An Internet Based Development Framework for Reconfigurable Computing

Reiner W. Hartenstein, Michael Herz, Ulrich Nageldinger, Thomas Hoffmann
University of Kaiserslautern
Erwin-Schroedinger-Strasse, D-67663 Kaiserslautern, Germany
Fax: ++49 631 205 2640, email: abakus@informatik.uni-kl.de
www: http://xputers.informatik.uni-kl.de/

Abstract. The paper presents a development framework for the Xputer prototype Map-oriented Machine with Parallel Data Access (MoM-*PDA*). The MoM-*PDA* operates as a reconfigurable accelerator to a host computer. It utilizes the KressArray as a coarse-grained reconfigurable architecture and implements concurrent data access to parallel memory banks. An Internet-based development framework allows remote prototyping of accelerator applications and enables worldwide access to the prototype via the Internet.

1 Introduction

The increasing interest in the Internet of the recent years results in a growing number of users as well as improved network infrastructure, quality and performance. This is the basis for a new kind of software, which is running via the Internet [5][6]. That means the software is installed on a server connected to the Internet and accessed and executed by a remote client. Such Internet based software provides several benefits. Once distributed by a server, Internet based software is available world-wide and depending on its implementation it can be executed without previous installation on the client machine. Further software updates and patches are not distributed to the users any more. Only the server has to be updated and the new version is available to all users immediately. For commercial use it is not necessary that customers buy the software any more, they download and pay only the modules they need. This pay by use method makes very expensive design software also available to small companies. It is also possible to place a high-performance or application specific computer at the server side and provide computation time for special tasks to the users.

In the area of reconfigurable computing the described techniques allow to provide public access to accelerator prototype hardware and related development software. Since experimental prototypes mostly consist of especially designed components, which are also very expensive, Internet based access increases the number of users efficiently. One of the first approaches, where remote testing was implemented, is WebScope [8][16]. A different approach, which is more going in the direction of operating systems, has been published in [4]. Here reconfigurable hardware is regarded more abstract and its computational resources can be used in a client server based approach.

In this paper a remote prototyping framework is presented, whereas the complete design software is executed via the Internet. For testing of applications a prototype hardware can be accessed remotely. The software part of the framework forms the Xputer Multimedia Development System (XMDS, [14][17][21]). It incorporates the advantages of Internet based software mentioned above. Further the XMDS enables access to the Map-oriented Machine with Parallel Data Access (MoM-*PDA*, [13]) for remote testing.

In the following section the MoM-*PDA* is presented first. After that the XMDS is introduced and it is explained how the prototype hardware is accessed via the Internet.

2 The Map-oriented Machine with Parallel Data Access

In this section the MoM-*PDA* is introduced. It is used as an accelerator for a host computer. The machine architecture is based on the Xputer paradigm [9] and uses the KressArray [13] as a reconfigurable ALU (rALU). To achieve high data throughput possible through parallelism on hardware level, the MoM-*PDA* has parallel access to computation data [13]. Therefore a dedicated memory organization scheme has been implemented.

For the readers convenience the Xputer principles are briefly summarized. The basic operation principles are shown and the prototype MoM-*PDA* is presented.

2.1 The Xputer Machine Paradigm

The basic Xputer architecture consists of a data sequencer, which can be seen as the control part, a 2-dimensionally organized data memory and a rALU (figure 1). All parts of the Xputer are programmable. While the datapath is typically built of reconfigurable devices, the data sequencer is fixed and programmed by only a few parameters for generic address generation [10]. During operations the data sequencer generates an address stream for the data memory and sequences data to the rALU and results back to the memory. The accessed data is passed from the data memory through a smart interface which optimizes and reduces memory accesses. It stores interim results and holds data needed several times. All data manipulations are performed by the rALU.

To clarify how computations are performed an execution model is shown in figure 1. A large amount of input data is typically organized in arrays (e.g. matrix, picture) where the array elements are referenced as operands of a computation in a current iteration of a loop. These arrays can be mapped onto a 2-dimensionally organized memory without any reordering. 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 1). Operations are performed by moving the scan window over the data map

