

---

# SafeSU User's Manual

---

Guillem Cabo Pitarch

August 29, 2022

## CONTENTS

|  |           |
|--|-----------|
| <b>1 Overview</b>                              | <b>3</b>  |
| <b>2 Operation</b>                             | <b>4</b>  |
| 2.1 General . . . . .                          | 4         |
| 2.2 Default input events . . . . .             | 4         |
| 2.3 Main configuration and self-test . . . . . | 12        |
| 2.4 Crossbar . . . . .                         | 13        |
| 2.5 Counters . . . . .                         | 14        |
| 2.6 Overflow . . . . .                         | 17        |
| 2.7 Quota . . . . .                            | 18        |
| 2.8 MCCU . . . . .                             | 18        |
| 2.9 RDC . . . . .                              | 21        |
| 2.10 Software support . . . . .                | 23        |
| <b>3 Configuration options</b>                 | <b>24</b> |
| <b>4 Signal descriptions</b>                   | <b>24</b> |
| <b>5 Library dependences</b>                   | <b>25</b> |
| <b>6 Instantiation</b>                         | <b>25</b> |

# 1 OVERVIEW

The SafeSU (Safe Statistics Unit) is an AHB slave capable of monitoring SoC events, enforce contention control, and identifying profiling errors on run-time. Figure "1.1" shows the structure of the unit. It is composed of an ahb wrapper (*ahb\_wrapper.vhd*) that maps the SystemVerilog implementation into a VHDL module that can be instanced in SELENE and De-RISC SoCs. The SystemVerilog AHB interface (*pmu\_ahb.sv*) offers support for a subset of AHB requests. This module also instances the interface agnostic SafeSU (*PMU\_raw.sv*). The latter is used as the generator of the statistic unit. It generates the memory map and the instances for each of the features.

The main features are:

- **Self-test:** Allows to configure the counters' inputs to a fixed value bypassing the crossbar and ignoring the inputs. This mode allows for tests of the software and the unit under known conditions.
- **Crossbar:** Allows to route any input event to any counter.
- **Counters:** Group of simple counters with settable initial values and general control register.
- **Overflow:** Detection of overflow for counters, interrupt capable with dedicated interruption vector and per counter interrupt enable.
- **Quota:** Deprecated (*It may be excluded in a future release*)
- **MCCU** (Maximum Contention Control Unit) : Contention control measures for each core. Interruption capable after a threshold of contention is exceeded. It accepts real contention signals or estimation through weights.
- **RDC** (Request Duration Counters): Provides measures of the pulse length of a given input signal (watermark). It can be used to determine maximum latency and cycles of uninterrupted contentions. Each of the counters can trigger an interrupt at a user-defined threshold.

The default configuration of this unit supports 4 cores and 32 input signals. In future releases such parameters (VHDL generics) will be exposed to the top level.

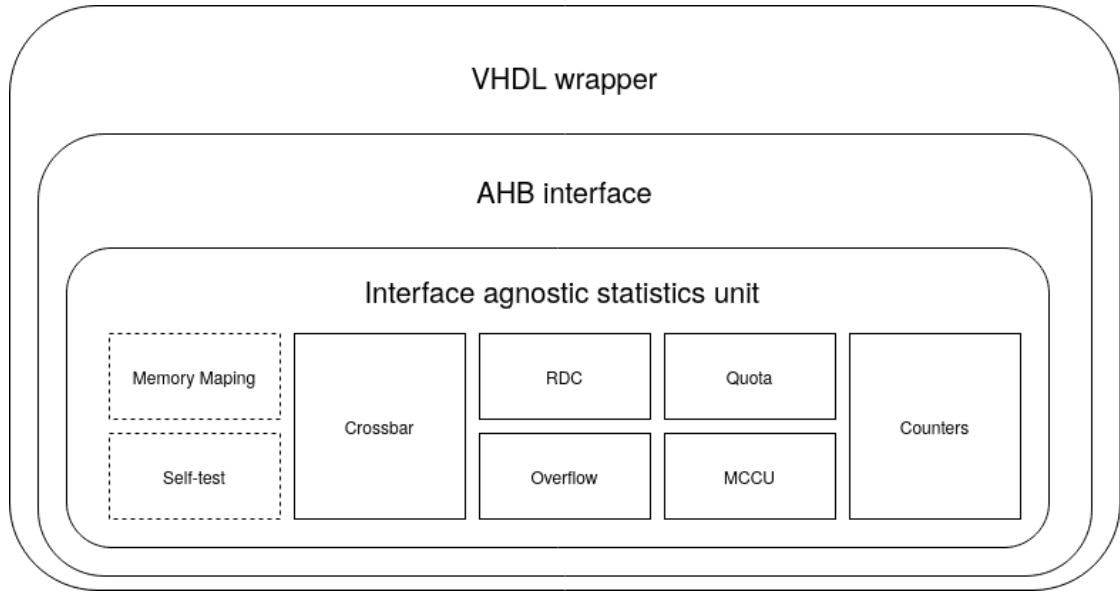


Figure 1.1: Block diagram statistics unit structure

## 2 OPERATION

### 2.1 GENERAL

The SafeSU attaches to a 128-bit wide AHB bus but only supports Single burst 32-bit accesses. AHB Lock accesses and protection control are not implemented.

The unit is compatible with GRLIB *plug&play*(P&P). Vendor and device's default configuration values are "BSC" and "AHB Performance Monitoring Unit" respectively. Regardless, release *v3.2.4* of GRMON, the unit may appear as "Unknown device".

### 2.2 DEFAULT INPUT EVENTS

In its default implementation, the unit provides up to 128 input events. Each one of the events can be routed to any mechanism of the module through the crossbar.

Table 2.1 shows the inputs and mapping to the crossbar input for the current SELENE release. The number of signals and arrangements may change in the upcoming versions.

Table 2.1: SELENE GPL input events. 256 inputs, 6 cores

| Index                  | Type  | Source | Description  |
|------------------------|-------|--------|--|
| 0                      | Debug | local  | Constant HIGH, used for debug purposes or clock cycles |
| 1                      | Debug | local  | Constant LOW, used for debug purposes                  |
| 2                      | Pulse | Core 0 | Instruction count pipeline 0                           |
| 3                      | Pulse | Core 0 | Instruction count pipeline 1                           |
| 4                      | Pulse | Core 0 | Instruction cache miss                                 |
| 5                      | Pulse | Core 0 | Instruction TLB miss                                   |
| 6                      | Pulse | Core 0 | Data chache L1 miss                                    |
| 7                      | Pulse | Core 0 | Data TLB miss  |
| 8                      | Pulse | Core 0 | Branch predictor miss                                  |
| 9                      | Pulse | Core 1 | Instruction count pipeline 0                           |
| 10                     | Pulse | Core 1 | Instruction count pipeline 1                           |
| 11                     | Pulse | Core 1 | Instruction cache miss                                 |
| 12                     | Pulse | Core 1 | Instruction TLB miss                                   |
| 13                     | Pulse | Core 1 | Data chache L1 miss                                    |
| 14                     | Pulse | Core 1 | Data TLB miss  |
| 15                     | Pulse | Core 1 | Branch predictor miss                                  |
| 16                     | Pulse | Core 2 | Instruction count pipeline 0                           |
| 17                     | Pulse | Core 2 | Instruction count pipeline 1                           |
| 18                     | Pulse | Core 2 | Instruction cache miss                                 |
| 19                     | Pulse | Core 2 | Instruction TLB miss                                   |
| 20                     | Pulse | Core 2 | Data chache L1 miss                                    |
| 21                     | Pulse | Core 2 | Data TLB miss  |
| 22                     | Pulse | Core 2 | Branch predictor miss                                  |
| 23                     | Pulse | Core 3 | Instruction count pipeline 0                           |
| 24                     | Pulse | Core 3 | Instruction count pipeline 1                           |
| 25                     | Pulse | Core 3 | Instruction cache miss                                 |
| 26                     | Pulse | Core 3 | Instruction TLB miss                                   |
| 27                     | Pulse | Core 3 | Data chache L1 miss                                    |
| 28                     | Pulse | Core 3 | Data TLB miss  |
| 29                     | Pulse | Core 3 | Branch predictor miss                                  |
| 30                     | Pulse | Core 4 | Instruction count pipeline 0                           |
| 31                     | Pulse | Core 4 | Instruction count pipeline 1                           |
| 32                     | Pulse | Core 4 | Instruction cache miss                                 |
| 33                     | Pulse | Core 4 | Instruction TLB miss                                   |
| 34                     | Pulse | Core 4 | Data chache L1 miss                                    |
| 35                     | Pulse | Core 4 | Data TLB miss  |
| 36                     | Pulse | Core 4 | Branch predictor miss                                  |
| 37                     | Pulse | Core 5 | Instruction count pipeline 0                           |
| 38                     | Pulse | Core 5 | Instruction count pipeline 1                           |
| Continued on next page |       |        |  |

