HOT CHIPS

Slide 1 "Introduction"

Good Afternoon.
The R10000 is the new high-end chip in the Mips family of RISC microprocessors.
We will describe its micro-architecture, memory hierarchy and system architecture,
We will conclude with a brief look at our verification and design methods.

Slide 2 "R10000"

The R10000 is a dynamic superscalar microprocessor which implements the 64-bit Mips-4 Instruction Set Architecture.
It fetches and decodes four instructions per cycle and dynamically issues them to five fully-pipelined low-latency execution units.
Instructions can be fetched and executed speculatively beyond branches.
Instructions graduate in order upon completion.
Although execution is aggressively out-of-order, the processor still provides sequential memory consistency and precise exception handling.

But our chip is more than just faster way to run the Spec benchmarks.
Its hierarchical non-blocking memory system efficiently runs larger and less well-behaved programs.
There are two levels of set-associative, write-back caches.
Two on-chip 32-Kbyte primary caches provide instructions and data with low latency.
A large external secondary cache uses synchronous static RAMs for high bandwidth, transferring 16 bytes per cycle.
The processor's speculative execution initiates cache refills early so it can hide memory latency.

The R10000 is connected to an external agent using a split-transaction system bus, which can directly connect up to four chips to form a multiprocessor cluster.
Cache directories or duplicate tags are supported to facilitate construction of large multiprocessor systems.

Slide 3 "R10000 Block Diagram"

This diagram shows the major blocks in the R10000.

Four 32-bit instructions are fetched and decoded during each cycle.
If one of these is a branch, its target address is calculated, the branch path is predicted, and instructions are speculatively fetched along the predicted path.

Decoded instructions are put into a 32-entry Active List and three 16-entry instruction queues.
The Active List keeps track of the original instruction order.
The queues dynamically issue each instruction to the appropriate execution unit after all its operands have become available.

The Floating-point Queue issues instructions to the floating-point multiplier and adder.
The Integer Queue issues instructions to two ALUs.
The Address Queue issues instructions to the Load/Store unit and the Data Cache.
The load/store pipeline calculates 44-bit virtual memory addresses and translates them to 40-bit physical addresses.
It uses a 64-entry TLB which is similar to but larger than the R4400's.

Slide 4 "R10000 Pipelines"

The processor was designed modularly using six nearly independent pipelines.
The complex interaction between these pipelines is managed by well-structured logic in the instruction queues. This modularity and regular structure greatly facilitated the design.

The instruction pipeline continues to fetch and decode instructions as long as there is room in the Active List and queues.
When resource conflicts or operand dependencies prevent the queues from issuing instructions in their program order, the queue's dynamic scheduling hardware tries to find other instructions that can be issued instead.

For frequent operations, each execution unit is fully pipelined with a single-cycle repeat rate.
The ALUs execute simple integer operations with single-cycle latency, so that dependent instructions can be issued on consecutive cycles.
The floating-point units have 3-stage pipelines, but special bypass logic reduces latency to only two cycles.
Integer operands are loaded from the Data Cache with 2-cycle latency. Floating-point loads take an extra cycle of latency, because these units are physically farther from the Data Cache.

Slide 5 "Register Renaming"

One challenge of superscalar design is to avoid issue restrictions.
Register renaming provides three crucial advantages.
First, speculative results can be written directly into the register file.
Second, write-after-read and write-after-write hazards are eliminated.
Finally, true dependencies can be easily detected.

During instruction decode, integer and floating-point registers are renamed using separate mapping tables.
This hardware handles almost any sequence of four instructions, including sequences with dependencies and instructions destined to the same functional units.

Renaming maps 32 "logical" register numbers into 64 physical registers.
The physical registers contain both committed and speculative values.
When each instruction is decoded, its result is assigned to a physical register from a "Free List" of currently unused registers.
At graduation, this register contains a new committed value, and the previously assigned physical register is returned to the Free List.
Thus, each physical register is uniquely associated with just one value; dependencies can be determined simply by comparing physical register numbers.

