A TCP Offload Accelerator for 10 Gb/s Ethernet in 90-nm CMOS

Yatin Hoskote, *Member, IEEE*, Bradley A. Bloechel, *Associate Member, IEEE*, Gregory E. Dermer, Vasantha Erraguntla, *Member, IEEE*, David Finan, Jason Howard, Dan Klowden, Siva G. Narendra, *Member, IEEE*, Greg Ruhl, James W. Tschanz, *Member, IEEE*, Sriram Vangal, *Member, IEEE*, Venkat Veeramachaneni, *Member, IEEE*, Howard Wilson, Jianping Xu, *Member, IEEE*, and Nitin Borkar

Abstract—This programmable engine is designed to offload TCP inbound processing at wire speed for 10-Gb/s Ethernet, supporting 64-byte minimum packet size. This prototype chip employs a high-speed core and a specialized instruction set. It includes hardware support for dynamically reordering out-of-order packets. In a 90-nm CMOS process, the 8-mm² experimental chip has 460 K transistors. First silicon has been validated to be fully functional and achieves 9.64-Gb/s packet processing performance at 1.72 V and consumes 6.39 W.

Index Terms—Gigabit Ethernet, offload, packet processing, special-purpose processor, TCP.

I. INTRODUCTION

THIS PAPER presents an experimental Transmission Control Protocol (TCP) offload engine that uses special-purpose hardware and is programmable via a specialized instruction set. Intended as a prototype to demonstrate this offloading approach, the chip performs a significant amount of TCP input processing on 64-byte minimum size packets at wire speed for 10-Gb/s Ethernet.

General-purpose microprocessors are rapidly becoming overwhelmed with the burden of processing TCP and Internet Protocol (IP) packets on Ethernet links that are growing exponentially in capacity. Fig. 1 shows a graph plotting CPU utilization on a state-of-the-art Pentium® 4 class server while processing IP packets on a saturated 1-Gb/s Ethernet link. The uni-processor server has its CPU at 100% utilization even at large packet sizes of 64 kbytes. For a dual-processor server, at larger packet sizes, one of the CPUs has to be completely dedicated to processing incoming packets. As the packet size decreases, the available processing time decreases and the burden on the CPU goes up. At packet sizes of 128 bytes, both the CPUs are completely utilized. This is clearly an undesirable situation. The challenge in wire speed protocol processing is that for 1-Gb/s Ethernet stream there are 1.48 M minimum size packets coming in per second. This gives the CPU only 672 ns to process each packet. For 10 Gb/s, the arrival rate is 14.8 M packets per second, giving only 67.2 ns to process each packet, a prohibitive requirement on the CPU. A generally accepted rule of thumb for network processing is that 1-GHz CPU processing frequency is required for a 1-Gb/s Ethernet

Fig. 1. CPU saturation at smaller packet sizes.

link. For smaller packet sizes on saturated links, this requirement is often much higher [1]. Ethernet bandwidth is slated to increase at a much faster rate than the processing power of leading edge microprocessors. Clearly, general-purpose MIPS will not be able to provide the required computing power in coming generations.

One approach to address this problem is to provide hardware support to the CPU by offloading some of the tasks involved in processing the Open Systems Interconnection (OSI) layers [2], [3]. The level of difficulty in offloading these tasks to hardware engines increases significantly as we move up the seven-layer OSI stack. All layers above the link layer (L2) have traditionally been processed in software. Layer 3 or IP layer is already moving toward hardware. The next logical step is to offload the transport layer (L4). From a functionality point of view, transport layer processing at an end station can be divided into inbound and outbound processing (Fig. 2). Inbound/outbound processing units provide fast data plane processing power while the host CPU through a fast input/output (I/O) interface, such as PCI-Express, provides control plane functionalities such as synchronization and intercommunication. The inbound Ethernet stream comes from the physical (PHY) layer through the medium access control (MAC) controller and I/O interface to the inbound processing unit. The packet filter is used to determine if the traffic belongs to the end station and is in the right format. An IP re-assembler puts the fragmented IP packet into its original form once it checks the fragment bit and the offset field. The protocol demultiplexer forwards the packet to the appropriate protocol processing unit. The outbound processing unit accepts an outbound packet from the send buffer. Both inbound and outbound processing tasks could be performed by the

Manuscript received March 31, 2003; revised June 14, 2003.

The authors are with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: yatin.hoskote@intel.com).

Digital Object Identifier 10.1109/JSSC.2003.818294

Fig. 2. Input processing block diagram.

same physical engine. This work focuses on TCP processing because it is very compute intensive and is the protocol most dominantly used on the Ethernet, amounting to roughly 82% of the protocol usage [4].

