The SCA Controller for the ATLAS LAr Calorimeter

Stephan Böttcher, John Parsons, William Sippach
Nevis Labs, Columbia University

Douglas Gingrich
University of Alberta

DRAFT 1.10, 2001/10/19

Abstract

This document describes the *deep sub-micron* (DSM) implementation of the SCA Controller for the ATLAS LAr Calorimeter front-end readout.

Contents

1 Introduction **5**

2 Controller Architecture **6**
2.1 SCA Address FIFOs 6
2.2 Readout Control 9
2.3 Configuration Interface 9
2.4 TTC Interface 9
2.4.1 L1Accept 10
2.4.2 BCID and Event Number Reset 10
2.5 IO 10

3 Operational Features **11**
3.1 Sampling Pattern 11
3.2 Special Modes of Operation 11
3.3 Level 1 Trigger Latency 11
3.4 Derandomising Buffer Capacity 12
3.5 Overlapping Events 13
3.6 Pileup Treatment 13
3.7 Invalid SCA Cell Numbers 14

4 Operation **15**
4.1 Option Pins 15
4.2 Configuration Parameters 16
4.2.1 Serial Configuration Address Space 16
4.3 Initialisation 17
4.4 Status Monitoring 17
4.4.1 Output Data Status 18
4.4.2 Status Register 18

5 SEU Errors **18**
5.1 SEU Error Detection 19
5.2 SEU Protection Measures 19
5.2.1 Configuration Storage 19
5.2.2 State Machines 20
5.2.3 SCA Address FIFOs 20
5.3 Sensitive Sequential Nodes Count 22

6 SRAM test interface **23**

7 Design Flow **24**
7.1 DSM Standard Cell Library 24
7.2 SRAM 25
7.3 Verilog Design 25
7.3.1 Parameterised FIFO module 26
7.3.2 SRAM modules 27
7.3.3 Explicitly Instantiated Library Cells 27
7.4 Clock Distribution 27
7.5 RTL Level Simulation 28
7.6 Synthesis 29
7.7 Gate level simulation 29
7.8 Place and Route 29
7.9 Timing Extraction and Simulation 29
7.10 Physical Verification 29
7.11 Design Status 30
7.12 Die Size 30

A SCAC Configuration **31**
A.1 Configuration Address Space 31
A.1.1 Chip select 31
A.1.2 Configuration and Status 31
A.1.3 TTC commands 32
A.2 Configuration Data Formats 32
A.2.1 Mode Register Format 32
A.2.2 Status Register Format 34
A.2.3 Event Trailer Status 36

B IO pins **39**

C Misfeatures of MPW4 chips **44**
C.1 The `free_fifo`bug. 44
C.1.1 Impact 44
C.1.2 Workarounds 44
C.1.3 Description 45
C.1.4 Fix 45
C.2 The fixed gain mode bug. 45

C.2.1 Impact 45
C.2.2 Workarounds 45
C.2.3 Description 45
C.2.4 Fix 46
C.3 The `BCntResEvCntRes`pin swap 46
C.3.1 Impact 46
C.3.2 Workarounds 46
C.3.3 Description 47
C.3.4 Fix 47
C.4 The clock buffer strength mismatch 47
C.4.1 Impact 47
C.4.2 Workarounds 47
C.4.3 Description 47
C.4.4 Fix 47

List of Figures

1 SCA Controller Block diagram. 7
2 SRAM test timing diagram. 23
3 Configuration and status register format. 33
4 SCA Controller Bonding diagram and pinout. 38

1 Introduction

As described in the TDR [1] of the liquid argon (LAr) calorimeter, the LAr front end boards (FEBs) [2] contain the electronics for amplifying, shaping, sampling, pipelining, and digitising the liquid argon (LAr) calorimeter signals.

On the FEB, the calorimeter signals are received, amplified, split into three gain scales, and shaped with a bipolar shaping function. The shaped signals are then sampled at the bunch crossing frequency of 40 MHz, with the samples stored in analog form in switched-capacitor array (SCA) analog pipelines [3] during the Level 1 trigger latency. Upon receipt of a Level 1 trigger accept, a train of typically five samples is read from the SCA and digitised by a 12-bit ADC. A gain selection algorithm is applied that ensure that an appropriate gain scale is chosen for each channel in each event. The digitised samples are formatted and then transmitted off-detector to the *Read Out Driver* (ROD) modules [4] for processing.

The SCAs on the FEB provide the L1 trigger latency pipeline of up to 2.5 μs, as well as a derandomising buffer. The SCA allows independent, simultaneous, directly addressed read and write operations in order to provide deadtimeless performance at Level 1 trigger rates up to 75 kHz. The SCA provides 144 analog storage cells for each signal channel.

Each calorimeter signal is sampled on three analog gain scales. The readout of the SCA is performed by the Gain Selector chip [5], which can operate in fixed gain modes, or auto-gain mode. In a fixed gain mode, one, two, or three of the gain scales are read out for each signal sample. In auto-gain mode, the *peak sample* of the signal is digitised at a low gain scale, and compared to thresholds to determine the gain scale to be used for each signal. The chosen gain scale of the peak sample is then read. The remaining signal samples of the event are only read and digitised at the chosen gain scale.

The SCA Controller (SCAC) provides the main digital control of the FEB. The SCAC receives the trigger information from the on-board TTCrx chip [6], and generates the write and read addresses for the SCA pipelines. In addition, the SCAC controls the flow of the SCA readout through its communication with the Gain Selector chips.

On the Module 0 FEB, the SCAC functionality was implemented in a Xilinx FPGA. However, the FEB placement in crates on the detector requires tolerance to ??? krad of ionising radiation and to ???$\times 10^{13}$ n/cm^2, over 10 years of LHC operation. In addition to total dose requirements, the

SCAC must be sufficiently tolerant to SEU-induced error rates in its internal logic to allow the readout to operate stably. A DMILL design of the SCAC Controller was built and tested [7]. However, the design is rather challenging in DMILL and the large die size of the DMILL design imply corresponding yield and cost concerns. A second implementation using a *Deep Sub-Micron* CMOS technology was built, tested, and finally chosen for the ATLAS readout. This DSM implementation is the subject of this document.

References

[1] ATLAS Liquid Argon TDR: `http://atlas.web.cern.ch/Atlas/GROUPS/LIQARGON/TDR/`

[2] ATLAS LAr FEB Documentation: `http://www.nevis.columbia.edu/~atlas/electronics/ATLASFEB/ATLASFEB.html`

[3] `!javascriptisnotenabled:-(`

[4] `!javascriptisnotenabled:-(`

[5] The ATLAS LAr Gain Selector ASIC: `http://www.nevis.columbia.edu/~atlas/electronics/asics/ggains/gainsel.ps`

[6] TTCrx ASIC: `http://ttc.web.cern.ch/TTC/TTCmain.html#TTCrx`

[7] DMILL SCA Controller: `ftp://jever.phys.ualberta.ca/pub/atlas/papers/scac-dmill.pdf`

2 Controller Architecture

A block diagram illustrating the basic architecture of the SCAC is shown in figure 1. The main functional blocks are described in more detail in the following subsections.

2.1 SCA Address FIFOs

The main SCAC functionality, namely the SCA write and read address bookkeeping, involves cycling the 144 possible cell addresses of the SCA through

