The MIPS32® 24KE™ Core Family: High-Performance RISC Cores with DSP Enhancements

Chinh Tran
Chijioke Anyanwu, Sanjai Balakrishnan, Anshul Bhargava,
James Jiang, Radhika Thekkath

MIPS Technologies, Inc.

The MIPS32® 24KE™ core family is the latest high-performance synthesizable microprocessor cores from MIPS Technologies, Inc. and features DSP enhancements at a negligible cost. The 24KE core family fills an important gap in the convergence of RISC processors and digital signal processors (DSPs). This convergence is enabled by emerging market trends, and comes with cost advantages and technical challenges. The 24KE design meets these challenges quite handily as shown in this paper. The 24KE cores were built by adding the new DSP enhancements to the existing pipeline in the 24K® cores. The performance benefit for DSP applications from these enhancements can be as much as two times compared to a 24K design. And yet the 24KEc™ core has minimal frequency degradation and increase in the die area compared to the 24Kc™ core. This paper discusses some of the novel implementation techniques used to achieve this result.

This paper is organized as follows. The market trends promoting the convergence of RISC processors and DSPs are described first. Then, an overview of the 24K core family is provided since this is the foundation upon which the 24KE core family is built. This is followed by a brief description of the MIPS32® DSP Application Specific Extension (ASE). This leads to a discussion of the design challenges, in which we focus on the base core of each family. The design challenges are described in the context of adding the DSP enhancements to the 24Kc core to create the 24KEc core. The paper concludes with a discussion of the achieved results and a summary.

The Convergence of RISC and DSP

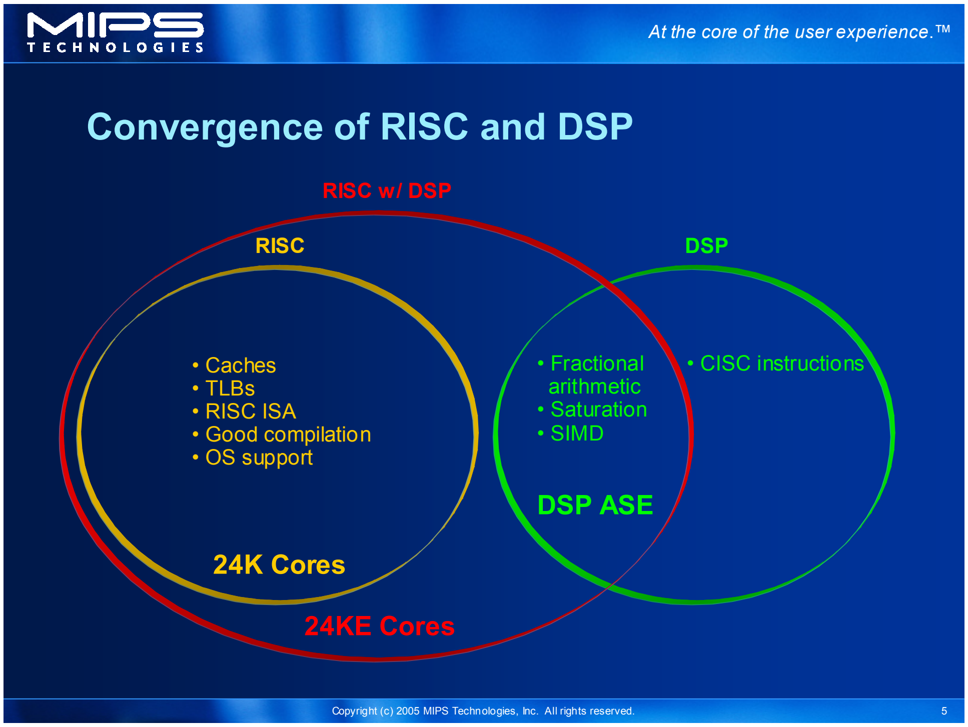
Embedded consumer devices require both general-purpose processing such as running the operating-system ***and*** digital signal processing such as decoding music streams. Examples of such devices include set-top boxes, DVD players and recorders, Voice over IP phones, MP3 players, mobile phones, and personal digital assistant devices (PDAs). Add to this need for combined processing the fact that RISC processors can already execute some DSP operations and the trend seems natural. According to DSP Analyst Will Strauss at Forward Concepts, “at least 30% of RISC processors are suitable for ‘real’ DSP applications [1].” Sample applications include audio codecs, packet protocol processing, and JPEG processing. Further, as RISC processors continue to increase their speeds, they are subsuming even more types of DSP applications. So this trend should continue if not accelerate.

There are advantages and challenges to combining general processing and digital signal processing on the same core. One big advantage is a more efficient hardware architecture [2].

The additional die area to implement DSP enhancements in the core can be kept small compared with the area for a programmable DSP. One reason is that the DSP enhancements can share much of the hardware already in the core. Also, a combined architecture eliminates the external data bus between the processor core and the DSP. This can also reduce power consumption because data is no longer driven back and forth through such an external bus [2].

Another advantage is that software developers need to use only a single tool chain for writing both general-purpose code and DSP code. Eliminating the need to deal with multiple tool chains should lower development costs and accelerate development.

The advantages discussed above are significant, but three key challenges must be overcome to permit combining general and digital signal processing onto a single core. First, the core must deliver enough performance for real-time execution of DSP tasks. Second, the DSP enhancements must be added to a high-performance RISC core without adversely affecting the clock frequency, which would impact both the general-purpose as well as the DSP performance. And, third, the die size increase of the enhanced core must be kept small. These challenges are the focus of this paper.


At the core of the user experience.™

Convergence of RISC and DSP

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 5

Figure 1: Features of RISC processors and DSPs are converging.

RISC processors have many benefits that enable them to achieve high performance. For example, caches and translation look-aside buffers (TLBs) help to improve software performance. RISC instruction sets and pipelines are usually designed for relatively high clock frequencies. And, of course, RISC instruction sets lend themselves to good compilation [3]. A distinguishing feature among RISC architectures is the ability to run several different operating systems which have been ported to that architecture. MIPS processors, such as those in the 24K core family, offer the advantage of being supported by operating systems commonly used in embedded systems.