The prototype described here focuses on inbound TCP processing, with outbound processing limited to acknowledgment messages. While inbound and outbound processing may be equally complex, the time budget for inbound processing is usually much tighter. The goal was to design and build an experimental chip that can handle the most stringent requirements: wire speed inbound processing at 10 Gb/s on a saturated wire with minimum size packets. Another priority was to ensure that the design cycle was short by keeping the design simple, flexible, and extensible. As opposed to a solution that uses a general-purpose processor dedicated to TCP processing, this approach involved design of a special-purpose processor targeted at this task. In order to adapt quickly to changing protocols, the chip was designed to be programmable. This feature also served to simplify the design and greatly reduce the validation cycle as compared to a fixed state machine architecture. Specialized instructions in the instruction set significantly reduce the processing time per packet. In addition, the chip is architected so that it is possible to easily scale down the high-speed execution core without any re-design if the processing requirements in terms of Ethernet bandwidth or minimum packet size are relaxed.

It is important to note that this chip is an experimental prototype designed to serve as a proof-of-concept vehicle to show the value of a special-purpose programmable offload engine. Constraints on available die area and short time to tapeout strongly influenced design decisions such as number of active connections supported. Consequently, we focused on the engine details rather than system level issues such as payload transfer, limited memory bandwidth, and host CPU interface. Extension of this prototype to a product would require additional work in those areas, for instance, minimize number of memory accesses and hide memory latency. Section II of this paper describes the prototype chip architecture, Section III gives details of the design, Section IV goes over the design methodology, and Section V describes the results.

II. ARCHITECTURE

This chip targets header processing and control dominated tasks, rather than the storage and forwarding of packet payloads.

Fig. 3. Major functional blocks.

It performs connection establishment and tear down, checks validity of incoming message, computes payload length, processes incoming flags, performs window management tasks, identifies and reorders out-of-order packets, and assembles response packets. Briefly, the steps in processing an incoming packet are as follows: the connection to which that packet belongs is identified and that connection state is loaded into a working register in the execution core for processing. The execution core performs TCP processing tasks using the state information under direction of instructions from an on-board instruction store. The connection state is updated and an output packet is assembled as a response to the input packet. These tasks are completed before arrival of the next packet.

In order to achieve this level of performance, the chip uses a dual-frequency design with two clocks, a major clock and a higher speed minor clock. The architecture, shown in Fig. 3, consists of a high-speed core operating in the minor clock domain, fed by memory units that operate on the major clock and store context information. This approach enables a buffer-free design that achieves wire speed processing. The input sequencer parses the incoming header information and forwards data appropriately inside the chip. With a 32-bit-wide input bus operating at the major clock frequency, a required bit rate of 10 Gb/s translates to a major clock frequency of 312.5 MHz. Lookup of a connection and loading of connection state into the working register is done by the context lookup block (CLB) and transmission control block (TCB), respectively. The 64-entry TCB memory unit stores the context information for an existing Ethernet connection at the same index location that the CLB stores the 96-bit connection identifier [5]. Successful lookup of a connection causes 33 bytes of connection state to be loaded into the working register from the TCB in a single major clock cycle. The high-speed execution core, controlled by instructions from the instruction ROM, performs the central part of the TCP processing. The results are stored back in the TCB and the output packet is generated and eventually assembled in the send buffer. The 33 bytes of stored context information for each connection is sufficient to implement the inbound processing tasks offloaded in this prototype. The reorder block (ROB) is used exclusively to dynamically reorder out-of-order packets. All memory operations occur only once for every packet, keeping performance degradation minimal. The memory units (TCB, CLB, and ROB) are thus largely idle during processing of a packet, decreasing power consumption.

General purpose minor clock instructions	Special purpose major clock instructions
LOAD A ← data	CLBCLR index
MOV A → B	ROBCLR index
AND A B → cond	CLBLKUP key → index, data
OR A B → cond	ROBLKUP key → index, data
ADD A B → C	ROBWR key ← data
SUB A B → C	ROBEMPTY → cond
CMP A B → cond	TCBWR index ← data
EQUAL A B → cond	TCBRD index → data
NOT A → C	
BREQZ / BRNEQZ label	
JMP label	
SHL2 A	
NOP	

Fig. 4. Specialized instruction set.

Die size constraints on the prototype chip limited the number of active connections to 64. Scaling the design to support larger number of connections (for example, 4000) will linearly increase the size of the memory units with minimal impact on the core. To support an even larger number of connections, the TCB can be viewed as a cache with support for additional connections in off-chip memory. In such an organization, an efficient replacement policy and a mechanism for hiding the memory latency, such as multiple cores or threads, would have to be implemented.

