A Hardware / Software Co-Design System using Configurable Computing Technology

John Schewel
Virtual Computer Corporation
6925 Canby Ave #103
Reseda, California, USA 91335

Abstract

Virtual Computer Corporation has combined the latest reconfigurable component technology with a number of advanced software tools in one easy to use 'system approach' to digital design. This Hardware Software Co-Design Development System contains all the components necessary for configurable computing implementation in one integrated package. This paper will describe the features and use of this system.

1: Introduction

With the rapidly changing demands for quicker turn-around times and higher performance at a lower cost, a development platform that allows for control and debugging of designs in Real-Time with real-data can enhance the engineer's success. The problems of integrating the hardware and software components are greater as the logic density and application sizes increase. A Hardware/Software Co-design Development System is becoming essential to success. Unlike co-design systems of the past, these new platforms must address the needs and skills of both the hardware and the software engineers. As the boundaries between hardware and software blur so do the boundaries between the hardware engineer and software engineer.

The H.O.T. Works Development System features both hardware and software components. The integration of the configurable PCI board and development tools makes it an easy to use platform for hardware / software co-design. Co-verification on the H.O.T. Works Development System is possible with the Real-Time connection between your hardware design and the system environment. The choice of options on both the hardware side and software side of the Development System makes it ideal for rapid product development. You have dynamic control of the communications between digital, structural logic, timing and data. With the use of a custom or third party mezzanine card, one can add analog components to the system.

Configurable Computing Systems are those computing platforms whose architecture can be modified by the software to suit the application at hand. The software code being downloaded into the configurable computer

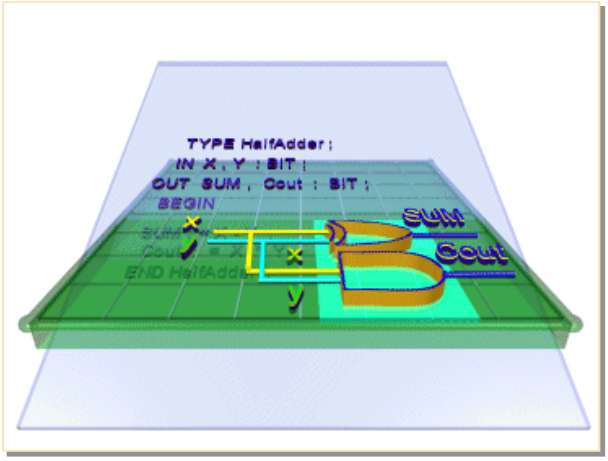

Figure 1 - Hardware Object Technology

is a formatted digital design created for that specific algorithm. The barriers between hardware and software begin to blur when the software can configure the hardware at run-time. This is the H.O.T. Crossover.

This process of altering configurable logic on-the-fly from within an executable program is called **Run-Time Reconfiguration.** The Run-Time Programming of Configurable Computer Systems gives executable programs the power to alter the 'logic' level of hardware to suit its own needs.

Hardware-On-Demand™ is possible with Hardware Object Technology and the Virtual Computer H.O.T. Works PCI Board. VCC's Hardware Object Technology enables the designer to use digital designs with standard 'C' language programs. Your digital design is downloaded from within an application program (as a 'C' language function). VCC's unique implementation of reconfigurablity and ease-of-use allows Real-Time debugging of digital designs.

2: The Configurable Element

The Reconfigurable Processing Unit (RPU) is a new Programmable Logic Device from Xilinx Inc. for Run-Time Reconfigurable Computing. Programmable logic devices, particularly FPGAs, continue to gain momentum over traditional ASICs as the logic solution of choice for today's systems design.

In the last five years, research has shown that reconfigurable logic devices have a great potential in algorithm acceleration within a computing system environment. The multiple run-time reconfiguration use of PLDs is called Configurable or Reconfigurable Computing. Applications such as data mining, image/signal processing and program acceleration, require 'in-system run-time' reconfigurablity, making new demands upon the programmable logic device. While there are many similarities between the architectures of the FPGA and RPU, their differences are significant and should be taken into consideration when evaluating the application at hand.

