Concept and model of a multiprocessor system for high resolution image correlation

Author(s): Cavadini, Marco

Publication Date: 1999

 Permanent Link: https://doi.org/10.3929/ethz-a-002077547

Rights / License: In Copyright - Non-Commercial Use Permitted
Concept and Model of a Multiprocessor System for High Resolution Image Correlation

A dissertation submitted to the SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH for the degree of Doctor of Technical Sciences presented by MARCO CAVADINI dipl. el.-Ing. ETH born 9 March 1969 citizen of Davesco, TI

accepted on the recommendation of Prof. Dr. Gerhard Tröster, examiner Prof. Dr. Lothar Thiele, co-examiner

1999
Alla mia famiglia
   e a Paola
Abstract

In this work the correlation based image localization problem is targeted towards implementation for real-time operation on high-resolution images. Its main contribution resides in a global approach to the 2D correlation problem which concurrently considers the algorithm, the system architecture and its implementation. As a result the relevance of image correlation traditionally limited to the image processing domain can be expanded into applied machine vision applications.

Based on an enhanced frequency based correlation algorithm, a scalable multiprocessor system is presented which achieves real time image correlation with performances exceeding currently available solutions, by an order of magnitude in terms of frame-rate, input image resolution and result accuracy. The system compactness is kept under high consideration.

A key aspect of the multiprocessor is a reconfigurable memory architecture where the peculiar data access patterns of the algorithm are optimally supported by means of an adapted reconfiguration of the parallel memory architecture at runtime. Additionally by the use of dedicated data communication mechanisms, the characteristics of emerging dynamic memory devices are taken into account, leading to optimal memory bandwidth exploitation.

The implementation with state of the art technology based on different system integration approaches is explored and a cost model of the system permits an early cost/performance evaluation of its final configuration.
Zusammenfassung


Kernstück der Multiprozessorarchitektur ist eine rekonfigurierbare Speicherorganisation, die aufgrund ihrer problemangepaßten Stuktur die Datenkommunikationsvorgänge mittels dynamischer Anpassung optimal unterstützt. Durch eine flexible Gestaltung der Datenkommunikation kann das System optimal auf die Anforderungen von modernsten dynamischen Speicherbausteinen angepaßt werden, woraus eine maximale Auslastung dieser Komponenten resultiert.

Acknowledgment

I would like to express my gratitude to my advisor Prof. Dr. G. Tröster for permitting me a very high degree of freedom with my research and for steadily reinforcing my motivation. Thanks also to my associate advisor Prof. Dr. L. Thiele for co-examining and for his valuable comments on this thesis.

Special thanks to my colleague Matthias Wosnitza for his precious cooperation in technical matters during the whole project without which this work would not exist.

Thanks to Etienne Hirt and Michael Scheffler, members of the multi-chip module group at Electronics Lab for the helpful suggestions on advanced integration techniques.

And at last but not least a thought goes to Paolo Orsatti and Francesco Piazza for the fruitful discussions during our breaks as well as to my girlfriend Paola for her spiritual support.

Zürich, April 1999

M. Cavadini
# Contents

Abstract i

Zusammenfassung iii

Acknowledgment v

1 Introduction 1
1.1 Motivation 1
1.2 Approach 2
1.3 Parallel Systems in Computer Vision 3
1.4 Design Environment 4
   1.4.1 Methodology 4
   1.4.2 Used Representations 6
   1.4.3 Dedicated Processor Design 8
1.5 Work Outline 8

2 High Resolution Image Correlation 11
2.1 Object Localization 11
2.2 Normalized Cross Correlation Function 12
2.3 Blue Screen Technique (BST) 15
2.4 Frequency Based BST Computation 17
   2.4.1 Concept 17
   2.4.2 FB-BST Computational Requirements 19

3 System Architecture 23
3.1 Concept 23
3.2 Frequency Domain Engine 26
   3.2.1 Extended 2D FFT indirect method 27
   3.2.2 SPMD Parallelization Approach 30
3.3 Reconfigurable Memory Architecture 33
3.3.1 Shared vs. Distributed Memory .................. 33
3.3.2 RMA Concept .................................... 35
3.3.3 FB-CCF mapping on RMA ....................... 37
3.3.4 Requirements to the RMA processing element .... 41
3.4 Real Valued FFT .................................. 42
  3.4.1 Concept ........................................ 42
  3.4.2 Application to the RMA ......................... 45
3.5 General Purpose Processing Engine ................ 48
  3.5.1 Maximum Extraction Procedure ................. 48
  3.5.2 Parallelization of GPE ........................ 51
3.6 System Level Task Scheduling ........................ 51

4 RMA Implementation .................................. 55
  4.1 Image Transposition on VBS ....................... 56
  4.2 Memory Bandwidth Exploitation .................... 58
    4.2.1 Dynamic Memory Enhancements ................. 58
    4.2.2 Memory Efficient Transposition ............... 60
    4.2.3 VBS Token Passing Network .................... 63
    4.2.4 Alternative to the VBS Transposition Implemen-
        tation ....................................... 65
  4.3 VBS Throughput Improvement ....................... 66
    4.3.1 RMA Scalability ................................ 66
    4.3.2 VBS Parallelization ........................... 67

5 Processors and Communication Controller ............. 71
  5.1 The SPITFFIRE .................................... 71
    5.1.1 ALU ......................................... 73
    5.1.2 Local Memory and I/O Port .................... 74
    5.1.3 Data flow processing during I/O ............... 75
    5.1.4 Programming ................................ 75
  5.2 SPITFFIRE Impact on FDE Task Scheduling .......... 76
  5.3 GPE Processor ................................... 77
    5.3.1 The SHARC DSP ................................ 77
    5.3.2 Task Synchronization and Control ............... 78
  5.4 Communication Controller ........................ 79
  5.5 BE Management ................................... 82

6 MPIC Model ........................................... 85
  6.1 Virtual Prototype ................................ 85
    6.1.1 Design Strategy ................................ 85
    6.1.2 FDE VHDL Model ............................... 88
6.1.3 Virtual Prototyping Output .................................. 91
6.2 Performance Models .............................................. 93
  6.2.1 HBS and VBS Bandwidth .................................... 93
  6.2.2 Programmable Logic Performance ............................. 94
  6.2.3 HBS Performance Model ....................................... 95
  6.2.4 VBS Performance Model ....................................... 96
  6.2.5 FDE Performance Model ....................................... 97
  6.2.6 GPE Performance Model ....................................... 98
  6.2.7 MPIC System Performance Model ............................... 98

7 System Performance ................................................. 101
  7.1 Performance Evaluation ......................................... 101
    7.1.1 2D FFT Performance ....................................... 109
    7.1.2 Frame Rate Driven System Configuration ..................... 110
  7.2 System Performance Comparison .................................. 112
    7.2.1 BST and NCCF ............................................... 112
    7.2.2 2D FFT .................................................. 114

8 FDE Integration and Cost Evaluation ................................ 117
  8.1 MCM FDE Realization ........................................... 118
    8.1.1 System Partitioning ....................................... 118
    8.1.2 MCM early dimensions estimation ............................ 120
  8.2 Full FDE PE integration ........................................ 123
    8.2.1 System on a Chip .......................................... 124
    8.2.2 Multi-Packed FDE Module .................................. 127
  8.3 Costs vs. Performance Evaluation ................................ 128

9 Outlook and Conclusions ............................................ 135
  9.1 System Performance Scaling ..................................... 135
    9.1.1 Scaling of the Single Elements .............................. 135
    9.1.2 Performance Prediction ..................................... 136
  9.2 RMA as general VSP architecture ................................ 138
  9.3 Application to Other Algorithms ................................ 139
  9.4 Conclusions .................................................. 141

A Used Abbreviations and Parameters .................................. 143
Chapter 1

Introduction

1.1 Motivation

Computer vision is a digital signal processing discipline constantly enlarging the spectrum of its applications and can nowadays supply a relevant contribution where the automation of technological processes is required. Application fields such as industrial automation, biomedical technologies, aerospace industry and robotics, are the most important domains where computer vision is constantly increasing its presence.

Due to the computational power delivered by current computer systems in conjunction with the improving image sensor technology, the implementation of classical image processing algorithms has become faster and more reliable at reasonable costs. Influenced by this situation the traditional computer vision end user community is numerically expanding and above all requiring the implementation of tasks of growing complexity. As a consequence the pressure exercised on the computer developers will not lessen in the near future.

The fact that a relevant part of the computer vision applications does involve real time operation on images with a steady growing resolution, further increases the requirements imposed on the image processing systems. Looking at state of the art image sensor technology and at the target environments of computer vision applications, it can be established that real-time high resolution involves the the processing of image resolutions in the range 256x256...1024x1024 pixels with a rate of up to 30 frames/s.
Object localization belongs to a category of image processing algorithms laying between the low-level processing (dealing with single pixels or small kernels) and the high-level processing (analyzing extracted image features), which up to now lacks of efficient real time implementations, particularly when high image resolution is involved. The important relevance of object localization in computer vision applications imposes to make this function available with appreciable robustness and reliability. Even though several efforts have been undertaken with respect to the theoretical aspects of object localization, only a few of them have lead to results which go beyond a software simulation of the algorithms. In many cases the proposed solutions show accurate and reliable mathematical results but, mainly due to their complexity, cannot be used for real time implementations on state of the art computer technology.

As a consequence there is the need to develop a dedicated solution to this problem which, from the beginning, approaches the image localization problem targeting at its real time implementation on a dedicated computer architecture.

1.2 Approach

The only way to advance beyond the solutions presented so far, is to maintain a strong interaction between the optimizations carried out at algorithm level with the ones undertaken during the processing system development. The high computational requirements resulting from the complexity of the algorithm do not make it possible to approach these aspects separately. According to this a dedicated processing element is developed in conjunction with a dedicated parallel system architecture which aims to support an optimized version of a localization method.

The chosen image localization method makes use of the cross correlation function which is slightly modified and computed in the frequency domain. The main benefit of this strategy is the massive reduction of the computational requirements compared to the ones imposed by a computation in the space domain. This effect is particularly amplified by the fact that large images are considered. Further enhancement with respect to the standard cross correlation algorithm concern lower constraints for the template form and result accuracy.

In this work the attention is concentrated on the development of the
1.3 Parallel Systems in Computer Vision

For a useful consideration of the presented work, the general constraints involved with the envisaged computer vision applications have to be considered. In fact the parallelization approach can’t ignore the final environment where the image processing system will be used. In most cases the typical computer vision end user asks for a system showing the highest possible compactness in order to place the image processing engine close to the image sensor. Industrial automation and robotics are very important applications which are concerned with these kinds of working conditions. Furthermore the image processing engine also has to provide stand alone functionalities so that no further resources need to be involved for embedding it into the target system.

In contrast to computer vision, general purpose parallel processing users merely ask for very high computational power for scientific, commercial or military computations and simulations [Gunzinger 95] [Bell 89] [Bell 94]. Compactness and price play in this case a less relevant role.

For the parallelization approach in computer vision, the compactness issues result in pushing push the utilization of the system resources to the highest level. In other words the sustained performance of the system has to narrow the peak performance given by the theoretical
performance of a single processor multiplied by the number of processors present in the system. Thus the number of used electronic components as function of the system specifications is minimized, leading to the highest compactness.

Obviously these kind of remarks are pertinent to every parallel architecture development, but in the end it is a question of setting priorities. Whereas in the general parallel processing domain, more sustained performance can be achieved by increasing the number of processors, in computer vision few of them have to be allocated and the performance has to be achieved through their maximal exploitation. The price of this maximal exploitation is mainly paid in terms of limited system flexibility. The specificity of the application gives very good optimization opportunities because the development can be based on a priori known computational steps and data access patterns, aiming for maximal load per logical gate.

Thus, the metrics driving the development of the multiprocessor system will have the highest considerations of performance/resource optimizations issues, with also some influence played by what is the final cost. Other metrics such as power consumption or fault-tolerance are not considered.

1.4 Design Environment

1.4.1 Methodology

The system development is characterized by a very high complexity which largely exceeds the one given by the sum of the complexities of the single algorithm tasks. In fact the heterogeneity of the possible implementations for every single functional block of the algorithm, in practice expands the design space with a further degree of freedom, producing a wide range of possible algorithm implementations which have to be evaluated with respect to their effectiveness in terms of performance and cost.

The concurrent development of system components realized with software (SW) programmable resources interacting with more or less specific hardware (HW) units goes under the definition of HW/SW codesign [Teich 97]. This discipline is steadily increasing its relevance in modern electronics design, providing methodologies aiming to manage the growing complexity of state of the art digital systems development within computer aided design (CAE) environments.
While HW/SW codesign methodologies involve each level of digital system design, from the highest abstraction level down to the transistor level, in this work most considerations are limited to the system level development. It has to be pointed out that while at chip level automated design methodologies show nowadays a satisfying level of maturity, lots of progress still has to be made at system level where no CAE tool can provide a complete and comfortable design environment. Nevertheless it is common to apply a design paradigm consisting of three main activities [Gajski 94]:

**Specification:** in a first stage of the development process an executable representation of the whole system is created. This has to provide the description of the system functionalities as well as the possibility to automatically verify critical system parameters. As a result, the effectiveness of the different solutions produced in the exploration phase can be compared.

**Exploration:** this step evaluates the systems functional sub-blocks with respect to their implementation on different platforms: the whole spectrum ranging going from programmable processors to application specific integrated circuits is considered. In this phase the complexity of the design can assume an unmanageable complexity since every system partitioning implies a considerably different implementation. Thus there is the need to foresee the fundamental repercussions of different platform on the system parameters, in order to limit the number of the evaluated solutions.

**Refinement:** according to the partitioning produced in the exploration phase, the specification is distributed onto the different HW and SW components. Subsequent exploration and refining iterations can now take place at sub-block level, producing a complete structural system description building the starting point for a subsequent successful system synthesis.

This design methodology has been exploited in order to allow a concurrent optimization of algorithms, system architecture and processor architecture. In particular concurrent developments of system and processor architecture have been carried out.
1.4.2 Used Representations

Formal representation methods are used in this work in order to build models at different abstraction levels, which efficiently represent the complexity of the computing and controlling processes involved by the parallel implementation of the localization algorithm. The use of modern modeling paradigms allows us to have a concrete representation of the system functionalities and to detect the best parallelization approach independent of the future system implementation. Furthermore, an adequate modeling builds the basis of the specification-refinement loop previously introduced.

At system level, the graphical modeling approach gives better results with respect to the procedural description with imperative hardware description languages (HDL's). In parallel digital signal processing, data-flow graphs are a widely used graphical representation. A data-flow graph is a directed graph whose vertex-set (also nodes) is in one-to-one correspondence with the set of tasks. The directed edges correspond to the data transfer from one operation to another [de Micheli 94]. In a data-flow graph any node can fire (perform its operation) whenever input data area available on its incoming edges. Two particular extensions

![Diagram of iteration representation with a sequencing graph.](image)

**Figure 1.1:** Example of iteration representation with a sequencing graph.
of the data-flow paradigm are used in this work:

**sequencing graphs (SG's):** modeling at the highest level, requires the introduction of control-flow related information such as *branching* (conditional) and *iteration* (looping). In this case *control/data-flow graphs* (CDFG's) are more efficient than simple data-flows. Sequencing graphs are hierarchical CDFG's, where control-flow primitives such as branching and iterations are modeled through the hierarchy, whereas data-flow and serialization dependencies are modeled by graphs [de Micheli 94].

**synchronous data-flows (SDF's):** it is a special case of data-flow [Lee 87], either atomic or large grain, in which the number of data samples produced or consumed by each node on each invocation is different from one (as in the case of data-flow) and it's known a priori. By this, multiple sample rates within the same system are easily handled. In an SDF a delay of \( d \) samples can be introduced on an edge: thus a delay \( d \) on an edge from node \( A \) to node \( B \) means that the \( n \)th sample consumed by \( B \) is the \( (n-d) \)th samples produced by \( A \). Delays in SDF are usually shown with a black dot (Fig. 1.2).

SDF's efficiently permit us to examine DSP algorithms with re-

---

**Figure 1.2:** Example of an SDF: representation of a voice-band data modem [Lee 87]. Note the multiplicity of the sample rates. Complex data paths are highlighted.
spect to their implementation on parallel hardware. Furthermore multi dimensional SDF’s (MD-SDF’s) permit the modeling of multidimensional data-streams [Lee 93].

Other modeling methods are currently being investigated which also aim to represent control and data-flow processes within the same model [Grötker 97] [Fettweis 97]. These approaches are still in an early development phase and have not been considered in this work. On the other hand the used modeling methods (SG’s and SDF’s) optimally fulfill the requirements for the presented system.

1.4.3 Dedicated Processor Design

While this work mainly deals with the development of the multiprocessor architecture, it has to be pointed out that this is only a part of a larger project dealing with high resolution image correlation. In particular, within the same project, a dedicated processor aiming to accelerate the kernel operations of the adopted correlation algorithm has been designed and fabricated. The resulting application specific integrated circuit (ASIC) is called SPITFFIRE (Smart Pipelined Interleaved Fast Fourier Processing Engine). Exhaustive discussions on the chip architecture as well as on the synthesis and layout processes are carried out in a parallel PhD. thesis [Wosnitza 99].

The optimizations carried out at algorithm level, the dedicated processor architecture and the multiprocessor system architecture developments are three strictly related processes strongly influencing each other. Strong interaction between system and processor development has occurred in particular in two relevant aspects: on one side the determination of the critical operations at system level which could benefit from a placement on the SPITFFIRE chip and on the side the definition of the chip I/O interface with respect to the multiprocessor environment.

According to this the support given by a modern HW/SW codesign methodology and CAE environment during the concurrent chip and system development has played a fundamental role.

1.5 Work Outline

The work begins with an overview of state of the art image localization methods. Chapter 2 introduces the object localization algorithm which has to implemented by the multiprocessor architecture: the involved
computational requirements are discussed and an efficient optimization is presented.

In Chapter 3 the concept of the multiprocessing system is presented with the peculiarities of the localization algorithm as driving impulse. The discussions on the architecture are bound to state of the art storage devices in Chapter 4 and processing elements in Chapter 5.

Chapter 6 focuses on the user design methodology applied during the system development: a virtual prototype of the multiprocessor system is presented and evaluated. Based on the informations delivered by the virtual model and by the implementation, a performance model is built whose output is evaluated in Chapter 7 and compared with available benchmarks.

Multi-chip module and ASIC integration of the system are evaluated in Chapter 8. In this context a cost-model is generated and used for cost/performance analysis.

Estimations concerning technology driven performance scalability are presented in Chapter 9 where also the mapping of other algorithms on the presented architecture is discussed.
Chapter 2

High Resolution Image Correlation

In this chapter the cross-correlation based object localization is analyzed with respect to it’s computational requirements, which are compared with the performance provided by state of the art digital signal processors. An extension of the classical cross-correlation algorithm improving the localization accuracy is presented. Besides this characteristic, the new algorithm permits a huge reduction of the computational requirements through it’s computation in the frequency domain, based on the Fourier transform.

2.1 Object Localization

Classical object localization is realized by shifting the target pattern (or template) over the image and computing a similarity measurement (SM) for every possible position (Fig. 2.1). The point where the maximal value of the SM is found, is considered as the candidate for the localization [Russ 95]. A wide range of visual SM’s are realized with area correlation algorithms [Aschwanden 93] which have the advantage of being applied directly on the grey level image. Depending on the kind of operation used for the area correlation (addition, multiplication) the computational requirements can become in some cases particularly high, but the reduced complexity of the algorithm kernels make them particularly well suited for VLSI implementation [Rothacher 94] [Siegel 82].
The drawback of area correlation algorithms resides in their reduced insensitivity to the template rotation ($\pm 5^\circ$).

Alternative SM’s, such as optical template matching methods [Leclerc 89] [Elizur 91] or methods based on very extensive image pre-processing [Lejeune 89] [Cideciyan 95] [Hu 62] [Belkasim 91], are particularly appreciated when template rotation insensitivity is required. The complexity involved in these kinds of algorithms has limited up to now implementations in the software domain. Furthermore the high level of data dependent computation and complex mathematical operations present in these methods, limit the opportunities of efficient implementation in VLSI.

### 2.2 Normalized Cross Correlation Function

Area correlation algorithms have been widely studied [Aschwanden 93] with respect to their sensitivity to several image noise parameters (multiplicative and additive intensity variation, image noise). The best trade-off between similarity robustness and computation complexity appears with the Normalized Cross Correlation Function (NCCF) defined by (2.1).
2.2. Normalized Cross Correlation Function

\[
\text{nccf}(x, y) = \frac{\sum_{v=0}^{T} \sum_{u=0}^{T} tp(u, v) \cdot im(x + u, y + v)}{\sqrt{\sum_{v=0}^{T} \sum_{u=0}^{T} tp^2(u, v) \cdot \sum_{v=0}^{T} \sum_{u=0}^{T} im^2(x + u, y + v)}}
\]  

\[
tp(u, v) \quad \text{square template} \quad 0 \leq u, v < T
\]

\[
im(x, y) \quad \text{input image} \quad 0 \leq x, y < N
\]

Equation (2.1) describes the computation of the NCCF of the template \( tp(u, v) \) for a search window at position \((x, y)\) in the image \( im(x, y) \). The method is a variation of the cross correlation function (CCF) usually applied in mathematics.

\[
ccf(x, y) = \sum_{v=0}^{T} \sum_{u=0}^{T} tp(u, v) \cdot im(x + u, y + v)
\]  

The difference in (2.1) resides in the fact that the cross correlated value of each image pixel is divided by the geometrical mean value of the energy contents of the search window and the template [Aschwanden 93]. With this step, defined as normalization, the correlation result becomes independent from the energy content of image and template. Without the normalization, the correlation results would have the highest values at the most luminous region (thus with high energy content) of the search area, thus the localization of templates with low brilliance would be excluded.

With the defined images and template sizes \((N, T)\) the NCCF algorithm has a complexity of order \(O(N^2 \cdot T^2)\).

The computational requirements of the NCCF algorithms are resumed in Table 2.1. The computational requirements are expressed in Multiply Accumulate (MAC) operations, since this is by far the most required operation. On the other side the MAC operation is what state of the art digital signal processors (DSP’s) are specialized into and can then be easily used for performance evaluation purposes.

Assuming input data with 8 bit resolution (both image and template), the dynamic of the numerical results do impose floating
Table 2.1: Computational requirements of the NCCF algorithm by high-resolution images: data relative to the processing of a single frame and to a frame rate of 30 frames/s

<table>
<thead>
<tr>
<th>Size [pixels]</th>
<th>1 Frame</th>
<th>30 Frame/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MACs</td>
<td>DIVs</td>
</tr>
<tr>
<td>N</td>
<td>T</td>
<td>26.3 · 10⁹</td>
</tr>
<tr>
<td>1024</td>
<td>128</td>
<td>1.64 · 10⁹</td>
</tr>
<tr>
<td>512</td>
<td>64</td>
<td>103 · 10⁶</td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>103 · 10⁶</td>
</tr>
</tbody>
</table>

The floating point performance provided by state of the art DSP’s is listed in Table 2.2, which gives an idea of the available peak performance (usually obtained by multiplying the number of processing units on the chip by the maximal clock frequency) but leaves a high grade of uncertainty concerning the effective observed performance on a given algorithm. In fact only in the case of the Sharc processor is it possible to assume that the peak performance can be exploited without particular problems, due to its classical single multiply-accumulate unit clocked at 60 MHz.

