1. Introduction

With the increasing speed of todays computational devices (microprocessors as well as configurable devices, etc.), the memory interface speed becomes the limiting factor for the overall system throughput [HS96]. Several techniques[1] to alleviate this problem have been developed. Since most of these techniques focus on the locality of instructions in a program [KT96], these methods can not be applied to reconfigurable computing machines, as they are configured before program execution [BHH98] and computation data doesn't has any loops.

Memory Bandwidth: Software vs. Accelerator

Since an accelerator for an application (hopefully) has higher throughput than its software solution, the accelerator makes the memory bandwidth problem worse. This means, memory/accelerator communication architectures are mostly more challenging for reconfigurable computing platforms than for procedural platforms (i.e. von Neumann like). Acceleration mechanisms are mainly based on:

- Reconfigurable accelerators abolish the instruction fetch overhead during runtime [HHW90]. The memory interface is not burdened by instruction fetch cycles and therefore also no addresses have to be computed for instruction fetches.
- Some reconfigurable accelerators compute data addresses on the basis of a generic formula. Compared to the software based address calculation of microprocessors, this method is much faster [HHW90]. Examples have been published [HHW88] [HHW90], where up to 90% of computation time of a microprocessor is wasted for address computation.
- The computation throughput of reconfigurable accelerators is increased by heavily pipelined designs. This is possible because reconfigurable architectures usually provide a large number of registers (an overview on fine-grained architectures can be found in [BFR92], [OD95], [Tri94], or [Wan98], for coarse architectures see chapter2), and therefore applications may be implemented in deep pipelines, which results in a high data throughput.

The result of these acceleration mechanisms is an increased computation speed and data throughput. Because of reconfigurable accelerators are able to process data faster than microprocessors, it must provided in the same time by the memory interface. Therefore sequencers or address generators are important means to create memory interfaces or powerful memory architectures such as e.g. interleaving or burst mode capability.

1. E.g. the introduction of hierarchical memory [PH93] [WH90].

While most research projects still struggle with general architectural issues, the memory communication problem is undervalued in the area of reconfigurable computing. Usually a straightforward application specific data sequencer implementation is chosen to address small SRAMs[1], ignoring that the lack of a common concept causes a lot of overhead in several areas.

The Memory Interface Performance

Already for microprocessors the memory communication bandwidth has become worse and worse for each new technology generation. Examples have been published where more than 75% of computation time of a state of the art microprocessor the processor is waiting because of busy memory [PAC97]. But for accelerators, especially for data-intensive applications like image processing, the memory communication bandwidth problem is drastically more dramatic than with microprocessor usage. An novel universal concept must addresses this topic with several strategies to solve memory communication performance problems under consideration of the specific characteristics of reconfigurable architectures.

Previous MoM architectures (see chapter 5) are based on external memory devices. But systems on chip (SoC), faster memory architectures, and their availability as IP[2] core create a new situation. Compared to multi chip solutions SoC provide faster and more interconnect resources. At present there is an increasing trend to integrate smaller and larger reconfigurable parts in SoC [Rab00] [MOR] [MAL] [SIL] in particular for network processors [Cra99] [Gil99] [Wal99]. Therefore it becomes newly a commercially relevant approach to integrate coarse-grain reconfigurable accelerators with memory on chip (see [LSL99] [Tri99] [Sid99]), because on chip memory is expected to drastically shorten the cycle time.

Further new highly flexible synthesis methods for coarse-grained reconfigurable accelerators [HHH00] support the integration of efficient solutions for the memory bandwidth problem. This is in so far important, as completely universal accelerators are an illusion and application domain specific solutions are needed. IRAM [KP98] [KAP97] [PAC97] seem to have tried a fine grain or medium grain mixture or merging of logic and memory on the same chip. But it seems to be less promising because of the lack of good algorithms to map applications onto such a platform. Therefore in this thesis a solution with a clear architectural separation of processing block from memory module (also including independent memory banks on the same chip) is preferred.

1. For information on SRAM see e.g. [Pri96].

2. IP = intellectual property

Thesis Overview

This thesis will focus on the memory communication problem of reconfigurable computers based on coarse-grained architectures. It gives a general overview on relevant technologies for the presented work. This overview is organized in four chapters. First several coarse-grained reconfigurable architectures will be presented as a target platform for reconfigurable computing systems (chapter 2). After that current memory technologies and their mechanisms to accelerate memory accesses will be explained (chapter 3). Interesting address generators will be presented (chapter 4), and then the data sequencing concepts of existing reconfigurable computing machines will be discussed (chapter 5).

As a basis, on which the memory communication bandwidth is improved, the thesis will present an address generation concept for sequencing (chapter 6). Starting with a classification of memory accesses, special operators for data sequencing will be developed, taking into account, that they will be used for reconfigurable computing, i.e. the configuration load has to be minimized. On this basis a new basic access pattern has been determined. With further combinations of the basic pattern a general model of a generic data sequencer has been developed, which uses a parameter stack to provide high flexibility in access sequences. Based on this model two hardware implementations of generic data sequencers will be presented: a mapping to the coarse-grained KressArray-3 (section 6.5), and a hardwired version to be used in the Map-oriented Machine with Parallel Data Access (MoM-PDA, appendix D).