The RPU is a natural evolution of the very successful Programmable Logic Device Class (Figure 2) and as such should be familiar to users of FPGAs. The primary difference between the RPU and FPGA come from the added features needed for reconfigurable computing. The RPU needed to integrate with the computer's microprocessing unit, operating system and the standard software language applications being used in the marketplace.

There are three unique features in an RPU:

- Open Architecture -- Resulting in 3rd development of tools and compilers
- Dynamically & Partially Reconfigurable Logic -- Enabling Hardware On Demand.
- A Microprocessor Interface -- Configuration times in micro seconds

	MPU	DSP	RPU	FPGA	ASIC
Performance	Limited	Better than MPU	Better than DSP	Near ASIC	Very Fast
Architecture	OPEN	OPEN	OPEN	CLOSED	CLOSED
Programmable	Easily	Programmable	Programmable	Difficult	Not
Development Tools	Lots Low Cost	Good Low Cost	First Low Cost	High Cost	Very High Cost
Who Programs	Software Engineer	Software Engineer	Software Engineer	Hardware Engineer	Hardware Engineer
Use	General Purpose	Embedded	Embedded & Gen. Purpose	Embedded	Embedded
Applications	Lots End Users	Lots Embedded	Embedded & End User	Lots Embedded	High Volume Low Cost

PROGRAMMABILITY ← → POWER

Figure 2

3: The Hardware Component

The Configurable Computer's Architecture

The H.O.T. Works PCI Board offers a standard platform with both fine and course grain reconfigurable components. It utilizes the new XC6200 Reconfigurable Processing Unit (RPU) and the XC4000 FPGA from Xilinx. Inc. The configurable computer board acts as a closely coupled co-processor through the PCI Bus. (Figure 3).

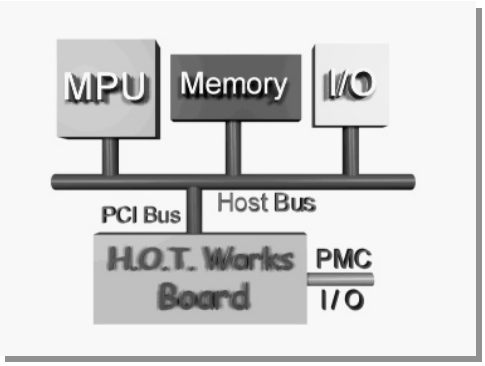

Figure 3

The Board includes one XC6200 RPU, one Field Programmable Gate Array device (XC4000 FPGA Family), 2MB Fast SRAM, On-Board Programmable Oscillator (360 KHz to 100MHz), PCI interface, and PCI Mezzanine Card (PMC) Standard interface connectors for the addition of optional daughter board cards.

The FPGA is used as the PCI bus interface. Approximately 50% of the chip is used for this function and the remaining area is used for card control logic. The FPGA is electrically and functionally 100% PCI compliant. For details of the PCI interface see the PCI LogiCORE product description which is available separately from Xilinx. The compute element is the XC6200 RPU. The board architecture allows the XC6200 to be reconfigured through the PCI interface during run-time. The PCI interface provides direct access from the host PC to logic cells within the user's circuit. The output of any cell's function unit can be read and the flip-flop within any cell can be written through the PCI interface (See Figure 4).

The compute element memory is organized into two banks. Each bank consists of a maximum of two 512K x 8 SRAM's. A bank of RAM can be accessed from either the PCI Interface or the XC6200. The banks of memory have two separate address busses and four read/write signals to control the RAMs individually. The development system provides a flexible architecture in order to implement a wide variety of algorithms. Multiple modes of operation can be set-up by selecting the muxes and bus switch in the desired manner. A 44-bit external data path is available to XC6200 Input Output Blocks (IOB's). This data path can be used to attach daughter boards for video I/O, network connections, or sensor I/O.

