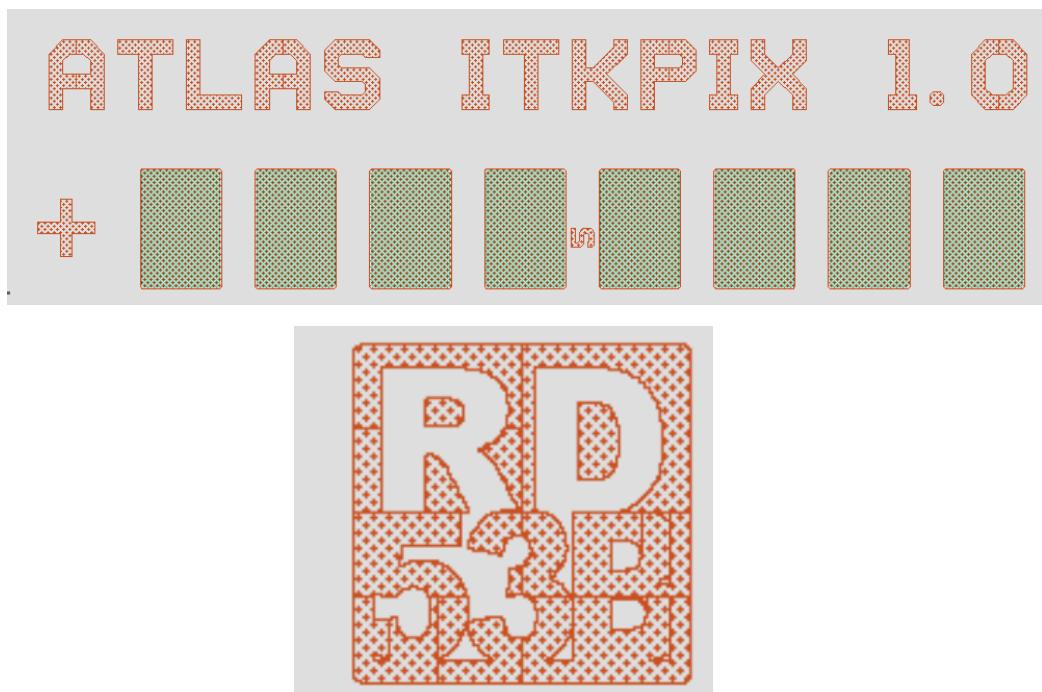


**CERN-RD53-PUB-xxxx**  
**Version 2.21, June 26, 2022**

## The RD53B-ATLAS Pixel Readout Chip Manual

---

ABSTRACT: Manual for the RD53B design in the ATLAS chip implementation.



---

## Contents

<b>1. Overview</b>	<b>2</b>
<b>2. Dimensions, Floorplan and Pads</b>	<b>4</b>
2.1 Bump Bond Pads	6
10 2.2 Wire Bond Pads	6
<b>3. Basic Operation and Reset (Intro and Quick Start Guide)</b>	<b>8</b>
3.1 Chip Startup	8
3.2 Reset	10
3.2.1 Chip Default Configuration	11
15 3.2.2 Command Activity Detector	13
<b>4. Power and References</b>	<b>14</b>
4.1 Shunt LDO Regulator	14
4.2 References and Startup	16
4.2.1 Offset Voltage Start-up	17
20 4.3 Offset Voltage and Low Power Mode	17
4.4 Under-shunt Current Protection	19
4.5 Over-voltage Protection	20
<b>5. Analog Front End</b>	<b>22</b>
5.1 Front End Bias Generation and Distribution	22
25 5.2 ATLAS Analog Front End	23
<b>6. Calibration Injection</b>	<b>28</b>
6.1 Generation of S0 and S1 signals	29
6.2 Cal Command	30
6.3 Injection Voltages	32
<b>30 7. Digital Core</b>	<b>34</b>
7.1 4-Pixel Region	34
7.2 Pixel Hit Logic	35
7.3 ToT counter and storage	35
7.4 Latency, Trigger and Readout (LTR) block	37
35 7.5 Pixel Addressing	39
7.5.1 $25\text{ }\mu\text{m} \times 100\text{ }\mu\text{m}$ pixels	39

<b>8. Commands and Configuration</b>	<b>41</b>
8.1 Receiver Circuit	41
8.2 Command Protocol	42
8.2.1 Short Commands	43
8.2.2 Long Commands	44
8.3 Command Protocol Initialization	45
8.4 Command Protocol Transmission	46
8.5 Command Protocol Decoding	46
8.6 Command Protocol Timing	47
8.7 Global Configuration	48
8.8 Pixel Configuration	48
<b>9. Trigger Processing, Tags, and Data Flow</b>	<b>51</b>
9.1 Bunch crossing ID selection and latency	53
50   9.2 Tags	53
9.3 Trigger Book Keeping	54
9.4 Self Trigger Source	57
9.5 Data Flow	58
<b>10. Data Output</b>	<b>61</b>
55   10.1 Data Output Drivers	61
10.2 Aurora and RD53B Data	63
10.3 Aurora and streams	65
10.4 Hit data encoding	66
10.5 Stream construction and efficiency	68
60   10.6 Hit map construction	68
10.7 Multi-chip encoding	70
10.8 Event size limit and data filtering	71
10.9 Precision ToT data	72
10.10 Format Options	74
<b>65 11. Multi-Chip Data Aggregation</b>	<b>75</b>
11.1 Data Receivers	75
11.2 Setup and Operation	75
<b>12. Sensing and Monitoring Functions</b>	<b>78</b>
12.1 Analog Multiplexer (MUX)	79
12.1.1 Multiplexer Configuration	79
70   12.2 General Purpose ADC	79
12.2.1 12-bit DAC	80
12.2.2 ADC comparator	81
12.2.3 ADC conversion timing	82
12.2.4 ADC Configuration	82
12.2.5 ADC Control Sequence	83

12.3	Transistor-based Temperature and Radiation Sensors	83
12.3.1	Transistor Sensor Theory	83
12.3.2	Precision Biases	84
80	12.3.3 Measurement Approaches	84
12.4	Resistive Temperature Sensors	85
12.5	Sensor Configuration	86
<b>13.</b>	<b>Test and Miscellaneous Functions</b>	<b>87</b>
85	13.1 General purpose LVDS and CMOS outputs	87
13.2	Bypass mode	87
13.3	Scan Chains	87
13.4	Hit OR	87
13.5	Heartbeat and test patterns	87
13.6	Ring Oscillators	87
90	13.7 Precision ToT module	89
13.8	Capmeasure circuit	90
<b>14.</b>	<b>Clock Generation and Data Recovery Technical Details</b>	<b>93</b>
95	14.1 I/O interface, data and clock distribution	93
14.2	Clock Data Recovery	93
14.3	Startup	94
14.4	Configuration	95
14.5	Bypass mode	95
14.6	Fine delay circuit (phase shifters)	96
14.7	Data serializer	97
<b>100</b>	<b>15. Reference Tables (pinouts, configuration, etc.)</b>	<b>99</b>
15.1	Wire Bonding Pinout	99
15.2	Global Configuration	101
15.3	Global Pulse Routing	106
15.4	Pixel Configuration	106
105	15.5 IMUX and VMUX selection values	107
15.6	General Purpose LVDS and CMOS Output Assignments	108
15.7	Internal and External Component Nominal Values	109
15.8	Command and Trigger Encoding	110
15.9	Output Data Encoding	112
110	15.10 ToT Tables	114
15.11	Ring Oscillator Assignments	116
<b>A.</b>	<b>Aurora 64b66b Technical Reference</b>	<b>117</b>

<b>B. Design for Test (DFT)</b>	<b>118</b>
B.1 Test signals	118
115     B.2 Chip Bottom testing	119
B.3 Pixel Matrix test	121
B.4 Testpoint description	122
B.5 Matrix test procedure	123
<b>C. Known Issues with RD53B ATLAS</b>	<b>124</b>

## 1. Overview

This manual provides a technical description of the RD53B chip design and operation adequate for simulation, testing, debugging and DAQ development. A basic familiarity with pixel systems and readout chips is assumed. A more general introduction explaining the basic functions and principles can be found in [6].

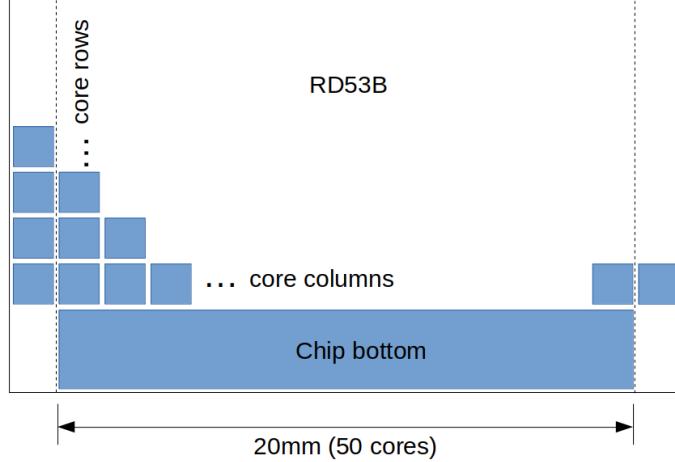
The readout chips for the ATLAS and CMS HL-LHC pixel detectors are two separate instances of a common design framework called RD53B. The main differences between ATLAS and CMS are the size of the pixel matrix and the pixel analog front end. There are other differences partly stemming from the sequential fabrication: RD53B-ATLAS in 2020 and RD53B-CMS in 2021. RD53B-CMS is thus a newer version of the RD53B framework with some bug fixes and added features relative to RD53B-ATLAS. For convenience this manual is compiled in two separate versions, RD53B-ATLAS and RD53B-CMS. This is the RD53B-ATLAS version. Both versions use the same revision number as most of the elements are common. RD53B is an evolution of the RD53A integrated circuit [1] to incorporate all production requirements defined by the experiments [2].

RD53B is a pixel readout chip framework that can be instantiated into different size physical chips. The design work and much of the verification are largely independent of the final instantiated size. RD53B consists of a *pixel matrix* and a *chip bottom*. The pixel matrix is built up of identical 8 by 8 pixel *cores* stepped and repeated in columns and rows. A core is physically 400  $\mu\text{m}$  by 400  $\mu\text{m}$ . The selected numbers of core columns and rows determine the chip size. The chip bottom contains all the system functionality and should be viewed as a fixed element that does not depend on matrix size. A physical chip, therefore, cannot be *narrower* than 20 mm (50 cores), because that is the width of the unique wire bonding pad frame in the chip bottom, but it can be wider. The *height* (number of core rows) is not constrained by the chip bottom, but is limited to a maximum of 50 by power and bias distribution as well as readout timing. This high level organization concept is shown in Fig. 1. The instantiated dimensions are detailed in Sec. 2.

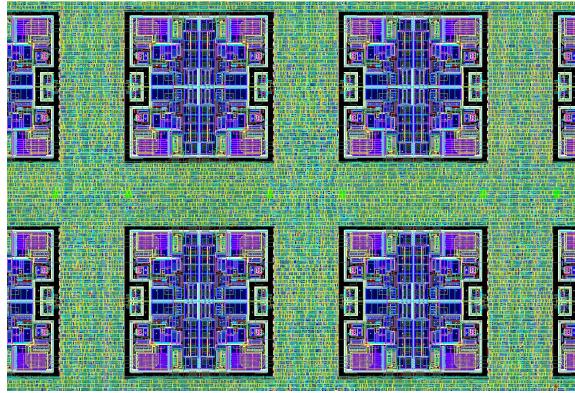
The core contains 64 pixel front ends organized in 16 identical so-called *analog islands* with 4 fronts ends each, which are embedded in a flat digital synthesized “sea” as shown in Fig. 2. The analog front end and island design are described in Sec. 5. The digital core design is described in Sec. 7. The pixel matrix is produced by stepping and repeating cores, which also takes care of the distribution of analog biases, as described in Sec 5.1.

The chip bottom contains all system functionality and the wire bond pads. RD53B is a system-on-chip including power management, sophisticated digital communication, sensing and monitoring. An overview of the basic operation, including a description of the reset scheme, are given in Sec 3. Sec 3 also serves as an introduction to the more detailed content of other sections. All tabular information, including pinout and configuration register values, is collected in the Reference section (Sec 15).

Power management, including design of the Shunt-LDO regulators are covered in Sec 4. The command and control interface (how one talks to the chip) and the configuration are covered in Sec 8. The data output (what comes out of the chip), including special (non-hit data) and the aggregation of data from multiple chips, are described in Sec. 10. The sensing and monitoring functions are described in Sec 12. Test features and miscellaneous functions are covered in Sec 13.



**Figure 1:** Conceptual depiction of RD53B framework, with a matrix composed of 50 or more columns by up to 50 rows of identical cores, and a fixed chip bottom. The dashed lines indicate the minimum width of 50 cores.



**Figure 2:** Layout view of analog islands within synthesized logic. Four complete islands can be seen in the center of the figure. One core contains four by four analog islands.

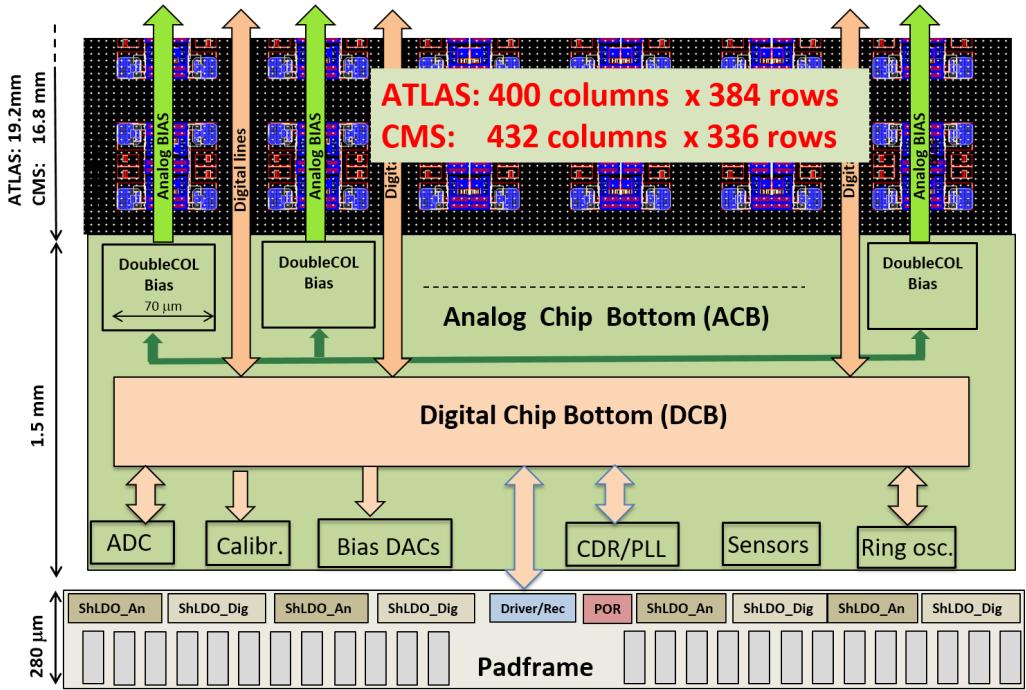
The designs of the bump bond and wire bond pads are covered in Sec. 2.1 and 2.2. RD53B only has wire bond pads along the bottom edge, to make it 3-side abuttable.

Parameter	ATLAS	CMS
Pixel bump pitch	$50 \mu\text{m} \times 50 \mu\text{m}$	
pixel rows (H)	384	336
pixel columns (W)	400	432
core rows	48	42
core columns	50	54
Chip width (including seal ring)	20.054 mm	21.654 mm
Chip height (including seal ring)	21.022 mm	18.622 mm

**Table 1:** Chip sizes for ATLAS and CMS chips.

## 165 2. Dimensions, Floorplan and Pads

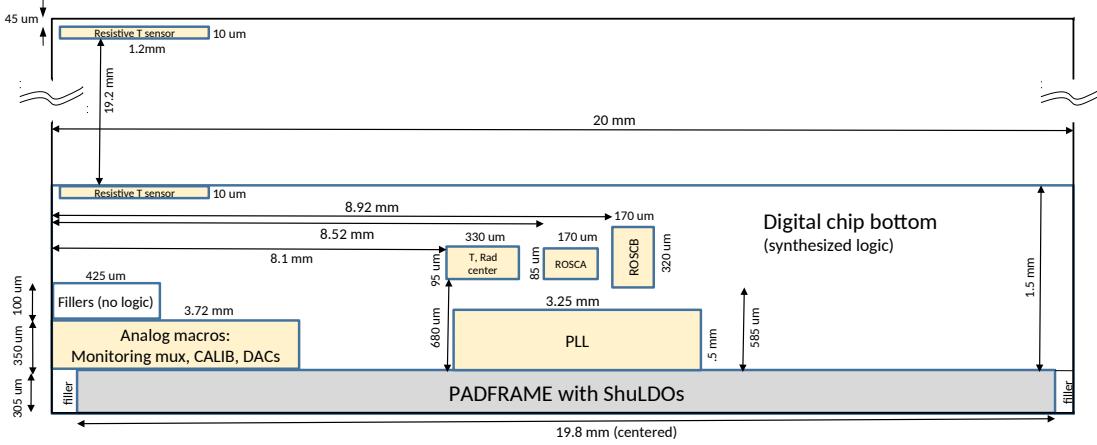
RD53B uses a 9 metal layer stack, consisting of 7 thin, 1 thick and 1 ultra-thick metal layers. In addition, the 28K AP layer is also used for power lines distribution. In Fig. 3 the layout and functional view of RD53B floorplan are shown. The sensitive area of the chip is placed at the top of the chip and is arranged as a matrix of pixels of  $50 \mu\text{m} \times 50 \mu\text{m}$  according to Table 1. The peripheral circuitry is placed at the bottom of the chip and contains all global analog and digital circuitry needed to bias, configure, monitor and readout the chip. The wire bonding pads are organized as a single row at the bottom chip edge and are separated from the first row of bumps by 1.7 mm in order to allow for wire bonding after sensor flip-chip (Sec. 2.2). The pad size is larger than in RD53A and no longer guaranteed to be compatible with Thru-Silicon Via post processing.



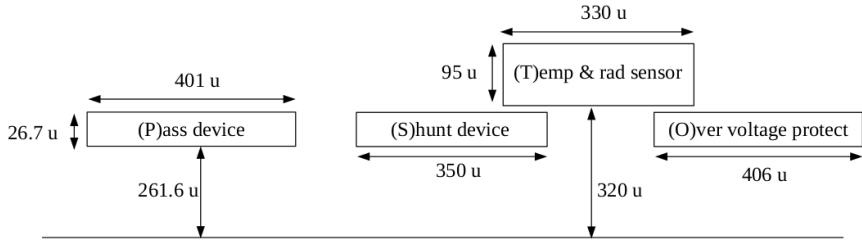
**Figure 3:** RD53B floorplan, functional view.

175 In the chip periphery, all the analog building blocks are grouped in a macroblock called Ana-

log Chip Bottom (ACB), which is fully assembled and characterized in an analog environment. The ACB block is surrounded by a synthesized block, called Digital Chip Bottom (DCB), which implements the Input, Output and Configuration digital logic.



**Figure 4:** Size and location of elements in the ATLAS chip bottom and top (Not to scale). The outline is the chip seal ring (tightest possible diced edge).



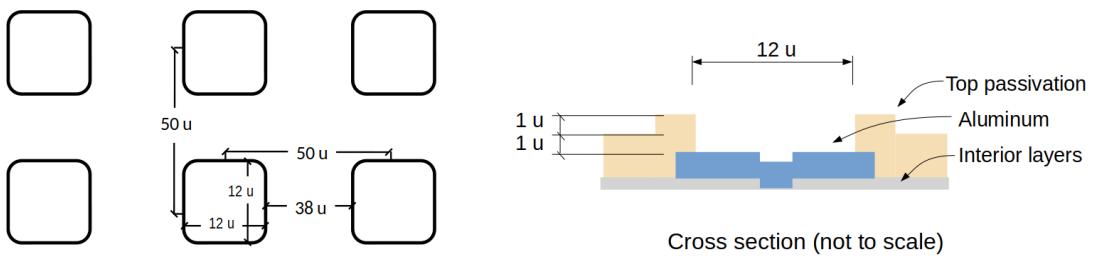
**Figure 5:** Size and vertical position of power devices relative to the chip seal ring (tightest possible diced edge). Horizontal placement is given in Table 2

Device	AO	AS	AP	DP	DS	DO	AO	AS	AP	DP	DS	DO
Left edge (um)	929	1346	1690	2709	3114	3465	6129	6546	6890	7909	8314	8665
<b>Device</b>												
Left edge (um)							AT	DT				
<b>Device</b>												
Left edge (um)							7125	7825				
Device	AO	AS	AP	DP	DS	DO	AO	AS	AP	DP	DS	DO
Left edge (um)	11329	11746	12090	13109	13514	13865	16629	17046	17390	18409	18814	19165

**Table 2:** Companion table to Fig. 5 showing the position of the left edge of power devices in ATLAS chip: A=Analog, D=Digital, O=Over voltage protection, P=Pass device, S=Shunt device, T= Temperature and radiation sensor. The devices are arranged in four groups (delimited by double lines) as can be seen in Fig. 8.

## 2.1 Bump Bond Pads

The bump bonds pads are defined by a regular pattern of openings in the passivation as shown in Fig. 6 (left). The alignment of aluminum metal shape under each passivation opening can vary by up to  $1\text{ }\mu\text{m}$  from pixel to pixel, but as the shapes are bigger than the opening there is always exposed aluminum for the entire pad. The bump bond pads do not have ESD protection. The passivation opening is square with 45 degree corners, which will appear rounded in the as-built chip. Fig. 6 (right) shows the expected height profile across the center of a bump pad as derived from the metal stack. Aluminum metal is exposed in the  $12\text{ }\mu\text{m}$  passivation opening and extends below the passivation beyond the opening, resulting in a passivation ridge surrounding the opening, as shown. The exposed metal may not be completely flat: it can have depressions less than  $1\text{ }\mu\text{m}$  deep due to vias below. The figure shows such a depression.



**Figure 6:** Bump bond pad dimensions. Matrix layout on the left and cross section on the right.

## 2.2 Wire Bond Pads

The wire bond pads are along the chip bottom on a  $100\text{ }\mu\text{m}$  pitch. The pad area is larger than in the RD53A chip to meet production requirements (Fig. 7). This RD53B pad size is not guaranteed compatible with thru-silicon via post-processing.

There are 198 pads, 4 of which are not used and not connected to any internal net. The location of these unused pads was chosen to eliminate wire bonding tool interference at the edges of fanout regions. Most pads are for power and ground and are grouped strategically for PCB/module layout as shown in Fig. 8. The detailed pinout is given in Sec. 15.1.

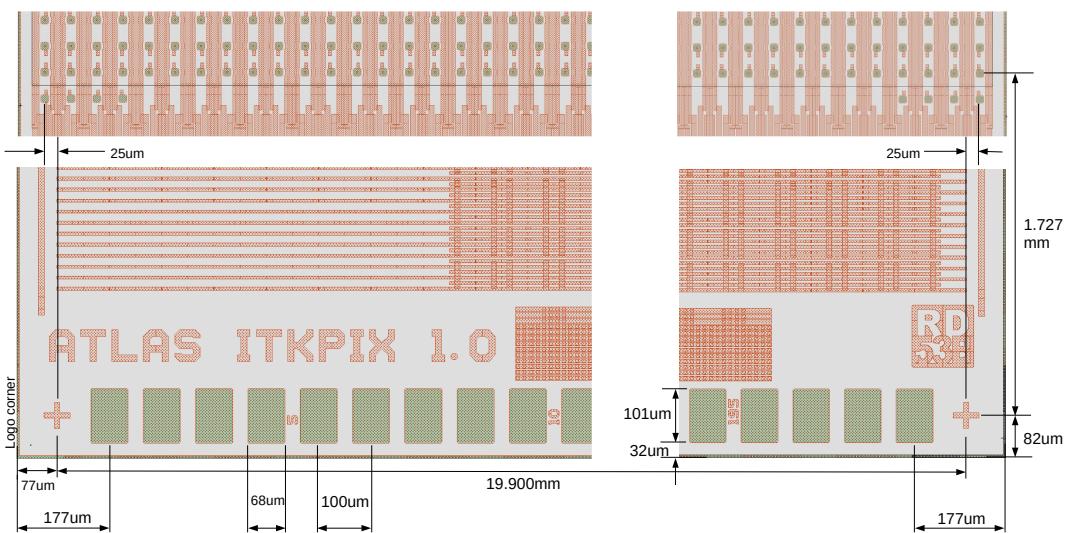
The wire bond pads have visible numbering on the chip (the numbers label the pads to their right), and are flanked by alignment marks, as can be seen in Fig. 7.

The RD53B chip has internally four separate power domains:

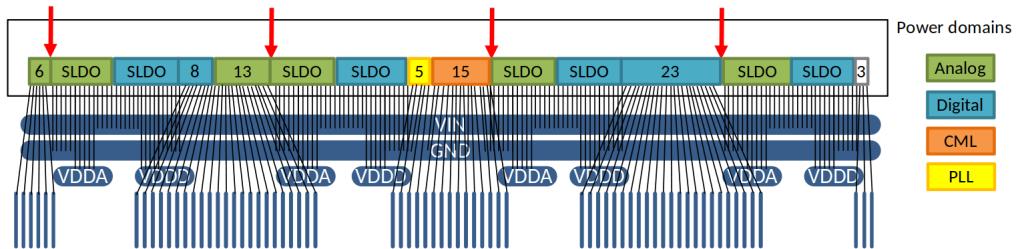
- Analog: VDDA, GNDA
- Digital: VDDD, GNDD
- PLL (PLL/CDR + CMD\_IN LVDS receiver): VDD\_PLL, GND\_PLL
- CML (serializer + cable driver): VDD\_CML, GND\_CML

The local ESD devices connect to both power and ground rail or to the ground rail only in case of over-voltage tolerant pads (OVT). OVT pads are used where the input voltage could potentially exceed the local power rail (see pad listing in Sec. 15.1). In a typical environment, all ground

rails are wire-bonded to the same system ground, which enables ESD paths between the (otherwise isolated) power domains. However, during assembly or wafer probing, a common external ground rail might not be established yet. To account for this, a common ESD bus (VSS) has been used to connect the different ground rails via on-chip anti-parallel diodes to create a safe ESD path between power domains at all times. This net (VSS, also used for connecting the global substrate VSUB) should be wire-bonded first (pads 9, 91, and 196), than all remaining ground pads, and finally the rest of the pads.



**Figure 7:** Detail of ATLAS chip bottom with dimensions (rounded to nearest micron).



**Figure 8:** Organization of wire bond pad frame and generic bonding scheme. All wire bonds are shown, including connections for testing (not used on detector modules). The number of fanned-out signal bonds is written in each box, while the power supply bonds run parallel (not fanned out). The red arrows indicate the four unused pads.

215 **3. Basic Operation and Reset (Intro and Quick Start Guide)**

This section walks through the steps for basic, beginner level operation of a single chip on the bench. It also describes how the chip is reset- a critical point for correct start-up. Advanced users will often do things differently than stated in this introductory section, and so the more detailed sections of the document are referenced as appropriate. Thus, this section can also be used as a  
220 guide to the rest of the document. Each item in this section is just one choice out of several possible connections and configuration values. Whenever registers or pins are mentioned they can be found in the reference section (15).

### 3.1 Chip Startup

The startup sequence is power, clock, communication, configuration, operation.

225 **Power** Typical bench testing will use the LDO powering option, in which the internal regulators are used as classic linear regulators fed from a constant voltage power supply, rather than serial power regulators fed from a constant current supply as they will be used in the experiment. LDO powering is more convenient for single chip testing (LDO stands for Low Drop Out voltage). A single chip card will contain jumpers to select LDO mode. A single power supply will be connected to all the chip's V\_IN pins, while the shunt mode controller voltage, VDD\_SHUNT, will be disconnected, which is all that is needed to disable shunt mode. For details see Sec. 4, which also describes the serial power and direct power configurations. External components should be set to their nominal values (Table 30). A power supply current limit of 2 A (half analog and half digital) will be typical.

235 LDO mode allows to view the internal current consumption. (VINA and VIND can be connected individually to monitor currents in analog and digital domains). The supply voltage should be a minimum of 1.4 V and never more than 2.0 V. 1.5 V should be a typical setting for RD53B, or perhaps 1.6 V to have comfortable margin for cable voltage drops. When power is turned on, the current consumption will be determined by the default configuration, which is low (normal)  
240 power for the ATLAS (CMS) chip. The VDDA and VDDD regulator outputs, which connect to external decoupling capacitors, should produce approximately 1.2 V, which is the default setting (one should verify that this is the case when first testing a chip). The ATLAS default configuration analog (digital) current consumption is simulated to be 140 mA (180 mA). Both the regulated voltages and current consumption will be affected by the main reference current ( $I_{ref}$ ), which has  
245 a nominal value of 20  $\mu$ A and can be trimmed with wire bonds (or jumpers on a single chip test card) if needed (Sec. 4.2).

**Clock** All the above happens upon application of power. The reference and power regulation circuits do not have any reset capability. The most critical function after power is clock.

The PLL Clock Data Recovery circuit will become active as soon as it has power and will  
250 produce clock edges on all the internal clocks even in the absence of any external command input. But these will have arbitrary frequency and phase. This arbitrary clock is useful as a diagnostic: it will drive the data output stage (CML output 0) and produce a “heartbeat” idle pattern that can be observed to confirm that the chip is alive and the data connections present. But this arbitrary clock

is not useful for operation. For that one needs a known frequency and phase clock that is obtained  
255 by locking to the incoming command bitstream.

The initialization and reset procedures needed to establish a proper clock and communication  
will be carried out automatically by the DAQ without user intervention, but they are described here  
to provide a basic introduction to how the chip operates and allow troubleshooting.

The reset organization is described in Sec. 3.2. Regardless of the state of the command input  
260 during power up, after power is stable, communication must be initialized by first “idling” the com-  
mand line to a nominal bitrate of 1 Mbps<sup>1</sup> for at least 10  $\mu$ s, and then supplying a 160 Mbps clock  
pattern (80 MHz effective clock frequency) for at least 1 ms. The clock pattern is an uninterrupted  
stream of PLL\_LOCK symbols (Sec. 8.2). This is equivalent to a No Operation (NOOP) command  
in many processors, and can be used as filler when no other commands must be sent, but it will be  
265 referred to as PLL\_LOCK or PLLlock in RD53B.

This “idling” of the command line is the main reset mechanism for RD53B. It can be done at  
any time to recover the chip from a bad state. After the command idle reset, the PLL will enter lock  
mode, and supplying a clock pattern is critical for it to lock to the correct frequency. The locking  
of the PLL can be verified with an optional diagnostic output (see below), but during detector  
270 operation this diagnostic will not be available and there will be no external indication that the PLL  
has locked. It will therefore be necessary to hold the clock pattern long enough to leave no doubt  
that there has been enough time to lock (1 ms). Further details are given in Sec 14.

**Optional Diagnostics** During bench testing it is possible to access a variety of test outputs. The  
chip has one CMOS and four LVDS general purpose outputs that can show a selection of internal  
275 signals (see Sec. 13.1). By default these carry the following information:

CMOS: gpo\_ch\_sync\_lock: 1 if the ChannelSync is locked, 0 when it is unlocked (see below).  
LVDS\_0: CMD\_raw: repeater of the chip command serial input. A buffered version of what the  
chip receives.  
LVDS\_1: cdr\_cmd\_data\_predel: sampled input command pattern before applying any delay.  
280 LVDS\_2: PorResetB: output of Power On Reset circuit (active low) in case it is needed (not nor-  
mally used in RD53B).  
LVDS\_3: gpo\_cdr\_lock\_status: PLL Lock signal. 1 if Locked, 0 otherwise (see clock above).

**Communication** Now all clocks will be at the correct frequency, but the chip is not yet ready to  
understand commands, because the alignment of the incoming command frames has not yet been  
285 established. This alignment is done by a circuit called the channel synchronizer, that recognizes  
unique bit patterns called sync symbols (all command symbols are described in Sec. 8). Therefore,  
after the clock pattern, the sending of sync symbols will be enabled (once again, the DAQ system  
will do this automatically). One can send a constant string of sync symbols or simply enable  
automatic insertion of one sync every N frames (where N is set by configuration, default 32).  
290 So either (sync, sync, sync,...) or (PLLlock, PLLlock, PLLlock, sync, PLLlock, PLLlock,...)- it  
makes no difference. The important thing is to send a large number of sync symbols (exceeding  
a minimum number set by configuration, default 16) before sending any commands. When the

---

<sup>1</sup>A DC level- low or high- instead of 1 Mbps will also work for initiating the reset, but is not advised for A/C coupled command lines.

channel synchronizer locks, the lock signal can be seen on the general purpose CMOS output (for diagnostic purposes). Commands will not be accepted (so the chip configuration cannot be changed) unless this lock signal is high.

**Configuration** The ATLAS chip starts up in a low power default configuration, intended to allow limited communication tests rather than full operation. The pixel matrix is entirely disabled (Analog DIFF FE off, clock off), but writing and reading configuration registers and temperature and radiation monitoring should all be possible. In general a new configuration will have to be loaded for most single chip testing. Test setups will include a baseline configuration suitable for most tests (which may also be called default in test setup documentation, should not be confused with the internal chip default configuration).

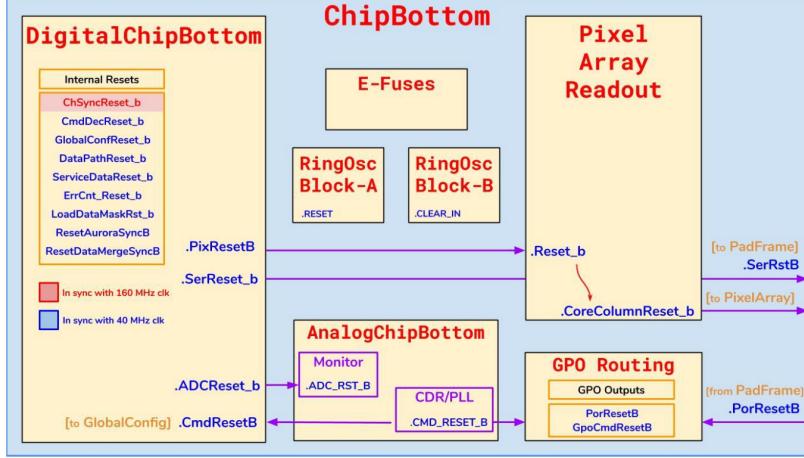
The default configuration enables a single lane Aurora output (lane 0) at 1.28,Gbps To use a programmed configuration other than the default, the configuration selection register must be programmed with the key that enables a non-default configuration to be used (Sec. 3.2.1). A soon as clock edges present in the above command idle reset state, the programmed configuration will also be reset to equal the default, so when the configuration is switched from default to programmed nothing will actually change. In general a new configuration will have to be loaded for most single chip testing. Test setups will include a baseline configuration suitable for most tests (which may also be called default in test setup documentation, should not be confused with the internal chip default configuration).

### 3.2 Reset

The driving requirements of the RD53B reset scheme are:

- Avoiding introduction of Single Event Effect vulnerability. This led to having reset capability only on circuits that absolutely need it, and to use only synchronous reset for them. A synchronous reset signal is a logic level that is sampled locally every clock edge. Spurious glitches on this reset signal have no effect (in contrast to an asynchronous reset, for which a transition produces a reset regardless of clock).
- Need for a default configuration present immediately upon power up. This is done without the use of a power-on reset, as this would require an asynchronous reset on the global configuration registers. The default configuration is not stored in registers, but hard-wired and selected by a 2-to-1 multiplexer (Sec. 3.2.1)
- The ability to reset a chip (or a subcircuit within) without cycling the power, which would require tuning off and on an entire serial chain. This is accomplished with activity detection on the command input (Sec. 3.2.2)

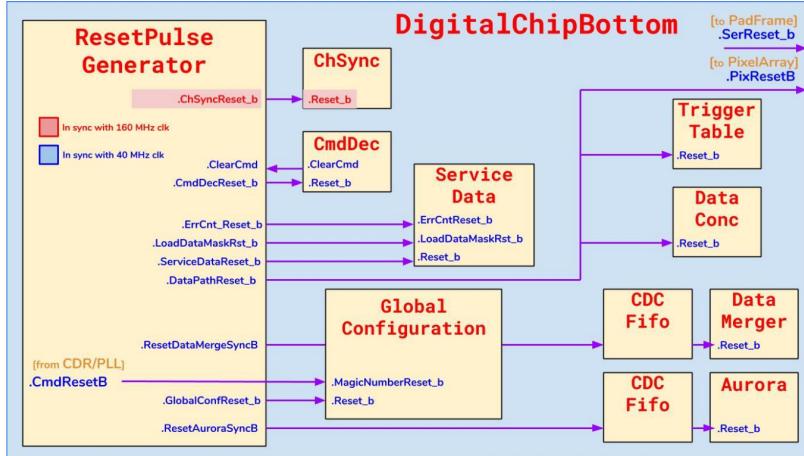
The overall reset organization is shown in Fig. 9. All signal use negative logic: low means reset. There is a power-on reset generation circuit in the chip, inherited from RD53A, but the output of this circuit is not used to reset anything in RD53B and is only sent to the general purpose output multiplexer so that is available for external routing. The only asynchronous reset signal that is used is the command activity detector (Sec. 3.2.2), labeled .CMD\_RESET\_B in the figure. This signal performs 3 functions: (1) it resets the PLL circuit that recovers the clock, (2) it selects



**Figure 9:** Block diagram of reset signals in the RD53B chip as described in the text.

the default configuration (Sec. 3.2.1), and (3) it is used (after synchronization) to actuate all the synchronous resets in the digital chip bottom.

All digital blocks have synchronous resets. These can be individually actuated at any time using the Global Pulse command, in addition to the actuation by the synchronized .CMD\_RESET\_B signal in (3) above. The organization of the digital block resets is shown in Fig. 10. The global configuration registers are explained in Sec. 3.2.1. The logic to write and read global configuration has its own synchronous reset, labeled .GlobalConfReset\_B in the figure.

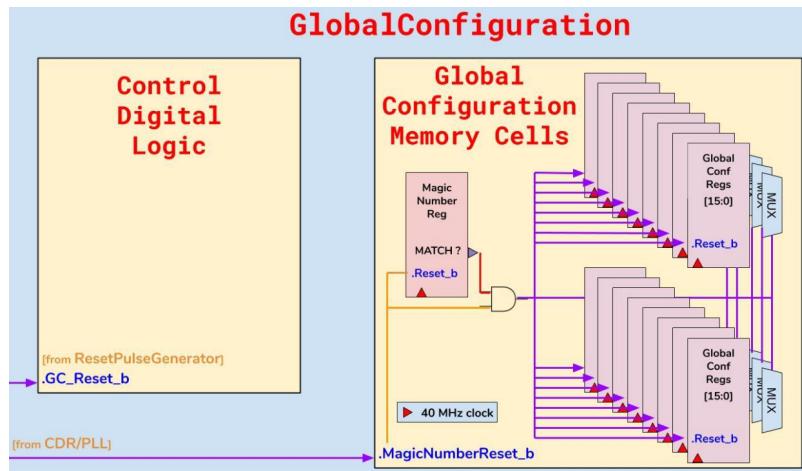


**Figure 10:** Block diagram of reset signals in the RD53B digital bottom as described in the text.

### 3.2.1 Chip Default Configuration

The global configuration registers have a synchronous reset in RD53B. The reset operation sets the registers to the default values. Because the reset is now synchronous, it cannot be guaranteed that the registers will be at the default value immediately upon power-up. Furthermore, the generation of the clock needed for the synchronous reset depends on biases that are controlled by configuration. For these reasons, in order to ensure that the default configuration is present immediately upon

345 power-up or upon CMD idle reset, there is a hard-wired configuration parallel to the registers that is selected by a multiplexer without any clock requirement. Thus there are in two configurations at all times, a default hard-wired one and a programmable one stored in the global registers, as indicated in Fig. 11 by the MUX symbols following each register. Only one configuration is active at any given time. The control of this multiplexing is a logic level. The same mechanism is implemented  
 350 for the pixel configuration, but unlike the pixel matrix, where the programmed configuration has no reset at all, the global configuration has both the MUX and a synchronous reset, so that whenever the MUX selects the hard-wired configuration, the programmed values will soon (when clocks arrive) be reset to equal the hard-wired values.



**Figure 11:** Configuration selection and reset.

The global configuration select level is controlled by both the CMD idle reset signal and a  
 355 logic comparator that compares the value stored in a pair of special configuration registers (32 bits total) to a hard-wired key code or “magic number” (labeled MagicNumberReg in the figure). When the stored value does not match the key the output is low and the default (hard-wired) global configuration is selected. At power up, the registers will contain something arbitrary, and soon after they will be reset to zero as soon as clock edges are present. Neither case will match the key code. The  
 360 user must write the key into the registers in order to switch to a programmed configuration (the key code is Hex AC75 in GCR\_DEFAULT\_CONFIG and Hex 538A in GCR\_DEFAULT\_CONFIG\_B, as can be seen in Table 23). Only the programmed configuration can then be changed to something other than the default. To guard against SEU, in addition to being triple redundant (as are all global configuration registers), any permutation of the key codes with one bit flipped will also select the  
 365 programmed configuration.

In addition to a possible bit flip in the key code, an SEE can put a glitch in the MUX control level and that will cause the active configuration bits to switch between hard-wired and programmed for the duration of the glitch. This is relatively benign and not persistent: after the glitch everything will be in the original state and there is no corruption of the stored configuration. Most  
 370 configuration bits control DACs, which have a slow response time and will therefore not propagate a glitch in their control bits to their output analog level. Furthermore, many DAC values will be close to their default, which means glitches in the selection are unlikely to have any effect.

### 3.2.2 Command Activity Detector

The purpose of this circuit is to provide a “hard reset” mechanism for PLL/CDR block that recovers the clock from the input command stream and controls internal resets. The command activity detector measures the rate of transitions in the incoming command signal. A positive edge rate below a nominal 10 MHz causes a reset to be asserted, while a higher frequency removes the reset. This nominal 10 MHz threshold has a significant uncertainty, with process, voltage, and temperature dependence. Thus, an edge rate  $\ll 10$  MHz (called idling) should be provided to guarantee reset, while normal command activity has a positive rate always between 30 MHz and 80 MHz. The circuit bandwidth is low enough that it takes of order 2  $\mu$ s after command line idling for the reset to be asserted. It will take of order 0.5  $\mu$ s to release the reset once the command line is returned to normal.

The activity detector is part of the PLL/CDR block. It directly resets the PLL, which means it puts it back into frequency lock mode.

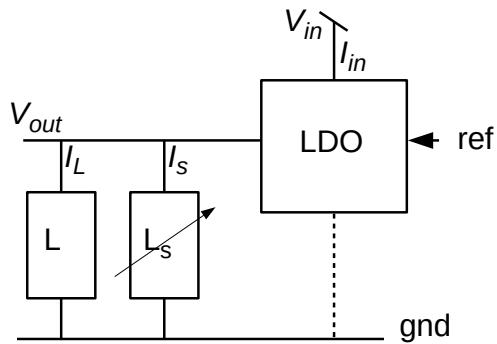
This is the main reset mechanism of the RD53B chip- conceptually equivalent to cycling the power in a typical system. This is necessary because in a serial power chain, cycling the power is truly an action of last resort that should never be needed.

## 4. Power and References

### 390 4.1 Shunt LDO Regulator

The Shunt LDO regulator (SLDO) regulator is a combination of a linear Low Drop-Out voltage regulator (LDO) and a shunt element. The goal is to provide constant current operation with multiple devices connected in parallel (which is not possible for conventional shunt regulators). The circuit was invented as part of the FE-I4 chip development [4], but the design has evolved significantly 395 towards the final implementation in RD53B. There are significant additions and improvements relative to RD53A, including redesigned startup and references (Sec. 4.2), low power mode (Sec. 4.3), “under-shunt” transient protection (Sec. 4.4), and overvoltage protection (Sec. 4.5).

The basic principle of operation of the circuit can be explained using Fig. 12. A conventional 400 LDO voltage regulator is used to power the main load,  $L$ , as usual, plus an internal load,  $L_s$ , in parallel. This internal load (referred to as the shunt element, hence  $L_s$ ) is actively controlled to achieve the desired behavior at the input, no matter what the main load  $L$  does. To first order, the desired behavior is  $I_{in} = I_L + I_s = \text{constant}$ .



