Science & Technology Facilities Council

ASIC Design Group

CBC3.1 User Manual

Author: Mark Prydderch

Issue: 1.5.2

Published: 31/05/23

Science & Technology Facilities Council
Technology

Contents

Tables 5

Figures 6

Document Change Log 7

Introduction 8

Top Level Architecture 9

Analogue Front End 11

- Channel Gain 12
- Channel Noise 12
- Pulse shaping 12
- Post-amp Offset Adjustment 13
- Overload recovery 13
- 'Hit' Comparator Time-walk 13
- 'Hit' Comparator threshold trim 13

Digital Back End 13

- Delay Locked Loop (DLL) 13
- Channel Mask 14
- Hit Detect Logic 14
- Pipelined Data Logic: 17
  - OR254 17
  - Pipeline SRAM 17
  - Pipeline Control Logic 17
  - Buffer SRAM 19
  - Output Data Serialiser 19
  - L1 Counter 21
- Stub Data Logic: 21
  - Layer swapping logic 22
  - Stub-Finding Logic 22

Stub Gathering logic 26
Bend Look-up Table 28
Data Assembly Logic 29
Interfaces 32
Fast Control Interface 32
I²C Slow Control Interface 34
Hard Reset 37
320MHz Input Clock & Fast Command Interface 38
Fast Data Outputs 38
Inter-chip Hit Signals 39
Programmable e-Fuses 41
Chip ID 41
Bandgap Trimming 41
e-Fuse Programming 41
Test Features 42
Charge Pulse Generator 42
Analogue Bias Multiplexer 44
40MHz Clock Output 44
Nearest Neighbour IO Test Features 44
Powering 46
Operating Temperature 46
I²C Programmable Registers 46
Page Function 46
Register 0 — Comparator Settings, Beta Multiplier Reset Polarity, Trigger Latency & Page selection 47
Page1 — Register 1: Trigger Latency 48
Page1 — Register 2: Beta Multiplier & SLVS Output Pad settings 48
Page1 — Register 3: Preamp Input Branch Current (Ipre1) settings 49

Page1 — Register 4: Preamp Cascode Branch Current (Ipre2) settings 49
Page1 — Register 5: Preamp Source Follower Bias (Ipsf) settings 50
Page1 — Register 6: Postamp Bias Current (Ipa) settings 50
Page1 — Register 7: Postamp Offset Adjustment (Ipaos) settings 51
Page1 — Register 9: Comparator Bias Current (Icomp) setting 51
Page1 — Register 11: Postamp Reference Voltage (VPLUS1/VPLUS2) setting 52
Page1 — Register 12: HIP Suppression, Serialiser Handshake Retiming & SLVS Disable setting 52
Page1 — Register 13: Test Pulse Amplitude setting 53
Page1 — Register 14: Test Pulse Delay & Channel Group Selection 54
Page1 — Register 15: Test Pulse Control & Analogue Mux setting 55
Page1 — Register 16: CAL_I bias setting 56
Page1 — Register 17: CAL_Vcasc bias setting 56
Page1 — Register 18: Pipeline & Stub Logic input selection, & Pt Width setting 57
Page1 — Register 19: Correlation Window Offset setting for regions 3 & 4 58
Page1 — Register 20: Correlation Window Offset setting for regions 1 & 2 59
Page1 — Register 21: Bandgap Fuse Register setting 59
Page1 — Register 22: Chip ID eFuse Register setting (Part 1) 60
Page1 — Register 23: Chip ID eFuse Register setting (Part 2) 61
Page1 — Register 24: Chip ID eFuse Register setting (Part 3) 61
Page1 — Register 27: Layer Swap control & Cluster Width setting 62
Page1 — Register 28: 40MHz Clock Delay, Test Pulse Clock, 40MHz Clock Output and OR254 63
Page1 — Register 29: Fast Command Interface and Error Flags 64
Page1 — Register 32 to 62: Mask Channels 1-248 65
Page1 — Register 63: Mask Channels 249-254 65
Page1 — Register 64 to 78: Bend register<0:14> 66

Page1 — Register 79: VCTH (Threshold Voltage part 1) 67
Page1 — Register 80: VCTH (Threshold Voltage part 2) 67
Page1 — Registers 81 to 84: **TOP** Nearest Neighbour IO Test Registers 68
Page1 — Registers 85 to 88: **BOTTOM** Nearest Neighbour IO Test Registers 70
Page2 — Register 1 to 254: Channel 1-254 Offset 72
Page2 — Register 255: Dummy Channel Offset 72
Pads 73
Overview 73
Pad Allocation 74
Pad Definitions 76
Further Reading 81
NOTES 82

Tables

Table 1 — Fast Control Interface commands and their corresponding serial codes 32
Table 2 — Fast Control Interface commands and their functions 33
Table 3 — Register Address 0 default settings 47
Table 4 — Trigger Latency default settings 48
Table 5 — Beta Multiplier & SLVS Output Pad settings 48
Table 6 — Preamp input Branch Current (Ipre1) 49
Table 7 — Preamp Cascode Branch Current (Ipre2) 49
Table 8 — Preamp Source Follower Bias (Ipsf) 50
Table 9 — Postamp Bias Current (Ipa) 50
Table 10 — Postamp Offset Adjustment Bias Current (Ipaos) 51
Table 11 — Comparator Bias Current (Icomp) 51
Table 12 — Postamp Reference Voltages (Vplus1 & 2) 52
Table 13 — HIP Suppression options & SLVS output enable/disable 52
Table 14 — Test Pulse Amplitude setting 53
Table 15 — Test Pulse Delay & Channel Group Selection 54
Table 16 — Test Pulse Channel Selection 54
Table 17 — Test Pulse Control & Analogue Mux 55
Table 18 — Valid Analogue Multiplexer codes and their corresponding bias voltages. 55
Table 19 — CAL_I bias settings 56
Table 20 — CAL_Vcasc bias settings 56
Table 21 — Pipeline & Stub logic input selection & Pt width setting 57
Table 22 — Selection codes for the Pipeline & Stub Logic input choice 57
Table 23 — Correlation Window Offset setting for regions 3 & 4 58
Table 24 — Valid codes for Correlation Window Offset programming. 58
Table 25 — Correlation Window Offset setting for regions 1 & 2 59
Table 26 — Bandgap Fuse Register settings 59
Table 27 — Chip ID eFuse programming register (Least significant 8 bits) 60
Table 28 — Chip ID eFuse programming register (Next most significant 8 bits) 61
Table 29 — Chip ID eFuse programming register (Most significant 3 bits) 61
Table 30 — Layer Swap control & Cluster Width settings 62
Table 31 — Effect of Cluster Width settings 62
Table 32 — 40MHz Clock Domain options and OR254 test feature 63
Table 33 — Fast Command Interface and Error Flags 64
Table 34 — Channel Masking registers 65
Table 35 — The last Channel Masking register 65
Table 36 — Bend Lookup Table mapping 66
Table 37 — Comparator threshold setting (Part 1) 67
Table 38 — Comparator threshold setting (Part 2) 67
Table 39 — TOP Nearest Neighbour IO Test Input Register 81 68
Table 40 — TOP Nearest Neighbour IO Test Input Register 82 68
Table 41 — TOP Nearest Neighbour IO Test Output Register 83, and Test Mode Select 69
Table 42 — TOP Nearest Neighbour IO Test Output Register 84 69
Table 43 — BOTTOM Nearest Neighbour IO Test Input Register 85 70
Table 44 — BOTTOM Nearest Neighbour IO Test Input Register 86, and Test Mode Select 70
Table 45 — BOTTOM Nearest Neighbour IO Test Output Register 87 71
Table 46 — BOTTOM Nearest Neighbour IO Test Output Register 88 71
Table 47 — Channel Offset Adjustment 72
Table 48 — Dummy Channel Offset Adjustment 72
Table 49 — CBC3.1 Bump Pad definitions (part 1) 77
Table 50 — CBC3.1 Bump Pad definitions (part 2) 78
Table 51 — CBC3.1 Bump Pad definitions (part 3) 79
Table 52 — CBC3.1 Probe/Wire Bond Pad definitions 80

Figures

Figure 1 — Block Diagram of CBC3.1 architecture. 9
Figure 2 — CBC3.1 Analogue Front-End 12
Figure 3 — CBC3.1 pulse shape Figure 4 — Post-amp Offset Voltage setting 12
Figure 5 — Hit Detect block diagram. 15
Figure 6 — Examples of Hit Detect processing of different input signal scenarios. 15
Figure 7 — Timing constraints for the Fixed Pulse Width section of the Hit Detect circuit. 16
Figure 8 — Timing constraints for the 40MHz Sampled section of the Hit Detect circuit. ... 16
Figure 9 — Block diagram of the pipeline 17
Figure 10 — Block diagram of the Pipeline Control Logic 18
Figure 11 — Illustration of the serial *Triggered Data* packet. 20
Figure 12 — Illustration of the *Triggered Data* packet appearance following the 1st *Trigger* command after a Fast Reset command, for a programmed latency of 1 and Clock 40 delay settings 0 to 10. 20
Figure 13 — Illustration of the Triggered Data packet appearance following the 1st Trigger command after a Fast Reset command, for a programmed latency of 1 and Clock 40 delay settings 11 to 24. 21
Figure 14 — An illustration of two sensor layers wire-bonded to a module hybrid substrate on which CBC3.1s are bump-bonded. 22
Figure 15 — Illustration of a valid correlation between layers and a rejected Cluster. 23
Figure 16 — Example of Correlation Window & Offset Correction 24
Figure 17 — Definition of Bend Codes generated by the Stub-Finding Logic. 25
Figure 18 — Illustration of the Stub Gathering Logic 27
Figure 19 — An example of mapping the 5 bit bend information to 4 bits. 28
Figure 20 — Illustration of the Stub Data packet. 29
Figure 21 — An illustration of the serial Triggered Data packet in relationship to the Stub Data Packet. 30
Figure 22 — Illustrating the timing of the *Stub Data* output following a *Fast Reset* command, for low numbered DLL settings. 31
Figure 23 — Illustrating the timing of the *Stub Data* output following a *Fast Reset* command, for high numbered DLL settings. 31
Figure 24 — Illustration of the Fast Control Interface timing. 33
Figure 25 — I^2C Master/Slave Configuration 34
Figure 26 — I^2C Write Transaction 35
Figure 27 — I^2C Write and Read commands. 36
Figure 28 — Suggested timing for RESET. 37
Figure 29 — Map of circuits affected by the hard reset. 38
Figure 30 — SLVS I/O Format. 39
Figure 31 — Illustration of the inter-chip signals required for finding stubs. 40
Figure 32 — Illustration of the inter-chip input pad structure. 40
Figure 33 — Block diagram of the Charge Pulse Generator 42
Figure 34 — Test pulse timing 43
Figure 35— Test pulse capacitor arrangement 44
Figure 36 — (a) Test feature for outputs to neighbouring CBC3.1, (b) Test feature for inputs from neighbouring CBC3.1 45
Figure 37 — CBC3.1 Pad layout – Overview (pads facing upwards) 73
Figure 38 — CBC3 Pad layout – Top half of chip (pads facing upwards) 74
Figure 39 — CBC3.1 Pad layout – Bottom half of chip (pads facing upwards) 75
Figure 40 — CBC3.1 Pad layout – Pad Numbering (pads facing upwards) 76

Document Change Log

Change Log		Page
27/02/2018	Comment added to clarify the settling time requirement for the input serial command data relative to the falling edge of the 320MHz clock.	32
22/05/2018	**Major update to text, figures, etc. New version 1.2 issued.**	
	Added text regarding new control bits for adjusting the timing of handshaking control signals between the L1 Data Serialiser and the Buffer SRAM.	19
	Note added to L1 Counter description to emphasise the counter start condition. L1 Counter Reset renamed to Orbit Reset in keeping with Fast Command section.	20
	Text and figure added to describe the effect of a Fast Reset command on the L1 Data Packet.	20
	Text added to end of 1st paragraph of Stub Data section to clarify the latency between bunch crossings and their associated stub data.	28
	Text and figures added to describe the effect of a Fast Reset command on the Stub Data timing.	29/30
	Last sentence of last paragraph of e-Fuse programming section was changed to reflect the fact that the 3.3V supply is resistively shorted to the 1.2V supply to improve the stability of the e-Fuse circuit when not in use.	40
	Register 3 bit 6 default state changed to 0 to match circuit.	48
	Register 12 table modified to include Select_Clk40_DLL and Select_Clk40_Ref control bits. Explanatory text also added.	51
09/07/2018	**New version 1.3 issued.**	
	Output Data Serialiser section, end of 2nd paragraph – Clarification of the channel data order.	19
	Section on Further Reading added with list of publications.	80
18/07/2019	**New version 1.4 issued.**	
	Figure 12 Modified to show timing with low delay settings for the on-chip 40MHz Clock	20
	Figure 13 added to show timing with high delay settings for the on-chip 40MHz Clock	21
	Document figures renumbered to accommodate new figure 13. Page numbering also updated	All
	New section on Operating Temperature added with text to be added later.	46
	Tables 49, 50 & 51 updated to correct some incorrect pad numbering and labelling.	77-79
04/12/2019	Table 6 corrected to 00001010 to match chip	49
13/07/2020	Figures 38, 39 & 40 corrected to show Odd & Even pad numbering	74-76
13/07/2020	Text on Bit 3 (Layer Swap) corrected for ODD & EVEN channels	62
22/03/2023	Note regarding hit detect blind spot added	16
31/05/2023	Clarification of SLVS current settings	48

Introduction

The CBC3 is designed for the readout of the Silicon Micro-strip Tracker of the CMS experiment on the HL-LHC. The HL-LHC will accelerate two beams of charged particles in opposite directions around a circular particle accelerator based at CERN in Geneva. The particles are concentrated into bunches and the paths of these bunches cross at intervals around the ring. The CMS experiment is based at one such point and the tracker forms the inner region of the experiment – its purpose is to track charged particles produced in the bunch collisions. The inner region of the tracker comprises pixellated silicon sensors whilst the outer region comprises silicon strip sensors.

Manufactured on a 130nm CMOS technology, the CBC3.1 chip reads out the charge generated by ionising events within the silicon strips of the CMS detector. It converts these events into a 'hit' or 'no hit' binary value for each of the channels. For the purposes of this document, this data will be referred to as 'Triggered Data'. The ionising events are synchronous with the bunch crossing event interval of 25ns and the chip must store the data from each event, up to a maximum of 512 bunch crossing intervals (12.8μs for a 25ns clock), in order to allow time for the external system to decide which event data should be read out. This time is known as the trigger latency.

When the CBC3.1 receives a trigger signal, the Triggered Data from the event that was stored one trigger latency period earlier, is read out into a 32-event-deep data buffer where it is stored until it can be read out from the chip. The data buffer is required because data is read out serially from the chip at a clock rate of 320MHz. Reading 254 channels of data, the event address, plus some additional 'house-keeping' data bits, requires 950ns (including dead-time for clock domain crossing). At the same time the CBC3.1 must continue to take data every bunch crossing (25ns).