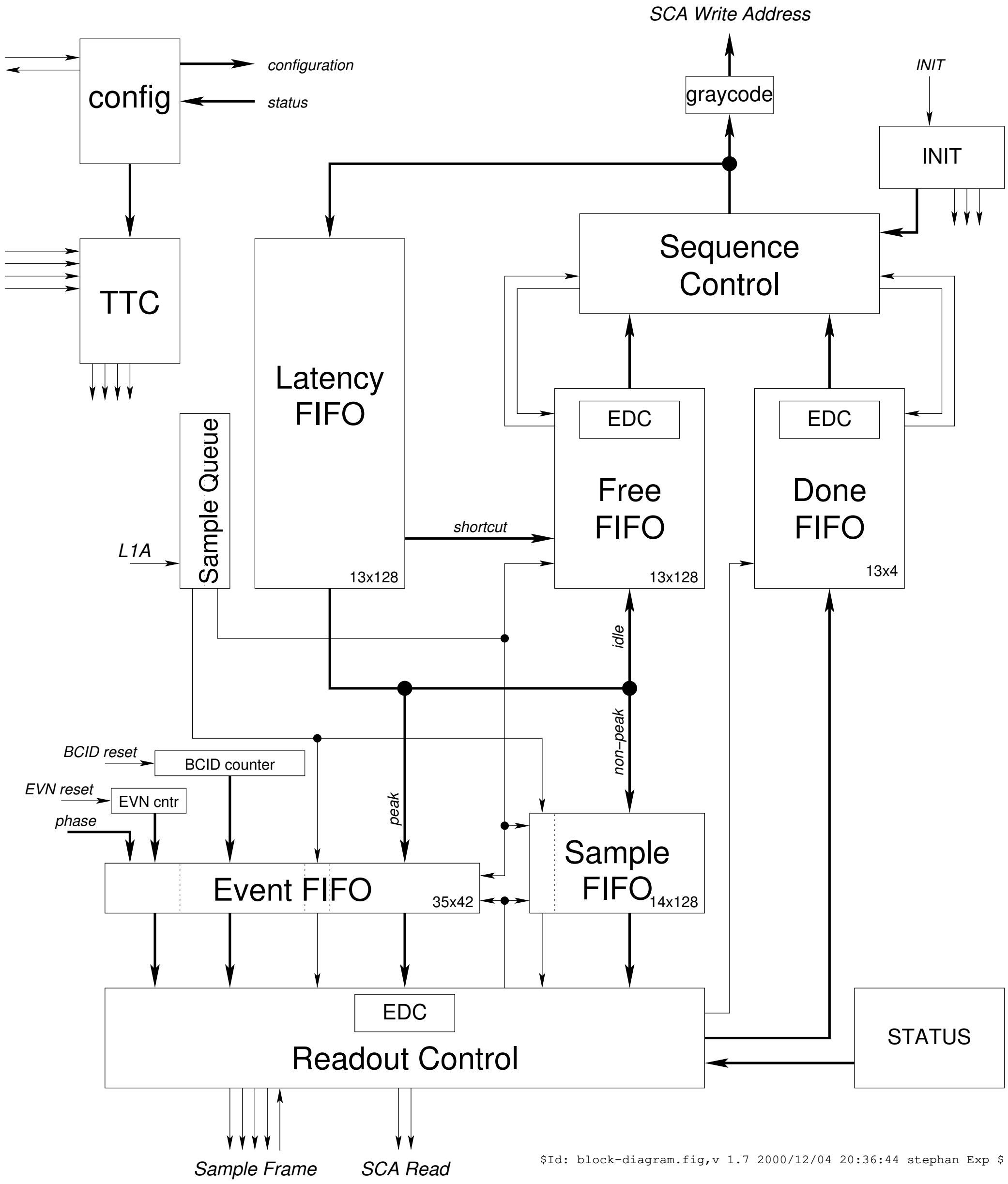

Figure 1: SCA Controller Block diagram.

a system of FIFOs. The SCA write operation runs at the LHC bunch frequency of 40.08 MHz. The SCA readout is performed at 1/8 of the bunch frequency, 5 MHz.

The SCA addresses are gray encoded, to reduce pattern noise on the FEB and inside the SCA. To achieve this noise reduction, the cell addresses must be maintained in sequential order as best as possible. While the 40 MHz write address is sent 8-bit parallel, the read address is sent bit-serially in order to reduce the number of address bus lines.

The 144 SCA cells are represented by 8-bit addresses which are moved in the SCAC through a series of FIFOs. As illustrated in figure 1, the movement of SCA addresses takes the following path:

1. After a SCA cell number is used for the SCA write address for a particular bunch crossing, it is written into the `latency_fifo`, which stores the sequence of SCA cells corresponding to the bunch crossings of the L1 trigger latency. The cell numbers are stepped through the `latency_fifo` every 25 ns. The depth of the FIFO is programmed such that a given cell number emerges from the FIFO at the time that the L1 trigger decision for the relevant bunch crossing arrives.

2. Cells which emerge from the `latency_fifo` and are not part of an L1 Accept are transferred to the `free_fifo`, which contains addresses which are available to be used as write addresses in upcoming bunch crossings.

3. Those addresses emerging from the `latency_fifo` which are part of a L1 triggered event are transferred to the derandomising buffer of samples awaiting digitisation. The SCAC will typically be configured to read out five consecutive signal samples for each triggered event. The relevant cell addresses are stored in either the `event_fifo` or the `sample_fifo`. The `event_fifo` stores some event status and the cell address of the peak sample of the event. The `sample_fifo` stores the remaining non-peak sample addresses.

4. After the signal samples corresponding to a particular cell address has been fully digitised, the address is again available for re-use. However, in order to keep the gray encoded addresses as sequential as possible, these cells are not immediately put into the `free_fifo`. Instead, they are transferred to the `done_fifo`.

5. Each 25 ns, the Sequence Control compares the addresses at the top of the `free_FIFO` and `done_fifo` and chooses the one which most closely preserves the ongoing sequence of write addresses. In this manner, the addresses in the `done_fifo`, corresponding to samples read from previous L1 triggers, will be inserted back into the correct sequence.

2.2 Readout Control

The readout operation must be synchronised with the Gain Selectors, depending on the gain mode in use. The readout control block handles the interface between the SCAC and the Gain Selectors. The SCAC provides the Gain Selector with the read addresses as well as other information (such as Bunch Counter number) required for the readout data stream.

The readout data stream consists of event data blocks, which in turn are composed of signal sample data blocks. The SCAC is responsible for forming complete event blocks, by requesting sample readout operations in the proper sequence. The Gain Selector chip is responsible for the formation of sample data blocks, including event header and trailers.

The sample command format is described in the documentation of the Gain Selector.

2.3 Configuration Interface

The configuration block provides the interface of the SCAC to the Configuration Controller chip used to download and read back serial data required to configure the SCAC. Thus, the configuration block provides the interface for configuring the SCAC in the requested operational mode. More details are provided in Section 4.2.

2.4 TTC Interface

The TTC block provides the interface of the SCAC to the TTCrx chip located on each FEB. In addition to the 40 MHz clock, the SCAC receives from the TTCrx chip the L1 Accept signal, Reset signals for the Bunch Counter and Event Counter, and a signal for initialisation. The SCAC response to these four TTC commands from the TTCrx can be independently enabled/disabled via configuration bits.

2.4.1 L1Accept

The `L1Accept` is decoded by the SCAC in a three cycle window. A L1A is accepted at the rising edge of the `lhc_clk`, when

- the `L1Accept` input is high,
- and the state of `L1Accept` in the previous cycle matches a configurable state,
- and the `L1Accept` was low in the cycle before that.

In other words, the sequence of L1Accept states that cause a L1A can be configured to be (earlier to later) 0 0 1 or 0 1 1.

In other words, the L1A can be encoded in one or two bits, and triggers must be at least three bunch clocks apart. The second option introduces one additional cycle of trigger latency, but reduces the rate of SEU induced fake triggers.

2.4.2 BCID and Event Number Reset

In order to minimise FEB noise, the Bunch Counter and Event Counter outputs of the TTCrx are not used. Instead, the SCAC includes an internal 12-bit Bunch Counter and 5-bit Event Counter, which are synchronously reset using the relevant Reset signals.

The value of the BCID counter is sent with the first sample command of every event to the Gain Selector, and included in the event header data. The `BCntRes` is synchronised with respect to the L1Accept timing such that the BCID sent with an event agrees with the number that would be produced by the TTCrx chip for that event. In case that the L1A is encoded in two bits, the BCID corresponds to the later L1Accept.

The event counter can optionally be reset by the `BCntRes`, instead of the `EvCntRes`, to count the events in each orbit.

2.5 IO

With few exceptions, all IO to and from the SCA Controller uses LVDS levels, with differential receivers on all input pairs, and current mode LVDS drivers on the output pairs. The exceptions are three option inputs, which are supposed to be hardwired on the PCB, and four TTC inputs. The two clock inputs are received differently as well.

In the case that some differential inputs need to be driven by single ended CMOS ot TTL signals, the reference input should be biased at some suitable level, and the single ended signal applied to the other pin of the pair. The receivers work well even when driven rail to rail, but not beyond the power rails.

3 Operational Features

3.1 Sampling Pattern

The SCAC can be configured to read a number of samples per event, between 1 and 32. These samples can be arbitrarily placed in a pattern of up to 32 bunch clocks in length.

One sample is designated as the *peak sample*, and is read first, to perform the gain selection. The remaining samples are read in sequence.

3.2 Special Modes of Operation

Two special modes of operation are independently configurable:

No Sequencing: The ordering of cell numbers into ascending sequence can be turned off. This is useful for testing how effective the gray code reduces the pattern noise contribution to the coherent noise on the FEB.

Single Event Mode: This mode configures the SCAC to ignore L1Accept as long as an event is being read out. (Actually, triggers are accepted as soon as the last cell number is being sent out to the Gain Selector.)

3.3 Level 1 Trigger Latency

The L1 trigger latency is configurable roughly between 14 and 127 clock cycles. The exact range depends on how exactly you define the trigger latency. The configuration specifies the number of samples stored in the `latency_fifo`, within the given range. The minimum is set by the available space in the `free_fifo`.

The SCAC contributes to the L1 trigger latency, in the sense that internal delays may reduce the number of SCA cells available for derandomising buffer storage.

The critical path is the time from arrival of a L1Accept signal, until the earliest time when the first cell address that belongs to the corresponding event could be sent to the SCA write address bus, in the case when the L1 trigger did not occur.

This path has two components

1. How fast can the SCAC decide to set a cell number aside for readout after the L1Accept arrived.
2. How fast can a cell number that is not selected for readout traverse to the write address output.

