DCD-Bv4-Pipeline Reference Manual

Document revision: 1
for chip version 4
January 13, 2015

Ivan Perić, Christian Kreidl, Peter Fischer
Group for Circuits and Simulations
University of Heidelberg

Contents

1 Overview

The Drain Current Digitizer - DCD - chips are used in the Belle-II Pixel Vertex Detector to receive and digitize the current signals generated by the in-pixel transistors of a DEPFET detector. The chips are placed at the edges of the DPEFET sensor. Each chip has 256 analogue channels that are connected by bump bonding to 256 DEPFET row-lines. The DCDB's 256 channels are organized in eight column pairs with 32 channels each. Digital outputs of 32 channels are digitally processed and multiplexed to one 8-bit wide parallel output that sends the ADC codes with 400MBits/s. The DCD has eight such 8-bit outputs. Eight 2-bit wide digital inputs are used to load the offset-correction values at the rate 400MBits/s. Every analogue channel uses one *pipeline* ADC. The detector signal can be sampled every 80 ns. Although we presently use 320MBits/s data rate, in this document we will assume 400MBits/s.

1.0.1 DCDB Specifications

This section gives the list of the specifications:

- ADC resolution: 8 Bit
- ADC sampling rate: 100ns
- Radiation tolerance: 20MRad
- Fixed pre-ADC offset correction (SubIn): up to 200 μA
- Variable pre-ADC offset correction: 2 Bits
- Analog Common Mode Correction (CMC): optional
- Analog Common Mode Correction range: up to 200 μA
- DCD Power consumption: < 2W
- DCD Current consumption: < 800mA
- DCD PSRR DC: < 50mV Vpp
- DCD PSRR AC: 2.5 μV in the frequency range 1 MHz to 250 MHz, in other frequencies about 10mV
- ADC average noise input referred: about 200e ($\sim$ 80nA). (Several noise hot spots within the ADC characteristics are allowed)
- Maximal noise in system: $\sim$ 300e ($\sim$120nA). (In this way we achieve SNR > 17 for MIPs which will lead to a good efficiency)
- ADC LSB: $\sim$ 200e ($\sim$80nA) (fine mode) $\sim$ 400e (160nA) (coarse mode). (Both modes should be implemented)

- ADC nonlinearity (peak to peak) in the range -+ 100 ADC counts: < 6 sigma noise (or 6 LSBs) that corresponds to 480nA in the fine mode or 960nA in the coarse mode
- Localized code jumps of 6 sigma noise (LSBs) allowed (6 missing codes)
- ADC calibration DAC - resolution 9 Bits

Explanation: DEPFET signal is 5000e (MIP), the signal range of interest is up to 4 MIP $\sim$ 20000e ($8\,\mu A$). The DEPFET pedestal dispersion will be after irradiation (peak to peak) $\sim 32\,\mu A$ (optimistic scenario) or $64\,\mu A$ (pessimistic scenario) per SWITCHER segment. By using offset correction bits, we can reduce the pedestal dispersion by factor 4, from $32\,\mu A$ to $8\,\mu A$ or from $648\,\mu A$ to $(16)\,\mu A$.

ADC Range is about $20\,\mu A$ (fine mode) or $40\,\mu A$ coarse mode, the pedestal dispersion occupies about 1/2 of the range (in pessimistic scenario we use coarse mode).

One half of the range is used for the signal. In the case of coarse mode the signal range is up to 50ke. In the case of fine mode the signal range is up to 25ke.

1.1 Chip Geomtry

DCDBv4Pipeline chip is implemented in UMC 180 nm technology using radiation tolerant design techniques. DCD occupies an area of $3240\,\mu m \times 4969\,\mu m$ that is determined by the fixed size of 2×3 "miniasic" blocks.

Analogue channels are arranged as a 16×16 matrix. One channel occupies an area of $200\,\mu m \times 180\,\mu m$.

The input of each analogue channel is connected to a bump-bond pad with the passivation opening of $60\,\mu m$ in diameter.

The minimum bump-bond distance is $200\,\mu m$. The bonding pads are implemented using the additional aluminium metal layer (metal 7); the bumps are placed by the vendor. This commercial technology allows chip-flipping without pressure and removing of already "flipped" chips without damaging the detector (rework).

The bonding pads are arranged hexagonally. This arrangement fits well to $180\,\mu m$-wide channels.

The chip footprint is shown in Fig. 1.1.

1.2 Pad Description

The DCD chip has the minimal number of pads. In this section we give the list of the pads and explain their electrical properties. We classify the DCD pads according to their *purpose* into I/O pads, power pads, fast control pads, slow control pads and test pads.

According to their *electrical properties* we classify the DCD pads into analogue pads (with protection), power pads (without protection), CMOS input pads, CMOS output pads, low-voltage (LV) single-ended output pads and low-voltage (LV) differential input pads.

Pad	Description	Type
A0<0>, ..., A7<31>	DEPFET currents	Analogue (inp.)
DO0<0:7> - DO7<0:7>	Digital out. pads for ADCs	Single-ended LV output
DI0<0:1> - DI7<0:1>	Digital inp. pads for DACs	CMOS inp.
AVDD	Analogue power	1.8V, 364mA + AMPLOW
AGND	Analogue power	0V, -364mA
RefIn	Analogue power	1V, 50mA
AmpLow	Analogue power - ADC	300mV, -247mA
AmpLowAmp	Analogue power - TIA	300mV
DVDD	Digital power	1.8V, 176mA
DGND	Digital power	1.8V, -176mA
Monitor	Calibration input/output	Analogue (i/o)
`CLK`	400MHz clock	CMOS inp.
Strobe	Synch. with Clear	CMOS inp.
`SYNC_RESET`	Synch. with DHP	CMOS inp.
`CLKP`	Digital input	LVDSIN
`CLKN`	Digital input	LVDSIN
TDI	JTAG data input	JTAG, CMOS inp.
TDO	JTAG data output	JTAG, CMOS out.
TMS	JTAG change state	JTAG, CMOS inp.
TCK	JTAG clock	JTAG, CMOS inp.
TRSTB	JTAG reset	JTAG, CMOS inp.
RefOut	Reference voltage	Analogue (out.)

Table 1.1: Pads.

1.2.1 I/O pads

DCD has 256 analogue input pads (A0<0:31>, ..., A7<0:31>) that are connected to DEPFETs and 64 digital output pads (DO0<0:7>, ..., DO7<0:7>) operating at 400MBits/s that transmit ADC codes to Data Handling Processor - DHP. 16 CMOS digital inputs (DI0<0:1>, ..., DI7<0:1>) are used to transmit the offset-correction values (400MBits/s) from DHP to DCD.

Electrical properties of I/O pads

The current input pads are of analogue type and they use small-size protection diodes connected to analogue power (AVDD and AGND).

The digital output pads generate non-standard single-ended signals with the logic-high level of nearly 1.24 V and the logic-low level $\sim 1\,V$. The signals are generated by terminated LVDS output drivers where only one output per driver is used. The other differential output is left unconnected. The termination resistance is $200\,\Omega$ and the bias current of one driver nearly $1.2\,mA$. The drivers use digital power (DVDD).

The digital output signals are received by DHP, that uses differential receivers for this purpose. The inverting input of such a receiver is connected to a reference potential (threshold) that equals the arithmetic mean between the high and low signal level - nearly $1.12\,V$. To simplify DHP design, such a reference voltage is generated on DCD and is accessed using a dedicated analogue pad RefOut.

All the digital input pads, except the LVDS `CLKN` and `CLKP` inputs, are of CMOS type and they use $1.8\,V$ digital supply (DVDD).

The `CLKN` and `CLKP` LVDS clock inputs can be used instead the CMOS clock input `CLK`.

1.2.2 Power-pads

The analogue part of the chip uses one 1.8 V supply voltage (*AVDD*), one ground (*AGND*). The 1.8 V voltage and the ground are connected each by 16 bumps. There are two additional supply voltages - *RefIn* ($\approx 1\,V$) and *AmpLow/AmpLowAmp* ($\approx 300\,mV$). The *AmpLow/AmpLowAmp* generator must drain the current. AmpLow/AmpLowAmp are contacted by 16 and RefIn by 3 pads.

If the chip is operated without the analogue common mode compensation (ACMC), AmpLowAmp is used as the negative power supply for the channel (pixel) amplifiers. If ACMC is used, the negative power supply is generated on the chip - AmpLowAmp pad is then disconnected from the amplifiers. The switch is placed on the chip.

The digital part uses one 1.8 V supply voltage (*DVDD*) and a ground (*DGND*) that are separated from the corresponding analogue voltages.The digital supply and ground are connected each by 16 bumps.

Power pads do not have any protection.

1.2.3 Fast-control pads

DCD receives from the DHP **one 400 MHz clock** (`CLK` or, optionally, `CLKP` and `CLKN`), **one synchronization signal** (`SYNC_RESET`) that repeats once in 128 (320 ns) `CLK` periods (see Fig. 4.1) and, in the case of double-sampling (DKS) operation, **one strobe** (`Strobe`) that is in phase with the "STOBECLR" signal sent from DHPs to Switchers. Notice: DKS is not implemented on DCDBv4Pipeline - `Strobe` is present as pad but not used.

The fast input signals are received either by 1.8V CMOS or by LVDS pads (`CLKP` and `CLKN`).

1.2.4 Slow-control pads

DCD can be configured using JTAG. JTAG pads are the following: Reset (*TRSTB*), Clock (*TCK*), TestModeSelect (*TMS*), DataIn (*TDI*) and DataOut (*TDO*). JTAG pads are CMOS input and output pads that use 1.8 V digital power. Note that TRSTB is active low! (In the first version of DCDB this signal was active high!) Note also that Global and Pixel register sample the input on falling *TCK* edge and release the output of rising *TCK* edge.

1.2.5 Test pads

DCD has one multi-purpose analog pad - *Monitor* - for monitoring the internal voltages and calibration. (The protection diodes are connected to AVDD/AGND.)

List of the pads is shown in Table 1.1.

1.3 Block Diagram

The overall chip architecture is shown in Fig. 1.2. The chip contains the following main blocks:

- 256 analogue channels ('pixels') with signal receivers and ADCs (A. Channels).
- Fully synthesized digital readout- and control block (Dig) that performs processing of the ADC codes and generates fast control signals (Fast ctrl.).
- JTAG interface for accessing the configuration shift registers (JTAG).
- Global configuration block that contains two configuration shift registers (GSR and PSR) and the DACs for bias voltages (GDACs).
- Pad frame that contains digital input- and output pads (Pads).
- 'Debug'-multiplexer for overriding of internal control signals (Dbg).

