Architectural Strengths of the MIPS32® 74K™ Core Family

K.R. Kishore, Vidya Rajagopalan,
Georgi Beloev, and Radhika Thekkath

Document Number: MD00652
Revision 01.00
May, 2000

1 Introduction

The MIPS32® 74K™ core family from MIPS Technologies introduces a completely new superscalar, out-of-order pipeline architecture, designed to maximize the performance achievable by synthesizable methodology. This 17-stage pipeline dual-issues instructions to make the most effective use of the load-store nature of the MIPS32 RISC instruction set architecture. Other innovations in the 74K core include out-of-order issue, support for fast memory copies, and new instructions to accelerate DSP and media processing. The 74K core is fully synthesizable and designed for custom integration into a variety of system-on-chip (SoC) applications. It is high performance, low power and highly portable across process technologies, allowing SoC designers to focus on the system design and end-product definition.

This paper first describes the basic architecture and microarchitectural design choices made in the 74K core. The paper then illustrates the architectural strengths and benefits of this processor family. These include, for example, a description of the design choices that allow the 74K cores to achieve high clock frequencies using a purely synthesizable methodology. This paper also describes in detail the optimizations done to obtain high performance memory copy functionality.

2 Base Architecture Description

The 74K core implements the MIPS32 Release 2 instruction set. The 74Kf™ core also implements the 64-bit floating point instruction set. To enable code compaction, the core implements the MIPS16e™ ASE (Application Specific Extension). The core implements the MIPS32 DSP ASE Revision2, which accelerates DSP and media processing applications. The implemented instruction set architectures are supported by a wide range of industry-standard tools and development systems. Also available are a wide range of third-party application suites such as audio codecs, VoIP codecs, PVR stacks, and middleware.

In addition, the 74K core family includes a CorExtend® block which allows customers to add a functional unit to the 74K core pipeline with access to all programmer-visible general-purpose registers (GPRs) and accumulators. With this functional unit a customer can implement their own set of specialized instructions, each selecting up to two source GPRs and one accumulator. This is a powerful feature that allows the user to extend the existing MIPS32 Release2 instruction set to improve performance and customize the core for their target application.

3 Pipeline Design of the 74K™ Processor Core Family

The standard 74Kc processor core is a superscalar processor capable of issuing two integer instructions every clock cycle. The 74Kf core can also issue two floating-point instructions concurrently with the integer instructions. The following sections describe how the various classes of instructions map to the pipeline design of the 74K core.

3.1 Integer Dual-Issue Pipeline

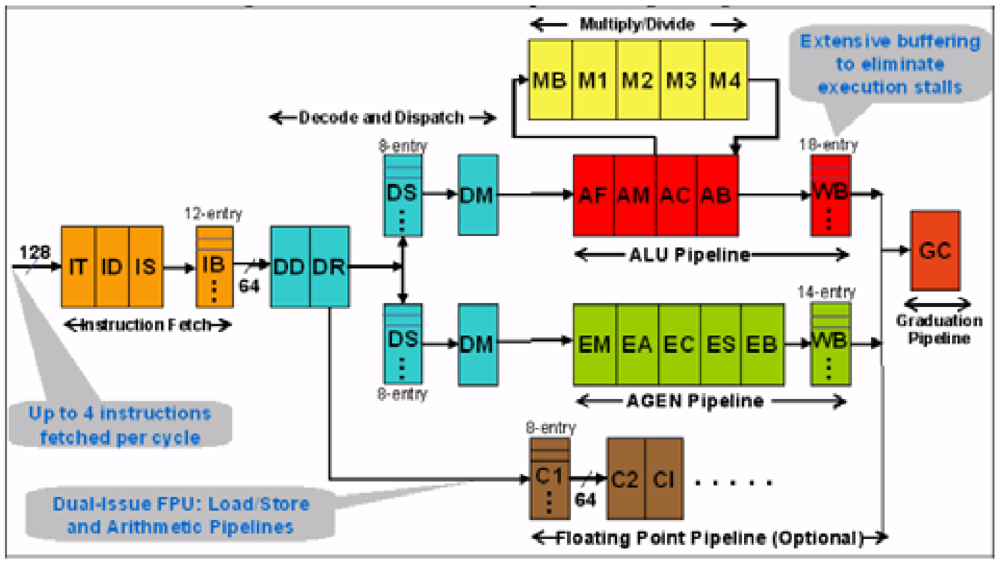
The integer instruction dual-issue is achieved through two pipelines:

