# A VLSI ARCHITECTURE FOR REAL TIME OBJECT DETECTION ON HIGH RESOLUTION IMAGES

*M. Cavadini* cavadini@ife.ee.ethz.ch Tel. +41 1 632 2749

*M. Wosnitza* wosnitza@ife.ee.ethz.ch Tel. +41 1 632 3391

*M. Thaler* thaler@ife.ee.ethz.ch Tel. +41 1 632 6558

*G. Tröster* troester@ife.ee.ethz.ch Tel. +41 1 632 3964

Swiss Federal Institute of Technology (ETHZ)
Electronics Laboratory, Gloriastr. 35, CH 8092-Zurich, Fax +41 1 632 1210

**ABSTRACT**

*This paper describes a VLSI-based SIMD multiprocessor system for the implementation of a set of basic object detection algorithms. The system architecture takes advantage of modern fast EDRAM-technology to support the communication requirements of 800 Mbytes/s between main memory and processors imposed by high resolution images. A specialized processing element (PE) architecture for implementation in VLSI which efficiently implements the basic set of algorithms is presented. The performance of a single PE is discussed with respect to the different algorithms. A system consisting of 4 processing elements realized in* $0.6\mu$ *CMOS-technology is able to localize a 128x128 pixel template in a 1024x1024 pixel image at a rate of 10 frames/second (sustained performance* $2.1 \cdot 10^9$ *Ops/s).*

## 1 Introduction

The availability of powerful, vision based systems for object detection is of growing interest. Medical imaging and robot vision are two of the most representative examples. Depending on the application, a combination of low- and high-level image processing algorithms has to be implemented to achieve the highest possible reliability for object detection. In general, the computational requirements can only be satisfied with special purpose image processing systems. At the same time, todays image sensor suppliers produce standard digital cameras providing 1024x1024 pixels resolution which in addition increases the need for high performance image processors.

The main goal of this work is to provide a system which is capable to implement real time localization of 128x128 pixel objects on image formats up to 1024x1024 pixels.

## 2 Object detection on irregular background

Studying the most important object localization algorithms has shown the importance of the well known normalized cross correlation function (NCCF) [1]. To cover rotation invariant object detection, the method of invariant moments (MI) is a promising approach to extract arbitrarily rotated features from a given image [3].

In many cases, the object to be found only sparely fills the rectangular template area and many template pixels do not represent any object information. In addition background pixels may not correspond with the actual image background. Thus 'template background' will be included in the correlation result and depending on the difference to the actual object background may have an adverse influence on the correlation peak and object position. Simulations based on representative image material have shown that by out-masking background template pixels within the correlation process, the precision and reliability of the correlation result can significantly be improved [2]. Since this procedure has an analogy with the *"blue-screen-technique"* used in television we have adopted this notation.

As a matter of fact implementation of both algorithms, NCCF and MI under consideration of the *"blue-screen-technique"* is a must.

## 3 VLSI implementation of object detection algorithms

The required number of multiplications and additions for the NCCF and the MI algorithm are listed in Tab. 1.

Facing todays technology, these high computational rates can only be achieved with large and expensive computing systems. To implement the NCCF and the MI-algorithm in a compact and low cost image processing systems the requirements on the computational rate must be reduced as much as possible.

|  | NCCF | MI |  |
|---|---|---|---|
| image size | MAC | ADD | MULT |
| 256x256 | $545 \cdot 10^6$ | $2.7 \cdot 10^9$ | $5.5 \cdot 10^9$ |
| 512x512 | $4.9 \cdot 10^9$ | $24 \cdot 10^9$ | $49 \cdot 10^9$ |
| 1024x1024 | $26.4 \cdot 10^9$ | $132 \cdot 10^9$ | $264 \cdot 10^9$ |

Table 1: Computational requirements for a single frame (template size 128x128, *"blue-screen-technique"* )

A first option to decrease computational requirements is to reduce the spatial resolution of the incoming image by applying a sub-sampling pyramid. Furthermore, according to the specific characteristics of the NCCF- and

MI-algorithm, additional dedicated methods to reduce the computational requirements do exist.

