# Deep Sub-Micron Implementation of the ATLAS LAr SCA Controller

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

Douglas Gingrich
University of Alberta

DRAFT 1.7, 2000/12/01 18:40:39

**Abstract**

This document describes the design, and implementation using a deep sub-micron (DSM) CMOS process, of the SCA Controller of the ATLAS LAr calorimeter readout.

# Contents

# 1 Introduction

As described in the TDR of the liquid argon (LAr) calorimeter, the LAr front end boards (FEBs) 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 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 guarantees that all samples for a given channel and given event are digitised using the same gain scale, in order to reduce possible systematic effects. The digitised samples are formatted and then transmitted off-detector to the Read Out Driver (ROD) modules for processing.

The SCA's on the FEB provide the L1 trigger latency pipeline of up to 2.5 $\mu$s, 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, 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, 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 100 krad of ionising radiation and to $5\times10^{13}$ 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 is being pursued. However, the design is rather challenging in DMILL, and the required performance is not yet demonstrated. In addition, the large size die of the DMILL design imply corresponding yield and cost concerns. As a result, an effort was launched to also implement the SCA Controller functionality using a *Deep Sub-Micron* CMOS technology; this DSM implementation is the subject of this document.

# 2 SCA Controller Architecture

A block diagram illustrating the basic architecture of the SCAC is shown in figure 1. These 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 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, and time-multiplexes eight calorimeter signal channels to one ADC.

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. In addition, 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