The DSM technology would allow implementation of the SCAC with zero additional latency. In such an implementation, the L1Accept input would, in some circumstances, directly control which cell number appears on the write address output on the next clock edge. However, such an SCAC design would require a large setup time on the L1Accept input, and thus restrict the flexibility of the clock distribution architecture on the FEB. Further, the L1Accept timing path inside the chip would be very demanding, when worst case timing models are considered. To relax the timing constraints both inside and outside the chip, a register has been added to the L1Accept path at the input. This register contributes one clock cycle to the L1 trigger latency.

3.4 Derandomising Buffer Capacity

The SCA analog storage fulfills two purposes: L1 trigger latency pipeline, and derandomising buffer. The number of events which can be stored in the derandomising buffer depends on the L1 trigger latency and on the number of samples to digitise per event. The 144-cell depth of the SCA was chosen to provide at least an 8-event buffer, assuming a maximum L1 latency of 2.5 μs (100 crossings) and 5 samples per event, yielding a deadtime of order 1% for a 75 kHz maximum L1 trigger rate.

The FEB system does not send a BUSY to the trigger system. Instead, the status of the SCAC must be modeled by the L1CTP to assert a BUSY within a few clock cycles of a trigger, when the busy condition is reached.

It is foreseen to implement a *leaky bucket* algorithm in the L1CTP to avoid issuing additional triggers when the SCA buffers are full.

3.5 Overlapping Events

Overlapping events occur when two L1 triggers happen in short succession, so that the first sample of the second event is not later than the last sample of the previous event. The L1CTP is supposed to prevent overlapping events for normal data taking. Triggers must be separated by at least the number clock cycles as there are samples per event (nominally 5).

Nevertheless, the SCAC must behave well in the presents of overlap, for two reasons:

1. Not to preclude running modes that require overlapping events.
2. To cope with SEU-induced fake triggers.

Due to limitations of the SCA, in case of overlap it is infeasible to produce two complete well formed events. It was decided that when overlapping events are triggered intentionally, the earlier samples should be deemed more important that later ones. This leads to a simple, yet useful treatment of overlapping events as follows:

The remaining samples of the first event are replaced with invalid cell numbers to complete the event data format. The second event will be processed as usual. This implies the lost of even those samples of the first, which are *not* claimed by the second event due to gaps in the sample pattern.

This also means that the Gain Selector will receive invalid cell numbers from the SCAC. The implications of reading from invalid cell numbers on the readout amplifier inside the SCA are not fully known (see section 3.7).

The peak sample will always be assigned to the proper event. When the overlapping events are so close that the peak sample of the first event overlaps with the second event, that sample will be assigned to the first event, and the second event will have one invalid cell number inserted in place of the first non-peak sample. That may distort the proper order of samples in the second event.

3.6 Pileup Treatment

The Gain Selector treats events in close succession differently from isolated events, to allow for proper digitisation of signals during the drift time of energy depositions of an earlier event. This condition is marked by the *backporch* bit, which is sent by the SCAC to the Gain Selector as part of the sample command for the peak sample.

The backporch bit is set for an event which follows the previous trigger by less than a configurable number of bunch crossings. This number can be set, via the SCAC configuration data, up to 31 bunch clock cycles from the first sample of the event. (A drift time of 400 ns is equivalent to 16 cycles, so a limit of 31 is more than adequate.)

3.7 Invalid SCA Cell Numbers

The SCA has 144 analog storage cells for each signal channel. These cells are addressed by 8 gray encoded bits. The 8 bit address space leaves 112 invalid cell numbers.

Implications of sending invalid cell numbers to the SCA were not yet studied in great detail. There is little worry that attempts to write to invalid cell addresses can cause any problems. Problems are not so easy to dismiss when attempts are made to read non-existing cells. The internal readout amplifiers of the SCA would operate without feedback path, and thus drive their output to a power rail. The open question is if the amplifiers would sufficiently recover during the following reset period (800 ns) in time for the next read operation, and how much noise all this causes on the chip.

The SCAC generates invalid cell number under the following conditions:

1. To fill overlapping events. (read)

2. When no free cell numbers are available. (write)

3. When an event claims a sample that was not available for writing (case 2). (read)

4. When the configuration is inconsistent, or the SCAC gets into some inconsistent state, so that not enough samples are set aside for readout. (read)

The SCAC generates an `sca_read` command signal, that will be used instead of the one supplied by a Gain Selector to drive the read command bus to the SCAs. The `sca_read` command from the SCAC is suppressed when the cell number is invalid for any reason. Using this signal will eliminate the problems with reading non-existing cells.

4 Operation

The configuration and operation of the SCAC is determined by four sources of control:

- Option pins on the chip, for hardwired configuration options on the FEB printed circuit board (PCB).
- Configuration parameters downloaded through the SPAC interface.
- Commands from the TTC system via TTCrx.
- Timing feedback from the Gain Selector (optional).

4.1 Option Pins

The DSM implementation of the SCAC has three options that are selected via static connections on the FEB PCB. These are:

- Chip address on the serial configuration bus.
- Order of the SCA write address bits.
- Select internal or external read timing control.

One FEB requires two SCAC chips. Each chip has its dedicated serial port to the Configuration Controller. However, to allow both SCAC chips to reside on a single configuration bus, one bit of the configuration address (bit 6) is configurable.

The two SCAC on the FEB drive mirror images of the SCA bus. To allow routing of these busses without extensive crossing of signal traces on the PCB, the bit order of the write address can be reversed. The polarity of the signal is not reversed, an inverting output does not turn into a non-inverting output.

The Gain Selectors return a timing signal `s_ready` to the SCAC, which depends on the time required to read a sample, and thus on the gain mode (auto, 1, 2, or 3 fixed gains). The DSM implementation has four differential `s_ready` inputs pairs, one for each Gain Selector, and takes a majority decision. On the other hand, this control loop can be eliminated entirely, by configuring the SCAC with the proper gain mode and having it generate the timing on its own. This mode of operation can be selected by an option pin.

4.2 Configuration Parameters

The configuration of the SCAC is possible through the serial SPAC configuration bus. Each FEB is equipped with a SPAC Slave which interfaces to the SPAC bus. The Configuration Controller chip interfaces the SPAC signals to the various programmable devices on the FEB, including the SCAC. The SCAC requires the download of two 32-bit words for configuration. Each bit of configuration information is stored with triple redundancy, with majority voting used to resolve SEU-induced errors.

One register of 32 bits defines the sample pattern. One bit is set for each sample to be read per event. Bit 0 of the sample pattern (first sample) should be set, to ensure proper operation with overlapping triggers. The selected peak sample must also be set in the sample pattern.

The second register of 32 bits defines various operational parameters, such as the number of samples to digitise per event and the position of the peak sample. The definitions of each of the bits are given in Appendix A.2.

A third register of 16 bits allows to introduce a delay between the readout of events. This register, like the other two registers, is reset to zero at power-up, so that no action is required for nominal running. This register specifies the number of read-clock cycles to wait after each event, before the next event is processed. This feature may be useful to reduce the readout bandwidth, to cope with limited early implementations of the ROD system. Another projected use of this feature is the measurement of the leakage current in SCA capacitor cells.

4.2.1 Serial Configuration Address Space

The SCAC occupies a subset of the 8 bit configuration address space that is not recognised by the Gain Selectors. This arrangement would allow to connect all eight Gain Selectors and both SCACs on a single configuration bus. However, the LVDS output cell of the DSM library does not provide a control to turn off the outputs, and therefore each SCAC will require its own configuration bus. This restriction is not serious, since there is a sufficient number of serial configuration busses available from the Configuration Controller. The addresses recognised by the SCAC, and the corresponding bit definitions, are summarised in Appendix A.1.

4.3 Initialisation

The SCAC receives initialisation commands from the TTCrx via one of its broadcast message bits. The TTC chain is used to transmit the init command, instead of the SPAC serial bus, so that the entire system of FEBs can, if desired, be synchronously initialised.

TTC broadcast commands are supposed to be validated by the BrcstStr output of the TTCrx chip. However, in order to minimise noise on the FEB, the SCAC does not use this strobe. Instead, the SCAC uses other available means to figure out when a broadcast happened, which are

- a BCntRes command is received, or
- a EvCntRes command is received, or
- a transition on the Init Brcst$n$ signal is detected.

When two successive broadcast messages include the Init command, but the second one does not include either BCntRes or EvCntRes, then the second Init command will *not* be detected. Init commands accompanied by EvCntRes are always detected.

An initialisation command resets the entire SCAC except for the configuration information and the BCID and event counters. In particular:

- Triggers are disabled for up to 240 clock cycles, depending on the configured trigger latency.
- The currently processed event is aborted, after the current sample command is sent to the Gain Selector. All further events waiting for readout are discarded.
- The cell number sequence is regenerated.

4.4 Status Monitoring

Status information for monitoring the performance of the SCAC and checking for possibly corrupted data is obtained in two ways. First, event status and SCAC status information is communicated by the SCAC to the Gain Selectors and included in the event data output to the ROD. Second, a 32-bit SCAC status register, containing additional information about the internal states of the SCAC can be read via SPAC.