DSPs usually offer special features, such as fractional arithmetic, saturation, and single-instruction multiple-data (SIMD) operations. They also often implement special complex instructions to boost their digital processing performance. These instructions do not follow the RISC paradigm because they perform expensive functions, such as operating directly on memory operands. As a result, they create many critical paths that limit the clock frequencies of the DSPs [2].

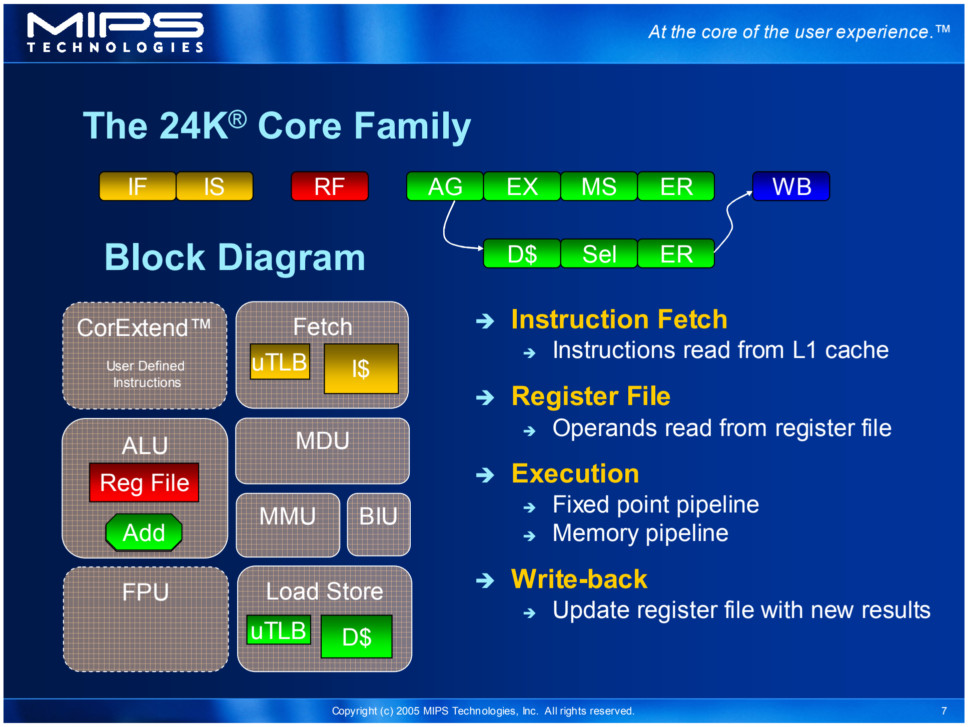
A RISC core with DSP enhancements should inherit all the benefits of the RISC architecture. If it incorporates DSP support in a RISC-like manner, it would avoid the increased cycle time required to implement complex instructions. This will enable it to achieve a higher clock frequency and better code compilation than a DSP. When compared to having separate RISC processor and DSP, the combined core should have a lower die area dedicated to DSP operations due to the hardware reuse and the elimination of the external bus discussed earlier.

The rest of this paper will focus on the creation of such a RISC core with DSP enhancements. The 24Kc core is the base RISC core for this design. At the Fall Processor Forum in 2004, MIPS Technologies introduced our DSP ASE, an architectural extension to enhance DSP performance with low implementation costs. Implementing this ASE in the base design results in the MIPS32 24KEc core.

The 24K® Core Family

The 24K core family consists of the highest-performance 32-bit synthesizable cores for the embedded market. The base core in this family, the 24Kc core, can achieve 400MHz to 625MHz in a 130 nm CMOS process and can execute 576 to 900 Dhrystone MIPS.

The 24K core family implements the MIPS32 Release 2 architecture, which includes vectored interrupts and shadow register sets to minimize and bound hardware interrupt response times. These features support real-time applications. The cores also feature a single-issue pipeline, a decoupled 32x32-bit integer multiplier, and an aggressive memory subsystem.



At the core of the user experience.™

The 24K® Core Family

Block Diagram

- **Instruction Fetch**
  - Instructions read from L1 cache
- **Register File**
  - Operands read from register file
- **Execution**
  - Fixed point pipeline
  - Memory pipeline
- **Write-back**
  - Update register file with new results

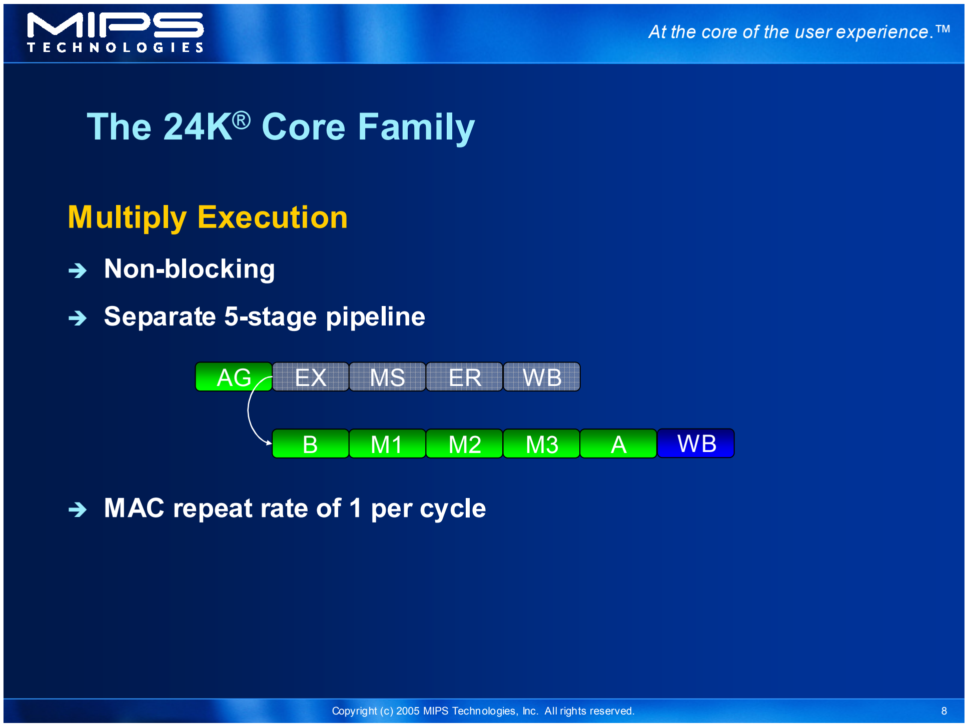
Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 7

Figure 2: The 24K core family features an eight-stage pipeline divided into four sections

The pipeline of the 24K core family is eight stages deep, logically divided into four sections. The fetch unit operates autonomously from the rest of the machine, decoupled by an eight-entry instruction buffer. The processor reads two instructions from the I-cache (instruction cache) each cycle, allowing fetches to proceed ahead of the execution pipeline. Speculation accuracy is improved with a branch history table, holding 512 bimodal entries, and a four-entry return-prediction stack. A full cycle is allocated for the I-cache RAM access, with the cache hit/miss determination and way selection occurring in the following stage.

