



International Journal of Engineering, Business and Enterprise Applications (IJEBA)

www.iasir.net

APPLICATION SPECIFIC INTEGRATED CIRCUIT CHIP BASED DATA RECOVERY SYSTEM

Ambika R¹, C S Mala², Vishakha Yadav³

^{1,2}Associate Professor, Dept. of ECE,

³ Assistant Professor, Dept. of CSE BMS Institute of Technology,
Bangalore-560064, INDIA

Abstract: In a network system, there needs to be a provision for recovery of intentional/ unintentional modification of data. Conventionally this is done by creating a backup/copy of data prior to its modification. In this paper, using an Application Specific Integrated Circuit chip based Data Recovery System, the original contents (of a data file) can be recovered, in case the modification done to the data is not authenticated. If the user is authenticated, the modifications shall be updated. In this work, the data recovery process is performed by contextually switching the semaphores which control the system resources. The designed hardware is placed in the PCI slot between the CPU and lower-level peripherals. A binary semaphore controls the data flow between the designed hardware and the CPU, whereas a mutually exclusive (context-switching) semaphore controls the data flow between the designed hardware and lower-level peripherals. The novelty of the proposed work is that, the atomicity of the operation is retained.

Keywords: ASIC, Context switching, Data Recovery, data security, DIFT, Semaphore, snooping

I. Introduction

Security is an important aspect of any operating system. It allows access of information to only properly authorized individuals or processes through the use of special security features. There are two types of security threats to an operating system namely active and passive. Active threats include amendment, fabrication, denial of service and jamming. Passive threats include tapping and disclosure.

To overcome the problems faced by the conventional data protection methods, flexible and secure information flow architecture for software security is required[8].

The proposed work provides software security based on Dynamic Information Flow Tracking (DIFT) by keeping track of the modified capabilities of the resources [4,5]. A novel feature of this work is that the proposed DIFT allows passive threats to co-exist thus making the architecture more user-friendly. The information is allowed to flow freely into fetch, decode and execute cycles and the resource protection i.e., the authentication is provided only before the write cycle. This helps in preserving the original information and also maintains the data integrity. The data recovery process is performed by contextually switching the semaphores which control the system resources. A semaphore is a protected variable or an abstract data type which restricts the access to shared resources such as shared memory in a multiprogramming environment. The proposed work regulates enabling and disabling of the cache structure with respect to the CPU. It allows mapping of access rights and does not directly change the capabilities unless authenticated. The capabilities are kept secured at hardware level and no permanent modification and manipulation of the original capabilities can be done without permission. The paper is organized as follows: Section 2 describes the objective of this work. Section 3 of this paper describes the block diagram of the proposed DIFT.

II. Objective of the work

1. To protect the system against active threats with increased user-friendliness by allowing the passive threats to co-exist.
2. To provide a unified security across all the seven layers of OSI model.
3. To make the security policies more flexible and software programmable so that the protection schemes are generic.
4. To retain the two main requirements of security namely the "Data Availability" and "Data Integrity".
5. To reduce the amount of extra OS resources required by the user and the entire hardware is made more compact and compatible.

The above objectives are achieved in this work, by allowing the information to flow freely through the fetch and decode cycles while an access or authentication is made between the decode and execute cycle before the data is permanently written into the memory by the user (if authenticated) and is shown in figure 1.

Figure 1. Information flow in the proposed DIFT

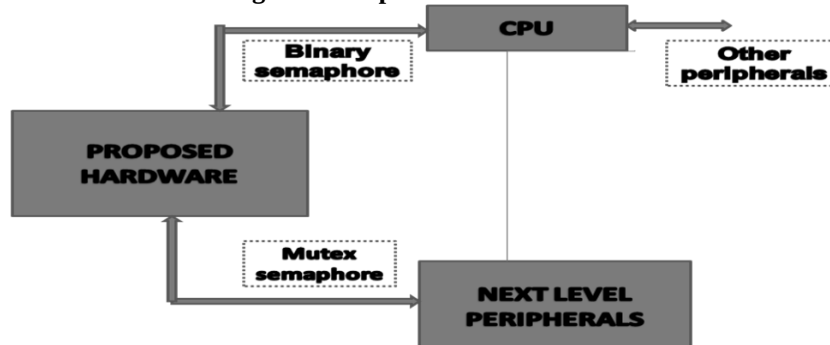


A. Novel features of proposed DIFT

- **ROBUSTNESS:** To provide defense against vulnerabilities with few false positives or false negatives.
- **FLEXIBLE:** It can adapt easily to cover the continuously evolving threats.
- **END-TO-END:** The security policy flows throughout all the seven layers of the OSI model.
- **SCALABLE:** It can co-exist with the existing circuitry without any modification.