The average trigger rate is 750KHz (1.33μs). However, due to the random nature of the trigger, the data readout has been designed to cope with several triggers arriving at up to 1MHz. As long as the data from one event is read out quicker than 1μs then the data buffer should rarely fill up and lose events.

Unlike earlier versions of the CMS Binary Chip, the CBC3.1 includes logic for detecting potential high momentum tracks in each event. These are the events of particular scientific interest and the data will be used by the Level 1 Trigger to decide whether the Triggered Data for an event is to be read out. This logic looks for coincidence between hits on channels that are connected to different sensor layers and generates positional data. For the purposes of this document, this data will be referred to as 'Stub Data', in reference to the data being only a part of the potential track. In order for the data to be useful input to the Level 1 Trigger system, it must be output from the CBC3.1 with as little delay as possible. In order to do this, the amount of Stub Data must be restricted and several parallel digital outputs used at 320MHz. A maximum of three sets of Stub Data per event, per CBC3.1, was chosen as a compromise between the expected detection efficiency and the available output data bandwidth. No prioritization is given to the Stubs, other than where they physically occur in relation to the output of the data multiplexer. The data for the Stubs is shared across five differential SLVS output pads, along with some bits to indicate excess stubs and error flags, and a bit for synchronization of the CBC3.1 output data with the Concentrator ASIC.

During operation the HL-LHC the environment will be highly radiated with charged particles, requiring the electronics to be radiation tolerant, both for total dose and for Single Event Transients (SETs).

Top Level Architecture

The following block diagram illustrates the main circuit blocks that make up the CBC3.1 design. There are two clock domains on the ASIC, as indicated by the pink and green shaded areas. The first domain is a 320MHz clock domain that is used for all of the Input/Output interface activity, with the exception of the 1MHz I^2C-type slow-control interface that is used to initialise register settings for such things as comparator thresholds and amplifier biasing. A 320MHz clock is provided to the chip via a differential SLVS input.

► **NOTE:** There is no fine tuning of the 320MHz clock on the chip.

The second clock domain is a 40MHz domain, which is derived from the Fast Control Interface (FCI) that operates on the 320MHz domain. This 40MHz clock domain matches the 25ns repetition rate of the Bunch Crossings (BX), and is used extensively throughout the chip.

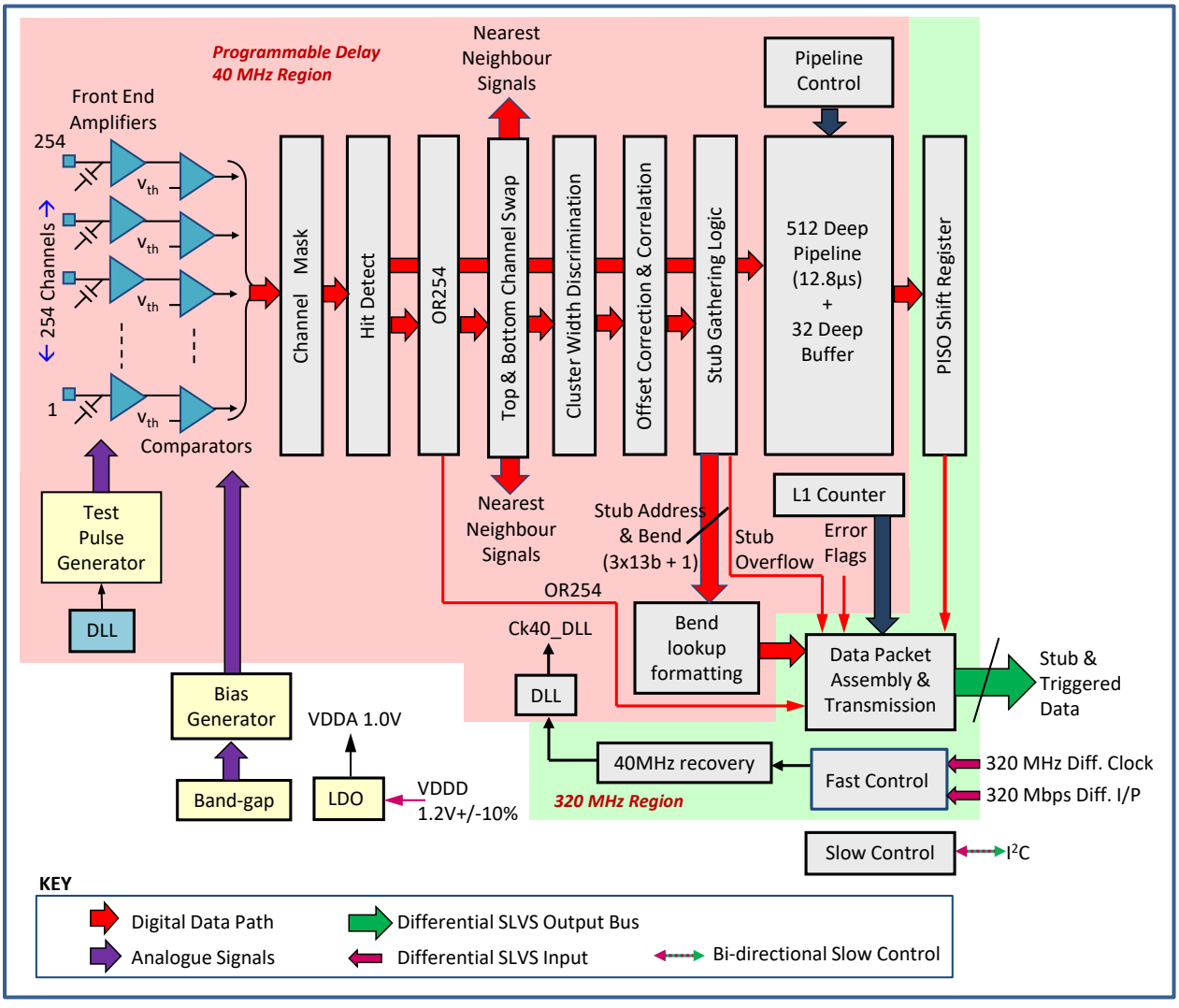

Figure 1 — Block Diagram of CBC3.1 architecture.

Some circuits, such as the Hit Detect circuit, require their timing to be adjusted relative to the BX. This adjustment is provided using a Delay Locked Loop (DLL). The 40MHz clock derived from the FCI is fed into the DLL, which produces multiple versions of the original clock, phase shifted by controlled delays with a resolution of 1ns. The user is able to select which delayed version of the clock is to be used by the circuits, and in this way tune their phase relationship to the BX timing.

► **NOTE:** It is also possible to select the source 40MHz as the clock input to these circuits. As this would suggest, there are sections of the ASIC where data is required to cross from a delayed 40MHz clock to the source 40MHz clock, before eventually crossing to the 320MHz domain, and vice versa. A 40MHz clock output is made available from the ASIC, primarily for test purposes.

There are 254 channels of pre-amplifier, post-amplifier and comparator, which take the charge deposited in the silicon sensors and convert it to binary Hit data in the form of logical 1's and 0's. It is intended that the CBC3.1 be operated with odd and even channels connected to different sensor layers of the module, such that one CBC3.1 ASIC services 127 strips from each of two sensor layers. Multiplexing logic is included to allow for the layers to be swapped.

The analogue front end amplifiers can be stimulated using the on-chip test pulse generator. The test pulse is instigated by a fast command applied through the FCI. The phase relationship of the test pulse to the delayed 40MHz clock is controlled by another DLL circuit (derived from the delayed 40MHz clock), with a programmable 1ns resolution available. The duration of the pulse is fixed by an on-chip timer circuit, however the amplitude and polarity of the pulse is set by programming configuration registers through the I^2C interface.

The analogue front end circuits are powered from a supply that is derived on-chip using a Low Dropout Regulator (LDO), which in turn is supplied by a 1.2V (nominal) power supply that is also used to supply the digital sections of the ASIC.

The various analogue biases used by the front end circuits are derived on-chip using bias generation circuits referenced by a Voltage Reference Bandgap. These bias circuits have default settings on start-up, but can be re-programmed via the I^2C interface. The bandgap circuit uses a PMOS transistor in place of the more traditional diode, which makes it more tolerant to radiation damage, but leaves it susceptible to process variation. To compensate for this, the CBC3.1 design includes trimming circuitry, which employs eFuse technology to allow the trimming to be hard-wired into the ASIC at the time of wafer probing by electronically blowing the fuses. After blowing the fuses, it remains possible to override this hard-wired value by writing a new setting via the I^2C, but this new setting will be lost after power-down. The eFuse technology is also used to hard-wire a unique identifier code into the ASIC, making it possible to track the ASIC's as they are integrated into the system.

After the Hit Detect stage, the Hit data takes two paths. The first path sees the raw Hit data transferred directly to the 512 deep pipeline memory, while the second path processes the Hit data looking for Stubs.

Hit data is written to the pipeline under the control of a Write Pointer operating off the delayed 40MHz clock. The pointer wraps around continuously (overwriting the stored data) unless a reset is applied. A Read Pointer shadows the Write Pointer, delayed by a pre-programmed separation. This separation is programmed to match the latency of the Level 1 Trigger. When a Level 1 Trigger is received as a command via the FCI, the Read Pointer will copy the Hit data from the corresponding pipeline address into the 32 deep Output FIFO, from where it is transmitted off-chip via a dedicated SLVS output operating at 320MHz.

Independent of the latter, the Stub Finding Logic (SFL) will asynchronously process the Hit data looking for coincidences of hits on channels that are connected to different sensor layers. The output from the SFL is latched using the delayed 40MHz clock, and then the Stub Gathering Logic multiplexes it to the Stub Data Assembly Logic to be arranged into the packet structure for output on dedicated SLVS outputs operating at 320 Mbps. At this speed it is possible to output the data corresponding to up to three Stubs within the time frame of one BX (25ns).

The SFL also generates data relating to the angle of the Stub, called Bend data. This is generated at 5 bit resolution, but passes through a programmable Look-Up-Table (LUT) to convert it to 4 bits prior to being assembled into the Stub data packet.

A 254 input OR of the Hit data is available as a test feature, the result from which is output along with the Stub Data.

The circuits that make up the CBC3.1 ASIC are described in the following sections.

Analogue Front End

AC coupled n-on-p type silicon sensors were chosen for the tracker, so unlike the earlier versions of the CMS Binary Chip, the CBC3.1 channel will only operate for electrons readout. As AC coupling is included in the sensor, there is no leakage current compensation circuit included in the CBC3.1 design.

The CBC3.1 front end channel is shown in Figure 2. Each input is bonded to one strip of a silicon sensor. Charge generated by an ionizing event in the strip is read out by a *pre-amplifier* and integrated onto a 100fF feedback capacitor. The feedback capacitor is discharged by a 100kΩ resistive feedback network.

The resulting voltage pulse from the pre-amplifier is further amplified by a capacitive gain *post-amp*. A large value feedback resistance stabilizes the amplifier. To compensate for any mismatch in amplifier and comparator thresholds from channel to channel, each *post-amp* has a programmable offset adjustment controlled using a differential current *Ipaos*. These control-currents are programmed using an 8 bit register in each channel.

The comparator stage detects signals which cross a defined threshold and will produce a digital “1” output for as long as the signal stays above the threshold. The polarity select circuit used in previous versions is retained, to allow the polarity of the Hit to be inverted if desired. The 4 bit programmable hysteresis is also retained, with a default setting for maximum hysteresis.

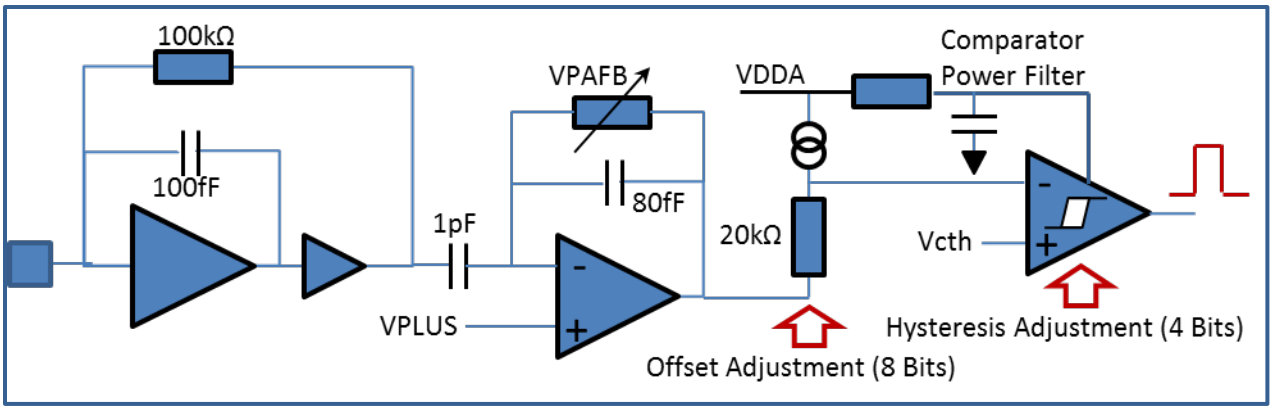

Figure 2 — CBC3.1 Analogue Front-End

Channel Gain

The target gain of the Pre-amplifier and Post Amplifier combined is 50mV/fC at the comparator input.

Channel Noise

The noise target of the Pre-amplifier and Post Amplifier combined is ≤1000 electrons for 5 cm strips with a leakage current up to 1µA. Design studies show that this can be achieved for an external capacitance (sensor + stray) up to 10 pF, for an input FET power of 240µW, at an operating temperature of 0 °C.

Pulse shaping

The amplifier pulse shape peaking-time is designed to be less than 20ns with a return to the baseline within 50ns (see figure 3).