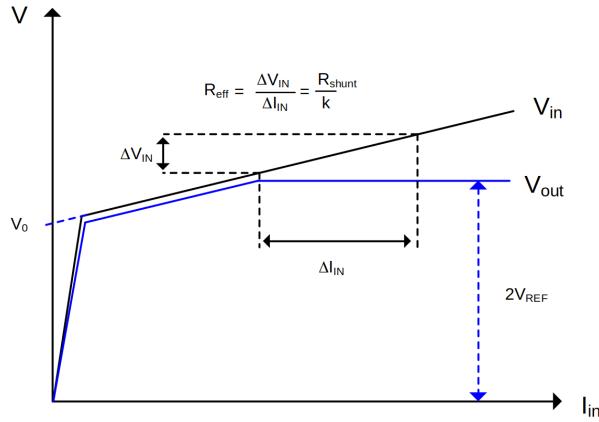
**Figure 12:** Concept of SLDO operation as a linear regulator (LDO) powering an main load  $L$  and a variable internal shunt load  $L_s$ .

The real needed behavior for serial power operation is more complex in order to achieve efficient current sharing among parallel chips and is given by Eq. 4.1.

$$I_{in} = I_L + I_s = \frac{V_{in} - V_0}{R_{\text{eff}}} \quad [V_{in} > V_0] \quad (4.1)$$

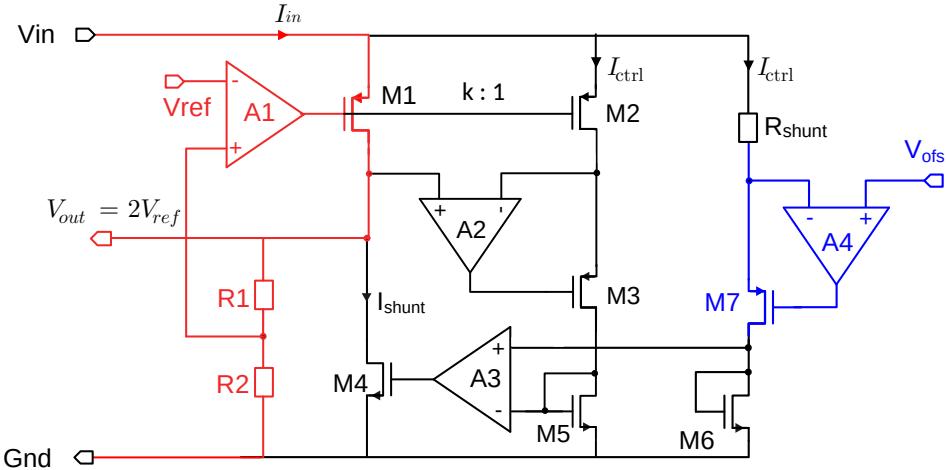
405 where  $V_0$  is a constant but programmable offset needed for high efficiency and  $R_{\text{eff}}$ , also user programmable, gives an ohmic behavior necessary to share current evenly among parallel devices. A diagram of the desired behavior is shown in Fig. 13.

The simplified circuit schematic of the RD53B SLDO is shown in Fig. 14. The red part of the circuit is a classic LDO regulator with pass device M1. The rest of the circuit can be disabled in 410 order to operate in pure LDO mode, which is useful for testing individual chips and for observing the current consumption. The main load  $L$  (external to the SLDO) is not shown- it is the chip itself. Device M4 is the internal load of Fig. 12 and the rest of the black circuitry is the active control. This



**Figure 13:** Desired current vs. voltage characteristics for SLDO. The unregulated input voltage and regulated output voltage are shown. Indicated values are discussed in the text.

control ensures that the current in the pass device M1 is equal to the the current through  $R_{\text{shunt}}$  ( $I_{\text{ctrl}}$  for control current) times the scale factor K, which has a design value of 1000.  $R_{\text{shunt}}$  is an external resistor to allow the user precise control  $R_{\text{eff}}$ . It can be seen that  $R_{\text{eff}} = R_{\text{shunt}}/1000$ . Finally, the blue circuit provides the offset  $V_0$ . This is controlled by a reference voltage labeled  $V_{\text{ofs}}$ . User control of  $V_{\text{ofs}}$  is described in Sec. 4.3.



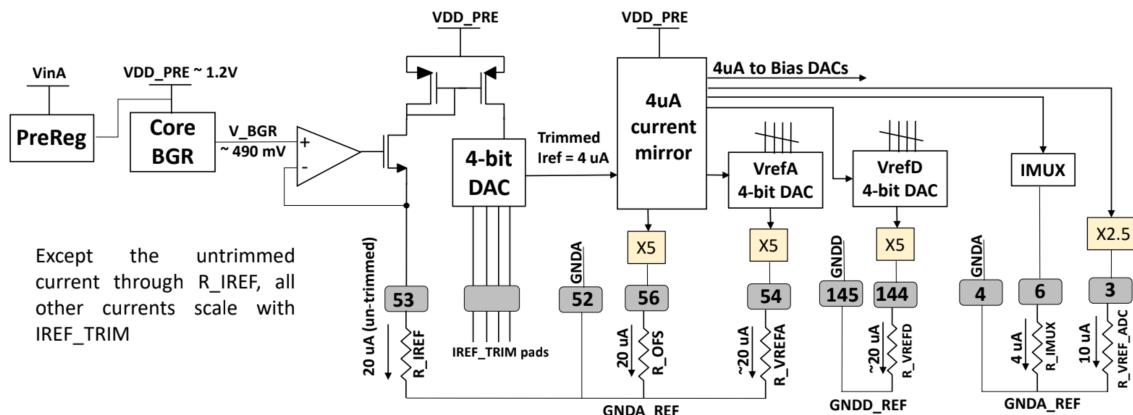
**Figure 14:** Simplified schematic RD53B SLDO regulator. The colors differentiate the LDO (red), shunt (black), and offset (blue) functions as discussed in the text.

The SLDO circuit is designed to be compatible with 2 V input voltage. Even during transients, no gate in the circuit will have more than the allowed maximum of 1.32 V across it. All transistors are cascoded in order to always have more than two transistors between voltage supply and ground, with supply voltage distributed across several transistors. Device voltage limits checks in static and dynamic simulations show that no transistor sees more than 1.32 V across any two terminals.

The one exception is the pass device M1, where cascoding to protect against over-voltage would cause higher drop-out voltage and therefore higher power consumption. This lack of cascoding of device M1 leads to the Vref lower limit to the undershoot protection range (Sec. 4.4). The SLDO circuit also uses a Low-ESR output capacitor compensation scheme, such that careful control of the external component equivalent series resistance (ESR) is not necessary.

## 4.2 References and Startup

For serial chain operation the SLDO must become active immediately upon current flow, before communication is possible. Once operational it must work with high efficiency and uniformity among chips in the chain, and it must support a low power mode for detector integration. These requirements often conflict and complicate startup. Furthermore, startup must work reliably over a wide temperature range, from room temperature for bench testing and wafer probing, to the evaporative cooling base temperature (taken to be -40°C, that may be reached before power is applied). The generation of current and voltage references is intimately connected to the startup behavior.

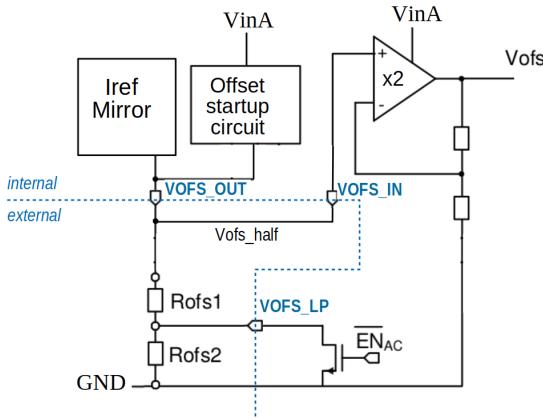


**Figure 15:** Generation of references and recommended connection to internal grounds. See Fig. 16 for further detail on  $R_{OFS}$ , shown here as a single resistor.

The RD53B reference scheme is fundamentally different from RD53A. An effort was made to remove circular connections in order for more robust startup. The RD53B scheme is shown in Fig. 15. RD53B does not use the SLDO output to power any reference circuit. A new low current linear regulator (the *preregulator*) is used to power the main reference current generator. All other references are then derived from this unique main reference current. The preregulator is outside of and in parallel to the chip power concept of Fig. 12, but because it is low current it does not noticeably alter the behavior of Eq. 4.1 and Fig. 13. The preregulator includes its own dedicated bandgap voltage reference, which does not need to be very precise, as the preregulator output does not need to be exactly 1.2 V, but merely between 1.1 V and 1.32 V. The preregulator is a low power device capable of a current of order 10 mA.

The Core Bandgap generates the main reference current, which can be adjusted with a 4-bit trim set by wire bond pads with internal pull-up resistors. This allows to compensate for process

variations and equalize all chips to the design reference current value of  $20\ \mu\text{A}$ . Note that unless some of the pads are externally grounded the reference current will be at its maximum value.  
450 The generated main reference current also depends on the external resistor  $R_{\text{Iref}}$  (Table 30), which is external in order to avoid the temperature variation of internal devices. The two LDO reference voltages,  $V_{\text{ref}}$  analog and  $V_{\text{ref}}$  digital, are each generated by a known current (derived from the main reference) across a dedicated external resistor (Table 30). Each  $V_{\text{ref}}$  is independently adjustable by configuration to allow some fine adjustment of the chip internal operating voltage.  $V_{\text{ref}}$  adjustment does not change the  $V_{\text{in}}$  vs.  $I_{\text{in}}$  behavior of the chip, making sudden jumps due to configuration upset or operator error harmless for serial chain operation. The offset reference  $V_{\text{ofs}}$  is common to both SLDOs and is not adjustable by configuration, as sudden jumps in  $V_{\text{ofs}}$  would be problematic for serial chain operation. The generation is shown in Fig. 16.



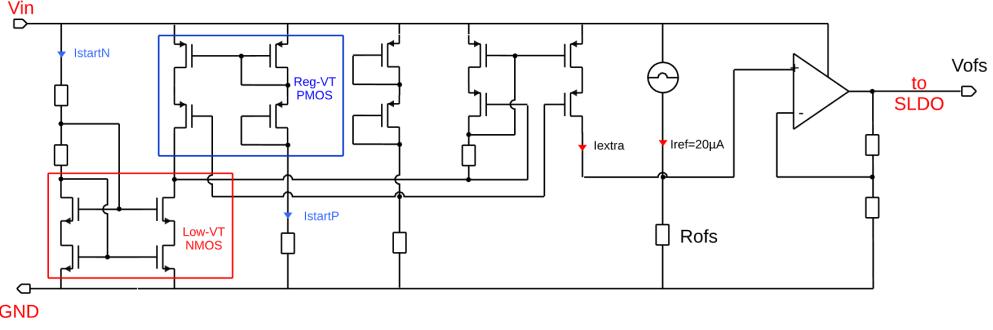
**Figure 16:** Offset voltage generation and startup. See text for description. The resistors in series  $\text{Rofs}_1$  and  $\text{Rofs}_2$  may sometimes be referred to as simply  $\text{Rofs}$ . The dashed blue line separates internal from external components and connections, while wire bonds pads are indicated by the pin symbol and blue labels. See Fig. 18 for optional connection of the  $\text{VOFS\_OUT}$  and  $\text{VOFS\_IN}$  wire bond pads on a module.

#### 460 4.2.1 Offset Voltage Start-up

While circular dependencies requiring start-up circuits have mostly been removed from RD53B, the offset voltage ( $V_{\text{ofs\_half}}$ ) does require a startup (Fig. 16), because a very low offset voltage will cause the shunt device (M4 of Fig. 14) to be fully on, and this will clamp the input voltage,  $V_{\text{in}}$ , to a low value even if a large current is supplied. This is especially critical for low power mode  
465 (Sec. 4.3). The built in start-up circuit shown in Fig. 17 boosts the offset voltage to follow  $V_{\text{in}}$  until  $V_{\text{in}}$  is high enough for the preregulator to work and all references to be at their correct values. The circuit injects a current into the offset voltage setting resistor until the preregulator reference voltage rises. The rise of the preregulator reference shuts off this startup circuit.

### 4.3 Offset Voltage and Low Power Mode

470 The SLDO offset voltage plays critical roles. It is the most important voltage for regulators operating in parallel, because the total current in a given SLDO,  $I_{\text{in}}$ , is very sensitive to  $V_0$  (and therefore



**Figure 17:** Offset voltage startup circuit. The Resistor  $R_{\text{ofs}}$  is either  $R_{\text{ofs}1}$  in normal operating mode or  $R_{\text{ofs}1} + R_{\text{ofs}2}$  in low power mode (see Fig. 16).

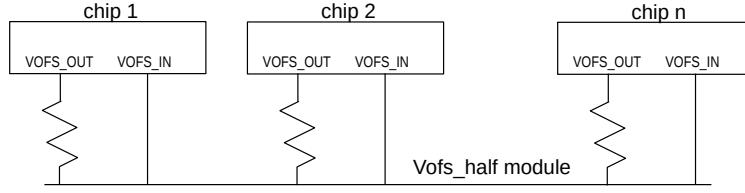
to  $V_{\text{ofs}}$ ), as can be seen by the  $dI_{\text{in}}/dV_0$  derivative to Eq. 4.1, which is  $-1/R_{\text{eff}}$ . (This is also true for  $dI_{\text{in}}/dV_{\text{in}}$ , but  $V_{\text{in}}$  is by construction equal for all SLDOs wired in parallel.) It is very important to note that the actual offset voltage,  $V_0$ , in Fig. 13 is twice the generated  $V_{\text{ofs\_half}}$  voltage. This is because the actual offset voltage may need to be higher than the preregulator output voltage, and so is impossible to generate directly. A  $\times 2$  buffer with  $V_{\text{in}}$  rail internally generates the true offset voltage from  $V_{\text{ofs\_half}}$  (Fig. 16). Only  $V_{\text{ofs\_half}}$  is accessible outside the chip and can be manipulated via the VOFS\_OUT, VOFS\_IN and VOFS\_LP wire bond pads.

$V_{\text{ofs}}$  may not necessarily be equal for different chips placed in parallel, and this can lead to current imbalance. While a small value of  $R_{\text{eff}}$  will make a single SLDO more efficient (lower voltage drop between  $V_{\text{in}}$  and  $V_{\text{out}}$ ), it can make a multi-chip module less efficient by amplifying a small  $V_{\text{ofs}}$  chip-chip mismatch into a large current imbalance. Two solutions to this problem are possible in RD53B<sup>2</sup>. First, it is possible to trim the main current reference to produce a target  $V_{\text{ofs}}$  value, rather than to produce a target current value. This will result in a larger chip-chip variation of reference current, but since all internal biases are adjusted with dedicated DACs this is not a problem. In return for the larger variation of reference current there will be a smaller variation in offset voltage. The second solution is to tie together the  $V_{\text{ofs}}$  outputs of all chips in the same module via resistors (Fig. 18). For this purpose, in RD53B the  $V_{\text{ofs}}$  output of Fig 16 and the  $V_{\text{ofs}}$  input of Fig 14 are on separate wire bond pads.

The common  $V_{\text{ofs}}$  wiring of Fig. 18 is robust against chip failure. Should one of the  $V_{\text{ofs}}$  outputs be grounded, the common  $V_{\text{ofs}}$  will be reduced. This will cause the working chips to draw more current for a given  $V_{\text{in}}$ , which is actually beneficial in the case the failing chip draws low current, as the working chips must now carry the extra current from the failing one.

Rather than a single external resistor to set  $V_{\text{ofs}}$ , Fig 16 shows two resistors with a center tap switch- effectively a 1-bit variable resistor. The switch is internal in RD53B, while the resistors are external. This allows implementation of a low power serial chain mode. For normal serial chain operation the switch is conducting and the resistor value is just  $R_{\text{ofs}1}$ . When the switch is off, the resistor becomes  $R_{\text{ofs}1} + R_{\text{ofs}2}$ , leading to a higher  $V_{\text{ofs}}$ . A higher  $V_{\text{ofs}}$  means that a small current will develop a high enough value of  $V_{\text{in}}$  needed for the SLDO output to reach 1.2 V. This also requires

<sup>2</sup>Assuming the  $R_{\text{ofs}}$  resistors cannot be practically trimmed individually

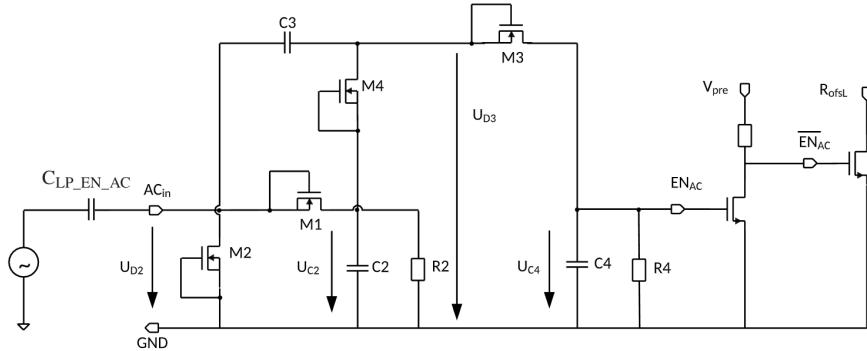


**Figure 18:** Common offset voltage wiring option for a multi-chip module. The Vofs\_OUT and Vofs\_IN wire bond pads can be seen in Fig. 16 and Table 23. For single chip operation Vofs\_OUT should be simply looped back to Vofs\_IN.

500 the default configuration to be very low current, as is the case in RD53B ATLAS.

The switch is controlled by a dedicated A/C input as this mode is only intended for use during detector construction, when additional contacts can be made. An A/C signal on this special input will turn the switch off, and the absence of a signal (as will be the case for normal operation) will leave the switch conducting. Note that if this high  $V_{ofs}$  low power mode is never needed one can 505 simply leave out  $R_{ofs2}$  and connect  $R_{ofs1}$  to ground, in which case it no longer even matters what the state of the switch is.

The amplitude of the A/C signal to activate low power mode should be 1.2 V peak-to-peak and should not exceed 1.32 V. A square wave with a rise time below 100 ns should have any frequency larger than 80 kHz, while a sine wave should have a frequency larger than 130 kHz. This assumes 510 the A/C signal is coupled to the chip by a 100 nF external capacitor (Table 30). The rectification circuit that turns the A/C signal into an internal logic level is shown in Fig. 19. It consists of a 2 stage rectifier using low threshold NMOS transistors with applied forward-body biasing as rectification elements to achieve the minimum possible threshold voltage.



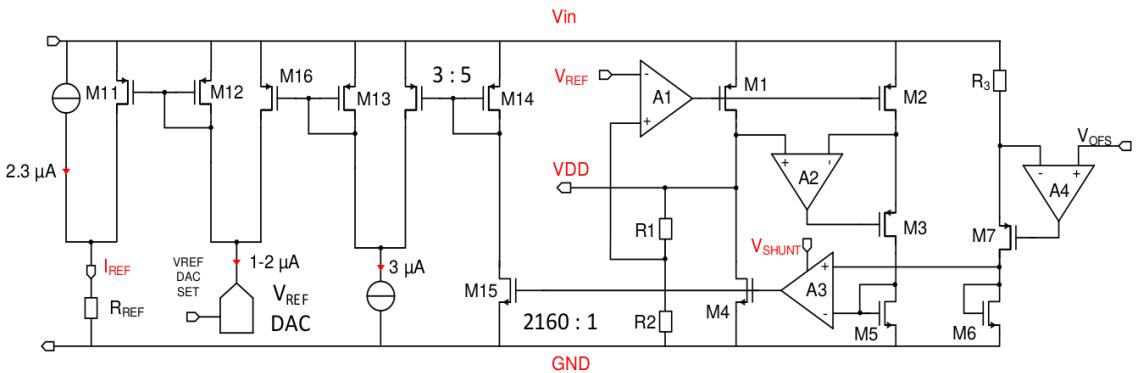
**Figure 19:** Rectification circuit for external A/C signal that enables low power mode.

#### 4.4 Under-shunt Current Protection

515 The variable internal shunt load  $L_s$  of Fig. 12 can act to keep the total current constant as long as the load current drawn by the chip,  $I_L$ , is less than the programmed total current  $I_{in}$ . But if due to an error or fault condition  $I_L > I_{in}$ , then there is nothing the variable load  $L_s$  can do to prevent the

total current from exceeding  $I_{in}$ . An additional function is need to react to the condition  $I_L > I_{in}$ .  
520 The under-shunt circuit acts to prevent the  $I_L > I_{in}$  condition. It is different from a classic current limit circuit, because the programmed value of  $I_{in}$  is not fixed in advance, but set by an external resistor. Thus it is not possible to have a hard-wired absolute current limit.

Turning around Eq. 4.1,  $I_s = I_{in} - I_L$ , where  $I_s$  is the internal shunt current in M4 of Fig. 14.  
525 The desired condition  $I_L < I_{in}$  is equivalent to a non-zero shunt current,  $I_s > 0$ . Thus, the under-shunt protection compares a scaled replica of the M4 current to a threshold (which does not have to be precise), and if it goes below threshold (known as the under-shunt condition), it reduces  $V_{ref}$  (by reducing the current it is derived from). Lowering the voltage powering the load  $L$  will reduce the load current  $I_L$ . The circuit is shown in Fig. 20. However, the  $V_{ref}$  is not allowed to drop below 0.35 V, to avoid the possibility of a voltage greater than 1.32 V across M1 of Fig. 20, which could cause permanent damage to the device.

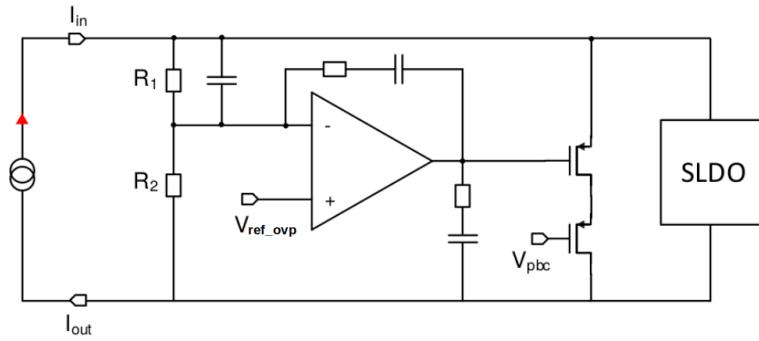


**Figure 20:** Under-shunt protection circuit.

530 The under-shunt protection is disabled by default and must be enabled in the global configuration. It can prevent internal shorts from being visible outside the chip, as long as their effective  
535 resistance is greater than  $0.7 \text{ V}/I_{in}$ . It can also prevent transient “shorts” (for example due to a simultaneous firing all comparators or a wrong configuration setting) from drawing more than the programmed  $I_{in}$ . Simulations of selected test cases show that the under-shunt protection generally mitigates both DC shorts and transients, but it can also lead to internal oscillation when the reduction of the load voltage removes the under-shunt condition, but then the condition returns when the load voltage recovers. These internal oscillations are not expected to be a problem for the system outside the chip. Ultimately, the use or not of under-shunt protection will have to be informed by system tests.

## 540 4.5 Over-voltage Protection

In the SLDO design the shunt element M4 of Fig. 14 is placed after the pass device M1. The total current draw is limited by the pass device and additional current cannot be shunted by M4. Therefore, classic over-voltage protection (OVP) is implemented with a current clamp in parallel to the SLDO. Since the voltage being clamped is  $V_{in}$ , which is common to both SLDOs, there is 545 only one single clamp for the whole chip. The circuit is shown in Fig. 21.



**Figure 21:** Over-voltage protection clamp.

The OVP must only become active if the input voltage is close to 2 V. The clamp threshold is  $0.333 \times V_{ref\_ovp}$ , where  $V_{ref\_ovp}$  is an internal reference obtained as a copy of the preregulator bandgap output voltage  $V_{ref\_PRE}$  and is expected to be around 0.6 V. This value can be overridden with the wire-bond pad  $VREF\_OVP$ , without affecting  $V_{ref\_PRE}$ . OVP can be disabled by driving  $VREF\_OVP$  to a high value (for example  $VDD\_PRE$ ).  
550

Note that if multiple chips in parallel go into OVP, there is no current balancing mechanism for this function, so the chip with the lowest effective OVP threshold will take most of the current.

## 5. Analog Front End

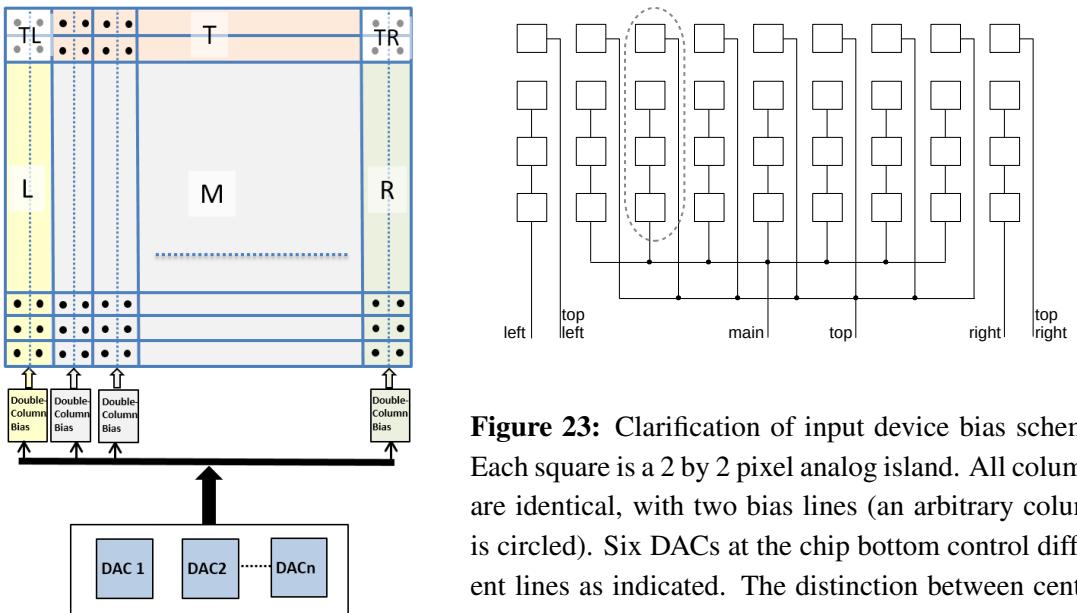
The ATLAS and CMS chips use different front ends (FE). However, they are treated the same way by the design framework and share many features. Much about the RD53B FE can be described generically, applying equally to ATLAS and CMS. The FE is a pure analog circuit: it contains no memory latches, flip-flops or counters. Static configuration values are provided by the digital core, which receives only the comparator out signal from the analog part. The design is a small-area, low-power, free-running amplifier and discriminator for negative input charge. All necessary biases are generated in the chip bottom as described in Sec. 5.1. The calibration charge injection circuit and operation are described in Sec. 6.

The FE circuits are laid out in analog islands of 4 FE's each, as was described in Sec. 1.

### 5.1 Front End Bias Generation and Distribution

The bias voltages for the analog front-ends are provided by a set of programmable 10-bit DACs placed in the Digital Chip Bottom, near the pad frame. The list of configuration registers of the bias DACs is available in Sec. 15.2.

The bias distribution to the pixel array is based on a 2-stage scheme, as shown in Fig. 22. The biases from the DACs are distributed in parallel to the DOUBLE\_COLUMN\_BIAS blocks placed at the feet of the pixel array. Then, each DOUBLE\_COLUMN\_BIAS generates and distributes the bias and threshold voltages to two pixel columns. The chosen granularity allows a certain level of redundancy, so that a hard failure in one pixel will not affect the bias of the full pixel array.



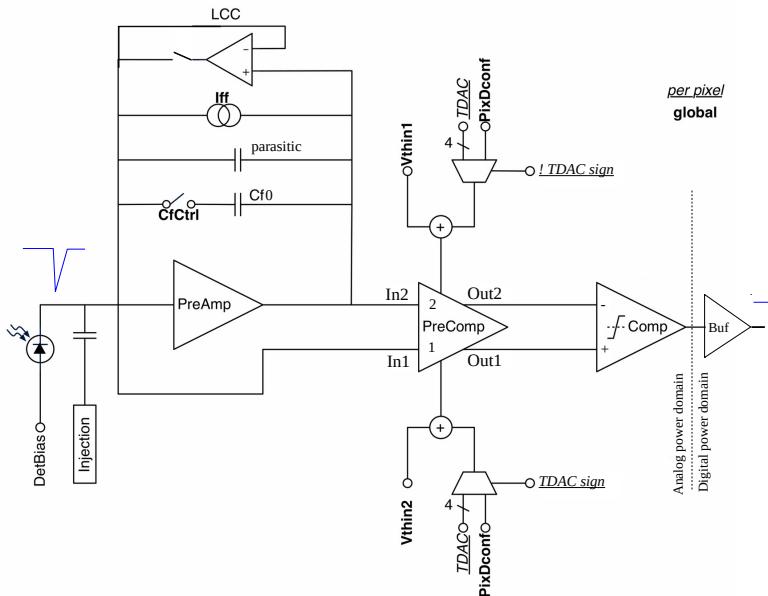
**Figure 22:** Bias distribution scheme.

The distribution scheme takes also care to provide dedicated biases to the edge and top pixels, that will be larger than normal pixels to span the gap between adjacent chips in quad or dual

chip modules. Edge/top pixels may need different bias to cope with greater capacitance and leakage current than the normal pixels. Simulations of the analog front-end showed that the only bias requiring different setting is the current of the preamplifier input transistor: a higher current allows both to align in time the response of the edge pixels and also partially recover the noise increase due to the greater capacitance. Therefore, only the input transistor bias can be adjusted differently for edge/top pixels, while all other biases are the same everywhere. The distribution of the input transistor bias is illustrated in Fig. 23. This distribution creates six groups of 4-pixel islands: Main (the interior if the chip), Left edge, Right edge, Top, Top Left corner, and Top Right corner. Each group has its own dedicated DAC for the input transistor bias, and these DACs can be set to the same or different values as needed by changing the value of the corresponding `DAC_PREAMP_{M,L,R,TL,TR}_DIFF` registers (ATLAS) or the `DAC_PREAMP_{M,L,R,TL,TR}_LIN` registers (CMS) (see Table 23).

In addition to the input transistor bias, the global thresholds setting is also modular for the left and right edge double-columns. Instead, the pixels of the top edge do not have an independent threshold. Therefore, the chip is equipped with three global threshold DACs: Left double column, Main array and Right double column. The global threshold is then set by setting a fixed value to the `DAC_VTH2_DIFF` register, and adjusting the value of the `DAC_VTH1_{L,M,R}_DIFF` registers (ATLAS) or adjusting the `DAC_GDAC_{L,M,R}_LIN` registers (CMS) as needed (see Table 23).

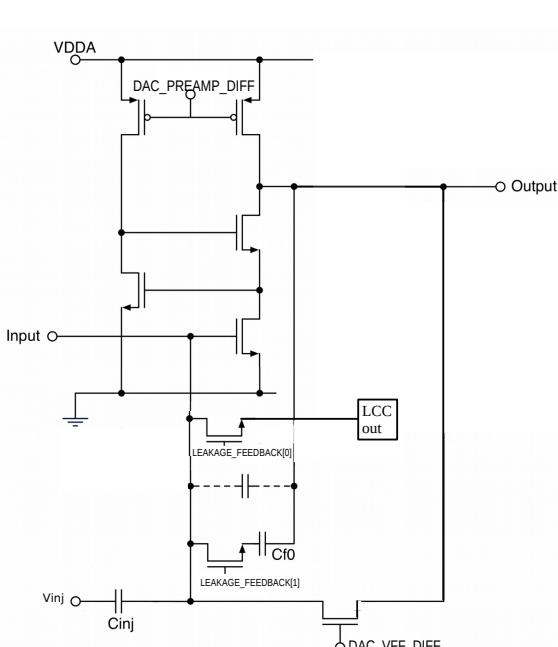
## 5.2 ATLAS Analog Front End



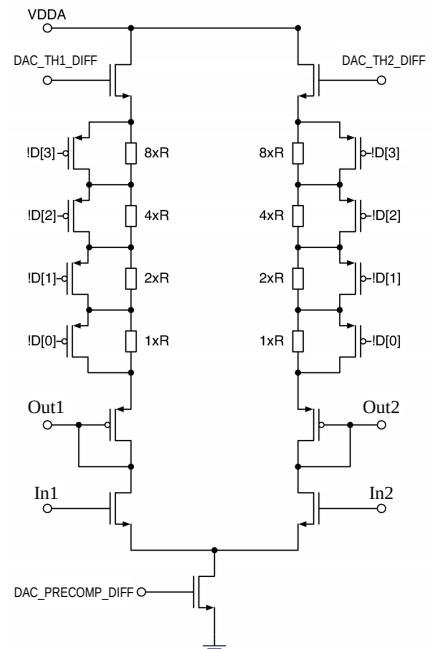
**Figure 24:** Schematic of ATLAS chip analog front end with differential second stage and comparator. The underlined italic signals are controlled per pixel (see sec. 15.4), while the boldface signals are global. A digital buffer is manually placed at the comparator output. The schematic for the Preamp, including its biases, Cff, Cf0, is shown in Fig. 25. The schematic for the LCC circuit is given in Fig. 29. The schematic for the PreComp, including its biases, TDACs, and Vthin, is shown in Fig. 26. The schematic for the Comp circuit is shown in Fig. 30.

The front end (FE) for ATLAS is based on the Differential FE of RD53A chip. Fig. 24 shows the FE block diagram. It contains a single ended charge integrator feeding a differential second stage and comparator. The ADC function is implemented entirely in the digital core (Sec. 7), by digitizing the time-over-threshold (ToT) of the comparator output pulse. A digital buffer is manually placed at the comparator output to ensure fast edges are available for the digital core.

The pre-amplifier (preamp) or first stage is exactly the same circuit as in RD53A (Fig. 25). It has a straight regulated cascode architecture with NMOS input transistor in weak inversion. A high-VT type input device is used. It has a continuous reset and adjustable gain by adding or not a feedback capacitor, cf0 (this choice is made globally using the least significant bit of the LEAKAGE\_FEEDBACK register, not per pixel). The preamp can operate at very low currents and has two biases: the main bias (input transistor current), and the continuous reset feedback current. The feedback current is set globally using the DAC\_VFF\_DIFF register, and cannot be trimmed in each individual pixel. The preamp is single ended, but the feedback ensures that, in the absence of signal, input and output are at the same potential. Input and output are thus taken as a differential input to the next stage.



**Figure 25:** First stage (preamp) schematic. LCC is a Leakage Current Compensation circuit. The cf0 capacitor to enable low gain mode has a value of 3.28 fF. In high gain mode cf0 is off and the only feedback capacitance is parasitic, with a value of 3.73 fF.



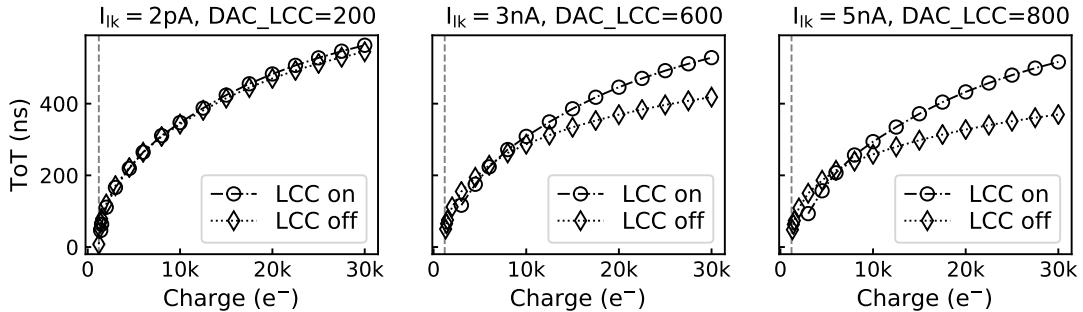
**Figure 26:** Second stage (pre-comparator) schematic. The threshold is introduced in this stage by offsetting the two branches using resistor ladders. The preamp output is connected to In2 and the preamp input to In1.

A leakage current compensation circuit (LCC) provides additional optional feedback in the preamp. The circuit is enabled/disabled using the most significant bit of the LEAKAGE\_FEEDBACK register. It is normally off and should only be used in case of large sensor leakage current (2 nA/pixel

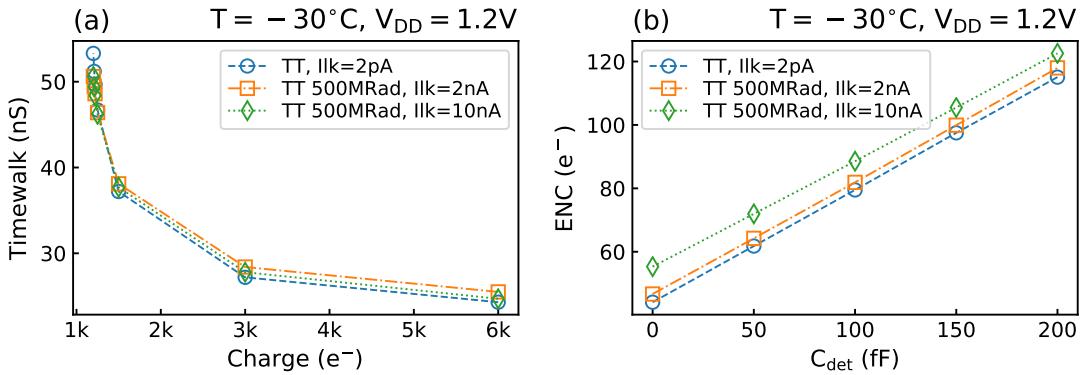
or higher). It is an active low pass filter, shown in Fig. 29. Because it is a low pass filter, it can sculpt the response to periodic burst injections as done in calibrations, so calibration scans to be run with LCC enabled should keep this in mind. The bias of the LCC is globally adjustable using the DAC\_LCC\_DIFF register. Table 3 shows the recommended configuration of the LCC circuit, depending on sensor leakage current. Figure 27 demonstrates the advantage of using the LCC circuit when high sensor leakage current is present. Figure 28 shows the simulated equivalent noise charge and timewalk for different levels of sensor leakage current.

Sensor leakage current (nA/pixel)	LCC status	DAC_LCC_DIFF
0 - 2	Disable	200
2 - 4	Enable	600
4 - 8	Enable	800

**Table 3:** LCC configuration.



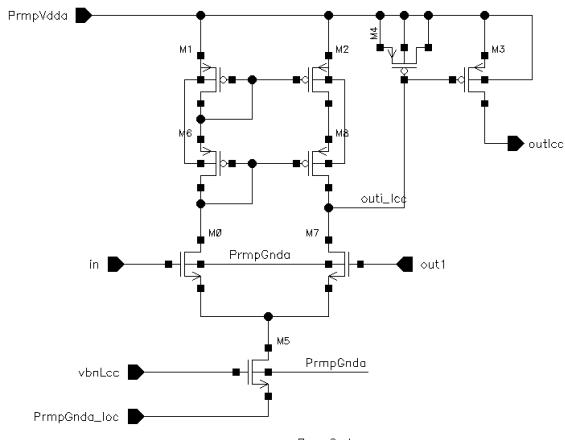
**Figure 27:** Comparison of ToT when LCC is on or off, for various sensor leakage currents  $I_{lk}$  with their corresponding LCC configuration.



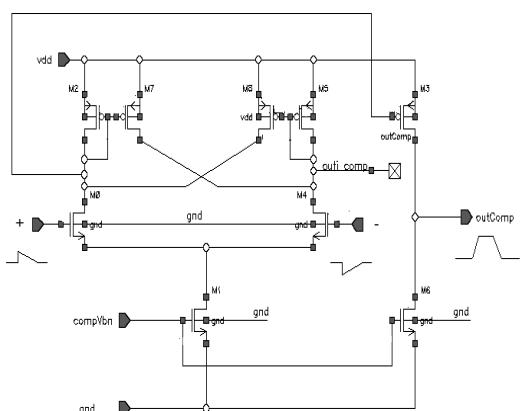
**Figure 28:** Simulated equivalent noise charge as a function of sensor capacitance (a), and timewalk as a function of input charge (b), before and after irradiation.

The DC-coupled pre-comparator or second stage maintains the same schematic as RD53A (Fig. 26), but two transistors (those connecting to Out1 and Out2 in the figure) have been resized.  
620 Their length has been reduced to increase operating margin at low temperature at the expense of reduced gain (from 3 V/V to 2 V/V). This has a negligible effect on overall performance, as the main function of the pre-comparator is to introduce the threshold, not to add gain. The global threshold is adjustable through two distributed voltages (VTH1 and VTH2) which introduce an offset between the two branches of the pre-comparator. The left and right analog island columns (including the  
625 corners) have dedicated VTH2 threshold settings. Thus, VTH2 is set in common for the whole chip, while VTH1 can be set separately for left, middle, and right (no top row distinction). The threshold is trimmed in each pixel using one 4-bit resistor ladder in each pre-comparator branch. The branch current is turned into additional voltage offset by these resistor ladders. An effective 5-bit adjustment is obtained by adjusting one branch resistance or the other using a single 4-bit value.  
630 The 5<sup>th</sup> bit is a “sign” bit, which determines which branch is adjusted. The branch that is not adjusted is set to all 1 or all 0, depending on a global configuration value. The bit values are defined in Sec. 15.4. The bias of the pre-comparator is globally adjustable using the DAC\_PRECOMP\_DIFF register.

The FE design is optimized for low-threshold operation. The pseudo-differential design reduces variation due to mismatch and provides improved power supply rejection. The pre-comparator stage is followed by a classic continuous time comparator stage (Fig. 30) with output connected to the digital pixel region through a buffer placed in close proximity to the comparator output. The polarity of the comparator has been flipped relative to RD53A, by taking the single output from the other differential branch than RD53A. Thus, when a negative input charge fires the RD53B front end, the comparator output switches from low to high. This has two advantages relative to RD53A (which switched from high to low). (1) The slew rate is higher, because it is limited only by the gain of PMOS transistor in the output stage, whereas the falling edge slew rate is limited by the bias current in the NMOS transistor, which is purposely small to save power. (2) The quiescent state of the output is low. Therefore, if the analog supply rail happens to be much lower than the digital supply rail (for example during an under-shunt current transient triggering the internal protection) this will not cause a digital transient, as would be the case if the quiescent state was high. Finally, similar to all the other circuits of the AFE, the bias of the comparator is globally adjustable using the DAC\_COMP\_DIFF register.  
635  
640  
645



**Figure 29:** Leakage current compensation circuit (LCC). The high impedance ports “in” and “out1” are connected to the preamp input and output, respectively. The LCC output is connected via a switch to the preamp input as shown in Fig. 25

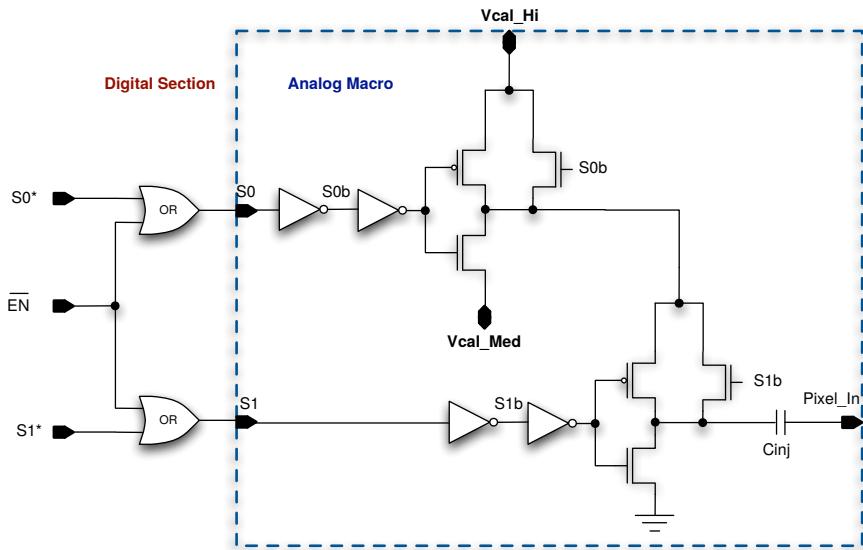


**Figure 30:** Comparator schematic. Note the a single bias is used for the tail current in the comparator proper as well as the output stage current.

## 6. Calibration Injection

650 There are two types of calibration injection: digital and analog. The same command (Sec. 6.2) is used for both, and which one is active is selected by configuration in register CalibrationConfig (Table 23). Digital injection bypasses the front end and inputs a pulse to the hit processing logic as shown in Fig. 40. It is therefore relatively simple: the digital pulse generated by the Cal command is directly what the processing uses and is fully deterministic (no noise). It is useful to 655 test proper functioning of the readout chain, as a timing reference for each pixel's FE analog delay, etc. The rest of this section is concerned with analog injection.