One cycle is allocated to read operands from the register file and collect other bypass sources. Separate execution pipelines handle integer and load/store instructions. For memory operations, address calculation occurs in the AG stage. Next, the processor reads the data cache. Like the instruction cache, the D-cache (data cache) is four-way set-associative, may range in size from 16KB to 64KB, allows line locking, and supports optional parity protection. The MS stage performs hit calculation, way selection, and load alignment. The processor can accommodate up to four non-blocking load misses, allowing hits under misses. Normal ALU operations pass through the AG stage and do their real work in the EX stage. This skewed ALU preserves the two-clock load-to-use relationship common to many other MIPS cores. The exception-recovery stage prioritizes any exceptions. Finally, the write-back stage updates the register file and other instruction destinations with new results.

In the block diagram in Figure 2, blocks with dotted line borders are optional. An optional floating-point unit is available on the 24Kf™ core. An optional CorExtend™ interface is available on the Pro Series® versions of the cores.



At the core of the user experience.™

The 24K® Core Family

Multiply Execution

- Non-blocking
- Separate 5-stage pipeline

AG EX MS ER WB

B M1 M2 M3 A WB

- MAC repeat rate of 1 per cycle

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 8

Figure 3: The 24K core family features non-blocking multiply instructions and a repeat rate of one multiply-accumulate instruction per cycle.

Multiply instructions are non-blocking in the pipeline of the 24Kc core; subsequent instructions that do not depend on the result of the multiply can proceed without delay. Depending on the multiply instruction, its result is written into either the HI and LO accumulator or a general-purpose register (GPR).

Multiply instructions execute in a separate five-stage pipeline. Execution starts in the Booth recoding (B) stage [4,5] which corresponds to the EX stage of the normal integer pipeline. Then, the operation propagates through the multiplier array during the M1, M2, and part of the M3 stage. The multiplier array produces a number in carry-save format [5,6]. A final addition to convert this number to two's complement format is performed in the latter part of the M3 stage. The result is available at the beginning of the A stage, which is used to select between the multiply data path and other sources for the accumulator value. The processor writes the new result into the accumulator at the clock edge between the A and WB stages.

The Multiply/Divide Unit, or MDU, achieves a repeat rate of 1 multiply-accumulate instruction

(MAC) every cycle. (More details about multiplication will follow later in this paper).

The MIPS32® DSP ASE

The MIPS32 DSP ASE consists mainly of new instructions that execute in the integer and multiply pipes of the processor. The ASE includes all the typical DSP features such as register-based SIMD instructions that can add, subtract, shift, or multiply. The SIMD instructions operate on as many as four operands simultaneously and support operands of eight, 16, and 32 bits. The ASE also provides fractional arithmetic with saturation and rounding. There are several flavors of multiply-accumulate, including dot-product-accumulate. Precision expansions and reductions enable scaling operations. Absolute, bit-reverse, and other instructions enable common DSP operations to be performed efficiently. The ASE adds three new accumulators to the architecture for a total of four.

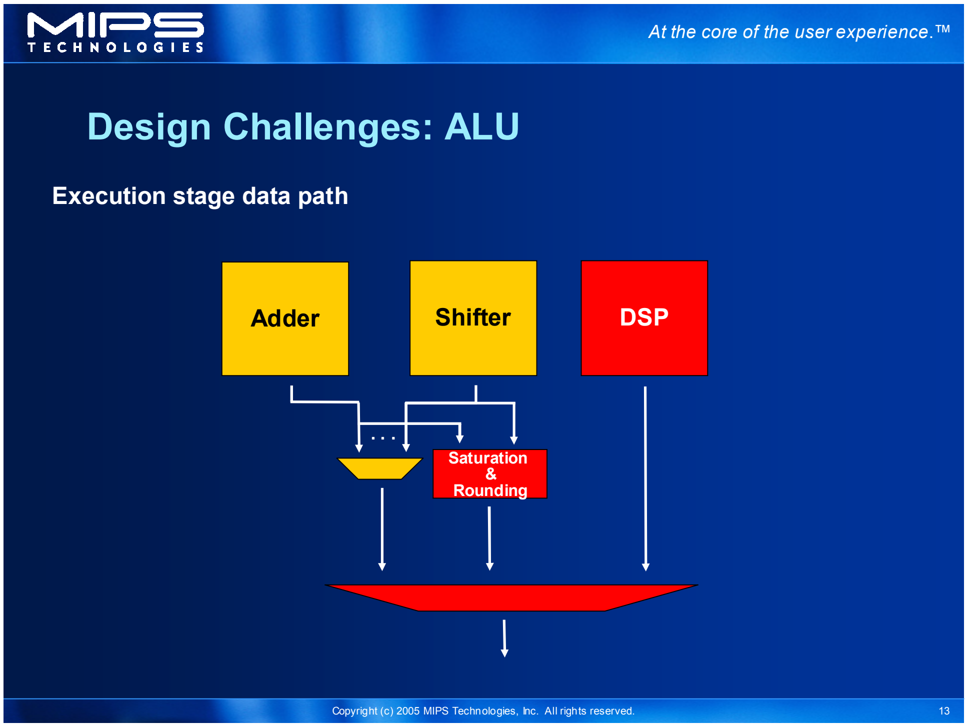
In addition to all the commonly found operations, the ASE also adds some advanced features that attain extra performance without requiring a complex implementation. An example is a variable bit extract method that efficiently extracts bits from an incoming stream. Another feature efficiently processes complex numbers. The ASE also includes a novel and efficient way to support virtual circular buffers.

