SafePMU Fault Tolerance Measures

Guillem Cabo Pitarch

June 29, 2021

CONTENTS

1 OVERVIEW

The SafePMU is capable of changing the execution of critical tasks, either if a control kernel is using its measures to perform software control or by the use of interrupt service routines signaled by the unit. Given this scenario the unit shall be analyzed to detect possible failure modes. We focus our efforts on failures due to single event upsets (SEU) and single transient effects (SET) that can be mitigated at RTL level. Additional measures can be taken at a physical level such us the use of hardened memory cells on ASIC or periodical reconfiguration on FPGAs, but they shall be undertaken by the IP integrator in a project by project basis.
Failure modes and fault tolerance measures have been analyzed for each RTL file. Common considerations among files and features are described under the general section.

- **pmu_ahb.sv:** Interface with AHB bus. Contains a PMU values and configuration registers, state machines for AHB control, combinational logic to manage register updates.
- **PMU_raw.sv:** Signal routing among instances, some signals with combinational logic for enables, IMPLEMENTATION OF SELFTEST MODE, registers RDC and MCCU signals.
- **PMU_counters.sv:** Internally registered counter values, combinational logic for adders and external update control.
- **crossbar.sv:** Externally driven muxes and registered outputs.
- **MCCU.sv:**Internally registered quotas. Capable of signaling interrupts.
- **PMU_overflow:** Mostly combinational with several internal registers. Interruption capable.
- **RDC.sv:** Mainly combinational logic but it has several internal registers. Capable of signaling interrupts.

The following sections describe possible failure modes and potential mitigations. Each solution will result in a tradeoff between performance, resources, ease of use, and development time. Thus the recommended reliefs may change at a later date to match project goals.

2 FAIL MODES AND PROPOSED CORRECTIVE ACTIONS

2.1 GENERAL

2.1.1. Fail mode: Single transient events on rstn_i will upset the whole design, this applies to all modules with resets sensitive to falling edges of the reset. **Very high priority**.
Corrective action: Replace asynchronous reset with synchronous reset.

2.1.2. Fail mode: Event generators (Signal gathering on SoC) have not been hardened. Purely combinational CCS signal generators that suffer a transient event have little consequences, counts may be out by one. CCS signals that depend on sequential logic may be analyzed on further detail, since upsets may cause prolonged misbehavior on the resulting event signal. **Medium priority**
Corrective action: Protect sequential logic with fault detection and user correction or hardware error correction and recovery. Transient errors on combinational logic are allowed due to their overall low impact, given the low probability of errors.

2.2 PMU_AHB

2.2.1. Fail mode: Failure on AHB external signals may cause miss behaviors thought the unit. Transient errors on data, address or control signals may cause miss configuration, incorrect lectures and invalid internal states. AHB external signals could be hardened by Cobham Gaisler. **Low priority**.
Corrective action: The unit assumes good behavior, even after a fault, from external interfaces. Thus adequate fault detection or fault correction shall be provided by the SoC design.

2.2.2. Fail mode: Single event upset (SEU) on slv_reg configuration registers (Any feature) may cause complete unit failure. At least registers in range BASE_CFG, BASE_MCCU_CFG, BASE_MCCU_WEIGHTS, BASE_CROSSBAR to their respective END range shall be protected. Error detection is required. **High priority.**
Corrective action: In this case, either error detection or error correction is recommended. On a resource-restricted system, we could use a hash function that is updated at each cycle. If changes in the hash function are detected at any other time than after an AHB write transaction, the IP can signal an error interrupt. When hardware resources are available, error correction is recommended through all configuration registers since it simplifies the use of the IP.

2.2.3. Fail mode: Single event upset (SEU) on slv_reg result registers have different conse-

quences depending on the instant of the event and the state of the system. Event upsets during a write request on BASE_COUNTERS for instance may have a critical effect since it has effect over overflow interrupts, while the same upset on any idle state may be harmless since the upset will be cleared in the next cycle by the refresh of the unit. If only upsets on write are deem dangerous the issue can be handled by software forcing several reads after a write. The same happen for reads, temporal redundancy on software can mitigate the issue. Hardware solutions may be considered. **Priority medium**
Corrective action: It is recommended to perform a read after each write to the PMU in order to detect transient errors on writes.