660 The calibration injection circuit uses two distributed DC voltages plus in-pixel switches to chop them and generate steps fed to an injection capacitor. Having two voltages allows a precise 665 differential voltage that will be independent of local ground drops in the chip, as well as two consecutive injections into the same pixel. The injection circuit is implemented in every pixel and its topology is shown in Fig. 31. The control signals, S0 and S1, are generated in the digital domain as explained in Sec. 6.1 and can be phase shifted relative to the bunch crossing clock with a fine delay, which is global for the whole chip. The enable bit (EN) is programmable for each 670 pixel. Injection takes place only for cal enabled pixels when either S0 or S1 switch from low to high. Analog injection must therefore be primed by setting at least one control signal low, prior to being able to inject. This priming is not automatic, so that the user is able to control the amount of settling time allowed prior to injection. The CAL command is used for both functions: prime and inject (see Sec. 6.2). The value of the injection capacitor can deviate from nominal due to process variations, so a dedicated circuit is provided to measure a replica capacitor array in each chip during wafer probing (Sec 13.8).



**Figure 31:** Calibration injection circuit in each pixel. The injection capacitor nominal value is 8.02 fF.

Just as in a common inverter, there will be a switching transient when a control signal switches

from low to high. Simulations show these transients to have a negligible impact on the distributed Vcal\_Hi and Vcal\_Med voltages. Note that at the top of each inverter there is an NMOS transistor in parallel with the PMOS, which switches first (before the PMOS) when injecting. This allows the  
675 switches to operate for any choice of voltages Vcal\_Hi>Vcal\_Med>GND, but since the top NMOS switches first, it does not contribute transients during injection. During priming, on the other hand, the top NMOS switches first, while the bottom NMOS is still conducting, resulting in a short circuit lasting one inverter delay. This will cause a transient on the Vcal\_Hi and Vcal\_Med voltages, and the user must therefore allow some settling time between priming and injection. In addition to this  
680 transient, priming injects a positive polarity pulse into each enabled front end, so one must allow for the front end to settle in any case. The two-voltage injection circuit allows injection of two successive pulses without priming in between, and with arbitrary delay between these pulses. The two voltages also mean that the charge injected by S0 is given by a differential voltage and not affected by local ground potential differences. Keeping S1=0 and only toggling S0 will result in  
685 single pulse differential injection. The two voltage system also makes it possible to inject different amount of charge simultaneously in neighboring pixels by changing the meaning of S0 and S1 in different pixels (see Sec. 6.1).

Finally, since the voltage distribution lines have finite impedance, injecting into too many pixels simultaneously will cause the voltages to droop, introducing a nonlinearity in the injected  
690 charge vs. number of pixels injected. Simulations show this nonlinearity to be less than 1% for simultaneous injection into 3 full rows of pixels and less than 2% for 4 rows, but the exact value of this nonlinearity should be measured in actual chips by measuring threshold vs. number (and pattern) of injected pixels.

## 6.1 Generation of S0 and S1 signals

695 The signals S0 and S1 of Sec.6 exist locally in each pixel but are derived from different internal signals produced by the command decoder and distributed to the array. This two-step scheme is necessary in order to have either sequential or simultaneous injection, and more importantly to avoid having to distribute two switching signals with precise timing. Since the calibration input is used to study and calibrate timing, it must occur simultaneously in all enabled pixels, just as is the  
700 case for the bunch crossing clock (here, simultaneously means within a 2 ns window). Two control signals are distributed: CAL\_edge and CAL\_aux. As the name implies, CAL\_edge needs to be simultaneous in all pixels, while CAL\_aux does not. CAL\_edge has a fine phase adjustment relative to the beam crossing clock, which is called CAL\_delay. In fact CAL\_edge looks like a traditional injection pulse, with user controlled leading edge time and duration. In *uniform* injection mode  
705 (which allows injecting two pulses close in time into all selected pixels), S0 and S1 are derived from CAL\_edge and CAL\_aux identically for all pixels:

$$S0 = \text{CAL\_edge} \text{ OR } \text{CAL\_aux} \quad (6.1)$$

$$S1 = \overline{\text{CAL\_edge}} \text{ AND } \text{CAL\_aux} \quad (6.2)$$

The rising edge of CAL\_edge throws the S0 switch, while the falling edge throws the S1 switch. The CAL\_aux starts low and then goes high after CAL\_edge, but not with precise timing. In uniform mode the injection switches can only be thrown in that order. Either only use S0 for single  
710 pulse, differential voltage injection, or use S0, then S1, for double pulse injection.

In order to allow injection of different size pulses simultaneously into adjacent pixels, there is an *alternating* analog injection mode that can be selected instead of the default uniform mode. In this mode the S0 and S1 signals are derived as above only for *even* pixels, but swapped for *odd* pixels:

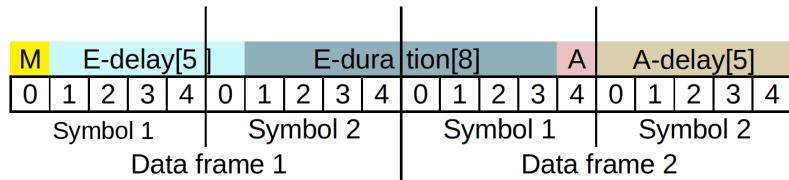
$$S1_{odd} = \text{CAL\_edge} \text{ OR } \text{CAL\_aux} \quad (6.3)$$

$$S0_{odd} = \overline{\text{CAL\_edge}} \text{ AND } \text{CAL\_aux} \quad (6.4)$$

715 where an even (odd) pixel is one for which the sum of row + column is an even (odd) number. Thus, for example, in single injection mode the CAL\_edge rising edge throws S0 for even pixels, but S1 for odd pixels. The S0 and S1 assignment options are independent of the cal enable bit in each pixel. The Analog Mode bit of the injection configuration controls whether injection is uniform (mode=0) or alternating (mode=1).

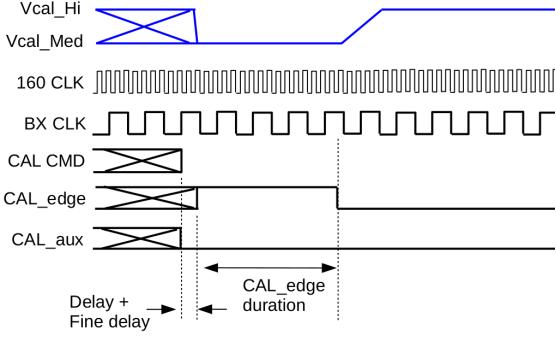
## 720 6.2 Cal Command

The Cal command controls the generation of the two internal signals CAL\_edge and CAL\_aux. For digital injection only the CAL\_edge signal is relevant. The CAL\_edge signal to be generated is specified by the first 14 data bits of the Cal command, while the CAL\_aux signal is specified by last 6 data bits. The detailed bit assignment of the command payload (four 5-bit fields) is shown in  
725 Fig. 32.

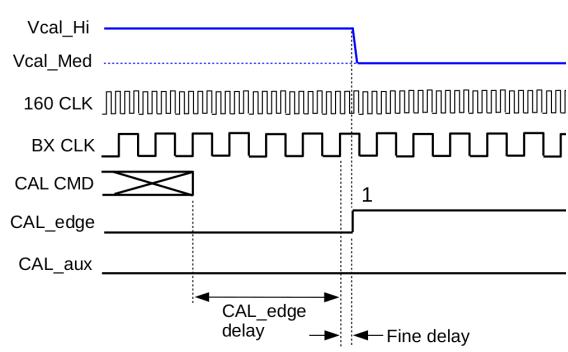


**Figure 32:** Bit assignment of the Cal command payload. Two data frames totaling 20 bits. M= mode bit, E= CAL\_edge parameters, A= CAL\_aux parameters (value and delay). All delays and duration are in units of 160 MHz clock cycles.

The CAL mode bit (M) selects between two behaviors for the CAL\_edge signal: a single step (mode=0) or a pulse (mode=1). The step is always from low to high, so an edge is only produced if the prior state of CAL\_edge was low; if it was high it remains high. Thus, for typical injection it is necessary to first arm the system to ensure CAL\_edge is low, as shown in Fig. 33. The  
730 standard injection sequence is then shown in Fig. 34. In this case voltage at the injection capacitor of the selected pixels is switched from Vcal\_Hi and Vcal\_Med, effectively providing a differential injection voltage that will be insensitive to power and ground local voltage variations across the matrix. The delay value controls the “coarse” delay from the Cal command to the injection (in cycles of the 160 MHz clock), and a global fine delay is added on top of that. This fine delay is  
735 pre-programmed in a global register and is in units of 1.28 GHz clock cycles. This allows precision scanning of the pixel timing. The duration serves no purpose in edge mode and the CAL\_aux signal is unchanged. But note that CAL\_aux must be low the whole time in order for the rising CAL\_edge to switch from Vcal\_Hi and Vcal\_Med. If the CAL\_aux signal were instead high, the switching



**Figure 33:** Timing diagram illustrating the arming of calibration injection, to set both CAL\_edge and CAL\_aux to the correct levels without knowing their prior state. The Cal command parameters are as follows: M=1, delay=1, duration=15, A=0, Adelay=0 (the exact delay and duration values are not important).

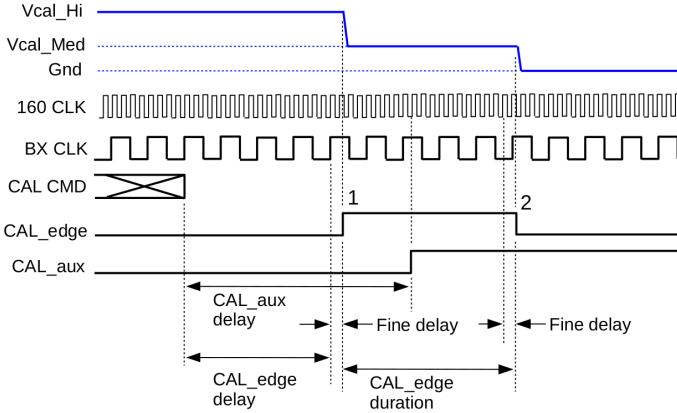


**Figure 34:** Timing diagram illustrating standard calibration injection. Only the CAL\_edge signal is active. The Cal command parameters are as follows: M=0, delay=16, duration=1, A=0, Adelay=0 (the exact delay and duration values are not important).

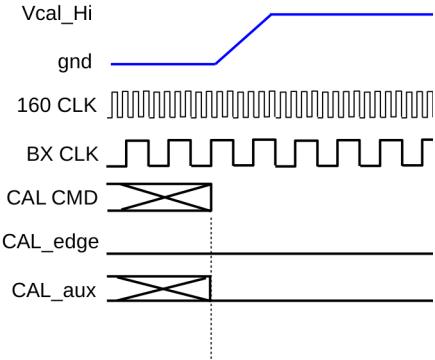
would be from ground to CAL\_edge. This would inject positive polarity charge, which the Front  
740 End is not designed for, but may be of interest for special tests.

In pulse mode (M=1) the CAL\_edge signal will be set to high after the coarse plus fine delay, just as it happened for edge mode, but then it will be set to low after the duration value elapses. Note that if duration is set to zero then CAL\_edge will simply go low after the delay (a duration zero pulse with final state low). Thus, pulse mode with duration zero is the complement of step mode:  
745 the former brings CAL\_edge low while the latter brings CAL\_edge high. Step mode is used to produce two consecutive injections. Starting from the armed state of Fig. 33, the rising CAL\_edge will inject from Vcal\_Hi and Vcal\_Med as usual, but then, before the falling CAL\_edge at the end of injection the CAL\_aux signal is set high, which then causes the falling CAL\_edge to switch from Vcal\_Med to ground. This sequence is shown in Fig. 35. The CAL\_aux signal will be set to the level indicated in the command (A) after the given delay value. The fine scale for the delay allows changing the CAL\_aux value in the middle of a bunch crossing cycle, as is needed to inject charge in two consecutive crossings. Since at the end of this sequence CAL\_edge is low, it's no longer necessary to repeat the Fig. 33 arming sequence. However, the CAL\_aux signal has to be returned low, which can be done with the sequence in Fig. 36.

755 Executing a double injection with a single Cal command in pulse mode is limited the injections being closer together in time than the maximum CAL\_edge duration. To perform a double injection separated by longer times, two separate Cal commands can be used. The first would be a standard injection (Fig. 34), while the second command would accomplish the Vcal\_Med to ground injection as shown in Fig. 37. Note that this is not the same as two consecutive standard injection sequences, because in between standard injections one must re-arm, which produces a positive polarity injection at the time of re-arming. In contrast, in the double injection there is no activity between the two injections. This can be important when studying threshold stability vs. time, for example. One can achieve the same thing using standard injection, but it requires more

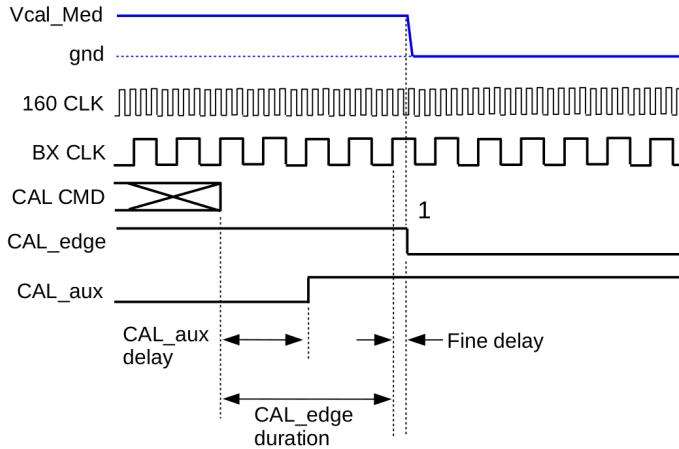


**Figure 35:** Timing diagram illustrating double calibration injection using the pulse mode of the Cal command. The Cal command parameters are as follows: M=1, delay=16, duration=19, A=1, Adelay=25 (the exact values are representative. The Cal\_AUX transition must be between the CAL\_edge rising and falling edges).



**Figure 36:** Timing diagram illustrating the re-arming of the CAL\_aux signal after double injection. The Cal command parameters are as follows: M=1, delay=0, duration=0, A=0, Adelay=0.

commands and, therefore, more time between injections.

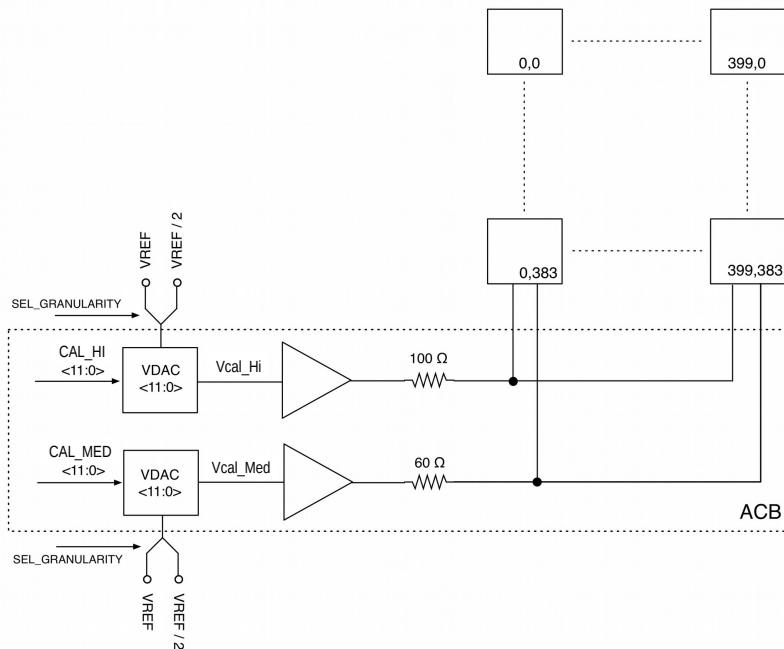


**Figure 37:** Timing diagram for an arbitrarily delayed second injection in a double injection sequence. The Cal command parameters are as follows: M=1, delay=0, duration=16, A=1, Adelay=8 (the exact values are representative. The Cal\_AUX transition must be before the CAL\_edge falling edge).

### 765 6.3 Injection Voltages

The two injection voltages Vcal\_Hi and Vcal\_Med are generated by two 12-bit DACs in the chip bottom as shown in Fig. 38. Using the SEL\_GRANULARITY configuration bit, the circuit can operate either with high dynamic range or with fine step sizes. In the former configuration, the DACs voltage reference is the same as the ADC voltage reference Vref\_ADC (Sec. 12.2), while

770 in the latter case it is  $V_{ref\_ADC}/2$ . As these voltages are relatively high impedance, injection  
 into many pixels at once will introduce a systematic bias, as the current pulse from the combined  
 effect of all injected pixels will cause a voltage shift of the injection voltages. This effect is small  
 and can be ignored for many applications, but should be considered for precision studies. An  
 additional consideration for any precision studies is that the absolute scale of calibration injection  
 775 depends on  $V_{ref\_ADC}$  and on the injection capacitor value,  $C_{inj}$ . Therefore, when converting to  
 units of injected charge (typically in electrons), the conversion is only as good as the knowledge of  
 $V_{ref\_ADC}$  and  $C_{inj}$ . This conversion is typically done automatically within the readout systems.  
 The value of  $V_{ref\_ADC}$  (which depends on the trimmable  $I_{ref}$  main current) should be checked  
 and  $C_{inj}$  should be measured (Sec. 13.8).



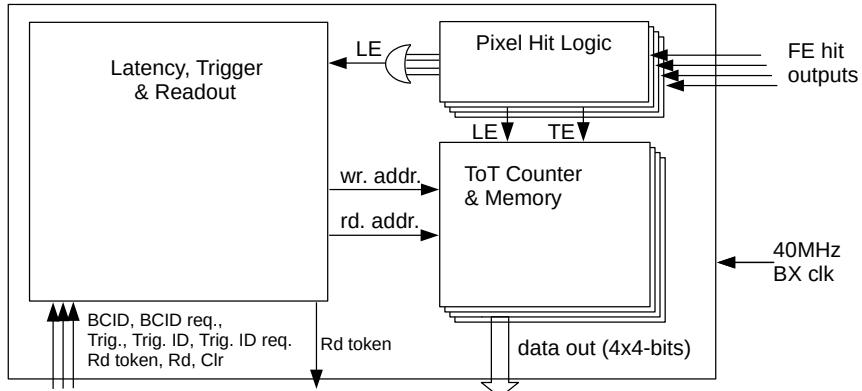
**Figure 38:** Generation of the injection voltages  $V_{cal\_Hi}$  and  $V_{cal\_Med}$ .

## 780 7. Digital Core

The digital core implements all the functionality of digitization, time stamping, storage, trigger retrieval, configuration memory, and injection control for the 64 pixels in the core. All cores are identical and the layout is stepped and repeated to from the matrix. The address of each core is generated by a combinatorial subtracter that subtracts 1 to the address of the previous core, starting 785 from a hard-wired seed address of 47 at the bottom of the matrix (thus the top-most core has address 0). The distribution of the bunch crossing clock and calibration pulse (CAL\_edge) along the column is not done with a global clock tree, but they are time aligned though delays that depend on the core address, compensating for the propagation delay accumulated along the column.

Within the core, the pixel hits are processed in pixel regions (Sec. 7.1), each made of 4 pixels 790 (i.e. in total there are 16 pixel regions in a digital core). Even though each analog island also has 4 pixels, the region pixels are organized in  $4 \times 1$  rows, so two pixels from one island and two from another. The 4 pixels in a region share timing information, while each individual pixel has its own hit processing (Sec. 7.2) along with dedicated ToT counter and memory (Sec. 7.3). The 795 pixel region aspect ratio is chosen to minimize the size of its timestamp and ToT memories for the areas of the detector with the highest hit rates (i.e. end of barrel, where a particle normally hits an elongated cluster of pixels). Additionally the region manages clock gating, which reduces digital power consumption.

### 7.1 4-Pixel Region



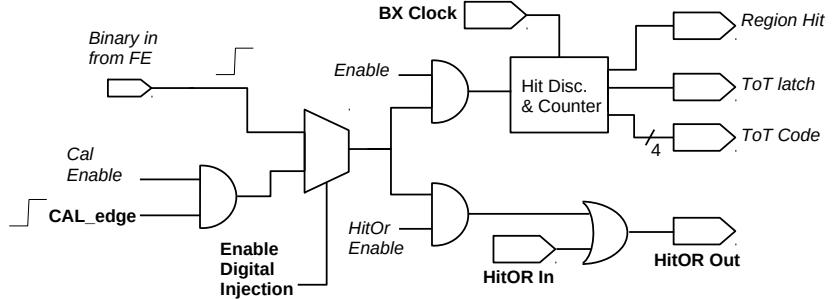
**Figure 39:** 4-Pixel region block diagram. LE is leading edge, TE is trailing edge, BCID is bunch crossing counter value. See text for details.

The pixel region logic contains three main blocks as indicated in Fig. 39: the Hit Logic (four 800 instances), the ToT counter and storage (four instances), and one Latency, Trigger and Readout (LTR) block. Sharing the LTR block among four pixels leads to a more compact layout than if each pixel was independent.

The Hit Logic (Sec. 7.2) determines if an hit is present in a given bunch crossing and drives the ToT counter (Sec. 7.3). The LTR block (Sec. 7.4) is the brain of the region. It keeps track of

805 the timing of all hits, decides which ToT memories to use, and manages triggering, reading and clearing of ToT data.

## 7.2 Pixel Hit Logic



**Figure 40:** Single pixel hit digital processing path. The bold text indicates global signals, while the italic text indicates local pixel signals.

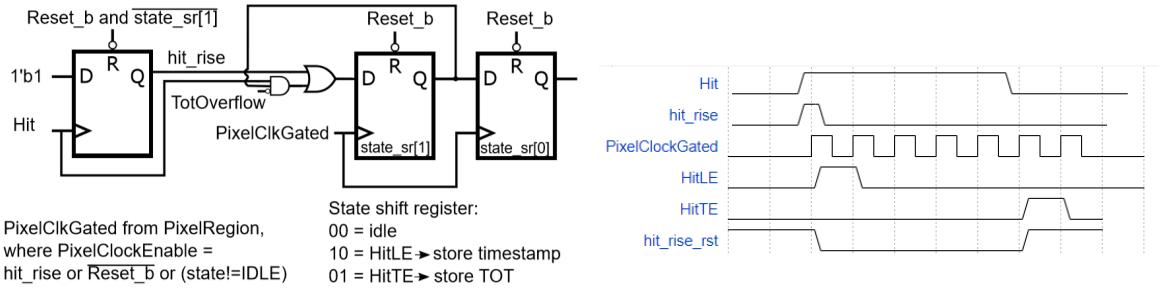
810 The hit output of the analog front end is processed to produce all the needed digital values and signals as shown schematically in Fig. 40. There are two parallel paths with independent enable bits: the DAQ path leading to hits being collected and encoded into the chip data output (Sec. 10, and the HitOr path, which feeds the wired OR core column lines as described in Sec. 13.4. The hit source can be selected to be the analog front end, or digital hit calibration injection, which checks the full digital functionality bypassing the analog front end. The Cal Enable, (data output) Enable, and HitOR Enable bits are independently set for every pixel (Sec. 15.4), while Enable 815 Digital Injection is globally controlled by the CalibrationConfig register (Table 23) The Hit Disc. & Counter block contains logic to detect and synchronize hits (Fig. 41), as well as a ToT counter dedicated to this pixel (Sec. 7.3).

820 Fig. 41 shows the schematic of the logic to process the asynchronous hit output of the analog front end, while Fig. 42 shows the corresponding waveforms. Any hit above threshold is detected, even if it is shorter than a clock cycle. It provides accuracy for ToT measurement at higher effective clock speed than 40 MHz. The block does *not* contain a method to separate “small hits” from “large hits” (as was done in the FE-I4 chip), because the RD53B front end time-walk is very small, resulting in very few late hits.

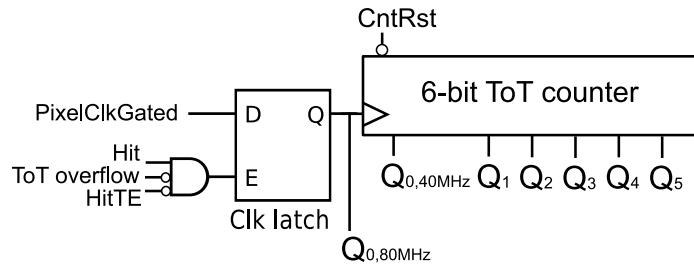
## 7.3 ToT counter and storage

825 ToT is counted independently for each pixel using the 40 MHz clock, but also provides 80 MHz effective resolution by capturing the clock phase with the asynchronous hit. In this case, the clock falling phase is latched, based on the assumption that hits are time-aligned on the rising-edge. 80 MHz ToT counting can be enabled through global configuration (the default is 40 MHz). The ToT counter schematic is shown in Fig. 43. A 6-bit counter is used, but only 4 bits are stored and 830 read out.

In addition to 80 MHz resolution, a dual slope ToT mapping is supported and can be enabled by global configuration. In this case, the 6 bits of the counter are compressed to 4 bits. In case



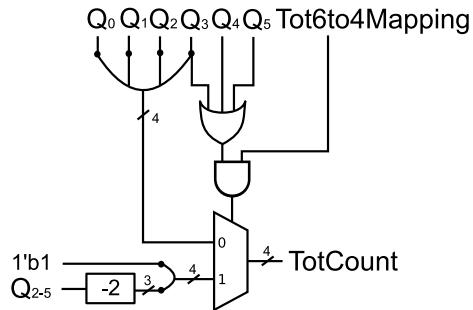
**Figure 41:** Schematic of the logic to process the asynchronous hit output of the analog FE. The reset<sub>b</sub> signal globally provided during initialization. The state shift register controls clock gating, timestamping, ToT counting and storage.



**Figure 43:** ToT counter schematic. At the end of the count, the Clk latch captures the clock falling phase, which is used as the LSB ( $Q_{0,80\text{MHz}}$ ) of the count, achieving 80 MHz ToT counting. The input signals reported on the schematic are the same shown in the pixel control logic in Fig. 41.

the features is disabled, no mapping is performed and only the 4 LSBs are stored. The logic implementing the dual slope mapping is shown in Fig. 44.

835 The meaning of each 4-bit ToT code in terms of true ToT value is shown in Table 4 for the two possible speeds and compression or not. (The inverse mapping for encoding true values can be found in Table 40.) Note that in the default mode (40 MHz and no compression) the output ToT code is the true ToT bin low edge.



**Figure 44:** ToT 6-to-4 bit mapping schematic.

Output 4-bit code	True ToT bin (low edge) [BX]			
	40 MHz speed		80 MHz speed	
	4-bit (DEF)	6-to-4 bit	4-bit	6-to-4 bit
0	0	0	0	0
1	1	1	0.5	0.5
2	2	2	1	1
3	3	3	1.5	1.5
4	4	4	2	2
5	5	5	2.5	2.5
6	6	6	3	3
7	7	7	3.5	3.5
8	8	8	4	4
9	9	11	4.5	5.5
10	10	15	5	7.5
11	11	19	5.5	9.5
12	12	23	6	11.5
13	13	27	6.5	13.5
14	$\geq 14$	$\geq 31$	$\geq 7$	$\geq 15.5$

**Table 4:** True ToT value in bunch crossing (BX = 25 ns units) for each output ToT 4-bit code, depending on speed (40 or 80 MHz) and compression (4 bit or 6-to-4 bit) settings. Always the low edge of the true ToT bin is shown. For example code 3 having a true ToT low edge of 3 means the true ToT was at least 3 bunch crossings and at most  $x$ , where  $x$  is the true ToT low edge of the next code (4 in this case). The last bin (code 14) has no high edge and includes all overflows. Code 15 means “no hit” and should never be seen because unhit pixels are internally suppressed.

In any mode, the ToT code that is read out goes from 0 to 14. Code 15 is reserved in the pixel region to identify non-hit pixels. Also, if the ToT counter reaches maximum while the pixel comparator output is still high, the counting concludes and the maximum ToT is recorded. The pixel region clock is gated off whenever there is no ToT counting taking place.

The ToT storage has 8 locations, 4 bits each. The value of the ToT counter is stored once the conversion is finished, given by a trailing edge pulse or by the counter reaching max count. Which of the memories the ToT is stored in is fixed at the start of the ToT conversion by a write address from the LTR, shown as wr. addr. in Fig. 39. This is common to the 4 pixels in the pixel region.

The pixel ToT memory bank has a 4-bit output port. Which ToT memory is presented on this port is given by a select address from the LTR block (rd. addr. in Fig. 39).

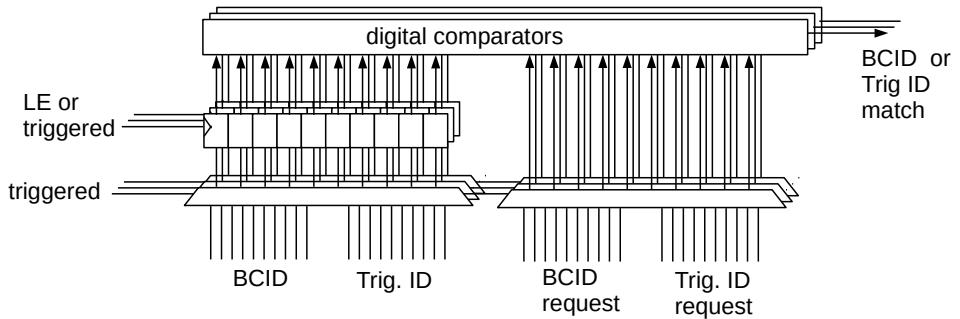
#### 7.4 Latency, Trigger and Readout (LTR) block

If any pixel in the pixel region is hit, the timestamp is stored in a memory common to the whole pixel region. If multiple pixels fire in the same region in the same crossing, still only one memory is written. The timestamp is necessary to determine when the trigger latency for a certain event expires and the storage elements are therefore often referred to as latency memories.

Each 4-pixel region has 8 latency memories just as each pixel has 8 local ToT storage registers. Each of the 8 ToT registers of a given pixel is associated with one region latency memory (hard-

wired). This way, when one pixel is counting ToT it does not prevent the other pixels in the region from being hit, i.e. there is no region dead time.

Each memory consists of a 9-bit buffer and comparator, as shown in Fig. 45, which is a low power solution compared to local latency counters. The 9-bit memory stores the value of a global bunch crossing counter (BCID) distributed from the chip bottom (9 bits translates to a maximum trigger latency of  $2^9 - 1 = 511$  bunch crossings or  $12.8 \mu\text{s}$ ). Every subsequent bunch crossing, this value is compared to a delayed bunch crossing counter (BCID request), also global, that is delayed by the trigger latency relative to the BCID. Both are Gray counters so that only one of the 9 bits changes every bunch crossing. If the event is triggered, the same memories are also used to store the trigger ID value.



**Figure 45:** Latency memory block diagram. The same memory and comparators are used to store BCID values or trigger ID values, as there is never a case when both need to be stored at the same time. Only 3 instances of each circuit are shown in the figure, but in reality there are 8 9-bit memories and comparators in the region.

The latency memories are 9 bits wide, allowing for a programmed latency of up to 511 bunch crossings. Each memory cell also contains a 2-bit state register to identify whether the memory is idle, triggered, to be read out or to be cleared. The latter is only used in second level trigger mode or for event truncation. The memory is in idle state until a LE signal arrives and with each new LE a new memory location is written. When a location reaches the programmed latency (BCID match) the presence of a trigger is checked. If no trigger is present, the latency buffer and associated ToT memory are released (marked available). If a trigger is present, then the ToT memories are marked triggered (not available) and a trigger ID is stored to label the hits for later readout.

When a region hit is selected by a trigger, the latency buffer mechanism is reused for queuing the hit for readout. The stored BCID value is overwritten with a trigger ID value, which is compared to a trigger ID request value. When the trigger ID request matches the stored value, the LTR will hold the read token and select ToT data to be placed on the output. The read token travels from one region to the next to scan a core column for data. The generation of the trigger ID is explained in Sec. 9.3.

The Precision ToT modules in the chip bottom reuse the region latency memory mechanism and reading, Sec. 13.7 may give further insight about the region operation.

## 7.5 Pixel Addressing

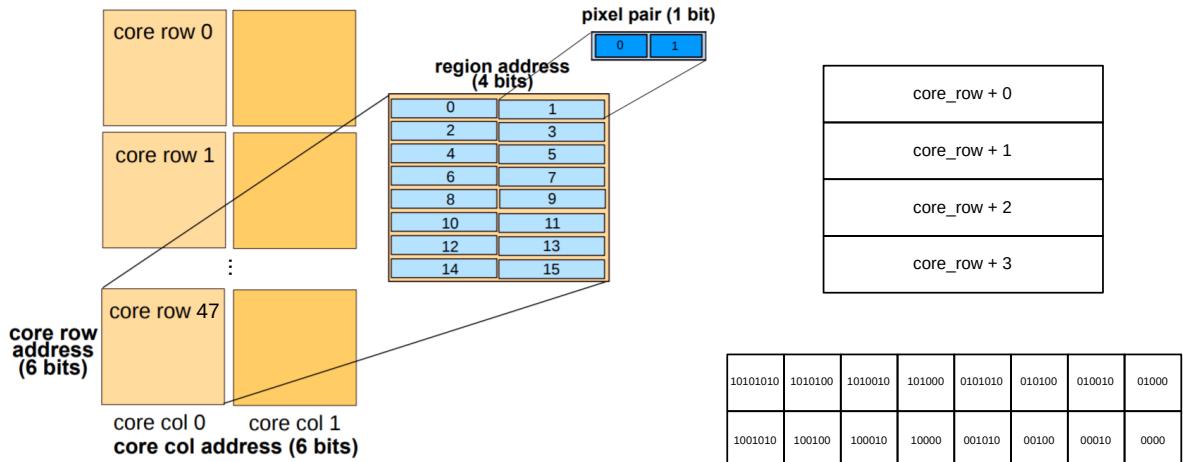
The pixel addressing is hierarchical, first in cores (like postal codes) and then regions within a core (like the street address). This structure is shown graphically in Fig. 46.

For pixel configuration, the basic unit which can be addressed is a pixel pair (see Fig. 46). This is achieved by writing to two registers of the chip global configuration. The Core\_Col and Core\_Row values are preserved in the global configuration registers 1 and 2, respectively, but the four Region\_in\_Core are divided between the two registers as follows. Additionally, there is a Pair\_in\_Region bit need (17 total bits instead of 16), because configuration is written in pixel pairs rather than pixel regions.

```
Register 1= [7:2]=Core_Col, [1]=Region_in_Core[0], [0]=Pair_in_Region
Register 2= [8:3]=Core_Row, [2:0]=Region_in_Core[3:1]
```

In this way, Register 1 identifies the pixel pair column address, while Register 2 contains the pixel row address.

In the RD53B data output (Sec. 10.4) the basic unit is instead a quarter core, which contains two rows of 8 pixels. Thus a core from Fig. 46 is vertically divided into four quarter cores, with the numbering of the quarter cores as shown in Fig. 47. A compressed binary tree scheme is used to address each pixel in a quarter core, as described in details in Sec. 10.4. Fig. 47 also shows an example of the compressed bit codes for all 16 cases of a single hit pixel in the quarter core.

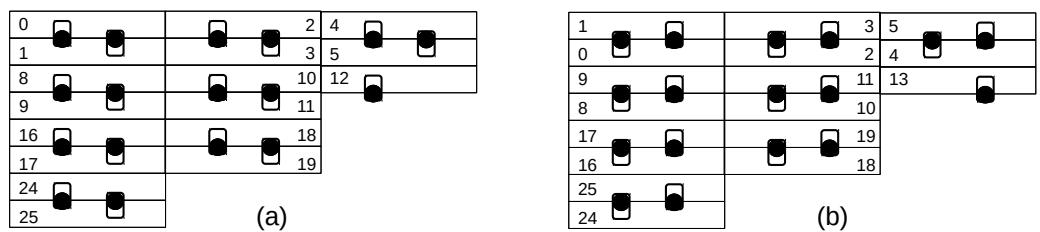


**Figure 46:** Pixel core addressing scheme, down to pixel pairs for pixel register configuration.

**Figure 47:** Numbering of quarter-cores for readout (top) and compressed binary code for each single hit pixel (bottom).

### 7.5.1 $25\text{ }\mu\text{m} \times 100\text{ }\mu\text{m}$ pixels

While for  $50\text{ }\mu\text{m} \times 50\text{ }\mu\text{m}$  sensors the chip pixel numbering will carry over unchanged to the sensor, for  $25\text{ }\mu\text{m} \times 100\text{ }\mu\text{m}$  a mapping is needed to know which sensor pixel is connected to which chip channel. This mapping is determined by the sensor metalization and there are two possible mappings as shown in Fig. 48.

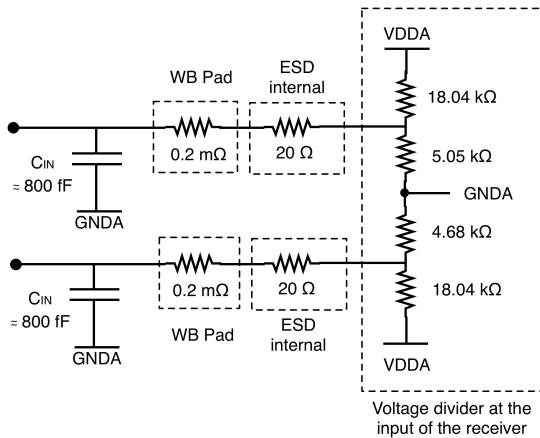


**Figure 48:** Two options (a and b) for mapping of  $25 \mu\text{m} \times 100 \mu\text{m}$  pixel sensors to the core pixel address. Which option is correct is determined by the sensor metalization. The top left corner of an 8 by 8 pixel core is shown.

## 8. Commands and Configuration

RD53B is fully controlled with a 160 Mbps differential serial input stream with a custom, DC-balanced encoding described in Sec. 8.2. The differential receiver circuit is described in Sec. 8.1. The received signal, without any processing, can be optionally repeated on the general purpose differential outputs (Sec. 13.1). The command input also contains an activity detector that will cause a reset when the rate of transitions falls to very low value (Sec. 3 and 14). A Clock and Data Recovery circuit (CDR) recovers the input bitstream and also produces the internal clocks for the chip, based on the transitions on input stream, as described in Sec. 14. A dedicated command (PLL\_LOCK) equivalent to a clock pattern is provided to ease locking the internal phase locked loop (Sec. 14).

### 8.1 Receiver Circuit



**Figure 49:** Equivalent circuit for differential receiver input.

The CMD receiver is implemented as a differential amplifier with a rail-to-rail input stage. The inputs are connected to an on-chip resistor bias network in the  $k\Omega$  range, which allows the receiver to be ac-coupled to the serial input stream. The resulting input common-mode voltage is  $0.21 \times VDD\_PLL$  (260 mV for nominal 1.2 V supply). The bias network also adds a small offset voltage to the differential input signal to keep the receiver in a static input state in case of a broken signal connection. A termination resistance is not implemented in the CMD receiver to allow multiple RD53B chips to be connected to the same CMD line (a multi-drop configuration with one termination at the end). The differential receivers for the data aggregation inputs (Sec. 11) use the same rail-to-rail input stage but have an internal  $100\Omega$  termination resistor (removed in V1.1). The input impedance of the receiver, together with the ESD protection and wire bond pads, have been simulated with extracted parasitics. The capacitive and resistive contributions are shown Fig. 49.

## 8.2 Command Protocol

The input stream is a continuous sequence of commands. All commands are built in 16-bit frames made out of two 8-bit symbols. As the bitrate is 160 Mbps each frame spans four periods of the 40 MHz bunch crossing clock. Commands that are one frame long (four BX clocks) and are called short commands (Sec. refsec:short-commands) of which Triggers are an example. Frames are interpreted one at a time and short commands are executed immediately, while long (multi-frame) commands (Sec. 8.2.2) are executed after their last frame is received. Long commands have the property that they can be interrupted by short commands without the need of restarting the interrupted command. This gives the ability to send Trigger commands (which are short) whenever needed, and to send long commands during data taking without worrying if trigger might be coming<sup>3</sup>.

The command input is intended to be shared by multiple chips (multi-drop). Commands can be *broadcast*, in which case all chips sharing the command input will execute them, or *addressed*, in which case only the chip with the selected address will execute it and all other chips receiving it will ignore it. A chip can have one of 16 possible chip ID values (set by 4 wire bonds to ground overriding internal pull-up resistors). The first frame of addressed commands consists of an 8-bit symbol identifying which of the 7 commands it is, and a data 8-bit symbol specifying a chip ID. Addressed commands can also be sent in broadcast mode by specifying a chip ID value greater than 15. A chip that receives a command not broadcast or addressed to it will still process it (so as not to produce “unexpected data frame” errors), but will not execute it. The PLL\_LOCK, Sync, and trigger command are always broadcast, while all others are addressed.

Each 16-bit frame is exactly DC balanced. DC balance is needed for A/C coupling, reliable transmission, and clock recovery. The symbols used also provide error detection<sup>4</sup>. There is a unique sync frame (used to perform frame alignment as explained in Sec. 8.3), plus 3 kinds of TTC (Trigger, Timing and Control) frames: trigger, command, or data. TTC frames contain two 8-bit symbols which are themselves DC-balanced. Furthermore, symbols that begin or end with three or more 1’s or 0’s are not used, resulting in a maximum run length of 4, except for the sync frame which has a run length of 6. The valid symbols and commands are given in Tables 5, 6, and 32. There is one sync frame, 7 non-trigger commands, 15 trigger symbols allowing the encoding of 15 trigger patterns (Tables 5, 6), and 32 data symbols allowing the encoding of 10 bits of content per data frame or 5 bits of chip ID per command frame (Table 32). All valid symbols are allowed to be used as trigger tags in the trigger frame; thus there are 54 possible tags (see Sec. 9). A single bit flip always results in an invalid symbol (formally, all symbols are separated by a Hamming distance of 2).

RD53B interprets the protocol in three phases (which will be transparent to the user): Initialization 8.3, Data Transmission 8.4 and Decoding 8.5. The decoding timing and exception handling are covered in Sec. 8.6

---

<sup>3</sup>Note that the RD53B command set is similar, but not identical to that used in the RD53A chip, and thus it is not backwards compatible with RD53A control sequences.

<sup>4</sup>All these properties could have been obtained with 8b/10b encoding, but the 10-bit frame length of 8b/10b would have required 200 Mbps link speed in order to maintain an integer number of bunch crossings per frame, as needed for synchronous triggering. The 160 Mbps bitrate of the RD53B custom protocol makes for better transmission on low mass cables and can be directly driven from GBT e-links.

Command	Encoding		(T)ag, (A)ddress or (D)ata 5-bit content									
Sync	1000_0001	0111_1110										
PLLlock	1010_1010	1010_1010										
Trigger	ttt_tttt	Tag[0..53]										
Read_trigger	0110_1001	ID<4:0>	00,T<7:5>	T<4:0>								
Clear	0101_1010	ID<4:0>										
Global Pulse	0101_1100	ID<4:0>										
Cal	0110_0011	ID<4:0>	D<19:15>	D<14:10>	D<9:5>	D<4:0>						
WrReg(0)	0110_0110	ID<4:0>	0,A<8:5>	A<4:0>	D<15:11>	D<10:6>	D<5:1>	D<0>,0000				
WrReg(1)	0110_0110	ID<4:0>	1,xxxx	xxxxx	N×(D<9:5>)	D<4:0>						
RdReg	0110_0101	ID<4:0>	0,A<8:5>	A<4:0>								

**Table 5:** List of protocol commands/frames and address or data fields associated with each. Unused padding bits are indicated by “0”. Double vertical lines denote frame boundaries. ttt\_tttt is one of 15 trigger commands (Table 6). The before-encoded bit content of chip ID, Address or Data is shown. These are all encoded as 8-bit data symbols (Table 32).

965    **8.2.1 Short Commands**

**PLL\_LOCK** (broadcast only):

This command allows a clock pattern to be sent to the chip without any action being executed by the command decoder. The clock pattern is needed to efficiently lock the Phase Locked Loop (PLL) to the correct frequency at start of operation (Sec. 14). Once locked, the PLL no longer needs a perfect clock pattern and regular commands and sync frames can be sent. This command can also be used as an idle when there is nothing to be sent during normal operation. This is equivalent to a No Operation (NOOP) command in many processors, but we do not use that terminology here. It repeats the same 8-bit symbol twice to produce a clock pattern (Table 5).

**Sync** (broadcast only):

The Sync is the only command where the two 8-bit symbols used are not themselves DC balanced (both together the 16 bits are DC balanced). This is what makes it unique and allows it to be recognized for frame alignment.

**Clear:**

980    Clears the entire data path. All pending triggers and stored hits will be erased.

**Global Pulse:**

The global pulse command sends a single pulse with a duration of  $2N$  bunch crossings, where  $N$  is the value of the 8-bit register GlobalPulseWidth (Table 23). The value  $N = 0$  is treated like  $N = 1$ . A pulse shorter than two bunch crossings is not possible. The global pulse can be routed to different places of the chip and has many uses. It can provide reset signals, control the ring oscillators, the ADC, etc. The global pulse routing table is 24.

