VLSI IMPLEMENTATION OF INTERNET CHECKSUM CALCULATION FOR 10 GIGABIT ETHERNET

Tomas Henriksson, Niklas Persson and Dake Liu
Department of Electrical Engineering, Linköpings universitet
SE-581 83 Linköping
{tomhe, nikpe, dake}@isy.liu.se

Abstract. *Computer networks are limited in performance by the electronic equipment. Terminals have received little attention, but need to be redesigned in order to be able to manage 10 Gigabit Ethernet. The Internet checksum computation, which is used in TCP and UDP requires specialized processing resources. The TUCFP hardware accelerator calculates the Internet checksum. It processes 32 bits in parallel and is designed for easy integration in the general purpose protocol processor. It handles UDP as well as TCP packets in both IPv4 and IPv6 environments. A synthesized implementation for 0.18 micron technology proves a throughput of over 12 Gigabits/s.*

1 Introduction

Computer networks keep increasing the transmission speeds, in the near future they will operate at several gigabits/s. It has been realized that the network equipment will have trouble to keep up with these increased network transmission speeds and several companies and research groups try to manage this problem with new hardware architectures, e.g. [1] and [2]. However, almost all of the published work concentrate on network infrastructure such as switches and routers. Less effort has been put in to deal with the end equipment, the network terminals (NTs) and their specific high-speed problems.

In NTs, traditionally layer 3 (IP) and layer 4 (TCP and UDP) processing have been handled in the CPU of the host. This will no longer be possible due to the heavy workload from the protocol processing. For 1 Gigabit Ethernet it is estimated that between 20% and 60% of the host CPU capacity is used for handling the network protocols and for the emerging 10 Gigabit Ethernet full network utilization will not be possible due to the host CPU bottleneck [3].

We have suggested a separation of the protocol processing, where only layer 7 (application layer) processing is conducted by the CPU and all other layers are taken care of by a specialized general-purpose protocol processor (GPPP) [4]. Fig. 1 shows the main idea of this separation of the processing. By doing such a separation the operating system (OS) must be reconstructed, which is a heavy but unavoidable, task.

Several benefits will be the result of such a separation. First of all, the OS will not have to execute the protocol processing code on the host CPU. This will free CPU time for the applications. Second, since the OS will execute less code, the footprint in memory will be smaller and thus less part of the cache memory will be used by the OS. Also this contributes to better host CPU utilization for the applications, since the number of cache misses will decrease.

The GPPP and the physical layer hardware can be implemented on one chip together with a similar processor for transmission and memory for payload storage. In an NT for wireless connectivity the transmission speeds will not be the critical factor, instead focus is

on low power consumption. By integrating the GPPP, physical layer hardware, memory, and transmission processor with the host CPU on one chip the total power consumption can be kept low since all communication is on chip. This requires the GPPP to be small in area as well as having a micro architecture where unnecessary switching is avoided.

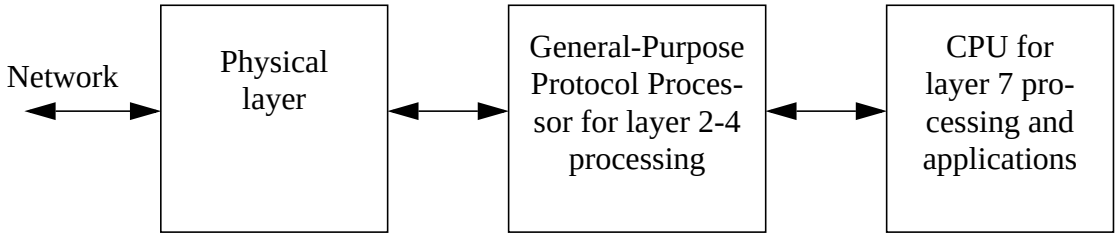

Figure 1: Overview of the protocol processing separation

By minimizing the functional overhead, such as instruction fetching and decoding, and providing flexibility by configuration possibilities instead, the high speed as well as low power operation is made possible.