A specialized instruction set, shown in Fig. 4, was developed for efficient TCP processing. It includes special-purpose instructions for accelerated context lookup, state loading, and write back. These instructions enable single-cycle CLB lookup, CLB write and clear, as well as single-cycle 33-byte-wide TCB reads and writes. Generic instructions operate on 32-bit operands. These make up the heart of the TCP processing code. The complete microprogram implemented to perform TCP inbound processing consists of 306 lines of code.

A. Processing Budget

A minimum size Ethernet packet consists of 64 bytes: 14-byte MAC header + 23-byte IP header + 23-byte TCP header + 4-byte frame check sequence. The time budget for processing such an incoming packet of 64 bytes (plus 20-byte interframe gap) is shown in Fig. 5. The packet transfer at 10 Gb/s requires 67.2 ns, corresponding to 21 major clock cycles. A larger packet, which includes payload, increases the available processing time. After reading context information from the TCB and by overlapping the TCB write back operation with the CLB lookup for the next packet, a total of 19 major cycles or 60.8 ns is available for the high-speed core to process a minimum size packet. At an operating speed of 5 GHz for the core, this would allow execution of up to 304 instructions [6]. Simulation traces show that the worst case path through the instruction program for in order packets arriving on an established connection takes only 116 instructions. After including CLB and TCB operations and branch and synchronization penalties, this worst case path translates to a total processing time of 57.6 ns (18 major clocks), which is within the processing budget of 67.2 ns. In the worst case, processing an out-of-order packet takes 73.6 ns. This increase is due to execution of three ROB major clock operations

Fig. 5. Processing budget.

to perform reordering, as well as synchronization penalty between clock domains for these ROB instructions. To maintain wire speed processing, this processing budget corresponds to a lower limit of 92 bytes on packet size, rather than the minimum 64 bytes. However, out-of-order packets will likely have some payload in them and therefore, span extra clock cycles, enabling the micro engine to take advantage of the additional processing time available without sacrificing wirespeed performance.

The processing budget is, thus, split between the minor clock and the major clock domains with synchronization at the clock domain boundaries. Due to the decoupling between the major clock domain and the high-speed core, we can modulate the performance of the core in accordance with the processing requirements. As the size of packets to be offloaded increases, the performance requirement on the core decreases. This inverse relation between the inbound packet size and the frequency of operation of the core is shown in Fig. 6 for different input Ethernet rates. The highlighted point shows our current operating point with a 5-GHz core processing 10 Gb/s at wire speed. Bringing the core down to 1 GHz allows us to offload only those packets larger than 364 bytes. If the Ethernet rate decreases to 1 Gb/s, a 500-MHz core is sufficient to handle minimum size packets. This core frequency modulation can result in significant savings in power consumption, as borne out by the measured results.

Another approach to power management is to use multiple engines operating at lower frequencies to achieve the same processing performance. Connections would be split across engines and each packet would be directed to the appropriate engine owning that connection by performing a parallel lookup on all the CLBs. This requires an external control unit to arbitrate between engines for new connections, for demultiplexing the input stream and multiplexing the output stream. The performance degradation due to added latency, added control complexity, design cost, and die size has to be traded off with savings in power.

B. Dynamic Reordering

Packets can frequently arrive out of order [7]. Reordering these packets in software or by implementing a sorting algorithm in hardware is expensive and cumbersome. This chip implements a novel dynamic reordering algorithm in hardware with the use of content addressable memories (CAMs) that eliminates the need for sorting. The ROB contains two CAMs that store pointers to packet payload that are indexed by the sequence numbers of the packets, the first sequence number of the

Fig. 6. Packet size versus required core frequency.

packet in one CAM and last+1 sequence number in the other CAM. Arrival of an out-of-order packet triggers a lookup in both CAMs, using the first and last+1 sequence numbers of the new packet as tags, to check if the new payload is adjacent to any existing out-of-order payload. If so, adjacent payloads are merged, thereby reducing CAM entries. If the lookup fails, a new entry is created in both CAMs for that out-of-order packet. Arrival of an in order packet requires only one lookup using the packet's last+1 sequence number to check if the succeeding adjacent payload exists. If so, it is forwarded to the user and the corresponding CAM entries are cleared. This method maintains the number of CAM accesses per packet to be a constant of two lookups and one write for an out-of-order packet and at most one lookup for an in-order packet. This constancy is critical to achieve wire speed processing. The out-of-order packets must be nonoverlapping for this scheme to be efficient. For effective communication between the two clock domains, the high-speed micro engine is stalled while reordering is performed. The penalty due to stalling is minimized by the re-ordering algorithm by limiting the number of CAM accesses per packet.

III. DESIGN DETAILS

A. Micro Engine

The working register-execution unit-instruction ROM loop shown in Fig. 7 is the high-performance micro engine at the heart of the design. The 264-bit-wide working register is loaded with initial values on initiation of a connection or with data from the TCB on resumption of a connection. The working register supports TCB loads on the major clock and core write back on the minor clock.

