SALT ver 3 chip documentation

Miroslaw Firlej, Tomasz Fiutowski, Marek Idzik,
Jakub Moron, Krzysztof Swientek
Szymon Bugiel, Roma Dasgupta, Marika Kuczynska

Created: January 2016

Last change: May 18, 2016

Contents

Documentation Conventions

To aid the readers understanding, a consistent formatting style has been used throughout this manual.

- Internal signals are written using *italic* font.
- External connections names (pads) like supplies use CAPITAL LETTERS only.
- External signals names, however, are in capital letters but using *ITALIC* font also.
- Configuration elements like register names are written in sans serif font.
- Signals controlled by configuration bits use *slanted sans serif* font.

For numbers the Verilog prefix style is used:

- **'b** for binary numbers e.g 'b1010,
- **'h** for hexadecimal numbers e.g 'hA7,
- **'d** for decimal numbers e.g 'd72,
- no prefix means that the number is in decimal notation.

There are also still some C style prefixes **0x** for hexadecimal numbers, especially in I^{2}C figures.

1. ASIC overview

Silicon strip detectors in the upgraded Tracker of LHCb experiment requires a new readout Application Specific Circuit (ASIC) . A project of a 128-channel ASIC called SALT (**S**ilicon **A**SIC for **L**HCb **T**racking), has been started. A block diagram of the first 128-channel SALT prototype (called just SALT) is shown in figure 1. SALT extracts and digitises analogue signals from the sensor, performs the digital processing and transmits the serial output data. It is designed in TSMC CMOS 130 nm technology, and uses a novel architecture comprising an analog front-end and an ultra-low power ($< 0.5\ mW$) fast (40 MSps) sampling 6-bit ADC in each channel.

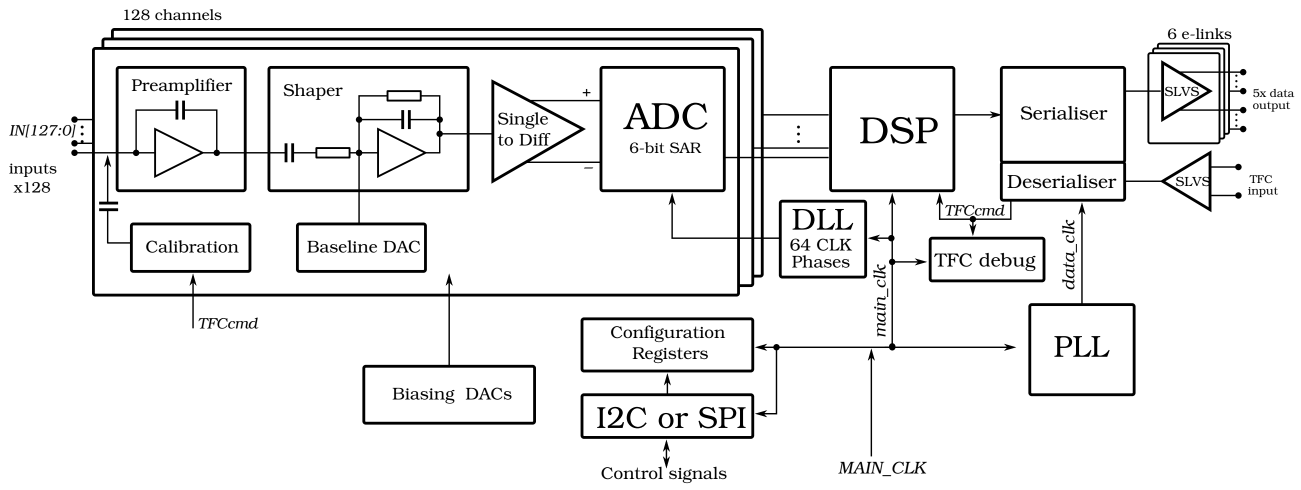

Figure 1: Block diagram of 128-channel SALT version

The front-end comprises a charge preamplifier and a fast shaper ($T_{peak} = 25ns$ and fast recovery) required to distinguish between the LHC bunch crossings at 40 MHz. To achieve this a specific non-standard shaper is required. The front-end should work with sensor capacitances between 5–20 pF. In the last stage of analog front-end a single-to-differential block converts a single-ended signal to a differential one. A fully differential 6-bit SAR ADC running at 40 MHz converts the analog signal to the digital domain. In order to achieve highest speed and lowest power consumption the SAR logic is asynchronous and dynamic circuitry is used in the ADC logic and comparator. To synchronize ADC sampling instance with beam collisions a dedicated ultra-low power ($< 1\ mW$) DLL is used to shift and align an external clock. The digital ADC output is processed by a Digital Signal Processing (DSP) block, which performs a pedestal subtraction, a mean common mode subtraction and a zero suppression. In following step the data packets are created and recorded in a local memory. After the DSP the data are serialized and the data rate is multiplied eight times. This is obtained by increasing the clock frequency four times (to 160 MHz) and by using a double data rate (DDR) transmission. An ultra-low power ($< 1\ mW$) Phase Locked-Loop (PLL) is used to generate the 160 MHz clock from the 40 MHz system clock. The data is sent out by the SLVS interface. The

ASIC is controlled via the LHCb common protocol consisting of two interfaces: the Timing and Fast Control (TFC) and the Experiment Control System (ECS) [1, 2]. The TFC interface delivers the 40 MHz clock and other crucial information and commands, synchronised with the experiment clock, while the ECS serves to configure and monitor the ASIC and it is realised through the Inter-Integrated Circuit (I^2C) interface.

The specifications and the functionality of the full 128-channel SALT ASIC are almost the same as the previous 8-channel SALT ASICs. The differences are caused either by different floorplan of these chips, or by using not a final version of certain blocks in the 128-channel prototype of Silicon ASIC for LHCb Tracking (SALT) ASIC, or by the modifications requested after the SALT ASIC tests.

The main specifications of the SALT ASIC are shown in Table 1.

Table 1: Summary of the specifications of the SALT ASIC.

Variable	Specification
Technology	TSMC CMOS 130 nm
Channels per ASIC	128
Input / Output pitch	80 μm / 140 μm
Total power dissipation	$< 768\ mW$
Radiation hardness	30 Mrad
Sensor input capacitance	5 – 20 pF
Noise	$\sim$1000 e^-@10 pF + 50e^-/pF
Maximum cross-talk	Less than 5% between channels
Signal polarity	Both electron and hole collection
Dynamic range	Input charge up to $\sim$30 000 e^-
Linearity	Within 5% over dynamic range
Pulse shape and tail	T_{peak} $\sim$25 ns, tail after 2$\times T_{peak}$ $\sim$5% amplitude
Gain uniformity	Uniformity across channels within $\sim$5%
ADC bits	6 bits (5 bits for each polarity)
ADC sampling rate	40 MHz
DSP functions	Pedestal and common mode subtraction, zero suppression
Output formats	Non-zero suppressed, zero suppressed
Calibration modes	Analogue test pulses, digital data loading
Output serialiser	Three serial e-links, at 320 MBit/s
Slow controls interface	I^2C
Digital signals, data, TFC, clock interface	Differential, SLVS

2. Analogue front-end

The analogue front-end has to be very fast with a peaking time $\leq$ 25 ns, should have a very short tail to minimise the pile-up and spill-over into the next bunch crossing, and should also have very low power consumption (1–2 mW/channel). It should work with different strip sensors (capacitance range 5–20 pF), with input signal of both polarities and with good enough signal to noise ratio (S/N>10), even in the worst operation conditions. One of the main challenges for the analogue block is to obtain a very short signal duration with a minimum possible power consumption. Preliminary studies showed that this is not possible with a standard semi-Gaussian shaping (with real poles in the transfer function), but that a more complex shaping (using complex poles and zeros in the transfer function) is required. A simplified block diagram of the preamplifier and shaper fulfilling the requested requirements is shown in figure 2.

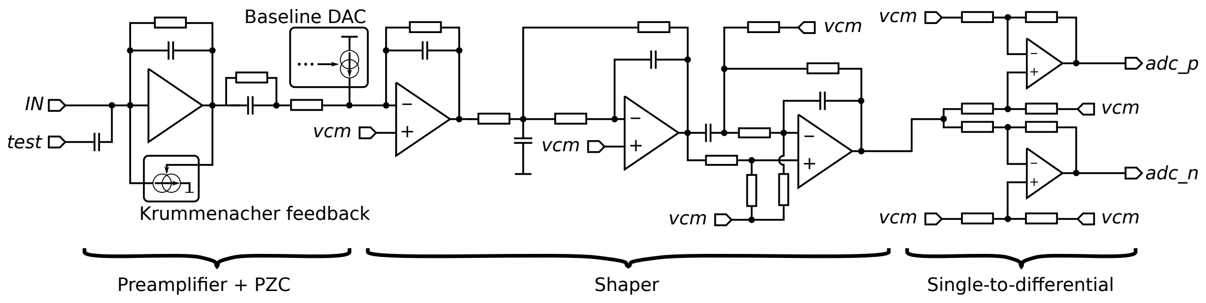

Figure 2: Simplified block diagram of preamplifier and shaper.

Each channel contains an 8-bit trimming DAC for a precise baseline setting. The *vcm* reference voltage should be set to half power supply. At the input of each channel there is a small (100 fF) test capacitance. The inputs of every second capacitance are shorted together so the injection of test pulse can be done either to all odd or even channels through a two separate test pads. The single-ended output is sent to a single-to-differential converter translating a single-ended signal to differential. In the Krummenacher circuitry and baseline DAC the ELT NMOS transistors were implemented in current mirrors working with very small currents.

The analog front-end has four common parameters which are adjusted through internal DACs. These are: preamplifier bias current I_{pre}; preamplifier feedback bias current I_{krum}; shaper bias current I_{sh}; and single-to-differential converter bias current I_{s2d}. The configuration of these DACs, together with eight DACs for independent setting of each channel baseline, is described later in table 14.

3. Analog to Digital Converter

According to the SALT specifications a 6-bit resolution ADC with 40 MHz sampling rate is needed for the signal digitisation. One of the most important constraints of the ADC design is a very low power consumption, significantly below 1 mW. This requirement has naturally led to a SAR ADC architecture. In order to further reduce the dissipated power, a Merged Capacitor Switching (MCS) scheme [3] was chosen. Since in the given technology the best capacitance matching is offered by a Metal-Insulator-Metal (MIM) capacitors, which are relatively large, a split capacitor Digital to Analogue Converter (DAC) approach was used, as shown in Fig. 3. In order to increase the linearity of

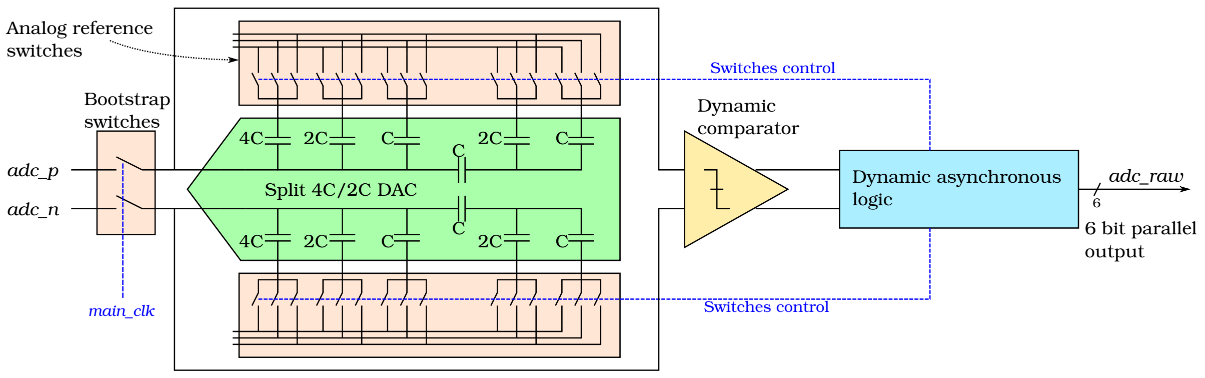

Figure 3: Block diagram of the 6-bit SAR ADC

the ADC the input switches are bootstrapped, what reduces significantly their dynamic resistance. In addition to the DAC switching scheme two other features are implemented to reduce the power consumption: a dynamic comparator and an asynchronous control logic, as shown in Fig. 3. The dynamic comparator dissipates power only during the bit cycling process, while the main power saving in the asynchronous control logic comes from the fact that the fast clock signal for the conversion of subsequent bits is eliminated. Finally, some more power is saved by implementing part of the logic with a dynamic flip-flops. To improve the ADC robustness against SEE events the ADC is reset by the sampling signal if the previous conversion is not yet completed. In practice it means that for SEE event two ADC conversions are lost (one caused by SEE and next when the reset is done), but after that the Analog to Digital Converter (ADC) is ready for the next conversion.;

The ADC has two 3-bit registers which are internally controlled. These are: ADC resolution *inactive_bits* which allows to decrease the resolution from the default 6 bits down to a 1-bit comparator in an extreme case; setling delay *set_del*, which allows to modify the internal delay between the bit cycling and so to tune the ADC performance. The configuration of these registers is described later in table 14.

4. Generation of common mode voltages

Both the front-end and ADC need several common mode voltage signals. All these signals generate the same voltage value of about 0.6 V. Most of them use as input a reference voltage from an internal bandgap reference circuit. The bandgap circuit generates a stable 0.6 V reference. Because the common mode signals have critical influence on the quality of signal processing a special block with four different common signals generators was designed:

- signal VCMA - delivers common mode voltage to the first two stages of the shaper (see fig. 2). It is generated as a buffered signal from the bandgap circuit. Since it has a pure capacitive load a single stage high-gain amplifier configured as a buffer is used.

- signal VCMB - delivers common mode voltage to the third stage of the shaper (see fig. 2). It is generated as a buffered signal from the bandgap circuit. Since it has resistive load of the order of 100 Ohm a two-stage high-gain amplifier configured as a buffer is used.

- signal VCMC - delivers common mode voltage to the single-to-differential converter (see fig. 2). It is generated as a buffered signal from the bandgap circuit. Since it has resistive load of the order of 100 Ohm a two-stage high-gain amplifier configured as a buffer is used.

- signal VCMD - delivers common mode voltage to the ADC (see fig. 3). Since this signal is very "dirty" and its precision is not important it is generaterd separately by resistive divider from VDDD power supply and then it is buffered by a single stage high-gain amplifier (since the load is capacitive).

4.1. Bandgap reference and temperature sensor

The bandgap based voltage references and temperature sensors (BGP) circuit is needed as standard block delivering constant reference voltage in various parts of readout system. Since temperature sensor is a part of reference voltage circuitry and temperature measurement is very useful itself, also temperature output has been added to the BGP block. In order to achieve good radiation hardness in the developed circuit a PMOS enclosed layout transistor (ELT) is used as a diode.

5. Clock generation

As already mentioned the internal 40 MHz clock is obtained by a precise shift/alignment, with a dedicated internal DLL block, of the external clock. A fast 160 MHz clock for data serialisation and transmission is obtained by the multiplication the 40 MHz clock by a dedicated internal PLL block. Both blocks are described in the following sections.

5.1. Delay Locked-Loop (DLL)

Figure 4 shows the block diagram of the DLL, which was designed to adjust the input clock phase. The VCDL contains 64 delay cells, so the same number of independent clock phases is available at the DLL output. The circuit is also equipped with an internal multiplexer, which provides the clock phase selection (1 from 64 phases). The VCDL delay can be controlled by changing the bias current via an internal 7-bit DAC, which allows to avoid current fluctuations, introduced from outside. The CP current can be configured in similar way via another DAC. The DLL contains also the output clock multiplexer, which provides a clock source selection. When the *dll_connect* is set

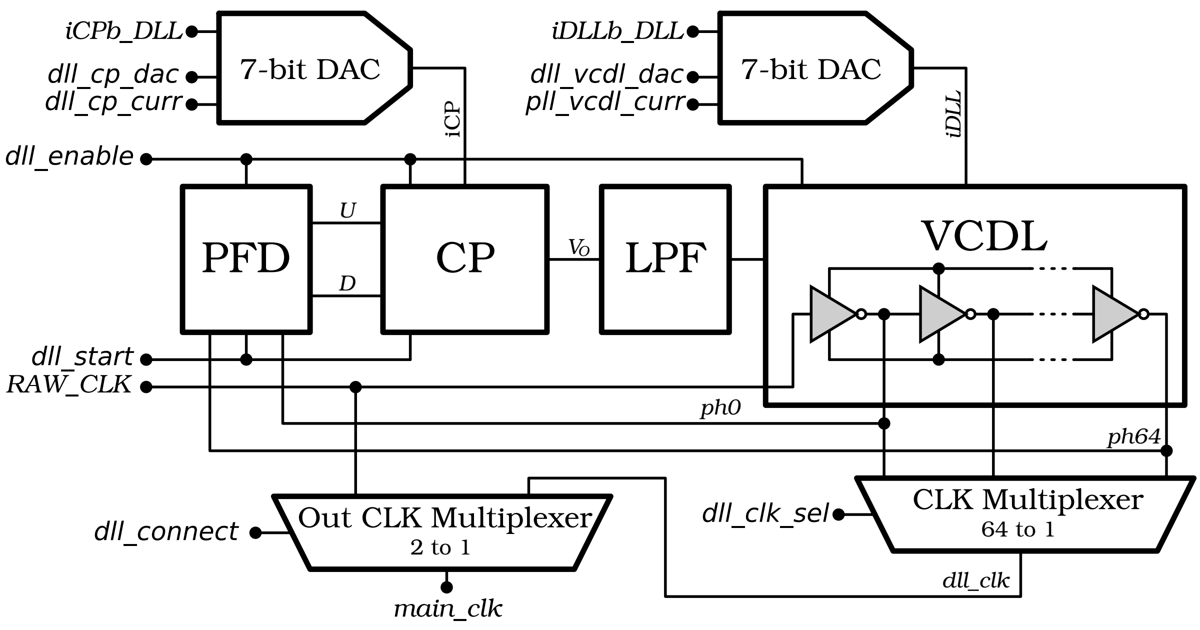

Figure 4: Delay Locked-Loop block diagram

to 0, the input clock is connected directly to the output (bypass). The selection of the clock phase is possible only when *dll_connect* is set to 1. This functionality is created to avoid randomly changed clock phase on the DLL output during the synchronization process, which may be disadvantageous for the digital logic.

When *dll_enable* is 0, the circuit is disabled. When *dll_enable* is 1 the DLL is on, but the synchronization of the circuit requires special startup procedure. When the DLL is enabled and *dll_start* is set to 0, the circuit is in standby mode. This mode should last at least 10 μs and after that time a high state on *dll_start* starts the DLL synchronization process.

5.1.1. DLL configuration procedure

1. Before starting DLL configuration make sure that the PLL works correctly and you are able to measure the delay between reference clock (40MHz) and output PLL clock (160MHz).

2. The dll_cp_cfg command sets the CP current. Its value should be: 'b10011010, and should not be changed.

3. The VCDL current, which can be configured by dll_vcdl_cfg command, is very important. Its default value is: 'b10011010, and it should be fine in most of cases. For such setting the default delay of all 64 DLL stages should be around 25ns (for 40MHz reference). It can be adjusted changing 7-bit dll_vcdl_cur field of the dll_vcdl_cfg command (higher values give shorter delays and vice versa) if needed.

