

Review of:
**Net Aware BitStreams that Upgrade FPGA
Hardware Remotely Over the Internet**

- Paper by:
 - Steve Casselman and John Schewel
 - Virtual Computer Corporation
- Published in:
 - Proceedings of SPIE, ITCOM 2002
 - Boston, July 2002
- Original copy on-line as:
 - <http://www.arl.wustl.edu/~lockwood/class/cs6812/NetAware.pdf>
- Survey by:
 - David V. Schuehler

The Challenge

- Create Tool to Upgrade Remote Hardware over the Internet
 - Has rich feature set
 - Supports multiple platforms
 - Provides object oriented solution
- Create Intelligent Bitstreams that
 - Know where to go
 - Know what to do when they get there
 - Can report back when they've completed

Style of the Paper

- Conference paper
 - Reads as a sales brochure
 - Trying to sell Hardware Object Technology
- Small number of references
 - Half of references are by same author
- Missing competitive analysis

Deploying Hardware for In-System Upgrades

- Requirements
 - Transmit bitstream
 - Configure remote hardware
 - Verify functionality
- Methodology
 - Simple interface
 - Robust feature set
 - Delivers bitstreams to remote locations

6 Keys to Remote Upgrade Methodology

- **Single Design Methodology**
 - Works with either embedded systems or any standard O/S based system
- **Portable and Customizable**
 - Cross-platform capabilities
 - Flexible API with functional controls
- **Wide Configuration Support**
 - Partial configuration
 - Target multiple devices
- **Robust Upgrade Architecture**
 - Push upgrades from central location
 - Version control
 - Target instructions
- **High Reliability**
 - Remote and automatic upgrades
 - Security
 - Compression
- **Low Maintenance Cost**
 - Cross platform migration
 - Reusable code
 - Easily adapted to new hardware

Hardware Object Technology (HOT)

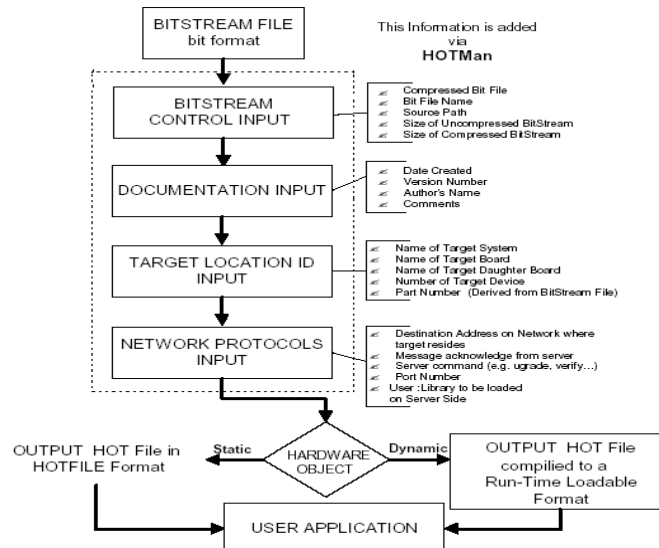
- Developed by Virtual Computer Corporation
- Provides transportation link between bitstream generation and their run-time use
- Hardware object is a self-contained object that contains data and methods

HOTMan Application Program

- Creates Hardware Object File
 - Input
 - user bit file: <name>.bit
 - Output
 - Static array file: <name>.hot
 - Dynamic file: <name>.dll or <name>.so

- Provides Program Control and Delivery
 - HOT API
 - JAVA
 - C++

Hardware Object Generation



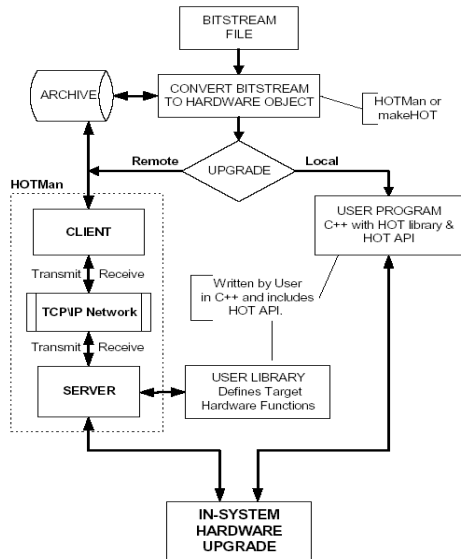
Hardware Object Content

- File Management Control
 - Compressed user bitfile
- Document Control
 - Author
 - Creation date
 - Security information
 - Comments
 - Version control
- Target Location ID
 - Remote system identifier
- Network Deployment Protocols
 - Network commands and addresses
- Object Methods
 - Deployment
 - Verification
 - Testing

Hardware Object

- Provides foundation for delivery solution
- Knows where to go
- Knows what to do when it gets there
- Reports back with final status

Design Flow



I. CREATE HARDWARE OBJECT

Step 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.

