

Interfacing the MoM-PDA to an Internet-based Development System

Michael Herz, Thomas Hoffmann, Ulrich Nageldinger, Christian Schreiber

University of Kaiserslautern,

Department for Computer Science

Erwin-Schroedinger-Strasse, D-67663 Kaiserslautern, Germany

WWW: <http://xputers.informatik.uni-kl.de> Email: abakus@informatik.uni-kl.de

Fax: ++49 631 205 2640

Abstract

The paper introduces an internet-based remote prototyping platform for accelerator applications. It gives an overview on the Xputer prototype Map-oriented Machine with Parallel Data Access (MoM-PDA), which operates as an accelerator to a host computer. The MoM-PDA utilizes the coarse-grained KressArray and implements concurrent data access to parallel memory banks. To give a large user community the opportunity to access the MoM-PDA prototype this paper describes how it can be connected to an Internet-based development system. This development system enables worldwide access to the prototype via the Internet.

1. Introduction

This paper introduces a prototyping platform for accelerator applications for reconfigurable computing. The system is based on an universal accelerator called Map-oriented Machine with Parallel Data Access (MoM-PDA) [6]. For the readers convenience it is briefly summarized as follows: The MoM-PDA is an FPGA-based custom computing machine, which is able to perform concurrent memory accesses by means of a dedicated memory organization scheme. A sophisticated Data Sequencer hardware performs the address generation for parallel memory banks and is the basis for several speed-up techniques. Further it utilizes the KressArray for highly area-efficient implementation of a reconfigurable datapath to perform computations. Details concerning the hardware implementation and speed-up techniques are presented in [6].

The method providing computation power to others via the Internet in combination with Internet based CAD systems is getting increasingly attention. In approaches based on the client/server paradigm, users with rather small computers are able to run complex design software which is partially executed on the server side [4].

With its sophisticated hardware concepts the MoM-PDA prototype is of interest for a large community. But because of the considerable costs for such a computing machine, it is more convenient to provide the users a virtual access via the Internet. Therefore a dedicated runtime system with multiple user support is needed. This runtime system is integrated into the Xputer Multimedia Development System (XMDS, [14]), an application development system for Xputers. The XMDS provides worldwide access to Xputer software via the Internet.

The paper is structured as follows. First the MoM-PDA is introduced and it is explained how it is physically connected to the XMDS server. After that it is shown how the XMDS controls the hardware and how user can access it via the Internet.

2. The Map-oriented Machine with Parallel Data Access (MoM-PDA)

In this section the MoM-PDA is introduced, which is used as an accelerator for a host computer. The machine architecture is based on the Xputer paradigm and uses field programmable logic as a reconfigurable ALU. To achieve high data throughput possible through parallelism on hardware level, the MoM-PDA has parallel access to computation data. Therefore a novel memory organization scheme is implemented.

2.1 The Xputer Machine Paradigm

The Xputer paradigm [13] is explained best by comparing it to the well-known von Neumann paradigm, which is used by most of today's computers. The main difference between the Xputer and von Neumann machines is, that the Xputer is controlled by a data stream instead of an instruction stream (but it is *not* a data flow machine [12]). The program to be executed is determined by first configuring the hardware. As there are no further instructions at run time, only a data memory is required.



Notice: This document has been provided by the contributing authors as a means to ensure timely dissemination of scholarship and technical work on a noncommercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

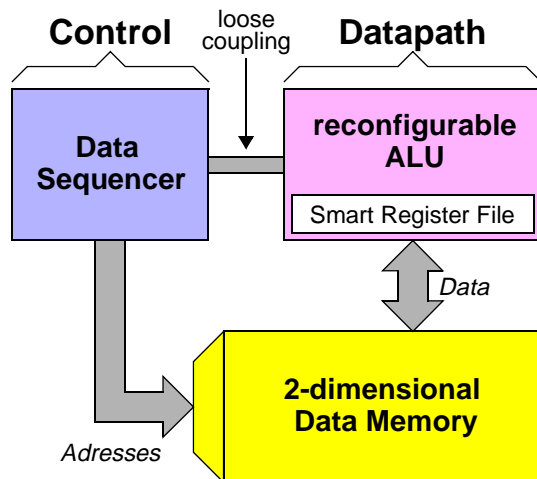


Figure 1. Basic Xputer Machine Architecture

This data memory is organized 2-dimensionally. At run time an address stream is generated by a Data Sequencer. The accessed data is passed from the data memory through a smart interface to the reconfigurable ALU (rALU) and back. The smart interface optimizes and reduces memory accesses. It stores interim results and holds data needed several times. Figure 1 shows all necessary components and their interconnect.

