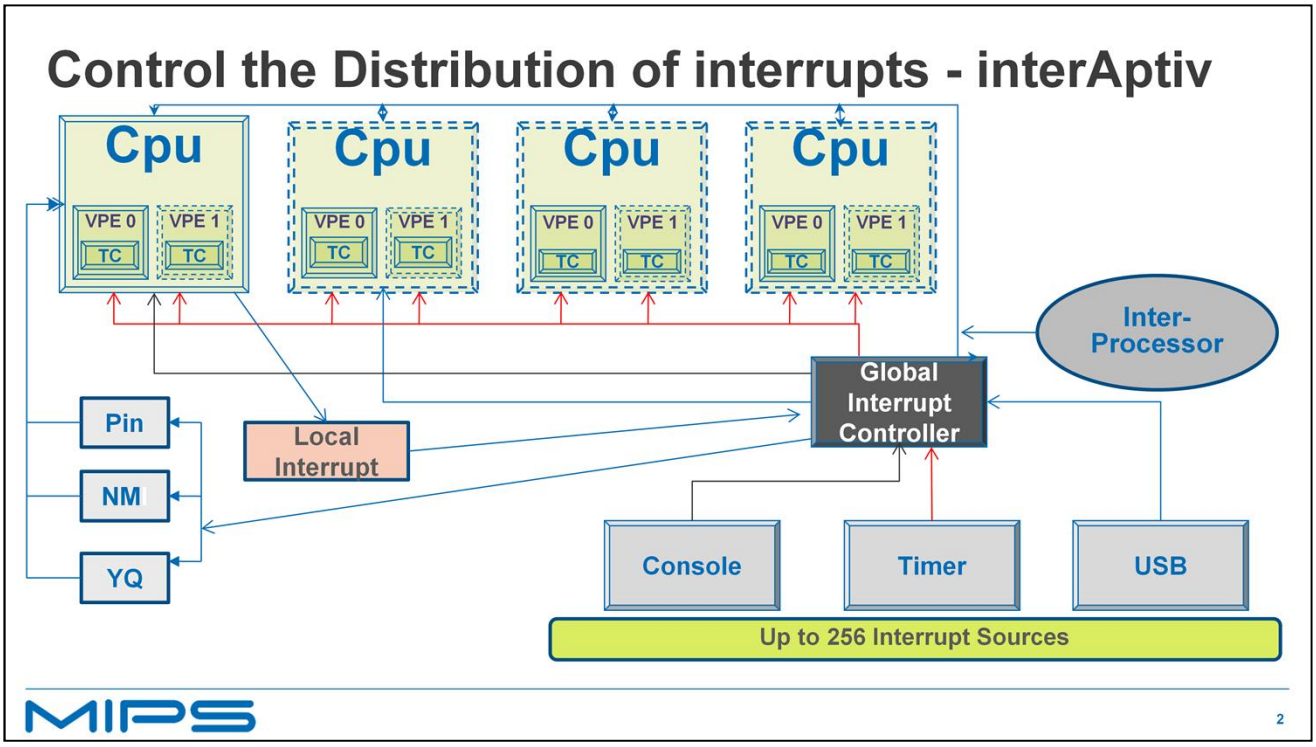
Our Coherent Processing systems can have an optional Global Interrupt Controller. This section covers its configuration and use.



The GIC handles the distribution of interrupts between the Processor Elements in the Multi core cluster. A processor element is a Virtual Processor Element or VPE in a multi processor system made up of MT cores or a single processor in a multi processor system made up of single non-MT cores.

+ Your system can have up to 256 external interrupt sources.

+ The GIC Distributes interrupt sources between the available processor Elements in the system. You can configure it to connect any interrupt source to any interrupt input which can be an Interrupt pin, an NMI pin or a Yield Qualifier pin on a MT Core.

+ It Allows any Processing Element to interrupt any other Processing Element.

+ It Routs local interrupts such as Timer, Watchdog, GIC Count/Compare, Performance Counters, and Software Interrupts.

to interrupt pin, NMI or Yield Qualifier on the local core.

It is Backward compatible with the legacy and vectored interrupt modes.

And it's able to integrate interrupt messages from peripherals such as HyperTransport and PCI-Express.

GIC Memory Mapped Register Interface

- **Programming the GIC Base Address**

Register Fields		Global Interrupt Controller Base Address Register (GCR_GIC_BASE Offset 0x0080)	Reset State
Name	**Bits**		
GIC_BaseAddress	31 - 17	Base address of the 128KB Global Interrupt Controller block	Undefined
GIC_EN	0	Setting to 1 enables GIC	0

- GIC Base Address programed by boot code example

```
#define GCR_CONFIG_ADDR     0xbfbf8000   // GCR registers base address
#define GCR_GIC_BASE        0x0080       // Offset of GIC Base address register from GCR base address
#define GIC_P_BASE_ADDR     0x1bdc0000   // physical address of the GIC

li      a1, GCR_CONFIG_ADDR + GCR_GIC_BASE      // Address of the GIC address register
li      a0, GIC_P_BASE_ADDR | 1                 // Physical address of the GIC + enable bit
sw      a0, 0(a1)                               // Write to GCR[GIC base address register]
```

MIPS

3

The interface to the GIC is through memory mapped registers. To get started using the GIC registers you must first know where they are in the memory map. The address of the GIC registers should be programmed by the boot code in the GCR GIC Base register. This register is located within the Global Configuration Register Block at offset 80 hex.

As you can see from the table the address is on a 128K boundary so the lower 17 bits will always be 0. This leaves space for additional information in the register. The GIC_EN field controls the enabling of the GIC. Once the boot code configures the GIC it should enable it by setting this bit.

GIC Address Segments

- GIC Address Map

GIC Address Segments	
Segment	Base Offset
Shared Section (External Interrupts)	0x0000
Local Section (Core local interrupts)	0x8000
Other Local Section	0xC000
User Mode section	0x10000

MIPS

4

Starting at the GCR GIC Base address mentioned in the previous slide, the GIC address space is accessed with uncached load and store commands. For each load or store command the hardware supplies the physical address and the number of the Processing Element of the requester. The Processing Element number is used as an index to reference the appropriate subset of the instantiated control registers. By using the Processing Element number information, the hardware writes or reads the correct subset of the control registers pertaining to the “local” Processing Element so your software does not need to explicitly calculate the register index for the “local” Processing Element – it’s done entirely by hardware.

The GIC is divided into sections:

The first section starts at the Base address of the GIC. This shared section is where the external interrupt sources are registered, masked, and assigned to a particular Processing Element and interrupt pin. This section is used by all Processing Elements.

