

GRIP: A Reconfigurable Architecture for Host-Based Gigabit-Rate Packet Processing*

Peter Bellows Jaroslav Flidr Tom Lehman Brian Schott Keith D. Underwood

Information Sciences Institute
University of Southern California
3811 N. Fairfax Drive
Suite 200
Arlington, VA 22203

{pbellows, jflidr, tlehman, bschott}@isi.edu, keithu@parl.clemson.edu

Abstract

One of the fundamental challenges for modern high-performance network interfaces is the processing capabilities required to process packets at high speeds. Simply transmitting or receiving data at gigabit speeds fully utilizes the CPU on a standard workstation. Any processing that must be done to the data, whether at the application layer or the network layer, decreases the achievable throughput. This paper presents an architecture for offloading a significant portion of the network processing from the host CPU onto the network interface. A prototype, called the GRIP (Gigabit Rate IPSec) card, has been constructed based on an FPGA coupled with a commodity Gigabit Ethernet MAC. Experimental results based on the prototype are presented and analyzed. In addition, a second generation design is presented in the context of lessons learned from the prototype.

1. Introduction

The bandwidth of high-performance network interfaces has often exceeded the capabilities of workstations to process network data. The advent of gigabit rate networks and the promise of ten-gigabit networks has once again made this the case. Protocol processing of gigabit rate data leads to CPU utilizations in excess of 80% on modern microprocessors; hence, there is clearly not enough processing power to handle the 10 Gigabit Ethernet standard now coming to market. To further complicate matters, the applications using that data require some CPU time. CPU time used by

applications is not available for protocol processing. This leads to a reduction in the achievable network bandwidth. New technology is needed to enable commodity machines to leverage these new network capabilities. The proposed solution is the augmentation of the network interface with reconfigurable computing to offload processing related to the network data stream.

The network stack is typically formed of several layers of processing, as shown in [Figure 1](#). Each of these layers can have processing requirements that vary by application. The goal of a network card enhanced with reconfigurable computing is to absorb as many layers of the network stack as possible (on a per application basis) to offer the maximum possible bandwidth to the application. For example, in a cluster computer, the network and transport layers can be extremely light-weight, allowing parts of the application to be absorbed into the network interface[29]. Alternatively, two servers at two distant military sites communicating over the internet require secure communications at the network level, such as those provided by IPSec[17]. Encryption with IPSec is an extremely compute-intensive process[9] that may require most (or all) of the FPGA resources.

Providing reconfigurable computing on the network interface allows a single network adapter to serve a variety of purposes. More importantly, it provides the capability to reconfigure the network adapter to meet the changing needs of the system. This paper presents a prototype reconfigurable network interface to meet this need. A number of potential applications are described with the focus being on the network level security protocol, IPSec. The capabilities of the prototype are assessed and preliminary results for IPSec are presented along with an analysis of the results. Based on experiences with the prototype and a further analysis of the possible application domain, a second-generation architec-

*This work is supported by the DARPA Information Technology Office (ITO) as part of the Next Generation Internet program and is funded via Grant No. F30602-00-1-0541.

OSI Layers			
	7	Application	
	6	Presentation	
	5	Session	
	4	Transport	
Routers	3	Network	IPsec
Bridges	2	Data Link	LLC MAC
WDM	1	Physical	Transceiver Fiber Optics Cables

Figure 1. OSI 7-Layer Model

ture is presented for gigabit rate packet processing.

The remainder of this paper is organized as follows. [Section 2](#) presents applications under consideration. [Section 3](#) then presents the prototype developed for experimentation with these applications. Preliminary results from the prototype are then presented in [Section 4](#). Based on the preliminary work, a second-generation design is presented in [Section 5](#). Related work is then presented in [Section 6](#) followed by conclusions in [Section 7](#).

2. Applications

A range of application domains can take advantage of a network interface enhanced with reconfigurable computing. These range from intrusion detection at the link layer and encryption at the network layer (IPSec) to protocol processing at the transport layer and parallel computing at the application layer. The goal of the technology is to absorb as much of the network processing stack as possible to maximize application performance. Four example applications are discussed here.

2.1. IPSec

IPSec is a protocol suite which allows hosts or entire networks to send protected (encrypted) and authenticated data across untrusted networks. It consists of three protocols: Encapsulating Security Payload (ESP), Authentication Header (AH) and a key management protocol [17]. ESP provides protection, integrity, authentication and anti-replay services to the IP layer. AH provides the same services as ESP with the exception of encryption. The key management protocol coordinates negotiations between the peers in order to establish security associations, exchange the keys and enforce the security policies.

The cryptographic algorithms used in the IPSec protocols are too complex to be scaled to modern network speeds by sequential CPUs. For example, a test machine (266 MHz Pentium II) running an implementation of AES (Advanced Encryption Standard) [1] produced only 26Mb/s while consuming 100% of the CPU time [9]. Similar results can be found in NIST AES candidates' efficiency testing [8]. Although these results are implementation dependent, even the most optimized implementations cannot compensate for the shortcomings of serial processors. Adding this bottleneck to the existing overhead of protocol processing causes further performance degradation.

Many modern cryptographic algorithms [13] have been designed to be optimized via parallelization. The obvious solution, hardware acceleration of the cryptographic algorithms is a necessary, but not sufficient, step to increase the performance. IPSec is not a fully independent part of the network layer. It provides security services to the IP layer, but in return requires IP stack services for networking; thus, a resource interdependence is created. As a result, a crypto-only accelerator, while decreasing the CPU load, will create a new bottleneck at the PCI bus by requiring three or more PCI transactions per packet. Measurements have shown that the best throughput achievable in such systems with dedicated crypto-hardware is 12Mb/s for 1500 MTU or 75Mb/s for 9k MTU packets [18]. By integrating cryptographic acceleration with the network interface, the GRIP architecture effectively removes both the CPU and PCI bottlenecks.

