INTEGRATION, the VLSI journal ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Dynamically adaptive real-time disparity estimation hardware using iterative refinement

Abdulkadir Akin*, Ipek Baz, Alexandre Schmid, Yusuf Leblebici

Microelectronic Systems Laboratory (LSM), Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland

ARTICLE INFO

Keywords:
Disparity estimation
Census transform
Sum of absolute differences
High resolution
Real-time implementation
Iterative refinement
FPGA

ABSTRACT

The computational complexity of disparity estimation algorithms and the need of large size and bandwidth for the external and internal memory make the real-time processing of disparity estimation challenging, especially for High Resolution (HR) images. This paper proposes a hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time reconfigurable hardware implementation that targets HR video with high quality disparity results. Moreover, an enhanced version of the AWDE implementation that uses iterative refinement (AWDE-IR) is presented. The AWDE and AWDE-IR algorithms dynamically adapt the window size considering the local texture of the image to increase the disparity estimation quality. The proposed reconfigurable hardware architectures of the AWDE and AWDE-IR algorithms enable handling 60 frames per second on a Virtex-5 FPGA at a 1024×768 XGA video resolution for a 128 pixel disparity range.

1. Introduction

Depth estimation is an algorithmic step in a variety of applications such as autonomous navigation, robot and driving systems [1], 3D geographic information systems [2], object detection and tracking [3], medical imaging [4], computer games and advanced graphic applications [5], 3D holography [6], 3D television [7], multiview coding for stereoscopic video compression [8], and disparity-based rendering [9]. These applications require high accuracy and speed performances for depth estimation.

Depth estimation can be performed by exploiting three main techniques: time-of-flight (TOF) camera, LIDAR sensor and stereo camera. A TOF camera easily measures the distance between the object and camera using a sensor, circumventing the need of intricate digital image processing hardware [10]. However, it does not provide efficient results when the distance between the object and camera is high. Moreover, the resolution of TOF cameras is usually very low (200×200) [10] when it is compared to the Full HD display standard (1920×1080). Furthermore, their commercial price is much higher than the CMOS and CCD cameras. LIDAR sensors compute the depth by using laser scanning mechanisms but they are also very expensive compared to CMOS and CCD cameras. Due to laser scanning hardware, LIDAR sensors are heavy and bulky devices. Therefore, they can be used mainly for static images. Consequently, in order to compute depth map, the majority of research focus on extracting the disparity information using two or more synchronized images taken from different viewpoints, using CMOS or CCD cameras [11].

Many disparity estimation (DE) algorithms have been developed with the goal to provide high-quality disparity results. These are ranked with respect to their performance in the evaluation of Middlebury benchmarks [11]. Although top-performer algorithms provide impressive visual and quantitative results [12–14], their implementations in real-time High Resolution (HR) stereo video are challenging due to their complex multi-step refinement processes or their global processing requirements that demand huge memory size and bandwidth. For example, the AD-Census algorithm [12], currently the top published performer, provides successful results that are very close to the ground truths. However, this algorithm consists of multi disparity enhancement sub-algorithms, and implementing them into a mid-range FPGA is very challenging both in terms of hardware resource and memory limitations.

Various hardware architectures that are presented in literature provide real-time DE [15–21]. Some implemented hardware architectures only target CIF or VGA video [15–18]. The hardware proposed in [15] only claims real-time for CIF video. It uses the Census transform [22] and currently provides the highest quality disparity results compared to real-time hardware implementations in ASICs and FPGAs. The hardware presented in [15] uses low complexity Mini-Census method to determine the matching cost, and aggregates the Hamming costs following the method in [12]. Due to high complexity cost aggregation, the hardware proposed in [15] requires high memory bandwidth and intense hardware

* Corresponding author. Tel.: +41 216934698.
E-mail address: abdulkadir.akin@epfl.ch (A. Akin).

0167-9260/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2013.11.002

resource utilization, even for low resolution (LR) video. Therefore, it is able to reach less than 3 frames per second (fps) when its performance is scaled to 1024×768 video resolution and 128 pixel disparity range.

Real-time DE for HR images offers some crucial advantages compared to low resolution DE. First, processing HR stereo images increases the disparity map resolution which improves the quality of the object definition. Second, DE for HR stereo images is able to define the disparity with sub-pixel efficiency compared to the DE for LR image. Therefore, the DE for HR provides more precise depth measurement than the DE for LR. Third, disparity values between 0 and 2 can be considered as background for LR images. In HR such disparities are defined within a larger disparity range; thus, the depth of far objects can be established more precisely.

Despite the advantages of HR disparity estimation, the use of HR stereo images brings some challenges. Disparity estimation needs to be assigned pixel by pixel for high-quality disparity estimation. Pixel-wise operations cause a sharp increase in computational complexity when the DE targets HR stereo video. Moreover, DE for HR stereo images requires stereo matching checks with larger number of candidate pixels than the disparity estimation for LR images. The large amount of candidates increases the challenge to reach real-time performance for HR images. Furthermore, high-quality disparity estimation may require multiple reads of input images or intermediate results, which poses severe demands on off-chip and on-chip memory size and bandwidth especially for HR images.

The systems proposed in [19–21] claim to reach real-time for HR video. Still, their quality results in terms of the HR benchmarks given in [11] are not provided. Authors [19] claims to reach 550 fps for 80 pixel disparity range at a 800×600 video resolution, but it requires extremely large hardware resources. A simple edge-directed method presented in [20] reaches 50 fps at a 1280×1024 video resolution and 120 pixel disparity range, but does not provide satisfactory DE results due to a low-complexity architecture. In [21], a hierarchical structure with respect to image resolution is presented to reach 30 fps at a 1920×1080 video resolution and 256 pixel disparity range, but it does not provide high-quality DE for HR.

In this paper, we present a hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time reconfigurable hardware implementation to process HR stereo video with high-quality disparity estimation results. A preliminary description of AWDE has been presented in [23]. In this paper, the AWDE algorithm and its real-time hardware implementation are explained with further details. In addition, the disparity estimation quality of the AWDE algorithm is improved using iterative disparity refinement process. The proposed enhanced AWDE algorithm that utilizes iterative refinement (AWDE-IR) is implemented in hardware and its implementation details are presented. Moreover, the algorithmic comparison is enhanced including with the results of additional algorithms.

The proposed AWDE algorithm combines the strengths of the Census transform and the Binary Window SAD (BW-SAD) [24] methods, thus enables an efficient hybrid solution for the hardware implementation. Although the low-complexity Census method can determine the disparity of the pixels where the image has a texture, mismatches are observed in textureless regions. Moreover, due to a 1-bit representation of neighboring pixels, the Census easily selects wrong disparity results. In order to correct these mismatches, our proposed AWDE algorithm uses the support of the BW-SAD, instead of using the complex cost aggregation method [12,15].

The benefit of using different window sizes for different texture features on the image is observed from the DE results in [24]. The selection of a large window size improves the algorithm performance in textureless regions while requiring higher computational load. However, the usage of small window sizes provides better disparity results where the image has a texture. Moreover, the use of BW-SAD provides better disparity estimation results than the SAD for the depth discontinuities [24]. The hardware presented in [24] is not able to dynamically change the window size, since it requires to re-synthesize the hardware for using different window sizes. In addition, the hardware presented in [24] does not benefit from the Census cost metric.

The proposed hardware provides dynamic and static configurability to have satisfactory disparity estimation quality for the images with different contents. It provides dynamic reconfigurability to switch between window sizes of 7×7, 13×13 and 25×25 pixels in run-time to adapt to the texture of the image.

The proposed dynamic reconfigurability provides better DE results than existing real-time DE hardware implementations for HR images [19–21] for the tested HR benchmarks. The proposed hardware architectures for AWDE and AWDE-IR provides 60 frames per second at a 1024×768 XGA video resolution for 128 pixel disparity range. The AWDE and AWDE-IR algorithms and their reconfigurable hardware can be used in consumer electronics products where high-quality real-time disparity estimation is needed for HR video.

2. Hardware-oriented adaptive window size disparity estimation algorithm

The main focus of the AWDE algorithm is its compatibility with real-time hardware implementation while providing high-quality DE results for HR. The algorithm is designed to be efficiently parallelized to require minimal on-chip memory size and external memory bandwidth.

As a terminology, we use the term “block” to define the 49 pixels in the left image that are processed in parallel. The term “window” is used to define the 49 sampled neighboring pixels of any pixel in the right or left images with variable sizes of 7×7, 13×13 or 25×25. The pixels in the window are used to calculate the Census and BW-SAD cost metrics during the search process.

The algorithm consists of three main parts: window size determination, disparity voting, and disparity refinement. The parameters that are used in the AWDE algorithm are given in Section 5.