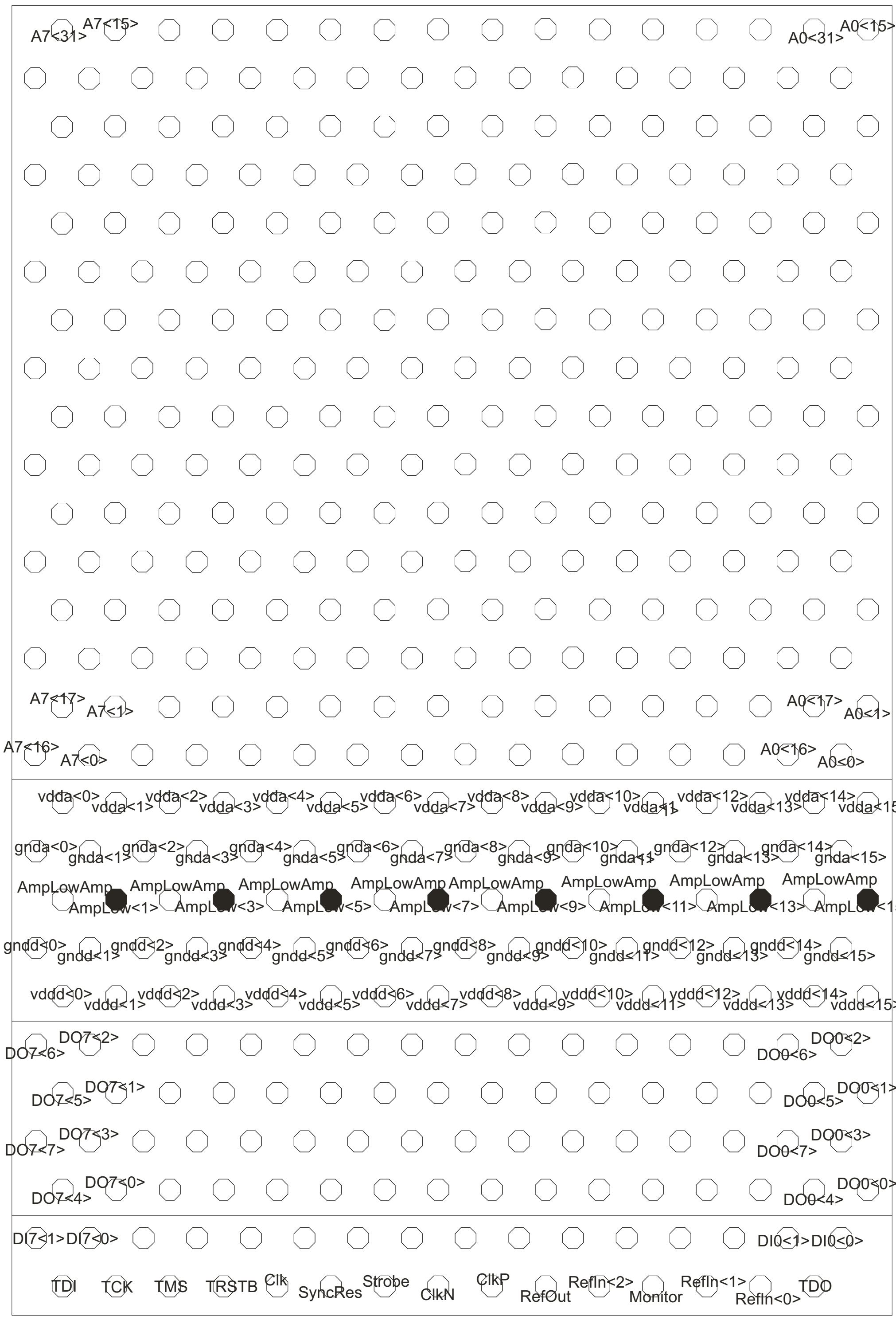

Figure 1.1: Chip footprint (on the module) and pad locations.

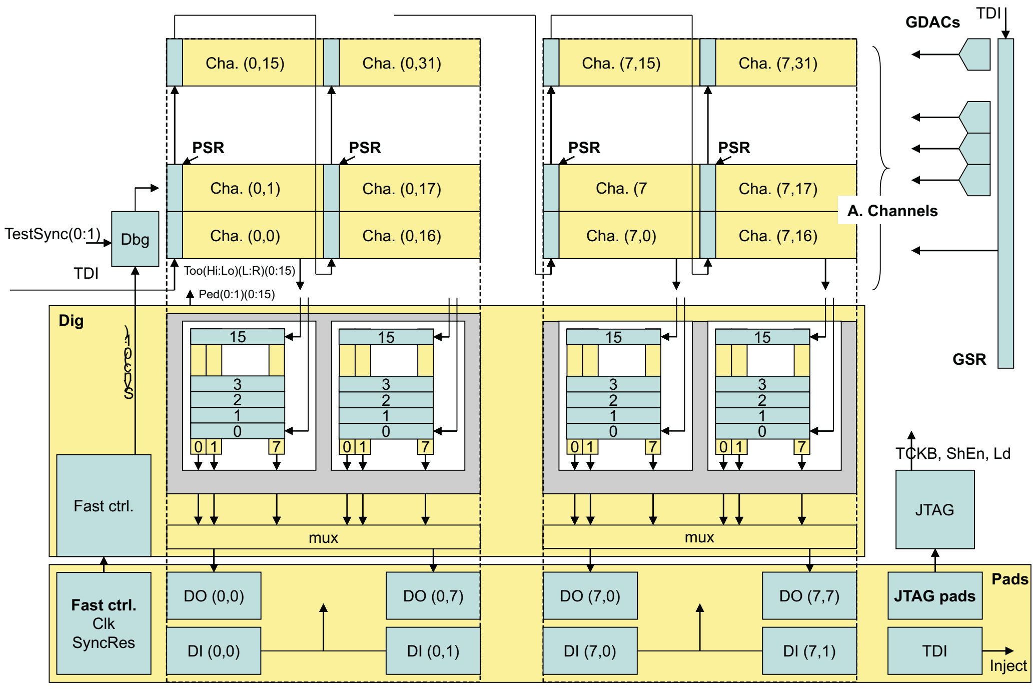

Figure 1.2: The chip architecture.

2 Analogue channel

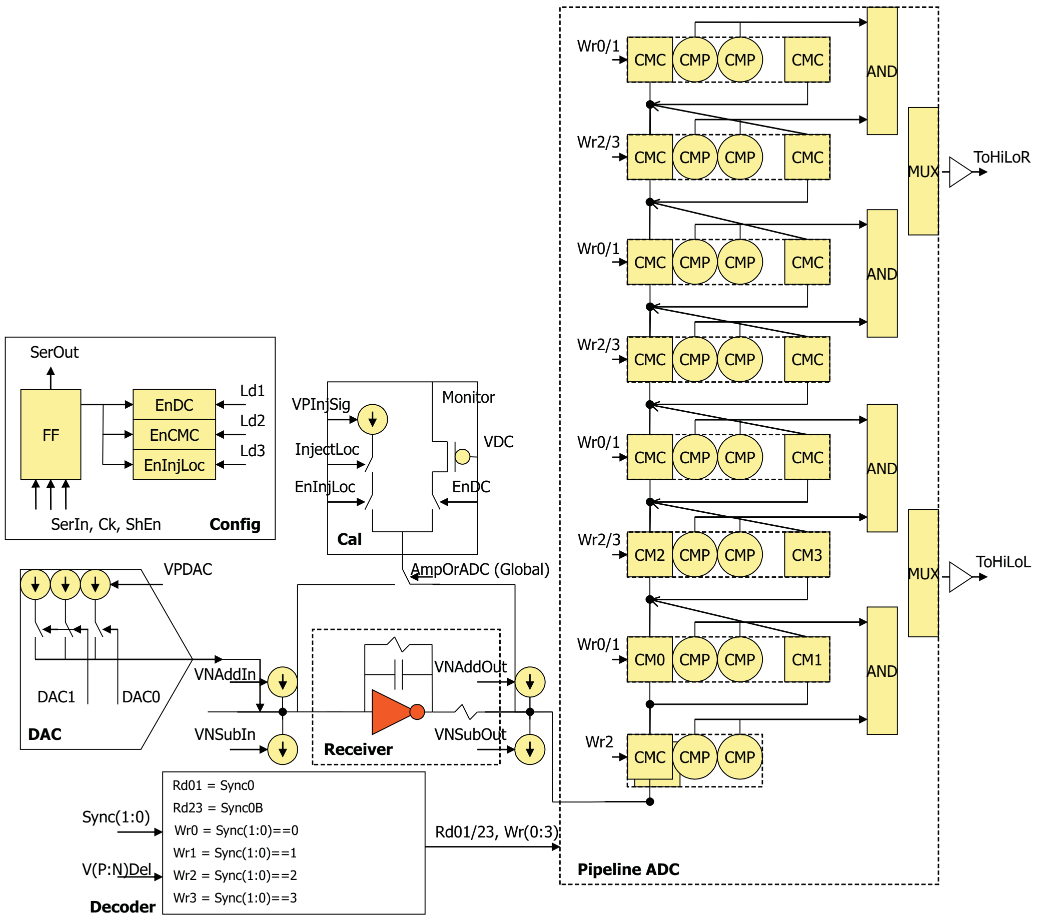

Figure 2.1: Basic circuits of the analogue channel.

The block diagram of the analogue channel is shown in Fig. 2.1.

The main circuits of the bock are:

- Resistive current receiver (Receiver) (based on a trans-impedance amplifier with a resistor on its output) that amplifies the DEPFET current.
- One current-mode pipeline ADC based on current-memory cells.
- 2-bit DAC for pedestal correction (DAC).
- Calibration circuit for the ADCs (Cal).
- Decoder for generating of the control signals for the ADC (Decoder).

- Pixel configuration register (Config).