Concerning the MI-algorithm, large reduction in computation rate can be achieved by restricting to binary images. In this case most of the weights can be stored in look-up-tables, eliminating all multiplications (Tab. 2) [7].

As stated in [1], computation of the NCCF is realized by dividing the unnormalized cross correlation sum (UCCF) by corresponding energy terms. A very efficient way to reduce computational requirements is the computation of the UCCF via the FFT algorithm as e.g. reported in [5], which reduces the requirements from $O(N^2 \cdot n^2)$ operations to $O(N \log_2 N)$ (Tab. 2) ($N$ being the geometric size of the rectangular image and $n$ of the template respectively). Considering the local data transfer rates, additional savings can be achieved within the PE by using a radix-4-FFT butterfly structure [6].

The two paths of the NCCF algorithm, computation of the UCCF and computation of the energy terms, are shown in Fig. 1. An efficient way to implement the *"blue-screen-technique"* is to assign the value "0" to all background pixels of the template in order to eliminate the influence of the background pixels on the UCCF.

Concerning high precision object detection, simulations have clearly shown that out-masking of irrelevant template pixels has also to be done on the second path of the NCCF algorithm, i.e. while computing the energy terms [2]. Unfortunately in this case data dependent processing is necessary, since only those pixels have to be squared and accumulated whose corresponding template pixels contain relevant information. Therefore the optimized computation of the energy terms proposed in [1] can not be applied to the *"blue-screen-technique"*.

Since the computation of the energy terms can be realized by a 2D filter-function applied to the squared camera image, energy terms can also be computed via the FFT, achieving at the same time a reduction in computational requirements. In this case, all image pixels have to be squared and the template has to be binarised. The resulting algorithm flow for the NCCF including the *"blue-screen-technique"* is shown in Fig. 1.

| image size | Rad-4-FFT ADD | Rad-4-FFT MULT | MI ADD |
|---|---|---|---|
| 256x256 | $2.6 \cdot 10^6$ | $6.7 \cdot 10^6$ | $3.8 \cdot 10^9$ |
| 512x512 | $12.1 \cdot 10^6$ | $30.4 \cdot 10^6$ | $34 \cdot 10^9$ |
| 1024x1024 | $51.4 \cdot 10^6$ | $113.2 \cdot 10^6$ | $185 \cdot 10^9$ |

Table 2: Optimized algorithms, single frame, *"blue-screen-technique"* [ops/frame]

Looking at the reduced computational requirements (Tab. 2) it is still clear, that real-time object detection can only be supported by using several processors which work in parallel.

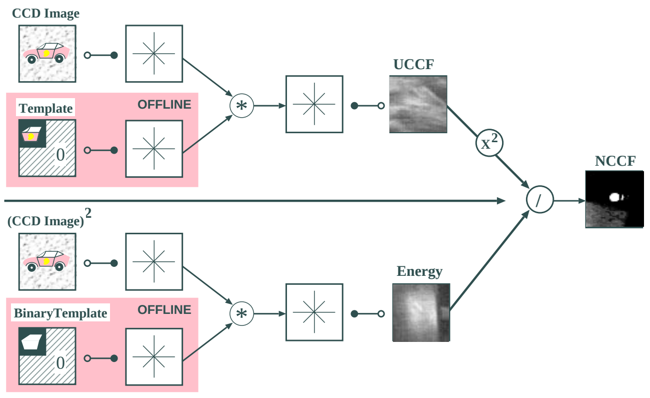

Figure 1: Flow chart of NCCF algorithm computed via FFT, including *"blue-screen-technique"*

## 4 System Architecture

The considered algorithms have been analyzed with respect to their implementation on a multiprocessor system. A common feature of the algorithms is the very high degree of data parallelism which leads to a SIMD architecture.