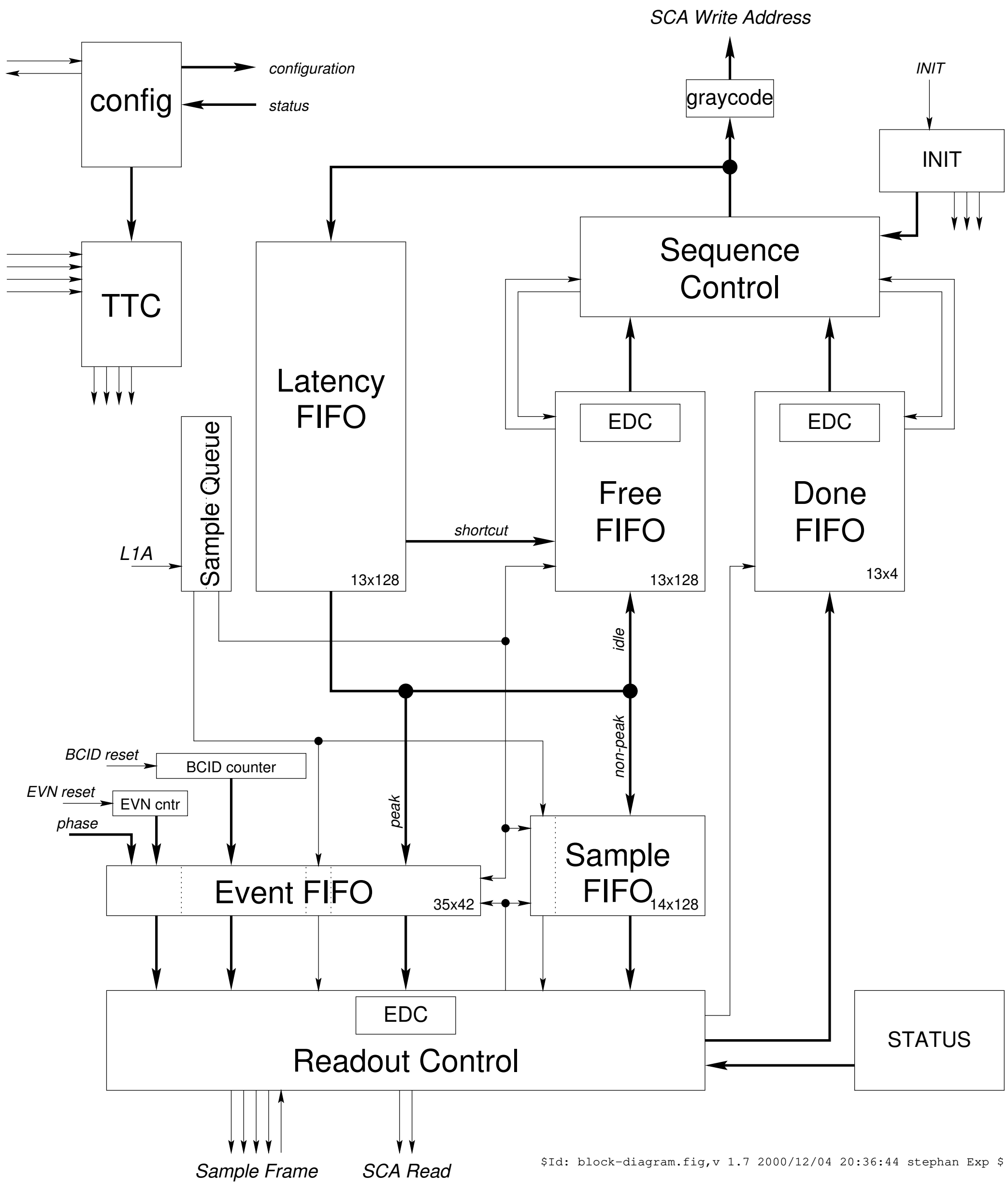


Figure 1: Block diagram.

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 de-randomising 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 config 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 config block provides the interface for configuring the SCAC in the requested operational mode. More details are provided in Section 4.

## 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.

### L1Accept

The `L1Accept` is decoded by the SCAC in a three cycle window. An internal L1A is generated when

- L1Accept is true,
- and the state of L1Accept in the previous cycle matches a configurable state,
- and the L1Accept was false 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.

#### 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 the L1Accept such that the BCID sent with the 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.

# 3 SCAC Operation

## 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.)

When cell number reordering is turned on (normal mode of operation), no more than 64 cells must be taken out of the sequence for readout at any time. The sequence will get out of order whenever a gap of 64 or more cell numbers appears in the sequence.

## 3.3 Level 1 Trigger Latency

The L1 trigger latency is configurable roughly between 64 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 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 fulfils 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 $\mu$s (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 modelled 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 means that even those samples of the first event are lost, 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. If the `sca_read` command is taken from the Gain Selector, the SCA would attempt to read a non-existing cell. The implications of this on the readout amplifier inside the SCA are not fully known (see section 3.7). The `sca_read` command generated by the SCAC is suppressed for invalid cell numbers.

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 part of the sample command for the peak sample.

The backporch bit is set by the SCAC for an event which follows the previous trigger by a number of bunch crossings which is less than a configurable number. 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 can 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 would eliminate the problems with reading non-existing cells addressed above.

# 4 SCAC Configuration

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).

## 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. 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 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.

## SCAC 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 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 word 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 word 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.

### 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 SCAC's 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 B.

# 5 SCAC 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, 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 by an 8-bit counter.

# 6 SEU-Induced 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.

## SEU-Induced 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.

## SEU Protection Measures

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

### Configuration Storage

The SCAC is configured by downloading 64 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 64 configuration bits are stored in three flip-flops each, and majority voting is used to resolve any ambiguities.

### State Machines

Since complete protection is neither required nor possible, 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 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.

### SCA Address FIFOs

The largest contribution to SEU on the SCAC are the SRAM of the FIFOs. The address pointers in the FIFO implementation are not protected against SEU errors. The SRAM block employ input registers at the address busses. These registers contribute about equally to the SEU cross section as the address pointers. Since the SRAM module was provided from an external source, we could not harden the input registers. Thus, protection the address pointers would only reduce the SEU cross section of the FIFO logic by 1/2, which was not considered worth the considerable effort required.

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 output 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. (There are actually two EDC unit in the `free_fifo`, one for cell addresses entering through the `latency_fifo` preview path, and one for the other data paths to the output register.)

- The cell numbers are corrected before they leave the `done_fifo`. The `done_fifo` is implemented with flip-flops, which are supposedly less SEU sensitive than SRAM. 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 word 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.

## 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 767 D-flip-flops. Among these are:

192: Configuration registers. Protected by majority voting.

60: FIFO address 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.

106: 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.

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

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

# 7 SCAC 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 word, containing additional information about the internal states of the SCAC, can be read back via SPAC.

## Output Data Status Information

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. The bits are:

**Bit 1:** Init. A reinitialisation of the sequence occurred.

**Bit 2:** BCID Reset. The BCID counter was reset.

**Bit 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.

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

**Bit 5:** Sequence Error: The sequence of cell addresses is corrupted. This error bit will be set without indicating harm if cell number sequencing is disabled in the configuration.

**Bit 6:** Free FIFO Underrun. The SCA run out of free cells. Invalid cell numbers were inserted into the sequence.

**Bit 7:** Done FIFO overflow. done fifo.

**Bit 8:** Chip ID (Serial address bit 6).

### SCAC Status Word

The 32-bit SCAC status word can be read back via SPAC. The status work 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 C.

