FPGA Security, FPGA Configuration, FPGA Bitstream, FPGA Authentication

Business Considerations for Systems with RAM–Based FPGA Configuration

CJ Clark, Intellitech Corporation
[cjclarkatintellitechperiodcom, 603-868-7116]

Abstract

Can an FPGA configuration choice hurt the company's bottom line? RAM–based FPGAs need on-board methods to program a design into the FPGA at power-up. Years ago, the primary solution was to add a specialized FPGA PROM to the PCB for this purpose. FPGAs now have eight or more ways to load design bits using external NOR FLASH, serial FLASH, a CPU, JTAG, or other specialty devices. Is it good use of engineering time to build an ad-hoc method? Complexity has increased as the solution must support multiple design versions, encryption, gray market protection, Trojan protection, fail-safe updates, mixed FPGA families, and more. This paper explores the methods and costs to the company.

Author Biography

CJ Clark is the president and CEO of Intellitech Corp.
He was the elected chairperson of the IEEE 1149.1 JTAG working group from 1996 to 2002. He has been active in other IEEE working groups, most recently P1687 or "IJTAG". He was instrumental in the P1687 working group adopting TCL for the IJTAG language. He has presented on FPGA configuration at the FPGA Summit, the IEEE lecture series on "Mission Critical Embedded Systems", International Test Conference, TECS (Testing Embedded Cores-Based Systems) Workshop, the Board Test Workshop, Ottawa Test Workshop and VLSI Test Symposium. He is on the program committee for VTS 2009.

CJ serves on the University of New Hampshire College of Engineering and Physical Science (CEPS) Advisory Board. He also serves on the UNH Department of Electrical Engineering Advisory Board and is a frequent guest lecturer. He is co-inventor on three US patent related to scan-based test and FPGA configuration, two European, two Hong-Kong, two Canadian, two Japanese, one Korean, one Taiwanese patent with others pending world-wide. His first job in test was in 1978 with Plantronics/Wilcom.

Introduction

Economic downturns are good times to examine costs, internal methodologies and corporate culture. Designing the ecosystem needed for multi-FPGA based PCBs continues to become more complex however engineering time budgets continue to decrease. Today's FPGA based PCBs need programmable DC/DC converters, watch dog timer ICs, power-on reset ICs, programmable clocks, multiple flash memories, serial eeproms and FPGA configuration devices. FPGA based products need security from hacking and cloning, not just bitstream reverse engineering.

The position in this paper is that using an off the shelf 'board manager' can reduce the parts costs, complexity and reduce many other downstream costs to the company that may traditionally not be considered. The SystemBIST IC described in the presentation allows user-defined FPGA configuration sequences, DC/DC voltage margining, programmable resets and watch-dog activity, PCB and FPGA anti-cloning security, anti-hacking security, IEEE standards based BIT or Built-In-Test, and built-in field updating with version control.

I can design, so I do

Let's say you're a manager and your designer tells you the PCB for your next project needs 2.5V and 1.8V. Your designer says, "I can design that. I need four weeks of design time, plus time to validate and debug it and time for the embedded software team to support it". You are going to do what? Design a DC/DC converter? What for? Why not buy one? Sounds crazy, I know, but bring this scenario into the digital domain and it is common practice. Designers needing a FPGA configuration method outside of what is provided by the FPGA vendors and the answer is 'design it myself'. One fellow who needed to configure eight FPGAs with the same design, came up with his own-FPGA configuration method, designed with, guess what another FPGA. "I have a PROM which loads an FPGA design into my FPGA which then the CPU uses to tell the other FPGA how to load the other eight FPGAs." While simple in concept, the cost of the PROM, FPGA and more importantly the debug, lack of tools, and downstream manufacturing costs to debug and troubleshoot far exceeded the initial estimates. Just because you can design something, doesn't mean that you should. I *can* solder rs-232 cables with 25 pin DSUB connectors, and build one hundred of them. I'm pretty sure this is not good use of my time, however.

FPGA vendors have done a great job in providing many different ways to program a RAM based FPGA. There exists a large number of application notes and ideas on how it can be done. FPGAs support parallel configuration, serial configuration, JTAG, FPGA as master, FPGA as slave, direct NOR flash, direct serial FLASH, CPLD and FLASH, CPU and FLASH, PROMs, configuration devices and compact flash. These design choices have further ramifications in that some methods will support an encrypted bitstream, some methods will not. Some of the methods will work with high-end FPGAs only like the Stratix and Virtex and some are designed only to work with the lower priced Spartan and Cyclone.

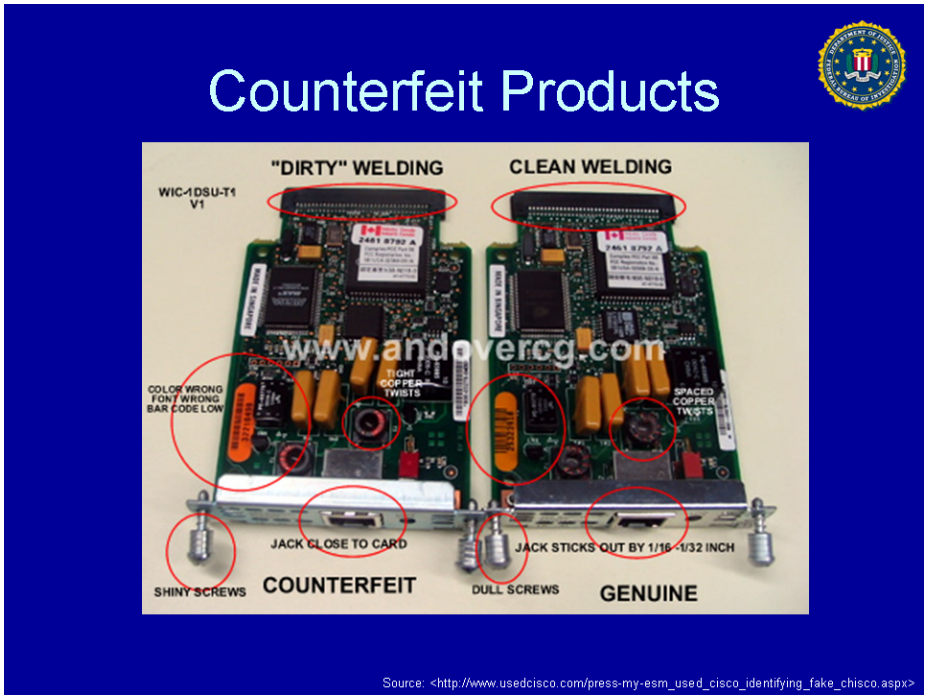
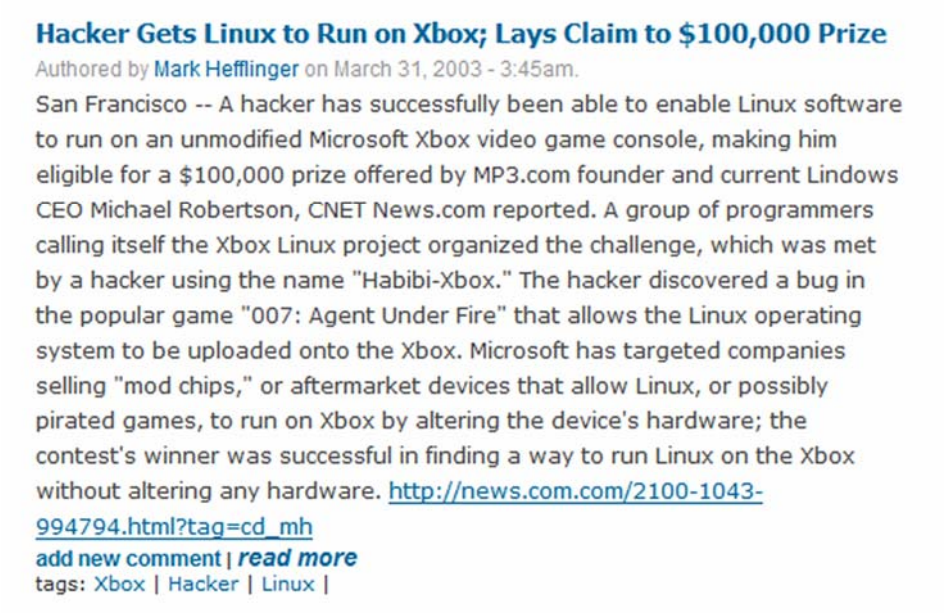


