Virtual Computing and The Virtual Computer

Steven Casselman
Virtual Computer Corporation
Reseda, CA 91335
E-Mail: sc@vcc.com

Abstract

Virtual Computing is an entirely new form of supercomputing that allows an algorithm to be implemented in hardware. Based on the Xilinx FPGA[1] *and ICubes FPID*[2] *the **Virtual Computer** is completely reconfigurable in every respect. Computing machines based on reconfigurable logic are hyper-scalable meaning they scale up better than 1-1.*

We have developed a completely reconfigurable hardware supercomputer that we call the Virtual Computer using advanced FPGA devices. Current computer architectures which have hardwired Central Processing Units, do not have the flexibility to respond to the ever increasing number of algorithms and computing paradigms. These new concepts are changing the way we think about computing as we head toward the twenty-first century. At VCC, we have designed a new type of computer that does not rely on the CPU as it's programmable element, but uses massively reconfigurable logic as its main computing engine.

Concept

Virtual Computing is a new paradigm for computing that involves the analysis of an algorithm to identify its computationally intensive inner-loops and then implementing those inner-loops in completely reconfigurable hardware. The hypothesis behind Virtual Computing is that for any algorithm to achieve maximum through-put, it should be implemented in hardware. The Virtual Computer was developed under Naval SBIR Phase II contract number N60921-91-C-A330.

Hardware Overview

The Virtual Computer features fifty two Xilinx 4010 FPGAs, twenty four ICUBE Field Programmable Interconnect Devices (FPID) , three 64-bit I/O ports - one for hardware configuration/read back (CBus) and two for general purpose I/O (VBus), 8 Megabytes of 25ns SRAM, and sixteen 8K by 16-bit 25ns dual port RAM. The general purpose I/O ports are completely symmetrical with respect to usage and can offer balanced I/O to two completely different systems. With 520,000 gates of reconfigurable logic per board the Virtual Computer can be completely reconfigured in under 25 milliseconds. Of the 520,000 gates, 400,000 lie in a special area called the Virtual Array. This virtual array area consists of 40 Xilinx 4010 along with 24 ICUBE FPIDs surrounded by 256K of dual port RAM. This area has been designed to serve as the main computing engine. The Virtual Computer is housed in a desktop case that any workstation can sit on top of. The Virtual Computer is bus interface independent, which means a specific bus interface can be developed for any computer with relative ease. To interface to the Sun SBus we used one XC4005 Xilinx (5000 gate FPGA). Thus, we envision quickly becoming compatible with most computer systems. A synthesis package (Vsyn based on UC Berkeley's sis/blis[3]) is used to implement computationally intensive software subroutines in "virtual" hardware and allows for many forms of input description.

Hyper-Scalability

Present day CPU based architecture follows the law - any processor X that has throughput Y will always result in throughput less than 2Y for 2X. This is because for every additional processor, additional overhead is required for task handling between those two processors.

Hyper-scalability implies that for any function which fits in a system with X area of logic, and has throughput Y, a system with 2X area has throughput of *greater* than or equal to 2Y. This is because no additional overhead is added for each additional function, and that additional efficiencies will be generated as functions are "bound" together and the logic is reduced.

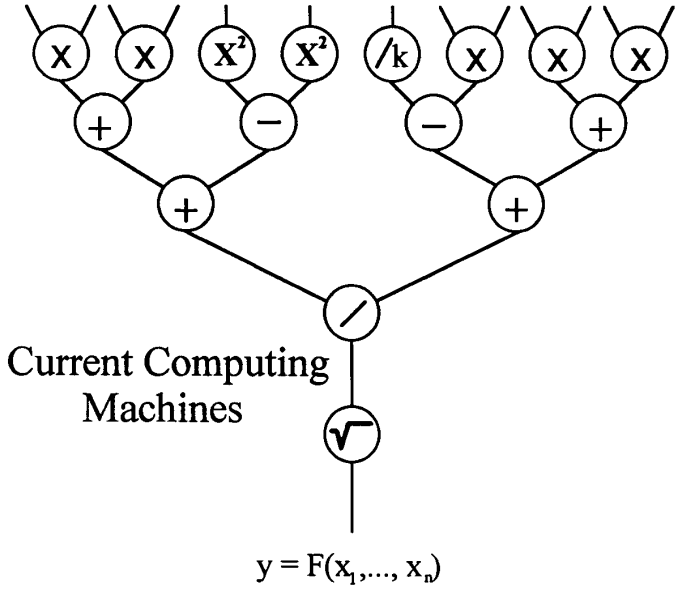
The law of *hyper-scalability* is exemplified by the following examples:

0-8186-3890-7/93 $03.00 © 1993 IEEE

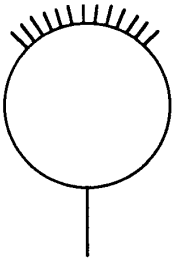
Imagine two processors, X_1 and X_2 , of a given throughput Y in a system where all of it's I/O requirements are satisfied. If two functions are running on X_1 and X_2 and have no data dependencies, the throughput of the system is at *most* 2Y. If there are data dependencies between the functions running on X_1 and X_2, then X_1 must wait for X_2 (or vice-versa) and therefore the throughput is *less* than 2Y. As results are passed on from CPU to CPU, additional overheads are incurred from task management of each processor's run queue.

Now imagine two functions, X_1 and X_2, that run on the Virtual Computer with each taking 1/2 of the logic and having throughput Y with all of their I/O requirements satisfied. If there are no functional data dependencies the throughput is at *least* 2Y. If there are data dependencies, the two functions can be bound together and reduced. Multiple functions can be joined and treated as one and redundant logic can be eliminated. This is done in hardware without incurring <u>any</u> additional overheads from CPU task management. Furthermore, logic left over from the bind can be used for other functions, implying a throughput of *greater* than 2Y.

Unlike any other type of computing system the Virtual Computer architecture scales up to be more efficient than smaller versions. This style of architecture is *Hyper-Scalable* since it scales up better than the "perfect" 1:1 ratio for increased areas of contiguous logic.