- The AGEN (Address Generation) pipeline has 17 stages and implements the memory transfer (load/store) and control transfer (branch/jump) classes of instructions.
- The ALU (Arithmetic Logic Unit) pipeline has 16 stages and implements the arithmetic, logic, and computation instructions.

These two pipelines have a common 8-stage front-end that includes instruction fetch, decode, and dispatch. These pipelines also have a common 2-stage backend that includes logic to deal with instruction graduation. Instructions that go through ALU/AGEN pipes generate their results and hold them in temporary storage called ALU/AGEN Completion Buffers.

As shown in Figure 1, the pipeline design consists of multiple pipelines that communicate via buffers to alleviate the global stall problem. The pipeline is non-stalling, that is, all stalling events are collected and dealt with at the end of the pipeline so that following instructions can complete without a bubble in the execution. This microarchitecture design leads to a great performance advantage.

Figure 1 74K Core Pipeline Stage Diagram



The Multiply Divide Unit (MDU) pipe is split from the ALU pipe after the 32-bit operands are read. This pipeline executes all integer multiply, divide, and some DSP ASE instructions. The pipeline has a 32x32 booth recoded multiplier array with support for DSP Q31, Q16 and quad operands. The MDU is able to support a single cycle repeat rate for multiply-accumulate class operations, and is 5 stages deep.

The instruction groupings for the two pipelines were carefully chosen to minimize bypassing requirements and maximize performance. The exact asymmetry was determined from studies done to ensure that the instructions executing across the two pipelines were balanced for occurrence in common code, thus resulting in the best possible performance. This has resulted in a minimization of bypassing across the two pipelines and two compact pipelines that are optimized for high frequency and low power.

3.2 DSP Instructions

The DSP ASE Revision2 instructions are implemented in the ALU and MDU pipes. The instructions are issued into the ALU pipe. When the instructions need to execute in the Multiply-Divide Unit (that is, when they access the accumulator or use the multiplier), they branch off to the MDU pipe in the appropriate stage.

3.3 Floating Point Instructions

In addition to the integer pipelines, the 74Kf core family includes support for executing floating point instructions. The dual-issue Floating Point Unit (FPU) pipeline is 7 stages deep and receives instructions from the integer dispatch unit. The FPU is capable of receiving two instructions every cycle, one data transfer instruction and one arithmetic instruction.

3.4 Out-Of-Order Dispatch

The 74K core can dispatch instructions out-of-order. Register renaming is used to execute out-of-order and still complete in order, thus preserving the correct architectural state. When an individual instruction completes its computation phase at the end of the pipeline, it writes its result into the completion buffer (CB). An entry from the CB is written into the GPRs (General Purpose Registers) only when the instruction is graduated. Exceptions are also taken at graduation time thereby ensuring correct priority and precise state. Out-of-order dispatch helps improve the performance for the 74K core's deep pipeline and comparatively long execution unit latency. The amount of performance gained is dependent on the instruction level parallelism that exists in a given code stream.

An additional advantage of out-of-order dispatch in the 74K core is its ability to provide excellent performance on existing binaries. This fills a critical need since embedded processors frequently need to execute binary code which has been compiled and optimized for older processor architectures. For example, there are literally millions of lines of existing middleware code for the MIPS32 architecture which exist in binary form without an ability to be recompiled. The 74K core thus provides a compelling advantage being not only code compatible with previous MIPS32 implementations, but also by being resilient to the need to re-schedule old code for the new pipeline design.

