# A Reconfigurable Disparity Engine for Stereovision in Advanced Driver Assistance Systems

Mehdi Darouich[1], Stephane Guyetant[1], and Dominique Lavenier[2]

[1] CEA LIST, Embedded Computing Laboratory,
PC94, Gif-sur-Yvette, 91191 France
mehdi.darouich@cea.fr
[2] ENS Cachan Bretagne / IRISA,
Campus de Beaulieu, Rennes, 35042 France

**Abstract.** Depth extraction in stereovision applications is very time-consuming and requires hardware acceleration in real-time context. A large number of methods have been proposed to handle this task. Each method answers more or less to real-time constraints, depending on the applicative context and user's needs. Thus, flexibility is a strong requirement for a generic hardware acceleration solution, particularly when ASIC implementation is targeted. This paper presents REEFS[1], a reconfigurable architecture for embedded real-time stereovision applications. This architecture is composed of three reconfigurable modules that enable flexibility at each step of depth extraction, from correlation window size to the matching method. It generates VGA depth maps with 64 disparity levels at almost 87 frames per second, answering hard real-time requirements, like in Advanced Driver Assistance Systems.

## 1 Introduction

Stereovision is widely used in many fields of computer vision, from robotics to intelligent transportation. Its principle is to calculate the depth of scene by triangulation from two images taken from different view points. The depth map is created using different methods and parameters, depending on the target application; in this article we focus on embedded stereovision applications. Some of these applications, such as advanced driver assistance systems (ADAS), have strong accuracy and real-time requirements. Other applications, like in mobile multimedia systems, have lower performance requirements but harder embedded constraints. Besides, a plethora of algorithmic solutions have been investigated to answer these issues.

Strong embedded and real-time requirements of stereovision applications imply the use of hardware accelerators. The algorithmic variability leads us to consider hardware flexibility as an important requirement as well. However, a trade-off must be found between efficiency and the level of flexibility. General

[1] Reconfigurable Embedded Engine for Flexible Stereovision.

Purpose Processors (GPP) are very flexible, through programming, but do not provide enough processing power to answer current requirements, as it consume too much power (40 to 100W). The use of Graphical Processing Units is more and more investigated to execute real-time vision applications as they provide high processing power and remain user-friendly through programmability. However, their power consumption (up to 100W) remains too high for many applications. FPGAs gives access to a high level of flexibility and high performances, but still with a low efficiency (regarding surface/performance/power ratio). Finally, application specific architectures provide high efficiency and allow real-time execution with a few Watts. However, their flexibility is limited to parameters customization (image size, disparity range, etc).

We propose a hardware architecture, targeting ASIC implementation. This architecture has a level of reconfiguration enabling area-based image processing techniques, and particularly advanced depth maps creation techniques, with a wide range of applicative conditions. Besides, our solution is designed to meet the requirements of applications in ADAS and can be scaled to adapt to any level of performance. The paper is organized as follows: section 2 presents the context of stereovision systems, outlining algorithmic flexibility needs and presenting related works. Section 3 presents the proposed architecture and its three reconfigurable modules. ASIC synthesis results and configuration examples are presented in section 4. Finally, conclusions and outlooks are given in section 5.

## 2 Stereovision Systems

Depth extraction is done by disparity map generation. Disparity is the gap between a point in the first image, called reference image, and the corresponding point in the second image. The disparity of a point is inversely proportional to its depth. Disparity map creation can be divided into two main steps: matching cost processing and corresponding pixels matching. The first step consists in evaluating the level of correspondence between pixels in the reference image and pixels from the other image: a matching cost is generated for each pixels pair. The second step uses these matching costs to extract the correct disparity. There are two main approaches for disparity map creation: feature-based approaches and area-based approaches. Feature-based approaches match features like edges, corners or points of interest and then require a reduced computational time. Area-based approaches process disparity for all the pixels of the reference image, generally taking into account their neighborhood. In this article, we focus on area-based approaches as they give access to high-density maps and are generally very regular and highly parallelizable. Furthermore, they are executed in deterministic time, which is required in high safety real-time applications.