Figure 1. Cloning is real

In some cases logistics and cost have made adding AES security for bitstreams more challenging, especially for high volume products or products with multiple contract manufacturers in multiple countries. AES security keys are generated by the FPGA tools and delivered as plain-text SVF programming files. Cloned products and product over builds are done by the unscrupulous contract manufacturer in receipt of the files needed for manufacturing or in some cases by ex-employees of those CM's with access to the gerber board files, bitstreams and security keys. Security keys can be programmed in by a trusted third party, but this has added cost and logistics. If the security keys are battery backed up keys they must be programmed after the PCB is assembled. This is not challenging with a single PCB that you can program yourself in the lab. However for a global product with volume production, or regional manufacturing it can be logistically challenging.



Hacker Gets Linux to Run on Xbox; Lays Claim to $100,000 Prize

Authored by Mark Hefflinger on March 31, 2003 - 3:45am.

San Francisco -- A hacker has successfully been able to enable Linux software to run on an unmodified Microsoft Xbox video game console, making him eligible for a $100,000 prize offered by MP3.com founder and current Lindows CEO Michael Robertson, CNET News.com reported. A group of programmers calling itself the Xbox Linux project organized the challenge, which was met by a hacker using the name "Habibi-Xbox." The hacker discovered a bug in the popular game "007: Agent Under Fire" that allows the Linux operating system to be uploaded onto the Xbox. Microsoft has targeted companies selling "mod chips," or aftermarket devices that allow Linux, or possibly pirated games, to run on Xbox by altering the device's hardware; the contest's winner was successful in finding a way to run Linux on the Xbox without altering any hardware. http://news.com.com/2100-1043-994794.html?tag=cd_mh

add new comment | *read more*

tags: Xbox | Hacker | Linux |

Figure 2. Legitimate businesses encourage hacking

AES decryption in the FPGA does not always prevent hackers from programming in a non-encrypted bitstream. Battery backed key methods which prevent non-encrypted bitstreams are easy to defeat by removing the battery or shorting the battery temporarily until the battery is dead. Non-volatile key storage that allows non-encrypted bitstreams to be programmed also allows non-trusted bitstreams to be programmed in. Key storage that does not allow non encrypted bitstreams to be programmed also creates challenges during production test when FPGAs need to be configured in different ways to increase test coverage. Test engineers may not be privy to the security keys especially when they are working in a company separate from the one responsible for the design. Currently only 'high end' FPGAs support AES encrypted bitstreams, another method must be used for the other FPGAs. Application notes from Altera and Xilinx show alternative methods for protecting bitstreams from copying using an external security device from Maxim.

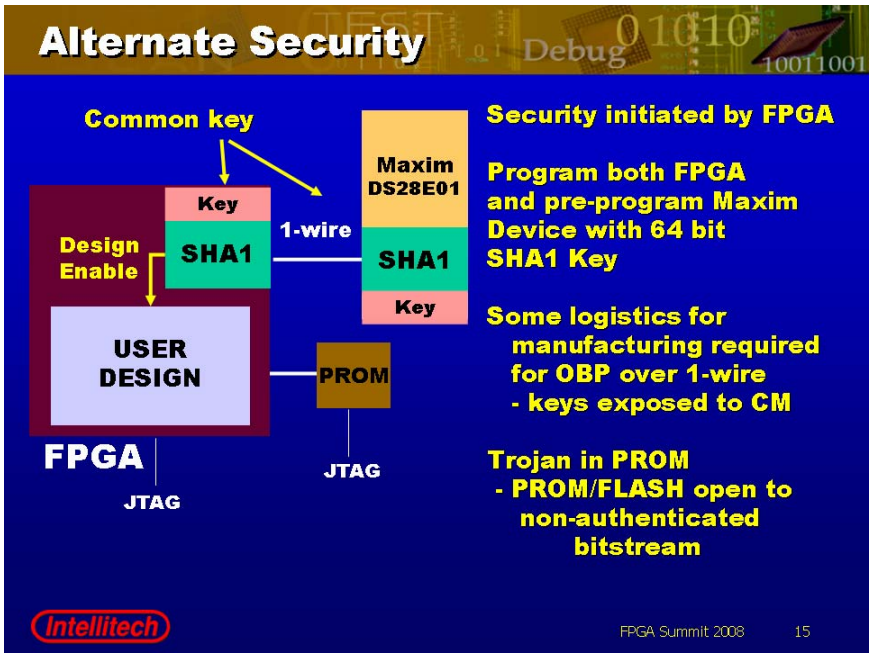


Figure 3. Maxim device doesn't prevent hacks or trojans

This method helps but requires either providing the key to the contract assembler, additional logistics for pre-programming the device or programming through a trusted third party after assembly. After these efforts and costs are expended, it remains easy to defeat by a hacker who programs the bitstream storage with an unauthorized bitstream which intentionally doesn't interact with the security device. The hacker looking to install different bitstreams, potential look-alike bitstreams which maybe Trojans or for other reasons is not interested in deciphering your bitstream but to make use of the platform you have developed for nefarious purposes.

Open, remote in-the-field updates which send bitstreams over the internet or use an ad-hoc updating mechanism to re-program bitstreams into well understood commodity flash also adds obvious security holes in making your product hacker resistant. At one time a FPGA programming language called "JAM" and later "STAPL" was being used by a few in the industry. It suffered from security problems in that FPGA updates in the field were plain text files. It further invited hacking as the embedded "STAPL player" would execute any STAPL code provided to it. STAPL further fell into disfavor as FPGA densities increased; loading 85 megabit FPGAs via JTAG took too long for many applications.

