Whitepaper

Fine-Grained Hardware Multi-Threading
A CPU Architecture for High-Touch Packet Processing

Bob Gelinas, Pat Hays, Sol Katzman
Lexra, Inc.
Waltham, MA 02453

1. Overview

This whitepaper describes Lexra's fine-grained Hardware Multi-Threading (HMT) architecture and its critical performance advantages for High Touch Internet packet processing. High Touch applications create unprecedented performance demands on general-purpose programmable CPUs. At the same time, the large data sets and absence of locality characteristic of these applications cause conventional CPUs to squander much of their processing power in stalls while waiting for main memory reads. HMT exploits the fundamental packet flow parallelism in High Touch applications by processing multiple application threads simultaneously. As a result, cache miss stalls from one thread can be hidden under useful work by another thread. Lexra's first HMT implementation is in the LX4580 CPU. The LX4580, designed to support Lexra's upcoming family of network communications ICs, is the highest performance MIPS32™ CPU. It achieves 500 MHz in 0.13 μm technology in a synthesizable Verilog design with a small silicon footprint. For High Touch applications, the LX4580 with HMT achieves over three times the effective performance of a single-threaded CPU.

The next section discusses the computational demands of the new High Touch applications. Section 3 describes multi-threading, in its various styles, as a natural fit to the problem posed by High Touch applications. Lexra's HMT architecture and its benefits are described in Section 4. In Section 5 these benefits are quantified. First, the LX4580 performance with HMT is compared to a single-threaded CPU. Second, the LX4580 performance is compared to other CPU architectures. Section 6 describes how software developers can use HMT with minimal effort. The summary is in Section 7.

2. Introduction

The search for the legendary "killer ap" has a new find: *High Touch* Internet packet processing is the class of applications most likely to drive new CPU innovations and sales. High Touch packet processing is characterized by the application of complex

and dynamic policies to Internet connections or packet flows. These higher layer applications thrive in the security and storage infrastructure of enterprise networks.

A common characteristic of High Touch applications is that *many* bytes of the packet, not just the headers, must be processed. In today's gigabit enterprise networks, this generates computational demand well beyond conventional CPUs developed for the desktop or workstation market. For example, intrusion detection requires ~10,000 instructions per packet; at gigabit data rates, billions of instructions per second are required. At future 10 gigabit data rates, the processing requirement grows to tens of billions of instructions per second.

In the past much of the computation was cast into hardware to minimize the demand on the programmable processor. That worked for simple applications like IPv4 forwarding but fails for High Touch because these applications are exceedingly complex and continually evolving. In other words, High Touch applications require general-purpose programmable processing at unprecedented performance levels.

All architects have attempted to deliver this high processing power by taking advantage of the inherent parallelism in packet or packet flow processing to integrate anywhere from two (2) to thirty-two (32) CPUs on one chip. This strategy, called Chip Multi-Processing (CMP), is alone insufficient because most of the time the CPUs are idle waiting for memory accesses, wasting area and power.

Figure 1 shows that over time, the gap between CPU performance and DRAM performance has grown wider and wider [Hennessy 1996, p. 374]. This trend will continue well into the future because DRAM technology is driven by cost, not performance. High touch applications operate on multi-gigabyte data sets that must be stored in off-chip DRAM. As a result of the performance gap between CPU and DRAM, increasing the CPU clock speed offers diminishing benefit. For example, with 5% level 1 cache miss rate, and 80ns mean level 1 service time from off-chip DRAM, increasing the CPU clock speed from 500MHz to 1GHz gives only 20% benefit to performance, at significant cost to area and power.

