

Intel[®] Technology Journal

Network Processors

Security: Adding Protection to the Network via the Network Processor

Security: Adding Protection to the Network via the Network Processor

Wajdi Feghali, Brad Burres, Gilbert Wolrich, Douglas Carrigan
Intel Communications Group, Intel Corporation

Index words: security, encryption, authentication, 3DES, AES, SHA-1

ABSTRACT

As information security becomes more important in today's world, it is necessary for networking equipment to enable cryptographic functions. The Intel IXP28xx network processor integrates the appropriate set of features to enable the addition of security functionality with significant advantages over a discrete solution.

The integration of security functionality in the IXP28xx network processor was initiated by the investigation of cryptographic systems requirements. Several possible architectures were investigated, but they all lacked certain key elements. It was proposed that an integrated solution capable of network processing and security processing offered the most advantages.

After determining that an integrated solution had the most advantages, we found it necessary to understand what new functions needed to be integrated and what existing IXP2800 functions could be leveraged. The new functions were identified, implemented, and interfaced to the rest of the network processor. These new functions, combined with the baseline IXP28xx, make it possible to provide secure traffic at 10Gigabits/second.

INTRODUCTION

With the increasing reliance of businesses and individuals on computer networks, there is a corresponding increase in the importance of information security. This increase requires that some base security functionality, such as data confidentiality and data integrity, be afforded to every packet placed on the network. Despite this requirement, security is sometimes an after-thought in many networking equipment designs. It is essential to provide security functionality as part of the networking building blocks from the beginning of the design cycle.

Previous designs have added security to the network through either a co-processor or an inline security processor. As data rates go up, the co-processor solution

becomes less and less practical. Inline security processors can actually scale to the higher data rates but must perform many of the same functions as the network processor does to achieve the high data rate.

Integrating security functions onto the IXP28xx makes it possible to provide secure network traffic at 10Gigabits/second while using the same system designs as for an ordinary network processor. This enables the design of security functionality in network equipment from the start and at a lower overall system cost in terms of power consumption, board real estate, and silicon investment.

This paper describes how security functionality is added to the IXP2800 network processor in order to support data confidentiality and data integrity. Specifically, it discusses the hardware features added and how these features can work in cooperation with the rest of the network processor functionality.

SECURITY SYSTEMS ARCHITECTURES

There are three primary ways to add security functions to networking hardware equipment.

The first and most common method today is to use a co-processor coupled with a network processor or a general processor. As data rates go up, this method becomes less practical because the packet must traverse shared resources such as data buses or memory four times.

The second method is to add a security processor inline with a network processor. While this approach can achieve high data rates, the inline security processor must perform many of the same functions that the network processor must do such as packet reassembly; thus work must be repeated and silicon area must be duplicated.

The third method is to add the security functionality into the same silicon as the network processor, thus adding security functionality into the network processor while maintaining wire rate and minimizing new silicon area. As

new network line cards are designed, an integrated solution will prove beneficial.

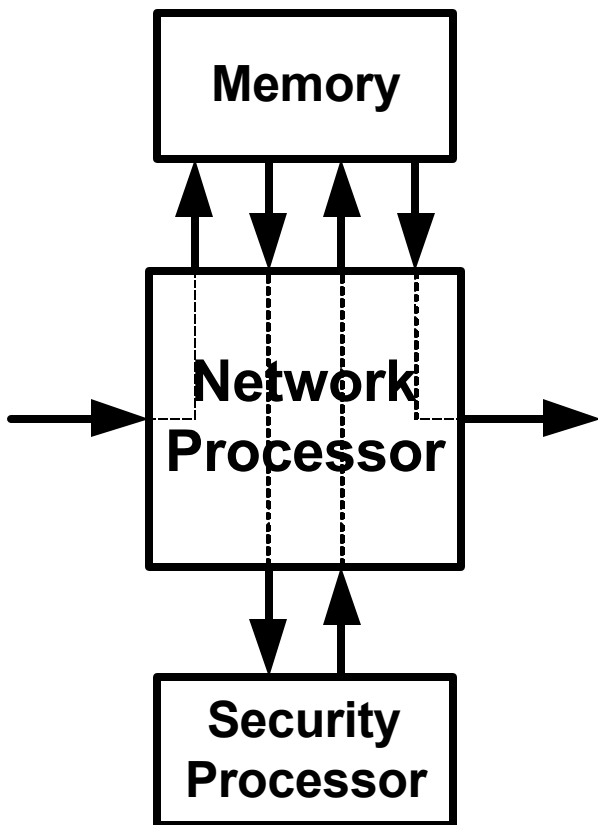


Figure 1: Look-aside architecture

assemble the packet in memory before sending it to the security processor and that the packet is then placed in memory before being transmitted. Although the bus between the network processor and the security processor must be able to handle twice the line data rate, requiring many more pins on the network processor, the network processor memory must be traversed four times, and that is not desirable for high packet rates. It is important that the bulk of the data traverse the memory just twice: once for write and once for read.