**Table 2.1 – continued from previous page**

| <b>Index</b>           | <b>Type</b> | <b>Source</b> | <b>Description</b>      |
|------------------------|-------------|---------------|-------------------------|
| 39                     | Pulse       | Core 5        | Instruction cache miss  |
| 40                     | Pulse       | Core 5        | Instruction TLB miss    |
| 41                     | Pulse       | Core 5        | Data chache L1 miss     |
| 42                     | Pulse       | Core 5        | Data TLB miss           |
| 43                     | Pulse       | Core 5        | Branch predictor miss   |
| 44                     | CCS AHB     | -             | Agressor C1 Victim C0   |
| 45                     | CCS AHB     | -             | Agressor C2 Victim C0   |
| 46                     | CCS AHB     | -             | Agressor C3 Victim C0   |
| 47                     | CCS AHB     | -             | Agressor C4 Victim C0   |
| 48                     | CCS AHB     | -             | Agressor C5 Victim C0   |
| 49                     | CCS AHB     | -             | Agressor C0 Victim C1   |
| 50                     | CCS AHB     | -             | Agressor C2 Victim C1   |
| 51                     | CCS AHB     | -             | Agressor C3 Victim C1   |
| 52                     | CCS AHB     | -             | Agressor C4 Victim C1   |
| 53                     | CCS AHB     | -             | Agressor C5 Victim C1   |
| 54                     | CCS AHB     | -             | Agressor C0 Victim C2   |
| 55                     | CCS AHB     | -             | Agressor C1 Victim C2   |
| 56                     | CCS AHB     | -             | Agressor C3 Victim C2   |
| 57                     | CCS AHB     | -             | Agressor C4 Victim C2   |
| 58                     | CCS AHB     | -             | Agressor C5 Victim C2   |
| 59                     | CCS AHB     | -             | Agressor C0 Victim C3   |
| 60                     | CCS AHB     | -             | Agressor C1 Victim C3   |
| 61                     | CCS AHB     | -             | Agressor C2 Victim C3   |
| 62                     | CCS AHB     | -             | Agressor C4 Victim C3   |
| 63                     | CCS AHB     | -             | Agressor C5 Victim C3   |
| 64                     | CCS AHB     | -             | Agressor C0 Victim C4   |
| 65                     | CCS AHB     | -             | Agressor C1 Victim C4   |
| 66                     | CCS AHB     | -             | Agressor C2 Victim C4   |
| 67                     | CCS AHB     | -             | Agressor C3 Victim C4   |
| 68                     | CCS AHB     | -             | Agressor C5 Victim C4   |
| 69                     | CCS AHB     | -             | Agressor C0 Victim C5   |
| 70                     | CCS AHB     | -             | Agressor C1 Victim C5   |
| 71                     | CCS AHB     | -             | Agressor C2 Victim C5   |
| 72                     | CCS AHB     | -             | Agressor C3 Victim C5   |
| 73                     | CCS AHB     | -             | Agressor C4 Victim C5   |
| 74                     | CCS AXI     | Write         | Agressor MQ1 Victim MQ0 |
| 75                     | CCS AXI     | Write         | Agressor MQ2 Victim MQ0 |
| 76                     | CCS AXI     | Write         | Agressor MQ3 Victim MQ0 |
| 77                     | CCS AXI     | Write         | Agressor MQ4 Victim MQ0 |
| 78                     | CCS AXI     | Write         | Agressor MQ5 Victim MQ0 |
| Continued on next page |             |               |                         |

**Table 2.1 – continued from previous page**

| <b>Index</b>           | <b>Type</b> | <b>Source</b> | <b>Description</b>       |
|------------------------|-------------|---------------|--------------------------|
| 79                     | CCS AXI     | Write         | Agressor MQ6 Victim MQ0  |
| 80                     | CCS AXI     | Write         | Agressor MQ7 Victim MQ0  |
| 81                     | CCS AXI     | Write         | Agressor MQ8 Victim MQ0  |
| 82                     | CCS AXI     | Write         | Agressor MQ9 Victim MQ0  |
| 83                     | CCS AXI     | Write         | Agressor MQ10 Victim MQ0 |
| 84                     | CCS AXI     | Write         | Agressor MQ11 Victim MQ0 |
| 85                     | CCS AXI     | Write         | Agressor MQ12 Victim MQ0 |
| 86                     | CCS AXI     | Write         | Agressor MQ13 Victim MQ0 |
| 87                     | CCS AXI     | Write         | Agressor MQ14 Victim MQ0 |
| 88                     | CCS AXI     | Write         | Agressor MQ0 Victim MQ1  |
| 89                     | CCS AXI     | Write         | Agressor MQ2 Victim MQ1  |
| 90                     | CCS AXI     | Write         | Agressor MQ3 Victim MQ1  |
| 91                     | CCS AXI     | Write         | Agressor MQ4 Victim MQ1  |
| 92                     | CCS AXI     | Write         | Agressor MQ5 Victim MQ1  |
| 93                     | CCS AXI     | Write         | Agressor MQ6 Victim MQ1  |
| 94                     | CCS AXI     | Write         | Agressor MQ7 Victim MQ1  |
| 95                     | CCS AXI     | Write         | Agressor MQ8 Victim MQ1  |
| 96                     | CCS AXI     | Write         | Agressor MQ9 Victim MQ1  |
| 97                     | CCS AXI     | Write         | Agressor MQ10 Victim MQ1 |
| 98                     | CCS AXI     | Write         | Agressor MQ11 Victim MQ1 |
| 99                     | CCS AXI     | Write         | Agressor MQ12 Victim MQ1 |
| 100                    | CCS AXI     | Write         | Agressor MQ13 Victim MQ1 |
| 101                    | CCS AXI     | Write         | Agressor MQ14 Victim MQ1 |
| 102                    | CCS AXI     | Write         | Agressor MQ0 Victim MQ2  |
| 103                    | CCS AXI     | Write         | Agressor MQ1 Victim MQ2  |
| 104                    | CCS AXI     | Write         | Agressor MQ3 Victim MQ2  |
| 105                    | CCS AXI     | Write         | Agressor MQ4 Victim MQ2  |
| 106                    | CCS AXI     | Write         | Agressor MQ5 Victim MQ2  |
| 107                    | CCS AXI     | Write         | Agressor MQ6 Victim MQ2  |
| 108                    | CCS AXI     | Write         | Agressor MQ7 Victim MQ2  |
| 109                    | CCS AXI     | Write         | Agressor MQ8 Victim MQ2  |
| 110                    | CCS AXI     | Write         | Agressor MQ9 Victim MQ2  |
| 111                    | CCS AXI     | Write         | Agressor MQ10 Victim MQ2 |
| 112                    | CCS AXI     | Write         | Agressor MQ11 Victim MQ2 |
| 113                    | CCS AXI     | Write         | Agressor MQ12 Victim MQ2 |
| 114                    | CCS AXI     | Write         | Agressor MQ13 Victim MQ2 |
| 115                    | CCS AXI     | Write         | Agressor MQ14 Victim MQ2 |
| 116                    | CCS AXI     | Write         | Agressor MQ0 Victim MQ3  |
| 117                    | CCS AXI     | Write         | Agressor MQ1 Victim MQ3  |
| 118                    | CCS AXI     | Write         | Agressor MQ2 Victim MQ3  |
| Continued on next page |             |               |                          |