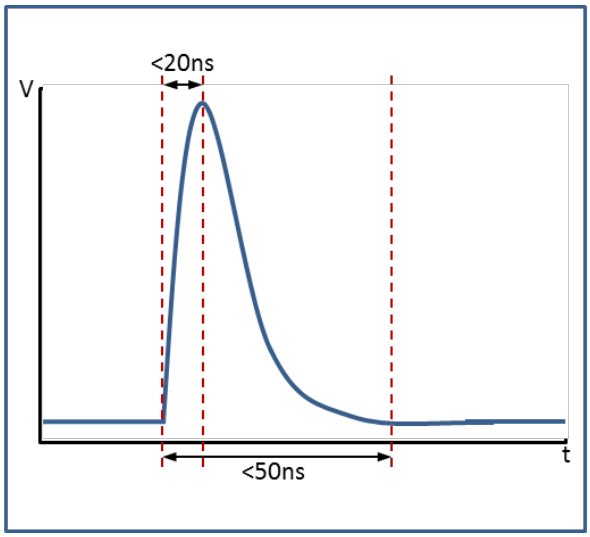

Figure 3 — CBC3.1 pulse shape

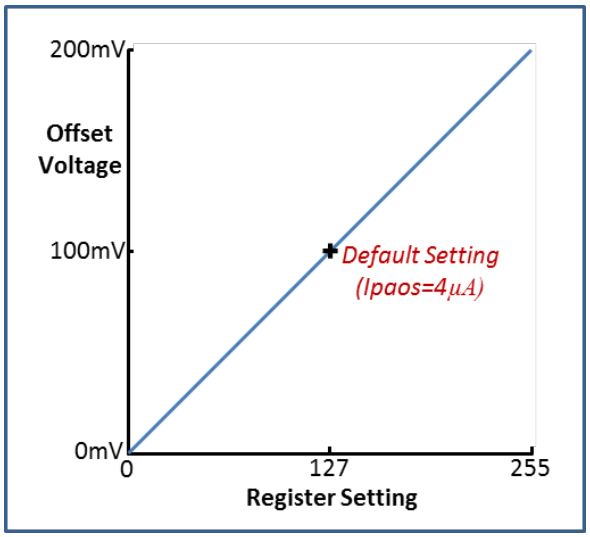

Figure 4 — Post-amp Offset Voltage setting

Post-amp Offset Adjustment

The comparator threshold (*Vcth*) to each channel is set global to the chip, so in order to compensate for different input offsets from channel to channel (due to process variations), the level of the input signal to the comparator can be trimmed on a channel by channel basis. This is achieved using a programmable, differential current to bias the output branches of the post-amp. The voltage at the output of post-amp is offset by adjusting the current through the 20k resistor shown in figure 2. Using the default setting for *Ipaos*, the default offset given is 100mV. 8 Bits of programming gives a range of offset adjustment of 200mV, as shown in figure 4 previous.

Overload recovery

An individual channel is designed to be able to respond to normal size signals within 2.5µs of having received a hip-type (Heavy Ionizing Particle) signal of up to 4pC.

'Hit' Comparator Time-walk

The comparator time-walk is designed to be no larger than 16ns, defined as the maximum time difference between the 50% amplitude points of the comparator digital output signals measured with a 1.25fC and a 10fC input signal, for a comparator threshold of 1fC.

'Hit' Comparator threshold trim

The global Comparator Threshold Voltage (*Vcth*) is provided by a 10 bit resistor ladder DAC (monotonic). VDDA and GND are used as the reference voltages for the ladder, giving millivolt resolution ($\frac{VDDA-GND}{1024}$).

There is no amplifier buffering this voltage.

Digital Back End

Delay Locked Loop (DLL)

An on-chip 40MHz clock derived from the serial fast command input is used for timing the data path from the output of the comparators through to the *Output Data Serialiser and the Data Assembly Logic* at the back-end, where it is necessary to cross to the 320MHz clock domain.

The derived 40MHz clock is not necessarily well aligned with the data generated by the BX, so a *Delay Locked Loop* circuit is included to provide phase adjustment of the 40MHz clock used by the *Hit Detect* and *Pipelined Data Logic*.

Selection of the clock phase is controlled by a register that is programmable via the I^2C Interface. The phase is programmable in 1ns increments from 0 to 25ns. There is a limit to the range of frequencies for which the DLL will operate, so for low frequency wafer level tests, the first setting of the DLL is a bypass that allows the derived clock to be distributed instead of the phase shifted version.

To ensure the correct alignment of the fast control commands with the shifted clock, they also pass through the DLL circuit and are automatically adjusted by the same amount as the clock.

Channel Mask

The Channel Mask is a 254 bit register programmable through the I^2C slow control interface. The outputs of the register are used to disable the Hit Detect circuits on a channel by channel basis. The Hit Detect output of a masked channel will be held at logic 0. L1 data output from the 'masked' channel will be logic 0, and there will be no Stub Data as a result of this channel.

► **NOTE:** In the case that charge is spread across three or more sensor strips, if the central strip is connected to a masked channel, then this will appear to the Stub Finding Logic as though there were two separate hits.

Hit Detect Logic

The Hit detect logic processes the output signal from the channel's Comparator circuit, and provides four resulting outputs, each with their own characteristic, as described in points 1 to 4 below:

1) **Fixed Pulse Width:** The output from the Hit comparator is latched for a full 25ns clock period. Non-synchronous comparator output transitions are captured. The output is a fixed 25ns pulse, regardless of the width of the comparator output pulse. Hits following immediately one after another in subsequent clock cycles will be captured provided the channel returns below the comparator threshold for each hit.
2) **40MHz Sampled Output:** The output from the comparator is sampled using the 40MHz clock from the Delay Locked Loop. Only comparator outputs present on the rising edge of the clock will be captured and the output will only return to zero on the first rising clock edge following the comparators return to zero. The minimum width of output pulse is one clock cycle. Hits following immediately one after another in subsequent clock cycles will be captured even if the channel does not return below the comparator threshold for each hit.
3) **Logical OR Output:** The outputs from (1) and (2) are passed through a logical OR to provide a combined result.
4) **HIP Suppressed Output:** Associated with (2) is a selectable Highly Ionizing Particle (HIP) suppression circuit. This circuit will check the length of the pulse and if it exceeds a pre-programmed number of clock cycles, the circuit will force the output to return to zero. The number of clock cycles for which the pulse can remain high is pre-programmed by setting 3 bits in a register, using the I^2C interface (see table 13). This suppression can be applied to either the 40MHz Sampled Output or the Logical OR Output by appropriate multiplexer selection.

Any of these results can be routed independently to the Stub Logic and/or Pipeline. Output selection is configured via the I^2C slow control interface. Details of the register settings can be found in tables 20 and 21.

Figure 5 shows a block diagram for the circuit, and figure 6 illustrates the different output options for different input signal scenarios.

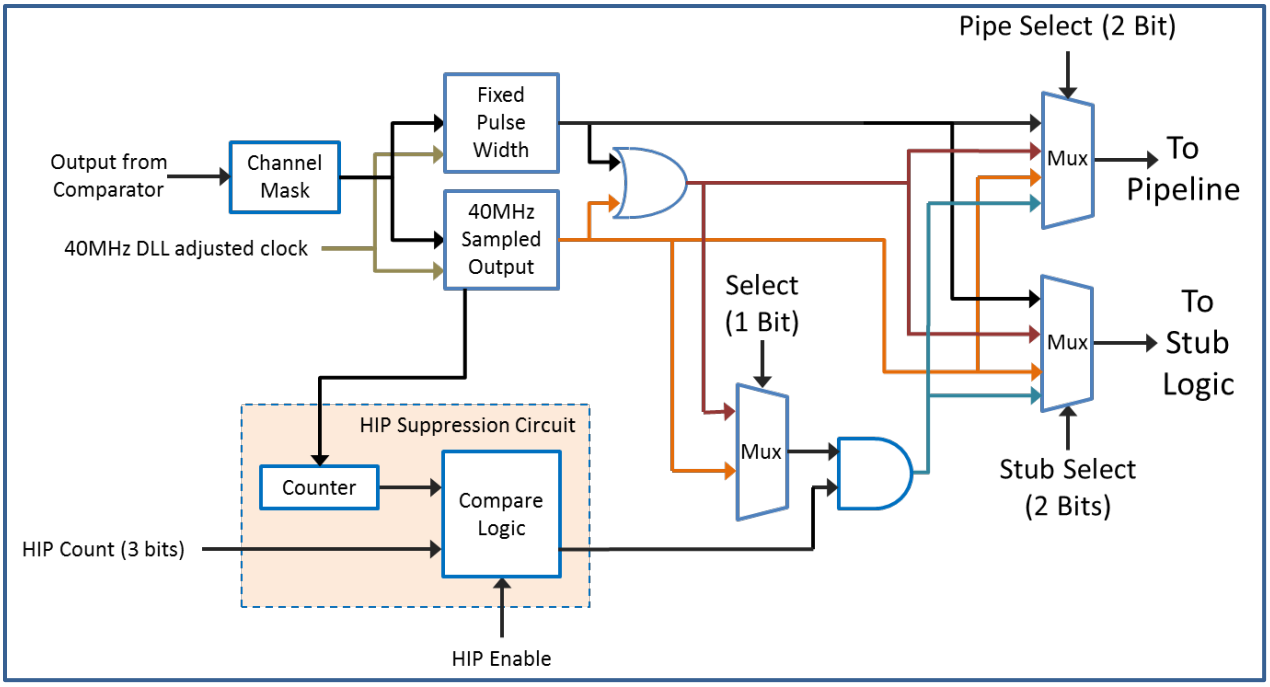

Figure 5 — Hit Detect block diagram.

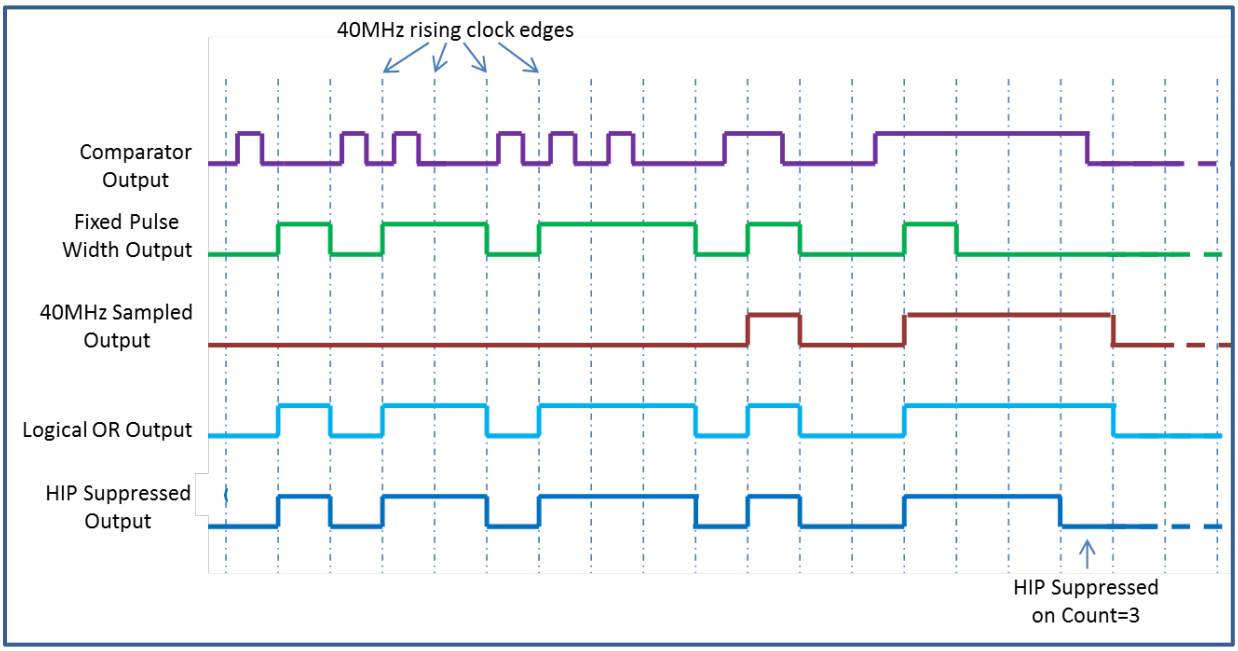

Figure 6 — Examples of Hit Detect processing of different input signal scenarios.

The timing of the circuitry for the Fixed Pulse Width and the 40MHz Sampled outputs is slightly different. Figure 7 and 8 demonstrate the timing constraints of each path.

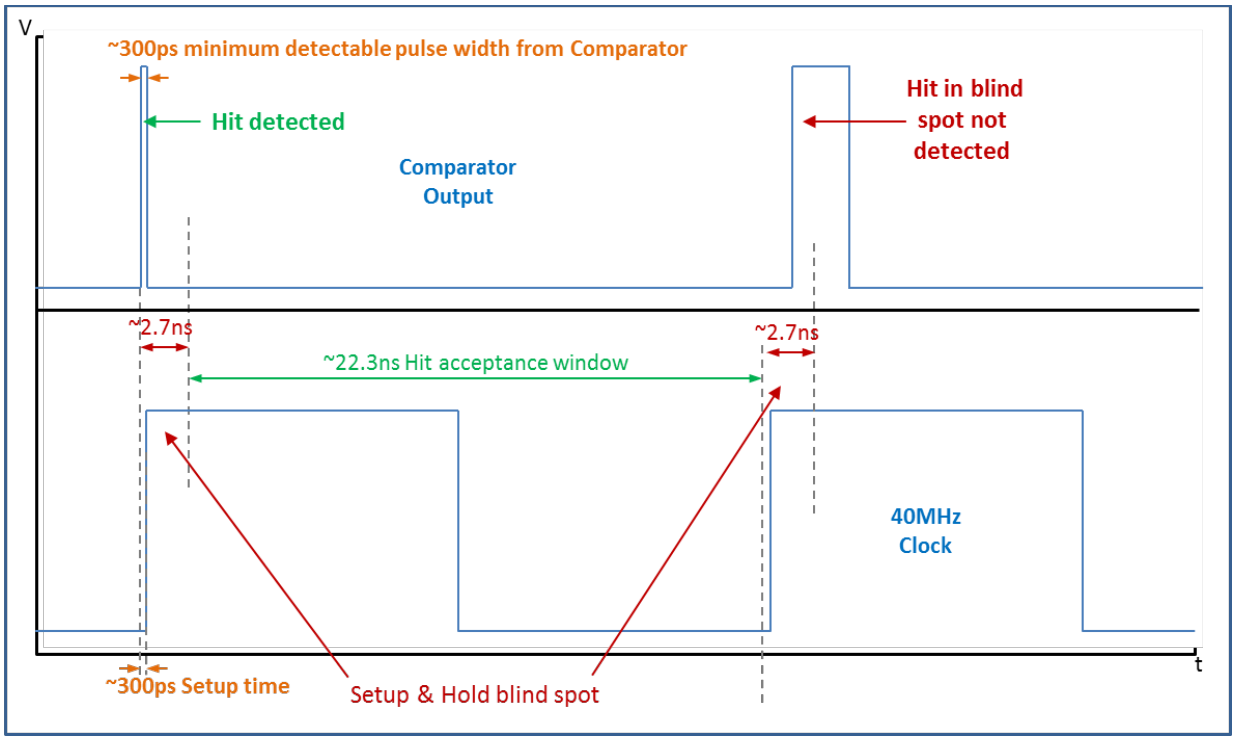

Figure 7 — Timing constraints for the Fixed Pulse Width section of the Hit Detect circuit.

► **NOTE:** When an asynchronous signal is captured in one clock period, the logic must be reset on the rising edge of the next clock period. This results in a period during which any new signals arriving will be lost. In the worst case this period is expected to be approximately 2.7ns.