4. Set the *dll_start* bit of dll_main_cfg command to 0 (DLL in idle state) and *dll_connect* bit to 0 (DLL skipped)
5. Wait at least 20us (probably slow control interface provides this delay)
6. Set the *dll_start* bit of dll_main_cfg command to 1 (DLL starts synchronization)
7. Set the *dll_connect* bit of dll_main_cfg command to 1 (DLL connected)
8. Set the clock delay (by dll_clk_cfg command) to 20 ('b010100) and measure delay Td1 between reference clock and output PLL clock
9. Set the clock delay to 30 ('b011110) and measure delay Td2 between reference clock and output PLL clock
10. Calculate Td2 – Td1, which is equivalent to delay of 10 VCDL stages. Delay between two consecutive clock phases should be around 390ps (25ns/64 phases) so the measured value of Td2 – Td1 should be around 3.9ns (at 40MHz reference). If Td2 – Td1 gives much smaller or much higher values then 3.9ns, the VCDL current (using dll_vcdl_cfg command) should be changed according to step 2.

General DLL remarks:

- Please note that any change of VCDL current by dll_vcdl_cfg command requires resynchronization of the DLL, which is done performing the steps 3-6.
- To verify proper DLL operation the reference frequency may be slightly changed, for example to 41MHz or 39MHz. The measured delay should change according to equation: Td2-Td1 [ns] = 10000/(Fref[MHz] * 64)

5.2. Phase Locked-Loop (PLL)

Figure 5 shows the PLL , which is used for clock multiplication in the data serialization circuit. The architecture of the circuit is typical and commonly described in literature, but few improvements were implemented. The DAC circuits were added for the better control of bias currents. The Voltage Controlled Oscillator (VCO) provides the gain selection (1 of 4), which can by useful for the jitter optimization (if necessary). This feature was added to make the PLL a general purpose block, although in the SALT a multiplication by four will be always used. The PLL works in frequency range 80 MHz – 400 MHz and is characterized by very low power consumption (1.0 mW@320 MHz). The VCO operates with 4 different gains. The center frequency is controlled directly by an internal 7-bit DAC, which allows to avoid the bias current fluctuations, introduced from outside (external current bias iVCOb_PLL are not available in SALT chip). The PLL works with four division factors (2, 4, 6 and 8) and generates 16 clock phases at their outputs. Two internal multiplexers provide a clock phase selection (2 from 16 phases).

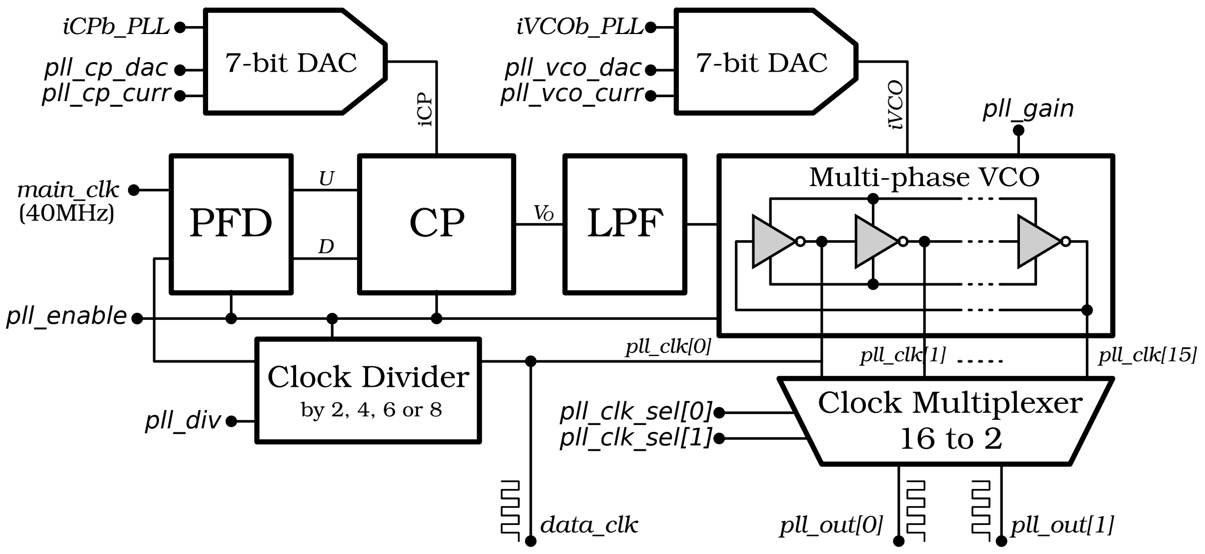

Figure 5: Phase Locked-Loop block diagram

5.2.1. PLL configuration procedure

1. Make sure that PLL is enabled and gain and divider are configured by pll_main_cfg command, which for SALT is: ’b10001101. It sets divider = 4 and maximum VCO gain.

2. The pll_cp_cfg command allows to set CP current. Its default value should be: ’b10011010, and should not be changed.

3. The pll_vco_cfg command allows to change the VCO bias current which is directly related to the output frequency. Default current value is: ’b10011010 – optimum for SALT (160MHz output, 40MHz reference, and divider 4). It can be changed when PLL has synchronization problems (output frequency too low or too high). If output frequency is lower than 160MHz the pll_vco_cfg value should be increased and vice versa.

4. The best method to verify the proper PLL operation is to check PLL output frequency (scope) vs reference frequency. For reference frequencies 35MHz and 45MHz, the PLL should generate respectively 140MHz and 180MHz. It is a good practice to adjust the pll_vco_cfg in such a way, that PLL operates correctly in the range at least 35MHz – 45MHz, keeping 40MHz the center frequency.

6. Digital Signal Processing

As it was already mentioned the DSP circuitry performs a pedestal subtraction, a mean common mode subtraction, and a zero suppression on the 6-bit data-stream coming from the ADC. The ADC samples are a signed 6-bit numbers coded as 2's complements. For a better testability the DSP allows also to transmit a raw ADC data or various combinations of partially processed data. A block diagram of the ASIC with the DSP components included is presented in figure 6. In the following sub-sections the detailed

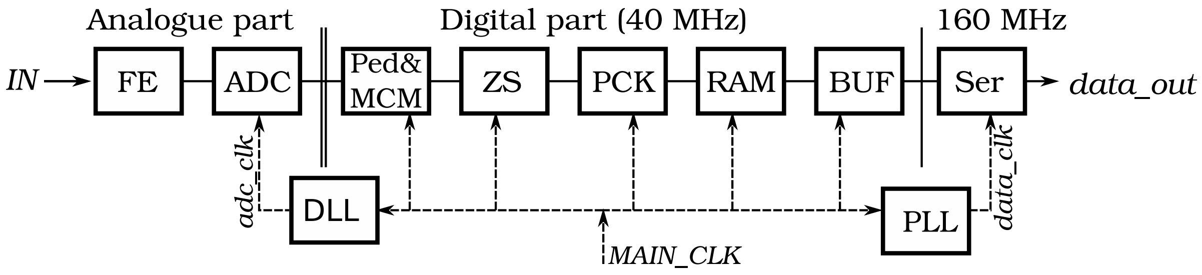

Figure 6: Data processing in the ASIC ; vertical lines separate clock domains; FE – analogue front-end, ADC – Analog to Digital Converter , Ped&MCM – Pedestal and Mean Common Mode Subtractions, ZS – Zero Suppression , PCK – Packet Formation, RAM – Main data memory, BUF – e-liks Buffers (Idle and Sync packets generation), Ser – Serialisers (just before e-link drivers), PLL – Phase Locked-Loop , DLL – Delay Locked-Loop .

information about the DSP processing is given.

6.1. Pedestal subtraction

The DSP processing starts from the pedestal subtraction, which block diagram is shown in figure 7. The global signals (same for all channels) *const_value* and *is_const* are for

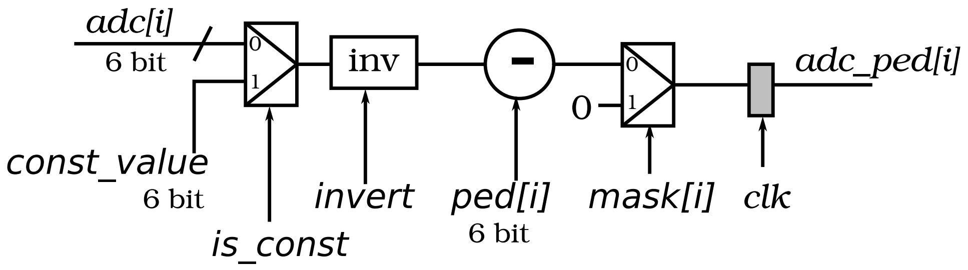

Figure 7: Pedestal subtraction block – identical in each channel; first multiplexer and signals *const_value* and *is_const* are for testing purposes and are inactive in during normal operations

testing purposes and in normal operation *is_const* should be zero. The signal *invert*, also global, turns on the arithmetic inversion of input data. It is necessary if ASIC receives the signal of negative polarity (for $n^+ - n$ silicon sensor), when the ADC sample value is expected to be negative. When the ASIC is connected to $p^+ - n$ sensor the signal *invert* is zero. This operation is shown in the formula below:

$$adc\_inv[i] = \begin{cases} adc[i] & \mathsf{invert} = 0 \\ -adc[i] & \mathsf{invert} = 1 \end{cases}, \tag{1}$$

where $adc[i]$ is input sample from ADC and i is the channel number from 0 to 7.

After the inversion a pedestal subtraction is performed. The data after pedestal subtraction is sent to the next block only if the signal *mask*, for a particular channel, is zero, as expressed in the formula below:

$$adc\_ped[i] = \begin{cases} 0 & \mathsf{mask}[i] = 1 \\ adc\_inv[i] - \mathsf{ped}[i] & \mathsf{mask}[i] = 0 \end{cases}. \tag{2}$$

The *mask* signal is intended to mask noisy or dead channels. In fact, there is one more multiplexer (not shown in the figure 7 but see figure 10), controlled by *mask*, at the input of the pedestal block, which helps to reduce the power consumption by avoiding not needed switching activity.

All the configuration signals used in this block are controlled by the configuration registers described in table 16 (page 61).

6.2. Mean Common Mode Subtraction

The idea of Mean Common Mode Subtraction (MCMS) block operation is to calculate an average value of all channels without a hit and subtract this value from all channels. In ideal case, when there are no disturbances and the pedestals from previous step are correct, the calculated average should be zero. We assume here that a possible disturbance is identical in each channel what, in general case, is an idealisation. A block diagram of the MCMS algorithm in SALT ASIC is shown in figure 8.

The channels with and without hit are distinguished by a comparison to a threshold. There are two thresholds in the MCMS block: *mcm_th* and *mcm_th2*, with different functions. The main threshold *mcms_th* selects the channels which should be used for the MCMS algorithm. The second threshold is to reject channels which values are extremely low – this idea comes from experience with the Beattle ASIC working presently in the LHCb experiment.

Before the subtraction can be applied the number of non active channels *mcm_channels* (channels without a hit) and the sum over non active channels have to be calculated. Both calculations are implemented as hierarchical pipelined structures and in both cases the operation takes two clock cycles. The next step is to calculate the average value . To perform this a division sub-block was designed. It is also implemented as a pipeline and the whole operation takes five clock cycles. As a result of the division the *mcm_value* signal is produced.

Mathematically the whole process can be described as:

$$mcm\_channels = \sum_{i=0}^{7} mcm\_ch[i], \tag{3}$$

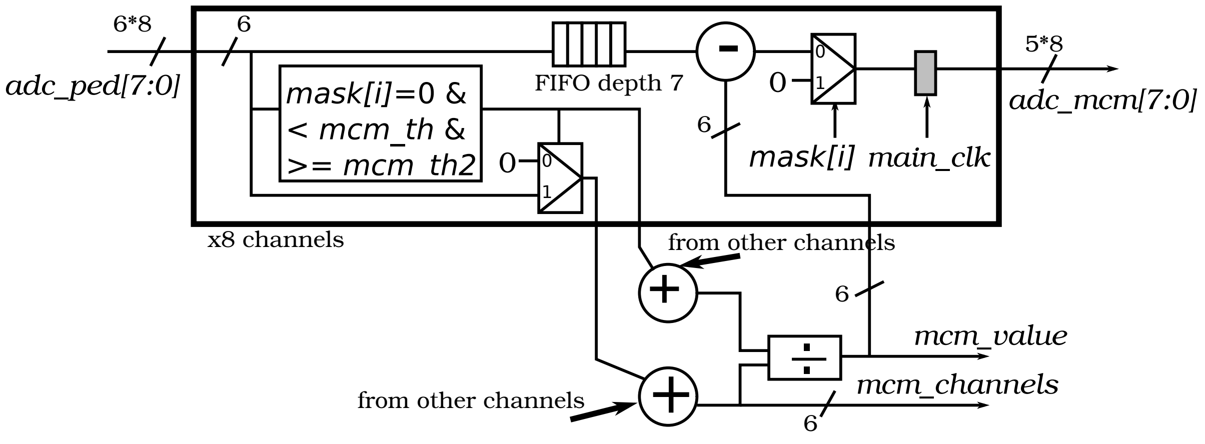

Figure 8: Mean Common Mode Subtraction block; elements in frame are repeated in each channel; elements outside the frame are single in the whole ASIC

where

$$mcm\_ch[i] = \begin{cases} 0 & \mathsf{mask}[i] = 1 \text{ or } \mathsf{mcm\_th2} > adc\_ped[i] \text{ or } adc\_ped[i] >= \mathsf{mcm\_th} \\ 1 & \mathsf{mask}[i] = 0 \text{ and } \mathsf{mcm\_th2} <= adc\_ped[i] < \mathsf{mcm\_th} \end{cases} . \quad (4)$$

The values $adc\_ped[i]$ are the results of pedestal subtraction (formula 2), while *mcm_th* and *mcm_th2* are the thresholds kept in configuration registers described in table 16.

A mean common mode value *mcm_value* (6-bit signed value) is evaluated as:

$$mcm\_value = \left[\frac{1}{mcm\_channels} \sum_{i=0}^{7} mcm\_ch[i] \cdot adc\_ped[i] \right]_{\text{round}} . \quad (5)$$

To keep the precision the result of division is rounded to the nearest integer instead of being truncated.

To have the ability to deactivate both thresholds the inequalities in formula 4 (and figure 8) are slightly different for *mcm_th* and *mcm_th2*. Deactivation of *mcm_th2* is possible by setting it to -2^5 (the smallest possible number), in such case non channel will be rejected by this threshold. The *mcm_th* also needs to be set to -2^5 for deactivation, and in such case there is not channel smaller than this value, and so the MCMS module is inactive since both *mcm_channels* and *mcm_value* are zero.

The very last step of MCMS is the subtraction and channels masking. To keep the data consistency there is a FIFO of depth 7 located just before the subtraction block, see figure 8. The depth of this FIFO is calculated as a sum of latencies of sum over channels and division operations. The masking have the same role as in pedestal subtraction block. As the result at the output of MCMS block the samples have following values:

$$adc\_mcm[i] = \begin{cases} 0 & \mathsf{mask}[i] = 1 \\ adc\_ped[i] - mcm\_value & \mathsf{mask}[i] = 0 \end{cases} , \quad (6)$$

where i is the channel number, and *mcm_value* is calculated with equation 5.

It is worth to mention that both *mcm_channels* and *mcm_value* values are sent to the packet building block to be used as a part of the Non Zero Suppression (NZS) packet.

6.3. Zero Suppression

The purpose of Zero Suppression (ZS) is to compress the output data by omitting the channels without hits and to send out only the channels with useful information (active channels). Active channels are selected by threshold *zs_th* which is set by one of the configuration registers (see table 16). Internally the ZS block, which simplified diagram

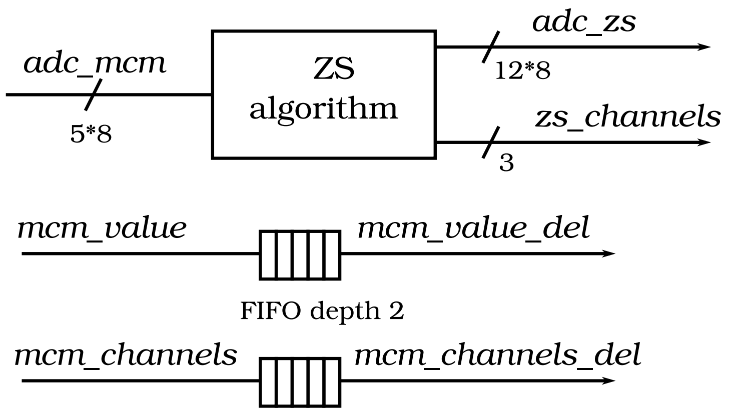

Figure 9: ZS block diagram

is shown in figure 9, is built as a large set of multiplexers forming a multibit output signal *adc_zs*. The internal structure is pipelined so the latency is constant and equal two.

The *adc_zs* signal is a 12-bit vector containing 7 bits for the channel number and 5 bits for the sample value. The structure is identical to the data field of normal packet shown later in figure 12. The *zs_channels* signal contains the number of valid elements in *adc_zs* vector. The *zs_channels* is calculated according to formula:

$$zs\_channels = \sum_{i=0}^{7} zs\_ch[i], \tag{7}$$

where

$$zs\_ch[i] = \begin{cases} 0 & adc\_mcm[i] < \textsf{zs\_th} \\ 1 & adc\_mcm[i] >= \textsf{zs\_th} \end{cases}. \tag{8}$$

It is clearly seen from formula 8 that if *zs_th* is zero the algorithm is inactive i.e. all sampled data go to output vector *adc_zs*, regardless of the value, and *zs_channels* is equal eight.

The *mcm_channels* and *mcm_value* signals are delayed by the ZS algorithm latency only to keep the data consistency.

6.4. NZS data

The NZS data purpose is to monitor/verify the DSP operation outside or in between the normal data taking. In such situations the NZS data packet replaces a normal data packet. The NZS packet is created on demand if ASIC receives the TFC NZS command. The data can be taken from four different places in the DSP chain as shown in figure 10. The default data source is the data after the first masking multiplexer.