<table>
<thead>
<tr>
<th>Vendor Device</th>
<th>2106x</th>
<th>Analog Devices</th>
<th>N/A</th>
<th>Texas Instr. C67xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nickname</td>
<td>Sharc</td>
<td>HammerHead</td>
<td>N/A</td>
<td>Velocity</td>
</tr>
<tr>
<td>Availability</td>
<td>1994</td>
<td>1Q99</td>
<td></td>
<td>3Q98</td>
</tr>
<tr>
<td>Architecture</td>
<td>DSP</td>
<td>DSP+SIMD</td>
<td></td>
<td>VLIW</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>60 MHz</td>
<td>100 MHz</td>
<td></td>
<td>167 MHz</td>
</tr>
<tr>
<td>16b MAC/s</td>
<td>60 · 10⁶</td>
<td>200 · 10⁶</td>
<td>2000 · 10⁶</td>
<td>333 · 10⁶</td>
</tr>
<tr>
<td>FP MAC/s</td>
<td>60 · 10⁶</td>
<td>200 · 10⁶</td>
<td>500 · 10⁶</td>
<td>333 · 10⁶</td>
</tr>
</tbody>
</table>

Table 2.2: State of the art floating point DSP peak performance [Wolf 98]

In the other cases, following today's major trend in DSP architec-
2.3 Blue Screen Technique (BST)

ture, vendors have committed to mixed DSP-VLIW oriented designs for the high-end part of their product lines, which promise impressive performance gains over their predecessors but bring many new challenges. Primary among these are the compiler issues, since the unprecedented complexity of the new architectures makes it difficult to exploit their performance. For example in the case of the C67xx, keeping all six FP units busy during every clock cycle requires an even mix of multiplications, additions and miscellaneous operations such as compare or absolute value. This instruction mix is uncommon in DSP applications, leading to sub optimal chip exploitation [Shoham 98].

Other considerations involve the relationship between on-chip memory space and the address space related to the application: in the case of high resolution NCCF this relationship is in the range of several orders of magnitude, thus the I/O characteristics of the DSP become determinant. Since the off-chip memory bandwidth of the emerging DSP is only a fraction of the on-chip bandwidth required to optimally support the highly parallelized CPU (1/10 in the case of TigerSharc), a further limitation is imposed on the peak performance exploitation.

Furthermore the clock frequencies announced by the different vendors have to be considered carefully [Shoham 98] as they are conditioned by aggressive marketing strategies.

According to these considerations, an available DSP performance of 100 MFLOP's is taken as reference, which leads to the fact that there is still a worst case factor of up to three orders of magnitude between the available and the required performance in the case of $(N, T) = (1024, 128)$ (Tab.2.1).

As a consequence, before entering any examination dealing with the implementation of the algorithm on a multiprocessor architecture, a different solution for the computation of the NCCF algorithm has to be found. This solution has to reduce the computational requirement to a large degree, since it is not feasible to face a parallelization grade of more than several hundreds with a computer vision system.

2.3 Blue Screen Technique (BST)

Before facing the reduction of the computational requirements of the NCCF, a modified version of this algorithm named Blue-Screen Technique (BST) [Cavadini 96] is presented which improves the accuracy and robustness of the localization attempt and on the other hand
permits the optimization step presented in Section 2.4.

In many cases, the object to be found only sparsely fills the rectangular template area and many pixels of this area (background) do not represent relevant object information. Thus these irrelevant template pixels have to be masked out every time an NCCF value is computed on a new search window. While this task can be easily performed on the numerator of (2.1) by forcing all the background pixels to zero, in the denominator of (2.1) the whole square template is considered in the computation of the search window energy, including the unwanted background area.

A binary version of the template $btp(u, v)$ is introduced where:

$$btp(u, v) \in [0, 1] \quad 0 \leq u, v < T$$

$$btp(u, v) = \begin{cases} 
1 & \text{if } tp(u, v) \text{ is a relevant pixel} \\
0 & \text{if } tp(u, v) \text{ is a background pixel}
\end{cases}$$

Using $btp(u, v)$ (2.3) the background masking-out step is carried out by a selective sum of the squared image pixels in the computation of the energy terms. These modifications lead to an extended version of the NCCF called Blue Screen Technique (BST) (2.4).

$$bst(x, y) = \frac{\left( \sum_{v=0}^{T} \sum_{u=0}^{T} tp(u, v) \cdot im(x + u, y + v) \right)^2}{\sum_{v=0}^{T} \sum_{u=0}^{T} btp(u, v) \cdot im^2(x + u, y + v)}$$

It has to be pointed out that in (2.4) the square-root operation present in (2.1) has been substituted by a squaring operation, basing again

![Figure 2.2: Template and binary version of the template.](image-url)
on the fact that no absolute NCCF values are required for the object localization.

The name blue-screen has been taken from a similar technique used in the television broadcast news for the super-imposition of the background-free image of the speaker over the weather map.

The improvement in terms of localization accuracy brought by the BST depends on the complexity of the template form with respect to an ideal $T \times T$ square template. In general the benefits of the BST are relevant: in particular, by very complex templates on highly textured backgrounds which are practically impossible to localize with a NCCF, the BST shows precise and selective localization peaks.

## 2.4 Frequency Based BST Computation

### 2.4.1 Concept

From the computational point of view a new characteristic introduced by the BST with respect to the NCCF is that both numerator and denominator of the SM computation, have a CCF character. An alternative method for the computation of the CCF consists in performing this operation in the frequency domain [Russ 95].

\[
CCF(r, s) = IM(r, s) \cdot TP^*(r, s) \quad 0 \leq r, s < N \quad (2.5)
\]

\[
CCF(r, s): \quad \text{Frequency spectrum of } ccf(x, y)
\]

\[
IM(r, s): \quad \text{Frequency spectrum of } im(x, y)
\]

\[
TP^*(r, s): \quad \text{Complex conjugated frequency spectrum of } tp(u, v)
\]

According to (2.5), the frequency based CCF (FB-CCF) computation is reduced to a point-wise complex multiplication of the image spectrum by the complex conjugate template spectrum.

In order to dispose of the frequency representation (spectrum) of image ($IM(r, s)$) and template ($TP(r, s)$), both have to be transformed by means of the 2 dimensional Fast Fourier Transform (2D FFT) [Cooley 65] defined as follows:

\[
IM(r, s) = \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} (im(i, k) \cdot W_N^{-iy} \cdot W_N^{-yk}) \quad (2.6)
\]
\[ TP(r, s) = \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} (tp(i, k) \cdot W_N^{ri} \cdot W_N^{sk}) \]  

\[ W_N = \exp(-2\pi j/N) \quad \text{with:} \quad j = \sqrt{-1} \]  

The FB-CCF imposes the computation of two 2D FFT for the generation of the image and template spectrums as well as the computation of a 2D inverse FFT in order to obtain the correlation results in the space domain. The inverse 2D FFT is computed with the same procedure of (2.6) with the use of the \( W_N \) defined in (2.9):

\[ W_N = \exp(2\pi j/N) \]  

The point-wise multiplication of \( IM(m, n) \) and \( TP^*(m, n) \) in (2.5) implies that, before being transformed, the template \( tp(u, v) \) has to be enlarged from the \( T \times T \) dimensions up to \( N \times N \) pixels: this is carried out by forcing to zero the remaining \((N-T) \times (N-T)\) area (called 0-PAD). Therefore the FB-CCF is independent of the template size which can be up to \( N \times N \). This is a relevant advantage of the FB-CCF if confronted with the space domain CCF.
2.4. Frequency Based BST Computation

The sequencing graph of the FB-BST computation (visualized depicted in Figure 2.3) is presented in Figure 2.4.

A comparison between Figure 2.1 and 2.4 shows how the frequency based approach eliminates the need to move the search window over the search area and compute the SM. Another aspect emerging from Figure 2.4 is that the transformed template values can be computed in a separate step. Considering object localization applications where the template is usually a static data-set to be found in a dynamic image stream, this means that the template data can be pre-processed off-line, reducing the requirements to the computing engine.

2.4.2 FB-BST Computational Requirements

The 2D FFT has a complexity of order $O(N^2\log_2 N)$, which refers to complex operations. More precisely, the complexity of the radix-4 2D FFT expressed with real multiplications and additions can be expressed
Chapter 2. High Resolution Image Correlation

by (2.10) and (2.11) respectively [Jamieson 86]. The radix-4 FFT is used here as reference, since it is the one implemented by the dedicated FFT processor used in the system.

\[ \text{mults}(2D \text{ FFT}) = 3N^2 \log_2 N \quad (2.10) \]

\[ \text{sums}(2D \text{ FFT}) = \frac{11}{2} N^2 \log_2 N \quad (2.11) \]

Assuming that the template data is computed off-line (Fig. 2.4), a FB-BST still requires four 2D FFT's and two \( N \times N \) point-wise complex multiplications. The total amount of real multiplications and additions required by the FB-BST (without the final square and divide) is expressed by (2.12) and (2.13). The computational requirements (MAC operations) of the space domain implementation of the BST are obtained with (2.14) (which also applies to the NCCF).

\[ \text{mults}(FB - BST) = N^2(12 \log_2 N + 8) \quad (2.12) \]

\[ \text{sums}(FB - BST) = N^2(22 \log_2 N + 4) \quad (2.13) \]

\[ \text{macs}(BST) = 2 \cdot (N - T)^2 \cdot T^2 \quad (2.14) \]

The computational reduction obtained by the frequency based approach as a function of the image and template size graphically represented in Figure 2.5, where the case of real-multiplications is considered (the addition reduction factor has qualitatively the same graph).

<table>
<thead>
<tr>
<th>Size ( im(x, y) ) [pixels]</th>
<th>( tp(u, v) )</th>
<th>Rad-4 FB-BST MULT’s</th>
<th>ADD’s</th>
<th>Gain Factor</th>
<th>30 Frames/s Req. DSP’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>128</td>
<td>134 \cdot 10^6</td>
<td>235 \cdot 10^6</td>
<td>196</td>
<td>40</td>
</tr>
<tr>
<td>512</td>
<td>64</td>
<td>30 \cdot 10^6</td>
<td>53 \cdot 10^6</td>
<td>54</td>
<td>9</td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>6.8 \cdot 10^6</td>
<td>11.8 \cdot 10^6</td>
<td>15</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2.3: Computational requirements of FB-BST (radix-4 FFT) implementation compared with the implementation in the space domain (Tab. 2.1) referring to a single frame. The last column shows the number of 100 MOPS DSP required for a 30 frame/s frame-rate.

The precise computational requirements of the FB-BST computation for some \( (N, T) \) pairs are listed in Table 2.3. As well as the reduction factor obtained by the frequency based approach, in the last
column the number of 100 MOPS DSP's required for a frame rate of 30 Frame/s is shown. This reference has already been used in the case of the NCCF implementation in the space domain.

After obtaining a substantial reduction of the computational requirements as well as a relevant improvement of the localization accuracy by means of the BST at algorithm level, the implementation of the proposed algorithm can be envisaged. The computational requirements of Table 2.3 are still in an order of magnitude greater than the available performance of state of the art processors. Thus the development of a dedicated parallel system architecture is unavoidable.
Chapter 3

System Architecture

The parallel system architecture is presented, beginning with its division into two main parts: the one dedicated to the frequency domain computation (FDE) and the other (GPE) supporting the maximum extraction procedure. Particular weight is given to the parallelization approach of the 2D FFT on the FDE, which is carried out by means of a new dedicated memory reconfiguration architecture. The mapping and scheduling of the different tasks is discussed, leading to the determination of the system frame rate.

3.1 Concept

The FB-BST (Fig. 2.4) is characterized by two main activities: on one side there is the SM computation in the frequency domain based on the 2D FFT and the complex multiplication and on the other side the maximum extraction procedure (MEP). More precisely, considering the requirements in terms of processing tasks and data-access, the following differences do appear:

Complex computation: from the point of view of the mathematical computation, the effort imposed by the 2D FFT’s (2.6) and (2.7) and in particular by its complex arithmetic and data format is considerably high. Already with elementary operation such as addition and multiplication, the complex data format requires respectively 2 and 6 real operations. A dedicated processor with a complex arithmetic unit can considerably speed-up the FFT computation. Concerning the complex arithmetic unit, a relevant role
is played by the high precision imposed to the SM computation by the subsequent normalization and MEP. This imposes the use of a dedicated processor for the SM computation, where fast complex computation joined with high accuracy has to be provided [Wosnitza 99].

Concerning the MEP, the computations can be executed on a state of the art floating-point processor. Furthermore a programmable CPU is required, since variations of the MEP can be imposed by the final system application (for instance if the extraction of more relative local maxima are required).

**2D data management:** in relation to the large amount of data involved in the high resolution input images, data handling resources are particularly critical in a multiprocessor system. The FB-CCF imposes the storage of temporary results implying memory writing operations (which involves bi-directional data access between CPU and memory) as well as the access of two input arguments during the complex multiplication. On the contrary the MEP is characterized by a more simple data-access scheme, which doesn’t impose to write temporary results (uni-directional).

**Data dependent computation:** is another aspect of the algorithm which has severe repercussions on the implementation. The MEP is characterized by a higher grade of data-dependent processing if compared with the FB-CCF, in particular if multiple maxima have to be detected. In the case of an application requiring more complex image post-processing, the data-dependent character of the processing could be amplified further.

**Available parallelism:** usually divided into *instruction and data parallelism*, is the basic condition in order to gain system speedup by use of parallel computation.

The available parallelism of in the FB-BST, is quite different in the two main algorithm parts. Basically, the whole algorithm flow shows very high inherent data parallelism with the exception of the final steps of the MEP where, the maximum sorting algorithms combined with high level processing tasks, assume more and more sequential characteristics. Therefore the FB-CCF is well suited for parallel VLSI implementation, whereas the last part offers lower parallelization chances.
Figure 3.1: Overview of the MPIC architecture. Example with 4 PE's in the FDE and 2 processors in the GPE.
As a consequence of this analysis, the basic division of the FB-BST into two main parts does appear in the architecture of the multiprocessor system for image correlation (MPIC) consisting of two processing engines (Fig 3.1):

**FDE: Frequency Domain Engine:** supports parallel implementation of the processing in the frequency domain, including the transformations and back-transformation operation.

**GPE: General Purpose Engine:** parallel implementation of the normalization, the MEP procedure as well as pre- and post-processing tasks which could be required by the system application.

The normalization procedure is placed together with the MEP on the GPE because of the fact that the floating-point division operation is provided by state of the art programmable processors already involved by the MEP.

### 3.2 Frequency Domain Engine

Figure 3.2 shows in details the operations which have to be accelerated by a parallel implementation of the FDE. The following remarks can be done:

- the template data are supposed to be pre-computed and stored on the FDE. The BIN and 0-PAD operation are supposed to be placed on the video-input path, since their complexity can be easily supported by state of the art advanced frame-grabbers, which dispose of a low-level image processing unit. The 2D FFT can be realized by the FDE.

- the squaring of the input image required by the energy-path is computed directly by the FFT processor used on the FDE (Section 5.1), while data is loaded into the processor memory.

According to these considerations the crucial task to be accelerated during system the run-time is the sequence consisting of 2D FFT, CMUL and 2D inv. FFT. This is practically the kernel of the FB-BST which requires a dedicated approach towards its parallel implementation.
The SDF of Figure 3.2 offers very few parallelization opportunities, residing practically only in the parallel computation of the two main FB-CCF’s. Thus, in order to obtain an higher parallelization grade, the parallel computing approach is carried out on the 2D FFT and CMUL tasks. Furthermore, being the predominant part of the FB-CCF, the parallel implementation of the 2D FFT is of central interest and practically determines the FDE multiprocessor architecture. The 2D data management of the 2D FFT algorithm is particularly complex in comparison with the one involved in the CMUL operation. Therefore the 2D FFT algorithm is exhaustively analyzed with respect to its parallelization.

3.2.1 Extended 2D FFT indirect method

The 2D FFT can be computed by means of the direct method [Kunieda 90] where the two image dimensions (row and columns) are computed concurrently based on 2D butterfly operations [Blauth 95].

An alternative is obtained by a reformulation of (2.6) where the fact
that each sum is independent of the other, permits the independent computation of row and column data (3.1):

\[
IM(r, s) = \sum_{i=0}^{N-1} \left( \sum_{k=0}^{N-1} (im(i, k) \cdot W_N^{rk}) \right) \cdot W_N^{si} \tag{3.1}
\]

Thus, following the indirect method, the 2D FFT is split into a 1D FFT’s performed over the image rows followed by a 1D FFT executed column-wise over the results of the first one.

For the FDE the indirect method has been chosen, mainly because of the fact that the system complexity can be kept within the 1D domain: this is a relevant factor considering the size of the processed images. In fact the parallelization of the 2D-FFT using the direct approach is usually based on large two dimensional processor arrays which with an image size of 1024x1024 implies a too high system complexity [Tanno 92]. Another variant is proposed in [Jamieson 86] where the direct method is implemented on a single instruction multiple data (SIMD) architecture also involving the use of \( O(N^2) \) processing elements.

The indirect method offers better parallelization opportunities with respect to the factor two of Figure 3.2 making the image row (column) become the fundamental data element on which the FDE architecture is based on. Having input images of size \( N \times N \) pixels, granularity of the problem is given by the 1D FFT of a row of \( N \) elements. According to this the 2D FFT task can be parallelized with up to \( N \) PE’s from the point of view of the algorithm. A parallelization grade in the range 1...\( N \) is sufficient in order to obtain the speedup required in Table 2.3.

The transposition (TRSP) operation carried out on a 2D data-set is defined by (3.2).

\[
TRSP : \quad im_{trsp}(r, s) = im(s, r) \quad 0 \leq r, s < N \tag{3.2}
\]

Applied to an image the TRSP moves its row-data to the columns and vice versa. The standard indirect method is slightly extended by transposing the complete dataset between the two 1D FFT’s, so that the second one is also carried out on the image rows.
Figure 3.3: FB-CCF with use of the extended indirect method. In the last two versions of the algorithm, the number of executions of a single operator (firing) is shown with a black background. The different firings are due to the fact that in the first one various data structures (rows, points) are processed while in the second one only the data-row is used.
The enhancement brought by the insertion of the transposition has several positive repercussions for a multiprocessor implementation, independently of the final FDE architecture:

**2D FFT Parallelization:** is made easier by the fact that the parallelization approach is carried out in the same dimension during the two 1D FFT's, making the tasks for the parallel architecture more homogeneous and thus better suited for VLSI implementation. In fact the transversal data access over the image columns is concentrated in the transposition operation and not in the 1D FFT computation phase where the complex CPU is involved.

**Data Access:** becomes homogeneous during the 1D FFT computation, because carried out row-wise only, simplifying the memory addressing schemes which have to be supplied by the processor-memory interface. As a consequence the requirements of the underlying HW are reduced.

**Memory Access Time** can be boosted to its theoretical limits. For state of the art Dynamic RAM's (DRAM), there's a great difference if consecutive accesses are made over the same row instead of over the memory columns. Exhaustive discussion of these aspects is carried out in Section 4.2.

Apart from the advantages resulting from the introduction of the transposition in the indirect method, there is the need to provide the infrastructure performing this data-rearrangement between the two 1D FFT's. The conditions under which the transposition is carried out are made worse by the FDE multiprocessing issues. The development of the parallel bus-memory architecture of the FDE therefore keeps the transposition step as an important issue.

### 3.2.2 SPMD Parallelization Approach

After the introduction of the extended indirect method, the FB-CCF is computed basing on a limited set of operations (1D FFT, CMUL and TRSP) which can be independently executed on different data-sets having various structures (points, rows,...). These characteristics involve the use of the *Single Program, Multiple Data Stream* (SPMD) [Flynn 72] parallel architecture for the FDE.

Furthermore, as shown in the lower part of Figure 3.3, the modification of the type of data structure processed by a single operator of the
algorithms leads to the same numbers of repetitions for every task. In the first version of the algorithm, the CMUL operator works on single points while the transposition is carried out on a whole image. This leads to different firing-rates. By the use of a data-row as the fundamental structure for the the CMUL and TRSP every node in the SDF has a firing rate of $N$.

This higher grade of homogeneity can be optimally exploited in a SPMD approach, because single operations (1D FFT, CMUL and TRSP) can grouped into larger programs, leading to a simple parallel task flow. The SPMD approach implies the partitioning (PART) of the data-set over the available number of PE's. The fact that every PE executes the same sequence of data-independent computation produces constant execution times which eliminate the arising of load-balancing problems, offering chances for optimal resource exploitation.

According to Figure 3.4 it is not possible to build a single program consisting of the whole sequence of operations of the FB-CCF, because a data element which can be independently processed during the whole algorithm doesn't exist. The parallel computation of the FB-CCF is divided into three phases. In fact the data rearrangement occuring after the transposition consist of a merge (MRG) operation requiring data processed by different PE's to be collected before the next phase can be executed.

An alternative to the SPMD approach is offered by a parallel data-flow architecture [Gunzinger 90]. Following this principle each node (or group of nodes) of the SDF graphs of Figure 3.3 is directly mapped onto the processing elements. The following drawbacks appear, if this approach is compared to the SPMD approach:

- load-balancing is made difficult leading to poor resource exploitation. Processors involved in the 2D FFT computation have far greater requirements with respect to the one performing CMUL or TRSP.

- 2D FFT parallelization based on the indirect method required $P = N$ PE's to be present in the system. Otherwise (with $P < N$) iteration of the 1D FFT over different rows is required, leading to a SPMD approach.

- according to the data-flow principle, where nodes fire as soon as input data is available, the presence of a non-blocking buffer between the nodes is necessary. The non-blocking buffer permits a
node to produce results while the following one consumes them without reciprocal influence. This imposes a huge investment in the storage infrastructure (mostly dual-post) as a consequence of the image dimensions.

- scalability of the system performance is difficult to obtain, since different performances are given by different mapping of the nodes on the PE's leading practically to different system architectures. On the contrary SPMD offers performance scalability merely by
changing the number of PE's.

As a result of these factors, the data-flow parallelization approach has been discarded in favor of the SPMD. The SPMD parallelization implies that a processing element (PE) can support the 1D FFT as well as the point-wise complex multiplication of two rows. Figure 3.5 gives a visual representation SPMD approach of Figure 3.4.

![Diagram](image)

**Figure 3.5:** SPMD parallelization approach of the FB-CCF. The procedure is divided into three phases, determined by the process synchronization points. Every PE independently processes a portion of data.

### 3.3 Reconfigurable Memory Architecture

#### 3.3.1 Shared vs. Distributed Memory

SPMD parallel machines are mainly characterized by their memory organization. The SPMD model is usually divided into two main categories [Bell 92] consisting of the *shared memory* and the *distributed memory* machines (Fig. 3.6). The first one is characterized by a global address space visible by all processors which have a common connection to the memory: every processor can directly access the whole memory space but in doing this it automatically makes the other processors wait. In the distributed memory organization every
processor has its own private memory non accessible by other processors: a local addressing scheme is used. The access of data stored in remote processors implies that inter-processor communication is be carried out. In some large multiprocessor systems, memories are physically distributed with the processors. If the physically separate memory can be addressed as one logically shared address space, meaning that a memory reference can be made by any processor to any memory location, these machines are called distributed-shared memory (DSM) multiprocessors [Hennessy 90] [Tseng 96]. A DSM system allows a natural and portable programming model on distributed-memory machines, making it possible to construct a relatively inexpensive and scalable parallel

**Figure 3.6:** Shared, distributed and reconfigurable memory architecture where by means of a flexible interconnection scheme, the same memory element can be accessed in shared or distributed mode.
system on which programmers can develop parallel application codes.

### 3.3.2 RMA Concept

Considering the SPMD approach for the FB-CCF (Fig. 3.4), it can be seen that the parallel processing on the different PE can be optimally supported by a distributed memory organization up to the point where the MRG and PART operations are carried out. In fact in these operations an access on the global data-set is involved which can take large benefits by the use of a shared memory architecture.

Thus for the FDE a new dedicated approach to the memory organization is used aiming to maintain the benefits of both distributed and shared memory organization. As a consequence the reconfigurable memory architecture (RMA) has been developed (Fig. 3.6). The driving idea behind the RMA is to provide both types of memory organizations in the system, but also to achieve fast and simple reconfigurability of the memory organization. The memory configuration is dynamically selected at runtime with respect to the type of computation or communication currently involved by the algorithm flow on a given dataset. When a dataset requires parallel computation based on distributed memory architecture, the physical memory location where these data are stored is organized in this way with respect to the processors. The same happens when shared memory is required.

