Net Aware BitStreams that Upgrade FPGA Hardware Remotely Over the Internet

CREATING INTELLIGENT BITSTREAMS THAT KNOW WHERE TO GO; WHAT TO DO WHEN THEY GET THERE; AND CAN REPORT BACK WHEN THEY'VE DONE.

Steve Casselman CEO-President, John Schewel VP Marketing & Sales
sc@vcc.com, jas@vcc.com
Virtual Computer Corporation,
6925 Canby Ave #103
Reseda, California, USA 91335

Success in the marketplace may well depend upon the ability to upgrade and test hardware designs instantly around the world. An upgrade management strategy requires more than just the bitstream file, email or a JTAG cable. A well-managed methodology, capable of transmitting bitstreams directly into targeted FPGAs over the network or internet is an essential element for a successful FPGA based product strategy. Virtual Computer Corporation's HOTMan? , Bitstream Management Environment combines a feature rich cross-platform API with an Object Oriented Bitstream technique for Remote Upgrading of Hardware over the Internet.

DEPLOYING HARDWARE FOR IN-SYSTEM UPGRADES

To upgrade hardware remotely, the engineer needs a simple yet robust methodology for managing and delivering bitstreams. Whether the target hardware is across the hall or across the country, transmitting the bitstream, configuring the hardware and verifying its functionality is required. This can be a cumbersome process at best. We will describe a point-to-point solution for the management of in-system upgrades. A well-managed remote upgrade methodology includes these features:

1. Upgrading FPGA designs with a single design methodology that works equally as well with embedded systems as it does with any standard O/S based systems.
2. A methodology, that's portable and customizable; with cross-platform capabilities for updating bitstreams with functional controls from within application software.
3. Remote FPGA configuration, capable of total or partial reconfiguration of single or multiple FPGAs via SelectMAP or JTAG ports. [1,2]
4. A Robust upgrade architecture enabling the upgrading of hardware based upon the initiative of a central command (a 'push' update) with version control and target instructions built-in.
5. A method flexible and powerful enough that it can enable highly reliable upgrades both Remotely & Automatically; including features like, secured and compressed bitstream files.
6. Designed for Low Cost Maintenance, with easy cross-platform migration; Code that is reusable across products and customer application; quickly modifiable for any new hardware in a product.

BITSTREAMS AS HARDWARE OBJECTS

Object-oriented programming (OOP) is a revolutionary concept that changed the rules in computer program development. An object is a bundle of variables and related methods. Programming objects are modeled after real-world objects in that they too have state and behavior. An object maintains its state in one or more variables. An object implements its behavior with methods. A method is a function (subroutine) associated with an object. By applying Object programming techniques to bitstream files, we get a bitstream file that includes bitstream data, other resource data and methods for the management of the bitstream itself. The bitstream configuration file alone is not enough to facilitate remote upgrading. It needs other data such as version history for tracking and maintenance, network protocol data, target location information, programmed commands and other essential data for use from within executable programs. This makes bitstream *Intelligent*. The *Intelligent BitStream* includes all the information required for delivery, verification, and use.

We call the Intelligent Bitstream a **Hardware Object**. Developed in 1994, VCC's **Hardware Object Technology (HOT)** provided the transportation link between bitstream generation and Run-Time use of the bitstream within executable programs [3,4]. The converted BitStream file was ready for in-program use and enabled Remote Run-Time configuration of the FPGA on our HOT2 Xilinx/PCI Board Development System [5]. The current Hardware Object Class includes a wider range of data and methods; containing all the necessary information and processes for remote configuration of Virtex FPGAs over the network.

The **HOTMan Application Program** creates the Hardware Object File from a user selected bit file (`<name>.bit`). It outputs a static array file (`<name>.hot`) or a dynamic file (`<name>.dll`, `<name>.so`). The Hardware Object (HOT File) is used with the HOT API, JAVA and a C++ compiler to provide program control and delivery. The Hardware Object contains the compressed bitstream data as well as a number of data fields (see below).