2.2.4. Fail mode: State machines depend on internally registered signals such as *state*, *address_phase.select*. Upsets may result on misbehavior regarding AHB protocol but also internal updates. The number of bits seems to be small and hardware redundancy may be feasible. **Priority high**
Corrective action: Due to the small number of signals, it is recommended to triplicate the registers and implement a voting mechanism that allows error correction.

2.3 PMU_RAW

2.3.1. Fail mode: *MCCU_enable_int* and *RDC_enable_int* are internally registered. Reset is active low. While active, the values are updated each cycle based on the corresponding *regs_i* values. Given that *regs_i* has error detection against permanent errors on higher levels, transient faults may cause the unit to be disabled for a single cycle. **Priority Low**.
Corrective action: Since a transient error will be self-corrected in the following cycles, and the consequences of the failures are not deemed catastrophic, additional protection can be ignored for most of the systems. In extreme cases, delayed sampling of the bus could be used to detect and recover from transients.

2.3.2. Fail mode: *MCCU_rstn* and *RDC_rstn* are internally registered. Reset is active low. While active, the values are updated each cycle based on the corresponding *regs_i* values and current reset state. Transient errors on *regs_i* during the clock's positive edge can cause the propagation of unexpected resets. **Priority high**.
Corrective action: Internally registered signals shall be replicated. Protection against transients on regs_i can be provided by hardware at the driver side, detecting mismatches between the past output and the current output during periods without external updates. Another solution would be to add redundancy bits on the driver side and compare them at the receiver. On the receiver end, time-delayed sampling could be implemented. Only one mechanism is required.

2.3.3. Fail mode: The self-test configuration feature is implemented as a combinational block within this unit. While permanent failures are addressed at the signal source, the feature may still be affected by transient errors. If *selftest_mode* is disturbed, all the input events may take incorrect values for a single cycle. **Priority Low**.
Corrective action: Transient errors in these signals are a low priority since they will correct themselves. If transients need to be mitigated, error detection can be implemented at the driver side by checking that regs_i remains stable unless a new ahb write to the corresponding slv_reg registers occurs.

2.3.4. Fail mode: This module's primary purpose is signal routing, thus point-to-point connections between the ports of *PMU_raw* and each of the PMU features. Given the combinatorial nature of the circuit, transient faults can occur. Such fail modes shall be considered and mitigated at the signals destination.
Corrective action: Consumers of PMU_raw signals shall include transient error detection if needed. Detection/correction can be placed at the source or destination of the signals.

2.4 PMU_COUNTERS

2.4.1. Fail mode: *Softrst_i* transient errors can occur. The unit shall handle error detection internally. One error can reset all counters. **Priority High**.
Corrective action: Provide redundant signals from the source of Softrst_i and do error detection within the module.

2.4.2. Fail mode: *en_i* transient errors can occur. The unit shall handle error detection internally. One error can disable the counters for a single cycle. **Priority low**.
Corrective action: Given that normal operation will be recovered after the transient priority is low. Nevertheless, en_i can be replicated at the source. Error correction mechanisms can be added if needed.

2.4.3. Fail mode: *we_i* transient errors can cause the unit to miss a user update to the counter values. Counter values are internally registered as *sly_reg* and mirrored in the wrapper interface (*pmu_ahb.sv*). Missing a we_i may cause metastability. If *we_i* is altered, the new value is not bypassed. In this scenario, the contents of the mirror and internal registers diverge. This failure mode is hard to detect since the unit will always swap two values around the mirror and the internal register. **Priority high**.
Corrective action: Add a hardware check to detect if counter values decrease or increase

by more than one without any reasonable cause such as reset, write, overflow. On a constrained system, error detection can be added by checking parity of the current value with a single-bit counter set and reset accordingly with the counter's initial values. As an alternative, extend the we_i signal with error recovery mechanisms such as replication and voting.

2.5 CROSSBAR

2.5.1. Fail mode: Transient errors on *vector_i* may cause routing faults. Input events may end up assigned to the incorrect output for a single cycle. **Priority low**.
Corrective action: Since transients shall occur at the clock's rising edge to allow the counters to register incorrect events, and the consequence is adding or dropping one event occurrence, priority is low. If needed, resource-constrained systems could compare a hash of vector_i could be recorded and compared with the source register after each cycle. When an error is detected, the unit could sign an interrupt.