Next is the local section which starts at the Base address plus 8 thousand hex. This is the section in which interrupts local to a Processing Element are registered, masked, and assigned to a particular interrupt pin.

The “local” Processing Element can access the local registers of another Processing Element by using the Other local address space. Software must write the Other Addressing Register before accessing these address spaces. The value of this register is used by hardware to index the appropriate subset of the control registers for the other Processing Element.

An additional section called the User-Mode Visible section is used to give quick user-mode read access to specific GIC registers. The use of this section is meant to avoid the overhead of system calls to read GIC resources, such as counter registers.

GIC Shared Section

- **All shared registers are visible to all Processing Elements**
  - Kernel mode access only
  - Can be read or written by any Processing Element
  - Configuration usually done at boot time by Processor Element 0

MIPS

5

I'm going to start with the Shared section of the Global Interrupt Controller. This section of registers can be accessed by all Processing Elements while in kernel mode. Most of the registers in this section are used to configure the Global Interrupt Controller. Usually you would do this configuration in your boot code using Processing Element 0. In most cases, Other Processing Elements only access these registers for status information. The names of the registers in this section all start with GIC underscore the SH to indicate they are Global Interrupt Controller Shared Section registers.

GIC Config Register

Register Fields		GIC Config Register (GIC_SH_CONFIG) Offset 0x0000	Reset State
Name	Bits		
COUNTSTOP	28	Setting this bit will stop counters GIC_CounterHi and GIC_CounterLo	0
COUNTBITS	27 - 24	Number of Implemented Bits = (32 + (COUNTBITS*4))	8
NUMINTERRUPTS	23 - 16	Number of External Interrupt Sources = 8 + (8 * NUMINTERRUPTS)	Preset
PVPE	8 -0	Total number of Processing Elements in the system	Preset

```
// extracting number of interrupt slices:
#define GIC_SH_CONFIG 0x0000
#define NUMINTERRUPTS 16
#define NUMINTERRUPTS_S 8

li    a1, GIC_BASE_ADDR // load virtual base address of the GIC registers NOTE: must be uncached address
lw    a0, GIC_SH_CONFIG(a1)          // dereference GIC_SH_CONFIG
ext   a0, NUMINTERRUPTS, NUMINTERRUPTS_S                    // NUMINTERRUPTS (actually slices - 1)
```

MIPS

6

The Configuration register is the first register in the shared section located at offset 0. It contains configuration information and a control bit.

The Count stop bit is a control bit that will stop the GIC counters.

The Count Bits field encodes the number of bits in the GIC counter. In the current implementation there are always 64 bits in the counter so count bit will always be 8.

I'll cover the counter registers a little later in this section.

The boot code will use the Numinterrupts field to determine how many external interrupt sources there are. Interrupt sources are configured in the core in groups of 8. This field tells you how many groups of 8 plus 1 the core has. 0 would indicate 1 group of 8, 1 would indicate 2 groups of 8, and so on.

PVPE tells you how many Processing Elements there are in the system.

Configuring an Interrupt Source: GIC Interrupt Polarity Registers

32 bit

- **Up to 8 Polarity Registers**
  - Number of registers = (8 * (8 * NUMINTERRUPTS)) / 32
  - If interrupt type is level, 0 = Active Low, 1 = Active High
  - If Single Edge, 0 = Falling toggled, 1 = Rising toggled

Register Fields		Global Interrupt Polarity Registers (GIC_SH_POL Offset 0x0100 – 0x011C)	Reset State
Name	Bits		
GIC_SH_POL	31 - 0	Each bit in this register represents an interrupt source. The state of the bit indicates the polarity of the interrupt.	0

```
#define GIC_SH_POL31_0 0x0100          // offset from the GIC base address for polarity bits for interrupt sources 0 - 31
li    a1, GIC_BASE_ADDR                // load virtual base address of the GIC registers NOTE: must be uncached address
li    a0, 0x8000000                    // interrupt source 31 (bit 31)
sw    a0, GIC_SH_POL31_0(a1) // (high/rise for 31)
```

MIPS

7

You can configure the polarity of the interrupt source by using the Polarity registers. There are enough registers instantiated for the number of interrupt sources in the Coherent Processing System. As I have gone over in the GIC Configuration register slide the NUMINTERRUPTS field can be decoded to give you the total number of interrupt sources. Since each bit in each Polarity register controls a source, to find out how many polarity registers there are you divide the total number interrupt sources by 32, always rounding up if there is carry over. Interrupt source 0 is bit 0 of the first polarity register located at offset 100 hex. This register will control the first 32 interrupt sources, then the next register will control the next 32 interrupt sources 32 through 63 and so on. There are other registers I will cover next that determine the interrupt type which can be level, single edge or dual edge sensitive.

+ If the interrupt type is level sensitive then setting the corresponding source bit to 1 will configure the source to active High, and setting it to 0 will configure it to be active low.

+ If the interrupt is single edge sensitive then setting the corresponding source bit to 1 will configure the source to rising edge toggle and setting it to 0 will configure it to falling edge toggle.

Configuring an Interrupt Source: Trigger Registers

- **Up to 8 Trigger Registers**
  - Number of registers = (8 * (8 * NUMINTERRUPTS)) / 32
  - 0 = Level sensitive
  - 1 = Edge sensitive

32 bit

Register Fields		Global Interrupt Trigger Type Registers (GIC_SH_TRIG Offset 0x0180 – 0x09C)	Reset State
Name	Bits		
GIC_SH_TRIG	31 - 0	Each bit in this register represents an interrupt source. The state of the bit indicates the nature of the signaling interrupt.	0

```
#define GIC_SH_TRIG31_0 0x0180      // offset from the GIC base address for trigger bits for interrupt sources 0 - 31
li    a1, GIC_BASE_ADDR             // load virtual base address of the GIC registers NOTE: must be uncached address
li    a0, 0x8000000                 // interrupt source 31 (bit 31)
sw    a0, GIC_SH_TRIG31_0(a1)       // (edge sensitive)
```

MIPS

8

You can configure the type of the interrupt source by using the trigger registers. There are enough Trigger registers to cover the number of interrupt sources implemented. The calculation of the number of registers is the same as it was for the polarity registers.

+ If a bit is set to 0 the interrupt is level type.

+ If a bit is set to 1 the interrupt is Edge type. The next slide show how to configure it as single or dual edge.