It has to be pointed out that even though RMA consist of both shared and distributed memory it doesn't fall into the DSM concept.

The RMA concept [Cavadini 97a] [Cavadini 97b] is realized by means of three basic elements depicted in Figure 3.7 which do affect the interconnection network between PE's and memory:

- **Vertical Bus System (VBS):** realizes shared memory
- **Horizontal Bus System (HBS):** realizes distributed memory
- **Crossbar Switch (XBAR):** provides memory reconfiguration

Moreover the main memory is physically partitioned into banks which are alternatively used in a shared or distributed manner.

#### Distributed Memory: HBS

The memory banks connected to the HBS through the XBAR are configured as distributed memory (Figure 3.7). The PE I/O port can in-
dependently access its corresponding memory bank, without any inter-
action by others PE. The addressing space is locally defined by a local
address generator in the I/O port. Each HBS connection from a PE to
a memory bank has its own data, address and control lines.

![Diagram](image)

**Figure 3.7:** RMA: memory banks connected to the PE’s as distributed
memory in the left image are configured as shared memory in the right
one and vice versa. The change is achieved by toggling the XBAR state.

**Shared Memory : VBS**

The memory banks connected to the VBS through the XBAR are con-
figured as shared memory (Figure 3.7). The I/O ports of each PE share
the VBS and all the memory banks connected to it through the XBAR’s,
with all the other PE’s. In contrast with the HBS, a global address
space is defined requiring the addressing operation to be assumed by
a central VBS controller (VBS-C). Only the data lines are routed be-
tween the memory to the PE’ I/O ports whereas the address signals are
driven by the VBS-C. In addition to these connections, further lines for
synchronization and control purpose are connected to the I/O ports.
The necessity of this synchronization infrastructure is motivated later
at implementation level (Section 4.1).
3.3. Reconfigurable Memory Architecture

Reconfiguration

Since the activities performed on the dataset, repetitively require shared and distributed memory configuration, frequent change of memory configuration is involved. The execution of this context switching with a data move operation from a shared to a distributed memory domain or vice-versa would be particularly unfavorable. This is because of the size of the data structures involved by the algorithms and by the high resolution computation.

The RMA realizes the context switching in a much more elegant way, by physically switching the memory component with a bus reconfiguration (Fig. 3.7). This is realized with the use of fast, zero delay crossbar switches (XBAR). The fact that all of the addressing schemes are a priori defined by the algorithm allows the switching of a given memory bank (with the data stored in its cells) from the shared to the distributed subsystem without causing problems to the data consistency. There is a fix known mapping between the shared global addressing scheme and the distributed one. Thus data consistency can be realized without particular problems, when switching data for one to the other kind of architecture.

3.3.3 FB-CCF mapping on RMA

The SDF representation of the FB-CCF of Figure 3.4 is slightly modified in order to better distinguish the communication and computation operations to be mapped on the RMA. The resulting SDF is depicted in Figure 3.8.

In particular, with respect to the SDF of Figure 3.4 the following modifications do occur:

**GET operation:** according to Figure 3.7 and 3.1 the PE is provided with a fast local memory imposed by the high data-throughput occurring at CPU level [Wosnitza 99]. Thus a GET operation is introduced in the SDF describing the operation of reading a row of data from the distributed part of the main memory into the local memory.

In phase B a GET operation has also to be performed on the template data: the template row corresponding to the image row currently loaded in the PE has to be read from the main memory, before the CMUL operation is performed.
Figure 3.8: Reorganization of the SDF of Figure 3.4 for mapping on the RMA architecture. Computation and communications operations are highlighted with different grey-levels.

The GET operation occurs with the distributed memory banks because each PE has a separate portion of data to process (as well as a corresponding template region). According to this the GET operation is mapped on the HBS.
3.3. Reconfigurable Memory Architecture

**TRSP and MRG operation:** these two operations of Fig. 3.4 are merged into a single one in 3.8 since they can be concurrently executed (Section 4.1) and both do involve the use of the shared memory part. This combined operation is called PUT and is performed by the VBS.

![Diagram of task scheduling of phase A, B and C on a PE.](image)

**Figure 3.9:** Task scheduling of phase A, B and C on a PE.

**Pipelined Execution:** the GET, PUT and processing operations are performed in a pipelined way. This is represented in Figure 3.8 by the introduction of black points on the edges connecting the computation and communication operations. This practically means that while the CPU is processing row r, the HBS is getting row r + 1 from the main memory and the VBS is executing a OUT on row r - 1.

**PART Operation:** this operation doesn’t have to be concretely executed because between two phases a main memory reconfiguration is carried out. By this the logical data sets TMP1 and TMP2 are
moved from the shared to the distributed physical memory before entering the next phase, actually performing the PART operation.

**Modified Phase C**: since the PUT operation consist of TRSP and MRG, in phase C a transposition of the data set is introduced which is not required by the FB-CCF algorithm. This permits us to maintain a regular process scheduling in the three phases aiming at an easy system control.

According to the discussion on the SDF of Figure 3.8 the task scheduling for the three phases can be generated as shown in Figure 3.9.

![Figure 3.10: Example of the FDE data management during phase A. After the 1D FFT transformation the image is written back, transposed and repartitioned into the main memory. At the end of the phase an XBAR state-toggle will make data ready to be processed by phase B.](image)

The computation of a phase is reduced to an iterative execution of a phase cycle, as defined in Figure 3.9. Since each of the P PE’s has N/P rows to process and the pipelined execution introduces two further
cycle iterations (initialization and finishing), the number of iterations for the three phases \( N_{PHx} \) can be defined by:

\[
N_{PHA} = N_{PHB} = N_{PHC} = \frac{N}{P} + 2 \quad \text{[cycles]} \quad (3.3)
\]

The division by \( P \) in (3.3) reflects the parallelization grade obtained by the SPMD approach. The conditions required in order to achieve an improvement of factor \( P \) in the FB-CCF computation (linear speedup) are discussed in conjunction with the RMA implementation.

A more visual representation of the RMA configuration and the involved data communication is shown in Figure 3.10, where the case of phase \( A \) is taken as an example.

### 3.3.4 Requirements to the RMA processing element

As already stated, the processing element has to support efficient computation of the complex 1D FFT combined with the complex multiplication. Further conditions concerning the data management units, have to be fulfilled by the PE, in order to optimally support full parallel computation and communication of the RMA. This concerns the logical and physical size of the local memory as well as its communication interfaces.

![Figure 3.11: Requirements in terms of logical memory space and I/O interfaces for the PE in order to realize full RMA parallel computation and communication (Fig. 3.9).](image)

**Local Memory Size:** in the ideal case, the local memory has to provide the space for two data rows in order to support concurrent GET and PUT operations. The logical size of a row is \( N \) complex
words. Furthermore, the CPU itself requires a certain amount of local memory space which greatly depends on its internal architecture and on how the FFT and CMUL operation are implemented. In order to obtain the required physical memory size, the used word representation has to be considered.

Advanced addressing methods at local cache level could be used in order to share the same local memory space by overwriting expired outgoing data with new valid incoming data. This method imposes a considerable overhead for the synchronization of the GET and PUT operation.

**Communication Controller (CC):** assumed that enough space is provided, there must be the possibility to concurrently send or receive data on the VBS and HBS. For this a communication controller placed between the local memory and the RMA provides two separated I/O interfaces, each one consisting of a control unit with the corresponding data, address and handshake signals. By means of the two I/O interfaces the CC implements the GET and PUT operations.

**Local Memory Access:** the computation and I/O tasks performed by the CPU and the CC, have to access the local memory without interfering with each other. This involves the use of a multi-port local memory architecture.

These conditions represent the qualitative requirement to the PE architecture and are impossible to quantify without knowing the exact CPU architecture and the used word representation. In Section 5.1 a precise discussion is made with respect to the SPITFFIRE case.

### 3.4 Real Valued FFT

#### 3.4.1 Concept

A relevant reduction of the computational requirements for the FDE can be realized by applying the *real valued FFT* (RV FFT) optimization to the 2D FFT computation [Mitra 93]. This approach exploits the propriety of the FFT arising when real data is used as input as in the case of the FB-CCF: the produced spectrum is symmetrical with respect to its central point.
This FFT peculiarity can be exploited in the FB-CCF where, after the first 1D FFT, only $N/2 + 1$ words of each row contain relevant information: the rest can be discarded (DIS operation in Figure 3.12). As a consequence the number of iterations which phase B has to go through is nearly halved. In fact discarding $N/2 - 1$ columns after phase A, reduces the iterations for phase B to $N/2 + 1$. After phase B, before entering phase C an image mirroring (MIRR operation in Figure 3.12) permits to rebuild the necessary information for the final back transform (Figure 3.12).

But real valued FFT can be further exploited so that a more consistent optimization is achieved, involving phase A and C as well. This is carried out by analyzing the influence of real input data in the first 1D FFT, particularly in conjunction with the complex CPU architecture. The aspect to be analyzed here is how a FFT processor can differentiate between the computation of real and complex input data [Sorensen 87].

Three techniques can be proposed for this approach:

**Zero Patching**: the complex CPU always computes FFT assuming complex data input. When real data is processed, the imaginary

---

**Figure 3.12**: Real Valued FFT effects: real input data produce symmetrical spectrum reducing the number of iterations in phase B.
part of the words is forced to zero. This solution requires the simplest CPU architecture view that only one FFT mode has to be supported. But the exploitation of the allocated resources isn't optimal, because dummy data is fed into the CPU and computed.

**Two FFT Modes CPU:** the complex CPU supports a dedicated mode for real input data achieving faster transformation. This can be realized for instance by a specific implementation of the last butterfly stage in the 1D FFT algorithm. The complexity of the CPU is increased by the flexibility it has to provide with respect to a single FFT mode architecture.

**Separate/Unseparate:** this solution exploits the symmetries of the 1D FFT to transform two real valued sequences simultaneously by computing one complex 1D FFT [Cooley 72]. Based on this, a more elegant solution with respect to the Zero Patching can be realized. Two real rows are merged together in a single complex row: one into the real part and the other into the imaginary part (R2C). The 1D FFT is carried out on the merged data. After the 1D FFT, it is possible to extract from the transformed $N$ point complex row two separated spectrums with size $N/2+1$ belonging to the original merged real rows (3.4).

The same approach is applied to the back transform step, knowing that the result will consist of real data only. Two $N/2+1$ words long complex half-rows are unseparated in order to build an $N$ point complex row which is back transformed (3.5). The results entail the back transformed data belonging to the two half-rows in the real and imaginary part respectively. Applying C2R, two real rows are obtained.

\begin{align*}
\text{R2C} & : \quad \text{operation merging two real rows into the real- and imaginary-part of a complex one} \\
\text{C2R} & : \quad \text{operation extracting two real rows from real- and imaginary-part of a complex one}
\end{align*}

The GPU has to support only one transform mode which is optimally exploited because the CPU is fed with relevant information. A speedup of factor 2 of the transformation is achieved by the concurrent computation of two real rows. The drawback of this
solution is the overhead imposed by the support of the separate and unseparate operation.

The SPITFFIRE processor used on the FDE implements this variant.

In details, the separate (3.4) and unseparate (3.4) operation are defined as follow [Mitra 93]:

Separate:

\[
X[k] = \frac{1}{2}(Z_r[k] + Z_r[N-k]) + j\frac{1}{2}(Z_i[k] - Z_i[N-k]),
\]
\[
Y[k] = \frac{1}{2}(Z_i[N-k] + Z_i[k]) + j\frac{1}{2}(Z_r[N-k] - Z_r[k]),
\]
\[
X[0] = \{Z_r[0], 0\} \quad \text{and} \quad Y[0] = \{Z_i[0], 0\}
\]

with \( k = 1, \ldots, \frac{N}{2} \)

\( Z[0 \ldots N] \): transformed row

\( X[0 \ldots N/2], Y[0 \ldots N/2] \): separated half-rows

Unseparate:

\[
Z[k] = (X_r[k] + Y_i[k]) + j(X_i[k] - Y_r[k]),
\]
\[
Z[N-k] = (X_r[k] - Y_i[k]) + j(-X_i[k] - Y_r[k]),
\]

with \( k = 1, \ldots, \frac{N}{2} \)

\( X[0 \ldots N/2], Y[0 \ldots N/2] \): half-rows

\( Z[0 \ldots N] \): unseparated row

The subscripts \( r \) and \( i \) in (3.4) and (3.5) denote the real and imaginary part, respectively.

Figure 3.13 presents the FB-CCF after the application of the RV FFT approach on each one of the phases.

### 3.4.2 Application to the RMA

The advantage brought by the RV FFT, with respect to the computational requirements, resides in the fact that the firing rates of the 1D FFT and CMUL operators are reduced by a factor of 2 if compared with the ones in Figure 3.3. The drawback is represented by the insertion of two new operators (SEP and UNSEP) in the algorithm flow. Both separate and unseparate have a complexity of \( 2N - 4 \) real additions,
which is considerably less than the FFT complexity. Thus the use of the RV FFT approach on the FDE is advantageous.

Furthermore a second relevant benefit is brought about by the RV FFT approach concerning the FDE: the reduction of the memory space requirements (practically a factor of 2 as well) which applies to every complex data set in the system.

Since both the SEP and UNSEP operations are computed on a row of complex data, they can be optimally integrated in the RMA mapping of Figure 3.8. A complete SDF of the RV FFT mapping on the RMA is depicted in Figure 3.14. With respect to the mapping without RV FFT the following differences are present:

- SEP and UNSEP have been introduced in phase A and C resp.
- the size of complex data sets TMP1 and TMP2 is reduced. This is visible by a comparison of the reduced indexes describing the size of TMP1 and TMP2 in Figure 3.14.
- the firing rates reflect the improvement achieved by RV FFT
- the GET and PUT are slightly modified in order to incorporate
3.4. Real Valued FFT

Figure 3.14: RV FFT mapping on the RMA. The firing rates of relevant operators are shown with a black background.

the R2C and C2R operations respectively, which are easily realized with a corresponding data handling during data transfer.

The iteration number for each phase cycle is modified in order to entail the improvement brought by the RV FFT and consider the pipelined execution still present in the RMA mapping. With respect to the previously introduced repetition numbers $N_{PHx}$ (3.3), which only consider
the effects of the SPMD approach, new $N_{PH_{ARV}}$ can be defined.

\[
N_{PH_{ARV}} = \frac{N}{2P} + 2 \quad (3.6)
\]

\[
N_{PH_{BRV}} = \frac{N/2 + 1}{P} + 2 \quad (3.7)
\]

\[
N_{PH_{CRV}} = \frac{N}{2P} + 2 \quad (3.8)
\]

Qualitatively the division by $P$ is given by the SPMD approach while the division by 2 is given by the FV-FFT optimization.

*Figure 3.15: Data handling on FDE in the tree phases by supporting RV FFT*

Figure 3.15 gives a more visual idea of the logical dimensions of the different data-set handled by the FDE with the RV FFT approach.

### 3.5 General Purpose Processing Engine

#### 3.5.1 Maximum Extraction Procedure

In contrast to the precise definition of computational tasks present in the first part of the FB-BST algorithm (Figure 2.4), the part relative to the MEP is characterized by a much more heterogeneous environment.
In fact the computational requirements can go from a minimal procedure represented by the normalization and maximum peak extraction of the FB-BST results, up to complex (application dependent) computations. For instance the complexity can be rapidly increased by more demanding searches in the images, targeting the extraction of several relative maximum values in the correlated image. Other applications can require post-processing tasks such as the tracking of localization candidates over the time.

Furthermore according to Section 3.1 the GPE has also to support the normalization procedure consisting of a squaring operation followed by a division of the results delivered by the FDE (Fig. 2.4). This operation is executed together with the MEP, since data has already been loaded into the CPU in (3.9) or (3.10).

Let’s consider the basic procedure to be supported by the GPE:

\[
\begin{align*}
PEAK &= 0; \ X\_PEAK = 0; \ Y\_PEAK = 0; \\
\text{for } (x\_index = 0..\text{SEARCH\_AREA}) & \quad \text{# for each image point} \\
& \quad \text{for } (y\_index = 0..\text{SEARCH\_AREA}) \\
& \quad \quad \text{SQ\_CCF} = \text{CCF}[x\_index][y\_index]^2; \quad \text{# square CCF result} \\
& \quad \quad \text{ENERGY} = \text{ENERGY}[x\_index][y\_index]; \\
& \quad \quad \text{BST} = \text{SQ\_CCF}/\text{ENERGY}; \quad \text{# NORMALIZATION} \\
& \quad \quad \text{if } (\text{BST} > \text{PEAK}) \text{ then} \\
& \quad \quad \quad \text{PEAK} = \text{BST}; \quad \text{# compare with previous peak} \\
& \quad \quad \quad \text{X\_PEAK} = x\_index; \quad \text{# store peak values} \\
& \quad \quad \quad \text{Y\_PEAK} = y\_index; \quad \text{# store position} \\
& \quad \quad \quad \text{end if;} \\
& \quad \quad \text{end for;} \\
& \quad \text{end for;} \\
\end{align*}
\]

The critical point of the normalization procedure (3.9) is the floating point division. In recent floating point units, emphasis has been put on designing faster adders and multipliers, with division receiving less attention. While the typical range for addition and multiplication latency is one to eight cycles, double floating point precision latency ranges from six to 60 cycles [Obermann 97].

For the absolute maximum localization task, the use of division can be avoided because there’s no interest in the normalized image but only in its highest peak. Thus in the procedure (3.9) the computational kernel can be rewritten as shown in (3.10): the division is
eliminated by doing a comparison between multiplied values instead of dividing them [Aschwanden 93]. This involves the storage of both CCF (CCF_PEAK) and energy terms (ENERGY_PEAK) found at the highest peak.

\[
\begin{align*}
\text{CCF}_{\text{PEAK}} &= 0; \quad \text{ENERGY}_{\text{PEAK}} = 0; \\
X_{\text{PEAK}} &= 0; \quad Y_{\text{PEAK}} = 0; \\
\text{for } (x_{\text{index}} = 0..\text{SEARCH}_{\text{AREA}}) & \quad \# \text{ for each image point} \\
& \quad \text{for } (y_{\text{index}} = 0..\text{SEARCH}_{\text{AREA}}) \\
& \quad \quad \text{SQ}_{\text{CCF}} = \text{CCF}[x_{\text{index}}][y_{\text{index}}]^2 \quad \# \text{ square CCF result} \\
& \quad \quad \text{ENERGY} = \text{ENERGY}[x_{\text{index}}][y_{\text{index}}]; \\
& \quad \quad \text{if } (\text{SQ}_{\text{CCF}} \times \text{ENERGY}_{\text{PEAK}}) > (\text{ENERGY} \times \text{CCF}_{\text{PEAK}}) \text{ then} \\
& \quad \quad \quad \text{CCF}_{\text{PEAK}} = \text{SQ}_{\text{CCF}}; \quad \# \text{ store peak values} \\
& \quad \quad \quad \text{ENERGY}_{\text{PEAK}} = \text{ENERGY}; \\
& \quad \quad \quad X_{\text{PEAK}} = x_{\text{index}}; \quad \# \text{ store peak position} \\
& \quad \quad \quad Y_{\text{PEAK}} = y_{\text{index}}; \\
& \quad \quad \text{end if;} \\
& \quad \text{end for;} \\
& \text{end for;} \\
\end{align*}
\]

(3.10)

It is now possible to define the minimal requirements of the normalization procedure in terms of clock cycles assuming that floating point arithmetic is supported with a single clock cycle latency for each operation. According to this the number of cycles required by the instructions internal to the loop is 10, consisting of a worst case evaluation of the conditional code giving 8 cycles increased by the 2 initial data load operations in (3.9) and (3.10). For a single frame the number of floating point operations (FLOPS) is shown in Table 3.1.

<table>
<thead>
<tr>
<th>Image/Template Size</th>
<th>1024/128</th>
<th>512/64</th>
<th>256/32</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Search</td>
<td>$10.5 \cdot 10^6$</td>
<td>$2.62 \cdot 10^6$</td>
<td>$655 \cdot 10^3$</td>
<td>FLOP</td>
</tr>
<tr>
<td>Reduced Search</td>
<td>$8 \cdot 10^6$</td>
<td>$2 \cdot 10^6$</td>
<td>$501 \cdot 10^3$</td>
<td>FLOP</td>
</tr>
</tbody>
</table>

Table 3.1: Computational requirements of the minimal GPE procedure

A reduction of the computational requirements can be achieved by assuming that a template of size $T$ can't be localized in a border region of $T/2$ pixels in the search image (search area height/width is then $(N-T)$). Thus the search loop can be reduced by $T$ in both dimensions, leading to the requirements shown in the last row of Table 3.1 with $T = 128$, $T = 64$ and $T = 32$ respectively.
3.6. System Level Task Scheduling

3.5.2 Parallelization of GPE

The order of magnitude of the required performance of Table (3.1), which has still to be multiplied by the frame rate, imposes parallel execution on the GPE. An SIMD parallelization approach has been chosen for the GPE since it offers the highest grade of flexibility (Fig. 3.1). Furthermore the programming model for a shared memory machine remains simple with an increasing number of CPU's, making program implementation easier to be realized by different final applications of the system. The minimal localization procedure requires only 2 read operations from the global memory and this keeps the communication vs. computation ratio of the algorithm in the range of 2:10. Thus assuming a CPU with direct memory access (DMA) capability and a reasonable local cache, it is possible to achieve linear speedup of the normalization procedure with up to 5 CPU's.

By a minimal NCCF implementation the inter-processor communication is involved only at the end of the computation, since the global maximum in the image has to be extracted among the GPE processors. Thus the sequential part of the algorithm is minimal and linear system speedup can be realized.

3.6 System Level Task Scheduling

The computations in the frequency domain and the MEP of the localization procedure (Fig. 2.4) are placed according to the scheduling shown in Figure 3.16 on the FDE and GPE respectively. Considering a single frame $F$, the two main tasks are scheduled following a pipelined execution. The FB-CCF procedure is executed twice by the FDE for each frame: one time for the computation of the CCF values and a second time for the computation of the energy terms. Concurrently to this, the GPE normalizes the previously processed frame and extracts the local maximums.

The data exchange between the GPE and FDE is carried out using the VBS, due to its global characteristics. According to the system level process scheduling (Fig. 3.16) the data is transfered from the FDE main memory into the GPE memory at the end of each FB-CCF procedure. This operation overlaps with the ongoing computation on the GPE which has to be halted during data transfer operations.

In order to connect the two subsystems, the transparent switch (Fig.
Figure 3.16: MPIC: system level process mapping and scheduling

3.1) present between the VBS and the GPE shared bus is turned on. When data I/O between FDE and GPE is terminated the switch divides the two shared buses again, so that computation can occur independently on the two subsystems.

The following timing parameters are defined here:

- \( T_{FD} \): time for computation of the FB-CCF
- \( T_{NR} \): normalization and maximum extraction time
- \( T_{IF} \): input frame time
- \( T_{OF} \): output frame time
- \( T_{FRAME} \): MPIC cycle time
- \( T_{LOC} \): total time for a localization

The system frame rate is determined by:

\[
FRAME_RATE = \frac{1}{T_{FRAME}} \quad (3.11)
\]

Due to the pipelined execution of the FDE and GPE procedures, a latency of \( T_{BST} \) is introduced.

\[
T_{LOC} = 2 \cdot T_{FRAME} \quad (3.12)
\]
This doesn’t have severe repercussions on most of the real image processing applications where the frame rate is much more important than the answer time of the system.
Chapter 4

RMA Implementation

In this chapter the RMA architecture of the FDE is confronted with the limitations imposed by state of the art technology. The FDE is particularly influenced by the memory device access times which are investigated in conjunction with the management of the data transposition during communication on the VBS. An HW efficient data transposition mechanism leading to maximal memory bandwidth exploitation is presented. Other investigations concerns the scalability of the FDE performance.