**Table 2.1 – continued from previous page**

| <b>Index</b>           | <b>Type</b> | <b>Source</b> | <b>Description</b>       |
|------------------------|-------------|---------------|--------------------------|
| 119                    | CCS AXI     | Write         | Agressor MQ4 Victim MQ3  |
| 120                    | CCS AXI     | Write         | Agressor MQ5 Victim MQ3  |
| 121                    | CCS AXI     | Write         | Agressor MQ6 Victim MQ3  |
| 122                    | CCS AXI     | Write         | Agressor MQ7 Victim MQ3  |
| 123                    | CCS AXI     | Write         | Agressor MQ8 Victim MQ3  |
| 124                    | CCS AXI     | Write         | Agressor MQ9 Victim MQ3  |
| 125                    | CCS AXI     | Write         | Agressor MQ10 Victim MQ3 |
| 126                    | CCS AXI     | Write         | Agressor MQ11 Victim MQ3 |
| 127                    | CCS AXI     | Write         | Agressor MQ12 Victim MQ3 |
| 128                    | CCS AXI     | Write         | Agressor MQ13 Victim MQ3 |
| 129                    | CCS AXI     | Write         | Agressor MQ14 Victim MQ3 |
| 130                    | CCS AXI     | Write         | Agressor MQ0 Victim MQ4  |
| 131                    | CCS AXI     | Write         | Agressor MQ1 Victim MQ4  |
| 132                    | CCS AXI     | Write         | Agressor MQ2 Victim MQ4  |
| 133                    | CCS AXI     | Write         | Agressor MQ3 Victim MQ4  |
| 134                    | CCS AXI     | Write         | Agressor MQ5 Victim MQ4  |
| 135                    | CCS AXI     | Write         | Agressor MQ6 Victim MQ4  |
| 136                    | CCS AXI     | Write         | Agressor MQ7 Victim MQ4  |
| 137                    | CCS AXI     | Write         | Agressor MQ8 Victim MQ4  |
| 138                    | CCS AXI     | Write         | Agressor MQ9 Victim MQ4  |
| 139                    | CCS AXI     | Write         | Agressor MQ10 Victim MQ4 |
| 140                    | CCS AXI     | Write         | Agressor MQ11 Victim MQ4 |
| 141                    | CCS AXI     | Write         | Agressor MQ12 Victim MQ4 |
| 142                    | CCS AXI     | Write         | Agressor MQ13 Victim MQ4 |
| 143                    | CCS AXI     | Write         | Agressor MQ14 Victim MQ4 |
| 144                    | CCS AXI     | Write         | Agressor MQ0 Victim MQ5  |
| 145                    | CCS AXI     | Write         | Agressor MQ1 Victim MQ5  |
| 146                    | CCS AXI     | Write         | Agressor MQ2 Victim MQ5  |
| 147                    | CCS AXI     | Write         | Agressor MQ3 Victim MQ5  |
| 148                    | CCS AXI     | Write         | Agressor MQ4 Victim MQ5  |
| 149                    | CCS AXI     | Write         | Agressor MQ6 Victim MQ5  |
| 150                    | CCS AXI     | Write         | Agressor MQ7 Victim MQ5  |
| 151                    | CCS AXI     | Write         | Agressor MQ8 Victim MQ5  |
| 152                    | CCS AXI     | Write         | Agressor MQ9 Victim MQ5  |
| 153                    | CCS AXI     | Write         | Agressor MQ10 Victim MQ5 |
| 154                    | CCS AXI     | Write         | Agressor MQ11 Victim MQ5 |
| 155                    | CCS AXI     | Write         | Agressor MQ12 Victim MQ5 |
| 156                    | CCS AXI     | Write         | Agressor MQ13 Victim MQ5 |
| 157                    | CCS AXI     | Write         | Agressor MQ14 Victim MQ5 |
| 158                    | CCS AXI     | Read          | Agressor MQ1 Victim MQ0  |
| Continued on next page |             |               |                          |



**Table 2.1 – continued from previous page**

| <b>Index</b>           | <b>Type</b> | <b>Source</b> | <b>Description</b>       |
|------------------------|-------------|---------------|--------------------------|
| 159                    | CCS AXI     | Read          | Agressor MQ2 Victim MQ0  |
| 160                    | CCS AXI     | Read          | Agressor MQ3 Victim MQ0  |
| 161                    | CCS AXI     | Read          | Agressor MQ4 Victim MQ0  |
| 162                    | CCS AXI     | Read          | Agressor MQ5 Victim MQ0  |
| 163                    | CCS AXI     | Read          | Agressor MQ6 Victim MQ0  |
| 164                    | CCS AXI     | Read          | Agressor MQ7 Victim MQ0  |
| 165                    | CCS AXI     | Read          | Agressor MQ8 Victim MQ0  |
| 166                    | CCS AXI     | Read          | Agressor MQ9 Victim MQ0  |
| 167                    | CCS AXI     | Read          | Agressor MQ10 Victim MQ0 |
| 168                    | CCS AXI     | Read          | Agressor MQ11 Victim MQ0 |
| 169                    | CCS AXI     | Read          | Agressor MQ12 Victim MQ0 |
| 170                    | CCS AXI     | Read          | Agressor MQ13 Victim MQ0 |
| 171                    | CCS AXI     | Read          | Agressor MQ14 Victim MQ0 |
| 172                    | CCS AXI     | Read          | Agressor MQ0 Victim MQ1  |
| 173                    | CCS AXI     | Read          | Agressor MQ2 Victim MQ1  |
| 174                    | CCS AXI     | Read          | Agressor MQ3 Victim MQ1  |
| 175                    | CCS AXI     | Read          | Agressor MQ4 Victim MQ1  |
| 176                    | CCS AXI     | Read          | Agressor MQ5 Victim MQ1  |
| 177                    | CCS AXI     | Read          | Agressor MQ6 Victim MQ1  |
| 178                    | CCS AXI     | Read          | Agressor MQ7 Victim MQ1  |
| 179                    | CCS AXI     | Read          | Agressor MQ8 Victim MQ1  |
| 180                    | CCS AXI     | Read          | Agressor MQ9 Victim MQ1  |
| 181                    | CCS AXI     | Read          | Agressor MQ10 Victim MQ1 |
| 182                    | CCS AXI     | Read          | Agressor MQ11 Victim MQ1 |
| 183                    | CCS AXI     | Read          | Agressor MQ12 Victim MQ1 |
| 184                    | CCS AXI     | Read          | Agressor MQ13 Victim MQ1 |
| 185                    | CCS AXI     | Read          | Agressor MQ14 Victim MQ1 |
| 186                    | CCS AXI     | Read          | Agressor MQ0 Victim MQ2  |
| 187                    | CCS AXI     | Read          | Agressor MQ1 Victim MQ2  |
| 188                    | CCS AXI     | Read          | Agressor MQ3 Victim MQ2  |
| 189                    | CCS AXI     | Read          | Agressor MQ4 Victim MQ2  |
| 190                    | CCS AXI     | Read          | Agressor MQ5 Victim MQ2  |
| 191                    | CCS AXI     | Read          | Agressor MQ6 Victim MQ2  |
| 192                    | CCS AXI     | Read          | Agressor MQ7 Victim MQ2  |
| 193                    | CCS AXI     | Read          | Agressor MQ8 Victim MQ2  |
| 194                    | CCS AXI     | Read          | Agressor MQ9 Victim MQ2  |
| 195                    | CCS AXI     | Read          | Agressor MQ10 Victim MQ2 |
| 196                    | CCS AXI     | Read          | Agressor MQ11 Victim MQ2 |
| 197                    | CCS AXI     | Read          | Agressor MQ12 Victim MQ2 |
| 198                    | CCS AXI     | Read          | Agressor MQ13 Victim MQ2 |
| Continued on next page |             |               |                          |