The matching cost processing step is different from an application to another by the use of various metrics and window sizes and shapes [8]. The common metrics are SAD (Sum of Absolute Differences), SSD (Sum of Squared Differences), NCC (Normalized Cross-Correlation). SAD is often used in embedded solutions because it implies low processing cost; SSD and NCC require complex operations, like square or multiply, and thus are hardly compatible with embedded constraints. Centered versions of these metrics (ZSAD, ZSSD, ZNCC) are less sensible to illumination variations between the two images, but require more processing time. CENSUS is another interesting metric as it is insensitive to illumination variations and has a low processing cost.

The pixel's neighborhood used in matching cost processing has an influence on the quality of generated disparity maps and the processing time. The window's shape is generally rectangular. Large windows are adapted to low-textured area but are less precise on edges than small ones. However, small windows produce noisy disparity maps. A way of improvement is to use multiple windows: in [4] for instance, the cost is calculated on a centered window and on four peripheral windows. The two peripheral windows with the best cost are added to the central cost. This way, a composite window is processed, that adapts its shape to the data.

The second step, the corresponding pixel matching, can be done by local methods, semi-global methods or global methods. The most common local method, called Winner Take All (WTA), consists in choosing the pixels pair with the best matching cost. This technique implies very low processing time but gives poor results on noisy stereo images and in ambiguous areas (occultation zones or repetitive textures). A possible enhancement is to detect and invalidate wrong matches: the error rate decreases as well as the disparity map density. The level of confidence of a match can be determined by analyzing matching costs distribution [7]. Another way of detecting wrong disparities is to use symmetry constraint [3] and compare the disparity map processed with the left image as reference and the disparity map processed with the right image as reference: disparities that do not correspond are thus invalidated. Semi-global matching methods give access to high density disparity maps with a lower error rate. In these methods, a constraint is applied to a part of the image to enhance disparity continuity and thus to reduce noise. Dynamic programming [6] is a well known semi-global method: disparity on image lines is optimized using order and unicity constraints. However, it requires much more memory than local methods ($\approx$ 100kB for VGA images). Global matching methods are too costly to be used in embedded solutions because their memorization needs are too high ($\approx$ 100MB for VGA images) and the number of operations can be more than $10^8$ operations per pixel. Detection quality can also be increased by using preprocessing or post-processing. Rank filter applied on both images before matching cost processing eliminates illumination variation sensibility [8]. In [7], a median filter is applied on the disparity map to eliminate wrong matches.

Image size is an important parameter regarding the quality of the result and the needed processing time. Smaller QVGA stereo pairs (320x240) are often used to shorten the processing time. However, higher image resolutions give access to more detailed disparity maps, especially for far objects, implying more processing time. The maximum disparity range is also an important parameter: it depends on the stereo cameras pair geometry and defines the minimum detection distance. Besides, as close objects detection is generally critical, maximum disparity range highly depends on detection requirements. For a VGA stereo pair, maximum range usually varies from 64 to 128, or even 256 pixels.

A lot of application specific architectures aim to answer stereovision problem, but without flexibility considerations. For example, the Deepsea processor [9] uses CENSUS metric on 7x7 windows for 52 disparity range and is able to generate 512x480 disparity maps at 200 fps, which corresponds to a performance of 2.6 GPDS[2]. SIMD architectures give access to more flexibility but with lower performance. For example, the IMAP VISION of NEC [6] processes dynamic programming based disparity maps on 128 per 128 regions of interest with 64 dispariy levels at 10 fps ($\approx$11 MPDS). In [2], the proposed architecture parameters (image size, disparity range and correlation window size among others) can be tuned by static FPGA reconfiguration. It reaches a maximum peformance of 8 GPDS. An adaptive architecture targeting both ASIC and FPGA implementations is proposed in [1]. It is flexible on disparity range and windows size, and achieves 2 GPDS. However, the flexibility of the correlation window is based on processing block merging, which limits the possibilities. The VoC [5] is based on a reconfigurable matrix and processes matching costs on windows with various sizes, but only with SAD metric. Its performance is about 1.4 GPDS.

As shown in this section, stereo matching methods taxonomy is wide and the quality of a given method strongly depends on targeted applications, even in the same applicative field. Thus, flexibility is a strong requirement for a generic hardware acceleration solution targeting ASIC implementation.

