

SPECIAL PURPOSE HARDWARE FOR IMAGE PROCESSING

Kaarthik Sivashanmugam
Dept. of Computer Science
University of Georgia

Submitted to : Dr. Hamid Arabnia

ABSTRACT:

This paper studies the Special purpose hardware for image processing and their types. Simple hardware for image processing have been considered. It also deals with the problems that necessitated the need for them. For these problems hardware solutions arrived at are also mentioned. This paper focuses the use of reconfigurable hardware in image processing and finally this paper will analyze some of the recent trends and papers published in this area.

Keywords: General purpose computing, Reconfigurable computing, Reconfigurable bus, FPGA, Image processing algorithms, High performance-parallel computing.

MOTIVATION:

Image processing applications demand significant amount of processing power. Traditionally, such applications are realized using software on top of different architectures such as DSPs and microprocessors with enhancements. Dedicated hardware were also applied in image processing applications. Most of the image processing algorithms are CPU time intensive normally being implemented in software however with lower performance than custom implementations. Custom implementation in hardware (ASIC) allows real-time processing, having higher cost and time-to-market than software implementation. Recently FPGAs emerged as reasonable alternatives to custom hardware (ASICs) for implementations of digital systems. Reconfigurable computing offers promising solutions for many applications, enjoying cost-effective and flexible technology. Our discussion will focus mainly on the reconfigurable hardware for image processing applications.

INTRODUCTION:

Image Processing is an application area which requires fast realization of certain computationally intensive operations and the ability for the system developer to experiment with algorithms. For many years electronic hardware used for computation could be divided into two main types, general purpose and application specific. General-purpose hardware is exemplified by microprocessors such as the Intel80x86, Pentium and the Motorola 68000 family which serve as the main processing unit in most personal computers. The architecture of these devices is fixed and includes specific hardware to implement a limited, pre-defined, set of instructions. These microprocessors run programs, which are lists of instructions to be executed that are stored in external memory. However these computers can be slow performing certain kinds of operations such as those involving floating point calculations or complex mathematical functions. For this reason most modern computers have one or more coprocessors which are application specific hardware that performs certain functions very quickly. Examples include graphics coprocessors that perform 3D rendering.

Application-specific computing hardware (ASIC) performs functions very quickly, but the price of this speed is limited flexibility. As the name implies this type of hardware can only perform one function or a group of closely related functions. They cannot be reprogrammed. If the application specific hardware is needed to perform a new function then a new hardware design will have to be created. Since ASIC are custom ICs, they are also very expensive to fabricate and it takes week to months to design a new ASIC and have it fabricated. Hence ASIC is only useful if the functions are known in advance and the requirements of the functions they perform are not going to change. In spite of this drawback ASIC hardware is widely used whenever speed is an important design consideration. By structuring hardware to match the problem, ASIC can often achieve computation speeds several orders of magnitude faster than general purpose hardware. This high-level of performance is obtained by utilizing several techniques like performing operations in parallel, organization of efficient data transfer, utilization of hardware structures that permit efficient data scheduling, reducing inefficiencies introduced when computation is halted to wait for new data.

In recent years, a new class of computing hardware has been gaining increasing research interest. Configurable computing hardware has some of the advantages of both general purpose computing and application specific hardware. Reconfigurable hardware devices in the form of Field Programmable Gate Arrays (FPGAs) have been proposed as viable system building blocks in the construction of high performance systems at an economical price. This type of hardware consists of large number of functional units with programmable interconnections. The functionality of the hardware is determined by how interconnections between functional units are configured and in some cases how the functional units themselves are configured. By changing the configuration the hardware can be made to perform a completely different function. Since the structure of the hardware has effectively been changed for the specific function to be implemented, many types of computations can be performed by CCMs at speeds close to those obtained using application specific hardware. In addition the configuration can be changed relatively quickly from one function to another.

