Rapid Single-Chip Secure Processor Prototyping on the OpenSPARC FPGA Platform

Jakub M. Szefer [*3], Wei Zhang [#1], Yu-Yuan Chen [*3], David Champagne [*3],
King Chan [#1], Will X.Y. Li [#1], Ray C.C. Cheung [#2], Ruby B. Lee [*3]

[#] *Department of Electronic Engineering, City University of Hong Kong*
[1] {wezhang6, kingychan8, xiangyuli4}@student.cityu.edu.hk, [2] r.cheung@cityu.edu.hk

[*] *Electrical Engineering Department, Princeton University, USA*
[3] {szefer, yctwo, dav, rblee}@princeton.edu

***Abstract*—Secure processors have become increasingly important for trustworthy computing as security breaches escalate. By providing hardware-level protection, a secure processor ensures a safe computing environment where confidential data and applications can be protected against both hardware and software attacks. In this paper, we present a single-chip secure processor model and demonstrate rapid prototyping of the secure processor on the OpenSPARC FPGA platform. OpenSPARC T1 is an industry-grade, open-source, FPGA-synthesizable general-purpose microprocessor originally developed by Sun Microsystems, now acquired by Oracle. It is a multi-core, multi-threaded 64-bit processor with open-source hardware, including the microprocessor core, as well as system software that can be freely modified by researchers. We modify the OpenSPARC T1 processor by adding security modules: an AES engine, a TRNG and a memory integrity tree. These enhancements enable security features like memory encryption and memory integrity verification. By prototyping this single-chip secure processor on the FPGA platform, we find that the OpenSPARC T1 FPGA platform has many advantages for secure processor research. Our prototyping demonstrates that additional modules can be added quickly and easily and they add little resource overhead to the base OpenSPARC processor.**

I. Introduction

As computing devices become ubiquitous and security breaches escalate, protection of information security has become increasingly important. Many software schemes, e.g., [1]–[3], have been proposed to enhance the security of computing systems and are effective in defending against software attacks. However, they are generally ineffective against physical or hardware attacks. Attackers who have full control of the physical device can easily bypass software-only protection, and the whole system is left unsafe and subject to hardware attacks. This results in an increasing need for hardware-enhanced security features in the microprocessor.

Considerable efforts have been made to build secure computing platforms that can address security threats. In this paper, we present our extensible secure computing model prototyped on the OpenSPARC FPGA platform. The platform consists of the OpenSPARC T1 processor and system software, including the hypervisor and the operating system. The OpenSPARC T1 processor is the open-source form of the UltraSPARC T1 processor (from Sun Microsystems, now Oracle) that gives designers the freedom to modify the processor according to their own needs [4]. This OpenSPARC T1 processor is also easily synthesizable for FPGA targets, which makes the implementation of the processor quite easy. Field-programmable Gate Array (FPGA) is an integrated circuit designed to be configured by the customer or the designer after manufacture. Because of its reconfigurability, FPGA can be used to implement any logic function that an Application-Specific Integrated Circuit (ASIC) chip could perform, which makes it a good platform for rapid system prototyping.

978-1-4577-0660-8/11/$26.00 ©2011 IEEE

Through the prototyping process, we have found that the OpenSPARC FPGA platform has many advantages for secure processor research. For example, the memory subsystem is emulated in a MicroBlaze softcore, which allows new features to be added without re-synthesizing the whole platform. Furthermore, new hardware components can be easily added as Fast Simplex Link (FSL) peripherals without worrying about strict timing and editing fragile HDL code of the processor core. However, to the best of our knowledge, currently there are only a few papers [5]–[7] about processor research on the OpenSPARC FPGA platform and none focusing on security. In this paper, we propose a single-chip secure processor architecture based on this platform. Furthermore, we find that the OpenSPARC FPGA platform is relatively friendly for secure processor prototyping. Although there are other open-source processors like OpenRISC and LEON available [8], they lack the infrastructure and components (e.g. hypervisor or emulated caches) of the OpenSPARC platform.

The main contributions of this paper are:

- A reconfigurable single-chip secure processor model.
- A prototype of the single-chip secure processor on the OpenSPARC FPGA platform with our new additions: AES engine, true random number generator (TRNG), and memory integrity tree (MIT).
- Evaluation showing the OpenSPARC FPGA platform's advantages for secure processor research.

The rest of the paper is organized as follows. Section II describes some existing secure processor models as background information. Section III proposes our single-chip secure processor architecture on the OpenSPARC FPGA platform. Section IV describes the single-chip secure processor features.