4.4.1 Output Data Status

The FEB data format includes 8 bits of event status in the event header which are generated by the SCAC. The SCAC sends five LSB of an event number, plus three bits encoding the phase of the 5 MHz `rclk` at the time of the trigger.

In addition, the FEB data format includes 8 bits of status information in the event trailer which are generated by the SCAC. The SCAC sends asynchronous status information in this space. These bits are not associated with the event, but accumulated events of the SCAC in general. Each bit is set in the next event after an event occurred. A description of the exact bit definitions is provided in Appendix A.2.2.

4.4.2 Status Register

The 32-bit SCAC status register can be read via SPAC. The status register includes information about the detection of single-bit or double-bit errors, FIFO overflow or underflow errors and empty flags, and information on the current state of the SCAC. A description of the exact bitfields definitions is provided in Appendix A.2.2.

5 SEU Errors

The SCAC is subjected to a flux of hadronic particles during operation, which can cause *Single Event Upset* (SEU) errors. SEU errors are changes of signals or stored states due to charge deposited in the silicon chip by ionising particles. SEU errors are not destructive to the circuit.

We distinguish three classes of SEU errors:

- Lockup. An SEU error that brings the chip into a state from which is cannot be recovered by external intervention, except a power cycle.
- Transient errors which do not require external intervention for recovery. These errors may effect a few physics events.
- Unrecoverable errors. These errors require external intervention for recovery of proper operation, like reset or initialisation procedures.

Any circuit on the FEB must be immune to lockup errors. Special care was taken in the design of the SCAC ensure that no lockup states exist. In particular, the configuration interface will always recover into a responsive state after a finite number of clock cycles. As additional protection, a Reset input is provided on the SCAC, and will be connected to the SPAC system via the Configuration Controller.

The proper functioning of the SCAC depends on the entire history since the last initialisation. Whenever the SCAC takes a wrong decision due to SEU, the sequence of SCA addresses is very likely to be corrupted, and recovery requires a reinitialisation. Due to this nature of the SCAC, its SEU cross section for transient errors is very low, and so efforts at "hardening" the SCAC design against SEU-induced errors have been focused on reducing the rate of SEU errors that corrupt the sequence of SCA cell addresses.

5.1 SEU Error Detection

SEU error detection of any kind is trivially accomplished in the *ReadOut Driver* (ROD). Since there are two SCAC chips per FEB, operated synchronously, the SCA read addresses and other information (e.g.,. BCID) which appear in the FEB output data, must be identical for both SCAC. Therefore, by comparing the readout data streams from the two halves of the FEB, any SEU-induced error in a SCAC will more or less immediately be visible at the ROD.

This diagnostic tool requires that, in the case of SEU in either SCAC on a FEB, the Reset must be applied synchronously to both SCAC on the FEB in order to keep them in lockstep.

5.2 SEU Protection Measures

All SEU protection measures work by storing redundant information and correcting any errors before the corrupted state takes effect.

5.2.1 Configuration Storage

The SCAC is configured by downloading 80 bits through the serial configuration interface. The corruption on any of these bits would result in incorrect operation of the SCAC. However, recovery from this type of error requires actually downloading the configuration bits again, whereas any other type of SEU error only requires a single INIT command for complete recovery.

To reduce the likelihood of corruption of the configuration, all 80 configuration bits are stored in three flip-flops each, and majority voting is used to resolve any ambiguities.

5.2.2 State Machines

Since complete protection is neither required nor feasible, most of the SCAC's logic is not protected against SEU errors. However, whenever a simple protection mechanism could be identified, it was implemented.

For example, the initialisation logic employs a single redundant flip-flop to prevent an 8-bit counter to issue random initialisation sequences. Also, the trigger queue contains redundant logic to suppress event samples beyond the configured length of the event pattern, which takes 27 flip-flops out of the chip's SEU cross section for normal 5-sample data taking.

5.2.3 SCA Address FIFOs

The largest contribution to SEU on the SCAC are the SRAM of the FIFOs.

The control logic of the FIFOs is also vulnerable to SEU. The SRAM blocks have address input registers. These registers are unprotected. The FIFO logic must store the read and write pointers in registers too. To eliminate half of this vulnerability, the FIFO logic uses the state of the SRAMs input register as read and write pointers. The SRAM block provides buffered address outputs reflecting the current state of the address registers.

However, all information stored *in* the FIFOs includes redundancy to protect against SEU-induced errors. Every 8-bit SCA cell address is accompanied by 5 Hamming code bits, which allow for correction of single bit errors, and detection of double bit errors. The *Error Detection and Correction* (EDC) algorithm is applied to the cell addresses before the SCAC takes a decision based on the address value, and before an address is sent off-chip.

Three EDC units are employed:

- In the readout logic, the cell number is corrected before is is sent to the Gain Selector, and before it is inspected for validity. Invalid cell numbers are not passed on to the `done_fifo`, nor is a read command issued to the SCA.
- The cell numbers are corrected before they leave the `free_fifo`, to ensure that the sequencer logic takes the correct decision. The out-

put register of the free_fifo is not protected. The EDC unit and sequencer logic cannot operate in a single pipeline step, when worst case timing models are applied during synthesis.

- The cell numbers are corrected before they leave the done_fifo. The done_fifo is implemented with flip-flops, which are less SEU sensitive than SRAM cells. The cell numbers were corrected in the readout logic before they were sent to the done_fifo. The output register is not protected, for the same reason as in the free_fifo. The cell numbers spend most of the time in the output register. In summary: this EDC unit has very little effect.

When any EDC unit encounters a single or double bit error, a bit is set in the next event's status word. The double bit error flag is likely to be set in multiple event, since the error is not correctable and will be encountered again. The occurrence of EDC errors can also be detected by reading the SCAC status register via the serial configuration interface.

A cell address that is transferred from the latency_fifo to the free_fifo is checked for validity, without being subjected to EDC. The most significant bits of the cell address are stored in the latency_fifo with sufficient redundancy and restored at the output in a way that ensures a correct decision. This redundancy comes at no extra cost, since the word size of the SRAM provides these bits which would otherwise be unused.

In addition to SCA cell addresses, the event_fifo and sample_fifo store some extra sample status bits. These additional bits are protected by triple redundancy and majority voting.

The event_fifo also stores the BCID, event number, and read clock phase. These bits are unprotected, since they are not used for any decision within the SCAC. This is a rare example for a transient SEU error in the SCAC, with an SEU corrupting the data for a single event.

The event_fifo implements a wide word width of 35 bits by storing the data in three consecutive words into the 18 bit wide FIFO. Four bits out of 18 are used to encode the sequence of words in the event_fifo. When the code for the last word is encountered at the output of the FIFO, the event record is considered complete, and sent to the readout. When the code for one of the first two words is encountered, the FIFO output is registered in the appropriate output registers. The codes are arranged such that a single bit error cannot fake or corrupt the *last-word* code. The *second word* code cannot be corrupted, but a *first word* code can fake a *second word* code. The first word contains the least sensitive information (BCID, event number, clock phase). Only this part can be corrupted by single bit errors in the sequence codes.

5.3 Sensitive Sequential Nodes Count

The SCAC contains 9216 SRAM cells, some of which are entirely unused, and most of which do not contain valid information at a given time. The number of stored bits is roughly the number of cell addresses times the address size plus Hamming codes: $144 \times (8 + 5) = 1872$ bits.

During trigger activity, the number of valid bits increases due to state information stored in the `event_fifo` and `sample_fifo`, and due to invalid cell numbers inserted into the sequence. Furthermore, the extra redundant bits in the `latency_fifo` are not included in the count above.

In addition to the SRAM, the SCAC contains 957 D-flip-flops, including those embedded inside the SRAM blocks. Among these are:

240: Configuration registers. Protected by majority voting.

80: FIFO state. Unprotected.

9: Initialisation counter. Special protection.

17: BCID and event counter. Unprotected. Non-critical.

51: Configuration interface. Unprotected. Low probability of SEU interfering with the SCAC function.

97: Trigger queue. Somewhat protected. Cross section depends on sample pattern configuration.

123: Readout machine. Partly cell address datapath. Some probability of SEU errors recovering without affecting the sequence. Recovery probability is lower when cell address sequencing is turned off.

100: Cell address data path registers, including `done_fifo`. Partly protected by Hamming codes.

15: TTC and clock phases. Unprotected. Critical.

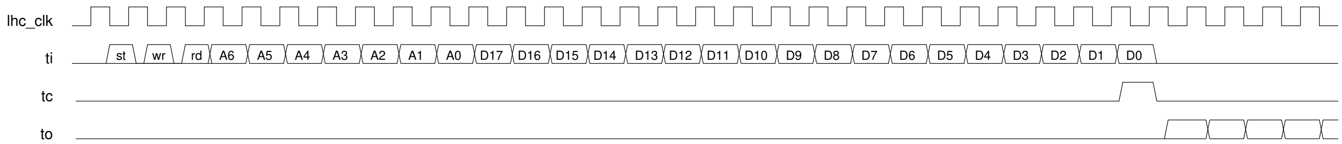

Figure 2: SRAM test timing diagram. The `tc` pulse is shown to validate the displayed command bits in the `latency_fifo` (first in chain). To apply the command to another FIFO issue the `tc` a multiple of 28 clock cycles later. The `to` output is shown as it would be coming from the `sample_fifo` (last in chain) in response to the shown `tc` pulse.