To minimize implementation cost, new state elements are limited to a DSP control register and the mentioned three new accumulators.

Design Challenges

Implementing the DSP ASE with negligible cost is the goal of the 24KE core family, which includes the 24KEc, 24KEf, 24KEc Pro, and 24KEf Pro cores. A customer can choose the appropriate core depending on whether an optional FPU and/or an optional CorExtend interface is needed. In describing the design challenges for this core family, we will focus on the 24KEc core.

Specifically, the goal is to implement the DSP enhancements without significantly impacting the speed or die area of the core. There are two major challenges. First, the single-cycle ALU execution path already has critical timing. Inserting any additional logic level in the data path would increase the cycle time. Second, many additional features are required in the multiply data path. Support for these features must be added without increasing the cycle time in any of the multiply pipeline stages. It is preferable to do this without adding another stage to the multiply pipeline. Also, with the wider data path in the MDU, it is even more important to minimize the area impact.



MIPS TECHNOLOGIES

At the core of the user experience.™

Design Challenges: ALU

Execution stage data path

Adder

Shifter

DSP

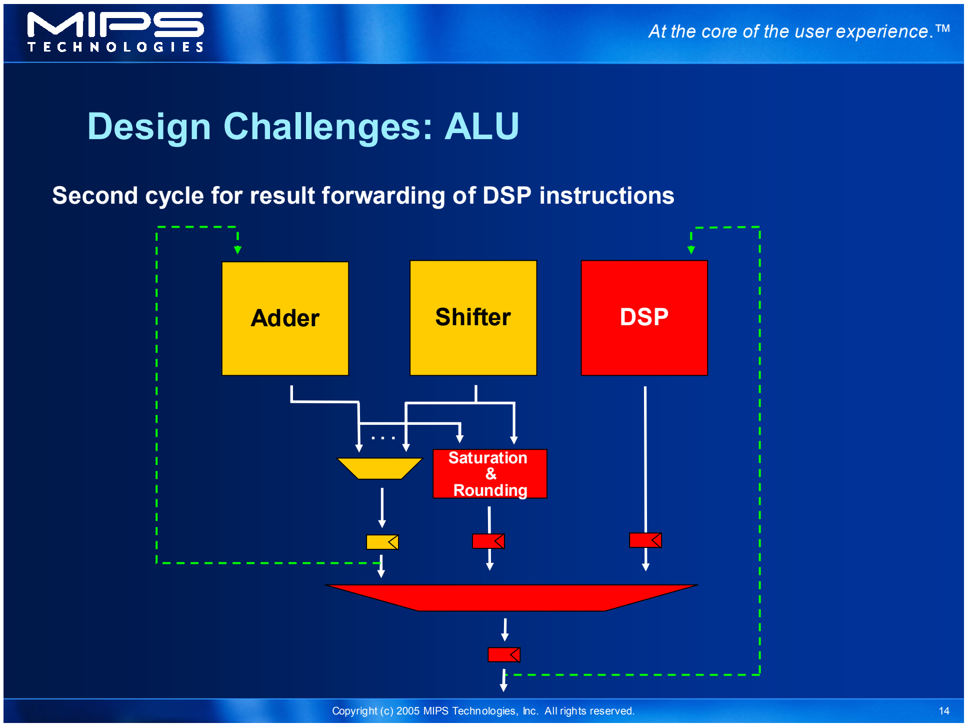
Saturation & Rounding

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved.

13

Figure 4: The execution stage data path has critical timing in 24KEc core.

The single-cycle execution of the ALU makes the timing of its data path critical. The most critical paths are through the 32-bit adder and shifter. In Figure 4, yellow blocks indicate existing logic in the 24Kc core. Red blocks indicate new DSP logic. The DSP enhancements require saturating the output of the adder and rounding the output of the shifter. Furthermore, the results from these and other DSP instructions require selecting from new sources so we need to add more mux levels. However, the additional saturation and rounding logic plus the additional mux levels would significantly increase the cycle time.



MIPS TECHNOLOGIES

At the core of the user experience.™

Design Challenges: ALU

Second cycle for result forwarding of DSP instructions

Adder

Shifter

DSP

Saturation & Rounding

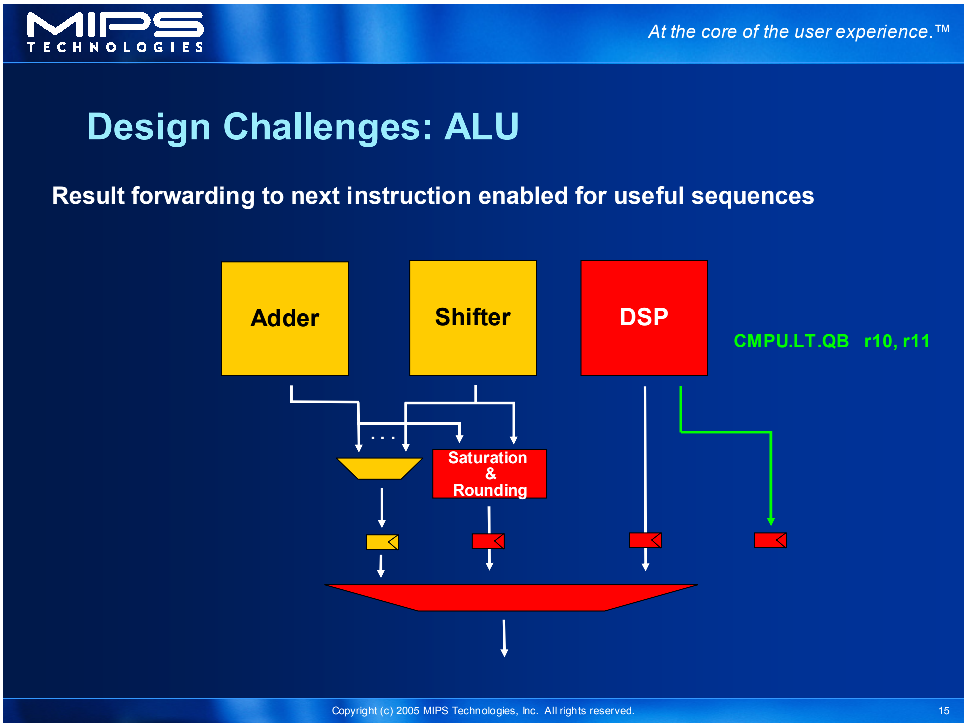
Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved.