iPhone unlocked by 17-year-old hacker

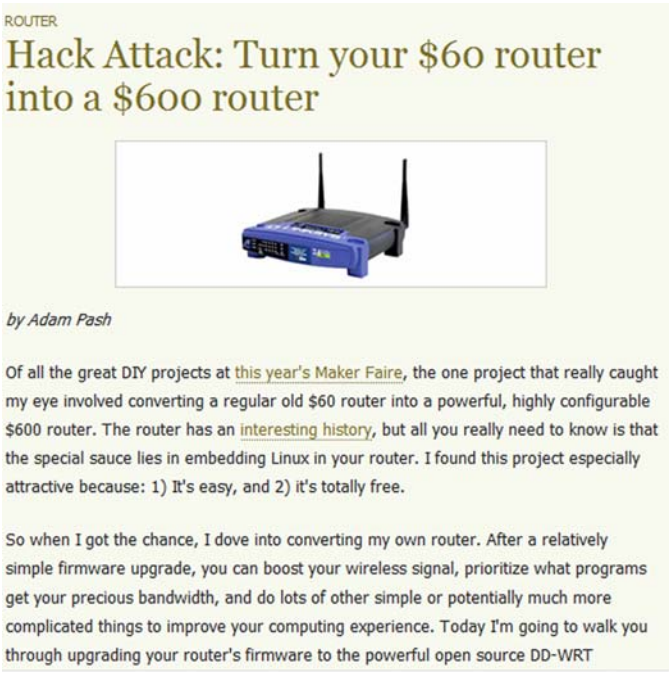
Posted on 25.08.2007 at 10:32 in Tech News by Martin

George Hotz remembers taking apart his first computer, an Apple II, when he was 4 or 5 years old. He cracked open an answering machine, remote control, vacuum cleaner and more computers. He scavenged for more products to tinker with on trash night in his neighborhood. Now the 17-year-old from Glen Rock, N.J., has reached the big leagues of hacking. He says he has "unlocked" the iPhone, finding a way to get around the device's restrictions and allow it to be used not only on AT&T's cell phone network but also on T-Mobile's network and overseas. Until now, however, the iPhone has come with a catch. Because of a revenue-sharing agreement between Apple and AT&T, the iPhone operates only on AT&T's network and requires a two-year subscription.

Figure 4.

Hackers cost the company more than just embarrassment. Some engineers may believe it is OK if the product is hacked. One engineer said to me, "We're still making money; they have to buy the product first to hack it". However, many business models are designed such that the hardware is sold at very low margins and become the platform that a company can make higher margins with the 'consumables' that go with it. Consumables could be games, songs, printer cartridges etc. Should engineers then spend more time on custom hardware and software to prevent hacks? Very competent engineers working in very recognizable companies, with more than reasonable time budgets have created products that a seventeen year old can hack, see Figure 4. Security from modifying the software and hardware in the product, even with the best intentions, can be difficult to achieve from an in-house design. This maybe better achieved by using components specifically designed to prevent hacking of your hardware.

Other types of hacks may circumvent controls on revenue generating features, enabling use of the features without paying for them. Freely downloadable software from the internet can enable average users to turn on features within a product that they didn't pay for, thus creating lost revenue for the company. Is your FPGA based PCB safe from this type of hack? FPGA based products are particularly vulnerable to hacking due to the open nature of the methods of configuration and the bitstreams themselves. There are many application notes from the FPGA vendors showing how to attach a NOR flash to an FPGA or use a CPLD and FLASH to load bitstreams. However, bitstreams stored in commodity memory are particularly vulnerable to hacking as the formats for programming the memory and the ability to re-program it with JTAG is not particularly challenging. Hackers will use the same tools you use to load bitstreams on your PCB.


ROUTER

Hack Attack: Turn your $60 router into a $600 router

by Adam Pash

Of all the great DIY projects at this year's Maker Faire, the one project that really caught my eye involved converting a regular old $60 router into a powerful, highly configurable $600 router. The router has an interesting history, but all you really need to know is that the special sauce lies in embedding Linux in your router. I found this project especially attractive because: 1) It's easy, and 2) it's totally free.

So when I got the chance, I dove into converting my own router. After a relatively simple firmware upgrade, you can boost your wireless signal, prioritize what programs get your precious bandwidth, and do lots of other simple or potentially much more complicated things to improve your computing experience. Today I'm going to walk you through upgrading your router's firmware to the powerful open source DD-WRT

Figure 5. Lost Profits when products are hacked

Other types of hacks can provide access to security passwords within a product. The company experiences revenue lost when customers feel uncomfortable about the security of the product or customer have a negative experience with the product when the product is compromised.

Figure 6. Security is compromised

Test? That's not my responsibility

Test is also one of the costs that may not be completely understood during the design phase of an electronic product. Companies don't ship products which are not tested. A product that is

delayed shipping by a week due to a test engineering problem cost the company the same amount of money as any other problem, an extra week of company wide expenses against the prior product's profits. This statement can be scaled of course for larger or smaller companies. In a larger company, there maybe business units with profit and loss responsibility, so it would be 'company wide' expenses but a week of business unit expenses.

	Product A	Product B	% chg
	Q4 2008	Q1 2009	
Sales	$ 7,500,000.00	$ 6,875,000.00	8.33%
Costs	$ 6,000,000.00	$ 6,000,000.00	
Earnings	$ 1,500,000.00	$ 875,000.00	41.67%
one week late cost		$ 625,000.00	

Figure 7. The real cost of 1 week of engineering time

In Figure 7, the company has quarterly sales of product A at 7.5 million dollars. Obviously many companies have more in sales or more products at one time; the example is simplified to show that as time marches on, the weekly, monthly and quarterly expenses of a company quickly work against the bottom line. In the example, Product B was to be introduced on January 1st, 2009, it was to replace product A. In the real world, of course, we have gradual slow downs of products and gradual ramp ups of new product sales but to make the numbers understandable, the products instantly ramp off and up. When Product B, misses its production schedule, it's delivery to distributors or retailers, by one week, then the sales drop by 8.33% but the profits for the quarter are negatively impacted by 42% in this example. There maybe additional penalties to the company for being late based on contracts the company may have with retailers and distributors. A manager may tend to think in terms of 40 hours of engineering time multiplied by fully loaded engineering hourly rates, but the cost is much higher.