The investigations on bus and memory bandwidth are restricted to the VBS because the implementation of the HBS doesn’t require any particular measurement to be carried out. Being a point-to-point connection between a communication controller and a memory device (the XBAR is fully transparent) it’s implementation falls into the class of classical DRAM controllers. The only scope for memory bandwidth optimization on HBS resides in the exploitation of emerging DRAM technology which is also considered for the VBS implementation.

Figure 4.1 shows how critical the VBS can be with respect to the the system communication balance. View that the PE’s produce on VBS the same amount of data they receive from the HBS, it can be inferred that VBS has to provide a throughput $P$-times higher than the one of HBS. The fact that a buffer is present between the communicating nodes and that the PE is involved in complex computations requiring a certain amount of time reduces the requirements to the VBS. The numerical examinations of the balance between communication and computation is faced in Chapter 6 and 7 whereas here the conceptual optimization
of the VBS throughput is analyzed. In general bus throughput is given by the multiplication of the number of bits of its data-path by the bus working frequency. Since HBS and VBS use the same physical word representation being connected to the same memory components, the only VBS optimization opportunity is given in principle by the maximization of its working frequency.

4.1 Image Transposition on VBS

The image transposition occurs concurrently with the PUT operation (Fig. 3.8 and 3.10) performed on VBS where the data present in the local cache of the PE’s are written back into the main memory banks configured in shared mode.

The transposition process is modeled with $P$ producers having a data stream to be delivered and distributed to $P$ consumers (the memory banks). Between these two entities there’s the VBS which is shared by all nodes (PE’s and memories) participating in the communication process. The image transposition is achieved by means of a centralized control of two main components of the communication paradigm:

**Producers Data Sequence:** each producer prepares the data-stream which has to be put on VBS in an output buffer. The data in
the buffer is organized in order to be put on VBS following a predefined sequence. The producer itself does not control the access to VBS, which is managed by a central VBS controller (VBS-C). Thus controlling all producers, the VBS-C can generate a data stream on VBS, consisting of a portion of the data streams made available by the producers.

**Memory Addressing Scheme:** In conjunction with the generation of the data bursts, the VBS-C activates the memory cells where the data has to be written following a predefined addressing scheme. This implies the global address generation as well as the cell activation with memory control signals.

The transposition is realized by means of the data-address relationship (DAR) existing between the produced data stream and the addressing scheme on the consumer side (Fig. 4.2). Assuming a number of $W$ words present in the producer side, there are $W!$ different data stream which can be produced and therefore $W!$ corresponding addressing schemes leading to image transposition. Furthermore other mappings of the data set can be realized by this kind of infrastructure merely by influencing the DAR between producers and consumers.

The FDE uses the DAR depicted in Figure 4.2. The presented transposition technique also carries out in an efficient way the data redistribution into the different memory banks connected to VBS. The choice of
this scheme is motivated by the result of investigations involving memory bandwidth exploitation which are discussed in the next sections.

4.2 Memory Bandwidth Exploitation

4.2.1 Dynamic Memory Enhancements

Considering that the RMA architecture targets high resolution image processing and that the complex data format is involved, the required storage resources assume the dimensions listed in Tab 4.1. Thus the use of static RAM (SRAM) for main memory has to be excluded due to its cost, leaving as only alternative the use of dynamic (DRAM).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Bank</th>
<th>Logical Size [k Words]</th>
<th>SPITFFIRE Size [MBytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( im(x, y) )</td>
<td>real</td>
<td>A</td>
<td>1024 256 64</td>
<td>4 1 0.24</td>
</tr>
<tr>
<td>TMP 2</td>
<td>comp.</td>
<td>A</td>
<td>513 131 33</td>
<td>4.7 1.18 0.3</td>
</tr>
<tr>
<td>( TP(r, s) )</td>
<td>comp.</td>
<td>B</td>
<td>513 131 33</td>
<td>4.7 1.18 0.3</td>
</tr>
<tr>
<td>( BTP(r, s) )</td>
<td>comp.</td>
<td>B</td>
<td>513 131 33</td>
<td>4.7 1.18 0.3</td>
</tr>
<tr>
<td>TMP 1</td>
<td>comp.</td>
<td>B</td>
<td>513 131 33</td>
<td>4.7 1.18 0.3</td>
</tr>
<tr>
<td>( ccf(x, y) )</td>
<td>real</td>
<td>B</td>
<td>1024 256 64</td>
<td>4 1 0.24</td>
</tr>
<tr>
<td>( Energy(x, y) )</td>
<td>real</td>
<td>B</td>
<td>1024 256 64</td>
<td>4 1 0.24</td>
</tr>
<tr>
<td>TOT</td>
<td></td>
<td></td>
<td>7186 1792 448</td>
<td>30.8 7.7 1.92</td>
</tr>
</tbody>
</table>

Table 4.1: *FDE memory space requirements: for the SPITFFIRE case a complex data format of 72 bit/word and 32 bit/word for real data are used (Section 5.1). The definitions of Bank A and B refers to the one of Figure 3.7.*

Influenced not only by technological aspects but also by volatile market conditions, the cost/bit of DRAM is a factor 6 to 8 lower compared to the one of SRAM. The drawback of DRAM resides in its higher access time (Tab. 4.2), making memory bandwidth the most critical element in the performance definition of today's computer industry. In fact since 1987 microprocessor performance improved 55% per year against a 7% performance improvement per year in main memory access time [Hennessy 90]. Hierarchical memory architectures, based on memories of different speed and size, have been the most used solution up to now aiming to reduce the performance gap. Concurrently to this several approaches have been proposed to improve the bandwidth of DRAM's
4.2. Memory Bandwidth Exploitation

components. The most successful products currently in use are known as Extended Data Out (EDO) RAM [Hitachi 97], Enhanced DRAM (EDRAM) [EMS 97] and the latest families based on synchronous access methods named SDRAM [Mitsubishi 97].

<table>
<thead>
<tr>
<th>Type</th>
<th>$T_{AROW}$ [ns]</th>
<th>$T_{ACOL}$ [ns]</th>
<th>$T_{PREC}$ [ns]</th>
<th>$T_{CYCLE}$ [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM</td>
<td>60</td>
<td>40</td>
<td>40</td>
<td>110</td>
</tr>
<tr>
<td>EDO RAM</td>
<td>60</td>
<td>25</td>
<td>40</td>
<td>104</td>
</tr>
<tr>
<td>EDRAM</td>
<td>35</td>
<td>15</td>
<td>25</td>
<td>60</td>
</tr>
<tr>
<td>SDRAM</td>
<td>50</td>
<td>10</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>Asynch. SRAM</td>
<td>10-15</td>
<td>10-15</td>
<td>-</td>
<td>10-15</td>
</tr>
<tr>
<td>Synch. SRAM</td>
<td>&lt;10</td>
<td>&lt;10</td>
<td>-</td>
<td>&lt;10</td>
</tr>
</tbody>
</table>

Table 4.2: Typical Advanced RAM access times (1998)

- $T_{AROW}$: Time to access the first cell of a new memory row
- $T_{ACOL}$: Access time within the same row
- $T_{PREC}$: Precharge time before initiating new row access
- $T_{CYCLE}$: Cycle time for a single random access

Available state of the art DRAM families are characterized by some common aspects [Katayama 97] which have to be examined in order to optimize bandwidth exploitation:

**Memory cell:** has basically remained the same which was already used in the first DRAM families. Based on a capacitor as carrying information device, it provides high silicon area efficiency but requires periodical refreshing of its content.

**Multiplexed Addressing:** the memory cells array is accessed in two steps. Firstly the array row is selected and then the required memory location is selected on the current row using the column address. The row and column address are put subsequently (multiplexed) on the same pins of the memory device. This reduces the package pin count but slows down the memory access.

**Conditioned Bandwidth Improvement:** the access time reduction provided by the latest DRAM evolutions is effective only when words belonging to the same memory row are accessed. In fact the different evolutions of DRAM are based on different solutions such as internal data buffering (EDO RAM), internal data caching with SRAM cells (EDRAM) or synchronous internal interleaving (SDRAM) [Katayama 97]. All these approaches are effective on a
reduced portion of the memory array, which is a complete physical row of the memory array or a part of it.

Table 4.2 presents a comparison between some of the most affirmed DRAM evolutions. It appears that a substantial improvement has been achieved in \( T_{ACOL} \) only, where a reduction of factor 2 or more has been realized with respect to the standard DRAM's. In contrast to this the cycle time, which considers initialization and precharge requirements for a new row access, have not been considerably improved.

As a consequence of these considerations, maximal memory bandwidth exploitation is realized only when in a large 2D data access pattern, the accesses on the same memory rows are grouped together and executed subsequently.

### 4.2.2 Memory Efficient Transposition

Among the various possible DAR's which realize the matrix transposition, the one presented in Figure 4.2 has been chosen because of the following characteristics: data bursts consist of all the data elements present in the producers output buffers which have to be written into the same consumer memory row.

A much simpler alternative DAR, from the point of view of the producers, is the generation of a data burst consisting of consecutive data from the complete local buffer for every producer. But in this case the memory addressing scheme generated by the VBS-C, requires column-wise access of the memory locations, leading to inefficient memory bandwidth exploitation.

The length of the data burst, which consist of \( P \) logical words with the proposed transposition DAR, is also influenced by another system parameter: the physical representation of a logical word in memory. Depending on relevant implementation parameters such as the arithmetical word representation, the depth of the used memory components and the width of the VBS data buses, the physical storage of a logical word can require more than one physical memory location. The complex data format with its vectorial word representation further increase required physical space per logical word. For simplicity purposes a parameter \( R \) is introduced representing the implementation dependent number of storage cells per logical word.
4.2. Memory Bandwidth Exploitation

Figure 4.3: Graphical representation of $RSM_{BW}$ (4.4) with 3 types of DRAM's and various $R$ factors.

$R$: number of physical memory cells per logical word

The value of $R$ directly influences the length of the produced VBS data burst ($VBS_{BL}$) by multiplying the logical length given by the number of PE's.

$$VBS_{BL} = P \cdot R \quad (4.1)$$

Once that the length of a data-burst is known, in (4.2) the duration of the transfer of $VBS_{BL}$ words is divided by the number of transmitted words, leading to the definition of the average access time $\hat{T}_{AC}$. For the computation of the duration, it is assumed that for the access to the first word of a new row $T_{AROW}$ is required. This is followed by the transmission of $VBS_{BL} - 1$ words requiring each $T_{ACOL}$. At the end, $T_{PREC}$ has to be respected before the transmission of a new burst can be initiated (4.2).

$$\hat{T}_{AC} = \frac{T_{AROW} + (VBS_{BL} - 1)T_{ACOL} + T_{PREC}}{VBS_{BL}} \quad (4.2)$$
\[ RSM_{BW} = \frac{1}{T_{AC}} \] (4.4)

The inverse of the average access time \( T_{AC} \) (4.4) is used as a measure of the realized sustained memory bandwidth \( RSM_{BW} \), depicted in Figure 4.3 for different DRAM devices and factors \( R \) as a function of the PE number. The values of Table 4.2 are used in (4.4) to generate the plots of Figure 4.3.

The sustained memory bandwidth shows a high dependency on \( P \) by values between 2 and 16, while with larger \( P \) the \( RSM_{BW} \) bandwidth curve nears the maximal available bandwidth for each memory type given by \( 1/T_{ACOL} \). After analyzing the impact of the number of PE's

![Figure 4.4: Sustained bandwidth gain factor obtained by the DAR used with respect to a worst-case DAR accessing the consumers column-wise.](image)

and the physical word representation on the main memory bandwidth assuming the best case DAR of Figure 4.2 being used, a comparison can be made with the memory bandwidth delivered by the use of a worst-case DAR accessing the consumer memory column-wise. This implies
4.2. Memory Bandwidth Exploitation

the investment of $T_{CYCLE}$ for every word transmitted on VBS.

$$BG_{DAR} = \frac{T_{CYCLE}}{T_{AC}}$$  

(4.5)

The bandwidth gain ($BG_{DAR}$) between best and worst case DAR can be established by (4.5) and is graphically represented in Figure 4.4. The benefits of the used DAR are visible in Figure 4.4 and thus lead to the development of a corresponding VBS data transmission paradigm.

4.2.3 VBS Token Passing Network

The transposition described above is implemented on VBS by means of a very simple and hardware efficient *token passing* architecture (Fig. 4.5). This solution is based on the fact that the DAR is a priori known by producers and consumers connected to the VBS. Thus, after an initialization step, the PE’s (preparing data) and the VBS-C (generating addresses) can work in an almost autonomous way with minimal interaction.

From the PE point of view, the words belonging to the row to be transposed are prepared in an output cell according to the DAR output data sequence. This output cell communicates with the data path of VBS. Moreover the output cell is filled by the PE but its connection to the VBS is controlled by the VBS-C. When a PE sees that its output cell has been activated by the VBS-C causing the transmission of its content, the next data of the output sequence is put into the output cell (Fig. 4.6). The VBS-C provides the output cell activation signals to every PE synchronized with the addresses and control signals for the consumers memory cells.

The activation signal is distributed to the PE’s by means of a token passing network (Fig. 4.5). Only one PE at time has the permission to put the data present in its output cell on the VBS. Once it has finished, it delivers the permission signal (the token) to the next PE in the chain. Thus the sharing of the VBS data bus among the PE is realized, without further arbitrage resources. The token passing process is initiated by the VBS-C by launching the token in the network.

Figure 4.5 shows the hardware implementation of the token passing network. The output cell is realized with a register provided with tristate buffered outputs connected to VBS. The tristate enable signal (the permission) is locally stored by the PE in a D flip flop (FF). The default situation on every PE is the one with the tristate output in the
Figure 4.5: *Token passing architecture for data output enable on VBS.*

high impedance position, thus leaving the VBS free to the other nodes. The token passing network is realized by connecting the enable FF's in a chain having the VBS-C at the beginning which also provides the clocking signal. In this way the VBS-C provides the first activation token to the chain and makes it propagate through the network by moving the clock signal. The activation token is directly followed by a deactivation one, so that a PE writes a word on VBS and becomes idle in the next clock cycle.
4.2. Memory Bandwidth Exploitation

Figure 4.6: Token passing state charts for VBS-C and PE.

The significant advantage offered by the token passing implementation of the transposition communication paradigm resides in the considerably reduced overhead imposed to the HW. In fact a tristate buffer is normally present on output register connecting internal chip data signals to a bus. The token passing network only requires a data line and a clock distributing network.

4.2.4 Alternative to the VBS Transposition Implementation

The communication paradigm of the transposition in a multiprocessor architecture similar to the one used for the FDE, is characterized by a *all to all* data redistribution. In fact every producer has to distribute and transpose a subset of data to every consumer. Moreover the VBS transposition scheme makes the producer data available for every consumer. The VBS-C selects which consumer has to grab a word from the data bus.

Starting from these remarks an effective implementation of the transposition can be realized with a pipelined ring network in conjunction with *intelligent communication* [Mühl 96] [Mühl 95]. In particular this topology shows very good scalability issues, imposing practically no limits to the physical system size. An increasing number of communication nodes has no effect on the network bandwidth: only the data latency is influenced by this. Each node connected to the ring can act as producer or consumer, depending on its role in the current communication phase.
But with respect to the RMA architecture and considering the strong requirements concerning compactness imposed on the system, this alternative solution for VBS appears impracticable. In particular the following aspects have to be pointed out.

The HW requirements for the pipelined ring are considerably higher. This is mainly because of the fact that on the producer as well as on the consumer side a communication controller has to be provided in order to manage the data transmitted on the ring. Producers have to be synchronized during the generation of the data stream, while consumers must be able to select the right data from the data stream. The data selection mechanism is based on communication parameters which are set during initialization and usually require operations such as address comparison which are quite expensive to be realized in hardware. Furthermore in conjunction with the RMA architecture, the presence of a controller on the consumer side makes the connection between bus system and memory modules different for HBS and VBS. Thus transparent switching from shared to distributed memory is considerably hindered.

On the producer side, the proposed VBS implementation is particularly HW efficient with respect to a ring communication controller, since only a register with tristate outputs and a local address generator are required. No synchronization has to be provided for the data injection in the bus system. The only global timing signals are provided by the VBS-C by clocking and initiating the token passing procedure.

Concerning data throughput the token passing solution for VBS has the drawback of being influenced by the number of PE’s but is based on a much simpler communication method with respect to a ring, leading to faster clock cycles. In fact a ring is normally considerably slowed down by the data selection procedure executed by every attached node on incoming data.

### 4.3 VBS Throughput Improvement

#### 4.3.1 RMA Scalability

The problem to be considered here is how the FDE system performance scales with respect to the number of PE’s.

Considering the scheduling of the three phases executed on the FDE (Fig. 3.9), it can be deduced that linear system speedup is achieved
if the communication processes don’t introduce wait states at CPU level when a larger number of PE’s is allocated. As a consequence, for phase A, B and C the conditions to be fulfilled which are directly readable for Figure 3.9 are:

\[ \max(T_{GET}, T_{PUT}) < T_{CPU} \quad \text{for phase A and C} \quad (4.6) \]
\[ \max(2 \cdot T_{GET}, T_{PUT}) < T_{CPU} \quad \text{for phase B} \quad (4.7) \]

If (4.6) and (4.7) are respected, the iterative repetition of the phase cycles lead to a 100% load of the CPU and the parallelization grade brought about by the the SPMD approach is fully exploited.

It is therefore determinant to analyze the dependency between \( P \) and \( T_{GET}, T_{CPU} \) and \( T_{CPU} \) in order to determine if (4.6) and (4.7) can be broken by an increasing number of PE. From a qualitative point of view the following considerations can be made:

\( T_{PUT} \): strongly depends on \( P \). This because of the shared characteristic of VBS where only one PE at a time can access the main memory. Thus by an increasing \( P \) it is possible that (4.6) and (4.7) aren’t satisfied any more.

\( T_{GET} \): isn’t influenced at all by \( P \) because of the distributed character of HBS. Every PE can independently communicate to the main memory bank connected to it, through its own HBS. It’s determined by bus and memory bandwidth as well as by the number of words to be transmitted for each row.

\( T_{CPU} \): is determined by the CPU performance only.

As a consequence, it can arise that the VBS is the system bottleneck concerning linear performance scaling.

### 4.3.2 VBS Parallelization

Before entering the discussion on which system parameters do determine \( T_{PUT} \) (faced in Section 6.2.1), a considerable improvement of the VBS throughput is offered at the architectural level. This is carried out by providing more than one data path in the VBS, thus multiplying its throughput.

The communication paradigm of the transposition allows to parallelize only the data lines of VBS and make a common use of the
addresses and control lines already present in a single data path VBS. On the producer side, each PE has to provide \( VBS_{PAR} \) I/O data ports, each one connected to a different VBS data-path.

\[ VBS_{PAR} : \text{ number of VBS data paths} \]

On the contrary on the consumer side only one data port is required, as in the case of a single VBS data path. This is because the transposition permits to pre-define into which memory bank, a word present in the producer data buffer has to be written. Thus it is possible to route the parallel data-paths from all the producers to a sub-set of the consumer.

![Diagram](image)

**Figure 4.7:** Parallelization of VBS (factor 2) on the producer side: a portion of the row being transposed is routed to VBS DATA1 and the other portion to VBS DATA2. Token passing network and PE addressing scheme are only slightly modified with respect to the \( VBS_{PAR} = 1 \) case.

Figure 4.7 depicts the effect of \( VBS_{PAR} = 2 \) on the producer infrastructure: the local address generation has to be extended in order to load data coming from the second half of the row into the output register. Assuming that the system has \( P \) PE’s, the data of VBS DATA1 are routed into memory banks \( 1 \ldots P/2 - 1 \) while the data of VBS DATA2 belong to the banks \( P/2 \ldots P \). A routing scheme for the \( VBS_{PAR} = 4 \) and \( P = 8 \) configurations is depicted in Figure 4.8.

The waiting time for the PE’s (Fig. 4.6) isn’t reduced with respect to a single data bus solution, but the throughput for each PE’s to the
Figure 4.8: A 8 PE' FDE with a 4 data bus VBS
main memory is multiplied by $VBS_{PAR}$. As a consequence, the global communication time is divided by the same factor.

In Figure 4.8 a system configuration with $VBS_{PAR} = 4$ is depicted. Note how with respect to a single VBS solution the routing of the data buses changes from an all to all character into an all to several character. This is another advantage introduced by the row-wise main memory access of the used DAR: all the rows destined to the same consumer memory bank are routed to it using the corresponding data bus.

Although the parallelization of VBS is based on a simple concept, the main drawback of this evolution of the RMA concept resides in the augmented requirements for the CC pin-out and on the board routing. In fact the CC has to provide $VBS_{PAR}$ I/O ports, each one with the number of data-signals imposed by the used word representation. The same multiplicative effect appears in the board-routing complexity.
Chapter 5

Processors and Communication Controller

The general discussions on the FDE and GPE of the previous chapters are bound to the peculiarities of the processors used. The SPITFFIRE chip is presented and the influence of its computing and I/O characteristics on the FDE implementation are discussed. Concerning the GPE processor, in addition to the computational requirements, particular weight is given to its role in the system-control concept, which is realized also by means of the CC. GPE processor and CC are also involved in the management at system level of the particular arithmetic format required by the SPITFFIRE.

5.1 The SPITFFIRE

The reduced number of functions as well as the low level of programmability present in the FDE are the best prerogatives for of an Application Specific Integrated Circuit (ASIC) implementation. Thus the SPITFFIRE (Smart Pipelined InTerleaved Fast FourIer pRocessing Engine) has been developed [Wosnitza 99] [Wosnitza 98].

The processor consists of a dedicated complex ALU and a local cache organized in three banks (Figure 5.1). The complex ALU is optimized for 1D complex FFT transform, complex multiplication and real valued
FFT support. The three memory banks can be used by the ALU as well as by a dedicated on-chip I/O controller, implementing parallel data communication and computation. On the input data path a data-flow processing unit supports some data preprocessing tasks needed to speedup the FB-BST algorithm.

Figure 5.1: *SPITFFIRE* architecture

<table>
<thead>
<tr>
<th>ALU Modes</th>
<th>FFT, element-wise complex multiply, separate, unseparate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Length</td>
<td>256 pt, 512 pt, 1024 pt</td>
</tr>
<tr>
<td>Computing Power</td>
<td>924 MOP's</td>
</tr>
<tr>
<td>Clock Rate</td>
<td>66 MHz</td>
</tr>
<tr>
<td>Accuracy</td>
<td>$2 \times 32 + 8$ Bits/Word</td>
</tr>
<tr>
<td>Technology</td>
<td>0.5 $\mu m$ 3 metal layer CMOS</td>
</tr>
<tr>
<td>Die Size</td>
<td>11.6 $\times 14.4$ mm</td>
</tr>
<tr>
<td>Complexity</td>
<td>2.3 M Transistors</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>1.7 W @ 66 MHz</td>
</tr>
<tr>
<td>On Chip Cache</td>
<td>3 $\times$ 9 KBytes</td>
</tr>
<tr>
<td>I/O Bandwidth</td>
<td>230 MB/s</td>
</tr>
<tr>
<td>I/O Port Clock Rate</td>
<td>50 MHz</td>
</tr>
</tbody>
</table>

Table 5.1: *SPITFFIRE* Specifications

The characteristics of the processing element are discussed with respect to their influence on system level aspects.
5.1. The SPITFFIRE

5.1.1 ALU

The SPITFFIRE disposes of an ALU specialized in the fast and accurate computation of the operation required by FDE (Table 5.2). Thus in addition to a radix-4 implementation of the FFT algorithm (for 1024, 512 and 256 input series) the point-wise complex multiplication (CMUL) as well as the special data rearrangement routines (SEP and UNSEP) required by the RV FFT are supported.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Row Size [points]</th>
<th>Used Banks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>FFT</td>
<td>80 µs</td>
<td>40 µs</td>
</tr>
<tr>
<td>CMUL</td>
<td>31.4 µs</td>
<td>15.7 µs</td>
</tr>
<tr>
<td>SEP</td>
<td>15.8 µs</td>
<td>16 µs</td>
</tr>
<tr>
<td>UNSEP</td>
<td>15.8 µs</td>
<td>16 µs</td>
</tr>
</tbody>
</table>

Table 5.2: Execution times of the SPITFFIRE for a single row. The last columns shows the number of local memory banks used by the operating modes.

