eSi-Watchdog

1 Contents

2 Overview

The eSi-Watchdog core can be used to generate an interrupt should a keep-alive sequence not be written to its control registers at a regular interval. This would typically be used to determine whether a program is running correctly, on the assumption that a program that has crashed would not write the correct sequence. The interrupt output would typically be configured to drive either the reset or non-maskable interrupt of the CPU, in order to allow the crashed program to recover. The register writes required for the keep-alive sequence are chosen so that they are unlikely to be generated by a program that has crashed.

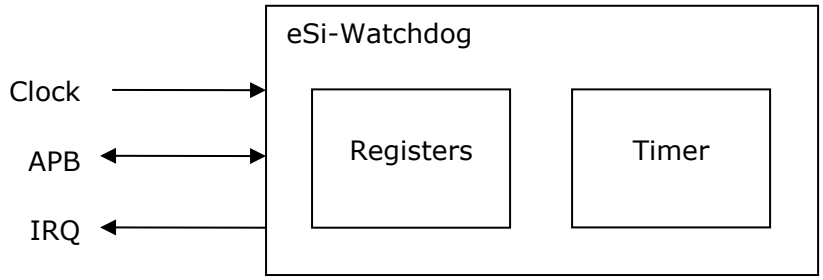

Figure 1: eSi-Watchdog

3 Hardware Interface

Module Name	cpu_apb_watchdog
HDL	Verilog
Technology	Generic
Source Files	cpu_apb_watchdog.v

Port	Type	Values	Description
apb_data_width	Integer	16, 32	Specifies the APB data bus width
apb_address_width	Integer	16, 32	Specifies the APB address bus width

Table 1: Parameters

Port	Direction	Width	Description
clk	Input	1	Clock used for the timer. This clock must be enabled when cactive is asserted.
pclk	Input	1	APB clock
presetn	Input	1	APB reset, active-low
paddr	Input	apb_address_width	APB address (only 8 LSBs are used)
psel	Input	1	APB slave select
penable	Input	1	APB enable
pwrite	Input	1	APB write
pwdata	Input	apb_data_width	APB write data
debug_active	Input	1	Indicates when debugger is active
cactive	Output	1	Clock active
pready	Output	1	APB ready
prdata	Output	apb_data_width	APB read data
pslverr	Output	1	APB slave error
interrupt_n	Output	1	Interrupt request, active-low

Table 2: I/O Ports

For complete details of the APB signals, please refer to the AMBA 3 APB Protocol v1.0 Specification available at:

http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

4 Software Interface

4.1 Register Map

Register	Address offset	Access	Description
counter_lo	0x00	R/W	Low 16-bits of counter
counter_hi	0x02	R/W	High 16-bits of counter
control	0x04	R/W	Control register
unlock1	0x08	W	Unlock 1
unlock2	0x0c	W	Unlock 2

Table 3: Register Map when BITS equals 16

Register	Address offset	Access	Description
counter	0x00	R/W	Counter register
control	0x04	R/W	Control register
unlock1	0x08	W	Unlock 1
unlock2	0x0c	W	Unlock 2

Table 4: Register Map when BITS equals 32

4.1.1 Counter Register

The counter register provides access to the 32-bit watchdog counter. The counter can only be written when the watchdog is in the unlocked state.

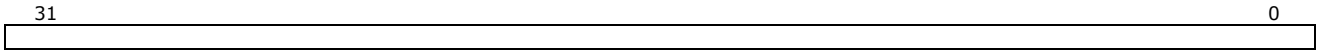

Figure 2: Format of the `counter` register

4.1.2 Counter Lo Register

The counter lo register provides access to the lower 16-bits of the 32-bit watchdog counter. The counter can only be written when the watchdog is in the unlocked state. When the counter lo register is written, the lower 16-bits of the counter are set to the value that is written to the counter lo register, while the upper 16-bits of the counter are set to the value held in a 16-bit buffer that is accessed via the counter hi register.