## 8 SCAC Design Flow

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

- Design entry. Behavioural Verilog.
- RTL level simulation.
- Synthesis. Synopsys, cmos6sf25 radhard cell library.

- Gate level simulation.
- Place and route. Silicon Ensemble.
- Timing extraction and simulation.
- DRC and LVS. Dracula.

## 8.1 DSM Standard Cell Library

The target standard cell library named `cmos6sf25` was developed at CERN, for a 0.25 $\mu$m 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.

## SRAM Building Block

The SRAM was/is developed at CERN by Kostas Kloukinas, who was also responsible for 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 earlier SRAM was manufactured and successfully tested.

The new design is modular, with separate block 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.

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

A 2 KByte by 9 bit configuration of the SRAM was submitted for production in November 2000. Silicon is expected for Christmas, first results in January 2001.

## 8.2 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`.

### 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. Actually, only the `sample_fifo` requires this property.

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.

### 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.

### Explicitly Instantiated Library Cells

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

## 8.3 Clock Distribution

The clock distribution path is modelled with two clock buffer modules, for the 40 MHz, and the 5 MHz clocks respectively. These module are instantiated in the top-level module, between the clock inputs and the clock networks of the design. These clock buffers are modelled in a Synopsys cell library for synthesis.

There are three options to implement the clock distribution:

1. Use *ctgen* to generate a balanced clock tree. The clock buffer modules would be implemented as feed troughs and flattened before synthesis.

2. Add Verilog modules to the netlist after synthesis, that implement sufficiently strong clock buffers from library buffer cells connected in parallel. These cells would be placed and routed inside the chip core. The clock net would be a single wire.

3. Create a layout for clock buffer blocks from library cells and link these blocks to the design when importing the netlist. The clock buffer blocks would be placed outside the chip core.

We are currently using the second, simplest method.

## 8.4 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 behaviour, and the master copy was replaced with the new output.

Simulation 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`.

The simulation is driven by a `Makefile`.

## 8.5 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. A detailed list of the steps performed by the main synthesis script is found in Appendix D.

### 8.6 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 are used, but Icarus Verilog uses simplified models for the library cells. For Verilog-XL, the models provided with the library were used, except for minor modification where Verilog-XL did not properly support some language features in the original models.

### 8.7 Place and Route

The design netlist is imported into a Cadence DFII library and reexported to Silicon Ensemble. A Silicon Ensemble command script is used to place and route the design. The RAM modules are declared as blocks, and placed separate from the chip core.

### 8.8 Timing Extraction and Simulation

A *Standard Delay Format* file (SDF) and a netlist is to be 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 must be identical to the previous simulation results.

### 8.9 DRC and LVS

DRC and LVS are to be performed with Dracula.

## 9 SCAC Design Status

The design entry, simulation, and synthesis are complete, pending minor modification during further review. The place and route design flow is mostly understood. Kostas Kloukinas was very helpful in developing the design method employing a complete design netlist and routing the whole chip at once with the Silicon Ensemble tool. The place and route scripts are nearing completion, so that most steps up to the final layout are fully automated.

The SRAM configuration block is expected to be complete by the end of November 2000. Again, Kostas Kloukinas is the one to do all the work.

Schedule problems may arise if the SRAM as submitted is not fully functional. In that case, Kostas will be very busy fixing the SRAM for the new submission, so that he will not be able to spend much time to fix our SRAM configuration. If the fixed modules drop into the existing configuration easily, then we can do this ourselves.

Expertise is still required for *Design Rule Check* (DRC) and *Layout Versus Schematic* (LVS) checks with the Dracula tool.

## 9.1 SCAC 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 $2.7 \times 2.7\,\mathrm{mm}^2$. The design would fit into $2 \times 4\,\mathrm{mm}^2$, but that shape would impose difficulties for packaging. This leaves the option of $4 \times 4\,\mathrm{mm}^2$ chip size, which matches well with the properties of the chosen package.

## 9.2 Schedule

The DSM SCAC design will be submitted for inclusion in the next CERN-coordinated MPW run using the DSM process. The submission date is February 28, 2001. The guaranteed 13-week turnaround time implies, allowing 4 – 6 weeks for packaging, that the first packaged chips should be received around the beginning of July 2001.