2.1 Resistive current receiver

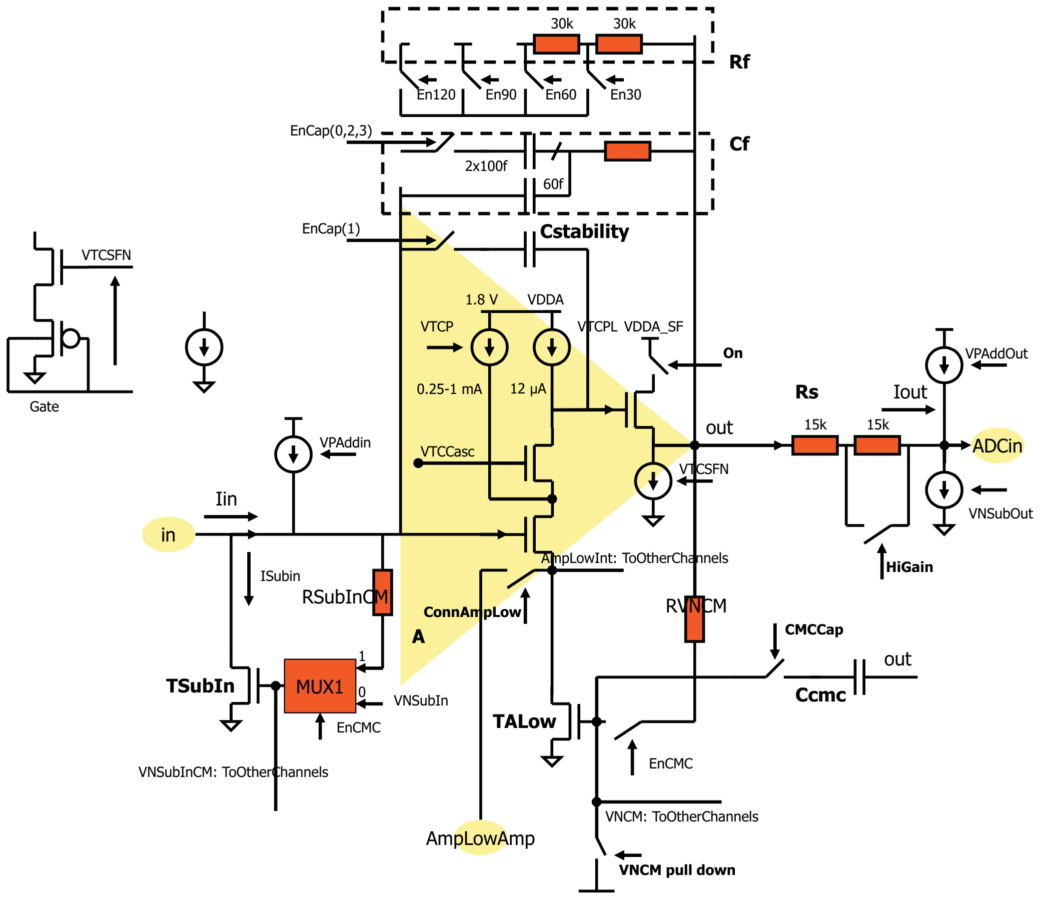

Figure 2.2: The receiver.

The schematic of the resistive current receiver is shown in Fig. 2.2. The hearth of the block is a trans-impedance amplifier (amplifier A with feedback resistors Rf) that receives DEPFET current (I_{in}) and converts it into voltage V_{out}. It holds

$$V_{out} = V_{in} - I_{in} \times Rf.$$

Resistor Rs is connected between V_{out} and the input of the ADC (ADCin). Since the ADC holds its input at a constant potential, the current flowing into the ADC is $I_{out} = (V_{out} - V_{ADCin})/Rs$ or substituting the previous equation

$$I_{out} = I_{offset} - G \times I_{in}$$

with the current gain $G \equiv Rf/Rs$ and offset current $I_{offset} \equiv (V_{in} - V_{ADCin})/Rs$. Clearly, if Rs has a smaller resistance than Rf, current gain G is higher than 1. Current sources SubIn/AddIn and SubOut/AddOut and the two-bit DAC shown in Fig. 2.1 can be used to compensate for I_{offset}. For convenience we define the ADC input current as $I_{ADC} \equiv -I_{out}$. By taking all current sources into account we obtain the following expression for I_{ADC}:

$$I_{ADC} = G \times (I_{in} - I_{SubIn} + I_{AddIn} + I_{DAC}) + I_{SubOut} - I_{AddOut} - I_{offset}. \quad (2.1)$$

It is possible to adjust the settling time of the current receiver (τ) and the current gain $G \equiv Rf/Rs$ by changing Rf, Rs and Cf values. This is done by setting bits En30 - En120 (in DCDBv4Pipeline En90 and En120 are not connected and used), EnCap(0:3) and "HiGain" in the global shift register. It holds $\tau \sim Cf \times Rf$. In the case of Rf = 30 $k\Omega$ the maximum input dynamic range is 16 μA. For HiGain = 0 the output range is 16 μA (gain = 1) and for HiGain = 1 the output range is 32 μA (gain = 2).

The current consumption of the amplifier is in the range of 250 μA to 1 mA (can be adjusted by a global current-mode bias DACs).

2.2 Analog common mode compensation

The amplifier can be operated in the regular mode (as DCDBv1) and in the mode where the common mode noise is compensated by the analog feedback. Let us call this mode "analog common mode correction" - ACMC. If we use ACMC, we need to set the global bits ConAmpLow to zero and CMCCap to one. Additionally we need to set the local bit EnCMC to one. ConAmpLow = 0 disconnects the line AmpLowInt (source of the input transistor) from the power supply line AmpLowAmp. ConAmpLow = 1 connects the feedback capacitor Ccmc to stabilize the common mode feedback. EnCMC = 1 includes the corresponding amplifier to the common mode loop. This means the input of the corresponding amplifier is used to "calculate" the bias voltage VNSubInCM. (VNSubInCM is the arithmetic mean of all input voltages *in*. The summing of voltages is done using resistors RSubInCM.) Similarly the output of the amplifier is used to "calculate" the bias voltage VNCM. (VNCM is the arithmetic mean of all output voltages *out*. The summing of voltages is done using resistors RVNCM.) Notice that VNSubInCM and VNCM are gate voltages of the transistors TSubIn and TampLow. These voltages are shared by all channels. If we want to use ACMC but exclude some channels from the CMC network (because they are not connected to DEPFET), we will set EnCMC in these channels to zero.

2.3 ADC

Beside the current receiver, a channel contains one pipeline- current-mode ADC.

2.3.1 The principle of cyclic A/D conversion

The ADC uses redundant signed-digit (RSD) conversion. The algorithm starts with the comparison of the input signal with two thresholds, one positive and one negative. If the input signal is larger than the positive threshold, the pair of output code bits is set to 10, meaning +1, and a reference current is subtracted. If the input signal is lower than the negative threshold, the output code is set to 01 (-1) and the reference is added. If the input signal value is between the thresholds, the bits are set to 00 (0) and no arithmetical operation is carried out. The residue signal is multiplied by two and the result undergoes the same operation for the next bits. In is interesting to note that the conversion is not influenced by the comparator offsets, providing the offsets are not larger than half of the threshold.

2.3.2 Current-memory cell

Current-mode memory cells are used to implement the described A/D conversion algorithm.

The block diagram of the current-mode memory cell we use is shown in Fig. 2.3.

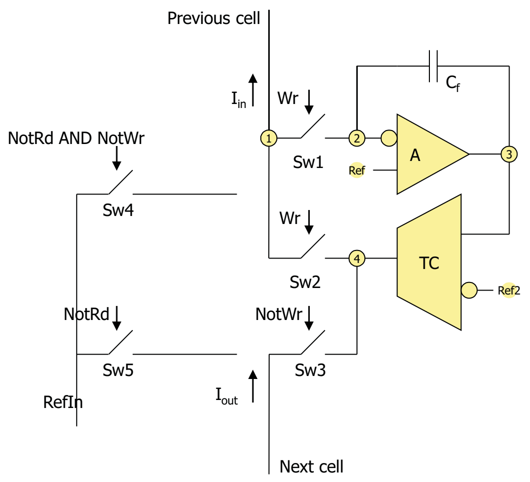

Figure 2.3: Current memory cell.

Capacitor C_f is used as the memory element. Transconductor TC is employed to convert the voltage across C_f into current. Switch Sw1 is used to freeze the voltage across C_f. Switch Sw2 is used to disconnect TC from the input when necessary. Switch Sw3 is used to connect the TC-current to the next cell during reading.

In the following analysis we assume that A has infinite gain and no offset.

In *write* state, Sw1 and Sw2 are closed. There is a negative feedback generated by A and TC. Potentials V_1 and V_2 are constant, signal independent, and equal to V_{Ref}: $V_1 = V_2 = V_{Ref}$. In the presence of input current, V_3 takes the value for which TC can drain the input current completely, so that no current flows into C_f.

Upon opening Sw1, the voltage across C_f remains *frozen*.

In *read* state, Sw1 and Sw2 are opened and Sw3 is closed. The copy of the input current I_{in} flows out of the cell.

The memory cell (a variant for the cyclic ADC) is explained in [1], it has the good feature that sampling errors caused by charge injection are signal-independent.

2.3.3 Current-mode realization of the pipeline ADC

The block diagram of the ADC is shown in Fig. 2.4. The ADC consists, actually, of 8 double cell blocks but for simplicity we show only first three. Figure 2.1 shows all 8 blocks with less details.

One double cell block contains two memory cells (MemCell1 and MemCell2) and two comparators (Comp Lo and Comp Hi). The comparators are connected to the voltage output (node 3 in Fig. 2.3) of the first memory cell. MemCell2 does not drive comparators. All memory cells have additional switchable current sources that can be used to add or to subtract the reference current.

The A/D conversion is performed in the following way:

- State 1 (takes 20 ns): The signals from previous cells are written in cells 1. (There is a difference in the first double block, which is explained later.) The comparators connected to these cells are in *reset* state.
- State 2 (takes 20 ns): The signals from previous cells are written in cells 2 The comparators connected to cells 1 are in *compare* state.
- State 3 (takes 20 ns): Comparators connected to cells 1 are *latched*, cells 1 and 2 are read out. Results of comparison (h, l) are used to decide about adding, subtracting or disabling the switchable reference sources in cells 1 and 2, according to the conversion algorithm. If the thresholds are properly chosen, the residue currents flowing out of the cells have half of the full signal range for an input signal within $\pm 2R$ (R is the reference current). In this case, the residuals can be summed and the sum fits into cells 3. Since residue currents are equal, the summing is equivalent to multiplication by two. Cells 3 is in write state, the comparators connected to this cell are in reset state.
- State 4 (takes 20ns): Sum of the residue currents is stored in cells 4. The comparators connected to cells 3 are in compare state.

The states 1 - 4 repeat. Notice that the first double block is controlled in slightly different way than the third (and seventh) double block. In State 1, both cells are written at the same time. In State 2 nothing happens. This has two consequences - the ADC takes only one sample within 20 ns. The input ADC range is doubled to 32 μA.

The comparators generate one digital output pair - (h_i, l_i) - when the corresponding double cell block is in read state. Notice that when one block generate valid outputs the next block generates invalid output 00. Therefore the digital outputs of two subsequent blocks can be merged by AND function. This is shown in Fig. 2.1. Additional time multiplex is performed (devices MUX in Fig. 2.1) to reduce the number of digital lines to 4 (2 x CompHi/Lo outputs) per ADC.

