

White Paper

Secure Processors for Embedded Applications

Black Hat DC 2007

Prepared by: James D Broesch

February 28, 2007

jbroesch@ucsd.edu

858-964-6842

© James D. Broesch, 2007

Introduction: There are many aspects to security in computing environments. In this paper we look at some of the options for incorporating security at the hardware level by using variety of secure processor options. Traditionally, developers have relied on FIPS / Common Criteria compliant processors when looking to secure processors. While there are many advantages to the use of these standard devices, it is also true that these devices are often relatively limited in their processing capability. In this paper we will look at number options for implementing high performance secure embedded processing. The options discussed range the use of simple security monitoring circuits through high performance FPGAs designed for secure applications.

Background: The maximum practical clock speeds for silicon based processors has probably been reached; until completely new materials and techniques are developed the clock speed on conventional processors will see no significant improvement. As a result of reaching the practical clock limits, as well as having reached the point of diminishing returns on conventional computer architectures, alternative computing architectures have been undergoing both a change in nature and significant growth in potential. This has significant impact on the design of high performance embedded applications in general, and secure embedded applications in particular [1]. These advances can be described under the general heading of Systems on a Chip (SoC). Specifically, the direction of advance has been along two lines: 1) multi-processor cores and 2) advanced Field Programmable Gate Arrays (FPGAs.) Particularly in the case of

FPGAs, which often contain processors, block RAM, general computing resources, and a large amount of user customizable logic, there can be significant overlap between the SoC and FPGA methodologies.

If the requirements for advanced computing and networking were merely a matter of computation performance, the standard make-verses-buy decisions of value engineering approaches would apply. However, the increasing reliance on commercial application of the internet, net-centric warfare, high performance sensors, and autonomous and semi-autonomous systems is rapidly changing the way systems are developed, procured, maintained, and deployed.

In the commercial sector, the utility of the net has been significantly compromised by various viruses, worms, denial of service (DoS) attacks, identity theft, and other forms of unauthorized use. To some extent, military networks do not face the same problems. On the other hand, the consequences of a successful penetration of a military network can have far more significant consequences. In the commercial sector these security issues have been largely addressed with a “band-aid” approach to patching the common commercial or open source operating systems combined with encryption of the network traffic. The encryption of the traffic has been to widely varying levels of effectiveness. The security of the underlying computational platforms has generally been ignored. Where applied, physical security has often been based on the physical security of the server, trusted operators, and a variety of ad-hoc security processes. In some applications,

commercial security modules such as IBM's Cryptographic Coprocessor Security Module (Model 4764-001) [2] have been applied to otherwise open conventional systems to provide a degree of enhanced security. These classical approaches become more problematic as ubiquitous computing becomes an ever greater part of our world. Whether it is an executive's laptop that is left in a taxi cab, or the sensor payload of a UAV lost over an adversary's territory, physically protecting the asset is becoming ever more of a challenge.

It is also often an overlooked fact that to achieve a high security embedded processing environment it is also often necessary to achieve a high reliability computing environment. In those cases where commercial computer networks require high reliability, industry has largely adopted an N+1 sparring strategy where computing "blades" are simply replicated until the desired degree of reliability is achieved. For embedded applications the high reliability requirement may mean the use of redundant processors, multiple core FPGAs, reconfigurable devices, or single event upset detection (SEU)[3,4], or other such techniques.

The Need for Secure Processors.

Secure processors are becoming an ever more important design consideration in the design of modern systems.

Key motivations include:

- Simple paranoia
- IP / design protection
- Protection of sensitive (i.e. trade secrets, business plans, etc.) data.

- Protection of legally liable data (i.e. medical records (HIPA), financial transactions (SOX), etc.
- Meeting ITAR or other regulatory agency compliance requirements.
- Classified information (which, needless to say, we will not be discussing.)

The consequences of failing to secure a design are equally wide ranging:

- Having to deal with a relatively minor virus (though on an enterprises wide basis even this can be a fairly significant expense.)
- Loss of trade secrets or loss of significant NRE investments by allowing competitors to easily copy designs.
- Spoofing the operation or messages from a remote system.
- Significant compromise of corporate and / or nationally important information.

The optimal system design becomes a tricky tradeoff between technical feasibility, cost, schedule, level of effort, and required degree of practical security.

Options for Secure Processing.

As the need for secure embedded processing increases, the increasing technological capability (both hardware and software) have allowed the marketplace to respond with a wide range of options for meeting the requirements of a secure processing environment.

Custom ASICs: These can be the most secure method for implementing high performance embedded applications if

rigorous design methodology is followed. This has been the traditional, but most expensive and time consuming, route to a secure processing environment.