3.5 Branch and Return Prediction

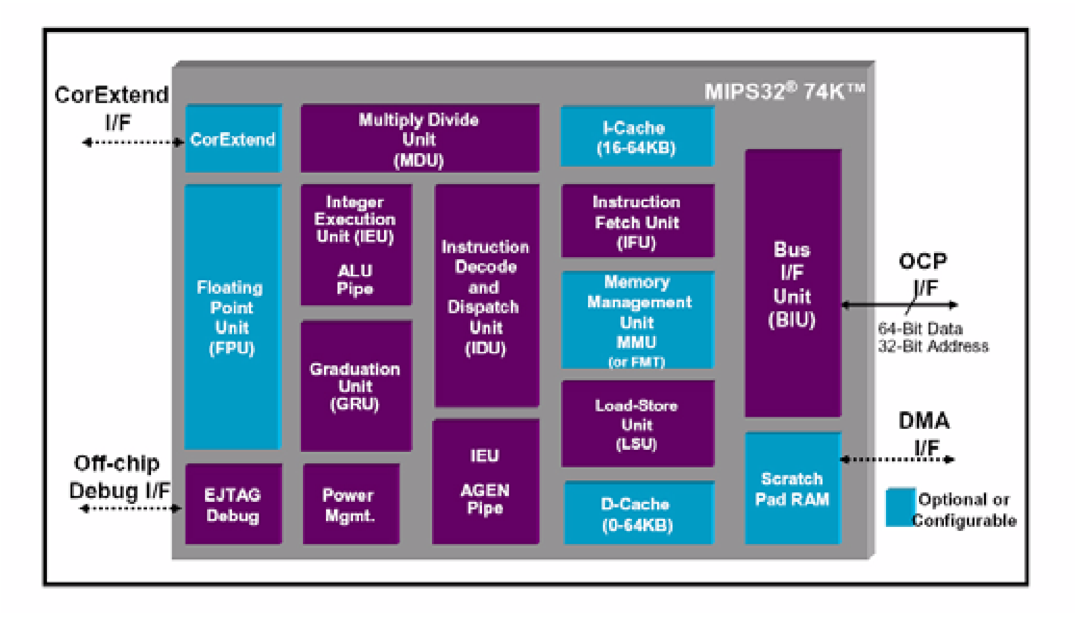
The 74K core implements a majority predictor based on the gshare branch prediction algorithm. Three 256-entry bimodal branch predictor tables are used to implement this algorithm. This predictor provides high accuracy prediction while minimizing aliasing effects with the least impact on area and timing. The 74K core also has an 8-entry return prediction stack to help in function return prediction.

4 Memory System Design of the 74K™ Processor Core Family

4.1 The Instruction Cache

The 74K cores have a 4-way set-associative level one instruction cache which is customer configurable at synthesis time to 16, 32, or 64 KB in size. This provides customers the flexibility to customize their design for either performance or size, based on the target application demands. The instruction cache is designed for high instruction fetch bandwidth to the pipeline by supporting two outstanding misses and allowing 4 instruction fetches per cycle. The instruction cache is virtually indexed and physically tagged to make the data access independent of virtual to physical address translation latency.

Figure 2 The 74K™ Core Block Diagram



4.2 The Data Cache

Similar to the instruction cache, the data cache is 4-way set-associative and allows customer configurations of sizes 0, 16, 32, or 64 KB. The data cache is non-blocking and supports up to four outstanding misses. Also, like the instruction cache, the data cache is virtually indexed and physically tagged. The data tag array has a virtual address part which is used to compare against the virtual address being accessed and generate a data cache hit prediction. This virtual address hit prediction is always backed up by a comparison of the translated physical address against the physical tag. This prediction allows efficient pipelining of data cache misses.