The proposed hardware flow is shown in Figure 2. The features of the proposed hardware are that it is (i) PCI compliant and (ii) Mounted in a single chip.

Figure 2. Proposed hardware flow.



By using a mutex semaphore, the atomicity of operations is maintained. The other peripherals cannot access the proposed hardware easily, since it is based on ownership concept. Computer machine languages include instructions designed specifically with the idea of semaphores in mind. These special instructions carry out a read-modify-write cycle to memory that is not only uninterruptible but also excludes all other operations to the same memory location by any other processor or input/output devices. Thus these special instructions guarantee that a semaphore procedure using them can test and alter a semaphore in a single atomic operation. Thus, these semaphores are very useful in a concurrency situation. They are faster than regular file locks and can be used on other things that are not files, such as Shared Memory Segments.

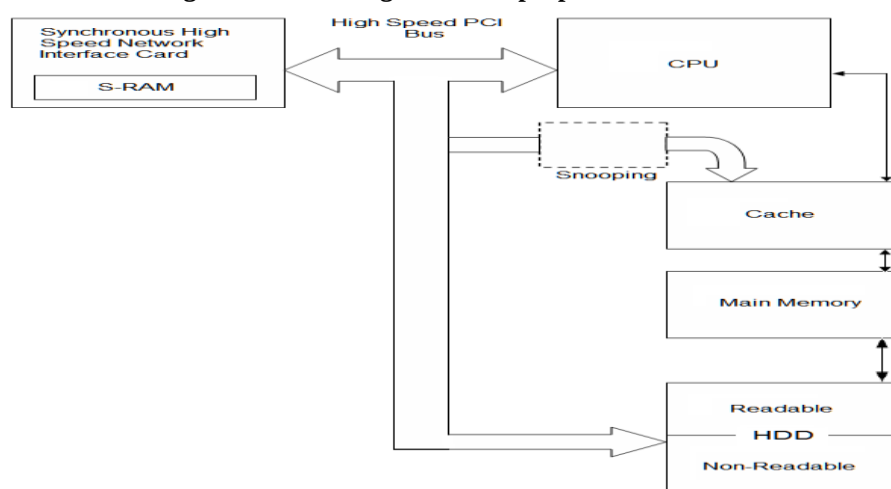
III. Block Diagram

The block diagram of the proposed work is shown in Figure 3.

A. Snooping

Snooping is done between CPU and cache. Its function is to disconnect the link between CPU and cache whenever data transfers occur between the CPU and developed ASIC Card. Also, snooping is OFF when data transfer occurs between cache and the proposed DIFT. Snooping will be ON for cache essential process like boot-up. Snooping is controlled through the PCI bus and is unidirectional in nature.

Figure 3. Block Diagram of the proposed Architecture.

**IV. Implementation Methodology**

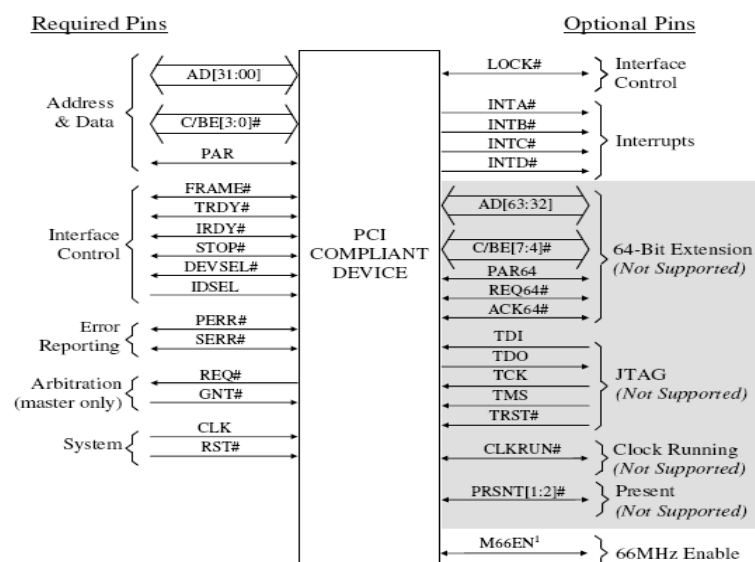
In the existing architecture, the bottleneck of connecting a high speed low memory device to a low speed high memory device is solved by using an intermediate memory module called the cache. The cache exists between the high speed CPU and the lower speed memories. However, there is no security between the data link layer of the CPU and the cache. The proposed DIFT is placed on the PCI bus at the maximum possible speed and a direct connection is established to the CPU via snooping. To understand how the security layer is to be implemented, a brief review of the three basic terms is described: subject, object and capability. Subject refers to