2.1. Window size determination

The window size of the 49 pixels in each block is adaptively determined according to the Mean Absolute Deviation (MAD) of the pixel in the center of the block with its neighbors. The formula of the MAD is presented in (1), where $\boldsymbol{c}$ is the center pixel location of the block and $\boldsymbol{q}$ is the pixel location in the neighborhood, N_c, of $\boldsymbol{c}$. The center of the block is the pixel located at block (4,4) in Fig. 1.

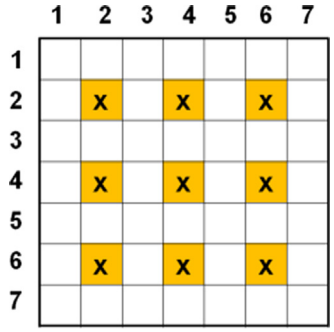

Fig. 1. 9 Selected pixels in a block for BW-SAD calculation. 49 Pixels in a block are searched in parallel in hardware.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

The high MAD value is a sign of high texture content and the low MAD value is a sign of low texture content. Three different window sizes are used. As expressed in (2), a 7×7 window is used if the MAD of the center pixel is high, and a 25×25 window is used if the MAD is very low.

$$\mathrm{MAD}(\boldsymbol{c})=\frac{1}{48}\times\sum_{\boldsymbol{q}\in N_c}|I_L(\boldsymbol{q})-I_L(\boldsymbol{c})| \tag{1}$$

$$\text{window size}=\begin{cases}7\times7 & \text{if}\quad \mathrm{MAD}(\mathbf{c})>tr_{7\times7}\\ 13\times13 & \text{else if}\quad \mathrm{MAD}(\mathbf{c})>tr_{13\times13}\\ 25\times25 & \text{else}\end{cases} \tag{2}$$

As a general rule, increasing the window size increases the algorithm and hardware complexity [24]. As shown in Fig. 2, in our proposed algorithm, in order to provide constant hardware complexity over the three different window sizes, 49 neighbors are constantly sampled for different window sizes. "1", "2" and "3" indicate the 49 pixels used for the different window sizes 7×7, 13×13 and 25×25, respectively. If the sampling of 49 pixels in a window is not applied and all the pixels in a window are used during the matching process, an improvement in the disparity estimation quality can be obtained. The overhead of computational complexity for this high-complexity case and the degradation of the DE quality due to sampling are presented in Section 5.

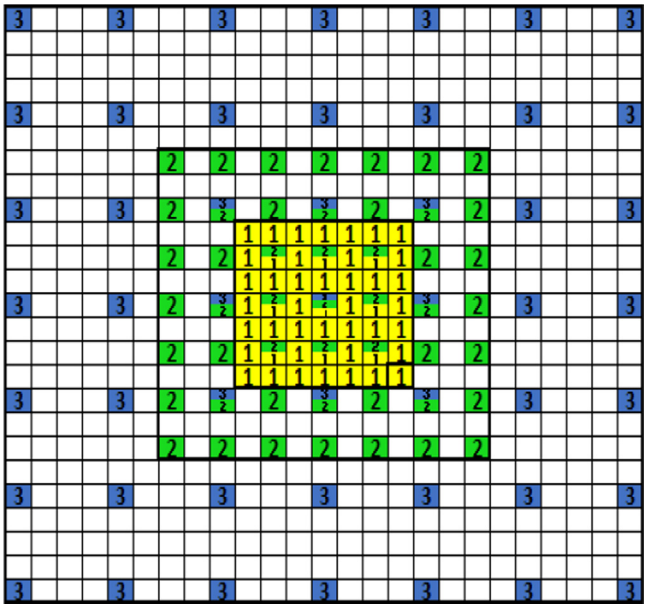

Fig. 2. 49 Selected pixels of adaptive windows (yellow (1) 7×7, green (2) 13×13 and blue (3) 25×25). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2. Disparity voting

A hybrid solution involving the Binary Window SAD and Census cost computation methods is presented to benefit from their combined advantages. The SAD is one of the most commonly used similarity metrics. The use of BW-SAD provides better results than using the SAD when there is disparity discontinuity since it combines shape information with the SAD [24]. However, the computational complexity of the BW-SAD is high, thus result of this metric is provided for nine of the 49 pixels in a block and they are linearly interpolated to find the BW-SAD values for the remaining 40 pixels in a block. The selected nine pixels for the computation of BW-SAD are shown in Fig. 1. The low complexity Census metric is computed for all of the 49 pixels of a block.

The formula expressing the BW-SAD for a pixel $\boldsymbol{p}=(x,y)$ is shown in (3) and (4). The BW-SAD is calculated over all pixels $\boldsymbol{q}$ of a neighborhood N_p, where the notation d is used to denote the disparity. The Binary Window, $\boldsymbol{w}$, is used to accumulate absolute differences of the pixels, if they have an intensity value which is similar to the intensity value of the center of the window. The multiplication with $\boldsymbol{w}$ in (4) is implemented as reset signal for the resulting absolute differences (AD). In the rest of the paper, the term, "*Shape*" is indicated by $\boldsymbol{w}$.

Depending on the texture of the image, the Census and the BW-SAD have different strengths and sensibility for the disparity calculation. To this purpose, a hybrid selection method is used to combine them. As shown in (5) and (6), an adaptive penalty (ap) that depends on the texture observed in the image is applied to the cost of the Hamming differences between the Census values. Subsequently, the disparity with the minimum *Hybrid Cost* (HC) is selected as the disparity of a searched pixel. Two's order penalty values are used to turn the multiplication operation into a shift operation. If there is a texture on the block, the BW-SAD difference between the candidate disparities needs to be more convincing to change the decision of Census, thus a higher penalty value is applied. If there is no texture on the block, a small penalty value is applied since the BW-SAD metric is more reliable than the decision of Census.

$$\boldsymbol{w}=\begin{cases}0 & \text{if}|I_L(\boldsymbol{q})-I_L(\boldsymbol{p})|>\text{threshold}_w, q\in N_p\\ 1 & \text{else}\end{cases} \tag{3}$$

$$\mathrm{BW-SAD}(p\ ,\ d)=\sum_{\boldsymbol{q}\in N_p}|I_L(\boldsymbol{q})-I_R(\boldsymbol{q}-d)|*\boldsymbol{w} \tag{4}$$

$$HC(\boldsymbol{p},\ d)=\mathrm{BW-SAD}(\boldsymbol{p},\ d)+\mathrm{Hamming}(\boldsymbol{p},\ d)\times ap \tag{5}$$

$$ap=\begin{cases}ap_{7\times7} & \text{if} \quad \text{window size}(p)=7\times7\\ ap_{13\times13} & \text{else if} \quad \text{window size}(p)=13\times13\\ ap_{25\times25} & \text{else if} \quad \text{window size}(p)=25\times25\end{cases} \tag{6}$$

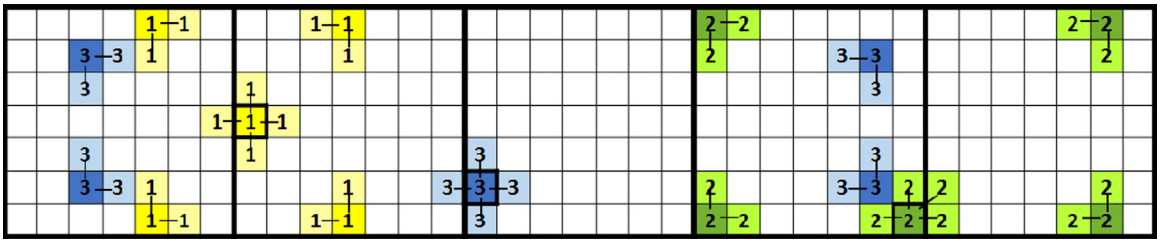

Fig. 3. (a) Examples for selecting 17 contributing pixels for 7×7, 13×13 and 25×25 window sizes during the disparity refinement process (yellow (1) 7×7, green (2) 13×13 and blue (3) 25×25). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

2.3. Disparity refinement

The proposed disparity refinement (DR) process assumes that neighboring pixels within the same *Shape* needs to have an identical disparity value, since they may belong to one unique object. In order to remove the faulty computations, the most frequent disparity value within the *Shape* is used.

As shown in Fig. 3, since the proposed hardware processes seven rows in parallel during the search process of a block, the DR process only takes the disparity of pixels in the processed seven rows. The DR process of each pixel is complemented with the disparities of 16 neighbor pixels and its own disparity value. Finally, the most frequent disparity in the selected 17 contributors is replaced with the disparity of that processed pixel.

The selection of these 17 contributors proceeds as follows. The disparity of the processed pixel and the disparity of its four adjacent pixels always contribute to the selection of the most frequent disparity. Four farthest possible *Shape* locations are pre-computed as a mask. If these locations are activated by *Shape*, the disparity values of these corner locations and their two adjacent pixels also contribute. Therefore, at most 17 and at least 5 disparities contribute to the refinement process of each pixel.