Section V gives an evaluation of the secure OpenSPARC platform. Finally, in Section VI we conclude the paper.

II. Related Work

With the emergence of hardware attacks, hardware-enhanced security has been given considerable attention by researchers and engineers. Different secure processor architectures have been proposed to provide a secure computing environment for protecting sensitive information against both software and hardware attacks.

The single-chip secure processors consider the processor to be trusted, but anything outside the processor, e.g. memory, is untrusted. One such example is the Aegis architecture, as described in [9], [10]. In this approach, the secure processor contains two key primitives: a physical unclonable function (PUF) and off-chip memory protection. Both primitives are realized within one single-chip processor so that the internal state of the processor could not be tampered with or observed directly by physical means.

The Secret-Protecting (SP) architecture is proposed in [11]. In SP, trusted software modules have their data and code encrypted and hashed when off-chip and a concealed execution mode is provided where these software modules are protected from other software snooping on them, e.g. registers are encrypted on an interrupt so a potentially compromised commodity Operating System cannot read them. Also, a hierarchical key chain structure is used to store all keys in encrypted and hashed form and only the root key needs to be stored in hardware.

Another secure processor architecture is Bastion [12] that can protect a trusted hypervisor, which then protects trusted software modules in the application or in the operating system. Bastion scales to provide support for multiple mutually-distrustful security domains. Bastion also provides a memory integrity tree for runtime memory authentication and protection from memory replay attacks. It also protects the hypervisor from physical attacks and offline attacks, not just software attacks.

Based on these previous work, we propose our secure computing model in Section III. Despite the many advantages of the OpenSPARC FPGA platform, it is not very widely used as a research platform. We propose a single-chip secure processor on this platform and hope that our work can provide some reference for researchers interested in OpenSPARC.

III. Single-Chip Secure OpenSPARC Processor

This section presents our secure computing model and single-chip secure OpenSPARC T1 processor.

A. Secure Computing Model

Figure 1 illustrates our secure computing model. We divide this computing system into two parts. The first part includes those components in the processor chip, shown inside the dashed box, and the second part consists of all components off the processor chip, shown outside the dashed box. All on-chip modules, including CPU core, cache, registers, encryption/decryption engine and integrity verification module, are assumed to be trusted and protected from physical attacks in our design, because the internal state of the processor chip could not easily be tampered with or observed directly by physical means. We do not consider such physical means as side-channel attacks that employ differential power analysis or electromagnetic analysis in this paper. On the other hand, all off-chip modules, including external memory and peripherals, are considered insecure because those modules can be easily tampered with by an adversary using physical attacks. In addition to hardware attacks, the system may suffer from software attacks from an untrusted operating system or malicious software.

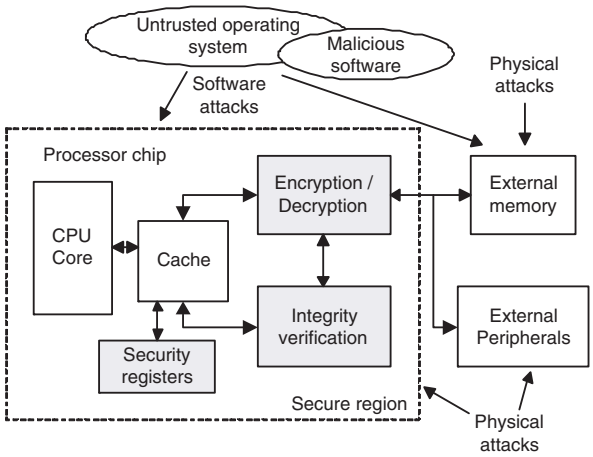

Fig. 1: Secure computing model. The processor chip is regarded as a physically secure region and gray blocks represent new security enhancements.

To ensure a secure computing environment, the computing system must have some secure functions that enable it to defend against either software or hardware attacks. The gray blocks in Figure 1 represent our initial set of new security enhancements that we add to the computing system. The encryption/decryption module encrypts all data evicted off the processor chip so that they are meaningless to an adversary. The integrity verification module verifies that all data coming from the off-chip memory has not been tampered with.

B. OpenSPARC FPGA Platform

Our single-chip secure processor design targets the FPGA-sythesizable version of the OpenSPARC T1 general-purpose microprocessor. The OpenSPARC T1 microprocessor is an industry-grade, 64-bit, multi-threaded processor and is freely available from the OpenSPARC website [4], [13]. In addition to the processor core source code (HDL), simulation tools, design verification suites, and hypervisor source code (C and assembly) are available for download [4]. The OpenSPARC FPGA platform consists of the following major components: OpenSPARC T1 microprocessor core, memory subsystem (L2 cache), DRAM controller and DRAM, hypervisor and a choice of operating systems (Linux or OpenSolaris).

