VciMultiIcu

1) Functional Description

This VCI target is a multi-channels memory mapped peripheral implementing a vectorized interrupt controller. It can concentrate up to 32 input interrupts **p_irq_in[i]** to 8 output interrupts **p_irq_out[k]**.

It behaves as 8 independant **VciIcu** components, and can be used in a multi-processors architecture to dispatch the peripheral interrupts to 8 processors, using the software programmable registers ICU_MASK[k].

There is one independant set of registers for each channel [k] (i.e. for each output interrupt) and each input interrupt can be individually masked through the programmable register ICU_MASK[k].

In principle, the values contained in the ICU_MASK[k] registers must be non-overlapping, because a given input interrupt should be routed to only one processor.

For a given channel, the priority scheme is fixed : The lower indexes have the highest priority.

For each channel [k], the ICU_IT_VECTOR[k] register can be addressed to return the index of the highest priority, non masked, active interrupt **p_irq_[i]**.

This hardware component checks for segmentation violation, and can be used as a default target.

For each channel [k] there is five addressable registers:

- `ICU_INT[k]` Each bit in this register reflects the state of the corresponding input interrupt line. This is read-only.
- `ICU_MASK[k]` Each bit in this register reflects the state of the enable for the corresponding interrupt line. This is read-only.
- `ICU_MASK_SET[k]` Each bit set in the written word will be set in the ICU MASK. (ICU_MASK = ICU_MASK | written_data). This is write-only.
- `ICU_MASK_CLEAR[k]` Each bit set in the written word will be reset in the ICU MASK. (ICU_MASK = ICU_MASK & ~written_data). This is write-only.
- `ICU_IT_VECTOR[k]` This register gives the index of the highest-priority active interrupt. If no interrupt is active, (-1) is returned. This is read-only.

For extensibility issues, you should access your ICU using globally-defined offsets.

You should include file `soclib/icu.h` from your software, it defines `ICU_INT`, `ICU_MASK`, `ICU_MASK_SET`, `ICU_MASK_CLEAR`, `ICU_IT_VECTOR`.

2) Component definition & usage

source:trunk/soclib/module/infrastructure_component/interrupt_infrastructure/vci_multi_icu/caba/metadata/vci_multi_icu.sd

```
Uses( 'vci_icu')
```

3) CABA Implementation

CABA sources

- interface :
  source:trunk/soclib/soclib/module/infrastructure_component/interrupt_infrastructure/vci_multi_icu/caba/source/inclu
- implementation :
  source:trunk/soclib/soclib/module/infrastructure_component/interrupt_infrastructure/vci_multi_icu/caba/source/src/v

CABA Constructor parameters

```
VciIcu(
     sc_module_name name,  // Component Name
     const soclib::common::InTab &index, // Target index
     const soclib::common::MappingTable &mt,  // Mapping Table
     size_t nirq_in,  // Number of input interrupts
     size_t nirq_out);  // Number of channels (output interrupts)
```

CABA Ports

- **p_resetn** : Global system reset
- **p_clk** : Global system clock
- **p_vci** : VCI target port
- **p_irq_in[i]** : Up to 32 input IRQ ports
- **p_irq_out[k]** : Up to 8 output IRQ ports