2.2. Intrusion Detection

Intrusion Detection (ID) is the process of detecting inappropriate activity on a network or host system. There are generally two types of ID systems: host-based and network-based. Host-based ID systems operate on a host to detect malicious or inappropriate activity. Alternatively, network-based ID systems monitor packets on the network looking for intrusion activity. A network-based ID system is usually placed at a strategic point on the network where it can observe traffic entering and leaving a site. The system can observe traffic through a hub or a switch (with port mirroring enabled), or through passive optical splitters for interfaces with fiber optic connections.

The GRIP system described in this paper can be applied to both the host-based and network-based ID solution. A key component to any ID system is the real time processing of network traffic. Software based techniques have trouble keeping up with traffic rates in excess of 100 Mb/s. However, rates of 1 Gb/s are common today, 10 Gb/s are available, and 40 Gb/s will be available soon. For these reasons, a reconfigurable hardware based system is an ideal platform for intrusion detection. The hardware based solution will allow capture, filtering, and processing of packets at line rates.

The reconfigurable feature is important in ID systems, because attacker activity and modes are constantly changing, and the filtering algorithms must be adjusted accordingly.

2.3. Protocol Processing

The key challenge to hosts which receive high-rate network data is not moving the data itself, but the high rate of header-processing and buffer-chaining that results from the relatively small size of a packet. While the IPSec application discussed leverages jumbo frames to achieve gigabit rate throughput to the host, jumbo frames are not practical for all network connections; hence, hosts need a mechanism for improving network performance in the presence of standard Ethernet frames. To highlight the problem, consider a Gigabit Ethernet connection using standard 1500 byte frames. If one interrupt per packet is received, the result is one interrupt every $12\mu s$. It takes approximately $50\mu s$ to process a single interrupt. While standard Gigabit Ethernet cards use interrupt mitigation schemes, the resulting increase in latency is unacceptable to many applications.

The architecture presented here is capable of offloading a significant portion of that protocol processing load. In the simplest case, the network card could generate all acknowledgment packets, significantly reducing the load on the PCI bus and the host processor. In a more advanced system, the majority of the packet handling could be moved onto the network interface. In this case, a stream of received packets would be re-assembled into a data stream on the network interface. This approach would facilitate true zero-copy receives in the TCP/IP socket model. The number of interrupts in this case would be drastically reduced as the host would no longer need to be informed of the arrival of individual packets.

2.4. Cluster Computing

The field of cluster computing depends on high-bandwidth, low-latency communications between nodes in a cluster of commodity systems. The goal of these clusters is high-performance computing, so any time spent in network processing is strictly overhead. In addition, parallel applications frequently rely on parallel computing primitives that are not available from commodity high-performance network interfaces (e.g. standard Gigabit Ethernet cards). Examples of this include barrier synchronization and parallel reduction.

A single card combining a high-performance computing engine with a high-performance network interface makes an attractive addition to a cluster computer. In a cluster, where lightweight network protocols are used, such a card can absorb all of the network processing and provide a stream-based, rather than a packet-based, interface to the network.

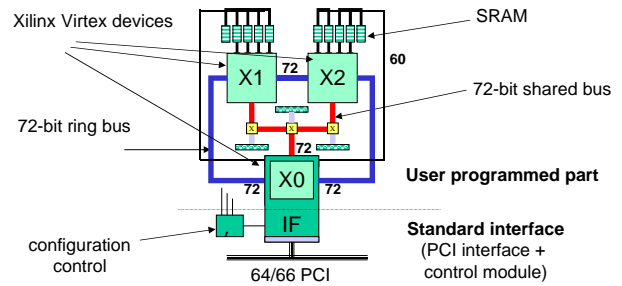


Figure 2. General SLAAC-1V architecture

This drastically reduces the load on the processor for a high-bandwidth connection. The same card can also absorb a significant portion of the application onto the network card along with the network processing (as seen in [28, 29]). As a final benefit, this enhanced network card can provide a hardware implementation of a variety of parallel communication primitives.

3. GRIP Architecture

In this section, the details of the GRIP architecture are described. First, a brief description of the SLAAC-1V card that forms the base of the GRIP architecture is given. Then, extensions to the normal SLAAC-1V system to provide GRIP functionality are presented, including the Gigabit Ethernet daughter card and the custom PCI interface / packet engine design, are described.

3.1. The SLAAC-1V Card

SLAAC-1V is a high-performance reconfigurable computing platform based on the Xilinx Virtex architecture. It was designed to address a wide variety of general memory bandwidth- and I/O-intensive applications. Its structure is shown in Figure 2. It has three user-programmable Virtex 1000 FPGAs, called X0, X1, and X2. The three FPGAs are interconnected by a 72-bit systolic ring path, plus an additional shared 72-bit bus. The FPGAs connect to a total of 10 fast ZBT SRAMs, for a total of 11 MB of SRAM memory and about 3 GB/s of memory bandwidth. The memories are individually accessible by the host, allowing for overlapped I/O and computation. The PCI interface is subsumed into the X0 FPGA, allowing for tight coupling to the PCI bus and easily customizable PCI behavior. About 80 percent of the X0 FPGA remains available for user-defined circuitry. The card provides two independent clocks to each FPGA, with programmable frequencies. The SLAAC-1V also has a bitstream cache capable of holding 7 full-sized Virtex-1000 bitstreams (5 in flash memory, 2 in SRAM), and is capable