In Fig. 3, examples of the selection of contributing pixel locations are shown for three different window sizes. Considering the proposed contributor selection scheme, the pixels in the same row with the same window size have identical masks. The masks for the seven rows of a block and three window sizes are different. Therefore, 21 different masks are applied in the refinement process. These masks turn out to simple wiring in hardware.

Median filtering of the selected 17 contributors provides negligible improvement on the DR quality, but it requires high-complexity sorting scheme. The highest frequency selection is used for the refinement process since it can be implemented in hardware with low-complexity equality comparators and accumulators. The maximum number of contributors is fixed to 17 which provides an efficient trade-off between hardware complexity and the disparity estimation quality.

3. Hardware implementation

3.1. System overview

The top-level block diagram of the proposed reconfigurable disparity estimation hardware and the required embedded system components for the realization of the full system are shown in Fig. 4. The Reconfigurable Disparity Map Estimation module involves 5 sub-modules and 62 dual port BRAMs. These five sub-modules are the Control Unit, Reconfigurable Data Allocation, Reconfigurable Computation of Metrics (RCM), Adaptive Disparity Selection (ADS) and disparity refinement. 31 of the 62 BRAMs are used to store 31 consecutive rows of the right image, and the remaining 31 BRAMs

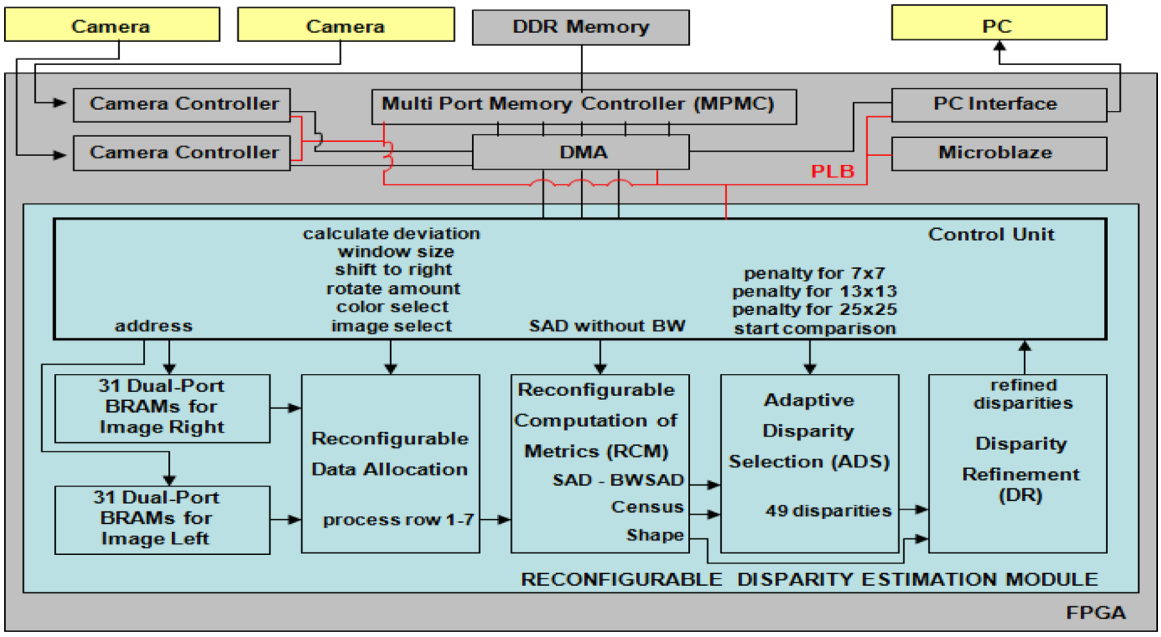

Fig. 4. Top-level block diagram of the system architecture.

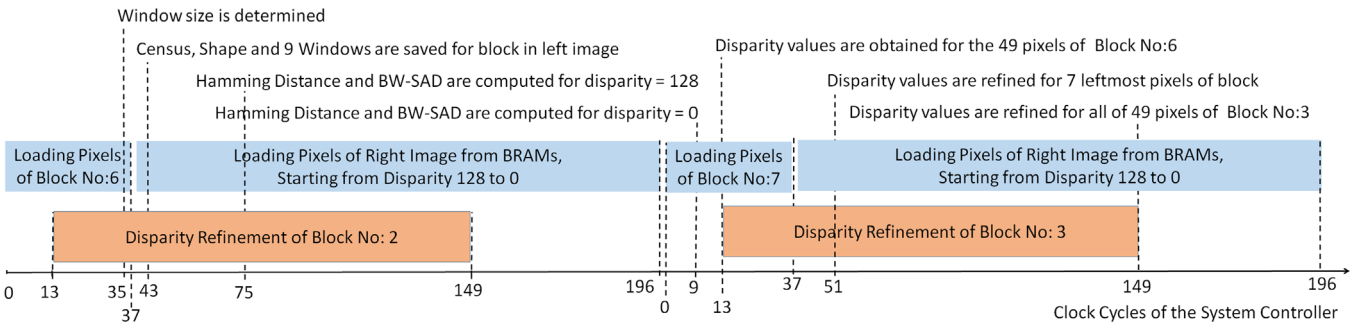

Fig. 5. System timing diagram.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

are used to store 31 rows of the left image. The dual port feature of the BRAMs is exploited to replace processed pixels with the new required pixels during the search process. The proposed hardware is designed to find disparity of the pixels in the left image by searching candidates in the right image. The pixels of the right image are not searched in the left image, and thus cross-check of the DE is not applied.

External memory bandwidth is an important limitation for disparity estimation of HR images. For example, the disparity estimation of a 768×1024 resolution stereo video at 60 fps requires 566 MB/s considering loading and reading each image one time. The ZBT SRAM and DDR2 memories that are mounted on FPGA prototyping boards can typically reach approximately 1 GB/s and 5 GB/s, respectively. However, an algorithm or hardware implementation that requires multiple reads of a pixel from an external memory can easily exceed these bandwidth limitations. Using multiple stereo cameras in future targets or combining different applications in one system may bring external memory bandwidth challenges. The hardware in [15] needs to access external memory at least five times for each pixel. The hardware presented in [19] requires external memory accesses at least seven times for each pixel assuming that the entire data allocation scheme is explained. Our proposed memory organization and data allocation scheme require reading each pixel only one time from the external memory during the search process.

The system timing diagram of the AWDE is presented in Fig. 5. The disparity refinement process is not applied to the pixels that belong to the two blocks at the right and left edges of the left image. For the graphical visualization of the reconfigurable disparity computation process together with the disparity refinement process, the timing diagram is started from the process of a sixth block of the left image. As presented in Fig. 5, efficient pipelining is applied between the disparity refinement and disparity selection processes. Therefore, the disparity refinement process does not affect the overall system throughput but only increases the latency. The system is able to process 49 pixels every 197 clock cycles for a 128 search range. Important timings during the processes are also presented with dashed lines along with their explanations.

3.2. Data allocation and disparity voting

The block diagram of the Reconfigurable Data Allocation module is shown in Fig. 6. The data allocation module reads pixels from BRAMs, and depending on the processed rows, it rotates the rows using the Vertical Rotator to maintain the consecutive order. This process is controlled by the Control Unit through the *rotate amount* signal.

The search process starts with reading the 31×31 size window of searched block from the BRAMs of the left image. Therefore, the Control Unit sends the *image select* signal to the multiplexers that are shown in Fig. 6 to select the BRAMs of the left image. Moreover, the *color select* signal provides static configurability to select one of the pixel's components (Y, Cb or Cr) during the search process. This user-triggered selection is useful if the Y components of the pixels are not well distributed on the histogram of the captured images. While the window of searched block are loaded to the D flip-flop (DFF) Array, the RCM computes and stores the 49 Census transforms, 49 *Shapes* and 9 windows pertaining to the pixels in the block for the computation of BW-SAD.

The Census transforms and windows of the candidate pixels in the right image are also needed for the matching process. After loading the pixels for the computation of metrics for the 7×7 block, the Control Unit selects the pixels in the right image by changing the *image select* signal, and starts to read the pixels in the right image from the highest level of disparity by sending the *address* signals of the candidate pixels to the BRAMs.

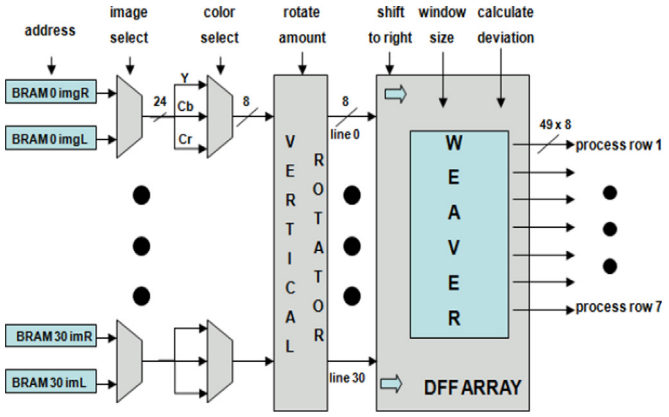

Fig. 6. Reconfigurable data allocation module.