32: Controller status registers. Unprotected. Non-critical.

113: SRAM test port. Protected.

Finally, the four SRAM blocks have 18 data output latches each, which benefit from the EDC protection of the FIFO's contents.

6 SRAM test interface

The SCA Controller chip provides three extra pins that allow to test the four SRAM blocks. Think of this as a Poor Man's JTAG.

The interface runs at the 40 MHz clock. Address, data, and command bits are shifted into the `ti` input, and shifted back out to the `to` output, with some fields replaced by a requested read operations. An SRAM test operation is performed by a signal on the `tc` input.

All four SRAMs are chained together. The data bits enter the `latency_fifo` first, followed by the `free_fifo`, the `event_fifo`, and the `sample_fifo`.

Each FIFO contributes 28 bits to the shift register: three command bits, 7 address bits, and 18 data bits. The three command bits enable (first to last) FIFO status readback, FIFO write, or FIFO read. See Fig. 2 for details of the timing.

FIFO write or read operations will overwrite the corresponding SRAM address register, and thus destroy the FIFO's state.

A write operation does not change the shift register bits. A read operation replaces the data bits. The status readback command replaces the address and data bits with: 7 read address bits, 8 zeros, the empty flag, the SRAM's `rd` command, the SRAM's `wr` command, and 7 write address bits. The status readback command does not interfere with the FIFO operation.

Operation of the test involves to shift in the required command, address, and data bits, and supply a command pulse right at the time when the bits are in the proper place, and then shift out the results, if any.

On the FEB, both `ti` and `tc` will be tied to GND. An single bit SEU in either the shift register or the `tc` input register will not have any effect. An SEU in the `tc` input register does not trigger a command because all bits in the shift register are zero, and a spurious bit in the shift register has no effect unless there is a `tc` hit at the same time when the spurious bit passes a command bit position.

7 Design Flow

The DSM design flow for the SCAC consists of the following steps:

- Design entry.
- RTL level simulation.
- Synthesis.
- Gate level simulation.
- Place and route.
- Timing extraction and simulation.
- Physical verification.

7.1 DSM Standard Cell Library

The target standard cell library named `cmos6sf25` was developed at CERN, for a 0.25 μm process.

Radiation hardness is achieved by using circular enclosed transistors, which completely eliminates the dominant *Total Ionising Dose* (TID) failure mode in CMOS circuits, namely *Gate Edge Leakage*, since the transistor gates do not have any edges.

The circular shape of the transistors impose an area penalty, which is more than compensated by the small feature size of the process technology.

7.2 SRAM

The SRAM was developed at CERN by Kostas Kloukinas, who was also responsible for most of the library development. The memory cell design was based on an earlier development by Alessandro Marchioro, which was a true dual ported SRAM cell, but lacked scalability and modularity features.

The new design is modular, with separate blocks for memory cells, row decoders, readout decoder, etc. The timing properties are self-adjusting to allow different configurations of SRAM width and depth.

The basic memory block is organised in 128 words of 9 bits, with a single access port, operating at frequencies well in excess of 80 MHz. The control blocks support pseudo dual ported operation at 40 MHz by splitting the cycle in a read and write operation.

Registers are added to the SRAM configuration on all data, address and control inputs. Latches are added on the data outputs.

7.3 Verilog Design

The SCAC design is described by a hierarchy of Verilog modules. The modules fall into these categories:

- Top level design module.
- Sub-design modules.
- Parameterised sub-design modules.
- Explicit IO pad instantiations.
- SRAM module instantiations.
- Explicit library cell instantiations.

The top level module `scac` describes the complete chip, with the physical inputs and outputs as module ports. The major blocks of the design are instantiated in this module.

The interfaces between the major sub-design modules are cleanly defined to allow for the replacement of sub-design implementations. In particular, the FIFO implementations can be switched to test various implementation ideas, possibly depending on the target process and library. For example, the fall-through time of the FIFOs does not affect the fundamental functionality of the circuit. Also, the use of Hamming codes and EDC to protect the cell addresses is optional, and can be suppressed by a directive to the Verilog preprocessor.

The major sub-designs are:

`scac_config` Configuration interface and parameter storage. The serial configuration interface submodule is identical to the one used in the Gain Selector.

`ttc` Trigger and timing interface.

`initsequence` Sequence initialisation machine. This module also controls the reset of the SCAC.

`sample_queue` Trigger queue. This module implements the event sample pattern generation.

`latency_fifo, event_fifo, sample_fifo, free_fifo`

The four SRAM based FIFOs. These are all implemented in terms of a parameterised FIFO model. A special preview data path is provided between the `latency_fifo` and the `free_fifo` to allows for zero fall-through implementation of the `free_fifo`.

`readout` Readout controller. This module watches the `event_fifo` and `sample_fifo` outputs and initiates readout operations.

`done_fifo` This FIFO is only four words deep, and thus implemented with flip-flop cells from the target library.

`select` The cell address sequencing logic. This module selects the cell address to be sent to the SCA write address bus, from three sources: `free_fifo`, `done_fifo`, and `initsequence`.

7.3.1 Parameterised FIFO module

The standard FIFO implementations are based on a parameterised module. The parameters specify the FIFO's width and depth. The DSM implementation of this module ignores the depth parameter, and uses the width parameter only for the width of the input and output data ports. The FIFO is then implemented with a 128×18 bits SRAM module. The excess bits in the SRAM data word are unused, with the unused inputs tied to GND.

The output semantics of this FIFO module is defined such that the next data word is presented at the output whenever the FIFO is signalling not-empty, and a read command pops that word and replaces it with the next one, if present.

The current zero fall-through implementation of the `free_fifo` requires that the underlying FIFO has a fall-through time of two cycles, meaning that the data appears on the output during the second cycle after the input data was asserted on the inputs.

7.3.2 SRAM modules

The FIFOs are build around a dual ported SRAM module with 18 bits data word width and 128 words depth. A Verilog module describes this RAM for simulations, and a synopsis library models the IO loads and drives for synthesis.

The SRAM was designed by Kostas Kloukinas at CERN. The design consists of a set of modules that can be combined to a wide range of RAM configurations. The basic SRAM module is 128×9 bits. The minimum SRAM word width for the SCAC with EDC is 13 bits, so that a single configuration with two basic SRAM modules and a single row decoder is appropriate for all FIFOs.

The SRAM is not a real dual port design, but rather alternating read and write cycles at 80 MHz. The read cycle is following the rising edge of the clock, and the write cycle during the low phase.

All inputs are registered at the rising clock edge. The outputs are latched, with an access time from the rising clock edge of about 4 ns typical. The output values remain latched when no read is requested during a cycle.

7.3.3 Explicitly Instantiated Library Cells

The top level module instantiates the LVDS input and output pad cells from the `cmos6sf25PadsLib`.

7.4 Clock Distribution

The SCA Controller operates internally on three clocks: The 40 MHz LHC clock, and both phases of the 5 MHz read clock. We designed two clock buffer cells, built of library inverter cells. These clock buffer cells provide the clock signal with sufficient drive strength for a clock trunk architecture.

The clock trunks were prerouted over the chip core, one clock line for each clock for sets of eight core cell rows. The route tool was instructed to connect each flip-flop clock input with a dedicated route to a trunk line.

The advantages of a clock trunk over a balanced clock tree are two. First we did not have to mess with yet another automated tool to generate a clock tree *(ctgen)*. Secondly, the high capacitance of he clock net makes the chip less vulerable to SEU. An SEU in the low capacitance root of the clock tree would cause additional clock edges in part of the chip. We did use *ctgen* for the DSM Gain Selector, though.

7.5 RTL Level Simulation

The Verilog design files are simulated with a testbench module that runs the chip through all major modes of operation, and logs all chip output to the output file.

The chip output is too complex to verify it correctness during simulation. Only verification of the data format of the sample commands is performed by the testbench module.

During the development, the correct functioning of the design was verified by visual inspection of the simulation waveforms and the log output. The output was then saved into a master copy, which was henceforth compared to each new simulation output. Whenever changes occurred, these were verified as intended changes in the designs behavior, and the master copy was replaced with the new output.

Simulations were performed with the Cadence Verilog-XL and the Icarus Verilog simulators. The outputs of both simulators were compared. Some differences were found to be due to race conditions in the testbench code. These races were removed from the testbench. Several bugs in the evolving Icarus Verilog simulator were found and fixed.

The occurrence of SEU-induced errors in the SRAM is simulated in the testbench by randomly switching bits in the four SRAM instances. The two simulators use different random number generators, so that their output is not directly comparable. All remaining differences were identified to result from the random SEU simulation test. All chip output is identical, except for

- SEU error status bits in the event status record.
- A few corrupted BCID records, due to SEU in the `event_fifo`.