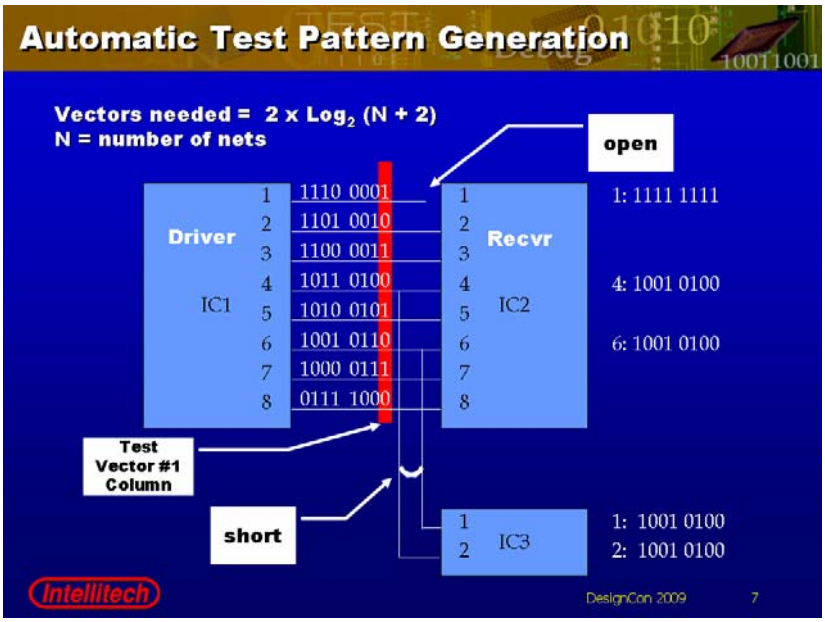


Figure 8. Netlist + software = Automated Test

IEEE 1149.1 based test, sometimes called JTAG, is usually thought of very late in the product development cycle. However, the earlier 1149.1 based tests are prepared the more value the company can extract from the investment. It's an invaluable tool to have during the prototype debug since it can provide shorts and opens testing for many of the BGAs on the PCB. It's not pogo-pin fixture based so there is little risk in having 1149.1 based tests developed as soon as the schematic capture is ready. In fact, PC based software can perform DFT Analysis during the design phase to determine fault coverage and where pads and vias are not needed during layout due to the coverage. IEEE 1149.1 tests are automated today. As shown in Figure 8, a netlist is imported into the software and patterns are generated automatically. Many of the ICs do not need to have IEEE 1149.1/JTAG and still tests can be generated automatically. FPGA to DDR memory test is also automatically generated with full fault stuck-at fault coverage. The skill level and costs required to generate the tests is a fraction of the costs for a software specialist to generate functional tests. Our service group in India develops tests for customers around the world at a $15/hr rate with maybe 10%-15% of management overhead. This compares very favorably with in-house functional test development costs. When tests are developed the software provides a fault coverage report to the pin level, another advantage over manually created tests. Further, software tools use the 1149.1 architecture to also provide diagnostics automatically, also speeding deployment of test for your product (Figure 9).

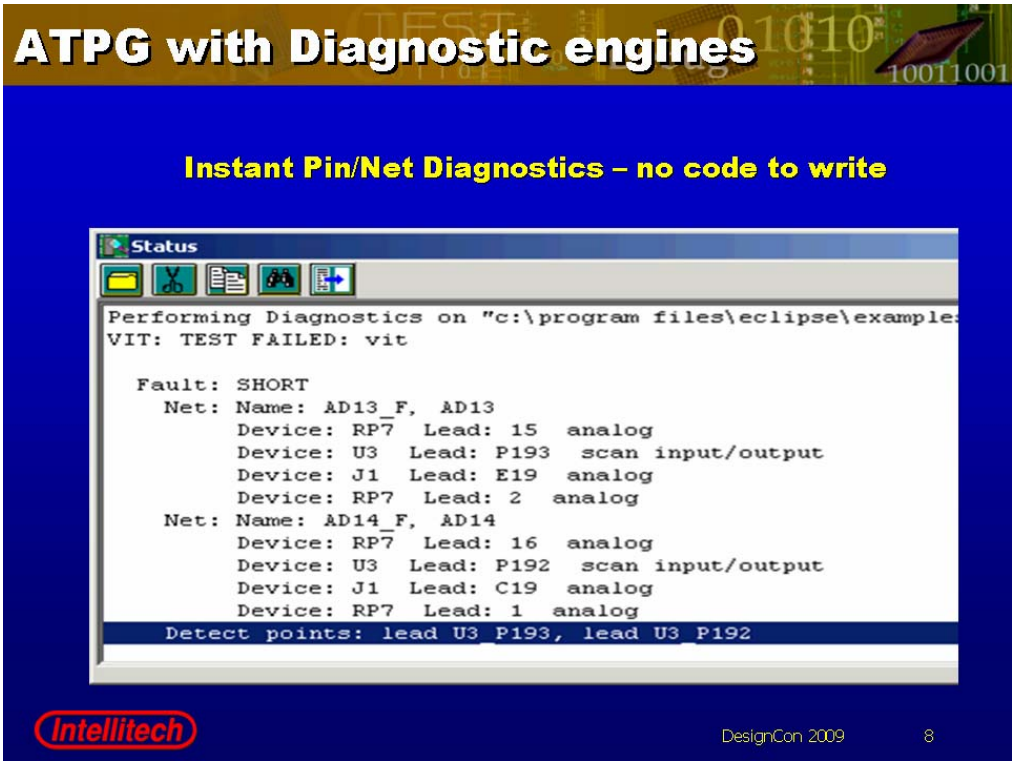


Figure 9. Pin Level Diags for all JTAG/1149.1 tests

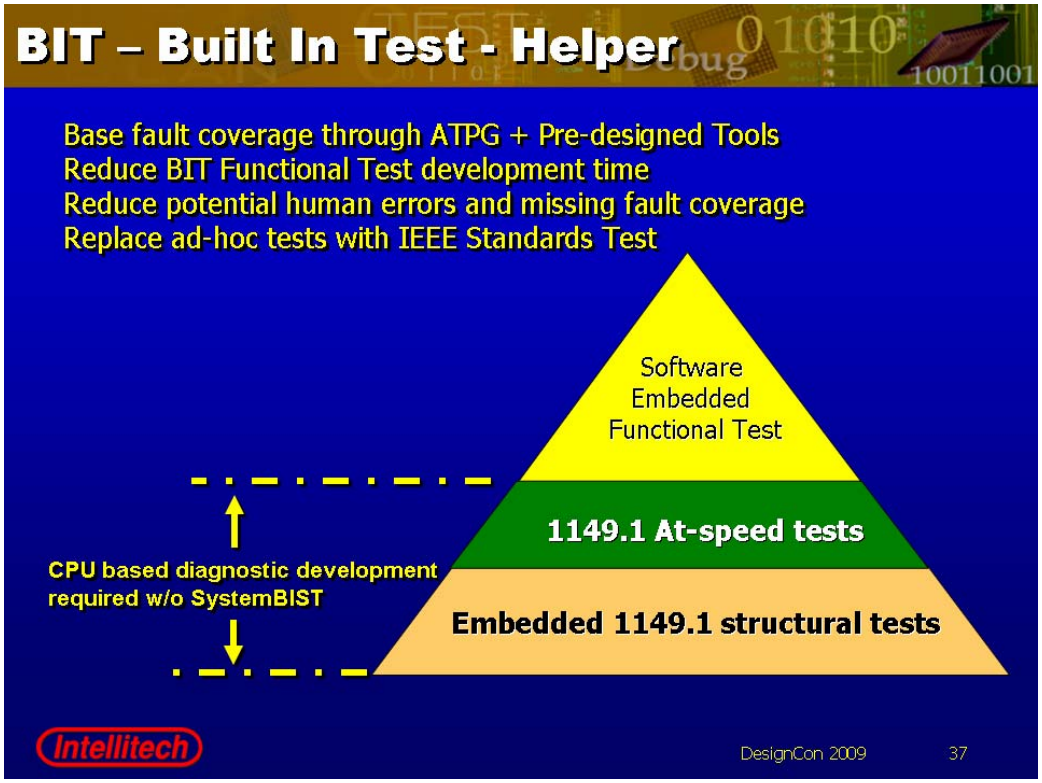


Figure 10. Reduced Functional Test Effort

Embedded software based test development using the mission mode CPU continues to grow in complexity. The mission mode function of the PCB is not well known outside of the originating company, CM engineers are not trained on developing the tests or debugging the functional test failures. It takes in-house resources to develop the tests, high level engineers which could be put towards higher value functions if a more structured approach was taken. In order to standardize development and possibly use third parties for development, it becomes necessary to separate out the mission mode software from the embedded test strategy. Mission mode software based test should start at a higher level, layered on top of 1149.1 structural tests and 1149.1 at-speed tests as shown in Figure 10. At-speed tests are performed by downloading test instruments into the FPGAs such as a Bit-Error-Ratio test for SERDES channels or Memory BIST (Built-in-Self-Test) for at-speed testing of DDR2/DDR3 memories, or even CPU based tests which are controlled by JTAG/1149.1 called 'emulation functional test'. Companies such as Intellitech offer the bitstreams pre-built for these test functions. Since the bitstream is not tied into the mission mode functionality of a system, it has universal use across many products. ASICS also contain JTAG executable on-chip BIST for logic, PLLs, on-chip memory and off-chip memory. Embedding the JTAG tests make it very easy to get high-speed tests that are debugged into the system without writing extensive software to do the same tests functionally. The newly emerging IEEE P1687 standard, sometimes called IJTAG will increase the use of more on-chip instruments accessible by JTAG. FPGAs have on-chip temperature and voltage monitoring through JTAG as well.