The disparity range can be configured by the user depending on the expected distance to the objects. Configuring the hardware for a low disparity range increases the hardware speed. In contrast, a high disparity range allows the user to find the depth of close objects. The architecture proposed in [19] is not able to provide this configurability since it is designed to search 80 disparity candidates in parallel, instead of providing parallelization to search multiple pixels in the left image. Therefore, a fixed amount of disparities is searched in [19], and changing the disparity range requires a redesign of their hardware.

The detailed block diagram of the DFF array and the weaver are shown in Fig. 7. They are the units of the system that provide the configurability of the adaptive window size. As a terminology, we used the term "weaving" to mean "selecting 49 contributor pixels in different window sizes 7×7, 13×13 and 25×25 by skipping 1, 2 and 4 pixels respectively". Seven rows and one column are processed in parallel by the Weaver, and the processed pixels flow inside the DFF Array from the left to the right. Additionally, the weaving process is applied to the location (15,8) of the DFF Array at the beginning of the search process only, to select the window size by computing the deviation of the center of the block from its neighbors for 7×7 and 13×13 windows.

The DFF Array is a 31×25 array of 8-bit registers shown in Fig. 7. The DFF Array has 25 columns since it always takes the inputs of the largest window size, i.e., 25×25, and it has 31 rows to process seven rows in parallel. While the pixels are shifting to the right, the Weaver is able to select the 49 components of the 7×7, 13×13 and 25×25 window sizes from the DFF Array with simple wiring and multiplexing architecture. Some of the contributor pixels of the windows for different window sizes are shown in Fig. 7 in different colors. The Weaver and DFF Array are controlled by Control Unit through the *calculate deviation*, *window size* and *shift to right* signals. The Weaver sends seven windows to be processed by RCM as *process row 1*–*process row 7*, and each *process row* consists of 49 selected pixels.

A large window size normally involves high amount of pixels and thus requires more hardware resource and computational cost to support the matching process [24]. By using the proposed weaving architecture, even if the window size is changed, always 49 pixels are selected for the window. Therefore, the proposed hardware architecture is able to reach the largest window size (25×25) among the hardware architectures implemented for DE [15–21]. The adaptability of window size between the small and large window sizes provides high-quality disparity estimation results for HR images.

During the weaving process of the 49 pixels in the block and the candidate pixels in the right image, the RCM computes the Census and *Shape* of these pixels in a pipeline architecture.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

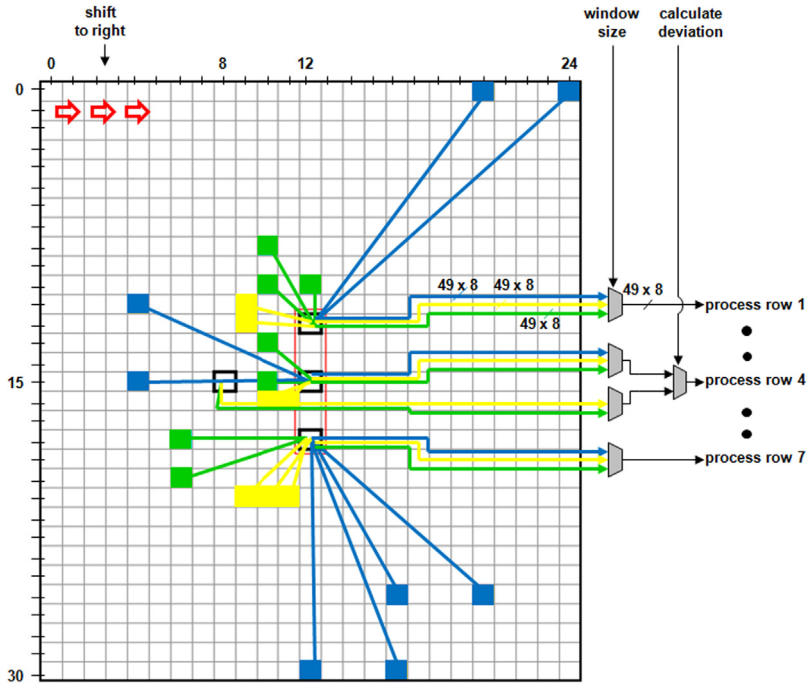

Fig. 7. DFF array and the weaver (yellow: 7×7, green: 13×13 and blue: 25×25). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

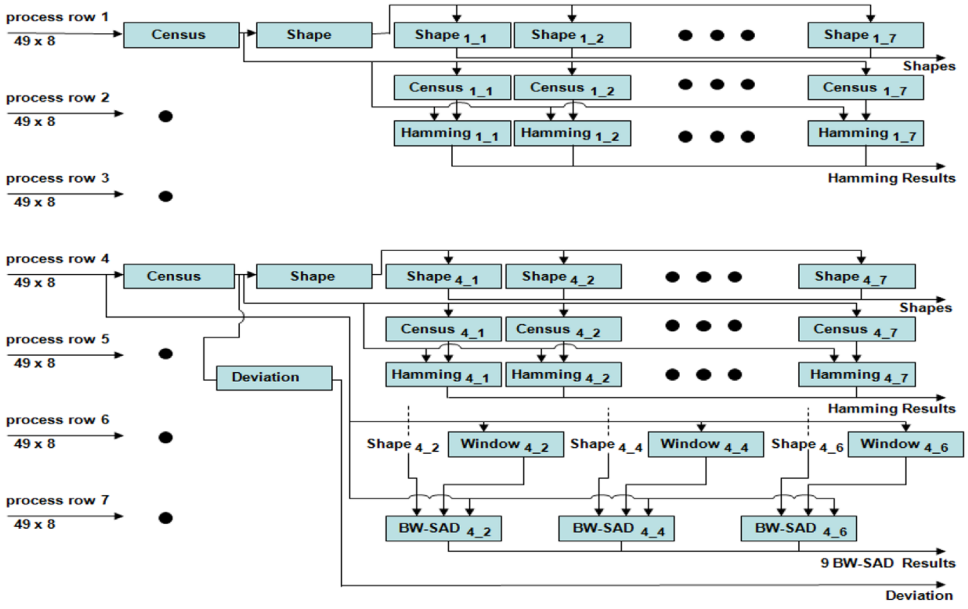

Fig. 8. Reconfigurable computation of metrics.

The block diagram of the RCM is shown in Fig. 8. The process for each block starts by computing and storing the Census and *Shape* results for the 7×7 block. In Fig. 8, the registers are named as "$Shape_{row\_column}$" and "$Census_{row\_column}$". Since the BW-SAD is only applied for 9 of the 49 pixels, the BW-SAD computation sub-modules are only implemented in process rows 2, 4 and 6.

The BW-SAD sub-module in Fig. 8 takes the *Shape*, registered window of the pixel in a block and the candidate window of the searched pixel as inputs, and provides the BW-SAD result as an output. The computation of the Hamming distance requires significantly less hardware area than the BW-SAD. Therefore, the Hamming computation is used for all of the 49 pixels in a block.

As shown in Fig. 8, when a new candidate Census for the *process row 1* is computed by the Census sub-module of the RCM, its Hamming distance with the preliminary computed seven Census1_[1:7] of the block is computed by the seven Hamming sub-modules. The seven resulting Hamming Results of the *process row 1* are passed to the ADS module. Since this process also progresses in parallel for seven process rows, the proposed hardware is able to compute the Hamming distances of 49 pixels in a block in parallel. This parallel processing scheme is presented in Fig. 9. While the proposed architecture computes the Hamming distance for the left-most pixels of the block, the Hamming for disparity d, rightmost pixels of the block computes their Hamming for disparity d+6.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

Therefore, the resulting Hamming costs are delayed in the ADS to synchronize the costs. This delay is also an issue of the BW-SAD results and they are also synchronized in the ADS.

The internal architecture of the Census transform involves 48 subtractors. The Census module subtracts the intensity of center from the 48 neighboring pixels in a window, and uses the sign bit of the subtraction to define 48-bit Census result. The *Shape* computation module reuses the subtraction results of Census module. The *Shape* module takes the absolute values of the subtraction results and compares the absolute values with the $threshold_w$. The Hamming computation module applies 48-bit XOR operation and counts the number of 1 s with an adder tree.

The Deviation module shown in Fig. 8 only exists on the process row 4 since it is only needed for the center of the 7×7 block to determine the window size. The module accumulates the absolute difference of the 48 neighboring pixels from the center. The Control Unit receives the deviation result of the 7×7 and 13×13 window sizes in consecutive clock cycles and determines window size. The mathematical calculation of the MAD requires dividing the total deviation by 48. In order to remove the complexity of the division hardware, the thresholds $tr_{7 \times 7}$ and $tr_{13 \times 13}$ are re-computed by multiplying them with 48 and compared with the resulting absolute deviations.