## 3 Presentation of the REEFS Architecture

### 3.1 Global Overview

The proposed architecture is flexible enough to execute a broad range of algorithms. Besides, targeting the ADAS domain, associated requirements are to be fulfiled. First of all, the image frequency for this kind of applications is typically 40 fps. Secondly, as we need high quality detection, we have based our study on VGA images (640x480), which implies a maximum disparity from 64 to 128 pixels. However, other image sizes can be used as well, depending on user's requirements. The maximum window size for matching cost calculation is established to 15x15, which is large enough for most known applications. As shown in figure 1, the REEFS (Reconfigurable embedded Engine for Flexible Stereovision) is a data-flow oriented architecture composed of three reconfigurable modules. Based on the assumption that input images are rectified, the architecture extracts disparity over lines. Thus, for a given line, needed pixels are located on 15 lines in both images which are stored in line buffers. This on-the-fly approach allows memory use reduction.

[2] PDS: Point Disparity per Second, *i.e.* the number of matching costs generated per second. Note that this metric does not take into consideration complexity of the cost generation method and the matching method.

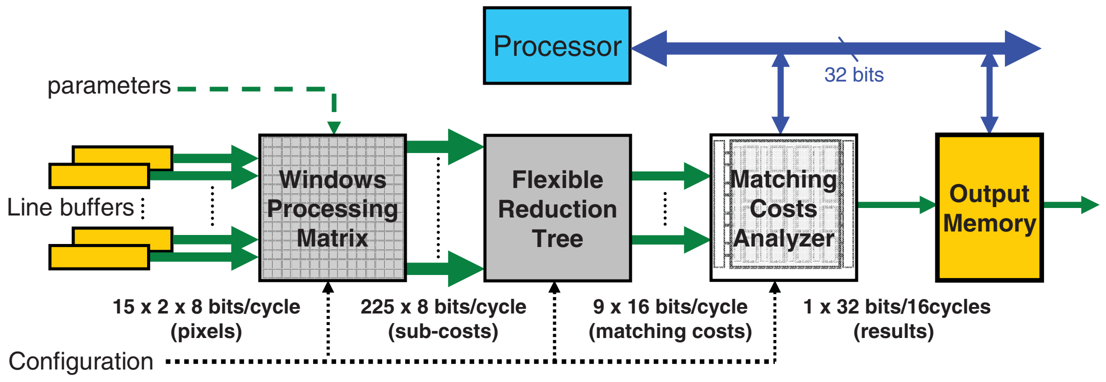

Processor
32 bits
parameters
Line buffers
Windows Processing Matrix
Flexible Reduction Tree
Matching Costs Analyzer
Output Memory
15 x 2 x 8 bits/cycle (pixels)
225 x 8 bits/cycle (sub-costs)
9 x 16 bits/cycle (matching costs)
1 x 32 bits/16cycles (results)
Configuration

**Fig. 1.** The REEFS allows the generation and the analysis of 9 matching costs per cycle. It is based on three reconfigurable modules, namely the Windows Processing Matrix (WPM), the Flexible Reduction Tree (FRT) and the Matching Cost Analyzer (MCA).

The first reconfigurable module, called Windows Processing Matrix (WPM), handles the first step of matching costs generation: sub-costs generation. It processes, every cycle, 15x15 windows on both images (225 pixels pairs) and generates as many sub-costs as pixel pairs. The WPM's reconfiguration allows to specify which metric is used for matching costs generation. To reduce the memory bandwith implied by window updates, currently processed pixels are stored inside the WPM. Exploiting data redundancy between two consecutive windows, the neighborhood update is done in one cycle by loading 15 pixels column from buffer lines. The second reconfigurable module, the Flexible Reduction Tree (FRT), sums the 225 sub-costs to obtain matching costs. Regarding current requirements, a minimum of 4 matching costs must be generated every cycle[3]. However, as multiple windows support is targeted, a maximum of 9 matching costs are generated each cycle. The shape and size of the window on which each matching cost is processed are application-dependant and can be modified by reconfiguring the FRT. Finally, the Matching Costs Analyzer (MCA) processes 9 matching costs each cycle to extract related disparity and metadata used by external high-level decision modules. These output data are stored in a buffer memory, named output memory in the figure 1, that is read by external modules. The MCA reconfiguration gives access to various extracting methods. A processor has been added to handle irregular processing; it has access to MCA input and output data and to the output memory. We now describe in detail the three reconfigurable modules: WPM, FRT and MCA.

## 3.2 Windows Processing Matrix

Most of the matching cost functions can be written as follows:

$$C = \sum_{N_R, N_A} f(P_R, P_A, c_R, c_A) \tag{1}$$

[3] With a given system frequency of 200MHz, generating VGA disparity maps at 40fps implies the processing of $\frac{640\times480\times40\times64}{200e6} \approx 4$ matching costs per cycle.