4.3 Scratch Pad RAM

The 74K core allows blocks of Scratch Pad RAM to be attached to the load/store unit. These RAMs can be used for fixed, low-latency access to memory. These memory blocks may be modified by the customer. MIPS Technologies includes a reference design with an external DMA port allowing direct system access to the memory array.

4.4 Memory Copy Function

Since block memory copy operations are very common in different system designs, the 74K core provides special configuration options to speed up data movement. This includes the addition of new hint fields to the Prefetch instruction and widening of the data cache data port width to 128 bits. The data and control path of load/store instructions is highly optimized to move data from memory in the event of a data cache miss. More details of the memory copy function optimizations are provided later in the paper.

5 An Introduction to the DSP ASE Rev2 and its Performance Benefits

The MIPS Digital Signal Processing Application Specific Extension—DSP ASE—was introduced with the MIPS32 24KE® family of cores. The DSP ASE enhances the MIPS32 architecture with a set of new instructions that improve the performance of signal processing and multimedia applications. It uses a Single Instruction Multiple Data (SIMD) approach to operate simultaneously on four 8-bit or two 16-bit data elements packed into 32-bit registers. For example, a single ADD instruction can compute four 8-bit sums or two 16-bit sums. Replacing normal operations with SIMD operations reduces the number of instructions and processor clock cycles needed to perform a computation.

The benefits of using SIMD instructions to process data extend beyond accelerating computations. Data can now be efficiently transferred as 32-bit memory words using load word and store word instructions. In many cases the data does not need to be unpacked before it is used or packed before it is stored. To further reduce the overhead of bringing data to the processor, the DSP extension offers indexed load instructions that add an index register and a base register to compute the effective address of a load.

The DSP ASE supports 8-bit, 16-bit, and 32-bit integer and fractional data types commonly used in multimedia applications. The integer data types are useful in image-processing and video-processing algorithms where the pixels are typically represented by 8-bit integers and the calculations are performed with 16-bit precision. The fractional data types represent numbers in the range from -1.0 to +1.0 and are convenient for implementing algorithms like Fast Fourier Transforms (FFT) and Finite Impulse Response (FIR) filters, which are basic building blocks for many signal processing applications including audio algorithms. Typically, 16-bit data types are sufficient for voice applications such as VoIP.

Multiply and Accumulate (MAC) operations are usually the most critical operations in DSP algorithms and are found at the heart of many algorithms. The DSP ASE provides a variety of MAC operations including SIMD instructions that compute and accumulate two full-precision products per cycle. To facilitate the implementation of MAC-intensive algorithms the DSP ASE (Revision 1) adds three additional 64-bit accumulator registers for a total of four accumulators. The availability of four accumulators permits loop unrolling of compute kernels, which reduces loop overhead and improves computation throughput.

Two special operations—rounding and saturation—are often found in DSP algorithms. The DSP ASE provides rounding and saturation options for many instructions, which improves performance and facilitates algorithm implementation and standards compliance.

The DSP ASE also offers SIMD precision expansion and precision reduction instructions, SIMD compare and pick instructions, variable-length bitfield extract instructions, and flexible accumulator value extract instructions. These instructions complement the functionality offered by the SIMD arithmetic, shift, and multiply instructions and address the needs of specific algorithms. For example, the SIMD compare and pick instructions can be used to implement efficient Viterbi decoding algorithms.

The 74K family of cores implements Revision 2 of the DSP Extension. The new revision provides additional support for image-processing and video-processing applications that use 8-bit integer data types. The orthogonality of the DSP instruction set is improved, making it an easier target for automatic compiler code generation. Programming is simplified because of the introduction of instructions performing more sophisticated operations. One such example is the new set of dot product instructions that cross-map the SIMD data first before the multiplication. These instructions offer flexibility to the programmer; it is no longer required to permute the data at run-time (at the expense of decreased performance) or rework the algorithm and rearrange its data in such a way that run-time data permutations are not required. Overall the DSP ASE Rev2 saves both development time and execution time.