**Table 2.1 – continued from previous page**

| <b>Index</b>           | <b>Type</b> | <b>Source</b> | <b>Description</b>       |
|------------------------|-------------|---------------|--------------------------|
| 199                    | CCS AXI     | Read          | Agressor MQ14 Victim MQ2 |
| 200                    | CCS AXI     | Read          | Agressor MQ0 Victim MQ3  |
| 201                    | CCS AXI     | Read          | Agressor MQ1 Victim MQ3  |
| 202                    | CCS AXI     | Read          | Agressor MQ2 Victim MQ3  |
| 203                    | CCS AXI     | Read          | Agressor MQ4 Victim MQ3  |
| 204                    | CCS AXI     | Read          | Agressor MQ5 Victim MQ3  |
| 205                    | CCS AXI     | Read          | Agressor MQ6 Victim MQ3  |
| 206                    | CCS AXI     | Read          | Agressor MQ7 Victim MQ3  |
| 207                    | CCS AXI     | Read          | Agressor MQ8 Victim MQ3  |
| 208                    | CCS AXI     | Read          | Agressor MQ9 Victim MQ3  |
| 209                    | CCS AXI     | Read          | Agressor MQ10 Victim MQ3 |
| 210                    | CCS AXI     | Read          | Agressor MQ11 Victim MQ3 |
| 211                    | CCS AXI     | Read          | Agressor MQ12 Victim MQ3 |
| 212                    | CCS AXI     | Read          | Agressor MQ13 Victim MQ3 |
| 213                    | CCS AXI     | Read          | Agressor MQ14 Victim MQ3 |
| 214                    | CCS AXI     | Read          | Agressor MQ0 Victim MQ4  |
| 215                    | CCS AXI     | Read          | Agressor MQ1 Victim MQ4  |
| 216                    | CCS AXI     | Read          | Agressor MQ2 Victim MQ4  |
| 217                    | CCS AXI     | Read          | Agressor MQ3 Victim MQ4  |
| 218                    | CCS AXI     | Read          | Agressor MQ5 Victim MQ4  |
| 219                    | CCS AXI     | Read          | Agressor MQ6 Victim MQ4  |
| 220                    | CCS AXI     | Read          | Agressor MQ7 Victim MQ4  |
| 221                    | CCS AXI     | Read          | Agressor MQ8 Victim MQ4  |
| 222                    | CCS AXI     | Read          | Agressor MQ9 Victim MQ4  |
| 223                    | CCS AXI     | Read          | Agressor MQ10 Victim MQ4 |
| 224                    | CCS AXI     | Read          | Agressor MQ11 Victim MQ4 |
| 225                    | CCS AXI     | Read          | Agressor MQ12 Victim MQ4 |
| 226                    | CCS AXI     | Read          | Agressor MQ13 Victim MQ4 |
| 227                    | CCS AXI     | Read          | Agressor MQ14 Victim MQ4 |
| 228                    | CCS AXI     | Read          | Agressor MQ0 Victim MQ5  |
| 229                    | CCS AXI     | Read          | Agressor MQ1 Victim MQ5  |
| 230                    | CCS AXI     | Read          | Agressor MQ2 Victim MQ5  |
| 231                    | CCS AXI     | Read          | Agressor MQ3 Victim MQ5  |
| 232                    | CCS AXI     | Read          | Agressor MQ4 Victim MQ5  |
| 233                    | CCS AXI     | Read          | Agressor MQ6 Victim MQ5  |
| 234                    | CCS AXI     | Read          | Agressor MQ7 Victim MQ5  |
| 235                    | CCS AXI     | Read          | Agressor MQ8 Victim MQ5  |
| 236                    | CCS AXI     | Read          | Agressor MQ9 Victim MQ5  |
| 237                    | CCS AXI     | Read          | Agressor MQ10 Victim MQ5 |
| 238                    | CCS AXI     | Read          | Agressor MQ11 Victim MQ5 |
| Continued on next page |             |               |                          |

**Table 2.1 – continued from previous page**

| <b>Index</b> | <b>Type</b> | <b>Source</b> | <b>Description</b>        |
|--------------|-------------|---------------|---------------------------|
| 239          | CCS AXI     | Read          | Agressor MQ12 Victim MQ5  |
| 240          | CCS AXI     | Read          | Agressor MQ13 Victim MQ5  |
| 241          | CCS AXI     | Read          | Agressor MQ14 Victim MQ5  |
| 242          | -           | -             | Filler signal, constant 0 |
| 243          | -           | -             | Filler signal, constant 0 |
| 244          | -           | -             | Filler signal, constant 0 |
| 245          | -           | -             | Filler signal, constant 0 |
| 246          | -           | -             | Filler signal, constant 0 |
| 247          | -           | -             | Filler signal, constant 0 |
| 248          | -           | -             | Filler signal, constant 0 |
| 249          | -           | -             | Filler signal, constant 0 |
| 250          | -           | -             | Filler signal, constant 0 |
| 251          | -           | -             | Filler signal, constant 0 |
| 252          | -           | -             | Filler signal, constant 0 |
| 253          | -           | -             | Filler signal, constant 0 |
| 254          | -           | -             | Filler signal, constant 0 |
| 255          | -           | -             | Filler signal, constant 0 |

**Table 2.2: Mapping from Master qos (MQ) to IPs on GPL release.**

| <b>MQ</b> | <b>Bus of origin</b> | <b>Bus ID</b> | <b>Description</b>     |
|-----------|----------------------|---------------|------------------------|
| 0         | AHB                  | 0             | NOEL-V CORE            |
| 1         | AHB                  | 1             | NOEL-V CORE            |
| 2         | AHB                  | 2             | NOEL-V CORE            |
| 3         | AHB                  | 3             | NOEL-V CORE            |
| 4         | AHB                  | 4             | NOEL-V CORE            |
| 5         | AHB                  | 5             | NOEL-V CORE            |
| 6         | AHB                  | 6             | GR Ethernet MAC        |
| 7         | AHB                  | 7             | GRDMAC2 DMA Controller |
| 8         | AHB                  | 8             | AHB Debug UART         |
| 9         | AHB                  | 9             | JTAG Debug Link        |
| 10        | AHB                  | 10            | EDCL master interface  |
| 11        | -                    | -             | Unused                 |
| 12        | -                    | -             | Unused                 |
| 13        | -                    | -             | Unused                 |
| 14        | -                    | -             | Unused                 |

Table 2.3: Mapping from Master qos (MQ) to IPs on non-GPL release

| MQ | Bus of origin | Bus ID | Description                |
|----|---------------|--------|----------------------------|
| 0  | AHB           | 0      | NOEL-V CORE                |
| 1  | AHB           | 1      | NOEL-V CORE                |
| 2  | AHB           | 2      | NOEL-V CORE                |
| 3  | AHB           | 3      | NOEL-V CORE                |
| 4  | AHB           | 4      | NOEL-V CORE                |
| 5  | AHB           | 5      | NOEL-V CORE                |
| 6  | AHB           | 6      | GR Ethernet MAC            |
| 7  | AHB           | 7      | GRSPW2 SpaceWire           |
| 8  | AHB           | 8      | GRSPW2 SpaceWire           |
| 9  | AHB           | 9      | CAN-FD Controller with DMA |
| 10 | AHB           | 10     | CAN-FD Controller with DMA |
| 11 | AHB           | 11     | GRDMAC2 DMA Controller     |
| 12 | AHB           | 12     | AHB Debug UART             |
| 13 | AHB           | 13     | JTAG Debug Link            |
| 14 | AHB           | 14     | EDCL master interface      |

Signals labeled as *debug* are fix inputs that can be used to test hardware or software with known inputs. **Event 0** (fix '1') can be used to measure the **number of elapsed cycles**.

Signals of type *CCS* can be used to compute the total contention cycle stack of the system. These signals become high at the first rising edge of the clock after a given condition or event has been detected. They remain active until the condition or event that they are measuring becomes low. This behavior allows measuring the length of clock cycles. When generating CCS signals, the user must consider if they want to allow back to back events and generate the input signals accordingly at RTL level.

### 2.3 MAIN CONFIGURATION AND SELF-TEST

Reset and enable of overflow, quota, and regular counters can be performed with register 2.1. All signals are active high.