The use of BW-SAD provides better results than using the SAD in presence of disparity discontinuity [24]. However, if the processed image involves a significant amount of texture without much depth discontinuity, using the SAD provides better results. Especially for 7×7 window size, using SAD instead of BW-SAD provides better visual results since it is the sign of significantly textured region. Thus, dynamic configurability is provided to change the BW-SAD computation metric to the SAD computation for a 7×7 window. The SAD module computes the ADs and the result of ADs are stored in registers prior to accumulation. An active-low reset signal is used at the register of the AD to make its result 0 when the architecture is configured for the BW-SAD and the respective *Shape* of the pixel in the block is 0. Otherwise, the AD register takes its actual value and participate to the SAD.

The ADS module that is shown in Fig. 4 receives the Hamming results and the BW-SAD results from the RCM and determines the disparity of the searched pixels. Since the BW-SAD results are computed for 9 of the 49 pixels, the RCM linearly interpolates these nine values to find the estimated BW-SAD results of the remaining 40 pixels in the block. Due to an efficient positioning of the nine pixels in a block, the linear interpolation requires a division by 2 and 4, which are implemented as shift operations.

The ADS module shifts the Hamming results of the candidate pixels depending on the 2's order adaptive penalty for the multiplication process as shown in formula (5). The ADS module adds the resulting Hamming penalty on the BW-SADs to compute *Hybrid Costs*. 49 comparators are used to select 49 disparity results that point minimum *Hybrid Costs*.

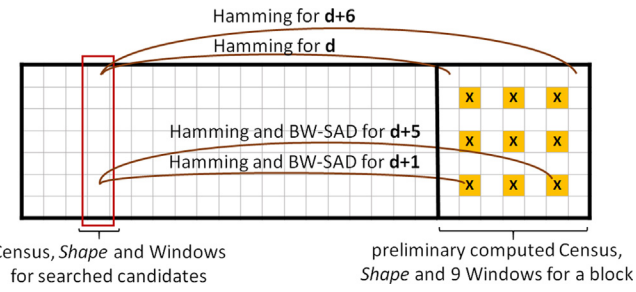

Fig. 9. Processing Scheme ("x" indicates 9 selected pixels in a block for BW-SAD calculations).

3.3. Disparity refinement

The DR module receives the 49 disparity results from the ADS and the *Shapes* of the 49 pixels of a block from the RCM and determines the final refined disparity values.

As presented in Fig. 10, after the ADS module has computed 49 disparity values in parallel, it loads this data in to the DFF Array of the DR module (DR-Array). The DR-Array has a size of five blocks for the refinement process. The Control Unit enables the DFFs by using the *Load Disparity* signal when the 49 disparity outputs of ADS module are ready for the refinement process. In each cell of the DR-Array, the respective *Shape* of a pixel is loaded from the RCM using the *Load Shape* signal. DR-Array is designed to shift the disparity and *Shape* values from right to left to allocate data for the refinement processes.

The DR hardware involves a Highest Frequency Selection (HFS) module that consists of seven identical processing elements (DR-PE). As presented in Fig. 10, DR-PEs are positioned to refine seven disparities in 15th column of DR-Array in parallel while the disparity and *Shape* values shift through the DR-Array. The hardware architecture of a single DR-PE is presented in Fig. 11. The location of a single DR-PE is shown in the 6th row of the DR-Array with bold square.

In Fig. 10, while 17 disparity values are selected by the multiplexers, the *Shape* information corresponding to the four corners are also selected from the 49-bit *Shape* information of the processed pixel. The selected 4-bits inform the DR-PE which of these 12 disparity values on the corners will be used while computing the highest frequency disparity. These 4 bits of the *Shape* are called *activation bits* in Fig. 11. Each *activation bit* activates itself together with its two adjacent disparities. Since the center disparity and its four neighbors are always activated, the 17-bit activation information is loaded to the DR-PE together with the respective disparities.

As presented in Fig. 11, the DR-PE hardware consists of two parts: Comparison of Disparities and Comparison of Frequencies. In the Comparison of Disparities part, the 17-bit activation information and the 17 disparities are stored in to two DFF Arrays. One of these DFF Arrays is used as a reference and the other one rotates to compare each disparity with the 16 other disparities. During the rotation process, 17 Compare and Accumulate (C&A) sub-modules compare the disparities in parallel. If the compared disparities are identical and both of them are activated, the values of the accumulators are increased by one. After 17 clock cycles, the values in the accumulators and their respective disparities are loaded in to the DFF Array in the Comparison of Frequencies part of the DR-PE. In the pipeline architecture, at the same time, the Control Unit shifts the DR-Array to the left by one to load new 17 contributors to the DR-PE. The Compare and Select (C&S) sub-module compares the values of the accumulators to find the highest value in the accumulators, and selects the disparity with the highest frequency as refined disparity. Since DR process works in parallel with other hardware modules of AWDE, it does not affect the throughput of the DE system if disparity range is configured as more than 70.

4. Iterative refinement for the enhance AWDE implementation

The intuition behind the proposed iterative refinement process of the IR-AWDE algorithm is identical to the DR process presented in Section 3: neighboring pixels within the same *Shape* need to have an identical disparity value, since they may belong to one unique object. Using the refinement process multiple times removes noisy computations more efficiently, and increases the disparity estimation quality.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

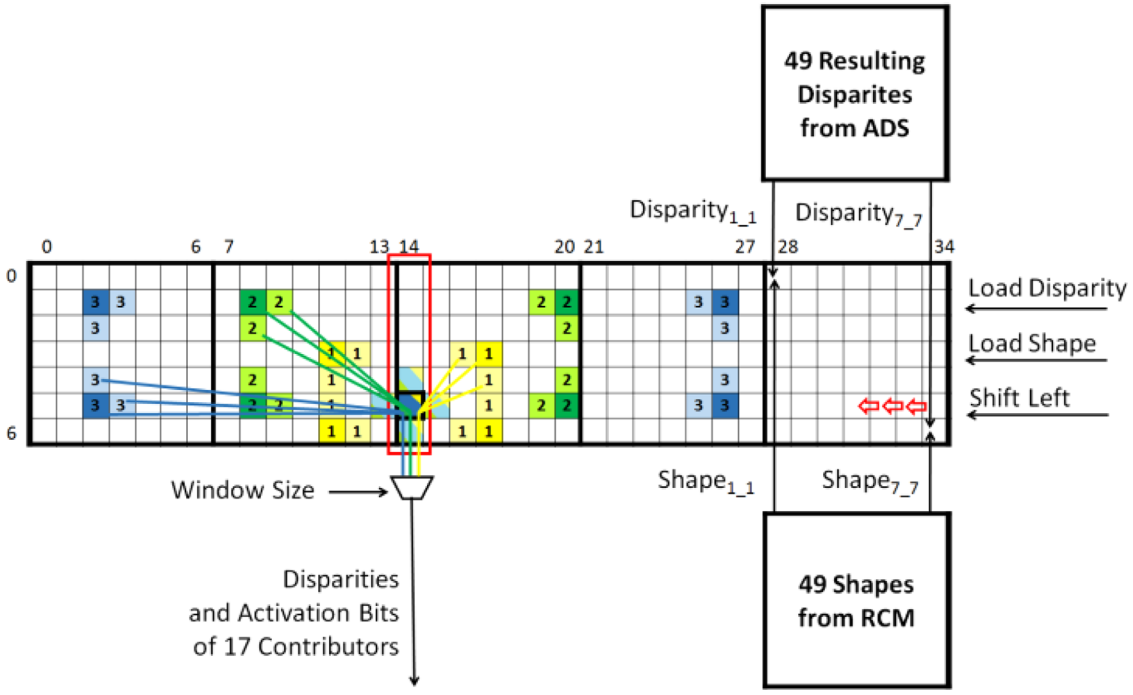

Fig. 10. DR-array of the disparity refinement module (yellow (1) 7×7, green (2) 13×13 and blue (3) 25×25). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

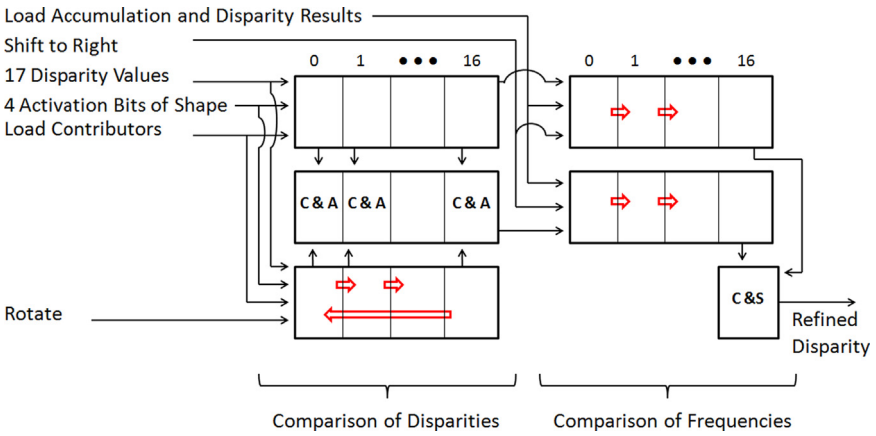

Fig. 11. Processing element of the disparity refinement module. The highest frequency selection module includes seven of these DR-PE elements.