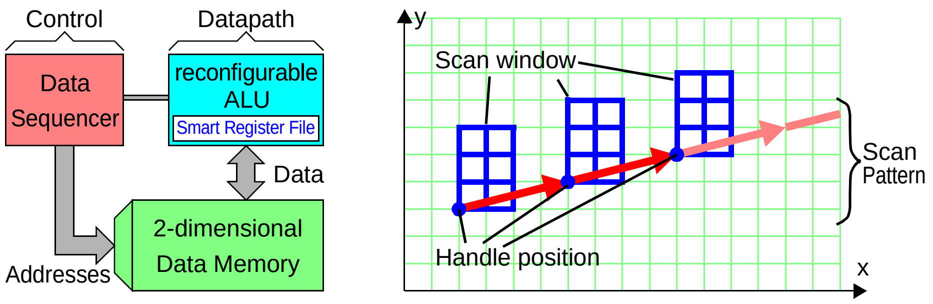

Fig. 1. Basic Xputer Architecture and Execution Model

and applying the configured operator of the rALU on the data in each step. Thus this movement of the scan window called scan pattern is the main control mechanism.

2.2 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. It is based on the Xputer paradigm and utilizes the KressArray [13] for reconfigurable computing. This section gives a short 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 [13]. Therefore it has 2 parallel banks of Multibank DRAM (MDRAM, [18]). Addresses for the MDRAMs are computed by the Data Sequencer and extended with burst information by the Burst Control Unit (BCU, [3]). In the current prototype implementation the Data Sequencer of the MoM-*PDA* is mapped to an Altera FLEX10K100 device [1].

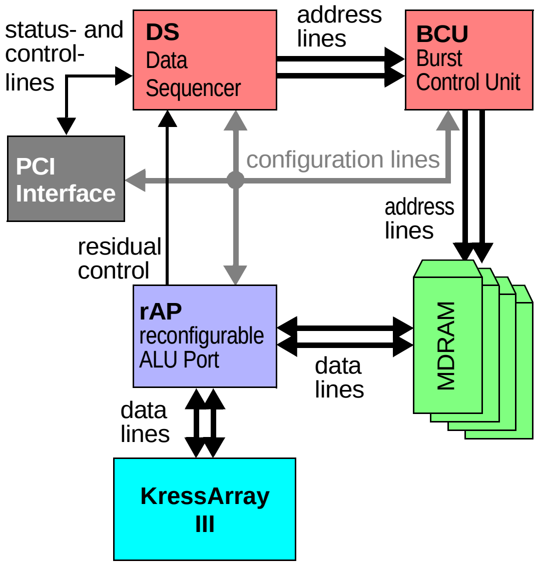

Fig. 2. MoM-*PDA* machine overview

The Data Sequencer handles up to 16 parallel tasks each consisting of a complex scan pattern [10]. Therefore up to 16 scan windows can operate on the data memory concurrently. The computation of the parallel tasks is done like known from multitasking systems. To access several locations in the data memory concurrently the Data Sequencer generates two parallel address streams. Based on these addresses the BCU performs MDRAM accesses. Usually computations are performed by the KressArray. Therefore data is routed to a reconfigurable ALU port (rAP, figure 2). It is implemented with a Xilinx XC6216 [20]. The rAP holds two functional units. One unit is the parallel memory interface for the MDRAMs 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:

- as rALU: Computations are performed directly in the rAP. This allows to build a small version of the MoM-*PDA* without the KressArray. For this an operator library has been implemented, mainly containing image processing applications [7][12].
- as interface to the KressArray: In that case the rAP optimizes data exchange between the KressArray and the data memory.

2.3 MoM-*PDA* Prototype Implementation

Except the KressArray the MoM-*PDA* prototype implementation is based on FPGAs. The complete prototype consists of 3 printed circuit boards.