Self-test mode allows to bypass the input events from the crossbar and instead use a specific input pattern where signals are constant. This mode can be used for debugging. After the addition of the crossbar and debug inputs, there is a certain overlap. The same results can be achieved with the correct crossbar configuration. Nevertheless, it has been included in this release for compatibility.

These are the Self-test modes for each configuration value of the fied *selftest mode* in register 2.1:

- 0b00: Events depend on the crossbar. Self-test is disabled

- 0b01: All signals are set to 1.
- 0b10: All signals are set to 0.
- 0b11: Signal 0 is set to 1. The remaining signals are set to 0.

Register 2.1: BASE CONFIGURATION REGISTER (0x000)

| Selftest mode |    | Reserved |  |  |  |  |  |  |  |  |  | Quota softreset |   |   |   |   | Overflow softreset |  |  |  |  | Overflow enable |  |  |  |  | General Softreset |  |  |  |  | General Enable |  |  |  |             |
|---------------|----|----------|--|--|--|--|--|--|--|--|--|-----------------|---|---|---|---|--------------------|--|--|--|--|-----------------|--|--|--|--|-------------------|--|--|--|--|----------------|--|--|--|-------------|
| 31            | 30 | 28       |  |  |  |  |  |  |  |  |  | 5               | 4 | 3 | 2 | 1 | 0                  |  |  |  |  |                 |  |  |  |  |                   |  |  |  |  |                |  |  |  |             |
| 00            |    |          |  |  |  |  |  |  |  |  |  |                 | 0 | 0 | 0 | 0 | 0                  |  |  |  |  |                 |  |  |  |  |                   |  |  |  |  |                |  |  |  | Reset value |

## 2.4 CROSSBAR

This feature allows routing any of the input signals of table 2.1 or 2.2 into any of the 24 counters of the SafeSU. Each one of the counters has a 5-bit configuration value for the 32 input version of the crossbar or 7-bit for the 128 version. For simplicity this section describes the 32 entry variant of the crossbar. Consider that the width of the configuration field is  $\log_2(N_{inputs})$  and the total amount of configuration registers  $\log_2(N_{inputs}) * N_{inputs}$ .

Configuration values are stored in registers 2.2, 2.3, 2.4, 2.5. All the configuration values are consecutive. Thus some values may have configuration bits in two consecutive memory addresses. Examples of this are Output 6, 12, 19 in our current configuration. As a consequence, the previous outputs may require two writes to configure the desired input signal.

Configuration fields match one to one with the internal counters. So the field *output 0* matches with *counter 0*, *output 1* with *counter 1* and so on.

As a usage example, suppose the user wants to route the signal *pmu\_events(0).icnt(0)* to the internal *counter 0*. The field *Output 0* of register 2.2 shall match the Index of the signal in table 2.1. In this case, the index is 2. After this configuration, the event count will be recorded in *counter 0*. The addresses for counter values are indicated in figure 2.1.

Register 2.2: CROSSBAR CONFIGURATION REGISTER 0 (0x0AC)

| Output 6 [1:0] |    | Output 5 |    | Output 4 |    | Output 3 |    | Output 2 |    | Output 1 |   | Output 0 |   |
|----------------|----|----------|----|----------|----|----------|----|----------|----|----------|---|----------|---|
| 31             | 30 | 29       | 25 | 24       | 20 | 19       | 15 | 14       | 10 | 9        | 5 | 4        | 0 |
| 00             |    | 00       |    | 00       |    | 00       |    | 00       |    | 00       |   | 00       |   |

Reset value

Register 2.3: CROSSBAR CONFIGURATION REGISTER 1 (0x0B0)

|                |    |           |    |           |    |          |    |          |   |          |   |                |   |
|----------------|----|-----------|----|-----------|----|----------|----|----------|---|----------|---|----------------|---|
| Output 12[3:0] |    | Output 11 |    | Output 10 |    | Output 9 |    | Output 8 |   | Output 7 |   | Output 6 [4:2] |   |
| 31             | 28 | 27        | 23 | 22        | 18 | 17       | 13 | 12       | 8 | 7        | 3 | 2              | 0 |
| 00             |    | 00        |    | 00        |    | 00       |    | 00       |   | 00       |   | 00             |   |

Reset value

Register 2.4: CROSSBAR CONFIGURATION REGISTER 2 (0x0B4)

|                 |    |           |    |           |    |           |    |           |    |           |    |           |    |                |    |
|-----------------|----|-----------|----|-----------|----|-----------|----|-----------|----|-----------|----|-----------|----|----------------|----|
| Output 19 [0:0] |    | Output 18 |    | Output 17 |    | Output 16 |    | Output 15 |    | Output 14 |    | Output 13 |    | Output 12[4:4] |    |
| 31              | 30 | 26        | 25 | 21        | 20 | 16        | 15 | 11        | 10 | 6         | 5  | 1         | 0  |                |    |
| 00              | 00 |           | 00 |           | 00 |           | 00 |           | 00 |           | 00 |           | 00 |                | 00 |

Reset value

Register 2.5: CROSSBAR CONFIGURATION REGISTER 3 (0x0B8)

|          |    |           |    |           |    |           |    |           |   |           |   |                |   |
|----------|----|-----------|----|-----------|----|-----------|----|-----------|---|-----------|---|----------------|---|
| Reserved |    | Output 24 |    | Output 23 |    | Output 22 |    | Output 21 |   | Output 20 |   | Output 19[4:1] |   |
| 31       | 29 | 28        | 24 | 23        | 19 | 18        | 14 | 13        | 9 | 8         | 4 | 3              | 0 |
| x        |    | 00        |    | 00        |    | 00        |    | 00        |   | 00        |   | 00             |   |

Reset value

Signal routing is important since some of the SafeSU features are only available at different crossbar outputs. Table 2.4 shows the available capabilities for each one of the outputs.

## 2.5 COUNTERS

The unit in the default configuration contains 24 counters, 32-bit each. Figures 2.1 and 2.2 indicate the memory address where each counter's value can be access. Counter values can be **read** or **written**, thus allowing to set the initial value of the counters.

Enable and reset is managed by the base configuration register 2.1.

Counters can overflow. In such a case, the count will wrap back to 0 and keep counting. Section 2.6 describes how to enable the overflow detection interrupts.

<sup>0</sup>RDC and MCCU signals are only available if the assigned core is synthesized.

Table 2.4: Crossbar outputs and SafeSU capabilities

| <b>Output</b> | <b>Counters</b> | <b>Overflow</b> | <b>MCCU</b> | <b>RDC</b> |
|---------------|-----------------|-----------------|-------------|------------|
| 0             | Yes             | Yes             | Core 0      | Yes        |
| 1             | Yes             | Yes             | Core 0      | Yes        |
| 2             | Yes             | Yes             | Core 1      | Yes        |
| 3             | Yes             | Yes             | Core 1      | Yes        |
| 4             | Yes             | Yes             | Core 2      | Yes        |
| 5             | Yes             | Yes             | Core 2      | Yes        |
| 6             | Yes             | Yes             | Core 3      | Yes        |
| 7             | Yes             | Yes             | Core 3      | Yes        |
| 8             | Yes             | Yes             | Core 4      | Yes        |
| 9             | Yes             | Yes             | Core 4      | Yes        |
| 10            | Yes             | Yes             | Core 5      | Yes        |
| 11            | Yes             | Yes             | Core 5      | Yes        |
| 12            | Yes             | Yes             | No          | No         |
| 13            | Yes             | Yes             | No          | No         |
| 14            | Yes             | Yes             | No          | No         |
| 15            | Yes             | Yes             | No          | No         |
| 16            | Yes             | Yes             | No          | No         |
| 17            | Yes             | Yes             | No          | No         |
| 18            | Yes             | Yes             | No          | No         |
| 19            | Yes             | Yes             | No          | No         |
| 20            | Yes             | Yes             | No          | No         |
| 21            | Yes             | Yes             | No          | No         |
| 22            | Yes             | Yes             | No          | No         |
| 23            | Yes             | Yes             | No          | No         |