The execution unit is a three-stage 32-bit ALU with the three pipelined stages as source select, ALU operation, and write back. The source and destination operands are chosen from among 26 fields of the working register, receive buffer, and internal scratch registers through wide multiplexer trees. The working register holds the data containing the current connection's state, the scratch registers contain intermediate processing data for the current packet, and the receive buffer contains the actual packet header data. Immediate data is part of the instruction word that is sent from the instruction ROM. The ALU inputs are determined by a 26:1 one-hot muxing scheme, implemented by two levels of multiplexers. This allows the pass gates to be directly driven by the appropriate segment

Fig. 7. Micro engine block diagram.

Fig. 8. ALU organization.

of the instruction word and results in minimum delay and maximum performance. Since the bulk of the packet processing is done in the high-speed core, it is critical to optimize it. Consequently, the ALU performs add, subtract, compare, and logical operations in parallel for added speed as shown in Fig. 8. The appropriate result is chosen and the appropriate destination register (or send buffer) is enabled, allowing for write back. The adder in the ALU uses a quaternary tree architecture, which is split between the second and third pipe stages. The paths through the compare and add blocks are critical paths. The condition register sends control bits back to the instruction ROM for use by branch instructions.

Proper care was taken to overcome the large interconnect penalty and extreme routing congestion in the core. The 264-bit-wide working register was split into two halves (MSB and LSB in Fig. 9). All fields were further split into groups according to bit number. The ALU was placed in the center and the two halves were aligned with the corresponding bits in the ALU to minimize the interconnect distance.

The instruction ROM (Fig. 10) is a two-stage 80-bit $\times$ 320-entry column-multiplexed pipelined array also operating on the fast minor clock. For reduced local bit line (LBL) length, the ROM is organized as five banks of 64 bits each. Each bank receives 16 wordlines (WLs) that select the appropriate bit to discharge one of 16 LBLs. A sense amplifier receives column-multiplexed data from 16 possible LBLs and performs a 16-way merge followed by a second five-way merge on the global bit line (GBL). For high performance, each LBL was restricted

Fig. 9. Execution unit organization.

Fig. 10. Instruction ROM organization.

to a maximum of eight devices. The final five-way merging and data latching is accomplished with a NAND set dominant latch (SDL). The ROM implements a single-phase domino design that achieves high frequency of operation by hiding the precharge latency. Traditional ROM designs precharge the LBL, the sensing, and the GBL stages simultaneously. In this design, the precharge and evaluation operations for the LBL, sense stage, and GBL are staggered with respect to each other as shown in Fig. 11. The set-dominant latch (SDL) at the output makes the design robust at low frequencies. This implementation provides the benefits of two-phase domino and the simplicity of single-phase domino designs.

The ROM has a two-cycle latency. During the first clock cycle, the 9-bit address is decoded, and during the second clock cycle, the decoded address is evaluated and the correct instruction word is read out. Instructions are stored in fully decoded form to avoid decode delay penalty. The ROM control block provides the correct address to the array every clock cycle. A simple static decoder was implemented to reduce power consumption and clock loading. For most instructions, the next address is the current address incremented by one. The target address for jumps is specified in the instruction word itself and, consequently, jumps incur no execution penalty. The target address for branches is dependent upon condition codes generated by the core. Since branch evaluation takes two clock cycles, the ROM control block dynamically inserts two NOPs after every branch instruction. Thus, in the absence of any branch prediction mechanism branches always incur an execution penalty. The control block also dynamically inserts a NOP between successive instructions

Fig. 11. Instruction ROM operation.

that exhibit data dependency. Whenever the ROM issues a major clock instruction, it stalls and waits for the operation to complete before resuming.

B. Memory Units

The CLB and the TCB store the context information for active connections. These memory units operate on the major clock and do not have stringent requirements on their performance. The CLB is a CAM used as a lookup table for the TCP connections. The 96-bit key input to the CAM corresponds to the source and destination ports and addresses of a TCP connection and a match is performed on the entire 96 bits. In case of a hit (match), the CLB output represents the address of the matched connection in the TCB. In case of a miss, the output represents an empty location in the TCB where the new connection state can be written. Each CAM lookup operation completes in a single major clock cycle.

The TCB is a register file (RF) used to store the context information specific to each TCP connection. Optimization of the TCB fields for only input processing resulted in each TCB entry consisting of 264 bits. The register file used is a single-cycle large-signal memory design that relies on a domino scheme for data reads/writes. Reads and writes to two different locations in the RF can occur simultaneously in a single clock cycle. To reduce the routing and area cost, the circuits for reading and writing registers are implemented in a single-ended fashion. LBLs are segmented to reduce bitline capacitive loading and leakage, thus improving address decode time, read access time, as well as robustness.