Slide 6 "Control Paths"

With renaming, control complexity is implemented in regular structures, such as the mapping tables, active list, and instruction queues, which contain only 6-bit physical register numbers.

The 64-bit data paths are no more complex than in an in-order processor, except that the register files are larger.
Operands are read directly from the register file, and results are written directly back.
Results may be bypassed into any operand register.
But there are no separate structures, such as reservation stations or re-order buffers, in the wide data paths.

Slide 7 "Instruction Fetch and Branch"

Four instructions are fetched in parallel at any alignment within a 16-word Instruction Cache line.
The direction taken by a conditional branch is predicted using a 2-bit algorithm, based on a 512-entry Branch History Table.
Simulations show an 87% prediction accuracy for Spec integer programs.

Each prediction is verified as soon as its branch condition is determined.
If its prediction was incorrect, all instructions fetched along the mis-predicted path are immediately aborted, and the processor state is restored from a 4-entry Branch Stack.
This allows rapid recovery for up to four mis-predicted branches.

Fetching along predicted paths may have initiated unneeded cache refills. However, the cache is non-blocking, and the correct path can be fetched while these refills are completed.

Slide 8 "Floating-Point Unit Block Diagram"

The R10000 has four independent 64-bit floating-point execution units.
The adder and multiplier are each fully pipelined with single-cycle repeat rate and latency of just two cycles.
This delivers good performance for general calculations which have serial dependencies, not just for vectorized code.

The adder includes a leading-zero predictor and a dual-carry-chain adder to round its result before it is normalized.
The multiplier includes a full radix-4 Booth-encoded array with a product compression tree using 4-to-2 carry-save adders.

Division and square root are performed in separate iterative units which operate concurrently with the pipelined units.
Each unit generates two bits per stage using an SRT algorithm.
To reduce latency, the divider cascades two stages, so it can generate four bits per cycle.
These units share the multiplier's issue and register file ports.
Separate ports are not justified, because the issue rate is low.

The new Mips-4 instruction set includes a combined multiply-add instruction for improved code density. This operation is not "fused".
Although a fused unit can reduce the latency of the combined instruction, it increases latency and reduces bandwidth for separate operations, and implementing it requires more area and power.

Slide 9 "R10000 Memory Hierarchy"

The R10000 design emphasizes an efficient memory hierarchy which tolerates latency.
It finds cache misses early, and begins refills in parallel with other useful work.
The on-chip caches provide concurrent access for instruction fetch, data load and store, and refill.
Refill data is transferred critical-word-first on wide high-bandwidth buses and can be streamed for immediate use.
All caches are 2-way set-associative with LRU replacement.

Slide 10 "Parallelism in Data Cache"

The Data Cache is 2-way interleaved with independent tag and data arrays for each bank.
These four arrays operate under shared control of the Address Queue and the External Interface.
The queue concurrently processes up to 16 load and store instructions in four separate pipelines.
It dynamically issues instructions to the address calculation and translation pipeline.
The other three pipelines can concurrently perform tag checks, execute loads, and graduate store instructions.
Although address calculation and loads can occur out-of-order, instructions appear to execute with strong memory order.

The External Interface can select either bank for refill, write-back, or tag interrogate operations.
This concurrency allows the R10000 to efficiently run real MP applications.

Slide 11 "Data Cache Pipelines"

The Data Cache pipelines interact during several common cache sequences.
Loads can perform the tag check and data read during the same cycle as the translation.
Store instructions immediately issue a tag check to initiate any required refill as early as possible, but writing data into the cache is delayed until the store becomes the oldest instruction and graduates.
A miss in the primary data cache will initiate a refill sequence from the secondary cache.
For loads, refill data can be bypassed directly into the register file.