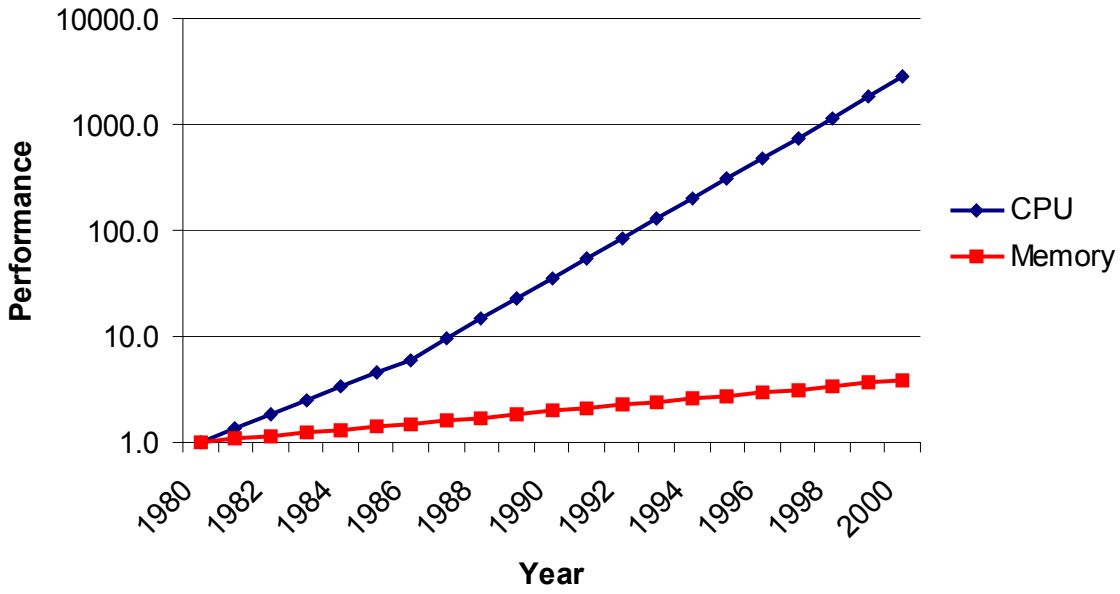

Figure 1. The Memory Bottleneck

The classic remedy is memory hierarchy, introducing level 2 and even level 3 (off-chip SRAM) caches. This works to the degree that the programs exhibit locality of reference. In network processing, there is almost no temporal locality; however, there is moderate spatial locality. In other words, when a cache line is accessed, other bytes in the line are also likely to be accessed but are unlikely to be accessed again. Moderate size level 2 caches are somewhat helpful, but increasing the level 2 size only wastes area and power.

Emerging High Touch applications demand innovative CPU architectures to cost effectively deliver the required processing power. Architects are turning to *multi-threading* as the most promising technology to achieve this goal.

3. Multi-Threading

A *thread* is a program segment along with state information. Multi-threading is an architectural technique that allows task switching between threads when a long latency event, like a cache miss, occurs. It is successful in hiding cache miss penalties under useful work whenever another thread is ready for execution. The biggest benefits of multi-threading occur when the threads are from the same program. In High Touch packet processing, many independent packet flows are available for processing, so multi-threaded architectures are a natural fit.

As was commonly true of CPU innovations, the first realizations were in research computers or supercomputers. Indeed, multi-threading dates from the 1964 CDC6600, among the earliest supercomputers [Thorton 1971]. The Tera supercomputer later used many parallel threads with fast context switching [Alverson 1991]. By the early 1980s, the increasing cost of communication between processors and off-chip memories was widely recognized [J. Smith 1982]. In an important paper, Burton Smith explained the benefits of multi-threading which he incorporated into HEP, a large scale scientific computer [B. Smith 1981]. Advancing technology has made multi-threading not only feasible but, in some cases, required for new monolithic CPUs.

There are two forms of multi-threading: *coarse-grained* and *fine-grained*. In coarse-grained multi-threading, a single thread consumes all of the CPU cycles until a context switch occurs. In fine-grained multi-threading, execution rotates cycle-by-cycle among different threads. Particularly in the fine-grained case, it is essential that the context switch itself occur at no cost to performance.

One of the first uses of coarse-grained multi-threading in a general purpose CPU chip is the dual-threaded IBM Power5 architecture [Borkenhagen 2000]. Intel, recognizing the value of multi-threading for on-line transaction processing and web services, has introduced a dual-threaded fine-grained multi-threaded architecture into the Pentium®III Xeon™ server CPUs [Intel Developer's Forum, August 28, 2001].

Even earlier, multi-threading became a dominant architectural theme for communications-centric processors. The Prism CPU in the Vitesse IQ2xxx product line is coarse-grained multi-threaded with five (5) contexts [Microprocessor Report, May 29, 2000]. The application switches from the current context to one of the other ready-to-run contexts automatically on cache miss. One of the contexts is reserved for kernel code. Similarly, the micro-engines deployed in Intel's IXP product line are coarse-grained multi-threaded with four (4) contexts per engine [Microprocessor Report, Sept. 13, 1999]. The IXP micro-engines perform a context switch in response to special assembly instructions. Instruction issue is fixed round-robin so CPU performance degrades incrementally for each thread executing a long latency load. By contrast with the coarse-grained multi-threading of Vitesse and Intel, Ubicom has incorporated fine-grained multi-threading into its proprietary 8-bit IP2022 network processor [Microprocessor Report, May 28, 2002]. The intent of its 8-threaded CPU is to deterministically allocate real-time to each thread.

The next section of this whitepaper describes Lexra's fine-grained hardware multi-threaded architecture. Lexra's instruction issue rules are novel and provide important performance benefits compared to other multi-threading architectures.

Before leaving this review section it is worth noting that a sophisticated form of multi-threading, called *simultaneous multi-threading (SMT),* permits two or more independent threads to issue in one cycle [Tullsen 1995]. By issuing instructions from independent threads SMT can overcome the data dependencies that limit the performance of superscalar data paths. Clearwater Networks, before its demise, developed an SMT

processor with eight datapaths called CNP810SP [EETimes, Oct 9, 2000]. SMT architectures, although promising for the future, are costly in both architectural complexity and silicon area.

4. Lexra's Fine-Grained Hardware Multi-Threading (HMT)

4.1 Introduction

Lexra's new LX4580 CPU implements a sophisticated fine-grained Hardware Multi-Threading (HMT) architecture. In general, CPU throughput is a function of (i) system clock speed, (ii) pipeline utilization during normal execution and (iii) the number of stall cycles incurred by cache miss. Lexra's HMT benefits all three parameters to maximize LX4580 throughput. First, HMT allows critical bypass paths to be eliminated, allowing higher clock speed. Second, HMT eliminates all branch and jump stalls and minimizes load interlock delay stalls for peak pipeline utilization during normal execution. Finally, cache miss processing is handled in the background, eliminating the major source of stalls for typical operating conditions.

