XMDS: The Xputer Multimedia Development System

Reiner Hartenstein, 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 development system for Xputers, which combines an application development framework with a runtime support to run applications on the real hardware. The key idea of the Xputer Multimedia Development System (XMDS) is to provide worldwide, platform independent access to Xputer software and hardware. Therefore the XMDS main system is running via the World Wide Web and is implemented in JAVA language to run on all common Web browsers.

1. Introduction

In the last years the use of the Internet grew up rapidly. Most computers are already connected to the Internet and because of the increased interest also the networks are improved. High performance networks allow the development of a new kind of software, which is running via the Internet [2]. That means the software is installed on a server connected to the Internet and accessed by a 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, which need the software seldom. 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.

These considerations led to the implementation of the Xputer Multimedia Development System (XMDS, [12]). The most important aspect during implementation was to reach as many users as possible. Therefore the XMDS is implemented in the platform independent programming language JAVA [10]. It is installed on a Web server and can easily be accessed by an URL (uniform resource location). On the client side only an Internet connection and a Web browser, which is capable to execute JAVA programs, are needed. The user do not need to install any software and since updates are performed on the server, they are immediately visible to the client.

The special interest of the XMDS was to access an already existent design software and overlay it with an public-access interface. This will ensure the connection to a real prototype hardware and allow a distributed application design. In the following, first the conventional C software is presented followed by the presentation of the XMDS. A case study will demonstrate the use of the XMDS.

2. Xputer Software

A complete software environment for Xputers is implemented in C programming language. The Hardware / Software Co-design framework CoDe-X takes C programs as input and partitions it on a host and an accelerator, based on the Xputer paradigm. The XMDS extends this framework by providing access to it via the Internet. Additionally multimedia tools are implemented as an extension.

To explain the software needed for Xputers, first the Xputer paradigm is briefly introduced. After that, the traditional Xputer software environment is explained and

the extensions by the XMDS are shown.

2.1 The Xputer Paradigm

The main difference between the Xputer machine paradigm [8, 13 - 15] and von Neumann machines is, that the computer is controlled by a data stream instead of an instruction stream (but it is *not* a data flow machine [5]). 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. 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 in [9] shows all necessary components and their interconnect.

These principles are derived from the fact that many computation-intensive applications iterate the same operations over a large amount of data. Xputers accelerate them 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. The rALU is implemented with the KressArray [4]. Computations are performed by applying a configured complex operator on the data. This 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 only needed when the data stream has to be influenced by the result of previous computations.

To clarify how operations are performed the execution model for Xputers is given in figure 2 in [9]. 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 can be mapped onto a 2-dimensional organized memory. This arrangement of data is called data map. The part of the data memory which holds the data for the current iteration is determined by a so called scan window. The scan window is a model for the smart interface which holds a copy of the data memory. Each position of the scan window is marked as read, write or read and write. The location of the scan window is determined by the lower left corner, called handle (see figure 2 in [9]). 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 of an Xputer.

2.2 The Xputer Compiler

After the introduction of the Xputer architecture, the software development framework will be described. A co-design compiler for a host/Xputer system must generate the following parts out of an input program:

- Code for the host,
- Parameter sets for the Data Sequencer,
- A configuration for the reconfigurable ALU (rALU),
- A mapping of the application data to the two dimensional data memory of the MoM.

The basic idea of the compilation will be explained on the example program shown in figure 2. The line numbers of the source code are only for explanation purposes. The program declares two arrays with two dimensions each, and loads data into these arrays by an external function. Then, a calculation is performed on the data using two nested loops. After that, the results are written back, by another external function.

Figure 2 illustrates the basic principles of co-compilation for a host/Xputer based accelerator. The example program is partitioned into several parts. The parts with operating system calls or dynamic structures must be mapped onto the host. The calculation in lines 10 to 18 is very regular and suits well for execution on the accelerator. Hereby, the data arrays (lines 3 and 4) are mapped to structures in the data memory, the loop guards are transformed to Data Sequencer parameters, and the loop body is mapped to a structure on the rALU.
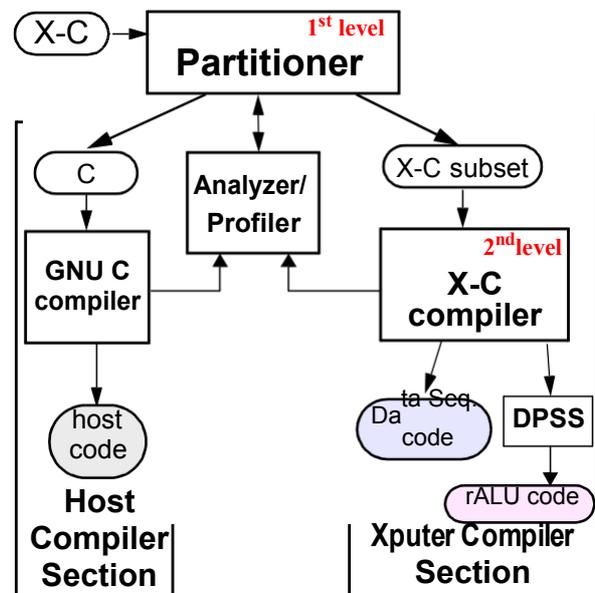