Configuring an Interrupt Source: GIC Interrupt Dual Edge Registers

- **Up to 8 Dual Edge Registers**
  - Number of registers = (8 * (8 * NUMINTERRUPTS)) / 32

32 bit

Register Fields		Global Dual Edge Registers (GIC_SH_DUAL Offset 0x0200 – 0x021C)	Reset State
Name	Bits		
GIC_SH_DUAL	31 - 0	Each bit in this register represents an interrupt source. The state of the bit indicates the nature of the interrupt signaling. 0 == Single edge 1 == Dual edge	0

```
#define GIC_SH_DUAL31_0 0x0200          // offset from the GIC base address for dual bits for interrupt sources 0 - 31
li   a1, GIC_BASE_ADDR    // load virtual base address of the GIC registers NOTE: must be uncached address
li   a0, 0x8000000                      // interrupt source 31 (bit 31)
sw   a0, GIC_SH_DUAL31_0(a1)            // Dual
```

MIPS

9

If you configured a source to be edge sensitive you can further configure it for single edge or dual edge using the Dual Edge registers. As was the case for the previous 2 registers there are enough Dual Edge registers to cover the number of interrupt sources implemented. The calculation is the same as it was for the polarity registers.

+ If the bit is set to 0 the interrupt is single edge.

+ If the bit is set to 1 the interrupt is dual edge.

Inter-processor Interrupts: GIC Interrupt Write Edge Register

- **Supports Inter-processor Interrupts**
  - Writing the interrupt number will activate or deactivate the corresponding pending interrupt and cause an interrupt if the interrupt is enabled.
  - Trigger type must be set to Edge sensitive.
  - Used to send an interrupt from one Processing Element to another.

Register Fields		Global Interrupt Write Edge Register (GIC_SH_WEDGE Offset 0x0280)	Reset State
Name	Bits		
RW	31	Controls whether this write is setting (1) or clearing t(0) he interrupt.	UD
Interrupt	30 - 0	This field is the encoded value of the interrupt that is being cleared or set. For example, a value of 0xB indicates interrupt 11 (decimal).	UD

GIC_SH_WEDGE = 0x80000000 + *interrupt number* ;

MIPS

10

The Write edge register is used to support inter-processor interrupts.

If you want to send an interrupt to another processor the interrupt source must be configured to be single or dual edge sensitive. Polarity does not matter -- setting the interrupt will always activate it.

+ The RW field determines if you are setting or clearing the interrupt. To send an interrupt set this bit, and to clear an interrupt clear this bit.

+ Along with the RW bit you need to set the interrupt field to the interrupt number to send the interrupt to. The processor that receives the interrupt would clear the RW bit and set the interrupt field to the interrupt number to clear the interrupt condition.

To set up the system for inter processor interrupts you would determine which interrupt sources to use for this purpose and which processors should enable a specific one of those interrupts. In this way each interrupt can be programmed for a specific processor or any number of processors could be interrupted by a single interrupt.

The Boot Code section of this class gives a coding example of setting up interrupts 24 through 39 for the purpose of inter processor interrupts.

GIC Interrupt Enabling and Detection

- **Three register groups**
  - Set Mask - enable or disable an interrupt (0 -255)
  - Mask - report enabled interrupts
  - Pending - report active interrupts

GIC_SH_RMASK 0x300 – 0x31C
GIC_SH_SMASK Offsets 0x0380 – 0x039C
GIC_SH_MASK Offsets 0x0400 – 0x041C
GIC_SH_PEND Offsets 0x0480 – 0x049C

Register Fields		Description	Reset State
Name	Bits		
GIC_SH_RMASK	31-0	Writing a 1 to any bit will disable the corresponding interrupt	UD
GIC_SH_SMASK	31 - 0	Writing a 1 to any bit will enable the corresponding interrupt	UD
GIC_SH_MASK	31 - 0	Any bit set to 1 designates an enabled interrupt	0
GIC_SH_PEND	31 - 0	Any bit set to 1 designates a pending interrupt	UD

MIPS 11

The next 3 Register groups control the enabling, disabling, reporting enabled state, and reporting pending interrupts. As with the Polarity, Trigger, and Dual Edge registers, there are enough instantiated registers in each group for the number of interrupt sources in the Coherent Processing System.

+ The table shows the address offset ranges for each register group.

+ The Reset Mask register is a write only register. Use it to Disable an interrupt by setting the bit that corresponds to the interrupt.

+ The set Mask Register is a write only register. Use it to Enable an interrupt source by setting the bit that corresponds to the interrupt.

When you write to the Set or Reset registers only the bits you set are affected so if you want to disable interrupt source 3 you would just set bit 3 of the first Reset Register and only interrupt source 3 would be affected.

+ To check to see if an interrupt source is enabled or disabled you read the Mask Registers. Any set bits indicate that the corresponding interrupt is enabled.

+ To check to see which interrupts are active, you read the Pending registers. Any bits that are set indicate the corresponding interrupt is active and waiting to be processed.

```
#define GIC_SH_RMASK31_0 0x0300          // offset fro the GIC base address for bits
for corresponding interrupt number
 li     a1, GIC_BASE_ADDR
 li     a0, 0x8000000                    // interrupt number 31 (bit 31)
 sw     a0, GIC_SH_RMASK31_0(a1)
```

Interrupt Mapping to Processor Element and Pin

- **Map Interrupt Source to Processor Element and to pin on that processor is a 2 step process:**
  - Interrupt Mapping to Pin using the Interrupt Map Source to Pin Registers
    - One register for each interrupt source
    - Maps to Interrupt pins 0 – 5
  - Interrupt Mapping to Processor Element using the Interrupt Map to Processor Registers (Single processor, VP or VPE)
    - One register for each source

MIPS

12

These next slides cover the mapping of an interrupt source to an interrupt pin on a specific Processor. The MIPS architecture defines 2 software interrupts and 6 hardware interrupt pins for each Processor. The Global Interrupt Controller can map any interrupt source to any hardware interrupt pin on any Processor. In addition any interrupt source can be mapped to the non-maskable interrupt pin or on an MT system any of the yield qualifier pins.

+ The map to pin registers are used to map to a hardware interrupt pin on a processor.

+ And the Map to VPE registers are used to map the interrupt source to a particular Processor.

