MIPS R4400 Features

May 27, 1993

MIPS Technologies Inc.
2011 N Shoreline Blvd
PO Box 7311
Mountain View, CA 94039-7311

R4400 Features

Following is a quick description of the features on the R4400 and the visible changes. This information will be reflected on the update of R4000 User's manual.

1. In addition to divide by 2,3 and 4, divide by 6 and 8 have been added. TClk for div6 & 8 has its rising edge aligned in the same place as during all other divide by modes. The waveforms for the divide by 2, 3 and 4 clocks are found in the interface spec. The new AC and DC specs will also contain this information. The modebits to set the interface are as follows:

Mode bits 15:17

value

0: div2

1: div3

2: div4

3: div6

4: div8

5-7: reserved

The SysAd interface divide ratio is also visible in the Config register in bits EC[28:30]. The value of EC is the same as in the chart above.

2. A new bit was added to CacheErr register to indicate if a cache error occurs while handling external requests. These errors get masked if the processor is executing in an exception handler. In such situations, cache coherency may not hold.

To fix this problem, a new bit was introduced called EW, in the CacheErr register. The description of this bit follows:

EW is the 23rd bit in CacheErr register.

EW bit indicates that a MP critical cache error occurred, but the CacheErr register is not updated, as it was busy holding the values of previous cache error.

EW bit could be set any time during the period starting from the time the CacheErr register is initially loaded due to an error until an ERET is executed. Once EW is set it can be cleared only by Reset.

Following errors can set EW bit,

1. SCache Tag errors due to an external request handling.
2. SCache Data errors due to an external update handling.
3. PCache Tag errors due to an external request handling.

Additional note for case 1:

Case 1 covers only multi bit tagerrors. The bit is set for all external requests including interventions and snoops. The reason this was done is because the system would reset the R4000 if it saw the snoop/intervention response having erroneous data bit set. Therefore, for consistency reasons, this is done for all external requests.

3. The R4400 has an uncached store buffer to improve the performance of uncached stores over the previous R4000.

On the R4000, when an uncached store reaches the WB stage of the pipeline, the CPU stalls until the uncached store has been sent out of the chip.

On the R4400, there is a single entry uncached store buffer. This means when an uncached store reaches the WB stage of the pipeline, the store drops into the uncached store buffer and the CPU pipeline may continue to run. If another uncached store reaches the WB stage of the pipeline before the first uncached store has been sent out of the chip, the CPU will stall until the uncached store buffer has completed the first uncached store.

If other useful instructions can be scheduled between successive uncached stores, then considerable performance improvement may be achieved using this mechanism.

To avoid CPU stalls, for sequential stores, there needs to be 7 cycles between uncached stores:

```
        sw r2, (r3) # uncached store
1:      nop
2:      nop
3:      nop
4:      nop
5:      nop
6:      nop
7:      nop
        sw r2, (r3) # uncached store
```

Remember that if any of the instructions between the stores are multiple cycles then the number of instructions between the stores can be reduced further without causing stalls (for example, variable shifts take 2 EX cycles)

If a sequence of uncached stores are executed in a loop, then the 2 killed cycles that are lost in branch latency still count towards the 7 cycles needed between the stores:

```
loop:   sw r2, (r3) #uncached store
1:      nop
2:      nop
3:      nop
4:      bloop
5:      nop
6:      killed
7:      killed
```

In this case only 4 instructions have to be scheduled (excluding the branch).

The latency between stores is really defined by the fact that back to back uncached stores on the system interface can be sent out at a maximum rate of one store every 4 external cycles (Address, Data, X, X, Address, Data...). This translates to 8 pipeline cycles for a chip running the system interface in divide by 2 mode. If a larger clock divisor is used (div3, div4), then the number of cycles between successive uncached stores goes up by the same ratio (div3 11 pipeline cycles, div4 15 pipeline cycles) if you want to avoid CPU pipeline stalls.

The SYNC instruction is used by the R4400 to ensure that any uncached store in the uncached store buffer is sent out of the chip before any load or store after the SYNC completes.

As external requests have higher priority than pending uncached stores, it is possible for an external agent to see the R4400 complete cached and uncached stores out of program execution order. For example, given the sequence:

```
sw r2, (r3) # uncached store
sw r4, (r5) # cached store
```

Given an external intervention or snoop is sent into the chip when the uncached store has entered the uncached store buffer, but has not yet been sent out of the chip, and the cached store has run successfully through the TC stage (i.e., it has hit in the primary cache). The external agent will see the state of the internal caches after the cached store but before the result of the uncached store is seen outside the chip. The SYNC instruction may be used to avoid this case if necessary, as shown below:

```
sw r2, (r3) # uncached store
SYNC
sw r4, (r5) # cached store
```

4. The primary caches on the R4400 are 16K for both instruction and data. This change is visible in the Config register in bits DC[6:8] and IC[9:11] which indicate the primary cache size according to the formula 2**(12+DC) and 2**(12+IC).

5. The status pins function in the R4400. The encoding is as follows. Note that the encoding is different than the 2.9.1 version of the interface specification.