**Trigger** (broadcast only):

Because one 16-bit frame spans 4 LHC bunch crossings, the trigger command must specify

a 4-bit map indicating which of the 4 bunch crossings are actually triggered; hence 15 trigger patterns. The triggering is synchronous, and therefore trigger frames must be sent at specific times. The second symbol in a trigger frame can be any legal symbol and is interpreted as one of 54 possible 6-bit tag bases to identify the trigger(s) in later readout (see Sec. 9.2). The mapping from symbol to tag base number is given in Table 34. The trigger tag will be returned with the data corresponding to that trigger (See Sec. 10).

Symbol Name	Encoding	Trigger Pattern	Symbol Name	Encoding	Trigger Pattern
Trigger_01	0010_1011	000T	Trigger_08	0011_1010	T000
Trigger_02	0010_1101	00T0	Trigger_09	0011_1100	T00T
Trigger_03	0010_1110	00TT	Trigger_10	0100_1011	TOTO
Trigger_04	0011_0011	0T00	Trigger_11	0100_1101	TOTT
Trigger_05	0011_0101	0T0T	Trigger_12	0100_1110	TT00
Trigger_06	0011_0110	0TT0	Trigger_13	0101_0011	TTOT
Trigger_07	0011_1001	0TTT	Trigger_14	0101_0101	TTTO
			Trigger_15	0101_0110	TTTT

**Table 6:** List of trigger symbols used to encode the 15 possible trigger patterns spanning four bunch crossings. Note there is no 0000 pattern as that is the absence of an trigger. The Trigger\_01 (000T) means that the first bunch crossing of the trigger window is meant to be readout, and the extended tag returned will have 00 following the supplied tag base.

### 8.2.2 Long Commands

#### Cal (Calibration Injection):

The same command is used for both analog and digital injection. Whether injection will be analog or digital is decided by global configuration register CalibrationConfig, but the Cal command produces the same output regardless. To understand the Cal command it is necessary to understand how the calibration injection circuit works. Therefore, the description of the command was given in Sec. 6.

#### WrReg(0) (Write Register, single):

The WrReg command has two modes: single write and multiple writes to register 0. The command frame is the same and the distinction between single and multiple is made by the first bit of the payload (0=single, 1=multiple). The WrReg(0) or single has 9 bits of Address and 16 bits of Data. Up to 512 16-bit wide registers can be addressed, but not all 512 possible register addresses are used. If an attempt is made to write to an unused address, the command will do nothing and no warning will be generated. The register memory map is given in Table 23. This command does not produce any output from the chip.

#### WrReg(1) (Write Register, multiple):

The WrReg command has two modes: single write and multiple writes to register 0. The command frame is the same and the distinction between single and multiple is made by the first bit of the payload (0=single, 1=multiple). The WrReg(1) or multiple must have the address value set to 0. It can only be used to initiate multiple writing to register 0. Register 0 is a virtual register called PIX\_PORTAL, used to write and read pixel configuration (Sec. 8.8).

1020

Following a WrReg(1) command, one may send data frames to the chip (as many as desired) without any preceding command. All these data frames will be written to the PIX\_PORTAL (register 0). This must be done in conjunction with auto-increment (see Sec. 8.8). This permits very efficient transfer of data to the PIX\_PORTAL. The write multiple mode remains in effect until a new long command is received (short commands will be executed and not end the WrReg(1) command mode). Note that the chips not addressed by the WrReg(1) command will still recognize the multiple write mode and will therefore not issue “unexpected data frame” warnings, but they will not write the data to their register 0. The placement of the 10 bits from each data frame into the 16 bits of register 0 is described in Sec. 8.8

1025

#### **RdReg (Read Register):**

1030

This command has 9 bits of address and no data. It initiates the readout of the addressed register. Address 0 is special: it is the pixel register as described in Sec. 8.8. The 16-bit register value is returned in the data stream as described in Sec. 10. Not all 512 possible register addresses are used. If readback of an unused address is requested, the data value returned will be 0, the address returned will be the requested (non-existent) one, without any warning generated. The register assignment list is given in Table 23.

1035

#### **RdTrig (Read Trigger):**

This command has an 8-bit extended tag value. In two-trigger mode, it selects a previously received tag for readout. It is not useful in single trigger mode. See Sec. 9 for details.

### **8.3 Command Protocol Initialization**

1040

Until the PLL is locked and produces a stable chip clock, the command decoder will be in its reset state. During this period, PLL\_LOCK frames should be sent to the chip. The transitions in the string of PLL\_LOCK frames will allow the clock recovery circuit to lock to the correct 160 MHz frequency. The user does not know when the PLL has locked, but simply sends PLL\_LOCK frames for a long enough time that the lock cycle is surely completed (see Sec. 14). For debugging, the PLL lock condition can be observed in the recovered CMD output of the general purpose LVDS, which is a default output (see Sec. 13.1). At this point the protocol initialization begins. Before any command decoding, the input bitstream is processed by the Channel Synchronizer circuit (Fig.50), and the initialization correctly sets up this circuit.

1045

The sync pattern (Table 5) can not be produced through any combination of TTC frames and therefore can be searched for to lock the correct frame boundaries (the search procedure is explained in the next paragraph). Sync frames must be sent at the start of operation so that the framing can be locked (this different from PLL lock!). It is mandatory to send one sync frame in every 32 frames or so in order to maintain lock or allow the command decoder to re-lock if lock was lost. If no sync frames are received in a long time frame lock will be declared lost and the command decoder will stop interpreting commands until a new lock is acquired. Typically at the start of operation (power up) there are no commands or triggers to immediately send, and so sending a large number of sync frames to ensure initial lock is not a problem. The channel synchronizer lock is available in the chip status CMOS output (Sec. 13.1).

1055

Using the 160 MHz recovered clock, the channel synchronizer will search for sync symbols and count each valid appearance of this pattern in 16 separate channels (one channel for each possi-

ble frame alignment). When the count for one of the channels,  $i$ , reaches a threshold  $N_{lock}$ , *sync lock* is declared as acquired, channel  $i$  is adopted as the correct channel, and the count of the remaining 15 channels is reset. The value  $N_{lock}$  has default value of 16 and can be changed in configuration register ChSyncConf (Table 23). At the start of transmission the command decoder will not interpret any commands until it has received  $N_{lock}$  Sync commands. Thus one should begin transmission by sending at least  $N_{lock}$  Sync commands. The 40 MHz bunch crossing clock is generated as the bit pattern 1100110011001100 aligned to channel  $i$ . Thus there are 4 bunch crossings with a fixed phase relationship to the sync frame, which can be labeled  $BXa..BXd$ . The counting of sync sequences continues in all the channels, but every new sync sequence detected on the lock channel  $i$  resets the count for all the other channels. If the count for a channel that is not the lock channel ever reaches a threshold of  $N_{lock}/2$ , lock is declared lost, and a new sync lock is acquired on the first channel that reaches the locking threshold  $N_{lock}$ . This allows for continuous channel monitoring and automatic sync lock as long as enough sync symbols are transmitted. Additionally, if zero sync frames are received in the lock channel within 64 frames (regardless of other channels), lock will be declared lost and no further commands will be decoded until a new lock is acquired. This value is hard-wired and cannot be changed. This is useful to prevent prolonged, random input due to an upstream exception from corrupting the chip operation, but makes it mandatory to regularly send Sync symbols.

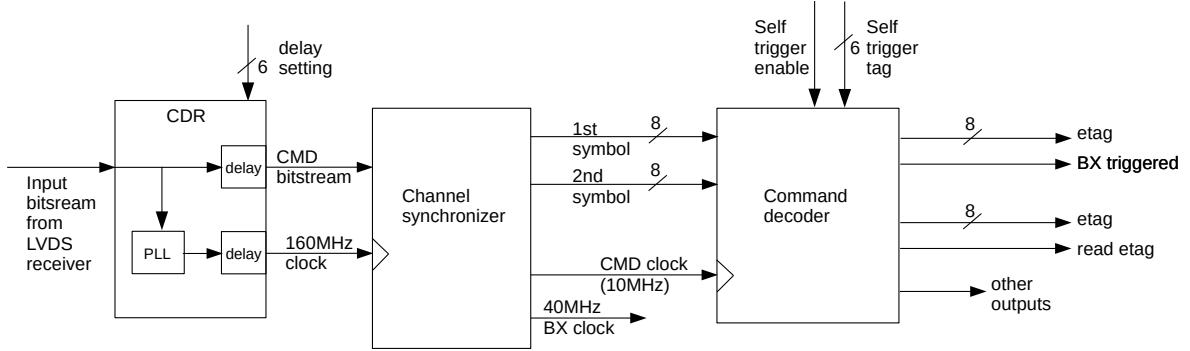
#### 8.4 Command Protocol Transmission

During transmission a correct sequence of commands is sent to control the chip. Trigger frames are sent at specific times, and the “space between trigger frames” is filled with commands (including the required Syncs). Long commands are decoded regardless of intervening short commands. The PLL\_LOCK command can be used as an idle frame, as it has the most transitions and will therefore best maintain PLL operation. Sync commands can also be used as idles, since they must be sent periodically anyway, but they have the fewest transitions, so are not ideal for maintaining PLL lock. The best approach is therefore to always send Syncs every 32 frames and PLL\_LOCK commands in between if and when there is nothing else to send.

#### 8.5 Command Protocol Decoding

The data bits recovered from the locked channel are fed to the Command Decoder as shown in Fig. 50. In the absence of a sync lock, nothing is fed to the command decoder, so until a lock happens no commands will be interpreted. The locked condition guarantees that the bits fed to the command decoder are correctly aligned with the 40 MHz bunch crossing clock. Protocol consistency is ensured by checking that the decoded frames are valid and also that they match what is expected (analogous to checking both spelling and grammar). The 16 bits are fed to the command decoder with a parallel bus. In case of correct detection, the indicated action is performed according to the command type and Chip ID. All symbols are always checked and decoded, even if they follow a Chip ID that does not match the wire bonded ID. However, the Command Decoder will act on the rest of the chip only if the command is a trigger, if decoded Chip ID matches the wire bonded ID, or if the decoded broadcast bit is 1 (the PLL\_LOCK command is not addressed, but has no internal action- no operation). The detection of an invalid symbol is handled differently

depending on the frame and expectation (current state). The handling of exceptions is shown in Table 7.



**Figure 50:** Clock and command recovery and decoding path from chip input to internal signals, showing trigger pulses and tags in particular. Other outputs of the command decoder, such as global register address and write signal, not shown. 16-bit Command patterns are successively loaded into the Command Decoder with the correct frame alignment as determined by the Channel Synchronizer.

Frame received	Frame Expected	Error/Action
invalid, data	data	Aborted command
data, invalid	data	Aborted command
invalid, invalid	data	Aborted command
invalid, data	not data	Lost trigger
invalid, invalid	not data	Corrupted frame
invalid, sync	any	Corrupted sync
sync, invalid	any	Corrupted sync
invalid, command	any	Execute with warning
command, invalid	any	Execute with warning
trigger, bit-flip (*)	any	Execute w/tag base 54
trigger, invalid (*)	any	Execute w/tag base 55
command, command	data	Ignored command

**Table 7:** Command Decoder response to invalid or unexpected symbols. (\*) bit-flip refers to an 8-bit pattern produced from flipping a single bit in a valid symbol, while invalid references to any other invalid 8-bit pattern.

## 8.6 Command Protocol Timing

The decoded commands are executed 25 ns after the end of the last frame of the command data. “Executed” means that the outputs of the Command Decoder block in Fig. 50 change state, which happens on a rising edge of the beam clock. In many cases the execution is instantaneous (outputs

change state and that's it), but the Trigger, Cal and Global Pulse commands have a delay and duration. The trigger command sends 1 to 4 pulses in 4 consecutive beam clock cycles, and thus  
1105 is completely finished before a new command can be completely received (since 1 frame is 4 beam clock cycles). The Cal and Global pulse commands can occupy their respective output lines (CAL\_edge, CAL\_aux, and Global\_pulse) for many clock cycles. A new Cal or Global pulse command should not be sent before the prior such command is complete (up to the DAQ to ensure this), but any other command can be sent and will be executed normally.

## 1110 8.7 Global Configuration

The global configuration is stored in 16-bit registers which are accessed like a RAM with the write and read register commands of Table 5. Each register has a default value that is provided as explained in Sec. 3.2.1. The main table of register names, content, and default values is given in Sec. 15.2.

## 1115 8.8 Pixel Configuration

Each pixel has 8 bits of local configuration as detailed in Sec. 15.4. From the point of view of the write and read register commands, each pixel is seen as one half of one configuration data register. All pixels are paired as shown in Sec. 7.5.

The 8 pixel bits are divided into 5 TDAC bits (threshold tuning bits) and 3 enable bits (also  
1120 known as mask bits). These two types of bits can be written together or independently (always for two pixels at a time). Thus one can choose to write all 8 bits at once, only the 5 TDAC bits, or only the 3 enable bits. The single write register command (WrReg(0)) of Table 5 always writes al 8 bits of both pixels, where the 16 bit data frame is subdivided as follows:

**Single Write:** left-pixel(TDAC[15:11]HitBus[10]InjEn[9]Enable[8]),  
right-pixel(TDAC[7:3]HitBus[2]InjEn[1]Enable[0])

1125 The multiple write register command (WrReg(1)) instead writes the mask bits or the TDAC bits depending on the Mask or TDAC bit of global configuration register PIX\_MODE. The mapping from 10-bit data frame to two pixel TDAC or mask bits is as follows:

**PIX\_MODE[1] = 0:** unused[9:8], right-pixel-mask[7:5], unused[4:3], left-pixel-mask[2:0]  
**PIX\_MODE[1] = 1:** right-pixel-TDAC[9:5], left-pixel-TDAC[4:0]

1130 Internally, the writing and reading of configuration values from the pixels uses an addressed bus to every  $2 \times 1$  pixel pair. All reading and writing is done two pixels at a time in a given column of 4-pixel regions. (See Sec. 7.5 for address encoding). However, multiple core columns can be written in parallel, while readback can only take place from one pixel-pair column at a time. There are thus two write modes, single pixel-pair and broadcast, while read is always single pixel-pair.

1135 The write and read operations are controlled by three global registers, the REGION\_COL, REGION\_ROW, and PIX\_MODE configuration registers. The pixel data is written into or retrieved from global register 0 (PIX\_PORTAL) with the normal write and read register commands (see Sec. 8.2). This is a virtual register acting as a portal to whatever pixel pair is pointed to by the column and row config registers (called PIX\_PORTAL). The row register has a special feature called

1140 auto increment (Auto Row), which reduces the number of commands needed to fully configure the chip. This mode is enabled by a configuration bit and increments the row register value after every write or read operation to PIX\_PORTAL.

1145 The typical pixel matrix configuration write sequence, using the write single register command, is given in Table 8. Note that this takes 77200 (73008) commands for ATLAS (CMS) chips to accomplish. These numbers should be multiplied times 4 to obtain number of frames, and each frame takes 100 ns to transmit. If one is only configuring a chip, it will therefore take about 30 ms. For the case of configuring during data taking (called trickle configuration), much of the command bandwidth will be taken up by trigger commands, and configuration will therefore take longer. The worst case is two-level trigger operation with 4 MHz L0 + 1 MHz L1 trigger operation. This will use up 60% of the command bandwidth. We should also remember that 6% of the command bandwidth must be used to send periodic Sync commands. With only 34% of the command bandwidth available, 77200 Write Register commands will take 88 ms instead of 30 ms. With some DAQ overheads we assume 100 ms. So for a 4-chip module trickle configuration in the worst case will take 400 ms. (If more chips share the same command line it will take proportionally longer). Writing a uniform (all pixels the same) configuration is 50 (54) times faster for ATLAS (CMS), because each Write Pixel command can write to all core columns Table 9. Alternatively, using the multiple instead of single Write Register command means one frame instead of 40 frames per write, which will reduce the above 88 ms to 22 ms (100 ms for a 4-chip module in worst case of trickle configuration). The readback of the pixel configuration for the whole matrix can proceed exactly as shown in Table 8, substituting the Read Register command instead of Write Register. This can be carried out in broadcast mode to any number of chips in parallel, so will always take 50 ms (half as much as writing a single chip because the read register command is two frames instead of four).

1155 Writing or reading an individual, arbitrary pixel pair follows steps 1-3 of Table 8. For calibration operations it is often required to write only the mask bits many times to shift a pattern through the matrix, leaving the TDAC bits alone. This can be done with broadcasted commands (same mask for all chips even though the TDACs are different), and it must use the write multiple command as the write single command always writes all the 8 configuration bits per pixel. Writing masks to a single pixel at-a-time will take 77200 write operations as in Table 8, but one frame per pixel write instead of four (still four frames per write for steps 1 and 2), resulting in 78400 frames, which takes 8.3ms if all the command bandwidth (minus 6% for Syncs) is used, or 23 ms in the worst case of trickle calibration. Writing one row at a time will take 1/50 of this per mask, regardless of the number of chips, as it is done in column broadcast mode using broadcasted commands. It can even be faster if not all rows need a new mask each time.

1160 When used in a radiation environment it is possible to write to an non-existing pixel address at the end of a configuration operation. This will make pixel configuration less sensitive to accidental SEU/SET caused overwriting a pixel register.

Step	Command	Address	Explanation
1	Write_Register	column and mode config	set columns 0-1 and auto row mode
2	Write_Register	row config	set row 0
3	Write_Register	0	config first 2 pixels
4	Write_Register	0	config for next row 2 pixels
386	Write_Register	0	config for last row 2 pixels in cols 0-1
387	Write_Register	column and mode config	set columns 2-3 and auto row mode
388	Write_Register	row config	set row 0
389	Write_Register	0	config for next row 2 pixels
77200	Write_Register	0	config last 2 pixels in chip

**Table 8:** Sequence to write an arbitrary pixel configuration to ATLAS size chip using write register single commands. Each column pair takes 386 commands, times 200 column pairs leads to 77200 commands. For readback replace Write\_Register 0 with Read\_Register 0 commands.

Step	Command	Address	Explanation
1	Write_Register	column and mode config	set broadcast, cols. 0-1, and auto row mode
2	Write_Register	row config	set row 0
3	Write_Register	0	config all pixels, first row in cols 0-1
4	Write_Register	0	config all pixels, second row
386	Write_Register	0	config all pixels, last row
387	Write_Register	column and mode config	set broadcast, cols. 0-1, and auto row mode
388	Write_Register	row config	set row 0
389	Write_Register	0	config all pixels, first row in cols 2-3
1544	Write_Register	0	config last 2 pixels in cols 6-7

**Table 9:** Sequence to write a default (all pixels the same) configuration for ATLAS size chip. Only the first core column (columns 0-7) are written because all core columns will be “CC-ed” in parallel.

## 9. Trigger Processing, Tags, and Data Flow

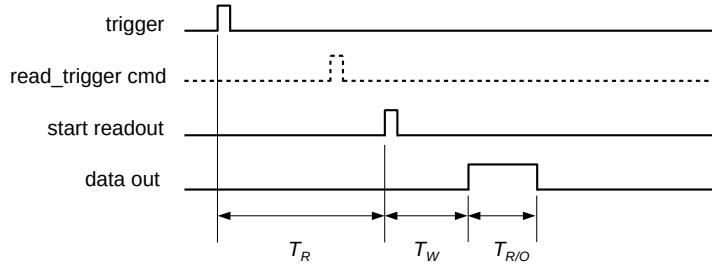
A simplified description of the chip triggered readout is as follows.

- 1180 1. A trigger command is received, including an identifier called tag,
2. The tag is stored along with a “read or clear” bit, and the hits from the appropriate bunch crossing are assigned to this tag (but left in the pixel matrix),
3. An countdown timer is started dedicated to this tag. The (programmable) start value is the same for all triggers (this value is  $T_R$  in Fig. 51),
4. When the countdown timer expires, the hits from this tag are either cleared or read out, depending on the value of the read or clear bit.
- 1185 5. The hits are read or cleared from all core columns in parallel, and then assembled into whole events at the chip bottom, along with the tag. Whole events (tagged) are placed in streams and sent to the Aurora encoder for output.
6. The processed tag is erased from storage and is now available to be used again for another trigger.

1190 There are two trigger modes, known as single level and two-level, selectable by configuration. They are more correctly called auto-read and manual read modes, respectively. They differ only in how the read or clear bit is set and in the countdown timer value, but otherwise follow the above processing identically. In auto-read mode, the value of the read or clear bit is set to read for every incoming trigger. There is no way it can be set to clear. Every trigger will be read out. The 1195 countdown timer delay is small (one bunch crossing). This is expected to be the operating mode used by both ATLAS and CMS.

1200 In manual read mode (two level trigger), the read or clear bit is initialized to clear for every incoming trigger and the countdown timer delay is long (second level latency). At any time before the countdown timer expires, a read\_trigger command can be received for the tag in question, in which case the read or clear bit will be set to read. The read\_trigger command has no action other than setting the read or clear bit. Readout will happen when the countdown timer expires, as usual. Any events for which the bit has not been set to read with a read\_trigger command will be cleared in due course, when the countdown timer expires. Note that latency buffers in the pixel 1205 regions remain occupied longer in two level trigger mode than in single level mode, and therefore the ultimate input hit rate the chip can accept (for a given hit loss) will be lower in two level than single level mode.

1210 Fig. 51 shows the timing from a trigger to the completion of data readout. The time to start of read ( $T_R$ ) is a fixed value (programmable) between 1 and 1024 bunch crossings. For single trigger level mode it will be 1 BX and there will be no read\_trigger command, while for two level mode it will be the level-1 latency (after level-0). In single level mode the start of read happens automatically, while in two level mode a read\_trigger command is needed for each trigger to be read out. This command can be received at any point within  $T_R$ .



**Figure 51:** Timing of trigger to data readout. The Read\_trigger command (dashed) is only used in two-level trigger mode.

Inside the chip the hit data remain in the pixel matrix until their turn to the read out of the chip or cleared. There is no other high capacity hit storage than the pixel matrix itself. There is some transient hit storage in the bottom of chip to buffer and format data on its way to the chip Aurora output, but the number of bits is small compared to the matrix. The hit data flow can be understood as a three stage process. The core column readout, the aggregation of data for each event from multiple core columns, and the Aurora encoding, serialization and output. Each stage pulls data from the previous stage when it needs it. Thus buffers are generally not empty. When buffers are empty idles are inserted into the output stream.

Each core column works as a self-contained unit with a small amount of storage at the bottom of the column. Its job is to pull data from the pixels to try to fill its bottom of column buffer (it pauses when buffer is full), regardless of whatever else is going on in the chip. All core columns do this in parallel and independently. Each core column has its own ordered list of triggered BCIDs to read out. Even if at  $T=0$  all these column lists are identical, that will not be the case for long, as each column will work through its list at its own pace given by what hits it contains. The encoding of hit maps and ToT is done in the column readout. Thus, the bottom of column buffers contain encoded bits, not individual hits.

The event building processes one triggered and selected-for-readout BCID at a time. It pulls the data for that BCID from the column buffers that contain any. Many columns may not contain any hits for that BCID and are skipped. The event building BCID will always be the first one present for columns that have hits, because columns and event building process triggers in the same order. Data for subsequent triggered BCIDs will generally still be in the pixel matrix, as the bottom of column buffers are small. The column numbers for a given event do not have to come out in numerical order- it depends on which column is ready first. A given column address could also appear more than once if it contains many hits. The event tag, and stream markers are added by the event building stage. The built event data are placed in a buffer to make them available to the Aurora encoder. More details are given in Sec. 9.5.

The Aurora encoder pulls data from the event buffer, performs the Aurora formatting, idle insertion, etc. The contents are serialized onto the output lanes. Data merging modifies how this stage works. The following subsections give a more detailed description of the trigger processing and book-keeping. Technical details of the bottom of chip data flow are given in Sec. 9.5.

## 9.1 Bunch crossing ID selection and latency

The pixel matrix operates in steps of the 40 MHz beam crossing clock (BX). Within the matrix each BX is triggered or not based on the state of a trigger signal (high is triggered, low is not). Thus a trigger is a one BX long pulse on this trigger signal. Trigger pulses are normally issued by the command decoder in response to commands received. The trigger command path, from chip input to internal trigger pulses, is shown Fig. 50. Note that the internal BX clock is generated by the channel synchronizer based on the frame alignment of the input control stream (see Sec. 8.3). An individual chip phase adjustment, in 1/(1.28 MHz) steps, is introduced by the clock and data recovery circuit (CDR). Thus, each chip can be individually “timed in” to the bunch crossings.

Each of the 15 trigger commands of Table 6 generates a different pattern of pulses spanning four BX’s. Trigger pulses can also be generated by the internal self trigger source (Sec. 9.4). The command decoder arbitrates the trigger sources, with trigger commands always having priority.

In RD53B each trigger pulse marks data in the matrix as triggered and associates it with a trigger identifier (ID), but *does not* initiate readout. The readout of data marked by a trigger ID is initiated later, as described in Sec. 9.3. Readout can be automatic (every trigger received is read out), which is the single level trigger operation mode, or deliberate with an explicit read trigger command for every previously supplied trigger, which is the two-level trigger mode. The timing was shown in Fig. 51.

The data bits for a given trigger do not instantly come out of the chip upon start of read, but are subject to queuing wait time and processing delay. The total wait time between start of read and bits at the chip output is  $T_W$  and varies depending on activity. It contains fixed delays including the 3 BX token transit to retrieve data from the pixel matrix, another 3 BX for column hit data encoding, the Aurora encoding, etc. The sum of all these fixed delays defines the minimum  $T_W$  is 31 BX, while the distribution is given by the added queuing time, which can be truncated with a programmable time-out. Finally, the readout time ( $T_{R/O}$ ) is given by the number of bits being sent out times the output multi-lane bit rate, which can be up to 5.12 Gbps (4 lanes at 1.28 Gbps each). At a given trigger rate, the average  $T_{R/O}$  must be less than the mean trigger period ( $\lambda$ ), and significantly less to avoid long queuing wait time. Since the whole chip can be analyzed as a single server queue, the wait time plus readout time  $T_W + T_{R/O}$  will have a distribution like Eq. 9.1,

$$P(W > t) = \frac{T_{R/O}}{\lambda} e^{-(\lambda - T_{R/O})t} \quad (9.1)$$

It is clear from Eq. 9.1 that as  $T_{R/O}$  approaches  $\lambda$  the total wait time diverges. The condition  $T_{R/O} = \lambda$  roughly corresponds to 100% data link occupancy.

## 9.2 Tags

The term tag is overloaded with two meanings. In the command protocol *tag* refers to the 6-bit code received with each trigger command (more correctly called tag base). The tag base can only take on 54 values, which is the total number of DC-balanced symbols. Inside the chip and in the output data *tag* refers to an 8-bit extended tag. The two additional bits indicate which of the four BX’s spanned by a trigger command the data correspond to. For example, command Trigger\_04 in Table 6 with tag base value abcdef will result in one extended tag: abcdef01, while Trigger\_05 will result in two extended tags: abcdef01 and abcdef11 (along with two trigger pulses).

While there are only 54 tag bases, which can lead to at most  $4 \times 54 = 216$  extended tags, the extended tag space in the chip spans all 256 8-bit codes. The extra codes that cannot be generated from one of the 54 tag bases are used to label self-triggered events and signal detected error conditions, as indicated in Table 10. In RD53B the main use of special tags is to label self-triggered events. Since self-triggers are generated internally in the chip, they are not constrained to the 54 tag bases, and so they are labeled with extended tags that could never result from a trigger command.

Tag values (decimal)	Meaning
0-215	extended tags from trigger command
216-231	Self triggers
216-219 (*)	Single bit-flip detected in tag symbol of a trig. command
220-223 (*)	unrecognized tag symbol
224-255	spares - not used in RD53B ATLAS

**Table 10:** Possible extended tag values and their meaning. (\*) The bit flip and unrecognized symbol values overlap the self-trigger tags in RD53B-ATLAS. This bug is corrected in future versions.

The number of tag bases available is large compared to the number of triggers expected to be pending at any given time, giving the DAQ flexibility for selecting and managing the tag base value sent with each trigger to make sure an extended tag value that is already in use is never requested. Requesting an extending tag value already in use will result in the chip skipping the trigger (and incrementing the skipped trigger error counter). The simplest approach the DAQ can take is to assign a new tag base value (eg. by incrementing a counter) to each new trigger command, regardless of the command. This is “wasteful” in the sense that two different single trigger commands could share the same tag base without resulting in duplicate extended tags, and statistically most trigger commands will be single trigger commands at 1 MHz trigger rate. However, since the number of available tag bases is large, the DAQ can afford this luxury. This is because the worst case wait time for a trigger to be fully read out is simulated to be about  $25 \mu\text{s}$  [5] and the Poisson probability of a random process with mean 25 (number of triggers in  $25 \mu\text{s}$  at 1 MHz trigger rate) to fluctuate up to 54 or more is negligible. Therefore, the DAQ tag base counter will never come around to the same value while a given trigger is still waiting to be read out. If higher trigger rate is desired for some applications, more complex tag base selection schemes can be used.

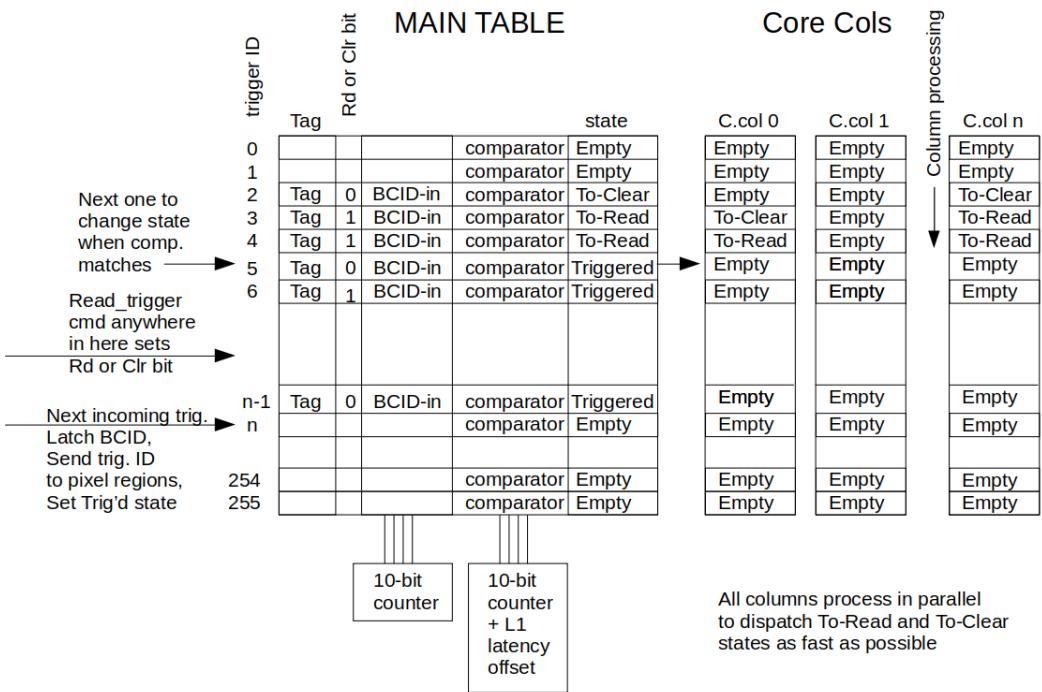
### 9.3 Trigger Book Keeping

RD53B keeps track of pending and in progress triggers using a main table holding all triggers plus one dedicated table in each core column with those triggers staged for readout or clearing, as shown in Fig. 52. The main table has 256 rows and so can hold every possible extended tag at once. The row number is used as a trigger ID to label triggered hits in the pixel matrix. The trigger ID is internally Gray coded so that only one bit ever changes from one ID to the next in the pixel matrix bus distribution bus. Each row contains:

- The extended tag received with the trigger command, which is not used in RD53B but simply stored so that it can be returned with the event data.
- The read-or-clear bit described in Sec. 9.

1315

- A time-keeping mechanism consisting of a latch to store the Bunch Crossing ID (BCID) timestamp when the trigger arrived and a comparator that compares this timestamp to an offset BCID counter, where the offset is the second programmed level trigger latency (this is the countdown timer of Sec. 9).
  - A state for this row. There are four possible states in the Main table: Empty, Triggered, To-Read, and To-Clear. The read-or-clear bit determines which state follows the triggered state.
- 1320 If 0, then To-Clear follows Triggered, while if 1 then To-Read follows Triggered.



**Figure 52:** Conceptual diagram of trigger tables in RD53B. The row number is used as the trigger ID in the pixel matrix. The tables are circular buffers filled and emptied as explained in the text.

1325

1330

The column tables also have 256 rows, but only contain the Empty, To-Read or To-Clear states in each row. They do not have a Triggered state as it is not needed in the columns. They can be regarded as extra columns added to the main table. When a trigger arrives, it is assigned to the next Empty row in the main table. Nothing happens in the column tables at this point. The trigger ID (row number) is sent to the pixel matrix to label corresponding hits (see Sec. 7.4 for how the hits are time-stamped and selected in the pixel matrix). The tag is saved, the BCID value is latched, and row state is changed to Triggered in the main table. As more triggers arrive others rows will change state to Triggered, but they will not affect this row. This row will remain in the Triggered state until the time-keeping mechanism asserts that the second level trigger latency has elapsed—that is to say, until the row BCID comparator fires. Note that only one row comparator can fire in any given bunch crossing clock cycle, because at most one row can latch the BCID value in any clock cycle (at most one trigger can arrive per clock cycle). Note also that the BCID comparators will fire in the same order that the triggers arrived, because they all fire the same number of clocks after arrival.

1335 When the BCID comparator fires the state of the row is changed to either To-Clear or To-Read, depending on the value of the read-or-clear bit as stated in Sec. 9. (The setting of the read-or-clear bit was also explained in Sec. 9, together with the fact that the only difference between single trigger and two trigger modes is how this bit is set.) The new row state is now propagated to all the column tables. The given row state changes from Empty to To-Read or To-Clear in call  
1340 column tables at the same time. While all the column tables are filled in parallel this way, they are emptied independently. Each core column has its own state machine and processing, whose job is to change rows back to Empty as fast as possible, regardless of what other columns are doing. Starting from the top down, the first non Empty row will be processed, by transitioning first from To-Read to To-Clear (if needed) and then from To-Clear to Empty. The state is changed as soon  
1345 as the readout or clear operations are performed on the all the hits labeled with the corresponding trigger ID (row number). Because they dispatch their non-Empty states independently from one another, each column can have a different number of rows in a given state, as illustrated in Fig. 52. Independent processing of all columns in parallel is necessary, because each core column takes at least 3 bunch crossing clock cycles to read between 1 and 8 hits (depending on hit pattern), whereas  
1350 the full chip output bandwidth can be as high as 15 hits per single clock cycle (5.12 Gbps and 10 bits per hit case).

In the main table, the state of each row with a to-Read state will change back to Empty when the event is fed to Aurora encoder. The stored tag value will be retrieved and returned in hit data stream (Sec. 10.3). Only after this point can the same tag be reused by the DAQ. Exceptions can  
1355 happen, for example if a trigger arrives with a tag value that is already stored in the main table, or a read\_trigger arrives with a tag value that does not match any value stored in the main table. Special tag values are reserved to mark such exceptions as covered in Sec. 9.2. It could also happen that row number is not arriving at the Aurora encoder for a very long time, preventing the row state from returning to Empty and also preventing processing of subsequent rows. To protect against this, a  
1360 time-out has been implemented that keeps track of how long a row has been in the To-Read state (reusing the same BCID comparator that timed the duration of the Triggered state). The timeout value is programmable (register TruncationTimeoutConf in Table 23). When the timeout is reached before the row state becomes Empty, the state is changed to To-Clear in all columns that have not yet processed the event and the state in the main table is changed to Empty.

1365 Each column operates as a single server queue, with a 40 MHz output rate. Each To-Read state requires 3 bunch crossings to be processed regardless of hit content (even if there are no hits in the core column for this event). Each hit quarter-core requires another 3 bunch crossings to be transferred out of the column. Note that the higher the rate of read\_triggers, the greater the bandwidth utilized by the To-Read processing overhead, leaving less bandwidth for hit transfer.  
1370 Thus, for a 1 MHz (4 MHz) read\_trigger rate, the input rate overhead to the column queue before any hits is 3 MHz (12 MHz), leaving room for 5.7 (2.7) hit quarter cores per event on average before the column readout bandwidth reaches 50% occupancy, beyond which point the simple queue wait time grows rapidly (Eq. 9.1). This corresponds to quarter core occupancies of 3% (1.5%), or, assuming an average of 3 hits per hit quarter core, 0.6% (0.3%) pixel occupancy. The higher the read\_trigger rate, lower the hit occupancy that can be accommodated before the column readout  
1375 saturates.

## 9.4 Self Trigger Source

The self trigger functionality is a stand alone block that, if enabled, can store triggers to be processed in the trigger table. The self trigger can operate in parallel to the normal single level trigger operation from the Command Decoder, but command decoder always has priority over the self trigger. The self trigger can not operate in two level trigger mode.

The block diagram of the self trigger processing pipeline is shown in Fig. 53. The mapping of the registers found in the drawing to global configuration can be found in Table 11. In a core column there are 4 HitOr lanes to which ORs the discriminator output of the pixels (Sec. 13.4).

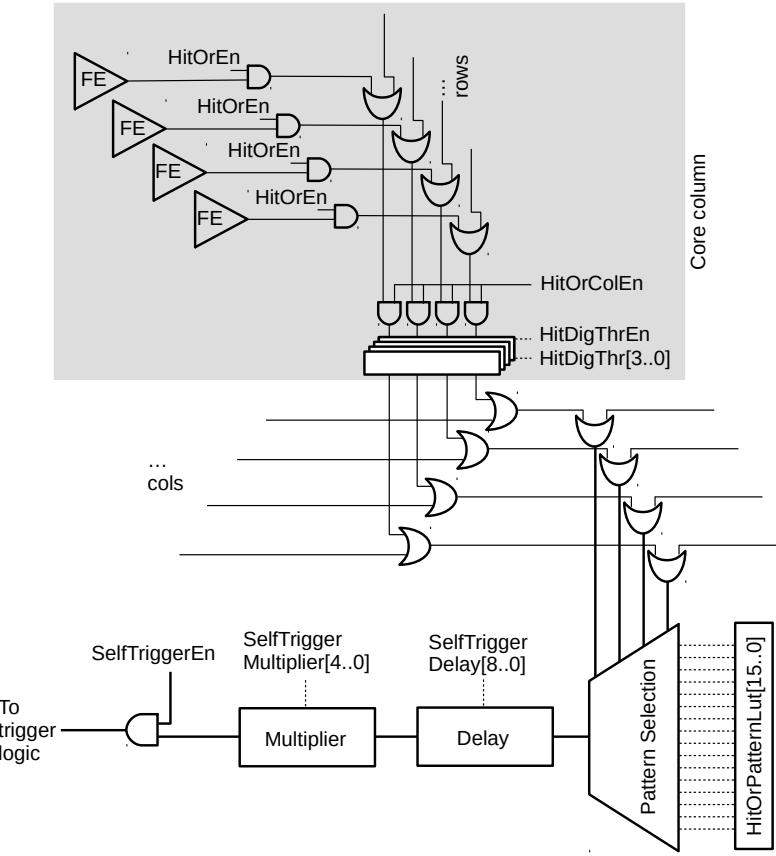
There is a configuration bit (in the pixel register) for each pixel to activate the pixel for the HitOr. At the end of the core column each HitOr lane can be enabled or disabled via the global register HITOR\_MASK\_1/2/3/4.

Register Name	Bits	Field Name	Description
SelfTriggerConfig_1	[3 : 0]	HitOrDigThr	If digital threshold enabled this is the length in clock cycles the HitOr has to be active. Values 0 and 15 are allowed, but will render the self trigger unusable.
	[4]	HitOrDigThrEn	Enables digital threshold, if disabled analog (not synchronized) signal
	[5]	SelfTriggerEn	Enables (gates) output of self trigger block
SelfTriggerConfig_0	[4 : 0]	SelfTriggerMultiplier	A single trigger pulse can be elongated to cover up to 31 bunch crossings. Value 0 is valid but will render the self trigger unusable. Note that only 16 eTags are available for the self trigger, so trigger multiplication beyond 16 might result in rejected trigger.
	[14 : 5]	SelfTriggerDelay	Delay applied to the HitOr pulse, has to match the configured Latency such that the resulting pulse triggers the right bunch crossing. The Self trigger pipeline has an internal delay of around 12BC (depends on digital threshold).
HitOrPatternLUT	[15 : 0]		Each bit represents a unique combination of the four HitOr, this enables to only trigger of coincidence of multiple (specific) HitOrs. Note that the LSB should never be high, as it represents all HitOrs being low. 0xFFFF represents an Or of all possible combinations.

**Table 11:** Selection of inputs to global OR operation feeding the self trigger generation.

There is one digital threshold block for each lane for each core column. The digital threshold can be disabled via HitOrDigThrEn which will also bypass any synchronization of the HitOr signal to the 40 MHz clock. Enabling the digital threshold will synchronize the HitOr signal to the 40MHz bunch crossing clock and the required threshold length can be set from 1 clock cycle up to 14 clock cycles. If a HitOr signal passes the threshold it will produce a single clock cycle pulse.

After each lane of the all core columns are ORed and fed into a large lookup table. Each entry in the table describes a unique state of all possible HitOr lane combinations. As the HitOrs are laid out in such a way that a coincidence on specific lanes corresponds to a multi pixel cluster hit in a specific direction (depends on sensor geometry).



**Figure 53:** Block diagram of the self trigger pipeline from pixel discriminator output (HitOr) to connection to the trigger logic table.

The Pattern LUT generates a single pulse, which can be delayed up to 511 clock cycles to match it with the L0 latency. Multiple trigger pulses can enter this delay shift register. The single pulse from the pattern LUT can be elongated (multiplied) to up to 31 bunch crossings , but only 16 extended Tags are available for self trigger in the trigger table. I.e. one can likely not use a multiplication beyond 16. The self trigger tags are full 8-bit tags picked sequentially in the range given in Table 10, not constructed from a tag base plus two bits. The self trigger tag counter is reset by the DataPathReset signal of the Global Pulse.

## 9.5 Data Flow

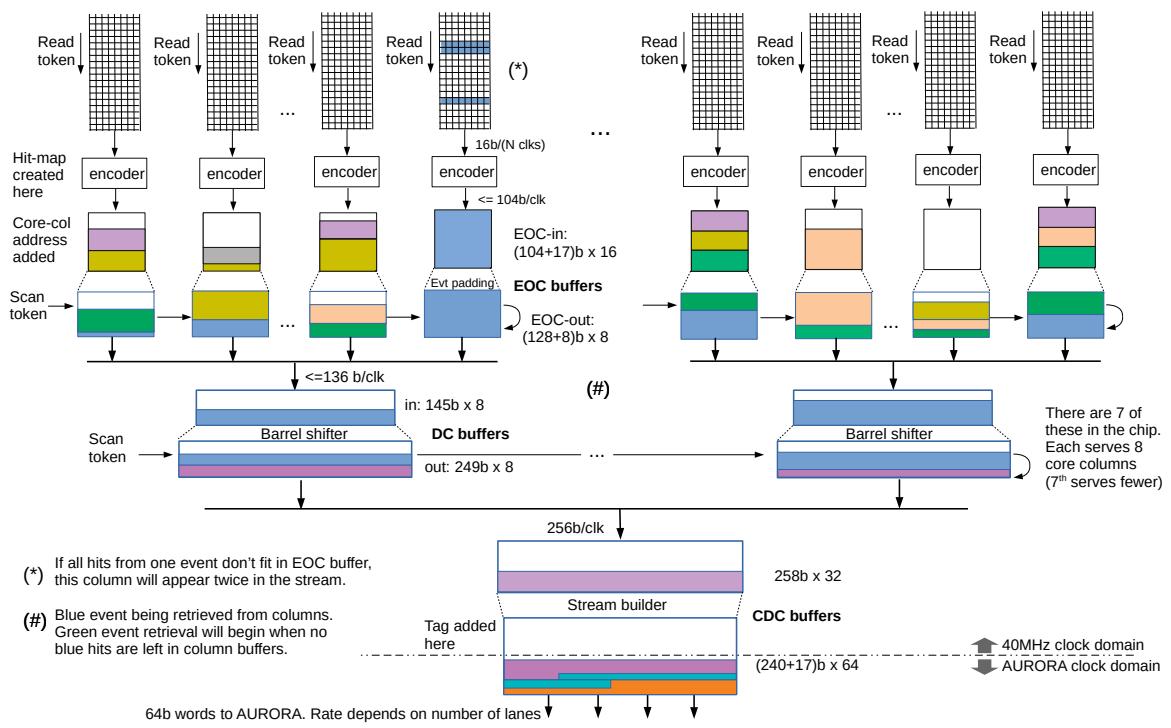
The starting point of the data flow is hit data stored in pixel regions waiting to be read out. Such stored hit data are labeled with a trigger ID. That labeling was carried out when the trigger arrived and all regions with hits from the BCID corresponding to that trigger were flagged (see Sec. 7).