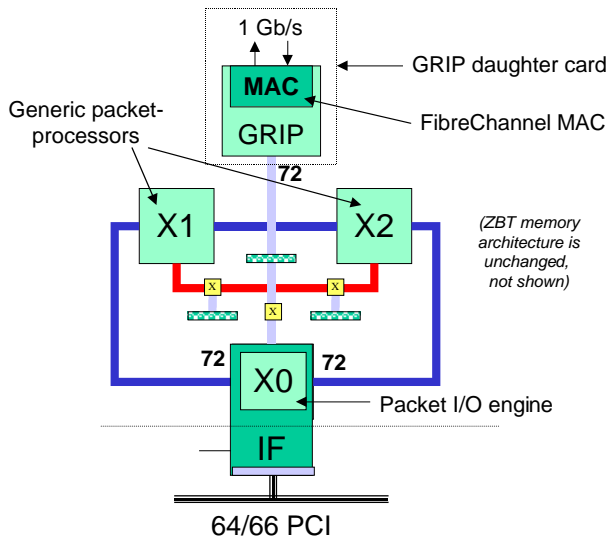


Figure 3. Block diagram of the SLAAC-1V architecture modified for GRIP

of partial configuration of the FPGAs.

3.2. GRIP extensions to SLAAC-1V

The GRIP architecture is motivated by two main factors. First, there is a need to insert hardware-assist functions in the network protocol stack in order for systems to perform useful computation at the high rates provided by modern networks. The primary bottleneck in system throughput is local bus bandwidth. Therefore, the hardware-offloading scheme must include an integrated physical interface, and the kernel driver must define a clear segregation in the protocol stack between host processing and co-processor processing; the host does all protocol processing above that segregation point, and the co-processor does all processing from that point down to the physical layer. If the co-processor cannot handle all packet processing from this “cutoff point” down to the physical layer, each packet must be transferred across the system bus at least three times, crippling performance. The second consideration in defining the GRIP architecture was the wide variety of possible hardware-assist functions for network processing. For this reason, it was important to define a standardized, generic packet-processing platform into which arbitrary packet operations could be inserted, with low overhead and design complexity.

In light of this, the GRIP architecture adds two key features to the basic SLAAC-1V system. The modified architecture is illustrated in Figure 3. First, a Gigabit Ethernet daughter card has been added to the SLAAC-1V base

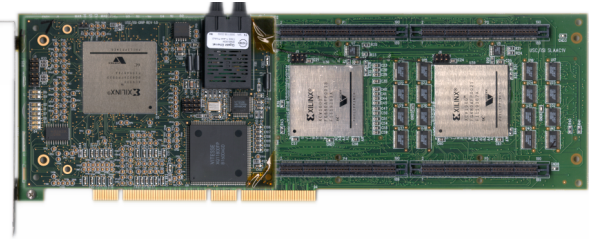


Figure 4. Photo of the assembled GRIP hardware

card to provide an integrated physical layer. The daughter card interfaces the MAC to the FPGAs on the SLAAC-1V card, and provides some packet buffering and filtering. Figure 4 shows a photo of the Gigabit Ethernet daughter card mounted on a SLAAC-1V base card. Second, the normal SLAAC-1V PCI interface in the X0 FPGA has been augmented with custom high-performance DMA capabilities, as well as packet switching and framing functions. X0 was essentially transformed into a packet mover, responsible for routing packets back and forth between the PCI bus and the other FPGAs in the system. It communicates with the other FPGAs using the 72-bit systolic ring and external I/O paths, with a simple packet-centric FIFO interface. The X1 and X2 FPGAs remain available for arbitrary packet-processing designs that implement a matching FIFO interface. The SLAAC-1V ZBT SRAMs are not used by the GRIP infrastructure, leaving them free for the packet-processing performed in X1 and X2. To date, a number of different X1 / X2 packet-processing cores have been developed by third parties for the GRIP framework, including AES (Rijndael), 3DES, packet filtering (firewall) and intrusion detection.

3.3. Gigabit Ethernet (GRIP) daughter card

The GRIP card, diagrammed in Figure 5, provides a Gigabit Ethernet I/O extension for the SLAAC-1V. It connects via a 72-pin external I/O connector on the SLAAC-1V base card. Connectivity to the network is provided by a commercial Gigabit Ethernet Media Access Controller (MAC), the Vitesse 8840. The interface to the MAC chip requires more pins than are available on the SLAAC-1V I/O connector, so a Xilinx Virtex 300 is included to provide a less complicated, narrower interface to the SLAAC-1V. Since the FPGA is needed to provide an interface for the MAC, two 512Kx16-bit ZBT-SRAMs are included for packet buffering.

Processing on the GRIP card currently consists of an interface to the MAC and some packet filtering capabilities. The packet filtering capabilities are needed because the use

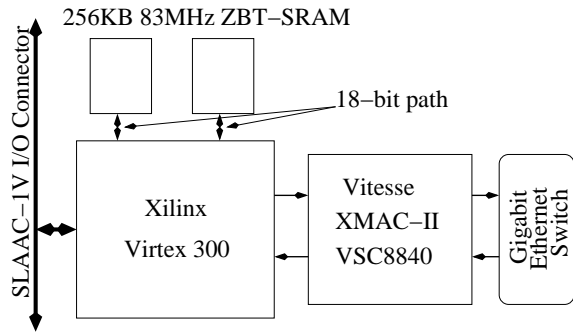


Figure 5. Block diagram of the GRIP daughter card

of jumbo frames is desired for the ongoing IPsec research effort. When configured for jumbo frames, the MAC used is unable to perform filtering of bad packets due to inadequate internal buffers. The FPGA on GRIP temporarily buffers received packets in the ZBT-SRAMs while waiting for the packet status word. The packet status word is checked for indications of an error. Good packets are passed to the X0 chip of the SLAAC-1V for processing while bad packets are discarded.