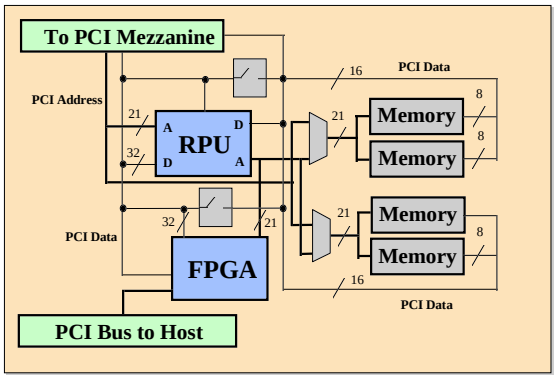

Figure 4

4: The Software Component

With the use of a High Level Hardware Description Languages (HDLs), Hardware Object Technology (H.O.T.) and the PCI plug-in co-processing board, the engineer can begin to use configurable computing techniques for algorithm acceleration, design emulation and rapid prototyping. Implementing and testing designs in Real-Time using real data by configuring hardware from executable programs.

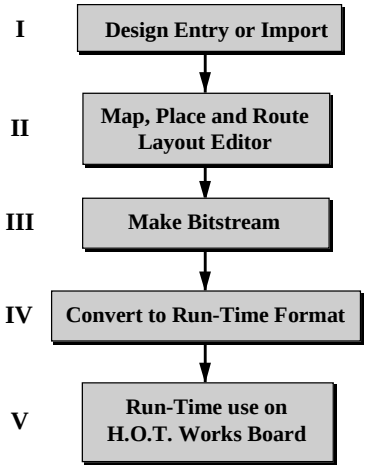

Figure 3

I . Design Entry or Import -- The first step in creating run-time reconfigurable Hardware with the H.O.T. Works Development System is entering a digital design with a design capture program. The Development Software Package contains two different design capture programs. The first is an HDL text editor and compiler called '*Lola*'; the other is a VHDL conversion program called '*Velab*'. Any third party design capture system outputting EDIF 2. 0 .0 netlist files and supports XC6200 libraries (e.g. Viewlogic) can be used to enter your design. The EDIF netlist file is imported directly by *XACT6000* for design implementation (Figure 6). The output of these design capture tools is mapped, placed and routed for the RPU (Step II.).

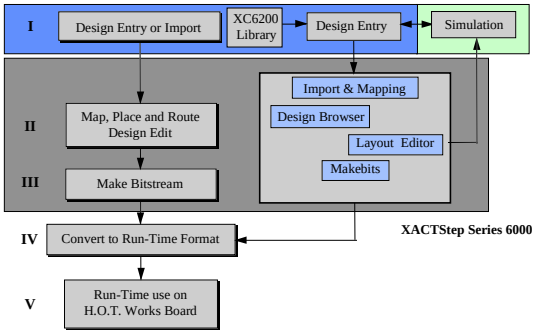

Figure 4

II . Design Implementation -- Once your design is entered, the next step is to map, place and route the design onto the XC6200 RPU. There are two software programs included in the Development System that implement designs.

Velab is a VHDL programming tool for conversion of structural VHDL coded designs into the XC6200 RPU EDIF file format. Velab takes structural VHDL code into EDIF Format file for use with the XACT6000 software.

The *Lola Programming System's XC Editor* imports the compiled *Lola HDL* design into a graphical layout tool for placement and routing. The output of this process includes the making of the RPU bitstream (Step III.).

The *XACT6000* imports EDIF files. The placement and routing may than be viewed and edited in the *Layout Editor* and *Design Browser*. *XACT6000* includes a sophisticated tool that analyzes timing for completed designs. After the design is finished, you generate the RPU bitstream file (Step III.).

III. Make Bitstream -- The bitstream file configures the RPU. The bitstream file has the extension .*CAL* and is generated by user's commands in both the *Lola Programming System* and XCATStep6000.