This principles are derived from the fact that many computation-intensive applications iterate the same operations over a large amount of data. They are accelerated by reducing the addressing overhead. All data needed for one computation step is held in the smart interface and can be accessed in parallel by the rALU. In contrast to von Neumann computers the rALU is not dependent of the sequencer. (Von Neumann machines do not efficiently support *soft* hardware. As soon as a data path is changed by structural programming, a von Neumann architecture would require a new tightly coupled instruction sequencer.) Therefore, it can be made application specific. The rALU is implemented with the coarse-grained KressArray [8][15]. Computations are performed by applying a configured complex operator on the data in the scan window. The hardware structure has the big advantage that, if the smart interface is integrated into the rALU, the rALU can be changed easily without modifying the whole machine. The residual control between Data Sequencer and rALU is needed when the data stream has to be influenced by the result of previous computations.

To clarify how operations are executed an execution model is shown in figure 2. A large amount of input data is typically organized in arrays (e.g. matrix, pictures) where the array elements are referenced as operands of a computation in a current iteration of a loop. These arrays

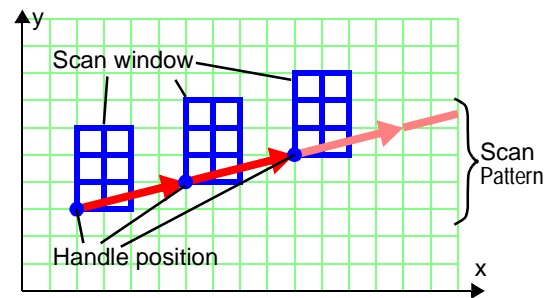


Figure 2. 2-dimensional memory organization and execution model

can be mapped onto a 2-dimensional organized memory without any reordering. This arrangement of data is called data map. The size of the data map depends on the application and varies in height and width. The part of the data memory which holds the data for the current iteration is determined by a so called scan window, which can be of any size and shape. The scan window can be seen as a model for the smart interface which holds a copy of the data memory. Each position of the scan window is labeled as read, write or read and write. The labels indicate the performed operation to the specified memory location. The position of the scan window is determined by the lower left corner, called handle (see figure 2). Operations are performed by moving the scan window over the data map and applying the compound operator on the data in each step. Thus this movement of the scan window called scan pattern is the main control mechanism. Because of their regularity, scan patterns can be described by only a few parameters. As a result no instruction cycles are needed for address generation.

In fact the execution model realizes a 2-level data sequencing. In the first level with the position of the scan window all data for one iteration is indicated. On hardware level the position of the scan window is determined by the x- and y-addresses of the scan pattern. On the second level the data is sequenced from the scan window into the rALU and back. This is done for each position depending on the read/write labels. On hardware level the Data Sequencer computes physical memory addresses for each scan window position.

2.2 2-dimensional Memory Organization

The 2-dimensional memory organization is achieved by cutting the memory in slices and mapping them on different memory modules. Because of the limited number of memory banks there might be more slices than parallel memory banks. In such a case several slices are appended and mapped in row major style to one memory bank. Figure 3 illustrates the mapping of a 2-dimensional data map onto 4 parallel memory banks. For bank 0 row major



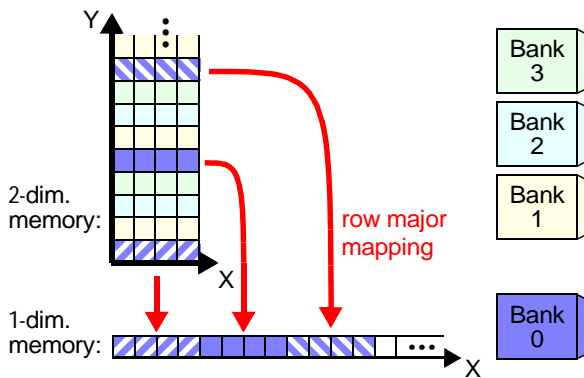


Figure 3. Mapping of 2-dim. memory to linear banks

mapping is shown.