Figure 1 shows a network processor and a security processor in the look-aside configuration. The packet data must typically traverse the memory four times, as opposed to two times, and the bus between the network processor and security processor must be capable of doing at least twice the desired wire rate.

Flow-through Architecture

The flow-through architecture solves the performance problems of the look-aside architecture, but it requires the security processor to do many of the functions that the network processor is targeted for. Some of these tasks include reassembly of packets, protocol processing, and exception handling.

It is very desirable to be able to use the same underlying hardware architecture to target multiple applications. This would be possible if the security functionality were added to the network processor.

Figure 2 shows partitioning of flow-through architectures. Some applications require that the network processor be placed before the security processor, some require that the security processor be placed before the network

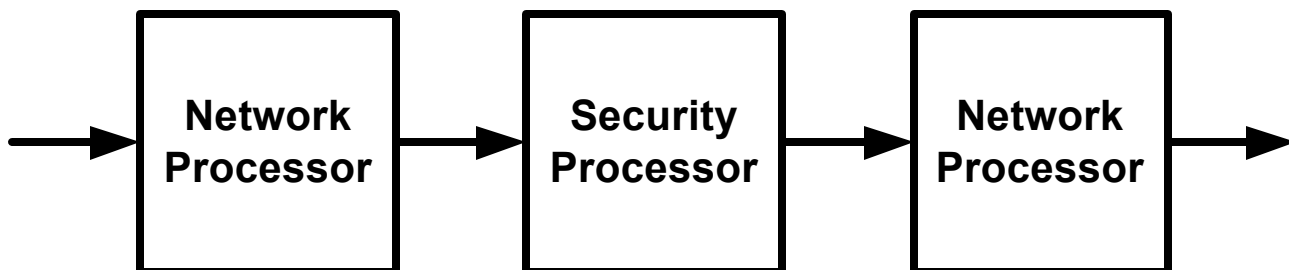


Figure 2: Flow-through architecture

Look-aside Architecture

The look-aside architecture is attractive because it places fewer burdens on the security processor to completely implement protocol processing. For a simpler security processor design, a greater burden is placed on the network processor. Current security processors require that the entire packet be available before processing can begin. This implies that the network processor must

processor, and some require a network processor before and after the security processor. For example, an SSL proxy application requires termination of a TCP connection, SSL processing, and then the establishment of a new TCP connection.

Protocol and Cryptographic Processing

The securing of network traffic can essentially be portioned into two partitions: protocol processing and cryptographic algorithm processing.

Protocol processing would include Encapsulating Security Protocol (ESP), Authentication Header (AH), Secure Sockets Layer (SSL), Transport Layer Security (TLS), and other non-security protocols such as Transmission Control Protocol (TCP) and the Internet Protocol (IP) processing.

Cryptographic algorithm processing includes the data manipulation that would need to be done on the entirety of the payloads, such as confidentiality and integrity.

The flow-through architecture requires the security processor to do similar functions as the network processor, mainly protocol processing. This can be done with dedicated hardware or perhaps one or many protocol processors integrated in the security processor.

An interesting question is raised: Is it better to add the necessary functions to a network processor to enable it to target security processing, or add network processor functions to the security processor to enable it to completely handle protocol processing?

We decided to leverage the extensive functionality and flexibility of the IXP2800 network processor and add to it the necessary cryptographic algorithms.

LEVERAGING THE IXP2800

The IXP2800 consists of several units that are connected via the chassis. The chassis connects the following units that can be directly used for security processing: microengines, SRAM, DRAM, lookup hash, PCI, and XScale™, via a set of command buses and data buses. The cryptography units are added to the chassis and are accessible by the microengines and the media switch fabric interface for data, and by the microengines for commands. For details on the chassis, see reference [7].

Microengines

Microengines are used to do the protocol processing, such as ESP processing for IPsec traffic. That processing includes constructing new headers, copying fields from one header to another, and modifying security state information. The microengines are designed for packet processing. For example, during replay checks on incoming ESP packets, it is necessary to read the sequence number state, modify the replay window, and write the data back to memory.

The microengine CAM is used to effectively manage the reading, make multiple modifications across several microengine threads, and write back the sequence number state to memory. This technique is called *folding* [6].

DRAM

The IXP2800 provides three independent channels of DRAM memory, yielding enough capacity to handle millions of security associations and enough throughput to handle 10Gigabit/second IPsec wire rates.