3.4. X0 Packet Processing

The X0 FPGA on the SLAAC-1V base card is the bridge between the other three FPGAs and the DMA engine, as shown in Figure 6(a). A sample packet flow is shown in Figure 6(b); X1 could hold an encryptor for outbound packets and X2 a decryptor for inbound packets, for instance. Packets are transferred between FPGAs via a common communications interface. This interface has two 34-bit, dual-clock FIFOs, allowing the core computation clock to be independent of the I/O clock. The 34-bit FIFO paths have 32 data bits and 2 packet framing bits. The framing bits delimit packet boundaries by indicating “start-of-frame” and “end-of-frame”. The communications interface also multiplexes a control channel into the chip-to-chip data stream. This control channel is used for infrequent slave accesses, like reading or writing control registers.

As shown in figure 6(a), a packet switch interconnects the three communications modules and the DMA controller. In the current prototype system, the device driver sets the switch to a static packet-routing configuration. The static switch supports a number of different routing configurations. In future implementations, a dynamic packet-switching network allowing routing decisions to be made inside X0 will be developed.

The DMA engine is a crucial component for attaining the

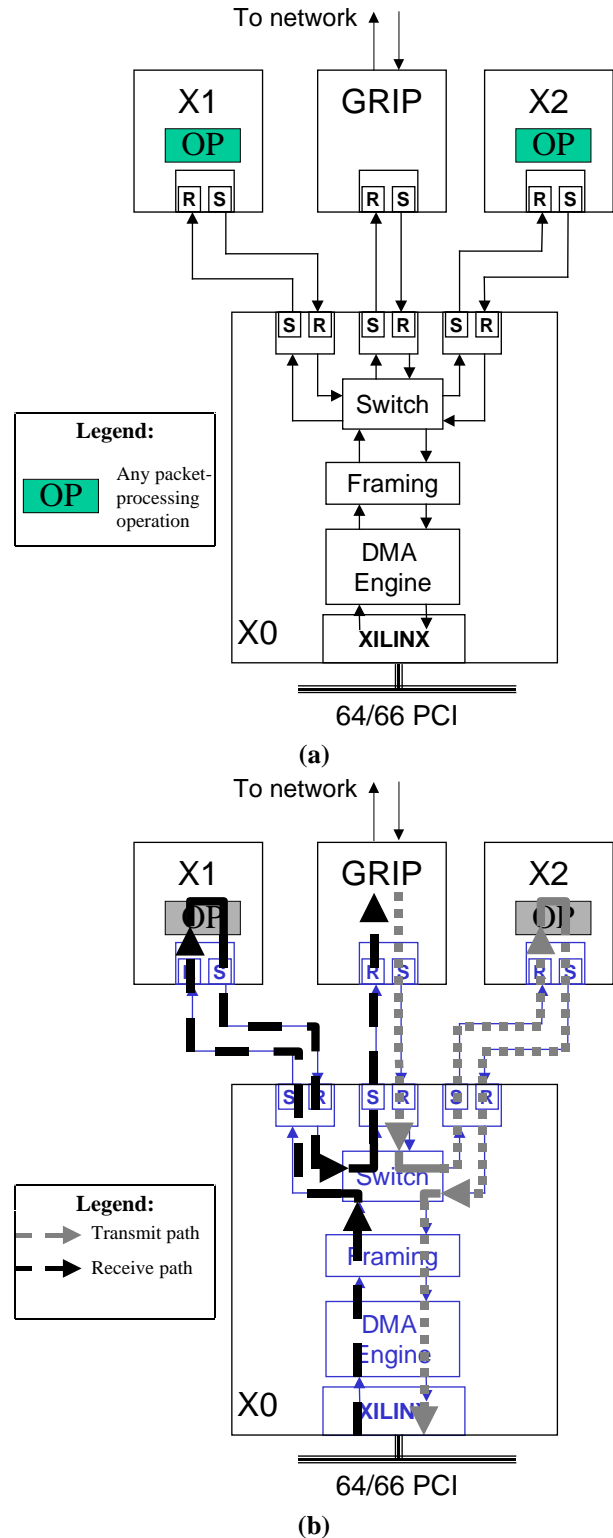


Figure 6. (a) Block diagram of the X0 design. (b) Sample packet flow through X0.