The number of necessary memory banks depends on the application but the mapping of several memory slices on one memory bank is mostly not a drawback. Because of the scan window technique the optimum number of memory modules is equal to the height of the scan widow. Figure 4 demonstrates how scan window positions are assigned to several memory modules (4 in the example). The neighboring memory slices are mapped to parallel memory banks. Because the scan window holds only a small part of the data map, only a few number of parallel memory banks are sufficient to have each row of the scan window accessible in parallel.

Because data needed in one iteration of a loop is mapped to parallel memory banks, instruction level parallelism (ILP) on hardware level becomes more feasible. By providing multiple datapaths between memory banks and a parallel rALU, loop bodies can be computed concurrently on hardware level.

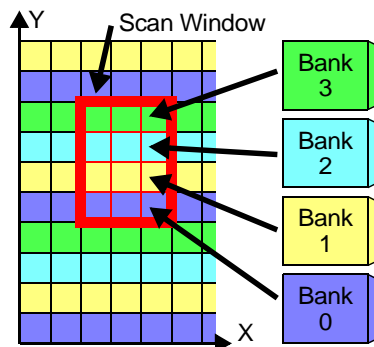


Figure 4. Mapping of the 2-dimensional memory to four linear memory banks

2.3 The Architecture of the Map-oriented Machine with Parallel Data Access (MoM-PDA)

The MoM-PDA is an accelerator to be connected to a host computer via a PCI interface [6]. It is based on the Xputer paradigm and utilizes the KressArray for reconfigurable computing. This section gives an short

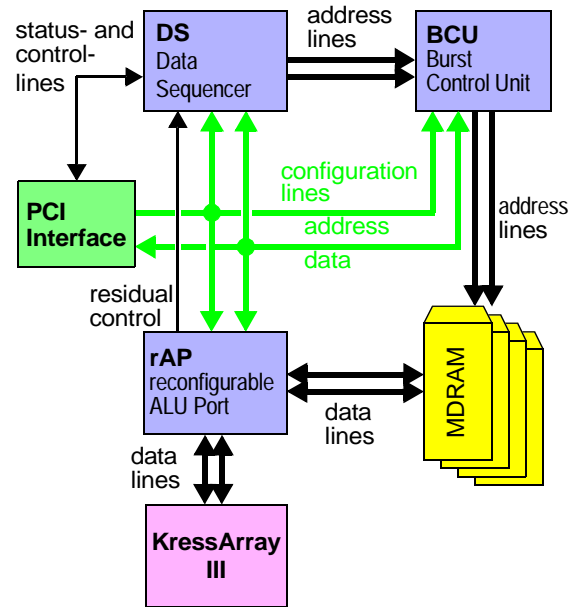


Figure 5. MoM-PDA machine overview

overview on the overall hardware structure of the MoM-PDA.

The most important new feature of the MoM-PDA prototype is parallel high speed access to the data as described above. Therefore it has 2 parallel banks of Multibank DRAM (MDRAM, [17]). Addresses for the MDRAMs are computed by the Data Sequencer and extended with burst information by the Burst Control Unit (BCU). In the current prototype implementation the Data Sequencer of the MoM-PDA is mapped to an Altera FLEX10K100 device [1]. The novel Data Sequencer structure handles up to 16 parallel tasks each consisting of an complex scan pattern [9][11]. Therefore up to 16 scan windows can operate on the data memory concurrently. The computation of the parallel tasks is done like known from multi-tasking systems. As described above the MoM-PDA has parallel data access capabilities. Therefore the Data Sequencer generates two parallel address streams. These addresses are passed to the Burst Control Unit (BCU, [3]), which initiates the accesses to the MDRAMs [17]. Computations are normally performed by the KressArray. Data is first routed to a reconfigurable ALU port (rAP, figure 5). It is implemented with a Xilinx XC6216 [19], which is connected to data lines of the data memory. The programmable space of the XC6216 device will be partitioned into two functional units. One unit will be the parallel memory interface for the MDRAMs [17] and the smart interface. This unit is the same for every application and is configured once at power up. The remaining programmable space can be used in two different ways:



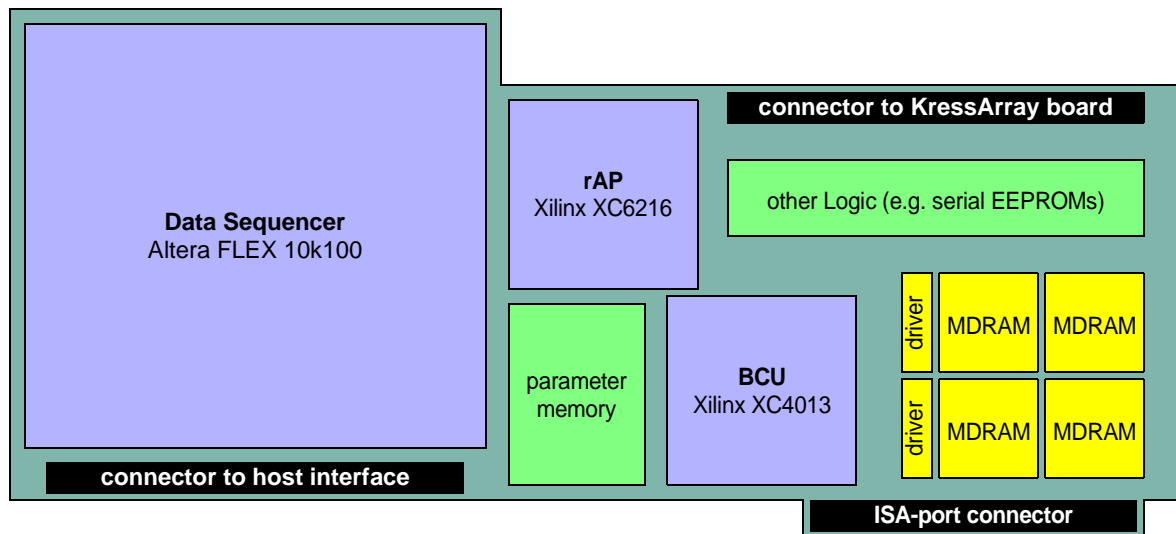


Figure 6. MoM-PDA Prototype Board

- for calculations it is (re-)configured for every task. Computations are performed by applying the configured operations on the data in the smart interface. This allows to build a small version of the MoM-PDA without the KressArray. Simple problems may be computed in the XC6216 only. Therefore a operator library has been implemented, mainly containing image processing applications (see [5] and [7]).
- as a connection to KressArray [10]. In that operation mode the XC6216 optimizes data exchange between the KressArray and the data memory.

2.4 Physical Prototype Implementation

The MoM-PDA prototype implementation is completely based on FPGAs. The only exception is the KressArray. The complete prototype needs 3 printed circuit boards (PCBs), which is mainly due to the prototype state.

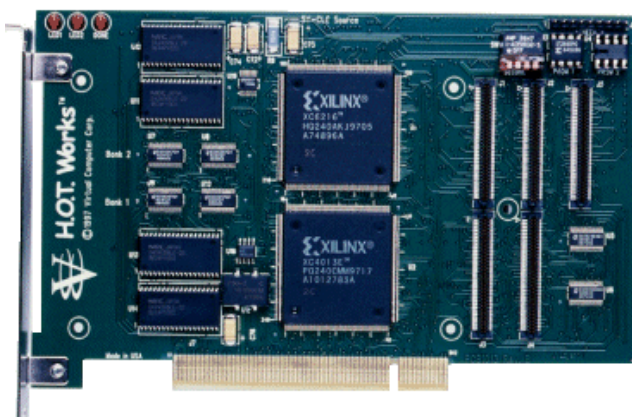


Figure 7. XC6216 PCI Board from VCC

The PCI interface is implemented with a commercially available FPGA board from Virtual Computer Corporation (VCC, [18], see figure 7). The so called H.O.T. Works board contains already a PCI interface and in addition it has a Xilinx XC6216 FPGA for user designs. This FPGA is mainly used to implement the glue-logic necessary to interface all MoM-PDA components. The FPGA board has further the advantage, that it is delivered with a C function library to program it. This library can be used with Java programs too (to see how to use the Java Native Interface together with the libraries refer to e.g. [16]).

The main components of the prototype are situated on the MoM-PDA prototype board (figure 6). This board contains the Data Sequencer, the reconfigurable ALU Port (rAP) and the Burst Control Unit (BCU), which can also be seen as a kind of MDRAM controller. Further 2 parallel banks each of 2 Siemens MDRAM chips are contained. Each MDRAM chip has a capacity of 1MB and operates at a clock frequency of 120 MHz.

The MoM-PDA prototype board is designed to work also in a stand-alone mode. Computations are then performed in the rAP and the parameter memory has to be replaced by an EEPROM or Flash memory programmed with application specific bytecodes for the Data Sequencer.