The iterative refinement hardware is presented in Fig. 12 which consists of an improved version of the DR hardware presented in Fig. 10. The proposed iterative refinement process utilizes three concatenated Highest Frequency Selection modules. Each HFS module includes seven identical DR-PEs, one of which is presented in Fig. 11. All DR-PEs receive 17 selected disparities from their own multiplexer. DR-Array in Fig. 10 includes DFFs to keep record of the computed disparities for five blocks. Instead, for the IR, the size of the DFF-Array is increased to six blocks since the disparities need to be pipelined for longer duration. Moreover, DR hardware presented in Fig. 10 provides most frequent disparities as an output as the refined disparities. Instead, the HFS modules for the IR hardware write back the refined disparities on DR-Array. Writing back the most frequent disparities into the DR-Array provides an iterative refinement of the estimated disparities. Since the disparity results shift inside the DR-Array, refined disparities are overwritten 2 pixels left of the consecutive pixel location. For example, as presented in Fig. 12, while the HFS module refines the disparities of the seven pixels in column 21 of the DR-Array, the DR-Array shifts the disparity values 2 times. Therefore, the computed seven highest frequency disparities in the column 19 of the DR-Array are overwritten.

In addition to removing noisy computations, IR provides efficient results in assigning disparities of occluded regions. While searching pixels from the left image inside the right image, occluded regions appear on the left side of objects [11]. Consequently, wrong computations due to occlusion appear on the left sides of the objects in the image, which should be replaced by the correct disparities that are assigned to the left adjacent pixels of the occluded ones. The proposed iterative refinement process scans the estimated disparities from left to right. In addition, HFS modules receive updated disparities from their left since they are already overwritten by the refined ones. Therefore, this process iteratively spreads the correct disparities to the occluded regions while considering the object boundaries with the *Shape* information. While disparities shift inside the DR-Array, the leftmost disparities in the column 0 of the DR-Array are provided as the refined disparity value outputs of the IR Module.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

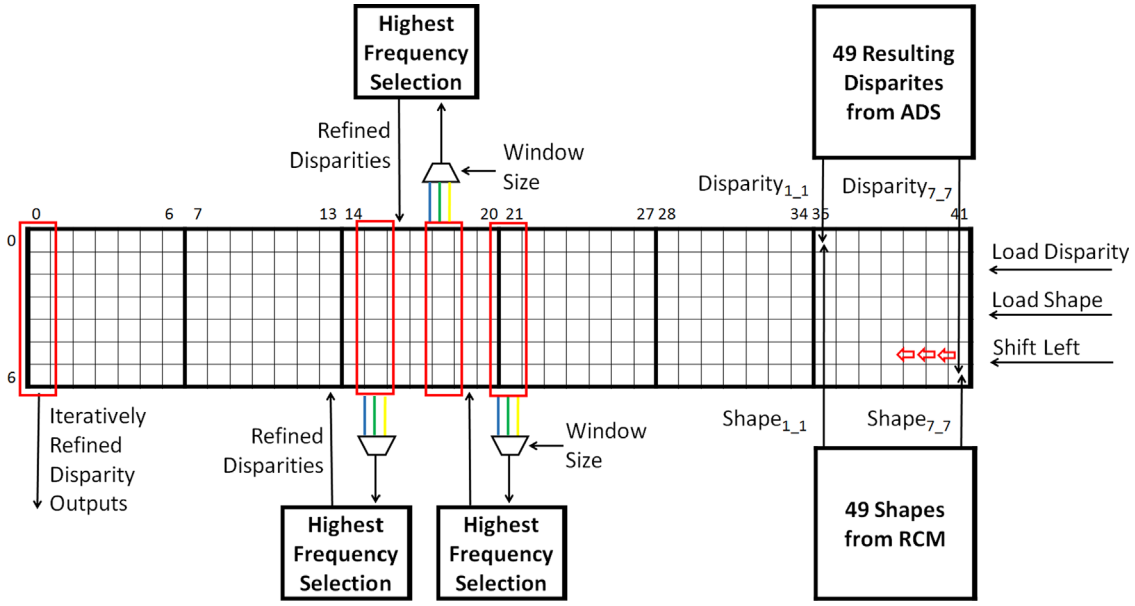

Fig. 12. DR-Array of the iterative disparity refinement module (yellow line: 7×17 candidates for 7×7 window, green line: candidates for 13×13, and blue line: candidates for 25×25). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Implementation results

The reconfigurable hardware architecture of the proposed AWDE algorithm is implemented using Verilog HDL, and verified using Modelsim 6.6c. The Verilog RTL models are mapped to a Virtex-5 XCUVP-110T FPGA comprising 69k Look-Up Tables (LUT), 69k DFFs and 144 Block RAMs (BRAM). The proposed hardware consumes 59% of the LUTs, 51% of the DFF resources and 42% of the BRAM resources of the Virtex-5 FPGA. The proposed hardware operates at 190 MHz after place and route, and computes the disparities of 49 pixels in 197 clock cycles for 128 pixel disparity range. Therefore, it can process 60 fps at a 768×1024 XGA video resolution.

The AWDE-IR is implemented to further improve the disparity estimation quality of AWDE using an efficient iterative refinement step. The hardware implementation of AWDE-IR is mapped to a same FPGA and verified using Modelsim 6.6c. The proposed AWDE-IR hardware consumes 70% of the LUTs, 63% of the DFF resources and 42% of the BRAM resources of the Virtex-5 FPGA. It can work at same speed performance due to the pipeline structure of the refinement process.

The parameters of the AWDE algorithm are shown in Table 1. Parameters are selected by sweeping to obtain high quality DE of HR images considering different features pertaining to the image content.

Tables 2 and 3 compare the disparity estimation performance and hardware implementation results of the AWDE architecture with other existing hardware implementations that targets HR [19–21] and currently the highest quality DE hardware that targets LR [15]. These papers do not provide the disparity estimation quality results for the HR benchmarks of the Middlebury data-set. Thus, we implemented [15,19,21] in software, and the software implementation of [20] is obtained from its authors. The DE results for the Census and the BW-SAD metrics for different window sizes are also presented in Table 2. The comparisons of the resulting disparities with the ground-truths are done as prescribed by the Middlebury evaluation module. If the estimated disparity value is not within a ± 1 range of the ground truth, the disparity estimation of the respective pixel is considered erroneous. 18 pixels located on the borders are neglected in the evaluation of LR benchmarks Tsukuba and Venus, and a disparity range of 30 is applied for all algorithms. Thirty pixels located on the borders are neglected in the evaluation of HR benchmarks Aloe, Art and Clothes, and a disparity range of 120 is applied for all algorithms.

Table 1
Parameters of the AWDE.

$tr_{7\times7}$	$tr_{13\times13}$	$ap_{7\times7}$	$ap_{13\times13}$	$ap_{25\times25}$	$threshold_w$
5	2	32	16	4	8

The Census and BW-SAD results that are shown in Table 2 are provided by sampling 49 pixels in a window. FW-DE indicates the combination of BW-SAD and Census for a fixed window size. The numbers terminating the name of the algorithms indicate the fixed window sizes of these algorithms.

Although the Census and the BW-SAD algorithms do not individually provide very efficient results, the combination of these algorithms into the FW-DE provides an efficient hybrid solution as presented in Table 2. For example, if a 7×7 window size and Census method are exclusively used for DE on the HR benchmark Art, 45.39% erroneous DE computation is observed from the result of Census7. Exclusively using a 7×7 window size and BW-SAD method for the same image yields 34.03% erroneous computation. However, if only a 7×7 window size is used combining the Census and BW-SAD methods, 20.87% erroneous computation is observed as presented in the result of FW-DE7. 20.87% erroneous computation is significantly smaller than 45.39% and 34.03%, which justifies the importance of combining the Census and BW-SAD in to a hybrid solution. For the same image, using the FW-DE13 and FW-DE25 algorithms yields 16.97% and 18.12% erroneous DE computations, respectively. Combining the FW-DE7, FW-DE13 and FW-DE25 into a reconfigurable hardware with an adaptive window size feature further improves the algorithm results as demonstrated from the results of AWDE. AWDE provides 16.33% erroneous computation for the same image which is smaller than 20.87%, 16.97% and 18.12%, thus numerically emphasizing the importance of adaptive window size selection. The algorithmic performance of AWDE, 16.33%, is considerably better than the DE performance results of HR DE hardware implementations [19–21] that provide 23.75%, 32.18% and 23.46% erroneous computations respectively for the same image.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

Table 2
Disparity estimation performance comparisons.