The bit stream (h_i, l_i) is the redundant binary representation of the input signal.

To evaluate final conversion result D, bit streams h_i and l_i must be subtracted: $D \equiv 2^7(h_7 - l_7) + ... + 2^0(h_0 - l_0)$. This is done in the digital readout block placed on the bottom of the chip.

An **8-cycle** quantization of the input signal gives as result a *signed* discrete number $D \times R/128$, where D takes all integer values between -255 and 255; in other words, D is the **8+1-bit** binary representation of the input current Iin. To simplify the digital data transfer, we neglect the LSB of D and transmit only 8-bit digital codes. Such a simplified code is within the range (-127, 127). We define a current flowing out of the ADC as positive. According to this, a current that flows into the current receiver generates a positive ADC current (see (2.1)). Increase of DEPEFT current leads to increase of the ADC codes.

Figure 2.5 shows the detailed transistor-level schematic of the used ADC- current-mode memory cell. Notice that switches Sw4 and Sw5 are used to short the internal nodes with RefIn when the cell is in idle state. Figure 2.6 shows the detailed transistor-level schematic of the comparator.

2.4 Digital sequence (decoder)

The analogue channel receives from the digital readout block the sequence signals: Sync(1:0), see Fig. 2.1. Out of them a decoder (Dec) generates signals Wr(4:0), Rd01 and Rd23 that are used to control the ADCs. Delay elements are used to assure proper timing of the delayed signals Wr* and

WrB*, shown in Fig. 2.5. Figure 2.7 show the signal waveforms. Sampling period is also indicated - it overlaps with the signal Wr(2). The section digital block describes the phase of the sampling period with respect to `SYNC_RESET`.

2.5 Calibration circuit

The schematic of the calibration circuit (Cal) is shown in Fig. 2.1. This circuit can apply test-signals and measure voltages or currents. The circuit can be connected either to the input of the amplifier or to the input of the ADC, which depends on the AmpOrADC bit polarity. This bit is stored in the global register. Following measurements can be conducted:

2.5.1 Fast current injection

One channel is selected by EnInjLoc = 1 and the injection is enabled at the chip level by setting of the global config bit EnInjGlobal = 1. **JTAG pad TDI is used as strobe.** If the strobe is one, the switchable current source (controlled by the global DAC VPInjSig) is enabled and its current flows into the amplifier or the ADC.

2.5.2 Slow current injection

One channel is selected by EnDC = 1 and the transistor connected to global line VDC is operated as cascode (global bit q(3) (pulldown VDC) equals 0 leading to VDC = 1.1 V). An external calibration current **or the global source ISigMirror** (explained in the table of global bits) can be sourced through the Monitor line into the amplifier or ADC.

2.5.3 Voltage measurements

One channel is selected by EnDC = 1 and the transistor connected to global line VDC is operated as switch (global bit q(3) (pulldown VDC) equals 1 leading to VDC = 0 V). Potential at the input of the amplifier or at the input of the ADC can be measured via Monitor line.

2.6 Pixel shift register

There are tree in-pixel configuration bits - EnDC, EnCMC and EnInjLoc per pixel. These bits are stored in pixel latches and are accessed by a pixel shift register (PSR) that has only one bit per channel, see Fig. 2.1. The pixel shift register can be written using JTAG that generates Ld, Ck and ShEn for this register. Since JTAG provides only one load signal for PSR, the 'target' for load is selected by setting of three global bits. E.g. if we want to load a bit into EnDC-latch, we set q(22) = 1 in the global shift register and then issue LdPSR using JTAG. Only Ld1 will be generated.

It is important to note that the clock of the pixel shift register is connected to **inverted** clock generated by JTAG to assure safer timing on chip. TDO is released on rising TCK edge. This is not according to JTAG standard. For more detailed description of JTAG see 5.3.2.

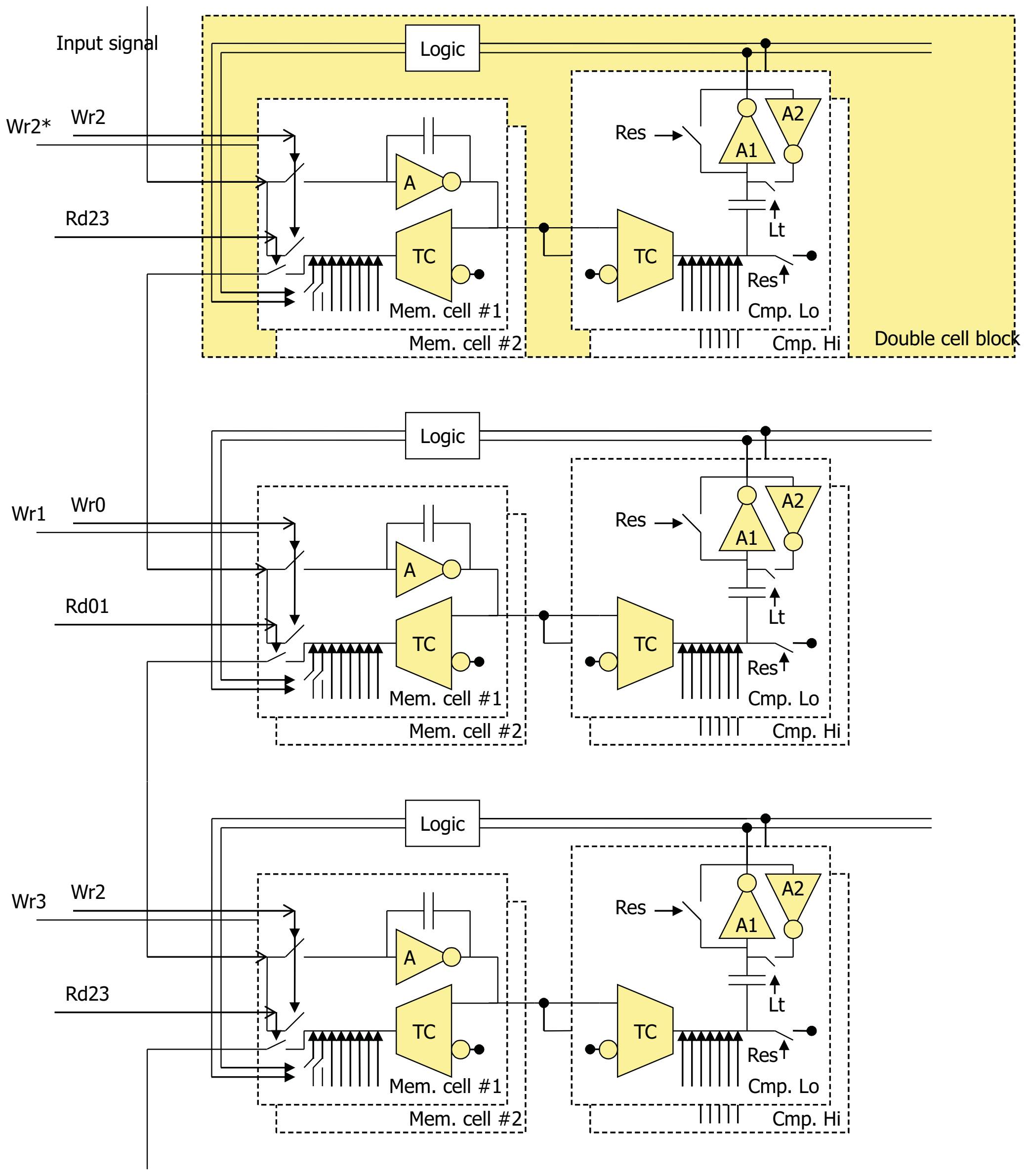

Figure 2.4: The block diagram of the ADC.

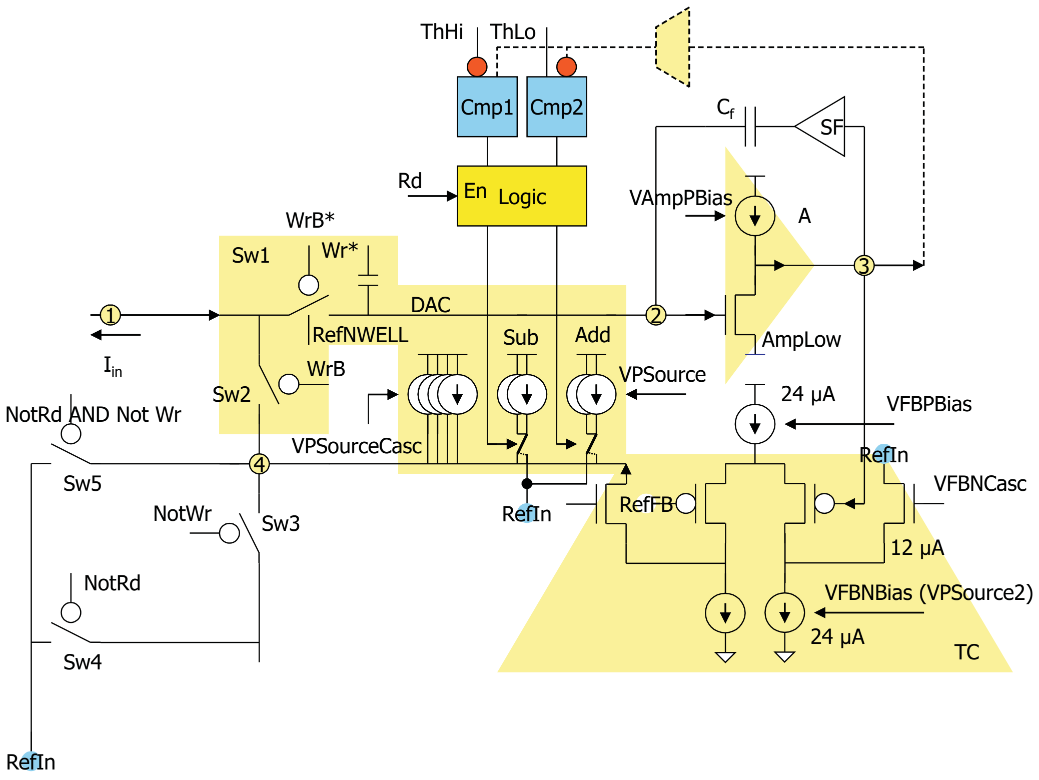

Figure 2.5: Transistor-level schematic of the current-memory cell used in the ADC.