the user or entity which acts on behalf of the user on the system. Objects may be defined as resources within the system. The main term however is capability which is basically a ‘token’. The possession of a capability by a subject confers access right for an object. They cannot be easily modified, but they can be reproduced. For a subject to access a particular object, it must possess the capability for doing so. Hence, before a subject accesses a resource via the CPU, it will first go through a screening check from the hardware on whether or not its capabilities allow it to access such resources. Hence, a security layer is now added to the data link layer between the CPU and the cache. Additionally, user also must be prevented from creating arbitrary capabilities. This can be accomplished by placing the capabilities in special ‘Capability Segments’ which users cannot access. Another approach is to add a tag bit to each primary storage location. This bit, inaccessible to the user is ‘ON’ if the location contains a capability. It should be noted that the hardware restricts the manipulation of the location contents to appropriate system routines. If the last remaining capability is destroyed, then that object cannot be used in any manner. In this work, special provisions are made for controlling the copying and movement of capabilities (as well as interpretation) depending on the hardware involved.

V. PCI device pin Configurations

The pin diagram of the PCI device is shown in figure 4.

Figure 4. Pin Diagram of PCI Device



For the purpose of brevity only the essential pin description is done in the following sections:

A. Cache Support Pins:

These pins permit cacheable memory to be implemented on the PCI bus. They transfer status information between bridge/cache and the target of the memory request. If a PCI transaction results in a hit on a “dirty” cache line, the bridge/cache will signal “snoop back-off” to the cacheable target. As a result, the target will issue the retries on all access to the modified cache line until the bridge/cache completes a write back operation. The target will then permit the access to complete. For performance reasons, cacheable memory is typically coupled very closely with the host processor that runs at a higher frequency than the PCI.

B. SBO#

Snoop back-off indicates a hit to a modified line when asserted. When SBO# is de-asserted and SDONE is asserted, it indicates a “clean” snoop result.

C. SDONE

Snoop done indicates the status of the snoop for the current access. When de-asserted, it indicates the result of the snoop is still pending. When asserted, it indicates the snoop is complete.

VI. DIFT Design

A. The Design Process

The HDL Design Capture phase implements the “top-down” methodology of design. It takes a designer from abstract concept or algorithm down to hardware in manageable and verifiable steps. This involves developing a design specification that will be used to create a high level behavioral abstraction with high-level programming languages such as C/C++. Additionally, this model may also be created using the Verilog HDL. The behavioral model should be simulated in order to verify that the desired functionality is captured completely and correctly. The behavioral abstraction is then used as a reference to create and refine a synthesizable register transfer level (RTL) abstraction that captures the desired functionality required by the design specification. With Verilog, the

difference between a purely ‘behavioral’ and an RTL abstraction may appear subtle at this point. Generally, we represent designs in Verilog at three levels of abstraction:

B. Methodology for Implementation

For the implementation of the hardware module in the proposed PCI compatible device it should function as a context switching semaphore device. For this feature, a snooping process has to be implemented and the two pins of the PCI bus- namely the SDONE and SBO# have to be utilized. Consequently for the above features to be implemented, the following is done:

Step 1: The tag representing the capability is stored in the readable portion of the HDD and has to be read from there. This tag has to be presented to the CPU to check for the access rights of the object so as to allow the subject to use the object within the allowed access rights. For example, if a file writing capability is prevented then that particular option will be disabled and whenever a subject will try to write into that object, the CPU will display a – ‘File Read Only’ message. Once the tag is found, its address location will be mapped into the DIFT.

Step 2: After step 1, Snooping is implemented and the CPU-cache architecture is disabled. Once this is done the capability will be read into the CPU (Read operation). The CPU will then process the capability and check for the access rights of the object.