In this formula, the matching cost C is the sum of all sub-costs $f(P_R, P_A, c_R, c_A)$. Each sub-cost is calculated using a pixels pair $P_R$ and $P_A$, belonging respectively to reference area $N_R$ and to area $N_A$, and constants $c_R$ and $c_A$ related respectively to area $N_R$ and $N_A$. The function $f$ corresponds to the metric used. As we target a system able to generate one matching cost per cycle, we need to process as many sub-costs as there are in the concerned areas, using the desired metric. Besides, among common metrics, SAD, ZSAD and CENSUS are chosen for their simplicity and the fact that each provides result with characteristics different from the others.

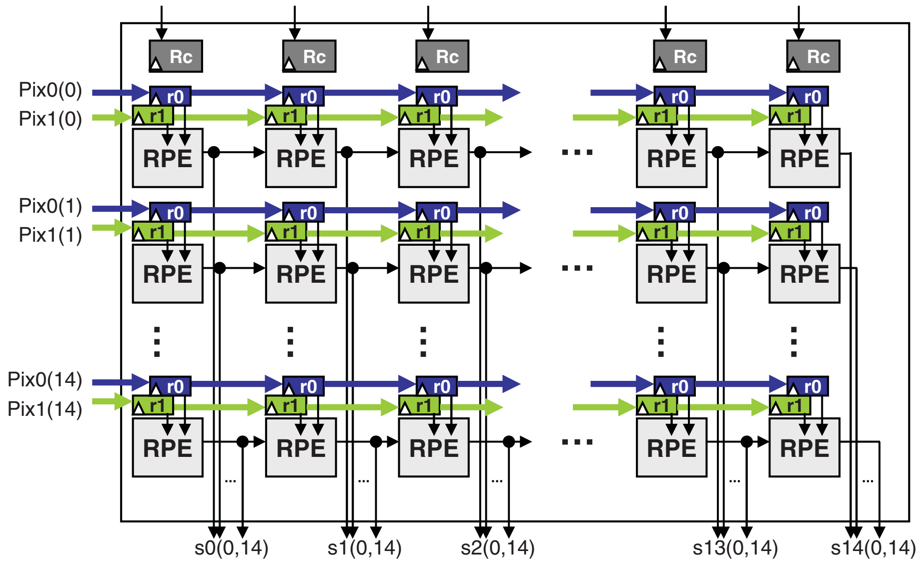


**Fig. 2.** The Windows Processing Matrix, composed of 15 columns of 15 RPEs each, processes 225 sub-costs necessary for matching costs generation on 15x15 areas each cycle. Reconfiguration of RPEs functionality and interconnection enables flexibility on the implemented metric.

As shown in figure 2, the WPM is composed of 15 columns of 15 Reconfigurable Processing Elements (RPE) each. These 225 RPEs allow generating one matching cost over an area of 15x15 pixels every cycle. Various operations can be executed by the RPE, depending on the chosen configuration: difference (D), absolute difference (AD), comparison (CMP), CENSUS and output weighting. The reconfiguration ability provides flexibility on the sub-cost generation function ($f$ in equation 1) used and thus on the metric implemented.

The inputs connexion of a RPE can also be reconfigured: each input can be connected to one of the two assigned pixel registers (r0 and r1 in figure 2), which store a pixel of the considered neighborhood ($P_R$ and $P_A$ in equation 1). It can also be connected to the column register Rc, which stores data that are common to the whole column. This register is useful when constant parameters ($c_R$ and $c_A$ in equation 1) are needed as inputs, for instance in *Zero-mean* metrics, *CENSUS* or even in *RANK* filter. At last, each input can be connected to the output of the previous RPE. Thus, complex operations can be executed at each cycle. For example in ZSAD, one RPE handles zero-mean rectification on pixels while the other one performs absolute differences.

## 3.3 Flexible Reduction Tree

The Flexible Reduction Tree's first purpose is to sum, each cycle, the 225 sub-costs generated by the WPM to obtain up to 9 matching costs. It must also answer flexibility requirements on the shape and the size of windows to ensure a good level of algorithmic variability. As displayed in figure 3, the FRT is divided in 3 levels. The first level is composed of reconfigurable sub-trees each assigned to a WPM column. The second level is also composed of the same type of sub-tree. The configuration of the first level allows to specify vertical patterns while the configuration of the second level impacts horizontal patterns. The last level is a simplified sub-tree (see figure 4(b)) which enables the combination of several matching costs.