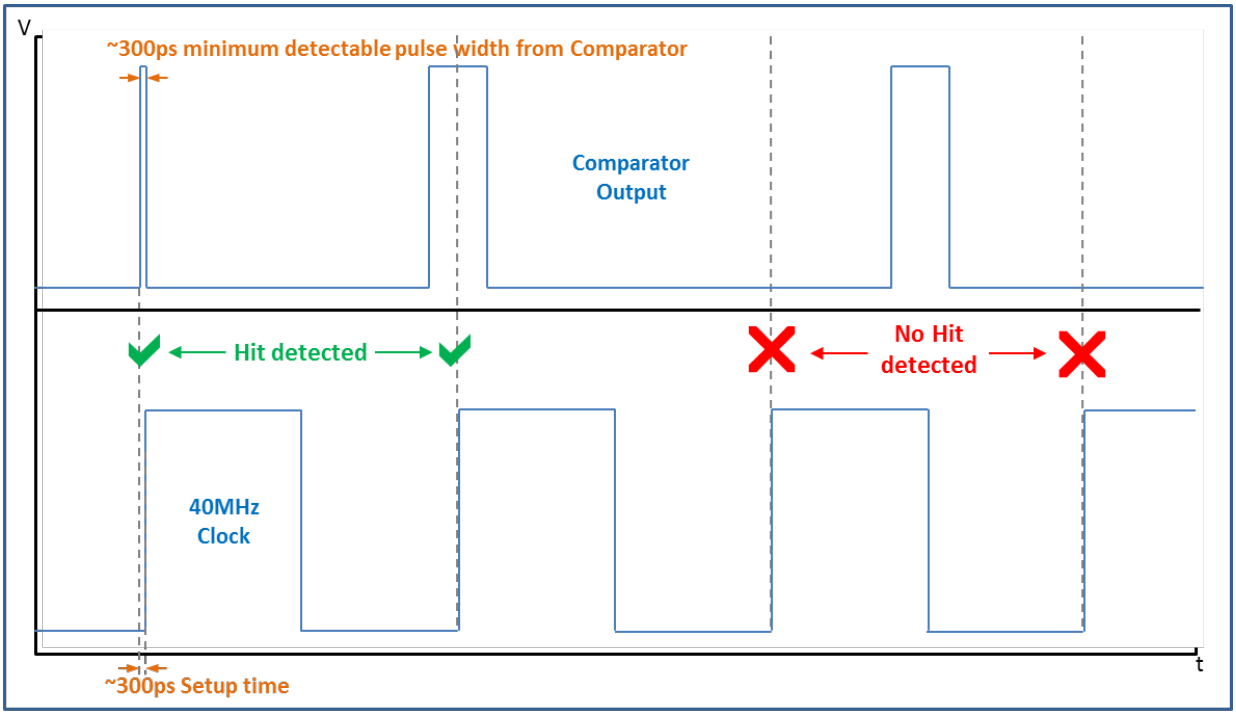

Figure 8 — Timing constraints for the 40MHz Sampled section of the Hit Detect circuit.

Pipelined Data Logic:

Figure 9 below, shows a simplified block diagram of the Pipelined Data logic.

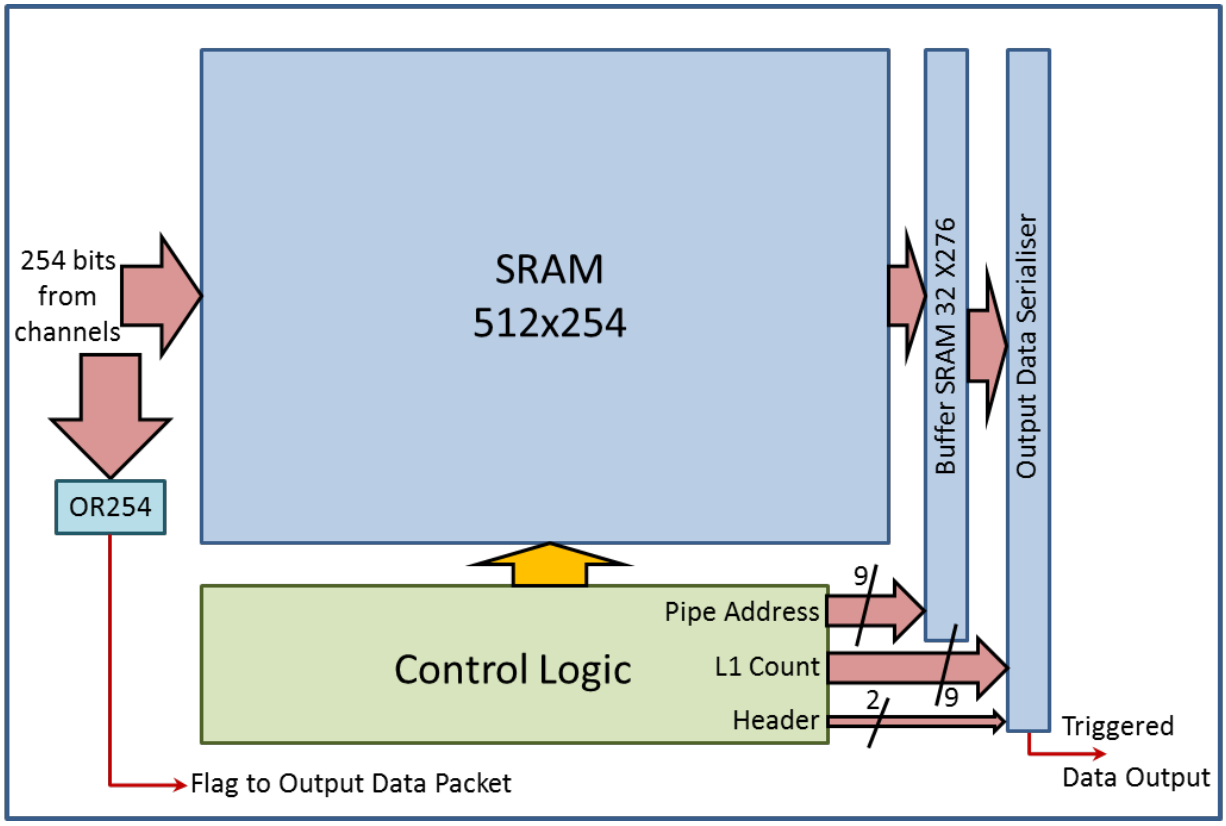

Figure 9 — Block diagram of the pipeline

OR254

Whatever the choice of output from the Hit Detect circuit to the Pipeline, this data feeds into a 254 input logical OR network, which produces an output if any of the channels are hit. This output, if enabled via I2C (table 32), appears as a flag in the output data packet.

Pipeline SRAM

In parallel with the logical OR, the Hit Data is input to the *Pipeline SRAM, a* 512 bits deep memory storing 512 clock cycles (12.8µs @ 40MHz) of data for all 254 channels. The SRAM cells are of simple dual-port architecture and although they are designed to cope with radiation induced leakage current issues, they are not required to be immune to SEU events since loss of data can be tolerated. *Pipeline Control Logic* sequences the writing and reading of data into the *Pipeline SRAM*.

Pipeline Control Logic

Figure 10 shows a block diagram of the *Pipeline Control Logic*. The logic is initialised by sending a *Fast Reset* command via the *Fast Control Interface*. In this way the pipelines for each chip on the module can be synchronized.

The sequencing of data being written into the pipeline RAM is controlled by two pointers, the *Write Pointer*, and the *Trigger Pointer*. These are realized using 9 bit counters feeding into 9:512 decoders.

When initialised, the *Write Counter* will start counting from 0. The *Write Decoder* will convert this count to a Pipeline SRAM position, into where the data from the front end will be written. The counter will count to 511 before wrapping around to start again. Any data previously written will then be overwritten with new data as the pointer increments.

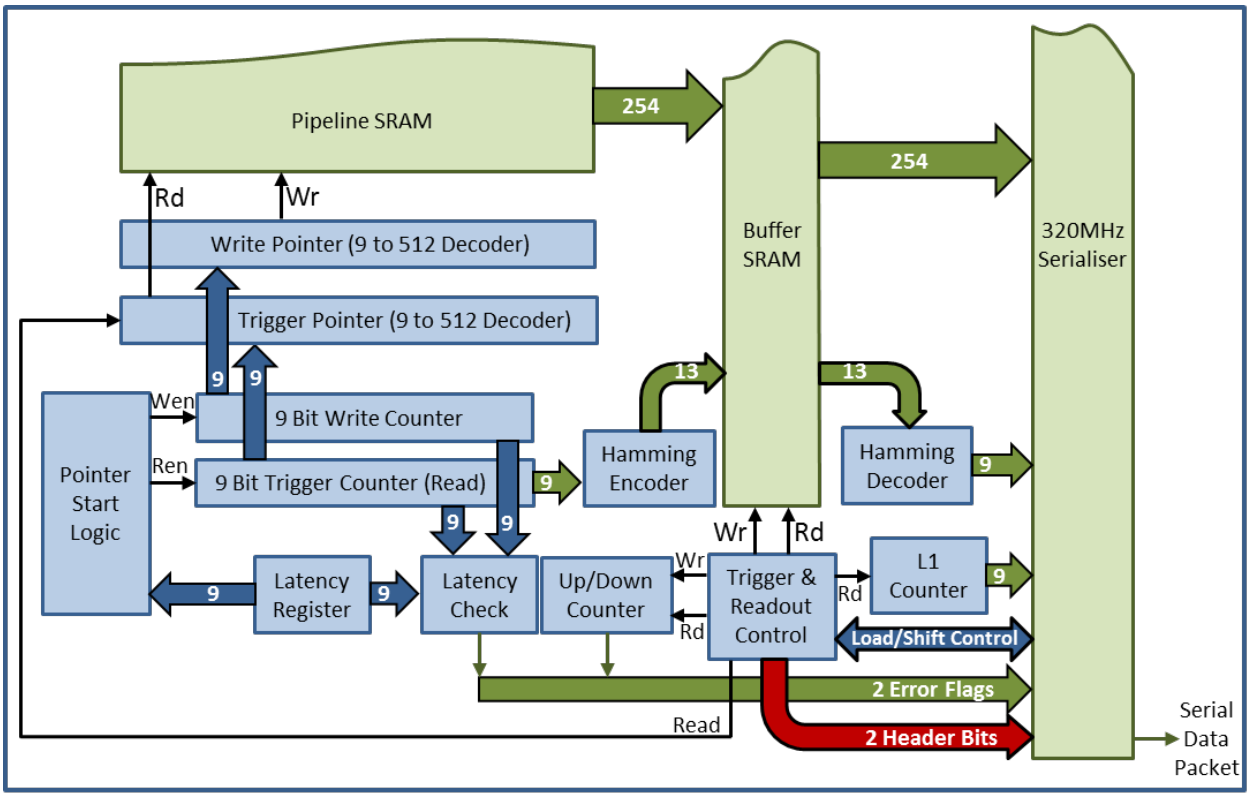

Figure 10 — Block diagram of the Pipeline Control Logic

The *Trigger Counter* waits for a predefined latency period after the *Write Counter* before it starts. Its decoder converts the counter output into the RAM address to be read should an external trigger be received.

Pointer Start Logic sets the correct latency as determined by the value programmed in the *Latency Register*. This 9 bit register is programmed using the I^2C interface. The latency between trigger and write pointers is constantly monitored by the *Latency Check Logic*, which sets an error bit in the L1 Data packet if the measured latency does not match the programmed latency (See figure 11).

The *Trigger and Readout Control* block has two functions. Firstly it controls the transfer of data from pipeline to buffer. If a *Trigger* fast command is received via the FCI, then a READ signal is applied to the pipeline and a WRITE signal is applied to the *Buffer SRAM* to transfer the pipeline data along with the Read address into the *Buffer SRAM*.

Its second function is to control the transfer of data from the *Buffer SRAM* to the *Output Data Serialiser* by monitoring whether there is any data in the *Buffer SRAM*. If there is no data, then it prevents the PISO register from loading or shifting. Once data becomes available, the control logic sequences the transfer of data from the *Buffer SRAM* to the *Output Shift Register*, and the data is then shifted out of the register.

Two error flags are used to indicate problems with the operation of the circuit. The first is set by the *Latency Check* circuit. This monitors the difference between the *Write Counter* and *Trigger Counter* and compares it to the value loaded into the *Latency Register*. If a difference occurs, then the *Latency Error Flag* is set. The second error flag comes from the *up-down counter*, which records the number of items stored in the *Buffer SRAM.* If the count reaches 32 then the *FIFO Full Flag* is set. Both flags are read out after the two bit header of each Triggered Data Frame.

Buffer SRAM

The *Buffer SRAM* stores data from bunch crossings which have been triggered as useful. When this happens data is read from the *Pipeline SRAM* and written into the buffer SRAM. It has a depth of 32 bits to store 32 events awaiting readout and is 267 bits wide, the extra width coming from the 13 bits required for the Hamming encoded pipeline address from which the data came. The SRAM cells used are of the same design as those used in the pipeline SRAM. The *Buffer SRAM* essentially operates as a FIFO, with Write and Read pointers to sequence data to and from it.

Output Data Serialiser

Data stored in the *Buffer SRAM* must be serialized for output from the CBC at 320Mbps. A Parallel-In-Serial-Out register (PISO) reads parallel data from the *Buffer SRAM* on the 40MHz clock domain, and converts it to a serial stream on the 320MHz clock domain. The Serialiser 'handshakes' with the *Pipeline Control Logic* to ensure correct timing of the load and shift functions across the clock domains. To address potential stability issues that could arise with clock domain crossing at different delay settings of the 40MHz clock, two control bits (*Select_Clk40_DLL* and *Select_Clk40_Ref*) have been added to enable the timing of control signals to be adjusted should the need arise. These control bits are set by programming the relevant register (Register 12) via the I^2C interface.

The serial *Triggered Data* packet is 276 bits long and includes a two bit header plus two error bits, the pipeline address (9), the L1 Count (9) and the channel data (254), as shown in figure 11 on the next page. The pipe address and L1 counter data is arranged to output MSB first; the channel data comes out channel 1 first and is ordered channel 1 to 254 consecutively, i.e. no reordering of the data into top and bottom sensor channel groups is performed.

Before being output at 320MHz via a differential SLVS driver, the *Triggered Data* packet is synchronized with the *Stub Data* packet such that the first header bit is aligned with the first suitable SYNC bit in the *Stub Data* packet (see Data Assembly Logic section).

► **NOTE:** Due to the data crossing multiple clock domains, the *Frame Period* is 950ns. As the *Frame Length* is only 862.5ns, there is a minimum gap of 87.5ns (28 Clock cycles at 320MHz) before the start of the next frame. This gap is padded with zeros.