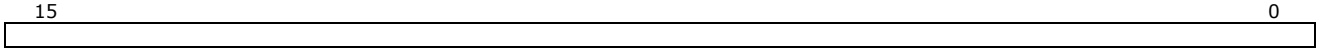

Figure 3: Format of the `counter_lo` register

4.1.3 Counter Hi Register

The counter hi register provides indirect access to the upper 16-bits of the 32-bit watchdog counter. When the counter lo register is read, the upper 16-bits of the counter are buffered, and the contents of this buffer are returned when the counter hi register is read. Similarly, when the counter hi register is written, the write value is written to this buffer. When the counter lo register is written, the contents of this buffer will be written to the upper 16-bits of the counter.

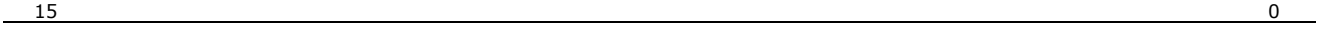

Figure 4: Format of the `counter_hi` register

4.1.4 Control Register

The control register contains a flag that enables the watchdog. The control register can only be written when the watchdog is in the unlocked state.

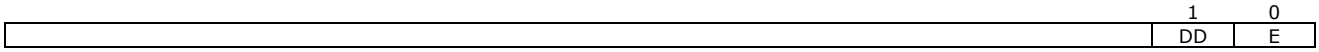

Figure 5: Format of the `control` register

Register	Values	Description
E	0 - Disabled 1 - Enabled	Enables the watchdog
DD		Disable counter when debugger is active

Table 5: Fields of the `control` register

4.1.5 Unlock 1 Register

The unlock 1 register is a register that must be written in a specific sequence in order to unlock access to the other registers.

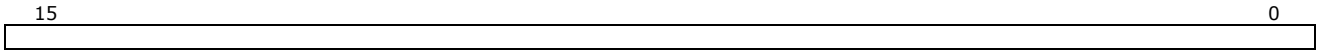

Figure 6: Format of the `unlock1` register

4.1.6 Unlock 2 Register

The unlock 2 register is a register that must be written in a specific sequence in order to unlock access to the other registers.

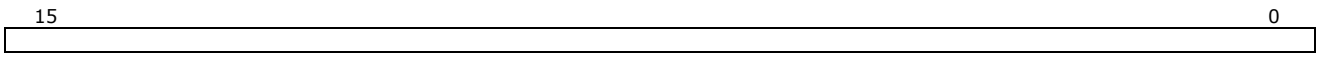

Figure 7: Format of the `unlock2` register

4.2 Interrupts

The watchdog supports a single interrupt. It will be raised when the watchdog is enabled, by the `E` flag in the `control` register having the value 1, and the `counter` having the value 0.

4.3 Watchdog Operation

In order to prevent a crashed program from accidentally writing any of the watchdogs registers, the watchdog contains a state machine, that will only allow writes to the `counter` or `control` registers, when in the unlocked state.

After reset, the state-machine will be in the locked state. In order to progress to the unlocked state, the follow sequence of registers must be written:

1. Write 0x1234 to `unlock1`.
2. Write 0x5665 to `unlock2`.
3. Write 0x4321 to `unlock1`.

Writing any other watchdog register part way through the sequence will result in the unlocked state not being reached. When in the unlocked state, writing any register will cause the state machine to reset to the locked state.

When the watchdog is enabled, the counter will be decremented until it reaches 0, when the interrupt will be raised. Therefore, the correct sequence of operations in order to use the watchdog is as follows:

1. Write the unlock sequence.
2. Write the desired timeout value to `counter`.
3. Write the unlock sequence.
4. Write 1 to `control.E` to enable the watchdog.
5. Perform operations that should take less that the specified timeout.
6. Write the unlock sequence.
7. Rewrite the desired timeout value to the `counter`.
8. Repeat from step 5.

5 Revision History

Hardware Revision	Software Release	Description
1	2.2.0	Initial release.
3	3.2.4	Config register in peripheral map indicates whether interrupt output drives NMI or reset.
4	6.0.2	Added `debug_active` input. Added `control.DD` field.

Table 6: Revision History