Hash for Lookups

Hashing for lookups can be used to find the required security association information for a given packet. Although other methods can be applied, combining a dedicated lookup hash unit with the use of external SRAM yields a cost-effective mechanism to conduct many required lookups.

SRAM

An important aspect of security processing is to find security associations in order to apply the appropriate cryptographic algorithms for a given flow. The four SRAM channels that the IXP2800 supports can be used to store hash tables.

PCI

When establishing security association, some operations, such as public key computations, are required. A co-processor that is connected to the PCI bus can conduct these operations.

XScale

The IXP2800 includes an XScale processor that executes general-purpose code. The XScale processor can be used for exception handling for packet processing or session setup protocols such as the Internet Key Exchange (IKE) protocol.

IXP2800 Feature Benefits

The IXP2800 allows several feature benefits for security processing. These benefits include flexibility of the implementation of the protocols, optimization of the protocols for certain applications, and the implementation of different applications using the same underlying hardware.

Protocol Flexibility

The security functionality is designed to allow support for many protocols, such as IPsec, SSL/TLS, ATM, and future protocols that use 3DES, AES, and SHA-1.

It has been proposed that the ESP protocol support larger sequence numbers. This can be achieved when combining the flexibility of the microengines and the flexibility of the cryptography unit. It is also possible to support IPv4 and IPv6 while using the underlying cryptographic hardware.

This flexibility is achieved by relying on the microengines to provide the packet processing such as header manipulation policy enforcement and post decryption payload inspection.

Protocol Optimizations

It is also possible to make optimizations depending on the application being targeted. For example, when optimizing for storage applications, large packets and few sessions are required. This makes it possible to use the microengines for adding more features. The flexibility of the network processor is key for enabling new applications because it is possible to allocate the available resources, given the new application requirements.

leverage, it definitely provided some design challenges. Since the security functions are somewhat orthogonal—depending on the configuration—it is desirable to fully utilize all security hardware in parallel. Enabling this parallelism in hardware requires careful management of common components such as global buses, local memory, and data stalling methods. It also requires substantial dexterity in switching the IVs, keys, and other state information when switching packet flows, without sacrificing performance.

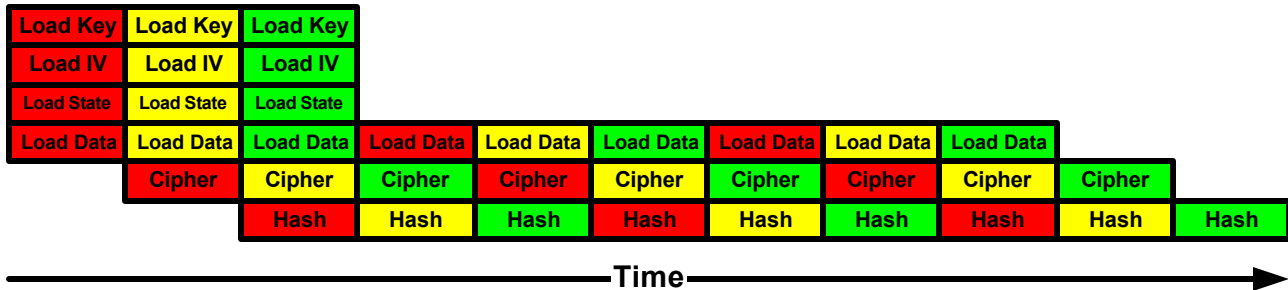


Figure 3: Security processing, pipelining, and interleaving using three wires and one core

Merging Packet Processing and Security

When designing a network security product, one must consider both the packet-processing requirements and the security requirements. A general-purpose processor coupled with a security co-processor will not be fast enough to achieve 10Gigabit/second rates with existing products. It is possible to couple an existing network processor, such as the Intel IXP1200, with a security co-processor, but today's security ICs offer only co-processor architectures, and these are insufficient.

Pipelining Security Processing

It is important to add the cryptographic functionality in such a way as to leverage the network processor features, such as multiple threads per microengine and multiple microengines.

Figure 3 shows pipelining of the functions that need to be executed to encrypt and hash data. Each color represents a different packet. The cipher key, IV, SHA-1 state and data loading can be viewed as one stage of the pipeline. While the red packet segment is being encrypted, new state is loaded for the yellow packet. While the red packet segment is being hashed, the yellow segment is being encrypted, and new state is being loaded for the green packet. For longer packets, no new state needs to be loaded, and only the data manipulation is pipelined.

Each microengine can have up to eight threads of execution running. While this multi-threaded model is one of the strengths of the architecture that we chose to

It is also important to pipeline the protocol processing. For example, when processing IPSec tunnel mode packets, it is possible to pipeline all the required processing.