14

Figure 5: A second cycle is used to forward GPR result of DSP instructions.

To avoid significantly increasing the cycle time, we added a second cycle to select between the existing ALU results and new DSP sources. Note that existing ALU operations are not affected; their results are still forwarded to a dependent instruction for execution the next cycle. Therefore, if the first instruction of a back-to-back sequence is not a DSP instruction, result forwarding can still take place.

If the first instruction is a DSP instruction, an additional cycle is required before forwarding the result. This one-cycle delay can usually be masked because of the vectorized nature of DSP code. Often, an instruction, such as a data load, can fill this delay slot between the two dependent instructions.



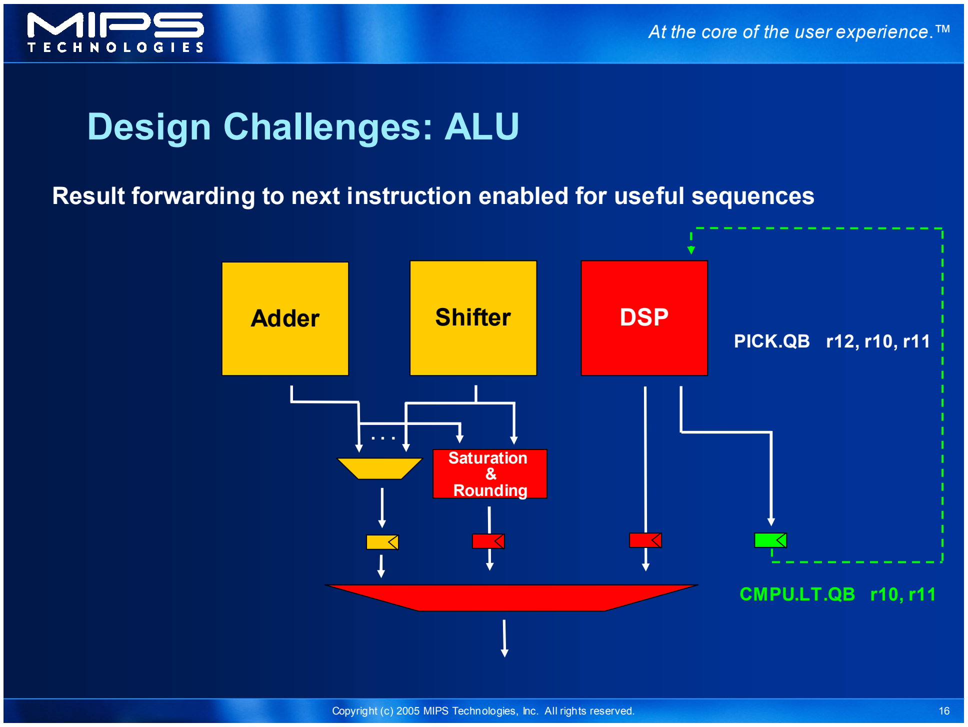
At the core of the user experience.™

Design Challenges: ALU

Result forwarding to next instruction enabled for useful sequences

Figure 6: A DSP compare instruction sets condition code bits.

However, there are some DSP instruction sequences in which it would be useful to forward the result to the next instruction. A compare followed immediately by a pick is such an example. In Figure 6, the first instruction is shown in green. It compares the corresponding bytes in register 10 and register 11. Based on the result of the comparison, it sets condition code bits in the DSP control register.



At the core of the user experience.™

Design Challenges: ALU

Result forwarding to next instruction enabled for useful sequences

Adder

Shifter

DSP

PICK.QB r12, r10, r11

Saturation & Rounding

CMPU.LT.QB r10, r11

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved.

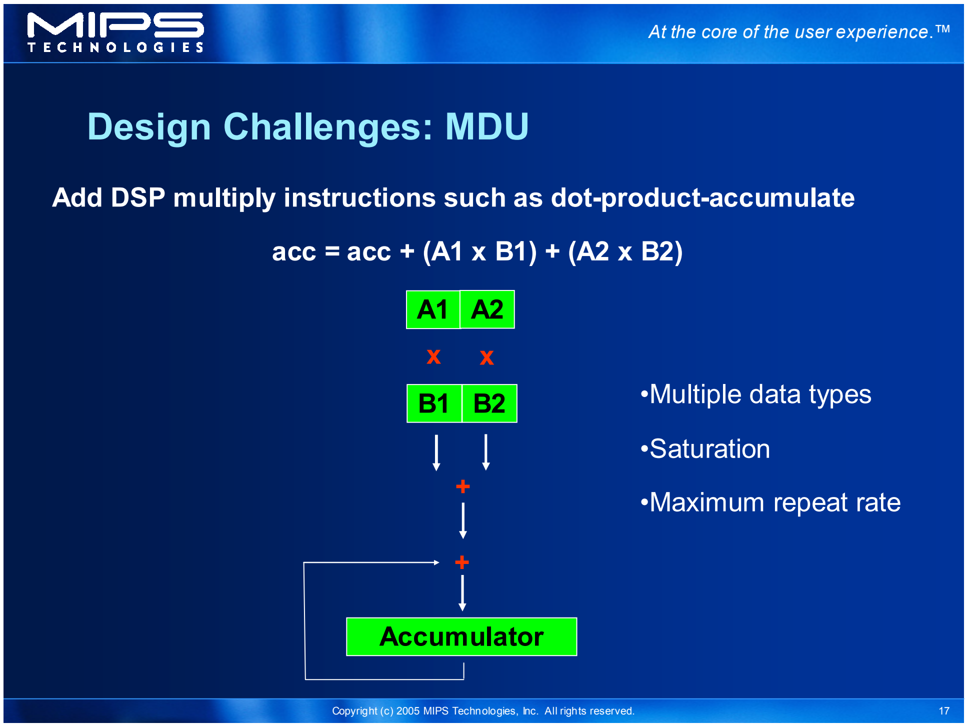
16

Figure 7: Result forwarding to the next instruction without a delay is possible for useful DSP sequences. .