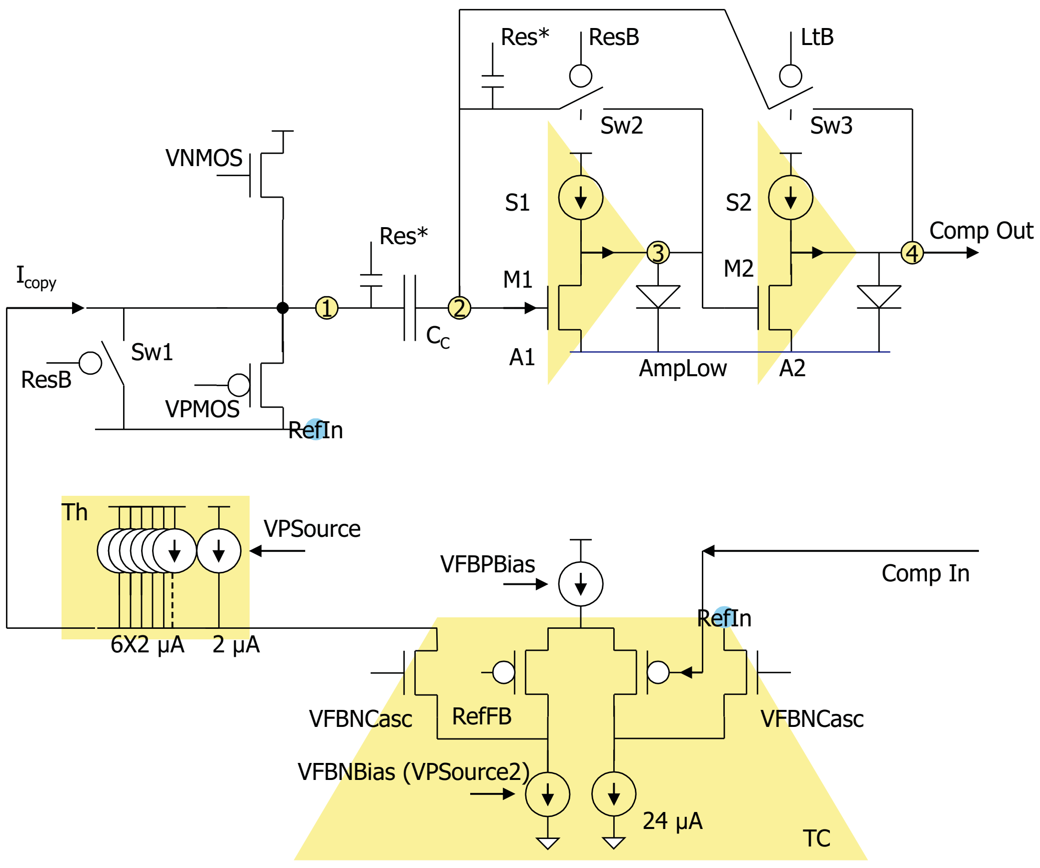

Figure 2.6: Transistor-level schematic of the comparator used in the ADC.

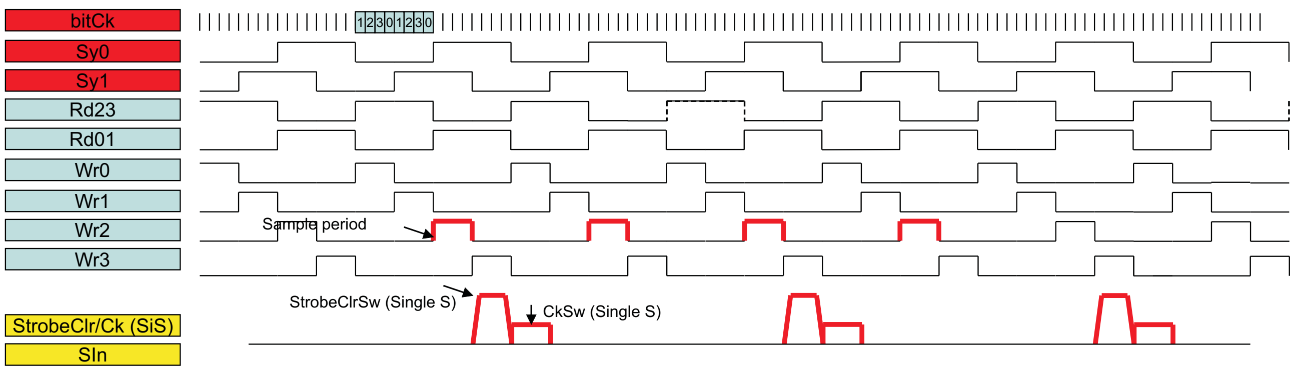

Figure 2.7: Timing diagram.

3 Global configuration block

DCD has 25 current-mode DACs (7-bits each) that generate bias voltages for the analogue channels. The current outputs of the DACs are connected to resistors or diode-connected transistors. The full scale DAC current is 10 μA. The DAC bits are stored in the global configuration register that can be accessed by JTAG. Beside the DAC bits, there are 40 other configuration bits in the register. The structure of the register is summarized in Table 5.3. For more detailed description of JTAG see 5.3.1.

The bias scheme for the global DACs is shown in Fig. 3.1. The DACs are based on switchable PMOS current sources. Their gates are connected to the bias voltage DACPBias. This voltage can be generated either by a simple bias-voltage generator (based on a resistor - as in DCDBv1-3) or by one of the two band-gap bias generators (band-gap references) - the simple band-gap (BG) bias generator and the standard BG bias generator. The choice is done by setting of the configuration bits.

Notice also, that DACs are generating currents, denoted as INxy. These currents are then converted into voltages VNxy, by connecting them to the diode-connected transistors. One such transistor is shown in Fig. 3.1 as an NMOS. The diode connected transistor can be also a PMOS. If the later case, an additional current mirror is attached to the DAC. We call all the PMOS-generated bias voltages VPxy, and the corresponding DAC currents IPxy. In the DCD's analog parts, VPxy bias voltages are used by their PMOS transistors and VNxy by the NMOS transistors. In the case when the resistive bias generator is used, DAC currents depend on AVDD and on temperature. In the case when the simple BG bias generator is used, DAC currents depend on AVDD and are (should be) temperature-independent. In the case when the standard BG bias generator is used, DAC currents are (should be) both temperature- and AVDD-independent.

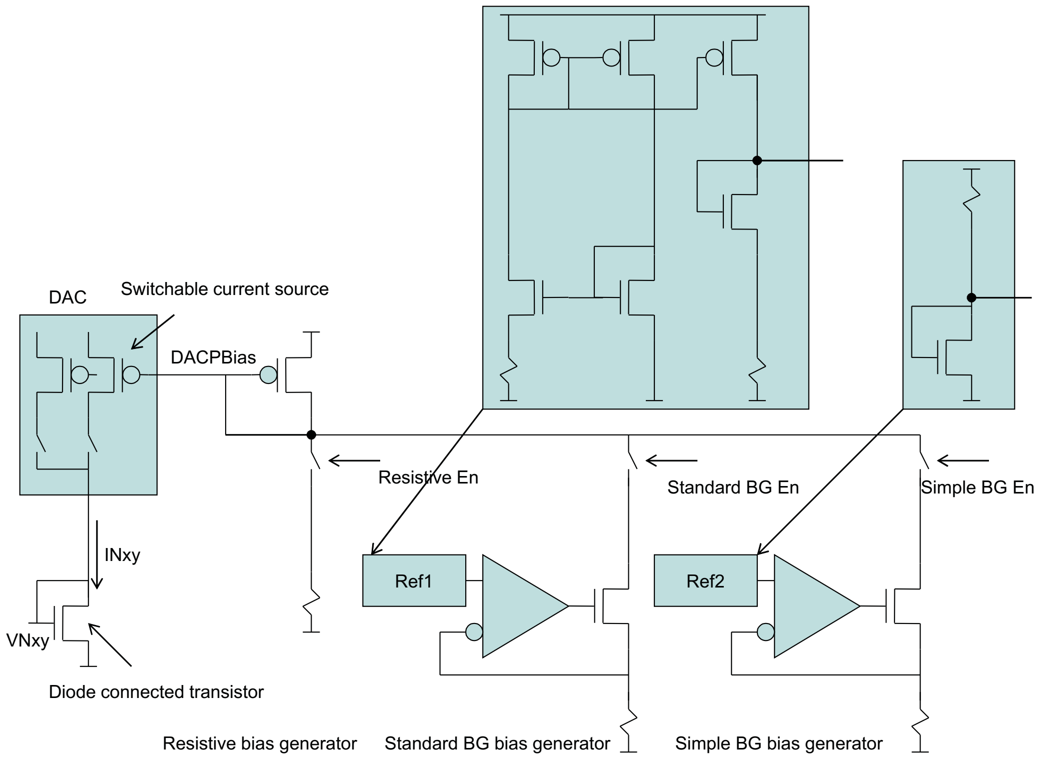

Figure 3.1: The bias scheme for the global DACs.

4 Digital Block

Besides the 256 ADC pixels, the DCDB consists of a large synthesized digital logic block. Its main tasks are generating control signals for the ADCs, converting the digitalized values coming from the ADCs in redundant binary representation (RBR) to standard binary format, and high speed data serialization. Furthermore, it implements a JTAG compatible interface for configuration purpose.
The following sub-sections describe the data read out scheme and the user interface to the ADC's pedestal subtraction. The configuration mechanism via the JTAG interface is focused by the next chapter.

4.1 Data Sampling and Read Out Scheme

The DCDB's 256 ADC pixels are organized in eight column pairs with 32 ADC pixels each. Every ADC pixel generates a byte of data within 32 clock cycles. So, the data produced by an entire column pair can be multiplexed to a single eight bit wide output bus. Since the DCDB consists of eight column pairs, there are eight such output buses. The timing behavior of all eight output buses is identical, therefore they are all commonly referenced by `DOX[7:0]` throughout this chapter.
The general DCDB read out scheme is shown in figure 4.1. After having complete the configuration, the data processing is controlled by only two signals: `CLK` and `SYNC_RESET`. At first, the clock (`CLK`) must be activated while holding `SYNC_RESET` high. Then, the DCDB starts data processing as soon as `SYNC_RESET` is released, here indicated in time step 1. Internally, the algorithmic ADCs start converting the input currents and the digital processing pipelines are filled with their output values. 191 clock cycles later (DCDv2: 171), the first valid byte of data can be read at the `DOX[7:0]` bus. From that cycle forth, every clock cycle brings a new valid byte of data. As shown in figure 4.1, the data shift out starts with the conversion result of ADC pixel number 0, then the conversion result of ADC pixel number 1 and so on up to number 31. Afterwards, results of the next conversion cycle are shifted out, beginning with ADC pixel number 0 up to number 31 and so on.
The DCDB's process flow is periodical with a length of 128 clock cycles. Therefore, pulling `SYNC_RESET` every 128th cycle does not interrupt the data flow.