7.6 Synthesis

The design synthesis is performed with Synopsys. In addition to the cmos6sf25 core and pad libraries, two self-made libraries are linked to the design: the SRAM module, and the clock drivers. The synthesis is driven via a `Makefile` and Synopsys command scripts.

7.7 Gate level simulation

The generated netlist is simulated with the same testbench as used for RTL level simulation. The logfile output is compared to the RTL level output and required to be identical. Both Verilog-XL and Icarus Verilog were used, but Icarus Verilog does not simulate the specified delays.

7.8 Place and Route

A Silicon Ensemble command script is used to place and route the design. The RAM modules and clock buffers are declared as subblocks, and placed separately from the chip core. The placement of the IO cells and the power routing to the pads is performed by scripts and IO constraints files that were automatically generated from the Verilog netlist using AWK.

7.9 Timing Extraction and Simulation

A *Standard Delay Format* file (SDF) was extracted from the layout. Using Verilog-XL, the wire delays are attached to the netlist timing paths and simulated in the usual testbench. The logfile output is identical to the previous simulation results.

7.10 Physical Verification

Design Rule Check DRC was performed with Diva, Dracula, and finally, at CERN, with Hercules.

Layout Versus Schematic comparisson (LVS) was performed with Diva before streamout. The LVS did not include the full SRAM, since the schematics for the inner parts of the RAM were not released by CERN. The full SRAM block was LVS checked at CERN, and our LVS covered all RAM block submodules that interface to the ports, so the the LVS check overlap. LVS after streamout was omitted, because of the difficulties of doing an LVS without full schematics.

The final DRC check with the Hercules tool were performed at CERN. The violations the were flagged by Hercules envolved libaray parts beyond our control, e.g., the SRAM cells and the LVDS drivers. The foundry accepted the waiver for these violations.

7.11 Design Status

The design was completed end of February 2001, and submitted for MPW production (MPW4). Samples were received in July, which appeared to be fully functional. A few misfeatures were discovered in the design during testing. These bugs are non-critical for FEB prototyping. A corrected design will be submitted in November 2001 for an MPW production (MPW6). See Appendix C for a description of the bugs and workarounds.

The current cell library does not provide proper ESD protection on all IO cells, and for the power rails. A new set of cells was made by CERN, based on a ESD kit from the foundry, and modified for TID radiation tolerance. The design for the MPW6 submission will use these new IO cells, and some additional devices for ESD protection of the power rails.

7.12 Die Size

Due to quantisation effects on the MPW reticle, the edges of the chip are required to grow in steps of 2 mm. The minimal chip size of the SCAC would be $2.7\times2.7\,\mathrm{mm}^2$. The design would fit into $2\times4\,\mathrm{mm}^2$, but that shape would impose difficulties for packaging. This leaves the option of $4\times4\,\mathrm{mm}^2$ chip size, which matches well with the properties of the chosen package.

A SCAC Configuration

The configuration and operation of the SCAC is controlled and monitored via a serial configuration port from the Configuration Controller. This port provides an 8-bit address space and up to 32 bits of data per transfer.

A.1 Configuration Address Space

The configuration address consists of two parts: the chip select part, and the on-chip resource address. The four MSBs of the address determine if the SCAC is addressed or not. The four LSBs tell what the SCAC is supposed to do.

A.1.1 Chip select

The SCA Controller responds to *read* and *write* commands from the Configuration Controller to address (binary) `1i00cccc`, where `i` is the status of the `chip_id` option input, and `cccc` is the code for the requested operation.

In addition, the SCAC controller accepts *write* commands for address (binary) `1x01cccc`, with bit 6 (`x`) being ignored. This is for broadcast commands to both SCA controllers.

Two classes of operation can be performed via the configuration interface. First, the configuration and status registers can be written or read, and secondly, a TTC command can be sent.

A.1.2 Configuration and Status

Four registers can be read or written:

`1i000r00`	Mode Register.
`1i000r01`	Pattern Register.
`1i000r10`	Read Delay Register (16 bits).
`1i000r11`	Status Register (read only).

Bit 2 (`r`) is the *reset* bit. If the *reset* bit is set with any register write command, the SCAC is reset. If the *reset* bit is set for reading the Status Register, the eight LSB of the accumulated status are updated. Writing to the Status Register has no effect, apart from a reset of the SCAC, when the *reset* bit is set.

A.1.3 TTC commands

TTC commands are performed by writing to one of the addresses listed below. The data bits are ignored. The SCAC can accept four different TTC commands:

`1i001f00`	L1A.
`1i001f01`	BCID reset.
`1i001f10`	Event Counter Reset.
`1i001f11`	INIT.

Each TTC command is accepted only when the corresponding command from the TTCrx chip is disabled in the Mode Register. Except, when bit 2 (`f`, *force*) is set in the command address, the TTC command is accepted regardless of the enable bits in the Mode Register.

A.2 Configuration Data Formats

The SCAC has three configuration registers: The mode register, the sample pattern, and the readout delay. The forth register is a read-only status register.

Writing into configuration registers immediately effects the SCAC operation. For reliable operation, the chip should be reset when any changes are written into these registers. Only the latency field of the mode register does not affect the SCAC operation. Changes to the latency field take effect only during initialization.

A.2.1 Mode Register Format

Fig. 3 gives an overview of the data format of the registers. The bit field of the mode register are explained below:

[3:0] TTC command enable bits. These four bits enable the four TTC commands from the TTCrx. The bits 0 to 3 enable `Brcst_Init`, `EvCntRes`, `BCntRes`, and `L1Accept`.

[5:4] Gain mode. If the internal read timing control is enabled, these two bits must correspond to the gain mode of the Gain Selectors. The values 0 to 2 select fixed gain modes, 3 selects auto-gain mode.

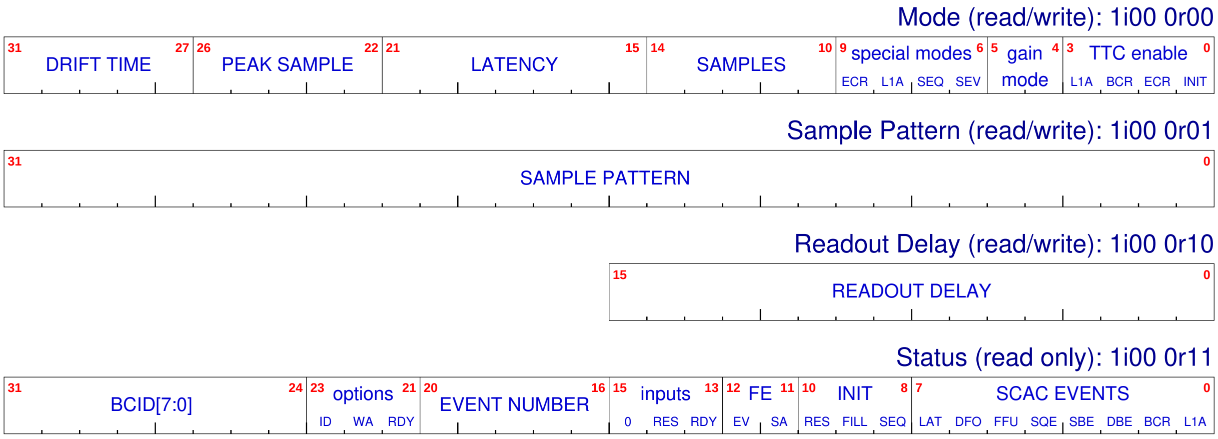

Figure 3: Configuration and status register format.

[6] Single event operation. When this bit is set, the SCAC rejects L1Accept while an event readout is in operation.

[7] Sequence disable. When this bit is set, the cell address sequence is not maintained. SCA cells are reused immediately after readout. Otherwise, the cell numbers are put on hold until they fit into the sequence. Reasons to disable sequencing include:

- Test measurements on the effect of write-address pattern noise.
- Minimisation of readout latency. Sequencing can hold a cell number for up to a full pipeline cycle.
- Certain pathologic sample pattern fail with sequencing enabled. These are fixed single gain pattern with three or less samples, when the first sample is not read first, and auto-gain mode with two samples, gain selection on the second sample.

[8] L1Accept encode. When this bit is set, the L1Accept signal is expected to be true on two consecutive clock cycles, for a trigger to be recognised. This helps suppressing fake triggers due to SEU.

[9] Event number reset mode. This bit selects the reset source for the 5-bit event number counter. If set, the counter is reset by the BCntRes signal, else by the EvCntRes signal.

[14:10] Number of samples. This bitfield specifies the number of non-peak samples, i.e., the number of samples minus one, in the range 0 to 31. This specification must agree with the number of bits set in the sample pattern.

[21:15] Trigger latency. This bitfield defines the L1 trigger latency in terms of the number of cells in the latency pipeline. The range is 14 to 127. This bitfield has effect only during sequence initialisation operation.

[26:22] Peak sample selection. This bitfield specifies the position of the peak sample in the sample pattern, in the range 0 to 31. The corresponding bit position must also be set in the pattern register.