One of the most critical parts of this GPPP to achieve high-throughput and low power operation is the Internet checksum calculation. The Internet checksum is used in both TCP and UDP and also for header check in IPv4. Thus it has to be calculated for almost every packet that is sent or received on the network. This is a very heavy task for the host CPU because all data in a packet has to be read from memory and then used in the computations. The GPPP handles this computation on the fly as the packet is received. All intra frame processing on a packet is conducted in the GPPP before the packet is stored in memory.

The GPPP consists of functional pages (FPs) which operate in parallel in order to allow full protocol processing at line speed. One FP is dedicated for the Internet checksum calculation. This paper describes its functionality and concentrates on the high speed operation.

The rest of this paper is organized as follows. Section 2 explains the Internet checksum calculation in detail. Section 3 introduces the concept of interlayer processing and discusses the advantages and disadvantages with that approach. In section 4 our hardware implementation is presented and section 5 then provides the results of it. In section 6 we discuss our results and finally some conclusions are drawn in section 7.

2 Internet Checksum Calculation

The Internet checksum is used for assuring correct end-to-end transmission for TCP and UDP. The sending host must calculate the checksum for the packet which is to be sent and include the complement of the checksum in the packet header. Likewise the receiver must calculate the checksum of the whole received packet, including the complement of the original checksum, and thus the result must be zero in the 1's complement number system (i.e. "1111..."). To perform the calculation the packet is split into 16-bit words, which are added by 1's complement addition. If the packet consists of an odd number of bytes it is padded with zeros.

Except for the TCP and UDP packet the IP pseudo header must also be included in the calculation. For IPv4 this pseudo header consists of the 32-bit source address, the 32-bit destination address, the protocol code (extended to 16 bit), and the 16-bit TCP/UDP packet length. For IPv6 the pseudo header consists of the 128-bit source address, the 128-bit desti-

nation address, the 32-bit TCP/UDP packet length, and the next header code (extended to 32 bits).

The 1's complement addition has the nice property of being associative, which means that the 16-bit words can be added in any order. This of course allows parallel implementation as suggested in [5] and [6]. Although a hardware implementation was suggested already 1996 [6], most operating systems still handle the Internet checksum calculation in software. As mentioned earlier this has to change soon.

3 Interlayer Processing

In the GPPP [4] protocol layers 2-4 are processed at the time of reception. This approach saves memory accesses since all operations are performed on the data as it arrives and it never has to be stored in memory before the application payload delivery. However, it also somewhat complicates the processing, especially when IP packets have been fragmented. Then layer 4 tasks, such as Internet checksum calculation, are performed on fragments rather than whole packets.

For the checksum calculation this means that partial results must be stored and caution must be taken so that the pseudo header is included exactly once in the calculation, although the IP header will be present for every fragment. It also complicates the TCP/UDP packet length calculation, which is normally calculated as (IP packet length) - (IP header length). The IP header length is specified in the IPv4 header and can be computed by tracing the extension headers in IPv6.

Totally the benefits from using interlayer processing outweigh the problems. Especially the fact that data does not have to be stored in memory more than once is crucial in a high speed implementation. The memory acts as rate converter from network speed to application speed. In traditional OSs where data is stored in memory by the Ethernet controller and then read from memory for processing on the IP and TCP/UDP layers the protocol processing is conducted at a rate which is independent of both the network and the application. It depends on the host CPU. When the protocol processing is ready, the whole packet must normally be copied in memory to the user area, so applications can access it. In case of using the GPPP the packet is directly stored in this memory area.

Interlayer processing can also be used in software implementations [7], but such an implementation cannot resolve all problems with a traditional software implementation. It has however, shown applicable to some parts of the processing and modern TCP/IP implementations use interlayer processing with great improvement over the original BSD UNIX TCP/IP implementation [8].