Fig. 54 shows how data flows out of the regions and though the chip. Each core column has its own list of pending triggers (as shown in Fig. 52) and processes that list as fast as possible, independent of all other columns. The processing pauses whenever the pending list is empty or the End Of Column buffer (EOC) is full. The hit data are retrieved one 4-pixel region at-a-time using a token that finds those region buffers matching the requested trigger. By default it takes 3 BX clocks

to retrieve and encode the data from one region, but this can be increased by configuration as may be needed after logic slows down due to radiation damage. The data from 4 regions (16 pixels total)  
1415 are accumulated and passed through a pipelined encoder that generates (and optionally binary tree encodes) the 16 bit hit map and discards empty ToT values. It also adds the row address, neighbor, and last hit flags. These data are placed in one row of an input EOC buffer that is wide enough to accept the maximum possible number of bits from one encoded 16-pixel region. Each row of this  
1420 buffer will necessarily contain many empty bits (the max number of bits occurs very rarely). These data are then barrel shifted and packed to remove the empty bits into the output EOC buffer. The aggregation of data from different core columns can now begin.

Every 8 core columns are combined into one Data Concentrator (DC). There are 7 Data Concentrators, but the last one contains fewer than 8 core columns because the number of core columns is not a multiple of 8. Each DC runs in parallel, independently of the others. It aggregates data  
1425 as fast as possible, pausing when the input buffers are empty or the output buffers are full. It processes one trigger at a time. As in the EOC, there are input and output FIFOs with a barrel shifter in between, in order to combine data from multiple columns without empty space. Data for a given event are pulled from one core column at a time until no more data from that event is present. Occasionally it may happen that a column has so many hits from a given event that they do not all  
1430 fit into the EOC FIFOs. In this case the DC processing will visit that column more than once, and data from that EOC will appear in two different places in the same event, correctly labeled with its column number (see Sec. 10.4). Note that there is also a programmable truncation limit on the column readout (see Sec. 9.3).

In the final stage, data from the eight Data Concentrators, one at time for a given event, are pulled into the Chip Data Concentrator (CDC). This is where stream building takes place, as this  
1435 is the first time that all hits from a given event are collected in the same FIFO. The tag and any other event-level information (Table 17) are added here. The total storage in the chip bottom (everything shown in Fig. 54) is 204 Kbits. As the number of bits used per hit pixel in detector readout simulations ranges between 9 and 15 (depending on occupancy and cluster distributions),  
1440 that means the bottom of chip memory can hold between 14K and 20K hits. In contrast, the pixel matrix has storage for 8 hits per pixel, or 1.2M hits, so almost 100 times more (while the silicon area of the pixel matrix is only 10 times greater than the periphery).



**Figure 54:** Schematic diagram of the data flow from the core columns to the Aurora output.

## 10. Data Output

The RD53B data output consists of tagged events, such that the readout will automatically recover from transmission errors without any action from the DAQ. While tagged data would permit event building to be performed off chip if desired, RD53B builds events on-chip, such that a full event is output before sending any data for the next event. The characteristics of the physical data output ports are described in Sec. 10.1. The transmission protocol used is a subset of the Aurora 64b66b protocol [3], as detailed in Sec. 10.2. This provides industry standard frame alignment, DC balance and multi-lane serial transmission suitable for high speed data, but does not define the data content.

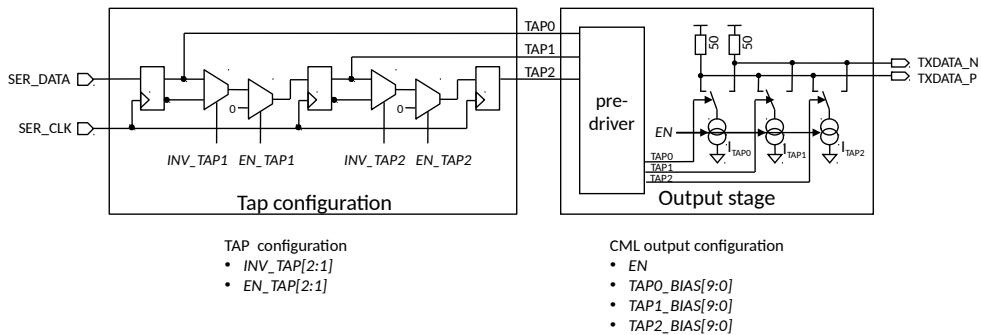
The Aurora protocol can be thought of as a “wrapper” placed around the RD53B data. Before the Aurora wrapper, the hit data are packaged in *streams*, not fixed frames. A stream is a self-contained, variable length data container beginning with a tag (8 bits) and followed by a mix of hit data and possibly other tags (called internal tags, which are 11 bits). Streams and their contents are described in Sec. 10.3 to 10.9. This variable length format is approximately 25% more efficient (fewer bits per hit) than the fixed frame format previously used in RD53A.

There are two encoding modes: single chip and multi-chip. Multi-chip encoding must be used when performing data aggregation. The encoding description in Sec. 10.4 is given for single chip mode, and the effect of multi-chip mode is described in Sec. 10.7. The use of multi-chip mode for data aggregation is described in Sec. 11.

The output is highly configurable and must be correctly set up to perform as required. The basic configuration for single chip operation was described in Sec. 3. Control of event size and data filtering options are covered in Sec. 10.8. Use of pre-emphasis for operation with lossy cables is included in Sec. 10.1. Use of test modes, for example for bit error rate studies, is covered in Sec. 13. Technical details of clock and data recovery and serialization are given in Sec. 14.

### 10.1 Data Output Drivers

RD53B contains four current mode logic (CML) differential output drivers (Fig. 55) with programmable pre-emphasis. Between 1 and 4 of these drivers will actually be used depending on the Aurora configuration (Sec. 10.2). Each driver is fed by a dedicated serializer circuit that produces the high speed bitstream. The default bitrate is 1.28 Gbps, but it can be reduced in factors of two down to 160 Mbps (and will be the same for all drivers). Serializer details are given in Sec. 14.7.



**Figure 55:** Detailed CML driver functional block diagram including TAP circuit

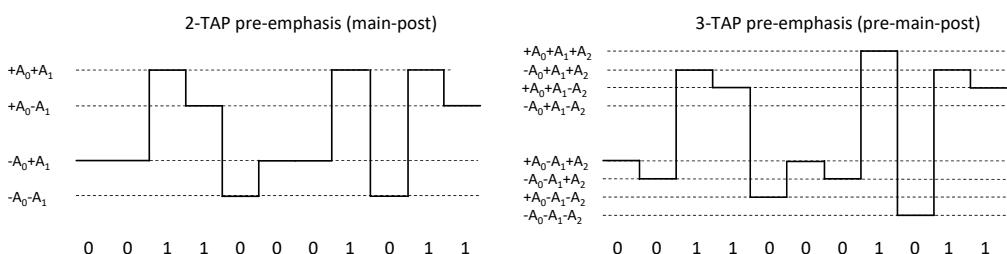
These differential drivers use back-termination with a  $50\Omega$  pull-up resistor to VDD\_CML on each wire to minimize back-reflections and thus improve the signal integrity with non-ideal transmission lines. Each driver can provide pre-emphasis via three current mode switches in parallel (so-called TAPs), which can be programmed to compensate for the high frequency damping of lossy transmission lines using the En. TAP 2,1 and Inv. TAP 2,1 fields of configuration register CML\_CONFIG (Table 23). The maximum current for each TAP is approximately 14 mA and the LSB is approximately  $14\mu A$ . The TAP configuration (Fig. 55, left) controls the type of pre-emphasis to be used:

- Single TAP: no pre-emphasis
- 2-TAP: programmable overshoot during transitions
- 3-TAP pre-main-post: programmable under-shoot followed by a programmable overshoot during transitions
- 3-TAP main-post1-post2: two levels of overshoot after the transition

The duration of the over/undershoot pulse is fixed by the SER\_CLK period, while the amplitude of the output levels can be programmed via the three configuration registers: DAC\_CML\_BIAS\_0 to DAC\_CML\_BIAS\_2. Table 12 shows the configuration settings for the pre-emphasis modes with recommended bias settings. Note that the inversion of TAP 1 is propagated to TAP 2 (Fig. 55).

Pre-emphasis	Inv. TAP[2:1]	En. TAP[2:1]	TAP 0 Bias	TAP 1 Bias	TAP 2 Bias
off	xx	0	700	0	0
2-TAP (main-post)	x1	1	500	200	0
3-TAP (main-post1-post2)	1	11	500	100	100
3-TAP (pre-main-post)	11	11	100	400	200

**Table 12:** Configuration settings for the pre-emphasis modes with recommended bias settings. Bias values are register settings (decimal).

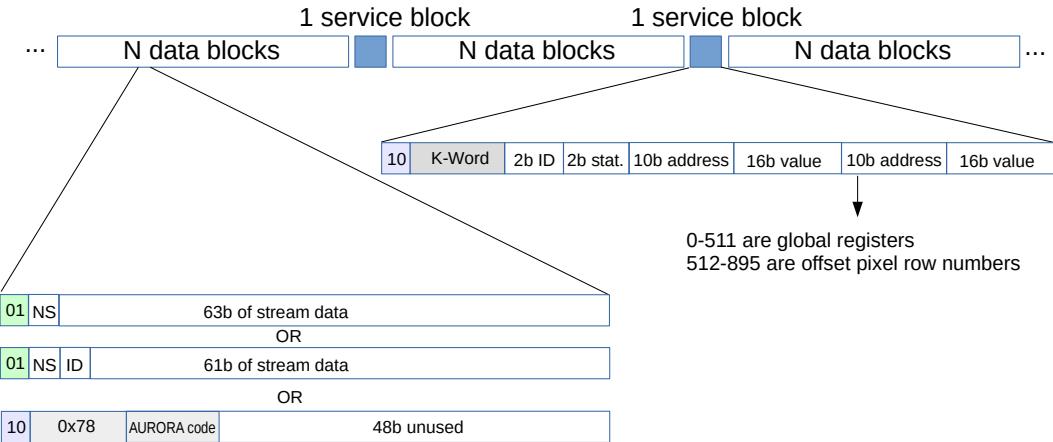


**Figure 56:** Output waveform with active pre-emphasis in 2-TAP (left) and 3-TAP pre-main-post mode (right). The amplitudes  $A_0$ ,  $A_1$ , and  $A_2$  are controlled by the bias settings of Table 12.

The effect of the pre-emphasis is shown in Fig. 56. In 2-TAP mode a programmable overshoot is added to every transition to compensate for the high frequency attenuation of the transmission line. The pre-main-post 3-TAP mode adds an additional undershoot in front of each transition which would compensate a higher order low-pass filter transfer function. Test have shown that 2-TAP pre-emphasis mode gives the best results with the ATLAS prototype cables (6 m long custom twinax cables). Using the bias settings given in the table the 2-TAP pre-emphasis achieves a boost

1495 of 10 dB at 640 MHz (1.28 Gbps). Note that the maximum output amplitude is not limited by the pre-emphasis mode but limited to 700 mV full swing by the saturation voltage of the NMOS current sinks. Thus, the same maximum amplitude is reached with the bias settings shown with pre-emphasis off, as with 2-TAP mode.

## 10.2 Aurora and RD53B Data



**Figure 57:** Schematic diagram of output data highest level format, consisting of  $N$  data or idle blocks followed by one RD53 service block. Each block consists of an Aurora 2-bit header that can be 01 or 10, plus 64 scrambled bits. The diagram shows the content of the 64 bits before scrambling. The gray shaded 8-bit fields with values given in hex have a meaning defined in the Aurora protocol. The possible Aurora K-Block values are given in Table 13. NS stands for New Stream bit and ID for the two least significant bits of the chip ID.

Aurora K-Word code (hex)	Meaning
0xB4	both register fields are of type AutoRead
0x55	first frame is AutoRead, second is from a read register command
0x99	first is from a read register command, second frame is AutoRead
0xD2	both register fields are from read register commands
0xCC	Indicates an error. Fields are meaningless

**Table 13:** Meaning of Aurora K-Word code (zz) in the periodic service blocks. This table is a companion to Fig. 57

1500 At the highest level, the RD53B output is encoded with a subset of the Aurora 64b66b protocol [3] (see App. A). RD53B implements a simplex channel configuration over 1 to 4 lanes using the *Strict Alignment* feature of the protocol. Each lane runs nominally 1.28 Gbps, but can be divided down by 2, 4, or 8. Pixel hit data are sent in one single infinite length Aurora *Data Frame* (binary

01 header), while service data (such as configuration register readback) are sent using Aurora *User K-Blocks* (binary 10 header).

GTx Out	DataMergingMux bits	Value (dec)	Internal lane selected
GTx0	[1:0]	0	0
GTx1	[3:2]	1	1
GTx2	[5:4]	2	2
GTx3	[7:6]	3	3

**Table 14:** Output switch matrix configuration using register DataMergingMux of Table 23. The right side table indicates which GTx output is connected to the internal Aurora lane based on the programmed 2-bit value.

The ability to configure 1 to 4 output lanes can accomodate different wiring configurations in actual chip usage. It is possible to route any of these four lanes to any of the available output GTx channels of Sec. 10.1. This routing is done configuring the output switch matrix according to table 1510 14. The default value is to route lane 0 to GTx0, lane 1 to GTx1, lane 2 to GTx2 and lane 3 to GTx3.

Aurora blocks consist of 66-bits (Fig. 57). Each block has a 2-bit sync header that can be 01 (Aurora Data type) or 10 (Aurora K-Block or K-Words), followed by 64 scrambled bits. Because the header is not scrambled, it permits frame alignment of the received data. Frame alignment identifies where each 66 bit block starts.

1515 RD53B takes advantage of the differentiation provided by Aurora between Data and K-Blocks to implement two independent output “channels”, hit data and service data, as depicted in Fig. 57. These two channels are effectively time-multiplexed onto the serial output. The Aurora encoded output basic unit (which repeats forever) consists of  $N_D$  Data or Idle blocks plus one service User K-Block, where  $N_D$  is programmable (range 1-256) and has default value 50 (ServiceDataConf 1520 register in Table 23). A fraction  $1/(N_D+1)$  of the output bandwidth is thus permanently reserved for service information and unavailable for hit data. Conversely,  $N_D/(N_D+1)$  is reserved for hit data and cannot be used for service information. If there are no hit data this fraction of the bandwidth will have Aurora idle blocks.

1525 Service blocks will not be sent except in their allocated turn every  $N$  data or idle blocks. The interval  $N$  is used on every lane regardless of how many lanes are active. For example, with the default  $N_D=50$ , 2% of the output bandwidth is permanently unavailable for hit data (in addition to the 3% consumed by the 2-bit 64b/66b header). At  $4 \times 1.28$  Gbps output bandwidth this 2% is sufficient for the maximum possible register readback of 64 Mbps, since 2% of 5 Gbps = 100 Mbps (See Sec. 8.8). In the service blocks, an 8-bit code follows the sync header, as specified by the 1530 64b/66b protocol, leaving 56 bits available for user information. These 56 bits are allocated as a 2-bit chip ID plus two 26 bit registers (10-bit extended address plus 16-bit value = 26 bits) plus 2 status bits, specified in Table 15.

ID[2 bits] 2x( [e-address (10 bits)] [value (16 bits)] ) [status (2 bits)]

Because of the chip ID, the service block is always compatible with multi-chip mode. The 10-bit extended addresses (e-address) in the service block are: MSB=0, followed by the 9-bit global

1535 register address, or MSB=1, followed by the 9-bit pixel row address in case of reading global register 0 (the pixel configuration portal register). The separation of the output into two time-multiplexed channels guarantees a certain bandwidth for both data and register information without the need for a complex priority arbitration containing safeguards against all possible pathologies.

1540 The periodic service block coming out every  $N_D$  data frames is filled automatically, even without there having been a read register command. The possible Aurora K-Words in Fig. 57 are given in Table 13. The two 16-bit registers are denoted  $A_i$  and  $B_i$ , where  $i$  is the lane number (0 to 3). The automatic filling of the  $A_i$  and  $B_i$  registers is controlled by eight configuration registers Auto- $A_i$  and Auto- $B_i$ , which have default values, but which the user is free to change. The auto-fill register addresses are specific to each lane. Thus if only lane 0 is used then only Auto- $A_0$  and Auto- $B_0$  are functional. RdReg commands will queue the registers specified by the command for output on lane 0 only, with priority over auto fill. Lanes 1 to 3 are unaffected by the RdReg command and only output their assigned auto-fill registers. If only one RdReg command has been received, then the  $A_0$  register will be auto-filled while the  $B_0$  register will contain the requested register. If more than one was received then both registers will be requested registers and auto-fill will wait. If read 1545 register commands are sent too fast for the reserved output bandwidth, the FIFO holding pending read registers may fill up, and any read register commands received while the FIFO is full will be ignored. All service block FIFOs have a depth of 16. The readout of registers staged in the FIFO may also happen out of order.

Status Code (decimal)	Meaning
0	Ready
1	There has been an error since the last register frame
2	There has been a warning since the last register frame
3	Both 1 and 2

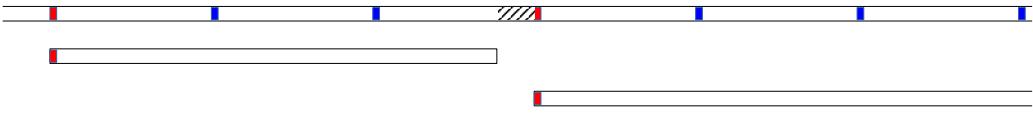
**Table 15:** Meaning of 2-bit status code

### 10.3 Aurora and streams

1555 The Aurora protocol transmits data in 64-bit blocks. These blocks are scrambled and each one is preceded by a 2-bit Aurora header that permits frame alignment. In the RD53B encoding, each 64 bit data block, before scrambling, begins with a New Stream bit (NS). If NS=1, this indicates the start of a new stream, which can be decoded without any information about the data that came before. If NS=0, this indicates that the previous stream simply continues, and therefore the data 1560 following NS=0 cannot be interpreted without having the prior part of the stream. .

Fig. 58 shows a continuous bit stream as would be seen after Aurora decoding. The position of the NSbits in this bitstream are known (red and blue), thanks to Aurora having taken care of frame alignment. The figure also shows two RD53B streams, which are self-contained, variable length data packets. The start of each stream in the continuous bitstream is flagged by NS=1 (red), while 1565 NS=0 (blue) are simply ignored.

Streams only contain hit and exception data. Configuration readback and monitoring data are not included in streams, but are sent in the periodically inserted Aurora service blocks.



**Figure 58:** Continuous bitstream after Aurora decoding (top) showing the NSbit positions (which correspond to Aurora block boundaries). Two streams are shown below. NS=1 is shown in red and NS=0 in blue. Orphan bits are shown as hatched.

A Stream contains  $N_E$  events, where  $N_E$  is programmable from 1 to 64. For short or empty events (as will occur in outer layers), single event streams will be inefficient, because the so-called 1570 orphan bits at the end of a stream (hatched in Fig. 58) are wasted. For long events (as in the inner layer) single event streams only waste a few percent of bandwidth on orphan bits. The default setting is  $N_E = 16$  (in register DataConcentratorConf). Note that single event streams are obtained by programming  $N_E = 0$ . Even when  $N_E$  is programmed  $>0$  single event streams will occur if the trigger rate is low as a stream must end when there is no more data to be sent.

1575 A new stream always begins with a tag (8 bits) and is followed by a mix hit data and if  $N_E > 0$ , other tags (called internal tags, which are 11 bits). A tag is always output for every trigger read out even if the event is empty. (In single level trigger mode every trigger results in an output tag, while in two level trigger mode every read\_trigger results in an output tag.) The possible tag values are given in Table 10. The hit data are compressed and zero-suppressed, and therefore, variable length 1580 (number of bits per hit varies).

The end of a stream is normally known by the start of a new stream (NS=1). However, there can be cases where the next NS=1 bit takes a long time because there have been no triggers. Therefore, an end of stream marker is available, consisting of 000000. This is neither a valid ccol 1585 address nor a valid internal tag (see Sec. 10.4), and is interpreted as an end of stream marker. Its use can be optionally disabled by zeroing bit “EoS marker” of register DataConcentratorConf (Table 23). Note that if a stream happens to end fewer than six bits from the 64-bit boundary when end of stream marker is enabled, an entire 64 bit block of all 0 will be added in order to complete the end of stream marker.

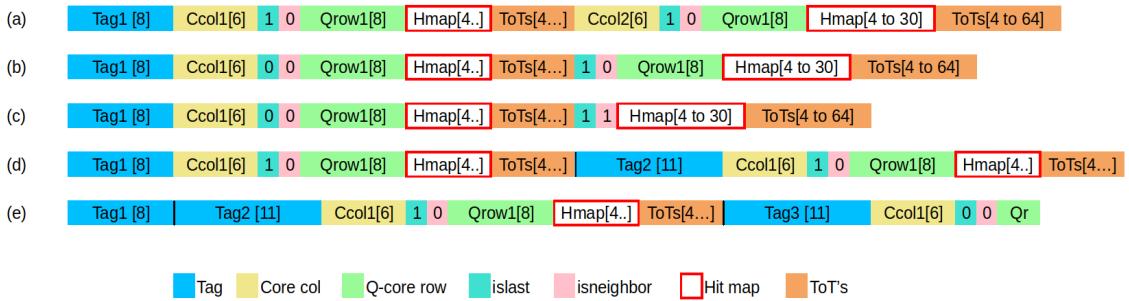
#### 10.4 Hit data encoding

1590 Within a stream, hit data encoding uses a hierarchical address of core-column (ccol), quarter-core row (qrow) within that column, and 2 pixel x 8 pixel quarter-core hit map, compressed or not as explained later (hit map compression can be disabled in CoreColEncoderConf configuration register). Following the quarter-core hit map are the ToT values for all hit pixels in the quarter-core 1595 (which can be suppressed by setting bit “Drop ToT” of CoreColEncoderConf register in Table 23).

Suppressing ToT will reduce data volume by about 30% at small radius). The order of the ToT values is top row first, from left to right, and bottom row second, from left to right (note that the row number increases from top down). The qrow address field begins with two flag bits called *islast* and *isneighbor*. The *islast* bit is set if this is the last qrow address in the ccol and zero otherwise, while the *isneighbor* bit is set if the previous address was qrow-1 and zero otherwise. When *isneighbor*

1600 is set, the qrow address is omitted, as it is known to be the previous address+1. (This is a form of Huffman coding: since the most frequently occurring qrow address is the previous address+1, due to the clustered nature of hits, a single bit is used to encode this address, while for all other cases a 0 followed by the full qrow address is used.)

1605 Fig. 59 shows the bit content of various hypothetical short streams, without showing Aurora block boundaries. Each of these streams could span one or more Aurora blocks and the NSbits are not shown. These examples illustrate the encoding hierarchy, where the different fields appear depending on the data content, and the functioning of the *islast* and *isneighbor* bits. Placing all ToT's in one block after the quarter-core map makes it simpler to drop ToT, should that be needed, but the default encoding contains the 4-bit ToT values. Table 37 provides a stream decoding reference.



Tag    Core col    Q-core row    islast    isneighbor    Hit map    ToT's

**Figure 59:** Examples of encoded stream data with no Aurora block boundaries shown and corresponding NSbits suppressed: (a) one hit quarter-core each in two ccols (note last hit bit is set for both), (b) two separated quarter-cores hit in same ccol (last hit set only for second), (c) two neighbor quarter-cores hit in same ccol, (d) one hit quarter-core each in two different events, (e) an empty event followed by an event with one hit quarter-core, followed by another event. A color key to the field types is shown at the bottom. The number of bits in each field is shown in square brackets.

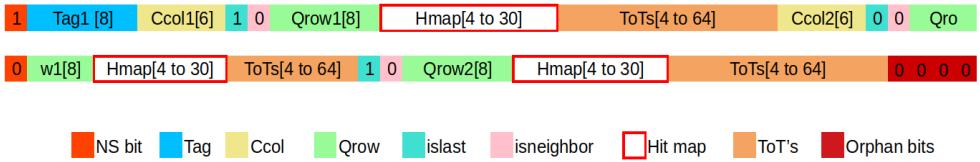
1610 The ccol address is not compressed. The allowed range is 1-55. The value 0 is reserved for the end of stream marker mentioned earlier. Since all valid ccol values are < 56 (binary 111000), an address 111xxx is interpreted as the first bits of an internal tag instead of a ccol. The full internal tag is thus 111xxx xxxxx (see Fig.59d,e). The qrow address begins with the two flag bits *islast* and *isneighbor* as explained before. There is only compression in the case of *isneighbor*=1, which is significant, as this condition is common for clustered hits.

1615 Typically all the data for one ccol will appear together, followed by all the data for another ccol, and so on. But this is not a rule of the encoding. Occasionally, depending on the number of hits and the timing of their extraction from a core column, it may happen that only some of the data for  $ccol_i$  appears and is followed by  $ccol_j$ , after which more data for  $ccol_i$  appears. This is perfectly valid and has important implication for the DAQ. The DAQ must store separately for every ccol the latest qrow value processed (latest qrow must be an array indexed by ccol, not a single variable). This way the DAQ will know where to continue when a ccol value appears more than once. Normally *isneighbor* will be zero for the first qrow address of a ccol, but when that ccol appears a second time in an event readout, it is possible for *isneighbor* of the first qrow to be 1, since the qrow addresses are simply continuing from the first installment of that ccol's readout.

## 10.5 Stream construction and efficiency

The stream builder (Fig. 54) must decide when to end a stream and start a new one. The builder does not know in advance when a stream will end; the data will determine that. A stream will end when (1)  $N_E$  events have been added, or (2) there is no more data to be sent.

For both of the conditions that end a stream, there will typically be a remainder of *orphan* bits between the end of the stream and end of the last 64 bit frame. These bits could in principle be used for something, but in RD53B they are padded with zeros. The DAQ should ignore orphan bits. For easy identification of orphan fragments the core column addresses start at 1 instead of 0. Thus, 000000 effectively marks the end of a stream, whether the end of stream marker is enabled or not. If the end of stream marker is disabled the number of orphan bits can be fewer than 6, even none, while if end of stream is enabled there will always be 6 or more orphan bits. For example if a stream would have 4 orphan bits (0000) with end of stream marker disabled, a new block would be added, increasing the number of orphan + end of stream marker bits to 67. Fig. 60 shows the bit content of a hypothetical stream extending across two Aurora blocks.



**Figure 60:** Encoded output for one hit quarter-core in one core column, and two adjacent hit quarter-cores in another core column, spanning two Aurora blocks. The new stream bit (red) is set for the first Aurora block (top) and zero for the second, indicating that the stream continues in the second Aurora block. Orphan bits set to zero (dark red) at the end of the stream in the second Aurora block.

The fraction of bandwidth wasted on orphan bits (inefficiency) can be easily estimated from the stream length. Taking the stream length as an approximately random variable, the average number of orphan (+ end of stream) bits per stream is 31 (37) if end of stream marker is off (on). This, in order to achieve a small fraction of wasted bandwidth, for example <2%, the average stream length must be  $>1550$  ( $>1850$ ) bits. So one should program  $N_E = 1550/\bar{W}_E$ , where  $W_E$  is the number of bits per event.

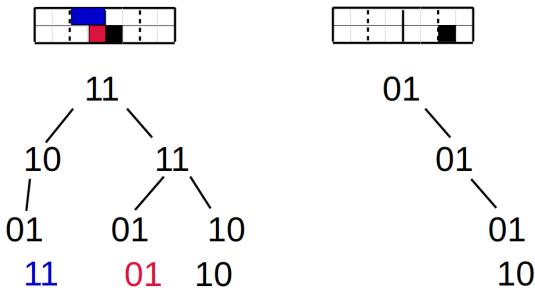
## 10.6 Hit map construction

A 16-bit hit map of the quarter core is constructed showing which of the 16 ToT values are not 1111. The ToT 1111 means “no hit”. If the pixel ToT value was 1111 then the corresponding bit in the hit map is zero and otherwise it is one. The default action is to compress the quarter-core 16-bit hit map to use fewer than 16 bits on average. This compression can be turned off in CoreColEncoderConf configuration register (Sec. 15.2). if compression is off, then the hit map will always be exactly 16 bits. In order to compress the hit map, it is (A) encoded using a binary tree and (B) the resulting code is then reduced with a bit code substitution. This section explains the

encoding in an algorithmic way that is easy to understand, but does not reflect how it implemented  
1665 in the chip.

**(A) Binary tree construction** This is done recursively in 3 steps (for the 16 pixel quarter-core) as follows

1. divide the quarter-core in top and bottom rows and label each row with 1 if it contains any hits and 0 if it does not. The top row is the first bit and bottom row is the second bit.
- 1660 2. Divide each row of 8 pixels into a left half (first bit) and right half (second bit). The bit is 1 if any of the 4 pixels are hit and 0 if not.
3. Divide each half-row of 4 pixels into a left pixel pair (first bit) and right pixel pair (second bit). The bit is 1 if the pixel pair has a hit and 0 if not.



**Figure 61:** Depiction of binary trees for two example quarter-core maps. The bottom tier of the trees consists of 2-pixel hit maps.

After these 3 steps one has identified all pixel pairs with at least one hit. The 2-bit map for  
1665 each hit pixel pair is saved (this a 4<sup>th</sup> step in the chip implementation). The results of the encoding are: 2 bits for step 1, from 2 to 4 bits for step 2, from 2 to 8 bits for step 3, plus the 2-bit maps of all the hit pairs. A quarter-core hit map with a single hit will have an 8-bit binary tree representation. A quarter-core with exactly 2 hits will have a binary tree with between 8 and 14 bits, etc. The maximum number of bits a compressed hitmap with up to 16 hit pixels can use is 30 ( $2 + 2 \times 2 + 4 \times 2 + 8$  2-bit maps). These numbers will all be potentially further reduced by action B.

One necessary ingredient for constructing a tree is a definition of the core subdivisions at each step, as specified in the steps 1-3 above (top-bottom for step 1 and left-right for steps 2 and 3). The trees for two example hit maps are depicted in Fig. 61. Additionally, one must specify in what order the values from Fig. 61 are to be listed. In RD53B the values are listed as follows: The step 1 result (top line Fig. 61) is listed first. This is followed by one step 2 result (the left one if there are two). Then all the step 3 results for this step 2 are listed (can be one or two), followed by all the 2-bit maps associated with those step 3's (can be one to four). A tree will therefore always begin like that: step 1, step 2, step 3. What follows can be another step 3 if there is one, and then the 2-bit maps. If there is another step 2 it will come after the last map for the first step 2. This is summarized as:

`s1, s2, s3, [s3], map, [3x[map]], [s2, s3, [s3], map, [3x[map]]]`

The right side of Fig. 61 only contains the minimal number of elements:

```
s1, s2, s3, map  
01 01 01 10
```

1685 while the left side contains additional branches:

```
s1, s2, s3, map, s2, s3, map, map  
11 10 01 11 11 01 10 01 10
```

**(B) Bit code replacement.** It should be clear from Fig. 61 that the bit code 00 never appears, since only maps with at least one hit are being encoded. As there are only three used 2-bit codes, one of them can be replaced with a 1-bit code. The substitution  $01 \rightarrow 0$  is made everywhere. This is a minimal case of Huffman coding. The encoded maps for Fig. 61 thus become:

- $11\ 10\ 01\ 11\ 11\ 01\ 10\ 01\ 10 \rightarrow 11\ 10\ 0\ 11\ 11\ 0\ 10\ 0\ 10$  (15 bits instead of 18)
- $01\ 01\ 01\ 10 \rightarrow 0\ 0\ 0\ 10$  (5 bits instead of 8)

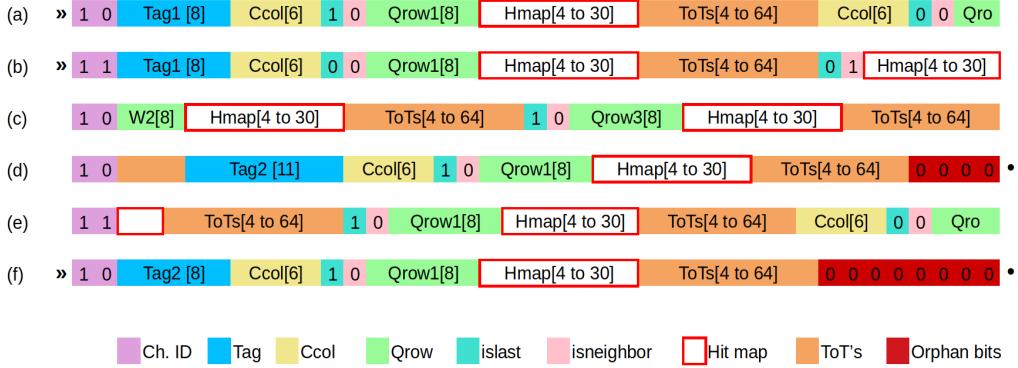
1695 Note that the choice  $01 \rightarrow 0$  instead of  $10 \rightarrow 0$  is arbitrary and makes no difference in the data volume for the given choice of subdivisions, as they are symmetric.

The binary tree encoding is elegant because it has an algorithmic form (as described above). However, in the chip it was implemented with an 8-bit lookup table where each binary value is mapped to its encoded value. Had this implementation choice been known in advance, a Huffman encoding could have been used and would have resulted in slightly higher efficiency.

## 1700 10.7 Multi-chip encoding

Data merging combines data from multiple chips onto a single Aurora output. In this mode each chip still produces streams, but the merged data contains Aurora data blocks from multiple chips. Each 66-bit Aurora block still contains data from only one chip, but blocks from different chips are interleaved. To reconstruct the streams from a given chip, the DAQ must be able to determine which Aurora block belongs to which chip. This is possible thanks two chip ID bits immediately after the NSbit at the start of every Aurora 64 bit block (before scrambling). The presence of these chip ID bits is enabled by default and can be disabled by zeroing bit “Ch. ID” of the DataMerging configuration register in Table 23. Thus, instead of 1/64 overhead from the NSbit, one has 3/64 overhead (NSbit plus 2 ID bits). These two ID bits are the least significant bits of the wire bonded chip ID. All other aspects of the stream encoding remain the same. Fig. 62 shows the bit content of a hypothetical merged data output containing two streams, one from chip ID=10 and another from chip ID=11, extending across multiple Aurora blocks.

1715 Because the stream protocol respects Aurora blocks, the decoder just needs to combine all blocks with the same ID in order to reconstruct the streams from that chip. Multi-chip encoding is the default setting, as the presence of ID bits upon power up will be a nice diagnostic tool even when not using data merging. For maximum data transmission efficiency, single-chip encoding would be selected upon configuring the chip.



**Figure 62:** Example of encoded and merged data outputs from two chips with ID LSBs 10 and 11. Six Aurora blocks are shown (a-f), four belonging to chip 01 and two to chip 11 (blocks b and e). The NSbits are not shown, but stream boundaries are shown instead: the start of streams is indicated with the '»' symbol and the end with a dot. Chip 10 has two streams containing 3 events. Event 1 starts in (a) and end in (d). It has two hit Ccols with one hit Qrow in the first and two in the second. Event 2 shares the same stream with event 1. It starts and ends in (d) with only only hit qrow and qcol and is followed by orphan bits. Event 3 starts and ends in (f) in its own one-block stream. Chip 11 has one event with three hit Qrows in the first hit Ccol, the second Qrow being a neighbor of the first. There may also have been Aurora idles or non-data words (such as register readback), which would have been removed or split off by the decoder and are not part of the streams.

## 10.8 Event size limit and data filtering

Unphysically large events due to exception conditions can cause readout problems and it may be desirable to suppress them. Two levels of truncation are available, applied prior to Aurora encoding, such that unwanted data are discarded as early as possible. The first is at the core-column level, where a maximum number of hits (in multiples of 4) allowed for any single core column can be programmed in the MaxHits[3:0] field of the CoreColEncoderConf configuration register (Table 23).

This feature must be enabled by setting bit “rem. hits” of CoreColEncoderConf. When enabled, hits in excess of MaxHits will be discarded prior to encoding and the unphysical qrow number 204 will be added with the islast bit set, to tell the DAQ that column truncation has taken place. A bitmap and a single ToT value will be included in qrow number 204 to comply with the encoding format even though they are meaningless. The added compressed bitmap will be binary 0000 and ToT also 0000. This protects against global occupancy extremes, but not against uniform high (but not extreme) occupancy everywhere.

The second truncation mechanism is a readout timeout for each event. If the time elapsed since the readout of an event started reaches a programmed threshold, any further data in that event will be cleared and the event readout will be ended. When this happens, the unphysical qrow number 205 will be added with the islast bit set, to signal to the DAQ that timeout truncation has taken place.

In addition truncating events too large to be meaningful, it can be desirable to filter out hits

known to be backgrounds. A configurable filter is implemented using isolation and/or ToT. The anticipated use case is discarding low ToT single hit clusters, but users may configure the filter any way they like. For each pixel hit the filter can cut on ToT and on the presence of hits in a programmed pattern of neighboring pixels. A 3-bit ToT cut value is programmed in the MaxToT[2:0] field of the CoreColEncoderConf configuration register (Table 23). Hits below this ToT value are discarded if there are no hits in the selected neighbor pixels.

The filter can only check for neighbor hits within a core column and not across core column boundaries. Consequently, it cannot check all the neighbors of pixels on the edges of the core column. Thus, only 75% (50%) of low ToT single hit clusters can be discarded by this filter for  $50 \times 50$  ( $25 \times 100$ )  $\mu^2$  pixels. As the purpose of this filtering is to reduce the impact of backgrounds on bandwidth utilization, the benefit from filtering 100% of backgrounds was not worth the increased complexity needed. Furthermore, not rejecting a fraction of the background (which can of course still be rejected off-line if desired), has the benefit of allowing to monitor the background level and the effectiveness of the on-chip filtering.

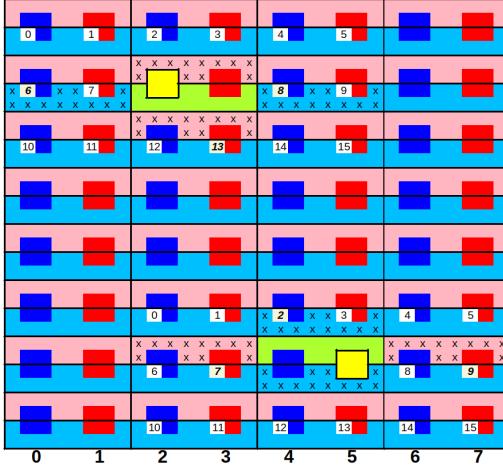
The desired neighbor map must be programmed as 16-bit patterns in the EvenMask and OddMask configuration configuration registers (Table 23). The neighbor bit patterns tell the filter which pixels to consider as a given pixel's neighbors. EvenMask (OddMask) is applied to even-numbered (odd-numbered) core columns, as illustrated in Fig. 63. To keep the mask to 16 bits, which conveniently fits one global register, the second pixel in the same row and same column pair is always considered a neighbor and, therefore, does not require a mask bit. The 16-bit mask allows selection of diagonal neighbors as well as up/down and left/right. Fig. 63 shows the more complex case of a  $25 \times 100$  sensor. For a  $50 \times 50$  sensor, the mask bits selecting all neighbors (including diagonal) would be 1,2,3,7,11,12,13 (2,3,4,8,12,13,14) for even (odd) columns.

In RD53B ATLAS the filter can only be enabled globally with bit [7] of the CoreColEncoderConf configuration register (Table 23), so it is applied to all core columns or to none. Thus, when using the filter, core columns where dead pixels will cause real non-isolated hits to appear isolated will result in a larger inefficiency than that from the actual dead pixels themselves.

The order of hit truncation and filtering is as follows. First column hit truncation is performed (if the option is enabled). Then Isolated Hit Removal is applied (on the remaining hits, if enabled). All Hits that survive the filtering process will then be encoded. Hit Removal, based on global timeout, is instead performed while building an event, so after those steps are performed.

## 10.9 Precision ToT data

A Precision ToT (PToT) block (Sec. 13.7) is present in every core column and generates four times 16 bits of data, one for each HitOr bus in the core column. These data are stored and triggered the same way as normal hit data, except that the timing can vary: because the HitOr can have a relatively long delay, PToT data can be recorded in the different bunch crossing than the regular hits. The 16 PToT bits consist of 5 Time of Arrival (ToA) bits and 11 ToT bits, as explained in Sec. 13.7. The bits are inverted prior to encoding to make the encoding (described next) more efficient: the bits should be inverted again to recover the correct ToA and ToT values after decoding. For the purposes of readout, the 16 bits are considered as a set of four 4-bit fragments. Thus there are sixteen 4-bit fragments. In this way the data can be encoded for readout exactly the same way as a pixel quarter core containing 16 pixels, each with its 4-bit ToT. After encoding there will be a hit



**Figure 63:** Bump pattern for one core bump bonded to a 25x100 pixel sensor. Column numbers are shown along the bottom. The red (blue) square bump pads are connected to the pink (light blue) pixels. Two pixels/bump pads are highlighted in lime/yellow and the 16 pixels identified by the 16-bit neighbor mask for each are numbered. The up,down,left,right neighbors (hatched) of the upper yellow pixel are selected by setting mask bits 6, 8, and 13; while for the lower yellow pixel by setting 2, 7, and 9. The companion pixel in the same column pair (not numbered) is always set as a neighbor regardless of the mask.

map (compressed or not), which is not actually mapping hits but simply indicating which of the 4-bit fragments are not 1111 (since 1111 is the ToT code for no-hit), followed by all the non-1111 4-bit fragments. (This is why the bits were inverted, since before inversion 0000 fragments will be common while 1111 will be rare.) For example, if the 16-4-bit fragments (expressed in Hex) were 0 0 0 0 5 C F 3 0 0 2 F 0 1 9 2, this will result in the hit map (in binary) 1111 1101 1110 1111 followed by the 4-bit codes (in Hex) 0 0 0 0 5 C 3 0 0 2 0 1 9 2. Only the two F's are missing, and two 0's in the hit map indicate where they belonged. For any given event, some of the four HitOrs may not have fired at all, which is the same as a quarter core without any hits and entirely suppressed from the readout. Within each 16-bit fragment corresponding to one HitOr bus, the bit assignment is shown in Table 16.

Bit position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Decoding	PToT [3:0]				PToT [7:4]				PToA [0:0]	PToT [10:8]			PToA [4:1]			

**Table 16:** Bit assignments for the 11-bit precision ToT and 5-bit precision ToA data within the 16-bit fragment corresponding to one HitOr bus.

The ToA and ToT functions are independently enabled with the PToA and PToT bits of the ToTConfig configuration register (Table 23). 9-Bit field “PToT Latency” of ToTConfig defines a dedicated trigger latency for the PToT block, analogous to the Latency value in TriggerConfig register for regular pixel readout. Additionally, each core column has a dedicated enable bit for its PToT block in registers PrecisionToTEnable\_0 to \_3.

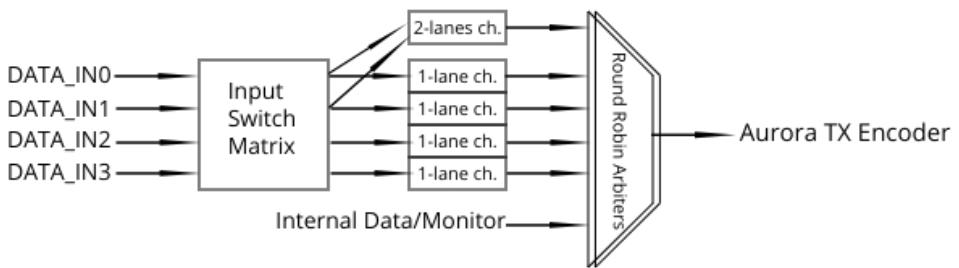
<b>Option</b>	<b>Register</b>	<b>Default</b>	<b>Description</b>	<b>Section</b>
En. Chip ID	68	on	Include Chip ID[1:0] after NSbit	10.7
EoS marker	74	on	000000 w/enable bit	10.5
En. BCID	74	off	Insert BCID[10:0] after Chip ID. <b>N/A in RD53B_ATLAS</b>	
En. Trig ID	74	off	Insert trigger ID[7:0] after Chip ID. <b>N/A in RD53B_ATLAS</b>	
Raw map	75	off	Don't compress hit maps. Map is always 16 bits	10.4
Drop ToT	75	off	No ToT values- hit/no hit information only	10.4

**Table 17:** Optional elements that can be enabled/disabled and affect stream efficiency. The Register column lists the global register number of Table 23 where the relevant enable bit resides. The “N/A in RD53B\_ATLAS” functions were not implemeted, but the configuration bits are reserved for them.

The stream format described in so far and exemplified in Fig. 59 is designed for maximum lossless efficiency in bandwidth utilization. More information can be added for debugging or for other functionality when such high efficiency is not needed, or conversely when lower bandwidth utilization must be obtained. Table 17 collects the available options. They can be used in any combination.