Interrupt Mapping to Pin

- **MAP register for each Interrupt Source**
  - Register Address = 0x0500 + (source number * 4)
  - Maps Interrupt pin, NMI, or Yield Qualifier (if MT)

Register Fields		Global Interrupt Map to Pin Registers (GIC_SH_MAP_PIN 0x0500 – 0x08FC)	Reset State
Name	Bits		
MAP_TO_PIN	31	When set, map to interrupt Pin per MAP field	1
MAP_TO_NMI	30	When set, map to NMI	0
MAP_TO_YQ	29	When set, map to Yield Qualifier (MT Only!)	0
MAP	5 - 0	Interrupt, EIC vector, or Qualifier Pin to Map to	0

MIPS

13

The Map to Pin register is used to map a interrupt source to a interrupt pin.

+ For every interrupt source there is a 32 bit, map to pin register. The address of the register that corresponds to any interrupt source is the GIC base address plus 500 hex, plus the interrupt source number times 4, because the addresses are on word boundaries of 4 bytes each, plus the offset into the GIC.

+ There are three bits that control the type of pin that is assigned to the interrupt source. While there is nothing to prevent you from setting more than one of these bits, only one of these bits should be set.

+ The map to pin bit means you will be mapping to an interrupt pin of the Processor that you will be assigning in Map to Vpe register. The actual pin or interrupt vector will be set in the map field of this register.

+ The map to NMI bit will map the source to the NMI input of the Processing Element.

+ The Map to YQ should only be used if your multi core is made up of multi-threaded processors. In that case, it will map the interrupt source to one of the Yield Qualifiers of a VPE.

+ The map field setting depends on whether you are mapping to a pin or a Yield qualifier. For pin mapping it is simply the pin number you are assigning the interrupt source to. If you are using EIC mode, then it is the vector number between 0 and 63. For a Yield Qualifier it is simply the number of the YQ pin you want this source to activate.

Interrupt Mapping to Processor Element

- **Two registers for each interrupt source are used to map the interrupt source to a Processing Element number from 0 to 63.**
  - Register Address = 0x2000 + (source number * 0x20)
  - Currently you can only have up to 4 Processing Elements for a multi-core system containing single cores and 8 Processing Elements for a system containing multi-threaded cores.

32 bit

Register Fields		Global Interrupt Map to VPE Registers (GIC_SH_MAP_VPE 0x2000, 0x2004 – 0x3FE0, 0x3FE4)	Reset State
Name	Bits		
GIC_SH_MAP0_VPE	31 - 0	Each bit in these registers represents a Processing Element. Setting a bit directs the interrupt source to a Processing Element.	0
GIC_SH_MAP1_VPE	63 – 32		0

```
#define GIC_SH_MAP0_VPE31_0 0x2000          // register offset from GIC base
#define GIC_SH_MAP_SPACER 0x20              // space between registers
li    a1, GIC_BASE_ADDR
a0,   1                                     // set bit 0 for CORE0 or for MT vpe0
sw    a0, GIC_SH_MAP0_VPE31_0 + (GIC_SH_MAP_SPACER * 31) (a1)   // source 31 to VPE 0
```

MIPS

14

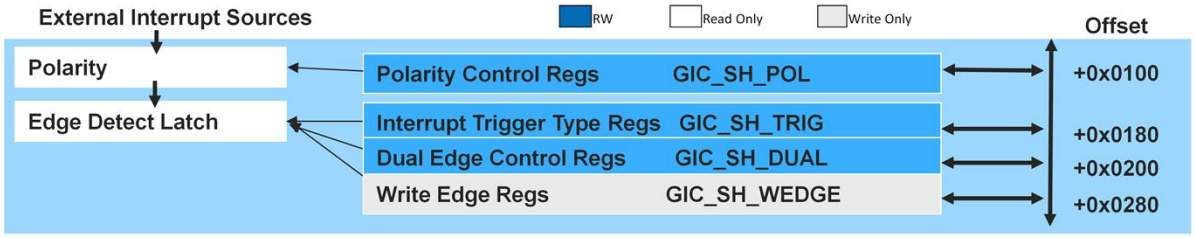
Each interrupt source can be routed to any Processor or VPE on a MT core. This is done by programming the Global Interrupt Map to VPE registers. There are 2 of these registers for each interrupt source so you can map Processing Elements from 0 to 63.

+ Each of these pairs is located on a 32 byte boundary. The first pair for interrupt source 0 is located at offset 2000 hex in the GIC address space. The second pair at offset 2020 hex and so on.

+ On a multi core system made up of single cores we currently support 4 cores so only bits 0 through 3 of the first register in the pair can be used. On a multi core system made up of Multi-Threaded cores you can have 4 cores with 2 VPEs each so in this case you can have 8 Processors. For this type of core, bits 0 through 7 of the first register are used.

You can map an interrupt source to more than one VPE that way all VPEs can synchronize on an interrupt. This is useful for processor to processor interrupts. Software would need to control which VPE did the house keeping if needed for the interrupt.

Summary for External Interrupts



- **Setting Interrupt Type**
  - Polarity Active High/low
    - Global Interrupt Polarity Registers
  - Trigger Level/Edge
    - Global Interrupt Trigger Type Registers
  - Edge Dual/Single
    - Global Dual Edge Registers
  - Write Edge
    - Inter-processor Interrupts

Polarity Control	Interrupt Trigger Type	Dual Edge Control	State
0	0	X	Active Low
1	0	X	Active High
0	1	0	Falling Edge
1	1	0	Raising Edge
X	1	1	Dual Edge

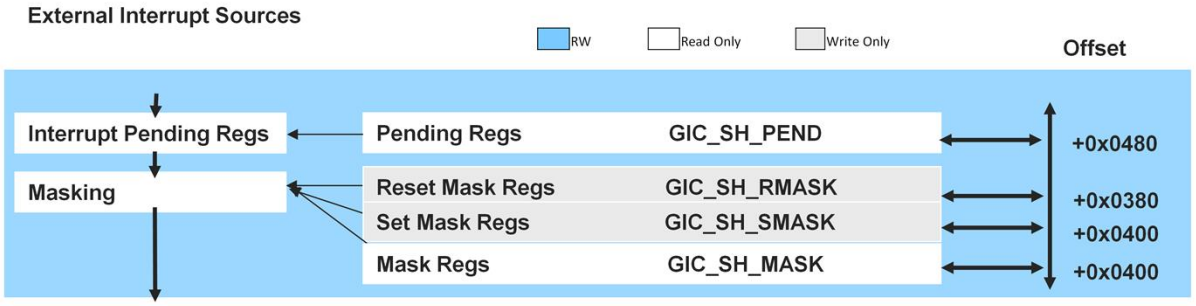
MIPS

15

In summary;