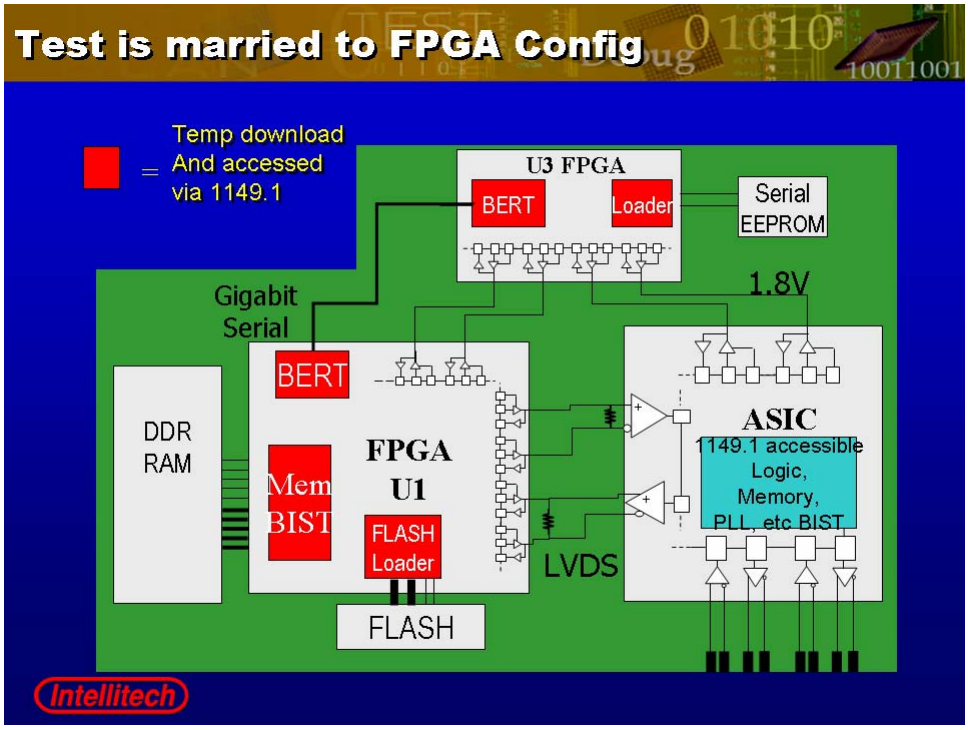


Figure 11 At-speed tests controlled by JTAG

Using the mission mode CPU to execute all possible tests creates the commonly found problem of having a single data point, not knowing whether the failure is in the software or the hardware. Separating out the embedded test and FPGA configuration infrastructure from the mission mode allows not only outsourcing of the development but also a system of checks and balances when failures in the field are encountered. A system that can store the failures in the field for later analysis eliminates the NFF, No Fault Found, enabling feedback to improve the product.

Putting it all together

A proposed solution is the SystemBIST IC which provides much of the on-PCB ecosystem needed for complex FPGA systems. PC based software is used to develop the ecosystem operation and strategy, and then the binary representation of that operation is downloaded to the device. SystemBIST provides parallel configuration of Altera and Xilinx FPGAs and JTAG based FPGA configuration. The designer has GUI access to describe how the IC configures the FPGAs and which bitstreams are to be used to configure them. Manufacturing tests based on JTAG can easily be imported from 3rd party JTAG providers or can be developed within the software itself. These tests would be used at prototype bring-up and re-used during manufacturing and in the field. When failures are found by SystemBIST they are stored in non-volatile memory for later retrieval or the failures can be delivered to Intellitech's website to retrieve a pin level fault diagnostic.

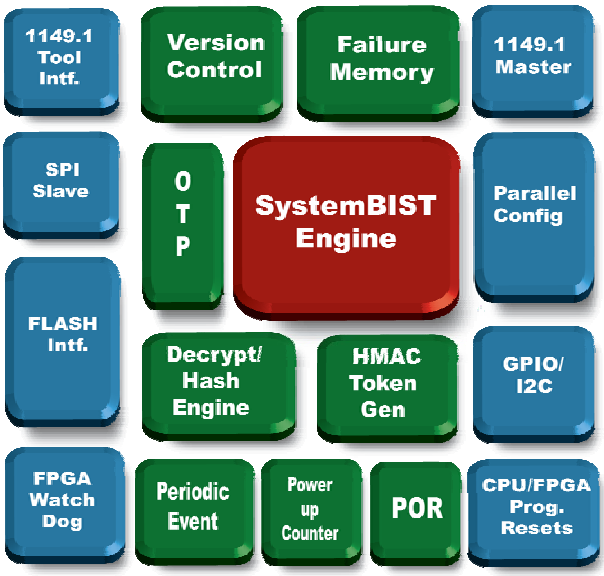


Figure 12. SystemBIST IC Block Diagram

The device provides a built in power-on-reset and programmable control of the board level resets. The device provides a user programmable watch dog for the FPGAs or the CPU. Rather than simply toggling reset the user can define a sequence of events to perform when the watch dog kicks such as saving FPGA registers, re-programming FPGAs or scanning the internal CPU register. Then the CPU is reset after these user defined events have been executed. The I2C and GPIO can be used for power sequencing DC/DC converters and programming adjustable DC/DC converters for voltage margining.