FIXED HARDWARE & RECONFIGURABLE HARDWARE:

Microprocessors are at the heart of the most current high performance computing platforms. They provide flexible computing platform and are capable of executing large class of applications. Software is developed by implementing higher level operations using instruction set architecture. Unfortunately this generality is achieved at the expense of performance. The software program stored in memory is to be fetched, decode and executed. In addition, data is fetched from and stored back into memory.

Re-configurable computing hardware combines the programmability of a general purpose processor with the processing efficiency of a dedicated hardware solution in a small and flexible package. Complex functions are mapped onto the architecture achieving higher silicon utilization and reducing the instruction fetch and execute bottleneck. Configurable platforms which have shown impressive results typically have configurable logic attached to a host system through some interface such as the system bus or the I/O channels. The limiting factor in this case in

achieving higher performance on all application is the delay in communicating with configurable logic.

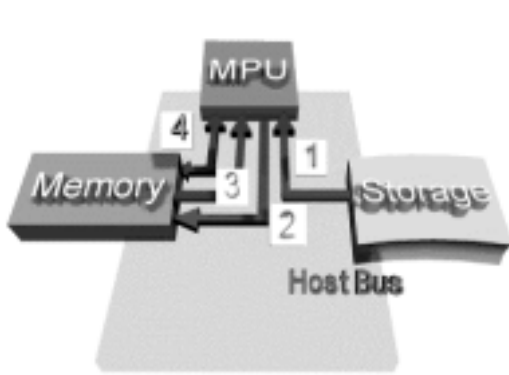


Fig 1

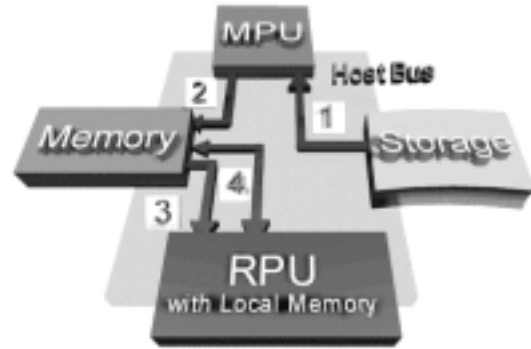


Fig 2

Reconfigurable Computing systems are those computing platforms whose architecture is modified by the software to suit the application at hand. This means that within the application program a software routine has been written to download a digital design (chip design) directly into the Reconfigurable Processing Unit (RPU) of the Reconfigurable Computer. It is like having custom processor chip manufactured for your application program. Most Reconfigurable Computing Systems are plug-in boards made for standard computers such as PCs and workstations. The Reconfigurable board acts as a Co-processor to the main MPU. Calculations normally done within the MPU are now carried out in the RPU instead. The main reasons for using this approach are:

1. Very High Speed Performance Gains; the fastest version of any program calculation is one in which a computer chip has been designed just to perform that specific calculation only,
2. Flexible Hardware; as new ways of solving problems arise, the RPU optimizes the compute power by implementing the calculations directly into a custom computing chip.

RECONFIGURABLE COMPUTING MODELS:

1. Reconfigurable Mesh

A regular mesh (Fig 1) is a $N \times N$ square grid with one processor per grid point. Except for the processors at the boundary, every other processor is connected to its neighbors to the left, right, above and below through bi-directional links. The instruction stream is MIMD (that is each node can send and receive a packet from all its (four or less) neighbors in one unit time).

Reconfigurable Mesh (Fig 2) is a theoretical model of a VLSI array of processors overlaid with a reconfigurable bus architecture. A reconfigurable bus architecture consists of a multi-dimensional array of processing elements connected to a bus through a fixed number of I/O ports. Bus reconfiguration is achieved by locally configuring the switches within each PE.

Different shapes of buses such as rows, columns, diagonals, zig-zag and staircase can be formed by configuring the switches/ports.