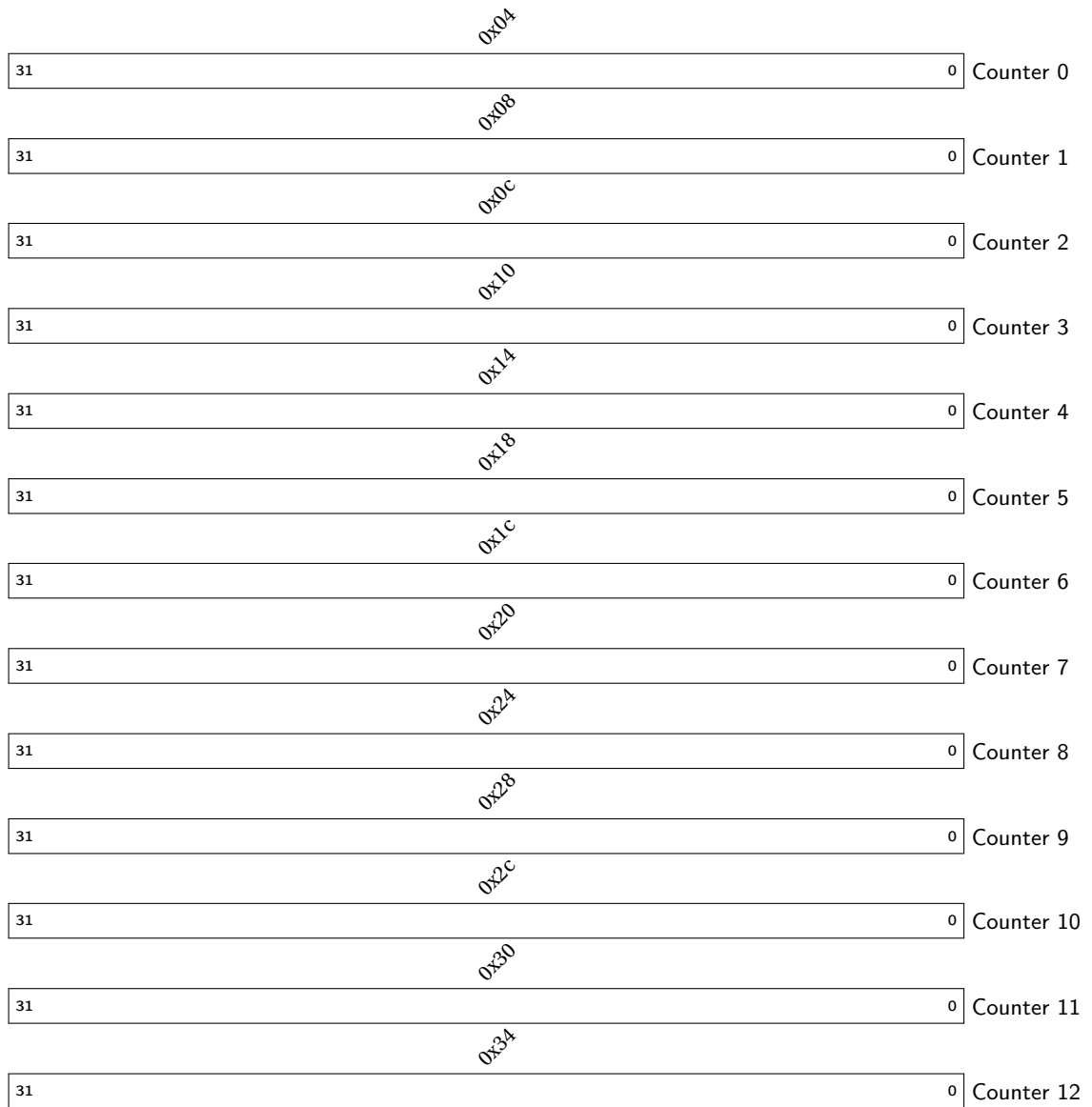


Figure 2.1: 0 to 12 Counter addresses





Figure 2.2: 12 to 24 Counter addresses

## 2.6 OVERFLOW

The user can enable overflow detection for each one of the counters in section 2.5. Enables are active high and individual for each counter, as indicated in register 2.6. If a counter with overflow detection active wraps over the maximum value, the corresponding bit of register 2.7 will become 1, and AHB interrupt number 6 will become active.

The default AHB interrupt mapping can be modified within the file *ahb\_wrapper.vhd*.

Register 2.6: OVERFLOW INTERRUPT ENABLE MASK (0x064)

| Reserved |  | Counter 23<br>Counter 22<br>Counter 21<br>Counter 20<br>Counter 19<br>Counter 18<br>Counter 17<br>Counter 16<br>Counter 15<br>Counter 14<br>Counter 13<br>Counter 12<br>Counter 11<br>Counter 10<br>Counter 9<br>Counter 8<br>Counter 7<br>Counter 6<br>Counter 5<br>Counter 4<br>Counter 3<br>Counter 2<br>Counter 1<br>Counter 0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |  |  |
|----------|--|--|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|--|--|
|          |  | 31   | 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 |  |  |
| x        |  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |

Reset value

Register 2.7: OVERFLOW INTERRUPT VECTOR (0x068)

| Reserved |  |  |  |  |  |  |  | Counter 23 | Counter 22 | Counter 21 | Counter 20 | Counter 19 | Counter 18 | Counter 17 | Counter 16 | Counter 15 | Counter 14 | Counter 13 | Counter 12 | Counter 11 | Counter 10 | Counter 9 | Counter 8 | Counter 7 | Counter 6 | Counter 5 | Counter 4 | Counter 3 | Counter 2 | Counter 1 | Counter 0 |   |
|----------|--|--|--|--|--|--|--|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---|
| 31       |  |  |  |  |  |  |  | 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 |
| 0        |  |  |  |  |  |  |  | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0          | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         |   |

Reset value

## 2.7 QUOTA

This feature has been replaced by the MCCU and will disappear in future releases. Usage is not recommended.

## 2.8 MCCU

The Maximum-Contention Control Unit (MCCU) allows to monitor a subset of the input events and track the approximate contention that they will cause. Currently, events assigned to counters 0 to 7 for the 4 core version or 0 to 11 for the 6 core version can be used as inputs of the MCCU. Thanks to the crossbar, any of the SoC signals can be used by the MCCU. In this section we use the 4 core version as a reference. In all configurations one quota register is assigned to each core and one weight is assigned to each signal.



Register 2.9: MCCU MAIN CONFIGURATION FOR 6 CORE CONFIGURATIONS (0x074)

