CONTROL PATH IN A PROTOCOL PROCESSOR

Ulf Nordqvist and Dake Liu
Computer Engineering Group
Linköping University, SE-581 83 Linköping, Sweden
Email: {ulfnor, dake}@isy.liu.se

Abstract- *In this paper a dual processor architecture dedicated for offloading of protocol processing tasks in network terminals is presented. The architecture includes one general purpose microcontroller handling control intensive tasks, and one programmable protocol processor (the PPP) accelerating data intensive tasks. The PPP uses a number of accelerators for datapath operation. The main focus of this paper is on how implementation of the PPP control path can be configured for different types of terminals, e.g. servers, gateways or desktop PCs. Static timing analysis in implemented layouts indicates that this architecture enables programmable TCP/IP offloading for multi-gigabit networks, when implemented in a mature process technology (0.35 μm).*

I. INTRODUCTION

Both computer and human communication networks use protocols with ever increasing demands on speed, cost, and flexibility. In order to let the end-users take advantage of the bandwidth enhancement in today networks, tomorrows Network Terminal (NT) hardware must support transmission speeds of Gbit/s. Hardware for such NT components is on the other hand sold on a cost-sensitive market share with high demands on flexibility and usability. Traditionally NT has been implemented using ASIC:s situated on the network interface card processing the lower layers in the OSI-Reference Model and a CPU-RISC based SW implementation of the upper layers. Usage of standard, general purpose CPU:s, is expensive in terms of cost, area and power due to their lack of dedicated hardware. Today it is easy to find Network Interface Card (NIC) supporting multi-gigabit networks but such bandwidth can not be utilized by the host since it requires the host to be fully loaded processing layer 3 and 4 protocols, leaving nothing for the application and system processing. The research focus has mainly been on router and switching applications so far [1], but in the future the terminals will also require offloading using programmable high-speed solutions.

In chapter 2 our dual-processor terminal protocol processing architecture is introduced. Chapter 3 discusses the control path of one of the two processors with focus on the performance limiting tasks of program flow selection. In chapter 4 three implementation strategies for this control path are proposed. In chapter 5 some performance figures for two of the alternatives are listed and finally in chapter 6 some conclusions and directions for further work are listed.

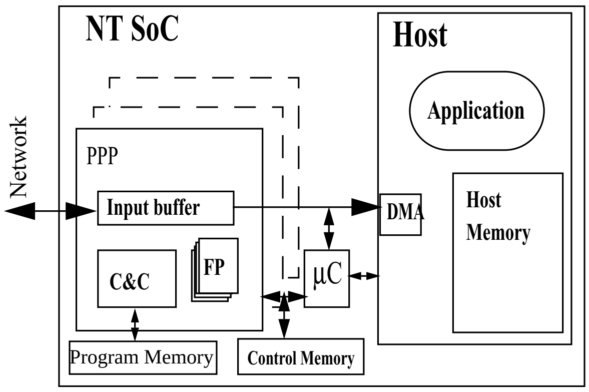

Figure 1. The PPP together with a general purpose micro controller (μC) handles the communication of one network media port.

II. PROGRAMMABLE PROTOCOL PROCESSOR

A Protocol Processor (PP) architecture intended to be used as a offloading device in a network terminal was proposed by the authors in [2]. As most PP, it consist of more or less programmable devices that can accelerate and offload a host processor, by handling the communication protocol processing. The protocol processor is a domain specific processor that have superior performance over general purpose CPUs but still provides flexibility through programmability within the application domain. The proposed architecture has a unique dataflow based strategy for storage and wirespeed processing of incoming packet data. Instead of storing the data in a input buffer before it is processed as traditional network processing hardware, the proposed architecture manage some of the fast path processing before the packet is either discarded or stored for further processing. An overview of the PP architecture is illustrated by. The PP is a dual processor architecture depicted in figure 1. The first processor is a general purpose micro controller responsible for the control intensive processing of the slow path. This type of processing tasks is common in upper protocol layers such as TCP/UDP. The second component is the Programmable Protocol Processor (PPP) which is responsible for the high-performance acceleration of the data-intensive processing tasks, i.e. packet decoding. This fast path process the data on wirespeed as it streams through a chain of flip-flop based registers. The PPP consists of a number of accelerators denoted as Functional Pages (FP) (e.g. [8]) implementing the fast path.