+I have shown you how to set the interrupt type by using the Global Polarity, Trigger, Dual Edge registers and how to send and clear inter-processor interrupts using the Write Edge register

Summary for External Interrupts



- **Set Interrupt State**
  - Enable
    - GIC_SH_SMASK Registers
  - Disable
    - GIC_SH_RMASK Registers
- **Detecting Interrupt State (Read only)**
  - Enabled State
    - GIC_SH_MASK Register
  - Pending State
    - GIC_SH_PEND Register

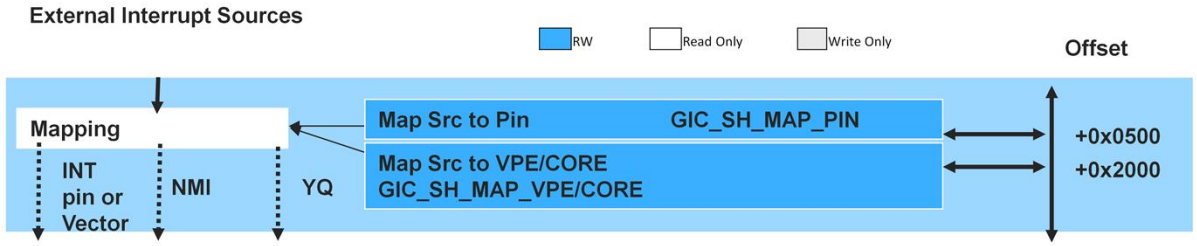
MIPS

16

+ How to set the interrupt state using the Set Mask Registers to enable and the Reset Mask Registers disable interrupts.

+ How to detect the Interrupt state using the Mask and Pending Registers.

Summary for External Interrupts

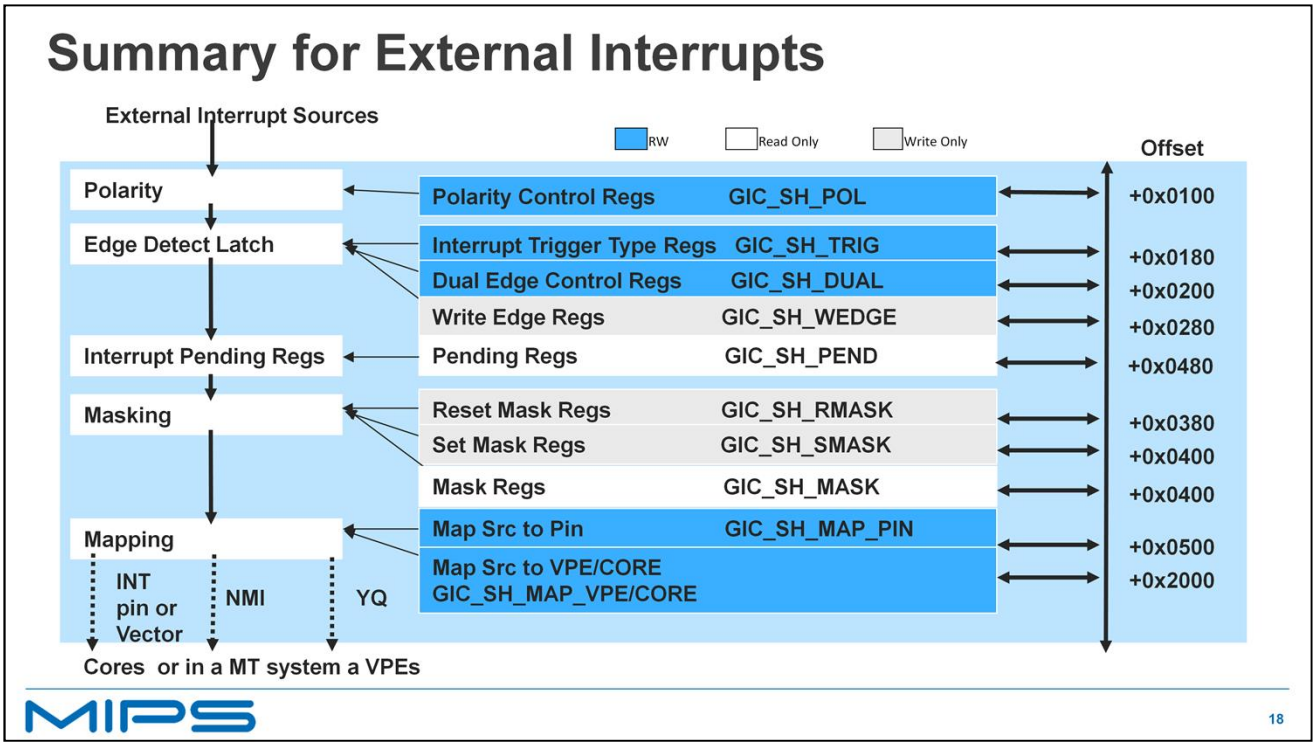


- **Map external interrupts to pins**
  - Setting type of pin (Pin, NMI or Yield Qualifier
    - Global Interrupt Map to Pin Registers
  - Mapping to Processor Element
    - Global Interrupt Map to VPE Registers

MIPS

17

+ How to map external interrupt sources using the Map to Pin and Map to VPE registers.



Here is the big picture summary of the registers for external interrupts.

Local and Other Sections of the GIC

- **Interrupts Generated internal to a Processing Element**
  - Watch Dog Timer
  - Count and Compare
  - Software Interrupts
  - Local Performance Counters
- **Shadow set register assignment**
- **Local Section for Processing Element that is executing**
- **Other Section set by VPE-Other Addressing Register**

MIPS

19

The local and other sections are used to process any interrupts which are generated locally within a processing element. This

includes the watchdog timer, software interrupts, local performance counters, and the count and compare interrupts. All of these interrupts can be routed to an interrupt pin on the local core.

+ A local shadow register set can be mapped to an EIC interrupt source.

+ There are 2 identical sections. The local section is the registers for the processor Element that is executing the code. The Other section is the section of another processor. By using the other section one processor can program all the other processors.

Other Addressing Register

- **Writing to another processor's local registers**
  - Uses the "Other Address Space Register"

Register Fields		VPE-Other Addressing Register (GIC_VPEi_OTHER_ADDRESS Offset 0x8080)	Reset State
Name	Bits		
VPENum	4 - 0	Number of the register set to be accessed in the Other address space	0

The "Other address register" is used to select the processors registers that are accessed through the other registers mapped section. The Processor number is written to the lower 5 bits of this register. If you are running on an MT system this would be a VPE number or a VP number.

