PMU_quota specification version v1

Guillem Cabo Pitarch

August 29, 2022

CONTENTS

1 General purpose of the module

This module checks the content of the counters selected by the mask register. The value of the selected counters is added and compared against the quota limit register. If the total quota is exceeded then an interrupt is risen. A single interrupt is generated for this module.

To account for a given event a 1 must be set to the Quota mask register. When the mask is 0 the value is not accounted for the quota.

The addition of counters is sequential, therefore you could notice up to N cycles (where N is the number of counters) from the point that the quota is exceeded until the point where the interrupt is risen.

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.

A local parameter called max_width sets the width of the internal adder. max_width shall guaranty that the addition of all counters will not overflow the internal counter.

4 Interface

Interface signals of the module are listed in table 4.1

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, called rstn_i it is asynchronous and active

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
counter_value_i	INPUT	288	[0:8][31:0]	Input wire from wrapper containi...	module port
softrst_i	INPUT	1	-	Active HIGH	module port
quota_limit_i	INPUT	32	[31:0]	sum_all_counters (counter_value_...	module port
quota_mask_i	INPUT	9	[8:0]	total quota that triggers the in...	module port
intr_quota_o	OUTPUT	1	-	Interrupt quota	module port

Table 4.1: Ports of module PMU_quota

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

Counter values are registered externally, and they shall not be registered. Addition of registers can be stored internally.

Interruptions shall be low while the unit is in reset.

The internal adder stores the addition of all counters in suma_int. Suma_int shall not overflow even if all counters have the maximum value and they have to be added.

7 Behavior

7.1 Counter masking

Counters are only added to compute towards consumed quota if the corresponding bit is active in the quota_mask_i signal. This signals is one-hot encoded, the LSB corresponds to the counter 0 and the MSB to the last counter.

The mask bit for each counter is replicated to match the reg_width. An and mask is applied between the replicated signal and the incomming counter value (counter_value_i). The outcome is routed to masked_counter_value_int. If the unit is in reset masked_counter_value_int entries remain 0.

7.2 SEQUENTIAL ADDITION

In order to minimize resources a single adder is used and values of counters are added sequentially.

Three structures are needed. First we have a **mask change detection** mechanism, the current mask is stored when the unit is active and compared against the previous value, if the stored mask (old_mask) is different than the incoming mask (quota_mask_i) a one bit signal called new_mask is set high. The second block is a simple **control state machine**, if we are in reset or a new mask is detected the state jumps to 0, otherwise the state increase each cycle. When the state machine count (state_int) reaches the N_COUNTERS value it is set back to 0. Finally we have the actual **sequential addition of counters**, it is stored in suma_int. If the unit is in reset the addition is set to 0, if a new mask or the state-machine in the second block is 0 the addition is set to 0. Otherwise suma_int adds its previous value to masked_counter_value_int[n], where n is the value of the state-machine in block two minus one (state_int-1).

7.3 INTERRUPTION

The interruption control is simple. It holds the previous state of the interrupt in a register called hold_intr_quota. This register is set to 0 at reset, otherwise it applies an or operation between its previous value and the current interrupt output (intr_quota_o).

The output interrupt (intr_quota_o) is high if the sequential addition of counters (suma_int) is larger than the input quota limit (quota_limit_i) or the interrupt was triggered since the last reset (hold_intr_quota is high).

To clear the interrupt a soft or hard reset shall be performed.

7.4 PACKAGES AND STRUCTURES

No packages and structures are used in this module.

8 SPECIAL CASES, CORNER CASES

If the quota mask is changed the state-machine that adds the content of the counters shall jump to reset. This will discard the previous addition and start over.

The number of N_COUNTERS shall remain reasonable for each specific implementation of the PMU, large values will increase the latency between the quota been exceeded and the trigger of the interrupt.

If the values feed into the unit through counter_value_i overflow there is the chance that they overflow after been added to suma_int. As a result the interrupt of quota may trigger but the values of the counters at that instant may be lower than the limit of quota.