[31:27] Drift time duration. The gain selection algorithm for an event that occurs during the drift time of the previous event is modified, to prevent saturation of the ADC. This bitfield tells how many clock cycles must go between the first sample of the first event to the peak sample of the second one, in order to not modify the gain selection of the second event.

The Sample Pattern register has 32 bits, each bit telling if the corresponding sample should be included in a triggered event or not. Bit 0 is the first sample, and should always be set. The bit at the position of the peak sample must also be set. The total number of bits set must agree with the number of bits specified in the mode register, plus 1, for the peak sample.

The readout delay register is 16 bit, telling how many 5 MHz cycles the SCAC should wait after each event, before it sends the next one to the Gain Selector. With the minimal value of 0, the events are processed back to back, obeying the constraints of the readout format in all modes. But see Appendix C for using fixed gain modes with MPW4 prototype chips.

A.2.2 Status Register Format

The bitfield definitions for the SCAC Status Register are as follows:

[0] L1A. An event was triggered.

[1] BCID Reset. The BCID counter was reset.

[2] Double Bit Error. A cell address was found with two bits in error. This bit will reoccur repeatedly, since the error will not be corrected.

[3] Single Bit Error: A cell address was found with a single bit error. This error was corrected.

[4] Sequence Error: The `free_fifo` emitted the same cell number twice in a row.

[5] `free_fifo` Underrun. The SCA run out of free cells. Invalid cell numbers were inserted into the sequence.

[6] `done_fifo` overflow.

[7] SEU in the `latency_fifo`'s MSBs.

[8] Sequence initialisation in progress.

[9] `latency_fifo` initialisation in progress.

[10] SCAC is held in reset state.

[11] `sample_fifo` is empty.

[12] `event_fifo` is empty.

[13] Gain Selector is ready.

[14] Level on the reset input.

[15] 0.

[20:16] Event number counter value.

[21] External `s_ready` is used.

[22] Write address bus is mirrored.

[23] `Chip_ID` input.

[31:24] Eight LSB of the BCID counter.

The lower eight bits [7:0] are registered events. The higher 24 bits [31:8] reflect current state of the SCAC.

The registered event bits are accumulated during operation. These accumulated bits are transfered into the status register when the reset bit 2 is set in the read address. The events accumulator will be reset at the same time. Reading the status register with the reset bit 2 zero yields the same data since the last update.

The bits [7:1] correspond to the same SCAC events as in the status word that is sent in the readout event trailer. However, the events are accumulated and reset independently for the two access methods, so that each SCAC event is visible both through the readout and the configuration interface.

The procedure to reliably read the status register with an unreliable SPAC bus is as follows:

1. Read the status register with the reset bit set. If the read succeeds, you are done.

2. In case the SPAC write phase of the read operation succeeded, but the read phase failed: retry with address bit 2 reset, until it succeeds.

3. If the address write phase of the read operation failed, retry with the address bit 2 set.

A.2.3 Event Trailer Status

Some status bits are available via SPAC as described above, and in the event trailer status word of each event. The 8 bits of the SCAC status transmitted with the event trailer are:

[1] Init. This is the first event after initialization.

[2] BCID Reset. The BCID counter was reset.

[3] Double Bit Error. A cell address was found with two bits in error. This bit will reoccur repeatedly, since the error will not be corrected.

[4] Single Bit Error: A cell address was found with a single bit error. This error was corrected.

[5] Sequence Error: The `free_fifo` emitted the same cell number twice in a row.

[6] `free_fifo` Underrun. The SCA run out of free cells. Invalid cell numbers were inserted into the sequence.

[7] `done_fifo` overflow.

[**8**] SEU in the `latency_fifo`'s redundant MSBs.

The bit positions are given as they appear in the data stream generated by the Gain Selector. Bits 0 and 11 of the event trailer are forced high by the Gain Selector. Bits 9 and 10 are EDC status bit generated by the Gain Selector. The higher bits carry protocol information.

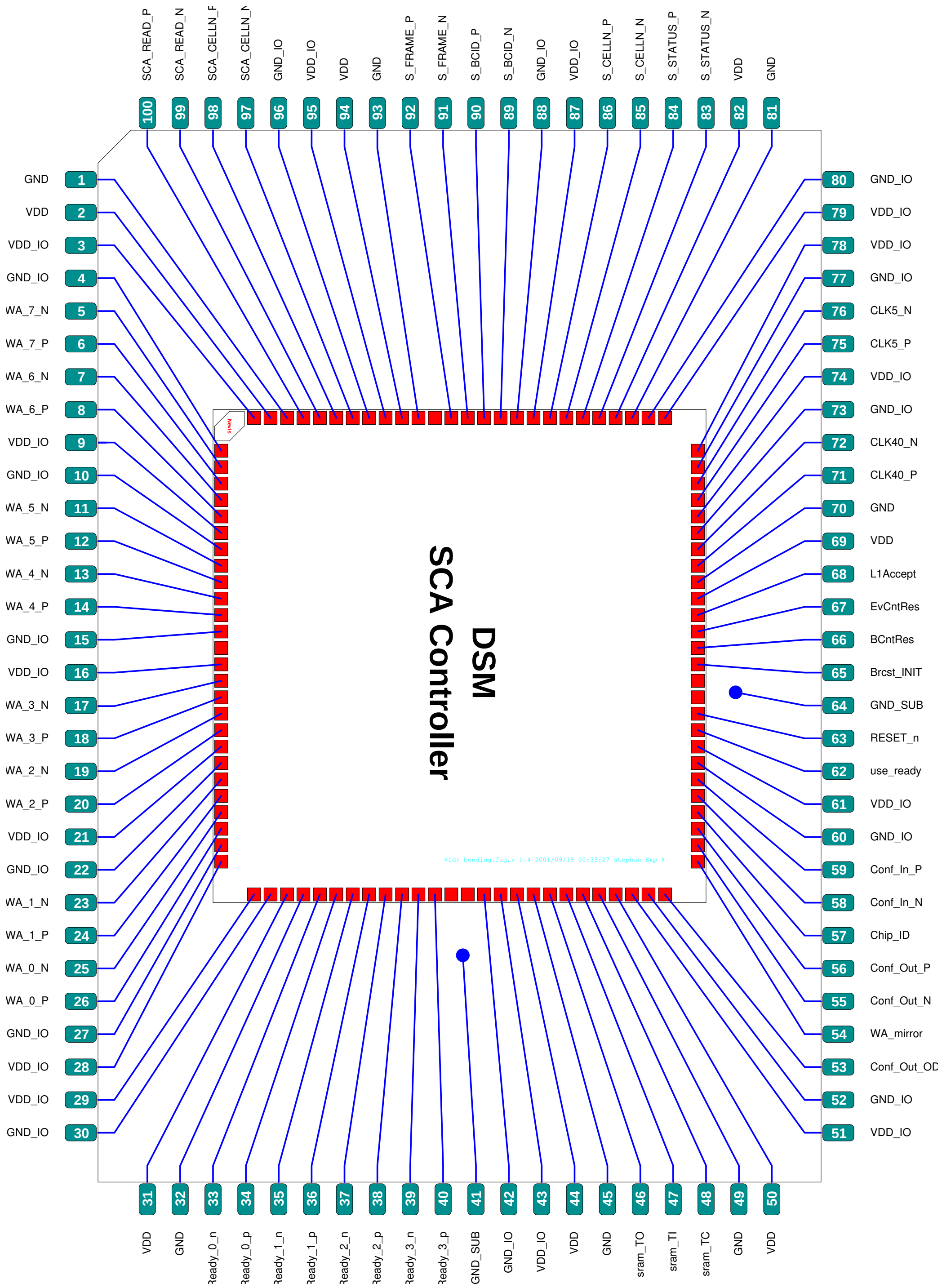

Figure 4: SCA Controller Bonding diagram and pinout.

B IO pins

Figure 4 shows the pinout and bonding diagram of the SCAC. The pins are:

1 GND core.

2 V_{DD} core.

3 V_{DD} IO.

4 GND IO.

5 `WA[7]_n`. LVDS output(-). SCA write address bit 7.

6 `WA[7]_p`. LVDS output(+). SCA write address bit 7.

7 `WA[6]_n`. LVDS output(-). SCA write address bit 6.

8 `WA[6]_p`. LVDS output(-). SCA write address bit 6.

9 V_{DD} IO.

10 GND IO.

11 `WA[5]_n`. LVDS output(-). SCA write address bit 5.

12 `WA[5]_p`. LVDS output(+). SCA write address bit 5.

13 `WA[4]_n`. LVDS output(-). SCA write address bit 4.

14 `WA[4]_p`. LVDS output(+). SCA write address bit 4.

15 GND IO.

16 V_{DD} IO.

17 `WA[3]_n`. LVDS output(-). SCA write address bit 3.

18 `WA[3]_p`. LVDS output(+). SCA write address bit 3.

19 `WA[2]_n`. LVDS output(-). SCA write address bit 2.

20 `WA[2]_p`. LVDS output(+). SCA write address bit 2.