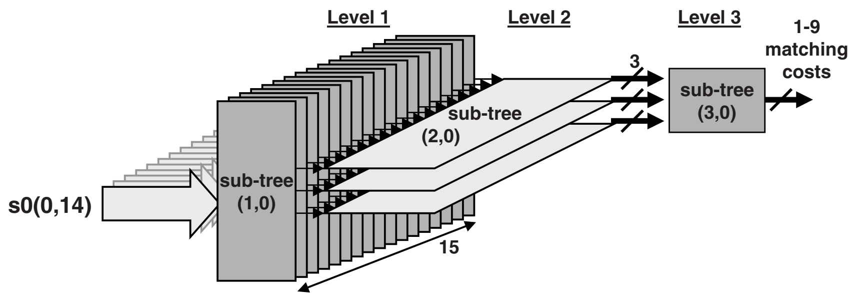
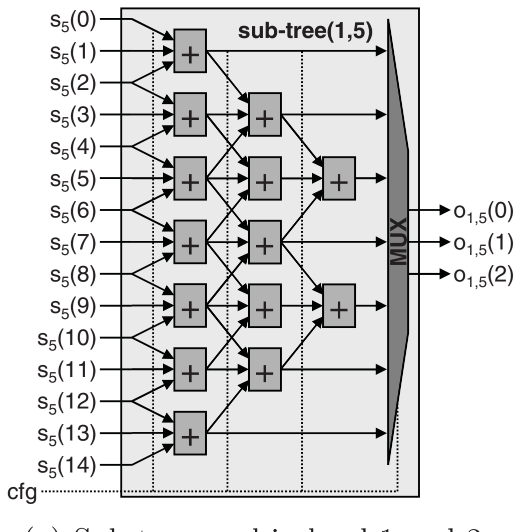
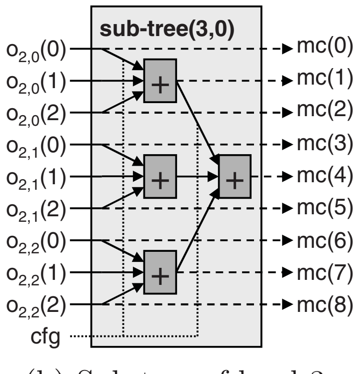


**Fig. 3.** The Flexible Reduction Tree sums the 225 sub-costs generated each cycle by the WPM. It is divided in 3 levels.



(a) Sub-tree used in level 1 and 2



(b) Sub-tree of level 3

**Fig. 4.** The first two levels of the FRT are composed of sub-trees shown in 4(a). The third level is a simpler sub-tree shown in 4(b). Active ways are selected by configuration.

The first type of reconfigurable sub-tree is shown in figure 4(a). It consists in a 3-ways adders tree with reconfigurable connectivity: each adder input can be disabled, in order to define the desired pattern, which is specified by a configuration register, *cfg* in figure 4(a). The overlapping of 3-ways adders inputs gives access to higher flexibility in multiple windows generation. Thus, the maximum vertical and horizontal overlapping is 5 pixels between two windows and 3 pixels between three windows, which increases combination possibilities. Nevertheless, the maximum window size depends on the number of matching costs to be processed simultaneously. For example, for a single matching cost, window size can vary from 2x2 to 15x15 pixels. In multiple windows mode, the maximum size is reduced to 11x11.

## 3.4 Matching Costs Analyzer

The purpose of matching costs analysis is to extract correct disparity and metadata from generated matching costs. This step uses simple operations like comparison or addition, and more elaborated operations like sorting or counting. However, the input data bandwidth is still high (9 matching costs per cycle) which implies the use of parallel processing units.