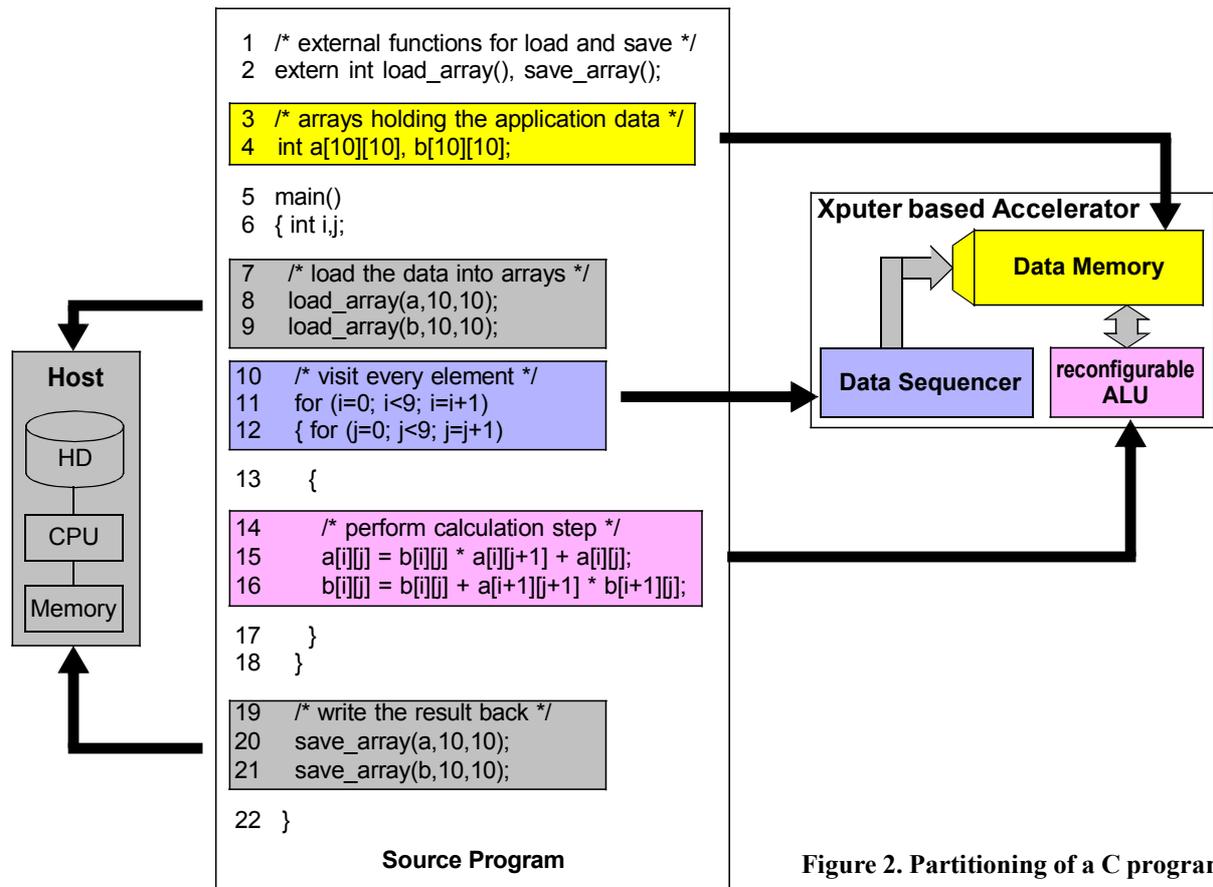**Figure 1. Xputer Compiler Framework without the XMDS integration**

```
1   /* external functions for load and save */
2   extern int load_array(), save_array();

3   /* arrays holding the application data */
4   int a[10][10], b[10][10];

5   main()
6   { int i,j;

7      /* load the data into arrays */
8      load_array(a,10,10);
9      load_array(b,10,10);

10     /* visit every element */
11     for (i=0; i<9; i=i+1)
12     { for (j=0; j<9; j=j+1)

13     {

14        /* perform calculation step */
15        a[i][j] = b[i][j] * a[i][j+1] + a[i][j];
16        b[i][j] = b[i][j] + a[i+1][j+1] * b[i+1][j];

17     }
18     }

19     /* write the result back */
20     save_array(a,10,10);
21     save_array(b,10,10);

22  }
```