Algorithm	Error rate (%)				
	Tsukuba 288 × 384	Venus 383 × 484	Aloe 1110 × 1282	Art 1110 × 1390	Clothes 1110 × 1300
Chang [15]	4.15	0.56	3.75	12.80	2.97
Ttofis [20]	13.21	4.56	8.88	32.18	7.67
Greis [21]	12.42	4.14	8.65	23.46	5.30
Georg [19]	12.38	15.20	6.97	23.75	9.15
Census7	26.05	30.80	20.36	45.39	21.80
Census13	18.19	18.83	11.21	31.65	9.36
Census25	15.94	15.38	10.41	29.66	7.16
BWSAD7	12.19	19.45	8.31	34.03	13.33
BWSAD13	11.23	15.16	7.13	28.57	9.27
BWSAD25	10.43	11.12	6.74	24.74	6.28
FW-DE7	9.53	12.59	5.38	20.87	5.39
FW-DE13	7.90	6.82	4.81	16.97	3.16
FW-DE25	8.03	5.66	5.16	18.12	3.87
AWDE	7.64	5.33	4.94	16.33	2.89
AWDE-HC	7.47	4.73	4.92	16.17	2.95
AWDE-IR	6.53	5.01	4.30	14.47	2.94

Table 3
Hardware performance comparisons.

Hardware	Technology	Image resolution	DFF consumption	LUT consumption	Disparity range	fps	Clock speed (MHz)
Chang [15]	ASIC-90 nm	352 × 288	562k Gates		64	42	95
Ttofis [20]	Virtex-5	1280 × 1024	31k	47k	120	50	100
Greisen [21]	Stratix-III	1920 × 1080	26k	54k	256	30	130
Georgoulas [19]	Stratix-IV	800 × 600	15k	146k	80	550	511
Proposed (AWDE)	Virtex-5	1024 × 768			128	60	
		640 × 480	35k	40k	64	221	190
		352 × 288			64	670	
Proposed (AWDE-IR)	Virtex-5	1024 × 768			128	60	
		640 × 480	43k	48k	64	221	190
		352 × 288			64	670	

If the sampling of 49 pixels in a window is not applied and all the pixels in a window are used during the matching process, the complexity of the AWDE algorithm increases by 12 times. The result of the high complexity version of the AWDE algorithm (AWDE-HC) is also provided in Table 2 for comparison. The AWDE-HC provides almost the same quality results as the AWDE. Considering the hardware overhead of AWDE-HC, the low complexity version of the algorithm, AWDE, is selected for hardware implementation, and its efficient reconfigurable hardware is presented.

Improving the results of AWDE is possible using the low complexity iterative refinement step as indicated from the results of AWDE-IR. AWDE-IR efficiently removes a significant amount of noisy computations by iteratively replacing the disparity estimations with the most frequent neighboring ones as can be observed from the results of Tsukuba, Venus, Aloe and Art. Moreover, IR does not require significant amount of additional computational complexity. Therefore, AWDE-IR is implemented in hardware for the further improvement of the disparity estimation quality.

The algorithm presented in [15] uses the Census algorithm with the cost aggregation method, and provides the best results for both LR and HR stereo images except the HR benchmark Clothes. As shown in Table 3, due to the high-complexity of cost aggregation, it only reaches 42 fps for CIF images, thereby consuming a large amount of hardware resource. If the performance of [15] is scaled to 1024 × 768 for a disparity range of 128, less than 3 fps can be achieved.

None of the compared algorithms that have a real-time HR hardware implementation [19–21] is able to exceed the DE quality of AWDE and AWDE-IR for HR images. The overall best results following the results of AWDE and AWDE-IR are obtained from [21]. The hardware presented in [21] consumes 20% of the 270k Adaptive LUT (ALUT) resources of a Stratix-III. It provides high disparity range due to its hierarchical structure. However, this structure easily causes faulty computations when the disparity selection finds wrong matches in low resolution.

The hardware implementation of [19] provides the highest speed performance in our comparison. However this hardware applies 480 SAD computations for a 7 × 7 window in parallel. The hardware presented in [19] consumes 60% of the 244k ALUT resources of a Stratix-IV FPGA. In our hardware implementation we only use 9 SAD computations in parallel for the same size window and this module consumes 16% of the resources of Virtex-5 FPGA on its own. Therefore, the hardware proposed in [19] may not fit in to 3 Virtex-5 FPGAs.

The visual results of the AWDE and AWDE-IR algorithms for the HR benchmarks Clothes, Art and Aloe are shown in Fig. 13(a–l). The disparity map result of the AWDE algorithm for the 1024 × 768 resolution pictures taken by our stereo camera system is shown in Fig. 13(m–o). Our hardware architectures provide both quantitative and visual satisfactory results and reaches real-time for HR.

6. Conclusion

In this paper, a hardware-oriented adaptive window size disparity estimation algorithm, AWDE, and its real-time reconfigurable hardware implementation are presented. The proposed AWDE algorithm dynamically adapts the window size considering the local texture of the image to increase the disparity estimation quality. In addition, an enhanced version of the AWDE, AWDE-IR, is presented. AWDE-IR iteratively refines disparity estimations to remove the noisy

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

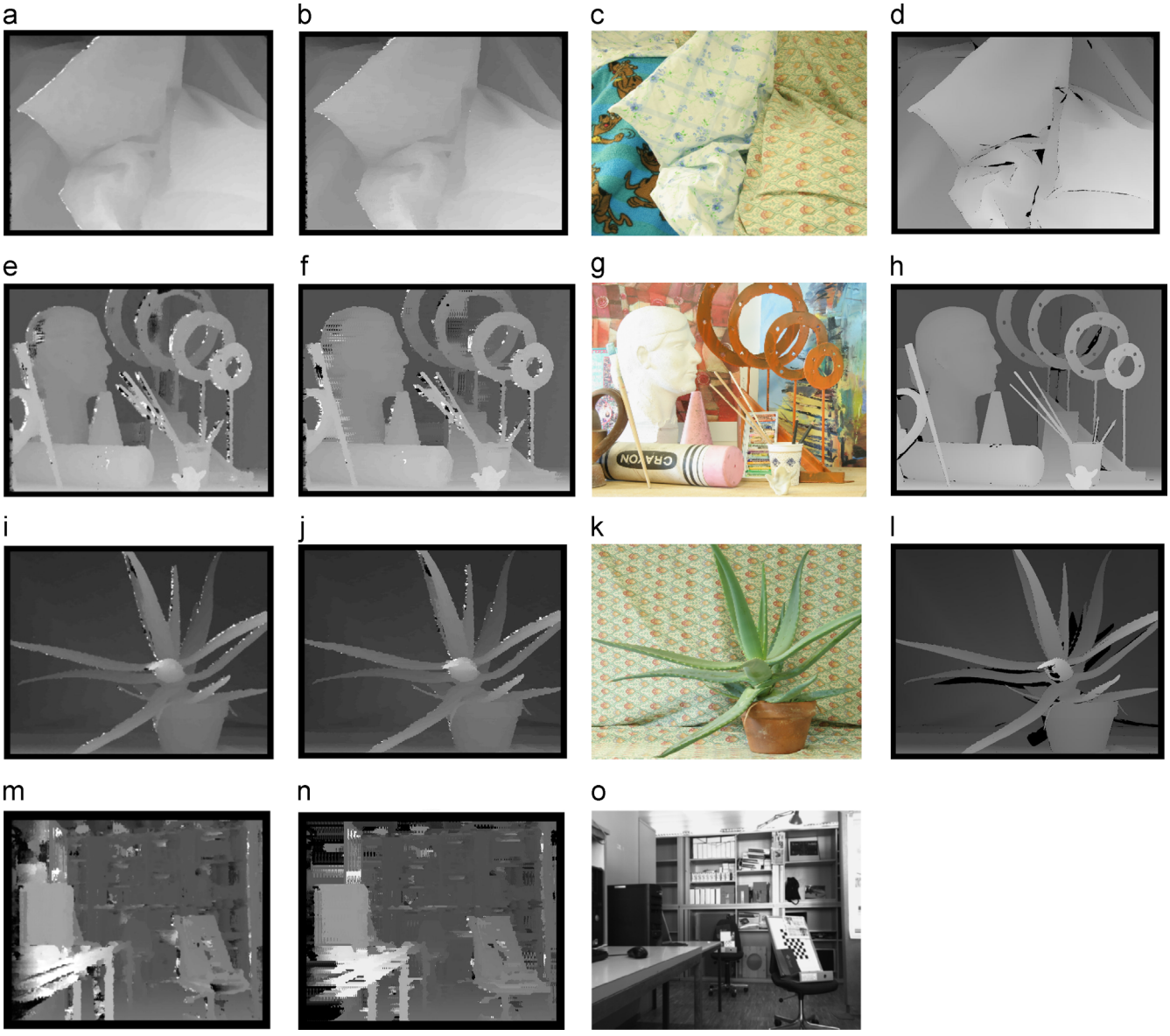

Fig. 13. Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmarks. From left column to right column: DE result of AWDE, DE result of AWDE-IR, left image, ground truth. Black regions in the ground truths are not taken into account for the error computations as explained in [11]. Ground truth for the image (o) is not available. (a–d) Clothes, (e–h) Art, (i–l) Aloe and (m–o) LSM lab.

computations of AWDE. Currently, the AWDE and AWDE-IR algorithms and their real-time hardware implementation reach higher DE quality than the existing real-time DE hardware implementations for HR images. The proposed reconfigurable hardware architectures of AWDE and AWDE-IR can process 60 fps at a 1024×768 XGA video resolution for 128 pixel disparity range. The AWDE and AWDE-IR algorithms and their reconfigurable hardware can be used in consumer electronic products where high-quality real-time disparity estimation is needed for HR video.