2.5.2. Fail mode: *vector_o* is internally registered. Values are updated at each cycle. Upsets may cause an incorrect output for a single cycle, but the failure will be cleared afterward. **Priority low**.
Corrective action: Since vector_o is updated at each cycle, no further action shall be taken to correct the upsets. Upsets may cause a single event to have an unreliable value for a single cycle. Software safety margins shall account for such small tolerances.

2.6 MCCU

2.6.1. Fail mode: Transient errors on *enable_i* may cause unintended updates of *quota_int* or missing updates. Propagation of transient errors on *interruption_quota_o*. **Priority medium**.
Corrective action: *enable_i* could be replicated or registered and compared with the source in the following cycle.

2.6.2. Fail mode: Transient errors on *events_i* may cause *events_weights_int* to have an incorrect value. This propagates to *ccc_suma_int* and *interruption_quota_o*. As far as failures are uncommon enough, faults can be absorbed by the margins implemented on the MCCU quota limits. **Priority Low**.
Corrective action: Upsets may cause a single event to have an unreliable value for a sin-

gle cycle. Software safety margins shall account for such small tolerances.

2.6.3. Fail mode: *quota_i* can suffer from transient errors. If such an error happens while *enable_i* is low and *update_quota_i* is high, Fail mode: *quota_int* will get miss configured. Users can detect faults by reading *quota_o*. A transient error may cause incorrect interrupts. **Priority medium.**
Corrective action: Configuration registers shall be read after a wite on the software side to ensure correct configuration.

2.6.4. Fail mode: *quota_int* is an internal register. It is forwarded to top modules with *quota_o*. *quota_int* can suffer permanent faults that are not cleared automatically. This failure mode may result in interrupts not triggering or triggering early. **Priority High**.
Corrective action: Replicating the internal register *quota_int* has a low overhead. Two instances will provide error detection, but tree instances are recommended, allowing for seamless recovery if a failure occurs.

2.6.5. Fail mode:*update_quota_i* transient errors can result in incorrect configurations due to unexpected or dropped updates. Misconfiguration affects *quota_int* and thus the resulting interrupts. **Low priority.**
Corrective action: Software can read quota_o values after each write to ensure no transients have occurred.

2.6.6. Fail mode: *quota_o* is a wire that takes the value of *quota_int*. Given that *quota_int* is protected against permanent upsets, a transient error on the output line may cause incorrect readings for a single cycle. *Quota_o* is not used as a control signal and does not affect interrupt generation. **Priority low**.
Corrective action: *quota_o* is signaled to the user-accessible registers to provide more information. Several readings could be performed in quick succession and determine if there was an update. Note that the values will be updated at each cycle if the unit is active. If transients over this signal are a real concern for a particular implementation, hardware error detection is recommended.

2.6.7. Fail mode: *events_weights_i* determines the contention impact of each MCCU input event. The source of this signal is the register bank in *pmu_ahb.sv*, and the source registers shall protect it against persistent errors. Transient errors could disturb the value of the weight for one signal. Such an event would cause quota mismatches of ±128 cycles over the intended value for a single event upsets. **Priority Low**.

Corrective action: Since the source register would have error detection and correction, transient errors would have a small effect. Software shall account for sporadical errors within the safety margins of the application.

2.6.8. Fail mode: *ccc_suma_int* contains the addition of all active events at a particular cycle. The value is updated at every cycle based on the incoming events and weights. The value is used to determine the interrupt value and remaining quota. Errors could significantly change the available quota if the bit-flip occurs on the MSB or trigger unintended interrupts. The potential severity of the error depends on the number of input events and the register's width. **Priority High**.
Corrective action: It is recommended to add error detection or correction by replication of the register.

2.7 PMU_OVERFLOW

2.7.1. Fail mode: *softrst_i* is signaled from a register outside the module. A permanent fault on this signal can render the unit disabled or rise unexpected interrupts. Permanent failures shall be prevented at the source register. **Priority High**.
Corrective action: Source register shall provide error detection or correction based on its particular implementation.