Particular weight has been given to the ALU accuracy. As a compromise between fix-point and floating-point arithmetic, the implemented block-floating point arithmetic provides the required accuracy with 20% less silicon area with respect to a pure floating-point solution [Wosnitza 98]. The complex numbers are represented with 32 bits for the real part and 32 bits for the imaginary part. Additionally, following the block-floating point arithmetic, an 8 bit block exponent (BE) common to all the words belonging to a row is generated by the ALU.

By the use of BE arithmetic the SPITFFIRE achieves a relative error of 0.001% with respect to a double-precision floating-point computation when processing a single row. The effect of this error on the whole FB-CCF computation is smaller than 0.2% (always compared with a double-precision floating-point computation). Exhaustive simulations with various image material, have shown that this level of accuracy is acceptable for successful template localization.

The BE arithmetic of the SPITFFIRE has relevant repercussions at system level, because the scope of a BE is restricted to a given row. In conjunction with the 2D processing and with the parallel computation where multiple PE’s generate a set of BE’s for different portions of data,
a global BE management infrastructure is needed (Section 5.5) for the FDE.

5.1.2 Local Memory and I/O Port

The local memory organization of SPITFFIRE, in conjunction with its I/O capabilities is the most critical aspect of the processor with respect to the system performance. In contrast to the ideal FDE PE (Fig. 3.11), some limitations are imposed by the SPITFFIRE storage and I/O capabilities.

The processor disposes of three memory banks: each one can store a whole complex row of $1024 \times 72$ bits. Depending on the mode currently executed by the ALU, one or more banks are reserved for computational tasks (Table 5.2), thus the I/O capabilities are strongly conditioned by the processing activity. The worst case situation occurs during the CMUL mode, where all the cache banks are in use and no I/O activity can be carried out. Another limitation with respect to the ideal process

![Figure 5.2: SPITFFIRE chip die](image)
5.1. The SPITFFIRE

scheduling is imposed by the fact that only one I/O port is provided. Thus assuming that the target memory bank is accessible for communication, it is not possible to realize parallel communication over VBS and HBS as it would be required by the ideal case (Figure 3.9).

The concrete effect of these limitation on the FDE process scheduling is treated separately in Section 5.2.

5.1.3 Data flow processing during I/O

On the input I/O data path, two processing units are provided in order to execute two kinds of operations required in the BST algorithm and by the BE arithmetic management: a real data square and a conditional shift.

The squaring operation has to be carried out on the real image data before the execution of the first FFT in the energy computation algorithm part (Figure 2.4). Using a look up table (LUT) this operation is very efficiently carried out with 256 memory cells, since a real data accuracy of 8 bit is assumed.

The shifting operation is required in conjunction with the BE management concept at system level (Sec. 5.5). When data is read into the PE, the 8 bit BE exponent of the incoming word is subtracted to a reference BE (RFBE) and a logical shift right operation is executed according to the resulting difference.

Both the squaring and shifting operations are carried out during the communication process, applying data-flow computation on the incoming data stream. The only effect on the input process caused by the squaring and normalization step is the introduction of a clock cycle latency on the input path. The increased latency has no repercussion on the communication processes.

5.1.4 Programming

The chip is programmed by means of the Mode-Sel and Len-Sel signals (Fig. 5.1) which set the mode to be executed and the length of the processed data series. To be pointed out is the fact that each mode operates always on a 1024 words data-set. In the case that Len-Sel requires the computation of 512 or 256 points operations, 2 or 4 series respectively are concurrently processed in the same mode execution. ALU-Start and ALU-Busy are used for process control purposes.
Each SPITFFIRE mode requires a certain number of local memory banks according to Table 5.2. The selection of which one the memory banks has to be accessed by the chip ALU is made through the Bank-Config signals before the processing is started. With exception of the CMUL mode, which requires all the memory banks, concurrent to the computation carried out by the ALU, data I/O can independently take place on the unused bank by means of the IO-Adr and IO-Data ports.

5.2 SPITFFIRE Impact on FDE Task Scheduling

The ideal FDE scheduling of Figure 3.9 can now be modified taking into account the restriction imposed by the SPITFFIRE internal memory access and by the serialization of the I/O previously discussed. Furthermore, with respect to Figure 3.9 in phase A and C the SEP and UNSEP task are introduced according to the RV FFT approach (Fig. 3.14).

With respect to the a full parallel utilization of CPU, VBS and HBS the more evident limitations verifiable in Figure 5.3 are the introduction of task inter-dependencies which impose resource synchronization (ALU, HBS and VBS) and the insertion of wait states in the scheduling.

In comparison with a full parallel solution of Figure 3.9, each phase is modified as follows:

**Phase A**: VBS and HBS communication are carried out sequentially with the cache bank not used by the CPU.

**Phase B**: before the execution of the CMUL step, the previously computed row has to be put back into main memory and subsequently a transformed template row has to be loaded into the cache. During the CMUL task no I/O activity is possible because the all of the cache banks are used by the CPU.

**Phase C**: like phase A, the PUT and GET operations have to be implemented sequentially.

The length of the 3 phase-cycles is therefore determined not only by the length of the single processes, but largely by their conditioned sequencing. Several alternatives concerning the scheduling have been investigated but the memory blocking CMUL mode as well as the reduced I/O resources don’t leave much variation in scope to the one presented.
5.3. GPE Processor

5.3.1 The SHARC DSP

The ADSP21060 SHARC processor [ADI 97] is used as processing element in the GPE. This particular DSP has been chosen mainly for the following reasons:

Floating Point CPU: the SHARC achieves 40 MFLOP's (Multiply and Accumulate) with 32 bit IEEE floating point. The normal-
Multiprocessing: the DSP has been developed in order to work in a multi processor environment. A system with up to 6 DSP can easily be realized without further HW requirements for resource arbitration.

Interprocessor Communication: the SHARC feature six link ports that provide up to 240 MB/s data throughput. Each port consist of four bidirectional data lines, a bidirectional clock line and a bidirectional acknowledge line.

Controlling Features: advanced interrupt servicing capabilities combined with multiple register sets for fast context switching, optimally support the development of multitasking real time software kernels.

For the parallel implementation of the normalization procedure (3.10) with the minimal overhead in the GPE, the SHARC multiprocessing capabilities play a relevant role [Tewksbury 97] [Sgro 98]. By means of distributed on-chip bus arbitration logic, glue-less shared memory parallelization with up to 6 DSP’s can be efficiently realized.

Other version of this processor have been produced after the ADSP-21060. Targeting at low cost applications the ADSP-21061 and ADSP-21065 also achieve higher clock frequencies (50 MHz and 60 MHz respectively) but do not provide the link-ports which are required for the FDE. Moreover the on-chip memory has been reduced by a factor 4 and 8 on the ADSP-21061 and ADSP-21065 respectively.

5.3.2 Task Synchronization and Control

Concurrently to the computational tasks, one of the GPE processors must also assume the function of system master controller (SMC) involving the generation of control and synchronization signals for the whole system. The control and synchronization task can be assumed only by a processor of the GPE because on this part of the system only, there’s the possibility to have an overview of the tasks running at system level. In fact the processors of the FDE access a reduced portion of data (row). Furthermore the length of the tasks on the FDE is a priori known while the computation on the GPE, due to its
data dependent flow, requires a variable amount of time. Thus the controlling of the main algorithm flow has to be carried out on the GPE.

Basically the main activity of the SMC deals with the initialization of operational parameters for the communication and computation nodes of the system. These parameters specify the kind of operation and the size of the datasets involved. In particular, the GPE provides the control signals in order to realize the process scheduling at different levels.

The fact that the SMC has to concurrently provide image processing capabilities combined with real time external process controlling, imposes particularly high requirements on the CPU used for this purpose. State of the art microprocessors are usually conceived for optimal supporting of only one of these kinds of activity. Since typical micro-controllers are a long way from providing the required computational power for the GPE, it is much more suitable to locate the GPE CPU in the high-end DSP domain. The SHARC fulfills these kind of specifications.

In order to support the SMC function, another characteristic of the SHARC is exploited: the link ports. The same communication interface is implemented on the FDE communication controllers (Section 5.4) providing a system-level point-to-point service network (Fig. 3.1).

Using this channel the SMC executes all the system controlling tasks. Initialization parameters depending on the kind of operations which have to be executed and setup values for VBS and HBS communication are transmitted to the CC’s. In the same way handshake protocols for task starting and finishing are transmitted from and to the PE’s for synchronization purposes at different levels.

5.4 Communication Controller

The communication controller (CC) is the interface between SPITF-FIRE and the RMA bus system. The kind of tasks to be supported range from data communication to job synchronization, both aiming to generate and control the data traffic between main memory and SPITF-FIRE cache involved with the computation on the FDE. Furthermore, closely related to these functionalities required by the RMA architecture, some more specific tasks have been placed on the CC in conjunction with the characteristics of the FDE and GPE processors (Figure 5.4).
The different sub-blocks of the CC are described here from a behavioral point of view:

**HBS Interface:** the base functionality provided by this part is a row-wise transfer of data from main memory to the SPITFFIRE cache (GET operation). Thus on one side the used DRAM technology timing and on the other the SPITFFIRE I/O protocol have to be implemented in HW. The HBS controller carries out the data transfer procedure autonomously once it has been initialized with the address boundaries of the memory portion it has to transfer.

**VBS Interface:** this sub-block implements the fast data transposition (PUT operation) based on the token passing approach (Section 4.1). Near to the specific VBS data interface (Figure 4.5) an address generation structure has to be provided in order to feed the VBS interface with the data sequences read from the SPITFFIRE cache. The activation of the VBS data driver by the externally driven token passing network, has to be detected by the VBS interface in order to trigger the next read operation from the cache. The VBS controller also works autonomously once it has been initialized for the current row and only signalizes the end of its
5.4. Communication Controller

task to the program controller.

**Program Controller:** in this block the different tasks running on each PE at *phase cycle* level are managed. In real terms the HBS and VBS controller are initialized with the parameters for the data transfer and the CPU is set up for current operation in his cache (FFT, CMUL, ...). When these entities have been started, the program controller waits for task ending signaling and reacts accordingly, delivering the next operation to be executed or the next transfer parameters.

**Maximum BE extractor (MBEE):** this unit extracts the *local maximum BE* (LMBE) generated by the SPITFFIRE within the computation of a single *phase*. It is part of the BE management infrastructure.

The main activities supported by the CC are shown in Figure 5.5. For reasons of simplicity only a high level representation of the processes is provided.

Considering the HW implementation of the CC, particular attention has to be given to the complexity and flexibility which has to be provided. In fact the address sequences generated by the HBS and VBS controller are quite simple to generate with an HW architecture specific to one image size (1024... 256). Since by a given image size all the addressing schemes are a priori defined, the address generators can be efficiently tuned. If on the contrary the CC has to work for different image sizes, more intelligent address generators based on small arithmetical units have to be used, considerably increasing the CC complexity.

The best solution to this problem is represented by modern fast *field programmable gate arrays* (FPGA’s) or *complex programmable logic devices* (CPLD). With this kind of HW platform, which can be easily reconfigured, the image size dependent optimization of the address generators is carried out during the CC design *phase*, leading to fast and efficient HW implementations. Different optimized HW configurations are created for different image sizes and loaded into the system during configuration or even during runtime, keeping HW complexity low.

State of the art CPLD’s and FPGA’s do provide enough HW resources in order to support the CC functionalities. These programmable devices achieve internal peak processing frequencies greater than 100 MHz. Limitations can be imposed by insufficient pin number, in particular by higher VBS parallelization grade. In this case the CC architecture can be easily partitioned on more than one chip.
5.5 **BE Management**

The objective of the BE management infrastructure is to guarantee data consistency at system level, since the BE exponents generated by the SPITFFIRE have a scope restricted to a single row. The BE arithmetic of the processor assumes that the words belonging to a row being loaded into it’s cache have the same BE: this condition is broken by the image transposition. In fact after the transposition of the data, the new rows fed into the PE’s consist of data belonging to different rows in the previous *phase*, thus having different BE exponents.
The BE management is based on 4 elements: the 72 bit data format, the MBEE placed on the CC, the SPITFFIRE input normalizer and the SMC.

The BE arithmetic theoretically requires the storage of an 8 bit exponent for every image row. But since 2D image reorganizations do occur, every 64 bit complex word (32 bit real and 32 bit imaginary part) is provided with a further 8 bit for its private BE: thus a 72 bit word representation is used. This permits the disposal of the BE relative to each word, in every part of the system.

The next thing to do is to provide all the data being loaded into the processor cache with the same BE for further computation: this implies the normalization of the $32 + 32$ bit words of the row with respect to a common BE value.

In order to generate a common BE with system scope, at the end of each phase the SMC gathers the local maximum BE's (LMBE) gen-

**Figure 5.6:** Main state transitions and inter-process communication for the implementation of the BE management.
erated on every PE and extracts the global maximum BE's (GMBE). Before entering a new phase the GMBE is redistributed to all the PE
which store it as reference BE (RFBE) in the SPITFFIRE input normalizer. By the next phase all the data loaded into the SPITFFIRE
cache (with a GET operation) are normalized with respect to the RFBE which has a system scope by means of the input normalizer. Every word
is right-shifted according to the difference between its BE (taken from the 72 bit word) and the RFBE.

After the normalization stage, the private 8 bit BE of each word is set to zero. The data now satisfy the conditions for BE arithmetic and are stored in the SPITFFIRE cache, ready for the next phase computation.
Chapter 6

MPIC Model

In the first part of this chapter the design strategy which has lead to the concurrent system and SPITFFIRE design is presented. A FDE virtual prototype in VHDL permits the detection of critical aspects of its HW implementation and build the basis for a future successful FDE synthesis. Based on the knowledge gained by the VHDL representation, a precise performance model of the MPIC is obtained.

State of the art development of electronic systems is driven more and more by highly developed design methodologies [Teich 97] [Gajski 94] aimed to achieve first pass success while considering system performance, power consumption and cost requirements. Virtual prototyping is the key for the successful estimation of fundamental decisional steps in the development, prior to investments in implementation or fabrication.

6.1 Virtual Prototype

6.1.1 Design Strategy

The development of the MPIC imposes the partitioning of the problem complexity into smaller units which are resolved in parallel and by different designers. Early system partitioning (which can be structural/functional, HW/SW,...) is the most common approach for the generation of smaller sub-problems. But this kind of approach reduces in an early phase the design-exploration space by excluding a subset of possible implementations. The opposite late-partitioning approach
tries to maintain the highest number of system implementations up to
the last moment, where the best architecture is determined from a very
large set of candidates according to a given metric.

For the MPIC design a trade-off between the two partitioning
strategies has been used in order to reduce the complexity of the
tasks (and permit parallel development) without imposing from the
beginning a too narrow design space which could lead to unsatisfying
solutions. Following this idea, after the explorations and optimizations
at algorithm level, the image row has been defined as the fundamental
data element of the system development. It is evident that this choice
has been strongly indicated by the presence of the 2D FFT, but this
has also permitted to generate two independent design sub-spaces: one
dealing with efficient data-row handling in a parallelized architecture
and the one dealing with fast and accurate implementation of the
processing tasks.

Based on the semi-automated interaction of high level programming
(C, Khoros Software Design Tool) and hardware description languages
(VHDL), an integrated design environment has permitted flexibility and
consistency in the two main design spaces. The first-silicon success of
the SPITFFIRE design can be without any doubt considered as a con¬
firmation of the effectiveness of the design environment used (Fig. 6.1).
The system and processor design have also strong interacted with each
other since as a consequence of the specify-explore-refine flow running
in the two sub-designs, new requirements have emerged in advanced
design phases with repercussion at system and processor level.

**Local Memory:** the settlement of the image row as fundamental data
element has implied the existence of a non-blocking data buffer
as interface between the system and processor level. In the early
development phase, the two sub-designs have both worked with a
generic behavioral SRAM model which has permitted an exhaust¬
tive inspection at system and processor level of the interfaces to
this common element.

When the SPITFFIRE chip has entered the late synthesis phase,
considerations concerning ALU performance and chip pin-out have
resolved in the decision to implement an on-chip local memory. At
this point the behavioral model of the SRAM has been substituted
by a structural, technology-dependent one. The finally defined
memory size, organization and timing of the SRAM have been
integrated into the system model.
Figure 6.1: *MPIC design partitioning and environment.*

**RV FFT:** investigations carried out at processor level have shown the effectiveness of RV-FFT implementation using the separate/unseparate variant. Thus at system level the measures to adapt the different task scheduling (from Fig. 3.8 to Fig. 3.14) have been introduced.

**Data Representation:** the BE arithmetic implementation with its 72 bit/word format is the result of investigations which occurred in an advanced phase of the SPITFFIRE CPU design. At system level the choice of the BE arithmetic has imposed the development of the BE management infrastructure which has been added to the already present functionalities. This has on one side considerably expanded the tasks for the CC, while on the other side
the 72 bit words has been split into 2x36 bit words exploiting the widely used 36 bit wide DRAM devices.

Input Normalizer: this element has also been introduced by the BE arithmetic. A behavioral model placed in the CC has been used at system level to verify the effectiveness of the BE management concept. Only in a second phase the normalizer has moved on the chip, because of the speed improvement obtained by the ASIC implementation (compared to the one of a PLD).

Input Squaring: as in the case of the shifting operation, the squaring unit, which has been present in the system model since the early design phase, has been placed on-chip in the late SPITFFIRE design phase.

These are only the most representative examples of how the integrated design environment has permitted independent development and adaptability to modified specifications at the same time. The modifications imposed by new specifications appeared during advanced design phases (which in the case of the BE arithmetic have been relevant) have been easily integrated in the design. During every change system-wide verification and code reusability have respectively brought consistent development and efficient design flow.

6.1.2 FDE VHDL Model

Concretely, the modeling activity carried out at system level described in the previous section, is mainly limited to the FDE part of the system. The main objective of the modeling effort, apart from the exploration activity, still remains the generation of design entry code for the successive design synthesis which is of lower interest in the GPE (although automatic SW generation tools seem to gain popularity). Furthermore the FDE presented a high grade of uncertainty with respect of its HW implementation, which requires a fine grade modeling and refinement activity. For the GPE only the performance model is relevant, which doesn’t require any particular prototyping approach.

For the FDE virtual prototyping, the VHDL [Bhasker 97] [Hsu 95] language is used. One of its characteristics consists of the the definition of external interfaces for every block (so called entity), while maintaining a very flexible modeling style of the block content going from
a fully behavioral down to a gate-level structural design entry. Most CAE tools do support VHDL as design entry language: additionally in most cases post-synthesis and post-layout VHDL descriptions of the generated net-list are generated by the same tools. Thus back annotated block models can easily be reintegrated into the system model. This permits the optimal realization of the modeling process described above (Fig. 6.1).

Since it is not relevant in this work to present the FDE model in its contents, the accent is posed on the presentation of the abstraction levels used for the modeling of the different blocks the FDE (Fig. 6.2 and 6.3). The modeling style is divided for simplicity purposes into three main abstraction levels:

**High Level Behavioral:** large blocks, abstract data structures (files, arrays, ...), complex mathematical representation (integers, signed integers), abstract programming style.

**Low Level Behavioral:** small blocks, small data structures (words), 'block-oriented' programming style, symbolic 'state' handling (for finite state machine entry)

**RTL Structural:** Register Transfer Level description, HW oriented signal representation (Standard Logic Library), logical 'state' handling (FF's content).

The three abstraction levels are represented with corresponding grey levels in Figure 6.2: a detailed representation of the CC modeling strategy with the same color scaling is depicted in Figure 6.3.

The choice of the abstraction levels used for the FDE sub-blocks has been made in function of several parameters:

- all the communication front-ends on the CC, on the VBS-C and on the SMC CPU1 are modeled at RTL. This is because a precise investigation of their HW implementation is required. The critical aspects of their HW implementation have to be detected as soon as possible in the design flow, in order to identify the critical communication nodes and evaluate alternatives solutions.

- in the CC in particular (Fig 6.3) the RTL modeling builds a solid basis for its implementation. The precise knowledge of the CC
complexity at RTL and gate level, permits to make reliable performance predictions and make reliable evaluations concerning the target HW platform for the CC (Section 7).

- the memory devices (EDRAM) are modeled at RTL with timing information. The same is done with the crossbars switches, which are directly modeled from the data-sheet at gate level with delay information. As a result the memory bandwidth optimization (Section 4.2) can be qualitatively and quantitatively verified.

- the VBS-C model is also mostly modeled at RTL due to the relevant role it plays in the transposition process efficiency.

- in general the parts modeled with Low Level Behavioral code present in the CC and VBS-S, concern the complex Finite State Machine (FSM) definition, which is made easier by a symbolic state handling.

Figure 6.2: Abstraction levels of modeling used in FDE virtual model
6.1. Virtual Prototype

The FDE VHDL model builds the basis for the synthesis, layout and fabrication of the proposed RMA. By attaining an RTL model of the CC

- the FDE CPU is modeled with two levels: RTL for the local memory in order to define and test the CC I/O ports. Concerning the complex ALU a high level emulator has been realized aiming to simulate the data modification in the local memory and the task controlling requirements. In this case the use of a back-annotated model imported from the SPIFFIRE design would be better, but the complexity of the simulation code would become unmanageable for useful computing times on state of the art workstations.

- also for the SMC SHARC an emulator has been modeled aiming to study and realize the system controlling environment (Fig. 5.5) as well as the BE management structure (Fig. 5.6).

6.1.3 Virtual Prototyping Output
and the VBS-C the following problems have been exhaustively analyzed:

- the addressing units present in the CC memory controllers (on both sides) as well as the addressing unit present in the VBS-C, have been investigated with respect to the mathematical requirements involved by the HBS and VBS addressing schemes. Efficient hard wired address generator architectures support the required addressing schemes, without the use of slow to implement mathematical units (adders).

- global data consistency with respect to the switching between the distributed and shared memory domain has been realized and verified.

- on the VBS-C the token passing network has been functionally modeled, verifying the optimized transposition DAR (Fig. 4.2) implemented with EDRAM memory technology. Memory control signal generation is synchronized with complex address generation and token passing initialization.

- the feasibility of VBS parallelization has been verified and its impact on memory control signal has been investigated and implemented.

- the process controlling requirements at different levels have been identified. This has delivered relevant decisional parameters for the choice of SHARC as processing and micro-controller unit on the GPE.

- a SHARC link based interprocessor communication protocol has been implemented in the program controller located in the CC. As a result all the system controlling tasks managed by the SMC are supported.

- based on the same communication channel and protocol, global BE management has been modeled.

The fact that a verified VHDL model of the CC is disposed of makes an automated synthesis for a given HW platform easily feasible. Nevertheless, according to the experiences made during the SPITFFIRE design it is to be assumed that a consistent effort in VHDL and net-list tuning will be required in order to manage technology related timing problems.
6.2 Performance Models

The MPIC system performance model is built from the bottom by examining the HW element determining the bandwidth of the communication infrastructure which in its turn determines the values of $T_{GET}$ and $T_{PUT}$. In conjunction with the SPITFFIRE computing times and the phase schedulings, a global system performance model is obtained as a function of $P$, number of SHARC DSP's ($SH_{NUM}$), VBS parallelization grade ($VBS_{PAR}$) and memory type.

Another aspect considered by the model is the influence on the overall performance of the reduced SPITFFIRE I/O capabilities with respect to a full parallel RMA implementation. According to this, two different models of the system are produced: one for a best case FDE PE with full parallel I/O capabilities (characterized by the letters $BC$) and a second one taking into account the specific SPITFFIRE I/O issues (characterized by the letters $SP$).