The interface to the host computer is implemented with a commercially available

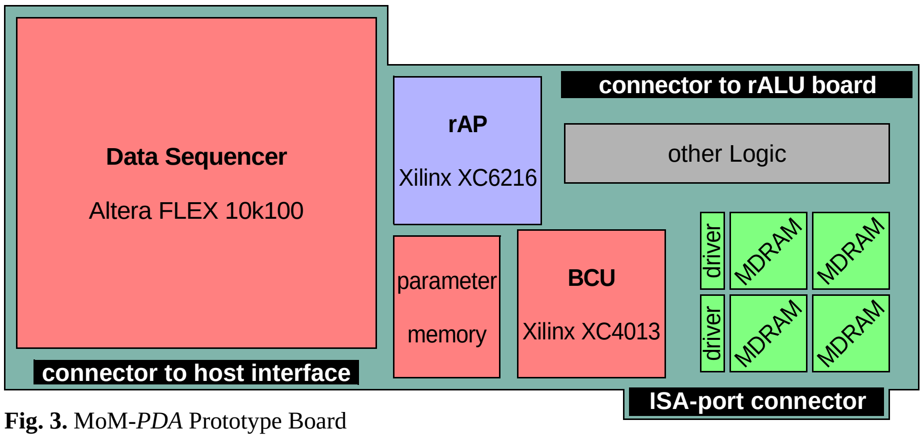

Fig. 3. MoM-*PDA* Prototype Board

FPGA board from Virtual Computer Corporation (VCC, [19]). It contains already a PCI interface and is delivered with a C library. The FPGA is used to implement necessary glue logic. Having the host interface on a separate board allows to use the prototype with different types of host computers.

The main components of the prototype are placed on the MoM-*PDA* prototype board (figure 3). This board contains the Data Sequencer, the reconfigurable ALU Port (rAP) and the Burst Control Unit (BCU). Further 2 parallel banks of 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 has 3 connectors:

- An ISA-port connector is used for power supply.
- A connector to the host interface.
- The connector to the rALU board allows to connect an external board with a KressArray. Since the rAP is able to perform computations, a KressArray is not always needed.

To set up the development framework all components are plugged into a Pentium processor PC running also the Xputer Multimedia Development System (XMDS). The XMDS integrates a special runtime system which enables users to test applications on the MoM-*PDA* via the Internet. This Internet based software is explained in the next section.

3 The Xputer Multimedia Development System (XMDS)

The most important part of the introduced Internet based development framework is the XMDS [14][17][21]. It is an Internet based CAD system which provides a comprehensive toolset for application development. Besides this remote access to conventional Xputer design software [2] written in C programming language is made possible. Further the XMDS enables remote testing of applications by executing them on the MoM-*PDA* prototype.

3.1 XMDS Concepts

The main part of the XMDS system is the XMDS server which holds the complete XMDS system including an user data library, a runtime system for the Xputer hardware and the MoM-*PDA* prototype hardware (figure 4). Parts of the XMDS system are executed on the server and other parts are downloaded on the fly to the users computer and executed there. For communication between these computers a special protocol (XMDS messaging, [17]) has been developed. Further an Unix host is connected to the XMDS server holding conventional design software written in C programming language.

Several concepts had been focussed during the XMDS implementation. In fact, the XMDS is a project that combines the features of a powerful Internet based CAD system with the specific requirements to access the experimental Xputer hardware and the traditional (C-programmed) software. This led to the following specification items. The XMDS should provide:

- a user front-end that is accessible via WWW for several platforms,
- a central user administration,
- a modularity of its components,
- a built-in support for further extensions,
- powerful network capabilities, and
- an embedding of multimedia features.

In the following, the realization of these concepts is described in detail.

WWW front-end and the Client /Server Model

Because the World Wide Web (WWW) is commonly explored with a WWW browser, the user front-end of the XMDS must have a WWW address, like normal WWW documents. By pointing his browser to that particular URL (Uniform Resource Location), 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 WWW browsers. Therefore a Java implementation has been chosen. The fact that a WWW browser forms the environment of the front-end ensures a certain independence against the underlaying operation system (e.g. Windows, MacOS, Unix). Portations of a particular WWW browser to different platforms ensure therefore the availability of the XMDS.

The different components that form the XMDS are realized following the client/server paradigm. Components are distributed in a logical manner on the client and on the server part. Every user front-end which is downloaded by a user forms a client of the XMDS. The user can reach it by requesting a specific URL targeting the computer where the XMDS files are installed. To allow the publishing of URLs, this computer must run a WWW server. The HTML document loaded in the users browser

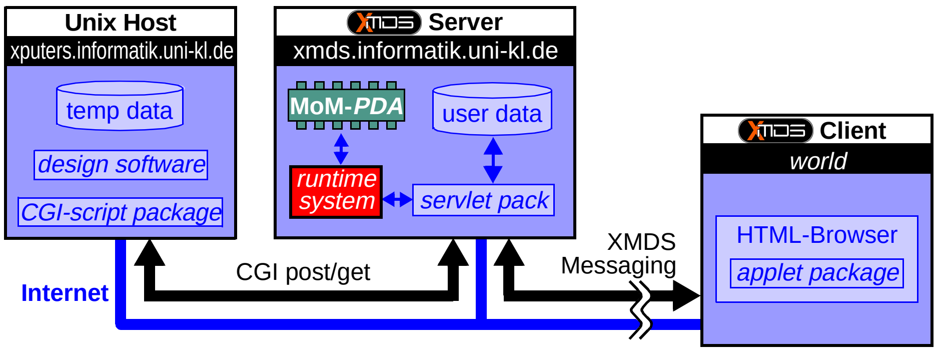

Fig. 4. Overview of the XMDS

window contains a reference to the XMDS client program.

