Efficient Hardware Accelerator for IPSec based on  
Partial Reconfiguration on Xilinx FPGAs

Ahmad Salman, Marcin Rogawski and Jens-Peter Kaps  
Volgenau School of Engineering  
George Mason University  
Fairfax, Virginia 22030  
email: *{*asalman, mrogawsk, [jkaps*}*@gmu.edu](mailto:jkaps%7d@gmu.edu)

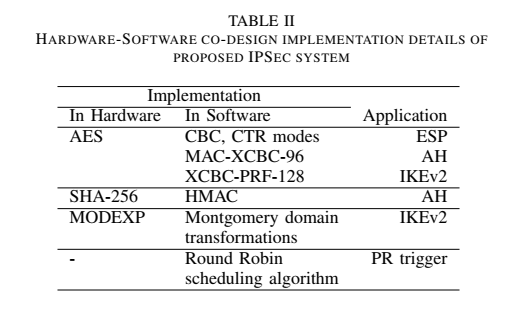
Trong bài báo này, chúng tôi trình bày một giải pháp hệ thống nhúng thực tế cấp thấp cho Internet Protocol Security (IPSec) được triển khai trên thiết bị Xilinx Field Programmable Gate Array (FPGA) trên họ Virtex 4 family .

The proposed solution supports the three main IPSec protocols: Encapsulating Security Payload (ESP), Authentication Header (AH) and Internet Key Exchange (IKE). This system uses efficiently hardware-software co-design and partial reconfiguration techniques.=> tiết kiệm đáng kể tài nguyên phần cứng

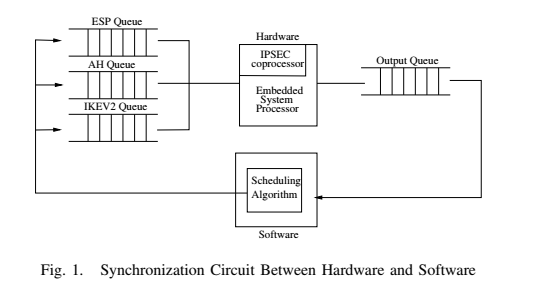
In this work we propose a division of the basic mechanisms of IPSec protocols, namely cryptographic algorithms and their modes of operation to be implemented either in software or hardware. Through this, we were able to combine the high performance offered by a hardware solution with the flexibility of a software implementation. We show that a typical IPSec protocol configuration can be combined with Partial Reconfiguration techniques in order to efficiently utilize hardware resources.

System decription

Our proposed IPSec embedded system’s structure is summarized in Table II. We implemented three cryptographic transformations in hardware where at any given point in time, only one is available for its utilization. In order to use a different algorithm than the one currently available, partial reconfiguration operation is performed. This process is controlled by a modified Round Robin with time sharing scheduling algorithm [18] implemented on the embedded processor.



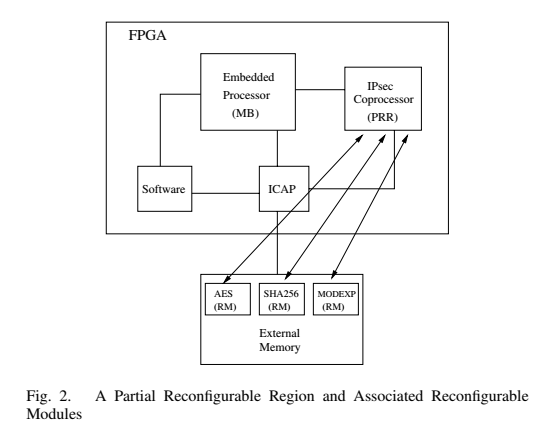
Packets are sent from and received by the system through input and output queues.



The Round Robin scheduling algorithm is used for switching the control between queues when packets are ready to be processed. In this system, we use a configurable time slot value which specifies for how much time packets from one queue are being processed before switching to another queue. Even if the current queue still contains data the processor switches to a different queue to make sure that no specific type of packets monopolize the co-processor.