Once the chip is submitted, we will design and fabricate a test jig for testing. In parallel, we plan to produce an FPGA version of the DSM SCAC design so that we can test and debug the various interfaces to the other FEB chips as much as possible in advance of receiving the DSM prototypes. This effort should also facilitate rapid testing of the DSM SCAC once it is received.

An engineering run could proceed in late 2001, with production slated for mid-2002.

# A SCAC Configuration Word

The bitfield definitions for the 32-bit mode configuration word for the SCAC are as follows:

[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 corresponds to auto-gain mode.

[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.
- Operation with 64 or more cells in the derandomising buffer function (80 or less cycles of trigger latency). Sequencing requires that no gaps of 64 or more cells appear in the write address sequence.
- 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, before a trigger is recognised. This helps suppressing fake triggers due to SEU.

[10:9] Event number reset mode. These two bit enable the reset of the event number counter by the BCntRes and EvCntRes respectively.

[15:11] 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.

[21:16] Trigger latency. This bitfield defines the L1 trigger latency in terms of the number of cells in the latency pipeline. The range is 64 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 word.

[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.

# B SCAC Configuration Addresses

The addresses recognised by the SCAC are binary 1x0x xxxx. The address bits are:

Bit 7: Must be 1.

Bit 6: Must match the level on the `chip_id` input, for read commands, or when address bit 4 is not set.

Bit 5: Must be 0.

Bit 4: Broadcast to both SCAC's. When set in write commands, then bit 6 is ignored.

Bit 3: Reset for configuration or status commands, set for TTC commands.

Bit 2: With configuration or status commands: reset bit. The SCAC is reset, when this address bit is set in a write configuration command.

With TTC commands: Force command bit. A TTC command with this bit reset is ignored, when the corresponding command is enabled from the TTCrx.

Bit 1: Set for status command, reset for configuration command.

TTC commands are encoded in bits [1:0]:

00: L1A.

01: BCID reset.

10: Event Counter Reset

11: Sequence Initialisation.

Status commands are used to read 32 bits of SCAC status. When address bit 0 is set, the event status bits are reset, even in write commands, which are otherwise ignored.

Configuration commands allow to read or write the configuration registers. Address bits [1:0]:

00: Mode register.

01: Pattern register.

10: SCAC status.

11: SCAC status with event bits reset.

## C SCAC Status Word

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

**[0]** Init. A reinitialisation of the sequence occurred.

**[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 sequence of cell addresses is corrupted. This error bit will be set without indicating harm if cell number sequencing is disabled in the configuration.

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

**[6]** Done FIFO overflow. done fifo.

**[7]** A L1 trigger was received.

**[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 first eight bits are accumulated during operation, and reset when the status word is accessed with address bit 0 set. The higher 24 bits reflect the current state of the SCAC. The first seven bits correspond to the same SCAC events as the first seven bits of the status word that is sent in the readout event trailer. However, the events are accumulated and reset independently for the two readout methods, so that each SCAC event is visible both through the readout and the configuration interface.

# D Synthesis Steps

1. Reading the target library setup, using a command script. This command script also disables the use of dynamic flip-flops.

2. Reading the design files, via command script for each design subdirectory. The command scripts run external `make` commands to run the Icarus Verilog preprocessor on the design files. The `Makefile` takes care of the required preprocessor directives. The command scripts perform all necessary design flattening and uniquification of the sub-designs.

3. The two clock inputs are defined.

4. Input timing constraints are defined. Configuration option inputs are false paths, serial inputs are defined with 100 ns setup time to the 5 MHz clock, the TTC inputs with 15 ns setup time to the 40 MHz clock.

5. Single ended input PADS (IB1) are added to the design ports.

6. Because of deficiencies in the Synopsys library models, the use of half adder cells and d-flip-flops with asynchronous controls is suppressed. Due to fake favourable timing specs, Synopsys would instantiate these cells inappropriately.

7. Add a custom wire-load model to the core library, and specify the use of that model throughout the design. The used model is slightly less pessimistic than the `10k` model of the original library.

8. Compile, first pass.

9. Flatten all sub-design with less than 60 cells.

10. Specify worst case operating conditions. Fastest for minimum timing, slowest for maximum timing.

11. Set clock skew on both clocks to 100 ps and require hold-time fixing.

12. Second pass compile with boundary optimisation and high mapping effort.

13. Change all names in the design to be Cadence compatible.

14. Save the Verilog netlist for simulation.
15. Save the Verilog netlist for Place and Route. Append the clock buffer modules to the netlist file
16. Flatten the entire design and save it again.