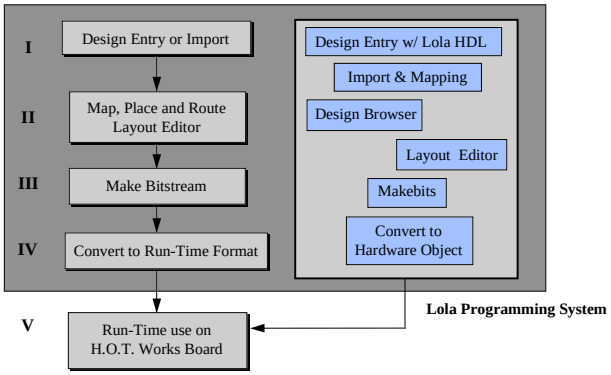

Figure 5

Lola Programming System -- A Hardware Object development system for the software enginner. (Figure 7 Figure 8). The Lola Programming System contains Lola HDL & Lola Compiler, Layout Editor, Circuit Checker, a technology mapper, a placer & router, and a bit-stream generator.

Lola Programming Flow

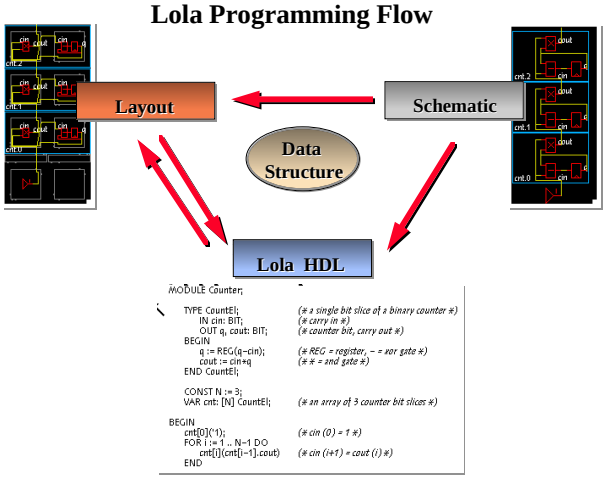

Figure 6

IV. Convert Bitstream File into Run Time Program Mode. -- The Development System supports two methods for Run-Time Reconfiguration.

Method one loads the CAL file (your digital design) from the hard disk to the Board via a program command. This method requires use of C++ support software. These C++ files contain routines to support Low-level board interface, plug and play support, Device configuration support, Runtime support and Debug support.

The other method compiles the design into the executable program. The downloading of the design occurs at program execution time. This method requires

the conversion of the *.CAL* file to a Hardware Object . It also requires C++ routines for control of the Board. This is VCC's **Hardware Object Technology** Method of Run Time Reconfiguration.

Hardware Objects are converted digital designs called from within a software program. The following describes a technique for creating Hardware Objects and embedding them into a compiled 'C++' program. You can reuse the Hardware Object over and over or in combination with other Hardware Objects. An integrated software driver and bus interface gives flexibilty and freedom from bus protocols. You need only convert the digital design into a Hardware Object to enable its use in an application programming environment.

After the final placement and routing of your design the resulting CAL file is converted into a **.h** file to be included within a 'C++' program application. Once the design has been converted by a program called **cal2h** and inserted into an application, the Hardware Object is ready for use. The H.O.T Conversion Program takes the `<filename>.CAL` and outputs `<filename>.h` The **.h** file must be included in your program for compilation with:

> **#include "<header file name>"**
> The file name is the array name of the design - created by the **cal2h** program used earlier in the design development cycle.

This Hardware Object downloading routine is used to load the Hardware Object.

> **void loadHOT(int *);**
> For example: **loadHOT(my_design)** downloads the Hardware Object called **my_design** into the H.O.T. Works Board.

With the above two commands and the other C++ routines provided, you can use your design from within your application program.

6: The Application Interface

The following classes provided a C++ interface to the board. The classes allow the user to interface to the PCI card from their own C++ code with a few simple function calls. The code uses the hotworks.vxd device driver to interface to the PCI board. The three classes making up the XC6200DS interface are:

XC6200DS is the top level class and provides a complete interface to the Development System. This class contains all the 6200 chip access functions, including the reading/writing of the control registers and loading and accessing the user design. The following are examples of XC6200DS functions:

> XC6200DS(XCAddr::DeviceDie); - initializes sets up the selected chip architecture.
>
> int loadCalFile(const char *filename); - This function is used to load a CAL file into the 6200.
>
> void setColumn(byte column, word data); - writes data to a user-defined register in the 6200.