Aggregating without Reassembly

Figure 4 shows a 3DES core with access to three IVs and three cipher keys. It is possible to choose which IV and key pair is used on a block-per-block basis without incurring any overhead cycles.

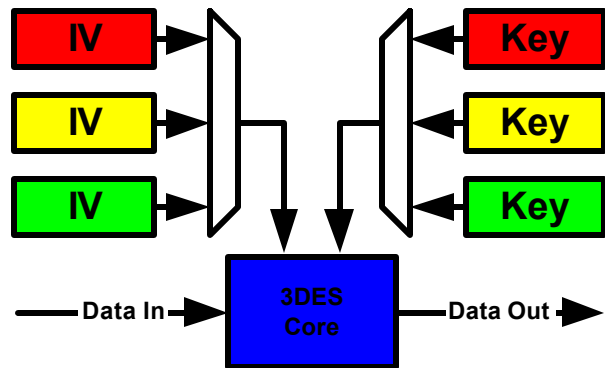


Figure 4: Multiple keys and IVs

Although it is important to achieve 10Gigabits/second rates on a single interface, it is also important to aggregate, for example, ten 1Gigabit/second interfaces. When multiple interfaces are connected to the network processor, the data of a particular packet might be interleaved with other packet data in the receive buffer. Typically, the network processor reassembles the packet

in memory. If encryption needs to be performed on the packet, then the packet must necessarily traverse the memory four times before it is sent off-chip.

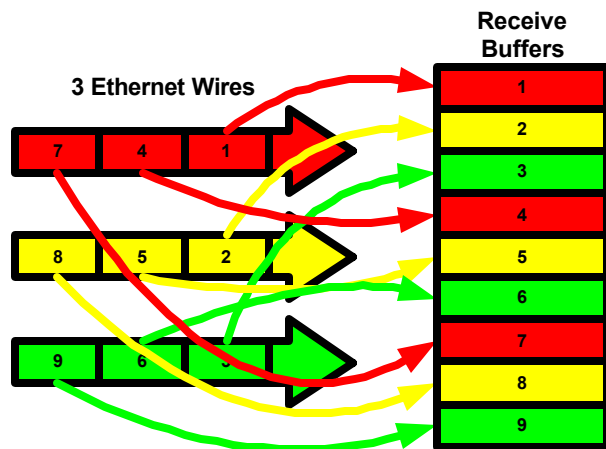


Figure 5: Multiple wire aggregation

Figure 5 shows aggregation of three wires. The packets on each wire are contiguous, but they are presented to the IXP2800 network processor interleaved with packet segments of other wires.

To avoid the reassembly before encryption, the cryptography unit supports enciphering several packets while maintaining state, such as cipher keys, cipher IVs and SHA-1 state, internally to the cryptography unit. This avoids having to reassemble the packet in memory and hence saves memory bandwidth. The states can be swapped on each block of an algorithm without any overhead cycles.

The IXP2800 supports software techniques to handle reassembly of segmented packets. The techniques are divided into two stages: Reassembly Pointer Stage (RPTR) and Reassembly State Update (RUPD). More information on RPTR and RUPD techniques can be found in [6].

Post-Decryption Processing

Given that once a packet is decrypted, further processing is required. To reduce latency, it is important that the post-encryption processing can be started before the packet is fully decrypted. This is very important for larger packets.

XScale™ is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

CRYPTOGRAPHY UNIT MICRO-ARCHITECTURE

The cryptography unit is comprised of several algorithms that in conjunction provide data confidentiality and data integrity. Each algorithm has its own set of trade-offs and challenges, in terms of silicon area, parallelism, and symmetry.

The added security functionality supports the Data Encryption Standard (DES), 3DES, and the Advanced Encryption Standard (AES) algorithms along with the Secure Hash Algorithm (SHA-1) for data authentication directly in hardware.

Figure 6 shows the data path of a cryptography unit. It consists of two 3DES cores, one AES core, and two SHA-1 cores. It is possible to process the data via the SHA-1 cores either before or after the ciphers have processed the data. The IXP2800 has two such cores.

Pipelining Around Cipher Block Chaining

During Cipher Block Chaining (CBC), encryption pipelining across several blocks of the same packet is not possible. This is because the results of the previous cipher operation are used in the calculation of the next block of data.