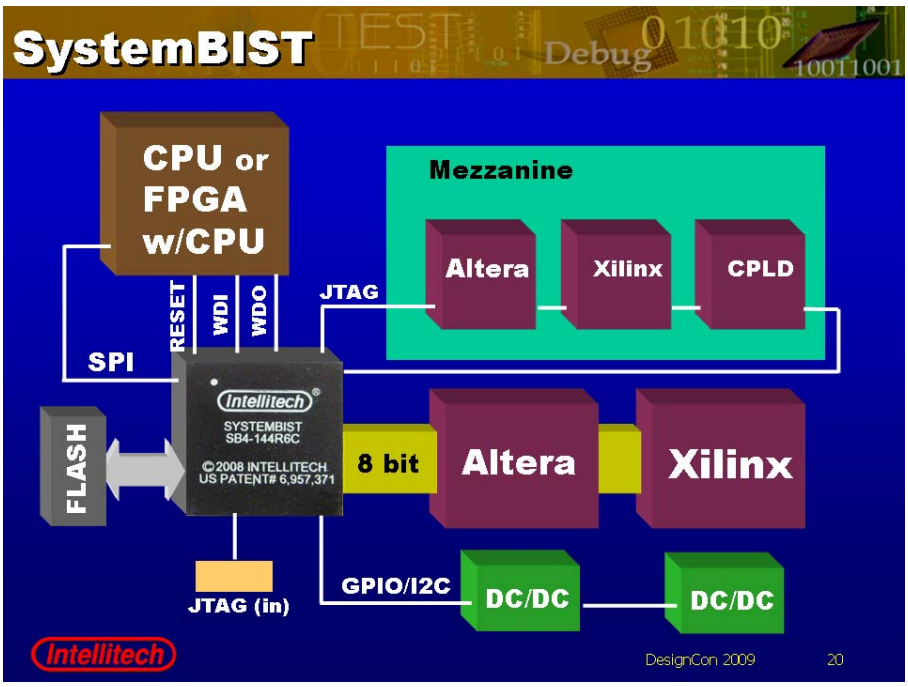


Figure 13. PCB Management with SystemBIST

SystemBIST can operate autonomously at power-up or can accept commands over the SPI (Serial Peripheral Interface). Updates in the field are done via the SPI and the on-chip version control checks for valid update images which can contain FPGA bitstreams, updated JTAG tests or even new CPLD designs. The software tools generate the update images, which are typically smaller incremental images containing new bitstreams and perhaps new test programs. The mission mode software for delivering the updates is relatively straight forward. The CPU takes the update image over a medium (internet, wireless, USB drive) to get it in the system and then opens the SPI port and streams the file into SystemBIST. SystemBIST manages the rest of the checking for authenticity and leaves results for the CPU to access over SPI.

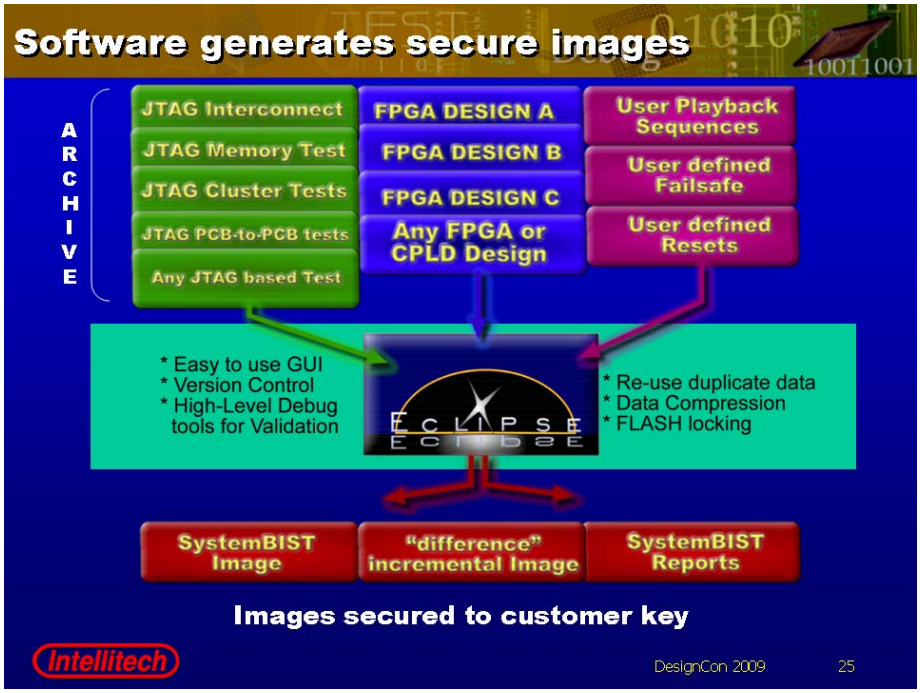


Figure 14. Software Block Diagram

Robust PC based software is a key differentiator from in-house designs and ad-hoc approaches. SystemBIST software generates or imports all of the JTAG based tests, it also allows the user to specify what FPGA designs are used, what FPGA designs must be failsafe and allows the user to define complex playback sequences that occur at power-up. Failsafe bitstreams are bitstreams which are loaded into a critical FPGA when the configuration fails. These bitstreams, and only these bitstreams are stored in a protected area of FLASH. Consider designing a PCIe based card with three FPGAs on it. One FPGA is the interface FPGA to the PCIe bus of the card. This FPGA is critical in that it must always be available otherwise the PCIe interface is lost. The bitstream for this FPGA would be marked 'failsafe' so it would always be available, even in the event the FLASH was corrupted during an update process.

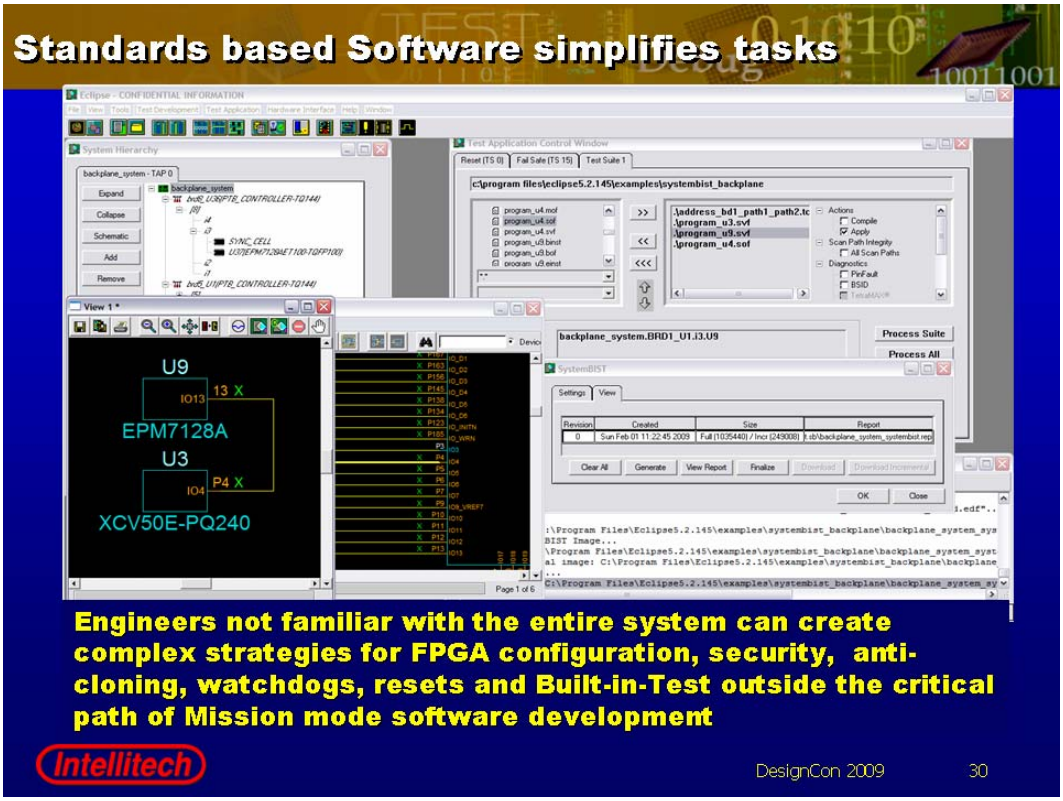


Figure 15. PC Software for Config/BIT/Security

SystemBIST contains a unique serial number and customer code in its one time programmable memory. This makes SystemBIST a physically un-clonable device. Non-authorized parties cannot obtain a SystemBIST IC with another customer's unique code. FPGA bitstreams and JTAG operations which are in the binary image are encrypted with two 128 bit keys and tied to the customer code. Anyone with the Intellitech software tools cannot generate compatible bitstreams without being authorized to do so. All SystemBIST bitstream images are coded and protected using the unique customer code and keys. SystemBIST can recognize bitstream images which are not authentic from the original source, or bitstream which have been tampered with. Bitstream storage is not erased unless the update image containing the new bitstreams passes the authenticity checks. Serial numbers and customer codes are available to the mission mode CPU over the SPI bus, however the keys for obvious reasons are not available. SystemBIST is compatible with AES encrypted bitstreams from Xilinx and Altera. Those methods can still be used. However, SystemBIST takes a more active role in checking for FPGA bitstream authenticity. SystemBIST passes tokens via JTAG to FPGAs that include a small design for hashing a unique response preventing non-authorized bitstreams from being present and protecting bitstreams from copy/reuse. The user can program via the SystemBIST software GUI the operation that occurs in the case of an incorrect response. This could be something as simple as resetting the FPGAs and re-programming it or something more complex such as shutting off the DC/DC converters.