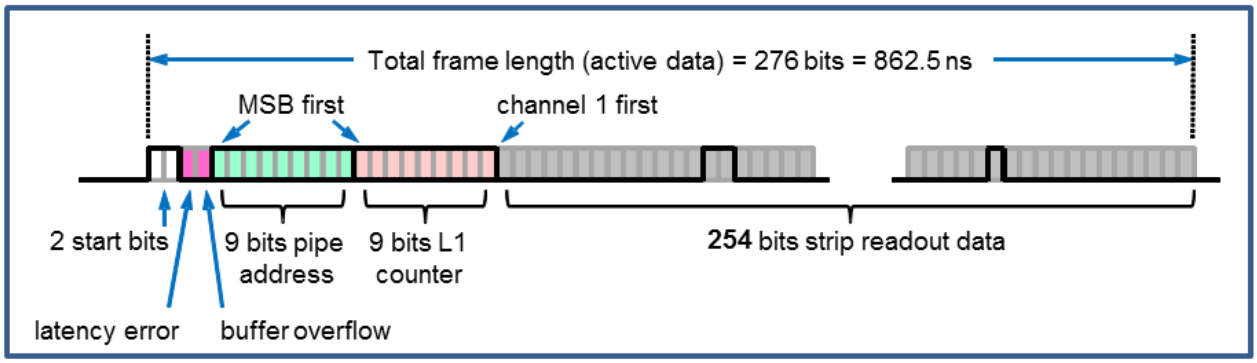

Figure 11 — Illustration of the serial *Triggered Data* packet.

On application of a *Fast Reset* command, any *Triggered Data* packet being transmitted will be truncated and the data lost. The pipeline pointers will be reset and any existing data will be overwritten with new data from the channels. It is important to remember that there is a one *Bunch-Crossing* period (BX = One 40MHz cycle) lag in the command word decoding, so a *Trigger* command should not be sent any sooner than two BX after the *Fast Reset* command. In practice, the timing of the first trigger following a reset will be determined by the latency of the trigger system. After receiving a *Trigger* command, the time to the appearance of the L1 Data packet header will depend on the 40MHz clock delay in use. For low delay settings (0 to 10) the header will appear in the 11th BX following the one in which the *Trigger* command was sent, as demonstrated in figure 12. This will be extended by 1 BX for higher delay settings (11 to 24), as demonstrated in figure 13.

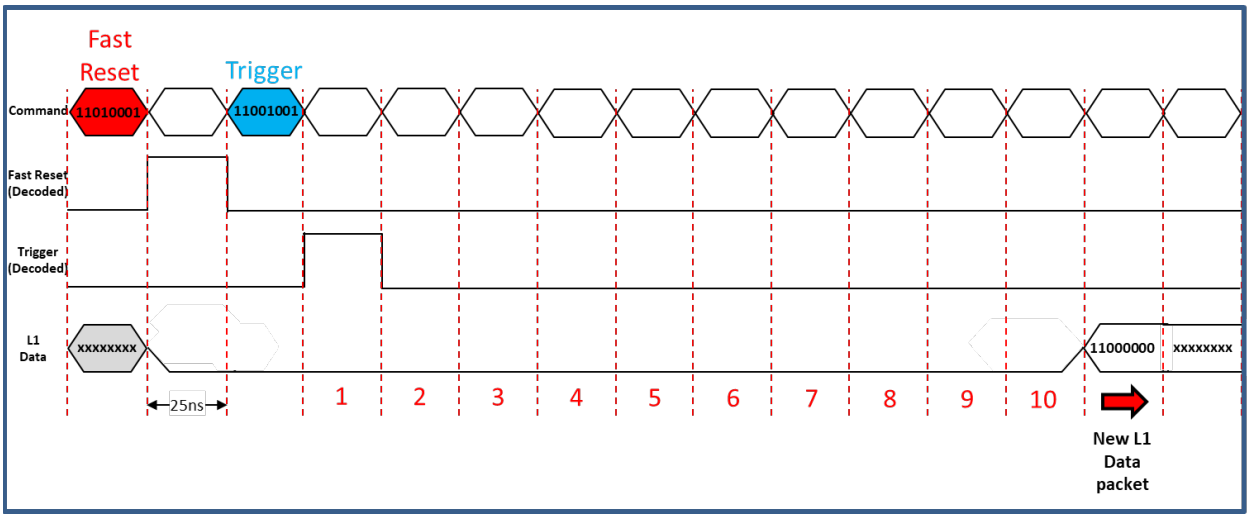

Figure 12 — Illustration of the *Triggered Data* packet appearance following the 1st *Trigger* command after a Fast Reset command, for a programmed latency of 1 and Clock 40 delay settings 0 to 10.

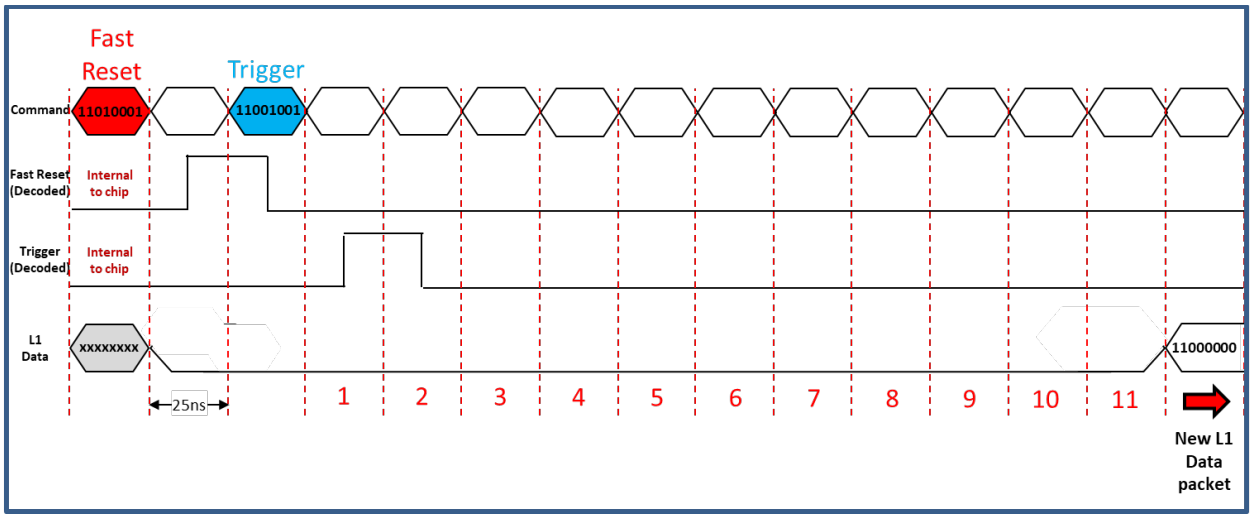

Figure 13 — Illustration of the Triggered Data packet appearance following the 1st Trigger command after a Fast Reset command, for a programmed latency of 1 and Clock 40 delay settings 11 to 24.

► **NOTE:** This timing relationship is different for the CBC3.

L1 Counter

A triple redundant 9 bit counter records the number of L1 triggers received (up to 511). The counter is reset through the Fast Control Interface by its own dedicated reset command, *Orbit Reset*. The count is output as part of the *Triggered Data* packet (figure 11).

► **NOTE:** The L1 counter is reset to zero and incremented on receipt of a trigger, which means that the first L1 event emerging from the chip will have a count of 1. The only exception to this occurs, if the *L1 Count Reset* is sent simultaneous with a trigger, or one clock cycle later. In this case there is insufficient time for the counter to recover from reset and the corresponding L1 event will have a count of zero.

Stub Data Logic:

The CBC3.1 is designed to be used in conjunction with two sensors, as illustrated in figure 14. Strips on the bottom sensor (*Seed Layer*) are connected to the CBC3.1's odd numbered channels, starting with channel 1. Conversely the strips of the top sensor (*Correlation Layer*) are connected to the CBC3.1's even numbered channels. The Hit Detect signals from the channels are processed in order to find correlations between the hits in both of these layers. The hits are processed differently depending on whether they are associated with the Seed layer or the Correlation layer. The result of this processing is the identification of potential *High Momentum* tracks. As only part of the track is being identified, these are known as *Stubs*.

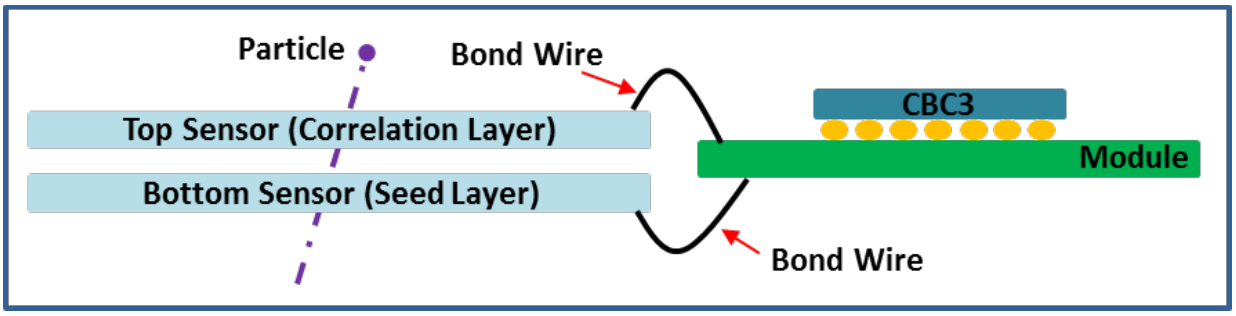

Figure 14 — An illustration of two sensor layers wire-bonded to a module hybrid substrate on which CBC3.1s are bump-bonded.

Layer swapping logic

Whatever the choice of output from the Hit Detect circuit to the Stub Finding Logic, this data feeds into a multiplexer that allows for the outputs from odd and even channels to swap place. The reason for this is that in some module arrangements the top and bottom sensor positions become swapped, such that the top sensor is now the *Seed layer*. Due to the asymmetric nature of the *Stub-Finding Logic*, it is necessary to swap the odd and even channels to accommodate this change of layer function. A multiplexer circuit at the output of the Hit Detect circuit can be programmed to perform a swap of all odd and even channel signals. Programming is via the I^2C interface and acts on all channels on the chip.

► **NOTE:** If this function is in use, then Hit signals that pass to neighbouring CBC3.1s will be swapped before they are output from the chip.

► **NOTE**: All chips on one module are expected to have the same layer set up, so the inter-chip logic does not account for different layer set ups on neighbouring chips.

Stub-Finding Logic

Hit Detect signals pass through the layer swapping logic, and are then processed in order to find correlations between the hits in both layers. A hit in either layer is referred to as a *Cluster*. These may be one channel wide, or greater if adjacent channels on the same layer have also produced a hit. The first stage of the *Stub-Finding Logic* examines the width of each *Cluster* in each layer, and rejects unsuitable candidates. This *Cluster Width Discrimination Logic*, compares the width of each *Cluster* against a programmable value, and suppresses them if they are found to be wider than this value. This *Rejection Width* is programmable up to and including a width of four adjacent channels. Any *Cluster* of five adjacent channels or more will always be rejected.

► **NOTE:** If a Cluster of multiple channels falls across a masked channel, this will appear to the logic as if there were two separate Clusters, and individually these smaller Cluster sizes might not be rejected.

An illustration of a valid correlation and a rejected Cluster is shown in figure 15.

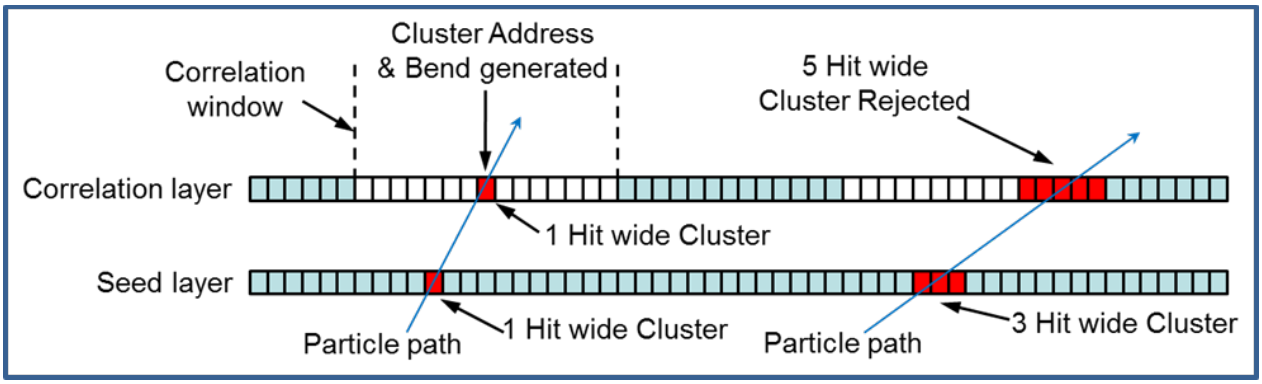

Figure 15 — Illustration of a valid correlation between layers and a rejected Cluster.

Having rejected any oversized Clusters, the next stage of logic tries to match any *Cluster* on the *Seed layer* with those on the *Correlation layer*. Matching is restricted to within a defined 'window' in the *Correlation layer*. This *Correlation Window* is essentially a group of neighbouring channels centred on a channel in the *Correlation layer*, and each *Correlation layer* channel has its own *Correlation Window* with which the *Cluster* in the *Seed layer* can be compared. The size of the *Correlation Window* is programmable in half channel steps, up to $\pm$7 channels about a centre channel, as illustrated in figure 16. To correct for geometrical offset due to the position of the module in the r-ϕ plane, the position of the *Correlation Window* can be offset in half channel steps up to ±3 channels, also illustrated in figure 16. This programmable offset is available on a regional basis only, with four independently programmable regions per chip. The offset value for each region is programmed into a register via the I^2C interface.

To summarise, I^2C programmability is provided for maximum *Cluster Width* (up to 4 channels, applied to both seed and correlation sensor layers), *Correlation Width* (up to $\pm$7 channels about a centre channel in the correlation layer), and *Correlation Window Offset* (up to ±3 channels for 4 independently programmable regions per chip).

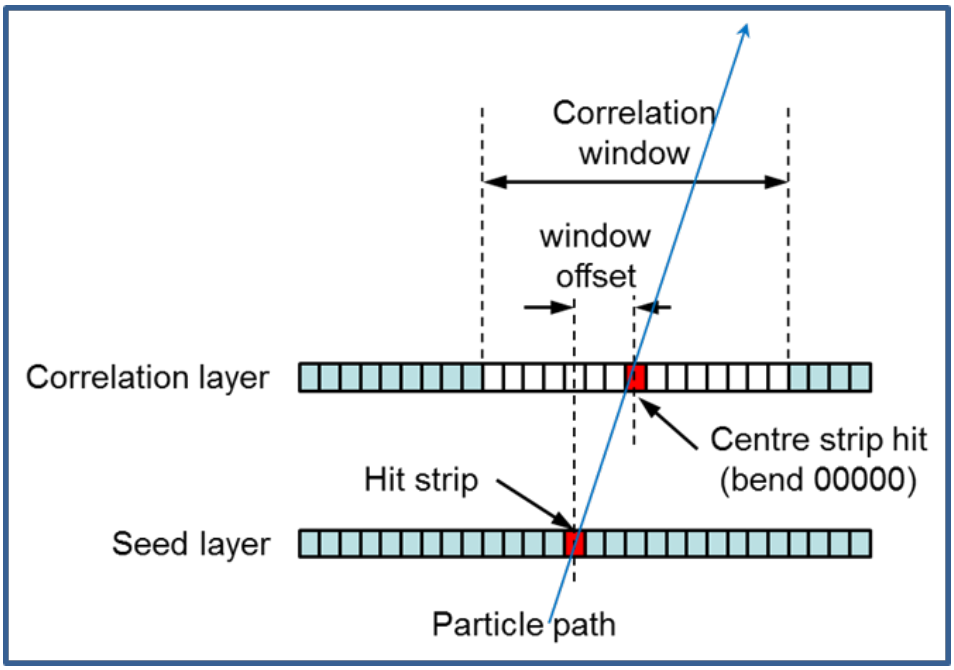

Figure 16 — Example of Correlation Window & Offset Correction

► **NOTE:** If there is more than one distinct *Cluster* within a valid *Coincidence Window*, then only the *Cluster* closest to the matching *Cluster* from the *Seed layer*, will be selected (i.e. The one giving the least amount of bend and therefore, in theory, the highest pT track).