Source Program

Figure 2. Partitioning of a C program

Xputer based Accelerator

Data Memory

Data Sequencer

reconfigurable ALU

Host

HD

CPU

Memory

Figure 1 gives an overview about the CoDe-X compiler framework [1]. The input is specified in X-C, an extension to C. The X-C program is partitioned in two levels. The first level is profiling driven and filters parts for the Xputer and parts for the host. The second level of partitioning is resource driven and filters the parts for the Data Sequencer (GAG code) and for the rALU. Hereby, the rALU part is processed by the Datapath Synthesis System (DPSS) [6] [7].

A limitation of the compiler framework described above is, that it is completely programmed in C for Sun SPARC stations running Sun OS. In order to use the framework first a user has to install it. A major problem occurs in that context if the user has not exactly the same unix hardware. Further the integration of multimedia features is difficult with such a configuration, especially when the implementation has to be platform independent.

2.3 The XMDS

Here, the XMDS as a superordinated system, has to access the Xputer software running on a Unix workstation. Therefore communication facilities must be provided. They are realized on the level of the Common Gateway Interface

(CGI) of the Unix host. As the particular Unix host running Sun OS is unable to run a Java Virtual Machine, the CGInterface provided by a running Web server is an alternative that allows a communication on a level that is quite higher than raw socket connections.

As described in the following section, the XMDS will be distinguished in a client and a server part. The server part of the XMDS (in the following called XMDSServer) will provide the connection to the Xputer software by offering an interface for CGI-connections. The XMDSServer is now able to control the execution of the compilers forming the design software and to coordinate them with the requests of the client. Client and server of the XMDS are communicating in a much more sophisticated way as would be possible by a CGI connection. This high-level communication is realized by the XMDS Messaging System [12].

So, if during the application development process, the intervention of the Xputer Compiler is needed, the client of the XMDS delegates the control flow to its server, which itself now contacts the Unix Host, starts the Xputer Compiler and awaits the result of the computation.
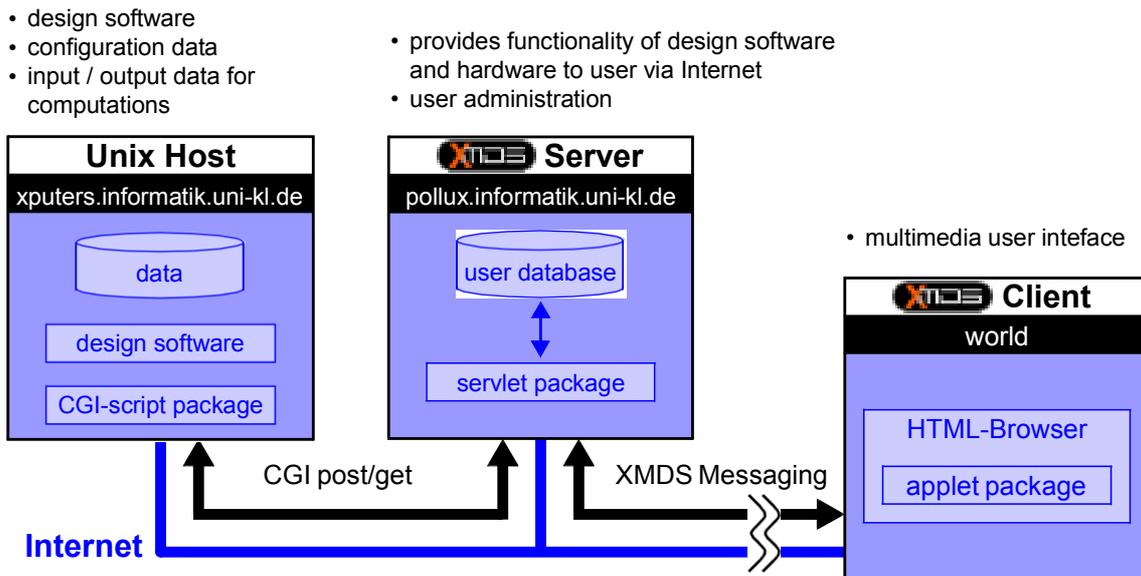
Figure 3. Overview of the XMDS.