4 Hardware Implementation

A hardware implementation of the Internet checksum calculation has been done. This hardware accelerator implementation can handle fragmented packets and fits into the GPPP as a functional page (FP), but the core calculation unit can be extracted and used in any processor environment with the adequate supporting of control. This TCP/UDP checksum calculation FP (TUCFP) handles both TCP and UDP packets and both IPv4 and IPv6. It extracts the correct pseudo header, calculates the packet length and calculates the Internet checksum for the entire packet.

The TUCFP consists of four major units, the calculation unit, the memory, the length counter and the control, see Fig. 2. Each of them is described in some detail in the following subsections, with most attention to the performance limiting calculation unit.

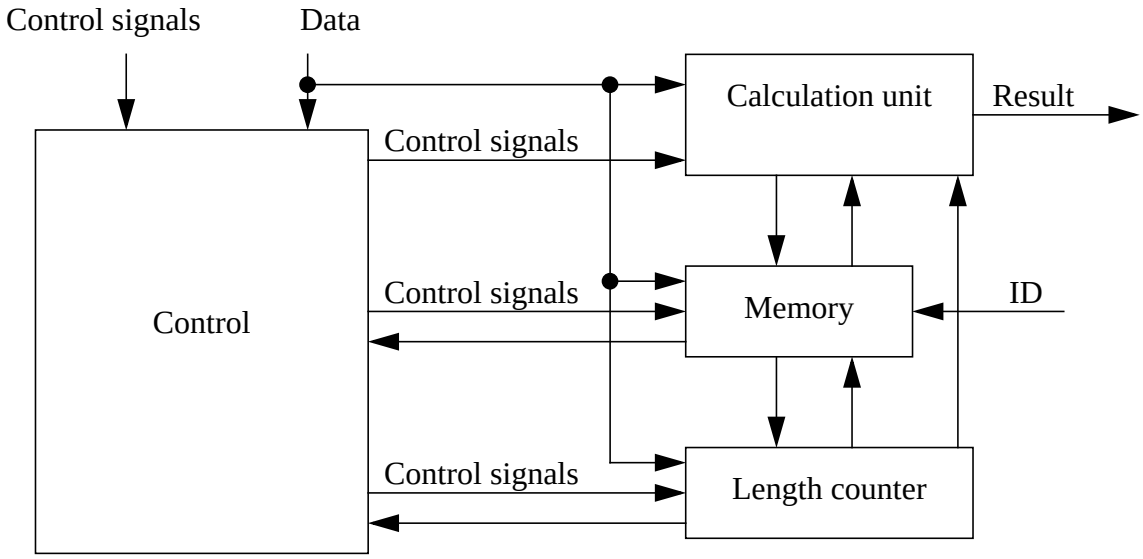

Figure 2: Overview of the TCP UDP checksum calculation functional page

4.1 The Calculation Unit

The calculation unit is the critical part of the data path. It handles all the 1's complement additions and has to support high throughput with reasonably small latency. In the current instance of the GPPP a word length of 32 bits is used. Therefore three terms have to be added each clock cycle, the accumulated value, the high part of the new data word, and the low part of the new data word.

Three different 1's complement adder structures, that can add two terms every clock cycle, have been implemented and evaluated. Table 1 shows propagation delays and area requirements for 2 sequential 2's complement adders (2 SEQ), 2 sequential pipelined 2's complement adders (2 PIPE), and direct 1's complement adder (DIR), implemented in a 0.18 micron process technology. The DIR is a carry lookahead adder, which is based on the adder structure in [6]. All values in the table are estimations after synthesis with Cadence Buildgates synthesis tool.

Table 1: Comparison of adder structures

Adder	Propagation delay [ns]	Area [mm^2]	AT[mm^2xns]
2 SEQ	0.88	0.0048	0.0042
2 PIPE	0.53	0.0095	0.0050
DIR	0.58	0.0052	0.0030

The DIR adder structure was chosen since it has the smallest area delay product. Two adders in series are needed since the input is 32 bits. However, the propagation delays for all three adders are small enough in order to support a throughput of 10 Gigabit/s, which is required for 10 Gigabit Ethernet. Later on it will be shown that the critical path consists of more parts than the actual adders, so for that reason the 2 SEQ was not sufficiently fast. The

reason for that the 2 PIPE implementation has so much larger area than the other two is that it requires a pipeline register, which was implemented as flip-flops. It also has registers on the inputs and outputs. The delay, however, was measured without the delay in the flip-flops for a reasonable comparison with the other adders.

