Component specification document

`xicu` : Programmable interrupt controller

Revision: 1.0 (svn: 4848)
Status: stable
Author: Nicolas Pouillon<nipo@ssji.net>
Date: $Date: 2009-06-13T14:31:43.251346Z $

Chapter 1

Description

1.1 Features

This controller is an interrupt hub, concentrating 3 types of interrupts:

- up to 32 internal programmable timer interrupts (PTI),
- up to 32 external hardware interrupt lines (HWI),
- up to 32 internal write-triggered interrupts (WTI).

All these interrupt sources can be routed to up to 32 interrupt outputs. Each output can mask individual interrupt sources.
Priority between interrupt source types is left to the handling operating system. Priority of interrupts inside an interrupt source type is from the lowest `Idx` (highest priority) to the highest `Idx` (lower priority). This is statically encoded in the $\texttt{PRIO}_{\texttt{OutIdx}}$ register.

1.2 Interrupt routing

Each of the output lines $\texttt{IRQ}_{\texttt{OutIdx}}$ can select any of the 96 interrupt sources. Selected sources are ORed together. Therefore, value of $\texttt{IRQ}_{\texttt{OutIdx}}$ is 1 as long as there is any of:

- unmasked AND unacknowledged PTI,
- active AND unmasked HWI line,
- unmasked AND unacknowledged WTI.

Chapter 2

Hardware component

2.1 Parameters

All hardware implementations of this component may not implement all the up-to-32 PTI (Timers), up-to-32 HWI lines, up-to-32 WTI registers and up-to-32 OUTPUT lines. Therefore component's model have parameters allowing the system designer to get just the needed hardware.
`XICU` implementations must have the following 4 parameters:

pti_count (in range `0..32`): number of programmable timers

hwi_count (in range `0..32`): number of external hardware interrupt lines

wti_count (in range `0..32`): number of write-triggered interrupt sources

irq_count (in range `1..32`): number of output interrupt lines

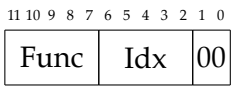
Chapter 3

Programmer's model

3.1 Registers

This component is Little-Endian, when using independant bits in a word, bits are numbered from least-significant (`0`) to most-significant (`31` for a 32-bit word).
This component can be mapped anywhere in the address space, on a 4-KiB boundary.
This component is 32-bit data-word based: arbitrary byte access is not supported.

12 lower address lines are used the following way:



Func (5 bits): function type, explained in the following pages

Idx (5 bits): indexing the function in the range `0..31`

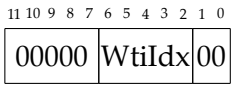
bits 1-0 (2 bits): as this component is 32bit word addressable, these should be always 0.

Register summary

		11	7 6	2 1 0
R/W	Function	Address		
R/W	WTI_REG$_{WtiIdx}$	00000	WtiIdx	00
R/W	PTI_PER$_{PtiIdx}$	00001	PtiIdx	00
R/W	PTI_VAL$_{PtiIdx}$	00010	PtiIdx	00
W	PTI_ACK$_{PtiIdx}$	00011	PtiIdx	00
R/W	MSK_PTI$_{OutIdx}$	00100	OutIdx	00
W	MSK_PTI_ENABLE$_{OutIdx}$	00101	OutIdx	00
W	MSK_PTI_DISABLE$_{OutIdx}$	00110	OutIdx	00
R	PTI_ACTIVE$_{OutIdx}$	00110	OutIdx	00
	Reserved	00111	Res.	00
R/W	MSK_HWI$_{OutIdx}$	01000	OutIdx	00
W	MSK_HWI_ENABLE$_{OutIdx}$	01001	OutIdx	00
W	MSK_HWI_DISABLE$_{OutIdx}$	01010	OutIdx	00
R	HWI_ACTIVE$_{OutIdx}$	01010	OutIdx	00
	Reserved	01011	Res.	00
R/W	MSK_WTI$_{OutIdx}$	01100	OutIdx	00
W	MSK_WTI_ENABLE$_{OutIdx}$	01101	OutIdx	00
W	MSK_WTI_DISABLE$_{OutIdx}$	01110	OutIdx	00
R	WTI_ACTIVE$_{OutIdx}$	01110	OutIdx	00
R	PRIO$_{OutIdx}$	01111	OutIdx	00
	Reserved	1	Reserved	00

3.1.1 `WTI_REG`$_{\texttt{WtiIdx}}$: Write-Triggered Interrupt register

Addressing



Description

This register retains the value written so it can be used as a mailbox between the interrupt source and the target.
One must consider the fact two different sources may write sequentially to this register, overwriting the value present in register. Therefore, usage as a mailbox should expect loss of information.