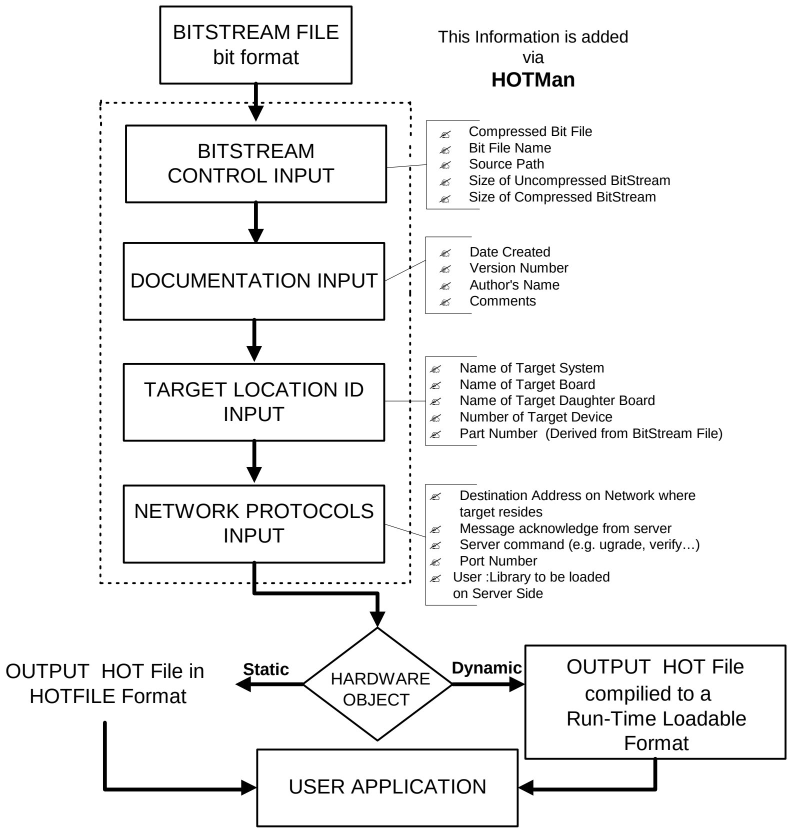
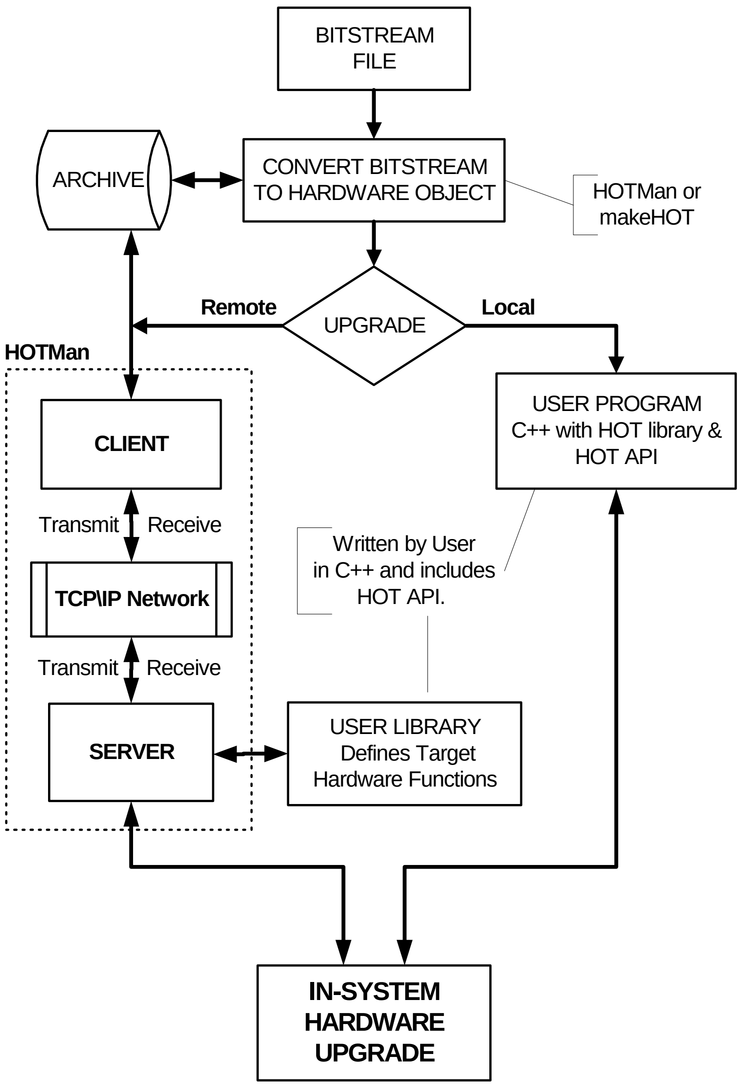