## 11. Multi-Chip Data Aggregation

The RD53B has four differential receivers (Sec. 11.1) that allow one chip (called primary) to aggregate serial data from one or more other chips (called secondaries) and merge it with its own output. The receivers are compatible with the differential data outputs (Sec. 10.1) of other chips. Fig. 64 shows the block-level schematic of the data merging path. The data receivers are designed to work at 320 Mbps, and so the secondary chip outputs must be configured to operate at 320 Mbps, instead of 1.28 Gbps. On the other hand, the primary chip output must be configured high enough to carry all inputs plus its own data. So either 640 Mbps for a single secondary input or 1.28 Gbps for multiple inputs. On an experimental basis the data inputs can be configured to operate at 640 Mbps, in which case the primary chip output must be configured for 1.28 Gbps for a single secondary input or two lanes at 1.28 Gbps each for multiple inputs.



**Figure 64:** Block level schematic of the data merging path.

### 11.1 Data Receivers

### 11.2 Setup and Operation

Lane	DataMergingMux bits	Value (dec)	Input selected
0	[9:8]	0	DATA_IN0
1	[11:10]	1	DATA_IN1
2	[13:12]	2	DATA_IN2
3	[15:14]	3	DATA_IN3

**Table 18:** Input switch matrix configuration using register DataMergingMux of Table 23. The right side table indicates which input is connected to the lane based on the programmed 2-bit value.

The first block in the data path shown in Fig. 64 is the input selection matrix. With this switch matrix it is possible to select which external serial input is connected to which internal serial lane. This provides flexibility to operate single chip modules in different output modes via configuration, allowing the use of identical modules in different regions of a detector. This configuration is done from the point of view of the internal serial lane: using register DataMergingMux according to table 18, each internal lane is fed from a specified data receiver. No sanity check is present to enforce that the selection of inputs is mutually exclusive, or that all connected receivers are used.

Register bit field	Description
[0]	Enable two-lanes Aurora channel using internal lanes 0 and 1
[1]	Enable single-lane Aurora channel using internal lane 0
[2]	Enable single-lane Aurora channel using internal lane 1
[3]	Enable single-lane Aurora channel using internal lane 2
[4]	Enable single-lane Aurora channel using internal lane 3

**Table 19:** Aurora decoder channels enables using register DataMerging of Table 23. Internal lanes are routed as configured according to Table 18.

All configurations, including unreasonable ones, are possible. For example, one can feed multiple input lanes from a single data receiver, which will lead to duplicate data blocks in the final output.

The internal lanes are then routed to Aurora decoder channels. Lanes 0 and 1 are routed to one two-lane Aurora channel, which interprets these two lanes as a single serial stream from a chip using two output lanes. In parallel, all lanes (0-3) are routed to four single-lane Aurora channels. Each Aurora channel can be separately enabled using the register DataMerging of table 19. As before no sanity check is performed and it is possible to enable an unreasonable combination of Aurora channels. Normally either the two-lane channel, or up to three single-lane channels will be enabled- never both. The enable bit provides clock to the channel so that it can function.

For example, one can read three chips on a single output link, by configuring one chip as primary (connected to the output link) and two as secondaries, each with their active output connected to one of the primary's data receivers. Assuming these receivers are DATA\_IN0 and DATA\_IN2, the primary chip would be configured with DataMergingMux[15:8] = binary xx.xx.10.00, which means internal lane 0 is fed from receiver 00, while internal lane 1 is fed from receiver 10 (little-endian for 1). The values for internal lanes 2 and 3 do not matter (xx), because these lanes will be disabled. Register DataMerging[4:1] is set to binary 0011, which enables the single-lane Aurora channels for internal lanes 0 and 1. Note that in this particular configuration the 1.28 Gbps primary output can never be saturated, as it is fed from three 320 Mbps lanes.

In the case of two chip sharing one 1.28 Gbps output link, there will be one primary chip and a single secondary chip with two outputs connected to two primary chip inputs. Assuming these are DATA\_IN3 and DATA\_IN1, The primary chip must be configured with DataMergingMux[15:8] = binary xx.xx.01.11 and DataMerging[4:0] = binary 00001 (which selects the two-lane Aurora channel). The data receivers must be routed to lanes 0 and 1 as they are the only lanes used by the two-lane channel. Additionally, the order of lanes matters in the two-lane configuration: output lane 0 of the secondary chip must be connected to the internal lane 0 of the primary chip.

The Aurora input channels are decoded and separated in two data buffers: one for hit data and the other for monitoring. This prepares the incoming data to look just like the internal data from the primary chip before Aurora encoding. Incoming idles, channel bonding blocks or any other Aurora protocol blocks are discarded and not buffered.

Every input lane plus the internal data from the primary chip are then routed to two round-robin arbiters, one for data and the other for monitor. The arbiter will select data from the next non-empty input according to the order: single-lane channel 0, 1, 2, 3, two-lanes channel, internal data path. Note that inputs that are not enabled are guaranteed to be empty, since their clock is

gated off and therefore they cannot be filled. The outputs of the arbiters are fed in input to the  
1855 primary chip's Aurora encoder block, that at this stage doesn't distinguish the origin of the data  
(from secondary chips or primary) and transmits a single stream as configured, inserting idles and  
other Aurora markers as needed. Consider for example a three-chip system with one primary  
and two secondaries. If all chips are full of data, the final output will have a round robin mix of  
Aurora blocks: secondary0, secondary1, primary, secondary0, secondary1, primary..., with Aurora  
1860 markers inserted as needed. However, If only the primary chip has data and the secondary chips  
do not, the final output will contain all Aurora blocks from the primary chip (primary, primary,  
primary, primary,...), with Aurora idles inserted only when the primary chip runs out of ready  
data, as opposed to a fixed time-domain division of the output link giving idle, idle, primary, idle,  
idle, primary... This is an important consideration when building systems. In particular, it makes  
1865 it possible for the link sharing to work with different data rates on different chips, as the output  
will always be filled with whatever input has data. For example if the primary chip produces data  
at 640 Mbps while two secondaries produce data at 320 Mbps each, half of the 1.28 Gbps output  
bandwidth will be taken by the primary chip and only one quarter by each of the secondaries. The  
chip with the highest data rate should always be made the primary, because secondary chips will  
1870 be limited by the inter-chip link bandwidth.

## 12. Sensing and Monitoring Functions

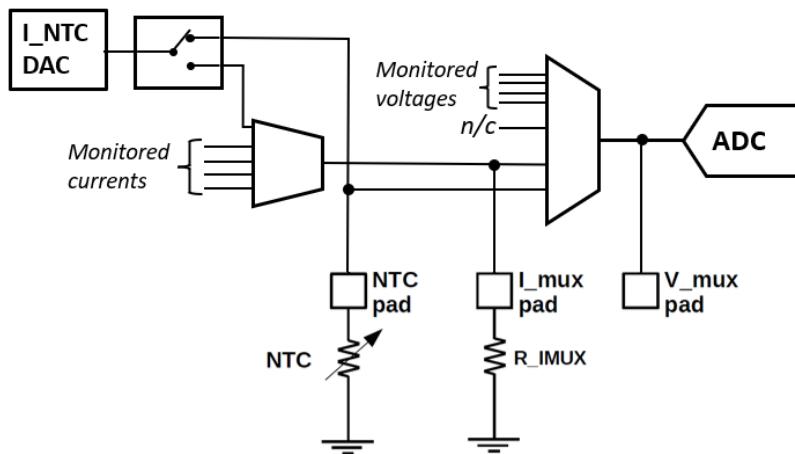
The Monitoring block in RD53B enables digitization and readout of internal parameters, such as the temperature, the total ionizing dose, and voltages or currents from different parts of the chip and even external ones. Monitoring can be performed at any time, including during data-taking.

<sup>1875</sup> Monitoring data are transmitted via the normal data output links, time-multiplexed with event hit data (Sec. 10.2). The monitoring procedure entails first selecting what to monitor by routing the given signal to the chip's internal ADC (Tables 26 and 27 show all the available signals), then triggering the digitization action so that the ADC digitizes the signal (Sec. 12.2.5), and finally reading out the digitized value via service data blocks using Read\_Register commands or the auto-read function (Sec. 10.2). Temperature and radiation sensors that feed the Monitoring block are distributed as shown in Sec. 2 and described in Sec. 12.3 and 12.4.

The Monitoring block is depicted by the Fig. 65 and contains two sub-blocks:

- An analog current multiplexer followed by an analog voltage multiplexer (MUX)
- A 12 bit Analog to Digital Converter (ADC)

<sup>1885</sup> The output of the current multiplexer has a dedicated wire bond pad ( $I_{\text{mux}}$  pad), which can be measured externally or turned into a voltage through connection of an external resistor to ground,  $R_{\text{IMUX}}$  (see Sec. 15.7). The voltage at the  $I_{\text{mux}}$  pad is then one of the inputs to the voltage MUX. Another dedicated wire bond pad sources a known current defined by a 10-bit DAC ( $I_{\text{NTC}}$  DAC) that can be sent to an external device to ground, nominally an NTC for silicon detector temperature <sup>1890</sup> measurement, and so is called NTC<sub>pad</sub>. The voltage at NTC<sub>pad</sub> is another input to the voltage MUX. The voltage MUX output feeds the ADC, and also has its own dedicated wire bond pad ( $V_{\text{mux}}$  pad) for optional external measurement and calibration of the ADC.



**Figure 65:** Diagram of monitoring block with current and voltage MUXes feeding the input of the ADC.

## 12.1 Analog Multiplexer (MUX)

The analog multiplexer has a set of CMOS transmission gates used to connect the analog inputs to a common output. This multiplexer is controlled through selection bits and only one transmission gate is set in the ON state at any given time. Each transmission gate is built with a parallel combination of NMOS and PMOS transistors driven by a complementary gate. When all the selection bits are set to all 1, the multiplexer output is at the high-Z state and the ADC can be calibrated with an external voltage source through the V\_mux pad.

### 1900 12.1.1 Multiplexer Configuration

Global configuration register 99 (MonitorConfig) is used to enable and select the routing of the multiplexers (See Table 23). The IMUX and VMUX have both a 6 bit selection value. Each value selects a different input, but not all 64 values may be used. The list of inputs is given in Tables 26 and 27 of the Reference Sec. 15. Each Mux can be disabled with the output in a high-Z state using the setting corresponding to 63 (all ones).

#### Examples:

- For monitoring the voltage of the TEMPSENS placed near the analog SLDO (see Sec. 2), channel 14 of the V\_mux should be selected (Table 27), so the MonitorConfig configuration register is set to binary 1000000001110 (decimal 4110)
- For Monitoring the NTC current bias, the channel number 9 of the I\_mux (Table 26) and the channel number 1 of the V\_mux (Table 27) should be selected. The MonitorConfig configuration register is set to binary 1001001000001 (decimal 4673).
- For temperature measurement with an external NTC, the channel number 2 of the V\_mux (Table 27) should be selected (after the above has been done, which provided a measurement of the actual current sent to the NTC). The MonitorConfig configuration register is set to binary 1000000000010 (decimal 4098).

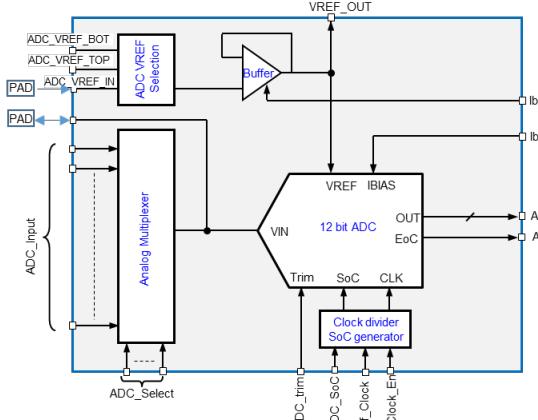
## 12.2 General Purpose ADC

Fig. 66 shows the main circuit elements of the monitoring block:

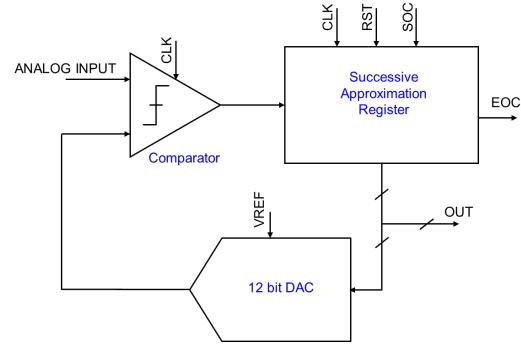
- The 12 bit General Purpose ADC proper, including the clock divider and the start of conversion signal generation circuit
- The reference voltage selection and buffering
- The input stage analog multiplexer as described earlier

The 12-bit ADC is based on a Successive-Approximation Register (SAR) architecture. It is the most popular architecture for data-acquisition applications, especially when multiple channels require input multiplexing. The circuit takes the chip bunch crossing clock, nominally 40 MHz, and divides it down with a 1024:1 frequency divider to generate the internal clock driving the ADC at 39 kHz. The SAR ADC consists of three main circuits (Fig. 67):

- A 12-bit DAC based on a capacitance network supplied through the reference voltage (Vref\_ADC) to generate the voltage scaling



**Figure 66:** Monitoring block diagram showing ADC.



**Figure 67:** Block diagram of 12-bit SAR ADC.

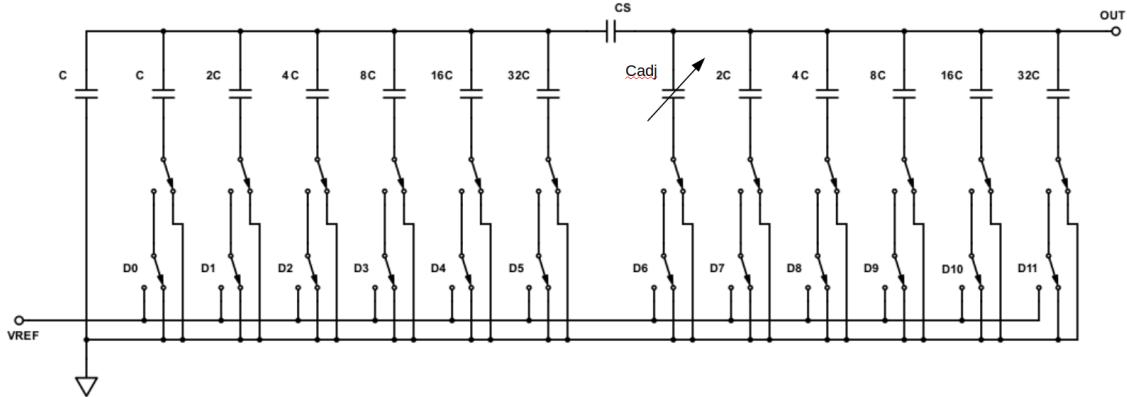
- 1930 • A high sensitivity comparator  
• A SAR logic block including the frequency divider mentioned above

The ADC digitized output depends on the input and references voltages:

$$\text{ADC}_{\text{out}} = A \times \frac{V_{in}}{\text{Vref\_ADC}} + B + \text{nonlinear terms} \quad (12.1)$$

where  $A$  is the conversion factor,  $B$  is an offset and the nonlinear terms are ideally of order of the LSB or less. Both  $A$  and  $B$  can be determined from calibration. Prototype measurements with 1935  $\text{Vref\_ADC} = 0.9 \text{ V}$  have shown a differential nonlinearity of less than  $\pm 1 \text{ LSB}$  ( $200 \mu\text{V}$ ) and an integral nonlinearity less than  $\pm 2 \text{ LSB}$ .

### 12.2.1 12-bit DAC



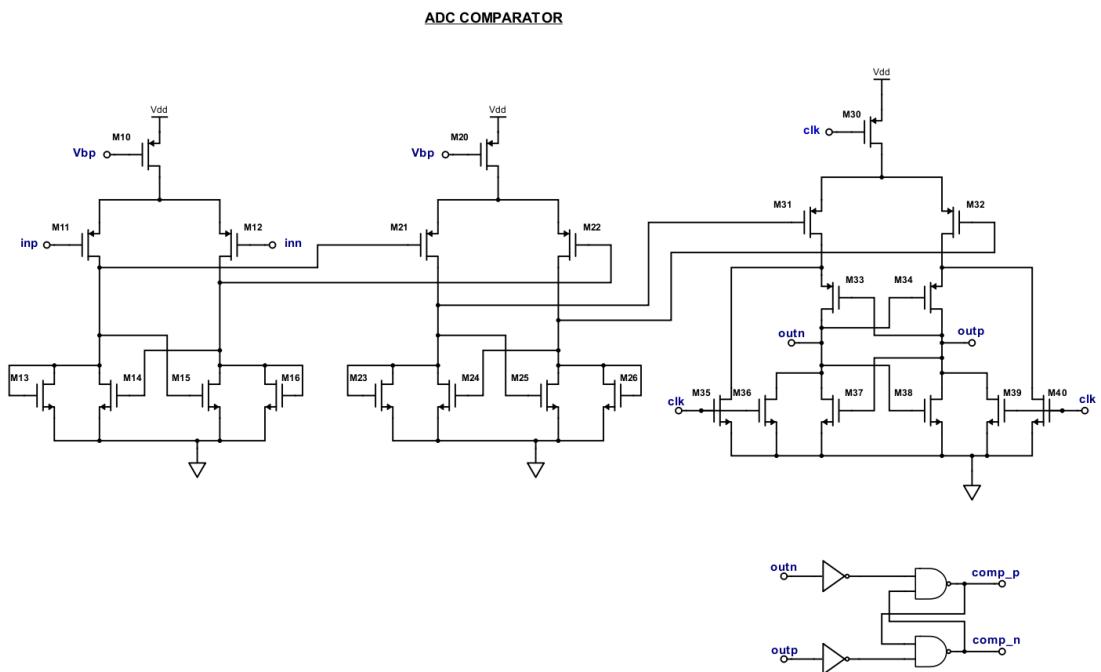
**Figure 68:** 12-Bit capacitive DAC.

The precision and the linearity of the SAR ADC rely mainly on the capacitive DAC. Two main DAC structures are in general used in SAR ADCs: Binary-weighted and bridge structures.

1940 The bridge structure shown by the Fig. 68 was adopted for this design, as it leads to larger unit capacitance, allowing better element matching and thus higher resolution. However, it suffers nonlinearity caused by mismatch of the bridge capacitance, CS, and by parasitic capacitance in the DAC array. Since this capacitance is insensitive to temperature variation, it can be calibrated to compensate the non-linearity. Six trimming bits are a part of the global register MON\_ADC,  
1945 allowing to adjust the capacitor Cadj between  $2C$  and  $C/2$  in order to compensate the non-linearity from CS and the parasitic capacitance.

The unit integrated capacitance, C, is chosen to keep the mismatch as low as possible and to achieve a very low noise for high accuracy and good linearity. The ratio of output voltage to the reference voltage of the DAC is determined by a capacitance ratio, which makes this stage very  
1950 tolerant to the radiation damage.

### 12.2.2 ADC comparator



**Figure 69:** Schematic of ADC comparator.

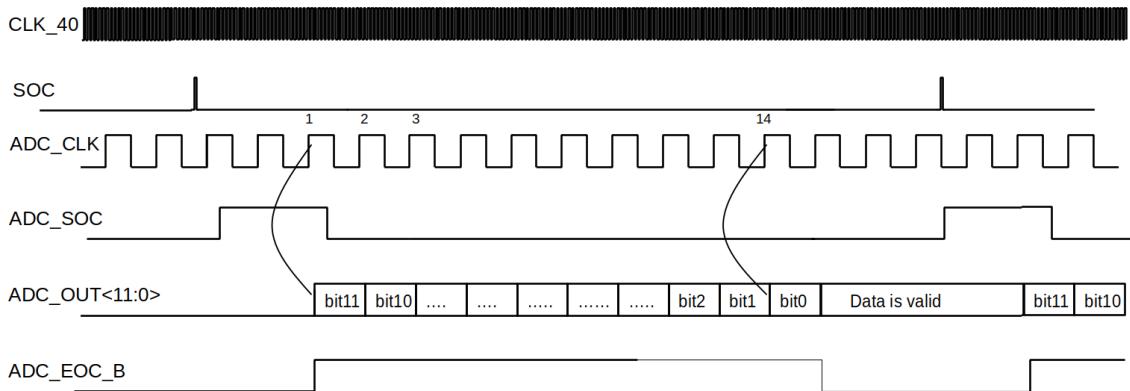
Fig. 69 shows the 3-stage comparator implemented in the 12-bit ADC design. Two differential operational transconductance amplifiers (OTAs) with diode and current source loads are followed by a dynamic latch comparator.

1955 The first stage input transistors M11-M12 sizes are critical for the linearity and the accuracy. A large transistor area reduces the offset and makes the performance less sensitive to radiation damage, but the gate-source capacitance of a large transistor, which is dependent on the input voltage, would increase non-linearity. A compromise has been found to keep low non-linearity with good tolerance to irradiation. There is no switch or reset transistor in the preamplifier stage, and  
1960 since the voltage gain of the first stage is moderate (10), this keeps the voltage swing at the drain

of the input transistor is kept small. This makes for very small kick-back noise at the input nodes. The dynamic latch comparator has a two phase of operation. When the clock signal is low, the comparator is reset and both the output nodes (outn and outp) are set to 0 V. When clock goes high, a regeneration phase starts and the cross coupled inverter pushes one output to ground and the other to VDD, which goes high and which low depending on the state of the input voltage of M31 and M32.

### 12.2.3 ADC conversion timing

The timing diagram for the SAR ADC is shown in Fig. 70. Each new conversion cycle is initiated by a Start of Conversion pulse (SOC), which is generated inside the monitoring block. This leads to an ADC start pulse (ADC SOC) with the low frequency ADC\_CLK. The ADC\_EOC\_B flag is asserted when the conversion is completed, indicating that the result can be readout from the ADC data register.



**Figure 70:** SAR ADC timing diagram.

### 12.2.4 ADC Configuration

The MON\_ADC global register holds the configuration of ADC. The 6 least significant bits hold the capacitor trimming value discussed in Sec. 12.2.1. As these bits primarily compensate for parasitic capacitance and not process, the value is not expected to vary much chip to chip. The 3 most significant bits of the MON\_ADC register select one of three reference voltage options. The default value is the Vref\_ADC pad voltage, generated by a replica of the main reference current to an external resistor. The 2 other values should be enabled when digitizing the voltages from the resistive temperature sensors (one for each sensor), as shown in Fig. 66.

Since the temperature measurement is a critical function of the monitoring block, the Vref\_ADC voltage must be stable in the range -40°C to +60°C. Since another critical function is radiation dose measurement, the Vref\_ADC voltage must also be stable vs. radiation dose. This is achieved by using a replica of the main reference current, designed to have such stability, together with an external resistor with negligible temperature coefficient. The external resistance value is chosen to

set the reference voltage around 850 mV. This determines the range and therefore the LSB of the 12-bit ADC.

### 12.2.5 ADC Control Sequence

The monitoring block is disabled most of the time (MonitorConfig[12] = 0, the default value). The

1990 command sequence required to make one conversion is as follows:

- Configure the ADC (see Sec. 12.2.4),
- Write the the MonitorConfig register: set MonitorConfig[12] to 1 to enable monitoring and set MonitorConfig[11:0] to select the channel to be monitored,
- Prepare to start the conversion: write the GlobalPulseConf Register to send the ADC StartOf-  
1995 Conversion pulse,
- Wait long enough for ADC\_EOC\_B to go low. For debugging purposes, the ADC\_EOC\_B signal can be seen on the general purpose CMOS or LVDS outputs.
- Read the ADC result register MonitoringDataADC.
- Disable the monitoring, if no other conversion is to be done.

2000 The conversion time is 14 times the ADC clock period: 358.4  $\mu$ s. The same conversion result will be read-back by every MonitoringDataADC read command until a new conversion is carried out. The read command itself does not trigger a new conversion (whatever the result was of the most recent conversion will be read back over and over). In case the auto-read register is configured to be the ADC value in order to automatically monitor some value vs. time, one will need to trigger  
2005 a new conversion periodically in order to be able to see any time variation.

## 12.3 Transistor-based Temperature and Radiation Sensors

Sensors can be made with any device that exhibits a reproducible temperature dependence. This includes resistors, MOS transistors, diodes, bipolar transistors, delay lines or delay of the logic gates. Most of the commercial temperature sensors are based on the bipolar transistors because the

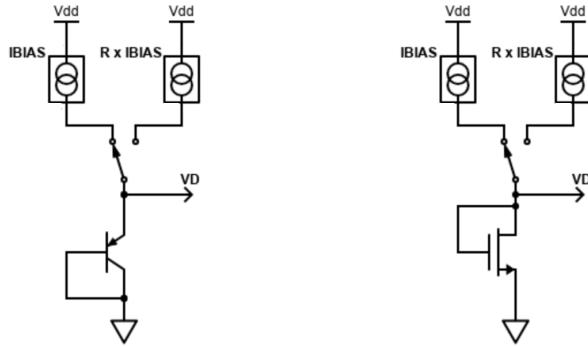
2010 difference in base-emitter voltage is very reproducible as function of temperature. (The bulk CMOS process of RD53B provides parasitic Bipolar Junction Transistors BJTs.) However, these BJTs have been found to be very sensitive to both ionizing dose and displacement damage. Therefore,  
2015 in RD53B BJTs are used as radiation sensors and diode-connected CMOS transistors are used as temperature sensors. Both cases use the same design of a single device with two switchable bias currents (Sec. 12.3.2) as shown in Fig. 71. This eliminates mismatch on the sensor itself assuming that the temperature and radiation dose are slowly varying.

The sensors have a voltage output that is an input to the VMUX (Sec. 12.1) so they can be digitized by the ADC (Sec. 12.2). The location of the sensors can be found in Sec. 2. Sec. 12.3.1 gives the theory of operation of the sensors.

### 2020 12.3.1 Transistor Sensor Theory

The voltage  $V_D$  across a diode shows a Complementary-To-Absolute Temperature (CTAT) variation. If two biases are applied,  $I_{bias}$  and  $R \times I_{bias}$ , the voltage difference will be given by:

$$\Delta V_D = V_D(R \times I_{bias}) - V_D(I_{bias}) = N_f \times \frac{k_B T}{q} \times \ln(R) \quad (12.2)$$



**Figure 71:** Diagram of BJT radiation sensor (left) and MOS temperature sensor (right) with switchable biases.

where  $N_f$  is an ideality factor (1 for an ideal diode),  $k_B$  is Boltzmann's constant,  $T$  is absolute temperature, and  $q$  the fundamental charge. The difference  $\Delta V_D$  is a Proportional-To-Absolute-Temperature (PTAT). Eq. 12.2 can be rewritten to extract absolute temperature from measured  $\Delta V_D$  and known  $R$ :

$$T = \Delta V_D \times \frac{q}{N_f \times k_B \times \ln(R)} \quad (12.3)$$

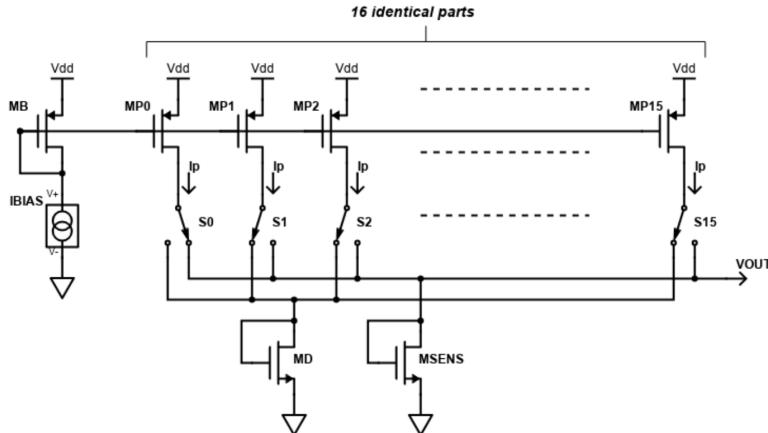
Formula 12.3 is also valid for a BJT with ideality factor close to 1.0, and for a MOS device biased in the sub-threshold region with ideality factor in the range 1.2 to 1.4. Although the temperature measurement depends logarithmically on  $R$ , to have a 1°C precision at 300 K requires a 0.3% precision on  $\Delta V_D$  and sub-percent precision on  $R$ .  $N_f$  must also be known precisely, but it is a constant that can be determined from calibration.

### 12.3.2 Precision Biases

Generation of bias currents with a ratio  $R$  is implemented with current mirrors having a ratio  $R$ . Despite all the precautions that can be taken at the layout level, it is not possible to achieve an accuracy better than 1% in this ratio. To reduce the error related to the current mirror mismatch, Dynamic Element Matching (DEM) is used. DEM consists of interchanging the unit transistor using a switch array. In the implemented design shown in the Fig. 72: 16 equal current sources generate a 1:15 current ratio, and up to 16 different values of  $\Delta V_D$  can be measured. The average value is determined offline and the error related to  $R$  ratio can be significantly reduced.

### 12.3.3 Measurement Approaches

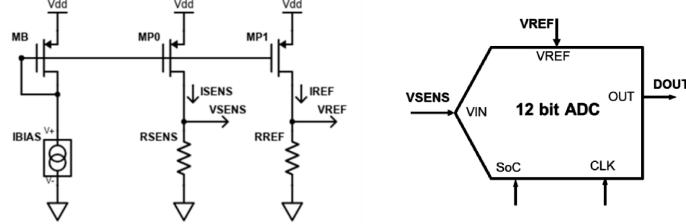
**Direct** For this approach simply measure  $V_D$  at the two different bias currents with ratio  $R$  and use Eq. 12.3. The digitized  $\Delta V_D$  is given by  $\text{ADC}_{\text{out}}(R \times I_{\text{bias}}) - \text{ADC}_{\text{out}}(I_{\text{bias}})$ . As can be seen from Eq. 12.1, the offset  $B$  cancels in this difference, leaving the conversion factor  $A$  and  $\text{Vref\_ADC}$  as sources or error. The  $A$  nonlinearity is an order 0.2% error. Although  $\text{Vref\_ADC}$  is based on a bandgap circuit, measurements of prototypes showed of order of 3 mV change in the range of -40°C to 40°C. This results in a temperature error of less than one degree, so similar to the  $A$  nonlinearity. However, irradiation of prototypes at room temperature showed a shift of -17 mV at 550 Mrad, which is a 2% or 6°C error on temperature.



**Figure 72:** Schematic of Dynamic Element Matching (DEM) circuit to generate precision biases.

**Indirect** For greater accuracy a self-compensating measurement can be made, using bandgap  
2050 voltage principle.

## 12.4 Resistive Temperature Sensors



**Figure 73:** Diagram of polysilicon resistor temperature sensor. To make a differential measurement both an input and a reference voltage are provided to the ADC.

RD53B has two temperature sensors based on polysilicon resistors as indicated in Fig. 4. They are designed to measure the temperature difference between the top and bottom edges of the chip. The small height of these devices allows one to be placed in the very limited available  
2055 space at the top of the chip. The CMOS process contains various resistor types with different temperature coefficients, allowing sensors to be implemented by comparing the resistance of two resistor types. The simple design for this differential measurement is shown in Fig. 73. Two copies of the same current are passed through two different types of resistor, and the the difference between the voltages across them is measured. Their different temperature coefficients will cause  
2060 the voltage difference to change in a known way as the temperature changes. The effect is small, so a trick is used to get maximum precision with the ADC. The trick is to use the voltage across one resistor as Vref\_ADC and the voltage across the other as the value to be measured. This results in the full 12 bits being available to measure the small difference in the two voltages (see Sec. 12.2.4). The nominal values are 16 K for the resistor connected Vref\_ADC and 10 K for

Bits	Field name	Description
MON_SENS_SLDO		
[11]	MON_SENS_SLDOD_EN	336 Enable sensor on digital SLDO
[10-7]	MON_SENS_SLDODDEM	Dynamic element matching bits
[6]	MON_SENS_SLDODSEL_BIAS	Bias selection switch
[5]	MON_SENS_SLDOA_EN	336 Enable sensor on analog SLDO
[4-1]	MON_SENS_SLDOADEM	Dynamic element matching bits
[0]	MON_SENS_SLDOASEL_BIAS	Bias selection switch
MON_SENS_ACB		
[5]	MON_SENS_SLDOA_EN	336 Enable sensor on analog SLDO
[4-1]	MON_SENS_SLDOADEM	Dynamic element matching bits
[0]	MON_SENS_SLDOASEL_BIAS	Bias selection switch

**Table 20:** Transistor sensor configuration.

2065 the one connected to the ADC input ( $V_{\text{sens}}$ ). The nominal bias current is  $32 \mu\text{A}$ . With this the condition  $V_{\text{ref\_ADC}} > V_{\text{sens}}$  over the full temperature range for all process corners.

These resistive sensors can precisely measure temperature changes, but not absolute temperature. Thus, they are ideal to measure the temperature difference across the chip. The ADC value is directly proportional to the relative temperature plus a fixed offset. Measurements on prototypes 2070 show good linearity in the range  $-40^{\circ}\text{C}$  to  $50^{\circ}\text{C}$ , with a resolution that can reach  $5 \text{ LSB} / ^{\circ}\text{C}$ .

## 12.5 Sensor Configuration

RD53B has three pairs of active temperature and radiation transistor sensors and two resistance temperature sensors. Their locations were given in Sec. 2. Two 12-bit registers (MON\_SENS\_SLDO and MON\_SENS\_ACB) are dedicated to the transistor sensor configuration as shown in Table 20. 2075 These control the sensors on the SLDO regulators and chip bottom center, respectively. To perform the measurement:

- Set the sensor enable bit,
- Set the bias switch to 0 ( $I_{\text{bias}}$ ),
- Cycle through all the DEM values and measure the voltage for each value. Average all the 2080 measurements,
- Set the bias switch to 1 ( $15 \times I_{\text{bias}}$ ),
- Cycle through all the DEM values and measure the voltage for each value. Average all the measurements.

The resistor temperature sensors do not have any configuration bits. However, the ADC must 2085 be specially configured in differential mode as explained in Sec. 12.2.4.

## 13. Test and Miscellaneous Functions

### 13.1 General purpose LVDS and CMOS outputs

RD53B contains four LVDS differential outputs and one CMOS output mainly for testing. During detector operation these outputs enable command link sharing, which means one chip can be used  
2090 as a repeater for the command serial link towards downstream chips (Sec. 3.1). These outputs also provide early diagnostics by default. They can be configured away from default to “spy” on a number of internal signals as detailed in Table 28.

### 13.2 Bypass mode

Bypass mode allows to control the chip without the internal PLL Clock and Data recovery function.  
2095 This mode can only be selected by driving a wire bond pad to high. It is available only for expert use to characterize performance of internal blocks.

### 13.3 Scan Chains

RD53B includes Design For Test (DFT) methodology in the digital flow. This allows structural testing of the bottom of chip logic and also much of the pixel matrix core logic. The General  
2100 purpose LVDS I/O as well as data merging inputs are used for DFT "scan chain" testing, which is intended to be done at the wafer probing stage. A detailed (technical) description of the DFT functionality is given in App. B.

### 13.4 Hit OR

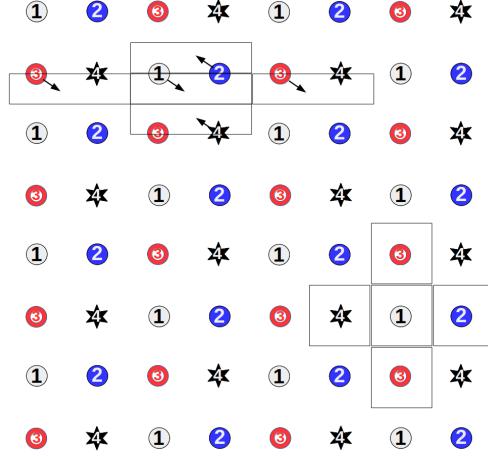
Within each RD53B core column there are four independent Hit OR nets, each one fed by one quarter of the pixels. Fig. 74 shows graphically how the 64 pixels in one core are grouped into the  
2105 4 OR networks. The figure also indicates two possible sensor formats of  $50 \mu\text{m} \times 50 \mu\text{m}$  (50x50) or  $25 \mu\text{m} \times 100 \mu\text{m}$  (25x100) pixels. It can be seen that in the 50x50 case, a given pixel in network 1 has its two up-down neighbors on network 3, and its left-right neighbors on 2 and 4. Conversely,  
2110 a given 25x100 pixel on network 1 has its left-right neighbors in network 3 and its up-down neighbors on 2 and 4.

Each net forms the logical OR of all individual pixel outputs that have been enabled by the HitOr mask bit (one bit per pixel). Because the signals travel in an OR network, there will be a different delay depending on which pixel core the signal comes from: the difference between top and bottom of chip is around 6 ns in unirradiated chips at nominal 1.2 V digital voltage. The delay  
2115 varies linearly with pixel row.

### 13.5 Heartbeat and test patterns

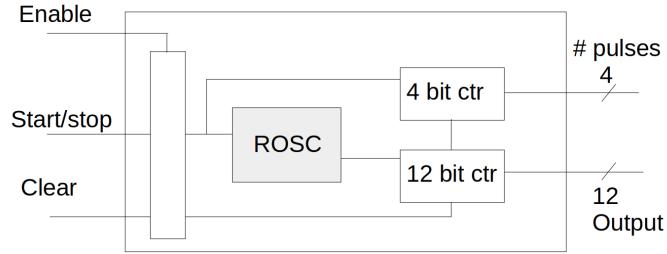
### 13.6 Ring Oscillators

RD53B contains a large variety of ring oscillators mainly intended to allow characterization of logic cell radiation tolerance, but which also allow other measurements, such variation of the pixel  
2120 injection capacitance value. The ring oscillators are located in two banks in the chip bottom: ROSCA and ROSCB, as specified in Sec. 2. Bank A (ROSRA) is a copy of the ring oscillator bank in the RD53A chip, so that a direct comparison with RD53A test results can be made. The two

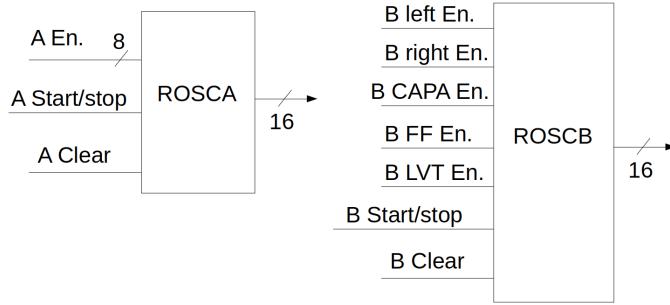


**Figure 74:** The four Hit Or nets in a 64 pixel core.

banks with their control signals are shown in Fig. 76, while a single oscillator diagram is shown in Fig. 75.



**Figure 75:** Diagram of a ring oscillator block. Different numbers and types of logic cells are used as specified in Tables 41, 42.



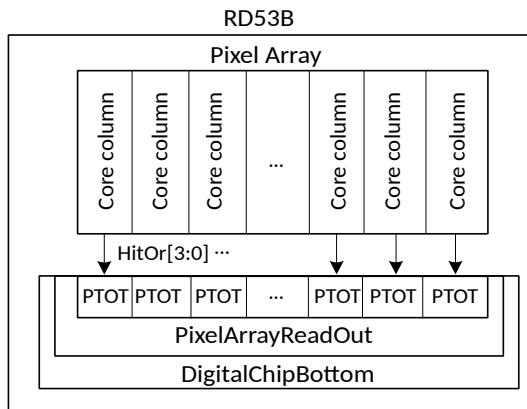
**Figure 76:** The two banks of ring oscillators with their control signals.

2125      Each ring oscillator is a chain made of different logic cells (Tables 41, 42). The number of cells in each ring was chosen to have approximately the same frequency, as given in the tables. Each oscillator drives a 12-bit counter, while a 4-bit counter is used to count the number of start/stop

pulses received. The counters will count while start/stop pulse is high as long as the Enable bit is high. The start/stop pulses are supplied using the Global Pulse command while Enable and Clear are static configuration bits. The 16-bit counter values from each oscillators are assigned to a read-only global configuration register for readout. There is one register serving the bank A (119: RING\_OSC\_A\_OUT) and one serving bank B (120: RING\_OSC\_B\_OUT).

When Enable is Low the counters hold their values. For bank B only, the clear state also causes the gates to be continually clocked at 40 MHz (not shown in the diagrams). This allows irradiation while the gates are being clocked rather than in a static value. The oscillators within a bank share some control signals and their 16-bit outputs are multiplexed. Bank A contains only 8 oscillators which have individual enable signals and common clear and start/stop. Bank B contains 4 groups of oscillators and each group has its own enable signal, common to all the oscillators in the group (see Table 42), while the clear and start/stop signals are common to the whole bank. The total number of oscillators in bank B is 34. All select and clear bits for both banks are contained in configuration register RingOscConfig (number 117). The A and B start/stop pulses are separate Global Pulse channels (Sec. 15.3). Which register is connected to each bank's output is selected with global register RingOscRoute (118), using the oscillator numbers from Tables 41, 42.

### 13.7 Precision ToT module



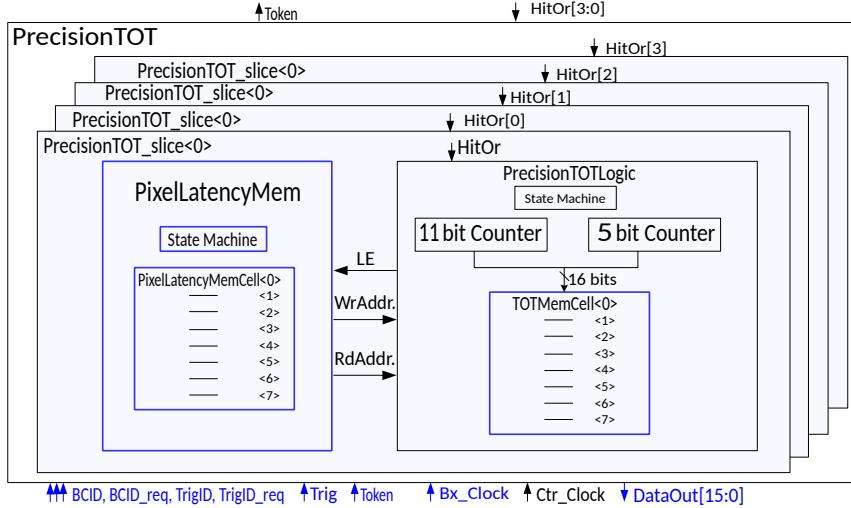
**Figure 77:** PTOT modules, one per core column.

The Precision ToT (PTOT) module makes measurements on the HitOr signals (13.4) coming out of each core column. There is one PTOT per core column as shown in Fig. 77. Two quantities are measured:

- ToT, just like in the pixels, but with an 11 bit counter counting as 640 MHz rate,
- Time of Arrival (ToA) of the leading edge, as a phase difference from the previous BX clock rising edge to the HitOr leading edge, with a 5 bit counter counting at 640 MHz rate.

These quantities are stored in memories just like the pixel region memories, associated to latency buffers that keep track of time just as in the pixel regions. A diagram the PTOT module

is shown in Fig. 78. The latency buffer depth and logic are the same as in the pixel regions. The 2155 readout of the PTOT data is trigger based, exactly as for regular pixels, and the data are included in normal data path as described in Sec. 10.9.



**Figure 78:** Single PTOT module diagram. The elements outlined in blue are copies (same code) of the pixel region logic.