Local Timer Registers

- **Three types of local timers**
  - CP0 Counter and Compare Timer
    - Within each Processing Element
    - Not Shared by any other Processing Element
    - Generates a Timer interrupt
  - Interval Timer
    - Counter shared among all Processing Elements - GIC_SH_CounterLo/Hi
    - Local Compare GIC_VPE_CompareLo/Hi registers for each Processing Element
    - Generates a Compare interrupt
  - WatchDog Timer
    - One for each Processing Element
    - Generates a WatchDog interrupt

MIPS

There are 3 types of counters available to each Processor.

+ The CP0 Counter that is within each Processing element and not shared by any other Processing element. When a value is set in CP0 "compare register" and then the "Count register" reaches that value an interrupt can be generated. This is covered in detail in the programming a MIPS core class. This interrupt generates a Timer interrupt. As you will see in upcoming slides the timer interrupt can be enabled and mapped by the GIC to an interrupt pin within the local Processor.

+ The interval timer is very similar to the CP0 counter. It too works with count and compare registers. The difference is that the Count register is global to all Processors whereas the Compare registers are local to each Processor. Both of these registers are part of the GIC and I will cover them in this class section. The interval timer generates a Compare interrupt which like the timer interrupt can be enabled and mapped by the GIC to a interrupt pin within the local Processor.

+ The WatchDog timer is a count down timer. It is meant to be a liveliness timer to tell if a Processor has gotten stuck. It needs to be kept from timing out by the software running on the local Processor. When the count in the WatchDog register reaches 0 it generates a WatchDog interrupt. Like the other 2 timers the WatchDog interrupt can be mapped by the GIC to an interrupt pin within the local Processing element. Typically this would be to the NMI pin.

GIC Interval Timer Registers

Register Fields		GIC_SH_CounterLo/Hi Offset 0x0010/0x0024	Reset State
Name	Bits		
GIC_SH_CounterHi/Lo	31 - 0	Upper and lower halves of GIC Counter (each 32 bits)	0

Register Fields		GIC_VPE_CompareLo/Hi Offset L - 0x80A0/0x80A4 U – 0xC0A0/0xCA4	Reset State
Name	Bits		
GIC_VPE_CompareHi/Lo	31 - 0	Upper and lower halves of GIC Local Compare Counter	0xffffffff

MIPS

22

For the interval timer the count is kept in a 2 register set that counts in the time units of the GIC.

+ Usually this is in cycles but it is implementation specific so you should check with your system designer to make sure what frequency the counter counts at. Since the number of count bits is fixed at 64 when the Lo counter overflows it increments the Hi counter. This is a read write register but you should always disable the counter by setting the Count Stop bit in the GIC configuration register before you write to it. Note again this is a shared register within the GIC.

+ The Compare value is set in a 2 register set within the local section of each Processor. When the GIC Shared counter reaches the value in these registers a compare interrupt is generated. Note again that the compare registers are per Processor.

GIC WatchDog Timer Register

Register Fields		GIC WatchDog Timer (GIC_VPE_WD_CONFIG Offset L – 0x8090 O – 0C090)	Reset State
Name	Bits		
WDRESET	7	Indicates this Processing Element generated SI_Reset due to a WatchDog expiration.	0
WDINTR	6	WatchDog expiration generated interrupt	Undefined
WAITMODE_CNTRL	5	Low Power stop behavior 0 - Stop 1 - Continue counting	0
DEBUGMODE_CNTRL	4	Debug Mode stop behavior 0 - Stop 1 - Continue counting	0
TYPE	3 - 1	0 - Assert interrupt and stop 1 - 2nd pass system reset 2 - Interrupt and restart	0
WD_START	0	0 – Stop the WD counter 1 – Start and/or reload initial count	0

MIPS

23

The WatchDog timer configuration register configures the actions of the WatchDog timer.

+ If the WatchDog Reset bit is set it indicates that this Processors WatchDog timer expired and has generated a system wide reset signal. This bit can be cleared by writing a 1 to it.

+ The WatchDog interrupt bit indicates that this WatchDog timer generated an interrupt. This interrupt can be cleared by writing a 1 to this bit.

+ If the Wait mode control bit is set to 1 the WatchDog timer will to continue to count if the Processing core is in low power mode; otherwise the WatchDog will stop decrementing when the Processing core enters low power mode.

+ If the Debug mode control bit is set to 1 the WatchDog timer will to continue to count if the Processing core is in Debug mode; otherwise the WatchDog will stop decrementing when the Processing core enters Debug mode.

+ The Type field determines what happens when the WatchDog reaches 0. If the type is set to 0 the Processor stops and asserts a WatchDog interrupt signal. If it is set to 1 the WatchDog will reset to the initial count and start decrementing again. If it reaches 0 on the second try then it will assert the system wide reset signal, SI_Reset. If it is set to 2 it will assert a WatchDog interrupt, reload the initial count and start decrementing again.

+ The WD Start field is used to Start or Stop the watchdog timer or reload the initial count. Setting this bit to 0 disables the timer. Setting this bit to 1 enables and reloads the count set in the Watchdog Timer Initial Count Register. To reload the count to a already enabled timer all you need to do to set this bit to 1.

GIC WatchDog Timer Registers

Register Fields		Watchdog Timer Count Register (GIC_VPE_WD_COUNT Offset L – 0x8094 O – 0C094)	Reset State
Name	Bits		
GIC_VPE_WD_COUNT	31 - 0	This read-only register indicates the state of the decrementing counter. The width of the counter is 32 bits.	UD

Register Fields		Watchdog Timer Initial Count Registers (GIC_VPE_WD_INITIAL Offset L – 0x8098 O – HC098)	Reset State
Name	Bits		
GIC_VPE_WD_INITIAL	31 - 0	Initial value to be loaded into the Watchdog counter. Needs to be done when the counter is disabled; otherwise, the results are UNPREDICTABLE.	UD

MIPS

24

The WatchDog timer is a 2 register set.

The WatchDog count register is a read only register that contains the decrementing count of the watchdog. If the count in this register reaches 0 it will take action depending on the “type” set in the WatchDog Configuration register.

The WatchDog Initial Count register is set by software with the number of counting Elements to count down. This value will be loaded into the WatchDog Count Register when write a 1 to the GIC WatchDog Timer configuration register.