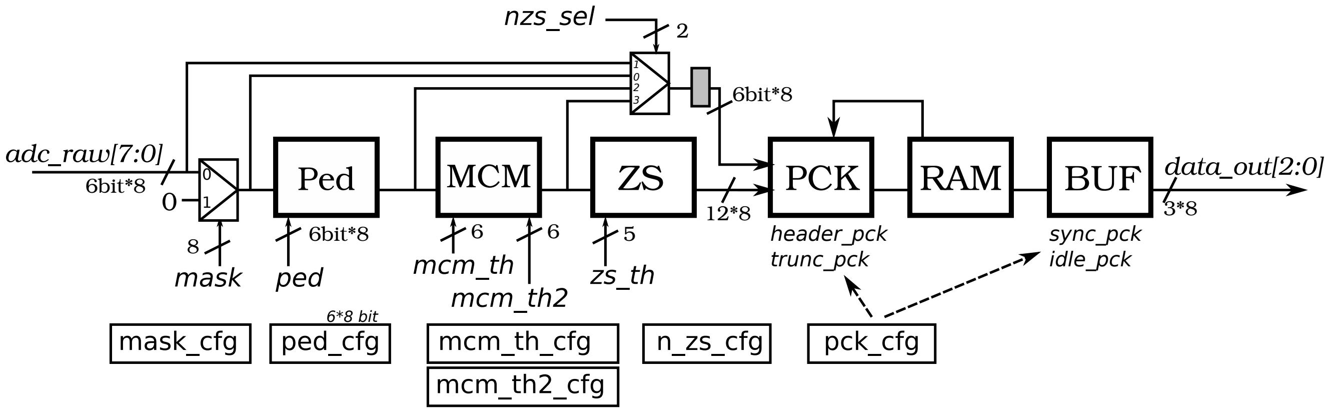

Figure 10: NZS data sources and DSP configuration registers

Because it is only a test/monitoring function there is no FIFO in parallel to the DSP chain but only one 48-bit register. This saves the power consumption but there is also a drawback, the TFC NZS command can't be sent more frequently than every 11 clock cycles. These 11 cycles correspond to the DSP pipeline depth: 1 for pedestals, 7 for MCMS , 2 for ZS plus 1 additional clock cycle. In other case the data would be overwritten before a data packet is built.

7. Back-end data processing

The back-end data processing (i.e. data processing after the core DSP chain) is directly related to the TFC commands. The effect of TFC commands on back-end data processing is shown in figure 11. To keep synchronisation of TFC commands with data processing a number of FIFOs is necessary with the depths matching the number of pipeline stages in particular blocks. The first FIFO in figure 11 is required by LHCb DAQ [2] and is implemented in the deserializer block (compare figure 31 in page 29).

The SALT ASIC should recognise eight TFC commands: Calib, Synch, Snapshot, BxVeto, NZS, Header, FEReset, and BXReset which interpretation is consistent with LHCb DAQ description [2]. Presently in the SALT the Calib and Snapshot are not implemented.

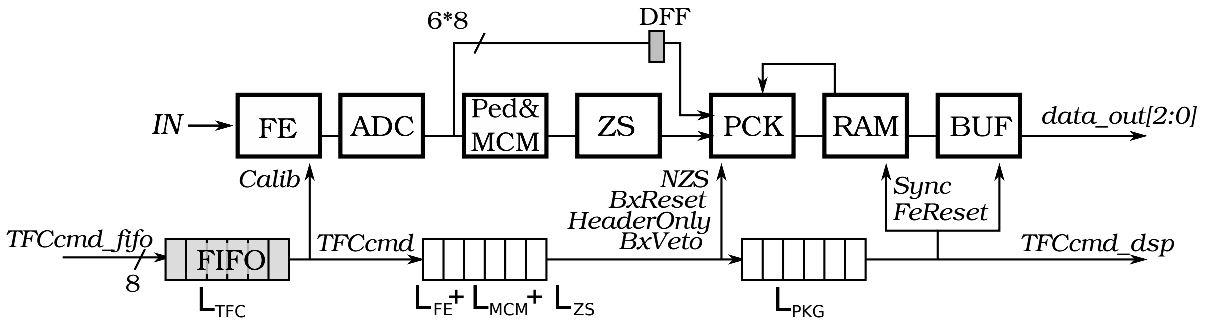

Figure 11: TFC commands control of back-end data processing

7.1. Packet building and recording

The Packet Building Block (PCK) shown in figure 10 receives two data streams the NZS and the ZS . The TFC commands control which of these streams, if any, is used to form the output data packet. The PCK block can create four types of data packets shown in table 2: Header, Normal, NZS and Truncated. The data packet type directly depends of the current TFC command.

If PCK receives Header or BxVeto TFC command the Header data packet is generated regardless of the data input. If there is neither Header nor BxVeto the Normal or the NZS data packet will be generated depending on whether the NZS is set or not in the current TFC command, however if there is no enough space in the memory the Truncated packet is sent to the memory. Other exception of this behaviour is the situation when the Normal packet is expected but there is no data to send (*zs_channels* is zero), in such a case the Header packet is put into memory. All cases of data packet creation are shown in table 3.

The PCK block contains the Bunch Crossing Identification number (BXID) , a 12-bit counter which counts clock cycles – the bunch crossing IDs. The last four bits of the counter are a part of header field in almost each data packet (see table 2). If the PCK block receive the BXReset TFC command the BXID counters is reset.

All data packets created in PCK block are sent to RAM memory (see figure 10). The size of memory is 128 bytes. If the memory if full it sends a feedback signal to PCK block and the Truncated packets are generated. The memory reacts for two TFC commands the Synch and FEReset. Both commands clear the memory.

7.2. Data stream buffering and splitting

The last data processing block, just before the serialisation, is called the BUF in figure 10. It receives the data from RAM and divides it to bytes (8 bit elements) which are sent directly to the e-link serialisers (*data_out[2]*, *data_out[1]*, *data_out[0]*). This block can add one or more Idle packet (see table 2) if there is no data to send. The Idle packets are sent out to keep the e-links active if there is no data. Also in this block the Sync packet is generated in answer to the Synch TFC command. The Sync packet contains a

	Header (8-bit)					Length (8bit)		Data
Packet name	BXID	NoData	IsTrunc	IsNZS	Parity	Len	Parity	
Header	4-bit	1	0	0	1-bit	—	—	—
Idle	0000	1	1	0	0	—	—	—
Normal	4-bit	0	0	0	1-bit	7-bit	1-bit	Hits
Truncated	4-bit	0	1	0	1-bit	7-bit	1-bit	—
NZS	4-bit	0	0	1	1-bit	1111111	1	Values
Sync	12-bit BXID + 12 configuration bits							

Table 2: Output data packet format. Data packet contains of three parts: Header, Length, and Data. All parts are byte aligned. In both packet parts: Header and Length the parity is even parity. Len field in Length part indicates number of hits or transmitted values reduced by 1. Data field contains ADC values in two forms: Hits or Values (see figures 12 and 13).

Normal Packet:

8 bits	8 bits	12 bits	12 bits		
Header	Length	Hit 0	Hit 1	...	Hit *Len*-1

Hit:

7 bits	5 bits
channel number	channel value after DSP

Figure 12: Normal packet structure. Hit part is repeated Len times according to Lenght part. Packet lenght in bits should be a multiplication of 8, so if there is odd number of hits two zero bits are added at the end.

NZS Packet:

8 bits	8 bits	8 bits	8 bits	6 bits	6 bits		
Header	Length	MCM	Non active	Ch 127	Ch 126	...	Ch 0

Ch:

6 bits
channel value (type selected by *nzs_cfg*)

MCM:

7 bits	1 bit
mcm_value	Parity

Non active:

7 bits	1 bit
mcm_channels	Parity

Figure 13: NZS packet structure: two control parameters and 128 channel values; for calculation of the control parameters see equation 5 and 3; for NZS channel type see figure 10 and *nzs_cfg* configuration bits.

Priority	Condition	Packet
0 (high)	Synch in *TFCcmd*	Sync
1	memSpace = memSize //empty memory	Idle
2	(Header in *TFCcmd*) or (BxVeto in *TFCcmd*)	Header
3a	(not NZS in *TFCcmd*) and ((memSpace < ⌈hitNum*3/2⌉+2) or (hitNum>=63))	Truncated
3b	(NZS in *TFCcmd*) and (spaceMem < 18)	Truncated
4	NZS in *TFCcmd*	NZS
5 (low)	—	Normal

Table 3: Data packet selection conditions in falling priority order. The condition is written in pseudo-code. Variable meaning are: TFCcmd – current set of TFC commands, memSpace – space left in memory in bytes, memSize – size of the memory, and hitNum – number of hits.

full BXID value and a constant value controlled by the configuration registers shown in table 16.

All data packets shown in table 2 have to be divided to bytes to be sent out through the e-link serializers, however the size of Data field in the Normal packet may not be a multiplier of eight, and in such case several zeros are added at the end of the packet to fill the last byte. In the present ASIC there are 3 e-links so 3 bytes are sent out in every main clock cycle.

Figure 14 (left side) shows an example data transmission splited to three e-links. It is assumed that only one data packed is in the buffer (RAM memory) with four consecutive bytes: header, length and two bytes of data (H, L, D1, and D0). Two additional Idle packets will be added at the end to keep all e-links active. The output *data_out[0]* is treated as the highest priority output i.e. if there is only a single byte to transmit (e.g. Header packet) it will be sent via this output and the others will be filled in by Idle packets. If there are two bytes the *data_out[0]* and *data_out[1]* will be used, etc..

Each e-link is completed with a differential Scalable Low-Voltage Signalling (SLVS) transmitter, therefore each output bit *data_out* is converted to a pair of differential signals *DDR_OUT_P* and *DDR_OUT_N* at the output of SLVS – see table **??** with pad list (page **??**).

The second part of figure 14 (right side) shows the transmission of Sync packet when the Synch TFC command was received twice. The Sync packet has always the size of three bytes. The first byte (Sync2) and the four most significant bits of the second (Sync1) contain 12 bits of BXID counter, while the four least significant bits of the second (Sync1) byte and the third byte (Sec0) are set by the configuration registers sync_cfg (see table 16).

Figure 15 shows more complicated example of packets transmission via 3 e-links. There are nine consecutive data packets numbered by lower index. It may be concluded from SALT operation description that Idle packet may appear only on outputs *data_out[1]*

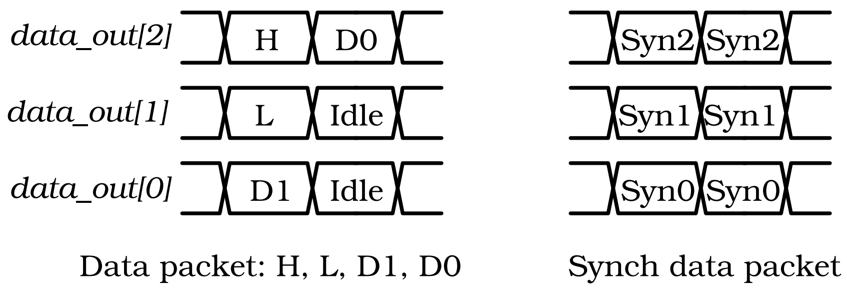

Figure 14: Data transmission example with three e-links: 4-byte data block (left) and two consecutive synchronisation patterns (right); each symbol is a single byte which correspond to one 40 MHz clock cycle.

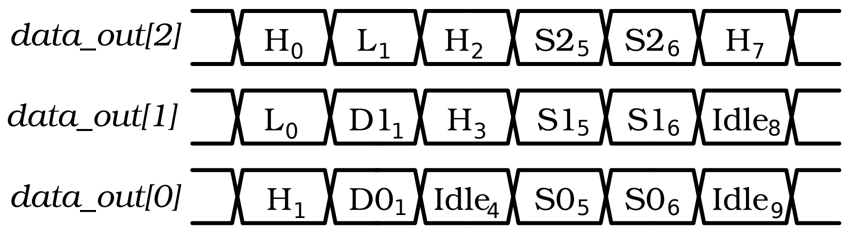

Figure 15: Data transmission example with three e-links; nine packets send numbered by lower index: Trunc, Data, 2xHeader, Idle, 2xSynch, Header, 2 x Idle; H – packet header, L – packet length, Dx – data elements, Idle – Idle packet, Sx – synch packet elements; each symbol is a single byte which correspond to one 40 MHz clock cycle.

and *data_out[2]* because in each main clock cycle at least 1-byte packet is build Header itself.

Figure 15 shows also more clearly the behaviour of Sync which always starts at *data_out[0]* and fulfil all active e-links. This packet may even interrupt ongoing transmission (not shown) to start at output *data_out[0]*.

8. TFC debug block

One of the very few digital functionalities required by LHCb DAQ system [2], which were not implemented in SALT ASIC are TFC debug registers: counters and their snapshots. They will be added in the SALT ASIC . The expected debug registers for TFC protocol are listed in tables: 4, 5, and 6.

9. Chip description

The SALT ASIC contains:

- Analogue and mixed-mode blocks:

RegName	Length	Snapshot	FeReset
CalibCnt	32	-	+
CalibCntSnap	32	+	-
SynchCnt	32	-	+
SynchCntSnap	32	+	-
SnapshotCnt	32	-	+
SnapshotCntSnap	32	+	-
BxVetoCnt	48	-	+
BxVetoCntSnap	48	+	-
NZSCnt	32	-	+
NZSCntSnap	32	+	-
HeaderCnt	48	-	+
HeaderCntSnap	48	+	-
FEResetCnt	32	-	-
FEResetCntSnap	32	+	-
BXResetCnt	32	-	+
BXResetCntSnap	32	+	-

Table 4: TFC counters (72 I^2C addresses)

RegName	Length	Snapshot	FeReset
BXIDCnt	12	-	-
BXIDSnap	12	+	-
MemFullCnt	32	-	+
MemFullCntSnap	32	+	-
MemEmptyCnt	32	-	+
MemEmptyCntSnap	32	+	-

Table 5: Other counters (20 I^2C addresses)

RegName	Length	Snapshot	FeReset
TFCCntOverflow	8	-	-
OtherCntOverflow	8	-	-

Table 6: Counter overflow registers (16 I^2C addresses); TFCCntOverflow – keeps overflow bits of TFC counters in positions defined in table 8; OtherCntOverflow – overflow bits shown in table 7

– Nine channels of analogue front-end (preamplifier and shaper). Channels [7:4] use the old front-end (from previous submission), while channels [3:0] use new modified front-end. The last channel, of a new modified type, is only for test

Bit	Counter name
7-3	—
2	*BXIDCnt*
1	*MemFullCnt*
0	*MemEmptyCnt*

Table 7: OtherCntOverflow register bits

purpose. Its input is floating but its test input is connected to TEST[1]. It has buffered output of each stage;

- Nine channels of single-to-differential converter. Channels [7:0] are connected to the outputs [7:0] of analogue front-end, and the last test channel is connected to the front-end test channel;
- Opamp-based buffers are used for test channel: *PRE_BUF* (preamplifier), *SH1_BUF* (1st shaper stage), *SH2_BUF* (2nd shaper stage), *SH_BUF* (shaper), *AP_BUF* (positive single-to diff), *AN_BUF* (negative single-to diff);
- Nine channels of 10-bit SAR ADC: channels [7:6] and [3:2] use 2-stage comparator, while channels [5:4] and [1:0] use 3-stage comparator. The last ADC channel is used only as the load for analogue test channel and its outputs are floating;
- PLL for fast clock generation for data serialization;
- DLL for alignment of sampling clock phase;

- Digital part comprising:
  - configuration registers read/write via I^{2}C interface;
  - serializer and deserializer for data output with some additional and functions: direct loop for configuration of communication and internal counters (ordinary and pseudorandom) for debugging;
  - DSP modules: pedestal subtraction, common mode, zero suppression;
  - packet building and buffering;
  - TFC debug counters.

- I/O interfaces:
  - SLVS differential drivers: data outputs (*DDR_OUT_P[2:0]*, *DDR_OUT_N[2:0]*), clock output (*DATA_CLK_OUT_P*, *DATA_CLK_OUT_N*);
  - SLVS differential receivers: clock input (*MAIN_CLK_P*, *MAIN_CLK_N*), TFC input (*DDR_TFC_P*, *DDR_TFC_N*)
  - I^{2}C signals: input *I2C_SCL*, data line *I2C_SDA*;
  - 3-bit I^{2}C chip address *I2C_ID[2:0]*;

10. I/O interfaces

The SALT ASIC communicates with external world with the aid of three different interfaces/protocols:

- A slow (< 1 MHz) I^2C read-write interface is used as digital control logic;
- A fast (320 Mb/s) write interface with DDR serializer and SLVS transmitter is used to send the data;
- A fast (320 Mb/s) read interface with DDR deserializer and SLVS receiver is used to read the TFC commands;

These interfaces are described in following sub-sections.

10.1. Slow I^2C interface

The SALT slow control Inter-Integrated Circuit (I^2C) interface needs a master 40 MHz clock to operate properly (see figure 1 on page 4). Both I^2C lines the *I2C_SCL* and *I2C_SDA* are treated in ASIC as ordinary digital signals, so the only real clock input in the ASIC is the master clock *MAIN_CLK*. The I^2C protocol was modified to achieve a maximum efficiency. The following examples show how to use the SALT slow control to obtain the desired functionality.

Since the internal registers address space is 12-bit long, therefore the 4 MSB address bits are transferred during the I^2C addressing phase while the following 8 bits are sent as the first byte of data (only during *write* operation). The 4 MSB address bits are sent between the 3-bit chip id and the last $R/\overline{W}$ bit (see fig. 16). These 4 bits are sent only during *write* operation i.e. when $R/\overline{W}$ is set to 0, otherwise they are ignored (see fig. 17). In the design an auto-incrementation feature of the address counter has been implemented – the address value is increased by one after sending/receiving each acknowledge bit.

The I^2C address is defined via inputs *ID*[2:0] bonded to ground or supply voltage. However, the ASIC recognise also a broadcast address 'b111. The broadcast is recognised simultaneously by all ASICs connected to common I^2C bus, so it should be used carefully e.g. reading is impossible.

A single register operations are shown in figures 16–18. In all these figures blue colour means sending by master, while red one means reading by master. As it is shown, it is possible to change the data transmission direction during one transfer (between start and stop conditions).

A multiple register writing and reading is also possible, thanks to the auto-incrementation feature. This however, requires caution as the 12-bit address is only set by the master during the write operation, using part of the first byte and the second byte. This means, that the data reading is started from the byte after the latest read/written. For example, the address was set to 'h002 and two bytes were written. Then the master sends stop and starts reading. The value sent to the master will be the register content from address 'h004. Figures 19–21 show the frames with multiple data transmission.

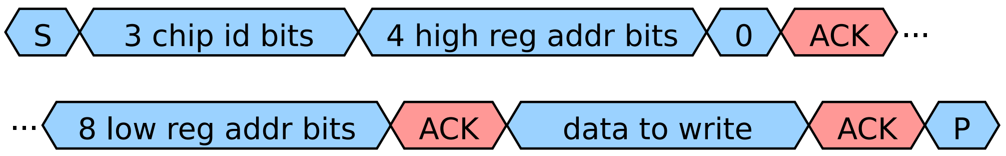

Figure 16: Single register write operation

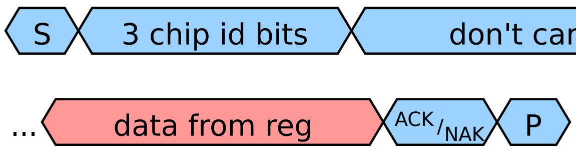

Figure 17: Single register read operation

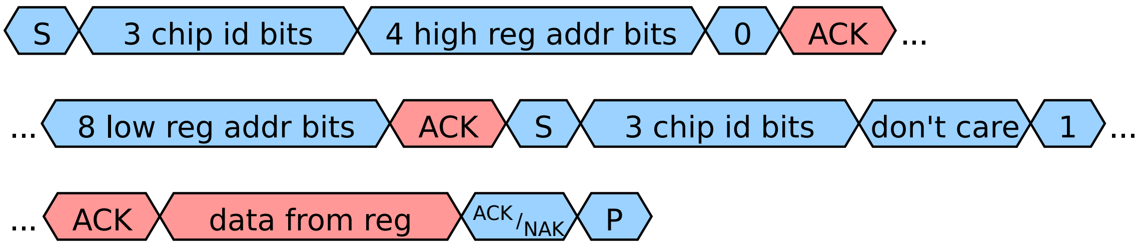

Figure 18: Single register mixed operation

Example 1 — writing Let's assume that the desired destination register address is 'h002 and the chip id is 'h3. Figure 22 is the way to save 'hFA value into the mentioned register. Figure 23 demonstrates how to write values 'hFA, 'h0B, 'hAF, 'hBC to registers 'h002–'h005.