Status<7..4> or Status<3..0>	Processor Internal State
0	Run cycle: Other Integer Instruction (not Load/Store/Conditional Branch)
1	Run cycle: Load
2	Run cycle: Untaken Conditional Branch
3	Run cycle: Taken Conditional Branch
4	Run cycle: Store
5	Reserved
6	MP stall
7	Run cycle: Integer instruction killed by slip
8	Stall cycle:Other stall type

9	Stall cycle:Primary instruction cache
a	Stall cycle:Primary data cache
b	Stall cycle: Secondary cache
c	Run cycle: Other FP Instruction (not Load/Store/Conditional Branch)
d	Run cycle: Killed by Branch/Jmp/ERET
e	Run cycle: Instruction killed by exception
f	Run cycle: Floating-point insstruction killed by slip

6. Master/Checker operation is also fully functional. The application by Ali Moayedian describes the operation of Master/Checker. These features can be enabled by Boot Time Modebits SIMasterMd (bit 18) and SCMasterMd (bit 42).

Master/Checker Mode in the MIPS R4400

Two R4400's can be configured as a lock-stepped pair to improve data integrity. For such configurations, outputs (and I/O busses during output) are checked between the two chips connected in parallel at bus cycle boundaries. Any discrepancies between them are reported through the assertion of the Fault* pin.

Two boot-time mode bits determine these Master/Checker configurations. For single-chip operations, these bits must be set to 00. This is the "Complete Master" mode, where the R4400's outputs operates normally.

There are two possible lock-step configurations. The simple case pairs a "Complete Master" R4400 (with mode bits set to 00) with a "Complete Listener" R4400, with mode bits set to 11. The latter has output drivers disabled. Since it receives all the same inputs as the former, both R4400's operate identically. On all output cycles, it samples the signals on output and I/O busses and compares with what it expects. Any discrepancies are reported through the Fault* pin.

The second lock-step configuration is called Cross-Coupled Checking. In general, one R4400 is set to drive the data pins of one bus and the other the check pins (parity or ECC) of the same bus. The mode bits to use are 01 and 10. Both chips sample and compare what are on the busses and indicate any discrepancies through their Fault* pins.

Additionally and for any configuration, the Fault* pin also reports error conditions not covered by R4400 exceptions. (These include input parity error at SysCmd, input addresses and data on SysAD for certain system interface operations.) Note that the fault detection mechanism related to this Fault* pin does not cause any exceptions for the R4400. It continues to run regardless of the state of the pin. External logic must act on the fault and take appropriate action to interrupt or reset the R4400's.

Connections

For lock-step operations, most pins of a pair of R4400's are intended to be connected in parallel. The following discussion refers to signal groups as defined in Chapter 8 - Signal Descriptions.

The signal groups to be connected in parallel are:

* System Interface

* Secondary Cache Interface

* Interrupt Interface

Some signals in the following groups are NOT to be connected in parallel:

* Initialization Interface

ModeClock, ModeIn and Reset*

* JTAG Interface

JTDO, JTMS

The remaining signals in these groups can be connected either way, independently or in parallel, according to other considerations.

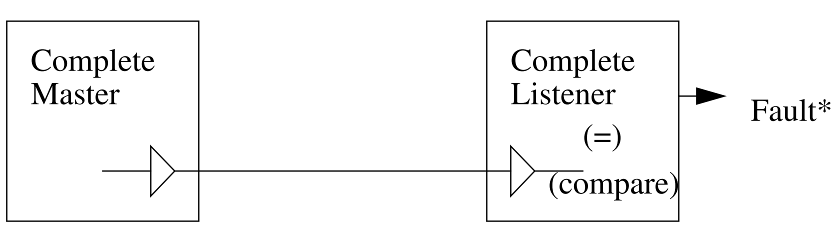
Finally, all of the signals in the Clock/Control Interface are NOT to be connected in parallel, except obviously VccP and VssP.

Modes of Operation

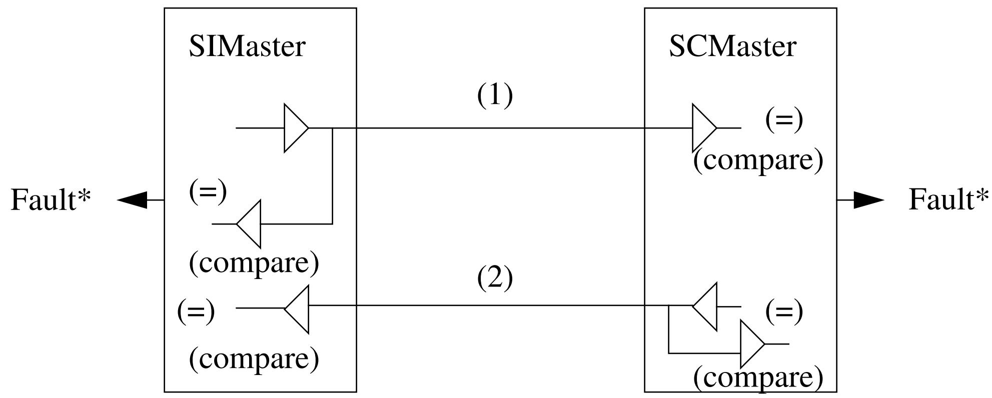
The Master/Checker mode of operation for each R4400 is set via boot-time mode bits: System Interface Master (SIMasterMd) and Secondary Cache Master (SCMasterMd). The following are defined:

SIMasterMd	SCMasterMd	Modes
0	0	Complete Master (required for single chip operation)
1	1	Complete Listener (to pair with Complete Master)
0	1	System Interface Master (SIMaster)
1	0	Secondary Cache Master (SCMaster, to pair with SIMaster)

Schematically, the Master-Listener configuration operates as follows:



And the Cross-Coupled Checking configuration operates as follows:



(1) Signals connected in parallel and driven from the SIMaster are:

SysAD(63:0), SysCmd(8:0) and SCAPar(2:0)

(2) Signals connected in parallel and driven from the SCmaster are:

SysADC(7:0), SysCmdP, ValidOut*, Release*,

SCAddr(17:1), SCAddr0(w,x,y,z), SCOE*, SCWr(w,x,y,z)*,

SCData(127:0), SCDChk(15:0), SCTag(24:0), SCTChk(6:0),

SCDCS*, SCTCS*

Fault Detection

Output miscomparison fault detection occurs at the end of bus cycles. The length of these cycles are determined by the appropriate boot-time mode bits.

At the System Interface, outputs are checked at the end of every System Interface cycle whenever the R4400 System Interface is in master state.

At the Secondary Cache Interface, outputs are check at the end of every read or write cycle.

A special note about the SCAPar(2:0) signals. They are delayed from the rest of the Secondary Cache Interface by exactly one PClock. This includes when they transition, and when they are checked. The transitions follow exactly one PClock after SCAddr transitions, and when the R4400 internally determines it is changing between a read and a write cycle without an address change. They do not follow the timing of the SCWr* signals, which are separately settable via the boot-time mode bits.

Fault Reporting

The external fault indication is reported through the Fault* pin. This is a non-persistent signal which operates on the same bus cycles as the System Interface. That is, it transitions at System Interface boundaries, as determined by the PClock-to-SClock divisor from the boot-time mode bits.

There is an internal fault detection latency of 4 PClocks. Then the fault signal must synchronize with the System Interface. Any output faults detected and fully-propagated through the internal fault logic within the previous System Interface cycle will be reported in the current cycle.

Note that in the Complete Master mode, output fault reporting is disabled for Secondary Cache Interface pins, but enabled for System Interface pins (SysCmd, SysCmdP, SysAD, SysADC, ValidOut* and Release*.)

Reset Operations

If the Master/Checker mode of the R4400 is Complete Listener, SIMaster or SCMaster, an assertion of Reset* is significant. On boot-up, the first time Reset* is deasserted, the Master/Checker mode is determined by the boot-time mode bits. On the next assertion of Reset*, the mode is forced to be a Forced Complete Master mode. Hence, if Reset* is deasserted, the R4400 will be driving all outputs. Upon yet another Reset* assertion, the R4400 returns to the previous mode as determined by the mode bits. This alternating behavior repeats on subsequent transitions of Reset*.

There is a slight difference between this Forced Complete Master mode and the normal Complete Master mode. That is, the Fault* pin continues to report all output faults when the R4400 was in the former mode, not just those of the System Interface.

Fault History Mechanism

Since both output faults and certain input faults are reported through the Fault* pin, there are two fault history bits that record which one or both of them has occurred. These bits are cleared at every deasserting transition of Reset*.

These bits are readable only when the Reset* is asserted. The Fault* pin changes from reporting live faults to indicating which fault history bit was set in the previous time period when Reset* was deasserted. The ModeIn pin acts as the selector. If ModeIn is 0, the Fault* shows the state of the Output Fault History bit (inverted). If ModeIn is 1, it shows the state of the Input Fault History bit (inverted).

There is also a mechanism to reset the fault history bits while the R4400 is running. If the ModeIn pin is asserted, these bits are cleared. This also means that for the duration when latching fault history bits are desired, ModeIn must be maintained at 0.

The following table summarizes some of the above information:

Boot/Reset Controls	ModeIn Pin	Fault History Bits	Fault* Pin	Master/Checker Mode
VCCOk just asserted (0 -> 1)	used as mode bits scan data	---	---	
Reset* just deasserted (0 -> 1)	---	cleared	---	

Boot/Reset Controls	ModeIn Pin	Fault History Bits	Fault* Pin	Master/Checker Mode
Reset* deasserted (normal operation)	0	set and latch if fault	live faults reported	
Reset * deasserted (normal operation)	1	cleared	live faults reported	
Reset* just asserted (1 -> 0)	---	---	---	changed, toggling between mode bits and forced master
Reset* just asserted (R4400 in reset)	0	Output Fault History bit connected to Fault* pin		
Reset* just asserted (R4400 in reset)	1	Input Fault History bit connected to Fault* pin		