4.2 The Memory

The memory unit stores partial results, which are present when IP packets have been fragmented. It also handles the communication with a reassembly unit, which is not part of the TUCFP. The partial results are identified by the connection state variables, which also have to be stored in the memory unit.

To be able to handle very short fragments (typically the last fragment) of IPv6 the identification process must be performed in maximum two clock cycles to fulfill the 5 clock cycle response time goal. Therefore the memory is implemented in a parallel way with registers instead of a SRAM. The memory unit is able to handle two different fragmented packets, but is designed to be easy to expand with more memory places if this is good for overall system performance.

4.3 The Length Counter

The length counter computes the lengths of the packets and the headers. It is also used to find out the total length of a fragmented packet and keep track of the accumulated length of the fragments that have arrived so far. Due to the concurrency of header length calculation and total packet length calculations, two hardware structures are necessary, one 4-bit counter for the header and one 16-bit counter, onto which all other computations are multiplexed. For ease of handling length calculations for fragmented IPv6 packets also a 13-bit subtractor is used.

4.4 The Control

The control of the TUCFP is according to the design philosophy of the GPPP mostly self contained. Only some trigger signals are received from outside. The internal control is managed by a finite state machine (FSM), which is subdivided into different branches dependent on the IP version of the received packet, if it is a fragment or a complete IP packet and so on. The FSM totally contains 76 states in order to handle all cases. Especially the fragmented packets and those with IPv6 extension headers make the handling complex from a control perspective. Fig. 3 shows the simplest case, when an IPv4 packet is not a fragment. For all other packets there are more checks that have to be made. The FSM of course also has to include the specific timing of all operations and thus is not as simple as the flowchart.

The control signals from the FSM are used for controlling all the other three units. The control always makes sure that the computation finishes within 5 clock cycles after the last data word has arrived. This exactly corresponds to the minimum time before the next packet can arrive and thus buffering of packets is never necessary.

5 Results

The TUCFP has been implemented in VHDL in RT-level. This section describes the results after the design and synthesis steps.