In a similar way, the Xputer Runtime System [9] that is implemented on the XMDSServer is initialized by the client and involves the Xputer hardware [3] via PCI. All intermediate synchronization between the Runtime System and the hardware is done locally and does not involve the client. The server informs the client in the following about the results of computations. During this process, the server transmits only the results to the client that are of value keeping the amount of data for the client as low as possible. This is especially important in case of low network bandwidth to the client, e.g. via modem.

The XMDS runtime interface to the hardware supports only testing/execution of applications on the real hardware. Hardware/software co-execution of applications like generated by CoDe-X is not supported by the XMDS.

An overview on the involved hosts and their relations is shown in figure 3.

3. XMDS Concepts

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

- an embedding of multimedia features, and

- a specially developed design for the user front-end.

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

3.1 WWW front-end and the Client /Server Model

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, like normal Web documents. By pointing his 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 underlaying operation system (Windows, MacOS, Unix,...). Portations of a particular Web 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: it is called XMDSClient. 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 Web server. The HTML document loaded in the user's browser window contains a reference to the XMDSClient program.

The visible parts of the XMDSClient are a starting-button inside the browser area and one or more windows. The button is embedded in an HTML document and can be seen as the 'hook' of the XMDSClient to the browser (figure 4).

XMDSClient and XMDSServer are programmed in Java. While the Java applet technology realizes a dynamic download of the XMDSClient to a client's machine via Internet, the XMDSServer 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 allows to take notice of the interested users of the XMDS.

3.2 Modularity and Extensibility

The XMDS is an open system: it offers a framework for future developments. These developments are not meant to be restricted on future add-ons to the functionality. The system itself is built up from elementary components which may be replaced or improved in an independent way. This modularity is realized on the client as well as on the server side of the XMDS and is based on precise interface definitions between the modules. In this context, 'system'
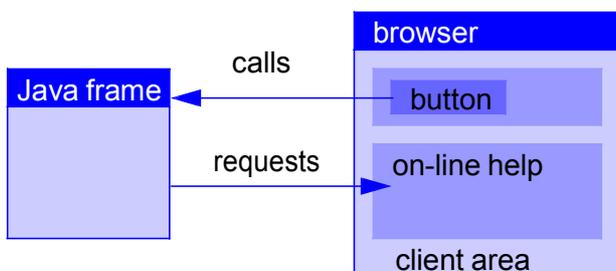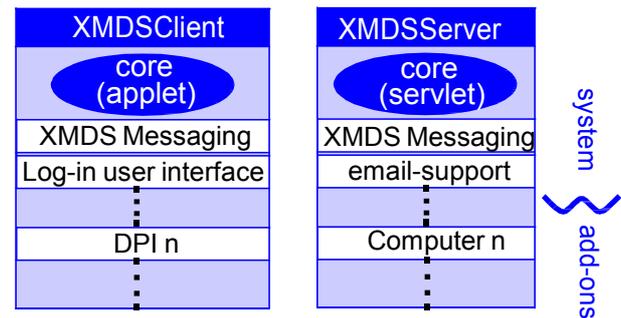

Figure 5. The Modularity of the XMDS.

means the parts of the XMDS which are essential for a correct running. This means the XMDS Messaging System [12], the user administration or the components that compose the user interface of the client. On client side, these components are called 'dynamic plug-ins' (DPI), on server side they are called 'Computers'. Figure 5 shows the modularity. Notice, that the separation between system components and add-ons is not precisely defined. It depends rather on what is said to be the 'essential functionality' of the XMDS.

Extensions increase the functionality of the XMDS and are realized by implementing client- and server-side add-ons and by integrating them within the execution flow of the system. In general, when a new application is prepared to be implemented for the XMDS, such as the Data Sequencer Tools (section 4), the application is distributed on several add-ons, which will cooperate on the client and on the server. Due to precise interface definitions and pre-built methods, this cooperation can easily be established. In that context, the XMDS Messaging System [12] provides methods for a transparent client/server distribution of the add-ons.

As an example of an XMDS extension, the Task Designer is presented in Section 4.1.

3.3 Central User Administration

The computer that hosts the different parts of the XMDSClient also contains the central user administration: the user database and the user management system. The central user administration is realized by the XMDSServer.