In the next cycle, the second instruction advances to the execution stage. The instruction shown in Figure 7 picks each byte from GPR 10 or GPR 11 based on the corresponding condition-code bit. The result is four bytes, some of which may be from GPR 10 and some from GPR 11.

The pick instruction is dependent only on the condition-code bits, which can be forwarded from the previous cycle. Unlike DSP results in a GPR, the condition-code bits need not pass through the mux levels in the extra cycle. By using dedicated bits, useful sequences such as compare and pick can be executed back to back without delay.

By using such techniques, we were able to maintain a high clock frequency for the 24KEc core. We also minimized additional die area by reusing the ALU’s 32-bit adder and shifter. We did so without affecting the existing ALU instructions while enabling result forwarding for useful back-to-back DSP instruction sequences.



At the core of the user experience.™

Design Challenges: MDU

Add DSP multiply instructions such as dot-product-accumulate

acc = acc + (A1 x B1) + (A2 x B2)

- Multiple data types
- Saturation
- Maximum repeat rate

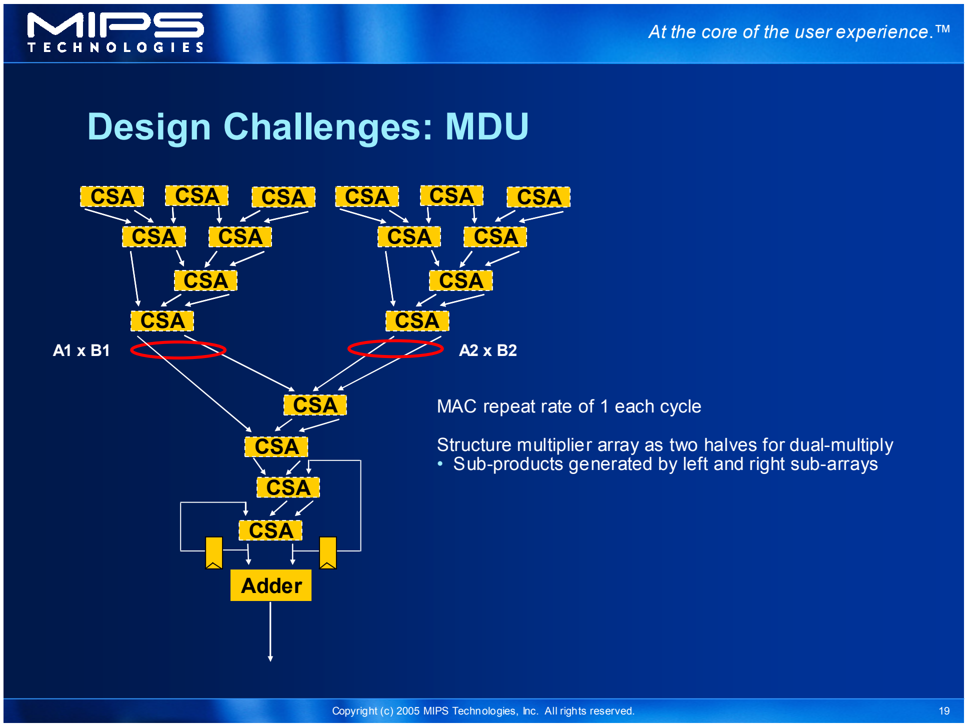
Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 17

Figure 8: The dot-product-accumulate instruction performs two multiplications and two additions.

Another design challenge was implementing the dot-product-accumulate instruction in the MDU. For this instruction, each half of a source GPR contains one Q15 operand. (Q15 is a 16-bit representation of a fractional number.)

This instruction performs two simultaneous multiplications of the corresponding halves from the two source registers. The two products are added to complete the dot-product operation. Then, that result is added to the content of the chosen accumulator. As illustrated in Figure 8, this instruction performs two multiplications and two additions.

Moreover, this instruction and other multiplies in the DSP ASE must support multiple data types and perform saturation. For optimal performance, it is desirable to maximize the repeat rate for as many types of MACs as possible.



MIPS TECHNOLOGIES

At the core of the user experience.™

Design Challenges: MDU

MAC repeat rate of 1 each cycle

Structure multiplier array as two halves for dual-multiply
- Sub-products generated by left and right sub-arrays

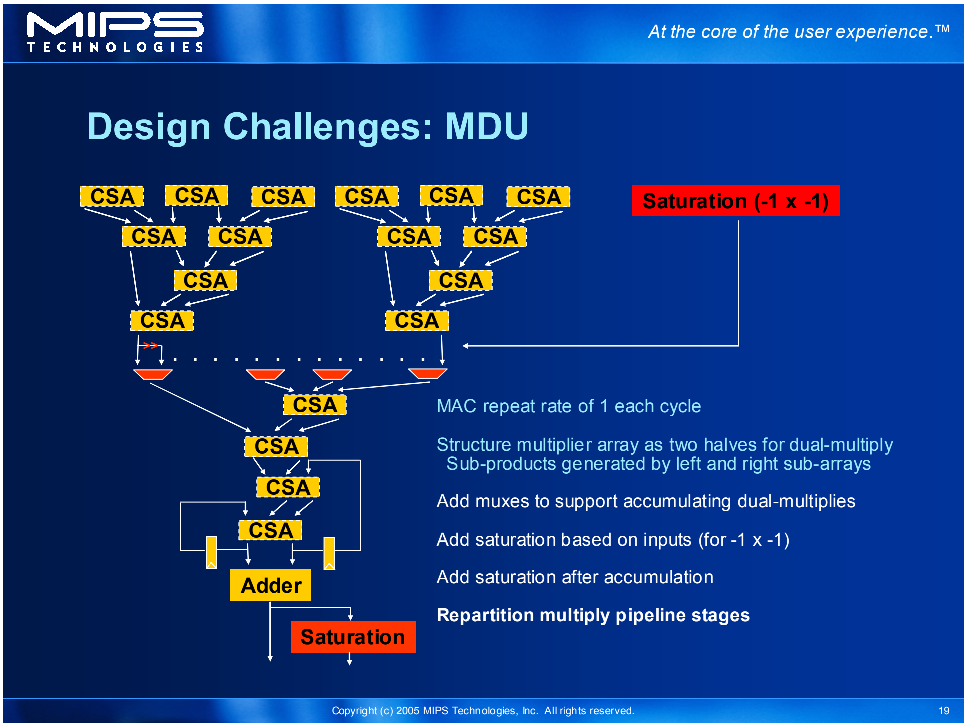
Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 19

Figure 9: The multiplier data path supports dual-multiply operations and a repeat rate of one MAC per cycle.