Due to the size constraint of the FPGA chip, the OpenSPARC FPGA platform that we use includes only one

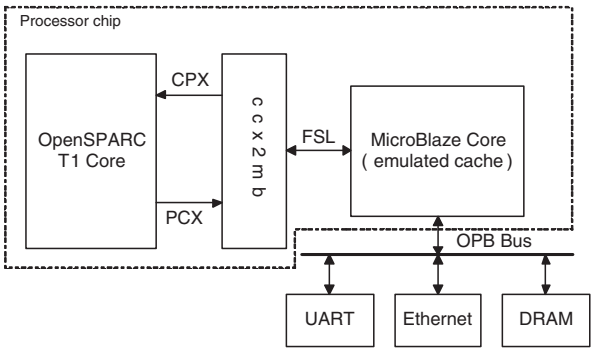

Fig. 2: Block diagram of stock OpenSPARC FPGA platform.

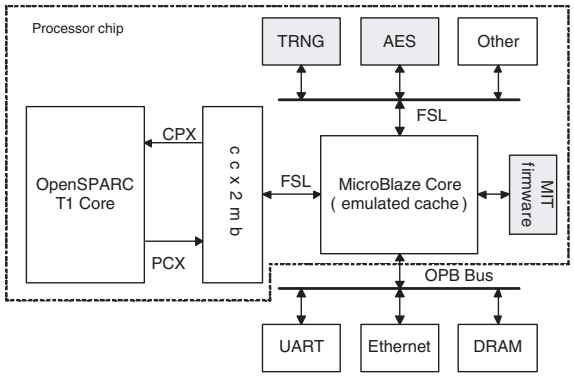

Fig. 3: Block diagram of secure OpenSPARC FPGA platform. Gray blocks show our new additions.

single-thread T1 CPU core to minimize its size. In addition, the L2 cache and the L2 cache controller are emulated in a MicroBlaze softcore, i.e. there is no physical L2 cache. A high-level block diagram of the OpenSPARC T1 processor is shown in Figure 2. The microprocessor core is connected to the L2 cache, emulated by a MicroBlaze softcore, through the CPU-Cache Crossbar (CCX). The DRAM controller is an IP (Intellectual Property) block synthesized and implemented in the FPGA fabric and connected to the MicroBlaze softcore. A physical DRAM is connected to the FPGA board to serve as the actual memory.

Due to the different complexities of these components and to place and route operations that determine critical paths when implementing on FPGA, the FPGA version of the OpenSPARC processor chip has multiple clock domains. The OpenSPARC T1 core is in one clock domain (50MHz), the MicroBlaze softcore is in another clock domain (125MHz). Peripherals synthesized in the FPGA fabric and connected to the MicroBlaze softcore are in another clock domain (e.g. 10MHz or 50MHz for the peripherals we create). Finally, the DRAM chip has its own clock domain (400MHz).

The OpenSPARC FPGA platform has a lot of advantages for security research [14]. It allows users to freely modify real hardware. Especially, the memory subsystem is emulated, rather than being fully implemented in HDL, so new features can be easily added (in C code and run on the MicroBlaze softcore) and re-synthesizing the whole platform can be avoided. In addition, new hardware components can also be easily added as firmware code or synthesized in FPGA fabric and connected to the emulated cache by buses, e.g. by FSL bus. The peripherals on the ML505 board are also very useful, e.g. the network port. Also, secure processors can be prototyped without fabricating a real processor chip.

C. Secure Processor Architecture

Based on the secure computing model described in Figure 1, we propose our single-chip secure processor architecture. Our approach is to add security modules to the stock OpenSPARC FPGA platform. We synthesize and implement the design in FPGA [15].

Figure 3 shows the block diagram of our single-chip secure processor architecture on the OpenSPARC FPGA platform. The gray blocks in the diagram represent security modules we add to the original platform, including a TRNG (true random number generator) (HDL code), an AES (Advanced Encryption Standard algorithm) engine (HDL code), and a memory integrity tree (MIT) (MicroBlaze firmware code). The TRNG and the AES engine are implemented in the FPGA fabric and the MIT is executed in the MicroBlaze softcore as firmware. The TRNG and the AES engine are connected through the FSL bus to the MicroBlaze. Through MicroBlaze, the OpenSPARC microprocessor communicates with these security modules. The MIT firmware calls the AES engine for memory integrity verification. Each module works in its own clock domain and a Digital Clock Manager (DCM) on the FPGA board serves to generate these clock frequencies. Table I shows the different clock domains in our secure OpenSPARC system.