Step 3: The capability of the presented object is modified as the subject uses the chosen object in different ways and this is now written back into the DIFT (Write operation). It should be noted that snooping is also implemented in this step.

Step 4: In the final step the snooping process will be turned off. Then the modified capability is written into the non-readable part of the hard disk by the DIFT. It should be noted that access to write in the non-readable portion of the HDD can be done only by installing the device driver for the hardware.

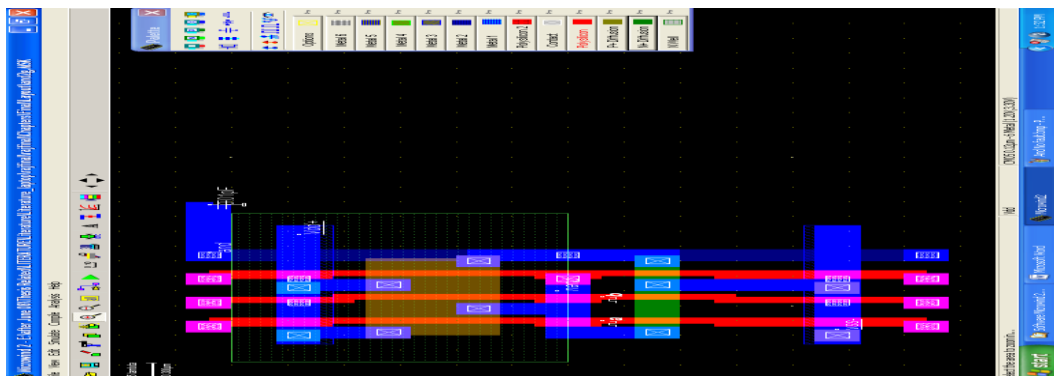
VII. Layout

The layout of the primitive on-chip DIFT (designed using the Microwind package) is presented in this section.

A. Evolved function is semaphore selector

The schematic of the control block performing the evolved function of semaphore selector is shown in figure 5.

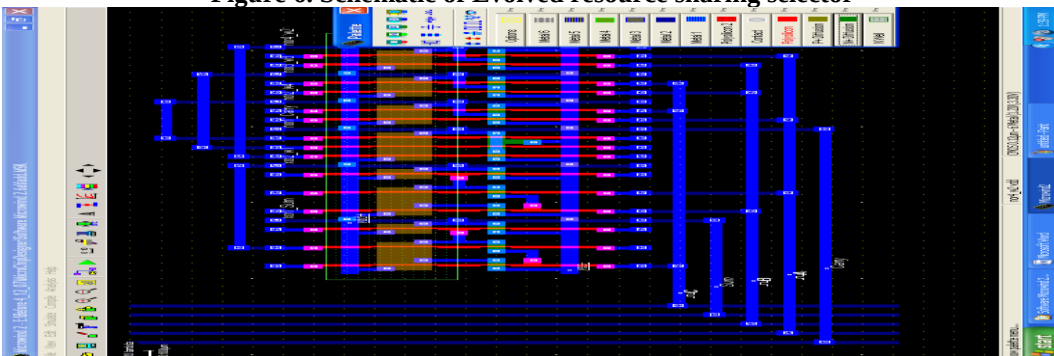
Figure 5. Schematic of semaphore selector



B. Evolved function is resource sharing selector

The control block performing the evolved function of “resource sharing selector” along with its power consumption is shown in figure 6.

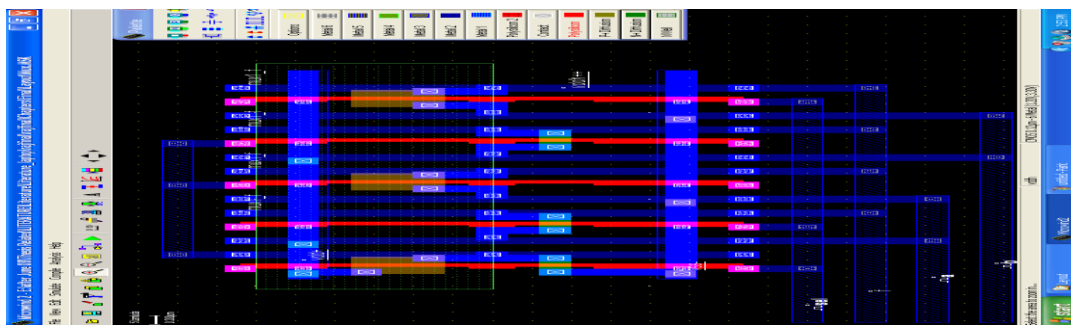
Figure 6. Schematic of Evolved resource sharing selector



7.3 Evolved function is Snoop selector operation

The schematic of the control block performing the evolved function of a Fundamental snoop selector circuit is shown in figure 7.