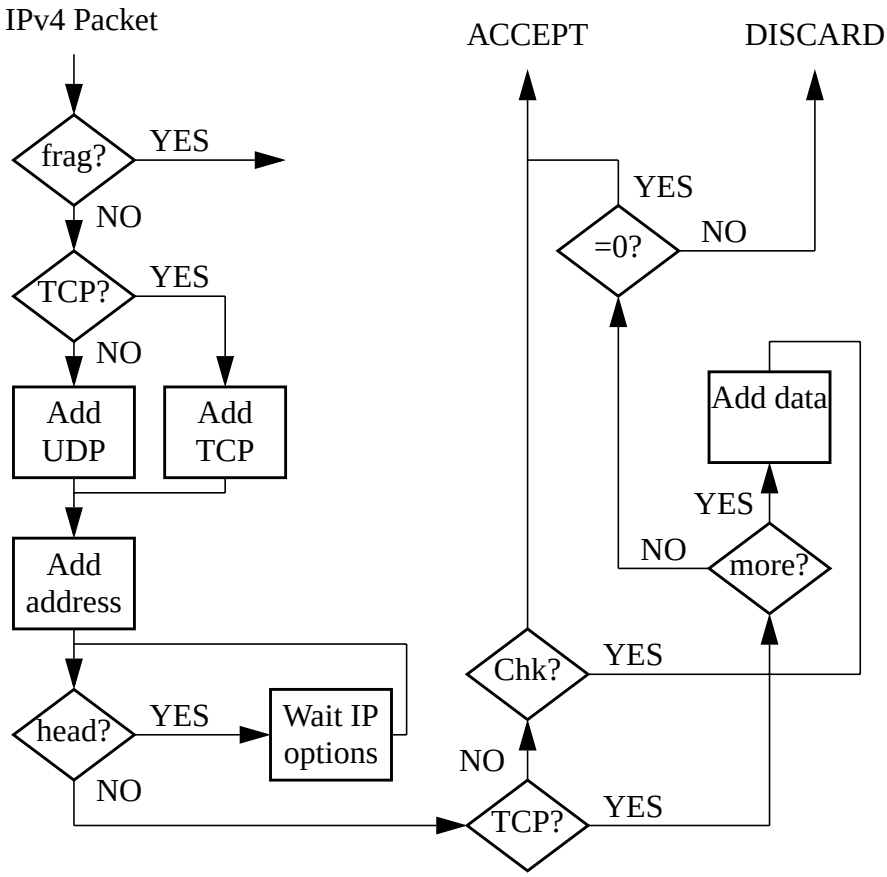

Figure 3: Flowchart for IPv4 handling when packet is not a fragment

5.1 Verification Results

The function of the TUCFP was verified by using a test bench in the VHDL environment and simulate it in Mentor Graphics Modelsim. The test bench read one binary test vector containing a packet and one with the corresponding control signals coming from other parts of the GPPP.

Binary packet test vectors were created with a packet generator written in C. The packet generator is able to create correct UDP and TCP IPv4 packets from random input data. It is also able to fragment packets. Real life IP packets were dumped with tcpdump and used in the tests. IPv6 test vectors were created by converting the IPv4 test vectors. Because of the their simplicity the control signal test vectors were created manually.

The verifications shows that the TUCFP can correctly handle IPv4 and IPv6 packets containing UDP and TCP. It can also handle fragmented packets regardless of which order the fragments arrive in or if fragmented packets are nestled into each other.

5.2 Synthesis Results

The VHDL code was synthesized by Cadence Buildgates, then static timing analysis (STA) provided the performance estimations. Performance and area estimations are presented in table 2. The target technology was a 0.18 micron 6 metal layer process. The 1.7% of the area that is not part of any module is used for interconnections. The critical path starts at the outputs of the finite state machine in the control and goes through the calculation unit to the

inputs of the registers in the memory unit. It is split up according to 0.73 ns from the finite state machine outputs to the control - calculation unit interface, 1.76 ns through the adder structure in the calculation unit, and 0.13 ns from the calculation unit - memory unit interface to the inputs of the registers in the memory unit, see Fig. 4.

Table 2: Estimations after synthesis

Module	Delay [ns]	Max. freq.	Throughput	Area [mm2]	Area [%]
Calculation Unit				0.053	31
Control				0.0128	7.5
Length Counter				0.017	9.8
Memory Unit				0.086	50
TUCFP	2.62	381 MHz	12.21 Gbit/s	0.171	100