C. Reorder Block

Two 54 bits $\times$ 32 entry CAMs (CAML and CAMR) are used to support dynamic reordering of out-of-order packets in the ROB. Again, the number of entries are limited by die area constraints imposed on the prototype chip. Each CAM entry includes sequence number, payload length, and CLB index for that connection. Each entry in CAML contains the first sequence number of the payload for an out-of-order packet as the tag part to be matched and the payload length as the data part. Similarly, each entry in CAMR contains the last+1 sequence number of the payload as the tag part and the payload length as the data part. Adding the CLB index for that connection as part of the tag enables sharing the CAMs for out-of-order packets

Fig. 12. Sparse-tree adder architecture.

from different connections. A ROB instruction issued by the instruction ROM causes the ROM to stall while the instruction is decoded and latched into the major clock domain. The ROM remains stalled till the results of the ROB operation are available, which is the next major clock cycle. The control logic in the instruction ROM de-asserts the stall signal appropriately. The CAMs perform single cycle lookup and write operations on the major clock. The output data must be latched into scratch registers in the execution core so that connection state can be updated through execution of instructions from the ROM.

D. Quaternary Tree Adder

The execution core uses a 32-bit sparse-tree adder [8] to perform high-speed add/subtract operations. The sparse-tree architecture divides the carry-merge tree into critical and noncritical sections, as shown in Fig. 12, with the intent of speeding up the critical path and moving a portion of the carry-merge logic to a noncritical path. Each adder core is composed of a critical sparse tree that generates 1 in 16 carries, with noncritical side paths generating conditional 1 in 4 carries and 4-bit conditional sums. Carry generated by the sparse-tree selects between the conditional carries to deliver 1 in 4 carries. These carries in turn select between the conditional sums to generate the final sum. The interstage wiring density, interconnect length, and generate/propagate fanouts all show significant reduction when compared to an equivalent Kogge-Stone adder.

E. Semidynamic Flip-Flops

To enable fast performance, the high-speed core uses implicit-pulsed semidynamic flip-flops [9] with small clock-to- Q delay and high skew tolerance. The flop has a dynamic master stage coupled to a pseudostatic slave stage (Fig. 13). As is shown in the schematic, the flip-flops are implicitly pulsed, with several advantages over nonpulsed designs. One main benefit is that they allow time-borrowing across cycle boundaries due to the fact that data can arrive coincident with, or

Fig. 13. Semidynamic flip-flop with selectable pulse width.

even after, the clock edge. Thus negative setup time can be taken advantage of in the logic. Another benefit of negative setup time is that the flip-flop becomes less sensitive to jitter on the clock when the data arrives after clock. They thus offer better clock-to-output delay and clock skew tolerance than conventional static master-slave flops. However, pulsed flip-flops have some important disadvantages. The worst case hold time of this flip-flop can exceed clock-to-output delay because of pulse width variations across process, voltage, and temperature conditions. A selectable pulse delay option is available, as shown in Fig. 13, to avoid failures due to pulsewidth variations and consequent min-delay failures. An external global signal allows selection between a narrow single-inverter delay pulse and a larger three-inverter delay pulse. However, no failures due to min-delay were observed in silicon even with the larger pulse option selected.

F. Clocking

The clock generation unit for the fast minor clock and its distribution is shown in Fig. 14. Clocking for the design includes two clock source options: an on-die phase-locked loop (PLL) and a secondary bypass clock source which uses an operational amplifier to convert external differential sinusoidal clock inputs to a single-ended clock. The single-phase clock output of the source selector is amplified and distributed to the high-speed

Fig. 14. Minor clock generation and distribution.

units through three stages of buffering. There are a total of five stages of clock buffering from the PLL to the clock inputs of the flip-flops in the core. All clock buffers are composed of two CMOS inverters to minimize variations and use local decoupling capacitors to minimize jitter. The entire clock distribution uses upper-level metals (M7/M6) with V_{cc}/V_{ss} shielding for noise isolation and for symmetric current return paths. The minor clock distribution network was simulated to have a maximum of 4.4 ps of total interunit skew.

G. Synchronization

Communication of data and control signals between the two frequency domains requires data synchronization. Special cells used for synchronization are shown in Fig. 15. On fast-to-slow synchronization, a high value on signal *fast-d* is held sticky until the signal *slow-o* goes high, which resets the sticky signal *stky*. This stickiness ensures that the data is transferred in this cycle or the next without being dropped. Only control bits that are active high are synchronized across domains, avoiding the need for valid bits. Similarly, for slow-to-fast synchronization, a rising edge on *slow-d* causes a fast output pulse on *fast-d*. A special “sync flop” is used to minimize metastability. The synchronization mechanism can cause a worst case delay of one slow clock cycle when a value is latched from the fast domain into the slow domain. Keeping the number of such synchronizations low minimizes this penalty. The specially designed sync flop was simulated to provide a mean time between failure (MTBF) approaching 10^4 years.