Slide 12 "Non-blocking Cache"

Although it is inappropriate to provide Spec numbers for unannounced systems which are not yet finalized, analysis of a few Spec benchmarks shows how the R10000's non-blocking caches reduce its cycles-per-instruction ratio.

For each benchmark, the entire lower bar is proportional to its CPI.
The darker right-hand area represents the time lost to cache misses.
The upper bar represents total delay if all cache refill operations were executed serially.
Thus, the arrow represents the time penalty that a blocking-cache would incur.

The benefit of non-blocking caches varies widely with the program's characteristics.
Non-blocking caches do little for small benchmark programs whose working sets fit entirely in the primary caches, but they greatly benefit more realistic programs, like "tomcatv" or the notorious cache-buster "compress".

Slide 13 "Data Cache Interleaving"

The two banks of the Data Cache are each divided into two logical arrays to support 2-way set-associativity. Unlike the usual arrangement, however, the cache ways alternate between these arrays, to efficiently support different-width accesses.

The processor simultaneously reads the same doubleword from both cache ways,
because it checks the cache tags in parallel and later selects data from the correct way.

The external interface refills or writes back quadwords by accessing two doublewords in parallel.
This is possible because the cache way is known in advance.

Slide 14 "System Configurations"

The R10000's flexible external interfaces support a wide variety of system configurations.
Its independent secondary cache and system interfaces operate concurrently, and can be individually optimized to meet packaging and layout constraints.
The secondary cache consists of 512K to 16M bytes of synchronous static RAM.

The system interface multiplexes address and data on a single 64-bit bus to conserve package pins.
It directly connects up to four processors to a memory controller or other "external agent".
Multiprocessor systems may use snoopy, duplicate tag, or directory coherency schemes.

Slide 15 "Clocking and Output Drivers"

An on-chip Phase-Lock Loop generates all timing synchronously with an external system clock. For system design and upgrade flexibility, independent clock divisors allow the choice of five secondary cache and seven system interface clock frequencies.

Four types of output buffers can be separately configured to conform to either low-voltage CMOS or HSTL standards.
The System Interface buffer contains additional open-drain pull-down transistors, which provide the extra current needed to drive HSTL Class-2 multi-drop buses.
The Secondary Cache Data buffer needs reduced overlap current, because nearly two hundred of these signals can switch simultaneously.
The Cache Address buffer drives multiple distributed loads.

The Cache Clock buffer drives low-impedance differential signals with minimum output delay. This clock's phase can be mode selected to adjust for propagation delays in the printed-circuit board clock net, so that its rising edge arrives at the cache simultaneously with the processor's internal clock.
This programmable delay element precisely aligns the clocks with low jitter.

Slide 16 "Refill from Set-Associative Secondary Cache"

Set-associativity is desirable to reduce conflict misses and to increase predictability.
For an external cache, however, this usually requires special RAMs or many more interface pins. Instead, the R10000 implements an novel 2-way pseudo-set-associative secondary cache using standard synchronous SRAMs and only one extra address pin.
This diagram shows how cache refills are pipelined.
The single set of RAMs contains both cache ways.
An on-chip bit array keeps track of which way was most-recently-used.
After a primary cache miss, two quadwords are read from this way in the secondary cache.
Its tag is read along with the first quadword. The tag of the alternate way is read with the second quadword by toggling the extra address pin.

Three cases occur. If the first way hits, the data is available immediately.
If the alternate way hits, the secondary cache is read again.
If neither way hits, the secondary cache must be refilled from memory.

Slide 17 "Secondary Cache Error Protection"

Large external caches require ECC protection for data integrity.
In the R10000, both a 9-bit ECC code and a parity bit are stored with each data quadword.
The extra parity bit reduces latency because it can be checked quickly to stop bad data from being used.
If a correctable error is detected, the read is retried through a 2-cycle correction pipeline.

The interface can be configured to use this correction pipeline for all reads.
Although this increases latency, it allows redundant lock-step processors to remain synchronized in the presence of correctable errors, which is important to our friends at Tandem.