The MoM-PDA prototype board has 3 connectors:

- A ISA-port connector is used to plug the PCB into a ISA slot of a PC. This connector is only used for power supply.
- The connector to the host interface connects the board to the host interface implemented with the H.O.T. Works board.



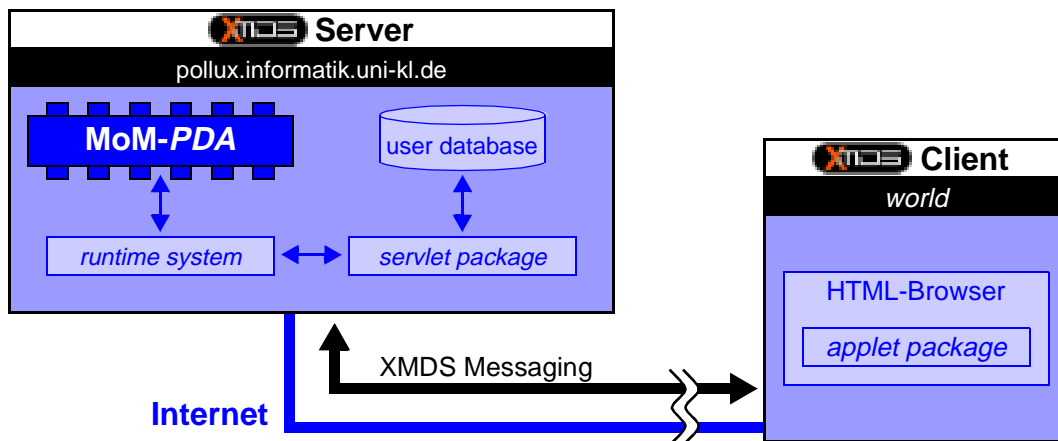


Figure 8. Overview of the XMDS.

- The connector to the KressArray board is to establish a connection to an external rALU, which will be a KressArray. Due to the prototype state using FPGAs instead of ASIC implementations, the KressArray has to be on a separated PCB.

All components are plugged into a Pentium processor PC running also the XMDS server [14]. A special runtime system allows users to run applications on the MoM-PDA via the Internet using the XMDS. This Internet based software is explained in the next section.

3. The Xputer Multimedia Development System (XMDS)

The XMDS is an Internet based CAD system for Xputers. It is a development System for the Xputer prototype MoM-PDA having the following features:

- a user front-end that is accessible via World Wide Web for several platforms,
- a central user administration,
- the modularity of its components, and so
- open system,
- the embedding of multimedia features, and
- an especially developed design user interface.

Because the World Wide Web is commonly explored with a Web browser, the user front-end of the XMDS must have a Web address, a URL (Uniform Resource Location), like normal Web documents. By pointing a browser to that particular URL, a user of the XMDS may download the front-end to his client machine and display it on the screen. Therefore, the part of the XMDS that should display on the user's machine must be executable by the most common Web browsers. The fact that a Web browser forms the environment of the front-end ensures a certain

independence against the underlying operation system (Windows, MacOS, UNIX, ...). Portations of a particular Web browser to different platforms ensure therefore the availability of the XMDS.

3.1 The XMDS Runtime Support

The XMDS as a distributed system accesses the Xputer prototype on the serverside. The MoM-PDA is directly connected to server which hosts the XMDS. While the front-end of the XMDS (client) realizes the interface to the user, the server part is designed to access the Xputer hardware through different communication layers. The organization of these layers and the realization of a control flow will build up the Xputer Runtime System. In this section, the Xputer Runtime System is discussed together with its hardware level connection to the MoM-PDA. The client/server relationship inside the XMDS and therefore the transmission of superordinated user requests is not shown here (refer to [14]).

The XMDS provides a runtime system for applications on the Xputer hardware. The runtime system consists on one hand of a set of functions accessing the MoM-PDA via the PCI bus of a PC. These functions are implemented in C because they have to operate on a sub-operating-system-level, which can not be reached by a conventional Java program. They form the lowest communication layer connecting directly the hardware ports of the PC. On the other hand an enveloping program realizes the control flow and calls the C functions. In approach to the XMDS modularization concepts [14], the program is implemented as an XMDS server module (called Computer) in Java, and calls the C functions using the Java Native Interface (Figure figure 9). While the XMDS server module forms the highest layer of the runtime system, the Java Native Interface defines an intermediate layer between control flow and C functions.