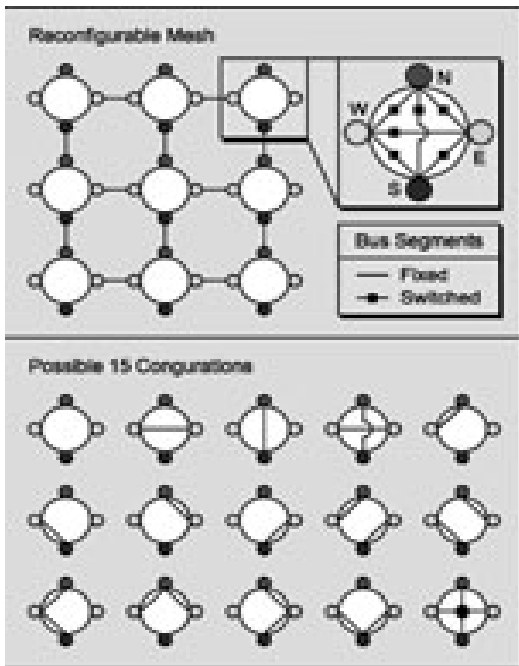


Figure 1

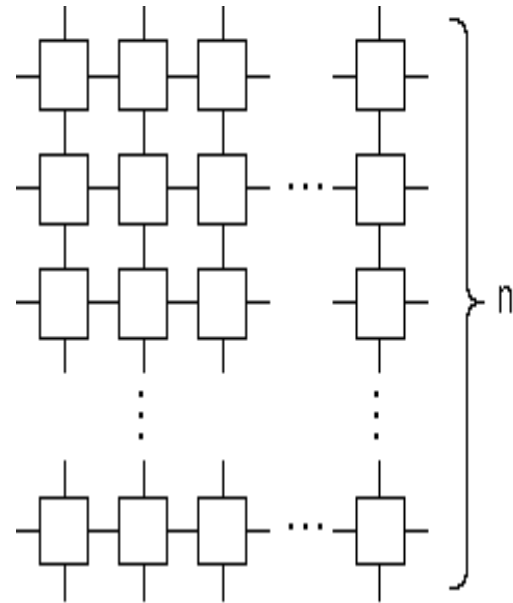


Figure 2

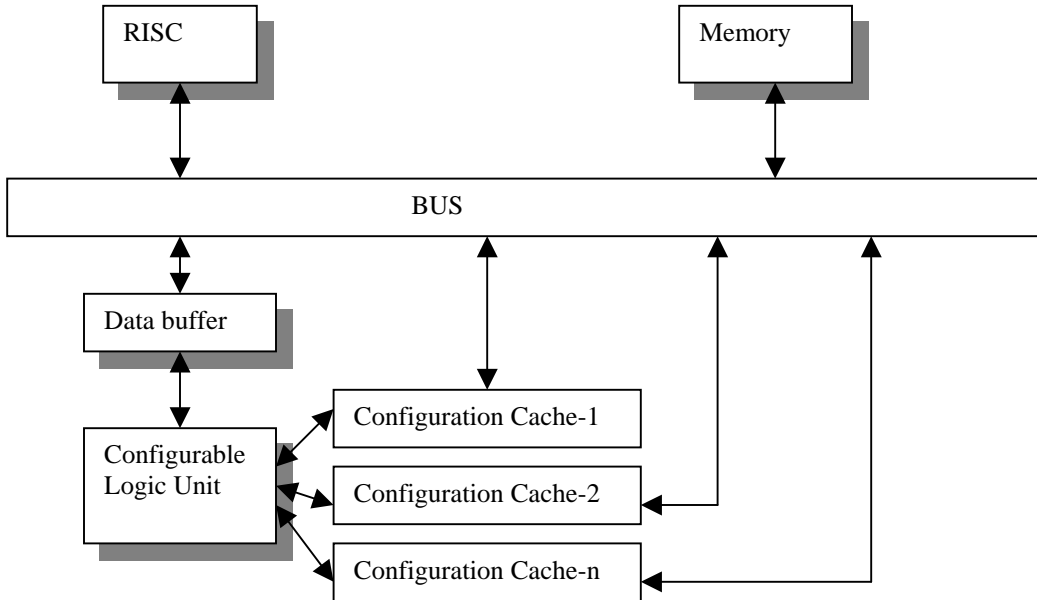
The basic computational unit of the reconfigurable mesh is the Processing Element (PE) which consists of a switch, local storage and an ALU. In a unit time, a PE can perform:

1. Setting up of a connection pattern.
2. Read from or wrote onto a bus or local storage
3. Logical or arithmetic operations on local data.

2. Hybrid System Architecture model

Hybrid System Architecture model (HySAM) is a parameterized model of a configurable computing system, which consists of configurable logic attached to a traditional microprocessor. The architecture consists of a traditional processor, standard memory, configurable logic, configuration memory and data buffers communication through an interconnection network.

The applications tasks to be executed are decomposed into a sequence of Configurable Logic functions. Execution of a funtion on the CLU involves loading the configuration onto the CLU and communication the required data to the CLU. After executing in a configuration to execute a different configuration, the CLU has to be reconfigured which takes some time. The cost is a measure of the amount of logic reconfigured and the time spent in reconfiguring. The total execution time is the time spent executing in each configuration and the time spent in reconfiguration between executions.



Hybrid System Architecture

RECENTLY PUBLISHED PAPERS:

HARDWARE ACCELERATOR FOR IMAGE PROCESSING

In order to achieve the real-time requirements, the programmable circuits mentioned above must exploit parallelism that can be extracted from the processed algorithm. The parallelism can concern either data, as in SIMD (Single Instruction Multiple Data) vector architectures, or instructions, as in superscalar or VLIW architectures. Data parallelism architectures were used for a long time because they were the only ones able to extract sufficient parallelism from signal processing programs. With the improvement of instruction scheduling algorithms, new perspectives have appeared, and DSP designers, such as Texas Instruments (with the TMS320C6xx series) and Philips (with the Trimedia) are going into instruction parallelism architectures, allowing to extend the range of algorithms where parallelism can be extracted. It combines both solutions, exploiting data and instruction parallelism, and is composed of a standard RISC processor or DSP, standing for algorithms that have few intrinsic data parallelism (e.g. for control tasks). However, if the processor has efficient compiling tools, parallel execution can still be achieved at the instruction level. A vector coprocessor is also implemented either in a large FPGA or in an ASIC, allowing to speed-up applications where operations are performed on