21 V_{DD} IO.

22 GND IO.

23 `WA[1]_n`. LVDS output(-). SCA write address bit 1.

24 `WA[1]_p`. LVDS output(+). SCA write address bit 1.

25 `WA[0]_n`. LVDS output(-). SCA write address bit 0.

26 `WA[0]_p`. LVDS output(+). SCA write address bit 0.

27 GND IO.

28 V_{DD} IO.

29 V_{DD} IO.

30 GND IO.

31 V_{DD} core.

32 GND core.

33 `Ready[0]_n`. LVDS input(-). ready from GS.

34 `Ready[0]_p`. LVDS input(+). ready from GS.

35 `Ready[1]_n`. LVDS input(-). ready from GS.

36 `Ready[1]_p`. LVDS input(+). ready from GS.

37 `Ready[2]_n`. LVDS input(-). ready from GS.

38 `Ready[2]_p`. LVDS input(+). ready from GS.

39 `Ready[3]_n`. LVDS input(-). ready from GS.

40 `Ready[3]_p`. LVDS input(+). ready from GS.

41 GND substrate.

42 GND IO.

43 V_{DD} IO.

44 V_{DD} core.

45 GND core.

46 `TO`. CMOS output. SRAM test serial output.

47 `TI`. CMOS input. SRAM test serial input.

48 `TC`. CMOS input. SRAM test command strobe.

49 GND core.

50 V_{DD} core.

51 V_{DD} IO.

52 GND IO.

53 `Conf_Out_OD`. Open Drain. Configuration output.

54 `WA_mirror_n`. CMOS input. Mirror WA bus when low.

55 `Conf_Out_n`. LVDS output(-). Configuration output.

56 `Conf_Out_p`. LVDS output(+). Configuration output.

57 `Chip_ID`. CMOS input. Configuration ID.

58 `Conf_In_n`. LVDS input(-). Configuration input.

59 `Conf_In_p`. LVDS input(+). Configuration input.

60 GND IO.

61 V_{DD} IO.

62 `Use_Ready`. CMOS input. Use Ready inputs when high.

63 `Reset_n`. CMOS input. Reset SCAC when low.

64 GND substrate.

65 `INIT`. CMOS input. INIT broadcast from TTCrx.

66 `BCntRes`. CMOS input. BCID reset broadcast from TTCrx.

67 `EvCntRes`. CMOS input. ECR broadcast from TTCrx.

68 `L1Accept`. CMOS input. L1Accept trigger from TTCrx.

69 V_{DD} core.

70 GND core.

71 `clk40_p`. LVDS input(+). 40.08 MHz clock.

72 `clk40_n`. LVDS input(-). 40.08 MHz clock.

73 GND IO.

74 V_{DD} IO.

75 `clk5_p`. LVDS input(+). 5 MHz clock.

76 `clk5_n`. LVDS input(-). 5 MHz clock.

77 GND IO.

78 V_{DD} IO.

79 V_{DD} IO.

80 GND IO.

81 GND core.

82 V_{DD} core.

83 `S_status_n`. LVDS output(-). Sample command status.

84 `S_status_p`. LVDS output(+). Sample command status.

85 `S_celln_n`. LVDS output(-). Sample command celln.

86 `S_celln_p`. LVDS output(+). Sample command celln.

87 V_{DD} IO.

88 GND IO.

89 `S_bcid_n`. LVDS output(-). Sample command bcid[7:0].

90 `S_bcid_p`. LVDS output(+). Sample command bcid[7:0].

91 `S_frame_n`. LVDS output(-). Sample command control.

92 `S_frame_p`. LVDS output(+). Sample command control.

93 GND core.

94 V_{DD} core.

95 V_{DD} IO.

96 GND IO.

97 `SCA_celln_n`. LVDS output(-). SCA read address.

98 `SCA_celln_p`. LVDS output(+). SCA read address.

99 `SCA_read_n`. LVDS output(-). SCA read control.

100 `SCA_read_p`. LVDS output(+). SCA read control.

In case of discrepancies, the pinout diagram Fig. 4 takes precedence.

C Misfeatures of MPW4 chips

The first DSM SCAC chips were produced by the CERN MPW4 run submitted in February 2001. Extensive testing of these chips revealed a few mistakes in the design of the logic. The implementation in silicon was flawless, bug for bug compatible with the design entry.

The bugs were, in the order of discovery:

1. The `free_fifo` bug.
2. The fixed gain mode bug.
3. The `BCntRes` `EvCntRes` pin swap.
4. The clock buffer strength mismatch.

Non of these bugs renders the chips unusable. All these bugs were fixed in the new version of the design, to be submitted for run MPW6 in November 2001. The new version also incorporates the new ESD protection scheme developed at CERN.

C.1 The `free_fifo` bug.

C.1.1 Impact

Under certain conditions, when the `free_fifo` runs almost out of free cells, the `free_fifo` swaps two cell numbers at the output to the sequencer. No cell numbers are lost or duplicated, but the sequence is split in two non-overlapping subsequences. The pattern noise at the SCA write address doubles.

This happens not very often, about once per second under normal running conditions.

C.1.2 Workarounds

A) Accept the higher pattern noise. B) prevent triggers that could cause the `free_fifo` occupancy to run below three free cells. C) run with low trigger latency (less than about 60 clock cycles).

C.1.3 Description

The `free_fifo` employs some tricks to achieve zero fall through time, in order to reduce the SCAC's contribution to the L1Accept latency. The logic that keeps track of the status of the bypass is getting confused when there are exactly two free cell numbers in the `free_fifo`, and at the same time two cell numbers return from readout into the pipeline in back to back clock cycles. This happens when a cell number needs to wait for almost a full rotation to find its place in the sequence, while the subsequent cell number become available too.

Small trigger latencies help, because the length of the pipeline become shorter, and the probability of two cell numbers becoming available while the `free_fifo` is empty goes down. Also, the `free_fifo` does not run empty so often.

C.1.4 Fix

The comments in the Verilog code correctly described the state transitions that need to be followed. The code did not implement the recipe correctly. This has been fixed in the Verilog design.

C.2 The fixed gain mode bug.

C.2.1 Impact

When operating the SCAC with internal readout timing running a fixed gain mode, the sample commands for events in sequence are issued two cycles too early. The Gain Selector cannot operate under those conditions.

C.2.2 Workarounds

A) (deprecated) Do not operate the SCAC with internal timing, but use the `s_ready` from the Gain Selector to get the readout timing. B) Program a delay of two cycles into the SCAC readout delay register.

C.2.3 Description

Fixed gain mode operation requires a gap of 5 read clock cycles between the last sample command of the previous event to the first sample command of the next one. This gap is required to leave space in the readout stream for event header and trailer words.

Autogain mode does not require this extra gap, since the autogain algorithm itself requires additional eight cycles, leaving a gap of three cycles in the readout stream.

The readout design was assuming, both in the comments and the actual code, that three cycles were required, instead of five.

C.2.4 Fix

Considering the availability of the readout delay register to produce the correct delay between events for any gain mode, no extra logic is required. Those three cycles that the MPW4 chips use are definitely wrong and confusing. The choice was to remove the fixed gain mode delay logic entirely, or to implement the correct delay of five cycles. The later was chosen, to avoid any further confusion for the users of the chip. The delay register remains truly optional in all gain modes.

The workaround of programming two extra delay cycles lead to exactly the same readout timing as the fixed design. But the sequence of cell numbers differs, because the fixed gain mode delay is taken into account for releasing a cell number back into the pipeline, while the programmable readout delay is not.

C.3 The `BCntRes` `EvCntRes` pin swap

C.3.1 Impact

Early pinout diagrams of the SCAC showed the `BCntRes` and `EvCntRes` pin swapped.

C.3.2 Workarounds

Make sure to use the latest version of the pinout diagram. Be aware that the order of the TTC input pins on the chip does not agree with the order of the enable bits in the mode register.

C.3.3 Description

The four TTC signals `L1Accept`, `BCntRes`, `EvCntRes`, and `BrcstINIT` where listed in the design files in all module port declarations in that order. They appeared in that order on the pinout, and in the mode register. Unfortunately, at one connection down the module hierarchy, the two Reset broadcast bits were swapped. The logic of the chip is not impacted in any way.

C.3.4 Fix

To maintain compatibility with future chip, nothing has been changed in the logic. The changes only affect the notation.

C.4 The clock buffer strength mismatch

C.4.1 Impact

Reduced long term reliability of the chips.

C.4.2 Workarounds

Do not use MPW4 chips on final ATLAS boards.

C.4.3 Description

The clock buffer module for the 5 MHz clock provides a non-inverted clock for the readout logic, and an inverted clock for the configuration interface. The drive strength of these outputs were not well adjusted to the respective capacitive loads on the clock trunks. The non-inverting clock was driving much stronger than the inverted output, although the inverted clock drives twice as many flip-flops.

C.4.4 Fix

The clock buffer layout was reorganized to adjust the drive strength appropriately. This was achieved by redistributing the existing inverters between the last to stages of the four-stage driver. The footprint and port locations (abstract) of the clock buffer module remained unchanged.