The DCDB is generating the ADC control signals internally and synchronously to `SYNC_RESET`. Hence, the user does not have to care about these signals. However, for the user it is important to know when the ADCs sample the incoming currents. Therefore, figure 4.2 illustrates the relationship of `SYNC_RESET` and the ADC's sample window. With respect to the numbering convention of the `CLK` cycles between two consecutive `SYNC_RESET` pulses, the ADCs sample their input currents in the time windows: 18-25, 50-57, 82-89. The sampling moments are therefore on 25, 57, 89... which is the same as in DCDBv2.

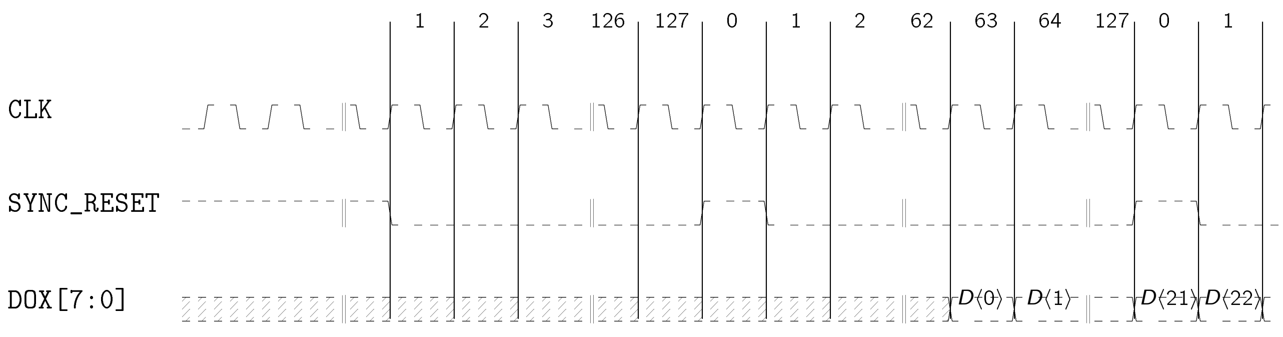

Figure 4.1: General data read out scheme

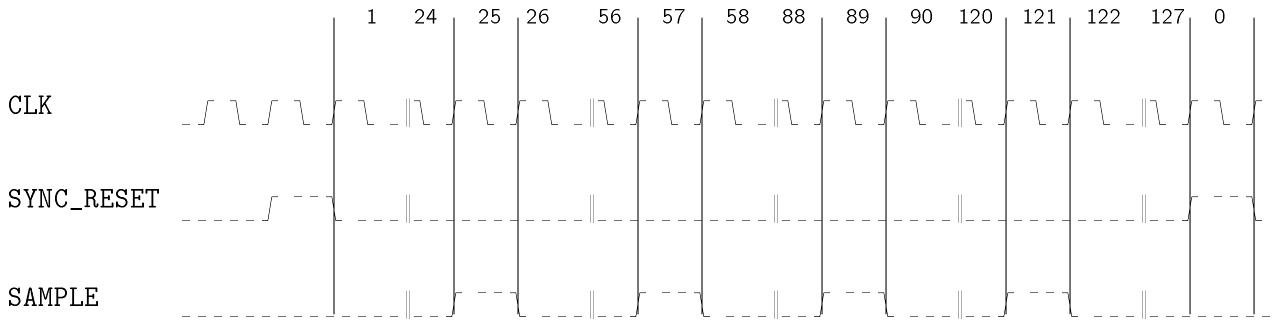

Figure 4.2: Relationship of SYNC_RESET and SAMPLE

4.2 Data Format

Each ADC conversion cycle results in exactly one byte of data, that is shifted out in parallel via the DOX[7:0] bus of the corresponding column pair. The value can be interpreted as signed and therefore represented by two's complement. The range is -127 to 127. A conversion value of 0 represents a zero current flowing out of DCDB's ADCs: $I_{ADC} = 0$, see (2.1). A larger ADC value represents a higher DEPFET current.

4.3 Controlling the Pedestal Correction

The DCDB's ADC pixels offer the possibility to dynamically compensate the offset input current before digitalization, see I_{DAC} on (2.1). I_{DAC} can be adjusted in every conversion cycle for every pixel individually. 4.1.

The technical realization is done by means of a two bit wide and 32 stages deep deserializer chain for

DIX[1:0]	I_{DAC}
2'b00	0
2'b01	IPDAC
2'b10	2×IPDAC
2'b11	3×IPDAC

Table 4.1: DAC for pedestal correction

every ADC pixel column pair. Simultaneously to reading out the conversion data of the entire column pair within 32 CLK cycles, the 32 offset current selections for the next conversion are shifted into the DCDB.
The column pair's pedestal correction input deserializer chain is accessed by means of the two bit wide input bus DIX[1:0]. Like with DOX[7:0], the timing behavior for the DIX[1:0] buses of all column pairs is identical. Figure 4.3 shows the order, in which the pedestal subtraction values are shifted in and the relation to SYNC_RESET. Beginning with clock cycle 22 (DCDBv2: 29), the pedestal subtraction values for ADC pixel number 31 down to 0 of the corresponding column pair must be given within 32 clock cycles. Then, starting with clock cycle 54 (DCDBv2: 61), the pedestal subtraction values for the next conversion cycle must be provided. The data that are send between clock cycles 22 and 53 will be valid at the DAC inputs between clock cycles 66 and 97.

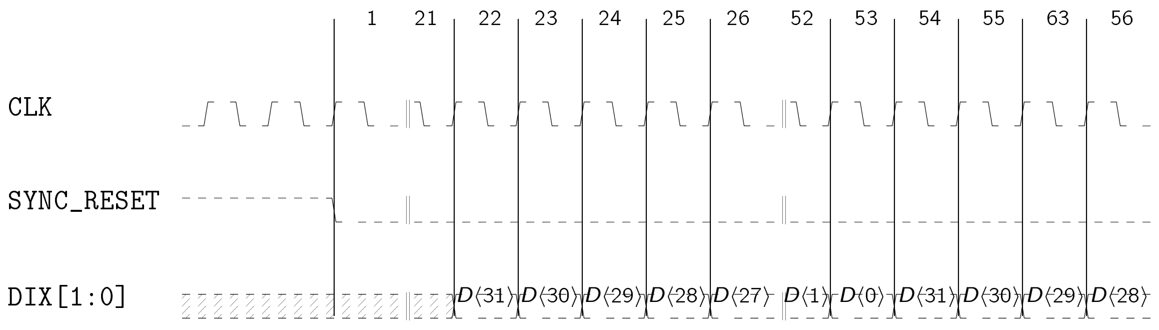

Figure 4.3: Pedestal correction input scheme

4.4 Digital test pattern

It is also possible that the DCD generates a digital test pattern. To enable this mode a configuration bit should be set (see JTAG chapter). The digital test pattern that corresponds to the ADC channel 0 is -127. It is generated as the signed binary number (MSB-LSB) 10000001. The digital test pattern that corresponds to the ADC channels 31 and 1 is 0. It is generated as the signed binary number 00000000. The digital test pattern that corresponds to all other channels is 127. It is generated as the signed binary number (DOX[7:0]) 01111111. Using such a pattern the position of the ADC channel "zero" in the byte stream can be determined. It is also possible to check whether all bits can change from 0 to 1 and vice versa within one clock period.

5 JTAG Configuration Interface

The DCDB offers a JTAG compliant interface for accessing both, the boundary scan chain and the internal configuration bits. This section gives some DCDB specific JTAG information. For a detailed and general description of JTAG and its operation mode, please refer to a more JTAG focused document.

5.1 General Information

The schematics of the JTAG block as implemented in DCD is shown in Fig. 5.1. The DCDB provides a JTAG interface for configuration and debugging purpose. It is accessible via the pins that are listed in table 5.1. The TAP controller (the state machine) of the DCDB's JTAG interface is sensitive to the rising edge of `TCK`. In this moment the TAP controller state machine executes the transition to its next state as indicated by `TMS`. `TRSTB` is the active low asynchronous reset pin that resets the TAP controller. Data bits are untouched by `TRSTB`. TMS controller state diagram is shown in Fig. 5.2.

Pin Name	**Purpose**
TCK	JTAG clock
TRSTB	Active low asynchronous reset
TMS	Mode Select for controlling the TAP controller
TDI	Data input pin
TDO	Data output pin

Table 5.1: JTAG Pin List

In order to access one of the various internal bit chains, it is necessary to first set the TAP controller instruction register accordingly. Table 5.2 lists the six valid instructions and their binary representation the TAP controller can deal with. The instruction register is written LSB first.

Instruction	**Binary Rep. [MSB:LSB]**	**Description**
EXTEST	4'b0000	Select boundary scan chain for external connection test
SAMPLE_PRELOAD	4'b0001	Select boundary scan chain for internal test
IDCODE	4'b0010	Read DCDB's ID code 32h12345678 (LSB goes to `TDO` first)
BYPASS	4'b1111	`TDI` is bypassed directly to `TDO`
SEL_GLOBAL_SR	4'b0100	Select global shift register chain
SEL_PIXEL_SR	4'b1000	Select pixel shift register chain