The complete XMDS system is programmed in Java. While the Java applet technology realizes a dynamic download of the XMDS client to a users machine via Internet, the XMDS server makes use of the Java servlet technology to accept the connections from the client and perform the server-side actions.

The advantages of such a dynamic client/server model compared to a static installation of client software are obvious. With this concept, the XMDS

- is available world-wide,
- may be used without previous installation,
- is always available in its latest version,
- downloads only the components that are actually needed by the user,
- enables the users to evaluate the MoM-*PDA* without acquiring the hardware.

This concept realizes an easy way for distribution and provides easy access for interested persons.

Central User Administration

The XMDS server also contains the central user administration: the user database and the user management system.

The users of the XMDS need to store the data, the state and the configuration of their XMDS sessions in order to continue their work in a next session. Because the XMDS-Client has no permission to store data on the client's machine due to the applet limitations [17], all the user-specific data must be stored on the XMDS server. Therefore the users of the XMDS must be registered. To secure intellectual property (IP) all users receive a personal password to protect their data.

Network capabilities

A very close binding between the client and the server is essential for a powerful distributed system like the XMDS. This is realized by the XMDS Messaging system which will not be discussed here. Refer to [17] for a detailed presentation.

An important task involving a network communication is to invoke the design software contained on an Unix host. Because this host runs a Web server, a medium-level communication is possible involving the Common Gateway Interface (CGI) of the Unix host. Because Java applets (like the XMDS client) are not allowed to communicate with an Internet host different from their own home host, it must be the XMDS server that establishes this communication. Therefore the XMDS server provides a module for a comfortable use of the CGI-request-response interaction and extends the communication to the XMDS client using the XMDS Messaging System.

Multimedia User Interface

The XMDS is prepared to cooperate with different media types to support the users in their application development process. It integrates complex animations within the user interface of the XMDS to allow visualization of non-trivial processes which efficiently supports application development.

On a more basic level, a support is integrated for playing audio files. This is useful for standard events like system sounds, or for requesting the user's attention in special situations. By the ability to add explaining speeches to particular key situations, the user is supported actively on his way through the different parts of the development process.

Finally the XMDS client window is able to load HTML documents in the browser window. This allows the construction of a context-sensual help system integrating text, image and hyperlink elements. The on-line-help is in fact expanded to a complete on-line-tutorial providing the most actual help documents available at the XMDS Server.

3.2 The XMDS Runtime Support

The XMDS provides a runtime system to test applications on the real hardware (figure 4). As a distributed system it accesses the Xputer prototype on the server side. The MoM-*PDA* is directly connected to the 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 purpose of the runtime system is testing of applications only. It has no operating system features and does not support host/accelerator co-processing. Further also the multitasking feature of the MoM-*PDA* is not used.

On a low level the runtime system consists 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 a higher an enveloping program realizes the control flow and calls the C functions. In approach to the XMDS modularization concepts [17], the program is implemented as an XMDS server module in Java, and calls the C functions using the Java Native Interface (figure 5). 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.

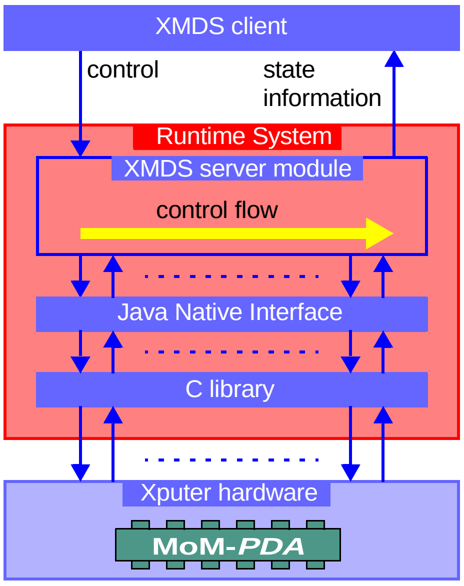

Fig. 5. Overview of the Runtime System

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. All process control comes from this client module. The user specifies the application to be executed and provides the test data. Since the test data is generated on the users machine it has to be transferred to the XMDS server. This is done using the file transfer protocol (ftp). After this preparations a start request is sent to the XMDS server. Besides normal execution the application can also be processed in a trace mode. In that mode the application is executed step by step and for evaluation the internal state of the MoM-*PDA* is sent to the client and displayed each time.