4.2 LX4580 Pipeline

The LX4580 7-stage pipeline is illustrated in Figure 2. This pipeline uses exclusively positive-edge clocking. It also allocates the maximum address-register-to-data-register clock cycle for I-Cache and D-Cache reads and isolates the caches from other internal logic. It is expected that this pipeline will readily scale into deeper sub-micron technologies. In applying this pipeline to a single threaded MIPS32™ implementation the following speed-critical bypass paths arise: (i) the bypass of ALU results from the execution unit through several levels of multiplexors to the ALU input registers, the so-called E-E bypass path. This bypass path is particularly difficult in super-scalar implementations where multiple ALU results must cross data paths. (ii) the bypass of load data after alignment shifting, bus transfer and multiplexing, to the input registers of the ALU, the so-called M-E forwarding path.

MIPS32 has a single exposed branch delay slot; as a result, incorrectly predicted branches, or data dependent jumps, incur a two-cycle penalty. MIPS32, unlike the earlier MIPS-I, does not expose a load delay slot. Therefore, in the most critical case where the load register is used in the next instruction, the single-threaded 7-stage pipeline also incurs a two-cycle penalty.

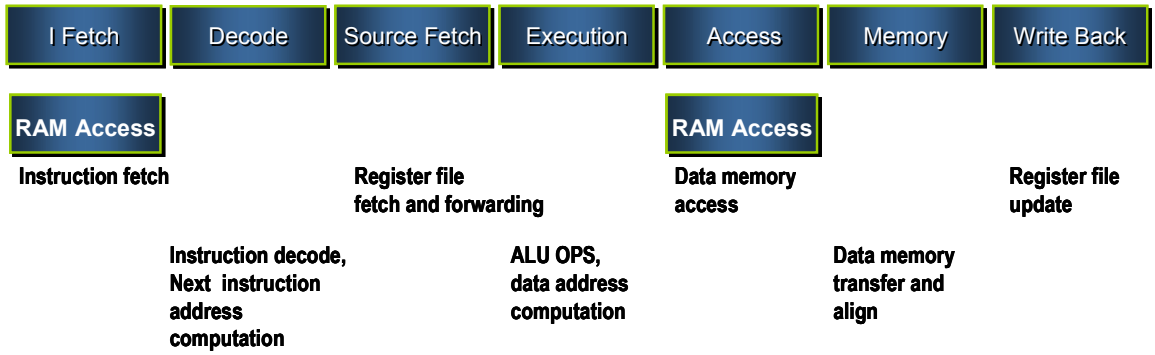

Figure 2. The LX4580 7-Stage Pipeline

4.3 LX4580 HMT Hardware

The LX4580 HMT architecture permits four active threads. Accordingly, the CPU includes four copies of most user-visible state registers. Registers which are replicated for each of the four threads are called "four-high". The LX4580 includes four copies of the general register file, the PC and the HI and LO multiply/divide target registers. Most, but not all, of the Coprocessor 0 control registers are replicated as well. Debug is "single-threaded"; after one context enters the debug exception handler, any other context that detects a debug exception will be inhibited. However, other "multi-threaded" exceptions may be executed by more than one context simultaneously.

The TLB is four-high because the threads are typically executing on independent sets of memory pages. The caches, however, are not four-high. The caches are 4-way set associative. This ensures that even if all four contexts experience a miss in consecutive cycles to the same cache miss index, there will be no thrashing. On the other hand, the replacement algorithm is a simple circular sequential ordering. That is, there is no particular connection between context number and way number. This allows more effective utilization of the cache as well as ease of sharing data. For example, one context may have data in several ways of the D-Cache at a particular set index, while the other contexts have no active data at that set index. Cache misses to the same line from different contexts are prevented, by hardware, from putting the same line into different ways of the cache.

The overall silicon area cost for adding HMT to the LX4580 CPU with 64 KB I-Cache and 16 KB D-Cache is 1.3X. It is shown in Section 5 that the performance benefit is significantly greater.

4.4 HMT Instruction Issue Rules

In the absence of cache misses, the four contexts issue in alternate pipe flows. (Figure 3). In the case of a cache miss in context X, X gives up its pipeline flows to other contexts that can take them until the cache miss is resolved. Any context can take the pipeline flows if it is not itself waiting for a cache miss, as long as it obeys the following rule:

CONTEXT EVEN-ISSUE RULE
If a context issues in flow N, it can not issue in flow N+1 nor in flow N+3.

This rule allows the complete elimination of the critical bypass paths including the E-E and M-E bypass paths described in Section 4.2. It is important to emphasize that *HMT requires no new MIPS32 ISA extensions*. Accordingly, no new operating system or code development tools are required.