IV. DESIGN METHODOLOGY

The starting specification for this design was a high-level finite state machine model that performs TCP operations outlined in the RFC 793 specification. Subsequent steps involved development of a C model to implement the finite state machine, construction of the instruction set required to implement the operations in the C model using special-purpose hardware where necessary, generation of the instruction program to be executed, development of the RTL model, generation of schematics, and finally, layout. Validation by simulation was performed at each step using the C model, an instruction set simulator, RTL simulator, and schematic switch level simulator. In addition, equivalence verification was done between the RTL and schematics. The use of the instruction set simulator enabled

Fig. 15. Synchronization cells.

Fig. 16. Die microphotograph and chip characteristics.

bugs in the instruction program to be flushed out early, which in turn made the task of RTL validation easier.

Schematic and layout generation was accomplished using a combination of custom design and automatic synthesis. The high performance blocks such as the core and the instruction ROM were completely custom designed, as was the data path in the memory units. The blocks operating on major clock, such as input sequencer and send buffer were automatically synthesized and sent through an auto-place and route flow. This flow was also applied to the control logic in the memory units. Using such a combination of approaches helped us optimize design time without sacrificing chip performance.

V. RESULTS

This chip was fabricated in 90-nm CMOS communication technology [10]. A die micrograph with the functional blocks identified and a summary of chip characteristics is shown in Fig. 16. The 8-mm² design contains 460 K transistors, with the core containing 129 K (28%) of the total device count. The test chip is packaged on a flip chip ball grid array (BGA) substrate with 306 pins, out of which 129 are signal pads and 177 are power pads. The 35 × 35 mm square flip-chip BGA package includes an integrated heat spreader. The package also has a ten-layer stackup to meet the various power planes and signal

Fig. 17. Evaluation board with packaged part.

Fig. 18. Processing rate versus power supply.

Fig. 19. Chip power versus processing rate.

requirements. The evaluation board used to characterize the design is shown in Fig. 17.

A plot of packet processing rate (Gb/s) versus V_{CC} characterizing execution of the chip is shown in Fig. 18. Measurements at room temperature show that the design achieves a wire speed processing rate of 9.64 Gb/s at 1.72 V. This corresponds to a minor clock frequency of 4.82 GHz. Measured average power consumption of the chip as a function of processing rate is shown in Fig. 19. For this measurement, the power supply for the chip is varied from 0.9 to 1.72 V. At a processing rate of

4.4 Gb/s and 0.9 V, the design dissipates 730 mW. The power consumption of the chip increases to 6.39 W at 1.72 V and at 9.64-Gb/s processing rate.

VI. CONCLUSION

This paper has presented the design of a programmable special-purpose hardware engine that is capable of wire speed TCP inbound processing for a saturated 10-Gb/s Ethernet link with minimum packet sizes. It is a dual-frequency buffer-free design with a high-speed execution core. The core performance can be modulated in accordance with processing requirements. Specialized instructions targeted at TCP processing are implemented that significantly reduce processing time per packet. The chip also implements a new algorithm for dynamically re-ordering packets in hardware.

The results show that the computing performance provided by a special-purpose engine with a simple but high-performance core is equivalent to a state-of-the-art general-purpose processor running at the same frequency, with significant savings in die area and power. Such an engine would form the centerpiece of a comprehensive system level solution for TCP offload.

ACKNOWLEDGMENT

The authors would like to thank J. Rattner, V. De, S. Borkar, and M. Haycock for their guidance, and K. Ikeda, K. Truong, H. Nguyen, and C. Parsons for help with layout, and the LTD team for the PLL design.