Example 2 — reading Lets assume that the internal registers were updated like in example 1 and the internal address counter points to 'h002 again. If such situation occurs, a single read operation will end with the result as on figure 24 and the multiple read operation (with the same assumption) will result in the waveform like on figure 25.

Example 3 — reading with address setting The presented earlier reading example must be prefaced by a write operation. To avoid using two frames, the transmission direction can be switched within one packet. Figure 26 presents reading a value from 'h002 address without ending the transmission. Figure 27 shows reading multiple values from a consecutive registers starting address 'h002.

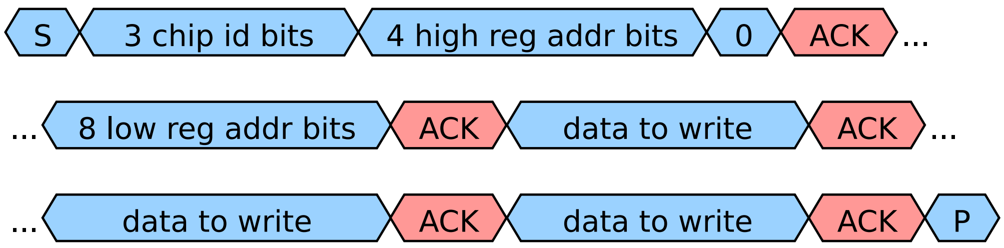

Figure 19: Multiple register write operation

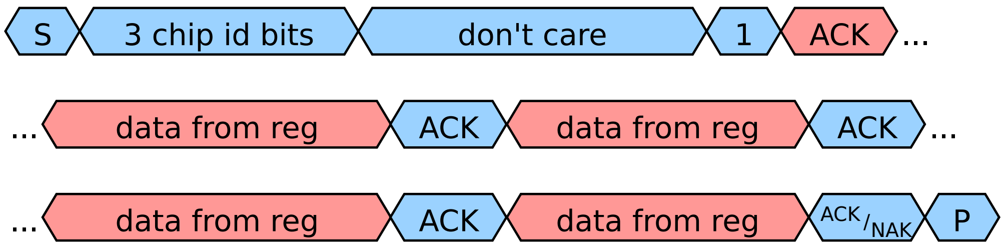

Figure 20: Multiple register read operation

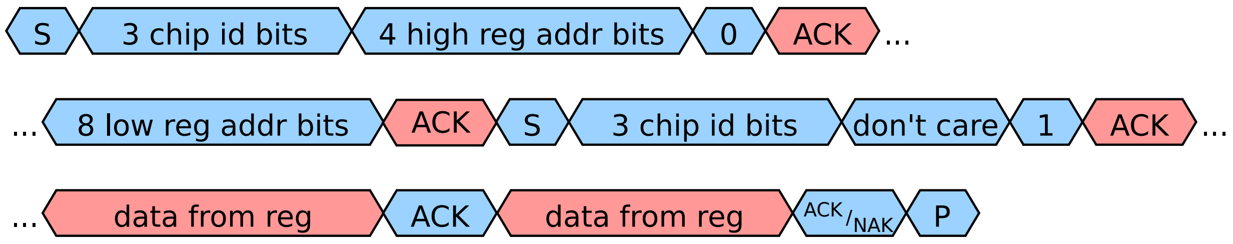

Figure 21: Multiple register mixed operation

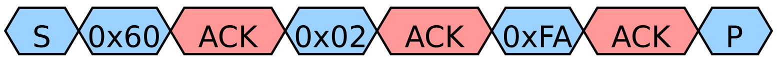

Figure 22: Saving ’hFA to ’h002 address on ’h3 chip

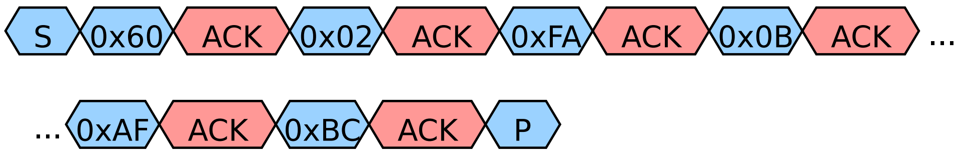

Figure 23: Saving multiple values to ’h002–’h005 addresses on ’h3 chip

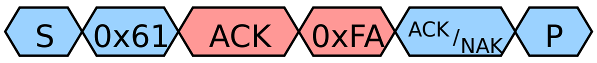

Figure 24: Reading ’h002 address on ’h3 chip

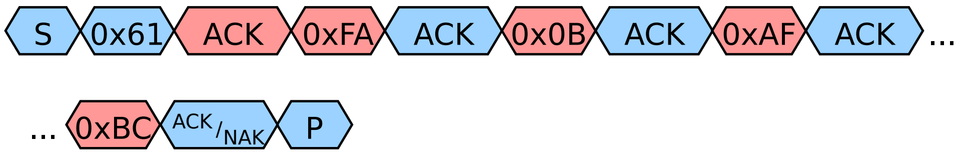

Figure 25: Reading multiple values from ’h002–’h005 addresses on ’h3 chip

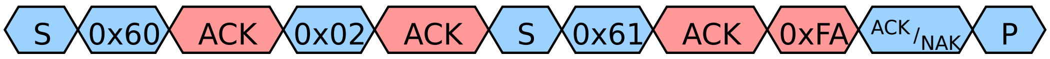

Figure 26: Reading ’h002 address on ’h3 chip without ending the transmission

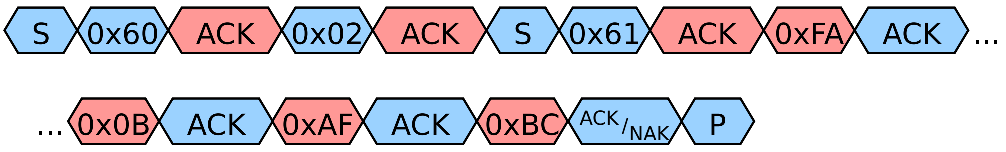

Figure 27: Reading multiple values from ’h002–’h005 addresses on ’h3 chip without ending the transmission

10.2. Fast data interface with DDR serializer

The base function of serializer circuit, shown in figure 28, is to serialise the data generated in DSP (input *data_out[i]*) and to send it out via SLVS interface. The serializer is also called the e-port. Although the data rate is 320 Mb/s, the internal clock frequency is 160 MHz to save power. The Data Double Rate (DDR) signaling scheme is used for this aim. Because the relation between the main clock *main_clk* and the output data rate is 8 the serializer sends a byte every main clock cycle. The bits are sent starting from MSb (bit 7) as shown in figure 29.

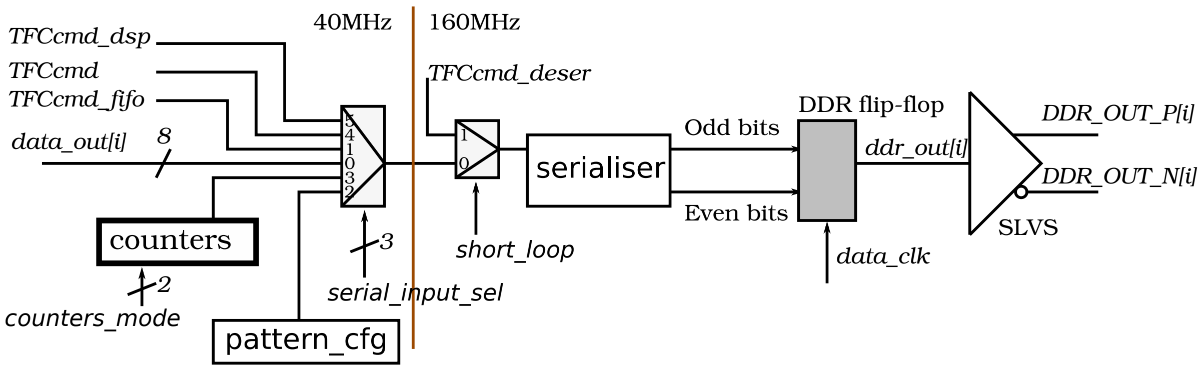

Figure 28: Serializer circuit with all possible data sources

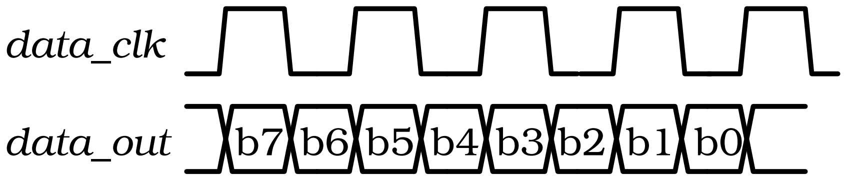

Figure 29: E-link byte transmission; MSb is first

There is no phase control circuitry in the serializer block, so the receiver (e.g. FPGA) is responsible for phase adjustment to receive the correct data. To facilitate this process there are two, alternative to *data_out[i]*, data sources with known data value or sequence: pattern register (constant value) or internal counters. There are four predefined counter sequences: counting up, counting down, and two pseudo-random sequences.

The data source selection and other settings of this block are controlled by the configuration registers described in table 17 (page 63). These settings are common to all three serializer blocks. Except the standard data (*data_out[i]*) sending mode all e-ports send out exactly the same data stream.

The other data sources are related to TFC commands and can be used for the TFC interface configuration or debugging.

10.3. Fast TFC interface with DDR deserializer

In contrast to many serializer there is only one deserializer circuit in the SALT . The deserializer, shown in figure 30, receives the TFC commands synchronous to main clock. As a result in each main clock cycle a different command is received and applied to DSP . The bit interpretation in deserializer, working also with DDR signaling scheme, is similar to serializer, so the first received bit is MSb (bit 7), as it was shown in figure 29 for the serializer. However, this interpretation is true only when the configuration of deserializer is correct.

It is required that the deserializer can adjust the phase of input DDR bit stream to receive the correct TFC commands. To cope with this, two phase synchronisation circuits, shown in figure 31, were added at the data input – one for each *data_clk* clock edge. The configuration procedure of deserializer requires an earlier configuration of correct serializer transmission. After that, the output of deserializer has to be sent to the serializer and only then the correctness of input data transmission is verified.

For the input flip-flop of the synchronisation circuit (figure 31) one of clock phases, generated by the PLL circuitry, can be selected by setting the configuration register described in table 17. Although the *pll_clk* signals are in constant phase shift to *data_clk* clock the final relation between the selected *pll_clk* signal and the data clock is not known. Because of that some hold or setup time violation is possible at the second flip-flop, working with *data_clk* clock. To avoid this there are two flip-flops, one working on rising and one on faling clock edge. The *clk_sel* chooses which one will be used.

A correctly synchronized bits don't end the configuration process of deserializer. The next step is to define the first bit of a byte. This parameter is controlled by the deser_cfg configuration register (see table 17 in page 63).

When the deserializer is properly configured each received byte contains a set of TFC commands which will control the DSP and readout operation. The meaning of the TFC command bits is presented in table 8.

To synchronize properly the TFC command stream with DAQ system a variable delay is needed in the ASIC . It was implemented as a FIFO, shown in figure 30 (TFC FIFO), and can be changed in range from 0 to 255 (see register tfc_fifo_cfg in table 17). The

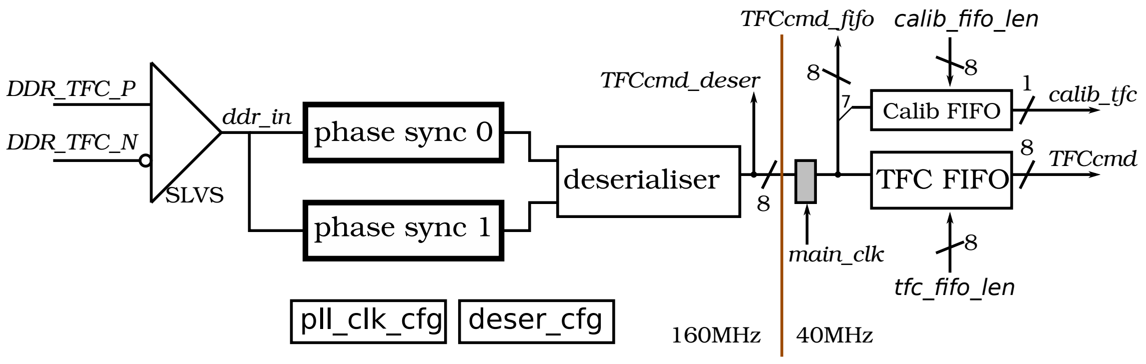

Figure 30: Deserializer circuit – TFC command interface

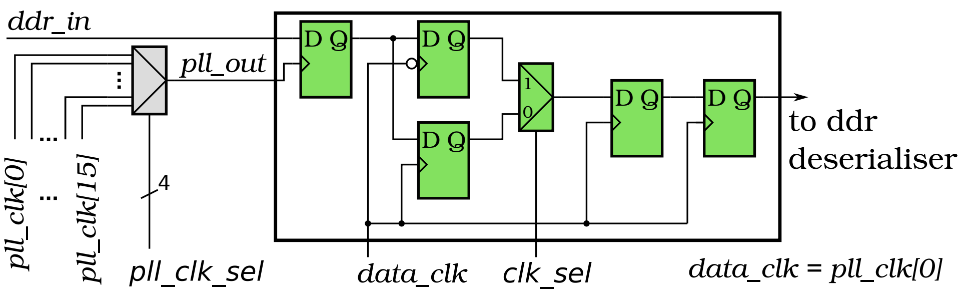

Figure 31: Deserializer – phase synchronizer circuit; each phase synchronizer (from figure 30) has its own external multiplexer for *pll_clk* phase clock selection

second FIFO (Calib FIFO) is intended for Calibration TFC command only. Calibration command *calib_tfc* is connected directly to calibration circuit described in section **??**. In is worth noting the Calibration TFC command is delayed independently in both TFC and Calib FIFOs. From the first one it is routed to TFC counters and DSP blocks while the second is connected only to Calib block.

bit	Name	Description
7	Calib	Run single calibration pulse (*not implemented*)
6	Synch	Empty data buffers and send Sync data packet instead of data
5	Snapshot	Rewrite selected counters to Snap registers
4	BxVeto	Equivalent to Header command
3	NZS	Omit zero suppression procedure
2	Header	Put only Header packet into data stream
1	FEReset	Reset TFC registers and empty data buffers
0	BXReset	Reset BXID counter

Table 8: Location of bits in TFC command word

11. Padring

Floorplan block diagram is shown of the SALT ASIC is shown in figure 32. Front (left in figure 32) pads have 80 μm pitch while back (digital) pads are placed in 140 μm pitch.

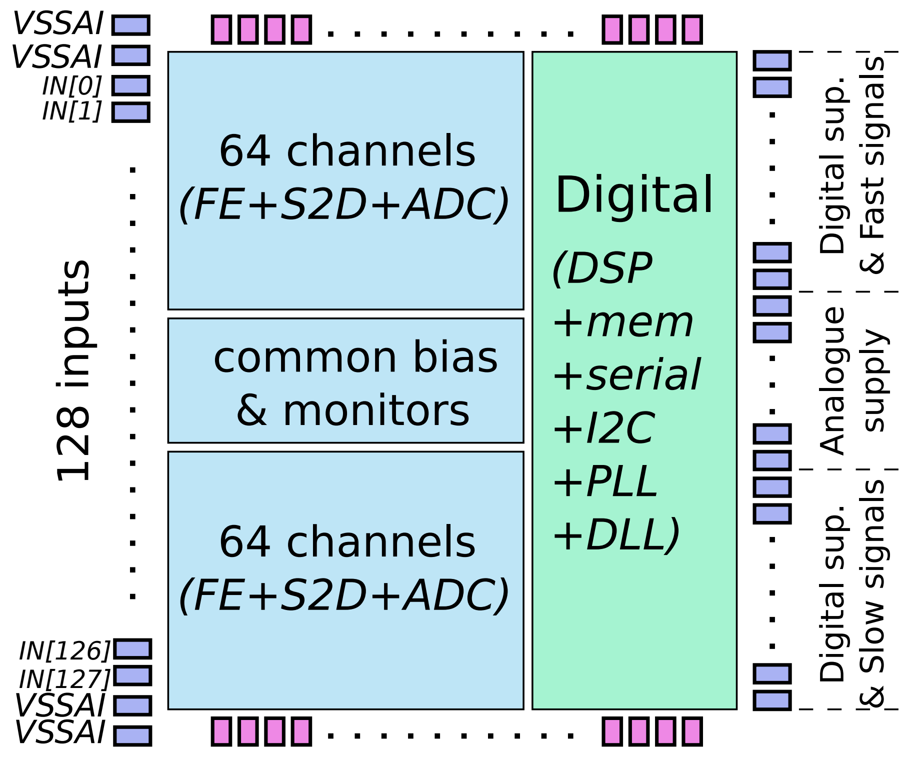

Figure 32: Floorplan of the SALT ASIC

The pad ring of the ASIC is shown in figures: 33 (input), 36 (back), 34 (top), and 35 (bottom). Input and back pads are longer then top and bottom pads. The former will be first used for wafer screening and thereafter bonded to hybrid, the latter are intended only for wafer screening.

