Brief Introduction to MIPS32® M4K® Core Shadow Registers for Microcontroller Applications

Document Number: MD00656
Revision 01.00
October, 2007

MIPS Technologies, Inc.
1225 Charleston Road
Mountain View, CA 94043-1353

1 Introduction

The RISC architecture brings many advantages to the microcontroller space including the fact that compared to CISC architectures it is much friendlier to compilers and very economical in silicon area. For silicon designers and their target customers, this translates to high performance at low cost.

There are, however, some additional considerations for the end user when using RISC-based processors, especially at the lower frequencies found in the general purpose microcontroller space.

One of these issues is related to the context save and restore sequence during an interrupt or exception handling routine. The context save and restore sequence in a RISC-based machine can potentially be very cycle-intensive, given that at least 16 if not 32 general purpose registers (GPRs) must be pushed to or popped from a stack or static storage space somewhere in RAM. In some cases, it is only after a significant amount of clock cycles that the core is ready to address the actual interrupt request.

Designers can take shortcuts by not saving all of the registers—either by writing the entire application in assembly language or instructing the compiler to use only certain registers during code translation. Either way, this adds additional constraints to the software design since the software developer must be aware of these requirements at all times.

The MIPS32® M4K® RISC core is no different than other RISC cores in this respect in that it has 32 GPRs and an application binary interface (ABI) that uses the full range of the 32 GPRs as defined by the MIPS32 Release 2 architecture.

However, the M4K core provides a more elegant solution in the form of GPR shadow register support, greatly reducing the interrupt response overhead and ultimately matching the highly responsive and deterministic structure of CISC-based microcontrollers. It offers the best of both worlds.

2 M4K Core Shadow Register Sets: Basic Introduction

A new feature in the MIPS32 Release 2 architecture greatly streamlines the ability to handle interrupts and exceptions. Processors implemented using the MIPS32 Release 2 architecture, including the M4K core, allow for up to eight copies of the GPRs to be present within the core itself. This feature is a build-time option that creates an advanced interrupt control structure not usually found in the microcontroller space.

Now chip designers can decide to balance speed against total core area, creating one, two, four or eight copies of the GPRs. By definition, shadow set 0 is the default GPR set, such that up to seven additional shadow sets are available exclusively for interrupt service tasks.

The additional GPRs are completely isolated from each other and normal memory space, except in the cases mentioned below. Each GPR set can be associated with one or multiple interrupt vectors, depending on the application.

When processing an interrupt or exception, the M4K core will determine which shadow set is to be used based on the values the designer sets in specific control registers. It then establishes the specified shadow register set as the active set of GPRs, allowing the interrupt vector to continue execution. This process completely eliminates the need for any context save or restore cycle, since the specified interrupt service routine is the sole owner of the shadow register that is currently active.

Not only does this mean that no time is wasted before the interrupt or exception code can begin actual implementation, but it also means that the content of the registers has been preserved since the last exception or interrupt event was active. This saves time on retrieving specific values from the SRAM space.

Following is the basic procedure for initialization of an interrupt controller scheme on the M4K core. It retains the standard steps of any general interrupt system initialization procedure, with the addition of one important new step:

1. Standard: Set base vector table address
2. Standard: Fill interrupt table with active vector addresses
3. Standard: Fill non-active vector table addresses with stub addresses
4. **M4K Specific: Assign vector specific IDs to associated GPR register set**
5. Standard: Clear all active and pending interrupt sources
6. Standard: Enable all active interrupt sources
7. Standard: Enable global interrupts

3 M4K Core Shadow Register Sets: Basic Operation

A detailed description of the entire implementation of shadow register sets is beyond the scope of this document; such a description can be found in the paper, “MIPS32® M4K® Processor Core Implementor’s Guide,” available through MIPS Technologies. The basic operation is summarized as follows.

Figure 1 illustrates the basic concepts involved with an M4K core built with a total of four register sets. It must be noted that these four register sets are completely contained within the processor core and require no external RAM to implement. Also indicated in the diagram are two key control registers, SRSCTL and SRSMAP, used in the configuration and implementation of the shadow GPR logic.