2.7.2. Fail mode: *softrst_i* transient errors can clear the interruption vector if the error aligns with the clock's positive edge. **Priority Low**
Corrective action: Reset signals could add redundancy bit to determine the intended value regardless of an upset. Error recovery is recommended.

2.7.3. Fail mode: *en_i* is signaled from a register outside the module. A permanent fault on this signal can render the unit disabled or rise unexpected interrupts. Permanent failures shall be prevented at the source register. **Priority High.**
Corrective action: Enable signals could add redundancy bit to determine the intended value regardless of an upset. Error recovery is recommended.

2.7.4. Fail mode: *en_i* transient errors can cause glitches on interrupts. Since *en_i* determines the value of *unit_disabled* and, consequently, the value of *intr_overflow_o* and *over_intr_vect_o* . **Priority Low**
Corrective action: Add redundancy bit to determine the intended value regardless of an

upset. Recovery is recommended over detection to avoid conflicts of priority between interrupts.

2.7.5. Fail mode: *counter_regs_i* can suffer from transient errors, and as a consequence, trigger or hide overflow signals. Most of the temporal errors would result in harmless scenarios that will correct themselves in the following cycles. Nevertheless, there is the potential to miss an overflow if the transient occurs while the counter reaches the maximum value, such scenario may but have severe effects. **Priority medium**.
Corrective action:

2.7.6. Fail mode: *over_intr_mask_i* is signaled from a register outside the module. A permanent fault on this signal can render the unit disabled or rise unexpected interrupts. **Priority high**.
Corrective action: Permanent failures shall be prevented at the source register.

2.7.7. Fail mode: *over_intr_mask_i* transient errors can cause glitches on interrupts. Transient errors can change the values of *past_intr_vect* (inducing a permanent error) by enabling overflow detection on signals that are not intended to be monitored. The transient shall occur at the clock's rising edge and trigger quota monitoring on a counter about to overflow. **Priority medium**.
Corrective action: The failure mode is considered unlikely, and results on a fail safe scenario that could be handled by software. If needed hardware error detection can be added to *over_intr_mask_i* by checking a hash of the signal and the source register.

2.7.8. Fail mode: overflow transient errors can cause glitches on interrupts. Transient errors can change the values of *past_intr_vect* (inducing a permanent error) by recording unexpected interrupts on monitored counters. An error shall align with a positive edge of the clock. **Priority medium**.
Corrective action: Overflow signal could be replicated and voted. Since it is one bit width for each counter hardware cost shall be acceptable.

2.7.9. Fail mode: *unit_disabled* transient errors can cause glitches on interrupts. Since it determines the value of *intr_overflow_o* and *over_intr_vect_o* with combinational logic. **Priority low**.
Corrective action: Interrupts remain high until they are cleared by software. Thus, a transient error may delay the actions of the processor by a cycle but will not cause significant consequences. No action is required.

2.7.10. Fail mode: *intr_overflow_o* and *over_intr_vect_o* are susceptible to transients and can generate glitches on interrupt lines. **Priority low**.
Corrective action: Interrupts remain high until they are cleared by software. Thus, a transient error may delay the actions of the processor by a cycle but will not cause significant consequences. No action is required.

2.8 RDC

2.8.1. Fail mode: *enable_i* is signaled from a register outside the module. A permanent fault on this signal can render the unit disabled or rise unexpected interrupts. Permanent failures shall be prevented at the source register. **Priority High**.
Corrective action: Fault-tolerance shall be granted by the source of the signal.

2.8.2. Fail mode: *enable_i* transient errors can cause permanent errors on *interruption_vector_rdc_o*, *past_interruption_rdc_o* , *max_value* and *watermark_int* if transients align with the clock. **Priority high**.
Corrective action: Enable signals could add redundancy bit to determine the intended value regardless of an upset. Error recovery is recommended.

2.8.3. Fail mode: *events_i* transient errors can cause discrepancies on *max_value*. After a transient, depending on its nature, *max_value* can contain the value of two consecutive events or a fraction of the actual event. As a consequence, *watermark_int* and *interruption_vector_int* may be disturbed. Transients must align with the positive edge of the clock. **Priority low**.
Corrective action: Disturbances on the events would need to align with an exceedance of the event's max value to cause a problem. Most of the systems shall be able to accommodate such failure safely. Otherwise, events_i shall be hardened at the source.