Figure 7. Schematic of Evolved Snoop selector circuit



7.4 Evolved function is Context switching semaphore

The schematic of Context switching semaphore block of the evolved function is shown in figure 8.

Figure 8. Schematic of Evolved Context switching semaphore

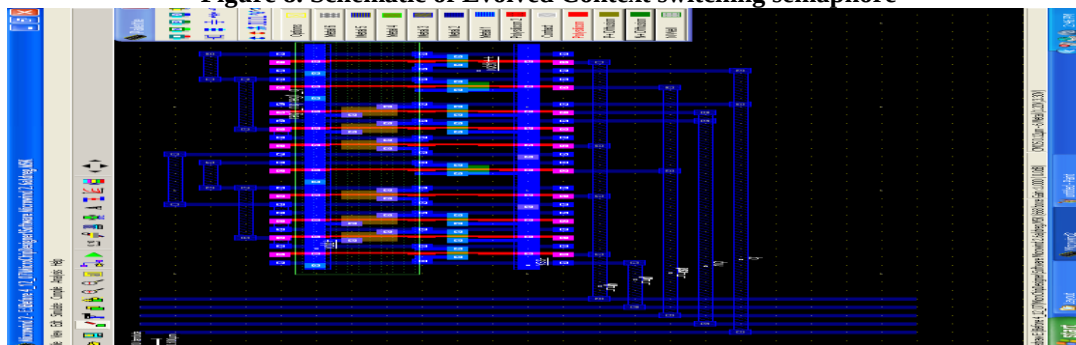


Table 1. Power Consumed by Evolved PE

Evolved Function of PE	Average Power Consumed by Evolved blocks	
	read	write
Semaphore selector	.3 mW	2.325mW
Resource sharing selector	.54mW	3.5mW
	.12mW	3.665mW
Context switching semaphore	.456mW	3.5mW

VIII. Conclusion

This work provides ideal data back-up architecture. The data saved in one session can be used till refreshing occurs. Security is provided independently, since the user does not actually interact with the HDD. In real time there is interaction only through a virtual HDD.

It provides OS flexibility and it is possible to work with the user friendliness of any operating system. It provides an easy and uncomplicated method of information recovery from virus (like Trojan horse) infections. As it is OS independent, it can be used in any compatible system and with any software. As a future scope, provision can be made selective, resulting in the option of both archive and discard of the modified files residing in different protected drives.

References

- [1]. Didier Samfat, Fefik Molva. IDAMN-An intrusion detection architecture for mobile network – IEEE journal on selected areas in communications, Vol 15, Sep 1997.
- [2]. Paul Amman, Sushil Jajodia and Peng liu. Recovery from malicious Transactions – IEEE transactions on knowledge and data engineering, Vol 14, Sep/Oct 2002.
- [3]. Dr. Bhavani Thuraisingham. Data and application security in the proceedings of 26th Annual international computer software and application conference, COMPSAC'02.
- [4]. G. E. Suh, J. W. Lee, D. Zhang and S. Devadas. Secure program execution via dynamic information flow tracking. Proceedings of the 11th International conference on Architectural support for programming languages and operating systems Oct' 2004.
- [5]. Lap Chung Lam, Tzi-cker Chiueh Reither networks, Inc. A General dynamic information flow tracking framework for security applications. Proceedings of the 22nd ACSAC'06, 2006 IEEE.
- [6]. Peng Liu and Jiwu Jing, Architectures for Self-Healing Databases under Cyber Attacks. Proceedings of IJCSNS. Vol,6 No.1B, January 2006.
- [7]. Feng Qin, Cheng Wang, Zhenmin Li, Ho-seop Kim. Yuanyuan Zhou and Youfeng Wu. LIFT : A low-overhead practical information flow tracking system for detecting security Attacks in the proceedings of the 39th Annual IEEE/ ACM international symposium on Micro architecture, Dec'2006.
- [8]. Michael Dalton, hari Kannan, Christos Kozyrakis, RAKSHA : A flexible information flow architecture for software security. Proceedings of ISCA' 07. June 2007.