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 XMDSClient has no permission to store data on the client's machine due to the applet limitations [10], all the user-specific data must be stored on the XMDSServer. Therefore the users of the XMDS must be registered. To secure intellectual property all users receive a personal password

Figure 4. Coexistence of a Java Frame and a Browser Window

to protect their data. User organization is one of the main tasks of the XMDSServer.

3.4 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 [12] for a detailed presentation.

An important task involving a network communication is to invoke the design software contained on the 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 XMDSClient) are not allowed to communicate with an Internet host different from their own home host, it must be the XMDSServer that establishes this communication. Corresponding to the modularity concept presented in Section 3.2, the XMDSServer provides a module for a comfortable use of the CGI-request-response interaction and extends the communication to the XMDSClient using the XMDS Messaging System. Figure 6 shows the information flow.

In order to contact all the users of the XMDS and to inform them of eventual software or hardware updates, the XMDS has a built-in e-mail module, able to send automatically generated e-mails.

3.5 Multimedia User Interface

The XMDS is prepared to cooperate with different media types to support the users in their application development process for the Xputer hardware.

For documents containing graphical animation and audio clips, Macromedia Flash [11] is a widely accepted file format and a quasi-standard. This is confirmed by the fact that Flash players are now integrated in the latest version of

Netscape's Navigator browser and pre-installed with every copy of Microsoft's Windows 98. A conversion of the Flash format to Java is performed by the authoring tool Macromedia Director in version 6.5 [11] or newer. The ability to integrate complex animations within the user interface of the XMDS allows to visualize some non-trivial processes that are essential to understand for the 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, etc. 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 XMDSClient 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 online-help is in fact expanded to a complete online-tutorial providing the most actual help documents available at the XMDS Server. Here the 'open' aspect of the XMDS is confirmed by integrating the hypertext features of the World Wide Web.

4. Case Study: Data Sequencer Tools

In this section the benefits of the multimedia user interface are demonstrated with some tools for experienced users to program the Data Sequencer (see section 2.1). Since the Xputer compiler introduced in section 2.2 generates all necessary parameters for the Data Sequencer, users are not forced to program it on low-level. But to achieve the highest speed-up and to benefit from all hardware features, it is nevertheless recommended. All Data Sequencer features can be used by the graphical tools.

To access the Data Sequencer tools, the user has to login to the system. The login is necessary to access the password
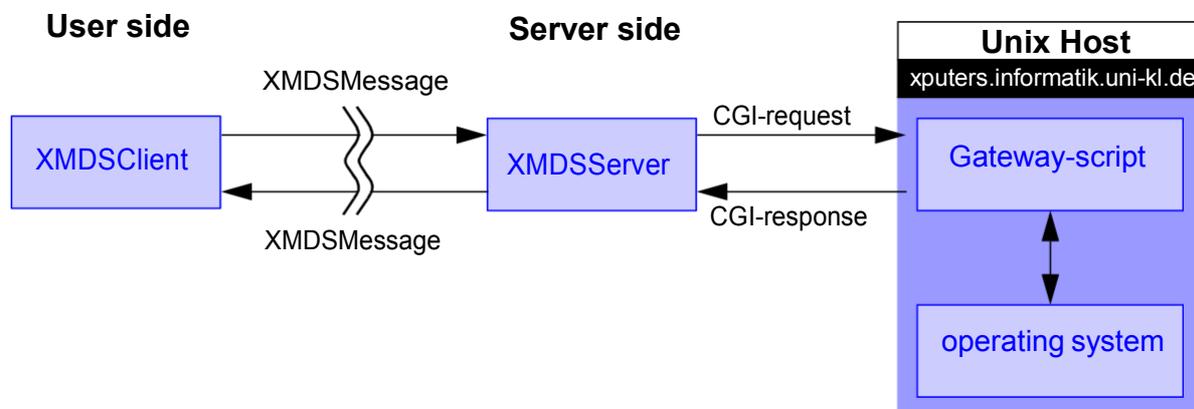