In addition to the FPGA chip, the FPGA board also has many on-board resources that can be utilized by the secure processor. In our experimental setup, we use the Xilinx Virtex-5 ML505 FPGA board. This board contains a 256MB DRAM module, in which an 80MB ramdisk is used to boot the Linux or Solaris operating system. In addition, the board has an Ethernet port that can connect the secure processor to the Internet if enabled. The Ethernet port can also serve as a communication port that provides high data exchange rate between the host computer and the secure processor.

Our secure processor works in one of the four modes:

- STD - standard mode, which has no additional security measures;
- CONF - confidential mode, which performs memory encryption to ensure data confidentiality;
- ITR - integrity tamper-resistant mode, which performs memory integrity verification to ensure data integrity;
- FTR - full tamper-resistant mode, which performs both memory encryption and memory integrity verification to ensure data confidentiality and integrity.

The secure processor can work in any of the four modes depending on the user's need.

TABLE I: Clock domains in OpenSPARC system

Module	Open-SPARC T1	Micro-Blaze	AES engine	TRNG	DRAM
Frequency	50MHz	125MHz	50MHz	10MHz	400MHz

IV. Single-chip Secure Processor Features

The proposed single-chip secure processor provides the following security features: memory encryption/decryption, secret key generation, and memory integrity verification.

A. AES Engine

Advanced Encryption Standard (AES) is a symmetric encryption algorithm approved by the National Institute of Standards and Technology (NIST). It is one of the most widely used symmetric encryption algorithms and is advanced in terms of both security and performance. While any symmetric key encryption algorithm suits our purpose, we adopt AES as the memory encryption/decryption algorithm and AES CBC MAC as the cryptographic hash primitive.

AES can process data blocks of 128 bits using cryptographic keys with size of 128, 192 or 256 bits. The encryption or decryption takes 10 to 14 rounds of array operations, depending on the key size. In our AES unit design, we employ the idea of parallel table lookup (PTLU) as in [16], [17]. The AES unit is based on AES-128 and takes a block of 128-bit input and 128-bit key to produce 128-bit output. The block diagram of the AES unit is shown in Figure 4.

The AES unit consists of one finite state machine (FSM) that controls the operation of `aes_round` and `aes_key_expander`. The `load` signal triggers the FSM to load the registers with input data and key. The `start` signal sets an internal counter value and starts the AES encryption/decryption cycle from round 0. The `mode` signal changes the AES operation between encryption and decryption. The latency of one block of AES-128 encryption and decryption is 14 cycles and 25 cycles, respectively. For encryption, the AES unit takes 11 cycles to produce the output data (1 cycle per round and 1 cycle for the initial AddRoundKey operation), and 3 cycles to assert `load`, `start`, and `done` signals. However, the decryption process incurs an extra 11 cycles in order to generate the first round key for decryption, since the first round key of decryption is the last round key of encryption. After the round operations are done, a `done` signal is asserted to signify that the output data is ready on the `data_out` bus.

The AES engine works in cipher-block chaining (CBC) mode, i.e., AES-CBC. The input and output data width is 512 bits (64 bytes) in our design in order to correspond to the common size of modern cache lines. The AES engine is connected to the MicroBlaze softcore through the FSL bus, which has a data bus width of 32 bits. All input data has first to be loaded to the AES engine to start the AES-CBC encryption, which needs (32+128+128+512)/32=25 cycles to load the `mode`, `initial_vector`, `key`, and `data_in`, and to completely output the data back to MicroBlaze takes 512/32=16 cycles. If

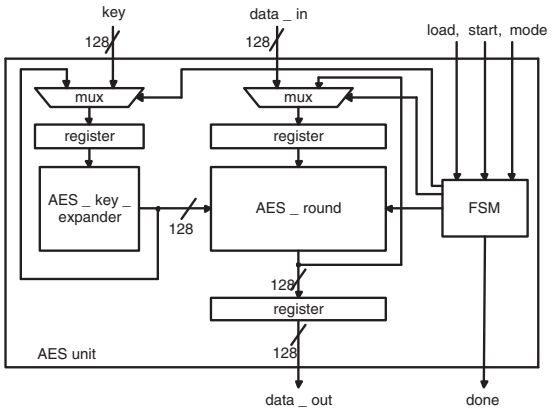

Fig. 4: Block diagram of AES-128 unit.

we include the 14 cycles for encrypting one block of 128-bit input (25 cycles for decryption), the total latency of encrypting 64 bytes of input data is 25+4×14+16=97 cycles. Similarly we get 25+4×25+16=141 cycles for decryption. We note the high overhead of the FSL bus for data transfer (41 of 97 cycles for encryption). Also, performance can be improved by storing the decryption round keys so they do not need to be regenerated, at the cost of using more hardware resources.