Table 9: Pad list of the top side; all pads here are only for tests purpose; pads marked with (W) are intended for wafer screening (there is no power supply provided for the test pads buffers inside the ASIC – during tests all power supply pads have to be connected)

No	Name	Type	Description
		From LEFT	
1	*PRE_BUF*[0]	voltage out	Test channel[0]: buffered pre-amp output
2	*SH1_BUF*[0]	voltage out	Test channel[0]: buffered shaper 1st stage output

Continued on next page

Table 9 – continued from previous page

No	Name	Type	Description
3	*SH2_BUF*[0]	voltage out	Test channel[0]: buffered shaper 2nd stage output
4	*SH_BUF*[0]	voltage out	Test channel[0]: buffered shaper 3rd stage output
5	*AP_BUF*[0]	voltage out	Test channel[0]: buffered s2d positive output
6	*AN_BUF*[0]	voltage out	Test channel[0]: buffered s2d positive output
7	*I_BUF*[0]	current in	Test channel[0]: buffers biasing current (default 25uA from *VDDA_BUF*[0])
8	*VSSPST_BUF*[0]	ESD (W)	Test pads ESD ground
9	*VSSA_BUF*[0]	power	Test pads buffers ground
10	*VDDA_BUF*[0]	power+ESD (W)	Test pads buffers and ESD power
11	*VCMA*	voltage out (W)	Common mode voltage A test pad
12	*VCMB*	voltage out (W)	Common mode voltage B test pad
13	*VCMC*	voltage out (W)	Common mode voltage C test pad
14	*VCMD*	voltage out (W)	Common mode voltage D test pad
15	*V_PRE_DAC*	voltage out (W)	Pre-amp biasing DAC output
16	*V_KRUM_DAC*	voltage out (W)	Krum biasing DAC output
17	*V_SH_DAC*	voltage out (W)	Shaper biasing DAC output
18	*DATA_CLK_OUT_P*	SLVS out	Data clock output
19	*DATA_CLK_OUT_N*	SLVS out	Data clock output

Table 10: Pad list of the bottom side; all pads here are only for tests purpose; pads marked with (W) are intended for wafer screening (there is no power supply provided for the test pads buffers inside the ASIC – during tests all power supply pads have to be connected)

No	Name	Type	Description
		From LEFT	
1	*PRE_BUF*[1]	voltage out	Test channel[1]: buffered pre-amp output
2	*SH1_BUF*[1]	voltage out	Test channel[1]: buffered shaper 1st stage output
3	*SH2_BUF*[1]	voltage out	Test channel[1]: buffered shaper 2nd stage output
4	*SH_BUF*[1]	voltage out	Test channel[1]: buffered shaper 3rd stage output

Continued on next page

Table 10 – continued from previous page

No	Name	Type	Description
5	*AP_BUF*[1]	voltage out	Test channel[1]: buffered s2d positive output
6	*AN_BUF*[1]	voltage out	Test channel[1]: buffered s2d positive output
7	*I_BUF*[1]	current in	Test channel[1]: buffers biasing current (default 25uA from *VDDA_BUF*[1])
8	*VSSPST_BUF*[1]	ESD (W)	Test pads ESD ground
9	*VSSA_BUF*[1]	power	Test pads buffers ground
10	*VDDA_BUF*[1]	power+ESD (W)	Test pads buffers and ESD power
11	*ADC_INP*	voltage in	Test channel[1]: ADC positive input
12	*ADC_INN*	voltage in	Test channel[1]: ADC negative input
14	*V_S2D_DAC*	voltage out (W)	S2D biasing DAC output
15	*V_PULSE_DAC*	voltage out (W)	Calibration test pulse amplitude DAC output
16	*V_SLVS_VBIAS_DAC*	voltage out (W)	SLVS biasing DAC output
17	*V_SLVS_INREF_DAC*	voltage out (W)	SLVS biasing DAC output
18	*SCAN_ENABLE*	CMOS in (W)	Scan/normal mode selection (pulled down)
19	*SDI*	CMOS in (W)	Scan chain input
20	*SDO*	CMOS out (W)	Scan chain output
21	*TEST_MODE*	CMOS in (W)	Switches chip into test mode (pulled down)

Table 11: Pad list of the back side; pads marked with (W) are intended for wafer screening (there is no default value for any input signal provided inside the ASIC, all signal pads have to be connected)

No	Name	Type	Description
		From BOTTOM	
1	*VDD*	power	Digital supply
2	*VSS*	power	Digital ground
3	*VDD*	power	Digital supply
4	*VSS*	power	Digital ground
5	*VDDPST*	ESD	Digital ESD supply
6	*ID*[0]	CMOS in	I^{2}C address

Continued on next page

Table 11 – continued from previous page

No	Name	Type	Description
7	*ID*[1]	CMOS in	I^{2}C address
8	*ID*[2]	CMOS in	I^{2}C address
9	*I2C_SCL*	CMOS in	I^{2}C clock line
10	*I2C_SDA*	CMOS inout	I^{2}C data line
11	*RST_N*	CMOS in	Asynchronous reset
12	*MAIN_CLK_N*	SLVS in	Main clock input (40 MHz)
13	*MAIN_CLK_P*	SLVS in	Main clock input (40 MHz)
14	*VSSPST*	ESD	Digital ESD ground
15	*VDD*	power	Digital supply
16	*VSS*	power	Digital ground
17	*VDD*	power	Digital supply
18	*VSS*	power	Digital ground
	RING POWER CUT		
19	*VDDADC*	power	Analogue ADC supply
20	*VSSADC*	power	Analogue ADC ground
21	*VDDADC*	power	Analogue ADC supply
22	*VSSADC*	power	Analogue ADC ground
23	*VDDADC*	power	Analogue ADC supply
24	*VSSADC*	power	Analogue ADC ground
25	*VREFD*	power	Analogue ADC supply
26	*VSSADC*	power	Analogue ADC ground
27	*VREFD*	power	Analogue ADC supply
28	*VSSVCM*	power	Analogue VCM buffers ground
29	*VDDVCM*	power	Analogue VCM buffers supply
30	*VSSVCM*	power	Analogue VCM buffers ground
31	*VDDVCM*	power	Analogue VCM buffers supply
32	*VSSAPST*	ESD	Analogue ESD ground
33	*VDDAPST*	ESD	Analogue ESD supply
34	*VSSAPST*	ESD	Analogue ESD ground
35	*VDDAPST*	ESD	Analogue ESD supply
36	*VSSAPST_IN*	ESD	Analogue input pads ESD ground
37	*VDDAPST_IN*	ESD	Analogue input pads ESD supply
38	*VSSAPST_IN*	ESD	Analogue input pads ESD ground
39	*VDDAPST_IN*	ESD	Analogue input pads ESD supply
40	*VSSAI*	power	Analogue input transistor ground
41	*VDDA*	power	Analogue front-end supply
42	*VSSAI*	power	Analogue input transistor ground
43	*VDDA*	power	Analogue front-end supply
44	*VSSAI*	power	Analogue input transistor ground
45	*VDDA*	power	Analogue front-end supply

Continued on next page

Table 11 – continued from previous page

No	Name	Type	Description
46	*VSSA*	power	Analogue front-end ground
47	*VDDA*	power	Analogue front-end supply
48	*VSSA*	power	Analogue front-end ground
49	*VDDA*	power	Analogue front-end supply
50	*VSSA*	power	Analogue front-end ground
51	*VDDA*	power	Analogue front-end supply
RING POWER CUT			
52	*VSS*	power	Digital ground
53	*VDD*	power	Digital supply
54	*VSS*	power	Digital ground
55	*VDD*	power	Digital supply
56	*VSSPST*	ESD	Digital ESD ground
57	*DDR_TFC_N*	SLVS in	TFC command input negative
58	*DDR_TFC_P*	SLVS in	TFC command input positive
59	*DDR_OUT_N*[0]	SLVS out	Data output [0] negative
60	*DDR_OUT_P*[0]	SLVS out	Data output [0] positive
61	*DDR_OUT_N*[1]	SLVS out	Data output [1] negative
62	*DDR_OUT_P*[1]	SLVS out	Data output [1] positive
63	*DDR_OUT_N*[2]	SLVS out	Data output [2] negative
64	*DDR_OUT_P*[2]	SLVS out	Data output [2] positive
65	*DDR_OUT_N*[3]	SLVS out	Data output [3] negative
66	*DDR_OUT_P*[3]	SLVS out	Data output [3] positive
67	*DDR_OUT_N*[4]	SLVS out	Data output [4] negative
68	*DDR_OUT_P*[4]	SLVS out	Data output [4] positive
69	*DDR_OUT_N*[5]	SLVS out	Data output [5] negative
70	*DDR_OUT_P*[5]	SLVS out	Data output [5] positive
71	*VDDPST*	ESD	Digital ESD supply
72	*VSS*	power	Digital ground
73	*VDD*	power	Digital supply
74	*VSS*	power	Digital ground
75	*VDD*	power	Digital supply
TOP of the ASIC			

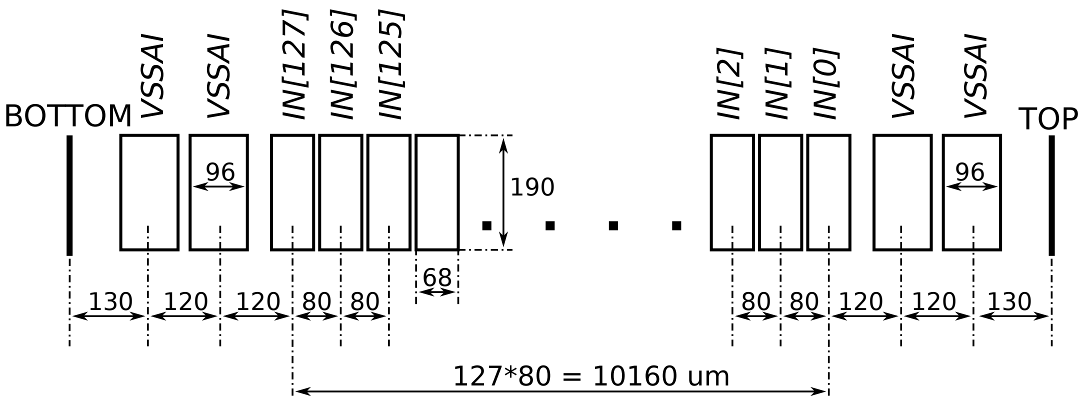

Figure 33: Front pads size and position

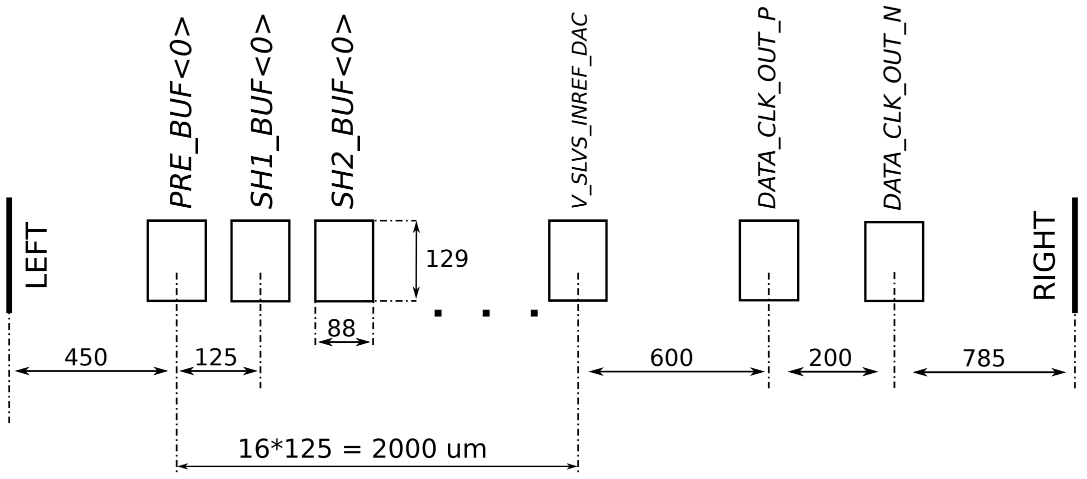

Figure 34: Top pads size and position

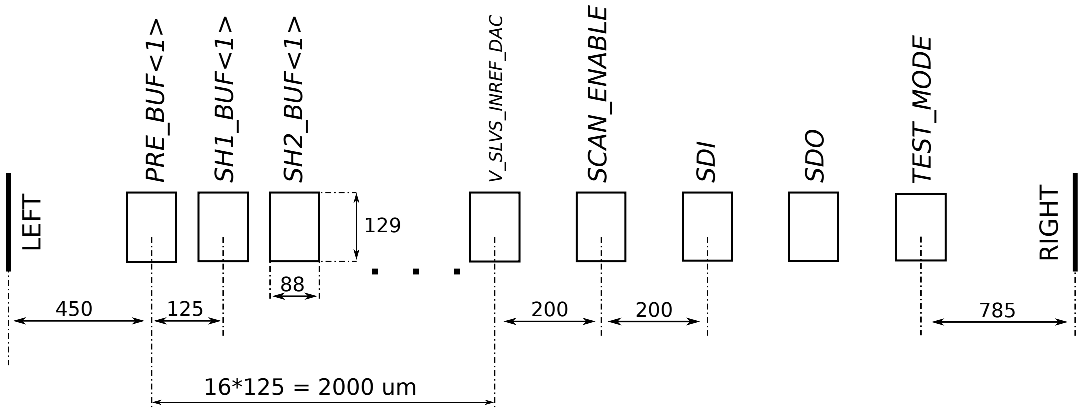

Figure 35: Bottom pads size and position

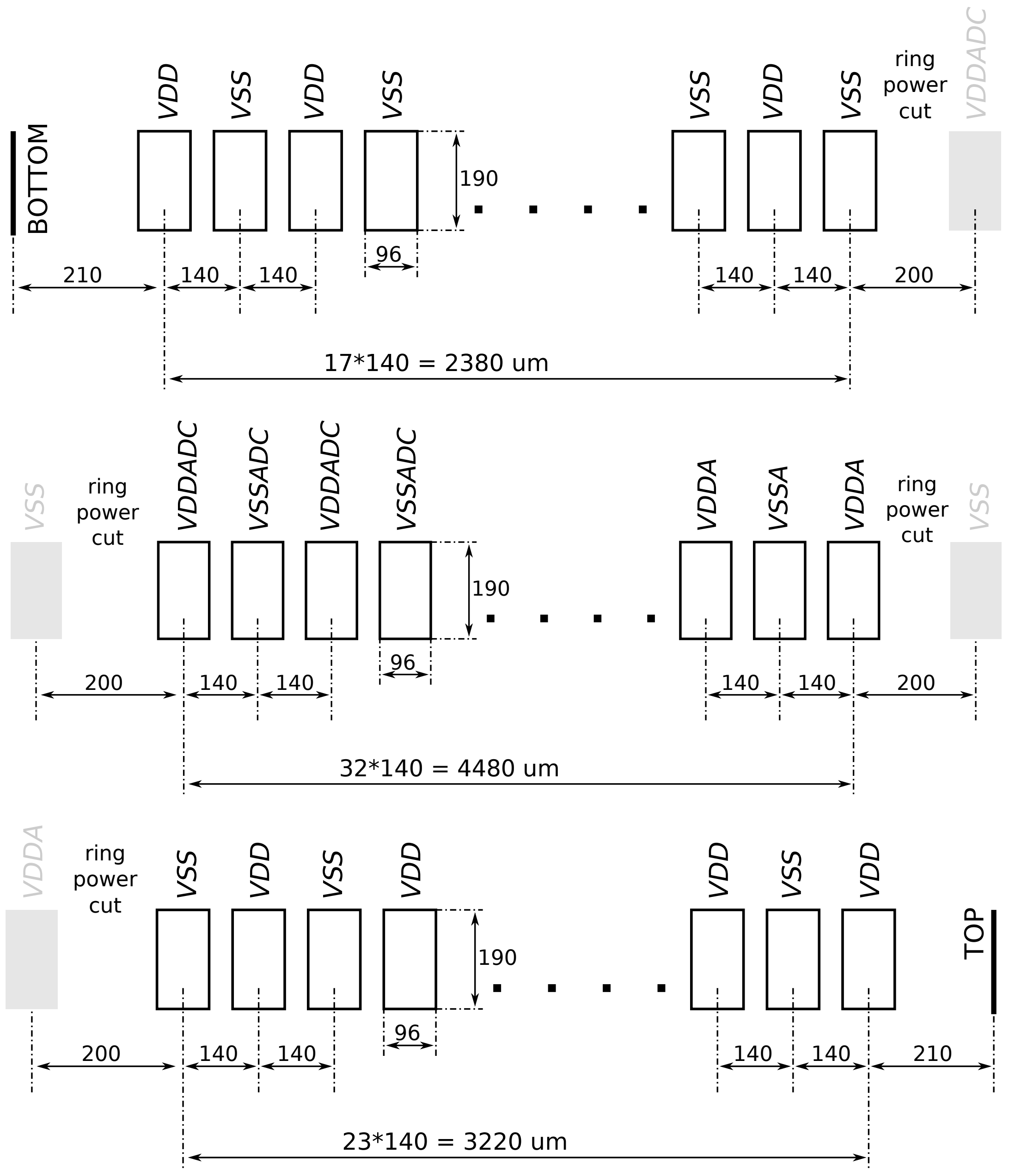

Figure 36: Back side pads size and position

12. Open issues, questions, proposals ...

12.1. DSP and back-end data processing modification

In the SALT the DSP block will be almost identical to the present one. The difference will be in memory (RAM) size, which should be about 3 kB, and in the maximum data size. It was already decide that in the ZS mode (Normal data packets) the largest data packet will contain no more then 63 channels. If the number of active channels will be larger then 63 then Truncated packet will be sent.

The data packet format will also be different (see figure 12) with 12-bit base element in contrast to 8 bits now. Having packet size being multiplication of 12 bits the serialisation procedure will be much more complicated because the e-link speed will stay at 320 Mbps.

Packet name	Header (12-bit)				Data	Comment
	BXID 4 bits	Parity 1 bit	Flag 1 bit	Length 6 bit	n·12 bits	
Idle	0000	1	1	'b11_0000	—	no enough data
BxVeto	BXID[3:0]	*	1	'b01_0001	—	BxVeto in TFCcmd
HeaderOnly	BXID[3:0]	*	1	'b01_0010	—	HeaderOnly in TFCcmd
BusyEvent	BXID[3:0]	*	1	'b01_0011	—	$nHits > 63$
BufferFull	BXID[3:0]	*	1	'b01_0100	—	no space in memory
BufferFullN	BXID[3:0]	*	1	'b01_0101	—	no space in memory
NZS	BXID[3:0]	*	1	'b00_0110	Values	NZS in TFCcmd
Normal	BXID[3:0]	*	0	$nHits$	Hits	Normal event
Sync	BXID[11:0]				pattern	fill one whole frame

Table 12: Output data packet format in SALT. Both parts Header and Data are 12-bit word aligned. The parity is calculated only for Header part as even parity. The Flag distinguish between special and Normal packet type. Length field contains number of hits for Normal packet and type for special packet. Data field contains ADC values in two forms: Hits or Values (see figures 12 and 13). Number of transmitted bits vary for Normal but is constant for special packets.

12.2. DSP modification propositions

In DSP algorithm, in each place were subtraction is performed a saturation arithmetic can be used. Saturation arithmetic is a version of arithmetic in which operations are limited to a fixed range between a minimum and maximum value, in SALT the range will be -32 to 31 which correspond to 6-bit signed number. If the result of an operation is greater than the maximum, it is set ("clamped") to the maximum; if it is below the minimum, it is clamped to the minimum.