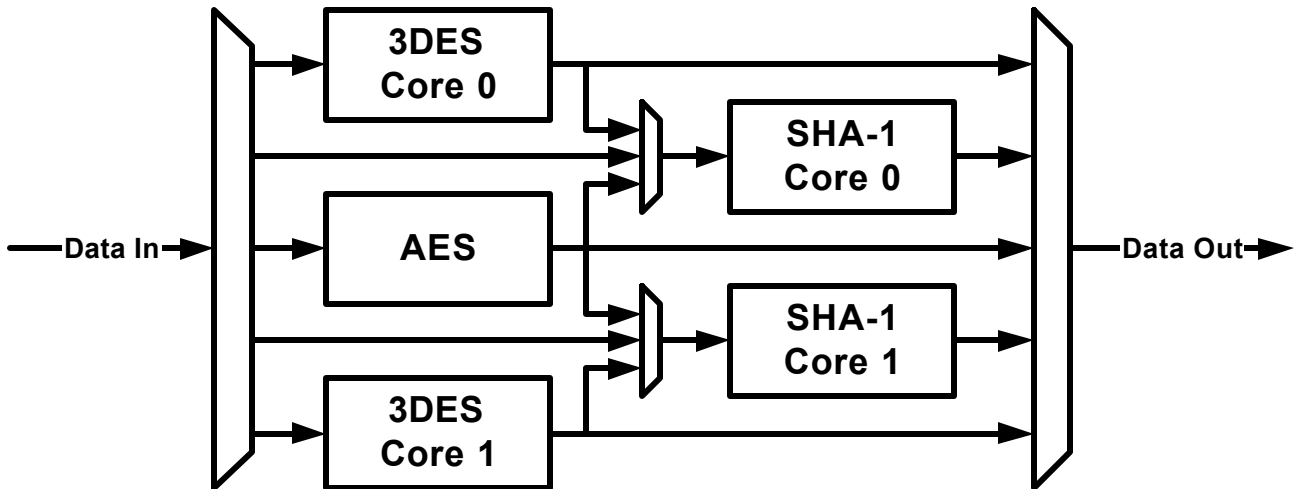


Figure 6: Cryptography unit overview

Figure 7 shows the dependency in cipher block chaining that requires the cipher text to be XORed with the next clear text block. This makes it very desirable to provide a low latency cipher algorithm implementation.

By providing each core with access to multiple keys and IVs with no overhead on transitions and by using more than one core, it is possible to overcome the CBC problem.

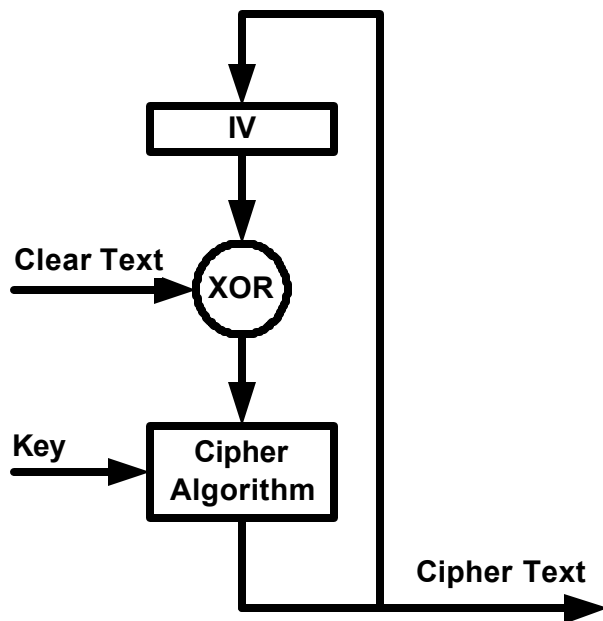


Figure 7: Cipher block chaining critical path

Although newer modes of operation do not have such a direct dependency between clear text and cipher text, it is important to support legacy modes.

Data Encryption Standard

This cipher block algorithm is a straightforward design. Since 3DES requires the performance of three DES rounds with different keys in varying modes, the hardware grabs all required keys and uses the same DES round hardware three times. The hardest design challenge was the SBOX implementation. The SBOX is a 64-entry lookup that is arrayed out 24 times and is used three times serially. In order to optimize the performance, a custom designed lookup table implemented as a passgate mux structure was used.

Advanced Encryption Standard

The AES algorithm represents unique challenges: 1) it is designed to favor encryption speed; 2) the encryption and decryption, at a first glance, require different circuits; and 3) the AES algorithm is the support for multiple key lengths.

AES is implemented both to give the same speed between encryption and decryption and to use the same circuit for all key sizes. This was challenging from an analysis and implementation point of view.

AES Key Scheduler

The key scheduler needs to support both encryption and decryption and all three key sizes. Figure 8 shows the key scheduler circuit. It is possible to load either a 128-bit, 192-bit, or 256-bit key in the available 32-bit registers. Once the key is loaded, it is possible to output the round keys for both encryption and decryption. For decryption it is necessary to compute the encryption key schedule at setup time and then load the end of the encryption key schedule for decryption processing. The same hardware is

capable of doing both the encryption and the decryption key schedule on the fly.