REFERENCES

- [1] A. Foong, T. Huff, H. Hum, J. Patwardhan, and G. Regnier, "TCP performance re-visited," in *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software*, Mar. 2003, pp. 70–79.
- [2] J. Chase, A. Gallatin, and K. Yocum, "End system optimizations for high-speed TCP," *IEEE Commun. Mag.*, vol. 39, pp. 68–74, Apr. 2001.
- [3] J. Kay and J. Pasquale, "Profiling and reducing processing overheads in TCP/IP networking," *IEEE/ACM Trans. Networking*, vol. 4, pp. 817–828, Dec. 1996.
- [4] M. Allman and A. Falk, "On the effective evaluation of TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 59–70, Oct. 1999.
- [5] *Transmission Control Protocol NIC-RFC 793, DDN Protocol Handbook*, vol. 2, Information Sciences Inst., Univ. Southern California, Los Angeles, Sept. 1981, pp. 2179–2198.
- [6] D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An analysis of TCP processing overhead," *IEEE Commun. Mag.*, vol. 27, pp. 23–29, June 1989.
- [7] V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, pp. 277–292, June 1999.
- [8] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, "A 4 GHz 130 nm address generation unit with a 32-bit sparse-tree adder core," in *Proc. VLSI Circuits Symp.*, 2002, pp. 126–127.
- [9] J. Tschanz, S. Narendra, Z. Chen, S. Borkar, M. Sachdev, and V. De, "Comparative delay and energy of single edge-triggered and dual edge-triggered pulsed flip-flops for high-performance microprocessors," in *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)*, Aug. 2001, pp. 147–151.
- [10] K. Kuhn, M. Agostinelli, S. Ahmed, S. Chambers, S. Cea, S. Christensen, P. Fischer, J. Gong, C. Kardas, T. Letson, L. Henning, A. Murthy, H. Muthali, B. Obradovic, P. Packan, S. Pae, I. Post, S. Putna, K. Raol, A. Roskowski, R. Soman, T. Thomas, P. Vandervoorn, M. Weiss, and I. Young, "A 90 nm communication technology featuring SiGe HBT transistors, RF CMOS," in *IEDM Tech. Dig.*, 2002, pp. 73–76.

Yatin Hoskote (M'96) received the B.Tech. degree in electrical engineering from IIT Bombay, India, and the M.S. and Ph.D. degrees in computer engineering from the University of Texas at Austin.

He joined Intel Corporation, Hillsboro, OR, in 1995 as a Member of the Strategic CAD Laboratories doing research in verification technologies. He is currently a Member of the Advanced Prototype Design Team, Intel Laboratories. He has authored three journal papers and over 10 conference papers. He has 11 patents pending in the fields of verification,

floating point computation, and processing network protocols.

Dr. Hoskote received a Best Paper Award from the 1999 Design Automation Conference. He is a member of the program committee for the High Level Design Validation and Test Workshop.

Bradley A. Bloechel (M'95–A'96) received the A.A.S. degree in electronic engineering technology from Portland Community College, Portland, OR, in 1986.

He joined Intel Corporation, Hillsboro, OR, in 1987 as a Graphics Design Technician for the iWarp project supporting the RFU and ILU design effort. In 1991, he transferred to Supercomputer Systems Division Component Technology, where he supported VLSI test/validation effort and extensive fixturing support for accurate high-speed test and

measurement of the interconnect component used in the Tera ops computer project (Intel, DOE, and Sandia). In 1995, he joined the Circuits Research Laboratory, Microcomputer Research Laboratory, where he is currently a Senior Lab Technician specializing in on-chip dc and high-speed I/O measurements and characterization.

Mr. Bloechel is a member of Phi Theta Kappa.

Gregory E. Dermer received the B.S. degree in electrical engineering from the Indiana Institute of Technology, Fort Wayne, in 1977 and the M.S. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 1983.

From 1979 to 1992, he held a variety of processor architecture, logic design, and physical design positions at Cray Research, Inc., Nicolet Instrument Company, Astronautics Corporation of America, and Tandem Computers, Inc. In 1992, he joined Intel Corporation's Supercomputer Systems Division,

where he worked on clock system design and reliability modeling for the Intel ASCI Red supercomputer. For the past six years, he has worked in the circuits research area of Intel Laboratories, Hillsboro, OR, on physical design and measurements for high-speed interconnections.

Vasantha Erraguntla (M'03) received the B.S. degree in electrical engineering from Osmania University, India, and the M.S. degree in computer engineering from the University of Southwestern Louisiana, Lafayette.

She joined Intel Corporation, Hillsboro, OR, in 1991 and worked on the high-speed router technology for the Teraflop machine. She then joined the Design Technology team, validating performance verification tools for high-speed designs. For the last six years, she has been engaged in a variety

of advanced prototype design activities at Intel Laboratories, implementing and validating research ideas in the areas of high-performance and low-power circuits and high-speed signaling. She has coauthored seven papers and has four patents pending in this area.

David Finan received the A.S. degree in electronic engineering technology from Portland Community College, Portland, OR, in 1989.

He joined Intel Corporation, Hillsboro, OR, working on the iWarp project in 1988. He started working on the Intel i486DX2 project in 1990, the Intel Teraflop project in 1993, for Intel Design Laboratories in 1995, for Intel Advanced Methodology and Engineering 1996, on the Intel Timna project in 1998, and for Intel Laboratories in 2000.

Jason Howard received the M.S.E.E. degree from Brigham Young University, Salt Lake City, UT, in 2000.

He was an Intern with Intel Corporation during the summers of 1998 and 1999 working on the Pentium 4 microprocessor. In 2000, he formally joined Intel, working in the Oregon Rotation Engineers Program. After two successful rotations through NCG and CRL, he officially joined Intel Laboratories, Hillsboro, OR, in 2001. He is currently working for the Prototype Design team.