B. True Random Number Generator (TRNG)

The secure processor needs a secret key for memory encryption and decryption. In addition, the secret key should be unpredictable for attackers. A true random number generator (TRNG) is utilized to generate a secret key for the AES engine.

Figure 5 shows the internal structure of the TRNG, which consists of many identically laid-out delay loops, or ring oscillators (ROs). We call this a ring oscillator TRNG, introduced by Sunar et al. in [18]. Based on the design in [19], which uses 110 rings with 13 invertors, our TRNG consists of 114 ROs, each of which is a simple circuit containing 15 concatenated NOR gates that oscillate at a certain frequency. One of the two inputs of the NOR gate is used to reset the TRNG. Because of manufacturing variations, each RO oscillates at a slightly different frequency. The outputs of all ROs are exclusive-ORed in order to correct bias and correlation and to generate a random signal. We sample the random signal output from the XOR gate at a frequency of 10MHz. Similar to the AES engine, the TRNG is also connected to MicroBlaze (the emulated cache) through the FSL bus and interacts with the OpenSPARC T1 core through MircoBlaze.

The TRNG module is separate from other modules, which requires overhead of data transfer over the FSL bus if the random bits are used by firmware or another module. For example, a transfer of 128 random bits is needed for the AES engine: TRNG → MicroBlaze → AES. The advantage is that the TRNG is not tied to AES and can be used for other purposes. Only TRNG → MicroBlaze transfer is needed if the random bits are used inside the firmware. Furthermore, the TRNG could be integrated with the AES unit to reduce the transfer overhead if it is dedicated to AES key generation. To the best of our knowledge, combining TRNG with an AES

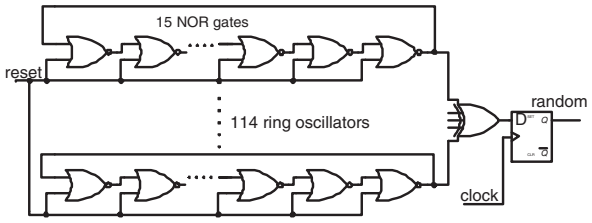

Fig. 5: Internal structure of true random number generator (TRNG).

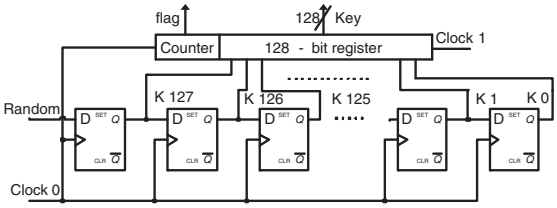

Fig. 6: 128-bit key generation circuit.

engine on a secure processor platform for key generation has not been explicitly discussed in previous works.

In testing TRNG on the FPGA platform, we devise a new method to collect enough random bits for testing. We connect to MicroBlaze an Ethernet core, which works at 10/100/1000Mb/s. The TRNG outputs random bits at 10Mb/s, so these random bits are able to be sent out to the host computer through the Ethernet port on the FPGA board. The MicroBlaze reads random bits from the TRNG and sends them to the Ethernet port which then sends out these random bits to the host computer for testing.

C. Cryptographic Key Generation

The output of the TRNG is single-bit. However, the AES encryption and decryption require a 128-bit key. To generate the 128-bit key using single-bit TRNG, a key generator is employed, shown in Figure 6.

A single-bit-in, 128-bit-out shift register is used in the key generator. The shift register contains 128 concatenated flip flops, each of which stores one bit for the 128-bit symmetric key. The TRNG generates only one single random bit per clock cycle, and its output is connected to the input of the shift register. It takes the shift register 128 clock cycles to store 128 random bits. These 128 random bits are then connected to a 128-bit register, which outputs them as a key. In addition to the registers, there is a counter in the key generator. The counter counts from 0 to 127. When the count reaches 127, it sets a `flag` signal to 1, indicating that the key is ready. After the key is read, the `flag` signal is reset to 0 until 128 new bits are available. This allows new randomness to be returned immediately if some time has passed since last read.