When the logic identifies a correlation between Clusters in the two layers (called a *Stub*), the *Seed Cluster* location is passed to the *Stub Gathering Logic* as a *Correlation Bit*. For a *Cluster* consisting of an even number of channels, the logic generates a signal to indicate that the *Cluster* centre lies between two channels, giving what's known as half-strip resolution. For an odd number of channels, the position of the centre channel is output.

The logic also calculates the offset between the centre of the *Seed Cluster* and the centre of the *Correlation Cluster*, and outputs a representative 5 bit code to the *Stub Gathering Logic*. The latter is known as *Bend* information, and 5 bits allow for the bend to be calculated to half-strip resolution. For the hardware defined *Bend Codes* from the *Stub-Finding Logic* see figure 17. These 5 bit codes are only used internally to the chip and are reduced to 4 bit codes by a programmable Look-Up Table (LUT) before they are output from the chip (see Bend Look-up Table section).

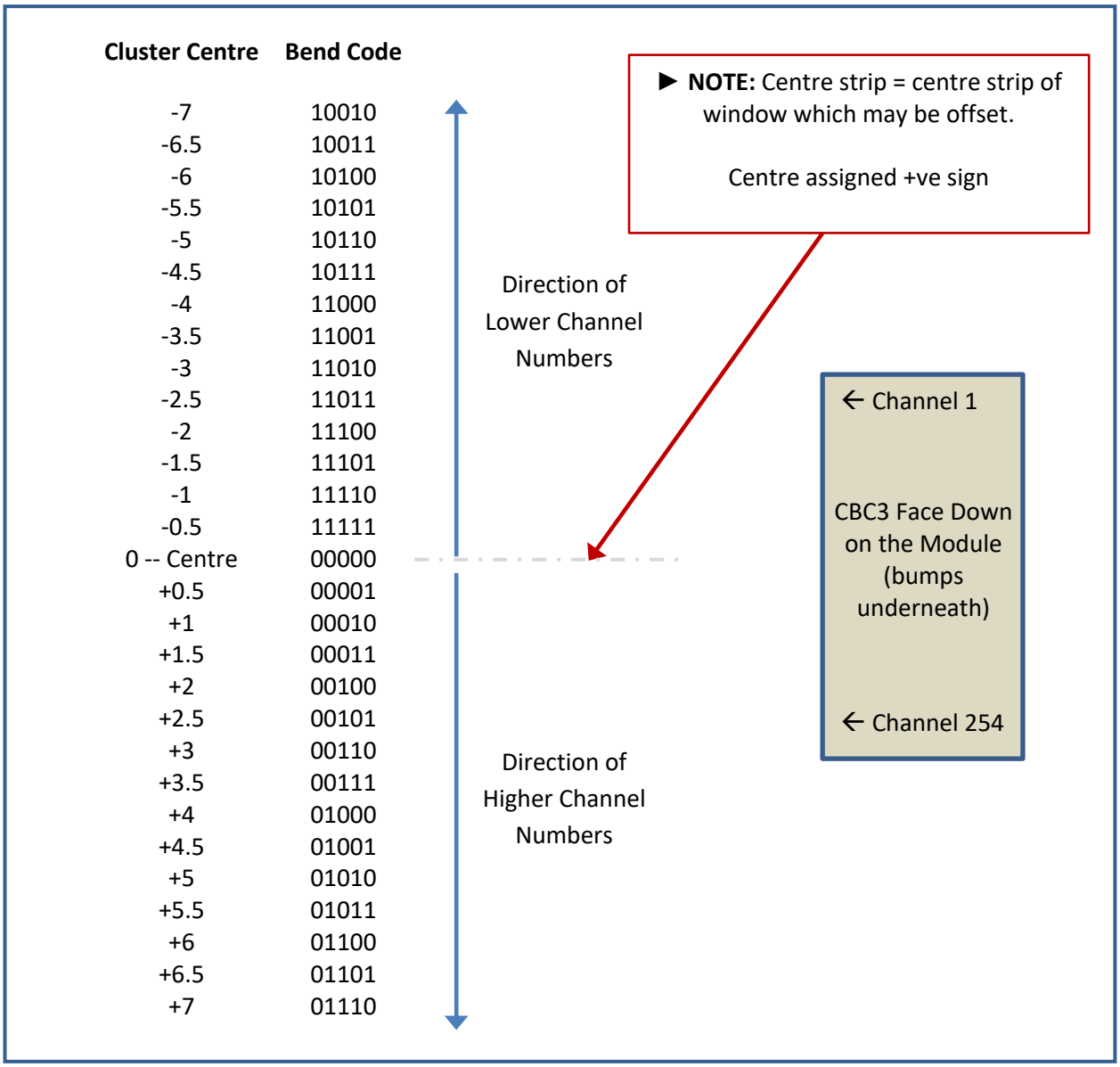

Figure 17 — Definition of Bend Codes generated by the Stub-Finding Logic.

► **NOTE:** The bend code is calculated after the offset correction, so it is independent of the geometrical position in the module or the Tracker.

For Stub-finding coverage to be uninterrupted across a module, Hit signals need to be passed between CBC3.1s in order to identify *Stubs* that straddle the border between chips. The *Stub-Finding Logic* contains additional circuitry to accept and process the inputs from neighbouring chips.

Stub Gathering logic

The *Stub Gathering Logic* generates an eight bit address for every *Correlation Bit* output from the *Stub-Finding Logic*. This is known as the *Stub Address* and can represent a whole or half strip.

The *Correlation Bits* from the *Stub Finding Logic* are latched at the end of each 25ns bunch crossing period to give the asynchronous *Stub Gathering Logic* time to generate the necessary address and bend information. This is achieved using an asynchronous multiplexer architecture which prioritises the *Stubs* based on their location in the CBC3.1, with lower addresses having priority over higher addresses, i.e. hits in the lower numbered channels are given priority over hits in the higher numbered channels.

If a *Stub* has sufficient priority, then its eight bit address and the associated five bit bend information is routed onto one of the three, thirteen bit wide multiplexer buses. An eight bit address is required for half-strip resolution (127 strips = 254 half strips). The data cascades through the multiplexer tree to the output where it is latched by the 40MHz clock at the start of the next bunch crossing period.

Due to data rate limitations, the *Stub Gathering logic* restricts the amount of *Stub Data* to a maximum of three *Stubs* per bunch crossing. If there are more than three *Stubs* per chip per bunch crossing, then only the three *Stubs* with the lowest numbered addresses will be output. The presence of more than three stubs will be indicated by a *Stub Overflow Flag* which will be output in the data packet.

► **NOTE:** As the *Stubs* output **are not** prioritised on the basis of their *Bend* information, if there are more than three stubs per chip per bunch crossing, it is not guaranteed that the *Stubs* read out will be those with the smallest bend, i.e. the highest momentum.

On the next page the *Stub Gathering Logic* architecture is illustrated in figure 18.

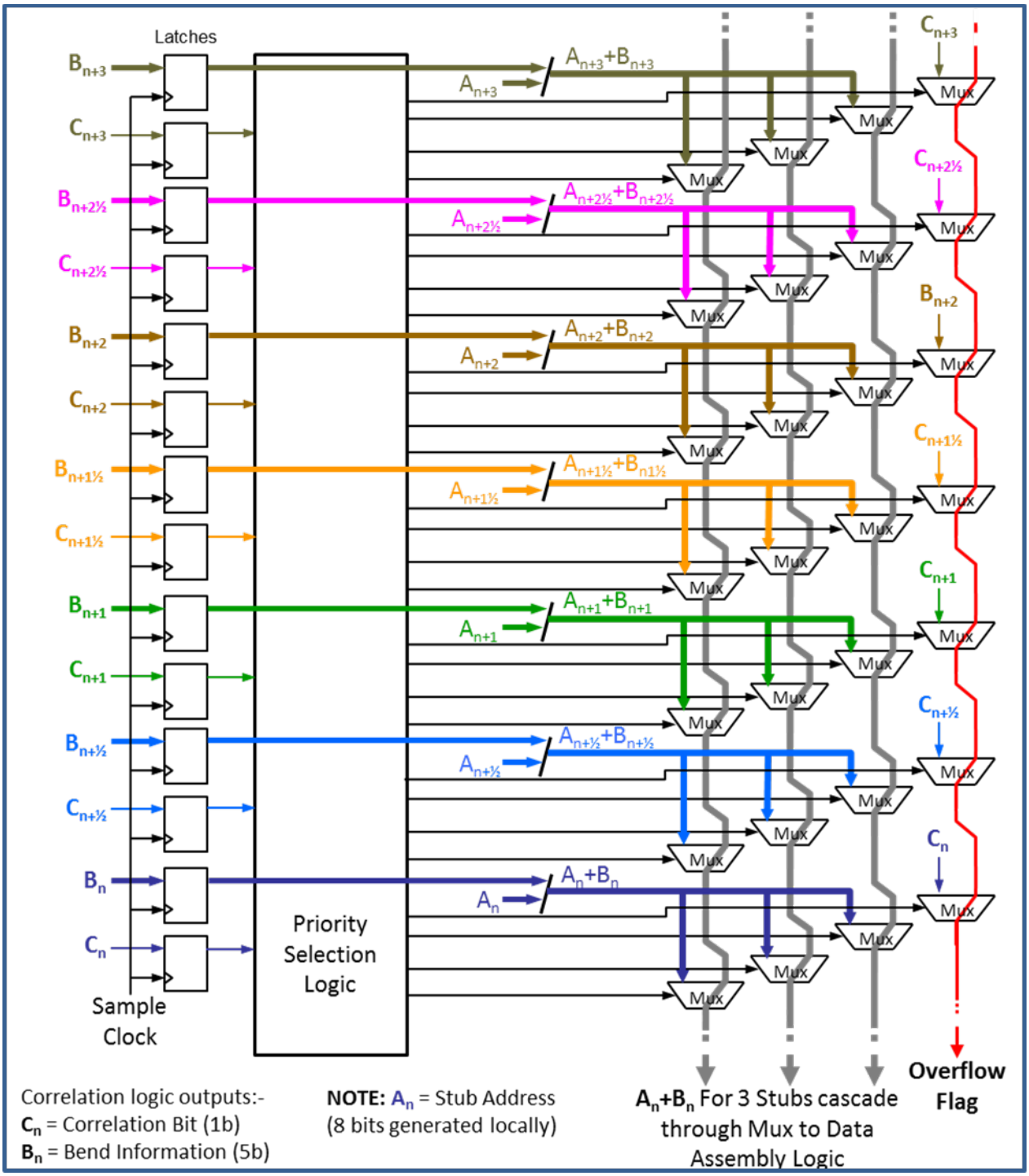

Figure 18 — Illustration of the Stub Gathering Logic

Bend Look-up Table

A LUT was added to convert the 5 bit bend information generated by the *Stub-Finding Logic* into 4 bits at the output, in order to make optimum use of the output data bandwidth. A programmable register block is used to define how the 5 bits map to 4 bits. This is programmed via the I^{2}C interface. The programmable nature of the LUT has the advantage of allowing the 5:4 mapping to be changed, allowing different mappings for different regions of the tracker, or even for different runs. It is also possible to reverse the bend polarity using this method, which might be useful in the event that modules are flipped on the rod/stave. The programming is achieved using 15 I^{2}C registers, where each register contains two sets of 4 bit bend codes corresponding to two of the original 5 bit codes from the *Correlation logic*. Figure 19 provides an example of how different bend resolutions can be programmed.

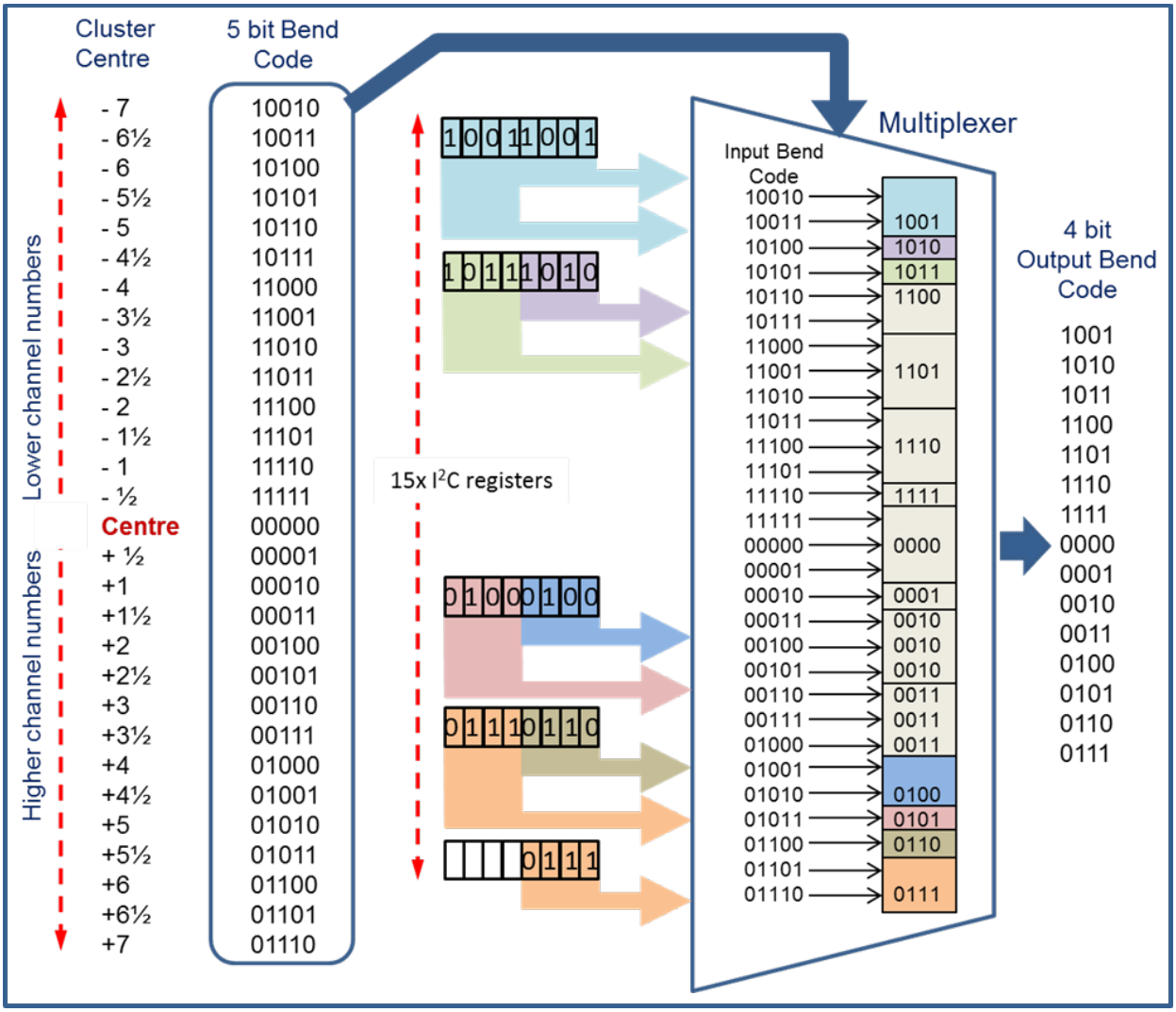

Figure 19 — An example of mapping the 5 bit bend information to 4 bits.