The performance improvement achievable by utilizing DSP ASE instructions depends on the algorithm and the data types it uses. Typical speedups are around 1.5x for 16-bit data types and around 2.0x for 8-bit data types. In some cases the speedup can be significantly higher if an algorithm maps particularly well to the operations offered by the DSP ASE.

5.1 The Benefits of Dual-Issue for DSP Inner Loops

In combined RISC/DSP processors like the 74K core, the primary bottleneck to obtaining better speedup for DSP code lies in the number of loads and stores needed for each computation instruction. Typically, each arithmetic instruction uses two operands and takes two load instructions to retrieve the operands to support this computation. An additional store instruction may be needed after the computation if the result needs to be immediately written back to memory. Using the dual-issue nature of the 74K core’s pipeline data load instructions can be overlapped with computation instructions, significantly reducing the number of clock cycles needed to execute a DSP function.

The impact of the 74K core’s dual-issue capability is even more pronounced when we consider the use of SIMD operations to process the data. Assuming 16-bit data types for the SIMD operations, which allows two 16 bit results to be computed using a single DSP ASE instruction, the 74K core can effectively compute one MAC result every cycle in a pipelined fashion. This provides a 2x speed improvement over the 24KE core family for DSP inner loop computations. An important point to note here is the fact that code written and optimized for the 24KE core will run unmodified on the 74K core with significant improvement in performance. For example, it is common for unmodified DSP kernel functions to take 35% fewer cycles on the 74K core compared to the 24KE core. Taking the frequency advantage of the 74K core into account (approximately 30%), the overall benefit is much higher (60% or more).

6 How Does the 74K Core Achieve High Speeds Using Only Synthesized Logic?

Providing IP cores in a fully synthesizable format allows for quick proliferation of the IP across multiple customers, applications, and fabrication technologies. Providing IP cores with high frequencies is desirable because it is a very tangible way to gain performance on existing applications. The main reason for a deep pipeline in the 74K core is to achieve high frequency in a fully synthesizable methodology. There are super-pipelined CPUs, especially in the world of high-end desktop processors, that achieve much higher frequencies than the 74K, or custom CPU pipelines which could achieve the 74K frequency with fewer pipeline stages. But all of these rely on process-specific custom libraries and circuit design for speed, unlike the 74K which uses a purely synthesizable methodology.

But a deeper pipeline with no other changes degrades performance (IPC—Instructions Per Cycle), hence the 74K core called for better architecture and engineering design of many of its other critical components. Some examples of these design optimizations are discussed in the rest of this chapter.

6.1 Better Branch Prediction

Similar deeply-pipelined processors use large structures for branch prediction. Rather than use a large table which would require a custom or structured logic block, the 74K core uses a relatively small table (256-entries) combined with other architectural techniques that make the smaller structure as effective as a bigger one. The relatively shallow table depth causes more aliasing cases and, thus, 74K core uses a majority predictor to disambiguate these aliases. This branch prediction algorithm along with the early resolution and two-step recovery mechanisms allows the 74K core to use only register file based structures and still attain very high accuracy.

6.2 Data Cache Hit Prediction Using a Virtual Tag-Based Hint

Independent timing experiments on the load/store pipe showed that the critical serial logic is [Effective Address Generation -> Dcache/TLB Lookup -> Physical Tag Compare and Data Select -> Data align and Bypass]. In this sequence, the data cache lookup is determined by the SRAM timing. The TLB in synthesizable processor cores is built using register files and has CAM functionality. Performing a TLB lookup in 1 cycle severely limits the size of the TLB. Previous generation processor cores from MIPS Technologies use a small TLB called microTLB. But even a small microTLB that provides a meaningful hit rate was not feasible at 74K core's frequency targets. Short of using a custom structure for TLB, the alternative was to use virtual tag based hint to determine a cache hit. This hint is used to predict a data cache hit and bypass the cache data to consumers. A full TLB lookup is done in parallel and the predicted data cache hit is validated. In the event of a mis-prediction, the load is deemed a cache miss and allowed to graduate. A data cache refill request is initiated only if necessary since the load data could be present in the cache. In either case, the data is returned to the consumer via the miss handling path.