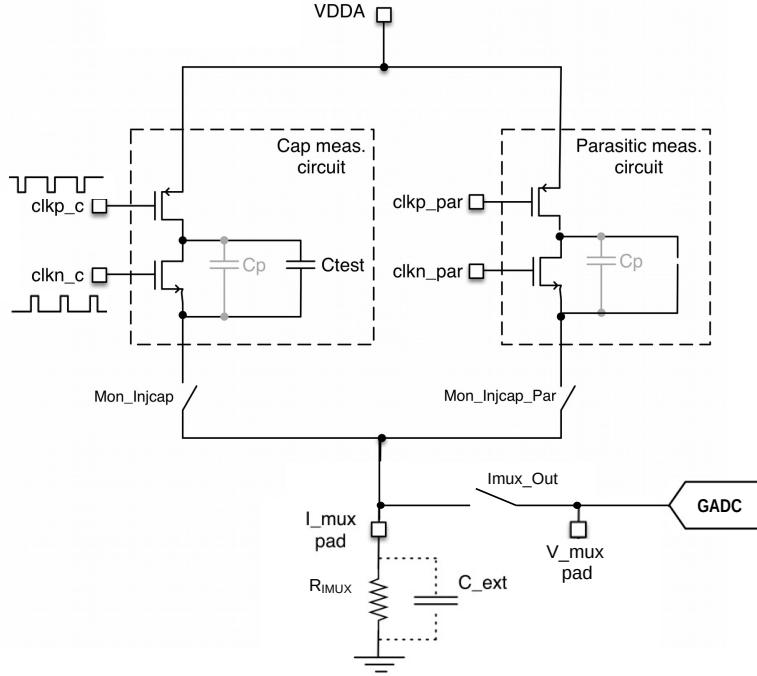
### 13.8 Capmeasure circuit

The capacitance measurement circuit (capmeasure, Fig. 79) allows the determination of the as-built front-end injection capacitor  $C_{inj}$ . It is integrated into the calibration block in the Analog Chip Bottom. The circuit consists of two sections: the capmeasure and parasitic capmeasure. The 2160 capmeasure circuit is connected to a parallel array of 100 capacitors, each identical to the injection capacitors, but connected differently than in the pixel (see Fig. 80) for use as a charge pump rather than a single charge injector. Also, in order to make an array, routing metal is needed, which adds a parasitic capacitance. An identical array but with the capacitors removed is therefore also provided in order to measure the parasitic capacitance. The total capacitance measured by this circuit is then,

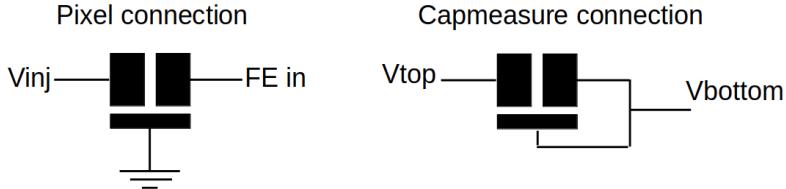
$$C_{meas} = 100(C_{pix} + \Delta C) + C_{par} \quad (13.1)$$

2165 where  $C_{pix}$  is the capacitance seen by a pixel,  $\Delta C$  is the added capacitance due to the different connection illustrated in Fig. 80, and  $C_{par}$  is the parasitic capacitance due to array metal routing. The Capmeasure circuit measures both  $C_{meas}$  and  $C_{par}$ , while  $\Delta C$  is obtained from layout extraction. The layout extracted value of  $\Delta C = 0.48$  fF, while the extracted value of  $C_{pix} = 8.02$  fF.

The capmeasure circuit is based on a charge pump with PMOS and NMOS transistors controlled by non-overlapping clocks. These clocks run at 1/4 of the bunch crossing clock (which is nominally 40 MHz). They are generated by combinatorial logic from the bunch crossing clock and are disabled by default. The En\_injcap\_meas and En\_injcap\_par configuration bits (there are 2170



**Figure 79:** Capacitance measurement circuit diagram.  $C_{\text{test}}$  is an array of 100 injection capacitors. Control of the switches  $\text{Mon\_Injcap}$ ,  $\text{Mon\_Injcap\_Par}$ , and  $\text{Imux\_Out}$ , as well as operation of the GADC generic ADC are described in Sec. 12.2.



**Figure 80:** The pixel injected charge is defined by the capacitance between the two main terminals of the injection capacitor (left). However, the capacitor has a third terminal which connects to a shield layer underneath (a poly layer usually connected to GND). Both main terminals exhibit a parasitic capacitance to this node. The capmeasure circuit charges this shield capacitance in addition to the capacitance between the two main terminals (right).

actually two control circuits, one for the array of the replica capacitors and one for the empty array to measure  $C_{\text{par}}$ ).

In a simulation of the circuit as shown in Fig. 79 with  $R_{\text{IMUX}} = 5 \text{ K}$  and  $C_{\text{ext}} = 22 \text{ nF}$ , the average current flowing in  $\text{VDDA}$  of the capmeasure circuit is  $11 \mu\text{A}$  for the  $10 \text{ MHz}$  nominal clock frequency, while the average current flowing in  $\text{VDDA}$  of the parasitic circuit is  $1 \mu\text{A}$ . This can be sensed by measuring the voltage across the external resistor with the generic ADC. The  $22 \text{ nF}$  capacitor in parallel will keep the output voltage constant for the GADC measurement (it is

2180 possible that the measurement will still be accurate without it- to be checked in bench tests).

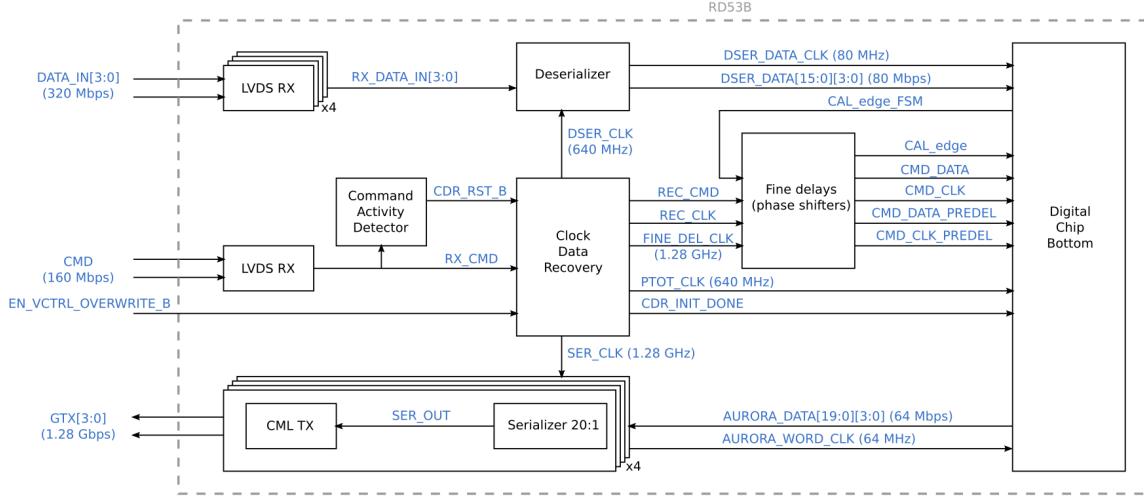
The measurements of  $C_{meas}$  and  $C_{par}$  are made separately. The capmeasure circuit should be reset before making a measurement. This is achieved by selecting capmeasure in the global pulse routing and issuing a global pulse with a 3 clock width (this is longer than the default width of 1 clock). For  $C_{meas}$  ( $C_{par}$ ), first enable the clock by setting En\_injcap\_meas = 1 (En\_injcap\_par = 1) and selecting the IMUX channel Mon\_injcap (Mon\_injcap\_par). One must wait at least 0.5 ms for the output to settle (with the nominal 5 K and 22 nF external components). At this point the voltage at the Imux\_Pad can be measured either with an external instrument or the GADC. To use the GADC select input channel Imux\_Out and follow the procedure from Sec. 12.2.

2185

## 14. Clock Generation and Data Recovery Technical Details

### 2190 14.1 I/O interface, data and clock distribution

Clock Data Recovery (CDR) circuit is responsible for recovering 160MHz clock (REC\_CLK) from the 160 Mbps CMD input and generating from it all other clocks needed inside the chip, as depicted in Fig. 81.



**Figure 81:** Simplified schematic of I/O interface together with data and clocks distribution. Configuration, bypass and monitoring related connections are omitted.

### 14.2 Clock Data Recovery

2195 The CDR circuit implemented in RD53B is significantly changed in comparison to RD53A implementation. Main changes are: different phase detector, added startup circuit and resized loop filter.

The circuit schematic is shown in Fig. 82. The CDR building blocks are:

- Phase and frequency detectors: compare the phase (and frequency in case of PFD) of edges of the CMD input with the phase (frequency for PFD) of the edges of recovered clock and the outputs UP and DN signals as a result. This functionality is implemented with three sub blocks:
  - Phase frequency detector (PFD): used during startup of the chip, makes the circuit operate in "PLL mode". This detector can only work with clock-like input (01010...), but has approximately infinite pull-in frequency range
  - Bang-Bang (Alexander) phase detector (PD): works correctly only if frequencies of input CMD and recovered clock are very close. Its role is to only adjust phase. Can work with any bit stream.
  - Rotational frequency detector (FD): compares the rate of the input CMD with the frequency of recovered clock. Has a frequency pull-in range of  $\pm 25\%$ . This detector

activates only in case the REC\_CLK frequency changes while the circuit is in normal operation mode i.e. after locking procedure. Such situation can happen e.g. due to SEE.

- Charge pumps (CP): sources or sinks the current to the LPF, based on received UP and DN signals. One CP is used by both PFD and PD, the second CP is for FD. Both CPs have the same drain switching architecture, the CP for PD & PFD has a buffer between branches for switching glitches reduction

2215 • Loop filter: second order RC filter optimized for CDR operation of the loop (PD + FD active). Integrates the CP currents and filters the VCO control voltage (VCTRL)

2220 • PMOS transistor for overwriting VC voltage: activated by pulling PLL\_VCTRL\_RST\_B pad to GND\_PLL. This will result in setting the net VC to 800 mV (generated with a resistive divider), which forces VCO to oscillate at frequency a bit higher than nominal 1.28GHz. This is a test feature to be used only in case of problems with CDR locking to correct frequency. In normal operation PLL\_VCTRL\_RST\_B pad can be left floating, since there is a 40 k $\Omega$  pull-up resistor connected to PMOS gate.

- 2225 • Voltage controlled oscillator (VCO): produces an output clock proportional to the voltage provided by the LPF. Based on a differential ring oscillator with differential to single ended converter at the output. In locked condition the VCO output frequency is 1.28 GHz
- Divider: generates several clocks (REC\_CLK, PTOT\_CLK, SER\_CLK, etc.) by dividing down the VCO output clock. Synchronous architecture, fully triplicated for better SEE immunity
  - Startup counter: responsible for automatic switching between locking mode (using PFD = "PLL mode") and normal CDR operation (using PD + FD = "CDR mode") at circuit startup

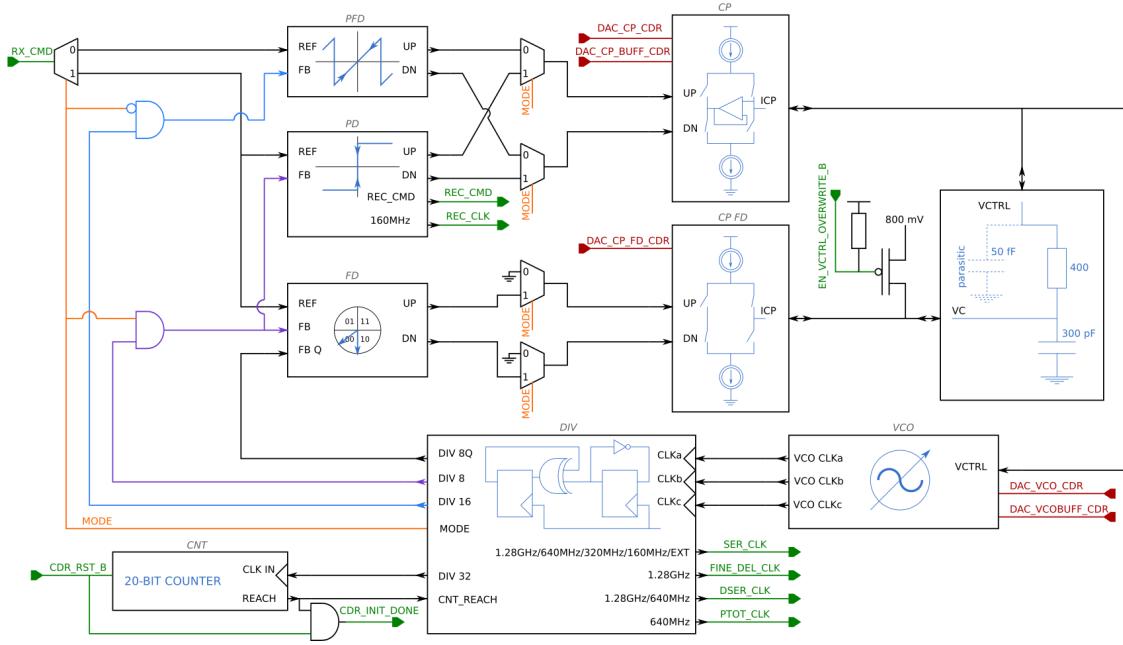
### 14.3 Startup

2235 This section expands on section 3. After the power is supplied to the chip the startup procedure for CDR is (the timing values already assume some safety factor):

2240 1. Send CMD = low frequency (approx. 1 MHz) clock signal for at least 50  $\mu$ s. This will cause assertion of CDR\_RST\_B signal by Command Activity Detector. The same can be achieved by sending a constant value (either 0 or 1), but using a slow clock is deemed a more reliable solution. As long as the CDR\_RST\_B is LOW:

- the startup counter inside CDR will be held in reset state
- CDR will enter the "PLL mode" and stay there
- CDR\_INIT\_DONE will be LOW

2245 2. Send CMD = PLL\_LOCK (noop = 010101... i.e. 80 MHz clock) for at least 2 ms (uninterrupted by any other symbols). This will cause:



**Figure 82:** Simplified schematic of the CDR. Nets drawn in green are either connected to pads or different blocks.

- release of CDR\_RST\_B after approx.  $1\ \mu\text{s}$
- after CDR\_RST\_B is released the startup counter counts down from 32768 to 0 (clocked by VCO output divided by 32). This will take approx. 0.8 ms
- once counter reaches 0 the circuit will automatically switch to "CDR mode" and shortly after that the CDR\_INIT\_DONE will change to HIGH

2250

### 3. Circuit can work now with any CMD stream.

After the startup procedure the PLL\_LOCK symbol should still be sent whenever possible to ensure best working conditions for CDR. If the global reset needs to be applied to the chip or the link has to be re-initialized, than the steps described above have to be repeated.

2255

## 14.4 Configuration

Configuration bits provided by the DCB (Digital Chip Bottom) are described in table 21.

In addition to the functionality described in this section the CDR provides features that are controlled by signals coming directly from IO pads - those are described in sections 14.3 and 14.5.

## 14.5 Bypass mode

2260

The CDR can be bypassed i.e. the REC\_CLK and SER\_CLK can be provided externally instead of being generated by CDR. To activate bypass more BYPASS\_MODE pad has to be set to HIGH. In such condition the EXT\_CMD\_CLK, EXT\_SER\_CLK and BYPASS\_CDR signals can be provided through SCAN\_CHAIN[2:0] (DATA\_IN[2:0] pads), as shown in Fig. 83.

There are two important aspects of bypass mode that the user should be aware of:

Address	Name	Size	Default	Description
54	CLK_DATA_FINE_DELAY	12	0	<ul style="list-style-type: none"> <li>bits [11:6]: Control of fine delay setting for CMD_CLK</li> <li>bits [5:0]: Control of fine delay setting for CMD_DATA</li> </ul>
58	CdrConf	5	0	<p>Contains few CDR-related settings:</p> <ul style="list-style-type: none"> <li>bit [4]: OverwriteCdrLimit. By default CDR-related DACs cannot be set below predefined limits. When this bit = 1 they can be set to any value.</li> <li>bit [3]: CdrSelPd. By default the PD uses both edges of the CMD input. When this bit = 1 it will use only the rising edge.</li> <li>bits [2:0]: Selection of SER_CLK: <ul style="list-style-type: none"> <li>0. 1.28 GHz (default)</li> <li>1. 640 MHz</li> <li>2. 320 MHz</li> <li>3. 160 MHz</li> <li>4. 1.28 GHz</li> <li>5. 640 MHz</li> <li>6. 320 MHz</li> <li>7. SER_CLK = EXT_SER_CLK</li> </ul> </li> </ul>
89	DAC_CP_CDR	10	40	Bias current of main charge pump. By default cannot be set below 15.
90	DAC_CP_FD_CDR	10	400	Bias current of FD charge pump. By default cannot be set below 100.
91	DAC_CP_BUFF_CDR	10	200	Bias current of unity gain buffer inside main charge pump.
92	DAC_VCO_CDR	10	1023	Bias current of VCO. By default cannot be set below 700.
93	DAC_VCOBUFF_CDR	10	500	Bias current of differential-to-single-ended converter. By default cannot be set below 200.

**Table 21:** Configuration registers for CDR. This table provides more detail than the main register table 23

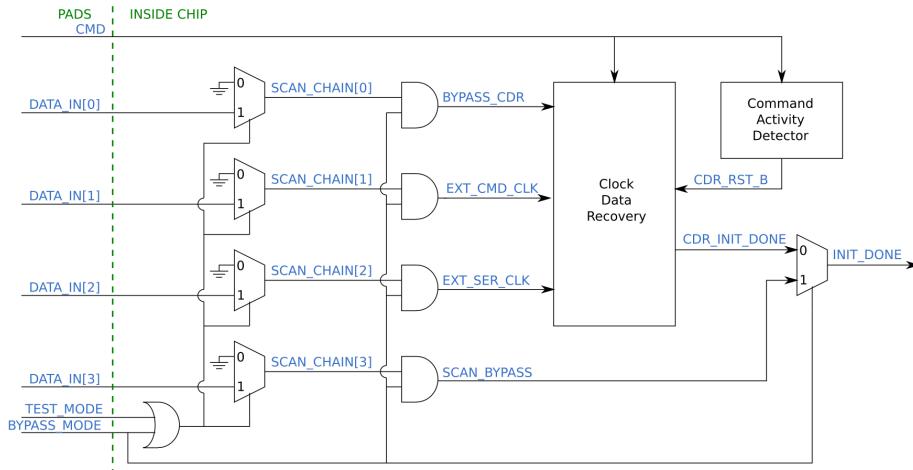
- 2265     • The PTOT\_CLK, FINE\_DEL\_CLK and DSER\_CLK will still be generated by CDR (they cannot be provided externally). This means that PTOT, data aggregation from other chips and fine phase adjustment (for CMD\_DATA, CMD\_CLK, CAL\_EDGE) cannot be used in bypass
- 2270     • When BYPASS\_MODE pad is set HIGH, the INIT\_DONE is equal to SCAN\_BYPASS, instead of CDR\_INIT\_DONE (see Fig. 83). Since INIT\_DONE is used as main reset signal of the chip (see section 3.2 for details), the user should use it appropriately

#### 14.6 Fine delay circuit (phase shifters)

CMD\_DATA, CMD\_CLK and CAL\_EDGE are each send to a dedicated shift register and output of a selected DFF in the given chain is used as the input to DCB. Each of the shift registers is com-

Address	Name	Size	Default	Description
53	CalibrationConfig	8	64	<ul style="list-style-type: none"> <li>bits [7:6]: See section 6 for details)</li> <li>bits [5:0]: Control of fine delay setting for CAL_EDGE_FSM</li> </ul>
63	ToTConfig	13	0	<ul style="list-style-type: none"> <li>bit [12]: Enable pToT clock (640 MHz)</li> <li>bits [11:0]: See section 13.7 for details</li> </ul>
68	DataMerging	12	192	<ul style="list-style-type: none"> <li>bits [11:7]: See section 10.7 for details</li> <li>bit [6]: Disable DSER_CLK (1 is default)</li> <li>bit [5]: Select DSER_CLK frequency. By default 640 MHz, increased to 1.28 GHz when this bit = 1.</li> <li>bits [4:0]: See section 10.7 for details</li> </ul>

**Table 22:** Configuration registers partially related to CDR.

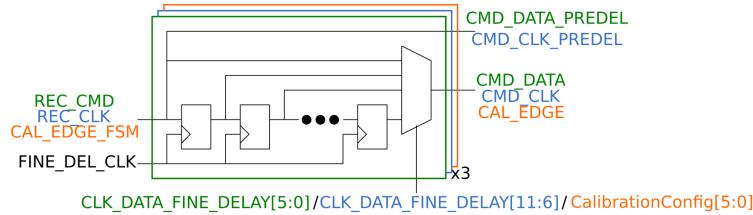


**Figure 83:** Block diagram of circuitry involved in bypassing CDR and providing external clocks.

posed of 31 triplicated DFFs clocked at 1.28 GHz, which allows to phase shift the CMD\_DATA, 2275 CMD\_CLK and CAL\_EDGE in steps of 781.25 ps. If fine delay setting is set to 0, the given signal is not re-clocked with FINE\_DEL\_CLK at all, as shown in 84.

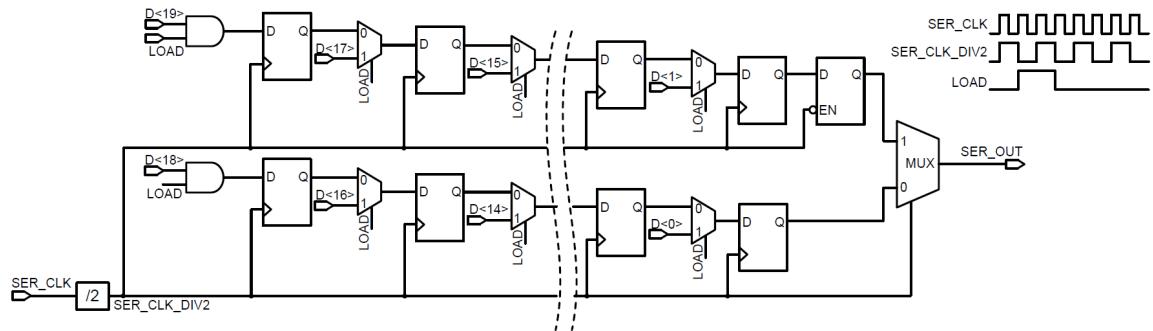
#### 14.7 Data serializer

The RD53B chip has 4 output lanes and each of them is driven by a dedicated 20-to-1 serializer. 2280 At each strobe of LOAD signal (64 MHz frequency, common for all 4 lanes and generated every 20 edges of SER\_CLK by one of serializers) each active serializer loads a 20 bit wide word from the data stream coming from DCB into serializer core. As shown in Fig. 85 the serializer core uses the SER\_CLK (nominally 1.28 GHz) divided by two to serialize data using double data rate architecture (utilizing both rising and falling edge of SER\_CLK\_DIV2). The produced data stream



**Figure 84:** Schematic of fine delay block. Each of CMD\_DATA, CMD\_CLK and CAL\_EDGE has a dedicated shift registers, as suggested by the colors. The \*\_PREDEL signal is not available for CAL\_EDGE

2285 is than again resynchronized to SER\_CLK phase with a single data rate DFF at the multiplexer output (visible in Fig. 55).

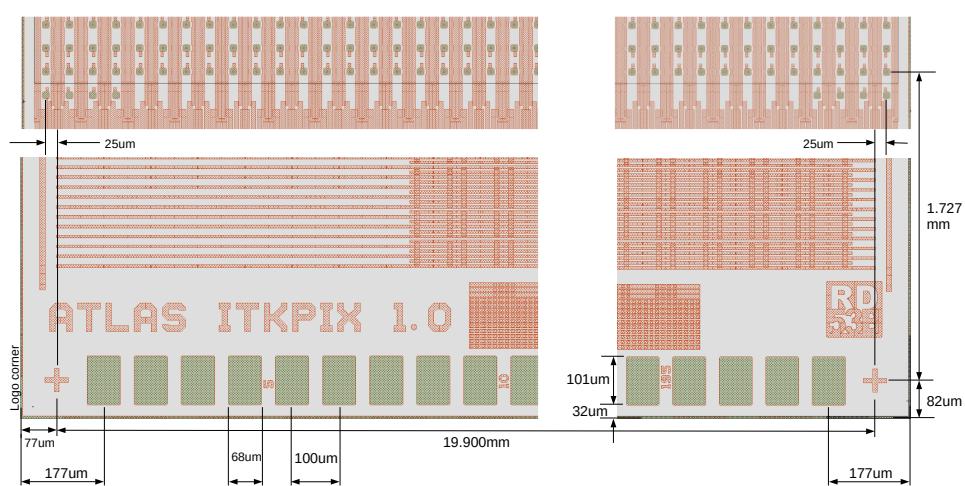


**Figure 85:** Circuit diagram for the serializer core with the timing diagram shown on the upper right.

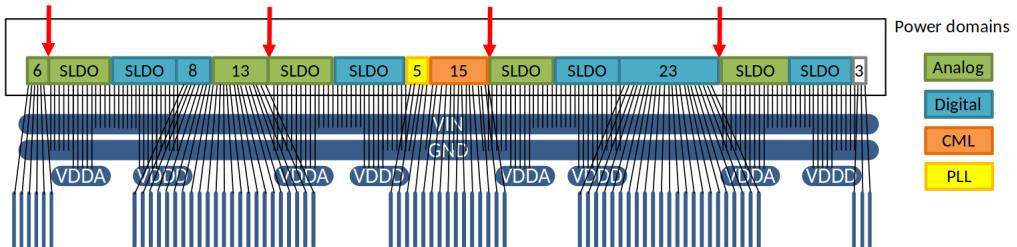
## 15. Reference Tables (pinouts, configuration, etc.)

### 15.1 Wire Bonding Pinout

The pinout is provided in a full page figure (next page, no figure number). The 198 pads are shown in two halves just to fit the page. All pads are shown and power and ground pads are color coded while all other pads are shown as open rectangles. Note that every 5<sup>th</sup> pad is numbered on the actual chip as shown in Fig. 7. Figures 7 and 8 are reproduced here for convenience (note the original figure numbers have been kept).



**Figure 7:** Detail of ATLAS chip bottom with dimensions (rounded to nearest micron). Pinout follows on next page, rotated relative to this figure (note position of logo corner).



**Figure 8:** Organization of wire bond pad frame and generic bonding scheme. All wire bonds are shown, including connections for testing (not used on detector modules). The number of fanned-out signal bonds is written in each box, while the power supply bonds run parallel (not fanned out). The red arrows indicate the four unused pads. Full pinout follows on next page.

(\*) CHIP\_ID0 (pin 43) becomes PIXEL\_MATRIX\_TEST\_MODE when scan mode is active

**Table 23:** Global configuration register main table. (\*) indicates further details are given following the table. Section references are given in (). The Size column may list multiple fields by their size in bits. The rightmost value is always the field starting at register bit [0]. This 1,3,2 would indicate 3 fields: [5],[4:2],[1:0].

Addr.	Name	Size (bits)	Description	ATLAS Default
0	PIX_PORTAL	16	Pixel portal: virtual reg. to access pix config (8.8)	0
1	REGION_COL	8	Pixel column pair connected to pixel portal (8.8)	0
2	REGION_ROW	9	Pixel row connected to pixel portal (8.8)	0
* 3	PIX_MODE	1,1,1	Pix config: Broadcast, Mask(0)/TDAC(1), Auto Row (8.8)	0,1,0
* 4	PIX_DEFAULT_CONFIG	16	Key 1 of 2: hex 9CE2 to exit pix default config.	0
* 5	PIX_DEFAULT_CONFIG_B	16	Key 2 of 2: hex 631D to exit pix default config.	0
* 6	GCR_DEFAULT_CONFIG	16	Key 1 of 2: hex AC75 to exit glob default config.	0
* 7	GCR_DEFAULT_CONFIG_B	16	Key 2 of 2. hex 538A to exit glob default config.	0
<b>Analog Front End (5)</b>				
8	DAC_PREAMP_L_DIFF	10	Input transistor bias for left 2 cols	50
9	DAC_PREAMP_R_DIFF	10	Input transistor bias for right 2 cols	50
10	DAC_PREAMP_TL_DIFF	10	Input transistor bias for top left 2x2	50
11	DAC_PREAMP_TR_DIFF	10	Input transistor bias for top right 2x2	50
12	DAC_PREAMP_T_DIFF	10	Input transistor bias for top 2 rows	50
13	DAC_PREAMP_M_DIFF	10	Input transistor bias for all other pixels	50
14	DAC_PRECOMP_DIFF	10	Precomparator tail current bias	50
15	DAC_COMP_DIFF	10	Comparator total current bias	50
16	DAC_VFF_DIFF	10	Preamp feedback (return to baseline)	100
17	DAC_TH1_L_DIFF	10	Neg. Vth offset for left 2 cols	100
18	DAC_TH1_R_DIFF	10	Neg. Vth offset for right 2 cols	100
19	DAC_TH1_M_DIFF	10	Neg. Vth offset all other pixels	100
20	DAC_TH2_DIFF	10	Pos. Vth offset for all pixels	0
21	DAC_LCC_DIFF	10	Leakage current compensation bias	100
22	DAC_PREAMP_L_LIN	10	not used in RD53B-ATLAS	300
23	DAC_PREAMP_R_LIN	10	not used in RD53B-ATLAS	300
24	DAC_PREAMP_TL_LIN	10	not used in RD53B-ATLAS	300
25	DAC_PREAMP_TR_LIN	10	not used in RD53B-ATLAS	300
26	DAC_PREAMP_T_LIN	10	not used in RD53B-ATLAS	300
27	DAC_PREAMP_M_LIN	10	not used in RD53B-ATLAS	300
28	DAC_FC_LIN	10	not used in RD53B-ATLAS	20
29	DAC_KRUM_CURR_LIN	10	not used in RD53B-ATLAS	50
30	DAC_REF_KRUM_LIN	10	not used in RD53B-ATLAS	300
31	DAC_COMP_LIN	10	not used in RD53B-ATLAS	110
32	DAC_COMP_TA_LIN	10	not used in RD53B-ATLAS	110
33	DAC_GDAC_L_LIN	10	not used in RD53B-ATLAS	408
34	DAC_GDAC_R_LIN	10	not used in RD53B-ATLAS	408
35	DAC_GDAC_M_LIN	10	not used in RD53B-ATLAS	408
36	DAC_LDAC_LIN	10	not used in RD53B-ATLAS	100
* 37	LEAKAGE_FEEDBACK	1,1	enable leakage curr. Comp; low gain mode	0,0
<b>Internal Power (4)</b>				
38	VOLTAGE_TRIM	1,1,4,4	Regulator: En. Undershunt A, D; Ana. V trim, Dig. V trim	0,0,8,8
<b>Pixel Matrix Control</b>				

**Table 23:** Global configuration register main table. (\*) indicates further details are given following the table. Section references are given in (). The Size column may list multiple fields by their size in bits. The rightmost value is always the field starting at register bit [0]. This 1,3,2 would indicate 3 fields: [5],[4:2],[1:0].

Addr.	Name	Size (bits)	Description	ATLAS Default
39	EnCoreCol_3	6×1	Enable Core Columns 53:48	0
40	EnCoreCol_2	16×1	Enable Core Columns 47:32	0
41	EnCoreCol_1	16×1	Enable Core Column 31:16	0
42	EnCoreCol_0	16×1	Enable Core Column 15:0	0
* 43	EnCoreColumnReset_3	6×1	Enable Reset for core cols. 53:48	0
* 44	EnCoreColumnReset_2	16×1	Enable Reset for core cols. 47:32	0
* 45	EnCoreColumnReset_1	16×1	Enable Reset for core cols. 31:16	0
* 46	EnCoreColumnReset_0	16×1	Enable Reset for core cols. 15:0	0
<b>Funtions: (T)rigger and timing, (I)nput/ooutput, (C)alibration, (M)asking, () Other</b>				
T 47	TriggerConfig	1,9	Trigger mode, Latency (9)	0,500
T 48	SelfTriggerConfig_1	1,1,4	Self Trig.: En., ToT thresh.: En., value (9.4)	0,1,1
T 49	SelfTriggerConfig_0	10,5	Self Trig.: delay, multiplier (9.4)	100,1
T 50	HitOrPatternLUT	16	Self Trig. HitOR logic program (9.4)	0
T * 51	ReadTriggerConfig	2,12	Col. Read delay, Read Trig. Decision time (BXs)	0,1000
T 52	TruncationTimeoutConf	12	Event truncation timeout (BXs, 0=off)	0
C * 53	CalibrationConfig	1,1,6	Cal injection: Ana/Dig (0/1), Ana mode, fine delay	0,0,0
T 54	CLK_DATA_FINE_DELAY	6,6	Fine delays for Clock, Data (14)	0,0
C 55	VCAL_HIGH	12	VCAL high level	500
C 56	VCAL_MED	12	VCAL medium level	300
C * 57	MEAS_CAP	1,1,1	Cap Meas: En. Par, En.; VCAL range bit	0,0,0
I 58	CdrConf	1,1,3	CDR: overwrite limit, phase det sel., CLK sel. (14)	0,0,0
I 59	ChSyncConf	5	Chan. Synch. Lock Thresh. (unlock is ×2)	16
60	GlobalPulseConf	16×1	Global pulse routing (15.3)	0
61	GlobalPulseWidth	8	Global Pulse Width (in BX)	1
I 62	ServiceDataConf	1,8	Service block En., Periodicity $N_D$ (10.2)	0,50
63	ToTConfig	1,1,1,1,9	En: PToT, PToA, 80MHz, 6b to 4b; PToT Latency	0,0,0,0,500
M 64	PrecisionToTEnable_3	6×1	Enable PToT for core cols. 53:48	0
M 65	PrecisionToTEnable_2	16×1	Enable PToT for core cols. 47:32	0
M 66	PrecisionToTEnable_1	16×1	Enable PToT for core cols. 31:16	0
M 67	PrecisionToTEnable_0	16×1	Enable PToT for core cols. 15:0	0
I 68	DataMerging	4,1,1,1,4,1	D. Merge: pol., Ch. ID, 1.28 CK gate, CK sel, En., Ch.bond	0,1,1,0,0,0
I 69	DataMergingMux	8×2	Input and Output Lane mapping	3,2,1,0,3,2,1,0
M 70	EnCoreColumnCalibration_3	6×1	CAL enable for core cols. 53:48	6×1
M 71	EnCoreColumnCalibration_2	16×1	CAL enable for core cols. 47:32	16×1
M 72	EnCoreColumnCalibration_1	16×1	CAL enable for core cols. 31:16	16×1
M 73	EnCoreColumnCalibration_0	16×1	CAL enable for core cols. 15:0	16×1
I 74	DataConcentratorConf	1,1,1,8	In stream En: BCID, L1ID, EoS marker; evts/strm	0,0,1,16
I 75	CoreColEncoderConf	1,1,1,4,1,3	Drop ToT, raw map, rem. hits, MaxHits, En. Isol., MaxToT	0,0,0,0,0,0
I 76	EvenMask	16	Isolated hit filter mask: Even cols.	0
I 77	OddMask	16	Isolated hit filter mask: Odd cols.	0
78	EfusesConfig	16	Efuses En. (to Read set 0F0F, to Write set F0F0)	0
79	EfusesWriteData1	16	Data to be written to Efuses (1 of 2)	0
80	EfusesWriteData0	16	Data to be written to Efuses (2 of 2)	0
I 81	AuroraConfig	1,4,6,2	AURORA: En. PRBS, En. Lanes, CCWait, CCSend	0,0001,25,3

**Table 23:** Global configuration register main table. (\*) indicates further details are given following the table. Section references are given in (). The Size column may list multiple fields by their size in bits. The rightmost value is always the field starting at register bit [0]. This 1,3,2 would indicate 3 fields: [5],[4:2],[1:0].

<b>Addr.</b>	<b>Name</b>	<b>Size (bits)</b>	<b>Description</b>	<b>ATLAS Default</b>
I 82	AURORA_CB_CONFIG1	8	Aurora Chann. Bonding Wait [19:12]	255
I 83	AURORA_CB_CONFIG0	12,4	Aurora Chann. Bond Wait [11:0], CBSend	4095,0
I 84	AURORA_INIT_WAIT	11	Aurora Initialization Delay	32
I 85	OUTPUT_PAD_CONFIG	4,1,1,4,3	GP_CMOS: pattern, En, DS, GP_LVDS: Enables, strength	5,1,0,15,7
I 86	GP_CMOS_ROUTE	6	GP_CMOS MUX select (15.6)	34
I 87	GP_LVDS_ROUTE_1	6,6	GP_LVDS(3), GP_LVDS(2) MUX select (15.6)	35,33
I 88	GP_LVDS_ROUTE_0	6,6	GP_LVDS(1), GP_LVDS(0) MUX select (15.6)	1,0
I 89	DAC_CP_CDR	10	CDR CP Bias (values <15 are set to 15)	40
I 90	DAC_CP_FD_CDR	10	CDR FD CP bias (values <100 are set to 100)	400
I 91	DAC_CP_BUFF_CDR	10	CDR unity gain buffer bias	200
I 92	DAC_VCO_CDR	10	CDR VCO bias (values <700 are set to 700)	1023
I 93	DAC_VCOBUFF_CDR	10	CDR VCO buffer bias (values <200 are set to 200)	500
I 94	SER_SEL_OUT	4×2	CML 3-0 content. 0=CK/2, 1=AURORA, 2=PRBS7, 3=0	1,1,1,1
I 95	CML_CONFIG	2,2,4	CML out: Inv. Tap 2,1; En. Tap 2,1; En. Lane 3,2,1,0	0,0,0001
I 96	DAC_CML_BIAS_2	10	CML drivers tap 2 amplitude (pre-emph)	0
I 97	DAC_CML_BIAS_1	10	CML drivers tap 1 amplitude (pre-emph)	0
I 98	DAC_CML_BIAS_0	10	CML drivers tap 0 amplitude (main)	500
<b>Monitoring and Test</b>				
99	MonitorConfig	1,6,6	Monitor pin: En., I. MUX sel., V. MUX sel.	0,63,63
100	ErrWngMask	8×1	Error and Warning Message disable Mask	0
101	MON_SENS_SLDO	1,4,1,1,4,1	Tsense LDO: En.A, DEM, Bias, En.D, DEM, Bias (12.5)	0,0,0,0,0,0
102	MON_SENS_ACB	1,4,1	Tsense center: En., DEM, Bias (12.5)	0,0,0
103	MON_ADC	1,1,1,6	Vref for Rsense: bot., top.; Vref in; ADC trim bits	0,0,1,0
104	DAC_NTC	10	Current output DAC for the external NTC	100
M 105	HITOR_MASK_3	6×1	HitOR disable for core cols. 53:48	0
M 106	HITOR_MASK_2	16×1	HitOR disable for core cols. 47:32	0
M 107	HITOR_MASK_1	16×1	HitOR disable for core cols. 31:16	0
M 108	HITOR_MASK_0	16×1	HitOR disable for core cols. 15:0	0
I 109	AutoRead0	9	Auto-Read register address A for lane 0	137
I 110	AutoRead1	9	Auto-Read register address B for lane 0	133
I 111	AutoRead2	9	Auto-Read register address A for lane 1	121
I 112	AutoRead3	9	Auto-Read register address B for lane 1	122
I 113	AutoRead4	9	Auto-Read register address A for lane 2	124
I 114	AutoRead5	9	Auto-Read register address B for lane 2	127
I 115	AutoRead6	9	Auto-Read register address A for lane 3	126
I 116	AutoRead7	9	Auto-Read register address B for lane 3	125
* 117	RingOscConfig	15×1	Ring oscillator enable bits (13.6)	1,5×0,1,8×0
118	RingOscRoute	3,6	Select which RO to read from block A, B (13.6)	0,0
119	RING_OSC_A_OUT	16	Ring oscillator block A output (rd. only) (13.6)	n/a
120	RING_OSC_B_OUT	16	Ring oscillator block B output (rd. only) (13.6)	n/a
121	BCIDCnt	16	Bunch counter (rd. only)	n/a
122	TrigCnt	16	Received trigger counter (rd. only)	n/a
123	ReadTrigCnt	16	Received or internal ReadTrigger ctr (rd. only)	n/a
124	LockLossCnt	16	Channel Sync lost lock counter (rd. only)	n/a

**Table 23:** Global configuration register main table. (\*) indicates further details are given following the table. Section references are given in (). The Size column may list multiple fields by their size in bits. The rightmost value is always the field starting at register bit [0]. This 1,3,2 would indicate 3 fields: [5],[4:2],[1:0].

Addr.	Name	Size (bits)	Description	ATLAS Default
125	BitFlipWngCnt	16	Bit Flip Warning counter (rd. only)	n/a
126	BitFlipErrCnt	16	Bit Flip Error counter (rd. only)	n/a
127	CmdErrCnt	16	Command Decoder error message ctr (rd. only)	n/a
128	RdWrFifoErrorCount	16	Writes and Reads when fifo was full ctr (rd. only)	n/a
129	AI_REGION_ROW	9	Auto Increment current row value (rd. only)	n/a
130	HitOr_3_Cnt	16	HitOr_3 Counter (rd. only)	n/a
131	HitOr_2_Cnt	16	HitOr_2 Counter (rd. only)	n/a
132	HitOr_1_Cnt	16	HitOr_1 Counter (rd. only)	n/a
133	HitOr_0_Cnt	16	HitOr_0 Counter (rd. only)	n/a
134	SkippedTriggerCnt	16	Skipped Trigger counter (rd. only)	n/a
135	EfusesReadData1	16	Readback of efuses 1 of 2 (Read Only)	n/a
136	EfusesReadData0	16	Readback of efuses 2 of 2 (Read Only)	n/a
* 137	MonitoringDataADC	12	ADC value (rd. only)	n/a
138-201	SEUxx_notmr (xx=0-63)	16	Dummies for SEU meas. Triple redundant	0
202-255	SEUxx (xx=0-53)	16	Dummies for SEU meas. Triple redundant	0

\* 3 **PIX\_MODE:** Broadcast mode causes the column pair value of Reg. 1 to be ignored and the entire row to be written with the same value. The selection of TDACs (5 bits) or Mask bits (3 bits) is valid when using the multiple mode of the write register command. In this mode only 10 bits are available in for two pixels in each write cycle, and they are assigned according to this flag. The bits 2300 that are not written will remain unchanged. Auto Row mode is described in Sec. 8.8.

\* 4,5 **PIX\_DEFAULT:** A special pair of registers that control the multiplexers for pixel configuration. When the correct value is programmed in both registers the programmed configuration (which must first be written) will be used. Until then the hard-wired default configuration will be active. Any single bit flip in the correct value of each register will still be interpreted as the correct value.

\* 6,7 **GCR\_DEFAULT:** A special pair of registers that control the multiplexers for global configuration. When the correct value is programmed in both registers the programmed configuration will be used. Until then the hard-wired default configuration will be active. Any single bit flip in the correct value of each register will still be interpreted as the correct value. In this case the default configuration will also be present in the registers soon after the chip clock is present, so the user 2310 need not program anything before switching (See 3).

\* 37 **LEAKAGE\_FEEDBACK:** Despite its address after Lin FE biases, it enables the Diff. FE leakage current compensation (bit 0) and a second feedback capacitance to reduce the gain (bit 1).

\* 43-46 **EnCoreColumnReset:** In the pixel matrix the clocks are gated unless there is hit activity. However, upon power up, the gating state is arbitrary and enabling the clock to the the columns can 2315 result in significant power consumption (up to double normal power). Therefore, a reset is provided to that columns can be reset to their full clock gated state as they have their clock enabled. The EnCoreColumnReset registers gate the reset signals to the columns, so that only a limited number

of columns can be reset at the same time, to avoid a large current spike. It is recommended to set enable EnCoreCol and EnCoreColumnReset equal to each other, and at start of operation cycle 2320 though selecting a small group of core columns at a time and issuing a reset signal for each group. The reset signal is issued by the Clear command (Sec. 8.2.1). After this initialization all core columns can be enabled.

- \* **51 ReadTriggerConfig:** The column read delay is the number of BX clocks allowed for the read token to propagate and data valid to appear at the bottom of column. The default (0) allows two 2325 clocks. This should normally not be changed but gives the option to allow more time in case of slower than expected propagation, especially after irradiation. The read trigger decision time is only relevant in 2-trigger mode. It is effectively the L0 to L1 latency. It is the number of bunch clocks allowed after a L0 trigger before readout or clear action is taken. A read trigger command must arrive before this delay for a readout action to be taken (See 9).
- \* **53 CalibrationConfig:** Injection can be either analog or digital, selected by bit [7]. Bit [6] selects the mode of analog injection, which can be regular (same for every pixel, default) or alternating (see Sec. 6)
- \* **57 MEAS\_CAP:** Contains the enable bits for the capmeasure circuit (see Sec. 13.8), as well as the VCAL range bit (see Sec. 6). If VCAL range is zero the injection voltage full scale is from 0 2335 to Vref\_ADC/2, if 1 it is from 0V to Vref\_ADC. Note Vref\_ADC is used as a reference for the calibration injection voltage as well as the Generic ADC.
- \* **117 RingOscConfig:** Enable bits for: Ring Osc. B Clear, Ring Osc. B Enable BL, Ring Osc. B Enable BR, Ring Osc. B Enable CAPA, Ring Osc. B Enable FF, Ring Osc. B Enable LVT, Ring Osc. A Clear, Ring Osc. A Enable[7:0].
- \* **137 MonitoringDataADC:** This value is updated every time the ADC performs a conversion, which is triggered by a global pulse. Reading the register multiple times between ADC conversions will return the same value. Reading the register does not trigger a conversion.

### 15.3 Global Pulse Routing

Any number of channels can be selected simultaneously.

Bit	Route to:	Bit	Route to:
0	Reset Channel Synchronizer	8	Reset Efuses
1	Reset Command Decoder	9	Reset Trigger Table
2	Reset Global Configuration	10	Reset BCID counter
3	Reset Service Data	11	Send Cal Reset pulse
4	Reset Aurora	12	ADC StartOfConversion
5	Reset Serializers	13	Send Start Signal to Ring Oscillators block A
6	Reset ADC	14	Send Start Signal to Ring Oscillators block B
7	Reset Data Merging	15	Send Start Signal to EfusesProgrammer

**Table 24:** Global pulse routing choices.

### 2345 15.4 Pixel Configuration

Bits	Name	Description
[0]	Enable	Include the pixel in the DAQ data path
[1]	Cal Enable	Turn on charge injection (*)
[2]	HitOr Enable	Add the pixel to its wired OR core col. hit line
[3:6]	TDAC value	Value for in-pixel threshold trim DAC
[7]	TDAC sign	Selects differential branch set to TDAC value

**Table 25:** ATLAS pixel configurations bits. (\*) Whether charge injection is digital or analog is controlled by global configuration register CalibrationConfig. The 5-bit pixel TDAC is made up of a 4-bit value and a sign bit.