Slide 18 "System Interface Block Diagram"

The System Interface dedicates substantial resources to support concurrency and out-of-order operation.
The "Cached Buffer" contains addresses for four outstanding read requests.
The memory data returned from these requests is stored in the 4-entry "Incoming Buffer", so that it can be accepted at any rate and order.
The "Outgoing Buffer" can hold up to five "victim" blocks to be written back to memory.
An 8-entry "Cluster Buffer" tracks all outstanding operations on the system bus.
It ensures cache coherency by interrogating, and if necessary invalidating, cache lines.

Uncached stores are often used for writing to graphics or other peripheral devices.
Such sequences typically consist of numerous sequentially or identically addressed accesses.
The "Uncached Buffer" automatically gathers these into 32-word blocks to conserve bus bandwidth.

Slide 19 "Functional Verification Methodology"

Our methodology has demonstrated that a complex superscalar microprocessor *can* be designed and verified.

The R10000 RTL uses an in-house hardware description language which runs on a fast software simulator. We chose not to use hardware emulation.

The simulator provides a graphical user interface which can back up and replay interactively.
The simulation is automatically checked against an instruction-level architectural model, which verifies the contents of registers and the memory hierarchy at each instruction graduation boundary.

Multiprocessor system models combined R10000 RTL code with secondary cache, bus controllers, and memory.
These models generate external events and detect bus protocol violations.

Slide 20 "Diagnostic Development"

Thorough testing relied on both directed and random diagnostics.

AVP's verify compatibility to the Mips instruction set architecture.
MVP's verify compliance to the R10000's micro architecture specification.
IVP's test specific implementations, based on a detailed knowledge of the RTL code.

Programmable random code generators focus on specific functional units.
These generators emit self-checking code for both UP and MP environments.

Finally, we simulated real applications including booting both NT and Unix operating systems.

This strategy worked.
We booted Unix on first silicon within two weeks, and we are now running MP Unix.

Slide 21 "Test Features"

To be manufactured economically, a microprocessor chip must be easily testable with a high fault-coverage.
The R10000 observes internal signals with ten 128-bit linear-feedback shift registers.
These internal test points partition the chip into three fully-observed sections.

The registers are separate structures which do not affect the processor’s logic or add a noticeable load to the observed signals.
They use the processor’s clock to ensure synchronous behavior and avoid any special clock requirements.
They are useful for debugging or production testing.

Slide 22 "Clock distribution methodology"

One design challenge for superscalar processors is to attain the same high clock speeds achievable in simpler architectures.

A balanced clock tree was designed to minimize skew.
The PLL generates a double-frequency clock, which is sent to the center and is then distributed to the four quadrants of the chip.
For added isolation, the main clock trunks are wired in metal-4 with metal-3 shields.
At the end of each branch, an area buffer divides the clock’s frequency to generate synchronized clocks for the processor, system, and secondary cache.
Finally, local buffers provide complementary clocks to registers, latches and dynamic logic.

Slide 23 "Repeater strategy"

The RC delay in long signal wires can limit the speed of large microprocessors built with sub-micron features.
To improve timing and signal integrity, 700 non-inverting repeaters were automatically inserted into the longest signal nets.
This graph shows the distribution of global wire lengths before and after the repeaters were inserted.
As can be seen, the number of wires above 12 mm has been significantly reduced.

Slide 24 "Die Photo"

The R10000 die is 16.6 by 17.9mm
It contains 6.8 million transistors, including 4.4 million in the cache arrays.
The main sections of the processor are outlined.

In conclusion, the R10000 is an aggressive superscalar microprocessor.
It delivers high performance from a sophisticated micro-architecture, fast clocks, and a non-blocking, set-associative memory subsystem.
Its design emphasizes concurrency and latency hiding techniques to efficiently run large real-world applications.   Thank you.