► **NOTE:** The default value in the I^{2}C registers is set as the four most significant bits of the original 5 bit bend code associated with each register. The invalid bend code of 10000 is not shown in the example, but will generate a 4-bit code of 1000 as a default.

Data Assembly Logic

The *Data Assembly logic* takes the *Stub Address* (8 bits for ½ strip resolution) and *Bend* information (4 bits) for up to three *Stubs* and combines it with the information flags, for output onto five SLVS differential outputs operating at 320Mbps. Figure 20 shows the data packet for the *Stubs*. The direction of time flow is shown **left** to **right**, i.e. S1<7>, S2<7>, S3<7>, B2<3> and SYNC are output first. All of the *Stub Data* corresponding to a *Bunch-Crossing* (BX) is read out in a 25ns period to be compatible with the *Bunch-Crossing* period. There is a gap of five BX between the one that generated the data and the one in which the data is observed on the outputs of the chip, as shown in figure 22.

The *Stub Overflow Flag* (1 bit) generated by the *Stub Gathering Logic*, will indicate the presence of more than 3 stubs, while the *OR254* flag (1 bit) will indicate a hit on any of the 254 channels. The *Flags* bit is the logical OR of the *FIFO Full Flag* and the *Latency Error Flag*, both of which also appear separately in the *Triggered Data* packet, and are also stored in a register which can be read via the I^2C interface.

The SYNC bit is output with every Stub Data packet (every 8 clock cycles at 320MHz) and can be used by the receiver for data synchronization purposes.

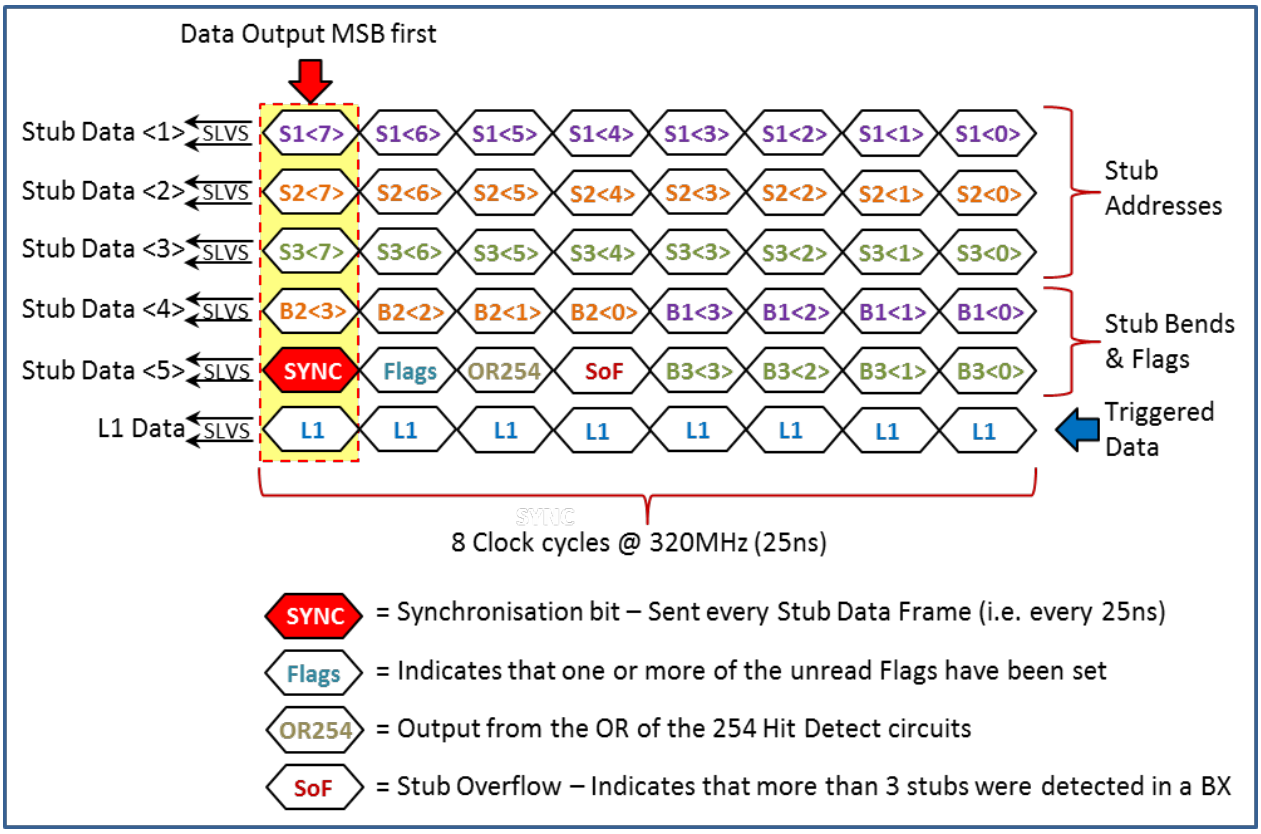

Figure 20 — Illustration of the Stub Data packet.

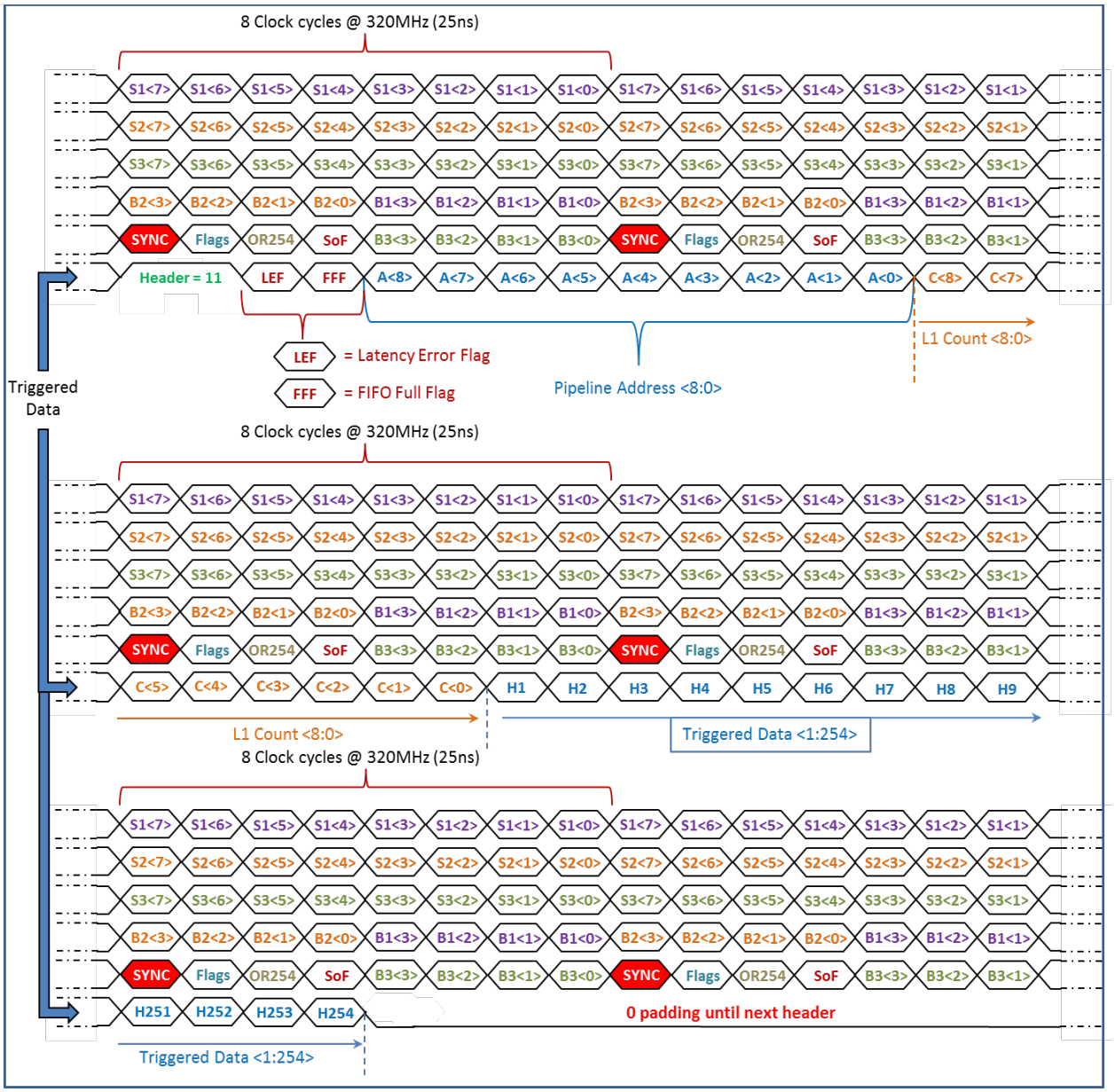

Figure 21 — An illustration of the serial Triggered Data packet in relationship to the Stub Data Packet.

► **NOTE:** The *Triggered Data* packet header (11) is always aligned with the SYNC bit.

Due to the latency between the data generation and its appearance at the output of the chip, a *Fast Reset* command will cause the loss of any data generated in the five BX before the command is sent, the BX in which the command is sent, and the following BX. The first valid BX for which data will appear following a *Fast Reset* command, will be the second BX following the BX in which the command was sent. The data for this event will appear on the outputs in the eighth BX following the BX in which the command was sent. Figures 22 and 23 illustrate the timing of events.

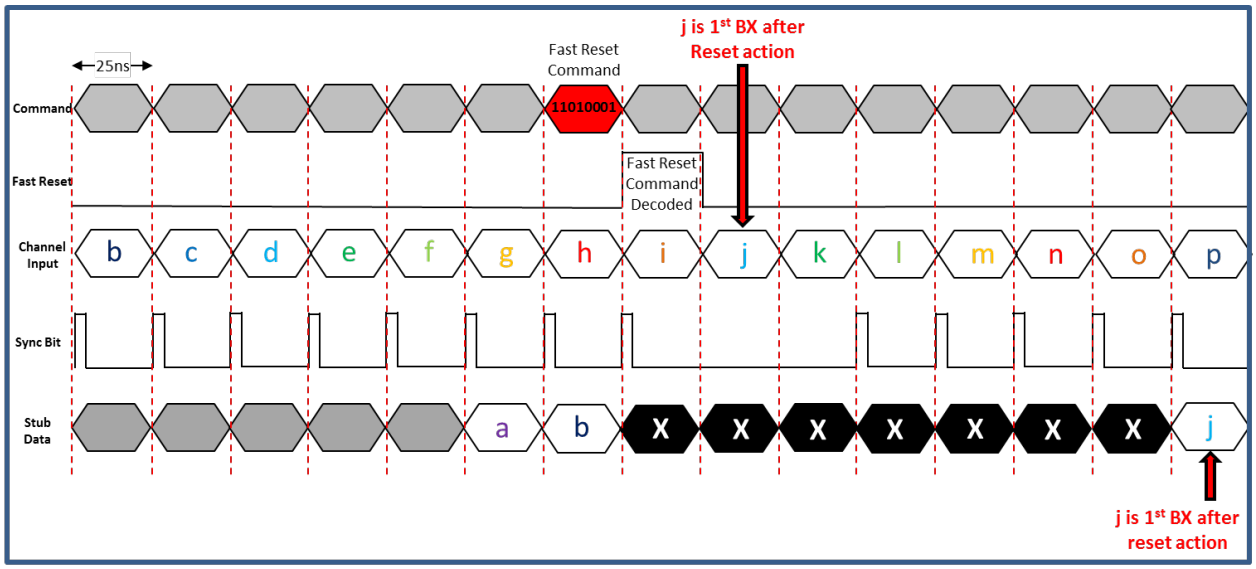

Figure 22 — Illustrating the timing of the *Stub Data* output following a *Fast Reset* command, for low numbered DLL settings.

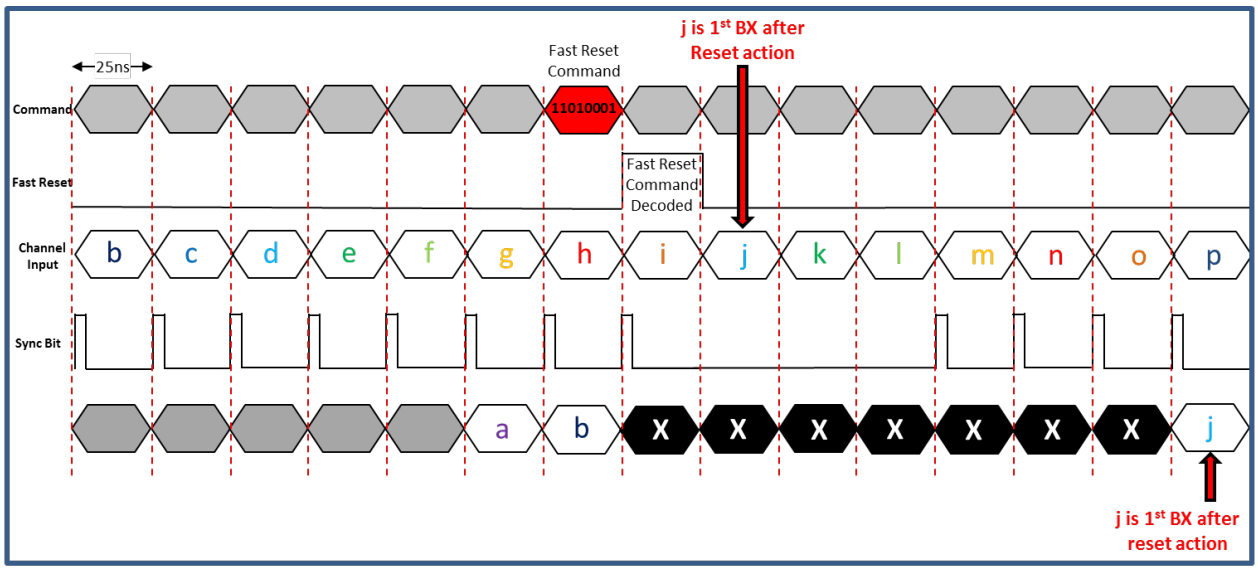

Figure 23 — Illustrating the timing of the *Stub Data* output following a *Fast Reset* command, for high numbered DLL settings.