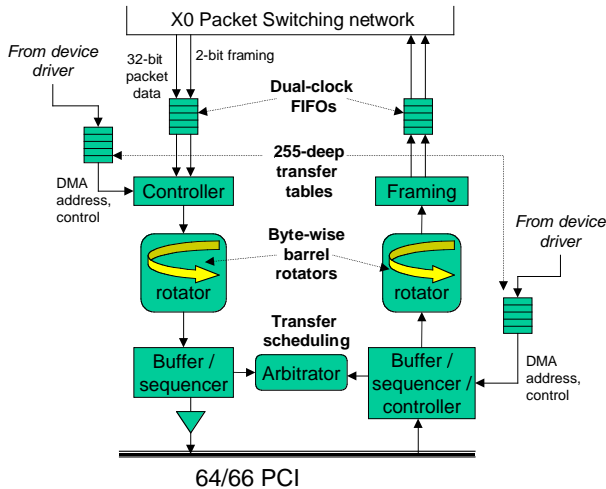


Figure 7. The GRIP X0 DMA Engine

desired line rates and packet-processing functionality. The original SLAAC-1V design incorporated an efficient 32/33 PCI design, using the Xilinx PCI core and a custom DMA engine which controlled the bus-master capabilities of the Xilinx core. This original DMA engine was optimized in four ways; the resulting design is illustrated in figure [Figure 7](#). First, it was modified to work with 64/33 or 64/66 PCI, in order to reach the bi-directional 1 Gb/s processing target. Second, deep scatter-gather tables were added using Virtex BlockRams in order to allow the DMA engine to process up to 255 independent packets in each direction between host interrupt responses. Third, a 64-bit, 8-way barrel rotator was added to the DMA engine to allow the engine to perform transfers with arbitrary byte-alignment. Finally, logic was added to the DMA engine to generate the framing bits used by the communications modules to delimit packet boundaries.

3.5. GRIP Software

The software side of the GRIP platform can be divided into two parts: a generalized driver and a modified, kernel/application function for which hardware assistance is used. The fundamental role of the driver is to present the GRIP card as a standard Gigabit Ethernet network interface to the operating system. The offloaded function is application-specific and, in this project, it is the IPSec protocol layer with its specified cryptographic transformations.

3.5.1. GRIP Driver

The most crucial component of the software is the driver. Unlike drivers for standard reconfigurable computing cards,

which are usually character devices, the GRIP driver registers itself as a network driver. The GRIP driver is partially derived from the driver for the SysKonnect [6] Gigabit Ethernet card, which shares the same Media Access Controller chipset, XMACII [7]. Currently, the GRIP driver registers itself as a regular Gigabit Ethernet driver. Its design uses the well-known ring-buffer approach with one addition. Because the SLAAC-1V board supports pseudo-scatter-gather DMA transfers (see [Subsection 3.1](#)), the driver is capable of scheduling up to 255 buffers for transmit in the DMA control FIFO without waiting for the completion of each particular transmission. Thanks to this feature, the performance can be greatly increased because the driver becomes independent of the jitter in the data flow introduced either by the hardware or software. Furthermore, when the transmit control FIFO gets full, the driver introduces a new pointer which will record the last buffer sent to the hardware and disassociate this pointer from the ring-buffer tail. In this fashion, the driver can withstand a temporary overflow without stopping the kernel transmit queue.

3.5.2. IPSec Protocol Stack

The GRIP project will use the FreeS/WAN implementation of IPSec [2] to fully integrate IPSec functionality into the Linux TCP/IP stack. In the demonstration testing, the entire IPSec layer is implemented in hardware, in very rudimentary form. In the final GRIP system, only the cryptographic transforms and checksumming will be delegated to the hardware. Since the IPSec stack registers its services with the kernel, the GRIP driver will have to do the same and present itself as both a Gigabit Ethernet interface and an IPSec interface. The IPSec stack must also be modified to accommodate the presence of the hardware accelerator. The transmit and receive functions will be modified such that packets are sent to the GRIP card without calling the software cryptographic transforms. The management of keys and the security policy database will not be accelerated initially.

4. Experimental Results

Testing of the SLAAC-1V / GRIP cards (hereafter called “GRIP cards”) has consisted of three aspects: basic functionality, functionality with packet-processing, and bandwidth. Basic functionality has been demonstrated by connecting hosts with GRIP cards to the internet. These hosts are able to perform normal network functions using the GRIP card with reasonable reliability. IPSec is being used as a proof of concept of packet-processing applications. Cores for 128-bit AES (Rijndael) encryption and decryption developed by [10] are placed within the GRIP framework to test IPSec functionality. Two GRIP cards connected to-

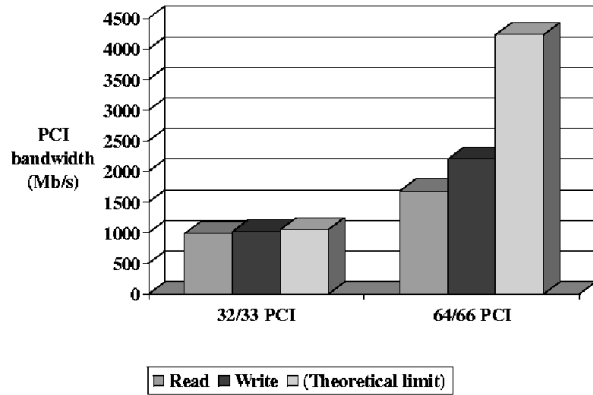


Figure 8. SLAAC-1V DMA throughput

gether are able to communicate with encrypted data. The final test is to measure the bandwidth achievable with a GRIP card. The remainder of this section details experiences with bandwidth testing.

4.1. Testing SLAAC-1V DMA performance

First, the DMA performance of the SLAAC-1V DMA engine was tested independently. The results are shown in Figure 8. Tests consisted of performing 512 DMA transfers of 2 MB each, in each direction. Tests were run using the standard SLAAC-1V device drivers on a Dell Dimension 4100 PC (850 MHz P-III) with a Linux 2.4 kernel for 32/33 PCI testing, and a Dell Precision 620 workstation (733 MHz Xeon) with Windows NT 4.0 for 64/66 PCI. The graph indicates that up to 96% PCI bus efficiency is achieved with 32/33 PCI (132 MB/s theoretical maximum). At 64/66 PCI, the bus efficiency decreases to 52% (520 MB/s theoretical maximum). Independent measurements of maximum PCI bandwidth for updated versions of the Dell Precision were similar to our findings (they measured 227 MB/s and 315 MB/s) [3]. This suggests that the lower bus efficiency is primarily due to bandwidth limitations of the test PC.