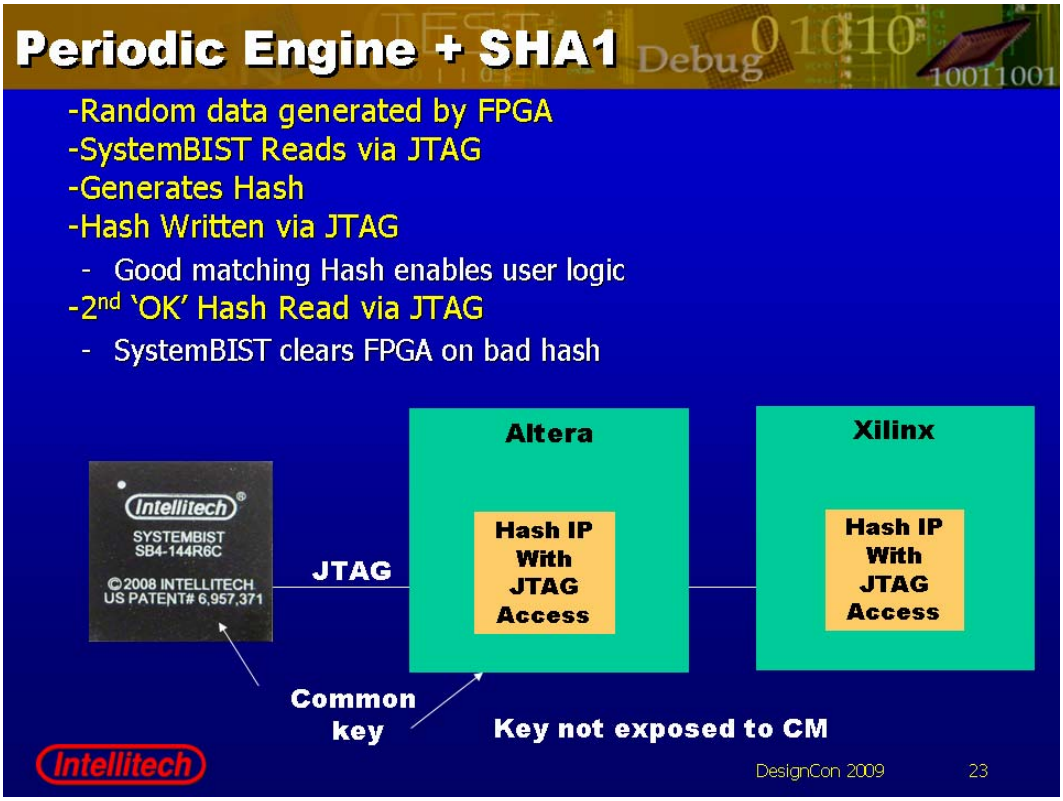


Figure 16. Checking for bitstream Authenticity

The periodic engine is designed to allow for this repeated checking, it can also be programmed to perform I2C functions such as voltage margining or FPGA configuration checks. For instance, the CRC_CHECK pin of Altera Stratix could be periodically scanned via JTAG SAMPLE to check that an SEU has not occurred. Since SystemBIST is stand-alone, this can be done without using mission mode CPU resources. SystemBIST would reprogram the FPGAs in the event the CRC_CHECK indicated an SEU event occurred. Another example use of the periodic engine is to have SystemBIST use JTAG to constantly monitor on FPGA temperature or on-chip voltage such as provided by Xilinx's new Virtex-6 family.

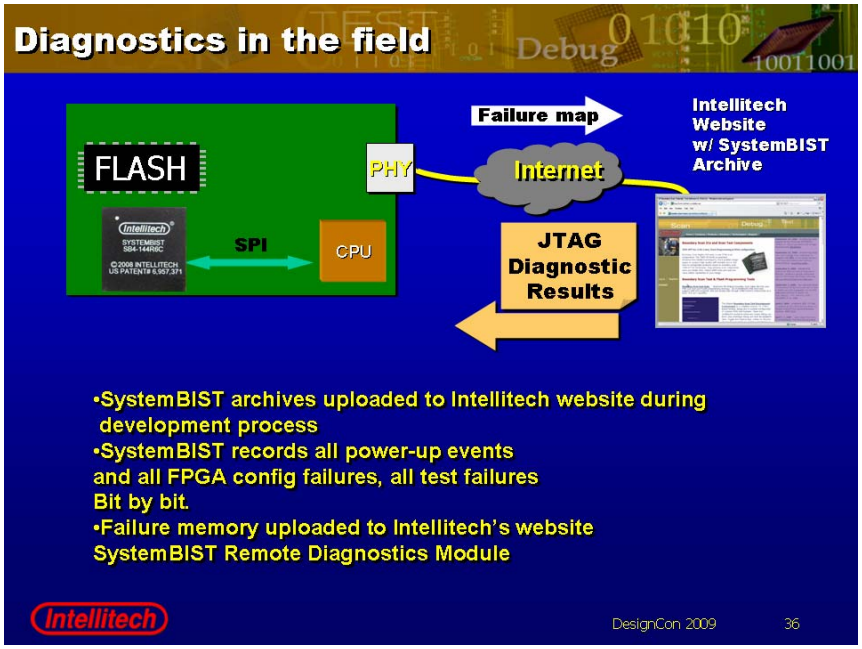


Figure 17. SystemBIST compatible with remote diagnostic engines

SystemBIST stores a diagnostic code for each test or FPGA configuration failure that occurs. The mission mode CPU can access this failure over SPI or the CPU can retrieve a full failuremap from SystemBIST to perform detailed diagnostics. Diagnostics in the field are not done by embedding the diagnostic engines into the mission mode platform. All diagnostics are done offline using Intellitech's off line diagnostic engine. During development of the SystemBIST image, an archive of all of the data files is made. Failures in your system are processed and diagnostics are retrieved (typically over a HTTP connection to a website) the same as if you had a PC running 1149.1/JTAG or FPGA configuration software.

