PMU_overflow specification version v1

Guillem Cabo Pitarch

August 29, 2022

CONTENTS

1 General purpose of the module

This module shall detect if an overflow of a given counter is about to happen, if any counter reaches the maximum value an interrupt shall be risen by the overflow module. Only one interrupt is required, the counter causing the interrupt shall be identified by a signal called overflow interrupt vector (over_intr_vect_o) that will encode in one-hot fashion the offending counter or counters.

Overflow detection shall be optional for each individual counter, hardware for overflow detection shall be present for all the counters but software shall be able to control through an overflow interrupt mask (over_intr_mask_i) what counters can actually generate an interrupt.

A dedicated enable signal shall be provided. Enable is active high.

Interrupt can not be active if the unit is disabled. The unit is considered enabled if the unit is not in hard or soft reset and the enable is high.

The whole unit shall use combinational logic, inputs and outputs are registered externally at the rising edge of the clock. The required inputs and outputs will be wires, names, types and default sizes are provided in section 4.

2 Design placement

This modules is meant to be instantiated by the interface agnostic PMU (PMU_raw.sv). Only one instance of this module is required.

3 Parameters

This module requires two integer parameters.REG_WIDTH and N_COUNTERS.
REG_WIDTH defines the number of bits that a counter has.
N_COUNTERS defines the amount of counters that are available in the PMU.
Given the previous parameters the unit shall generate correct RTL without any manual modification to the source code.

4 Interface

Interface signals of the module are listed in table 4.1

Port Name	Direction	Width	Index	Comment	Comment Source
clk_i	INPUT	1	-	Global Clock Signal	module port
rstn_i	INPUT	1	-	Global Reset Signal. This Signal...	module port
softrst_i	INPUT	1	-	Active HIGH	module port
en_i	INPUT	1	-	Active HIGH	module port
counter_regs_i	INPUT	288	[0:8][31:0]	Input wire from wrapper containi...	module port
over_intr_mask_i	INPUT	9	[8:0]	updated either	module port
intr_overflow_o	OUTPUT	1	-	Global interrupt overflow	module port
over_intr_vect_o	OUTPUT	9	[8:0]	Output of the Overflow interrupt...	module port

Table 4.1: Ports of module PMU_overflow

5 RESET BEHAVIOR

The module contains two reset signals. One synchronous active high reset enabled by software called softrst_i, the other is the global reset, it is asynchronous and active low.

Any reset when active shall clear any internal register and set them to 0. Interruptions shall become inactive while the unit is in reset.

6 WHAT COULD NOT HAPPEN

All signals except registers used to track previous state of the interrupts shall not be registered.

Interruptions shall be low while the unit is in reset.

Overflow interruption vector shall not decrease in value unless the unit is reset or soft-reset.

The overflow interruption shall not decrease unless the unit is reset or soft-reset.

7 BEHAVIOR

The module identifies the overflow of a counter by performing an and reduction of the incoming counter signals. Detected overflowing counters will only be signaled to the output after applying a bit by bit and operation between the detected overflow signal and the current overflow mask.

The unit is considered disabled when rstn_i is 0 or softrst_i is 1 or en_i is 0.

The previous state of the interrupt vector is stored to prevent missing the interrupt once the counter overflows. The output interrupt is te result of an or reduction of the masked overflow signals (masked_overflow) and the past interruption vector (past_intr_vect) if the unit is enabled, otherwise the output is 0. In other words, if any bit in the past interruption vector or masked overflow signals the interuption while the unit is active the interrupt is active.

7.1 PACKAGES AND STRUCTURES

No packages and structures are used in this module.

8 SPECIAL CASES, CORNER CASES

If the unit is disabled instead but soft reset is never activated the previous state of the interrupt vector will not be cleared and therefore the interrupt will trigger again as soon as the unit is enabled. Since the internal state is not visible for software developers if not used correctly it may lead to unexpected behavior.