Table 5.2: JTAG Instruction List

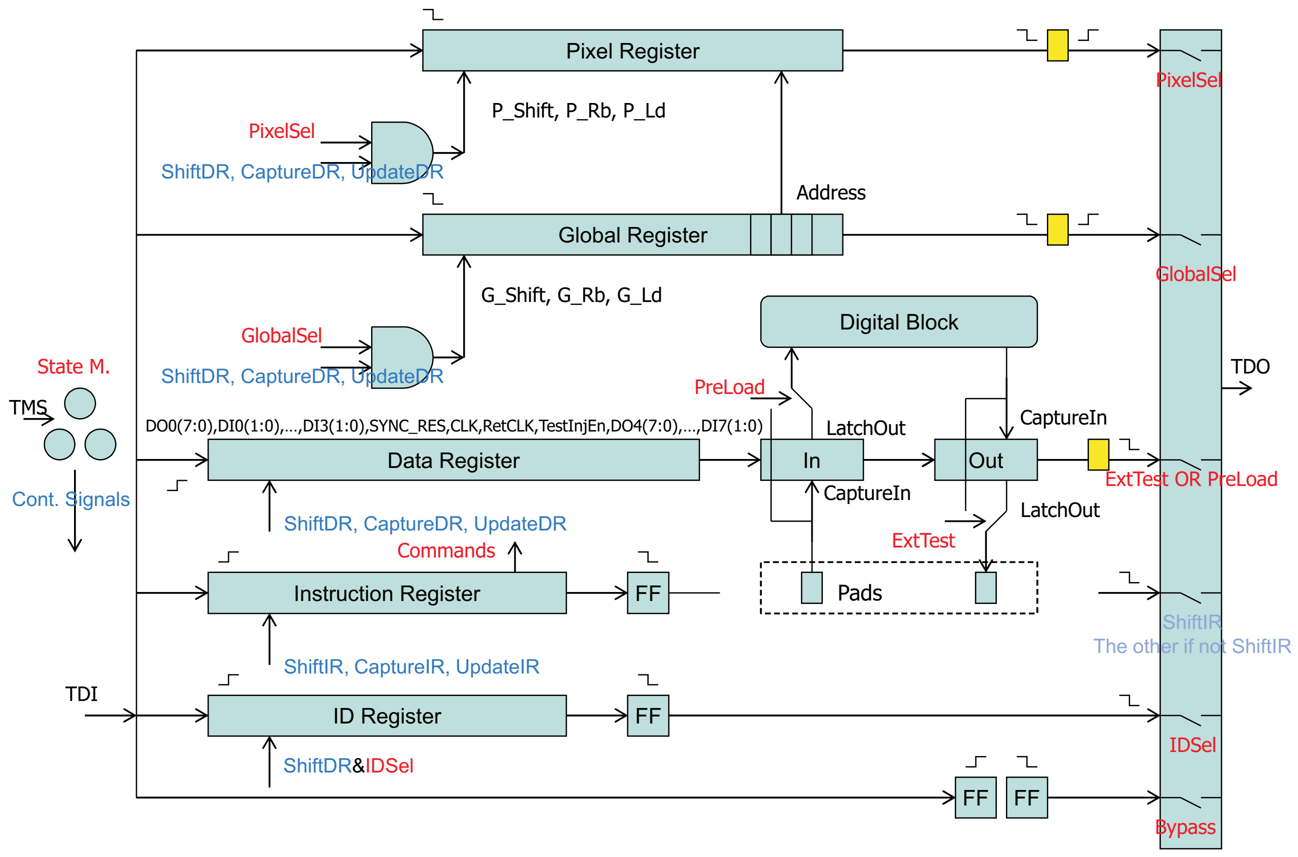

Figure 5.1: The schematics of the JTAG block.

5.2 EXTEST and PRELOAD

The EXTEST command allows 1) to "capture" (load) the states of the digital block outputs to the JTAG data register, 2) to shift out these bits to TDO while 3) the new bits are shifted in through TDI and finally 4) to send the bits that were shifted-in to the DCD's digital pads (such as DOX[7:0]). Notice that the bits received at TDO are generated by the DCD's digital block, while the bits seen on DCD's digital pads are coming from outside via TDI.

In this way we can check the functionality of the DCD using only JTAG pads and the pads CLK and SYNC_RESET.

We can also check the connection from DCD to DHP by sending a certain digital pattern to DOX[7:0] pads via JTAG. For illustration see Fig. 5.1.

The SAMPLE_PRELOAD command allows 1) to "capture" (load) the states of the digital input pads (such as DIX[1:0]) to the JTAG data register, 2) to shift out these bits to TDO while 3) the new bits are shifted in through TDI and finally 4) to send the bits that were shifted-in to the DCD's digital block inputs. Here, there is an exception - the fast digital block inputs CLK and SYNC_RESET are always connected to the corresponding pads. The CLK and SYNC_RESET pad-states can be captured but it is not possible to drive these digital block inputs via JTAG. This is not needed.

With SAMPLE_PRELOAD we can check the connection from DHP to DCD by capturing DIX[1:0] pad states via JTAG.

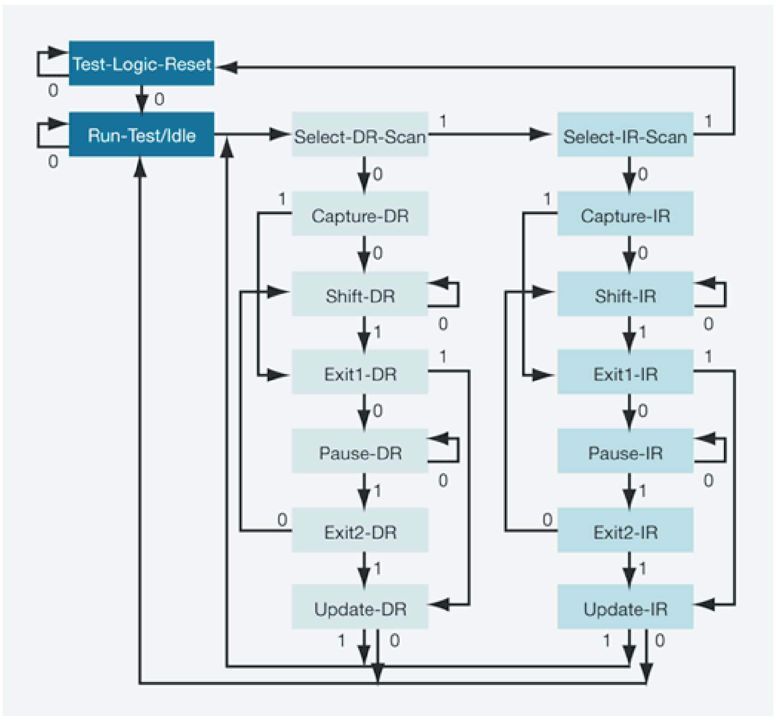

Figure 5.2: TMS controller state diagram. Taken from www.danafosmer.com test-engineering-articles.

Following signals are connected to the data register in the direction from `TDI` to `TDO`: DO0(7:0), DI0(1:0), DO1(7:0), DI1(1:0), DO2(7:0), DI2(1:0), DO3(7:0), DI3(1:0), `SYNC_RESET`, `CLK`, ReturnCLK, TestInjEn, DO4(7:0), DI4(1:0), DO5(7:0), DI5(1:0), DO6(7:0), DI6(1:0), DO7(7:0), DI7(1:0).

Among these signals the following two are not connected to DCD's pads: Signal ReturnCLK is generated by the digital block - this is the generated 100 MHz clock. Signal TestInjEn is the input of the digital block - TestInjEn is stored as a configuration bit in the global shift register.

5.3 DCDB Configuration

The DCDB is configured by writing its configuration registers. For obvious reasons, it is mandatory to configure the DCDB before trying to read out data.
This section describes the DCDB's configuration registers in details.

5.3.1 Global Chain (Global Shift Register)

The order of the 240 bits of the global configuration shift register is shown in table 5.3. It is written MSB first and accessed by previously writing "SEL_GLOBAL_SR" to the instruction register. Every cell of the global configuration shift register has an attached latch. The bits stored into the shift register cells (let us call them the "original bits") are written into the latches when state machine enters "UpdateDR" state (see Fig. 5.2). The transfer of bits from latches into shift register cells, that would be done in the "CaptureDR" state is not implemented. This means, during shifting-out (while simultaneously shifting-in the new values), we receive the "original bits" from the global shift register and not the bits that are copied into latches.

Notice the important thing: When the global shift register is written, the data from TDI are sampled at falling `TCK` edge (see Fig. 5.1). There is an additional flip flop attached to the global shift register output that samples data on rising TCK edge. The output of this flip flop is connected to `TDO`.

The full-scale DAC current is 10 μm. Full-scale DAC value is 127. DAC current is denoted with I.

Bit No	**Name**	**Purpose**	**Description**
[0:6]	DAC0	I/VPAddOut Negative offset current	Fig. 2.2 When increased decreases ADC value
[7]	Mon0	VPAddOut mon.	Connects the bias voltage to Monitor
[8:14]	DAC1	I/VFBPBias (DAC: 70) **Important!**	Fig. 2.5 TC positive current
[15]	Mon1	VRefNWell	Connects the bias voltage to Monitor
[16:22]	DAC2	I/VPSource (DAC: 70) **Important!** Influences ADC LSB granularity: value 70: LSB about 70 nA if high gain is used Increase to remove missing codes	Fig. 2.5
[23]	Mon2	AVDD top ADC	Connects the voltage to Monitor
[24:30]	DAC3	I/VPSource2 (DAC: 70) **Important!**	Fig. 2.5 TC negative current
[31]	Mon3	AVDD top TIA	Connects the voltage to Monitor
[32:38]	DAC4	I/VPDel (set at max)	Delays for ADC signals Wr* and WrB*
[39]	Mon4	AVDD bottom	Connects the voltage to Monitor
[40:46]	DAC5	I/VInjPSignal (DAC current x 4) **Important!**	Fig. 2.1 Switchable test-current source
[47]	Mon5	AVDD bottom	Connects the voltage to Monitor
[48:54]	DAC6	I/VPDAC (DAC current x 10) **Important!**	Fig. 2.1 Two-bit DAC LSB
[55]	Mon6	DACPBias	Connects the DAC bias voltage to Monitor
[56:62]	DAC7	I/VTCP (DAC: 30-40) **Important!**	Fig. 2.2 Amplifier bias current If increased ADC value decreases
[63]	Mon7	vddaTest (For yield testing - connected by many vias to AVDD)	Connects to Monitor
[64:70]	DAC8	I/VTCPL (DAC: 10-40) **Important!** Amplifier is faster for higher value)	Fig. 2.2 Amplifier load current
[71]	Mon8	I/VPMOS	Connects the bias voltage to Monitor
[72:78]	DAC9	V/IPSourceCasc (DAC: 64)	Fig. 2.5 Cascode voltage for CMC If DAC increased decreases