Note that our dual processor architecture means that the PPP only is responsible for packet decoding. All connection state (inter-packet processing) is handled by the micro controller. Further only the FPs accesses packet data. This means that the C&C only need to control FPs based on flags generated by FPs. The C&C can therefore be regarded as the control path of the PPP.

The remaining of this paper will focus on the control path of the PPP.

III. CONTROL PATH

The Counter and Controller unit (C&C) depicted in figure 1, is responsible for starting and stopping FP processing, based on the program and the result flags from the FPs. The C&C is also responsible for the decision to discard or accept a packet. Further the C&C select program counter (PC) values based on the flags from the FPs. The C&C must for example select the correct program flow when the incoming packets protocol type has been checked.

A Program Flow Selection

In protocol processing for terminal reception a significant part of the tasks is program flow selection, i.e. determine what to do based on header information. After header data has been extracted, the header data must be compared with a number of different values in order to identify the type of packet. The result of this comparison then decides which processing (program flow) the protocol processor shall perform on the packet. In order to fulfill the requirements set by protocol standards, many packet header fields must be extracted and processed. The complexity of the comparison grows with the number of protocols, protocol (e.g. IP source and destination) addresses etc. This program flow selection essentially translates as a case-statement in software.

A case-statement can be executed sequentially using (at assembly level) load, subtract and conditional branch instructions in most processors. The number of required clockcycles it takes to perform such an case-statement depends on the number of entries and which case is selected (e.g. protocol type). Hence, the C&C must run at a higher clock frequency compared to accelerators (FP) or perform the complete case-statement in a single clock-cycle in order to produce all control signals needed by accelerators.

IV. IMPLEMENTATION ALTERNATIVES

This paper proposes three different implementation strategies for the control path (C&C) of the protocol processor. Implementation strategy is selected based on the protocol coverage, i.e. the type of applications and terminals it is intended to offload. Hence, a domain specific solution is obtained.

A Pipelined C&C

By dividing the execution of an tasks into smaller parts, each of the parts takes smaller amount of time to perform. If parts of different tasks then are executed in parallel using pipeline scheduling the performance can be dramatically improved. I.e. since only a part of the instruction has to be executed each clock cycle, the clock frequency can be increased compared to that of a non-pipelined processor. Pipelining is a technique used in almost all modern processors. A pipelined processor can run on high clock-frequency and therefor manage many instructions for each network-clock cycle. In the experimental pipelined TIPP-processor [3] from Intel the execution core runs at a clock-frequency which is 32 times higher than the network and classification memory speed. Hence, theoretically up to 608 clock-cycles are available for each packet. This number is in fact reduced by the use of low-frequency CAM-memories (classification and reassembly). Further these 608 clock-cycles include the cycles lost as pipe-line penalties when conditional branch instructions are executed. This pipeline penalty is dependent on the depth of the pipeline. Since conditional branches are common instructions in protocol processing this may seriously limit the performance. Hence there is a trade-off between high-frequency (deep pipeline) and branch cost (few or no pipeline-stages).

The proposed pipelined C&C uses an application specific ISA. The ISA is very simple since most processing tasks are handled by FPs. The C&C is essentially only function as the PPP control path. The simple instruction set consists of 11 instructions. They are listed in table 1. Arithmetic instructions are used for counting and handling of inter packet information for fragmented packets, i.e. accessing fragmented packets temporal length and checksum results.

Table 1: ISA for the pipelined C&C

Type of instruction	Name
Logic	And
	Or
	Not
Arithmetic	Add (16 bits)
	Sub (16 bits)
Transfer	Load (immediate)
	Move
Jump	Jump unconditionally
	Brneqz
	Breqz

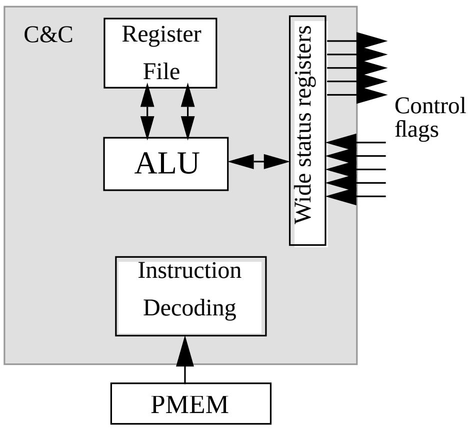

Figure 2. C&C organization for general purpose terminal processing. The register file is not needed if fragmented packets are not allowed.