In saturation arithmetic $adc\_ped[i]$ from equation 2 (page 13) will be calculated as

$$adc\_ped[i] = \begin{cases} 0 & \mathsf{mask}[i] = 1 \\ -32 & \mathsf{mask}[i] = 0 \text{ and } adc\_inv[i] - \mathsf{ped}[i] < -32 \\ adc\_inv[i] - \mathsf{ped}[i] & \mathsf{mask}[i] = 0 \text{ and } -32 <= adc\_inv[i] - \mathsf{ped}[i] <= 31 \\ 31 & \mathsf{mask}[i] = 0 \text{ and } 31 < adc\_inv[i] - \mathsf{ped}[i] > 31 \end{cases} . \tag{9}$$

Similarly the calculation of mean common mode correction (equation 6 from page 14) will be evaluated as

$$adc\_mcm[i] = \begin{cases} 0 & \mathsf{mask}[i] = 1 \\ -32 & \mathsf{mask}[i] = 0 \text{ and } adc\_ped[i] - mcm\_value < -32 \\ \begin{array}{l} adc\_ped[i] \\ -\ mcm\_value \end{array} & \mathsf{mask}[i] = 0 \text{ and } -32 <= \begin{array}{c} adc\_ped[i] \\ -\ mcm\_value \end{array} <= 31 \\ 31 & \mathsf{mask}[i] = 0 \text{ and } 31 < adc\_ped[i] - mcm\_value > 31 \end{cases} . \tag{10}$$

References

[1] K. Wyllie et al., *Electronics Architecture of the LHCb Upgrade*, CERN, Geneva, *http://cds.cern.ch/record/1340939*LHCb-PUB-2011-011.

[2] F. Alessio, R. Jacobsson, *Readout Control Specifications for the Front-End and Back-End of the LHCb Upgrade*, CERN, Geneva, *http://cds.cern.ch/record/1491666*LHCb-PUB-2012-017.

[3] V. Hariprasath, J. Guerber, S-H. Lee, U-K. Moon, *Merged capacitor switching based SAR ADC with highest switching energy-efficiency*, *Electronics Letters*, vol. 46, no. 9, April 2010.

A. ASIC registers

In this section the registers which are set internally and which control various SALT parameters are described. The 8-bit registers from a 12-bit address space are used. As shown in table 13 the registers are divided into four groups related to: serializer parameters, DSP parameters, analogue and mixed-mode parameters, and other parameters. The first 4 bits of the address are used to mark the group while the next 8 bits (set to 00 in address field of table 13) distinguish various parameters within a group. Presently in SALT only a small fraction of 12-bit address space is used. The parameters settings

Address	Name
'h000	BASE_SER
'h100	BASE_DSP
'h200	BASE_ANA
'h300	BASE_OTHER
'h400	BASE_TFC
'h500	BASE_MEM

Table 13: Base addresses for configuration registers blocks

of all groups are described in the following sub-sections.

A.1. Analogue and mixed-mode registers

Table 14: Analogue and mixed-mode configuration registers; all addresses in the table are related to BASE_ANA address

Addr	Name	Bit	Bit Name	Description
0	ana_g_cfg		reset value	'h00
		7	*glob_trim*	Use global base line register for all trim DACs
		6	–	Reserved
		5:3	*inactive_bits*	Number of deactivated bits in all ADCs: • 0 – all bits active • 1 – only bit 0 is always zero • 2 – bits 1:0 are always zero • 3 – bits 2:0 are always zero • 4 – bits 3:0 are always zero • 5 – bits 4:0 are always zero

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		2:0	*set_del*	ADC internal delay; the smaller the value the faster is the ADC
1	preamp_cfg	 7:0	reset value *preamp_dac*	'hBF $I_{pre} = 24\mu A$ Preamplifier biasing DAC value
2	s2d_krum_cfg	 7 6:0	reset value *s2d_low_gain* *krum_dac*	'hDF $I_{krum} = 6nA$ Single-ened to differential converter gain setting; if *s2d_low_gain* is one gain is twice smaller then if it's zero Krumenacher feed-back DAC value
3	shaper_cfg	 7:0	reset value *shaper_dac*	'h9E $I_{sh} = 2\mu A$ Shaper biasing DAC value
4	s2d_cfg	 7:0	reset value *s2d_dac*	'h9E $I_{s2d} = 2\mu A$ Single-ened to differential converter biasing DAC value
5	baseline_g_cfg	 7:0	reset value *all_trim[7:0]*	'h7F $I_{base} = 0\mu A$ Value for all trim DACs if *glob_trim*=1
6	baseline0_cfg	 7:0	reset value *trim_dac[7:0]*	'h7F $I_{base} = 0\mu A$ Trim DAC for channel 0 and test channel 0
7	baseline1_cfg	 7:0	reset value *trim_dac[15:8]*	'h7F Trim DAC for channel 1
8	baseline2_cfg	 7:0	reset value *trim_dac[23:16]*	'h7F Trim DAC for channel 2
9	baseline3_cfg	 7:0	reset value *trim_dac[31:24]*	'h7F Trim DAC for channel 3
10	baseline4_cfg	 7:0	reset value *trim_dac[39:32]*	'h7F Trim DAC for channel 4
11	baseline5_cfg	 7:0	reset value *trim_dac[47:40]*	'h7F Trim DAC for channel 5
12	baseline6_cfg	 7:0	reset value *trim_dac[55:48]*	'h7F Trim DAC for channel 6
13	baseline7_cfg	 7:0	reset value *trim_dac[63:56]*	'h7F Trim DAC for channel 7
14	baseline8_cfg	 7:0	reset value *trim_dac[71:64]*	'h7F Trim DAC for channel 8
15	baseline9_cfg	 7:0	reset value *trim_dac[79:72]*	'h7F Trim DAC for channel 9
16	baseline10_cfg	 7:0	reset value *trim_dac[87:80]*	'h7F Trim DAC for channel 10

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
17	baseline11_cfg		reset value	'h7F
		7:0	*trim_dac[95:88]*	Trim DAC for channel 11
18	baseline12_cfg		reset value	'h7F
		7:0	*trim_dac[103:96]*	Trim DAC for channel 12
19	baseline13_cfg		reset value	'h7F
		7:0	*trim_dac[111:104]*	Trim DAC for channel 13
20	baseline14_cfg		reset value	'h7F
		7:0	*trim_dac[119:112]*	Trim DAC for channel 14
21	baseline15_cfg		reset value	'h7F
		7:0	*trim_dac[127:120]*	Trim DAC for channel 15
22	baseline16_cfg		reset value	'h7F
		7:0	*trim_dac[135:128]*	Trim DAC for channel 16
23	baseline17_cfg		reset value	'h7F
		7:0	*trim_dac[143:136]*	Trim DAC for channel 17
24	baseline18_cfg		reset value	'h7F
		7:0	*trim_dac[151:144]*	Trim DAC for channel 18
25	baseline19_cfg		reset value	'h7F
		7:0	*trim_dac[159:152]*	Trim DAC for channel 19
26	baseline20_cfg		reset value	'h7F
		7:0	*trim_dac[167:160]*	Trim DAC for channel 20
27	baseline21_cfg		reset value	'h7F
		7:0	*trim_dac[175:168]*	Trim DAC for channel 21
28	baseline22_cfg		reset value	'h7F
		7:0	*trim_dac[183:176]*	Trim DAC for channel 22
29	baseline23_cfg		reset value	'h7F
		7:0	*trim_dac[191:184]*	Trim DAC for channel 23
30	baseline24_cfg		reset value	'h7F
		7:0	*trim_dac[199:192]*	Trim DAC for channel 24
31	baseline25_cfg		reset value	'h7F
		7:0	*trim_dac[207:200]*	Trim DAC for channel 25
32	baseline26_cfg		reset value	'h7F
		7:0	*trim_dac[215:208]*	Trim DAC for channel 26
33	baseline27_cfg		reset value	'h7F
		7:0	*trim_dac[223:216]*	Trim DAC for channel 27
34	baseline28_cfg		reset value	'h7F
		7:0	*trim_dac[231:224]*	Trim DAC for channel 28
35	baseline29_cfg		reset value	'h7F
		7:0	*trim_dac[239:232]*	Trim DAC for channel 29
36	baseline30_cfg		reset value	'h7F
		7:0	*trim_dac[247:240]*	Trim DAC for channel 30

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
37	baseline31_cfg		reset value	’h7F
		7:0	*trim_dac[255:248]*	Trim DAC for channel 31
38	baseline32_cfg		reset value	’h7F
		7:0	*trim_dac[263:256]*	Trim DAC for channel 32
39	baseline33_cfg		reset value	’h7F
		7:0	*trim_dac[271:264]*	Trim DAC for channel 33
40	baseline34_cfg		reset value	’h7F
		7:0	*trim_dac[279:272]*	Trim DAC for channel 34
41	baseline35_cfg		reset value	’h7F
		7:0	*trim_dac[287:280]*	Trim DAC for channel 35
42	baseline36_cfg		reset value	’h7F
		7:0	*trim_dac[295:288]*	Trim DAC for channel 36
43	baseline37_cfg		reset value	’h7F
		7:0	*trim_dac[303:296]*	Trim DAC for channel 37
44	baseline38_cfg		reset value	’h7F
		7:0	*trim_dac[311:304]*	Trim DAC for channel 38
45	baseline39_cfg		reset value	’h7F
		7:0	*trim_dac[319:312]*	Trim DAC for channel 39
46	baseline40_cfg		reset value	’h7F
		7:0	*trim_dac[327:320]*	Trim DAC for channel 40
47	baseline41_cfg		reset value	’h7F
		7:0	*trim_dac[335:328]*	Trim DAC for channel 41
48	baseline42_cfg		reset value	’h7F
		7:0	*trim_dac[343:336]*	Trim DAC for channel 42
49	baseline43_cfg		reset value	’h7F
		7:0	*trim_dac[351:344]*	Trim DAC for channel 43
50	baseline44_cfg		reset value	’h7F
		7:0	*trim_dac[359:352]*	Trim DAC for channel 44
51	baseline45_cfg		reset value	’h7F
		7:0	*trim_dac[367:360]*	Trim DAC for channel 45
52	baseline46_cfg		reset value	’h7F
		7:0	*trim_dac[375:368]*	Trim DAC for channel 46
53	baseline47_cfg		reset value	’h7F
		7:0	*trim_dac[383:376]*	Trim DAC for channel 47
54	baseline48_cfg		reset value	’h7F
		7:0	*trim_dac[391:384]*	Trim DAC for channel 48
55	baseline49_cfg		reset value	’h7F
		7:0	*trim_dac[399:392]*	Trim DAC for channel 49
56	baseline50_cfg		reset value	’h7F
		7:0	*trim_dac[407:400]*	Trim DAC for channel 50

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
57	baseline51_cfg		reset value	'h7F
		7:0	*trim_dac[415:408]*	Trim DAC for channel 51
58	baseline52_cfg		reset value	'h7F
		7:0	*trim_dac[423:416]*	Trim DAC for channel 52
59	baseline53_cfg		reset value	'h7F
		7:0	*trim_dac[431:424]*	Trim DAC for channel 53
60	baseline54_cfg		reset value	'h7F
		7:0	*trim_dac[439:432]*	Trim DAC for channel 54
61	baseline55_cfg		reset value	'h7F
		7:0	*trim_dac[447:440]*	Trim DAC for channel 55
62	baseline56_cfg		reset value	'h7F
		7:0	*trim_dac[455:448]*	Trim DAC for channel 56
63	baseline57_cfg		reset value	'h7F
		7:0	*trim_dac[463:456]*	Trim DAC for channel 57
64	baseline58_cfg		reset value	'h7F
		7:0	*trim_dac[471:464]*	Trim DAC for channel 58
65	baseline59_cfg		reset value	'h7F
		7:0	*trim_dac[479:472]*	Trim DAC for channel 59
66	baseline60_cfg		reset value	'h7F
		7:0	*trim_dac[487:480]*	Trim DAC for channel 60
67	baseline61_cfg		reset value	'h7F
		7:0	*trim_dac[495:488]*	Trim DAC for channel 61
68	baseline62_cfg		reset value	'h7F
		7:0	*trim_dac[503:496]*	Trim DAC for channel 62
69	baseline63_cfg		reset value	'h7F
		7:0	*trim_dac[511:504]*	Trim DAC for channel 63
70	baseline64_cfg		reset value	'h7F
		7:0	*trim_dac[519:512]*	Trim DAC for channel 64
71	baseline65_cfg		reset value	'h7F
		7:0	*trim_dac[527:520]*	Trim DAC for channel 65
72	baseline66_cfg		reset value	'h7F
		7:0	*trim_dac[535:528]*	Trim DAC for channel 66
73	baseline67_cfg		reset value	'h7F
		7:0	*trim_dac[543:536]*	Trim DAC for channel 67
74	baseline68_cfg		reset value	'h7F
		7:0	*trim_dac[551:544]*	Trim DAC for channel 68
75	baseline69_cfg		reset value	'h7F
		7:0	*trim_dac[559:552]*	Trim DAC for channel 69
76	baseline70_cfg		reset value	'h7F
		7:0	*trim_dac[567:560]*	Trim DAC for channel 70

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
77	baseline71_cfg		reset value	'h7F
		7:0	*trim_dac[575:568]*	Trim DAC for channel 71
78	baseline72_cfg		reset value	'h7F
		7:0	*trim_dac[583:576]*	Trim DAC for channel 72
79	baseline73_cfg		reset value	'h7F
		7:0	*trim_dac[591:584]*	Trim DAC for channel 73
80	baseline74_cfg		reset value	'h7F
		7:0	*trim_dac[599:592]*	Trim DAC for channel 74
81	baseline75_cfg		reset value	'h7F
		7:0	*trim_dac[607:600]*	Trim DAC for channel 75
82	baseline76_cfg		reset value	'h7F
		7:0	*trim_dac[615:608]*	Trim DAC for channel 76
83	baseline77_cfg		reset value	'h7F
		7:0	*trim_dac[623:616]*	Trim DAC for channel 77
84	baseline78_cfg		reset value	'h7F
		7:0	*trim_dac[631:624]*	Trim DAC for channel 78
85	baseline79_cfg		reset value	'h7F
		7:0	*trim_dac[639:632]*	Trim DAC for channel 79
86	baseline80_cfg		reset value	'h7F
		7:0	*trim_dac[647:640]*	Trim DAC for channel 80
87	baseline81_cfg		reset value	'h7F
		7:0	*trim_dac[655:648]*	Trim DAC for channel 81
88	baseline82_cfg		reset value	'h7F
		7:0	*trim_dac[663:656]*	Trim DAC for channel 82
89	baseline83_cfg		reset value	'h7F
		7:0	*trim_dac[671:664]*	Trim DAC for channel 83
90	baseline84_cfg		reset value	'h7F
		7:0	*trim_dac[679:672]*	Trim DAC for channel 84
91	baseline85_cfg		reset value	'h7F
		7:0	*trim_dac[687:680]*	Trim DAC for channel 85
92	baseline86_cfg		reset value	'h7F
		7:0	*trim_dac[695:688]*	Trim DAC for channel 86
93	baseline87_cfg		reset value	'h7F
		7:0	*trim_dac[703:696]*	Trim DAC for channel 87
94	baseline88_cfg		reset value	'h7F
		7:0	*trim_dac[711:704]*	Trim DAC for channel 88
95	baseline89_cfg		reset value	'h7F
		7:0	*trim_dac[719:712]*	Trim DAC for channel 89
96	baseline90_cfg		reset value	'h7F
		7:0	*trim_dac[727:720]*	Trim DAC for channel 90

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
97	baseline91_cfg		reset value	'h7F
		7:0	*trim_dac[735:728]*	Trim DAC for channel 91
98	baseline92_cfg		reset value	'h7F
		7:0	*trim_dac[743:736]*	Trim DAC for channel 92
99	baseline93_cfg		reset value	'h7F
		7:0	*trim_dac[751:744]*	Trim DAC for channel 93
100	baseline94_cfg		reset value	'h7F
		7:0	*trim_dac[759:752]*	Trim DAC for channel 94
101	baseline95_cfg		reset value	'h7F
		7:0	*trim_dac[767:760]*	Trim DAC for channel 95
102	baseline96_cfg		reset value	'h7F
		7:0	*trim_dac[775:768]*	Trim DAC for channel 96
103	baseline97_cfg		reset value	'h7F
		7:0	*trim_dac[783:776]*	Trim DAC for channel 97
104	baseline98_cfg		reset value	'h7F
		7:0	*trim_dac[791:784]*	Trim DAC for channel 98
105	baseline99_cfg		reset value	'h7F
		7:0	*trim_dac[799:792]*	Trim DAC for channel 99
106	baseline100_cfg		reset value	'h7F
		7:0	*trim_dac[807:800]*	Trim DAC for channel 100
107	baseline101_cfg		reset value	'h7F
		7:0	*trim_dac[815:808]*	Trim DAC for channel 101
108	baseline102_cfg		reset value	'h7F
		7:0	*trim_dac[823:816]*	Trim DAC for channel 102
109	baseline103_cfg		reset value	'h7F
		7:0	*trim_dac[831:824]*	Trim DAC for channel 103
110	baseline104_cfg		reset value	'h7F
		7:0	*trim_dac[839:832]*	Trim DAC for channel 104
111	baseline105_cfg		reset value	'h7F
		7:0	*trim_dac[847:840]*	Trim DAC for channel 105
112	baseline106_cfg		reset value	'h7F
		7:0	*trim_dac[855:848]*	Trim DAC for channel 106
113	baseline107_cfg		reset value	'h7F
		7:0	*trim_dac[863:856]*	Trim DAC for channel 107
114	baseline108_cfg		reset value	'h7F
		7:0	*trim_dac[871:864]*	Trim DAC for channel 108
115	baseline109_cfg		reset value	'h7F
		7:0	*trim_dac[879:872]*	Trim DAC for channel 109
116	baseline110_cfg		reset value	'h7F
		7:0	*trim_dac[887:880]*	Trim DAC for channel 110

Continued on next page

Table 14 – continued from previous page

Addr	Name	Bit	Bit Name	Description
117	baseline111_cfg		reset value	'h7F
		7:0	*trim_dac[895:888]*	Trim DAC for channel 111
118	baseline112_cfg		reset value	'h7F
		7:0	*trim_dac[903:896]*	Trim DAC for channel 112
119	baseline113_cfg		reset value	'h7F
		7:0	*trim_dac[911:904]*	Trim DAC for channel 113
120	baseline114_cfg		reset value	'h7F
		7:0	*trim_dac[919:912]*	Trim DAC for channel 114
121	baseline115_cfg		reset value	'h7F
		7:0	*trim_dac[927:920]*	Trim DAC for channel 115
122	baseline116_cfg		reset value	'h7F
		7:0	*trim_dac[935:928]*	Trim DAC for channel 116
123	baseline117_cfg		reset value	'h7F
		7:0	*trim_dac[943:936]*	Trim DAC for channel 117
124	baseline118_cfg		reset value	'h7F
		7:0	*trim_dac[951:944]*	Trim DAC for channel 118
125	baseline119_cfg		reset value	'h7F
		7:0	*trim_dac[959:952]*	Trim DAC for channel 119
126	baseline120_cfg		reset value	'h7F
		7:0	*trim_dac[967:960]*	Trim DAC for channel 120
127	baseline121_cfg		reset value	'h7F
		7:0	*trim_dac[975:968]*	Trim DAC for channel 121
128	baseline122_cfg		reset value	'h7F
		7:0	*trim_dac[983:976]*	Trim DAC for channel 122
129	baseline123_cfg		reset value	'h7F
		7:0	*trim_dac[991:984]*	Trim DAC for channel 123
130	baseline124_cfg		reset value	'h7F
		7:0	*trim_dac[999:992]*	Trim DAC for channel 124
131	baseline125_cfg		reset value	'h7F
		7:0	*trim_dac[1007:1000]*	Trim DAC for channel 125
132	baseline126_cfg		reset value	'h7F
		7:0	*trim_dac[1015:1008]*	Trim DAC for channel 126
133	baseline127_cfg		reset value	'h7F
		7:0	*trim_dac[1023:1016]*	Trim DAC for channel 127 and test channel 1
134	ana_seu_cnt_reg		reset value	'h00
		7:0	*ana_seu_counter*	SEU counter for analogue register block