Current Computing Machines

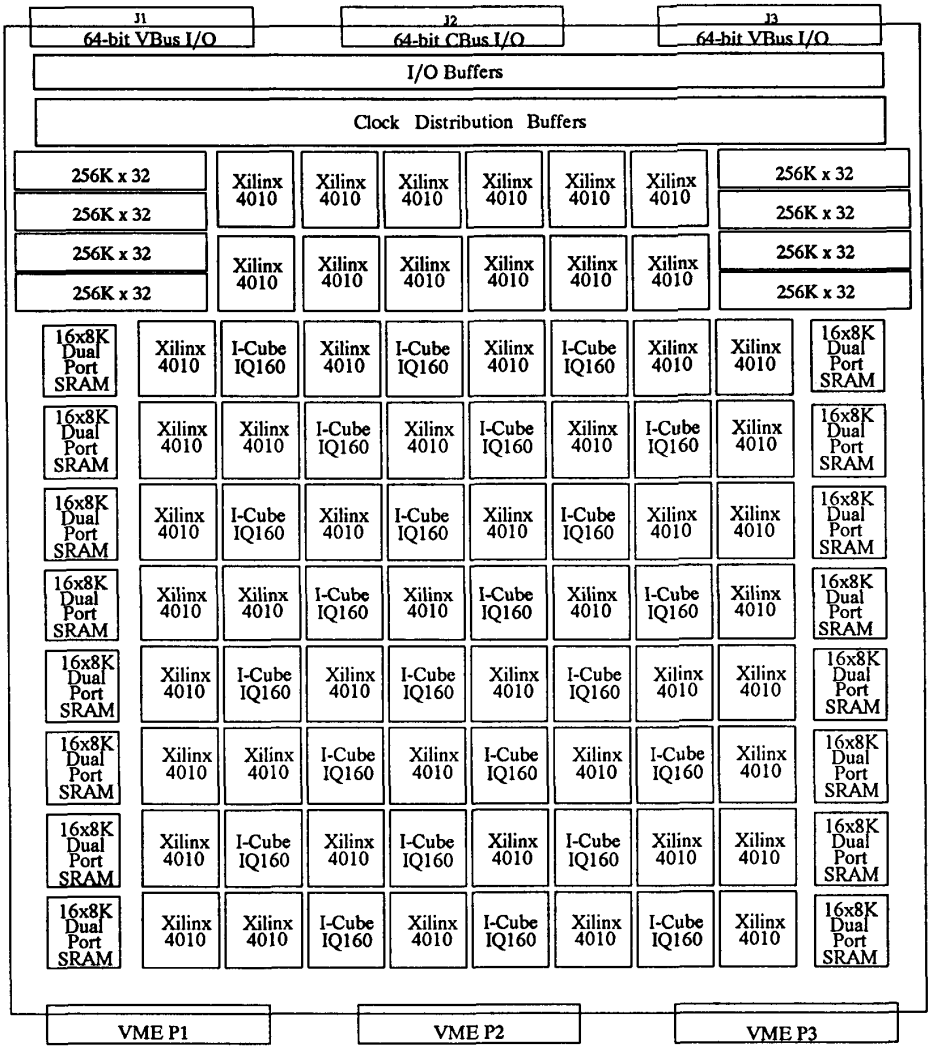


Compacted single function

Virtual Computer

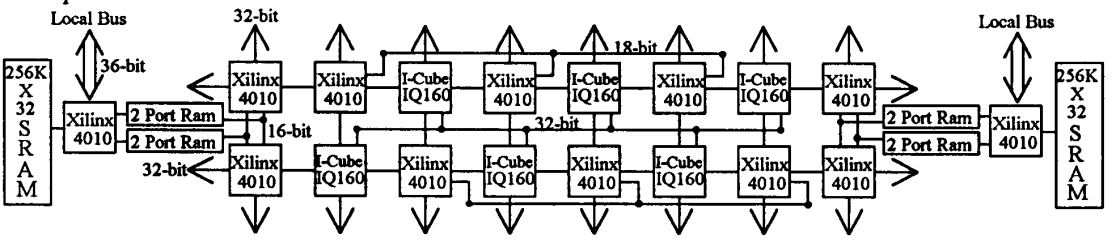
In the above example, multiple functions must be handled independently with a conventional architecture. Each operation as shown (left) requires a specific amount of time to accomplish. The more operations, the more time it takes to complete. The Virtual Computer architecture (right) allows a function to expand as additional operations are required. Additional operations are bound to the original function (if need be) thus forming a contiguous logic pipe through which data can stream at the system clock rate.

Virtual Computer Corporation
Virtual Computer ™
Patent Pending



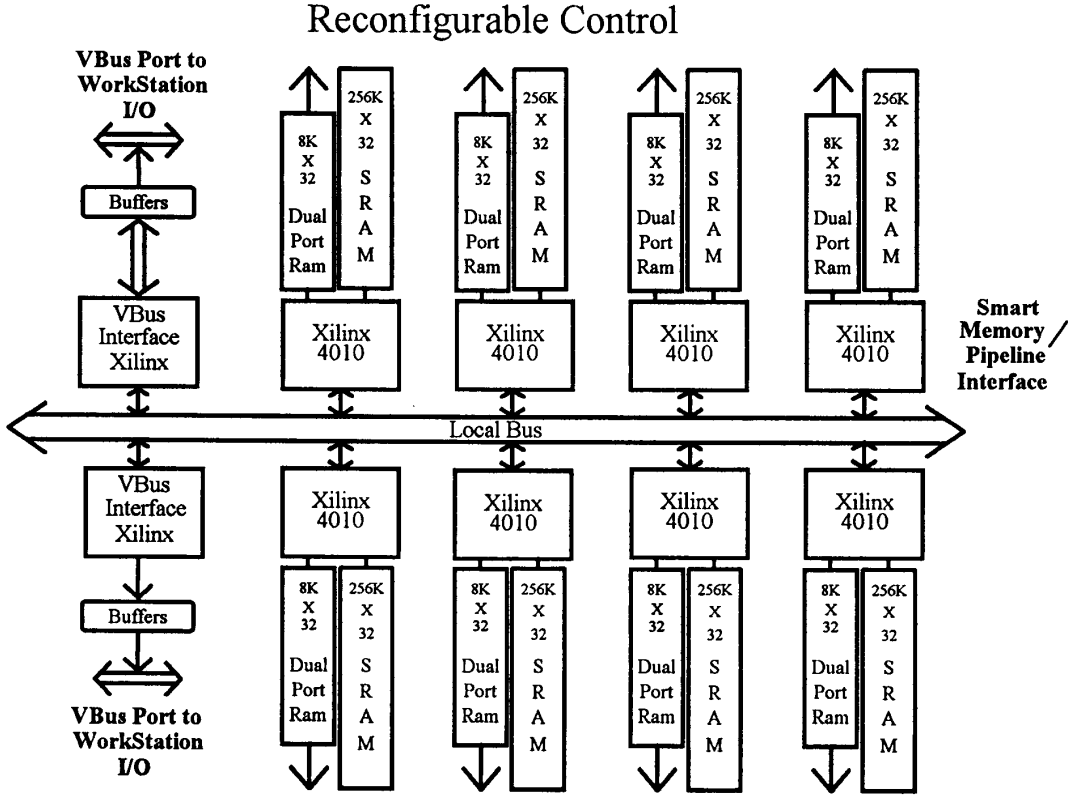
- **520,000 Reprogrammable Gates**
- **8 Megabytes of SRAM for DMA and Scratch Pad Memory**
- **256 Kilobytes of Dual Ported RAM for Data and Coefficients**
- **IQ160 Provides Reprogrammable Interconnect (any pin to any other pin)**
- **Fully Reconfigures in 25 milliseconds**
- **Single-Step/Readback any Resister**