Special considerations have to be set on the parallelization of the 2-D FFT because of its data access scheme. Following the separation approach the 2-D FFT is split into 2 consecutives 1-D FFT's where the image is first transformed in row order and then in column order (Fig. 2). This can be realized by transforming all rows, then transposing data and again transforming all rows. Fig.2 shows a flow graph for the UCCF computation of Fig.1 based on this procedure.

This imposes strong requirements on the memory organization which must guarantee independent data access for each PE for optimal SIMD operation.

The two main categories of memory organization for a SIMD machine are the shared and distributed architectures [4]. Regarding the FFT, a distributed memory architecture offers optimal data independence for the transformation of the image rows while a large communication overhead results for the column transformation. On the other hand the shared memory concept makes all the data accessible for each PE, but suffers from heavy bus contention problems. Combining the advantages of both concepts, a mixed shared-distributed memory architecture shown in Fig.3 has been developed.

The image data is partitioned and then distributed over independent memory modules. The image rows are accessed from each PE's over a horizontal bus system (HBS) for 1D-FFT computation of rows. The HBS represents the distributed memory character of the architecture. Concurrently already transformed rows are uploaded from the PE's local memories into the memory modules over a vertical bus system (VBS). The VBS implements the shared memory system aspects. During the upload process over the VBS, transposition of the entire dataset is achieved. Thus with a second 1-D FFT in row order (data access again over the HBS) the image column transformation is computed. In order to support the different algorithms steps (see Fig.2), several memory banks are connected dynamically to the two bus systems through a crossbar switch. The image transposition during upload is realized by a controller which generates the main memory addresses and concurrently enables the PE's to write onto the data bus. The implemented relationship between generated memory addresses and data bus access by the PE's realizes the image transposition.

The multiplication of the transformed images by the template can efficiently be computed directly on the PE, where the respective template values have been prepared during the second 1-D FFT computation (column transformation). After the multiplication phase the computation of the inverse FFT of the current row in the PE can start right away without communication.

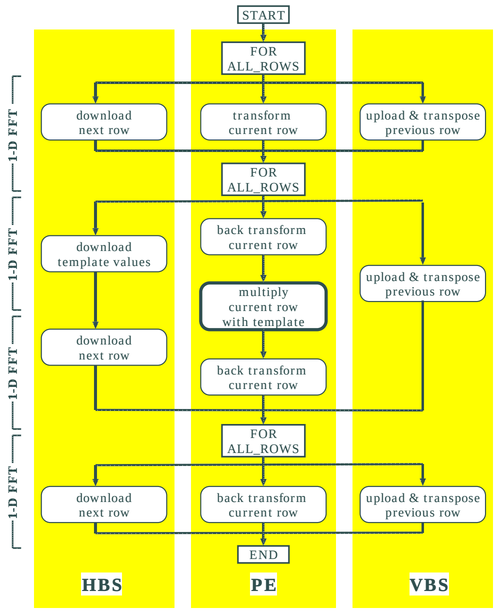
START
FOR ALL_ROWS
download next row
transform current row
upload & transpose previous row
FOR ALL_ROWS
download template values
back transform current row
multiply current row with template
upload & transpose previous row
download next row
back transform current row
FOR ALL_ROWS
download next row
back transform current row
upload & transpose previous row
END
1-D FFT
1-D FFT
1-D FFT
1-D FFT
HBS
PE
VBS

Figure 2: UCCF via FFT scheduling

Each PE contains two local memory modules in order to run transformation and communication in parallel. The 1024 points complex FFT on the PE takes $80\mu s$. Using 32 bit words for the real and imaginary part the required bandwidth is $200 Mbytes/s$ ($100 Mbytes/s$ for downloading the next row and $100 Mbytes/s$ for uploading the transformed row). The most stringent requirements are imposed by the VBS which has to support $400 Mbytes/s$, because 4 PE are connected to it. Even using 32 bit wide data busses, a memory cycle time of $10ns$ is necessary. The cost of current SRAM memory technology makes it impossible to be used for the system main memory.