On write

Raise $\texttt{WTI}_{\texttt{WtiIdx}}$

On read

Acknowledges $\texttt{WTI}_{\texttt{WtiIdx}}$

3.1.2 `PTI_PER`$_{\texttt{PtiIdx}}$: Timer period register

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
00001								PtiIdx		00	

Description

This register contains the reset value for `TIMER`$_{\texttt{PtiIdx}}$ when it wraps to 0.
If this register is set to 0, corresponding timer is disabled and no interrupt is ever raised.
Setting this register to 0 when there is a pending interrupt clears it without need to read `PTI_ACK`$_{\texttt{PtiIdx}}$.

On write

Resets the period of `TIMER`$_{\texttt{PtiIdx}}$. If the timer is currently running, the corresponding timer counter is not reset.

On read

Gets the period of `TIMER`$_{\texttt{PtiIdx}}$

3.1.3 PTI_VAL$_{PtiIdx}$: Timer value register

Addressing

11 10 9 8 7 6 5 4 3 2 1 0		
00010	PtiIdx	00

Description

This register is decremented by 1 on each clock's raising edge. When it gets to 0, the value is reset to the corresponding period register value (PTI_PER$_{PtiIdx}$), and the corresponding timer interrupt line is asserted until acknowledged.
Decrementation goes on whether interrupt is acknowledged or not.

On write

Resets the current value of TIMER$_{PtiIdx}$.
Writing a value greater than PTI_PER$_{PtiIdx}$ in this register has no particular side-effect: value will normally decrement to 0 and then be reset to PTI_PER$_{PtiIdx}$ when wrapping.

On read

Gets the current value of TIMER$_{PtiIdx}$.

3.1.4 PTI_ACK$_{PtiIdx}$: Timer acknowledge register

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	PtiIdx					0	0

Description

This register is used by the software to deassert an interrupt raised by wrapping of PTI_VAL$_{PtiIdx}$ register.

On write

Unsupported

On read

Acknowledges the PTI$_{idx}$. Read value has no useful meaning.

3.1.5 `MSK_PTI`$_{\texttt{OutIdx}}$: PTI mask for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	OutIdx					0	0

Description

Each bit in this register is a mask for the corresponding PTI IRQ. A 1 in bit `x` enables the PTI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

Sets the current mask

On read

Gets the current mask

3.1.6 `MSK_PTI_ENABLE`$_{\texttt{OutIdx}}$: PTI mask enabler for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11 10 9 8 7 6	5 4 3 2	1 0
00101	OutIdx	00

Description

Each bit written here unmasks the corresponding PTI IRQ. Writing a 1 in bit `x` enables the PTI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

ORs `MSK_PTI`$_{\texttt{OutIdx}}$ with written value

On read

Unsupported

3.1.7 `MSK_PTI_DISABLE`$_{\texttt{OutIdx}}$: PTI mask disabler for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	OutIdx					0	0

Description

Each bit written here masks the corresponding PTI IRQ. Writing a 1 in bit `x` disables the PTI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

ANDs `MSK_PTI`$_{\texttt{OutIdx}}$ with negation of written value

On read

Unsupported

3.1.8 `PTI_ACTIVE`$_{\texttt{OutIdx}}$: Current unmasked PTI vector for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	OutIdx					0	0

Description

Each bit read here corresponds to an active and unmasked PTI IRQ, according to current global PTI status and `MSK_PTI`$_{\texttt{OutIdx}}$.

On write

Unsupported

On read

Gets the current PTI status: `CURRENT_PTI` AND `MSK_PTI`$_{\texttt{OutIdx}}$.

3.1.9 `MSK_HWI`$_{\texttt{OutIdx}}$: HWI mask for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	OutIdx					0	0

Description

Each bit in this register is a mask for the corresponding HWI IRQ. A 1 in bit `x` enables the HWI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

Sets the current mask

On read

Gets the current mask

3.1.10 `MSK_HWI_ENABLE`$_{\texttt{OutIdx}}$: HWI mask enabler for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	OutIdx					0	0

Description

Each bit written here unmasks the corresponding HWI IRQ. Writing a 1 in bit `x` enables the HWI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

ORs `MSK_HWI`$_{\texttt{OutIdx}}$ with written value

On read

Unsupported

3.1.11 `MSK_HWI_DISABLE`$_{\texttt{OutIdx}}$: HWI mask disabler for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	OutIdx					0	0

Description

Each bit written here masks the corresponding HWI IRQ. Writing a 1 in bit `x` disables the HWI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

ANDs `MSK_HWI`$_{\texttt{OutIdx}}$ with negation of written value

On read

Unsupported

3.1.12 `HWI_ACTIVE`$_{\texttt{OutIdx}}$: Current unmasked HWI vector for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	OutIdx					0	0

Description

Each bit read here corresponds to an active and unmasked HWI IRQ, according to current global HWI status and `MSK_HWI`$_{\texttt{OutIdx}}$.