The Matching Costs Analyzer goal is to apply these operations on generated matching costs. We based the MCA architecture on sorting networks structure as it suits well to data analysis and exchange. Instead of simple compare-exchange operators, our network is composed of reconfigurable elements named ASCE (Add-Sub-Compare-Exchange). These processing elements have two inputs and two outputs and provide a set of operations summarized in table 1. Interconnection between ASCEs from a column to another is fully flexible by reconfiguration. It is also possible to plug inputs and outputs of the MCA together to create longer data paths. Figure 5 shows a MCA composed of 4 columns of 9 ASCEs each. In this figure, *cfg* specifies the configuration for interconnections and ASCEs functions. As the processing power is limited by the number

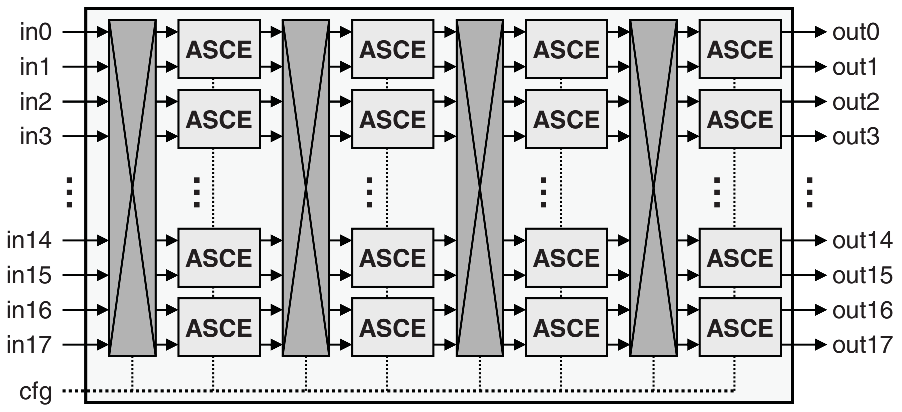


**Fig. 5.** The Matching Cost Analyzer is composed of reconfigurable elements (ASCE) and handles matching methods. Each ASCE's input can be connected to the previous column ASCE's output through a reconfigurable interconnection (crossed box).

**Table 1.** Operations supported by an ASCE. *a* and *b* are the inputs. *c* and *d* are the outputs. *Loop* indicates if the operation can be looped, *i.e.* *b* receives the data present the previous cycle in output *c*. In counter mode, *count* is the internal counter value.

| Name | Loop | Outputs | Name | Loop | Outputs |
|---|---|---|---|---|---|
| **ID** | yes | c=a, d=b | **Mm** | yes | c=max(a,b), d=min(a,b) |
| **ADD** | yes | c=(a+b), d=0 | **mM** | yes | c=min(a,b), d=max(a,b) |
| **CRb** | yes | c=count if a=1, d=b | **MC** | yes | c=max(a,b), d=cmp(a,b) |
| **CRbc** | yes | c=count if a=1 else c=b, d=0 | **mC** | yes | c=min(a,b), d=cmp(a,b) |
| **Necb** | no | c=a if b=0 else c=-a, d=a | **Subs** | no | c=(a-b), d=sign(a-b) |

of ASCEs, the allowed analysis complexity depends on the number of simultaneous reference pixels processed by the matrix, *i.e.* the number of independent processes executed.

### 3.5 Reconfiguration

The reconfiguration is done through the configuration port: each configuration register has its own address and thus, each module can be reconfigured independently. The bitstream total size is about 1800 bits (225 bytes). The architecture supports two reconfiguration modes. The first one is the static reconfiguration mode: each module is reconfigured once at the initialization of the system and no reconfiguration is needed while the application is running. Dynamic reconfiguration mode is also supported, as the whole architecture reconfiguration is done in less than 100 cycles. However, the frequency of reconfigurations is limited by performance requirements as bitstream loading implies penalties. The typical reconfiguration granularity is the image line: several processings can be alternately executed on each line, allowing hardware reuse. For instance, at first, the architecture is configured for Rank filtering. All the pixels of a given line are filtered and stored in a line buffer, which takes about 1300 cycles (640 cycles for each image). Then the architecture is reconfigured to perform disparity extraction on filtered pixels. The extraction step takes approximately 41000 cycles ($640 \times 64$ cycles). The reconfiguration overhead in this case study is less then 0.5% of the whole processing time.