The PCIBoard is a subclass of the XC6200DS class. This class contains the board IOSpace register access functions.

> PCIBoard(); - initializes the board.
>
> void clockOn(void); - writes to continuous clock mode location to put the clock on.
>
> word getCon(void); - reads the "CON" port of the 6200, thus allowing direct reading of data from the 6200

The supporting low level class PCICore handles the actual reading and writing to the PCIBoard as a memory mapped device. This class contains all the necessary code for interfacing to the device driver.

> PCICore(); - Initialises the device driver and the memory map interface
>
> void write6200(word addr, word data); - writes to the 6200 chip on the board
>
> word read6200(word addr); - reads the 6200 chip on the board
>
> void writeRAM(word addr, word *data, word count); reads and writes to SRAM

Figure 9 shows the Initialization of the H.O.T. Works Board and the Design File (CAL) downloading C++ functions.

C++ -- Notes

```
main(int argc, char* argv[])
{
  Pci6200 *board = new Pci6200();

  ...

  board->initialize();
  board->clock_on();
  board->reset();
  ref_freq = 16.0;
  des_freq = 66.0;
  board->set_ref(0); /* use PCI clock */
  freq = board->set_clock_freq(16.0, 66.0);

  if (board->load_cal_file(CALFILE))
  {
      cout << "Problem loading CAL file" <<  endl;
      exit(1);
  }
  board->set_bus_width(32);
  board->set_mask(all32Bits);
```

Create new Pci6200 C++ object

Intialize & define clock

Load Bitstream File

Figure 9

Figure 10 shows The Read and Write C++ functions.

C++ -- Notes

```
for(int i = 0; i < 1000; i++)
{
  // Generate 4 random nos (11-bit)
  a = rand()%2048;
  b = rand()%2048;
  x = rand()%2048;
  y = rand()%2048;

  // Write these data input registers
  board->set_map(all32Bits, noBits);
  board->set_column(multin, a + (x << 16));
  board->set_map(noBits, all32Bits);
  board->set_column(multin, b + (y << 16));
  // Read back the answer
  board->set_map(all32Bits, noBits);
  c = (board->get_column(addout)) & 0x007fffff;

  fout << a << "*" << x << " + " << b << "*"
       << y << " = " << c << endl;
```

Write

Read

Figure 10

7: Conclusions

The H.O.T. Works Development System is the first Hardware/Software Co-Design in one integrated package. The combination of a configurable hardware platform (based upon industry standards), software development tools (based upon proven design flow methods) and an API library (based upon C++ code standards) provides both the experienced hardware engineer and software engineer with a viable tool to explore the trade-offs between hardware and software implementation of application ideas.

Over 300 H.O.T. Works Systems are in use worldwide. Application implementations include motion JPEG at 10X the performance of a Pentium 233MHz implementation and a genetic algorthim optimizer at 1000X the performance of a Pentium 233MHz implementation to name but two. Both hardware and software enginners are currently using this Development System. Users include universities, government research groups, electronic systems companies and telcom companies.

VCC's goal is to offer a simple but powerful standard configurable computing platform for those interested in breaking the bondries between hardware and software.

Further work on the integration of debugging tools, macro libraries and alternate types of design entry systems is continuing. VCC intends to bring to the commerical market a next generation Configurable Computing based Hardware/Software Co-Design using the latest in reconfigurable technology. We hope to develop inexpensive systems with 100,000+ gate capacities by the end of this year.

Biliography

The XC4000 Data Book Xilinx, Xilinx Inc. August 1992

The XC6200 Data Sheet Xilinx, Xilinx Inc.1997

S. Casselman
Virtual Computing and the Virtual Computer™, Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, April 1993.

S. Harbison & G. Steele Jr.
Programming in C, Hayden Book Company, 1991.

Stephen Kochan.
C A Reference Manual Third Edition, Prentice Hall, 1991.

John Schewel, Steve Casselman
HOT Users Guide, Virtual Computer Corp., Sept. 1997.