Incoming applications for the MoM-*PDA* are organized in a sequential way. Only if one application is executed at a time precise measurement of execution time is possible. Therefore the XMDS integrates a time-management for multiple requests. This is real-

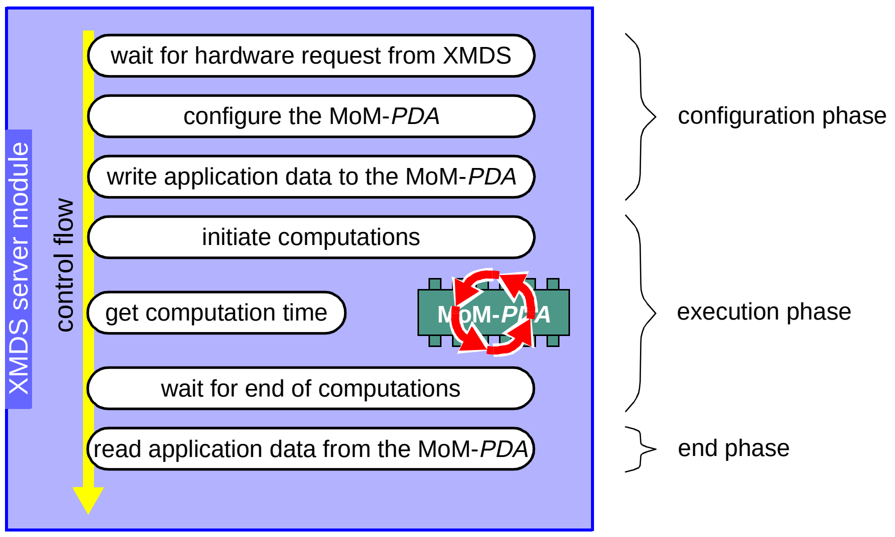

Fig. 6. Control Flow of the Runtime System

ized by a first-come-first-served strategy. The server holds a list of waiting jobs to be executed. This list can be read by the clients. In case that the hardware is currently used by another client, a started application is appended to the list of waiting jobs. Then the client has to wait until all earlier applications are processed or he may cancel his request. If the hardware is not used when an application is started, it is immediately executed.

Before an application is really executed on the hardware, the hardware has to be configured first. Also the test data has to be stored in the data memory of the MoM-*PDA*. This is done automatically by the runtime system. If the application is invoked in normal execution mode a time stamp is recorded to measure the execution time. During normal execution the user has to wait until computations have finished. The only possible interaction is to interrupt the execution. In that case the user can select between canceling the application or restarting it in trace mode. If the execution of the application is started or resumed in the trace mode the user can watch all internal processes of the hardware but no measurement of overall execution time is possible. At the end of computation the state of the hardware, the results of computation and depending of the execution mode the execution time is transferred to the client.

Inside the XMDS server module, the control flow can be divided on an abstract level into 3 phases (figure 6). 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. In normal execution mode a time stamp is recorded and the XMDS server module waits 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 XMDS client is informed.

If the hardware is invoked in the trace mode client and server are working interactively. The hardware stops each time the data sequencer generated a handle position (figure 1). The internal state of the hardware is transmitted to the client and displayed to the user. The user may manipulate internal registers and send the contents back to the server to reconfigure the hardware. Further the user triggers the hardware to perform computation steps. The user may operate the hardware in single steps, a specific number of steps or by setting a breakpoint on an internal state.

3.3 XMDS Design Flow

Depending on the users knowledge of the hardware principles the XMDS provides different design tools which results also in different degrees of performance. Unexperienced users may use automatic compilation of applications. For this the CoDe-X (Co-Design for Xputers, [2]) framework has been implemented. Because CoDe-X is programmed in C language, it is installed on an Unix host and accessible via the XMDS. There are also several tools implemented for application development on expert level. Figure 7 shows the XMDS client window with a clickable design flow on expert level. On this level there are 3 different possibilities to edit an application. Besides a specialized programming language (MoPL, [2]) a graphical task editor and a hardware level parameter editor are available. The design validation phase is supported by simulation tools animating the computations on a hardware model as well as on the execution model (figure 1). The Application Analyser calculates the exact runtime of an application. After design validation it can be downloaded to the users computer or executed with test data on the MoM-*PDA* prototype.