6.2.1 HBS and VBS Bandwidth

The different HW component building the VBS and HBS combined with the communication protocols used on these resources are analyzed in order to determine the two bus working frequencies. The prediction is done by analyzing the maximum clock rates and data throughput of the communicating nodes involved in the VBS and HBS: the memory devices, the CC implemented with programmable logic and the FDE processor.

Basing on the information gained from the fine grade model of the RMA communication interfaces, the influence of board level routing as well as the XBAR propagation delay are not considered. In the case of the XBAR, zero propagation delay switches introduce a delay smaller than 1 ns [Quality 95] on the data path. Concerning the influence of the board routing delays, it is assumed that the range of frequencies involved by the FDE communication (40 MHz...100 MHz) are very well supported by state of the art board technology.

Regarding the SPITFFIRE, in addition to the already discussed limitations imposed by the limited I/O resources, it has to be pointed out that the maximal working frequency on its I/O path is 50 MHz. Accordingly a SPITFFIRE I/O cycle time ($T_{SP_{IO}}$) of 20 ns is considered for the SP performance model.

Particular considerations have to be made concerning the CC working frequency, since programmable logic implementations are not easily
predictable with respect to their switching characteristics.

### 6.2.2 Programmable Logic Performance

The complexity of the CC functionalities resulting from the virtual model requires a high-density programmable logic device (PLD) with a capacity in the range of up to 100,000 logic gates. State of the art PLD's, which are mainly divided into the two main categories of Field Programmable Gate Arrays (FPGA) and Complex Programmable Logic Devices (CPLD) do provide this level of complexity with the currently used 0.35 $\mu$m and 0.25 $\mu$m technologies.

Maximum performance of the implemented CC design, in terms of circuit speed and throughput, is determined by the maximum achievable frequency of the internal clock as well as by the propagation delay introduced by the external I/O pads. Unlike many other types of IC, the maximum system frequency is not a parameter that can be found in a data sheet. Estimating the achievable maximum performance of a design implemented on a PLD is a very challenging task. Data sheets do provide valuable information quantifying the performance of individual resources within the PLD. But overall design performance is also dependent on how the PLD's individual logic resources are interconnected and used to generate larger, more complex system functions and how those functions are integrated into the total design. Complicating matter further, published system-level benchmarks from PLD vendors are very often misleading, as they do not indicate the relative levels of effort or the tricks that are used to achieve a particular performance. For the designer it is therefore particularly difficult to find reference values to be used as decisional parameters, without implementing the design (down to the place and route) on the given device.

The only reliable starting point for working and I/O frequency estimation is represented by performance test based on speed metrics, which are published by the different PLD vendors in order to compare their products [Altera 97] [Xilinx 97].

Based on these kind of publications of representative PLD vendors and considering the kind of operations to be implemented by CC which have been exhaustively examined by means of the system virtual prototype, the following statement is made in order to quantify the influence of the CC on the HBS and VBS bandwidth: the maximal working frequency (I/O and internal clock rate) supported by the CC is 66 MHz.
6.2. Performance Models

This value is quite conservative with respect to the published peak performance data, but is considered as a representative and reliable for state of the art high density PLD’s. The value of 66 MHz and its corresponding time period of 16 ns ($T_{PLD}$) are used in the following performance model generation for the RDS architecture.

6.2.3 HBS Performance Model

For the determination of the HBS bandwidth and the associated value of $T_{GET}$, a model of the HBS is considered, consisting of the different communicating nodes involved. For each node the maximal communicating frequency is taken into account (Figure 6.4). The maximal HBS bandwidth ($HBS_{BW}$) is determined by the slowest node present in the data communication path.

In the BC evaluation, the memory module is considered as the HBS bottleneck. View that on the HBS, during the GET operation, a row stored in a single physical memory row is transferred, the maximal memory bandwidth exploitation is assumed. Thus the $T_{ACOL}$ value is taken for the computation (6.1). The SP HBS performance is influenced by all the nodes on the data-path (6.2).

\[
HBS_{BW_{BC}} = \frac{1}{T_{ACOL}} \quad (6.1)
\]

\[
HBS_{BW_{SP}} = \frac{1}{\max(T_{ACOL}, T_{PLD}, T_{SP\_IO})} \quad (6.2)
\]

Relevant for the generation of the system model are the $T_{GET\_BC}$ and $T_{GET\_SP}$ values which are required for the computation of the length of the phase A, B and C cycle times.

\[
T_{GET\_BC} = DATASIZE \cdot T_{ACOL} \quad (6.3)
\]

\[
T_{GET\_SP} = DATASIZE \cdot \max(T_{ACOL}, T_{PLD}, T_{SP\_IO}) \quad (6.4)
\]
The \( T_{\text{GET}_XY} \) values of (6.3) and (6.4) are computed basing on the considerations already made for the HBS bandwidth concerning the limiting node on the data-path. The resulting clock period is multiplied by the length of the data burst (\( DATA_{\text{SIZE}} \)) transmitted over HBS during the GET operation.

6.2.4 VBS Performance Model

The same approach used for the HBS is applied to the VBS model. Every node in the subsystem is considered with respect to its maximum communication clock rate.

In the BC modeling for VBS also, the memory is assumed to be the determining component. The mean memory access time \( \tilde{T}_{AC} \) resulting from the transposition process (Section 4.1), is used as determining value for the VBS bandwidth (6.5).

When considering the limitations imposed to the VBS throughput by the other communication nodes, it has to be noted that the influence of the speed of SPITFFIRE I/O port is not relevant since, according to the transposition mechanism of VBS (Fig 4.5), a CC has enough time to fetch data from the local cache, while the other CC’s are writing on the bus. Thus only the PLD related I/O frequency and the memory bandwidths play a role (6.6).

\[
VBS_{\text{BW}_{\text{BC}}} = \frac{1}{T_{AC}} \cdot VBS_{\text{PAR}}
\]  

(6.5)
6.2. Performance Models

\[ VBS_{BW\_SP} = \frac{1}{\max(T_{AC}, T_{PLD})} \cdot VBS_{PAR} \quad (6.6) \]

In both (6.5) and (6.6) the elementary bandwidth obtained with the working frequencies has to be multiplied by the number of data-path of VBS (\(VBS_{PAR}\)) according to VBS parallelization concept.

\[ T_{PUT\_BC} = \frac{T_{AC}}{VBS_{PAR}} \cdot DATASIZE \cdot P \quad (6.7) \]

\[ T_{PUT\_SP} = \frac{\max(T_{AC}, T_{PLD})}{VBS_{PAR}} \cdot DATASIZE \cdot P \quad (6.8) \]

The duration of the PUT process (6.7) (6.8) is computed assuming the same amount of data transferred as in the HBS case. The total time has to be multiplied by the number of PE’s present and divided by the parallelization grade of VBS.

6.2.5 FDE Performance Model

After modeling the low level communication processes determining the duration of the single FDE tasks, the role played by their scheduling is expressed. In the BC situation, the full parallel process scheduling (Fig. 3.9) is used to determine the duration of the 3 processing phases.

\[ T_{PHA\_BC} = N_{PHA} \cdot \max(T_{GET}, T_{PUT}, T_{FFT} + T_{SEP}) \quad (6.9) \]

\[ T_{PHB\_BC} = N_{PHB} \cdot \max(2 \cdot T_{GET}, T_{PUT}, 2 \cdot T_{FFT} + T_{CMUL}) \quad (6.10) \]

\[ T_{PHC\_BC} = N_{PHC} \cdot \max(T_{GET}, T_{PUT}, T_{FFT} + T_{SEP}) \quad (6.11) \]

Concerning the case where the SPITFFIRE I/O is considered, the corresponding tasks scheduling is used (Fig. 5.3).

\[ T_{PHA\_SP} = N_{PHA} \cdot \max(T_{GET} + T_{PUT}, T_{FFT} + T_{SEP}) \quad (6.12) \]

\[ T_{PHB\_SP} = N_{PHB} \cdot \left( \max(T_{GET} + T_{PUT}, T_{FFT}) + T_{CMUL} + \max(T_{GET}, T_{FFT}) \right) \quad (6.13) \]

\[ T_{PHC\_SP} = N_{PHC} \cdot \max(T_{GET}, T_{PUT}, T_{FFT} + T_{SEP}) \quad (6.14) \]
In both cases the duration of a single phase \( (T_{PHx-y}) \) is computed by multiplying the length of the phase cycle (determined by the task scheduling) by the number of repetitions required by the RV FFT implementation. Thus for \( N_{PHA} \), \( N_{PHB} \) and \( N_{PHC} \) the values defined by (3.6), (3.7) and (3.8) are used in the model.

The time required for the computation of FB-CCF procedure is given by the sum of the durations relative to a single phase.

\[
\begin{align*}
T_{FD_{BC}} &= T_{PHA_{BC}} + T_{PHB_{BC}} + T_{PHC_{BC}} \\
T_{FD_{SP}} &= T_{PHA_{SP}} + T_{PHB_{SP}} + T_{PHC_{SP}}
\end{align*}
\] (6.15) (6.16)

### 6.2.6 GPE Performance Model

The duration of computation of the MEP (3.10) on the GPE is determined by the number of clock cycles required for every pixel, the clock frequency of the DSP and by the parallelization grade of the GPE subsystem (6.17).

According to Section 3.5.1 the number of clock cycles per pixel \( (CPP) \) required for the normalization is 10 while for \( T_{SHARC} \) the value of 25 ns is used (40 MHz). Moreover linear system speedup for \( SH_{NUM} = 1 \ldots 4 \) is assumed.

\[
T_{NR} = \frac{(N - T)^2 \cdot T_{SHARC} \cdot CPP}{SH_{NUM}}
\] (6.17)

In the performance model a reduced region of interest of the normalization procedure is used, according to the reduced search area approach (Tab. 3.1). For an image size \( N \) of 1024 pixel a template size \( T \) of 128 is used in the model. The same ratio is used by \( N = 512 \) \((T = 64)\) and \( N = 256 \) \((T = 32)\).

The influence of the controlling tasks carried out by one of the SHARC's (SMC) are not considered in the model, since the amount of code to be executed is very small and the SHARC can very efficiently manage context switching and interrupt service [ADI 97].

### 6.2.7 MPIC System Performance Model

At this point the system level task scheduling (Fig. 3.16) can be mathematically expressed, after determining the duration of the image transfer
between the FDE and GPE. For this purpose the used memory technology is assumed to be the limiting factor. Thus, since the image transfer permits to the building of very long data transfer rows, a best case memory bandwidth (determined by its $T_{ACOL}$) is used for the computation of the frame input ($T_{IF}$) and output ($T_{OF}$) times.

$$T_{IF} = T_{OF} = N^2 \cdot T_{ACOL}$$ \hspace{1cm} (6.18)

According to the system resource scheduling of Figure 3.16, the duration of a whole NCCF computation and the corresponding system frame rate can be determined as following:

$$T_{FRAME} = T_{IF} + 2 \cdot T_{OF} + \max(T_{NR}, 2 \cdot T_{FD})$$ \hspace{1cm} (6.19)

$$T_{BST} = 2 \cdot T_{FRAME}$$ \hspace{1cm} (6.20)

$$FRAME RATE = \frac{1}{T_{FRAME}}$$ \hspace{1cm} (6.21)

At this point all the system parameters have been considered into the performance model, so that an exploration of the achievable frame rates can be undertaken.
Chapter 7

System Performance

The performance model previously conceived is evaluated by varying input system sizes such as $N$, $P$, $\text{VBSP}_{\text{AR}}$, $\text{SHNUM}$ and memory type. Representative performance charts are presented and confronted. As well as the BST performance the achievable 2D FFT frame rate is investigated. In the second part of the chapter comparisons with existing realizations are provided.

7.1 Performance Evaluation

Basically the system performance is defined by the setting of 5 system parameters: the image size $N$, the number of FDE PE’s $P$, the number of SHARC’s $\text{SHNUM}$, the parallelization grade of VBS $\text{VBSP}_{\text{AR}}$ and the memory technology. These degrees of freedom directly determine the different time values which are used for the model evaluation. The

<table>
<thead>
<tr>
<th>Parameter</th>
<th>BC Model</th>
<th>SP Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase Scheduling</td>
<td>full parallel</td>
<td>half sequential</td>
</tr>
<tr>
<td>PE Cache Usage</td>
<td>free</td>
<td>conditioned by CPU</td>
</tr>
<tr>
<td>PE I/O Frequency</td>
<td>not relevant</td>
<td>50 MHz</td>
</tr>
<tr>
<td>$R$ Factor</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$\text{DATASIZE}$</td>
<td>2048</td>
<td>2048</td>
</tr>
</tbody>
</table>

**Table 7.1:** System parameters used for the BC and SP models
model evaluation output space is doubled by considering a best case (BC) and a SPITFFIRE based (SP) performance evaluation. The first one (BC) has a theoretical significance and gives the upper end of the currently achievable frame rates with a FDE CPU respecting the ideal

<table>
<thead>
<tr>
<th>N</th>
<th>P</th>
<th>SH</th>
<th>VBS</th>
<th>Mem. Type</th>
<th>Frame Rate [Frames/s]</th>
<th>$T_{FRAME}$ [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SP</td>
<td>BC</td>
</tr>
<tr>
<td>1024</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>3.08</td>
<td>3.61</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>4.12</td>
<td>5.21</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>4.92</td>
<td>5.59</td>
</tr>
<tr>
<td>1024</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>4.04</td>
<td>4.07</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>6.04</td>
<td>6.78</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>6.78</td>
<td>6.78</td>
</tr>
<tr>
<td>1024</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>4.35</td>
<td>4.35</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>6.95</td>
<td>7.59</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>7.59</td>
<td>7.59</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>12.05</td>
<td>14.13</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>15.86</td>
<td>20.16</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>18.46</td>
<td>22.35</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>15.79</td>
<td>15.90</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>23.22</td>
<td>26.11</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>27.11</td>
<td>27.11</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>16.96</td>
<td>16.96</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>26.60</td>
<td>29.09</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>30.35</td>
<td>30.35</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>44.40</td>
<td>52.28</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>55.31</td>
<td>71.48</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>EDO</td>
<td>59.06</td>
<td>74.48</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>57.85</td>
<td>58.26</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>80.24</td>
<td>90.78</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>ED</td>
<td>89.51</td>
<td>108.45</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>61.80</td>
<td>61.80</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>90.75</td>
<td>99.67</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>SD</td>
<td>99.37</td>
<td>121.39</td>
</tr>
</tbody>
</table>

Table 7.2: Relevant System Performance Data: System with 2 SHARC's and a parallelization grade of VBS of 2.
Figure 7.1: System Performance by 1024x1024 and 256x256 pixel images with $VBS_{PAR} = 2$ and $SHARC_{NUM} = 2$: upper and lower part of Tab. 7.2.
requirements discussed in Section 3.3.4. The second one is representative of currently implementable solutions considering the I/O and cache usage limitations related to the SPITFFIRE (Sec. 5.2). Table 7.1 summarizes the parameters used in the BC and SP case.

<table>
<thead>
<tr>
<th>N</th>
<th>P</th>
<th>SH</th>
<th>VBS</th>
<th>Mem. Type</th>
<th>Frame Rate [Frames/s]</th>
<th>$T_{\text{FRAME}}$ [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SP BC</td>
<td>SP BC</td>
</tr>
<tr>
<td>1024</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>3.56 3.61</td>
<td>280.90 277.01</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>5.12 5.59</td>
<td>195.31 178.89</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>6.31 7.47</td>
<td>158.48 133.87</td>
</tr>
<tr>
<td>1024</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>4.07 4.07</td>
<td>245.70 245.70</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>6.78 6.78</td>
<td>147.49 147.49</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>9.51 10.17</td>
<td>105.15 98.33</td>
</tr>
<tr>
<td>1024</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>4.35 4.35</td>
<td>229.89 229.89</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>7.59 7.59</td>
<td>131.75 131.75</td>
</tr>
<tr>
<td>1024</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>11.27 12.11</td>
<td>88.73 82.58</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>13.96 14.13</td>
<td>71.63 70.77</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>19.79 21.66</td>
<td>50.53 46.17</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>23.90 28.57</td>
<td>41.84 35.00</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>15.90 15.90</td>
<td>62.89 62.89</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>26.11 26.11</td>
<td>38.30 38.30</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>35.81 38.45</td>
<td>27.93 26.01</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>16.96 16.96</td>
<td>58.96 58.96</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>29.09 29.09</td>
<td>34.38 34.38</td>
</tr>
<tr>
<td>512</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>42.00 45.29</td>
<td>23.81 22.08</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>51.62 52.28</td>
<td>19.37 19.13</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>69.88 77.03</td>
<td>14.31 12.98</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>EDO</td>
<td>78.89 97.25</td>
<td>12.68 10.28</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>58.26 58.26</td>
<td>17.16 17.16</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>90.78 90.78</td>
<td>11.02 11.02</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>ED</td>
<td>116.03 125.91</td>
<td>8.62 7.94</td>
</tr>
<tr>
<td>256</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>61.80 61.80</td>
<td>16.18 16.18</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>99.67 99.67</td>
<td>10.03 10.03</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>SD</td>
<td>132.06 143.69</td>
<td>7.57 6.96</td>
</tr>
</tbody>
</table>

Table 7.3: Relevant System Performance Data: System with 4 SHARC’s and a parallelization grade of VBS of 4.
Figure 7.2: System Performance by 1024x1024 and 256x256 pixel images \( VBS_{PAR} = 4 \) and \( SHARC_{NUM} = 4 \): upper and lower part of Tab. 7.3.
In both cases an $R$ factor of 4 is used in (4.2) which is the valued experienced in the FDE virtual prototype as a consequence of the 72 bit data format, 36 bit memory devices and effects of the RV-FFT. The data burst size of 2048 is given by the organization of the SPITFFIRE cache.

Being determined by 5 parameters, it is difficult to graphically represent the whole space built by the available performances in a single chart. After numerically presenting an overview of some system configurations which are considered representative (Tables 7.2 and 7.3), the impact of the single system parameters on the frame rate is graphically drawn (Figures 7.1 and 7.2).

The performance data for a system configuration with a 2 SIIARC’s GPE and a VBS with 2 data paths are resumed in Table 7.2 and Figure 7.1. The same is done for a system with a 4 SHARC’s GPE and a VBS with 4 data buses in Table 7.3 and Figure 7.2.

A qualitative inspection of the performance charts leads to the following considerations:

**FDE/GPE Limitation:** the charts are basically divided into two domains: one determined by the FDE performance and the other by the GPE performance. Whereas by lower values of $P$, the performance scales with the number of FDE PE’s, in the upper region a performance limitation is introduced by the GPE in some systems configurations. This effect is more important with fast memory technologies which improve the FDE CPU usage, making the FDE performance scale more rapidly.

In the 4 SHARC system, the limit imposed by the GPE is much higher, so that the curve saturation is reached by a higher frame rate (if compared to the 2 SHARC system) and only with the fastest memories. The image size is another factor influencing the limit imposed by the GPE performance, since the computational requirements of normalization and MEP depend on the squared input image size (6.17).

**Speedup:** linear system speedup is neared only in the lowest region of the $P$ range and with fast memory types. According to the scalability considerations (Section 4.3.1) the VBS throughput dependency on $P$ shows its effects, introducing wait states in the FDE CPU scheduling (see also Fig. 7.3).
7.1. Performance Evaluation

**VBS Parallelization:** The effects of a better parallelized VBS appear in the performance charts in form of an improved scalability which maintains linear character by a larger number of PE's. The benefits of an advanced VBS parallelization ($VBS_{PAR} > 2$) are better exploited in small image computation, since with the larger ones the GPE limitations intervene, vanishing the invested HW overhead on the FDE.

**BC/SP** The limitations imposed by the SPITFFIRE I/O capabilities have a very low influence on the frame rate in the lower $P$ region and by fast DRAM technologies. With fast memories the relevant drawback of the SP case consists mainly of the CMUL mode blocking the ALU operation. On the contrary the EDORAM case shows slightly different curve because the effect of the larger access time leading to longer $T_{GET}$ and $T_{PUT}$ is amplified by the sequential execution of the I/O tasks in the SP case. The same effect also appears with a larger $P$, where longer $T_{PUT}$ times have a stronger negative influence on the SP case in comparison to the BC case.

A representative measure of the system scalability is depicted in Figure 7.3, where the load of the FDE CPU's is represented as a function of the number of processing elements. Linear speedup of the FB-CCF procedure is realized only when every CPU is 100% used. In the two charts of Figure 7.3 a comparison between a 2 VBS and a 4 VBS system can be made. The effect of the VBS parallelization appears in the fact that the CPU load curves leave the 100% value by a higher number of PE.

It is interesting to observe the combined effect of memory type and SP/BC situations. Three configurations have the same behavior in the middle of the charts: the one with slow memory (EDO) but fast SPITFFIRE interface and two other with fast memory (SD and ED) but the not ideal SPITFFIRE case.

In Figure 7.4 the influence on the system frame rate of the DAR used in the VBS transposition process is depicted. The performance measurement is made with a SPITFFIRE based system with $VBS_{PAR} = 2$ and 4 SHARC's: the memory technology is also varied. The effect of the optimized VBS DAR, which have already been discussed in relation with the memory bandwidth (Fig. 4.3 and 4.4), also appear at system frame rate level.
Figure 7.3: FDE CPU Usage by $VBS_{PAR} = 2$ and $VBS_{PAR} = 4$ and an image size of $N = 1024$
7.1. Performance Evaluation

7.1.1 2D FFT Performance

Table 7.4 shows the performance achieved by the FDE (inclusive image I/O) by the computation of the FB-CCF procedure (Fig. 3.5). Near to the FB-CCF performance, the data regarding a standard real-valued 2D FFT are listed in Table 7.4.

The data of Table 7.4 is extremely relevant since it permits to better compare the performance of the MPIC systems with other solutions, since the 2D FFT is an often used benchmark. Furthermore the column with the FB-CCF data can be used to predict the usability of the MPIC system as a general image filtering engine. In fact by acting on the template content and using it as filter-pattern, any filtering function can be realized with a modified FB-CCF.
7.1.2 Frame Rate Driven System Configuration

After observing the influence of the 5 system parameters on the achievable frame rate, a question to be faced is how the system configuration can be determined as a function of the required frame rate.

Due to the number of parameters required to configure the MPIC architecture it is impossible to express a mathematical function computing the inverse function of the MPIC performance model. Thus if a given frame rate has to be obtained, a possible system configuration can be extracted by the performance charts of Section 7.1. It is evident that more than one configuration can lead to a given frame rate. Thus
the choice of the best configuration has to be made by means of further decisional parameters such as the flexibility required by the system. In fact if exactly only one frame rate is required, then a minimal system configuration can be chosen. This could concern for example the $VBS_{PAR}$ value: a single VBS can achieve a certain range of frame rates but offers lower system scalability opportunities. In the case where final application requires several frame rates to be supported (by different image sizes), a more flexible configuration has to be preferred.

![Figure 7.5: Qualitative influence of the system parameters on the performance determination.](image)

If a required frame rate is above the delivered performance curves, following consideration can be made in order to determine which system parameter has to be modified (Fig. 7.5):

- the slope of the first part of the curve (where linear speedup is effective) is determined by the speed of the FDE CPU.

- if linear speedup begins to decrease slightly, then the communication bandwidth is slowing the FDE. VBS parallelization has to be increased or faster memory technology has to be used.

- if linear speedup is abruptly stopped, then the GPE is the limiting factor.

Further relevant input for the evaluation of the optimal MPIC configuration is brought by cost-performance charts which are provided in
conjunction with the discussion of higher system integration options (Sec. 8.3).

7.2 System Performance Comparison

Because of the high specificity of the MPIC system, it is difficult to find representative solutions to be compared with. This is because of the fact that the different solutions found in the literature do not support the correlation algorithms with the same input image accuracy and resolution, as well as with the same 32 bit computation accuracy achieved by the presented architecture. Other works merely present theoretical discussions of the parallel implementation [Siegel 82] [Armstrong 94], without entering into the implementation aspects.