Conclusion

I recall a conversation I had some seven years ago with a group of designers at a telecom company in North America. At the time I was presenting on the advantages of using our first generation SystemBIST IC for enabling plug-n-play built-in-self-test through JTAG for their PCBs. "But we write software to do that" one firmware engineer quipped, as if that was not obvious and the software would appear at zero cost to the company by saying this. "Yes", I said, "but software has costs as well; there is firmware engineering time to write that software, software maintenance, bug fixes, release management, engineering turnover requiring relearning and knowledge transfer, integration with the hardware, debug and validation. Since the software is integrated with the mission mode of the PCB, the cycle of develop-debug-validate starts all over again with each subsequent product." I was told "If you write the software right, you won't have any further work to do, it will be completely reusable." I'm completely aware of object oriented programming and source code reuse for applications running on standardized platforms, but when software talks with new hardware, new technologies, there is work to be done. Each generation PCB has new processors, new support ICs, new technologies that are difficult to account for during first generation software

development. In this economic environment, one may even question the wisdom of adding to the software development time such that it will be bullet-proof for any new technology that may arrive on generation two and generation three of the product. There may not be a 2nd or 3rd generation product if the first generation is delayed.

A few years after these discussions, we were selling first generation SystemBIST ICs into an India based company. They found it cheaper to put the ten dollar part on their PCB to re-use their manufacturing tests on-board then invest in in-house techniques for embedded test. As it turns out they were producing a PCB assembly for a new product to be used by the aforementioned telecom company. But, unfortunately, it was a little too late; the telecom company recently just filed for chapter 11 bankruptcy protection. I'm not saying that failure to use SystemBIST caused the bankruptcy but I am saying it was partly cultural. Within the company there was lack of focus on their core competency, choosing to build large engineering teams in-house rather than buying or outsourcing non-core functions.

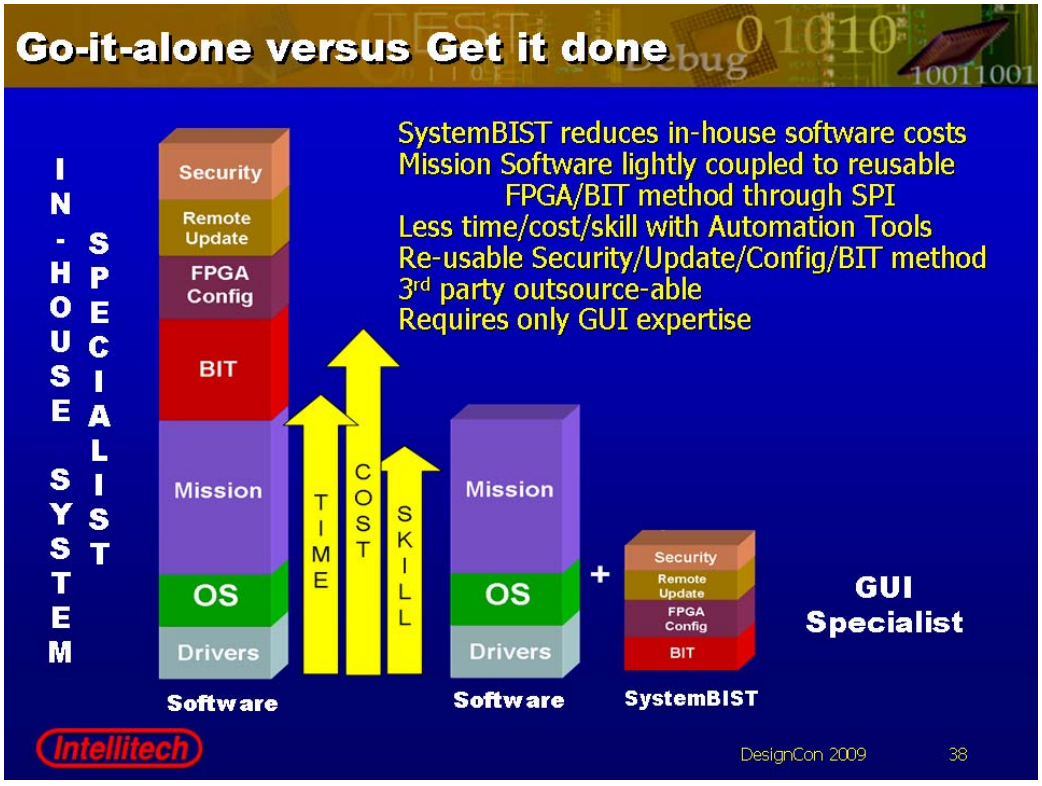


Figure 18. Left side is software only, right reduced software & SystemBIST

SystemBIST has more benefit than just reducing the parts on the PCB, but also the advantages of offloading the mission mode software development. On the left side of Figure 18, the mission mode software is integrated with the Built-In-Test (BIT) strategy which is also integrated with the FPGA configuration, the remote updating for FPGAs and CPLDs and the bitstream security. Many of these items have linear time dependencies; the BIT is not going to be complete until the FPGA configuration can be done. The left side 'go-it-alone' in-house approach typically requires a very knowledgeable, skilled system software developer(s) to meet the system objectives. On the right hand side we see how SystemBIST breaks these functions apart, leaving the mission mode software separate from the FPGA configuration and

BIT strategy. SystemBIST allows a less skilled person to work independently to implement the designer's FPGA configuration strategy, using pre-developed GUI based tools. There is little integration with the mission mode software other than to interface to the SystemBIST SPI interface. Since expertise in the mission mode of the system is not needed, turn-key third party services can be used to deliver much of the FPGA and BIT related components that previously had to be developed internally.

DesignCon attendees can contact the author for access to the power point slides that go with this paper.

Bibliography:

Using the Design Security Feature in Stratix II and Stratix II GX Devices, Altera Corporation, July 2008. http://www.altera.com/literature/an/an341.pdf

Trusted Design in FPGAs, Steve Trimberger, Xilinx, Design Automation Conference, 2007
http://videos.dac.com/44th/papers/1_2.pdf

Authentication of FPGA Bitstreams:
Why and How, Saar Drimer, ARC 2007
http://www.springerlink.com/content/t71pqn4g7565w806/

A Code-less BIST Processor for Embedded Test and in-system configuration of Boards and Systems, CJ Clark, Intellitech Corp, Mike Ricchetti, ATI Research, ITC 2004,
http://www.intellitech.com/pdf/itc04sb.pdf

Design Security in Stratix III FPGAs, Altera Corporation
http://www.altera.com/products/devices/stratix-fpgas/stratix-iii/overview/architecture/st3-design-security.html

Secure Update Mechanism for Remote Update of
FPGA-Based System, Benoît Badrignans1,2, Reouven Elbaz3 and Lionel Torres. SEIS 2008,
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/4569831/4577669/04577703.pdf?temp=x

Physical Unclonable Functions for Device Authentication and Secret Key Generation
G. Edward Suh, Srinivas Devadas
http://videos.dac.com/44th/papers/1_3.pdf

Xilinx® FPGA IFF Copy Protection with 1-Wire SHA-1 Secure Memories, Maxim,
http://www.maxim-ic.com/appnotes.cfm/an_pk/3826

An FPGA Design Security Solution Using a Secure Memory Device, Altera,
http://www.altera.com/literature/wp/wp-01033.pdf

Altera Configuration Handbook
http://www.altera.com/literature/lit-config.jsp

Xilinx Virtex-5 FPGA User Guide
http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

US Patent 6,957,371- Method and apparatus for embedded built-in self-test (BIST) of electronic circuits and systems
http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=10&f=G&l=50&co1=AND&d=PTXT&s1=Intellitech&OS=Intellitech&RS=Intellitech

US Patent 7,467,342 - Method and apparatus for embedded built-in self-test (BIST) of electronic circuits and systems
http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=2&f=G&l=50&co1=AND&d=PTXT&s1=Intellitech&OS=Intellitech&RS=Intellitech