Figure 6. The Information Flow Involving the Common Gateway Interface (CGI).
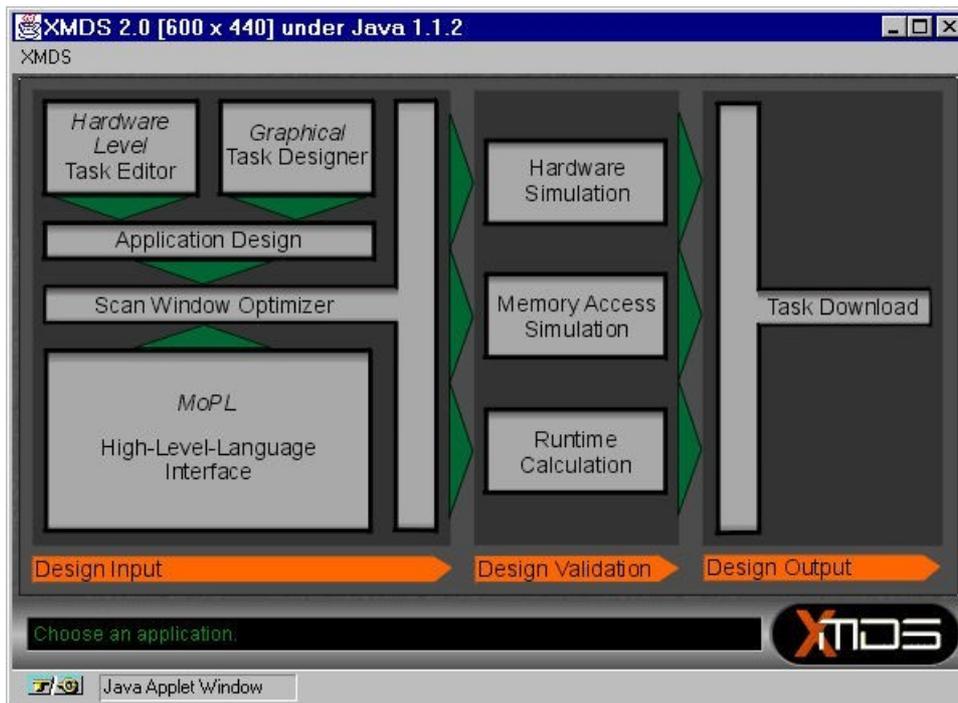
Figure 7. XMDS Window with Application Chooser

protected user database. The XMDS window illustrates a design flow (figure 7) for low-level Data Sequencer programming. By clicking on a design step, the corresponding tool is launched.

There are 3 ways for low-level design input for the Data Sequencer. First the behavior of the Data Sequencer can be described in the MoM Programming Language (MoPL). Since MoPL programs can also be included as an extension to C programs [1], this tool is entitled 'High-Level-Language Interface'.

The 'Hardware Level Task Editor' [12] allows to edit directly the byte code, which is used to program the Data Sequencer. This tool is only for users, which are very familiar with the hardware.

The 'Graphical Task Designer' allows to program the Data Sequencer with an illustration of the execution model (figure 2 in [9]).

While the MoPL programming interface allows to specify complete applications, with the Task Editor and the Task Designer the user can only program single tasks. These tasks have to be combined to complete applications using the Application Designer.

The last step of the design entry phase is a scan window optimizer, which allows to schedule the inner scan window memory accesses in order to optimize the overall memory access time. Although this step is optional, it is highly recommended. Even the automatic compilers perform such a scheduling of the memory accesses [1].

The design validation step provides two tools to simulate the memory accesses of applications. One tool simulates the events inside the Data Sequencer hardware. The second tool simulates the memory accesses with the execution model (figure 2 in [9]).

Because the data sequencing is based on a deterministic approach, the number of memory accesses for an application is known exactly. With real hardware parameters of the prototype, the runtime can be calculated. For hardware designers concerned with the implementation of the Data Sequencer, this tool allows to play with the hardware parameters to find out what parts of the hardware are most time critical for a specific application.

Since the presented design flow focuses only on the programming of the Data Sequencer, an application download to the hardware is not included at this point. Nevertheless the bytecode for the Data Sequencer hardware may be downloaded.

In fact, all these tools can also be applied to Data Sequencer programs initially generated by other Xputer software.

4.1 The Task Designer

As one of the most impressive tools, the task designer makes extensive use of the multimedia features of the XMDS. By pressing its button in the Application Chooser (figure 7) the Task Designer front end is loaded from the XMDS server to the client machine.

Figure 8. Startscreen of the Taks Designer

The startscreen of the Task Designer is shown in figure 8. Several icons simplify the user interaction. To start working with the Task Designer, first a new task must be specified or some existing task must be loaded.