7.2.1 BST and NCCF

The difficulty of confrontation with the existing solutions [Rothacher 94] [Peisl 88] is particularly high for the NCCF algorithm. This is because of the fact that in most cases only the unnormalized cross correlation function is supported and additionally the search for the maximum correlation peaks is not implemented. In addition to this the standard NCCF algorithm has been improved with the BST which further reduces the comparability of the MPIC system, because of the different computational requirements involved.

In [Peisl 88] a dedicated bit array processor is presented which is limited to the unnormalized cross correlation function implementation. The processor implemented in 2µs CMOS achieves 25 frames/s with 144 × 80 pixel images and 32 × 32 pixel templates. A more recent solution is presented in [Rothacher 94]. Applying data reduction techniques to the original grey level image, a dedicated processor has been implemented in VLSI, achieving the implementation of the normalized cross correlation with adjusted mean values: in this type of correlation function the mean value of the image is subtracted from each pixel value before the NCCF computation [Aschwanden 93]. The realized frame rate is 72 frames/s (13.9 ms) with a search area size of 64 × 64 pixels and a template of 32 × 32 pixels: the word width of the input data is 4 bits. The maximum extraction takes further 2ms giving a total frame period of 16ms.
In both cases relevant limitations are imposed on the size and accuracy of the input data, in order to guarantee frame rates usable in real time applications.

The simulation of the BST algorithm on state of the art workstations takes times in the range between 3.5 to 4.5 minutes (1024x1024 images). These values are not relevant for comparison since the algorithm is specified in high level C language. Moreover, being the BST algorithm modeled within the Khoros environment, a considerable overhead is involved in the Khoros-specific data handling. Nevertheless this value gives a tangible measure of the involved complexity.

A better comparison can be made with an implementation of the same algorithm on a SHARC DSP. According to the computational requirements of the FB-BST (Tab. 2.3) and assuming that 50% of the SUM operations can be computed in parallel with the MULT operation exploiting the MAC operation the following comparison can be made. In order to make the comparison more valuable, the MEP complexity is also included, since it has already been analyzed for the SHARC as GPE processor. The comparison of Table 7.5 gives an idea of the speedup factor achieved by the dedicated HW implementation. The performance data achieved by the SHARC implementation is a very optimistic expectation, which assume a 100% load of the floating point CPU, without any wait state imposed by data communication. Considering the size of the involved temporary data and the low computation to communication ratio of the FFT kernels, it has to be assumes that not all the data exchange to the main memory can be sustained with DMA operation. Thus a sub-optimal CPU exploitation has to be envisaged which leads to an even lower frame rate.

<table>
<thead>
<tr>
<th>Platform</th>
<th>256 × 256</th>
<th>512 × 512</th>
<th>1024 × 1024</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARC @ 40 MHz</td>
<td>620</td>
<td>2800</td>
<td>12300</td>
<td>ms</td>
</tr>
<tr>
<td>FDE-4/GPE-2</td>
<td>10</td>
<td>34.3</td>
<td>131.75</td>
<td>ms</td>
</tr>
<tr>
<td>Speedup</td>
<td>62</td>
<td>81</td>
<td>94</td>
<td></td>
</tr>
</tbody>
</table>

**Table 7.5:** *BST execution time comparison between single SHARC DSP and a 4FDE/2GPE MPIC with SDRAM memory and 4 VBS.*
7.2.2 2D FFT

With the 2D FFT algorithm, the performance confrontation can be made with more representative solutions [Sgro 98] [Hartley 94], since this procedure receives more attention due to its wide range of applications in 2D signal processing. Furthermore the 1D FFT computation is quite often presented as a benchmark for several off-the-shelf processor classes [BDTI 97] and can be used to extract 2D computing time following the indirect approach for 2D FFT.

In [Hartley 94] and [Sgro 98] a concept and resp. a commercial implementation of a multiprocessors system based on state of the art floating point DSP's are presented. In the first case the 50 MHz Texas Instrument TMS320C40 is used which is particularly appreciated because of the advanced inter-processor communication capabilities. The second case bases on the SHARC processor already presented in this work. In both cases a dedicated network is implemented in order to exploit the peculiar processor communication capabilities and realize a message-passing channel which operates in parallel to a standard shared memory SIMD approach.

<table>
<thead>
<tr>
<th>Platform</th>
<th>1024 × 1024</th>
<th>256 × 256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 PE [ms]</td>
<td>4 PE [ms]</td>
</tr>
<tr>
<td>Par. TI C40</td>
<td>3164</td>
<td>1169</td>
</tr>
<tr>
<td>Par. SHARC</td>
<td>1045</td>
<td>272</td>
</tr>
<tr>
<td>FDE SP-EDRAM</td>
<td>90.7</td>
<td>23.0</td>
</tr>
<tr>
<td>FDE BC-SDRAM</td>
<td>90.7</td>
<td>23.0</td>
</tr>
<tr>
<td>Pentium 266 MHz</td>
<td>616</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.6: 2D FFT performance comparison: the FDE has 4 VBS.

Table 7.6 resumes the frame rates achieved by the two solutions proposed in [Hartley 94] [Sgro 98], compared with two different configurations of the FDE. The data relative to the FDE listed in Table 7.6 do not comprehend the image I/O time (which on the contrary is considered in Tab. 7.4) in order to have a more representative comparison. A further comparison is made with respect to a full SW implementation of the 2D FFT on a general purpose micro processor. Based on [BDTI 97] where a performance comparison for 1D FFT implementation on a In-
7.2. System Performance Comparison

tel Pentium processor is given, the performance of this μP for the 2D procedure can be estimated.
The speedup brought by the FDE achieves the highest values in the range of 40 with respect to the TMS320C40 and 12 with respect to the multi SHARC architecture. The improvement factor is reduced by a smaller image size. Furthermore the speedup is maintained with the higher processor number only if the high end system consideration with a BC SPITFFIRE and SDRAM memory is considered.
Chapter 8

FDE Integration and Cost Evaluation

Two different approaches for higher system integration are investigated. Firstly a multi-chip module implementation of the FDE PE is considered and a series of possible realizations with different substrates is presented. In the second part of the chapter a full ASIC integration of the same functional blocks is discussed. Based on the evaluation of the multi-chip-module integration a cost-model of the MPIC is generated which permits to determine the relationship between cost and performance.

The evaluation of higher integration approaches for the MPIC is limited to the FDE as a consequence of different aspects. First of all the FDE parallelization grade is considerably higher if compared with the parallelization range 1...4 present in the GPE. Therefore a highly integrated solution can bring relevant benefits to the FDE in terms of a higher number of PE's present in the system as well as lower inter-PE signal degradation leading to higher system clock frequency. Additionally the reusability of the design specification carried out for the SPITFFIRE and the CC permits the facing of the integration from an advanced starting point and the disposal of an efficient testing infrastructure.

On the other hand the integration of the GPE is of a lesser interest because of the reduced number of chips and the simple routing. Furthermore, in order to guarantee system flexibility with respect to the final
application and to permit an easy upgrade to leading DSP evolutions, a PCB fabrication offers a better solution.

8.1 MCM FDE Realization

The first high density integration which is evaluated for the FDE PE is a *multi-chip module* (MCM) fabrication. MCM packaging is based on the direct connection of Bare Die (BD) chips on a multi-layer substrate which realizes the module level connectivity. For first-level interconnect (from IC to substrate) wire-bond, tape automated bonding (TAB) and Flip Chip (FC) are the available alternatives. Usable substrates are MCM-L (Laminate), MCM-D (Deposited) and MCM-C (Ceramic). The

<table>
<thead>
<tr>
<th>Substrate Type</th>
<th>$L_{width}$ [µm]</th>
<th>$P_{w eff}$ [µm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB</td>
<td>125</td>
<td>510</td>
</tr>
<tr>
<td>MCM-L</td>
<td>75</td>
<td>260</td>
</tr>
<tr>
<td>MCM-C</td>
<td>125</td>
<td>290</td>
</tr>
<tr>
<td>MCM-D</td>
<td>20</td>
<td>65</td>
</tr>
</tbody>
</table>

*Table 8.1: Generic substrate parameters*

reduction of the system dimensions given by the MCM is the effect of the package-less IC assembling and of the smaller signal lines ($L_{width}$) provided by the different substrates. However the effective wiring pitch ($P_{w eff}$) of a given substrate also depends on the via-pitch influence [Hirt 99]. The effective wiring pitches of relevant substrates are listed in Table 8.1.

8.1.1 System Partitioning

The objective of the partitioning is the determination of a base-module in the FDE architecture which is assigned to the MCM implementation. In evaluating different partitioning, the reusability of the module is of the highest interest. For the FDE architecture, the module has to permit the construction of a parallel system built of P PE's. These considerations leads to the FDE Module (FDEM) of Figure 8.1.
Some further explanations of the chosen partitioning are required:

**DRAM:** the two memory banks are not placed on the FDEM for the following reasons: state of the art 32/36 bit DRAM's are mostly available as SIMM or DIMM modules which already provide a high grade of compactness. Furthermore these modules are connected to the substrate through standard socket providing flexible memory configuration (size and type).

Another relevant consideration concerns yield. A 2Mx36 DRAM SIMM usually consist of 36 (or 9) 1Mbit (or 4 Mbit) chips. Since BD are usually delivered untested by the IC manufacturers (because of the absence of the package), the placement of up to 72 memory components on the FDEM would dramatically reduce successful production.

**VBS:** in order to support a wide range of system configurations the FDEM is provided with a VBS which can support up to 4 data paths \((VBS_{PAR} = 4)\). Thus in addition to the 50 signals of a single data-path VBS, three more 36 bit data paths are available on the MCM.

**SEL Box:** the FDEM must provide the highest possible reusability. In conjunction with its use in the FDE and in relation with the VBS parallelization issues of Section 4.3.2 the insertion of a SEL box on the VBS is required. By means of this element it is then possible to realize the all-to-several routing characteristic of a parallel VBS (Fig. 4.8). By means of the SEL box, one of the 4 data paths of...
VBS is connected to the XBAR at system configuration, according to the position occupied by the PE in the FDE.

### 8.1.2 MCM early dimensions estimation

According to the method proposed in [Hirt 97], it is possible to make an early estimation of the size of the FDEM, as a function of the used substrate and the number of used routing layers. In this discussion only the I/O signals of the IC’s are considered while it is assumed that the power pins are connected through two dedicated layers in the substrate.

**Chip Area Estimation**

Firstly the size of the used IC dies has to be determined. In order to obtain the keep-out size, expressing the actual size occupied by the dies on the substrate, the space used for the first level interconnect (FLI) is considered. For the FDEM wire bonding is considered, being the most mature and widely used option. The number of chips for the XBAR and SEL is determined by the number of pins available on the used IC’s with respect to the number of signals to be switched.

According to [Hirt 97] in the third column of Tab. 8.2 the bare die size is augmented by the space required for the wire bonding. For a single row bonding 1 mm is used while for the two rows bonding (required by the CC FPGA due to its high pin-count) 1.5 mm per side are added.

A total area of \( A_{CHIPS} = 750.3\, mm^2 \) is required by the bare dies on the FDEM.

<table>
<thead>
<tr>
<th>Function and IC</th>
<th>Die ((L \times W)) [mm]</th>
<th>FLI [mm]</th>
<th>Keepout ((L \times W)) [mm]</th>
<th>IC Numb.</th>
<th>Area ([mm^2])</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPITFFIRE</td>
<td>11.6x14.4</td>
<td>1</td>
<td>13.6x16.4</td>
<td>1</td>
<td>268.96</td>
</tr>
<tr>
<td>CC (XC4062XL)</td>
<td>15x15</td>
<td>1.5</td>
<td>18x18</td>
<td>1</td>
<td>324</td>
</tr>
<tr>
<td>SEL (QS32x245)</td>
<td>1.1x1.1</td>
<td>1</td>
<td>3.1x3.1</td>
<td>10</td>
<td>96.1</td>
</tr>
<tr>
<td>XBAR (QS32x383)</td>
<td>1.5x1.5</td>
<td>1</td>
<td>3.5x3.5</td>
<td>5</td>
<td>61.25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>Total</strong></td>
<td><strong>Total</strong></td>
<td><strong>Total</strong></td>
<td><strong>Total</strong></td>
<td><strong>750.31</strong></td>
</tr>
</tbody>
</table>

**Table 8.2:** FDEM IC’s: bare die size of the XC4062, the QS32x245 and the QS32x383 are estimated values.
Routing Area Estimation

An approximation of the conductor length required to interconnect electronic components based on empirical analysis can be computed as follows [Seraphim 77]:

\[ L_{ROUT} = 1.5 \cdot P_D (1.5 \cdot \frac{N_T}{2}) \cdot \frac{1}{50\%} \]  

(8.1)

where:
- \( P_D \): average chip to chip distance
- \( N_T \): number of chip to chip connections

In (8.1) the number of module-internal connections \( N_T \) can easily be extracted from the FDEM architecture of Fig. 8.1: a value \( N_T = 752 \) is obtained with \( VBS_{PAR} = 4 \). For the determination of \( P_D \) a trial placement of the FDEM has to be carried out (Fig. 8.2). In addition to the keep-out zones from Table 8.2 a routing channel is reserved for the VBS routing. Assuming about 160 signals to be placed in this region, its size is computed by a multiplication of this value by the \( P_{WEFF} \) of Table 8.1. Following a rule of thumb, the routing space required for a 1-layer MCM-D substrate is reserved (10.4 mm). The same space is required by a 4-layers MCM-L substrate. By computing the average chip to chip distances (measured in the centers) considering the horizontal and vertical directions a value of \( P_D = 6.8 \) mm is obtained.

Other relevant sizes extracted from Figure 8.2 are the minimal FDEM width \( FDEMW_{\text{min}} = 18 \) mm, the minimal FDEM length

**Figure 8.2:** FDEM trial placement of the IC's from Table 8.2: a routing channel has been introduced for the high demanding parallel VBS. All dimensions are in mm.
\( FDEM_{L_{\text{min}}} = 51.7 \text{ mm} \) and a ratio between FDEM width and length which is 2.9.

The overall wiring \( A_W \) area can be estimated according to (8.2):

\[
A_W = \frac{L_{\text{ROUT}} \cdot P_{W\text{eff}}}{N_L}
\]  

(8.2)

**FDEM Area Estimation**

Assuming that the FDEM area is determined by the chip size \( A_{\text{CHIPS}} \) added to the wiring area \( A_W \), by means of (8.2) the fabrication of the FDEM using different substrates and layer-numbers can be explored. The result is depicted in Figure 8.3. From Figure 8.3 a series of potential

**Figure 8.3:** FDEM length with different substrates and number of signal layers. The width of the FDEM is obtained by dividing the length by the factor 2.9.

FDEM fabrications can be extracted. The interesting ones are listed in Table 8.3.
8.2 Full FDE PE integration

The evaluation of a full integration of the FDE processing element can be undertaken by the same starting point used for the MCM exploration: the partitioning of Figure 8.1.

The integration of the functional blocks already discussed for the MCM case is of little interest, since there aren't any particular problems which inhibit the production of a full-custom ASIC design of the current SPITFFIRE concept with the addition of the CC and XBAR functionality. In fact the complexity of the last two elements can be estimated as being lower than 100 KGates. In [Wosnitza 99] an estimation of the SPITFFIRE chip-size considering state of the art and future CMOS technologies is presented. Table 8.4 summarizes some relevant data. The chip size increment due to the integration of the

<table>
<thead>
<tr>
<th>CMOS Techn.</th>
<th>Layout</th>
<th>Chip Area [mm²]</th>
<th>Clock Freq. [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 µm</td>
<td>StdCell</td>
<td>167</td>
<td>66</td>
</tr>
<tr>
<td>0.5 µm</td>
<td>Full-Custom</td>
<td>95.8</td>
<td>100</td>
</tr>
<tr>
<td>0.25 µm</td>
<td>Full-Custom</td>
<td>21</td>
<td>161.2</td>
</tr>
<tr>
<td>0.18 µm</td>
<td>Full-Custom</td>
<td>11.0</td>
<td>215</td>
</tr>
</tbody>
</table>

Table 8.4: SPITFFIRE characteristics by advanced CMOS technologies

CC and XBAR should not alter the data of Table 8.4 in a relevant way. Also in the case of the routing issues of a parallel VBS, modern CMOS technology with more than 6 wiring layers [SIA 97] should manage the
required complexity without consistent problems.

8.2.1 System on a Chip

A much more notable problem is presented by a complete integration of the FDE PE element including the two banks of DRAM. In fact the integration on the same ASIC of high speed logic as well as DRAM memory is one of the most challenging issues that system-on-a-chip (SOC) designers are currently facing. In fact ASIC and DRAM CMOS integration has been historically characterized by two optimization approaches (clock-speed and cell-size) which have produced ASIC and DRAM specific production processes. Thus the placing of DRAM and CPU core on the same ASIC is usually related to a strong performance loss of the logic elements or a huge size increase of the memory elements.

Pushed by the growing demand of high-speed/high-bandwidth multimedia and video encoding/decoding applications, DRAM macros have been developed in order to be embedded into ASICs designs. Such DRAM are called embedded DRAM [Crowder 98] [Kimura 99]. The attraction of embedded DRAM is the on-chip bandwidth, achieving a typical first data access time of about 10 ns and a column access time of 6 ns. These numbers are obscured when the cost overhead (in terms of silicon area and fabrication) of the embedded DRAM is considered and compared to the data provided by high-volume DRAM chips.

In [Crowder 98] some precise information is provided in this context: an embedded DRAM is presented as a macro for an 0.18\(\mu\)m CMOS 6ML Copper process. It is asserted that the obtained embedded DRAM cell size is of a factor 1.5 larger than one of a similar DRAM stand-alone technology. Furthermore a 25% production cost overhead is introduced in the ASIC fabrication as a consequence of additional mask layers. Thus the ratio between logic and memory complexity of the envisaged SOC becomes the relevant factor to be considered in the evaluation of this integration approach. In fact conceiving a SOC where the embedded DRAM part largely exceeds the logic part, leads to an economically unattractive solution.

A precise evaluation can be made in this sense for the FDE PE. Firstly the minimal amount of DRAM to be placed with the SPITFIRE core on the SOC has to be determined. In Table 8.5 the minimal
8.2. Full FDE PE integration

DRAM space requirements for the two memory banks are listed (starting from the RMA memory requirements of Tab. 4.1). In Table 8.5 the required size for the two banks is confronted with the available DRAM organization: a minimal DRAM block-size of 1Mx36 is assumed which is quite realistic (but since memory organization usually scale with a factor of 2, a worst case scenario requiring 8Mx36 for bank B wouldn’t be abnormal).

The minimal memory requirements (Tab. 8.5) refer to the amount of memory which has globally to be present in the FDE for a 1024x1024 pixel mode. Since the main memory is partitioned into smaller banks placed on the PE’s, the minimal amount of DRAM to be placed on each PE’s is obtained by dividing the global requirements by $P$. Once again this minimal value has to be confronted with the available granularity of the embedded DRAM macros, but for simplicity purposes this aspect is not considered.

In Figure 8.4 the influence of $P$ on the integration costs of a SOC integration are compared to the ones of a multi-chip solution based on a specific logic/memory ASIC integration (applying the cost/size factor reported in [Crowder 98]). Assuming 0.18 μm CMOS a chip area of 11 mm$^2$ is used for the SPITFFIRE while for the estimation of the DRAM area a density of 115 Mbit/cm$^2$ is inferred from [Crowder 98].

Using these values it can be deduced that a full integration of the system memory requirements as embedded DRAM leads to a cost augmentation of a factor 1.8, which in this highly memory-dominated SOC practically results from the multiplication of the area and production overhead. More attractive solutions can be envisaged if less DRAM is placed on the SOC, making it competitive from the point of view of the costs involved in a multi-chip integration. Advantages already appear from a DRAM size reduction of factor 2...4.

It has to be pointed out that the cost-advantage achieved through

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8.7 MByte</td>
<td>1.9M x 36</td>
<td>2M x 36</td>
<td>72 MBit</td>
</tr>
<tr>
<td>B</td>
<td>22.1 Mbytes</td>
<td>4.9M x 36</td>
<td>5M x 36</td>
<td>180 MBit</td>
</tr>
</tbody>
</table>

Table 8.5: Minimal DRAM space on the FDE for the 1024x1024 pixel mode.
the reduction of the DRAM memory space is strongly contradictory to the integration principles. This is because the reduced-memory SOC pays a high tribute in terms of flexibility and system-scalability by not providing enough memory space for a single-SOC FDE. In fact a minimal number of reduced-memory SOC's have to be present in the system in order to allow 1024x1024 operation.

In conclusion it is quite difficult to decide whether or not the SOC solution is the best full integration approach for the FDE, even though a feasible compromise between cost and flexibility seems to exist for DRAM partitioning in the range 2...4. Furthermore this uncertainty is confirmed by the current grade of acceptance embedded DRAM has in the semiconductor industry where in some cases DRAM is placed on the chip [Herrmann 98] while some others with high demanding multimedia designs still prefer the cost and flexibility characteristics of high volume off-chip DRAM [Kuntaragi 99].

**Figure 8.4:** Cost comparison between SOC and multi-chip integration of the FDE PE. Data based on the IBM 0.18 μm CMOS process presented in [Crowder 98]
8.2.2 Multi-Packed FDE Module

A feasible advanced integration of the FDE PE can be envisaged with a multi-packed module where the SPITFFIRE, XBAR and CC are integrated onto the same ASIC and two banks of DRAM are integrated separately using a specific process. Figure 8.5 shows the concept of such a solution.

![Diagram of multi-packed FDE PE with DRAM](image)

**Figure 8.5:** *FDE PE with DRAM integrated as multi-packed module.*

The DRAM chips are placed within the same package where the ASIC also resides. The inter-chip connections are realized by directly wire-bonding the corresponding chip-pads. Further ASIC area optimization can be realized by the use of smaller pads for the chip-to-chip connections, which do not have to drive the package-pins and the external lines. This optimization could theoretically be carried out also on the DRAM side, assuming that these chips are specifically developed.

This solution offers the advantage of offering a high grade of flexibility for the choice of the DRAM bare dies, which can take advantage of the highly volatile and evolving memory market. Further variations can occur with respect to memory size, organization, speed and interface. Lastly the production yield is dramatically improved with respect to a single chip solution, due to the silicon area partitioning.
8.3 Costs vs. Performance Evaluation

The objective of building a MPIC cost model is not to make a precise estimation of its commercial price but much more to obtain a cost function which permits to obtain a normalized cost-performance chart. Thus a relatively simple cost modeling approach is used based on the estimation of three relevant parameters: IC cost, board/MCM cost and production yield (8.3). No engineering or manufacturing costs are considered.

\[
MPCI_{\text{COST}} = \frac{IC_{\text{COST}}(P, SHNUM) + A_{\text{COST}}(P, SHNUM, VBSPAR)}{\text{yield}}
\]  

(8.3)

For the cost calculation as a function of the MPIC complexity, two system units are assumed (Fig. 8.6):

- an FDE unit (FDU) consisting of the processing part already considered for implementation as an MCM FDE module (Fig. 8.1) and two 4Mx36 DRAM banks. In the cost model a PCB as well as a 4-layers MCM-L and 2-layers MCM-D implementation of the FDU are included. Once again the DRAM parts are implemented on PCB only. The MCM dimensions obtained in Table 8.3 are used for the cost computation.

- a GPE unit (GPU) built by a SHARC DSP and 4Mx36 DRAM. This unit is realized on PCB only.

The \( P \) and \( SHNUM \) in (8.3) refer to the number of FDU’s and GPU’s present in the system.

For the estimation of the chip cost (\( IC_{\text{COST}} \)), representative prices have been obtained from different sources (Sales Points, WWW) assuming 10000 IC’s being purchased (Tab.8.6). The chip cost of a single unit is obtained by accumulating the IC costs according to the early placement of Figure 8.6.