BITSTREAM FILE
bit format
This Information is added via
HOTMan
BITSTREAM CONTROL INPUT
Compressed Bit File
Bit File Name
Source Path
Size of Uncompressed BitStream
Size of Compressed BitStream
DOCUMENTATION INPUT
Date Created
Version Number
Author's Name
Comments
TARGET LOCATION ID INPUT
Name of Target System
Name of Target Board
Name of Target Daughter Board
Number of Target Device
Part Number (Derived from BitStream File)
NETWORK PROTOCOLS INPUT
Destination Address on Network where target resides
Message acknowledge from server
Server command (e.g. ugrade, verify…)
Port Number
User :Library to be loaded on Server Side
HARDWARE OBJECT
Static
Dynamic
OUTPUT HOT File in HOTFILE Format
OUTPUT HOT File compilied to a Run-Time Loadable Format
USER APPLICATION

Figure 1. Hardware Object Content

BitStream File Management Control: Contains user selected Bit File, compressed and inserted within the Hardware Object. **Documentation Control:** Contains Author Information, creation date, comments, security field as well as options for auto-version/revision control numbers. **Target Location ID:** Information on the system, board and device location of the targeted FPGA to be updated. **Network Deployment Protocols:** Network Addresses & Server Command data. **Object Methods:** Methods for bitstream deployment, verification, and testing of circuit as well as data transfer mechanisms to the hardware. Methods are activated via a combination HotMan Program Functions, HOT API and/or User Libraries & User written programs.

HARDWARE OBJECT DESIGN FLOW

The conversion of the standard bitstream to a Hardware Object provides foundation for a well-managed bitstream delivery solution. The Hardware Object is a bitstream that knows where to go, knows what to do when it gets there and report back on it's status. Figure 1 shows the design flow for Hardware Object Use. HotMan is an easy to use, cost effective tool for advanced FPGA based IP network deployment management. [6]


BITSTREAM FILE
ARCHIVE
CONVERT BITSTREAM TO HARDWARE OBJECT
HOTMan or makeHOT
Remote
UPGRADE
Local
HOTMan
CLIENT
Transmit
Receive
TCP\IP Network
Transmit
Receive
SERVER
USER PROGRAM C++ with HOT library & HOT API
Written by User in C++ and includes HOT API.
USER LIBRARY Defines Target Hardware Functions
IN-SYSTEM HARDWARE UPGRADE

I. CREATE HARDWARE OBJECT

tep 1: Convert BitStream to Hardware Object using HOTMan or makeHOT enabled programs.

Step 2: Add Network Deployment Protocols, Target Location ID and Documentation Control Information to the Hardware Object.

II. LOCAL

Step 1: Program: Interface Hardware Object w/ Target Hardware via user written C++ program using the HOT API and HOT File.

Step 2: Upgrade: Activate User Program for Hardware Object Deployment.

III. REMOTE

Step 1: Program: Interface Hardware Object w/ Target Hardware via user written C++ program using the HOT API, User Library and HOT File.

Step 2: Upgrade: Activate HotMan/Server; Run Hardware Object on Client Side.

IV. Acknowledge

Step 1: Program: Implement desired Acknowledgment routine in Custom User Program or HOT File data.

Step 2: Acknowledged Data from Upgraded via Client HotMan or Custom User Program.