Dan Klowden received the B.S. degree in computer engineering from the University of Washington, Seattle, in 2000. He is currently working towards the M.S.E.E. degree at Oregon Health and Sciences University, Portland.

He joined Intel Corporation, Hillsboro, OR, as part of the Rotation Engineer Program and worked on a variety of projects in low-power circuit research and Linux device drivers for Internet Appliance products. He is currently with the Intel Laboratories prototype team, working on both circuit design and design au-

tomation activities.

Siva G. Narendra (M'99) received the B.E. degree from the Government College of Technology, Coimbatore, India, in 1992, the M.S. degree from Syracuse University, Syracuse, NY, in 1994, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 2002.

He has been with Intel Laboratories, Hillsboro, OR, since 1997, where his research areas include low-voltage MOS analog and digital circuits and impact of MOS parameter variation on circuit design. He has authored or coauthored over 16

papers and has 15 issued and 27 pending patents in these areas. He is an Adjunct Faculty with the Department of Electrical and Computer Engineering, Oregon State University, Corvallis.

Dr. Narendra is an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS and a Member of the Technical Program Committee of the 2002 International Symposium on Low Power Electronics and Design.

Greg Ruhl received the B.S. degree in computer engineering and the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 1998 and 1999, respectively.

He joined Intel Corporation, Hillsboro, OR, in 1999 as a part of the Rotation Engineering Program where he worked in ESG (Infiniband I/O switch), CRL (circuit research), and NCG (gigabit Ethernet validation). After completing the REP program, he joined Intel Laboratories Prototype Design team where he currently works on implementing and

validating research in the areas of low-power high-performance circuits and high-speed signaling.

James W. Tschanz (M'99) received the B.S. degree in computer engineering in 1997 and the M.S. degree in electrical engineering in 1999, both from the University of Illinois at Urbana-Champaign.

Since 1999, he has been a Circuits Researcher with Intel Laboratories, Hillsboro, OR. His research interests include low-power digital circuits, design techniques, and methods for tolerating parameter variations. He is an Adjunct Faculty Member with the Oregon Graduate Institute, Beaverton, and has authored several papers and patents pending.

Howard Wilson was born in Chicago, IL, in 1957. He received the B.S. degree in electrical engineering from Southern Illinois University, Carbondale, in 1979.

From 1979 to 1984, he was with Rockwell-Collins, Cedar Rapids, IA, where he designed navigation equipment and electronic flight display systems. From 1984 to 1991, he was with National Semiconductor, Santa Clara, CA, designing telecom components for ISDN. With Intel Corporation since 1992, he is currently a Member of Intel Laboratories,

Hillsboro, OR, engaged in a variety of advanced prototype design activities.

Sriram Vangal (S'90–M'98) received the B.S. degree from Bangalore University, India, in 1993, and the M.S. degree from the University of Nebraska, Lincoln, in 1995, both in electrical engineering.

With Intel since 1995, he is currently a Member of Intel Laboratories, Hillsboro, OR. His research interests are in the area of low-power high-performance circuits and advanced prototyping. He has authored or coauthored five papers and has six issued and 15 patents pending in these areas.

Jianping (Jane) Xu received the M.S.E.E. degree and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1995 and 1998, respectively.

She joined Intel Corporation in 1999 as a Research Scientist in emerging platform research and later in circuit research at Intel Laboratories, Hillsboro, OR. Her research interests include high-speed Internet networking infrastructure and acceleration chips, electrical and optical interconnect system and circuits, power delivery, and acoustic device design

and development.

Venkat Veeramachaneni (M'02) received the B.E. degree in electrical engineering and the M.S. degree in physics from the Birla Institute of Technology and Science, Pilani, India, in 1997 and the M.S. degree in electrical engineering from the University of Virginia, Charlottesville, in 1999.

He has been with Intel Laboratories, Hillsboro, OR, since 1999, where his work includes design of prototypes in the areas of low-power high-performance circuits and high-speed signaling. He has authored or coauthored three papers and has two

patents pending in these areas.

Nitin Borkar received the M.Sc. degree in physics from the University of Bombay, Bombay, India, in 1982 and the M.S.E.E. degree from Louisiana State University, Baton Rouge, in 1985.

He joined Intel Corporation in 1986, where he worked on the design of the i960 family of embedded microcontrollers. In 1990, he joined the i486DX2 Microprocessor Design Team and led the design and the performance verification programs. After successful completion of the i486DX2 development, he worked on high-speed router technology for

the Teraflop machine. He now leads the Prototype Design Team at Intel Laboratories, Hillsboro, OR, implementing and validating research ideas in the areas of high-performance low-power circuits and high-speed signaling.