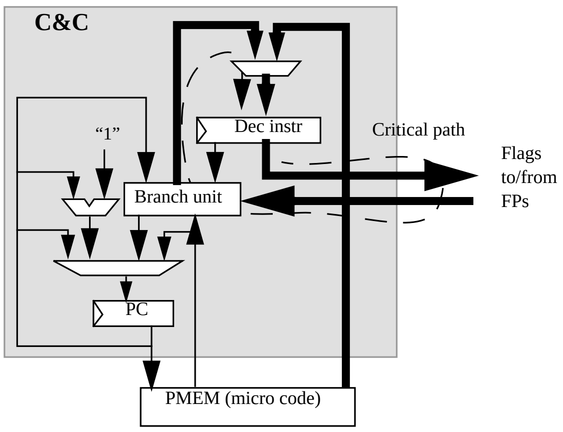

Figure 3. Branch unit supporting single clock cycle program flow selection. The critical path includes extraction of packet header data, comparison and branch decision.

According to behavioral simulations the C&C needs three instructions per network clock cycle to handle UDP or TCP/IPv4/Ethernet using the specified ISA when only one destination address is checked. I.e. we need three times as many instructions compared to the synchronized implementation due to increased task complexity. Complexity comes from a more complex classification and reassembly handling. This number (three) is rather low compared to many other (e.g. [10.6] uses 32) pipelined protocol processors but the number of tasks and protocols handled by the C&C is also lower. E.g. checksum calculations has been offloaded. High frequency operation is enabled using pipelining techniques. The number of pipeline stages used is five. It requires the program memory to be pipelined. We have not implemented such a memory but the size of the memory is small and using pipelining it should be possible to achieve very high speed memories using full custom design techniques.

B Synchronized C&C With Branch Unit

A synchronized data-flow processor runs at the same clock-frequency as the incoming data. This require each instruction to be more complex compared to pipe-lined processors. Further pipeline penalties are not allowed since the data- and the instruction-stream must be fully synchronized at all times. In order to achieve this the processor must accelerate complex tasks using self-contained accelerators. Further conditional branches must be executed in a single clock-cycle. If these two conditions are met the processor can manage high-performance protocol processing operating at a low clock frequency.

If the number of entries to the case-statements is limited, the program flow selection can be implemented in parallel dedicated hardware. This allows for the program flow selection to be performed in a single clock cycle. If all conditional branches are supported by this type of program flow selection hardware, all branch penalties can be eliminated. This means that the C&C unit can use the same clock frequency as the data-flow pipeline (i.e. the network clock). By using the same clock frequency the requirements on synchronization with the data flowing through the PPP becomes very strict. Since the data only is available to each FP during one clock cycle it is necessary to start and stop the processing in the FPs at the exact clock cycle. The concept of synchronized protocol processing was first introduced by Henriksson et al in [4] even if the proposed implementation is slightly different.

The use of a single clock domain simplifies the layout and reduces synchronization problems between FPs and the control path (C&C). I.e. the need for synchronization registers is eliminated. An implementation alternative for the C&C suitable for synchronized special purpose protocol processing is illustrated in figure 3.

There are many alternative ways of implementing a hardwired case-statement. Using Content Adressable Memories (CAM) is one rather straight forward alternative ([5] and [6]). The CAM based branch unit depicted in figure 4 uses the PC value and flags generated in FPs as inputs. If there is a match in one of the CAM entries, i.e. a conditional branch is taken, a new instruction and instruction fetch address are provided. Since the branch unit will be a part of the critical path of the PPP and thereby determine the maximum clock frequency it can operate at, it is very important to optimize this CAM search. The latency of a CAM search is mainly dependent on the size of the two search fields and the number of entries in the memory. The latency of the CAM search must be added to the latency of the FP and Muxes to find the critical path of the PPP.