## 15.5 IMUX and VMUX selection values

Setting	Selected Input	Setting	Selected Input	Setting	Selected Input
0	IREF main ref. current	11	Capmeasure parasitic	22	DIFF FE Preamp Top-Left
1	CDR VCO main bias	12	DIFF FE Preamp Main array	23	DIFF FE VTH1 Right
2	CDR VCO buffer bias	13	DIFF FE PreComp	24	DIFF FE Preamp Top
3	CDR CP current	14	DIFF FE Comparator	25	DIFF FE Preamp Top-Right
4	CDR FD current	15	DIFF FE VTH2	26	not used
5	CDR buffer bias	16	DIFF FE VTH1 Main array	27	not used
6	CML driver tap 2 bias	17	DIFF FE LCC	28	Ana. input current/21000
7	CML driver tap 1 bias	18	DIFF FE Feedback	29	Ana. shunt current/21600
8	CML driver main bias	19	DIFF FE Preamp Left	30	Dig. input current/21000
9	NTC_pad current	20	DIFF FE VTH1 Left	31	Dig. shunt current/21600
10	Capmeasure circuit	21	DIFF FE Preamp Right	32-62	not used
				63	high Z

**Table 26:** Current multiplexer (I\_mux) assignments for ATLAS chip.

Setting	Selected Input	Setting	Selected Input	Setting	Selected Input
0	Vref_ADC (GADC)	10	DIFF FE VTH1 Main array	31	Vref_CORE
1	I_mux pad voltage	11	DIFF FE VTH1 Left	32	Vref_PRE
2	NTC_pad voltage	12	DIFF FE VTH1 Right	33	VINA/4
3	VCAL_DAC/2 (Sec. 6.3)	13	RADSENS Ana. SLDO	34	VDDA/2
4	VDDA/2 from capmeasure	14	TEMPSENS Ana. SLDO	35	VrefA
5	Poly TEMPSENS top	15	RADSENS Dig. SLDO	36	VOFS/4
6	Poly TEMPSENS bottom	16	TEMPSENS Dig. SLDO	37	VIND/4
7	VCAL_HI	17	RADSENS center	38	VDDD/2
8	VCAL_MED	18	TEMPSENS center	39	VrefD
9	DIFF FE VTH2	19-30	Ana. GND	40-62	not used
				63	high Z

**Table 27:** Voltage multiplexer (V\_mux) assignments for ATLAS chip.

## 15.6 General Purpose LVDS and CMOS Output Assignments

The same selection codes apply to all outputs. Each output has a dedicated multiplexer and there is no issue selecting the same signal for multiple outputs.

Val	Selected signal	Val	Selected signal
0	Serial CMD input straight from differential receiver	21	Not used (see Sec. C)
1	Recovered serial CMD from CDR, before any delay	22	CDR/PLL mon_up_cdr signal
2	Recovered serial CMD from CDR, after phase delay	23	CDR/PLL mon_dn_cdr signal
3	Recovered 160 MHz clock before any delay	24	CDR/PLL mon_up_fd_cdr signal
4	Recovered 160 MHz clock after phase delay	25	CDR/PLL mon_dn_fd_cdr signal
5	BX Clock to Pixel Matrix	26	
6	Clock used for Precision ToT (640 MHz)	27	Serial stream from DATA_IN1 diff. Rcvr.
7	Pattern bit [3] of OUTPUT_PAD_CONFIG reg.85	28	Serial stream from DATA_IN2 diff. Rcvr.
8	Pattern bit [2] of OUTPUT_PAD_CONFIG reg.85	29	Serial stream from DATA_IN3 diff. Rcvr.
9	Pattern bit [1] of OUTPUT_PAD_CONFIG reg.85	30	unused
10	Pattern bit [0] of OUTPUT_PAD_CONFIG reg.85	31	unused
11	CalEdge as produced by the CMD Decoder	32	unused
12	CalAux as produced by the CMD Decoder	33	Power On Reset out (not used internally)
13	GlobalPulse as produced by the CMD Decoder	34	Channel Synchronizer Lock signal
14	Trigger pulses as produced by the CMD Decoder	35	PLL Lock signal
15	Read_Trigger pulses from the CMD Decoder	36	Activity detector on CMD input (Sec. 3.2.2)
16	HitOr [3]	37	Activity detector on CMD input (Sec. 3.2.2)
17	HitOr [2]	38	Goes low when low power mode is activated
18	HitOr [1]	39	GADC end of conversion signal
19	HitOr [0]		
20	HitOr logic result from self trigger block		

**Table 28:** Signal selection for GP\_LVDS and GP\_CMOS outputs.

2350 15.7 Internal and External Component Nominal Values

Component	Type	Value	Tolerance	Reference
Injection capacitor	MOM	8.02 fF	$\pm 10\%$	Fig. 31
Diff FE feedback cap (high gain)	parasitic	3.73 fF	n/a	Fig. 25
Diff FE feedback cap (low gain)	parasitic	3.28 fF	n/a	Fig. 25
Resistive temp. sensor	p <sup>+</sup> Poly w/silicide	10 K	$\pm 30\%$	Sec. 12.4
Resistive temp. reference	p <sup>+</sup> Poly w/o silicide	16 K	$\pm 12.5\%$	Sec. 12.4
Internal receiver terminations	Poly	100 $\Omega$	$\pm 12.5\%$	Sec. 8.1
Chip ID pull up resistors	Poly	40 k	$\pm 12.5\%$	to VDDD
IREF trim pull up resistors	Poly	40 k	$\pm 12.5\%$	to VDD_PRE

**Table 29:** Internal passive component types and values.

Component	Function	Value	Reference	Notes
R <sub>Iref</sub>	Current reference resistor	24.9 K	Fig. 15	$V_{BGR} = 490$ mV makes Iref = 20 $\mu$ A
R <sub>VrefA</sub>	Sets analog regulator ref. V.	30 K	Fig. 15	V <sub>refA</sub> = 0.6 V baseline
R <sub>VrefD</sub>	Sets analog regulator ref. V.	30 K	Fig. 15	V <sub>refD</sub> = 0.6 V baseline
R <sub>shuntA</sub>	Sets shunt slope	400 $\Omega$	Figs. 13, 14	$V_{IN}=1.4$ V at 1 A
R <sub>shuntD</sub>	Sets shunt slope	400 $\Omega$	Figs. 13, 14	$V_{IN}=1.4$ V at 1 A
R <sub>VOFS1</sub>	Sets the shunt offset V.	24.9 K	Figs. 13, 15	VOFS/2 = 0.5 V baseline
R <sub>VOFS2</sub>	Sets the low power offset V.	10 K	Fig. 15	VOFS/2 = 0.7 V for low power
R <sub>IMUX</sub>	Converts monitoring I output to a voltage	4.99 K		
R <sub>VrefADC</sub>	Sets the ADC reference voltage	84.5 K		V <sub>REF_ADC</sub> = 845 mV baseline
R <sub>CMD</sub>	termination resistor	100 $\Omega$		
C <sub>VDDA</sub>	Regulator output bypass	$\geq 0.55 \mu$ F	Fig. 8	4 instances (2.2 $\mu$ F total)
C <sub>VDDD</sub>	Regulator output bypass	$\geq 0.55 \mu$ F	Fig. 8	4 instances (2.2 $\mu$ F total)
C <sub>VPRE</sub>	Pre-regulator output bypass	0.1 $\mu$ F		
C <sub>VDD_PLL</sub>	Phase locked loop bypass	2.2 $\mu$ F		
C <sub>VDD_CML</sub>	Output driver bypass	2.2 $\mu$ F		2 instances
C <sub>VINA</sub>	Regulator input cap	$\geq 6.8 \mu$ F	Fig. 8	TBD by SP chain operation
C <sub>VIND</sub>	Regulator input cap	$\geq 6.8 \mu$ F	Fig. 8	TBD by SP chain operation
C <sub>VrefA</sub>	Vref bypass	100 nF		
C <sub>VrefD</sub>	Vref bypass	100 nF		
C <sub>IREF</sub>	Iref bypass	100 nF		
C <sub>IMUX</sub>	filter for current mux	22 nF		only for wafer probing
C <sub>IVMUX</sub>	filter for voltage mux	22 nF		optional
C <sub>CMD</sub>	command A/C coupling	10-100 nF		2 instances
C <sub>LP_EN_AC</sub>	Low power mode enable A/C coupling	100 nF	Fig. 19	1 instance

**Table 30:** External component nominal values.

## 15.8 Command and Trigger Encoding

Command	Encoding		(T)ag, (A)ddress or (D)ata 5-bit content				
Sync	1000_0001	0111_1110					
PLLlock	1010_1010	1010_1010					
Trigger	ttt_tttt	Tag[0..53]					
Read_trigger	0110_1001	ID<4:0>	00,T<7:5>	T<4:0>			
Clear	0101_1010	ID<4:0>					
Global Pulse	0101_1100	ID<4:0>					
Cal	0110_0011	ID<4:0>	D<19:15>	D<14:10>	D<9:5>	D<4:0>	
WrReg(0)	0110_0110	ID<4:0>	0,A<8:5>	A<4:0>	D<15:11>	D<10:6>	D<5:1> D<0>,0000
WrReg(1)	0110_0110	ID<4:0>	1,xxxx	xxxxx	N×(D<9:5>)	D<4:0>)	
RdReg	0110_0101	ID<4:0>	0,A<8:5>	A<4:0>			

**Table 31:** This is a duplicate of Table 5. List of protocol commands/frames and address or data fields associated with each. Unused padding bits are indicated by “0”. Double vertical lines denote frame boundaries. ttt\_tttt is one of 15 trigger commands (Table 6). The before-encoded bit content of chip ID, Address or Data is shown. These are all encoded as 8-bit data symbols (Table 32).

Symbol Name	Encoding	Data Value	Symbol Name	Encoding	Data Value
Data_00	0110_1010	5'b00000	Data_16	1010_0110	5'b10000
Data_01	0110_1100	5'b00001	Data_17	1010_1001	5'b10001
Data_02	0111_0001	5'b00010	Data_18	<i>0101_1001</i>	5'b10010
Data_03	0111_0010	5'b00011	Data_19	1010_1100	5'b10011
Data_04	0111_0100	5'b00100	Data_20	1011_0001	5'b10100
Data_05	1000_1011	5'b00101	Data_21	1011_0010	5'b10101
Data_06	1000_1101	5'b00110	Data_22	1011_0100	5'b10110
Data_07	1000_1110	5'b00111	Data_23	1100_0011	5'b10111
Data_08	1001_0011	5'b01000	Data_24	1100_0101	5'b11000
Data_09	1001_0101	5'b01001	Data_25	1100_0110	5'b11001
Data_10	1001_0110	5'b01010	Data_26	1100_1001	5'b11010
Data_11	1001_1001	5'b01011	Data_27	1100_1010	5'b11011
Data_12	1001_1010	5'b01100	Data_28	1100_1100	5'b11100
Data_13	1001_1100	5'b01101	Data_29	1101_0001	5'b11101
Data_14	1010_0011	5'b01110	Data_30	1101_0010	5'b11110
Data_15	1010_0101	5'b01111	Data_31	1101_0100	5'b11111

**Table 32:** List of command symbols used to encode data values. All symbols are the same as in RD53A except for Data\_18, which is shown in italics. The RD53A Data\_18 symbol is now the PLLlock command.

Symbol Name	Encoding	Trigger Pattern	Symbol Name	Encoding	Trigger Pattern
Trigger_01	0010_1011	000T	Trigger_09	0011_1100	T00T
Trigger_02	0010_1101	00T0	Trigger_10	0100_1011	TOT0
Trigger_03	0010_1110	00TT	Trigger_11	0100_1101	TOTT
Trigger_04	0011_0011	0T00	Trigger_12	0100_1110	TT00
Trigger_05	0011_0101	0T0T	Trigger_13	0101_0011	TTOT
Trigger_06	0011_0110	0TT0	Trigger_14	0101_0101	TTTO
Trigger_07	0011_1001	0TTT	Trigger_15	0101_0110	TTTT

**Table 33:** This is a duplicate of Table 6. List of trigger symbols used to encode the 15 possible trigger patterns spanning four bunch crossings. Note there is no 0000 pattern as that is the absence of an trigger. The Trigger\_01 (000T) means that the first bunch crossing of the trigger window is meant to be readout, and the extended tag returned will have 00 following the supplied tag base.

Tag base	Extended tag range	Symbol code	Symbol name	Tag base	Extended tag range	Symbol code	Symbol name
0	0-3	0110_1010	Data_00	27	108-111	1100_1010	Data_27
1	4-7	0110_1100	Data_01	28	112-115	1100_1100	Data_28
2	8-11	0111_0001	Data_02	29	116-119	1101_0001	Data_29
3	12-15	0111_0010	Data_03	30	120-123	1101_0010	Data_30
4	16-19	0111_0100	Data_04	31	124-127	1101_0100	Data_31
5	20-23	1000_1011	Data_05	32	128-131	0110_0011	Cal
6	24-27	1000_1101	Data_06	33	132-135	0101_1010	Clear
7	28-31	1000_1110	Data_07	34	136-139	0101_1100	GlobalPulse
8	32-35	1001_0011	Data_08	35	140-143	1010_1010	PllLock
9	36-39	1001_0101	Data_09	36	144-147	0110_0101	ReadReg
10	40-43	1001_0110	Data_10	37	148-151	0110_1001	ReadTrigger
11	44-47	1001_1001	Data_11	38	152-155	0010_1011	Trigger_01
12	48-51	1001_1010	Data_12	39	156-159	0010_1101	Trigger_02
13	52-55	1001_1100	Data_13	40	160-163	0010_1110	Trigger_03
14	56-59	1010_0011	Data_14	41	164-167	0011_0011	Trigger_04
15	60-63	1010_0101	Data_15	42	168-171	0011_0101	Trigger_05
16	64-67	1010_0110	Data_16	43	172-175	0011_0110	Trigger_06
17	68-71	1010_1001	Data_17	44	176-179	0011_1001	Trigger_07
18	72-75	0101_1001	Data_18	45	180-183	0011_1010	Trigger_08
19	76-79	1010_1100	Data_19	46	184-187	0011_1100	Trigger_09
20	80-83	1011_0001	Data_20	47	188-191	0100_1011	Trigger_10
21	84-87	1011_0010	Data_21	48	192-195	0100_1101	Trigger_11
22	88-91	1011_0100	Data_22	49	196-199	0100_1110	Trigger_12
23	92-95	1100_0011	Data_23	50	200-203	0101_0011	Trigger_13
24	96-99	1100_0101	Data_24	51	204-207	0101_0101	Trigger_14
25	100-103	1100_0110	Data_25	52	208-211	0101_0110	Trigger_15
26	104-107	1100_1001	Data_26	53	212-215	0110_0110	WrReg

**Table 34:** Tag base codes. All 8-bit symbols are re-used to provide the maximum number of tag bases.

## 15.9 Output Data Encoding

Tag values (decimal)	Meaning
0-215	extended tags from trigger command
216-231	Self triggers
216-219 (*)	Single bit-flip detected in tag symbol of a trig. command
220-223 (*)	unrecognized tag symbol
224-255	spares - not used in RD53B ATLAS

**Table 35:** This is a copy of Table 10. Possible extended tag values and their meaning. (\*) The bit flip and unrecognized symbol values overlap the self-trigger tags in RD53B-ATLAS. This bug is corrected in future versions.

Bits	Tag range	Meaning
1	1-215	Tag base autoread on/off
2	216-231	Self trigger autoread on/off
3	232-235	Corrupted tag base autoread on/off
4	236-243	Tag still in use autoread on/off
5	251	Exception record autoread on/off
6	252	Exception record autoread on/off
7	253	column data truncation marker on/off
8	254	event data truncation marker on/off
9	255	End of event on/off

**Table 36:** This is a copy of Table 10. Configuration bits to control auto-read action for each tag type.

Field	Value	Followed by	Decided by
NS bit	0 1	anything Tag[8] OR Chip ID[2] then tag[8]	data configuration
Tag	0-251	Tag[11] OR ccol[6] OR orphan bits	data always data data config. and data
	251, 252	32 bits of metadata	
	253	Tag[11] OR ccol[6] OR orphan bits	
	254 255	Tag[11] OR orphan bits Tag[11] OR orphan bits	
ccol	1-55	islast[1] then isneighbor[1]	always
isneighbor	0 1	crow[8] compressed map[var] OR map[16]	always configuration
crow	0-193	compressed map[var] OR map[16]	configuration
map	any	ToT[N <sub>hits</sub> × 4] OR lines below	configuration
ToT (islast)	any (0) any (1)	crow[6] Tag[11] OR ccol[6] OR orphan bits	always data

**Table 37:** Stream decoding reference. Each stream begins with an NS bit set to 1. The fields following the NS bit are identified in this table.

Register	Bits	Default	Meaning
DataMerging	4	1	Add 2 LS bits of chip ID after stream NS bit
DataConcentratorConf	0	0	Enable BCID output (not implemented in RD53B_ATLAS)
DataConcentratorConf	1	0	Enable L1ID output (not implemented in RD53B_ATLAS)
DataConcentratorConf	2	0	Enable end of stream marker (binary 000000)
DataConcentratorConf	3-10	15	number of events per stream - 1 (0 means 1 event/stream)
CoreColEncoderConf	0	0	Drop ToT values from output (output hit map only)
CoreColEncoderConf	1	0	Drop hit map compression (always 16-bit hit maps)
CoreColEncoderConf	2	0	Truncate too many hits in a column
CoreColEncoderConf	3-6	0	Truncation threshold exponent E. Hits in excess of $2^E$ will be removed
CoreColEncoderConf	7	0	Remove low ToT isolated hits
CoreColEncoderConf	8-11	0	Remove isolated hits below this ToT value
EvenMask	0-15	0	Isolation mask for even columns
OddMask	0-15	0	Isolation mask for odd columns

**Table 38:** Configuration bits to control output data format options including those in Table 37. See also Table 36. These options must be known in order to be able to decode the data stream.

## 15.10 ToT Tables

Output 4-bit code	True ToT bin (low edge) [BX]			
	40 MHz speed		80 MHz speed	
	4-bit (DEF)	6-to-4 bit	4-bit	6-to-4 bit
0	0	0	0	0
1	1	1	0.5	0.5
2	2	2	1	1
3	3	3	1.5	1.5
4	4	4	2	2
5	5	5	2.5	2.5
6	6	6	3	3
7	7	7	3.5	3.5
8	8	8	4	4
9	9	11	4.5	5.5
10	10	15	5	7.5
11	11	19	5.5	9.5
12	12	23	6	11.5
13	13	27	6.5	13.5
14	$\geq 14$	$\geq 31$	$\geq 7$	$\geq 15.5$

**Table 39:** This is a copy of Table 4. True ToT value in bunch crossing (BX = 25 ns units) for each output ToT 4-bit code, depending on speed (40 or 80 MHz) and compression (4 bit or 6-to-4 bit) settings. Always the low edge of the true ToT bin is shown. For example code 3 having a true ToT low edge of 3 means the true ToT was at least 3 bunch crossings and at most  $x$ , where  $x$  is the true ToT low edge of the next code (4 in this case). The last bin (code 14) has no high edge and includes all overflows. Code 15 means “no hit” and should never be seen because unhit pixels are internally suppressed.

counter value	Default ToT code (4-bit)	6-to-4-bit ToT code	BXs (40 MHz) ½ BXs (80 MHz)	counter value	Default ToT code (4-bit)	6-to-4-bit ToT code	BXs (40 MHz) ½ BXs (80 MHz)
0	0	0	$\leq 1$	17	14	10	18
1	1	1	2	18	14	10	19
2	2	2	3	19	14	10	20
3	3	3	4	20	14	11	21
4	4	4	5	21	14	11	22
5	5	5	6	22	14	11	23
6	6	6	7	23	14	11	24
7	7	7	8	24	14	12	25
8	8	8	9	25	14	12	26
9	9	8	10	26	14	12	27
10	10	8	11	27	14	12	28
11	11	8	12	28	14	13	29
12	12	9	13	29	14	13	30
13	13	9	14	30	14	13	31
14	14	9	15	31	14	13	32
15	14	9	16	32	14	14	$\geq 33$
16	14	10	17				

**Table 40:** Default (4-bit) and dual slope (6-to-4 bit) ToT count to ToT code mapping table. The first column shows the mere ToT count, the second column the 4-bit ToT code (read out in default mode), and the third column the 6-to-4 bit compressed value. The correspondent duration of the hit in number of clocks cycles (BXs) is reported in the last column. The readout ToT code goes from 0 to 14 in any mode. In 6-to-4 bit mode, the values in the third column are read out and the maximum value (14) is reached for hits with longer duration (i.e. higher charge).

### 15.11 Ring Oscillator Assignments

<b>ROSC Nbr.</b>	<b>Type</b>	<b>Len.</b>	<b>ROSC Nbr.</b>	<b>Type</b>	<b>Len.</b>
0	Strgth. 0 inv. clk. drvr.	55	4	Strgth. 0 4-input NAND	19
1	Strgth. 4 inv. clk. drvr.	51	5	Strgth. 4 4-input NAND	19
2	Strgth. 0 inverter	55	6	Strgth. 0 4-input NOR	19
3	Strgth. 4 inverter	51	7	Strgth. 4 4-input NOR	19

**Table 41:** Bank A ring oscillator types and lengths (in number of gates). The given lengths result in a typical frequency of about 600 MHz before irradiation. Each oscillator has its own Enable bit.

<b>ROSC Nbrs.</b>	<b>Type</b>	<b>Eff. Len.</b>	<b>Group</b>
0 & 1	Strgth. 0 inv. clk. driver	38.2	B-left, B-right
2 & 3	Strgth. 4 inv. clk. driver	44.5	B-left, B-right
4 & 5	Strgth. 0 inverter	38.1	B-left, B-right
6 & 7	Strgth. 4 inverter	44.3	B-left, B-right
8 & 9	Strgth. 0 4-input NAND	12.6	B-left, B-right
10 & 11	Strgth. 4 4-input NAND	16	B-left, B-right
12 & 13	Strgth. 0 4-input NOR	14.5	B-left, B-right
14 & 15	Strgth. 4 4-input NOR	14.5	B-left, B-right
16 & 17	Strgth. 0 scan D-flip-flop	6.1	FF
18 & 19	Strgth. 1 D-flip-flop	6.2	FF
20 & 21	Strgth. 1 Neg. edge D-flip-flop	5	FF
22	Strgth. 0 LVT inverter	40.6	LVT
23	Strgth. 4 LVT inverter	56	LVT
24	Strgth. 0 LVT 4-input NAND	16.5	LVT
25	Strgth. 4 LVT 4-input NAND	22.8	LVT
26-33	Strgth. 4 inj-cap-loaded 4-input NAND	16.8	CAPA

**Table 42:** Bank B ring oscillator types and lengths (in number of equivalent gates). The lengths given result in a typical frequency of about 800 MHz before irradiation. The oscillators are connected in separately enabled groups (there are no individual enable bits for each oscillator)

2355 **A. Aurora 64b66b Technical Reference**

## B. Design for Test (DFT)

The RD53B has two main digital domains: Chip Bottom and Pixel Matrix. The Chip Bottom controls all functions of the circuit and any fault in it will affect the whole chip. Therefore, maximum test coverage is desired for the chip bottom. To achieve this, it has been synthesized with  
2360 automatically inserted DFT features in the full logic code.

The Pixel Matrix on the other hand is a regular array of identical cores connected to each other by abutting. Since the core logic density is close to 100%, insertion of DFT logic for maximum test coverage was not possible. Luckily, faults in a single core do not force the rejection of the chip,  
2365 since chips having few faulty pixels are accepted as good. Therefore, limited structural testing of the cores is acceptable.

A scan chain is an industrial standard technique used in design for testing. The objective is to make testing easier by providing a simple way to set and observe every Flip-Flop in the Integrated Circuit. The basic element for this methodology is the Scan Flip-Flop, which features a multiplexer at the input selecting between the functional input D or the Scan-In (SI). The selection between D  
2370 and SI is governed by the Scan Enable (SE) signal. Using this basic Scan Flip-Flop as the building block, all the flops in the circuit are connected via the SI inputs to form a shift register. The first flop of the scan chain is connected to the scan-in port and the last flop is connected to the scan-out port. Scan chain operation involves three stages: Scan-in, Scan-capture and Scan-out. Scan-in involves shifting in and loading all the flip-flops with an input vector. During scan-in, with Scan Enable high,  
2375 the data is shifted from the output of one flop to the SI of the next. Once the sequence is loaded, Scan Enable is put low, one clock pulse (the capture pulse) is allowed to excite the combinatorial logic block and the output is captured by the Flip-Flops of the chain. The data is then shifted out (Scan Enable high) and the signature is compared with the expected signature. This is repeated for thousands of test vectors.

### 2380 B.1 Test signals

The chip has wire bond pads devoted to the production testing, as well as pads that change their function when the chip is put in test mode, as shown in Table 43 (see also full pinout in Sec. 15.1).

When the input pin TEST\_MODE is high, all pins in the table switch to the alternate function, the LVDS drivers for the two GPO\_LVDS[1:0] outputs are forced to be enabled at the maximum  
2385 output current. In TEST\_MODE most flip-flop clock inputs are connected to the primary input pins TEST\_CLK40 and TEST\_CLK160. In detail:

- TEST\_CLK40 drives all flip-flop of clock domains clk40 (most of the logic running at 40 MHz) and clk160\_merge (Data merger running at 160 MHz).
- TEST\_CLK160 drives all flip-flop of clock domains clk160 (Channel logic running at 160 MHz)  
2390 and clk64 (Aurora logic running at 64 MHz)

This flip-flop partitioning helps to avoid issues related to the different clock distribution delays from test clock input pins and flip-flop clock pins that can lead to hold violation during capture phase of the test.

Pin name	Normal function	Function in test mode
TEST_MODE_PAD	Connected to GND	TEST_MODE
CHIP_ID0_OR_PIXEL_MATRIX_TEST_MODE_PAD	CHIP_ID0	PIXEL_MATRIX_TEST_MODE
DATA_IN0_OR_SCAN_IN0_P_PAD	DATA_IN0	SCAN_IN0
DATA_IN0_OR_SCAN_IN0_N_PAD		
DATA_IN1_OR_SCAN_IN1_P_PAD	DATA_IN1	SCAN_IN1
DATA_IN1_OR_SCAN_IN1_N_PAD		
DATA_IN2_OR_TEST_CLK40_P_PAD	DATA_IN2	TEST_CLK40
DATA_IN2_OR_TEST_CLK40_N_PAD		
DATA_IN3_OR_TEST_CLK160_P_PAD	DATA_IN3	TEST_CLK160
DATA_IN3_OR_TEST_CLK160_N_PAD		
GPO_LVDS0_OR_SCAN_OUT0_P_PAD	GPO_LVDS0	SCAN_OUT0
GPO_LVDS0_OR_SCAN_OUT0_N_PAD		
GPO_LVDS1_OR_SCAN_OUT1_P_PAD	GPO_LVDS1	SCAN_OUT1
GPO_LVDS1_OR_SCAN_OUT1_N_PAD		
SCAN_EN_PAD	-	SCAN_EN

**Table 43:** Wire bond pads used in logic testing

## B.2 Chip Bottom testing

2395 Almost all flip-flops in the Chip Bottom are scan flip-flop connected in two scan chains. The only flip-flop excluded from the scan chains are those in the Precision ToT (PTOT) blocks. That means that this block is not tested, which is acceptable as PTOT is not a critical common function, but a secondary data generator similar to the pixels.

2400 Testing of the chip bottom can then be performed applying a set of test vectors generated by an Automatic Test Pattern Generator (ATPG), resulting in a fault coverage of about 99.7%.

Test vectors (or test patterns) are saved in Standard Test Interface Language (STIL) Procedure File (SPF), that describe how to apply the test vectors and the test vectors itself (See IEEE 1450). The format is the standard input file for Automatic Test Equipment (ATE) and should be parsed to extract all useful information.

2405 Code blocks in the SPF file describe primary inputs and outputs, scan structure, waveforms and patterns. The I/O pins controlled and monitored are defined in the Signals block:

```
Signals {
    SCAN_IN_0" In { ScanIn; } "SCAN_IN_1" In { ScanIn; } "SCAN_EN" In; "BYPASS_MODE" In;
    "CHIP_ID1" In; "CHIP_ID2" In; "CHIP_ID3" In; "PMTM" In; "TEST_MODE" In; "TEST_CLK160_64" In;
2410 "TEST_CLK40" In; "SCAN_OUT_0" Out { ScanOut; } "SCAN_OUT_1" Out { ScanOut; }
}
```

Test clock waveforms and in/out timing are defined in the Timing block:

```
Timing {
    WaveformTable "_default_WFT_" {
        Period '50ns';
        Waveforms {
            "TEST_CLK160_64" { 0 { '0ns' D; } }
            "TEST_CLK160_64" { P { '0ns' D; '16ns' U; '40ns' D; } }
            "TEST_CLK160_64" { 1 { '0ns' U; } }
            "TEST_CLK160_64" { Z { '0ns' Z; } }
2420
}
```

```

    "TEST_CLK40" { 0 { '0ns' D; } }
    "TEST_CLK40" { P { '0ns' D; '16ns' U; '40ns' D; } }
    "TEST_CLK40" { 1 { '0ns' U; } }
    "TEST_CLK40" { Z { '0ns' Z; } }
2425   "_default_In_Timing_" { 0 { '0ns' D; } }
    "_default_In_Timing_" { 1 { '0ns' U; } }
    "_default_In_Timing_" { Z { '0ns' Z; } }
    "_default_In_Timing_" { N { '0ns' N; } }
    "_default_Out_Timing_" { X { '0ns' Z; } }
2430   "_default_Out_Timing_" { H { '0ns' Z; '14ns' H; } }
    "_default_Out_Timing_" { L { '0ns' Z; '14ns' L; } }
    "_default_Out_Timing_" { T { '0ns' Z; '14ns' T; } }
}
}
2435 }

```

According with definitions, all primary input signals are applied at time 0 ns of the ATE cycle, primary outputs are tested at 14 ns, while the test clocks rise at 16 ns and fall at 40 ns. The cycle period is 50 ns. The SPF defines also a set of procedures and a macro:

```

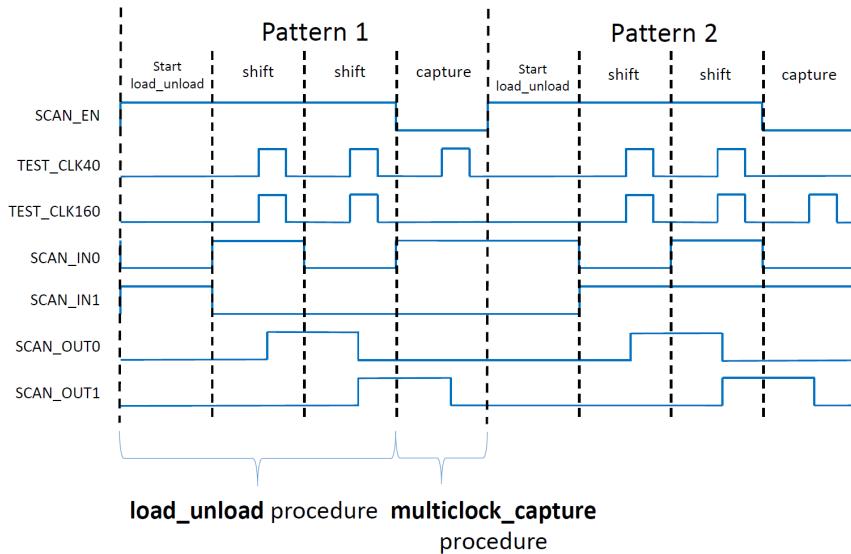
Procedures {
2440   "load_unload" {
      W "_default_WFT_";
      C { "SCAN_OUT_1"=X; "SCAN_OUT_0"=X; "SCAN_IN_1"=0; "SCAN_IN_0"=0; "TEST_CLK160_64"=0;
           "TEST_CLK40"=0; "SCAN_EN"=1; "PMTM"=0; "TEST_MODE"=1; }
      Shift {
          W "_default_WFT_";
2445     V { "_si"=###; "_so"=###; "TEST_CLK160_64"=P; "TEST_CLK40"=P; }
      }
    }
    "multiclock_capture" {
      W "_default_WFT_";
2450     C { "SCAN_EN"=0; "BYPASS_MODE"=0; "SCAN_OUT_1"=X; "SCAN_OUT_0"=X; "TEST_CLK40"=0;
           "SCAN_IN_1"=0; "CHIP_ID3"=0; "SCAN_IN_0"=0; "CHIP_ID2"=0; "CHIP_ID1"=0;
           "TEST_CLK160_64"=0; }
      F { "PMTM"=0; "TEST_MODE"=1; }
      V { "_pi"=\j #####;\n         "_po"=\j ##; }
2455   }
}

.....
}
2460 MacroDefs {
  "test_setup" {
    W "_default_WFT_";
    V { "PMTM"=0; "TEST_MODE"=1; "TEST_CLK160_64"=0; "TEST_CLK40"=0; }
2465   }
}

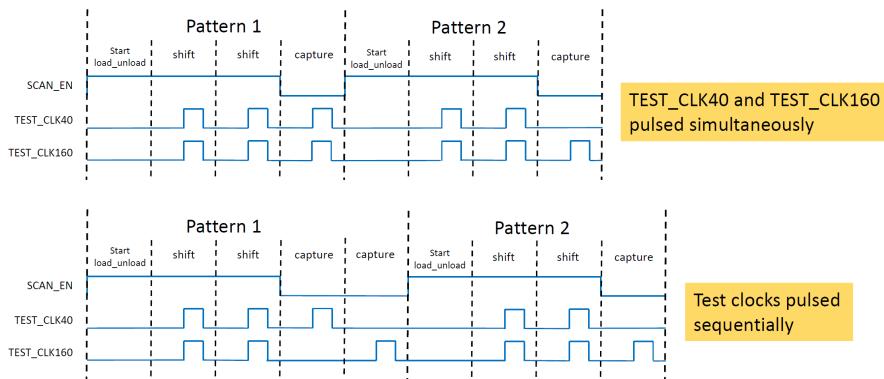
```

The `test_setup` macro is called at begin of the test to set the initial values of clock and Test select inputs. The two procedures `load_unload` and `multiclock_capture` are called at every test pattern. The `load_unload` puts the `SCAN_EN` to 1 and shifts the scan chains to load the new pattern, while unload the old one (SHIFT phase). After pattern loading, the `multiclock_capture` procedure is called to set `SCAN_EN` to 0 and primary inputs to the needed value and primary outputs are checked (CAPTURE phase). Input value passed as parameter to the procedure can include the “P” state that means: “Pulse the input”. Fig. 86 shows the state diagram of main

signals involved in test pattern shift and capture. Note that, while during shift phase all test clocks are always active, during capture phase each test clock can be active or not depending on the test pattern. Moreover the test program can require two or more consecutive capture phases as shown in Fig. 87. In case of two consecutive capture phases, the `multiclock_capture` procedure is called two times before the next `load_unload`.



**Figure 86:** State diagram of main signals involved in test pattern shift and capture

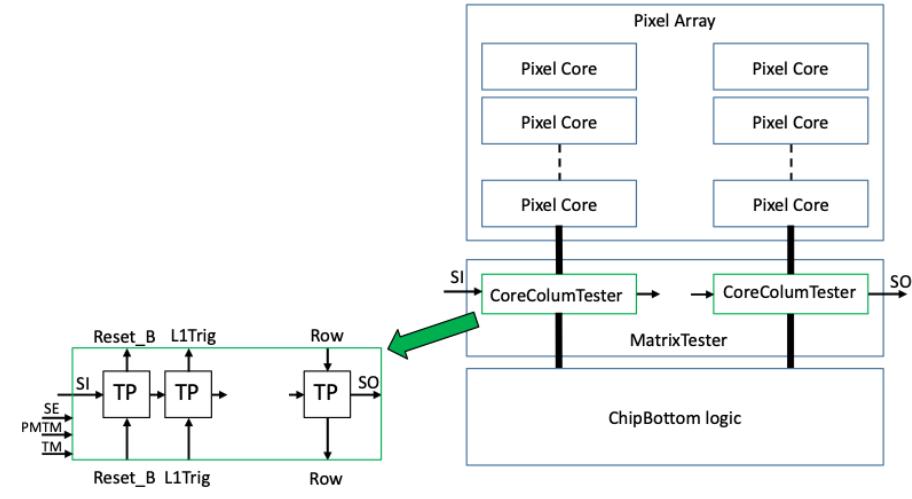


**Figure 87:** Example of consecutive capture phases

### B.3 Pixel Matrix test

Due to the high density in the pixel matrix region, flip-flops in this part of the chip are not scan flip-flop connected in chains. To perform the production testing a dedicated block has been inserted

between the matrix and the Chip Bottom called MatrixTester. As shown in Fig. 88, the Matrix tester is made by a chain of “CoreColumnTester” blocks, one for each column of cores.



**Figure 88:** Matrix tester block

When the primary input PIXEL\_MATRIX\_TEST\_MODE (PMTM) is set to high, the pixel array logic is disconnected from the chip bottom logic and connected to custom test points inside the CoreColumnTester. To speed up scan phase the primary I/O SCAN\_IN0 and SCAN\_OUT0 are connected MatrixTester SI and SO. Depending on the signal function, different type of test points has been inserted. The test point order and size is shown in Table 44.

#### B.4 Testpoint description

**ScanTestPointB2M** The default test point for configuration lines and synchronous signals. When SE is low, the state of the Flip Flop is hold.

**ScanTestPointM2B** The default test point for signals coming from pixel matrix. When SE is low, the Flip Flop samples the state of the line on the rising edge of the test clock.

**ScanTestPointReset** Test point for the asynchronous active low reset signal. The output to the matrix is hold high when SE is high. After scan phase, the output state is equal to the state of Flip Flop.

**ScanTestPointEnDigHit** Test point for the “Enable Digital Hit” signal. The output to the matrix is hold high when SE is high. After scan phase, the output state is equal to the state of Flip Flop.

**ScanCalEdge** Digital injection circuit. Internal Flip Flops can be loaded during scan phase (SE = 1) with the width of the pulse to be injected. During capture phase (SE=0) the output line is pulsed at the programmed time width. Pulse width ranges from 1 to 64 clock cycles in steps of half clock cycle.

Signal	Direction	TP type	Number of FF in scanchain
Reset_B	B2M	ScanTestPointReset	1
L1Trig	B2M	ScanTestPointB2M	1
TrigClear	B2M	ScanTestPointB2M	1
TrigId	B2M	ScanTestPointB2M	8
TrigIdReq	B2M	ScanTestPointB2M	8
LatCnt	B2M	ScanTestPointB2M	9
LatCntReq	B2M	ScanTestPointB2M	9
EnDigHit	B2M	ScanTestPointEnDigHit	1
AnaInjectionMode	B2M	ScanTestPointB2M	1
CalEdge	B2M	ScanCalEdge	7
CalAux	B2M	ScanTestPointB2M	1
ConfDefault	B2M	ScanTestPointB2M	1
ConfWr5bit	B2M	ScanTestPointConfWr	1
ConfWr3bit	B2M	ScanTestPointConfWr	1
ConfAddr	B2M	ScanTestPointB2M	12
ConfDataWr	B2M	ScanTestPointB2M	8
ConfDataRd	M2B	ScanTestPointM2B	8
TotDualEdgeCount	B2M	ScanTestPointB2M	1
Tot6to4Mapping	B2M	ScanTestPointB2M	1
Read	B2M	ScanTestPointB2M	1
Token	M2B	ScanTestPointM2B	1
HitOr	M2B	ScanTestPointM2B	4
Data	M2B	ScanTestPointM2B	16
Row	M2B	ScanTestPointM2B	6

**Table 44:** DFT test points in order, starting from SI input.

**ScanTestPointConfWr** Test point for configuration write pulse. The output to the matrix is hold low during scan phase (SE=1). During capture phase (SE=0), if the Flip Flop state is high, the test clock is propagated to the matrix.  
2505

## B.5 Matrix test procedure

The test of pixel matrix logic cannot be performed using a simple pass-fail criteria, since it is acceptable to have few pixels not working in a large front-end chip. The test procedure has to be dynamically changed to exclude defective pixels from test once they are detected. At the end the  
2510 pass-fail criteria should be applied based on the number and special distribution of faulty pixels.

A set of test procedures has been developed and simulates to evaluate the achieved fault coverage. The full set of tests is part of the RD53B design repository (file names tb\_matrix\_tester.sv and ChipBottomTV.sv) The set of tests tries to detect faults in the configuration registers and related logic, TOT counters, TOT registers, six to four encoder, TOT memories, Latency counter registers,  
2515 and trigger ID registers. It achieves a core fault coverage of about 90%. Additional tests can be developed if needed.

## C. Known Issues with RD53B ATLAS

This appendix collects information about bugs or strange features (quirks) in the RD53B ATLAS design.

- 2520 • Bug in 4-bit latch circuit. The storage of ToT values in the pixel regions uses a custom memory called 4-bit latch. This circuit has a design error that manifests in two ways. (1) High digital current consumption when inputs differ from stored values in latched state. Since the latch state is random upon power-up, this results in high digital current consumption upon power-up. The current can be reduced by storing a ToT Of 0 in all ToT memories via repeated digital injection. (2) Corruption of stored ToT values when multiple bits are 1. In particular, when 3 out of 4 bits are 1, it is very likely ( 90%) that the stored value will be corrupted to 1111, which is the code for “no hit”, and so will result in the hit being lost.
- 2525 • Triple redundant global registers 138-201. These registers were intended to measure the single latch SEU upset rate and, therefore, should not have been triple redundant. But they are triple redundant in the RD53B ATLAS chip.
- 2530 • Tags 216-224 are generated by the Self-Trigger and the same numbers are also generated by the command decoder in response to bit flips in the tag base symbols received with a trigger command. This should normally not be a problem since the self trigger will normally not be used in high dose rate irradiations causing bit flips.
- 2535 • One can only enable or disable the IsolatedHitRemoval feature globally. Per core column enables must still be added.
- Non-isolated hits sometimes removed. Very rarely (a few  $\times 10^{-5}$  level) some non-isolated hits are removed. This is related to a processing step when the end of column hit buffer is full and can only occur when hit removal is enabled.
- 2540 • Precision ToT encoding. The PToT output in the absence of hits is all zeros. However, the hit data encoder interprets binary 1111 as “no hit”, while binary 0000 is a valid ToT code. Therefore, the PToT data will not suppress zero hits. Whenever a PToT block fires (due to a HitOR pulse), the output will be equivalent to a fully occupied quarter core: a bitmap with 16 1’s (which when “compressed” takes 30 bits) plus 16 4-bit values (the ToT fields in normal data). So 94 bits in total. This is only if at least one of the four HitOr has fired. If the HitOr’s are disabled, then there will be no output.
- 2545 • LSB of trigger ID not triplicated. The LSB of the trigger ID internal value is not triplicated. This could lead to a minor SEU vulnerability.
- The ATLAS chip input differential receiver common mode is 260 mV, following SLVS standard, and additionally the receiver has some duty cycle distortion. These. For future versions (including CMS chip) the common mode will be raised to 600 mV to improve common mode noise rejection and the duty cycle distortion will be reduced.

- Missing self-trigger output in signal selection for GP\_LVDS and GP\_CMOS outputs. Selection value 21 in Table 28 should select the self trigger internal signal.
- Only HitOR 0 counter works. Counters 1-3 do not work due to a bug in the code. The HitOR signals themselves do work- only the counters are affected.

## References

- [1] RD53 Collaboration, “The RD53A Integrated Circuit,” CERN-RD53-PUB-17-001 (2017).
- [2] RD53 Collaboration, “RD53B Design Requirements,” CERN-RD53-PUB-19-001 (2019).
- 2560 [3] Xilinx, “Aurora 64B/66B Protocol Specification,” SP011 (v1.3) October 1, 2014.
- [4] M. Karagouins *et. al*, “An Integrated Shunt-LDO Regulator for Serial Powered Systems,” in Proc. of IEEE ESSCIRC ’09, (2009).
- [5] A. Uhr-Rehman *et. al*, “Performance simulations and characterization of RD53 pixel chips for ATLAS and CMS HL-LHC upgrades ,” in TWEPP 2021 (2021)  
2565 <https://indico.cern.ch/event/1019078/contributions/4443947/>.
- [6] J. Christiansen, “RD53B users guide: Introduction to RD53B pixel chip architecture, features and recommendations for use in pixel detector systems.” CERN-RD53-PUB-21-001 (2021)