4.2. Testing the GRIP infrastructure bandwidth

The bandwidth limitations of the prototype GRIP card are being tested using Dell Precision 620 workstations with the Linux 2.4 kernel. The GRIP driver presents the GRIP card as a standard network interface. The network utility “iperf” is used to test the GRIP card over a range of packet sizes (“MTU”). The maximum throughput listed is the highest data rate at which the GRIP card completed the iperf test with less than 1% packet loss. Figure 9 indicates that achievable bandwidths range from 610 Mb/s with “jumbo-

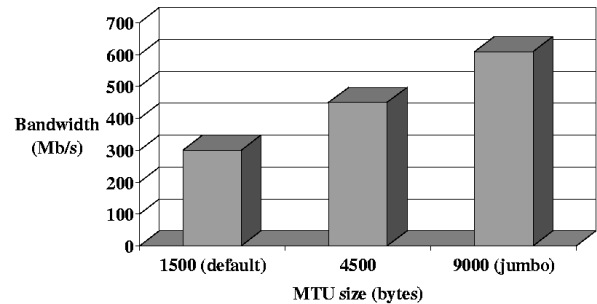


Figure 9. GRIP network throughput

frame” packets (8900 bytes) to 300 Mb/s with default packet sizes (1500 bytes).

Initial performance results are promising; however, testing is still on-going. Significant system bottlenecks still exist that are currently being addressed. When these bottlenecks are addressed, the GRIP architecture is expected to achieve the 1 Gb/s performance target. The current limitations include:

1. Clock rate limitations: the maximum clock rate for the GRIP daughter card currently limits the maximum throughput. The source of this limitation is still being investigated but the potential causes will be addressed in the second-generation card described in Section 5.
2. Asynchronous flow control: the GRIP system requires a number of independent clocks. Communications between clock domains is error-prone and is currently believed to limit the performance seen.
3. GRIP driver: the GRIP driver is in the early stages of development and has not been fully optimized.

These results are significant, but not because of the performance achieved thus far. Indeed, the results are promising, and 1 Gb/s host-to-host network traffic is expected to be achieved in the near future. However, the more important contribution is an architecture that provides an open, programmable, reconfigurable framework to offload arbitrary packet operations. Most other network accelerators are either not openly programmable or are limited to the physical or link layers of the protocol stack. The unique position of the GRIP card in the network protocol stack allows it to offload higher levels of network processing.

4.3. An example application: IPSEC

As a proof-of-concept, the X1 and X2 FPGAs have been loaded with AES (Rijndael) encryption cores devel-

oped for the SLAAC-1V board at George Mason University. These encryption cores have been demonstrated to run on a SLAAC-1V at system speeds of up to 80 MHz and bandwidth of 887 Mb/s [10]. Although these measurements were taken outside the GRIP framework, the cores have been modified and integrated with the TCP/IP stack as part of the GRIP framework. The modifications consisted of adding header-processing capabilities and extra flow control to the core. The modifications also include the addition of “encryption rules” about which packets to encrypt (e.g. control packets such as ARP are un-encrypted). This system was for demonstration purposes only. The real GRIP IPsec system will implement the IPsec header processing with a Linux kernel patch.

Testing utilized two connected GRIP cards loaded with encryption cores. Performance measured with iperf is currently limited to 50 Mb/s. Two issues are currently being investigated as performance-limiting factors: decryption failures with larger than 1500 byte packets and flow-control issues that might have been introduced with the addition of the extra header-processing logic. Header-processing will soon be moved back into the kernel. Full gigabit rate encrypted bandwidth is expected when these issues are addressed.

5. Next-Generation Architecture

The original GRIP card was built as a prototype device to allow experimentation with an FPGA-based network accelerator. The original design has a number of limitations. Key among these are a lack of memory resources, a lack of memory bandwidth, a lack of logic resources, and cost. For the second-generation card, GRIP2, a new requirements analysis has been performed.

5.1. Requirements Analysis

Work with current applications has revealed four requirements for the next-generation GRIP card that are not addressed by the current GRIP card: additional logic resources, additional RAM, additional RAM bandwidth, and low cost. Additional logic resources are needed to support the network security applications. The current GRIP card is expected to deliver gigabit rate encryption, but additional features such as key management are desired. In addition, achieving full-duplex gigabit speeds may require further protocol processing be offloaded from the host. Another application that needs additional logic resources is intrusion detection which cannot be easily implemented with the current design.

Additional RAM resources are needed for network security, parallel computing, and protocol processing applications. For IPsec, in particular, the hardware accelerator

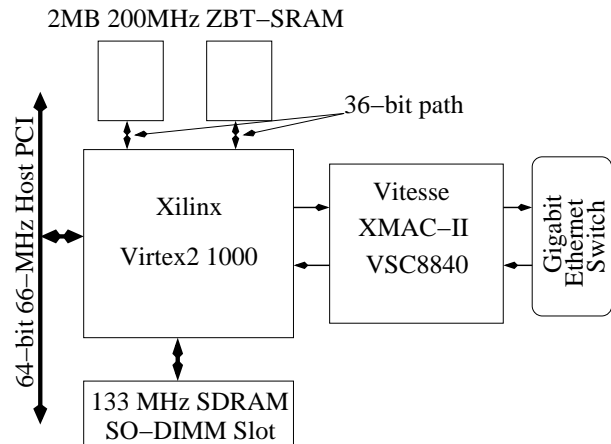


Figure 10. A block diagram of the second-generation GRIP card

needs to be capable of reassembling IP packets that are fragmented by the network. This requires storage for the packets as they are reassembled. Similarly, protocol processing requires storage for received packets, transmitted packets that have not been acknowledged, and state information about each connection. Some applications discussed for parallel computing[29] could also take advantage of additional RAM resources.