A.2. DSP registers

Table 15: DSP configuration registers; all addresses in the table are related to BASE_DSP address

Addr	Name	Bit	Bit Name	Description
0	ped_g_cfg		reset value	'h00
		7	*invert*	Arithmetic negation of all input ADC samples
		6	*is_const*	Use *const_value* instead of ADC sample value at the input of all pedestal blocks
		5:0	*const_value*	Value applied at all *adc[i]* inputs if *const_value* = 1
1	mcm_th_cfg		reset value	'h20
		7:6	–	Reserved
		5:0	*mcm_th*	Threshold value for MCMS block; signed number
2	mcm_th2_cfg		reset value	'h20
		7:6	–	Reserved
		5:0	*mcm_th2*	Second threshold value for MCMS block; signed number
3	n_zs_cfg		reset value	'h00
		7	–	Reserved
		6:5	*nzs_sel*	Non-zero suppression data source: • 00 – masked ADC data • 01 – raw ADC data • 10 – data after pedestals • 11 – data after MCMS
		4:0	*zs_th*	Zero suppression threshold value; unsigned number
4	pack_cfg		reset value	'h00
		7:3	–	Reserved

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		2:0	*packet_types*	Avaliable packet types: • 'b000 — all (normal operation) • 'b011 – default Header & Idle; Sync if TFC • 'b100 – default Truncated & Idle; Header & Sync if TFC
5	mask0_cfg		reset value	'h00
		7	*mask[7]*	Mask bit for channel 7
		6	*mask[6]*	Mask bit for channel 6
		...	...	
		0	*mask[0]*	Mask bit for channel 0
6	mask1_cfg		reset value	'h00
		7	*mask[15]*	Mask bit for channel 15
		6	*mask[14]*	Mask bit for channel 14
		...	...	
		0	*mask[8]*	Mask bit for channel 8
7	mask2_cfg		reset value	'h00
		7	*mask[23]*	Mask bit for channel 23
		6	*mask[22]*	Mask bit for channel 22
		...	...	
		0	*mask[16]*	Mask bit for channel 16
8	mask3_cfg		reset value	'h00
		7	*mask[31]*	Mask bit for channel 31
		6	*mask[30]*	Mask bit for channel 30
		...	...	
		0	*mask[24]*	Mask bit for channel 24
9	mask4_cfg		reset value	'h00
		7	*mask[39]*	Mask bit for channel 39
		6	*mask[38]*	Mask bit for channel 38
		...	...	
		0	*mask[32]*	Mask bit for channel 32
10	mask5_cfg		reset value	'h00
		7	*mask[47]*	Mask bit for channel 47
		6	*mask[46]*	Mask bit for channel 46
		...	...	
		0	*mask[40]*	Mask bit for channel 40
11	mask6_cfg		reset value	'h00

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7	*mask[55]*	Mask bit for channel 55
		6	*mask[54]*	Mask bit for channel 54
		...	...	
		0	*mask[48]*	Mask bit for channel 48
12	mask7_cfg		reset value	'h00
		7	*mask[63]*	Mask bit for channel 63
		6	*mask[62]*	Mask bit for channel 62
		...	...	
		0	*mask[56]*	Mask bit for channel 56
13	mask8_cfg		reset value	'h00
		7	*mask[71]*	Mask bit for channel 71
		6	*mask[70]*	Mask bit for channel 70
		...	...	
		0	*mask[64]*	Mask bit for channel 64
14	mask9_cfg		reset value	'h00
		7	*mask[79]*	Mask bit for channel 79
		6	*mask[78]*	Mask bit for channel 78
		...	...	
		0	*mask[72]*	Mask bit for channel 72
15	mask10_cfg		reset value	'h00
		7	*mask[87]*	Mask bit for channel 87
		6	*mask[86]*	Mask bit for channel 86
		...	...	
		0	*mask[80]*	Mask bit for channel 80
16	mask11_cfg		reset value	'h00
		7	*mask[95]*	Mask bit for channel 95
		6	*mask[94]*	Mask bit for channel 94
		...	...	
		0	*mask[88]*	Mask bit for channel 88
17	mask12_cfg		reset value	'h00
		7	*mask[103]*	Mask bit for channel 103
		6	*mask[102]*	Mask bit for channel 102
		...	...	
		0	*mask[96]*	Mask bit for channel 96
18	mask13_cfg		reset value	'h00
		7	*mask[111]*	Mask bit for channel 111
		6	*mask[110]*	Mask bit for channel 110
		...	...	
		0	*mask[104]*	Mask bit for channel 104
19	mask14_cfg		reset value	'h00

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7	*mask[119]*	Mask bit for channel 119
		6	*mask[118]*	Mask bit for channel 118
		. . .	. . .	
		0	*mask[112]*	Mask bit for channel 112
20	mask15_cfg		reset value	'h00
		7	*mask[127]*	Mask bit for channel 127
		6	*mask[126]*	Mask bit for channel 126
		. . .	. . .	
		0	*mask[120]*	Mask bit for channel 120
21	ped0_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[0]*	Pedestal bits for channel 0
22	ped1_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[1]*	Pedestal value for channel 1
23	ped2_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[2]*	Pedestal value for channel 2
24	ped3_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[3]*	Pedestal value for channel 3
25	ped4_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[4]*	Pedestal value for channel 4
26	ped5_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[5]*	Pedestal value for channel 5
27	ped6_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[6]*	Pedestal value for channel 6
28	ped7_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[7]*	Pedestal value for channel 7
29	ped8_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[8]*	Pedestal value for channel 8
30	ped9_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[9]*	Pedestal value for channel 9
31	ped10_cfg		reset value	'h00

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:6	–	Reserved
		5:0	*ped[10]*	Pedestal value for channel 10
32	ped11_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[11]*	Pedestal value for channel 11
33	ped12_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[12]*	Pedestal value for channel 12
34	ped13_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[13]*	Pedestal value for channel 13
35	ped14_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[14]*	Pedestal value for channel 14
36	ped15_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[15]*	Pedestal value for channel 15
37	ped16_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[16]*	Pedestal value for channel 16
38	ped17_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[17]*	Pedestal value for channel 17
39	ped18_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[18]*	Pedestal value for channel 18
40	ped19_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[19]*	Pedestal value for channel 19
41	ped20_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[20]*	Pedestal value for channel 20
42	ped21_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[21]*	Pedestal value for channel 21
43	ped22_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[22]*	Pedestal value for channel 22
44	ped23_cfg		reset value	'h00
		7:6	–	Reserved

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		5:0	*ped[23]*	Pedestal value for channel 23
45	ped24_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[24]*	Pedestal value for channel 24
46	ped25_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[25]*	Pedestal value for channel 25
47	ped26_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[26]*	Pedestal value for channel 26
48	ped27_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[27]*	Pedestal value for channel 27
49	ped28_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[28]*	Pedestal value for channel 28
50	ped29_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[29]*	Pedestal value for channel 29
51	ped30_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[30]*	Pedestal value for channel 30
52	ped31_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[31]*	Pedestal value for channel 31
53	ped32_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[32]*	Pedestal value for channel 32
54	ped33_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[33]*	Pedestal value for channel 33
55	ped34_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[34]*	Pedestal value for channel 34
56	ped35_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[35]*	Pedestal value for channel 35
57	ped36_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[36]*	Pedestal value for channel 36

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
58	ped37_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[37]*	Pedestal value for channel 37
59	ped38_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[38]*	Pedestal value for channel 38
60	ped39_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[39]*	Pedestal value for channel 39
61	ped40_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[40]*	Pedestal value for channel 40
62	ped41_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[41]*	Pedestal value for channel 41
63	ped42_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[42]*	Pedestal value for channel 42
64	ped43_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[43]*	Pedestal value for channel 43
65	ped44_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[44]*	Pedestal value for channel 44
66	ped45_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[45]*	Pedestal value for channel 45
67	ped46_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[46]*	Pedestal value for channel 46
68	ped47_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[47]*	Pedestal value for channel 47
69	ped48_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[48]*	Pedestal value for channel 48
70	ped49_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[49]*	Pedestal value for channel 49
71	ped50_cfg		reset value	'h00

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:6	–	Reserved
		5:0	*ped[50]*	Pedestal value for channel 50
72	ped51_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[51]*	Pedestal value for channel 51
73	ped52_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[52]*	Pedestal value for channel 52
74	ped53_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[53]*	Pedestal value for channel 53
75	ped54_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[54]*	Pedestal value for channel 54
76	ped55_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[55]*	Pedestal value for channel 55
77	ped56_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[56]*	Pedestal value for channel 56
78	ped57_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[57]*	Pedestal value for channel 57
79	ped58_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[58]*	Pedestal value for channel 58
80	ped59_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[59]*	Pedestal value for channel 59
81	ped60_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[60]*	Pedestal value for channel 60
82	ped61_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[61]*	Pedestal value for channel 61
83	ped62_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[62]*	Pedestal value for channel 62
84	ped63_cfg		reset value	'h00
		7:6	–	Reserved

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		5:0	*ped[63]*	Pedestal value for channel 63
85	ped64_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[64]*	Pedestal value for channel 64
86	ped65_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[65]*	Pedestal value for channel 65
87	ped66_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[66]*	Pedestal value for channel 66
88	ped67_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[67]*	Pedestal value for channel 67
89	ped68_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[68]*	Pedestal value for channel 68
90	ped69_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[69]*	Pedestal value for channel 69
91	ped70_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[70]*	Pedestal value for channel 70
92	ped71_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[71]*	Pedestal value for channel 71
93	ped72_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[72]*	Pedestal value for channel 72
94	ped73_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[73]*	Pedestal value for channel 73
95	ped74_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[74]*	Pedestal value for channel 74
96	ped75_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[75]*	Pedestal value for channel 75
97	ped76_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[76]*	Pedestal value for channel 76

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
98	ped77_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[77]*	Pedestal value for channel 77
99	ped78_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[78]*	Pedestal value for channel 78
100	ped79_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[79]*	Pedestal value for channel 79
101	ped80_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[80]*	Pedestal value for channel 80
102	ped81_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[81]*	Pedestal value for channel 81
103	ped82_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[82]*	Pedestal value for channel 82
104	ped83_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[83]*	Pedestal value for channel 83
105	ped84_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[84]*	Pedestal value for channel 84
106	ped85_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[85]*	Pedestal value for channel 85
107	ped86_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[86]*	Pedestal value for channel 86
108	ped87_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[87]*	Pedestal value for channel 87
109	ped88_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[88]*	Pedestal value for channel 88
110	ped89_cfg		reset value	’h00
		7:6	–	Reserved
		5:0	*ped[89]*	Pedestal value for channel 89
111	ped90_cfg		reset value	’h00

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:6	–	Reserved
		5:0	*ped[90]*	Pedestal value for channel 90
112	ped91_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[91]*	Pedestal value for channel 91
113	ped92_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[92]*	Pedestal value for channel 92
114	ped93_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[93]*	Pedestal value for channel 93
115	ped94_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[94]*	Pedestal value for channel 94
116	ped95_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[95]*	Pedestal value for channel 95
117	ped96_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[96]*	Pedestal value for channel 96
118	ped97_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[97]*	Pedestal value for channel 97
119	ped98_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[98]*	Pedestal value for channel 98
120	ped99_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[99]*	Pedestal value for channel 99
121	ped100_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[100]*	Pedestal value for channel 100
122	ped101_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[101]*	Pedestal value for channel 101
123	ped102_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[102]*	Pedestal value for channel 102
124	ped103_cfg		reset value	'h00
		7:6	–	Reserved

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		5:0	*ped[103]*	Pedestal value for channel 103
125	ped104_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[104]*	Pedestal value for channel 104
126	ped105_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[105]*	Pedestal value for channel 105
127	ped106_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[106]*	Pedestal value for channel 106
128	ped107_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[107]*	Pedestal value for channel 107
129	ped108_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[108]*	Pedestal value for channel 108
130	ped109_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[109]*	Pedestal value for channel 109
131	ped110_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[110]*	Pedestal value for channel 110
132	ped111_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[111]*	Pedestal value for channel 111
133	ped112_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[112]*	Pedestal value for channel 112
134	ped113_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[113]*	Pedestal value for channel 113
135	ped114_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[114]*	Pedestal value for channel 114
136	ped115_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[115]*	Pedestal value for channel 115
137	ped116_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[116]*	Pedestal value for channel 116

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
138	ped117_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[117]*	Pedestal value for channel 117
139	ped118_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[118]*	Pedestal value for channel 118
140	ped119_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[119]*	Pedestal value for channel 119
141	ped120_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[120]*	Pedestal value for channel 120
142	ped121_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[121]*	Pedestal value for channel 121
143	ped122_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[122]*	Pedestal value for channel 122
144	ped123_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[123]*	Pedestal value for channel 123
145	ped124_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[124]*	Pedestal value for channel 124
146	ped125_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[125]*	Pedestal value for channel 125
147	ped126_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[126]*	Pedestal value for channel 126
148	ped127_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*ped[127]*	Pedestal value for channel 127
149	zs_channels_cfg		reset value	'h3f
		7:0	*zs_channels*	Maximum number of channels in ZS package
150	dsp_seu_cnt_reg		reset value	'h00
		7:0	*dsp_seu_counter*	SEU counter for DSP register block
151	dsp_overflow_reg		reset value	'h00 (read-only)

Continued on next page

Table 15 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:4	–	Reserved
		3	*trunc_cnt_overflow_snap*	trunc counter overflow snapshot
		2	*bxid_cnt_overflow_snap*	bxid counter overflow snapshot
		1	*trunc_cnt_overflow*	trunc counter overflow bit
		0	*bxid_cnt_overflow*	bxid counter overflow bit
152	bxid_cnt0_reg		reset value	'h00 (read-only)
		7:0	*bxid_cnt[7:0]*	BXID counter
153	bxid_cnt1_reg		reset value	'h00 (read-only)
		7:4	–	Reserved
		3:0	*bxid_cnt[11:8]*	BXID counter
154	bxid_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*bxid_cnt_snap[7:0]*	BXID counter snapshot
155	bxid_cnt1_snap_reg		reset value	'h00 (read-only)
		7:4	–	Reserved
		3:0	*bxid_cnt_snap[11:8]*	BXID counter snapshot
156	trunc_cnt0_reg		reset value	'h00 (read-only)
		7:0	*trunc_cnt[7:0]*	Trunc packet counter
157	trunc_cnt1_reg		reset value	'h00 (read-only)
		7:0	*trunc_cnt[15:8]*	Trunc packet counter
158	trunc_cnt2_reg		reset value	'h00 (read-only)
		7:0	*trunc_cnt[23:16]*	Trunc packet counter
159	trunc_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*trunc_cnt_snap[7:0]*	Trunc counter snapshot
160	trunc_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*trunc_cnt_snap[15:8]*	Trunc counter snapshot
161	trunc_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*trunc_cnt_snap[23:16]*	Trunc counter snapshot

A.3. Memory registers and counters

Table 16: Memory subsystem configuration registers and counters; all addresses in the table are related to BASE_MEM address

Addr	Name	Bit	Bit Name	Description
0	mem_pack_cfg		reset value	'h00
		7:2	–	Reserved
		1:0	*mem_packet_types*	Available packet types: • 'b00 — all (normal memory operation) • 'b01 – Sync packet only • 'b10 – Idle only; Sync if TFC
1	sync0_cfg		reset value	'h0F
		7:0	*sync_ptt[7:0]*	sixth byte of Sync data packet (sent to e-link 0)
2	sync1_cfg		reset value	'h99
		7:0	*sync_ptt[7:0]*	fifth byte of Sync data packet (sent to e-link 1)
3	sync2_cfg		reset value	'h55
		7:0	*sync_ptt[7:0]*	fourth byte of Sync data packet (sent to e-link 2)
4	sync3_cfg		reset value	'hAA
		7:0	*sync_ptt[7:0]*	third byte of Sync data packet (sent to e-link 3)
5	sync4_cfg		reset value	'hC
		7:4	–	Reserved
		3:0	*sync_ptt[11:8]*	4 MSb in second byte of Sync data packet (sent to e-link 4)
6	elinks_cfg		reset value	'hC
		7:2	–	Reserved
		1:0	*add_elinks*	additional active elinks (over 3)
7	mem_seu_cnt_reg		reset value	'h00
		7:0	*mem_seu_counter*	SEU counter for memory register block
8	mem_overflow_reg		reset value	'h00 (read-only)
		7:6	–	Reserved
		0	*idle_cnt_overflow*	idle counter overflow bit
9	mem_space_reg		reset value	'h00 (read-only)

Continued on next page

Table 16 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*mem_space[7:0]*	memory space value
10	mem_space_snap_reg		reset value	’h00 (read-only)
		7:0	*mem_space_snap[7:0]*	memory space snapshot
11	idle_cnt0_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt[7:0]*	Idle packet counter
12	idle_cnt1_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt[15:8]*	Idle packet counter
13	idle_cnt2_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt[23:16]*	Idle packet counter
14	idle_cnt3_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt[31:24]*	Idle packet counter
15	idle_cnt4_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt[39:32]*	Idle packet counter
16	idle_cnt4_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt[47:40]*	Idle packet counter
17	idle_cnt0_snap_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt_snap[7:0]*	Idle counter snapshot
18	idle_cnt1_snap_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt_snap[15:8]*	Idle counter snapshot
19	idle_cnt2_snap_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt_snap[23:16]*	Idle counter snapshot
20	idle_cnt3_snap_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt_snap[31:24]*	Idle counter snapshot
21	idle_cnt4_snap_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt_snap[39:32]*	Idle counter snapshot
22	idle_cnt5_snap_reg		reset value	’h00 (read-only)
		7:0	*idle_cnt_snap[47:40]*	Idle counter snapshot

A.4. Serializer registers

Table 17: Serializer configuration registers; all addresses in the table are related to BASE_SER address