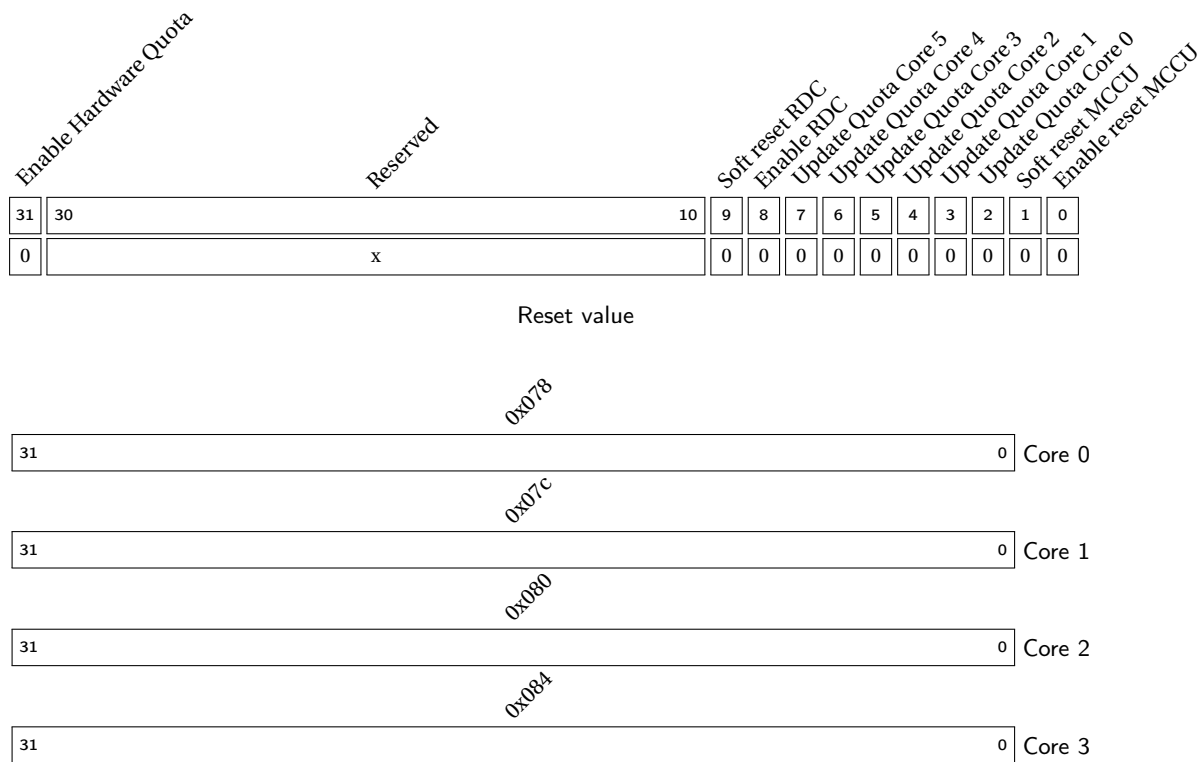


Figure 2.4: MCCU Quota limits for each core

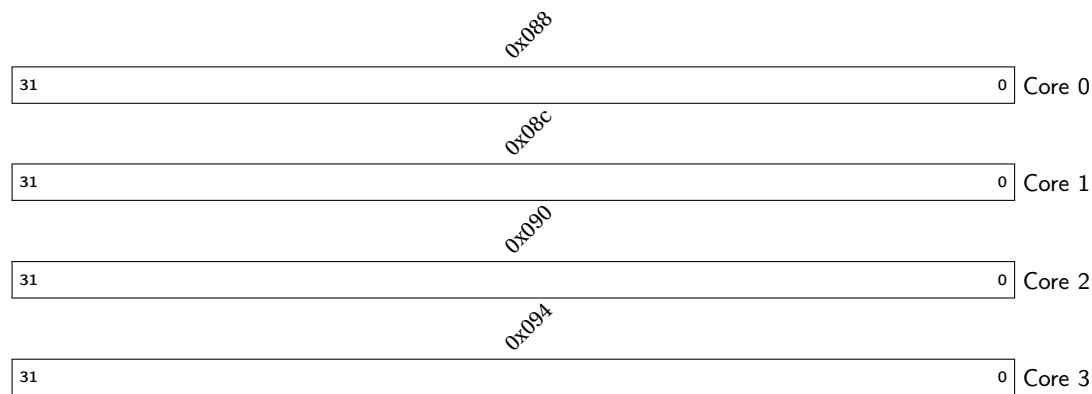


Figure 2.5: MCCU Current remaining Quota for each core

In the current release, the MCCU can be reset and activated with the respective fields of register 2.8. The fields labeled as *Update Quota core* are used to update the available quota of

each core (figure 2.5). While *Update Quota core* is high, the content is assigned to the available quota. Once released (low), the available quota can start to decrease if the MCCU is active. The current quota can be read while the unit is active.

In the current release, each core can monitor two input events. The MCCU module is parametric and more events can be provided in future releases. Table 2.4 shows the features available for each crossbar output. Under the column MCCU, you can see towards which core quota the event will be computed. The unit provides one interruption for each of the monitored cores. Quota exhaustion for cores 3, 2, 1, and 0 is mapped to AHB interrupts 10, 9, 8, and 7, respectively.

Weights for each monitored event are registered in registers 2.10 and 2.12. Currently, each weight is an 8-bit field. Each input of the MCCU maps directly to the outputs of the crossbar. Thus the weight for the MCCU input 0 corresponds to the signal in crossbar output 0.

Register 2.10: MCCU EVENT WEIGHTS REGISTER 0 (SHARED WITH RDC) (0x098)

| Input 3 |    |    |    | Input 2 |   |   |   | Input 1 |  |  |  | Input 0 |  |  |  |
|---------|----|----|----|---------|---|---|---|---------|--|--|--|---------|--|--|--|
| 31      | 24 | 23 | 16 | 15      | 8 | 7 | 0 |         |  |  |  |         |  |  |  |
| 00      |    |    |    | 00      |   |   |   | 00      |  |  |  | 00      |  |  |  |

Reset value

Register 2.11: MCCU EVENT WEIGHTS REGISTER 1 (SHARED WITH RDC) (0x09c)

| Input 7 |    |    |    | Input 6 |   |   |   | Input 5 |  |  |  | Input 4 |  |  |  |
|---------|----|----|----|---------|---|---|---|---------|--|--|--|---------|--|--|--|
| 31      | 24 | 23 | 16 | 15      | 8 | 7 | 0 |         |  |  |  |         |  |  |  |
| 00      |    |    |    | 00      |   |   |   | 00      |  |  |  | 00      |  |  |  |

Reset value

Register 2.12: MCCU EVENT WEIGHTS REGISTER 2 (SHARED WITH RDC) (0xa0)

| Input 11 |    |    |    | Input 10 |   |   |   | Input 9 |  |  |  | Input 8 |  |  |  |
|----------|----|----|----|----------|---|---|---|---------|--|--|--|---------|--|--|--|
| 31       | 24 | 23 | 16 | 15       | 8 | 7 | 0 |         |  |  |  |         |  |  |  |
| 00       |    |    |    | 00       |   |   |   | 00      |  |  |  | 00      |  |  |  |

Reset value

## 2.9 RDC

The Request Duration Counter or RDC is comprised of a set of 8-bit counters and comparators that allow monitoring the length of a CCS signal, record the number of clock cycles of the

longest pulse and compare such value with the defined weight.

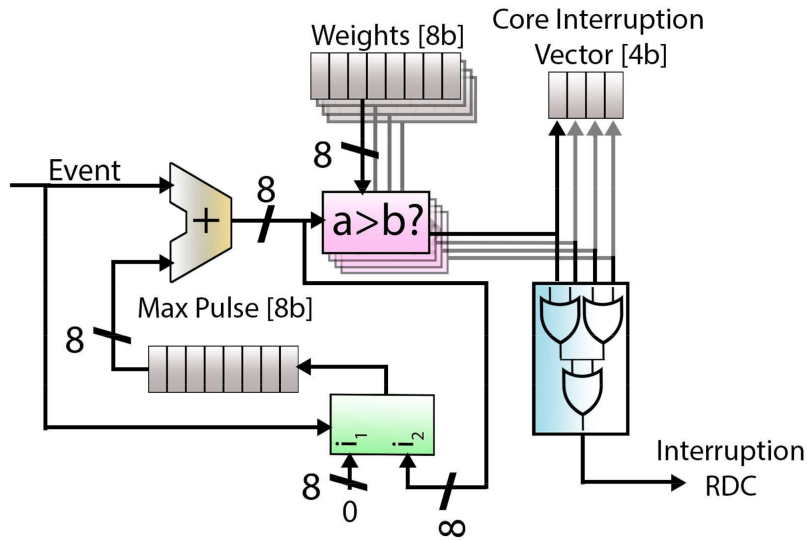


Figure 2.6: Block diagram RDC mechanism.

The current release provides monitoring for crossbar outputs 0 to 7 for the quad-core configuration and 0 to 11 for the hexa-core. Note that the amount of registers changes with the amount of cores, and figures and addresses are representative of the 4 core configuration. The weights for each signal are shared with the MCCU and are stored in registers 2.14 and 2.15. Weights are 8-bit fields. Counters have overflow protection, preventing the count from wrapping over the maximum value. The maximum value for each event (watermarks), are stored in registers 2.16 and 2.17.

The RDC shares the main configuration register with the MCCU (register 2.8). Through this register, the unit can be reset and enabled through the corresponding fields. Such fields are active high signals.

The unit does provide access to the internal interrupt vector (register 2.13), but such information is redundant and may be removed in future releases. Given the current watermarks and assigned weights, the events responsible for the interrupt can be identified. The RDC interrupt has been routed to AHB interrupt 11.