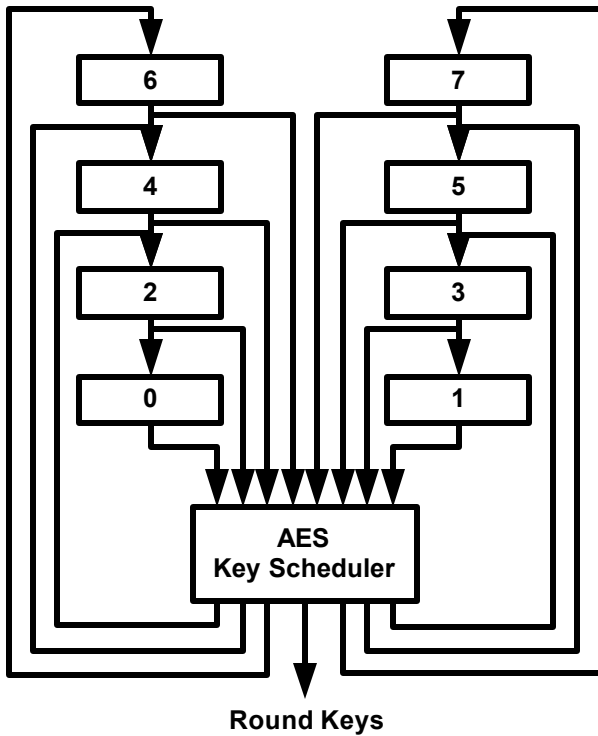


Figure 8: AES key scheduler

Secure Hash Algorithm and HMAC

The secure hash algorithm is used for data authentication. Depending on the current protocol, SHA-1 can operate either on the unmodified packet data or on the packet data after it has been modified by one of the cipher algorithms. It operates on a 512-bit block size and requires a data buffer to accumulate ciphered data. The data buffer design is important because it breaks the dependency between the data source and the actual hashing algorithm. By breaking this dependency, it creates a less complex design that enables the cipher algorithms and the SHA-1 algorithm to run at different rates.

The secure hash algorithm is used in the HMAC protocol and adds significant overhead on short packets. For example, processing a 40-byte IP packet with IPSec requires that 3 hashes be processed, assuming that the inner pad and outer pad XORed with the key are precomputed.

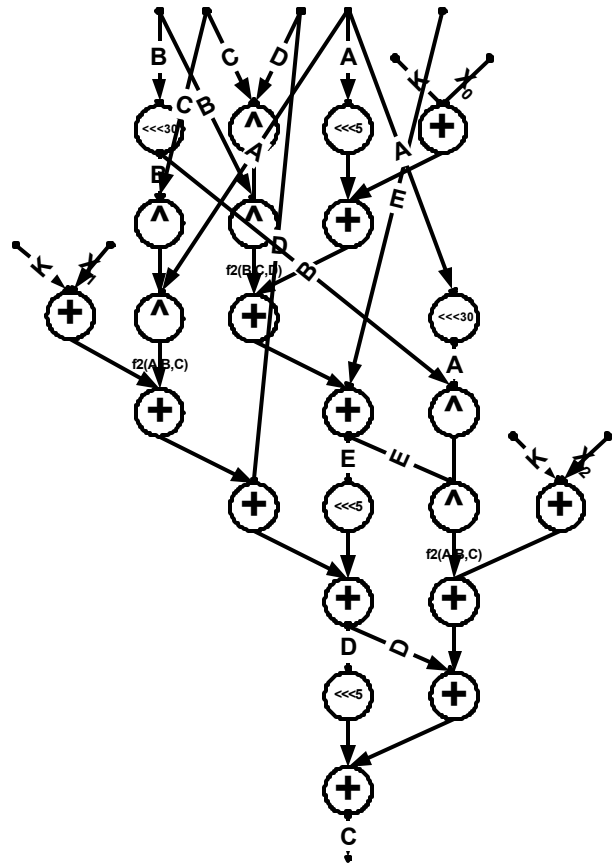


Figure 9: SHA-1 critical path

Figure 9 shows the start of the SHA-1 critical path analysis. This was done to exploit as much parallelism from the algorithm as possible while minimizing the required silicon area. The critical path analysis is then converted into a circuit.

Verification

Verification of the cryptography unit was conducted using multiple approaches. Given the amount of parallelism, it was important to validate functionality with different instruction sequences, and it was also important to validate the algorithms independently.