In this way, generating one key needs 128 clock cycles. The generated key can also be used as a seed to generate more 128-bit keys using the AES engine. The 10MHz working frequency of the TRNG might be a little slow. To get faster key generation speed, more than one TRNG can be used. For example, if using two TRNGs, the generated random bits can be stored in two 64-bit shift registers respectively, in which case a key can be generated in only 64 clock cycles and the speed can be doubled. However, this is accomplished at the price of more hardware resource utilization.

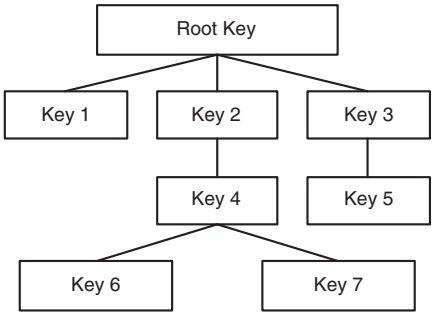

Fig. 7: Hierarchical key chain.

D. Key Management

Cryptographic keys in the secure OpenSPARC system are critical secrets. In the future, as more secure modules and applications are added to the system, the number of keys may increase and key management and protection will become a problem. We address this problem by utilizing the concept of a key chain, which is described in [11].

The key chain is a hierarchical structure which stores all keys of the secure OpenSPARC system in encrypted form, as shown in Figure 7. Each key in the chain is encrypted by its parent key. At the root of the chain is the Root Key. Only a leaf key can be used to encrypt a user's data. This tree structure allows an unlimited number of keys to be stored on the key chain. The Root Key is most critical and it is stored in the Root Key register on the processor chip, which is assumed secure from physical probing. Because all the other keys are in encrypted form, they can be stored in off-chip repositories (external memory) for retrieval and need no special protection. This has greatly enhanced key security and also reduced the on-chip storage for keys.

E. Memory Integrity Verification

Even though off-chip data stored in external memory are encrypted, an adversary can still tamper with them by modifying them. In ITR or FTR mode, the secure processor performs memory integrity verification on all off-chip data fed into the processor to ensure that they are not tampered with. The memory integrity verification is realized using the hash tree, which is shown in Figure 8.

The external memory is divided into different data blocks, and each data block is computed to get one hash value. These hash values are further computed to get their parent hash values (in the nodes above them). Thus, a hash tree (called a Memory Integrity Tree, MIT) is formed and at the top of the tree is the root hash. The root hash value is stored in the root hash register on-chip, which is assumed secure. When data are evicted to off-chip memory, the processor performs

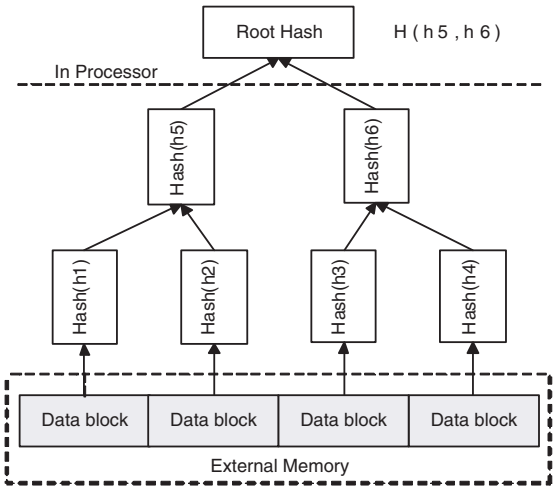

Fig. 8: Memory Integrity Tree

cache line hashing and updating of the non-leaf nodes of the MIT, in the path from the leaf node to the root hash. When external data are read in, the processor performs cache line verification in the path from the leaf node to the root hash. Whenever there is a mismatch between the new root hash and the old root hash, it can be asserted that the contents of the external memory has been tampered with, and an exception will be generated.

The memory integrity tree (MIT) is realized as OpenSPARC firmware in our design, rather than as real FPGA fabric. The MIT firmware calls the AES engine to perform the hash algorithm using AES CBC MAC. Using the firmware to emulate new hardware has some benefits, one of which is that the firmware can be updated without having to re-synthesize any of the existing components.

V. System Evaluation

This section evaluates our single-chip secure OpenSPARC T1 processor, including its performance and hardware costs. The secure processor is prototyped on Xilinx Virtex-5 ML505 FPGA board with a XC5VLX110T FPGA chip. We find that the OpenSPARC FPGA platform is relatively easy for secure processor prototyping. The synthesizing time in Xilinx ISE environment for the whole secure platform is about 4 hours. However, if firmware is altered, there is no need to re-synthesize the whole system and recompiling the firmware takes only several minutes, which saves a lot of time.