Figure 2. Design Flow for Creation & Use of Hardware Objects

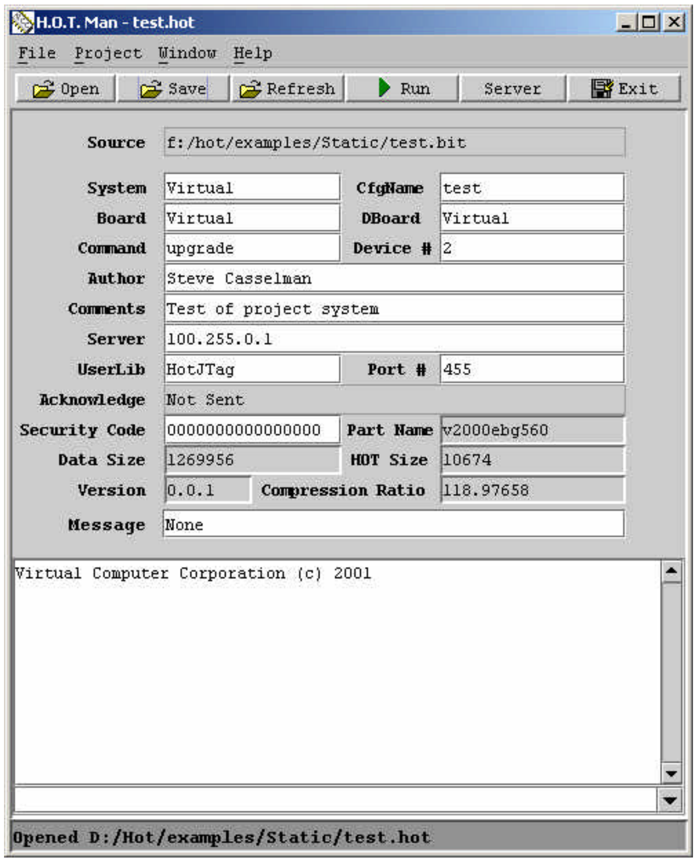
THE APPLICATION PROGRAMMING INTERFACE

The HOT API is a set of file, compression, file manipulation and data retrieval methods for controlling the Hardware Object. Besides the data setting functions relating to the HOT File data shown in the tables above, the HOT API includes `Compress()`,`UnCompress()`,`Load()`and `Save()`. The Hot RunTime Libraries and `Hot.h` files are the link between the Hardware Object and programmed execution. They are called by the makeHot/ HOTMan programs and must be called by any user written programs. User written programs in C++

use the HOT API to provide access to the Hardware Object locally or remotely [7]. Below is an example program code for printing the Server Address information:

```
        // Output the Destination Server Address of the target hardware
        printf("\tDestination Server = %s\n", Hottest.getServer());
```

The HOTMan Bitstream Management Environment includes: HotMan , a JAVA based program for creation, management and deployment of Hardware Objects. It includes a 'RUN' feature for activating Hardware Objects. If the target hardware is located on a remote server, the HOTMan program in Server mode must be running on the target server. The Bitstream Management Environment includes the MakeHot program, a command line program for creating Hardware Objects and HOT API files for C++ programming.



Source	f:/hot/examples/Static/test.bit		
System	Virtual	CfgName	test
Board	Virtual	DBoard	Virtual
Command	upgrade	Device #	2
Author	Steve Casselman		
Comments	Test of project system		
Server	100.255.0.1		
UserLib	HotJTag	Port #	455
Acknowledge	Not Sent		
Security Code	0000000000000000	Part Name	v2000ebg560
Data Size	1269956	HOT Size	10674
Version	0.0.1	Compression Ratio	118.97658
Message	None		

```
Virtual Computer Corporation (c) 2001
```

```
Opened D:/Hot/examples/Static/test.hot
```

Menu and Tool Bar

RUN Button: Activates the Hardware Object

Location of bit file

Names of Hardware Object, System, Board & Device #

Command (e.g. upgrade, verify, test)

Author & Comments

Remote User Library & Port Number

Acknowledge: Status of Command

BitStream Information Fields

Version Control Number

Command Line Window and Status Bar

Figure 3. HotMan -- Main Window

The HOTMan BitStream Management Environment is not bound by any specific OS restrictions. There are two components necessary to implement an upgrade strategy on the client-side; the HOTMan program and HOT dynamic library for bitstream conversion and activation. On the server side, three software components are needed; HOTMan Server Program, HOT dynamic library and the HotJTag dynamic library for upgrading any hardware connected via the JTAG cable. The HOTMan environment also provides the necessary API for the user to write a stand-alone program and user dynamic library for remote upgrading.