On write

Unsupported

On read

Gets the current HWI status: `CURRENT_HWI` AND `MSK_HWI`$_{\texttt{OutIdx}}$.

3.1.13 `MSK_WTI`$_{\texttt{OutIdx}}$: WTI mask for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
01100					OutIdx					00	

Description

Each bit in this register is a mask for the corresponding WTI IRQ. A 1 in bit `x` enables the WTI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

Sets the current mask

On read

Gets the current mask

3.1.14 `MSK_WTI_ENABLE`$_{\texttt{OutIdx}}$: WTI mask enabler for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	OutIdx					0	0

Description

Each bit written here unmasks the corresponding WTI IRQ. Writing a 1 in bit `x` enables the WTI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

ORs `MSK_WTI`$_{\texttt{OutIdx}}$ with written value

On read

Unsupported

3.1.15 `MSK_WTI_DISABLE`$_{\texttt{OutIdx}}$: WTI mask disabler for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
01110					OutIdx					00	

Description

Each bit written here masks the corresponding WTI IRQ. Writing a 1 in bit `x` disables the WTI `x` as an interrupt source for `IRQ`$_{\texttt{OutIdx}}$.

On write

ANDs `MSK_WTI`$_{\texttt{OutIdx}}$ with negation of written value

On read

Unsupported

3.1.16 `WTI_ACTIVE`$_{\texttt{OutIdx}}$: Current unmasked WTI vector for `IRQ`$_{\texttt{OutIdx}}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
01110					OutIdx					00	

Description

Each bit read here corresponds to an active and unmasked WTI IRQ, according to current global WTI status and `MSK_WTI`$_{\texttt{OutIdx}}$.

On write

Unsupported

On read

Gets the current WTI status: `CURRENT_WTI` AND `MSK_WTI`$_{\texttt{OutIdx}}$.

3.1.17 PRIO$_{OutIdx}$: Output line priority encoder for IRQ$_{OutIdx}$

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
01111					OutIdx					00	

Description

This register holds the type and index of the most important unmasked and unacknowledged interrupt source.

On write

Unsupported

On read

Get the highest-priority active interrupt source for masks corresponding to OUTPUT$_{OutIdx}$. Read value is:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000			PRIO_WTI					000			PRIO_HWI					000			PRIO_PTI					00000					HAS_WTI	HAS_HWI	HAS_PTI

PRIO_WTI: index of the most important unmasked WTI source. This corresponds to index of the lowest 1-bit in WTI_ACTIVE$_{OutIdx}$, counting from LSB=0. If WTI_ACTIVE$_{OutIdx}$ is all-zero, PRIO_WTI has an undefined value.

PRIO_HWI: index of the most important unmasked HWI source. This corresponds to index of the lowest 1-bit in HWI_ACTIVE$_{OutIdx}$, counting from LSB=0. If HWI_ACTIVE$_{OutIdx}$ is all-zero, PRIO_HWI has an undefined value.

PRIO_PTI: index of the most important unmasked PTI source. This corresponds to index of the lowest 1-bit in PTI_ACTIVE$_{OutIdx}$, counting from LSB=0. If PTI_ACTIVE$_{OutIdx}$ is all-zero, PRIO_PTI has an undefined value.

HAS_WTI: whether WTI has at least one unmasked source active (i.e. WTI_ACTIVE$_{OutIdx}$ is not all-zero).

HAS_HWI: whether HWI has at least one unmasked source active (i.e. HWI_ACTIVE$_{OutIdx}$ is not all-zero).

HAS_PTI: whether PTI has at least one unmasked source active (i.e. PTI_ACTIVE$_{OutIdx}$ is not all-zero).

3.1.18 Reserved

Addressing

11	10	9	8	7	6	5	4	3	2	1	0
00111					Res.						00
01011					Res.						00
1	Reserved										00

Description

Reserved

On write

Unsupported

On read

Unsupported