RAM bandwidth is always a concern when processing gigabit rate data. The current GRIP card has approximately 2 Gb/s of RAM bandwidth which is only enough to buffer packets in a single direction. The current configuration requires that this be used as a receive FIFO. For a next-generation GRIP card, the desire is to provide enough RAM bandwidth for buffering in both send and receive directions (4 Gb/s). This is required for protocol processing and for parallel computing applications. In addition, it is important to have additional RAM bandwidth for the storage of the state of connections and for other application processing.

Cost is a factor for any application, though it is particularly relevant to parallel computing. Parallel computing applications of this technology require a card in each node of a cluster computer. This is an issue because the current GRIP card requires an expensive SLAAC-1V reconfigurable computing card as a carrier.

5.2. GRIP2

Figure 10 illustrates a design to address the requirements outlined in Subsection 5.1. This design will be implemented as a PMC form-factor card that can be coupled with the upcoming Osiris card (Figure 11). The Osiris card provides a Xilinx Virtex-II 6000 with an SO-DIMM interface and ten

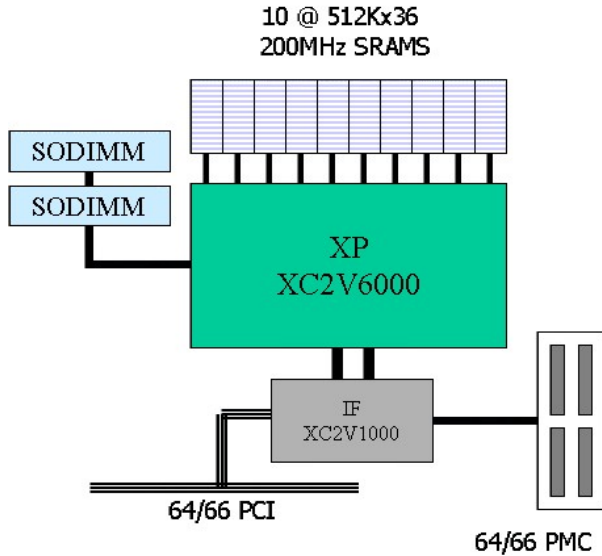


Figure 11. A block diagram of the Osiris card

banks of 250 MHz ZBT-SRAM. This provides significant additional logic resources and RAM bandwidth for use by applications. In addition, the PMC interface is easily coupled to a PMC-PCI converter card allowing the GRIP2 card to operate stand-alone, lowering the cost. On the GRIP2 card, a Xilinx Virtex-II 1000 is used as the FPGA device to provide adequate resources to implement the PCI interface as well as some application logic while minimizing costs. An SO-DIMM slot was used to provide a large buffer capable of at least 4 gigabits per second of sequential access bandwidth. In addition, two 36-bit wide, 2MB ZBT-SRAMs are provided to hold protocol related state and to provide the random access bandwidth needed by the 2D-FFT application[28].

6. Related Work

Since IPSec software implementations pose a substantial burden for the system resources, hardware accelerators have been in development since the dawn of the IPSec standard. In general, the approaches could be divided into three categories: *stand-alone*, *dedicated hardware platforms (VPN gateways)*, *crypto-accelerators* and *crypto-accelerators with a built-in network interface*. Nearly all vendors of perimeter VPN devices employ the first approach. It does not pose a significant challenge for system resource utilization because it uses proprietary technologies and essentially limitless resources. The dedicated cryptocard solution reduces the CPU overhead by offloading the computationally expensive cryptographic transforms

to hardware, but overwhelms the PCI bus at high data rates. The GRIP project has been developing the integrated packet processing/crypto-accelerator solution. While we are not aware of similar research efforts investigating the integration of the reconfigurable crypto- and packet processing, a number of people have been pursuing FPGA crypto-core research and development [21, 11, 14, 15, 16, 24, 19]. Similarly, FPGAs were used in an ATM firewall application as presented in [20]. A few commercial products employing proprietary, dedicated hardware such as custom crypto/network processors are available as well from Hifn [5] and Corrent [4].

A number of other efforts have demonstrated the usefulness of dedicated network processing. These efforts have shown that an embedded processor on the network interface can enhance parallel computing by improving bandwidth or latency or by adding special features. Examples of these efforts include HARP[23], Typhoon[25], RWCP's GigaE PM project[27], and the University of British Columbia's GMS-NP project[12]. Most recently, [26] demonstrated that the two embedded processors on an Alteon Gigabit Ethernet card could provide protocol processing at near gigabit speeds. Each of these efforts relies on embedded processor(s) and only attempts to achieve peak data rates with unencrypted data in a single direction on the network. The goal of GRIP, by contrast, is to achieve simultaneous bi-directional gigabit throughput with encrypted data. For this, additional processing power is needed on the network interface.

The advantages of an FPGA on a network interface for an application were first presented as Sepia[22], a system using a smart network adapter in a 3D rendering application. The usefulness of this type system was further illustrated for parallel applications in [28, 29]. The card architecture presented here addresses the short-comings of the prototype used in that work.

7. Conclusions

Network interface speeds today are such that general purpose CPU host systems struggle to process network data and still have resources left for application processing. In addition, the amount of processing required on network data is increasing due to the need for security protocols, cryptographic processing, intrusion detection, and IP fragmentation / reassembly. To further complicate the problem, many of these new network data processing requirements do not lend themselves to efficient processing with a general purpose CPU architecture. However, this type of processing is very well suited to hardware systems where the packet nature of network data allows easy exploitation of the parallel processing that can be implemented in hardware systems.