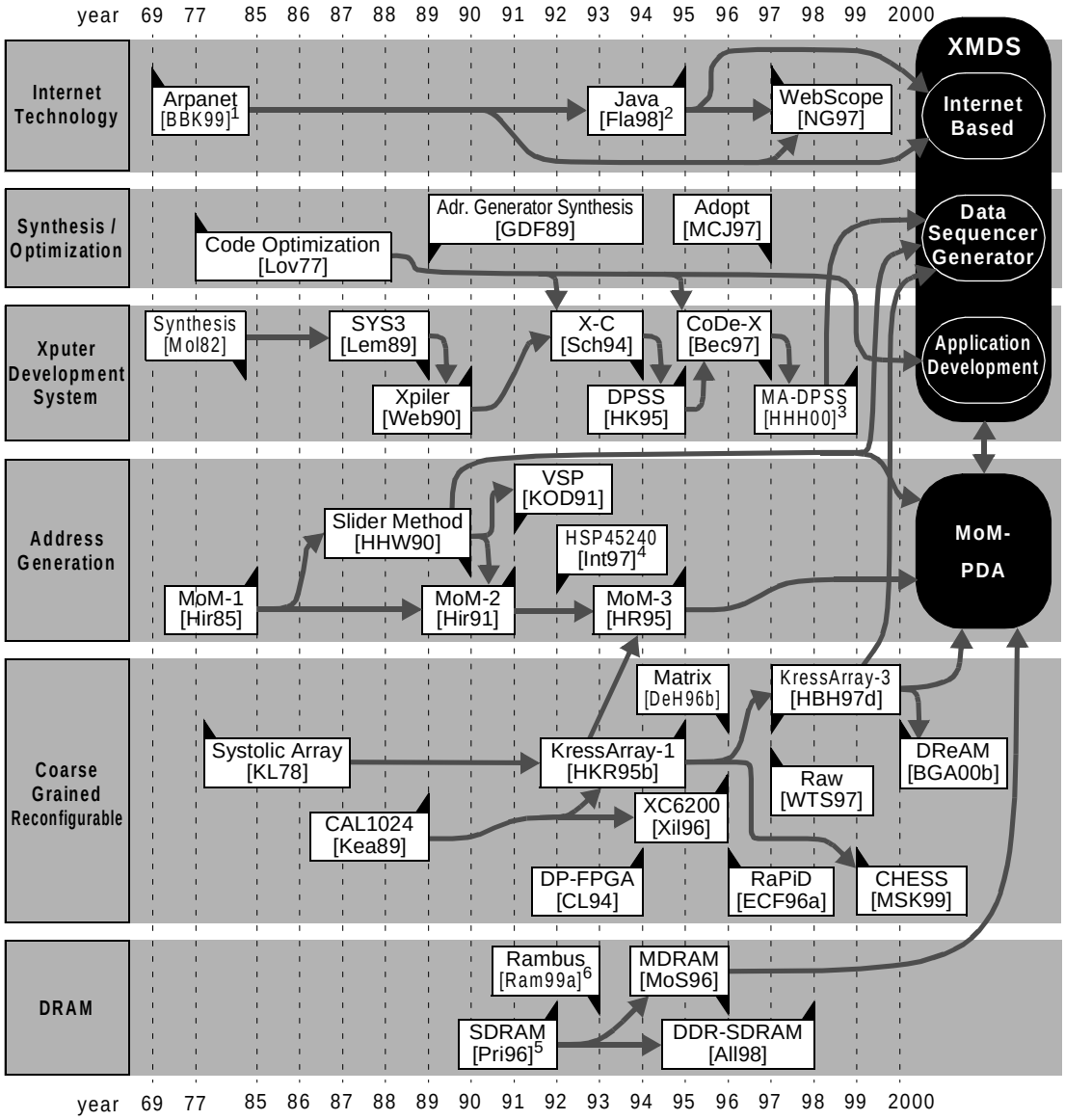
On hardware level a special 2-dimensional data memory organization enables the exploitation of speed-up mechanisms like burst mode and concurrent data accesses. The exploitation of the hardware level optimizations is further enhanced by loop-unrolling, modification of storage schemes and scheduling of the accesses (chapt er7).

Figure 1-1 at page 4 gives an overview on the direct influences of previous work, the technological and theoretical basis for the presented memory communication concept.

A Development Framework

Since computational machines based on data sequencing follow a new computing paradigm, which is quite different to the von Neumann paradigm, rethinking is necessary. To assist programming the data sequencer an Internet-based development framework, the Xputer Multimedia Development Framework (XMDS), has been developed. It supports the programmer in all development phases with a comprehensive toolset. Besides several possibilities to enter applications, it provides different tools for program validation and design output. One key feature of the XMDS is the way it is implemented. As an Internet-based software, the XMDS is installed on a Web-server. The user invokes the system by addressing a specific URL, then the XMDS is dynamically downloaded to the users computer.

The XMDS is introduced in appendix A and will be utilized for the application example in chapter8.



Legend: influence or technological basis

[1] [BBK99] gives an overview on the evolution of the Internet, which has its roots in the Arpanet. The Arpanet has been started in 1969.

[2] Java has been introduced in 1995. [Fla98] describes its progress.

[3] [HHH00] is the first publication on the MA-DPSS, which has already been developed in 1999.

[4] Since 1992 the HSP45240 has been distributed by Harris Semiconductor. Later Harris Semiconductor has been renamed to Intersil.

[5] [Pri96] is a survey on memories, which includes SDRAM technology, already been introduced in 1992.

[6] [Ram99a] is the most recent description of the Rambus technology, already been introduced in 1993.

Figure 1-1: Table of direct influences of previous work, the technological and theoretical basis for the presented memory communication concept.