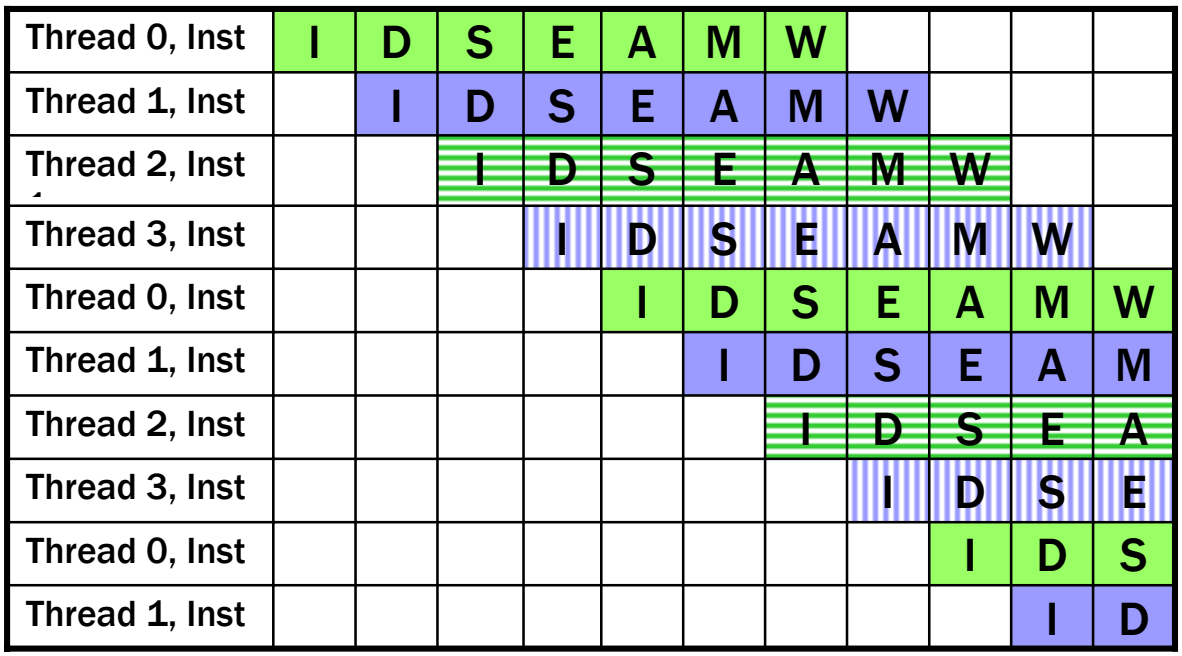

Figure 3. HMT Four Context Instruction Issue

The even-issue rule may be compared with two alternatives: (a) the simpler strict round-robin issue, (b) dynamic pipe flow allocation with forwarding paths in place (no even-issue rule). The probability of one or more misses outstanding depends on the system design and application code. However, it is obvious that in all cases the even issue rule is superior to (a). Option (b) differs from even-issue in allowing 100% CPU utilization with three outstanding misses; however, since three outstanding misses is rare, even-issue delivers higher performance because the missing critical paths permit higher clock speed in the even issue CPU. In Lexra's LX4580 design the clock speed benefit of the even-issue rule is about 20%. Furthermore, the even-issue design is less routing-dependent, so its performance scales better than (b) into deeper sub-micron technology.

It is worth noting that the even-issue rule could potentially be enhanced by permitting two instructions from the same context that do not depend on the missing bypass paths to issue in successive cycles. Lexra's analysis indicated that the complexity of the dependency checking needed for this feature extended key critical paths, offsetting the improved pipeline utilization.

4.5 HMT Benefits for Pipeline Utilization

HMT improves pipeline utilization by eliminating pipeline bubbles that typically occur for mis-predicted branches, data dependent jumps and load interlock delay.

Figure 4 illustrates a MIPS32 branch instruction based on the comparison of two registers. The delay slot is exposed. The figure illustrates the worst-case timing condition in which instructions from two threads are issuing in alternate cycles. Even so, the branch flag from instruction 1 is available in time for instruction 3 to correctly select the next address *without prediction*. Similarly, if instruction 1 is a data dependent jump, the jump destination is available in the E-stage of instruction 1, in time for instruction 3 after the delay slot. Decision-making is a particularly high proportion of packet processing code and high-performance pipelines have branch prediction penalties of at least two cycles. Conventional CPUs resort to complex branch prediction schemes to mitigate stalls however, the forward branches in packet processing are rarely seen a second time, so these schemes add significant area without improving prediction success over about 70%.