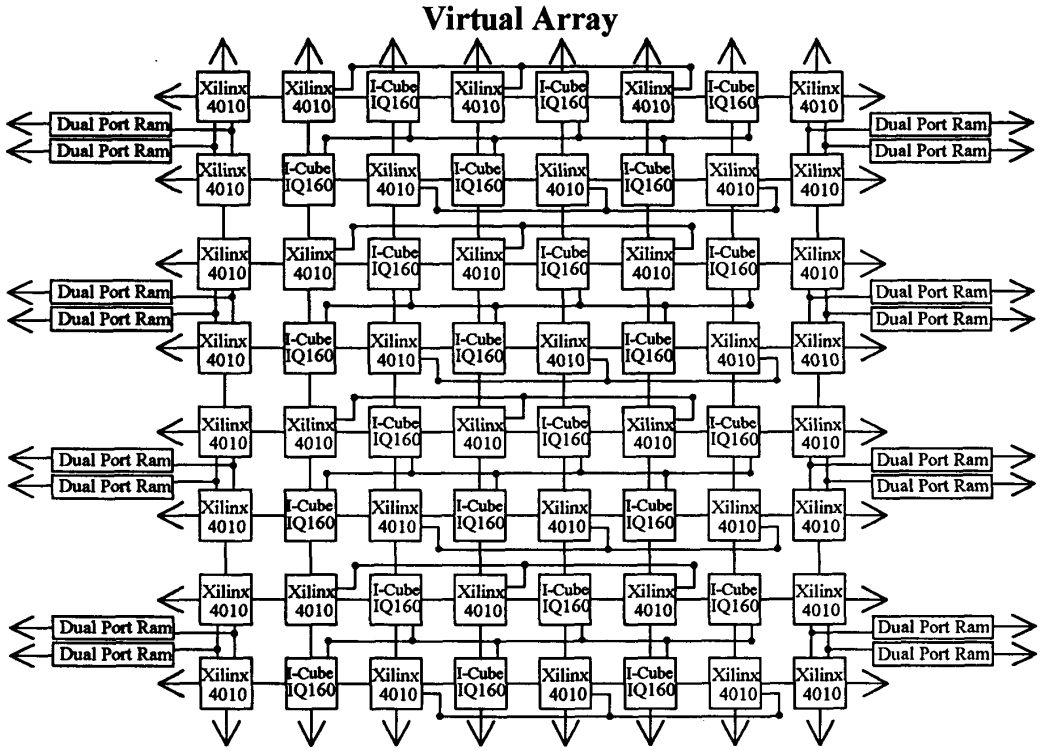
Virtual Pipeline



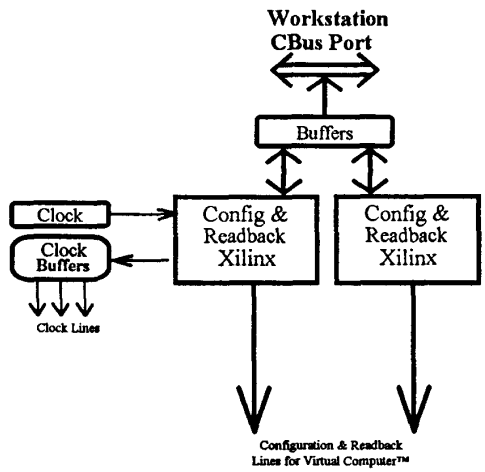
The block diagram above depicts the Virtual Pipeline (VP) which is the basic unit of the Virtual Computer. There are four identical pipes per board. Each of the pipes have 100,000 gates of logic in between the dual port ram and 20,000 gates that allow for smart memory and local bus interface.

Below we show the control section (kernel) of the *Virtual Computer.* Next to the kernel is the GERM (Generic Environment Resource Manger) which handle the kernel's data I/O. The Virtual Computer supports two 64 bit general purpose I/O ports (VBus) that can be interfaced to almost any system through the GERM. Because the GERM is reconfigurable, differing bus protocols can be supported by loading the appropriate configuration files. Current configuration will be to a SUN SparcStation II. Other interfaces will be supported as required.