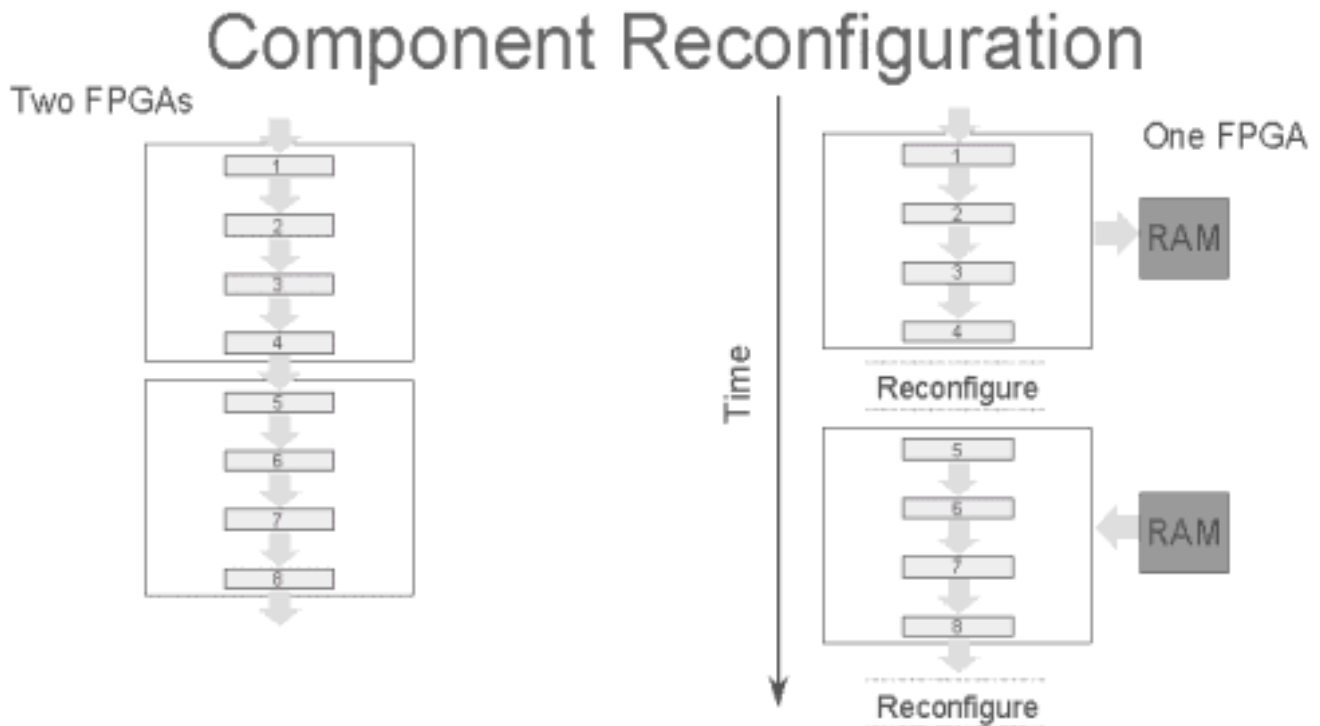
The system should be able to do digital signal processing in conjunction with graphics at real-time or near-real-time rates. Where algorithmic complexity is too high for real-time computation, the system should be able to run asynchronously, storing the results for later display at synchronized real-time rates. The system should be easily programmable, without undue attention to hardware resource management or process synchronization. The system should support time-sharing among multiple processes. The

system should be easily upgradable as processing requirements change or as technology improves. The system should be compact and relatively inexpensive, so that multiple copies can be built.

PIPERENCH

This is a coprocessor for streaming multi media acceleration. Piperench uses a technique called Pipeline reconfiguration to solve the problems of compatibility, reconfiguration time and forward compatibility. Piperench is currently used as an attach processor. It is known that application specific configurations of reconfigurable fabrics can be used to accelerate certain applications. The static nature of these configuration causes two significant problems. Computation may require more hardware than what is available and given more hardware there is no way that a single hardware design can exploit the additional resources that will inevitable become available in future process generations. This paper reviews a technique called pipeline reconfiguration that allows a large logical design to be implemented on a small piece of hardware through rapid reconfiguration of that hardware. This behaves like Pipelined hardware with different static configuration and each is loaded one per cycle into the hardware.

Example: 2 stage is simulated in 1 stage device



- FPGA size determines unit of reconfiguration
- Need memory for intermediate results
- Reconfiguration time impacts performance