A. Encryption/Decryption Performance

In CONF and FTR mode, the secure processor has to perform encryption and decryption operations. As mentioned in Table I, the secure OpenSPARC system has multiple clock domains. It takes the AES engine 97 clock cycles to encrypt 64 bytes of input plaintext, and 141 clock cycles to decrypt 64 bytes of input cyphertext, where many of these cycles are due to data transfer via the 32-bit FSL bus. The TRNG works at the frequency of 10MHz, and to generate one 128-bit key needs 128 TRNG cycles. However, a new key is infrequently needed. Data from AES and TRNG have to be first processed by MicroBlaze, and they are fetched by MicroBlaze at the frequency of 125MHz. The clock cycles needed for all these operations are shown in Table II.

TABLE II: Clock cycles needed for various operations.

Operation	Cycles	Cycle frequency
AES encryption of 64-byte block	97	50MHz (AES cycles)
AES decryption of 64-byte block	141	50MHz (AES cycles)
128-bit key generation from TRNG	128	10MHz (TRNG cycles)
128-bit key transfer TRNG $\rightarrow$ MicroBlaze	4	125MHz (FSL cycles)
128-bit key transfer MicroBlaze $\rightarrow$ AES	4	125MHz (FSL cycles)

B. Overall Performance

The STD mode causes no extra performance overhead for the processor. In CONF and FTR mode, the processor has to encrypt/decrypt off-chip data, which incurs performance overhead, but this only happens for cache misses which are relatively infrequent. For each 64-byte data evicted off the secure processor chip, a 97-cycle delay will be caused due to the encryption operation. Similarly, for each 64-byte off-chip data fed into the processor, a 141-cycle delay will be caused due to the decryption operation. This overhead can be reduced if the FSL bus could be widened or multiple FSL buses can be used. Counter-mode AES can also be used to reduce effective encryption/decryption latency. In ITR and FTR mode, the processor performs cache line hashing on data evicted to off-chip memory, and cache line verification on data read into the processor, which also causes additional performance overhead.

In the FPGA version of OpenSPARC, the frequency of the OpenSPARC T1 core is 50MHz, which is a bit slow for performance research running large software applications. Regardless, the OpenSPARC platform is still suitable for secure processor research because of its many advantages. In this paper, we mainly focus on how to modify the platform to add security features rather than on performance.

C. Hardware Costs

Our single-chip secure processor is implemented on Xilinx Virtex-5 XC5VLX110T FPGA. Table III shows the total resources of the FPGA chip and hardware costs after the security modules are added. Table III shows that the OpenSPARC T1 core has taken up most of the slices of the FPGA chip, up to 78%, while the new security modules consume much fewer slices. After both AES and TRNG are added, the slice utilization has increased by only 10%, which is far less than the 78% consumed by the T1 core. In our design, the OpenSPARC T1 microprocessor has been tailored to include only one CPU core. The resource utilization ratio of AES and TRNG will be even lower if two or more cores are used.

TABLE III: Logic utilization of single-chip secure OpenSPARC processor on Xilinx Virtex-5 FPGA

Module	Slice	LUT	Register	BRAM
Virtex-5 FPGA XC5VLX110T	17280	69120	69120	148
OpenSPARC T1	13561 (78%)	40270 (58%)	30087 (43%)	119 (80%)
OpenSPARC T1 with AES added	14030 (81%)	43174 (62%)	30945 (44%)	143 (96%)
OpenSPARC T1 with TRNG added	15166 (87%)	42162 (60%)	30146 (43%)	119 (80%)
OpenSPARC T1 with AES and TRNG added	15181 (88%)	45030 (65%)	31004 (44%)	143 (96%)

Table III also shows that the secure OpenSPARC T1 processor has almost taken up all the resources of the Virtex-5 XC5VLX110T FPGA chip. This will restrain the system from further development. For example, if more security modules are to be added, or two OpenSPARC T1 cores are desired in the system, the remaining resources may not be enough. One solution to this problem is to move the system to a larger FPGA chip with more logic resources, for example, the Xilinx Virtex-6 or Altera Stratix V FPGAs.

VI. Conclusions

OpenSPARC T1 is an open-source, FPGA-synthesizeable general-purpose microprocessor. In this paper, we have described a secure computing model which assumes that only the on-chip environment is secure from physical attacks, and proposed a single-chip secure processor architecture. Further, we have prototyped the secure OpenSPARC T1 processor on FPGA and evaluated the resulting system. The new security modules added to the OpenSPARC system incur little extra hardware costs and performance overhead.