Bit No	Name	Purpose	Description
[79]	Mon9	I/VFBPBias	Connects the bias voltage to Monitor
[80:86]	DAC10	I/VFBNCasc (DAC: 0)	Fig. 2.5 Cascode voltage for CMC If DAC increased decreases
[87]	Mon10	not used	not used
[88: 94]	DAC11	I/VFBRef (DAC: 64) **Important!**	Fig. 2.5 Reference node for CMC If DAC increased decreases
[95]	Mon11	not used	not used
[96:102]	DAC12	I/VNMOS (DAC: 120)	Fig. 2.6 Controls negative voltage limit If DAC increased decreases
[103]	Mon12	not used	not used
[104:110]	DAC13	I/VTCCasc (DAC: 0) **Important!**	Fig. 2.2 Cascode voltage for amplifier If DAC increased decreases
[111]	Mon13	not used	not used
[112:118]	DAC14	I/VNSubIn **Important!** (DAC current x 24)	Fig. 2.2 If increased ADC value decreased Compensates DPFET pedestal Not used in ACMC mode
[119]	Mon14	not used	not used
[120:126]	DAC15	I/VNSubOut (DAC: 0)	Fig. 2.2 Positive offset if increased ADC value increases
[127]	Mon15	not used	not used
[128:134]	DAC16	I/VTCSFN (DAC: 120) **Important!**	Fig. 2.2 Amplifier output stage
[135]	Mon16	not used	not used
[136:142]	DAC17	I/VNDel (set at max)	Fig. 2.5 Delay settings for Wr* and WrB*
[143]	Mon17	not used	not used
[144:150]	DAC18	I/VPAddIn (DAC: 0 - use for tests)	Fig. 2.2 Emulates DEPFET pedestal
[151]	Mon18	not used	not used
[152:158]	DAC19	I/VRefNWell (DAC: 64)	Fig. 2.5
[159]	Mon19	not used	not used
[160:166]	DAC20	I/VAmpPBias (DAC: 30-60) **Important!**	Fig. 2.5 CMC amplifier bias If increased ADC value increases
[167]	Mon20	not used	not used
[168:174]	DAC21	I/VPMOS (DAC: 120)	Fig. 2.2 Comparator positive voltage limit If DAC increased increases
[175]	Mon21	not used	not used
[176:182]	DAC22	I/VSigMirror0 (the 3 ISigMirror DAC currents are added and they flow into monitor)	Fig. 2.1 Test current source

Bit No	**Name**	**Purpose**	**Description**
[183]	Mon22	not used	not used
[184:190]	DAC23	I/VSigMirror1	Test current source
[191]	Mon23	not used	not used
[192:198]	DAC24	I/VSigMirror2	Test current source
[199]	Mon24	not used	not used
[200:202]	q(0), q(1), q(2)	101 Enable resistive bias gen. (as in DCDBv2) 100 Enable band gap bias gen.	
[203]	q(3)	Pulldown for VDC	Enables voltage measurements in Fig. 2.1
[204]	q(4)	VNCM pulldown	If ACMC is used set to 0, else 1 (it connects VNCMC line to gnda if 1)
[205]	q(5)	1: Simple band gap bias gen. is used 0: Normal band gap bias gen. is used	
[206]	q(6)	Power on band gap if 0	Not needed set to 1
[207]	q(7)	Enables monitor capacitive filter if 0	Set to 1
[208:211]	q(8:11)	NU	
[212]	q(12)	Enables CMOS BitCLK input	Set to 1 if no LVDS is used
[213]	q(13)	AmpOrAdcGlobal	Fig. 2.1
[214]	q(14)	TestInjEn	Enables test pattern injection into the DCDB's digital logic block for debugging purpose
[215:216]	q(15:16)	TestSync(0:1)	Static Sync(0:1) values (for debug mode)
[217:218]	q(17:18)	TestSampleEn(L:R)	Static SampleEn(L:R) values (for debug mode) Not used DCDBPipeline
[219]	q(19)	EnableInjGlobal	Enables injection - TDI used then as Inj Fig. 2.1
[220]	q(20)	ExternalStrobeEn	Enables external Strobe Not used DCDBPipeline
[221]	q(21)	EnableTest	Enables debug mode
[222]	q(22)	Enables load for EnableDC	Enables DC measurements (Fig. 2.1)
[223]	q(23)	Enables load for `EnCMC`	Pixel is in CMC mode (Fig. 2.1)
[224]	q(24)	Enables load for EnableInjLocal	Enables injection in pixels (Fig. 2.1)
[225]	q(25)	On	Enables the amplifier (Fig. 2.2)
[226:229]	q(26:29)	En30, En60, En90, En120 (in DCDBv4Pipeline only 30/60 used)	gain control (Fig. 2.2)
[230:233]	q(30:33)	EnCap(0:3); EnCap(1) Stability cap	Bandwidth control (Fig. 2.2)
[234:237]	q(34:37)	(34): High gain (set to 1), (35): NU, (36): CMC cap (if ACMC is used set to one), (37): ConnAmpLow (if ACMC is used set to zero)	Fig. 2.2

Bit No	Name	Purpose	Description
[238]	q(38)	EnDoubleSamplingB	Zero enables double sampling mode (Fig. 2.1)
[239]	q(39)	EnDoubleSampling	One enables double sampling mode (Fig. 2.1)

Table 5.3: Global shift register

5.3.2 Pixel Chain

In addition to the configuration bits of the global chain, each of the DCDB's 256 ADC pixels has three more individual configuration bits, called `EnDC`, `EnCMC` and `EnInjLoc` (refer Fig. 2.1). These 256x3 bits are accessible via the pixel shift register (pixel chain). For implementation reasons, these bits are not lined up to a single 768 bit long chain. Instead, the pixel chain is only 256 bits long and after shifting these bits in, they can be stored into three different locations: Either the pixel chain content represents the `EnDC`, or the `EnCMC`, or the `EnInjLoc` bits of all 256 ADC pixels. This selection has to be done before writing the pixel chain by setting the bits 222 to 224 of the global chain accordingly.
Like the global chain, the pixel chain is written MSB first (here that means: at first, write corresponding bit of ADC pixel 256 ([31][7]), then, write corresponding bit of ADC pixel 255 ([30][7]), and so on). During shifting out (while simultaneously shifting in new values), we receive the "original bits" from the global shift register and not the bits that are copied into latches. Important: The data from `TDI` are sampled at falling `TCK` edge when the pixel shift register is written. There is an additional flip flop attached to the global shift register that samples the input on rising `TCK` edge. The output of this flip flop is connected to `TDO`.

5.3.3 Register Writing with DHP

The fact that DCD's global and pixel registers sample `TDI` at falling `TCK` edge may lead to errors in shifting of bits from DHP to DCD if `TCK` falling edge arrives on the input of DCD after `TDI` change. This can happen, since DHP changes output on falling TCK as well and there is a large capacitive load on `TCK` connected to Switchers. To cope with this, DHP chip offers the possibility to invert `TCK` going to DCD. Using this feature we can make writing into DCD's global and pixel register safe. This section explains the procedure of writing into DCD global register.

Writing into DCD global register

We assume that we have a module with 4 DHPs, 4 DCDs and 6 Switchers. DCDi receives JTAG signals from DHPi. We would like to write into DCDi.

1 Step:

Put all chips by a `TMS` sequence into ShiftIR state (see Fig. 5.2).

Write into instruction registers of all chips the following commands: DCDi SelGlobalSR, DHPi JTAG-GlobalRegister, all other chips Bypass.

2 Step:

Put the chips by `TMS` sequence into ShiftDR state. The global JTAG register of DHPi, the global shift register of DCDi and the bypass registers of all other chips form a long shift register.

Shift in the proper bits into the global JTAG register of DHPi. Set the bit "DcdInvertTCKPolarity" to active. Go through UpdateDR state by `TMS`.

Now the `TCK` going from DHPi to DCDi is inverted. Before inversion the bits entering DCDi may be sampled wrongly, but now the bit transfer is safe.

Notice that after `TCK` inversion, the state machine of DCDi captures TMS at the falling edge of the original `TCK` (see Fig. 5.3). Moreover, the `TCK` active edge received by DCDi will arrive 1/2 `TCK` period later (delay 1 in Fig. 5.3). To assure proper and synchronous operation of the state machines, `TMS` should always be changed in the middle of original `TCK` low period, as shown in the figure, and not on `TCK` edges.

Step 3:

Put again all chips into ShiftDR state. Shift in the proper bits into the JTAG register of DHPi and the proper bits into the global register of DCDi . Set the DHP bit "DcdInvertTCKPolarity" to zero. Come into UpdateDR state. After the DHP's shadow register is written, `TCK` going to DCD has again the normal polarity. The return to normal polarity will cause an additional delay of the active `TCK` edge of 1/2 `TCK` period. This means that the DCD state machine is now delayed by one `TCK` period versus the state machine of the other chips (delay 2). To synchronize the chips again, keep `TMS` high after UpdateDR state until all state machines end up in Reset state.

In this way it is also assured that the DCD state machine goes through UpdateDR state and that the content of the global register is stored into the latches.

Writing into DCD pixel register

The DCD's pixel register is written in similar way. The only difference is that SelPixelSR command is used instead SelGlobalSR.

Writing of DCD pixel latches

Writing of pixel latches requires more cycles.

Let us assume that we want to write a bit-pattern into `EnDC` latches.

As first, we need to write the bit-pattern into the pixel register. This is done as explained above.

Then we need to write the global register with a correct pattern - the bit controlling load into `EnDC` (q(22)) is one. The content of the pixel register is now copied to the pixel latches. Finally, we need to write the global register with the pattern where q(22) is zero. By doing this we assure that the pixel latches will not be overwritten when we shift a new pattern through the pixel shift register.

The same procedure should be repeated for `EnCMC` and `EnInjLoc`.

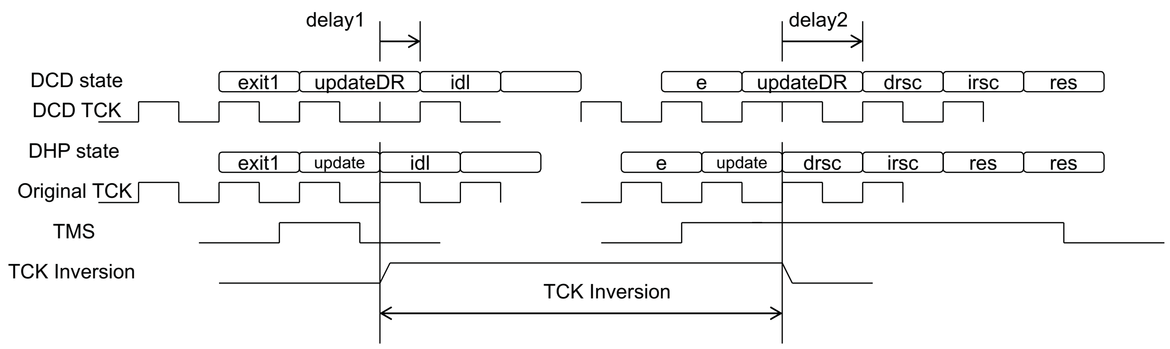

Figure 5.3: Inversion of `TCK`

List of Figures

List of Tables

Bibliography

[1] Ivan Peric, Tim Armbruster, Manuel Koch, Christian Kreidl, Peter Fischer, 'DCD - the Multi-Channel Current-Mode ADC Chip for the Readout of DEPFET Pixel Detectors'.