Figure 1 MIPS32® Vectored Interrupt / Shadow Register Structure

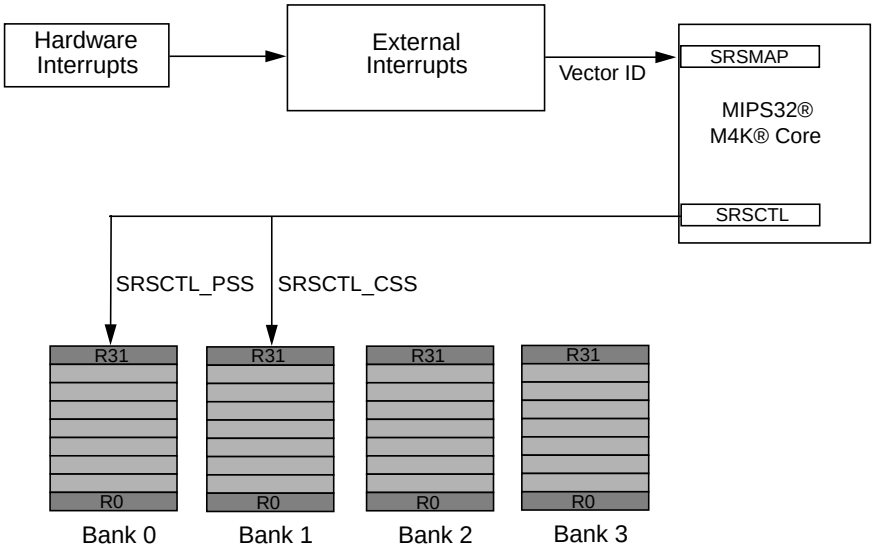

Figure 1 indicates the state of the core just before execution of the interrupt service routine is about to begin:

- A hardware interrupt of a specific source has been acknowledged by the external interrupt controller.
- The external interrupt controller has passed this vector ID onto the M4K core. For this example, we will assume the vector ID is 1.
- The core uses this vector ID to determine the target GPR set to use. The M4K core does this by the configuration set in the SRSMAP, which contains shadow GPR IDs for Vector ID 0 through Vector ID 7.

The M4K core then uses this information to perform the following:

- The current GPR ID is copied into the SRSCTL_PSS field of the SRSCTL control register. This is the Previous Shadow Set ID to be recovered later when the interrupt or exception request is completed.
- The GPR set ID pulled from the SRSMAP control register is copied into the SRSCTL_CSS field of the STSCTL control register. This now specifies the Current Shadow Set.
- Execution of the interrupt or exception service routine proceeds to completion and an ERET instruction is executed.
- The M4K core then copies the values from the SRSCTL_PSS bit field into the SRSCTL_CSS bit field and execution resumes from the point at which the interrupt or exception was issued.

There is a clear advantage of this process over the standard-context save and restore procedure: the M4K core wastes no time in setting up a GPR environment that allows it to begin processing the interrupt service request or exception request almost immediately. Using the shadow GPR function of the M4K core, the interrupt latency is reduced to only 22 cycles, or a total of 550 ns.

4 M4K Core Shadow Register Sets: Finer Grain Control

During normal execution of user-space code on the M4K core, each copy of the GPR set remains invisible to the processor core unless its specific ID is active. However there are some special processor instructions that allow the M4K core, when executing in kernel mode, to access any given register set. RDPGPR and WRPGPR instructions allow a privileged execution thread to access any and all GPR shadow sets at will. This may be required when a specific GPR set is dual-purposed between two discrete but similar interrupt vectors. Knowing how the compiler ABI places the active data into the GPR sets would allow a user to pick and choose specific GPR values for safekeeping, depending on the specific interrupt vector that is executing. An example of where this would be useful is in the case of two Ethernet devices that share the same register structure, but are processing different TCP/IP streams.

5 M4K Core Shadow Register Sets: Benefits in the Microcontroller Space

Use of the shadow set feature of the M4K core allows system designers to maximize the interrupt response time to external events, even when dropping the processor core speed to save power.

The compromises of the software design now shift from dealing with the inherent waste of clock cycles needed to preserve the current state of the system to deciding which interrupt sources deserve their own shadow register sets and which ones can share a dedicated GPR set between them. This is a much easier challenge.

The shadow GPR system is straightforward in initialization and operation with most of the heavy work happening transparently to the executing code. This allows software designers to concentrate on the efficiency of the actual interrupt service routine rather than on system housekeeping. By taking advantage of shadow GPRs, the M4K core can greatly reduce the interrupt response overhead and offer a highly responsive, deterministic solution for 32-bit microcontrollers.

Template: nW1.03, Built with tags: 2B