Cycle-Accurate Model

A C++ cycle accurate model was hand-written to provide a concrete model and a method for comparing the AES key scheduler against known results and for providing intermediate data for circuit debugging. The C++ cycle-accurate model also provided an initial overview of the control logic that would be required.

This model proved to be very successful in allowing us to try out new ideas in a short time and to validate the circuit implementation.

Instruction-Accurate Model

An important aspect of the cryptography unit architecture was the amount of parallelism that was provided. Although the individual algorithms can be verified with the cycle-accurate model, it was also important to verify that a sequence of commands produced the appropriate results. The instruction-accurate model served that purpose. Given the initial state of the cryptography unit, a sequence of commands, and the final state of the cryptography unit, it was possible to verify that the appropriate steps were carried through appropriately.

PERFORMANCE

The cryptography unit is designed to achieve 10Gigabits/second Ethernet performance when using IPSec. Although packet rate performance is important, consideration must be given as to the overall power consumption and the key agility of the system.

IPSec Performance

The cipher data path delivers over 25 million IPSec packets per second with a 40-byte clear text payload.

The SHA-1 data path delivers over 10 million HMACs per second for 40-byte clear text payloads.

This is sufficient performance to encrypt and authenticate IPSec at 10Gigabit/second Ethernet rates when 100% of the traffic needs to be secured.

Figure 10 shows the IPSec performance for a 10 Gigabit/second Ethernet wire. When securing packets with

IPSec tunnel mode, the original packet is encapsulated in a new packet, and the secure packet grows in size. The graph shows the IPSec and clear packet rate in Gigabits/second and the IPSec packet rate in millions of packets/second.

Power Consumption

One big benefit of adding security functions to the IXP2800 is a dramatic savings in power consumption for the overall solution. If current security processors were scaled to 10Gigabits/second rates, they would require between 13 and 30 Watts. Between the meticulous attention given to reducing power consumption during the design phase and, more importantly, the lack of I/O devices and overlapping silicon functionality, the IXP2800 with the Cryptography Unit reduces power by a 10X factor over its competitors. This is a tremendous benefit given the tight power constraints facing many networking equipment manufactures.

Key Agility

Key agility is important when considering the small packet rates of 10 Gigabit/second networks. The cryptography unit allows loading keys while the cryptographic algorithms are running. The cipher algorithm key schedulers run independently from the ciphers data circuits. This enables the encryption of all traffic, even minimum packet sizes, without sacrificing performance.