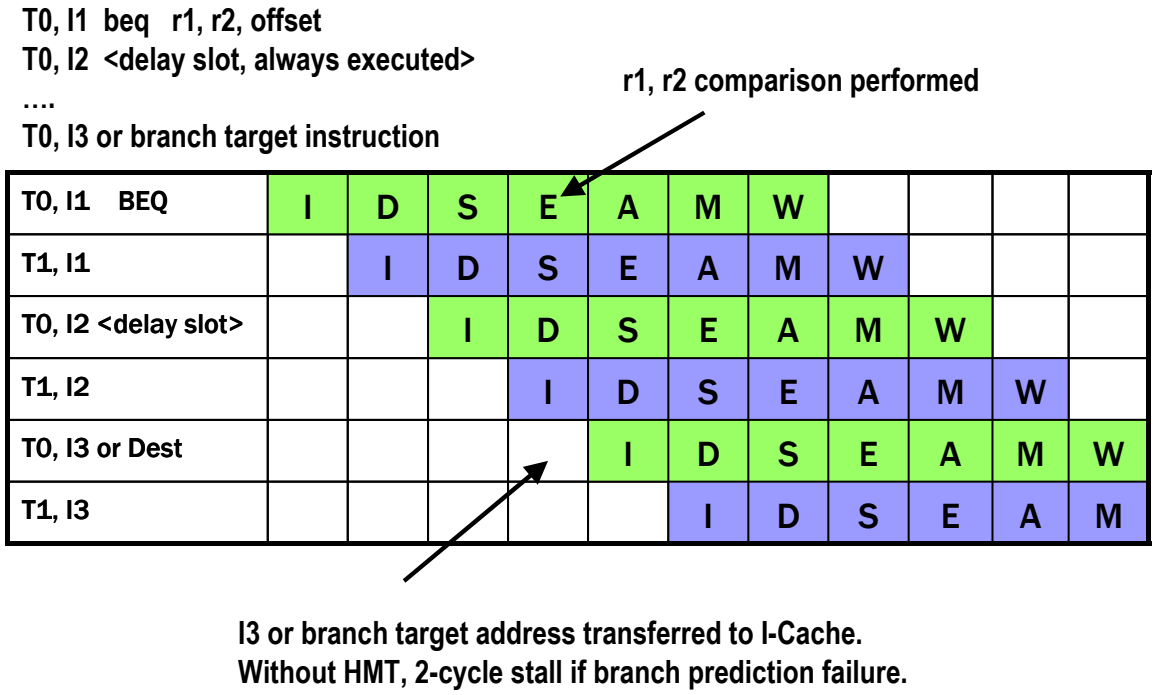

Figure 4. HMT Eliminates Branch Prediction

Figure 5 illustrates a MIPS32 load instruction. In the case illustrated, the load is used in the second instruction and no load to use stalls are required. Typically, compilers are able to insert an instruction between the load and its use so that no stalls are required even in the critical case where Thread 0 gets alternate pipeline flows; if not, one pipeline

flow must be killed. By contrast, a single-threaded version of the LX4580 would always require two stall cycles for the illustrated sequence.

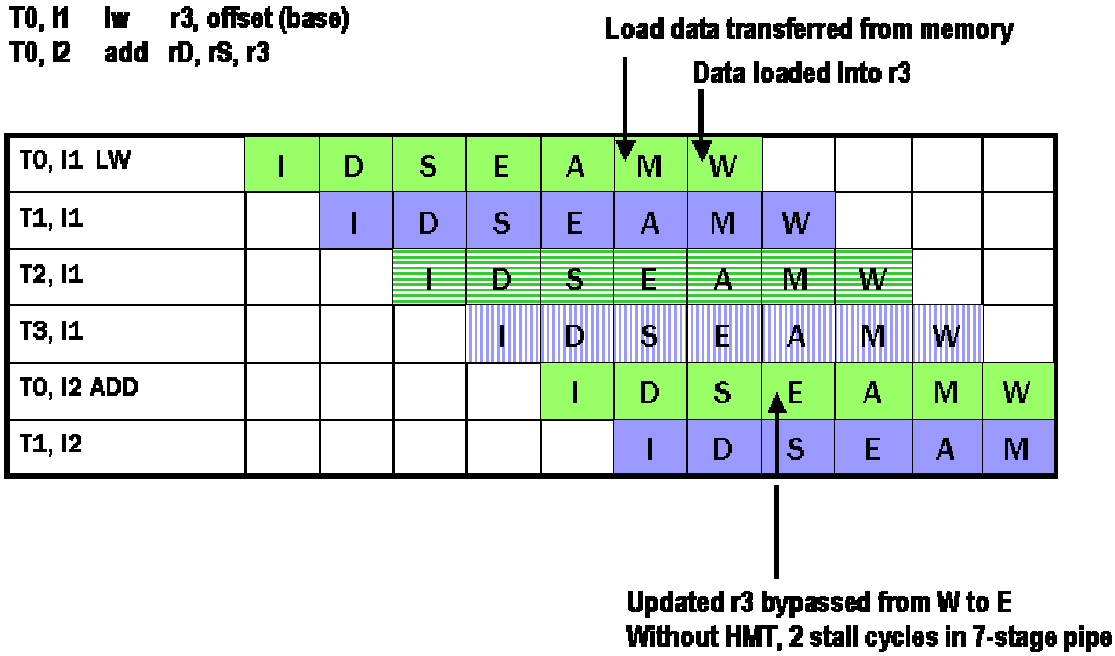

Figure 5. HMT Minimizes Load-Use Delay

4.6 Illustration of HMT Cache Miss with the Even-Issue Rule

Figure 6 illustrates how HMT re-allocates pipeline flows following cache miss. In the example, Thread 0 Instruction 1 suffers a cache miss. The miss is detected in the M-Stage, too late for Thread 0 Instruction 2 which has been fetched, but must be killed. The next pipeline flow that would have been allocated to Thread 0 is allocated to Thread 2 (second to last row in Figure 6). Similarly, after one Thread 1 instruction is killed, its issue slots are allocated to Thread 3 (last row in Figure 6).