Local Interrupt Control Register

Register Fields		Local Interrupt Control Register (GIC_VPE_CTL Offsets L - 0x8000 O - 0xC000)	Reset State
Name	Bits		
FDC_ROUTABLE	4	If this bit is set, the CPU Fast Debug Channel Interrupt is routable within the GIC.	Preset
SWINT_ROUTABLE	3	If this bit is set, the CPU Software Interrupts are routable within the GIC.	Preset
PERFCOUNT_ROUTABLE	2	If this bit is set, the CPU Performance Counter Interrupt is routable within the GIC.	Preset
TIMER_ROUTABLE	1	If this bit is set, the CPU Timer Interrupt is routable within the GIC.	Preset
EIC_MODE	0	Writing a 1 to this bit will set this VPE local interrupt controller to EIC (External Interrupt Controller) mode.	0

MIPS

25

The local interrupt control register contains

+ 4 state bits that correspond to the routability of 4 local interrupts which are: the Fast Debug Channel interrupt, Software interrupts , Performance Counter and Timer interrupts.

+ These bits are preset at core build time.

If a bit is set the corresponding interrupt is software routable.

If its bit is not set the interrupts would be hardwired to one of the hardware interrupt pins. Which pin they are hardwired to is implementation dependent.

+ The EIC_MODE bit is the only writable bit. It controls the External Interrupt Controller mode of the core. Setting this bit will set the processing element to external interrupt mode.

Note that the state of the EIC_MODE bit is driven onto the SI_EICPresent pin. Hardware uses this pin to update the state of the CP0 Config3.VEIC

bit to indicate support for and status of the EIC mode.

Local Interrupt Pending Register

- Local Interrupt Status

Register Fields		Local Interrupt Pending Register (GIC_VPE_PEND Offsets L - 0x8004 O - 0xC004)	Reset State
Name	Bits		
FDC_PEND	6	If set, Fast Debug Channel interrupt is pending.	UD
SWINT1_PEND	5	If set, Software 1 interrupt is pending.	UD
SWINT0_PEND	4	If set, Software 0 interrupt is pending.	UD
PERFCOUNT_PEND	3	If set, Performance Counter interrupt is pending.	UD
TIMER_PEND	2	If set, Timer interrupt is pending.	UD
COMPARE_PEND	1	If set, local GIC Count/Compare interrupt is pending.	UD
WD_PEND	0	If set, WatchDog interrupt is pending.	UD

The next 4 slides all deal with the local processor (or VPE in an MT system) interrupts. This register is the interrupt pending register. This is a read only register that will tell you if an interrupt is pending for one of the local interrupts regardless of whether or not the interrupt is enabled.

Local Interrupt Mask Register (Enable Status)

Register Fields		Local Interrupt Mask Register (GIC_VPE_MASK Offsets L - 0x8008 O - 0xC008)	Reset State
Name	Bits		
FDC_MASK	6	If set, Fast Debug Channel interrupt is enabled.	UD
SWINT1_MASK	5	If set, Software 1 interrupt is enabled.	UD
SWINT0_MASK	4	If set, Software 0 interrupt is enabled.	UD
PERFCOUNT_MASK	3	If set, Performance Counter interrupt is enabled.	UD
TIMER_MASK	2	If set, Timer interrupt is enabled.	UD
COMPARE_MASK	1	If set, local GIC Count/Compare interrupt is enabled.	UD
WD_MASK	0	If set, WatchDog interrupt is enabled.	UD

MIPS

27

The Local Interrupt Mask register is a read only register that reports the enabled status of a local interrupt. If an interrupt's bit is set the interrupt is enabled and if the interrupt occurs it will interrupt the processor (or VPE in a MT system).

Local Interrupt Reset Mask Register (Disable interrupt)

Register Fields		Local Interrupt Disable (GIC_VPE_RMASK Offsets L - 0x800C O - 0xC00C)	Reset State
Name	Bits		
FDC_MASK_RESET	6	Writing a 0x1 to this bit disables the local Fast Debug Channel interrupt.	UD
SWINT1_MASK_RESET	5	Writing a 0x1 to this bit disables the local Software 1 interrupt.	UD
SWINT0_MASK_RESET	4	Writing a 0x1 to this bit disables the local Software 0 interrupt.	UD
PERFCOUNT_MASK_RESET	3	Writing a 0x1 to this bit disables the local Performance Counter interrupt.	UD
TIMER_MASK_RESET	2	Writing a 0x1 to this bit disables the local Timer interrupt.	UD
COMPARE_MASK_RESET	1	Writing a 0x1 to this bit disables the local GIC Count/Compare interrupt.	UD
WD_MASK_RESET	0	Writing a 0x1 to this bit disables the local WatchDog interrupt.	UD

MIPS

28

The interrupt Reset Mask register is a write only register. Setting an interrupt's bit to 1 will disable that interrupt and the processor will not be interrupted if this interrupt is activated. But it will still show up in the interrupt pending register as a pending interrupt that you could poll for.

Local Interrupt Set Mask Register

Register Fields		Local Interrupt Enable (GIC_VPE_SMASK Offsets L - 0x8010 O - 0xC010)	Reset State
Name	Bits		
FDC_MASK_SET	6	Writing a 0x1 to this bit enables the local Fast Debug Channel interrupt.	UD
SWINT1_MASK_SET	5	Writing a 0x1 to this bit enables the local Software 1 interrupt.	UD
SWINT0_MASK_SET	4	Writing a 0x1 to this bit enables the local Software 0 interrupt.	UD
PERFCOUNT_MASK_SET	3	Writing a 0x1 to this bit enables the local Performance Counter interrupt.	UD
TIMER_MASK_SET	2	Writing a 0x1 to this bit enables the local Timer interrupt.	UD
COMPARE_MASK_SET	1	Writing a 0x1 to this bit enables the local GIC Count/Compare interrupt.	UD
WD_MASK_SET	0	Writing a 0x1 to this bit enables the local WatchDog interrupt.	UD

MIPS

29

The Local Interrupt Set Mask register is a write only register. When an interrupt's bit is set the interrupt will be enabled and will generate an interrupt when the interrupt is activated.

Local Interrupt Mapping to Pin Registers

GIC_VPE_WD_MAP L - 0x8040 O - 0xC040
GIC_VPE_COMPARE_MAP L - 0x8044 O - 0xC044
GIC_VPE_TIMER_MAP L - 0x8048 O - 0xC048
GIC_VPE_FDC_MAP L - 0x804C O - 0xC04C
GIC_VPE_PERFCTR_MAP L - 0x8050 O - 0xC050
GIC_VPE_SWInt0_MAP L - 0x8054 O - 0xC054
GIC_VPE_SWInt1_MAP L - 0x8058 O - 0xC058

Register Fields		Description	Reset State
Name	Bits		
MAP_TO_PIN	31	When set, map to interrupt Pin per MAP field	1
MAP_TO_NMI	30	When set, map to NMI	0
MAP_TO_YQ	29	When set, map to Yield Qualifier (MT)	0
MAP	5 - 0	Interrupt or Qualifier Pin to Map to	0

MIPS

30

For every local interrupt

+ there is a 32 bit map pin register as shown in the table. This allows a local interrupt to be mapped to an interrupt pin of the local Processor if it is routable in your core. Check the GIC_VPE_CTL register routable bits to see if a particular interrupt is routable and not already hardwired.