One of the features of the transposition process over the VBS is that the PE's can generate 8 word-bursts which have to be transfered to the same memory row. Enhanced DRAM (EDRAM) technology can take advantage of this fact and thus achieves an average cycle time of $20ns$ which results in bandwidth of 200 $Mbytes/s$ (32 bit/word). Therefore the VBS consist of two data buses in order to achieve the required $400 Mbytes/s$. The total amount of memory which is required to compute the different steps of the flow chart of Fig.1 is 32 MB.

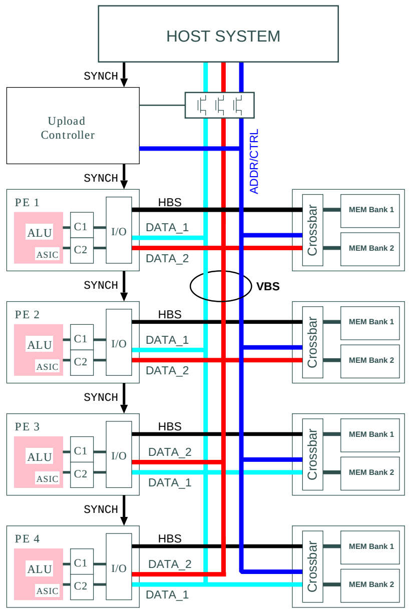

Figure 3: System architecture

Each PE has to compute a 1-D FFT on one quarter of the image lines. Further reduction in computing time is achieved by assuming that the input image consist of real data. This leads to data symmetries which can be exploited to compute the FFT on only half of the data. Thus, using a specialized data packaging concept, in the case of 1024x1024 pixels images each PE has finally to transform 128 rows. Assuming $80\mu s$ for one row computation, a 1-D FFT of the image can be computed within $10.2ms$ and a total cycle time of $81.6ms$ for the NCCF via FFT is achieved.

In addition, the distributed memory character of the system can be used to support the MI algorithm where each PE actually needs a portion of the image data and operates independently on it. The memory space required by the FFT enables to download a whole image to each main memory module. This avoids problems imposed by data along the borders of the image sections which has to be shared but belongs to different PE's.

In the same way the system supports standard NCCF computation. In this case the template values are downloaded into the local memories of the PE's. The standard NCCF computation is e.g. useful when a template has to be searched for in a very small image section.

A special property of the presented bus architecture is that the memory modules are directly connected to the bus without any need of a communication controller.

Thus the memory modules can be accessed over the VBS by standard addressing schemes used by common "of the shelf" processors without additional hardware overhead. The host system, which is responsible for the I/O and controlling tasks has also to compute the operations involved in the NCCF algorithm (Fig. 1) such as squaring UCCF values, normalizing the UCCF values and finding the local maximums.

## 5 Processing Element

Each processing element of Fig. 3 consists of a 32bit fix point arithmetic unit (ALU), two local memories and a control unit.

Implementation of two local caches will guarantee concurrency among computation and communication processes and thus provides high throughput. The control unit controls data flow as well as program flow within the ALU for the different algorithms.

The basic structure of the ALU is shown in Fig. 4. The ALU consists of a register file for storing intermediate results and data reorganization, two fast complex adder/subtracter units (CADD) and a highly pipelined complex multiplier (CMUL).