T0, T1 suffer cache miss
T2, T3 get T0, T1 issue slots

The CPU is fully utilized with only 2 active threads while T0, T1 await cache fill

T0, I1 LW (MISS)	I	D	S	E	A	M	(Dcache miss)			
T1, I1 LW (MISS)		I	D	S	E	A	M	(Dcache miss)		
T2, I1			I	D	S	E	A	M	W	
T3, I1				I	D	S	E	A	M	W
T0, I2 (KILLED)					I	D	(killed)			
T1, I2 (KILLED)						I	D	(killed)		
T2, I2							I	D	S	E	A
T3, I2								I	D	S	E
T2, I3									I	D	S
T3, I3										I	D

Figure 6. HMT Minimizes Cache Miss Penalty

As a consequence of the even-issue rule, there is one pathological case that can arise when two threads suffer cache miss. In this case, the two active threads "get off on the wrong foot" and pipeline utilization is degraded by 50%. Fortunately, it is straightforward to detect this special case and resolve it with one additional issue rule. It can be shown that no other issue rules are needed to ensure efficient pipeline utilization.

5. Comparison with Conventional CPUs

The previous section described Lexra's HMT architecture and its benefits. This section will attempt to quantify those benefits. First, the LX4580 performance is compared with and without HMT. The LX4580 performance with HMT was simulated by Lexra. Simulation is required to model the benefits of the sophisticated instruction issue rules in the multi-threaded architecture. For a single-threaded architecture, performance is a straightforward function of clock frequency, efficiency of the instruction set and stall rate. The problem is that in comparing the LX4580 to third party CPUs the necessary CPU parameters will rarely be available.

A reasonable approximation for the actual performance of a single-threaded CPU is:

Performance (MIPS) = DMIPS / (1 + L1*S)

where,

DMIPS = Dhrystone 2.1 DMIPS without in-lining

L1 = Level 1 cache miss rate

S = mean service time for L1 miss

No single benchmark is ideal but Dhrystone 2.1 has the major benefit that it is publicly available for comparison. Dhrystone will roughly account for differences in instruction set, superscalar vs. scalar issue, branch prediction stalls and 64-bit vs. 32-bit data path width. The most important drawback is that because Dhrystone fits completely into modest size level 1 caches, after the first loop's compulsory misses, Dhrystone doesn't suffer cache misses. The cache miss rate is, of course, a critical parameter for realistic CPU performance, especially for packet processing. In using Dhrystone to compare CPUs it is also important to know whether in-lining was used; other effects resulting from compiler differences are less important.

The LX4580 DMIPS is 700 at 500 MHz (simulated). The effective MIPS degrade with increasing level 1 miss rate (L1) and miss service time (S) as expected. The LX4580 with HMT was simulated for many data points with L1 in the range 3 – 7% and S in the range 50 – 250 ns. This range is sufficient to model a wide variety of packet processing software as well as a wide range of system architectures. S = 50 ns requires a tightly coupled on-chip level 2 cache with very high hit rate. On the other extreme, S = 250 ns is consistent with direct access to off-chip SDRAM in the absence of level 2 cache. A comparable single-threaded LX4580 was modeled using the formula above after de-rating the peak DMIPs by 20%. This de-rating roughly accounts for the benefit of HMT to processor clock as well as to branch, jump and load stall reduction.

Figure 7 demonstrates that over a wide range of operating conditions, *the HMT benefit is at least 2X and in some cases over 3X as compared to a single-threaded version of the same MIPS32 processor.*

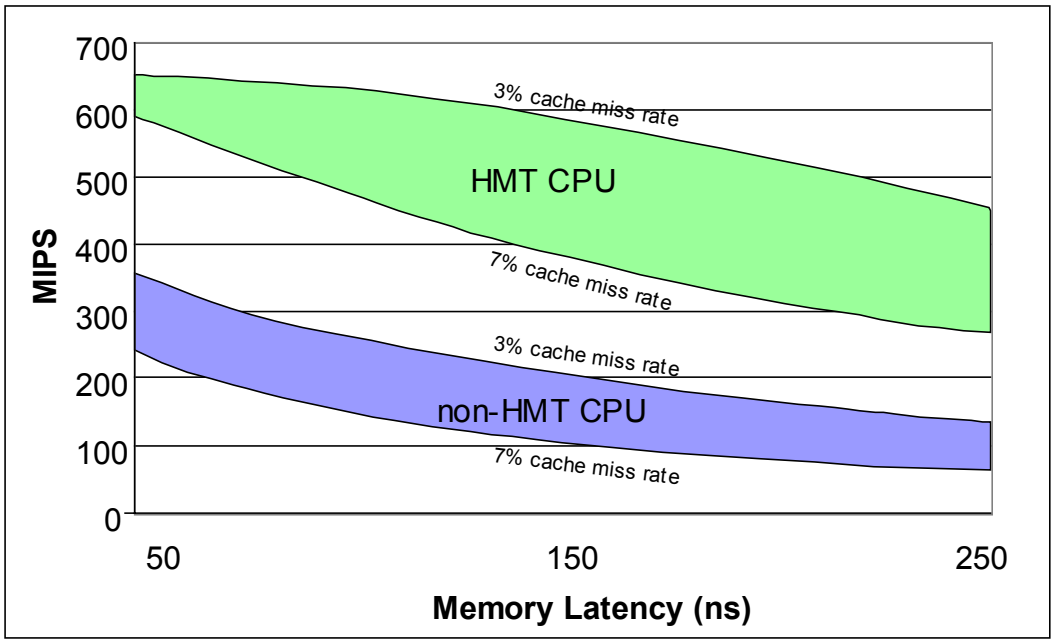

Figure 7. Comparison of LX4580 Performance with and without HMT

The LX4580 has also been compared to single-threaded CPUs from other vendors. In this case, other differences between the architectures will also influence the comparison. CPUs have used a number of strategies to reduce the level 1 miss rate and miss service time, most notably, prefetch (hardware or software controlled), non-blocking caches and early restart with critical word first. These features are not deployed in the LX4580. However, other CPUs have achieved a portion of the HMT performance benefit with these features.