Designing of tasks is separated in 3 design steps. First the

size and shape of the scan window must be specified and the operations inside the scan window are determined. Figure 9 shows the graphical user interface of the scan window editor. The main element is an illustration of the scan window, which can be modified in its size. The kind of
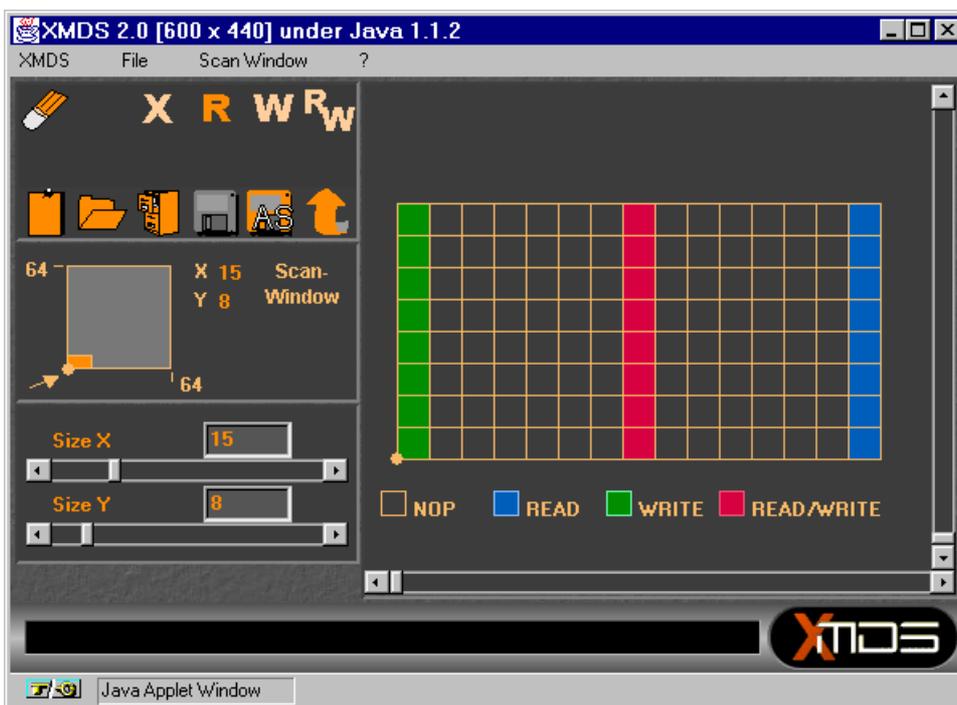

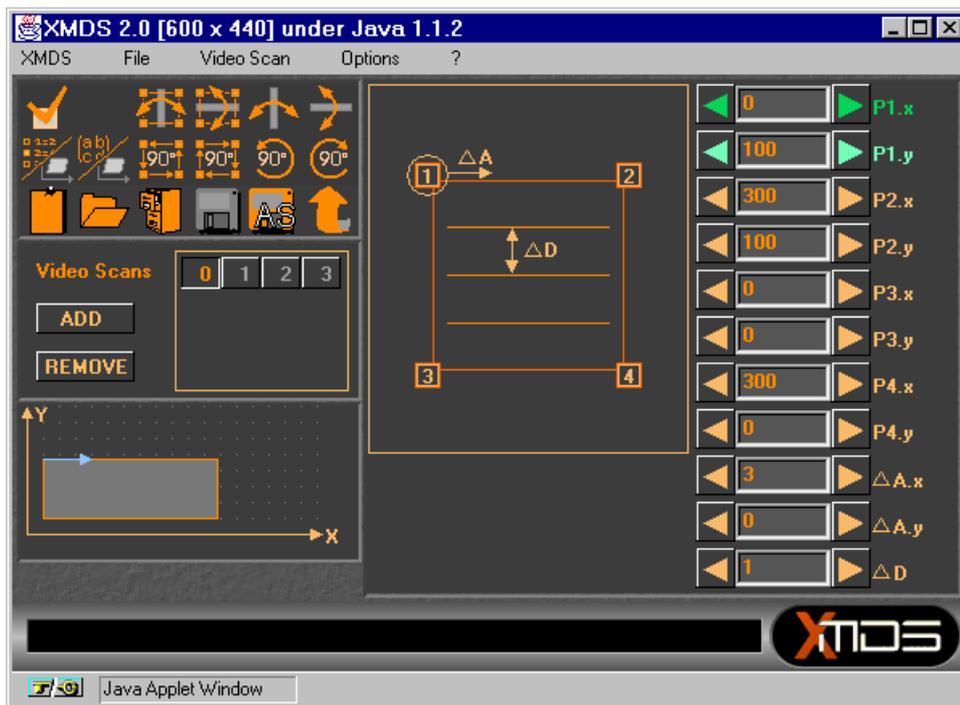**Figure 9. Scan Window Editor of the Task Designer**

Figure 10. Video Scan Editor of the Task Designer

performed operation is set by clicking with the mouse into scan window positions.