The GRIP architecture presented provides an example of

how hardware offload and assist mechanisms can allow host resources to be dedicated to their main purpose which is application processing. In addition, the combination of hardware based processing with real (or near real) time reconfiguration provides many opportunities for further assistance to host systems in response to new application or network processing requirements. Future areas of research will include extension of the GRIP architecture to offload additional network processing requirements including IP,TCP, and SSL.

References

- [1] Advanced Encryption Standard development effort, <http://www.nist.gov/aes>.
- [2] FreeS/WAN IPsec implementation, <http://www.freeswan.org/>.
- [3] <http://www.conservativecomputer.com/myrinet/perf.html>.
- [4] <http://www.corrent.com>.
- [5] <http://www.hifn.com>.
- [6] Syskonnect gigabit ethernet card, <http://www.syskonnect.com/>.
- [7] XMACII chipset, <http://www.vitesse.com/>.
- [8] Nist's efficiency testing for round1 aes candidates. Technical report, NIST, 1999.
- [9] M. R. et al. Performance of protocols. In *Security Protocols - 7th International Workshop*, pages 140–146, 2000.
- [10] P. Chodowiec, K. Gaj, P. Bellows, and B. Schott. Experimental testing of the gigabit ipsec-compliant implementations of rijndael and triple-des using slaac-1v fpga accelerator board. In *Proceedings of the 4th International Information Security Conference*, Malaga, Spain, Oct. 2001.
- [11] P. Chodowiec, P. Khuon, and K. Gaj. Fast implementations of secret-key block ciphers using mixed inner and outer-round pipelining. In *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2001.
- [12] Y. Coady, J. S. Ong, and M. J. Feeley. Using embedded network processors to implement global memory management in a workstation cluster. In *Proceedings of The Eighth IEEE International Symposium on High Performance Distributed Computing*, Redondo Beach, California, USA, Aug. 1999.
- [13] J. Daemen and V. Rijmen. Rijndael: Algorithm specification. In <http://csrc.nist.gov/encryption/aes/rijndael/>.
- [14] A. Dandalis, V. Prasanna, and J. Roli. A comparative study of performance of aes candidates using fpgas. In *The Third Advanced Encryption Standard (AES3) Candidate Conference*, Apr. 2000.
- [15] A. Dandalis, V. K. Prasanna, and J. D. P. Rolim. An adaptive cryptographic engine for IPsec architectures. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 132–141, Napa Valley, CA, April 2000.
- [16] A. Elbirt, W. Yip, B. Chetwynd, and C. Paar. An fpga implementation and performance evaluation of the aes block cipher candidate algorithm finalists. In *The Third Advanced Encryption Standard (AES3) Candidate Conference*, Apr. 2000.
- [17] S. Kent and R. Atkinson. RFC2401: Security architecture for the internet protocol, 1998.
- [18] A. Keromytis. Some ipsec performance indications. In *52rd Internet Engineering Taskforce meeting, IPsec WG*, 2001.
- [19] M. Leong, O. Cheung, K. Tsoi, and P.H.W.Leong. A bit-serial implementation of the international data encryption algorithm IDEA. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 122–131, Napa Valley, CA, April 2000.
- [20] J. T. McHenry, P. W. Dowd, F. A. Pellegrino, T. M. Carrozzi, and W. B. Cocks. An FPGA-based coprocessor for ATM firewalls. In *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pages 30–39, Napa Valley, CA, April 1997.
- [21] M. McLoone and J. McCanny. High performance single-chip fpga rijndael algorithm implementations. In *Third International Cryptographic Hardware and Embedded Systems Workshop - CHES 2001*, May 2001.
- [22] L. Moll, A. Heirich, and M. Shand. Sepia: scalable 3d compositing using PCI Pammette. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 146–155, Napa Valley, CA, April 1999.
- [23] T. Mummert, C. Kosak, P. Steenkiste, and A. Fisher. Fine grain parallel communication on general purpose LANs. In *In Proceedings of 1996 International Conference on Supercomputing (ICS96)*, pages 341–349, Philadelphia, PA, USA, May 1996.
- [24] C. Patterson. High performance DES encryption in Virtex FPGAs using JBits. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 113–121, Napa Valley, CA, April 2000.
- [25] S. K. Reinhardt, J. R. Larus, and D. A. Wood. Tempest and typhoon: User-level shared memory. In *International Conference on Computer Architecture*, pages 260–267, Chicago, Illinois, USA, Apr. 1994.
- [26] P. Shivam, P. Wyckoff, and D. Panda. Emp: Zero-copy os-bypass nic-driven gigabit ethernet message passing. In *Proceedings of the 2001 Conference on Supercomputing*, Nov. 2001.
- [27] S. Sumimoto, H. Tezuka, A. Hori, H. Harada, T. Takahashi, and Y. Ishikawa. The design and evaluation of high performance communication using a Gigabit Ethernet. In *International Conference on Supercomputing*, pages 260–267, Rhodes, Greece, June 1999.
- [28] K. Underwood, R. Sass, and W. Ligon. Acceleration of a 2D-FFT on an adaptable computing cluster. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2001.
- [29] K. D. Underwood, R. R. Sass, and W. B. Ligon. A reconfigurable extension to the network interface of beowulf clusters. In *Proceedings 2001 IEEE Conference on Cluster Computing*, pages 212–221, Newport Beach, CA, October 2001.