Application Programming Interface

- File Manipulation
- Compression/Decompression
- Data Retrieval

HOTMan Bitstream Management Environment

- **HOTMan application**
 - JAVA based program
 - Creates of objects
 - Manages of objects
 - Distributes of objects
 - Activates objects via RUN feature
 - Operates in Client or Server mode
- **MakeHot application**
 - Command line program for creating objects
- **HOT API**
 - Supports C++ programming environment

HOTMan Main Window

The screenshot shows the HOTMan Main Window application. The window title is "HOTMan - test.hot". The menu bar includes "File", "Project", "Window", and "Help". The toolbar contains buttons for "Open", "Save", "Refresh", "Run", "Server", and "Exit". The main area is a form with the following fields:

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	HotTag	Port #	455
Acknowledge	Not Sent		
Security Code	0000000000000000	Part Name	v2000ebg500
Data Size	1269956	HOT Size	10674
Version	0.0.1	Compression Ratio	118.97659
Message	None		

At the bottom of the window, there is a status bar that reads "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

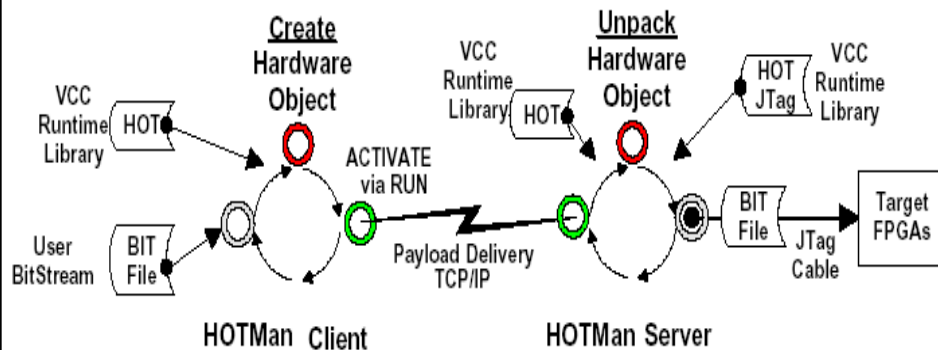
Version Control Number

Command Line Window and Status Bar

HOTMan Components

- Client Side
 - HOTMan program
 - HOT dynamic library
 - Bitstream conversion
 - Activation
- Server Side
 - HOTMan server program
 - HOT dynamic library
 - HOTJTag dynamic library
 - Upgrading hardware via JTag probe
- Programming API
 - Allows user to develop custom solution

Remote Programming Environment



Implementation

- **HOTMan averaged 54% bitfile compression**
 - Doubles the number of configurations that can be stored in a PROM
 - **Not really true**
- **Flexible Hardware Object Technology**
 - Supports Windows, Linux, or embedded target environments
- **Target Limitations**
 - Processor at end system
 - Network connectivity

Potential Cost Savings

- Bitfile compression could reduce the number of PROMs needed
- Direct Remote programming could eliminate PROMs altogether

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) Partner

Example

- Location of Remote Target

```
//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);
```

- Upgrade Operation

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

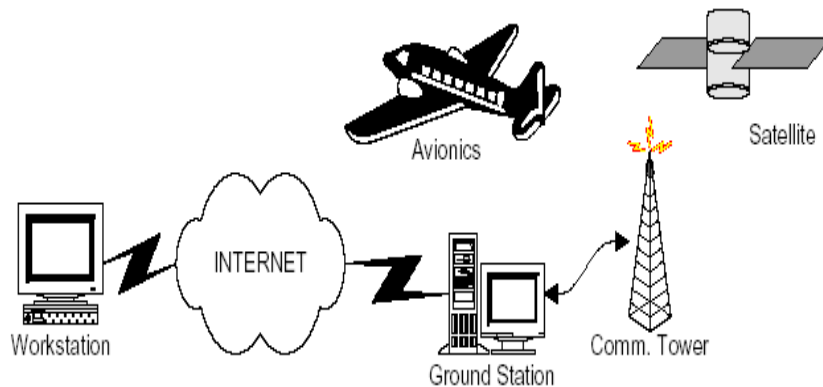
Server Side Setup

- Copy HotJTag Library to Target Server
- Connect Parallel JTAG Cable on Target System
- Run HotMan as Server on Remote System
 - Listen on port 455

Client Side Setup

- Start HOTMan Client
- Open HOT File
- Execute HOT File by Pressing RUN Button
- Check for Completion Status

Aviation & Satellite Environment



Conclusion

- Relevant Topic
- Reads as Advertisement
- Weak References
- No Comparative Analysis