6.3 Out-Of-Order Execution

The AGEN and the ALU pipes each have an 8-instruction window for use with out-of-order instruction dispatch. This size was chosen as an optimal balance of performance, size, and power. This size is radically different from the windows typically seen on modern desktop processors (128 or 256 entries). For desktop processors, one of the primary motivations for using out-of-order is to hide latencies to the second-level cache. The primary intent here is to hide instruction latencies in the pipeline, to hide load-to-use penalties, and to provide added pipeline flexibility to enable higher frequencies. Out-of-order execution, as already stated, provides the right trade-off for a synthesizable core by increasing both architectural performance and design frequency.

7 Optimizing Memory Copy Function to Address Current SoC Designs

Microprocessor cores within a SoC are often tasked with the job of copying arbitrary blocks of memory from one location to another. While some special-purpose processing engines may use DMA to move data, generic applications will use normal C library functions to accomplish this task. This is common practice simply because it is the easiest model for software applications and requires no special management. Only when the amount of memory to be moved is significant and other conditions specific to the application are favorable will a DMA based solution be better. Further, a MemCopy function skews the mix of instructions compared to a typical mix. With a MIPS (or a general RISC) instruction set architecture, a typical workload consists of about 40% helper instructions (30% load/store and 10% branches) and the remaining 60% are computation instructions. However, in a MemCopy function no computation is performed and the instruction mix is heavily skewed towards load/store (approximately 80% to 90%). The data structures such as Completion Buffers, which determine the number of instructions in flight are sized to accommodate a typical workload and not a skewed workload such as MemCopy. Further, these structures cannot be made arbitrarily large because they have to be built using register files while being able to meet the frequency target. To improve the performance, the 74K core has several advanced micro-architectural features. Some of these are:

- Widening the data cache read/write width to 128 bits so that fills and evictions can be done faster. The read path from the core pipeline is multiplexed down to 64 bits to keep the same data-path.
- A new hint value (31) was added to the Prefetch instruction to effectively reduce one cache access cycle in the Prepare-for-Store operation. Since the Prepare-for-Store operation is always followed by a real store the zeroing of the cache is eliminated.
- A miss request is launched as soon as the load/store is graduated and is determined to be a unique miss.

- Overlaying data cache and tag access for different instructions. Since store instructions perform the tag lookup from the main pipe but write the cache only after they graduate, these two distinct operations are overlaid so that on a given cycle the tag and data arrays are addressed and accessed independently.

- A fill/eviction that requires both tag and data access is optimized such that they can request a bubble for use in a forthcoming cycle. The dispatch unit grants this request only if there is no natural bubble in flight and generates an empty slot to meet the fill/eviction at the “right” time. This is done to minimize the lifetime of an entry in the Fill Store Buffer (FSB). (The FSB holds in-flight cache transactions).

Other design refinements in the 74K core to make MemCopy fast include: store merging in the FSB, intelligent arbitration for cache access, fast bypass of miss data to consumer, aggressive allocation of Completion Buffer IDs to compensate for shallow Completion Buffer table in the AGEN pipeline, and minimizing the number of cache probes for dirty/LRU state determination.

8 Summary

This paper describes the architectural and microarchitectural design strengths of the MIPS32 74K family of cores from MIPS Technologies. This core is designed for the highest frequencies using a purely synthesizable methodology. This design has led to key features such as a deep pipeline, asymmetric dual-issue, out-of-order execution, highly effective branch prediction, aggressive data cache design, and optimized memory copy functions. Taken together, the 74K core is an innovative design in the embedded market and will play a key role in bringing high-end, compelling consumer devices to the marketplace.