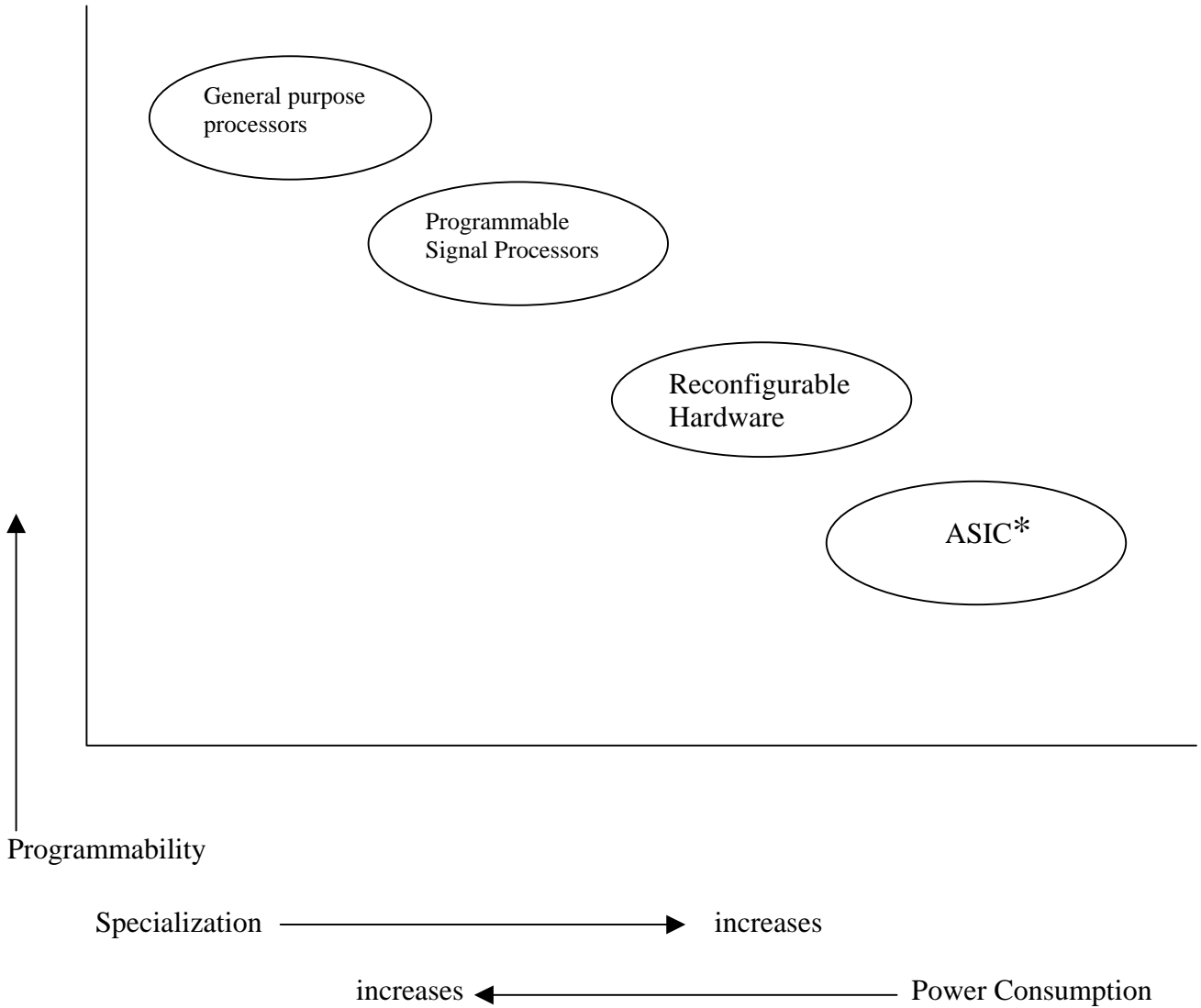
Pipeline reconfiguration is a method of virtualizing pipelined hardware application designs by breaking the single static configuration into pieces that correspond to pipeline stages in the application. These configurations are then loaded one per cycle into the fabric. This makes it possible to perform the computation even if though the whole configuration is never present in the fabric at one time. The virtualization process is illustrated in Fig which shows a two stage is virtualized in a single stage fabric. Because the configuration happens concurrently with the execution of the other stages, there is no loss in performance due to reconfiguration. As the pipeline is filling with data stages of the computation are being configured ahead of that data. Even if there is no virtualization configuration time is equivalent to the pipeline fill time of the application and therefore does not reduce the maximum throughput of the application. In order for the virtualization to work the state in any pipeline stage must be a function only of the current state of that stage and the current state of previous stage. In other words cyclic dependencies must fit within one stage of the pipeline. Interconnections that directly skips over one or more stages is not allowed nor are the connections from one stage to a previous stage.