Prefetch can reduce level 1 miss rates by 20–40% depending on the implementation and application [Hennessy 1996, pp. 400-404]. Non-blocking caches have limited value in reducing stalls for the SPECint92 tasks (20–30%) as compared to certain SPECflp92 tasks [Farkas 1994]. In the latter case, algorithms are executed on arrays of data stored in main memory. Non-blocking caches are very effective in eliminating stalls when loops are unrolled. Packet processing code has no such loops and therefore experiences only modest benefit from non-blocking caches. Finally, for an LX4580 CPU without HMT, stalls will be reduced by about 10% with critical word first and early restart caches.

Considering these factors, in comparing the LX4580 to other CPUs the L1*S product of other CPUs was reduced by a somewhat arbitrary 40% to account for the benefits of prefetch, non-blocking cache and early restart, absent from the LX4580:

Competitive single-threaded (L1*S) = 0.60 * LX4580 (L1*S)

For example, if S is fixed, the LX4580 with level 1 miss rate of 5% will be compared to the competitor's single-threaded CPU with a level 1 miss rate of 3%. The 0.6 factor is intended to generously account for the benefit of alternative strategies for reducing processor stall from cache miss.

Figure 8 compares the MIPS32 LX4580 to the most advanced 32-bit embedded CPU cores available from ARM and IBM. The new ARM1026EJ-S is a single issue pipeline implemented in synthesizable RTL and therefore a very direct comparison to a single-threaded LX4580. Despite the somewhat larger LX4580 area resulting from HMT, not only the performance but the bottom-line MIPS/mm^2 is significantly greater. On the other hand, with the latest PowerPC core, IBM has invested silicon area in a more powerful superscalar architecture. The 440 is not synthesizable and is implemented in SA-27E a fast 0.15 µm process with L_{eff} of 0.11 µm. Nevertheless, the LX4580 also significantly exceeds the IBM MIPS/mm^2. Both the ARM and IBM CPU effective performance was calculated by the formula DMIPS/(1 + 0.6 * L1 * S).

COMPANY	**ARM**	**Lexra**	**IBM**
CPU	ARM1026EJ-S	LX4580	PowerPC 440
INSTRUCTION SET	ARMv5TEJ	MIPS32 Release 2	PowerPC Book E
IMPLEMENTATION	Synthesizable	Synthesizable	Hard macro
TECHNOLOGY	0.13 µm	0.13 µm	0.15 µm SA-27E
CPU CLOCK (w.c.)	325 MHz	500 MHz	400 MHz
PIPELINE	6-Stage	7-Stage	7-Stage, 2-Issue, Out-of-order exec.
DMIPS	400	700	720
EFFECTIVE MIPS [@ 5% Level 1 miss, 100ns miss service time]	207	560	327
AREA (16KB I-Cache, 16 KB D-Cache	4.6 mm^2	5.5 mm^2	5.5 mm^2
MIPS/mm2	**45**	**102**	**59**

Figure 8. LX4580 Compared to ARM and IBM 32-bit CPUs.

6. Software Model for the LX4580 with HMT

For software developers an LX4580 looks like four virtual CPUs or hardware threads running in parallel. Because each hardware thread is a full MIPS32 compliant CPU with its own set of registers, TLB, and other resources, developers can use commercially available tools to develop application software.

The four hardware threads of a LX4580 constitute a Symmetrical Multi-Processing (SMP) core capable of running Linux SMP and other real time kernels. Each hardware thread can equally execute the Linux kernel, process interrupts, access memory and other resources, or execute user applications. There are two fundamental

programming models that are used on multi-processors systems: *share-everything* and *share-something*. Both models allow programs to communicate with loads and stores to and from memory. The difference between the two models comes from the fact that share-everything places all the data in shared memory while share-something places all the data by default in private memory unless the user explicitly requests that data be placed in shared memory. The Linux kernel follows the share-everything model but most user applications follow the share-something model. The LX4580 allows user applications to share large blocks of memory (up to 64 MB) using only one TLB entry (super page). Moreover, the TLB entry may be locked in place. Each thread can also send to the other threads an Inter-Processor-Interrupt (IPI) to trigger synchronization events.

Applications using concurrent kernel threads or user processes will run on the LX4580 virtually unchanged. Because the level 1 cache is shared and coherency is maintained between the four virtual CPUs of the LX4580 there is no need to set the processor affinity or modify the caches.