+ There are three bits that control what the interrupt source is routed or mapped to. While there is nothing to prevent you from setting more that one of these bits, only one of these bits should be set.

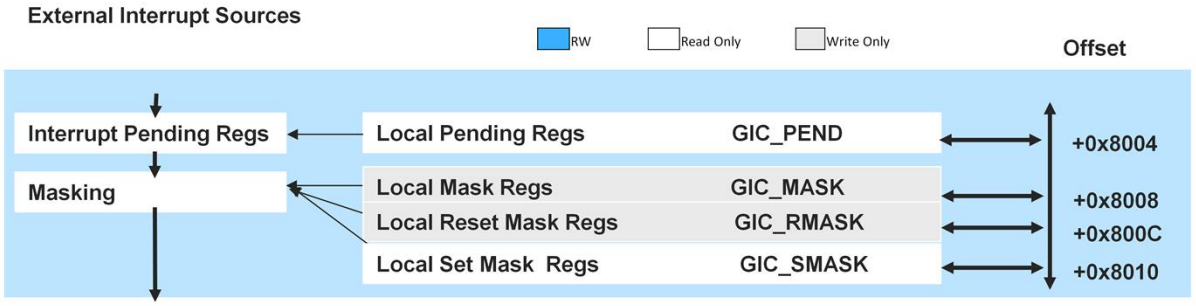
+ The map to pin bit will map the interrupt source to an interrupt pin of the processor. The actual pin will be set in the map field of this register.

+ The map to NMI bit will map the interrupt source to the NMI input of the Processor.

+ The Map to YQ should only be used if your multi core is made up of multi-threaded processors. It will map the interrupt source to one of the Yield Qualifiers of the VPE that is set in the map field.

+ The map field setting depends on if you are mapping to a pin or a Yield qualifier. For pin mapping it is simply the pin number you are assigning the interrupt source to. For a Yield Qualifier it is simply the number of the YQ pin you want this source to activate.

Summary for Local Interrupts



- **Set Interrupt State (Write only)**
  - Enable
    - GIC_SMASK Registers
  - Disable
    - GIC_RMASK Registers
- **Detecting Interrupt State (Read only)**
  - Enabled State
    - GIC_MASK Register
  - Pending State
    - GIC_PEND Register

MIPS

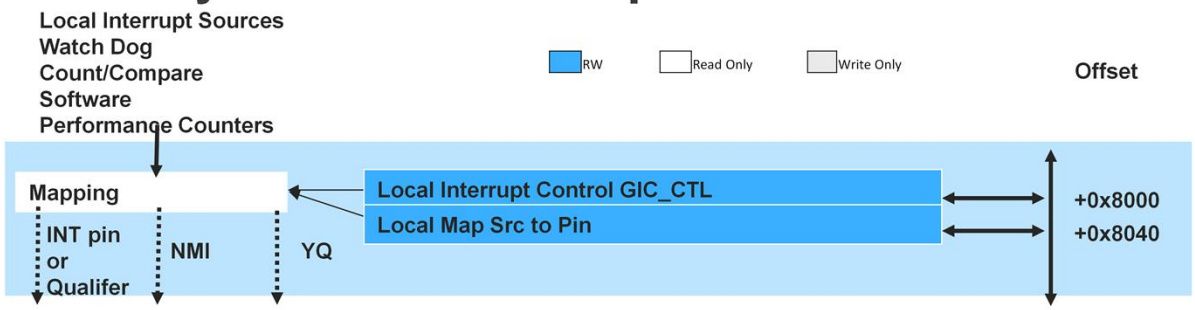
31

In summary I have shown you

+ How to set the interrupt state using the Set Mask Registers to enable and the Reset Mask Registers disable interrupts.

+ How to detect the Interrupt state using the Mask and Pending Registers.

Summary for Local Interrupts



- Local Interrupts Mapping
  - Routing possible?
    - Local Interrupt Control Register
  - Mapping
    - Local Interrupt Mapping to Pin Registers

MIPS

32

+ Check the Local Interrupt Control Register to see if routing/ Mapping is possible and enables EIC mode

Then Rout the interrupts to a pin, NMI, or Yield Qualifier for a MT system

Processor ID

- **Number of the Processing Element that is reading the local registers.**

Register Fields		Processor ID (GIC_VPE_IDENT Offset 0x8088)	Reset State
Name	Bits		
VPENum	31 - 0	Number of the Processing Element accessing the local registers	-

MIPS

33

The Core local identification register identifies the Processor (or VPE in a MT system) that is executing the code that is reading this register. It is also the number the hardware uses to locate the specific group of local registers for this processor.

EIC Shadow Register Set Assignment

- **External Interrupt Controller Mode Shadow Register Set Usage**
  - If using EIC mode, this field must be initialized!
    - If there are no shadow sets, you must set this field to 0
    - One register per interrupt source (63 total spaced on word boundary)

Register Fields		EIC Shadow Set Registers (GIC_VPEi_EICSS Offset L - 0x8100 – 0x81FC Offset O - 0xC100 – 0xC1FC)	Reset State
Name	Bits		
EIC_SS	3 - 0	Shadow Set # to use for this particular interrupt.	Undefined

MIPS

34

If you are using EIC mode you must set which register set to use for each interrupt source. This is not an initialized register so if your system does not have shadow register sets you must set this field to 0.

+ Note that in the slide and others, offset L is the offset in the address segment of the currently executing Processing Element,

+ and offset O is the offset in the address segment of the Processing Element whose ID is in the VPE-Other Addressing Register.

User Mode Section Offset 0x10000

- Only registers available in user mode

Register Fields		GIC CounterLo/Hi (Offset 0x10000/0x10004)	Reset State
Name	Bits		
GIC_CounterLo/Hi	31 - 0	Read-only alias for GIC Shared CounterLo/Hi	0xffffffff

The GIC Counter is the only register available in user mode. It is a alias of the GIC Shared Counters. All other registers are only available in a privileged mode.