Acknowledgment

The authors would like to thank Baris Atakan and Irem Boybat for their support in algorithm development. This research has been partly conducted with the support of the Swiss NSF under Grant number 200021-125651, and in part by the Science and Technology Division of the Swiss Federal Competence Center armasuisse.

References

[1] F. Tombari, S. Mattoccia, L. Di Stefano, Stereo for robots: quantitative evaluation of efficient and low-memory dense stereo algorithms, in: Proceedings of the International Conference on Control Automation Robotics and Vision, IEEE Computer Society, December 2010, pp. 1231–1238.

[2] S. Yang, G. Huang, Z. Zhao, N. Wang, Extraction of topographic map elements with SAR stereoscopic measurement, in: Proceedings of the IEEE International Symposiom on Image and Data Fusion, August 2011.

[3] P.M. Cheung, K.T. Woo, Human tracking in crowded environment with stereo cameras, in: Proceedings of the 17th International Conferance on Digital System Processing, July 2011, pp. 1–6.

[4] M. Field, D. Clarke, S. Strup, W.B. Seales, Stereo endoscopy as a 3-D measurement tool, in: Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society 2009, September 2009, pp. 5748–5751.

[5] G. Yahav, G.J. Iddan, D. Mandelboum, 3D imaging camera for gaming application, in: Proceedings of the International Conference on Consumer Electronics (ICCE), January 2007, pp. 1–2.

[6] M. Grosse, J. Buehl, H. Babovsky, A. Kiessling, R. Kowarschik, 3D shape measurement of macroscopic objects in digital off-axis holography using structured illumination, Opt. Lett. 35 (2010) 1233–1235.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002

[7] Dongbo Min, Donghyun Kim, SangUn Yun, Kwanghoon Sohn, 2D/3D freeview video generation for 3DTV system, Elsevier J. Signal Process.: Image Commun. 24 (1–2) (2009) 31–48.
[8] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Müller, P.H.N. de With, T. Wiegand, The effects of multiview depth video compression on multiview rendering, Elsevier J. Signal Process.: Image Commun. 24 (1–2) (2009) 73–88.
[9] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, M. Tanimoto, View generation with 3D warping using depth information for FTV, Elsevier J. Signal Process.: Image Commun. 24 (1–2) (2009) 65–72.
[10] C. Lee, H. Song, B. Choi, Y.S. Ho, 3D scene capturing using stereoscopic cameras and a time-of-flight camera, IEEE Trans. Consum. Electron. 57 (3) (2011) 1370–1376.
[11] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Comput. Vision 47 (1–3) (2002) 7–42.
[12] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, X. Zhang, On building an accurate stereo matching system on graphics hardware, in: Proceedings of GPUCV, November 2011.
[13] Z. Wang, Z. Zheng, A region-based stereo matching algorithm using cooperative optimization, in: Proceedings of the IEEE Conference on Computer Vision Pattern Recognition, June 2008.
[14] A. Klaus, M. Sormann, K. Karner, Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure, in: Proceedings of ICPR, vol. 3, 2006.
[15] N.-C. Chang, T.-H. Tsai, B.-H. Hsu, Y.-C. Chen, T.-S. Chang, Algorithm and architecture of disparity estimation with mini-census adaptive support weight, IEEE Trans. Circuits Syst. Video Technol. 20 (6) (2010) 792–805.
[16] Y. Miyajima, T. Maruyama, A real-time stereo vision system with FPGA, in: Proceedings of the 30th Conference of IEEE Industrial Electronics Society, 2003.
[17] S. Jin, J. Cho, X.D. Pham, K.M. Lee, S.-K. Park, M. Kim, J.W. Jeon, FPGA design and implementation of a real-time stereo vision system, IEEE Trans. CSVT (2010) 15–26.
[18] Sang Hwa Lee, Siddharth Sharma, Real-time disparity estimation algorithm for stereo camera systems, IEEE Trans. Consum. Electron. 57 (3) (2011) 1018–1026.
[19] C. Georgoulas, I. Andreadis, A real-time occlusion aware hardware structure for disparity map computation, image analysis and process, ICIAP 5716 (2009) 721–730.
[20] C. Ttofis, S. Hadjitheophanous, A.S. Georghiades, T. Theocharides, Edge-directed hardware architecture for realtime disparity map computation, in: Proceedings of the IEEE Transactions on Computers, January 2012.
[21] P. Greisen, S. Heinzle, M. Gross, A.P. Burg, An FPGA-based processing pipeline for high-definition stereo video, EURASIP J. Image Video Process. 2011 (2011) 18.
[22] R. Zabih, J. Woodfill, Non-parametric local transforms for computing visual correspondence, in: Proceedings of the 3rd European Conference on Computer Vision, vol. 2, 1994, pp. 151–158.
[23] A. Akin, I. Baz, B. Atakan, I. Boybat, A. Schmid, Y. Leblebici, A hardware-oriented dynamically adaptive disparity estimation algorithm and its real-time hardware, in: Proceedings of the GLSVLSI Conference, Paris, France, May 2013.
[24] A. Motten, L. Claesen, A binary adaptable window SoC architecture for a stereo based depth field processor, in: Proceedings of the IEEE VLSISOC-2010, 18th IEEE/IFIP International Conference on VLSI and System-on-Chip, September 2010, pp. 25–30.

Abdulkadir Akın received B.S. and M.S. degrees in Electronics Engineering from Sabanci University, Istanbul, Turkey in July 2008 and July 2010, respectively. He is currently studying towards a Ph.D. degree at the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland. His research interests include digital hardware design for video processing and coding.

İpek Baz received B.S. degree in Electronics Engineering from Sabanci University, Istanbul, Turkey in July 2011. She is currently studying towards a M.S. degree at the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland. Her research interests include algorithm development for image processing and coding.

Alexandre Schmid (S′98–M′04) received the M.Sc. degree in Microengineering and the Ph.D. degree in Electrical Engineering from the Swiss Federal Institute of Technology (EPFL) in 1994 and 2000, respectively.

He has been with the EPFL since 1994. Since 2011, he is a "Maître d'Enseignement et de Recherche" (MER) faculty member in EPFL, conducting research in the fields of bioelectronic interfaces, non-conventional signal processing and neuromorphic hardware, and reliability of nanoelectronic devices. He is co-author and co-editor of three books, and over 100 articles published in journals and conferences.

Dr. Schmid has served as the General Chair of the Fourth International Conference on Nano-Networks in 2009, and serves as an Associate Editor of the IEICE Electronics Express since 2009.

Yusuf Leblebici (M′90–SM′98–F′09) received his B.Sc. and M.Sc. degrees in Electrical Engineering from Istanbul Technical University, in 1984 and in 1986, respectively, and his Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign (UIUC) in 1990. Since 2002, Dr. Leblebici has been a Chair Professor at the Swiss Federal Institute of Technology in Lausanne (EPFL), and director of Microelectronic Systems Laboratory. His research interests include design of high-speed CMOS digital and mixed-signal integrated circuits, computer-aided design of VLSI systems, intelligent sensor interfaces, modeling and simulation of semiconductor devices, and VLSI reliability analysis. He is the coauthor of 6 textbooks, as well as more than 300 articles published in various journals and conferences.

He has served as an Associate Editor of IEEE Transactions on Circuits and Systems (II), and IEEE Transactions on Very Large Scale Integrated (VLSI) Systems. He has also served as the general co-chair of the 2006 European Solid-State Circuits Conference, and the 2006 European Solid State Device Research Conference (ESSCIRC/ESSDERC). He is a Fellow of IEEE and has been elected as Distinguished Lecturer of the IEEE Circuits and Systems Society for 2010–2011.

Please cite this article as: A. Akin, et al., Dynamically adaptive real-time disparity estimation hardware using iterative refinement, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.11.002