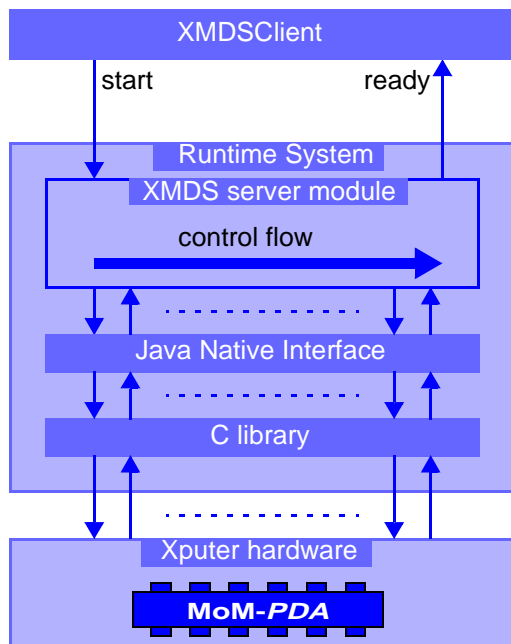


Figure 9. Overview of the Xputer Runtime System

The Java Native Interface allows to utilize special capabilities of a computer, operating system or non-Java software that the Java Class library does not already provide. This includes interfacing to plug-in cards like the H.O.T. Works board. More details about the Java Native Interface can be found in several Java books, e.g. in [16].

Beside the "start" and "ready" signals that involve the XMDSClient, asynchronous interaction from the client is possible e. g. for requesting a status report about the actual configuration of the hardware. The Xputer Runtime System will proceed the execution flow once initiated by the client. The control of the hardware computations is completely transmitted to the server module; only eventual status reports and the "ready" signal is noticed by the client.

Inside the XMDS server module, the control flow can be divided on an abstract level into three phases (figure 10). In a first phase, the necessary data and programs are loaded into the MoM-PDA; this is the configuration period. Next, the computation of the hardware is started and a time stamp is recorded. The XMDS server module waits now until the MoM-PDA reports an "end-of-computation"-signal. In the last phase, the execution time is determined based on the recorded time period, the results are read, and the XMDSClient is informed. If, during the execution flow, the user that made the request, disconnects from the server, the flow is not interrupted; only the final results are not reported to the client.

Both, the XMDS and the Xputer hardware allow a multitasking operation. Nevertheless, the runtime system

has to organize the incoming applications for the MoM-PDA in a sequential way. By allowing only the execution of one application at one time, it is possible to observe an application in isolation and retrieve information concerning the execution time of the particular application. Under multitasking conditions execution time observations could not be associated with a particular application. Therefore the XMDS has to provide a time-management for multiple requests. This is realized by a first-come-first-served strategy on server side.

The status visualization of a running application and the information about the application queue on the server is displayed for the user by a specialized XMDS client module called Dynamic Plug-In (DPI). A concurrent development of client and server parts in approach to the XMDS modularization concept assures a transparent distribution of the local and remote system units.

4. Conclusions

We have introduced an internet-based platform for remote prototyping of applications using reconfigurable accelerators. The paper presented the connection between the XMDS and the Xputer prototype hardware MoM-PDA. This configuration allows to execute applications on the hardware via the Internet. As mentioned in the beginning this enables not hardware/software co-execution of larger software but only the processing of isolated computation intensive applications. Therefore the purpose of the presented Xputer Runtime System is rather testing and demonstration.

Future work will be to think about runtime system for hardware/software co-processing as supported by the CoDe-X (Co-Design for Xputers, [2]) compiler. Such a

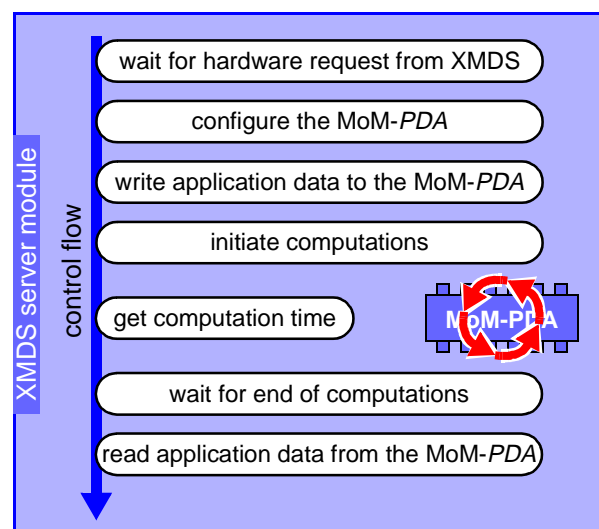


Figure 10. Control Flow of the Runtime System



runtime system will not be connected to a distributed system like presented here. The hardware has to run in multitasking mode accepting all tasks, which are partitioned to the accelerator. Then the Xputer will work like a reconfigurable co-processor.