## 4 Implementation Results and Case Studies

### 4.1 Real-Time Performances and ASIC Synthesis

An ASIC synthesis of the proposed architecture has been carried out, with ST 65nm Low Power technology, at 200MHz. This synthesis permits us to obtain a good estimation of the surface for each reconfigurable module : 0.7 mm$^2$ for the WPM, 0.3 mm$^2$ for the FRT and 0.5 mm$^2$ for the MCA. The total surface is around 1.5 mm$^2$[4], which satisfies the embedded constraints of ADAS. At 200MHz frequency, and considering a VGA stereo pair with a disparity range of 64 pixels, the generation of 1 to 9 matching costs per cycle gives access to a frame rate from 10 fps to 87 fps. It corresponds to a maximum performance of 1.7 GPDS, which is quite high regarding the flexibility provided. Besides, the REEFS architecture is fully compliant concerning real-time requirements for Advanced Driver Assistance Systems which are usualy set to 40 fps.

## 4.2 Configuration Cases

In the following, two configuration examples are presented to show the flexibility and the real-time abilities of the architecture. The first example enables a high frame rate with basic matching method. In the second example, both matching costs generation and matching method are more complex, which implies a lower frame rate. For each case, results on performance (frames per second) and utilization level are presented. The utilization level represents the number of reconfigurable processing elements used in each module. We consider that the left image is the reference image.

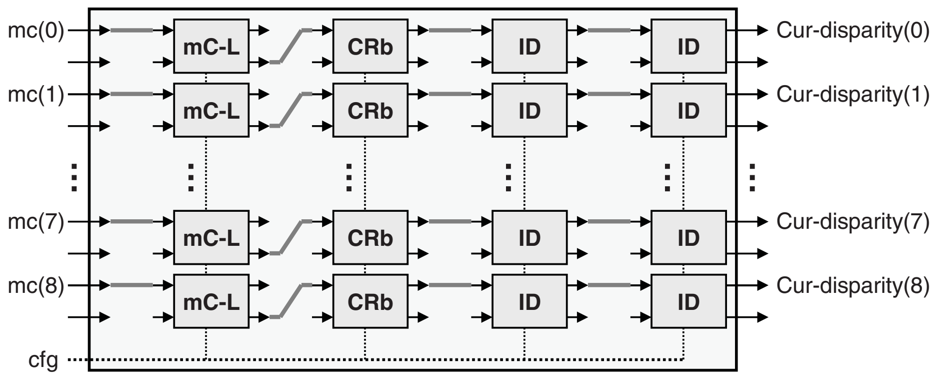


**Fig. 6.** Example of a MCA module's configuration that allows the processing of WTA method on 9 independent matching costs. The reached frame rate is 87 fps. The suffix *-L* indicates a looped operation.

In the first example, WPM and FRT modules are configured to generate 9 independent CENSUS matching costs on 5x5 windows, without overlapping. The MCA module applies a Winner-Take-All method simultaneously on each cost, as shown in figure 6. Thus, for each MCA line, the first ASCE compares the cost *mc(k)* in input with the current minimum cost (stored inside the element). If a new minimum value is detected, the second ASCE, which is configured as a counter, updates its output with its internal counter value. The result *cur-disparity(k)* represents the gap that gives the best matching cost for reference

[4] The surface occupied by registers is about 28% of the total surface.