2.8.4. Fail mode: *events_weights_i*, given protection against permanent faults on the source register, can suffer transient errors that may produce glitches on *interruption_vector_int*. **Priority low**.
Corrective action: A glitch on events_weights_i can delay interrupts for one cycle or trigger an unexpected RDC interrupt. Overall the failure will be harmless, but it will require processing an additional interrupt. No corrective measures are required.

2.8.5. Fail mode: *interruption_rdc_o* can suffer transients that will generate a glitch on the interrupt line. **Priority low**.
Corrective action: Overall the failure will be harmless, but it will require processing an unexpected interrupt. No corrective measures are needed.

2.8.6. Fail mode: *interruption_vector_rdc_o* can suffer of permanent faults. The current value of the signal depends on the previous, so faults have the potential to remain present over time. **Priority high**.
Corrective action: Given the size of interruption_vector_rdc_o it is recommended to add triple redundancy to enable error recovery.

2.8.7. Fail mode: *watermark_o* is a wire coming from register *watermark_int*. Bitflips on the second register can produce permanent faults if the resulting value is higher than *max_value*. This may lead to incorrect profiling. **Priority High**.
Corrective action: Error detection by duplicating the watermark registers is recommended.

2.8.8. Fail mode: *past_interruption_rdc_o* is a registered signal, and holds the previous state of the RDC interrupt. Its content can be affected by transients on *rstn_i* and *enable_i*. The error could clear RDC interrupts without user notice. **Priority medium**.
Corrective action: Given the small size, adding error correction with redundancy and a voting mechanism is recommended.

2.8.9. Fail mode: *max_value* is an internal register. Its value depends on the previous value of its own. It has the potential to hold permanent upsets. Such fail mode could cause *interruption_rdc_o* to become active and induce errors on the watermark register. **Priority high**.
Corrective action: max_value can only transition from its current value to the value plus one or to zero. So most upsets can be detected if a different transition is detected. Once the error is detected an interrupt can be signaled.

2.8.10. Fail mode: *interruption_vector_int* is an internal wire. Upsets on the signal can induce permanent upsets on *interruption_vector_rdc_o* if they occur at the rising edge of the clock. **Priority low**.
Corrective action: Given a failure in this signal, the most likely scenario is to trigger an unexpected interrupt or delay by one a legitimate interrupt. Both results can be accommodated by most of the systems, and no further action is required.

3 DEPENDENCIES

3.1 PMU_AHB

Correct behavior depends on external AHB signals, event generators and *PMU_raw* outputs.

3.2 PMU_RAW

It depends on *pmu_ahb,* and assumes that registers driving the inputs are fault-tolerant to upsets. Outputs of the module are affected by the correct behavior of *PMU_counters*, *PMU_overflow*, *MCCU*, and *PMU_counters*.

3.3 PMU_COUNTERS

The module depends on the correctness of *pmu_ahb* configuration register *slv_reg* and glitch-less propagation of all module inputs through *PMU_raw*.

3.4 CROSSBAR

The module depends on the correctness of *pmu_ahb* configuration register *slv_reg* and glitch-less propagation of all module inputs through *PMU_raw*.

3.5 MCCU

This module depends on the correctness of *pmu_ahb* configuration register *slv_reg* and glitch-less propagation of all module inputs through *PMU_raw*.

3.6 PMU_OVERFLOW

This module depends on the correctness of *pmu_ahb* configuration register *slv_reg*. Glitch-less propagation of all module inputs through *PMU_raw*, and correct behavior of *PMU_counters* is assumed.

3.7 RDC

This module depends on the correctness of *pmu_ahb* configuration register *slv_reg* and glitch-less propagation of all module inputs through *PMU_raw*.

4 Fault Tolerance IPs

4.1 Parity encoder / decoder

4.2 Hamming encoder / decoder

4.3 Reed Solomon encoder / decoder

4.4 Triple simultaneous voter

4.5 Triple time delayed voter