Addr	Name	Bit	Bit Name	Description
0	ser_source_cfg		reset value	'h22 = 'b00_10_0_010
		7:6	–	Reserved
		5:4	*counter_mode*	Internal counter mode: • 'b00 – pseudorandom sequence 0 • 'b01 – pseudorandom sequence 1 • 'b10 – count up (+1) • 'b11 – count down (−1)
		3	*short_loop*	TFC deserializer output connected directly to serializers in *data_clock* domain. Only if this bit is zero the field *serial_input_sel* is taken into account.
		2:0	*serial_input_sel*	Selection of serializer input: • 'b000 – DSP data • 'b001 – TFC deserializer output sent through *main_clock* domain • 'b010 – pattern register • 'b011 – internal counters • 'b100 – TFC deserializer output after FIFO • 'b101 – TFC command after whole distributed FIFO in DSP block
1	pattern_cfg		reset value	'hAA = 'b1010_1010
		7:0	–	Pattern value
2	deser_cfg		reset value	'h02 = 'b000_000_10
		7:5	–	Reserved
		4:2	*deser_byte_start*	Start point of a byte (in deserializer) in relation to rising edge of main clock
		1:0	*data_clk_sel[1:0]*	First synchronizer clock edge selection; 0 is rising edge
3	pll_clk_cfg		reset value	'h08
		7:4	*pll_clk_sel[1]*	Pll clock phase selection for input synchronizer 1
		3:0	*pll_clk_sel[0]*	Pll clock phase selection for input synchronizer 0
4	pll_main_cfg		reset value	'h0D = 'b0000_1101
				Continued on next page

Table 17 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7	*pll_enable*	Enable PLL when set to 1
		6:4	–	Reserved
		3:2	*pll_gain*	PLL gain
		1:0	*pll_div*	PLL divider
5	pll_cp_cfg		reset value	'h9A = 'b1_0011010; don't change it
		7	–	Reserved
		6:0	*pll_cp_curr*	Charge pump current selection 0–40μA
6	pll_vco_cfg		reset value	'h9A = 'b1_0011010; don't change it
		7	–	Reserved
		6:0	*pll_vco_curr*	VCO current selection
7	tfc_fifo_cfg		reset value	'h04
		7:0	*tfc_fifo_len*	Current TFC FIFO delay
8	ser_g_cfg		reset value	'h00
		7:3	–	Reserved
		2:0	*ser_byte_start*	Start point of a byte in relation to rise edge of main clock
9	calib_fifo_cfg		reset value	'h04
		7:0	*calib_fifo_len*	Current calibration FIFO delay
10	ser_seu_cnt_reg		reset value	'h00
		7:0	*ser_seu_counter*	SEU counter for serialiser register block

A.5. Other configuration registers

Table 18: Other configuration registers; all addresses in the table are related to BASE_OTHER address

Addr	Name	Bit	Bit Name	Description
0	others_g_cfg		reset value	'h04
		7	–	Reserved
		6	*dll_start*	Start DLL
		5	*dll_connect*	source selection for *adc_clk* & *calib_clk*: • 0 – *MAIN_CLK* connected • 1 – multiplexer outputs connected
		4	*slvs_termination*	connect termination (100 Ω) to SLVS receivers
		3	*adc_mon_off*	turn off monitoring ADCs: Band-gap, DLL, PLL, Band-gap, and Temperature
		2	*adc_test_off*	turn off test ADCs
		1:0	–	Reserved
1	dll_cp_cfg		reset value	'h9A = 'b1_0011010; don't change
		7	–	Reserved
		6:0	*dll_cp_cur*	Charge pump current 0–40μA
2	dll_vcdl_cfg		reset value	'h9A = 'b1_0011010;
		7	–	Reserved
		6:0	*dll_vcdl_cur*	Delay line biasing current 0–40μA
3	adc_clk_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*adc_clk_sel*	DLL clock phase selection for *adc_clk*
4	calib_clk_cfg		reset value	'h00
		7:6	–	Reserved
		5:0	*calib_clk_sel*	DLL clock phase selection for *calib_clk*
5	calib_main_cfg		reset value	'h05
		7	*calib_inv*	invert calibration impulse
		6:5	–	Reserved
		4:0	*calib_pulse_len*	length of calibration impulse (in clock cycles)
6	calib_volt_cfg		reset value	'h7f $V_{cal} = ??mV$

Continued on next page

Table 18 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*calib_volt_dac*	Calibration impulse voltage
7	calib_mask0_cfg		reset value	'h00
		7	*calib_mask[7]*	Mask bit for channel 7
		6	*calib_mask[6]*	Mask bit for channel 6
		...	...	
		0	*calib_mask[0]*	Mask bit for channel 0
8	calib_mask1_cfg		reset value	'h00
		7	*calib_mask[15]*	Mask bit for channel 15
		6	*calib_mask[14]*	Mask bit for channel 14
		...	...	
		0	*calib_mask[8]*	Mask bit for channel 8
9	calib_mask2_cfg		reset value	'h00
		7	*calib_mask[23]*	Mask bit for channel 23
		6	*calib_mask[22]*	Mask bit for channel 22
		...	...	
		0	*calib_mask[16]*	Mask bit for channel 16
10	calib_mask3_cfg		reset value	'h00
		7	*calib_mask[31]*	Mask bit for channel 31
		6	*calib_mask[30]*	Mask bit for channel 30
		...	...	
		0	*calib_mask[24]*	Mask bit for channel 24
11	calib_mask4_cfg		reset value	'h00
		7	*calib_mask[39]*	Mask bit for channel 39
		6	*calib_mask[38]*	Mask bit for channel 38
		...	...	
		0	*calib_mask[32]*	Mask bit for channel 32
12	calib_mask5_cfg		reset value	'h00
		7	*calib_mask[47]*	Mask bit for channel 47
		6	*calib_mask[46]*	Mask bit for channel 46
		...	...	
		0	*calib_mask[40]*	Mask bit for channel 40
13	calib_mask6_cfg		reset value	'h00
		7	*calib_mask[55]*	Mask bit for channel 55
		6	*calib_mask[54]*	Mask bit for channel 54
		...	...	
		0	*calib_mask[48]*	Mask bit for channel 48
14	calib_mask7_cfg		reset value	'h00
		7	*calib_mask[63]*	Mask bit for channel 63
		6	*calib_mask[62]*	Mask bit for channel 62
		...	...	

Continued on next page

Table 18 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		0	*calib_mask[56]*	Mask bit for channel 56
15	calib_mask8_cfg		reset value	’h00
		7	*calib_mask[71]*	Mask bit for channel 71
		6	*calib_mask[70]*	Mask bit for channel 70
		...	...	
		0	*calib_mask[64]*	Mask bit for channel 64
16	calib_mask9_cfg		reset value	’h00
		7	*calib_mask[79]*	Mask bit for channel 79
		6	*calib_mask[78]*	Mask bit for channel 78
		...	...	
		0	*calib_mask[72]*	Mask bit for channel 72
17	calib_mask10_cfg		reset value	’h00
		7	*calib_mask[87]*	Mask bit for channel 87
		6	*calib_mask[86]*	Mask bit for channel 86
		...	...	
		0	*calib_mask[80]*	Mask bit for channel 80
18	calib_mask11_cfg		reset value	’h00
		7	*calib_mask[95]*	Mask bit for channel 95
		6	*calib_mask[94]*	Mask bit for channel 94
		...	...	
		0	*calib_mask[88]*	Mask bit for channel 88
19	calib_mask12_cfg		reset value	’h00
		7	*calib_mask[103]*	Mask bit for channel 103
		6	*calib_mask[102]*	Mask bit for channel 102
		...	...	
		0	*calib_mask[96]*	Mask bit for channel 96
20	calib_mask13_cfg		reset value	’h00
		7	*calib_mask[111]*	Mask bit for channel 111
		6	*calib_mask[110]*	Mask bit for channel 110
		...	...	
		0	*calib_mask[104]*	Mask bit for channel 104
21	calib_mask14_cfg		reset value	’h00
		7	*calib_mask[119]*	Mask bit for channel 119
		6	*calib_mask[118]*	Mask bit for channel 118
		...	...	
		0	*calib_mask[112]*	Mask bit for channel 112
22	calib_mask15_cfg		reset value	’h00
		7	*calib_mask[127]*	Mask bit for channel 127
		6	*calib_mask[126]*	Mask bit for channel 126
		...	...	

Continued on next page

Table 18 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		0	*calib_mask[120]*	Mask bit for channel 120
23	slvs_cur_cfg		reset value	'hB4 $I_{slvs} = 215\mu A$
		7:0	*slvs_cur_dac*	SLVS biasing current
24	slvs_vcm_cfg		reset value	'h4B $V_{cm} = 390mV$
		7:0	*slvs_vcm_dac*	SLVS common voltage
25	bandgap0_cfg		reset value	'h00
		7:0	*bandgap_control[7:0]*	Band-gap control bits
26	bandgap1_cfg		reset value	'h00
		7:0	*bandgap_control[15:8]*	Band-gap control bits
27	vcm_cur0_cfg		reset value	'h10
		7:5	–	Reserved
		4:0	*vcm_cur[4:0]*	Biasing for vcm_a buffer
28	vcm_cur1_cfg		reset value	'h10
		7:5	–	Reserved
		4:0	*vcm_cur[9:5]*	Biasing for vcm_b buffer
29	vcm_cur2_cfg		reset value	'h10
		7:5	–	Reserved
		4:0	*vcm_cur[14:10]*	Biasing for vcm_c buffer
30	vcm_cur3_cfg		reset value	'h10
		7:5	–	Reserved
		4:0	*vcm_cur[19:15]*	Biasing for vcm_d buffer
31	dac_mon_cfg		reset value	'h00
		7:0	*dac_mon_sel[7:0]*	Select DACs for monitoring
32	other_seu_cnt_reg		reset value	'h00
		7:0	*other_seu_counter*	SEU counter for other register block
33	dll_vcdl_mon		reset value	'h00 (read-only)
		7	*dll_curr_ok*	1 if VCDL current OK when *dll_start*=0
		6	–	Reserved
		5:0	*dll_vcdl_voltage*	Delay line control voltage (2's complement)
34	pll_vco_mon		reset value	'h00 (read-only)
		7:6	–	Reserved
		5:0	*pll_vco_voltage*	PLL VCO control voltage (2's complement)
35	bandgap_mon		reset value	'h00 (read-only)
		7:0	*bandgap_voltage*	Band-gap voltage; 2's complement in relation to half supply voltage
36	temp_mon		reset value	'h00 (read-only)

Continued on next page

Table 18 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*temp_voltage*	Temperature voltage (2's complement)
37	adc_test0_reg		reset value	'h00 (read-only)
		7:6	–	Reserved
		5:0	*adc_test0_value*	Test ADC laying before channel 0 in the layout (2's complement)
38	adc_test1_reg		reset value	'h00 (read-only)
		7:6	–	Reserved
		5:0	*adc_test1_value*	Test ADC laying after channel 127 in the layout (2's complement)
39	dac0_mon		reset value	'h00 (read-only)
		7:0	*dac0_voltage*	DAC0 control voltage
40	dac1_mon		reset value	'h00 (read-only)
		7:0	*dac1_voltage*	DAC1 control voltage

A.6. TFC counters

Table 19: TFC read-only registers; all addresses in the table are related to BASE_TFC address

Addr	Name	Bit	Bit Name	Description
0	tfc_seu_cnt_reg		reset value	'h00
		7:0	*tfc_seu_counter*	SEU counter for TFC counters block
1	tfc_overflow_reg		reset value	'h00
		7	*calib_overflow*	calib counter overflow bit
		6	*synch_overflow*	synch counter overflow bit
		5	*snapshot_overflow*	snapshot counter overflow bit
		4	*bxveto_overflow*	bxveto counter overflow bit
		3	*nzs_overflow*	nzs counter overflow bit
		2	*header_overflow*	header counter overflow bit
		1	*fereset_overflow*	fereset counter overflow bit
		0	*bxreset_overflow*	bxreset counter overflow bit
2	tfc_overflow_snap_reg		reset value	'h00 (read-only)
		7:0	*tfc_overflow_snap*	Snapshot of tfc_overflow_reg register
3	calib_cnt0_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt[7:0]*	TFC calib command counter
4	calib_cnt1_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt[15:8]*	TFC calib command counter
5	calib_cnt2_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt[23:16]*	TFC calib command counter
6	calib_cnt3_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt[31:24]*	TFC calib command counter
7	calib_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt_snap[7:0]*	TFC calib counter snapshot
8	calib_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt_snap[15:8]*	TFC calib counter snapshot
9	calib_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt_snap[23:16]*	TFC calib counter snapshot
10	calib_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*calib_cnt_snap[31:24]*	TFC calib counter snapshot
11	synch_cnt0_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt[7:0]*	TFC synch command counter
12	synch_cnt1_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt[15:8]*	TFC synch command counter
13	synch_cnt2_reg		reset value	'h00 (read-only)

Continued on next page

Table 19 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*synch_cnt[23:16]*	TFC synch command counter
14	synch_cnt3_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt[31:24]*	TFC synch command counter
15	synch_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt_snap[7:0]*	TFC synch counter snapshot
16	synch_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt_snap[15:8]*	TFC synch counter snapshot
17	synch_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt_snap[23:16]*	TFC synch counter snapshot
18	synch_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*synch_cnt_snap[31:24]*	TFC synch counter snapshot
19	snapshot_cnt0_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt[7:0]*	TFC snapshot command counter
20	snapshot_cnt1_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt[15:8]*	TFC snapshot command counter
21	snapshot_cnt2_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt[23:16]*	TFC snapshot command counter
22	snapshot_cnt3_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt[31:24]*	TFC snapshot command counter
23	snapshot_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt_snap[7:0]*	TFC snapshot counter snapshot
24	snapshot_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt_snap[15:8]*	TFC snapshot counter snapshot
25	snapshot_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt_snap[23:16]*	TFC snapshot counter snapshot
26	snapshot_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*snapshot_cnt_snap[31:24]*	TFC snapshot counter snapshot
27	bxveto_cnt0_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt[7:0]*	TFC bxveto command counter
28	bxveto_cnt1_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt[15:8]*	TFC bxveto command counter
29	bxveto_cnt2_reg		reset value	'h00 (read-only)

Continued on next page

Table 19 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*bxveto_cnt[23:16]*	TFC bxveto command counter
30	bxveto_cnt3_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt[31:24]*	TFC bxveto command counter
31	bxveto_cnt4_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt[39:32]*	TFC bxveto command counter
32	bxveto_cnt4_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt[47:40]*	TFC bxveto command counter
33	bxveto_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt_snap[7:0]*	TFC bxveto counter snapshot
34	bxveto_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt_snap[15:8]*	TFC bxveto counter snapshot
35	bxveto_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt_snap[23:16]*	TFC bxveto counter snapshot
36	bxveto_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt_snap[31:24]*	TFC bxveto counter snapshot
37	bxveto_cnt4_snap_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt_snap[39:32]*	TFC bxveto counter snapshot
38	bxveto_cnt5_snap_reg		reset value	'h00 (read-only)
		7:0	*bxveto_cnt_snap[47:40]*	TFC bxveto counter snapshot
39	nzs_cnt0_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt[7:0]*	TFC nzs command counter
40	nzs_cnt1_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt[15:8]*	TFC nzs command counter
41	nzs_cnt2_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt[23:16]*	TFC nzs command counter
42	nzs_cnt3_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt[31:24]*	TFC nzs command counter
43	nzs_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt_snap[7:0]*	TFC nzs counter snapshot
44	nzs_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt_snap[15:8]*	TFC nzs counter snapshot
45	nzs_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt_snap[23:16]*	TFC nzs counter snapshot
46	nzs_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*nzs_cnt_snap[31:24]*	TFC nzs counter snapshot
47	header_cnt0_reg		reset value	'h00 (read-only)
		7:0	*header_cnt[7:0]*	TFC header command counter
48	header_cnt1_reg		reset value	'h00 (read-only)
		7:0	*header_cnt[15:8]*	TFC header command counter
49	header_cnt2_reg		reset value	'h00 (read-only)

Continued on next page

Table 19 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*header_cnt[23:16]*	TFC header command counter
50	header_cnt3_reg		reset value	'h00 (read-only)
		7:0	*header_cnt[31:24]*	TFC header command counter
51	header_cnt4_reg		reset value	'h00 (read-only)
		7:0	*header_cnt[39:32]*	TFC header command counter
52	header_cnt4_reg		reset value	'h00 (read-only)
		7:0	*header_cnt[47:40]*	TFC header command counter
53	header_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*header_cnt_snap[7:0]*	TFC header counter snapshot
54	header_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*header_cnt_snap[15:8]*	TFC header counter snapshot
55	header_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*header_cnt_snap[23:16]*	TFC header counter snapshot
56	header_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*header_cnt_snap[31:24]*	TFC header counter snapshot
57	header_cnt4_snap_reg		reset value	'h00 (read-only)
		7:0	*header_cnt_snap[39:32]*	TFC header counter snapshot
58	header_cnt5_snap_reg		reset value	'h00 (read-only)
		7:0	*header_cnt_snap[47:40]*	TFC header counter snapshot
59	fereset_cnt0_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt[7:0]*	TFC fereset command counter
60	fereset_cnt1_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt[15:8]*	TFC fereset command counter
61	fereset_cnt2_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt[23:16]*	TFC fereset command counter
62	fereset_cnt3_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt[31:24]*	TFC fereset command counter
63	fereset_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt_snap[7:0]*	TFC fereset counter snapshot
64	fereset_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt_snap[15:8]*	TFC fereset counter snapshot
65	fereset_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt_snap[23:16]*	TFC fereset counter snapshot
66	fereset_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*fereset_cnt_snap[31:24]*	TFC fereset counter snapshot
67	bxreset_cnt0_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt[7:0]*	TFC bxreset command counter
68	bxreset_cnt1_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt[15:8]*	TFC bxreset command counter
69	bxreset_cnt2_reg		reset value	'h00 (read-only)

Continued on next page

Table 19 – continued from previous page

Addr	Name	Bit	Bit Name	Description
		7:0	*bxreset_cnt[23:16]*	TFC bxreset command counter
70	bxreset_cnt3_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt[31:24]*	TFC bxreset command counter
71	bxreset_cnt0_snap_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt_snap[7:0]*	TFC bxreset counter snapshot
72	bxreset_cnt1_snap_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt_snap[15:8]*	TFC bxreset counter snapshot
73	bxreset_cnt2_snap_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt_snap[23:16]*	TFC bxreset counter snapshot
74	bxreset_cnt3_snap_reg		reset value	'h00 (read-only)
		7:0	*bxreset_cnt_snap[31:24]*	TFC bxreset counter snapshot

B. Changes in comparison to previous version

- DLL moved from clock input to ADCs inputs; Input signal *RAW_CLK* is no longer needed, clock in now called *MAIN_CLK* everywhere.
- Asynchronous FIFO added between ADC output and DSP input in each channel; these FIFOs delayed ADC data by 4 clock cycles.
- Broadcast I^2C address of value 7.
- Memory subsystem redesigned to cope with variable number of e-links and new packet format (12-bit based)
- Separate SEU counter in each register block added (counting SEU errors in block)
- Analogue registers
  - baseline_g_cfg register added
  - baseline registers added for channels from 8 to 127
- DSP registers
  - number of mask registers enlarged to 16
  - number of pedestal registers increased to cover all 128 channels
  - synch pattern registers moved to memory block
- New register block related to memory subsystem
- Serializer registers
  - tfc_fifo_cfg register reset value set to 'h04.
  - new fifo for Calib command going to calib block – calib_fifo_cfg