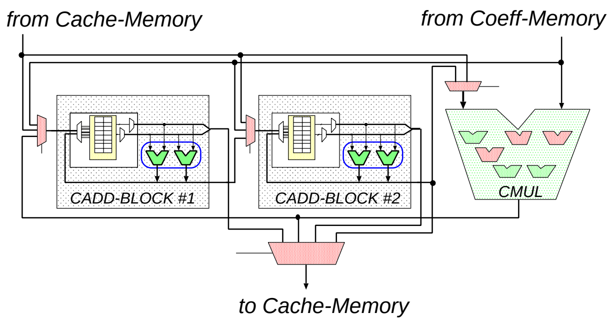

Figure 4: Block diagram of fix point arithmetic unit, optimized for radix-4 computation

The ALU design shown in Fig. 4 is optimized for efficient computation of a radix-4-FFT butterfly structure. The two embedded CADDs in the ALU and the highly pipelined CMUL (Fig. 4) compute a radix-4 butterfly structure within 4 clock cycles.

| Module | FFT | NCCF | MI |
|---|---|---|---|
| CADD | 100% | 50% | 100% |
| CMUL | 75% | 75% | 0% |

Table 3: Utilization of CADD and CMUL

To increase data throughput of the ALU computing the NCCF, both 32bit multipliers within the CMUL can be configured to compute two 16bit products within a single step. Thus the CMUL can be configured either in 'FFT-mode', performing complex multiplications, or in 'NCCF-mode'.

Computation of the MI-algorithm is based on calculation of moments $m_{pq}$ up to third order ([3]). By iterative computation of the $m_{pq}$, taking into account weights stored in a LUT, updating of all required weights in a single clock cycle is possible.

The utilization of CADD and CMUL with respect to the implemented algorithms is listed in Tab. 3.

The computational performance of the arithmetic-unit is shown in Table 4. The specifications are based on the assumption of a 66MHz data-cycle, which is typical for commercially available $0,6\mu$ CMOS-processes.

| Algorithm | computation power |
|---|---|
| NCCF | 16100 local 128x128 cross-correlations /sec |
| FFT | 12500 1D-1024-point FFT's /sec |
| MI | 1000 local 128x128 image segments /sec |

Table 4: Peak performance of a single PE at 60MHz

## 6 Conclusions

In this paper a VLSI-based SIMD multiprocessor system for real-time object detection has been propose targeting at high resolution images up to 1024x1024 pixels. The system supports normalized cross correlation function NCCF and the moment invariant methods which are considered as fundamental. The NCCF is implemented using radix-4 FFT algorithm to speed up computation of images larger than 256x256 pixels .

A scalable SIMD multiprocessor system based on special purpose processing elements (PE's) and on a dedicated memory architecture provides the necessary computational resources as well as the required fast access to image data. A single PE, running at a 66MHz clock rate computes the FFT algorithm on a 512x512 pixel image in real time (10 frames/s), whereas 4 of these PE's process a 1024x1024 pixel image at the same rate with a search template of 128x128 pixels.

## References

[1] P. Aschwanden: *"Experimental Comparison of Correlation Criterions in Image Analysis" (german)*, PhD-Thesis , ETH-Zurich, Hartung-Gorre Verlag, Konstanz, 1993

[2] M. Cavadini, M. Wosnitza: *"Technical Report No: 04/95"*, Electronics Laboratory, ETH-Zurich, April 1995

[3] S. O. Belkasim, M. Shridhar, M. Ahmadi *"Pattern recognition with moment invariants: a comparative study and new results"*, Pattern Recognition, Vol. 24, No. 12, pp. 1117-1138, 1991

[4] Volkers H. *"A contribution on memory architectures for image processing programmable mulitprocessor (german)"*, VDI Verlag,Nummer 152, 1992

[5] J .W .Cooley, J.W.Tukey: *"An Algorithm for the Machnine Calculation of Complex Fourier-Series"*, Math. Comput., Vol. 19(90), pp. 297-301, 1965

[6] E. Bidet, et. al.: *"A Fast Single-Chip Implementation of 8192 Complex Point FFT"*, IEEE Journal of Solid State Circuits, Vol 30, No 3, pp. 300-305, March 1995

[7] M. Wosnitza, M. Cavadini, M. Thaler, G. Tröster: *"A Scalable VLSI Architecture for Real Time Object Detection"*, ISCAS-96, Vol. 2, May 13-15, Atlanta, GA