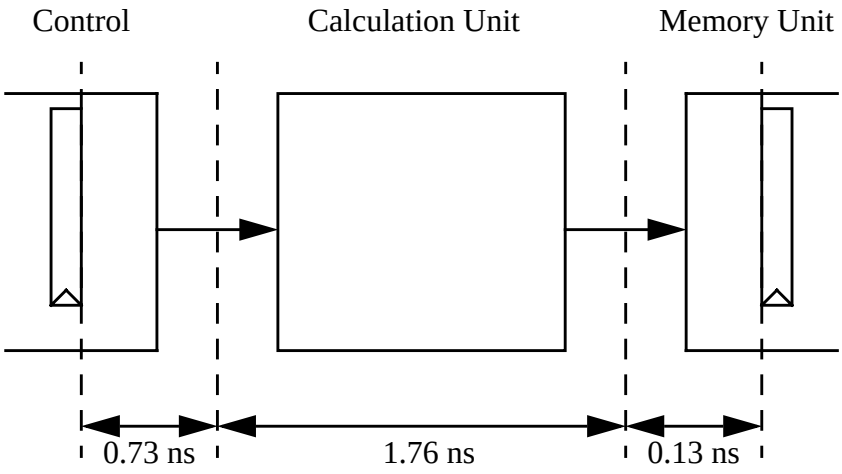

Figure 4: Analysis of the critical path

6 Discussion

The TUCFP that was designed and implemented supports 10 Gigabit Ethernet. The largest part of the area is used for the memory unit. The memory unit is needed only for fragmented packets. In IPv6 there is a mechanism for path maximum transmission unit discovery which should avoid fragmented packets. If it can be guaranteed that no fragmented packets will appear on the network the TUCFP can be significantly simpler since the memory unit can be removed and the complexity of the control path can be reduced. Also the length counter can be simplified.

The TUCFP was aimed for 10 Gigabit Ethernet, but in the future there might be a need for even higher transmission speeds for data base and web servers. The calculation unit of the TUCFP can already now support more than 10 Gb/s. If new process technologies do not offer enough speedup, pipelining can be used in order to increase the capacity even more. This would imply that the control path must be redesigned.

The reason for that the calculation unit contributes 1.76 ns to the critical path, but only had 0.58 ns delay in the adder comparison is that the calculation unit consists of two adders

in series with a multiplexer. A pipeline stage in the calculation unit would split the critical path almost in the middle and a factor of almost 2 can be expected in the throughput gain.

All estimations here are after synthesis, but before layout. The layout will add some interconnect delay, which is not included in our estimations, however it will be less than the margin of over 20% down to 10 Gb/s throughput requirement.

In a wireless communication environment the transmission speeds are much lower than those discussed here. The TUCFP and the GPPP can be used in an appliance for such conditions simply by lowering the clock frequency. The low functional overhead assures low power consumption.

7 Conclusions

A hardware accelerator for TCP and UDP checksum calculation has been designed and implemented in 0.18 micron technology. It can support 10 Gigabit Ethernet transmission speeds by using 32 bit parallel computation at a frequency of 312.5 MHz. This accelerator is a critical device in a protocol processor which is used to separate the protocol processing from the application processing in a network terminal. For 10 Gigabit Ethernet networks the capacity limitation is the network terminal. This accelerator helps circumventing that bottleneck in order to allow higher capacity data base and web servers.

The next phase of our project will integrate the accelerator with the rest of the general-purpose protocol processor and a host CPU.

References

[1] Kobayashi, M., Murase, T., Kuriyama, A.: A Longest Prefix Match Search Engine for Multi-Gigabit IP Processing, Proceedings of the IEEE Conference on communications 2000, Vol. 3, pp. 1360-1364

[2] Bux, W., Denzel, E. D., Engbersen, T., Herkersdorf, A., Luijten, R. P.: Technologies and Building Blocks for Fast Packet Forwarding, IEEE Communications Magazine, Jan. 2001, pp. 70-77

[3] Merritt, R.: iWarp interface spec could bust IP bottlenecks, http://www.eetimes.com/story/OEG20011207S0091

[4] Henriksson, T., Nordqvist, U., Liu, D.: Specification of a configurable General-Purpose Protocol Processor, CSNDSP 2000, Bournemouth, U.K., pp. 284-289

[5] Braden, B., Borman, D., Partridge, C.: Computing the Internet Checksum, RFC 1071, Network Working Group, September 1988

[6] Touch, J., Parham, B.: Implementing the Internet Checksum in Hardware, RFC 1936, Network Working Group, April 1996

[7] Ahlgren, B., Björkman, M., Gunningberg, P. G., The Applicability of Integrated Layer Processing, IEEE Journal on Selected Areas in Communications, Vol. 16, No. 3, April 1998, pp. 317-331

[8] Wright, G. R., Stevens, W. R.: TCP/IP Illustrated, Volume 2: The Implementation, Addison-Wesley 1995, ISBN: 0-201-63354-X