4 Summary

A development framework for reconfigurable accelerator applications has been introduced. The framework consists of an Xputer prototype (Map-oriented Machine with Parallel Data Access, MoM-*PDA*) and an Internet based development software (Xputer Multimedia Development System, XMDS). The XMDS provides a set of development tools for the MoM-*PDA* and supports also remote execution of other design software. Further the XMDS enables testing of applications on the MoM-*PDA* prototype hardware without acquiring the hardware.

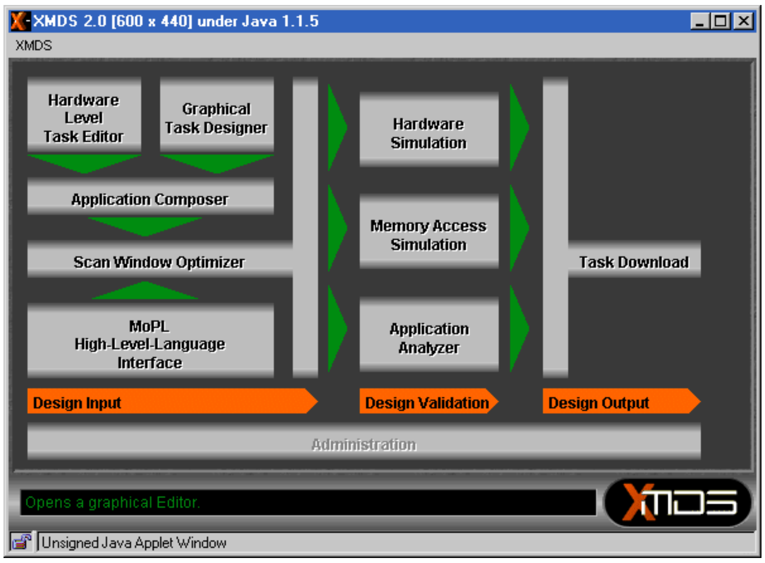

Fig. 7. XMDS client window showing a selection of tools as a design flow for expert level application development

References

[1] N.N.: Altera 1998 Data Book; Altera Corporation, San Jose, CA, 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, May, 1998

[4] G. Brebner: Circlets: Circuits as Applets; FCCM'98, Napa Valley, CA, USA, 1998.

[5] P. Denyer, J. Brouwers: Java in Electronic Design Automation; Proc. of ASP-DAC'97, Chiba, Japan, Jan. 28-31, 1997.

[6] L. Geppert: IC Design on the World Wide Web; IEEE Spectrum, June 1998.

[7] F. Gilbert: Development of a Design Flow and Implementation of Example Designs for the Xilinx XC6200 FPGA Series; Diploma Thesis, University of Kaiserslautern, May, 1998.

[8] S. Guccione: WebScope: A Circuit Debug Tool; FPL'98, Tallinn Technical University, Estonia, Aug. 31 - Sept. 31, 1998

[9] 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.

[10]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

[12]R. Hartenstein, M. Herz, F. Gilbert: Designing for the Xilinx XC6200 FPGAs; FPL'98, Tallinn Technical University, Estonia, Aug. 31 - Sept. 31, 1998

[13]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.

[14]M. Herz, T. Hoffmann, U. Nageldinger, C. Schreiber: XMDS: The Xputer Multimedia Development System; HICSS-32, Hawaii, USA, Jan. 5-8, 1999.

[15]M. Herz, T. Hoffmann, U. Nageldinger, C. Schreiber: Interfacing the MoM-*PDA* to an Internet-based Development System; HICSS-32, Hawaii, USA, Jan. 5-8, 1999.

[16]D. Levi, S. Guccione: BoardScope: A Debug Tool for Reconfigurable Systems; 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.

[17]C. Schreiber: Design of an Internet Based Development System for Xputers in Java, Diploma Thesis, University of Kaiserslautern, July 1998.

[18]N.N.: Siemens Multibank DRAM, Ultra-high performance for graphic applications; Siemens Semicond. Group, Oct., 1996.

[19]URL: http://www.vcc.com/

[20]N.N.: The Programmable Logic Data Book; Xilinx Inc., San Jose, CA, USA, 1996.

[21]URL: http://xmds.informatik.uni-kl.de/