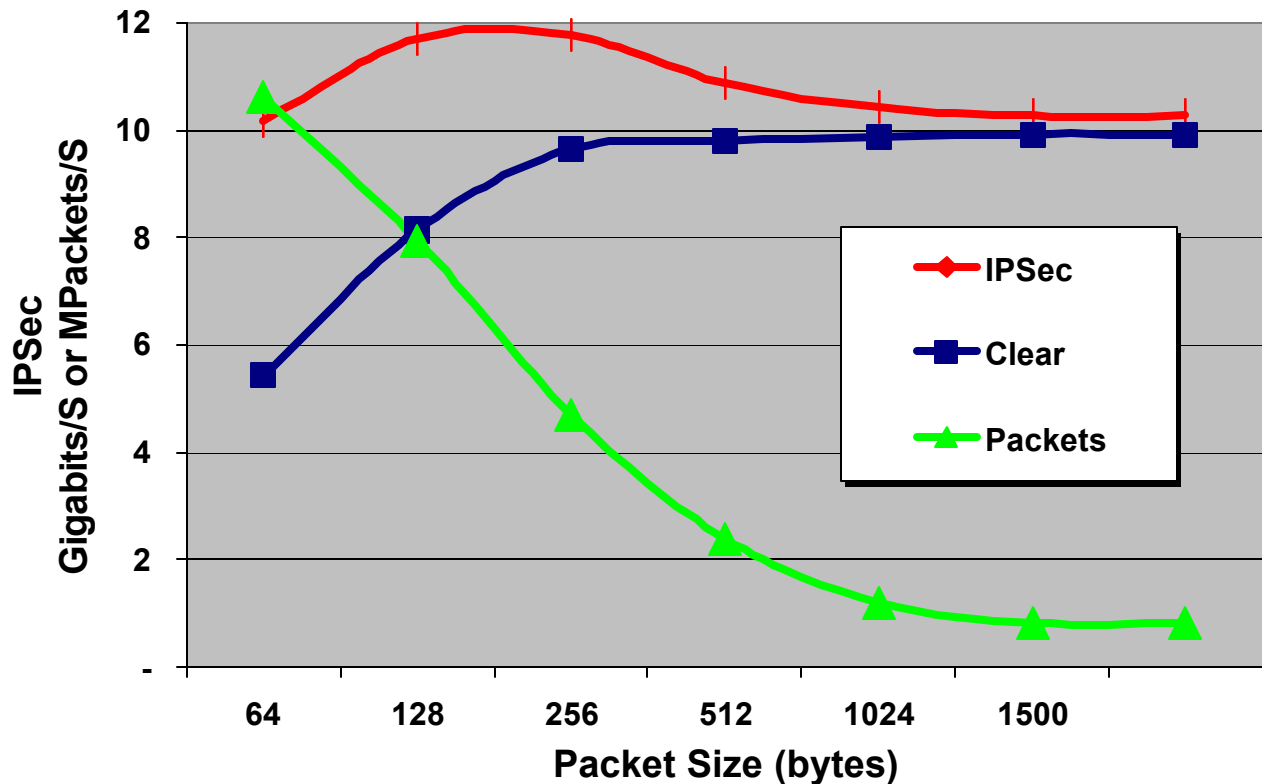


Figure 10: Performance estimates of the cryptography units doing IPsec ESP 3DES SHA-1 for 10Gigabits/second

CONCLUSIONS

The problem of providing a secure network is one that networking equipment manufacturers must tackle. The integration of security functionality onto the IXP28xx enables these manufacturers to easily solve the problem of providing that security.

The cryptographic algorithms that are implemented in silicon work closely with the other IXP28xx silicon and software resources to achieve secure data rates of up to 10Gigabits/second. Achieving the security processing requirements on a network processor makes it possible to provide not only a high-speed secure connection but also one that is achievable at a relatively low cost.

Due to the flexible nature of the IXP28xx network processor, future generations will integrate more security functionality. These functions might include public key acceleration, random number generation, and intrusion detection.

By providing both the general-purpose network processor and the security processing integrated onto a single silicon die, Intel is working to secure its position as the leading silicon provider for the networking equipment manufacturers.

ACKNOWLEDGMENTS

The authors acknowledge the contributions of John Cyr and Matthew Adiletta.

REFERENCES

- [1] Federal Information Processing Standards Publication 180-1, Secure Hash Standard, May 11, 1993.
- [2] Federal Information Processing Standards Publication 46-3, Data Encryption Standard, October 25, 1999.
- [3] Federal Information Processing Standards Publication 197, Advanced Encryption Standard, November 26, 2001.
- [4] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
- [5] W. Feghali and B. Burres, IXP2000 Cryptography Unit, Engineering and Architecture Specification, April 30, 2002, Intel internal document.
- [6] "Packet over SONET: Achieving 10 Gigabit/sec Packet Processing with an IXP2800," *Intel Technology Journal*, Vol. 6 issue 3, August 2002.

[7] M. Adiletta, et. al, "The Next Generation of Intel IXP Network Processors," *Intel Technology Journal*, Vol. 6 issue 3, August 2002.

AUTHORS' BIOGRAPHIES

Wajdi Feghali is a security architect in the Network Processor Group. He has been with Intel for two years leading the IXP2800 hardware and software security architecture. Prior to Intel, Wajdi worked for TimeStep and then NewBridge Corporations, developing secure VPN gateways. Wajdi attended the University of Ottawa and obtained a degree in mathematics in 1997. He resides in Ottawa, Ontario, and can be reached via e-mail at the at wajdi.k.feghali@intel.com.

Brad Burres is a senior component design engineer in the Network Processor Group in Hudson. He is a five-year veteran at Intel, having spent four of those years working on network processors, including the IXP12xx and IXP28xx families. Brad did the micro-architecture and ASIC implementation for the IXP28xx security solution. Brad received a B.S. degree in Computer Engineering from the University of Arizona. He resides in Cambridge, Massachusetts, and can be reached via e-mail at brad.a.burres@intel.com.

Gilbert Wolrich is a senior architect in the Network Processor Group in Hudson. He has contributed to the definition of both the IXP1200 and IXP2000 solutions. Gil has worked on high-performance network and general-purpose processors, and numerous floating-point units, and is interested in network security. Gil received a B.S. degree from R.P.I. and an M.S. from Northeastern University in electrical engineering. He resides in Framingham, Massachusetts, and can be reached via e-mail at gilbert.wolrich@intel.com.

Douglas Carrigan is a strategic marketing manager in the Network Processing Group in Hudson, Massachusetts. His e-mail address is douglas.carrigan@intel.com.

Copyright © Intel Corporation 2002. This publication was downloaded from <http://developer.intel.com/>

Legal notices at <http://developer.intel.com/sites/developer/tradmarx.htm>

For further information visit:

developer.intel.com/technology/itj/index.htm