C Pipelined C&C With Branch Unit

Even if a pipelined processor is used it is possible to accelerate program-flow selections using a branch unit. The size of the branch unit and the clock frequency to be used can be optimized after the protocol coverage has been set. The branch unit then makes it possible to accel-

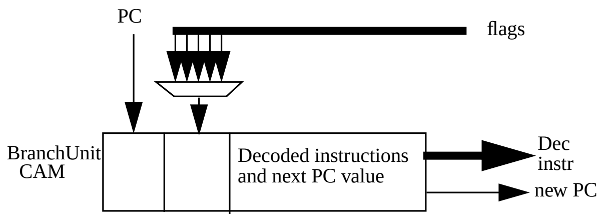

Figure 4. The branch unit makes its branch decision based on the program counter value and flags created in FPs.

erate case-statements. Normally classification is accelerated using pipelined classification engines, e.g. CAM but also other types of case-statements can be covered using branch unit acceleration.

Since the branch unit can be pipelined it is possible to reduce the critical path compared with alternative A, thereby enabling a higher clock frequency while still allowing for a large number of entries in the branch unit memory. In fact alternative C is the only possible for complex terminals with a large number of protocols, source and destination addresses since the number of entries in various case-statements implemented is to large. Note, that synchronization registers are still needed.

V. PERFORMANCE

Using AMS 0.35 µm standard cell library an implementation of the synchronized data-flow version of the C&C has been completed. The implemented branch unit supports 16 different conditional branches, each with four case-entries. Static timing analysis of the implemented layout shows that the critical path is 10.9 ns long which indicates that it can support wire speed processing at 2.9 Gbit/s. This is comparable with the performance of a configurable CRC FP implemented using the same process. Four entries to the case-statement means that only four destination addresses (including multi-cast addresses) can be checked.

The critical path of the pipelined C&C is the ALU. Static timing analysis of the layout of the C&C datapath shows that the simple 2-pipeline stage ALU can run at 588 MHz when implemented using AMS 0.35 µm standard cell library. If the C&C runs at a four times as high clock frequency this enables the PPP can support for a network speed of 4.7 Gbit/s. If the protocol coverage is increased, the C&C might have to run at eight times as high clock frequency compared to the network interface (GMII). This however still allows for more than 2 GBit/s of wirespeed processing.

VI. CONCLUSIONS AND FURTHER WORK

This paper proposes three different implementation strategies for the control path of a data-flow based protocol processor. Based on protocols covered and terminal type the C&C can be optimized for high speed operation. By enabling high speed and low latency program flow selection, the overall throughput is optimized. Timing analysis indicates that multi-gigabit network speeds are feasible for a restricted set of protocols when the C&C is implemented in a mature standard cell technology.

As future work a more complex protocol stack would be interesting to investigate implementation alternative C. Specially the size of the branch-unit and number of pipeline stages have to be carefully optimized using benchmarks.

REFERENCES

[1] Crowley, Patrick, et al, “Network Processor Design”, first edition, Morgan Kaufman Publishers, ISBN: 1-55860-875-3

[2] D. Liu, U. Nordqvist, and C. Svensson, “Configuration-Based Architecture for High Speed and General-Purpose Protocol Processing”, *IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, 1999, pp. 540-547.

[3] Y. Hoskote, V. Erraguntla, D. Finan, J. Howard, D. Klowden, S. Narendra, G. Ruhl, J. Tschanz, S. Vangal, V. Veeramachaneni, H. Wilson, J. Xu, N. Borkar, “A 10GHz TCP offload accelerator for 10Gbps Ethernet in 90nm dual-VT CMOS”, IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Paper, 14.7.

[4] T. Henrikson, U. Nordqvist, and D. Liu, “Specification of a Configurable General-Purpose Protocol-Processor”, Proceedings of CSNDSP 2000, Bournemouth

[5] McAuley A. et al., “Fast Routing Table Lookup Using CAMs”, IEEE INFOCOM ‘93, March 1993

[6] van Lunteren J., Engbersen A.P.J., “Multi-field packet classification using ternary CAM”, Electronics Letters, Volume: 38 Issue: 1, 3 Jan. 2002, pp 21 -23

[7] M. B. Abbott, L. L. Peterson, “Increasing network throughput by integrating protocol layers”, IEEE/ACM Transactions on Networking, vol. 1, pp 600-10, 1993

[8] U. Nordqvist, T. Henriksson, D. Liu, “CRC Generation for Protocol Processing”, in proceedings of the NORCHIP 2000, Turku, Finland