Security Monitor ICs: The most basic approach is to use an IC dedicated to building secure modules. One such example is the Dallas Semiconductor's DS3600 Secure Supervisor. These types of devices provide environmental monitoring, key management, and other security related features that assist in providing tamper resistance (AT) and information assurance (IA). Their primary weakness is that once successfully penetrated, the traffic within the module is visible and any relevant information can generally be obtained by a variety of attacks. This will generally be true even though the cryptographic keys or other key security information may remain safe in the security monitor device itself.

Secure Microcontrollers / Coprocessors: One step above the security monitor is the secure processors. Generally, these devices can be used as standalone secure processors for relatively low performance systems or as secure coprocessors for higher performance conventional processors. As secure processors these devices are generally rather secure. When used as secure coprocessors the same comments apply as for the security monitor ICs. An example of a secure microcontroller / coprocessor is the Maxim's DS5250, a secure processor based on the venerable 8051 architecture.

Secure enabled FPGA: Recently, the requirements for high performance, high reliability, and security, have become

practical to address with modern FPGAs. Traditionally, FPGAs have been considered extremely vulnerable from a security perspective due to the fact that most versions required an external PROM to load the configuration for the FPGA. Since the contents of the PROM were not encrypted, reverse engineering (RE) the FPGA based designs was simply a matter of recording the load sequence of the FPGA. With this information the function of the FPGA could be easily copied, and the contents of the FPGA could even be "decompiled" into source code form. Modern FPGAs possess the ability operate from PROMs that have been encrypted using 3DES or AES, depending upon the device. The key for decrypting the PROMs on load are either stored statically (that is, no external power is required), dynamically (that is an external battery is required to maintain the key), or on some devices the user may choose between a combination of the two. Examples of static storage are members of Altera's Stratix II family. Examples of the dynamic storage of keys can be found in Xilinx's Virtex II (3DES) and Virtex IV (AES) devices. An example of a device with the option of allowing the user to choose either option is the new Stratix III devices [5].

With dedicated hardware resources such as multipliers, block RAM, hard core processors, and large amount of fabric logic FPGAs can provide both high performance and high security. Particularly interesting are those devices that have dual hard processor cores. Generally, the processors can be synchronized, and the fabric logic can be used to create a comparator that monitors the output of each processor as

they execute parallel, and identical, instruction and data streams. Should any tampering occur with either independent processor, or should some fault occur that would produce conflicting results, the comparator can detect the problem and instigate corrective action [4].

The viability of FPGAs as low cost cryptographic engines has been extensively evaluated by NSA and Xilinx. The results indicate that these devices do indeed make reliable secure processors [6]. Threat vectors from SPA, DPA, DEMA, etc. were rigorously evaluated.

Other Key Factors: The key point here is that we are at a critical inflection point where several factors are coming together that will have a significant impact on many future commercial and military acquisitions. These factors include, but are not limited to:

- As mentioned above, processor clock rates have reached the maximum practical value for silicon. Thus computing improvements will not come from faster processors in the foreseeable future.
- Securing of the hardware and software design of the system is becoming a key requirement for both the protection of intellectual property (IP) and to meet government requirements.
- The need for high performance, low power, high density processing is increasing

dramatically, particularly for autonomous platforms such as UAVs.

- The introduction of new component level silicon to enable addressing these issues will require new design methodologies at the software, firmware, and hardware level.

Reference:

1) *The Past, Present, and Future of High Performance Embedded Computing (HPEC) in Reconnaissance Applications*, James D Broesch, Invited Talk, HPEC 2006.

2) *IBM eServer Cryptographic Coprocessor Security Module Model 4764-001, Security Policy*, 2005.

3) *Single Event Upset (SEU) Detection and Correction Using Virtex-4 Devices*, Les Jones, Xilinx Application Note xapp714, January 2007.

4) *Triple Module Redundancy Design Techniques for Virtex FPGAs*, Carl Carmichael, XAPP197, July 2006

5) *Design Security in Stratix III Devices*, Altera White Paper, November, 2006

6) *FPGA-BASED SINGLE CHIP CRYPTOGRAPHIC SOLUTION*, Mark McLean, National Security Agency, Jason Moore Xilinx Corporation, MILCOM 2006 paper and presentation.

Acronyms

ASIC	Application Specific Integrated Circuit
ASSP	Application-Specific Standard Product
AT	Anti-Tamper
DEMA	Differential Electromagnetic Analysis
DMA	Direct Memory Access
DPA	Differential Power Analysis
EFP	Environmental Failure Protection
EMA	Electromagnetic Attack
HIPA	The Health Information Protection Act
IA	Information Assurance
IP	Intellectual Property
ITAR	International Traffic in Arms Regulation
NRE	Non-Recurring Engineering
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RE	Reverse (Recurring) Engineering
SoC	System on a Chip
SOX	Sarbanes-Oxley Act of 2002, Public Company Accounting Reform and Investor Protection Act of 2002
SPA	Single (Simple) Power Analysis