Register 2.13: RDC INTERRUPT VECTOR (0x0a0)

|          |  |  |  |  |  |  |  |  |  |        |  |        |  |        |  |        |  |        |  |        |  |   |  |
|----------|--|--|--|--|--|--|--|--|--|--------|--|--------|--|--------|--|--------|--|--------|--|--------|--|---|--|
| Reserved |  |  |  |  |  |  |  |  |  | Core 5 |  | Core 4 |  | Core 3 |  | Core 2 |  | Core 1 |  | Core 0 |  |   |  |
| 31       |  |  |  |  |  |  |  |  |  | 4      |  | 3      |  | 3      |  | 3      |  | 2      |  | 1      |  | 0 |  |
| x        |  |  |  |  |  |  |  |  |  |        |  | 00     |  | 00     |  | 00     |  | 00     |  | 00     |  |   |  |

Reset value

Register 2.14: RDC EVENT WEIGHTS REGISTER 0 (SHARED WITH MCCU) (0x098)

| Input 3     |  |    |  | Input 2 |  |    |  | Input 1 |  |   |  | Input 0 |  |   |  |
|-------------|--|----|--|---------|--|----|--|---------|--|---|--|---------|--|---|--|
| 31          |  | 24 |  | 23      |  | 16 |  | 15      |  | 8 |  | 7       |  | 0 |  |
| 00          |  |    |  | 00      |  |    |  | 00      |  |   |  | 00      |  |   |  |
| Reset value |  |    |  |         |  |    |  |         |  |   |  |         |  |   |  |

Register 2.15: RDC EVENT WEIGHTS REGISTER 1 (SHARED WITH MCCU) (0x09c)

|         |    |  |  |         |    |  |  |         |   |  |  |         |   |  |  |             |
|---------|----|--|--|---------|----|--|--|---------|---|--|--|---------|---|--|--|-------------|
| Input 7 |    |  |  | Input 6 |    |  |  | Input 5 |   |  |  | Input 4 |   |  |  |             |
| 31      | 24 |  |  | 23      | 16 |  |  | 15      | 8 |  |  | 7       | 0 |  |  |             |
| 00      |    |  |  | 00      |    |  |  | 00      |   |  |  | 00      |   |  |  | Reset value |

Register 2.16: RDC WATERMARK REGISTER 0 (0x0a4)

| Input 3     |  |    |  | Input 2 |  |    |  | Input 1 |  |   |  | Input 0 |  |   |  |
|-------------|--|----|--|---------|--|----|--|---------|--|---|--|---------|--|---|--|
| 31          |  | 24 |  | 23      |  | 16 |  | 15      |  | 8 |  | 7       |  | 0 |  |
| 00          |  |    |  | 00      |  |    |  | 00      |  |   |  | 00      |  |   |  |
| Reset value |  |    |  |         |  |    |  |         |  |   |  |         |  |   |  |

Register 2.17: RDC WATERMARK REGISTER 1 (0x0a8)

|         |    |  |  |         |    |  |  |         |   |  |  |         |   |  |  |             |
|---------|----|--|--|---------|----|--|--|---------|---|--|--|---------|---|--|--|-------------|
| Input 7 |    |  |  | Input 6 |    |  |  | Input 5 |   |  |  | Input 4 |   |  |  |             |
| 31      | 24 |  |  | 23      | 16 |  |  | 15      | 8 |  |  | 7       | 0 |  |  |             |
| 00      |    |  |  | 00      |    |  |  | 00      |   |  |  | 00      |   |  |  | Reset value |

## 2.10 SOFTWARE SUPPORT

The unit can be configured by the user at a low level following the description of previous sections and the documents associated for each unit. In addition we provide a small bare-metal

driver under the *drivers* directory of the SafeSU repository.

Currently 3 configurations are provided. *4-core* and *6-core* folders have the driver for default configurations. *6-core-256e* folder has a driver for an hexacore with 256 entries on the crossbar.

The drivers are composed of three files.

- *pmu\_vars.h* : Defines a set of constants with the RTL parameters that generated the unit. Such constants allow the reusing of functions among hardware configurations.
- *pmu\_hw.h*: Defines the memory position of the SafeSU within the memory map of the SoC. It also contains the prototypes of each function of the driver.
- *pmu\_hw.c*: Contains the definition of each of the functions.

Given the current development status, the driver is not autogenerated, and **modifications to the RTL may require manual changes on the previous files.**

### 3 CONFIGURATION OPTIONS

Table 3.1 shows the configuration parameters exposed by *ahb\_wrapper.vhd*. Given the current development status, changes to the configuration options may require manual modifications to internal modules and software drivers. Future releases will expose parameters to enable individual SafeSU features and allow for more flexibility.

Table 3.1: Configuration options (VHDL ports)

| Generic         | Function   | Allowed range   | Default |
|-----------------|--|-----------------|---------|
| HADDR           | AHB base address                                     | 0 to 16#fff#    | 0       |
| HMASK           | AHB address mask                                     | 0 to 16#fff#    | 16#fff# |
| N_REGS          | Total of accessible registers                        | 2 to 64         | 43      |
| SafeSU_COUNTERS | Number of generic counters. Same as crossbar outputs | 1 to 32         | 24      |
| N_SOC_EV        | SoC signals. Inputs to the crossbar                  | 1 to 64         | 32      |
| REG_WIDTH       | Size of registers and counters                       | 32 <b>or</b> 64 | 32      |

### 4 SIGNAL DESCRIPTIONS

Table 4.1 shows the interface of the core (VHDL ports).



Table 4.1: Signal descriptions (VHDL ports)

| Signal name    | Field | Type   | Function   | Active |
|----------------|-------|--------|--|--------|
| RST            |       | Input  | Reset  | Low    |
| CLK            |       | Input  | AHB master bus clock   | -      |
| SafeSU_EVENTS  |       | Input  | Input for regular SoC events   | -      |
| CCS_CONTENTION |       | Input  | Input for contention cycle stack signals that measure contention     | -      |
| CCS_LATENCY    |       | Input  | Input for contention cycle stack signals that measure access latency | -      |
| AHBSI          | *     | Input  | AHB slave input signals  | -      |
| AHBSO          | *     | Output | AHB slave output signals, includes interrupts                        | -      |

## 5 LIBRARY DEPENDENCES

Table 5.1 shows the libraries used when instantiating the core (VHDL libraries).

Table 5.1: Library dependencies

| Library | Package        | Imported units        | Description                                     |
|---------|----------------|-----------------------|---|
| IEEE    | std_logic_1164 | Types                 | Standard logic types                            |
| GRLIB   | amba           | Signals               | AMBA signal definitions                         |
| GRLIB   | config         | Types                 | Amba P&P types                                  |
| GRLIB   | devices        | Types                 | Device names and vendors                        |
| GRLIB   | stdlib         | All                   | Common VHDL functions                           |
| GAISLER | noelv          | Signals               | Counter vectors and types                       |
| BSC     | pmu_module     | Instances and signals | Instances and signal definitions for the SafeSU |

## 6 INSTANTIATION

An example design is provided in the context of SELENE and De-RISC project. Integration examples of earlier releases of the unit along LEON3MP can be provided under demand.

Listing 1: SafeSU instance example for gpp\_sys

```
-- Include BSC library
library bsc;
use bsc.pmu_module.all;

--Provide a non-overlapping hsidx
constant hsidx_pmu : integer := 6;

--Update the hsidx of the next slave
```

```

constant nextslv      : integer := hsidx_pmu + 1;

--Declare events and signals of interest. They may change
--for each use case. Route such events to pmu_events,
--ccs_contention and ccs_latency ports

--Instance of the unit

PMU_inst : ahb_wrapper
generic map(
ncpu    => CFG_NCPU,
hindex  => hsidx_pmu,
haddr   => 16#801#,
hmask   => 16#FFF#
)
port map(
rst      => rstn,
clk      => clk,
pmu_events    => pmu_events,
ccs_contention    => ccs_contention,
ccs_latency    => ccs_latency,
ahbsi    => ahbsi,
ahbso    => ahbso(hsidx_pmu));

```

Given the development status of the module and drivers it is recommended to not modify the internal parameters of the module.