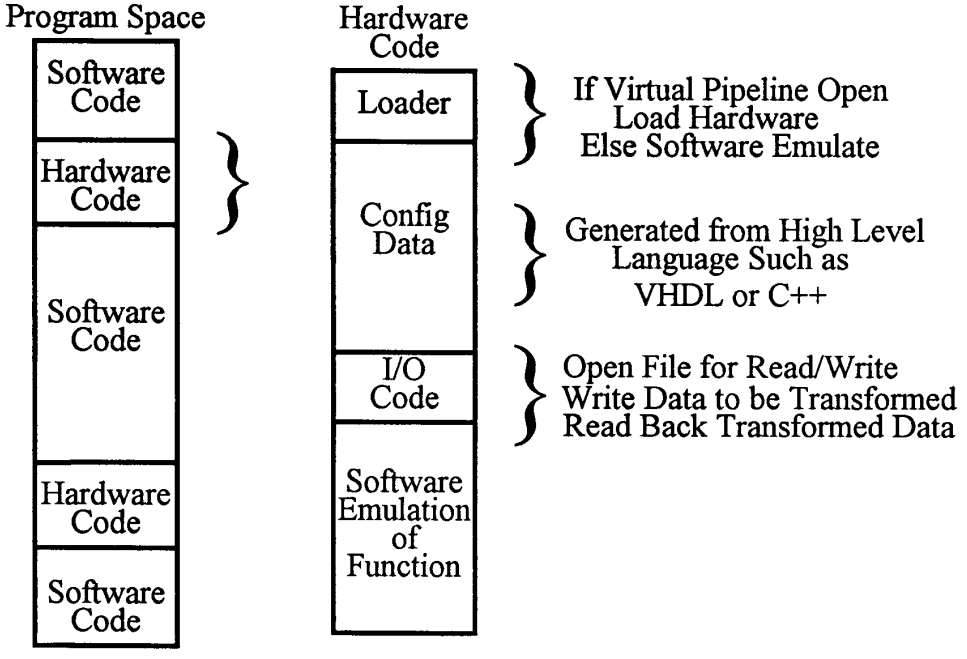
The block diagram above depicts the interconnection of the four Virtual Pipes. Most data paths shown are 32-bit connections with the exception of the dual-port ram and the Ugly Bus which are connected via an 18-bit bus. The dual-port ram can address 32-bit, 16-bit, or 8-bit data. 33-bit to 8,000-bit mathematics (+,-,*,/,Σ,√ ∫,∂f(x,y...),...) can be implemented in hardware by using bit-serial methods. The signals at the edges of the array wrap around creating a spherical topology. The basic unit is called a Virtual Pipeline and consists of two rows of logic terminated by two dual-port ram on either side. The other port of the dual-port ram connects to a control Xilinx that also has a one megabyte static ram. This Xilinx is also connected to our 36-bit local bus and acts as a Smart Memory/Pipeline interface. There are four of these "Virtual Pipes" on one board; each pipe has 120,000 gates of logic. We have made our board bus interface independent by running buffered control and data lines off to three 96 pin connectors on the back of the board. The first interface is to a Sun SparcStation II through the SBus. The *Virtual Computer* can be reconfigured in under 25 milliseconds with this configuration.



The Configuration Bus (CBus) is an independent 64-bit bus used to configure the board. The configuration and readback control (CRC) section is in charge of configuration, readback and clock control for single stepping and debugging. This will allow the state of any set of registers to be examined.

Programming Model

The *Virtual Computer* (like any other computer) has different programming models associated with it. At the highest level a programmer would call out a function in their code and link it with a run-time library. This is as simple as calling out a function in your code and opening a file for I/O. One level down from this would be writing a function in some object oriented language like Verilog or C++ code (which will be supported) and linking with a large library of hard macros. This function could then be used as described above. At a lower level is Verilog which we are targeting as our "assembly" language. At the lowest level is a full custom hand placement and routing.



The bottom line in compiling a program for the *Virtual Computer* is "it should just be another line in the make file."

Conclusion

As the need for higher through-put computer system grows only hyper-scalable system will meet the needs of the supercomputing community.

References

[1]The XC4000 Data Book, Xilinx Corporation, 1992.

[2] FPIDPro User's Guide, I-Cube Design Systems, Inc, 1992 .

[3] SIS: A System for Sequential Circuit Synthesis, ERL Memorandum No. UCB/ERL M92/41