Secondly, video scans are defined to be used to move the scan window over the 2-dimensional memory. In fact video scans can generate single addresses, linear scans or may cover a triangle or polygon by regular movements. Therefore only the covered area and the stepwidths have to be determined. Consequently between one and four corner-points have to be specified. A small window in the lower left part of the video scan designer (figure 10) shows the shape of the covered area. Several graphical aids and a schematic representation of a video scan in the center of the window help to specify the necessary parameters.

The Task Designer is further equipped with a parameter test function. This check function assists the user to find errors as well as it outputs a test report in the Web browser's window. Figure 11 shows an example of a correct video scan tested by the check function. It reports the shape of the covered memory area and all hardware parameters necessary for the Data Sequencer to generate the memory accesses.

If the user enters parameters which result in a corrupted video scan, the check outputs an error report (figure 12) describing the reason for the errors as well as possible, i.e the quality of the error report depends on the degree of the errors.

Video-Scan No. 0

Video-Scan is rectangle.

FAULTLESS

· B0.x:	0	· B0.y:	100
· L0.x:	300	· L0.y:	100
· dA.x:	3	· dA.y:	0
· dB.x:	0	· dB.y:	-1
· dL.x:	0	· dL.y:	-1
· F.x:	0	· F.y:	-100
· C.x:	0	· C.y:	-100
· stopAt:	0	· schedule:	1
· SMode.x:	0	· SMode.y:	0

Figure 11. Test Report of a Video Scan

Video-Scan No. 1

Video-Scan is single point.

ERRORS

∘ [4] delta A specified, must be 0

Figure 12. Error Report of a Video Scan

5. Conclusions

We have introduced an Internet-based development system for Xputers. Therefore Java has proven as a very good selection for implementation. During an intense

occupation with the user interface design of the XMDSClient, several inconveniences have occurred. They are mainly due to the defectiveness of the Java Abstract Window Toolkit (AWT). The platform independence - a major advantage of the Java language - is not yet guaranteed if graphical elements get involved. Maybe the adoption of a unique 'look and feel' for the GUI components on every platform will avoid these problems in future releases of the Java API.

In contrast, the 'non-visible' parts of the Java API show an impressive reliability. The Java network package makes it easy to develop a comprehensive distributed application like the XMDS. In addition, the perfect complement of applets and servlets reveals Java as a serious alternative to 'conventional' languages as C/C++ for network based low and medium scale applications. Due to the URL class and the possibilities to communicate with the Web browser, a Java applet can make use of the hypertext paradigm of the World Wide Web to integrate information and multimedia items, as realized for the XMDS help system.

Unfortunately, the main Web browser manufacturers are always in some delay to the actual Java API standard. As a consequence, the developers of a project like the XMDS have to consider that new API features aren't immediately embedded in the newest browsers *and* that not all the users take advantage of the newest browser. Further the programmers of the Web browsers have often different interpretations of the Java standard.

All this leads to the conclusion that although Java is specified as platform independent a lot of software validation an different platforms has to be performed before Java programs really work. But the efforts are worth of because of the benefits of the Java implementation. Real Internet based software can be programmed much easier compared to other programming languages because of the sophisticated network support.

6. References

[1] J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators; Ph. D. Thesis, University of Kaiserslautern, 1997.

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

[3] 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

[4] 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

[5] 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.

[6] R. Hartenstein, et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; Asia and South Pacific Design Automation Conference, ASP-DAC'95, Nippon Convention Center, Makuhari, Chiba, Japan, Aug. 29 - Sept. 1, 1995

[7] R. Hartenstein, et al.: A Scalable, Parallel, and Reconfigurable Datapath Architecture; Sixth International Symposium on IC Technology, Systems & Applications, ISIC'95, Singapore, Sep. 6-8, 1995

[8] 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.

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

[10] L. Lemay, C. Perkins: Teach Yourself Java 1.1 in 21 Days; 2. Edition, Sams.net Publishing, Indianapolis; IN, USA, 1997.

[11] http://www.macromedia.com

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

[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] R. Hartenstein: Xputer: Rechner nach neuartigen Prinzipien; GI Informatik Spektrum, Dezember 1989,

[15] R. Hartenstein, A. Ast, A.G Hirschbiel, M. Riedmüller, K. Schmidt, M.Weber: Using Xputers as Inexpensive Universal Accelerators in Digital Signal Processing; Bilkent'90 Int'l Conf. on New Trends in Communication, Control and Signal Processing; Juli 1990, Ankara, Turkey,