The multiplier array and adder make up most of the multiply data path. The multiplier array consists of rows of carry-save adders (CSAs). These rows of CSA blocks add the partial products of the multiplication. Each CSA adds three bits to produce two bits, a carry and a save. In Figure 9, each CSA block represents enough individual CSAs to operate on its input partial products. The final row produces two carry-save numbers, which must be combined using a 64-bit adder to convert them into the final two's complement result [5,6]. Not all staging registers are shown in the figure.

The final two carry-save numbers are forwarded through staging registers to the next instruction so that MACs can have a repeat rate of one per cycle. The data path forwards the accumulated value in carry-save format before it propagates through the 64-bit adder.

To support dual-multiply operations in the DSP ASE, the multiplier array is configured as two halves. For a single 32-bit multiplication, both halves function part of a common array. For dual-multiply operations, the left and right sub-arrays each yield a product in carry-save format. The dot-product-accumulate instruction requires these products to be added then accumulated. However, other dual-multiply instructions in the ASE simply require these products as results. For those instructions, the products are available at the point indicated by the red ovals on Figure 9. Not shown are the two 16-bit adders required to convert these carry-save values into two's complement numbers. These adders are small relative to the rest of the multiply data path and have no timing impact. They are omitted from the figure since they are not part of the multiplier array.



At the core of the user experience.™

Design Challenges: MDU

MAC repeat rate of 1 each cycle

Structure multiplier array as two halves for dual-multiply
Sub-products generated by left and right sub-arrays

Add muxes to support accumulating dual-multiplies

Add saturation based on inputs (for -1 x -1)

Add saturation after accumulation

Repartition multiply pipeline stages

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 19

Figure 10: Minimal additional logic was added to support DSP multiply instructions..

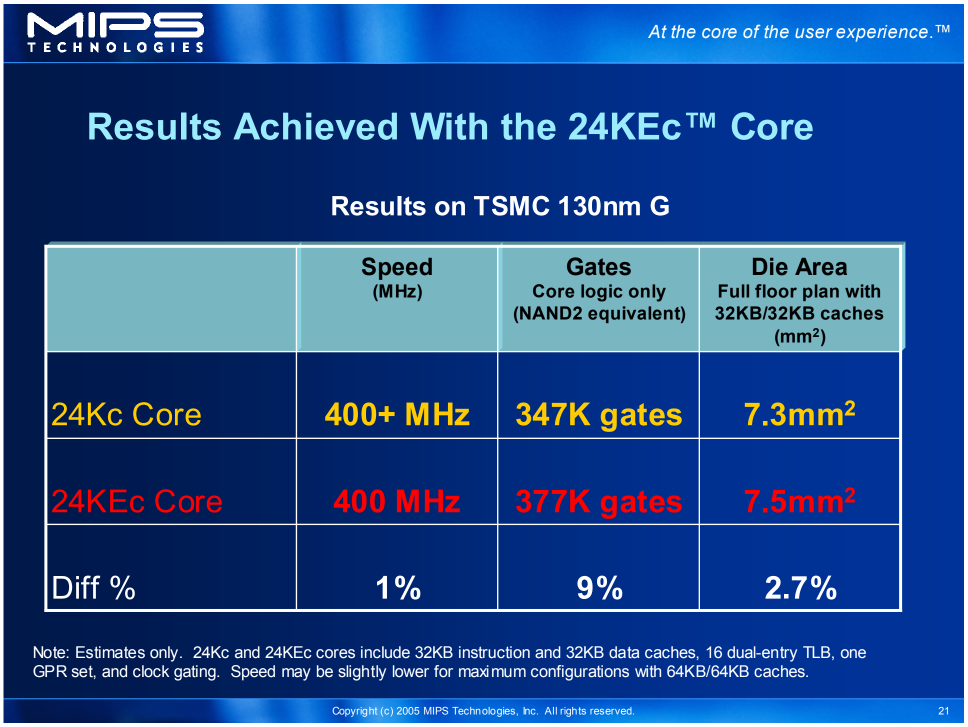
A row of muxes added to the left sub-array supports the dot-product-accumulate instruction described earlier. These muxes can shift the bits from the left sub-array so that they have the same significance as the bits in the right sub-array. This effectively aligns the two products so they can be added. The elegance of this approach is that we extensively reuse the existing logic. We use the left sub-array for the second multiply since it is not needed for 16-bit multiplication. Also, later stages of the multiplier array are used to perform the two additions for the dot-product-accumulate instruction. Therefore, the additional cost for this second addition is limited to a single level of muxes.

Saturation for the special case when both multiplier and multiplicand are –1 is performed in parallel with the multiplier array. Another level of muxes is required to substitute the products with the maximum fractional value. Additional logic after the 64-bit adder handles saturation after accumulation. Multiply instructions which saturate after accumulation have a repeat rate of one every two cycles. This is because final saturation occurs after the accumulated value would have been forwarded to the next instruction. All other MACs have the best repeat rate possible — one per cycle.

Note our high reuse of existing hardware as indicated by the large ratio of existing (yellow) logic to new (red) logic in Figure 10. We avoided adding any additional CSA level, and the added logic was small and introduced little delay. In the dot-product-accumulate instruction example, performing a dual-multiply, aligning the products, and adding them together required only one additional level of muxes in the multiplier array.

Our approach to redesigning the MDU allowed us to maintain a high clock frequency on the 24KEc core. We minimized additional die area by reusing the multiply data path to implement most of the DSP multiply logic, and we did so without affecting existing multiply instructions. Furthermore, the implementation includes DSP MACs which can be executed back-to-back for an optimal repeat rate.