The simplest method of upgrading hardware in the field is described below. Install HOTMan on the client and server side, along with the HotJTag dynamic library on the server and you are just a 'click' away from configuring your remote hardware via the JTAG cable. (see figure 4)

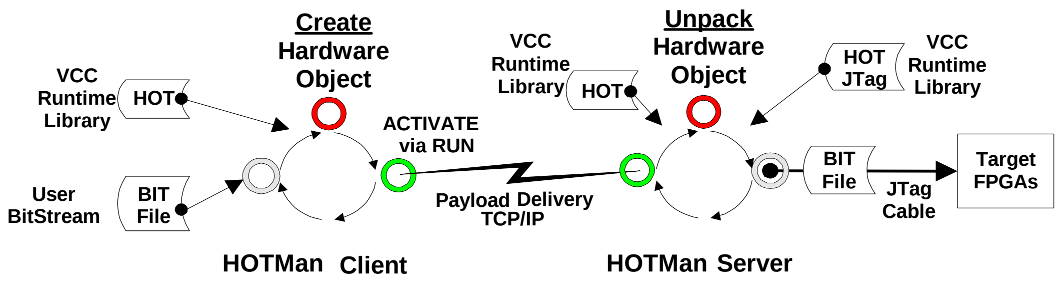


Figure 4. Using HOTMan BitStream Management Environment

IMPLEMENTATION

In a series of bitstream compression benchmarks [8], HOTMan averaged 54% compression on a diverse number of designs. This would double the number of configurations one could place inside a PROM (see table below). With HOTMan, the same Hardware Object can easily be delivered, whether the target hardware is in a Windows managed PC, a Linux based cluster or an embedded system. The combination of this delivery feature and the compression of the bitstream offers alternative design choices that can save money. The table below shows the configuration requirements for example Xilinx FPGAs, using the Xilinx In-System programmable PROM solution. The target hardware must have a processor and connectivity for remote programming PROM and/or FPGA. In the high end cases below, the compression feature alone could cut the number of PROMs needed in half. In all the cases, one can even eliminate the PROM altogether with direct remote configuration using HOTMan; savings can be calculated in components used and power costs.

FPGA Device	Number of Configuration Bits	Number of PPROMS needed *	FPGA Cost **	PROM Cost**	Config. Cost Ratio
HIGH END					
XCV1000E	6,587,520	2 -- XC18V04	1,300.00	92.00	7%
XCV2000E	10,159,648	3 -- XC18V04	3,300.00	138.00	4%
XCV3200E	16,283,712	4 -- XC18V04	8,000.00	184.00	2%
LOW END					
XC2S50	559,200	1 -- XC18V01	16.60	17.25	104%
XC2S15	197,696	1 –XC18V256	8.85	10.75	121%

NOTES: * -- In-System Programmable Xilinx PROM (Data from: Xilinx Inc. DS026 (v3.0) Nov 2001)
** -- Avg. Dist. Price 2002 (Qty. 1) Partminer

The example task below is to upgrade the #2 FPGA on Virtual (Board) over the internet via the JTAG parallel port connector (Figure 4). The bitstream configuration file is converted into a Hardware Object called `test.hot` on the Workstation via HOTMan.

SETUP: Settings within Hardware Object -- **`test.hot`**
Hardware Object Data is set via -- HOTMan or makeHot or from within a User written program.
Where to go: Locate Target Device for Upgrading.

```
//location of target hardware
Hottest.setServer (100.255.0.1);
Hottest.setBoardName(“Virtual”);
Hottest.setPortNumber (455);
// location of target Device #2, the third FPGA on the board chain
(Single device location is set to 0)
Hottest.setUnitNumber (2);
```

What to do: Perform an Upgrade on target Hardware with the JTAG cable.

```
// Command Functions
Hottest.setCommand (upgrade);
Hottest.setUserLib (HotJTag);
```

Server Side: VCC Library: HotJTag -- Method: upgrade
Setup: FTP `HotJTag` dynamic library to target server. Connect the Parallel JTAG cable on the server to target system. Run HotMan on Server with Server Activated at Port 455.
Client Side: Upgrade Remote HardwareUse: 1) Start `HOTMan` Client, 2) Open `test.hot`, 3) Press RUN Button, 4) Check the Status Bar on HotMan window for confirmation of successful upgrade.