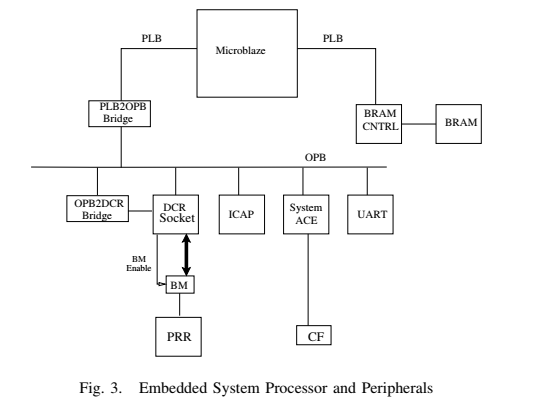
***B. Partial Reconfiguration***

A typical PR system is composed of static regions known as Base Region (BR) and a dynamic region known as Partial Reconfigurable Region (PRR). The BR holds the portion of the design that does not get affected by partial reconfiguration while the PRR holds the portion of the design that gets swapped during partial reconfiguration process which is known as Reconfigurable Modules (RM).



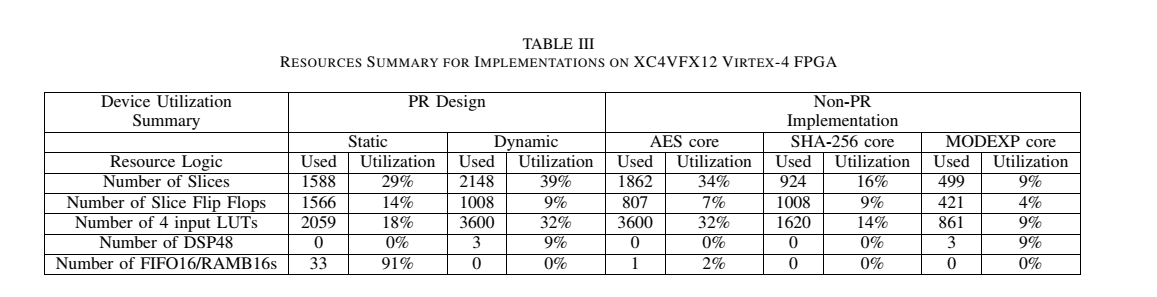
**In our design, the BR includes an embedded processor that controls the PR process and some supporting peripherals while** the PRR includes hardware accelerators for IPSec protocols. Initially the system is configured with the BR and the PRR is loaded with one of the RMs or left blank with no RM loaded.  
The remaining RMs are stored on an external memory and are swapped with other RMs by the scheduler.

During PR, the embedded processor communicates with the Internal Configuration Access Port (ICAP) which loads the partial bitstream that holds the information of the requested RM from the external memory and replaces the currently running RM or the blank space in the PRR with it during run-time as shown in Fig. 2.

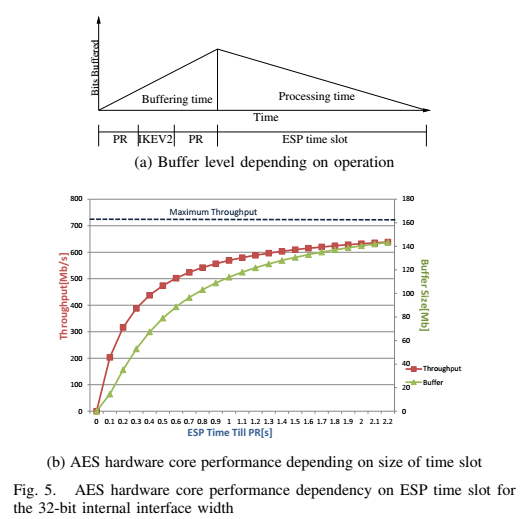


**Kết quả:**

Thực hiện trên kit Xilinx Design Suite 9.1 vì nó hỗ trợ triển khai PR

The implementation results are summarized in Table III. The first two columns summarize the resources of the static and dynamic portions of the system. The other three columns are implementation results for each of the three cores (AES, SHA- 256 and MODEXP) implemented independently in non-PR designs. When comparing the dynamic portion of the design to the non-PR implementations, it can be noticed that the PR design uses 2148 slices compared to 3285 slices used by the three non-PR designs combined together. This is an area improvement of more than 34%.

* Latency vs Throughput



It can be shown that in order to reduce the effect of PR on the total time (time needed from packet received by the system to sent back after being processed), the amount of time assigned for computations (encryption/hashing) should be increased. For example, if the PRR is loaded with the AES core module, then the scheduling algorithm should direct the Microblaze to fetch tasks from the ESP queue as long as it is not empty or the other queues are not full. The less PR is triggered, the higher the throughput.