5. References

- [1] N.N.: Altera 1998 Data Book; Altera Corporation, San Jose, California, USA, 1998.
- [2] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators; Ph. D. Thesis, University of Kaiserslautern, 1997.
- [3] M. Bednara: A Burst Control Unit to Perform Optimized Access to Multibank DRAMs; Diploma Thesis, University of Kaiserslautern, Kaiserslautern, Germany, May 29, 1998
- [4] P. Denyer, J. Brouwers: Java in Electronic Design Automation; Proc. of ASP-DAC'97, Chiba, Japan, Jan. 28-31, 1997.
- [5] F. Gilbert: Development of a Design Flow and Implementation of Example Designs for the Xilinx XC6200 FPGA Series; Diploma Thesis, University of Kaiserslautern, Kaiserslautern, Germany, May 29, 1998.
- [6] R. Hartenstein, M. Herz, T. Hoffmann, U. Nageldinger: Using the KressArray for Reconfigurable Computing; Proc. of Conference on Configurable Computing: Technology and Applications, part of SPIE International Symposium on Voice, Video, and Data Communications (Photonics East), Boston, MA, USA, Nov. 1-5, 1998
- [7] R. Hartenstein, M. Herz, F. Gilbert: Designing for the Xilinx XC6200 FPGAs; 8th International Workshop on Field Programmable Logic and Applications, FPL'98, Tallinn Technical University, Estonia, Aug. 31 - Sept. 31, 1998
- [8] R. Hartenstein, M. Herz, T. Hoffmann, U. Nageldinger: On Reconfigurable Co-Processing Units; Proceedings of Reconfigurable Architectures Workshop (RAW98), held in conjunction with 12th International Parallel Processing Symposium (IPPS-98) and 9th Symposium on Parallel and Distributed Processing (SPDP-98), Orlando, Florida, USA, March 30, 1998
- [9] R. Hartenstein, J. Becker, M. Herz, U. Nageldinger: A Novel Universal Sequencer Hardware; Proceedings of Fachtagung Architekturen von Rechensystemen ARCS'97, Rostock, Germany, September 8-11, 1997
- [10] R. Hartenstein, J. Becker, M. Herz, U. Nageldinger: An Embedded Accelerator for RealWorld Computing; in Proceedings of IFIP International Conference on Very Large Scale Integration, VLSI'97, Gramado, Brazil, August 26-29, 1997
- [11] R. Hartenstein, J. Becker, M. Herz, U. Nageldinger: A Novel Sequencer Hardware for Application Specific Computing; Proceedings of 11th International Conference on Application-specific systems, Architectures and Processors, ASAP'97, Zurich, Switzerland, July 14-16, 1997
- [12] R. Hartenstein, J. Becker, M. Herz, U. Nageldinger: A General Approach in System Design Integrating Reconfigurable Accelerators; Proc. IEEE Int'l Conf. on Innovative Systems in Silicon; Austin, TX, Oct. 1996.
- [13] R. Hartenstein, A. Hirschbiel, M. Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90, Int'l. Conf. memorizing the 30th Anniv. of Computer Society Japan, Tokyo, Japan, 1990.
- [14] M. Herz, T. Hoffmann, U. Nageldinger, C. Schreiber: XMDS: The Xputer Multimedia Development System; Proc. of the HICSS-32, Hawaii, USA, Jan. 5-8, 1999.
- [15] R. Kress: A Fast Reconfigurable ALU for Xputers; Ph. D. Thesis, University of Kaiserslautern, 1996.
- [16] L. Lemay, C. Perkins: Teach Yourself Java 1.1 in 21 Days; 2. Edition, Sams.net Publishing, Indianapolis; IN, USA, 1997.
- [17] N.N.: Siemens Multibank DRAM, Ultra-high performance for graphic applications; Siemens Semicond. Group, Oct., 1996.
- [18] <http://www.vcc.com>
- [19] N.N.: The Programmable Logic Data Book; Xilinx Inc., San Jose, California, USA, 1996.