Results Achieved



At the core of the user experience.™

Results Achieved With the 24KEc™ Core

Results on TSMC 130nm G

	Speed (MHz)	**Gates** Core logic only (NAND2 equivalent)	**Die Area** Full floor plan with 32KB/32KB caches (mm^2)
24Kc Core	**400+ MHz**	**347K gates**	**7.3mm^2**
24KEc Core	**400 MHz**	**377K gates**	**7.5mm^2**
Diff %	**1%**	**9%**	**2.7%**

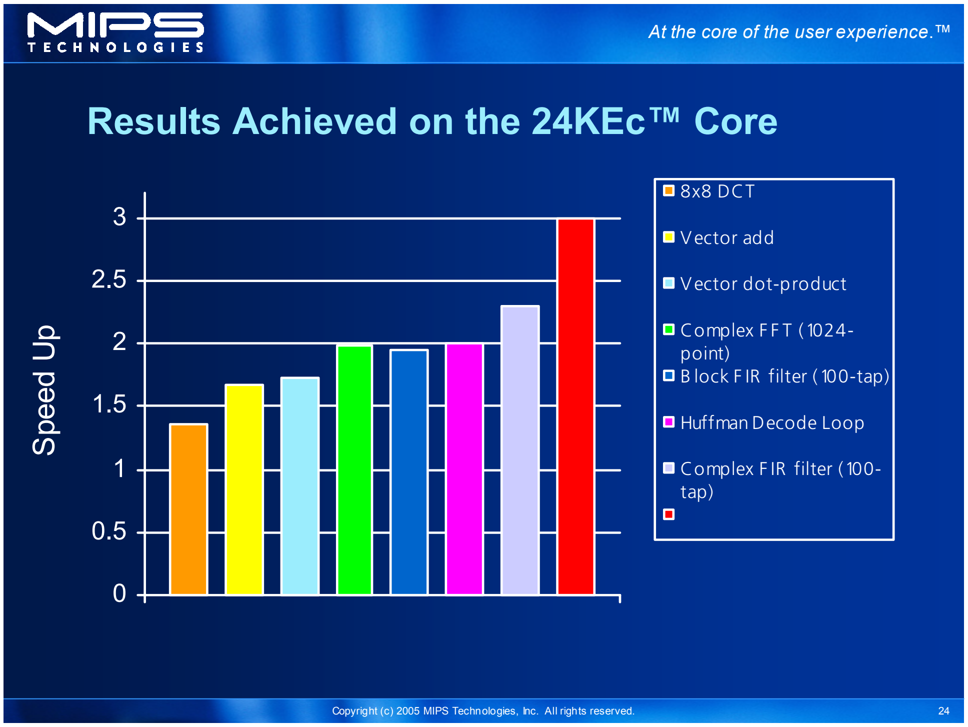
Note: Estimates only. 24Kc and 24KEc cores include 32KB instruction and 32KB data caches, 16 dual-entry TLB, one GPR set, and clock gating. Speed may be slightly lower for maximum configurations with 64KB/64KB caches.

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 21

Figure 11: The DSP enhancements were added with negligible effect on clock frequency and die area..

The key characteristics of the 24KEc core in the TSMC 130 nm G technology is shown in Figure 11. The 24Kc core numbers are also included for comparison. Both cores are capable of 400 MHz, worst case, using a high-speed library. The 24Kc core can achieve a slightly higher speed, but we target our synthesis runs at 400MHz. The additional DSP logic is about 9% of the core logic. This corresponds to an increase of only 2.7% of the total core die area with 32KB caches. Therefore, the DSP logic has negligible effects on processor core speed and total area.

At 400 MHz, the 24KEc core can reach 800 million multiply-accumulate operations per second, or MMACS, on 16-bit data. A wide range of DSP performance measurements are shown in Figure 12.


At the core of the user experience.™

Results Achieved on the 24KEc™ Core

Copyright (c) 2005 MIPS Technologies, Inc. All rights reserved. 24

Figure 12: The 24KEc core achieves DSP function code speedups of up to three times the performance of the 24Kc core.

To illustrate the enhancements of the 24KEc core, Figure 12 shows a range of DSP functions and their speedups using the new DSP instructions and state. These measurements are mostly for inner loops of functions so the overall application speedup will depend on the percentage of time these loops account for in the application. Of course, the MIPS architecture can execute general-purpose code in the rest of the application with inherently good performance. As Figure 12 indicates, the speedup ranges from a factor of 1.3 to 3. Note that the speedup numbers are comparing hand-optimized MIPS32 assembly code on 24Kc core (without using DSP instructions) and 24KEc core (using DSP instructions). These experiments were run on 24Kc and 24KEc core simulators.

Summary

The 24KE core family delivers accelerated DSP performance at a negligible cost. Simulation results show that the 24KEc core significantly improves DSP performance over the 24Kc core. The 24KEc core demonstrates that such meaningful DSP enhancements can be made to a RISC processor core with negligible effects on clock speed and die area. The success in overcoming the design challenges of implementing the 24KEc core further confirms the viability of moving digital signal processing onto the main processor core.

Merging digital signal processing onto the main processor also offers other benefits. The combined processing on a single processor creates a more efficient system architecture and results in lower area and power. The common tool chain for general and DSP code lowers software development costs and shortens software development time. All these factors make the 24KE core family ideal for embedded applications requiring a high-performance low-cost synthesizable microprocessor with accelerated DSP performance.

The 24KE core family is already available for licensing to early adopters and will become generally available later this summer. Processor cores from MIPS Technologies Inc. have traditionally been used as the host processor in embedded devices. But MIPS cores, specifically the 24KE core family, can run digital signal processing applications very efficiently and hence are well-suited to meet the convergence of RISC processors and DSPs to yield System-on-Chip integration benefits.

References

[1] W. Strauss, from correspondence.
[2] W. P. Hays, “DSPs: Back to the Future,” *ACM Queue*, pp. 42-51, Mar. 2004.
[3] J. L. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufman, San Francisco, CA, 1996.
[4] A. D. Booth, “A Signed Binary Multiplication Technique,” *Quart. J. Mech. Appl. Math.*, vol. 4, pt. 2, pp. 236-240, 1951.
[5] V. C. Hamacher, Z. G. Vranesic, and S. G. Zaky, *Computer Organization*, McGraw-Hill Book Company, San Francisco, CA, 1984.
[6] C. G. Wallace, “A Suggestion for a Fast Multiplier,” *IEEE Trans. On Electronics Computers*, vol. EC-13, Feb. 1964, pp. 14-17.