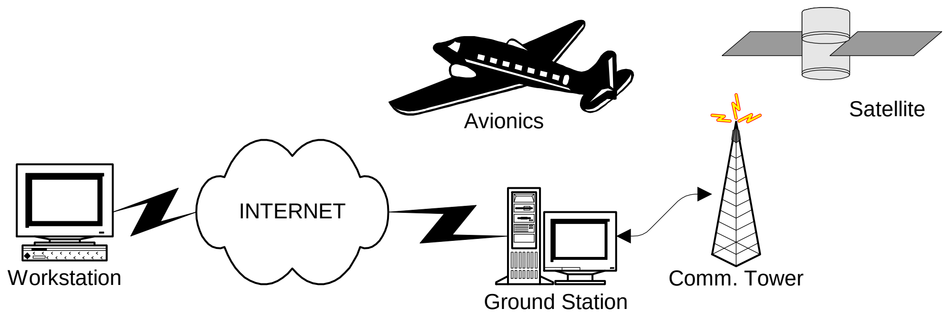


Figure 5. Using HOTMan BitStream Management Environment for Avionics & Satellite Space Application [9]

CONCLUSION

The combination of design complexity, component size and the use of multiple FPGAs in target systems increase the number of design versions used by customer products. A method for handling remote hardware upgrades in the field is necessary for a successful reconfigurable product line strategy. The bitstream file needs to carry with it all the information required for delivery, verification, and use.

With only minor changes to your software, the HOTMan Bitstream Management Environment can enable your systems for Remote Upgrading and add value for both you and your customers. The addition of HotMan to your product will extend it's life and simplify support and distribution models. With HotMan, you could manufacture a single physical version of your hardware and ship multiple different hardware versions. Your customers will appreciate the speedy, hassle-free upgradability of your products. The VCC HOTMan BitStream Management Environment provides a powerful software framework that allows designers to easily integrate Remote Upgrading of FPGAs into their designs. The object oriented nature of Hardware Objects eliminates the need to handle low level issues with hardware interfacing designs, JTAG or SelectMap programming, allowing the designer to focus on the end-user's application. The modular nature of HOTMan allows you to add new features without disturbing your current application framework.

The software architecture of the HOT environment places enough information into the hardware object for it to get to it's target no matter where it is. While other vendor solutions try and make their system 'push button', they in fact, take away flexibility [10]. They force you to design your PCB to their specification and their operating systems. Designers need this flexibility to lower cost of the system they are trying to implement. The HOTMan BitStream Management Environment is a reliable and low cost software-based solution for remote upgrading of single or multiple FPGAs.

REFERENCES

1. *The Virtex-II Platform FPGA Handbook* Xilinx, Xilinx Inc. December 2000
2. *Virtex Configuration Architecture Advanced Users' Guide XAPP 151* Xilinx, Xilinx Inc.1999.
3. S. Casselman, M. Thornburg, J. Schewel *User's Guide*, *EVC1*, Virtual Computer Corp., August 1994
4. S. Casselman, M. Thornburg, J. Schewel, *Hardware Object Programming on the EVC1*, Proceedings of SPIE FPGAs for Fast Board Development and Reconfigurable Computing, April 1995.
5. S. Casselman, C. Beaumont, J. Schewel, *IP Validation for FPGAs using Hardware Object Technology*, Proceedings of 9th International Workshop FPL'99, August 1999.
6. S. Casselman, *IP Reconfigurable Logic Based Fibre Channel Network Card with Sub 2us Raw Latency*, Proceedings of 65th International Workshop FPL'96, September 1996.
7. Bjarne Stroustrup,. *The C++ Programming Language*, *Addison-Wesley*, Mass. USA. February 2000.
8. *Z. Li*, S. Hauck, *Configuration Compression for Virtex FPGA*, IEEE Symposium on FCCM, 2001.
9. S. Casselman, J. Schewel *Exploration & Experimentation with Reconfigurable Computing*, Workshop on Remote Exploration and Experimentation (REE), Jet Propulsion Laboratory/NASA, August 1995.
10. PAVE Framework (PLD API for VxWorks Embedded Systems), Xilinx Inc. DS084 (v1.0) September, 2001.