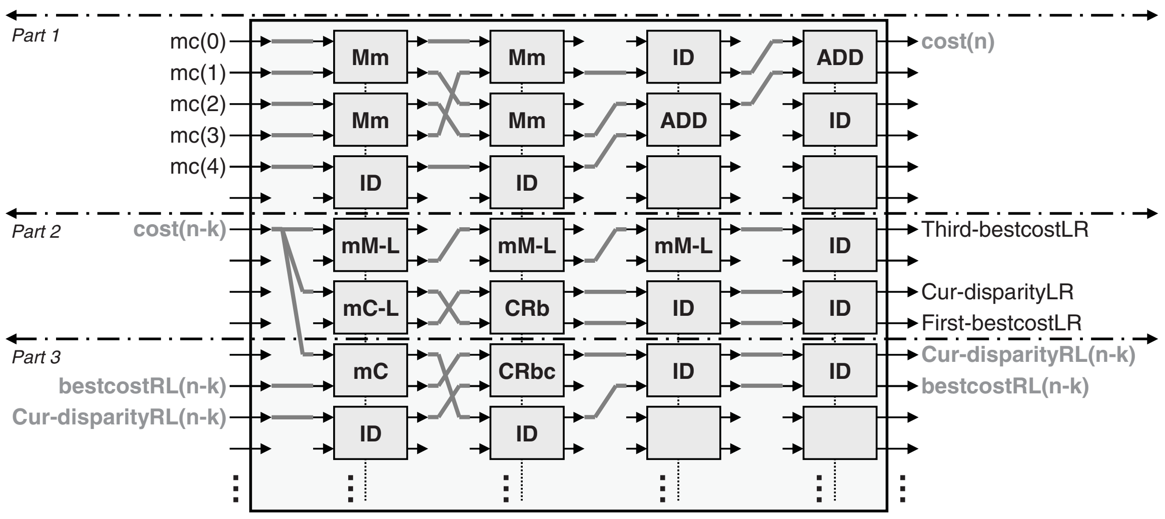


**Fig. 7.** Example of multiple windows costs generation followed by the extraction of disparity and metadata for symmetry constraint checking and confidence level calculation. Inputs and outputs in bold are plugged together.

pixel $k$. In this configuration, the theorical frame rate is 87 fps. Concerning the utilization level, both WPM and MCA are used at 100%, and the FRT is used at 72%.

The second configuration example enables the processing of one SAD matching cost on multiple windows, as proposed in [4]. The FRT module generates one central 7x7 window and four 7x7 peripheral windows, with overlapping. Figure 7 shows the MCA's configuration. In a first step, the 2 minimum peripheral costs are extracted and summed with the central cost to generate a combined cost (part 1). Then, this combined cost is used in a second step for the extraction of the left reference disparity (cur-disparityLR) with WTA method and the first and third best related matching costs (part 2). The combined cost is also used for extracting the right reference disparity (cur-disparityRL) with WTA method (part 3). After 64 shifts, all metadata are processed by the processor to filter wrong disparities with symmetry constraint checking and confidence level calculation. As one matching cost is processed each cycle, the frame rate is 10 fps. Only 54% of the WPM's RPEs are used while the FRT is still used at 72%. The MCA module is used at 67%.

## 5 Conclusion

Disparity maps generation is a key stage in real-time embedded stereovision applications. A high number of algorithmic solutions exists and the choice depends on the targeted application and applicative constraints. Thus, we propose a reconfigurable architecture that enables a high algorithmic flexibility. The REEFS architecture provides flexibility on the metrics, the size and the shape of correlation windows, and on matching methods. Besides it answers real-time requirements as it can produce 640x480 disparity maps at a maximum frame rate of

87 fps. The three main reconfigurable modules have been synthesized and the total surface is about 1.5 mm$^2$. The characteristics of our architecture satisfy the requirements of Advanced Driver Assistance Systems.

In the future, this architecture will be integrated in a System on Chip for ADAS. This SoC will integrate high-level processing units for disparity map analysis: road-plan extraction or obstacle localisation for instance. In this context, it will be possible to scale and to parallelize the current architecture to increase the maximum frame rate when using large windows or performing complex analysis. The execution of other types of area-based image processing will also be considered in this SoC.

## References

1. Ambrosch, K., Kubinger, W., Humenberger, M., Steininger, A.: Flexible hardware-based stereo matching. EURASIP J. Embedded Syst. 2008, 1–12 (2008)
2. Cuadrado, C., Zuloaga, A., Martin, J., Lazaro, J., Jimenez, J.: Real-time stereo vision processing system in a fpga. In: Proc. IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics, pp. 3455–3460 (2006)
3. Fua, P.: A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features. Machine Vision and Applications 6, 35–49 (1993)
4. Hirschmüller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. Int. J. Comput. Vision 47, 229–246 (2002)
5. Jacobi, R., Cardoso, R., Borges, G.: Voc: a reconfigurable matrix for stereo vision processing. In: Proc. 20th Int. Parallel and Distributed Processing Symposium (2006)
6. Kraft, G., Jonker, P.: Real-time stereo with dense output by a simd-computed dynamic programming algorithm. In: PDPTA 2002: Proc. of the Int. Conf. on Parallel and Distributed Processing Techniques and Applications (2002)
7. Muhlmann, K., Maier, D., Hesser, R., Manner, R.: Calculating dense disparity maps from color stereo images, an efficient implementation. In: Proc. IEEE Workshop on Stereo and Multi-Baseline Vision (2001)
8. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. of Computer Vision 47, 7–42 (2002)
9. Woodfill, J., Gordon, G., Buck, R.: Tyzx deepsea high speed stereo vision system. In: Proc. Conference on Computer Vision and Pattern Recognition Workshop CVPRW 2004, p. 41 (2004)