By prototyping the secure OpenSPARC T1 processor, we find that the OpenSPARC FPGA platform has many advantages for secure processor research and prototyping: ability to modify real hardware, ease of modification due to the emulated cache, ability to run commodity OS and benchmarks, the availability of an open source hypervisor, etc. The low working MHz rate of the stock OpenSPARC T1 processor and the high overhead of data transfer cycles using the 32-bit FSL bus affect the performance of the prototype. Hence, performance monitoring, performance estimation and performance improvement are fruitful areas for further research.

In this work, we have added the AES engine, TRNG and MIT to the OpenSPARC platform. More security modules can be added to further enhance its security features. On the other hand, high-level applications can be developed to make use of these security modules. In summary, we find that the OpenSPARC FPGA platform is relatively easy for secure processor prototyping with many advantages including its advanced software and hardware platform components.

Acknowledgment

This work is supported in part by NSF CCF-0917134 and NSF EEC-0540832, and by the City University of Hong Kong Start-up Grant 7200179.

References

[1] W. Ford and B. S. Kaliski, Jr., "Server-assisted generation of a strong secret from a password," in *Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE Computer Society, 2000, pp. 176–180.

[2] J. Garay, R. Gennaro, C. Jutla, and T. Rabin, "Secure distributed storage and retrieval," in *Distributed Algorithms*, M. Mavronicolas and P. Tsigas, Eds. Springer Berlin / Heidelberg, 1997, vol. 1320, pp. 275–289.

[3] P. MacKenzie and M. K. Reiter, "Networked cryptographic devices resilient to capture," *International Journal of Information Security*, vol. 2, pp. 1–20, 2003.

[4] OpenSPARC, World's First Free 64-bit CMT Microprocessors. [Online]. Available: http://www.opensparc.net/

[5] I. Parulkar, A. Wood, J. C. Hoe, B. Falsafi, S. V. Adve, and J. Torrellas, "OpenSPARC: An open platform for hardware reliability experimentation," *the Fourth Workshop on Silicon Errors in Logic-System Effects (SELSE)*, April 2008.

[6] K. Chandrasekar, R. Ananthachari, S. Seshadri, and R. Parthasarathi, "Fault tolerance in OpenSPARC multicore architecture using core virtualization," *International Conference on High Performance Computing*, December 2010.

[7] D. Lee, "OpenSPARC - a scalable chip multi-threading design," in *21st Intl. Conference on VLSI Design, VLSID*, 2008, p. 16.

[8] P. Pelgrims, T. Tierens, and D. Driessens, "Overview of embedded processors: Excalibur, LEON, MicroBlaze, NIOS, OpenRISC, Virtex II Pro," *De Nayer Instituut, Tech. Rep.*, 2003.

[9] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas, "Aegis: architecture for tamper-evident and tamper-resistant processing," in *Proceedings of the 17th annual international conference on Supercomputing*. ACM, 2003, pp. 160–171.

[10] G. E. Suh, C. W. O'Donnell, and S. Devadas, "Aegis: A single-chip secure processor," *IEEE Design and Test of Computers*, vol. 24, pp. 570–580, 2007.

[11] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, and Z. Wang, "Architecture for protecting critical secrets in microprocessors," in *Proceedings of the 32nd annual international symposium on Computer Architecture*. IEEE Computer Society, 2005, pp. 2–13.

[12] D. Champagne and R. Lee, "Scalable architectural support for trusted software," in *16th IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2010, pp. 1–12.

[13] D. Weaver, *OpenSPARC Internals*. Sun Microsystems, Inc., 2008.

[14] J. Szefer, Y.-Y. Chen, R. Cheung, and R. B. Lee., "Evaluation of OpenSPARC FPGA platform as a security and performance research platform," *Princeton University Department of Electrical Engineering Technical Report CE-L2010-002*, September 2010.

[15] PALMS OpenSPARC Security and Performance Research Platform. [Online]. Available: http://palms.ee.princeton.edu/opensparc

[16] R. Lee and Y.-Y. Chen, "Processor accelerator for AES," in *Proceedings of IEEE Symposium on Application Specific Processors (SASP)*, June 2010, pp. 16–21.

[17] J. Szefer, Y.-Y. Chen, and R. B. Lee, "General-purpose FPGA platform for efficient encryption and hashing," in *Application-specific Systems, Architectures and Processors conference*, July 2010, pp. 309–312.

[18] B. Sunar, W. Martin, and D. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 109 –119, 2007.

[19] D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," in *International Conference on Field Programmable Logic and Applications, FPL*, 2006, pp. 1–6.