A few words have to be spent here concerning the price of the EDRAM devices which according to Table 8.6 are nearly more expensive than SRAM’s. This kind of DRAM has been considered for the MPIC since its early developments, when SDRAM were being announced and other advanced DRAM’s evolutions were trying to gain market shares. In the meantime, SDRAM has become state of the
8.3. Costs vs. Performance Evaluation

![Diagram](image)

**Figure 8.6:** *MPIC partitioning for cost modeling.*

<table>
<thead>
<tr>
<th>Device</th>
<th>Price</th>
<th>I/O's</th>
<th>Pkg</th>
<th>IC Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPITFFIRE</td>
<td>75 USD</td>
<td>100</td>
<td>PGA180</td>
<td>16.5 cm²</td>
</tr>
<tr>
<td>XC 40062XL</td>
<td>180 USD</td>
<td>250</td>
<td>PQ304</td>
<td>17.6 cm²</td>
</tr>
<tr>
<td>XBAR</td>
<td>not. cons.</td>
<td>100</td>
<td>QVSOP48</td>
<td>0.5 cm²</td>
</tr>
<tr>
<td>SHARC DSP</td>
<td>265 USD</td>
<td>120</td>
<td>PQ240</td>
<td>12.25 cm²</td>
</tr>
<tr>
<td>EDOGRAM</td>
<td>1.2 USD/MB</td>
<td>50</td>
<td>SIMM</td>
<td>11 cm²</td>
</tr>
<tr>
<td>SDRAM</td>
<td>1.2 USD/MB</td>
<td>50</td>
<td>SIMM</td>
<td>11 cm²</td>
</tr>
<tr>
<td>EDRAM</td>
<td>22.6 USD/MB</td>
<td>50</td>
<td>SIMM</td>
<td>11 cm²</td>
</tr>
</tbody>
</table>

**Table 8.6:** *Price and complexity of the IC’s considered in the cost model. The SPITFFIRE price is estimated using cost/transistor values obtained from [SIA 97] for 0.25 μm CMOS.

...art main memory for PC’s and workstations. The result of the high volume production is that two similar technical solutions (in terms of chip design cost) for the memory bandwidth bottleneck have now a difference in price of a factor 20. As a consequence, although EDRAM achieves quite the same performance as SDRAM for the MPIC, it can't be considered as a usable alternative.

For the estimation of the PCB routing-cost the same method already applied for the MCM integration estimation is used in the cost model. Thus with (8.1) and (8.2) the PCB routing area in the GPU and FDU are obtained with an estimated PCB chip-pitch $P_{DPCB} = 40$ mm, assuming 6 wiring layers being used and a $P_{w,eff}$ taken from Table 8.7.

When the wiring area is known, the area cost of the GPU (8.4) and
<table>
<thead>
<tr>
<th>Substrate Type</th>
<th>( P_{\text{eff}} ) [( \mu \text{m} )]</th>
<th>Layers Number</th>
<th>Cost/Area [USD/cm(^2)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB</td>
<td>510</td>
<td>6</td>
<td>0.1</td>
</tr>
<tr>
<td>MCM-L</td>
<td>260</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>MCM-D</td>
<td>65</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 8.7: Effective pitch and cost per area of representative substrates used in the cost model

FDU (8.5) can be determined adding the area needed for placing of the IC’s (Tab. 8.6) and converting the total area into a cost with the cost per area values of Table 8.7. In the case of the FDU, if an MCM implementation is considered then the cost of the PCB considers the place for DRAM as well as the place for an MCM: the MCM cost is obtained by multiplying the MCM size by the substrate price (8.7).

\[
GP_{\text{ACOST}} = (GPU_{\text{PCB}_w} + GPU_{\text{IC}_a}) \text{PCBUSD/a} \quad (8.4)
\]

\[
FDU_{\text{ACOST}} = (FDU_{\text{PCB}_w} + GPU_{\text{IC}_a}) \text{PCBUSD/a} \quad (8.5)
\]

\[
FDU_{\text{MCM-ACOST}} = (DRAM_a + MCM_a) \text{PCBUSD/a} + (MCM_a) \text{MCMUSD/a} \quad (8.6)
\]

In (8.4), (8.5) and (8.7) the sub-indexes \( a \) and \( w \) refer to area and wiring area respectively.

According to the considerations made in the MCM evaluation where a routing channel has been introduced on the FDE PE (Fig. 8.2), the influence of the number of VBS data paths present in the system is considered in the PCB FDU cost model with the introduction of a VBS routing factor \( (VBS_{RF}) \).

\[
VBS_{RF} = 1.05 + (VBS_{PAR} - 1) \cdot 0.05 \quad (8.7)
\]

The \( VBS_{RF} \) is applied to the GPU area cost as well as to the PCB FDU area cost.

The unit cost of FDU and GPU are then determined by the sum of chip cost and area cost which are multiplied by the number of units presents in the MPIC system (\( P \) and \( SH_{NUM} \)).
The last value to be specified is the yield. In (8.8) the yield is obtained assuming a defect IC over 100 and a defect substrate-wiring over 100000. The number of chips and wires present in a single FDU and GPU are considered in the yield computation.

\[
\text{yield} = 0.99^{(FDU_{\text{chips}} + GPU_{\text{chips}})} \times 0.99999^{(FDU_{\text{wires}} + GPU_{\text{wires}})} \quad (8.8)
\]

On the contrary the \( P \) and \( SH_{NUM} \) values are not considered in the yield function assuming that the single units are tested before an MPIC is assembled.

![Figure 8.7: Output of the MPIC cost model in absolute values.](image)

In Figure 8.7 the estimated MPIC costs for the considered substrates as a function of different configuration is represented. The chart refers to a SDRAM based system and two configurations are concurrently drawn: a minimal \( (VBS_{PAR} = 1, SH_{NUM} = 1) \) and maximal \( (VBS_{PAR} = 4, SH_{NUM} = 4) \) situation. The lowest connected lines refer to the chips costs which then are incremented by the substrate dependent overhead. Apart from the fact that the different substrates do influence the system cost in a reduced manner, it can also be observed in the figure how the
GPE parallelization on the low $P$ range has a very high impact (factor 2) on the global costs. In the higher $P$ range, the relative cost of the GPE parallelization is lower (factor 1.2 for $P=32$).

Figure 8.8: MPIC: estimated area [mm$^2$] with PCB, MCM-D and MCM-L substrates.

While the substrate has a reduced impact on cost, a remarkable difference is noted in the area comparison of Figure 8.8. The PCB curve has a linear growth of about 100 cm$^2$/PE. By the two MCM implementations this value is about 38 cm$^2$/PE. Thus an area reduction factor of 2.5 between PCB and MCM realization arises from the model. The fact that the two MCM solutions do not provide different growth rates is due to the fact that for the DRAM and GPU a PCB substrate is used, which reduces the benefits of the miniaturization brought by advanced MCM-D.

The cost model can also be exploited to obtain normalized cost/performance curves. The MPIC costs are normalized by the cost of a minimal system consisting of an 1 FDU, 1 GPU SDRAM based system with a single VBS. Chip cost only is considered.
Figure 8.9: Normalized costs per frame rate by 1024x124 pixel images.

Figure 8.10: Normalized costs per frame rate by 256x256 pixel images.
By combining the output of the performance model of Section 6.2 with the one of the cost model, the charts of Figure 8.9 and 8.10 are created.

In the cost/performance charts interesting break-even points do appear, which help to decide which configuration is suited for a target frame rate. In conjunction with the considerations about system configuration of Section 7.1.2, system flexibility does appear again as an important decisional parameter in relations to costs.

If performance scalability is required, than a configuration has to be chosen that by the low $P$ number can be considerably more expensive than a minimal one (factor $2\ldots 4$). If on the contrary only a maximal system frame rate has to be achieved, then cost optimization can be considered.

The influence of the $SH_{NUM}$ and $VBS_{PAR}$ parameters on the system costs is quite different. Whereas the number of SHARC's has a strong influence on the system cost in particular with low PE numbers, the number of VBS data paths doesn't have relevant repercussions.
Chapter 9

Outlook and Conclusions

According to the envisaged evolution of semiconductor technology the scaling of the MPIC performance is investigated. The usability of the RMA for general video processing tasks is considered as well as an evaluation of other kinds of algorithms which can be efficiently ported on the MPIC. The chapter is then closed with some concluding remarks.

9.1 System Performance Scaling

9.1.1 Scaling of the Single Elements

In order to make a reliable prediction concerning the scalability of the MPIC performance as a consequence of the envisaged technological evolutions [SIA 97], the attention has to be firstly focused on the scalability of the single elements building the system.

**FDE PE:** according to the investigations carried out in [Wosnitza 99] the highly pipelined architecture of the SPITFFIRE chip is predicted to have high impact on the clock frequency by technology scaling. Thus the expected frequencies of Table 8.4 are envisaged: these values have a direct repercussion on the 1D FFT, CMUL, SEP and UNSEP execution time of Table 5.2.

The CC is considered as not determinant in the MPIC performance estimation, assuming on one side a full-integration of the communication infrastructure combined with the use of high density packaging solutions.
GPE PE: without a deep knowledge of the ADSP21xxx architecture, it is rather difficult to estimate its performance scalability. This is because in contrast to highly the pipelined architectures present in CISC and RISC processors, DSP are conceived to have one clock-cycle latency for the MAC operation. Thus for the GPE PE the only possible performance prediction can be made by considering the announced evolutions as the one of Table 2.2.

Apart from the performance improvement brought about by the increasing DSP clock frequency, in the case of the GPE a relevant boosting factor can be obtained by efficient exploitation of the emerging SIMD/VLIW parallel DSP's.

DRAM: this is without doubt the most difficult element of the MPIC to predict. First of all the introduction of the high performance DRAMs presented in Table 4.2 has brought an acceleration with respect to the historical 7% DRAM access-time improvement cited in [Hennessy 90]. Secondly the next evolution of the DRAM characteristics is conditioned by the effect of the introduction of Silicon On Insulator (SOI) technology which in [Park 99] is predicted to bring a 25% improvement factor. At last, the strongest influence on the next generation DRAM's will be exercised by the definition of the DRAM interface. Particularly in regards to this aspect it is currently very difficult to make a reliable prediction, since huge commercial interests are involved in the definition of the new standard, thus resulting in a reduced influence of the technical aspects. An overview on the interfaces relating to the next DRAM generation is listed in Table 9.1.

While the communication protocol is quite different, the solutions proposed in Table 9.1 are once again characterized by efficient bandwidth exploitation during burst-wise row memory transmission only. Thus the HBS/VBS communication methods will remain effective in the next DRAM generation.

9.1.2 Performance Prediction

As the MPIC is a heterogeneous system and considering the uncertainty present in the performance scaling of some of its processing elements, a general performance road-map for the next years is not feasible. But a quite reliable prediction can be made for the near future. The following system parameters are considered:
9.1. System Performance Scaling

<table>
<thead>
<tr>
<th>Interface</th>
<th>Throughput</th>
<th>Size</th>
<th>Techn.</th>
<th>Manuf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR DRAM</td>
<td>333 MBit/s/pin</td>
<td>1 GBit</td>
<td>0.14 μm</td>
<td>Samsung Inc.</td>
</tr>
<tr>
<td>DDR DRAM</td>
<td>250 MBit/s/pin</td>
<td>1 GBit</td>
<td>0.18 μm</td>
<td>NEC Inc.</td>
</tr>
<tr>
<td>SL DRAM</td>
<td>400 MBit/s/pin</td>
<td>72 MBit</td>
<td>0.25 μm</td>
<td>Various</td>
</tr>
<tr>
<td>RD DRAM</td>
<td>800 MBit/s/pin</td>
<td>72 MBit</td>
<td>0.25 μm</td>
<td>Toshiba Inc.</td>
</tr>
</tbody>
</table>

Table 9.1: Emerging DRAM interfaces: Double Data Rate DRAM [Yoon 99] [Takai 99], Synchronous Link DRAM [Paris 99] and the proprietary Rambus DRAM [Tabake 99]

- The DRAM options presented in Table 9.1 refer to research activities which are expected to be in production in year 2001 [SIA 97]. A throughput of 400 Mbit/pin/s is considered as feasible for the HBS ($T_{ACOL} = 2.5$ ns for the HBS model) while a more conservative value is taken for the VBS. This is because the token-passing network implementation can create some timing-problems in conjunction with the advanced clocking strategies used in the emerging DRAM interfaces. Thus for the VBS model a value of $T_{AC} = 5$ ns (200 Mbit/pin/s) is considered.

- 0.18 μm technology is used for the FDE PE performance prediction (production in 2001). This implies a 1024 1D complex FFT execution time of 24.6 μs.

- In 2001 the announced DSP of Table 2.2 are considered to be mature and well established (inclusive of the compiler technology). Since the GPE is closely related to the SHARC DSP family for its multi-processor capabilities, a clock frequency of 250 MHz near to an architecture boosting of factor 2 with respect to the current generation is used for the performance prediction.

Inserting these values in the MPIC model already used for the performance evaluation with state of the art technology, the chart of Figure 9.1 is produced.

With respect to the performance charts obtained with state of the art technology (upper chart of Figure 7.2), the same qualitative behavior of the speedup is observed in Figure 9.1. The predicted system linearly scales its performance by a larger number of FDE PE's. This is due to the considerable improvement brought by the assumed DRAM throughput. Another quantitative effect appearing from a confrontation is the boosting of the GPE which in Figure 7.2 consists of 4 DSP's.
while in Figure 9.1 only one or two DSP have to be used for a quadruple frame rate.

9.2 RMA as general VSP architecture

Leaving apart the already considered integration problems of logic and memory, the RMA can be investigated with respect to its usability as Video Signal Processor (VSP) architecture. The following aspect are relevant:

- SPMD and MPMD architectures are mostly used for the parallelization of computer vision and image processing algorithm
- the dynamic re-configuration of shared and distributed memory
is also to be considered of great advantage for many algorithms, when 2D computation does occur

- the VBS transposition method can be expanded in order to implement more complex data reorganizations. This results in substituting the token passing mechanism with a more general PE activation structure. On the CC also a more complex data sequencing has to be supported with all the repercussions on the HW implementation.

- being optimized for the high regularity and data independence of the FB-CCF, the RMA approach loses its strength when data-dependent computation and high level image processing is envisaged since there is no direct connection between the PE's. In the case where a better inter-processor connectivity is provided, the RMA could perform a little better in these situations.

- RMA also implies very high homogeneity of the computational tasks, which leads to ideal load balancing and furthermore to the synchronized data reorganization in the shared memory part.

As a consequence the RMA is effective only on a subset of the large spectrum of image processing algorithms. This is already to be considered as a big merit for a solution developed under stringent performance constraints for a dedicated algorithm. But the broad range of processing and communication tasks to be supported by a general VSP architecture go beyond the RMA usability.

9.3 Application to Other Algorithms

The fact of implementing high resolution accurate 2D FFT under real-time conditions, opens a wide range of applications for the MPIC. The system can be reprogrammed by acting on the process scheduling programmed on the SMC, allowing for instance to process data present in the FDE main memory with the GPE SHARC processors. Using the link service network, dedicated task controlling and global data redistribution can be realized by merely modifying the software kernel on the SMC and the program controller on the CC. In this way the 2D FFT-CMUL-2D iFFT sequence used by the FB-CCF can be modified aiming to more complex computation in the frequency domain.
Let's consider some of the most relevant applications that can be efficiently mapped on the MPIC system:

**Filtering:** complex image filtering can be realized by merely changing the template used for the BST with any complex 2D pattern. Periodic structure analysis and noise filtering [Russ 95] for example is particularly improved by a computation in the frequency domain with respect to the case in the spatial domain. The achieved frame rates are the ones referring to the FB-CCF computation listed in Table 7.4.

**Convolution:** Real time image convolution is nowadays limited to kernels sizes of $3 \times 3$ and $5 \times 5$ computed in the space domain. More complex convolutions kernels can be used for example to correct blurring effects introduced by optical devices into the sampled CCD image. The operation of physical optics is readily modeled in the frequency domain and furthermore the point-spread function of the optical system (the degree to which a perfect point in the object plane is blurred in the image plane) can be determined and used for removing some of the blur. The convolution can be carried out in the frequency domain by a point-wise multiplication, without conjugating the template complex values as in the CCF case. As in the filtering application, the achievable frame rate is the of the FB-CCF computation of Table 7.4.

**SAR and Tomography:** synthetic aperture radar (SAR) and computer aided tomography (TAC) are both emerging fields which intensively deals with high resolution 2D computation in the frequency domain [Wang 97] [Webb 98]. Both fields are characterized by a strong demand of computing performance for the support of the intensively used 2D FFT and inverse 2D FFT [Jenkins 82]. The high accuracy provided by the FDE in conjunction with the real time processing capabilities, are factors which further increase the usability of the BST system in these application fields.

**Watermarking:** this is an emerging discipline aiming at embedding relevant information into multimedia data-streams. In order to achieve high robustness of the introduced signature against scaling, reproduction, compression and piracy while maintaining its invisibility, the use of frequency domain based techniques such as spread spectrum modulation is widely used [Cox 97] [Tirkel 96].
Even if most of the watermarking can be made off-line by most applications, it is presumable that real time operation will acquire importance, boosted by the growing impact of real-time video and multimedia applications. In this case the GPE/FDE system can be exploited for 2D FFT, signature insertion and 2D inverse FFT of video streams achieving video real time (25 Frames/s) by images $512 \times 512$, since a slightly modified FB-CCF procedure is required.

9.4 Conclusions

The analysis of the NCCF implementation carried out in this work, improves the usability of this very important class of algorithms in the high resolution, real time application field which up to now has been considered only in theoretical and SW based investigations. The implementation of the BST in the frequency domain has brought about a wide range of improvements from the point of view of the computational requirements as well as from the point of view of accuracy. With respect to other system targeting the same class of algorithms, frame rates of a higher order of magnitude are realized by the MPIC system, without any restriction on the input data format: all of this is achieved maintaining a very high compactness. This could only be achieved with a dedicated system and processor architecture based on a specific algorithm-optimization.

The frequency based approach opens new interesting perspectives from the point of view of the algorithms. The presented architecture revalues a large number of algorithms and methods in the 2D signal processing domain, which up to now have been banished to a secondary role because of their high computational aspects. Up to now by reading 2D signal processing literature, the impression a reader gets when frequency based methods are examined, is that this class of methods has considerable potential, but its relevance in applied machine vision is still too small.

The presented system makes it possible to use high resolution image material, transform it into the frequency domain, process it and (if needed) transform it back into the spatial domain: all of this while achieving real time operation and maintaining block floating point accuracy. Further investigations should reconsider this methods and analyze which improvements should be applied to the GPE/FDE architecture (system as well as CPU architecture) in order to efficiently
support a wider range of algorithms.

As with every other parallel computer, the MPIC has to maintain a consistent performance gap with respect to standard micro processors and digital signal processors, in order to justify its higher complexity. Particularly critical factors for the effectiveness of the RMA architecture with respect to the future DSP generations can be localized in the bandwidth limitations imposed by IC interconnection bandwidth and memory technology. In the first case the implementation of the FDE in Multi Chip Module (MCM) technology represents a solution with very attractive perspectives. Furthermore the VBS bandwidth should benefit from the oncoming evolution of memory device technology which surely will be based on synchronous transmission protocols, giving more tuning possibilities to the token passing network by increasing clock frequencies. A further bandwidth improvement can be achieved by an on-chip integration of the CC leading to a faster access of the data present in the SPITFFIRE cache.

In any case, the RMA and the FDE/GPE approach have shown its cost/performance validity in the range of 4...16 PE. This range of parallelization is of a high significance in every processor or technology generation, since there are always high-end applications which are supposed to invest a certain overhead in order to achieve performance boosting of an order of magnitude but without entering into the expensive general purpose parallel processing domain. Thus the MPIC, which represents independently of the technology an optimal solution for the supported class of algorithms, will maintain its relevance during the next technology evolutions.
Appendix A

Used Abbreviations and Parameters

Abbreviations

BST : Blue Screen Technique
BE : Block Exponent
C2R : Extraction of two real rows from a complex one
CC : Communication Controller
CCF : Cross Correlation Function
CMUL : Complex Multiplication
DAR : Data-Address Relationship
DIS : Half-image discard in RV-FFT
FB-CCF : Frequency Based CCF
FDE : Frequency Domain Engine
FFT : Fast Fourier Transform
GMBE : Global (at FDE level) maximum BE
GPE : General Purpose Engine
HBS : Horizontal Bus System
LMBE : Local (at PE level) maximum BE
MPIC : Multiprocessor-System for Image Correlation
MEP : Maximum Extraction Procedure
MIRR : Half-image mirroring in RV-FFT
NCCF : Normalized Cross Correlation Function
MRG : Merging of the parallel computed data
PART : Partitioning of a data-set for SPMD computation
R2C : Merging of two real rows into a complex one
RV-FFT : Real Valued FFT
RMA : Reconfigurable Memory Architecture
SEP: : Separate Operation in RV-FFT
SMC : System Master Controller
TRSP : Image Transposition
UNSEP: : Unseparate Operation in RV-FFT
VBS : Vertical Bus System

Parameters

\[ btp(u,v) \] : Binarised template: \( 0 \leq u, v < T \)
\[ CPP \] : Number of clock cycles per pixel on GPE
\[ im(x,y) \] : Input image: \( 0 \leq x, y < N \)
\[ HBS_{BW} \] : HBS bandwidth
\[ N \] : Size in pixels of the input image
\[ N_{PH} \] : Number of iterations in phase \( x \)
\[ N_{PH_{RV}} \] : Number of iterations in phase \( x \) with RV-FFT
\[ P \] : Number of processor elements in the FDE
\[ R \] : Number of physical memory cells per logical word
\[ RSM_{BW} \] : Realized sustained memory bandwidth
\[ SH_{NUM} \] : Number of DSP elements in the GPE
\[ tp(u,v) \] : Searched template: \( 0 \leq u, v < T \)
\[ T \] : Size in pixels of the searched template
\[ \bar{T}_{AC} \] : Mean main memory access time on VBS
\[ \bar{T}_{ACOL} \] : Access time within the same row
\[ \bar{T}_{AROW} \] : Access time for data on new memory row
\[ T_{CPU} \] : FDE CPU processing time in a phase
\[ T_{CYCLE} \] : Cycle time for a random memory access
\[ T_{GET} \] : Duration of the GET operation on HBS
\[ T_{IF} \] : Time for loading a frame in the FDE
\[ T_{FD} \] : Computation time for the FB-CCF
\[ T_{FFT} \] : Time for a FFT computation
\[ T_{FRAME} \] : MPIC system cycle time
\[ T_{LOC} \] : Time for a localization
\[ T_{NR} \] : Computation time for normalization and MEP
$T_{OF}$ : Time for moving a frame from the GPE to the FDE

$T_{PLD}$ : Considered maximal frequency of PLD's

$T_{PREC}$ : Precharge time before initiating new row access

$T_{PUT}$ : Duration of the PUT operation on HBS

$T_{SEP}$ : Time for the computation of the SEP/UNSEP

$T_{SP\_IO}$ : SPITFFIRE I/O cycle time

$VBS_{BL}$ : Size (words) of the data-burst produced on the VBS

$VBS_{BW}$ : VBS bandwidth

$VBS_{PAR}$ : Number of VBS data paths
Bibliography


Bibliography


Curriculum Vitae

9. March 1969 Born in Chiasso, Switzerland
1975-1980 Primary school in Davesco
1980-1984 Secondary school in Viganello
1984-1988 Gymnasyum, Lugano 1
   1988 Matura Typus C (science)
1988-1993 Swiss Federal Institute of Technology Zürich
   (ETH Zürich)
   1993 Diploma in Electrical Engineering
      (Dipl. El.-Ing. ETH)
1993-1999 Electronics Laboratory, ETH Zürich
      Research Assistant, Image Processing Group