DYNAMICALLY RECONFIGURABLE ARCHITECTURES FOR IMAGE PROCESSOR APPLICATIONS

This paper presents an overview of the principles that underlie the speed-up achievable by dynamic hardware reconfigurations and demonstrates the advantage of dynamic reconfiguration in the new implementation of a image processor called DRIP which achieves a real-time performance that is faster than its pipelined non reconfigurable version. First step in the definition of the DRIP architecture was the customization of its PEs. The image processing algorithm is described using a high level language like C and translated to an intermediate representation which matches the DRIP architecture. After specification the algorithm is compiled/synthesized using previously designed function library, fully optimized to achieve better performance. The configuration bitstream that customizes the FPGA for the optimized algorithm implementation is stored in a configuration library. Once the configuration bit stream data is stored it can be used repeatedly and over several modules of the entire architecture. A dynamic image processing system that relies on DRIP flexibility can take advantage of a web-based environment for remote processing. A web-based framework can allow the distribution of the design/execution tasks. Dynamic reconfiguration can obtain a considerable gain in area, performance and cost for an application specific system. This reconfigurable dynamic image processor achieves 80% area reduction and 3 times faster clock rate. The web-based framework suggested will allow to define a methodology for dynamic reconfiguration on a widely distributed environment. That could become a relevant market for reconfigurable computing for efficient supply of an architecture on demand. Hardware upgrading, maintenance and adaptation could be performed from a remote host.

APPENDIX

Digital Signal / Image processing hardware spectrum



*Application Specific Integrated Circuits

REFERENCES

[1] R.Miller, V.K.P.Kumar, D.Reisis, and Q.F.Stout, Parallel Computations on Reconfigurable Meshes, IEEE Transactions on Computers, 42 (1993), 678-692

[2] S.Rajasekaran, Mesh-connected computers with fixed and reconfigurable buses: Packet routing, sorting and selection. In Proc. 1st Annual European Symposium on algorithms, September 1993, 309-320

[3] Y.Ben-Asher, D.Peleg, R.Ramaswami, A.Schuster, The power of reconfiguration, 18th International colloquium on Automata, Languages and Programming, July 1991, Madrid.

[4] H.Li and Q.Stout, Reconfigurable massively parallel Computers, Prentice-Hall, 1991

[5] Alexandro M S Adario, Eduardo L Roehle, Sergio Bampi, Dynamically Reconfigurable Architecture for Image Processor Applications, ACM, 1999

[6] Seth Copen Goldstein, Herman Schmit, Matthew Moe, Mihai Budiu, Srihari Cadambi, Reed Taylor, Ronald Laufer, PipeRench: a Coprocessor for Straming Multimedia Acceleration, The 26th Annual International Symposium on Computer Architecture, May 1999

[7] L. Petit, J. D. Legat, Hardware Techniques for the Real Time implementation of Image Processing Algorithms, University Catholique de Louvian, Belgium

Reference websites:

[a] <http://bwrc.eecs.berkeley.edu/Classes/CS252/Notes/67>

[b] http://www.krellinst.org/UCES/archive/classes/ASC/tableofcontents2_1.html

[c] <http://www-unix.mcs.anl.gov/dbpp/text/node1.html>

[d] <http://www.mv.com/org/rmesh/Tutorial.html>

[e] www.rsise.anu.edu.au/annrep/1998/researchcsl.html

[f] www.iti.fh-flensburg.de/lang/papers/trans/transcl.htm

[g] www.cs.umass.edu/~weems/CmpSci635/635lecture14.html