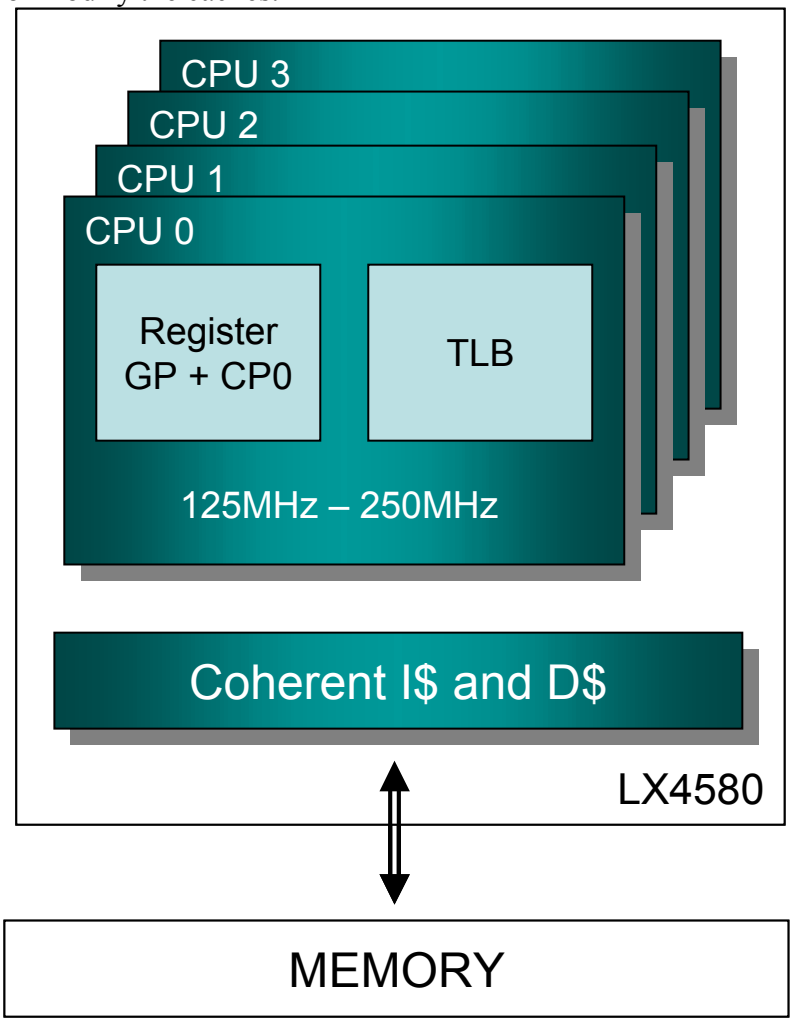

Figure 9. Software View of the LX4580

Another advantage of HMT is that it provides temporal partitioning of the applications. In essence, HMT gives the user free context switching that would otherwise require software support (context save/restore). This feature can be used in I/O intensive applications such as networking. For example on non-HMT systems, the device drivers have to be interrupt-driven but the overhead of the interrupts is high. With HMT it is possible to dedicate one more hardware threads to the device drivers and use polling methods to move data in and out of the processor.

7. Summary

This whitepaper described how Lexra's HMT architecture has been successfully applied to meet the performance demands of High Touch Internet packet processing. Under typical operating conditions the LX4580 CPU delivers three times the performance of a comparable CPU without HMT. This performance is achieved in a small footprint 32-bit synthesizable CPU and compares favorably with the performance of far larger full-custom CPUs. High Touch software can easily realize the HMT benefits because each CPU looks like four virtual CPUs or hardware threads running in parallel.

Later in 2002, Lexra will announce its first network communications chips based on the LX4580. Looking ahead, Lexra's next generation HMT architecture will permit even higher clock speeds while further isolating the CPU from cache miss stalls.

References

Alverson, R., D. Callahan, D. Cummings, B. Koblenz, A. Porterfield and B. Smith [1990]. "The Tera Computer System," *in International Conference on Supercomputing*, pp. 1-6.

Borkenhagen, J, R. J. Eikenmeyer, R. N. Kalla, S. R. Kunkel [2000]. "A Multithreaded PowerPC Processor for Commercial Servers," *IBM Journel of Research and Development*, Vol. 44, No. 6, pp. 885-899.

Farkas, K. I. and N. P. Jouppi [1994]. "Complexity/Performance Tradeoffs with Non-Blocking Loads," *Proc. 21st Annual International Symposium on Computer Architecture.*

Hennessy, John L., David A. Patterson [1996]. Computer Architecture: A Quantitative Approach, Morgan Kaufmann, San Francisco.

Smith, B. J. [1981]. "Architecture and Application of the HEP Multiprocessor Computer System," in *SPIE Real Time Signal Processing IV*, pp. 241-248.

Smith, James E. [1982] . "Decoupled Access/Execute Computer Architectures," IEEE 1982.

Thorton, James E. [1971]. *"Parallel Operation in the Control Data 6600,"* in "Computer Structures: Readings and Examples," C. Gordon Bell and Allen Newell, McGraw-Hill.

Tullsen, Dean M., Susan J. Eggers and Henry M. Levy [1995]. "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proc. of the 22nd Annual Symposium on Computer Architecture* (June), Ligure, Italy.

Figures