► **NOTE:** As shown in figures 22 and 23, the SYNC bit will stop for between two to three BX due to the *Fast Reset* command. The time for which the SYNC bit disappears is determined by the phase shift of the 40MHz clock, as programmed in the DLL settings, but this should be constant and predictable for each setting. The SYNC bit will be present for at least three BX prior to the first data appearing at the outputs.

Interfaces

Fast Control Interface

In addition to the *I²C Slow Control Interface,* there is a *Fast Control Interface* operating serially at 320MHz via a differential SLVS input. Running at the 320MHz clock frequency allows eight bits of serial data to be input within a Bunch-Crossing period of 25ns. These eight bits of data have pre-defined functions, as set out in Table 1 below.

Fast Command	B7	B6	B5	B4	B3	B2	B1	B0
Fast Reset	1	1	0	1	0	0	0	1
Trigger	1	1	0	0	1	0	0	1
Test Pulse Trigger	1	1	0	0	0	1	0	1
Orbit Reset	1	1	0	0	0	0	1	1
Orbit Reset & Fast Reset	1	1	0	1	0	0	1	1
Orbit Reset & Trigger	1	1	0	0	1	0	1	1
Orbit Reset & Test Pulse Trigger	1	1	0	0	0	1	1	1

Table 1 — Fast Control Interface commands and their corresponding serial codes

The CBC3.1 detects the **110** synchronisation pattern on bits <7:5> and uses this to produce the on-chip 40MHz reference clock. In the event that this synchronisation pattern is not received, the on-chip 40MHz clock will freeze until the synchronisation pattern is resumed. Two flags are made available via the I²C interface, which report whether synchronisation was lost and the current synchronisation status. The condition of these flags can be reset by sending a *Fast Reset* command.

Following on from the synchronisation pattern are the four bits that define which of the four fast commands is being sent to the CBC3.1. Each command has an associated bit in the serial data, for example *Fast Reset* is identified by a logical 1 in the bit 4 position. The only command that can be sent at the same time as other commands is *Orbit Reset*. All other command combinations are forbidden and will result in an error flag, which for the purpose of de-bugging can be read via the I²C interface. Forbidden command combinations will be ignored and no command signal will be generated on-chip. The CBC3.1 Commands and their functions are set out in table 2 over page.

The final bit of the frame (bit 0), is set to 1 to prevent the 110 synchronisation pattern being confused with a *Test Pulse Trigger* in combination with an *Orbit Reset*. In normal operation no other combination of fast control bits should ever reproduce the 110 sync pattern.

Fast Command	Bit	Function
Fast Reset	4	Reset to the *Hit Detect* circuits, the *Pipeline Control Logic* & the *Output Data Serialiser*, the *L1 Counter and the Data Assembly Logic*. It also resets the flags. The Fast Reset pulse is synchronous with the rising edge of the 40MHz on-chip clock, and is $1^1/_2$ clock cycles wide.
Trigger	3	This initiates a trigger signal to the *Pipeline Control Logic* . The decoded *Trigger* pulse is output to the Pipeline in the following 40MHz clock cycle.
Test Pulse Trigger	2	This initiates a test pulse from the *Test Pulse Generator* circuit.
Orbit Reset	1	Reset to the *L1 Counter*. This is the only command that can be sent in parallel with the three other commands.

Table 2 — Fast Control Interface commands and their functions

Figure 24 below illustrates the timing for the serial command data packet and the signals generated on-chip. The direction of time flow shown is **left** to **right** (i.e. bit 7 arrives before bit 6). The input serial command data packet is aligned with the rising edge of the 320MHz clock and is therefore sampled on-chip using the falling edge of the 320MHz clock. The input serial command data must settle at least 350ps before the falling edge of the 320MHz clock, as seen by the chip. There is no fine timing adjustment included on the CBC3.1, so the timing of these signals must be tuned by the transmitter.

The first full 40MHz clock cycle will start as soon as the 110 sync pattern is received, but commands are synchronized with this clock and as such will be output from the *Fast Control Interface* in the 40MHz clock cycle after they were received. Each command will last for one 40MHz clock cycle unless sent repeatedly. The only exception to this is the *Fast Reset* command which is one and a half 40MHz clock cycles wide in order to make sure that circuits using a clocked reset are correctly initialized.

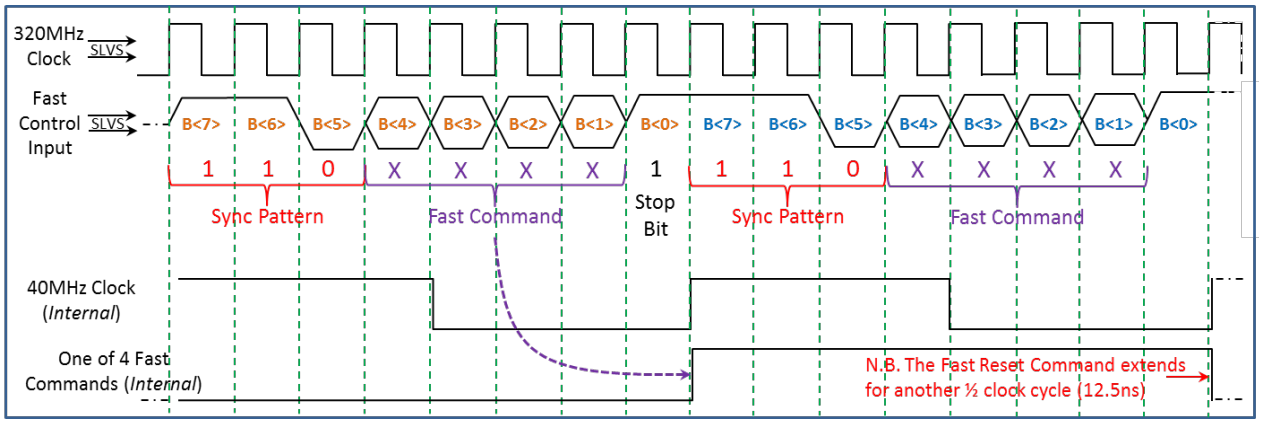

Figure 24 — Illustration of the Fast Control Interface timing.

The output data packet framing (figure 21) is synchronized to the internal 40MHz clock derived from the 110 sync pattern, however the timing of the data arriving at the data packet assembly and transmission block (figure 1) is determined by the choice of DLL setting in the programmable delay 40MHz region. Consequently there is a possibility that the appropriate choice of DLL setting for optimum comparator output sampling will be close to the transition point between output data packet frames. In such a critical

timing region any jitter on the clock could result in an unstable situation, and there could be differences between neighbouring chips on the same hybrid due to process variations affecting speed. To avoid this condition it is possible to coarsely adjust the timing of the 40MHz clock by selecting an internal delay path for the serial fast command data. This delays the decoding process by a choice of one or two 320MHz clock cycles relative to the start of the command data packet. The choice of delay used can be programmed via the I²C interface (FCI delay, table 33). Shifting the internal 40MHz clock phase results in a corresponding shift in the transition point between output data packet frames, and the optimum comparator output sampling time will in turn be shifted away from the critical timing region.

I²C Slow Control Interface

This is the interface between the external system and the control registers that configure how the CBC2 will operate. An I²C slave architecture is implemented.

Figure 25 shows a typical I²C configuration, with one master controlling several slaves. In addition to sending and receiving data (*SDA*), the master also generates the clock (*SCL*). I²C output drivers are open-drain, requiring pull-up resistors. The maximum value of resistor that can be used is determined by the clock frequency and total capacitance on the I²C bus. In addition it must be remembered that for small values of resistor, the open-drain outputs will not be able to pull the signals all the way down to 0V.

Since all devices share a common bus, only one may use it at a time.

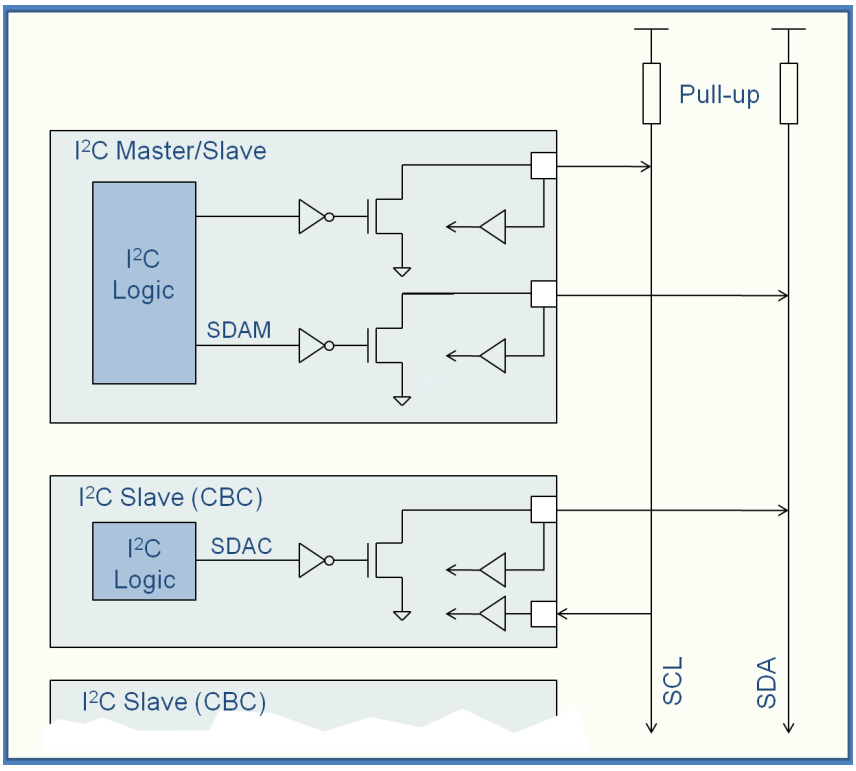

Figure 25 — I²C Master/Slave Configuration

The CBC3.1 has 5 address inputs, which compose the 5 LSBs of the address. I²C chip addresses must consist of 7 bits, so the remaining CBC3.1 address bits have been internally wired to 1 and 0. Therefore, when sending an I²C command to an individual chip, the 2 most significant bits of the chip address must be set as 1 and 0. For example, to address a chip with the 5 address inputs set to 00000, the I²C command must have a chip address of 1000000.

Figure 26 shows the signals for an I²C "write" transaction. When the I²C is inactive, both the clock and data buses are high. A transaction must start with an I²C start condition. This occurs when the data line (*SDA*) goes low when the clock (*SCL*) is high. This is initiated by the master on *SDAM*. During this time the slave is inactive (*SDAC*). All data must then change only when the clock is low.

The master sends the address of the slave it wants to communicate with, which in this case is 1000001, followed by a "0" which indicates it wants to write information to the slave. It then releases *SDA* by taking *SDAM* high and the slave acknowledges receipt of the address by taking *SDAC* (and hence *SDA*) low.

The master then sends the address of the register it wants to write to (in this case 00000001), and the slave acknowledges. The last piece of information is the data that is to be written into the register (00001110). The slave acknowledges, and the master sends the I²C stop condition (*SDA* goes high when clock is high).

Every register on any chip can be written to in this way by supplying the relevant chip and register address. The chip address 1111111 is reserved for addressing all chips connected to the same bus at the same time.

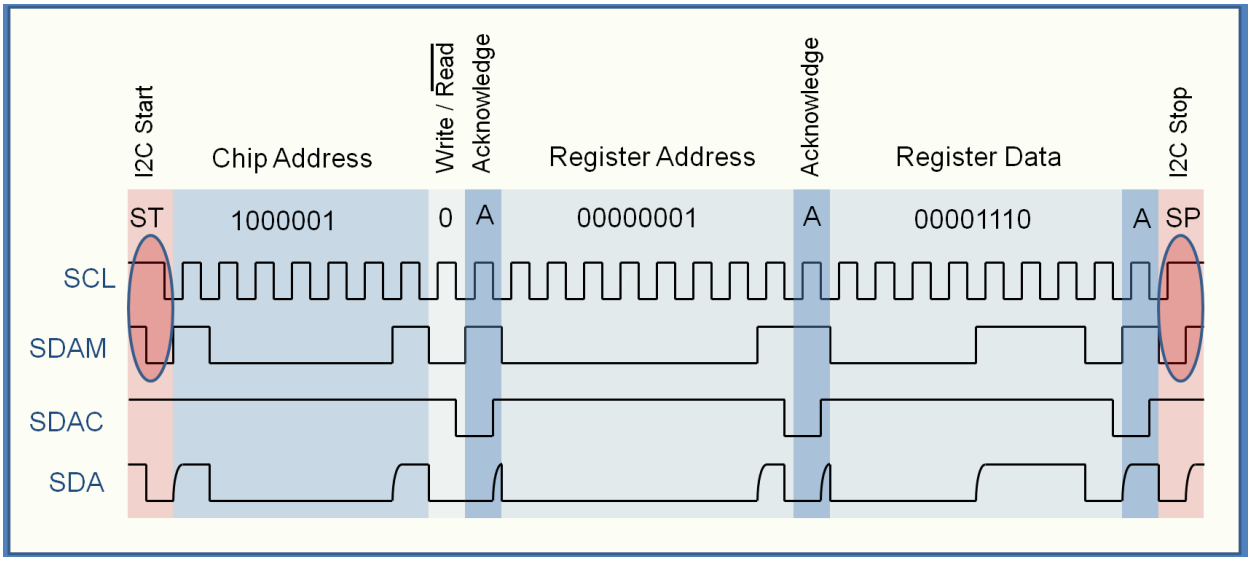

Figure 26 — I²C Write Transaction

Figure 27 shows the format for read from a register as well as writing to it. As the write operation has already been described this section will deal with the read transaction.

To begin a read transaction, the register on the chip you want to read from has to be addressed. This is done by firstly sending a start condition and addressing the chip with the write/read bit set to 0 (write

mode). Then the register that is to be read is addressed and a stop condition is sent. This has set up the internal I²C interface to address the relevant register.

Secondly, beginning with a start condition, the chip has to be addressed for a second time, but in this case, the write/read bit is set to 1 to indicate a read transaction. The register data is then loaded onto an internal bus, the master releases the *SDA* bus and the slave, detecting a read condition, outputs the register data over the *SDA* bus. The slave releases the bus, and the master acknowledges.

All control register are written to and read from using the I²C interface. The next sections list all available control registers.

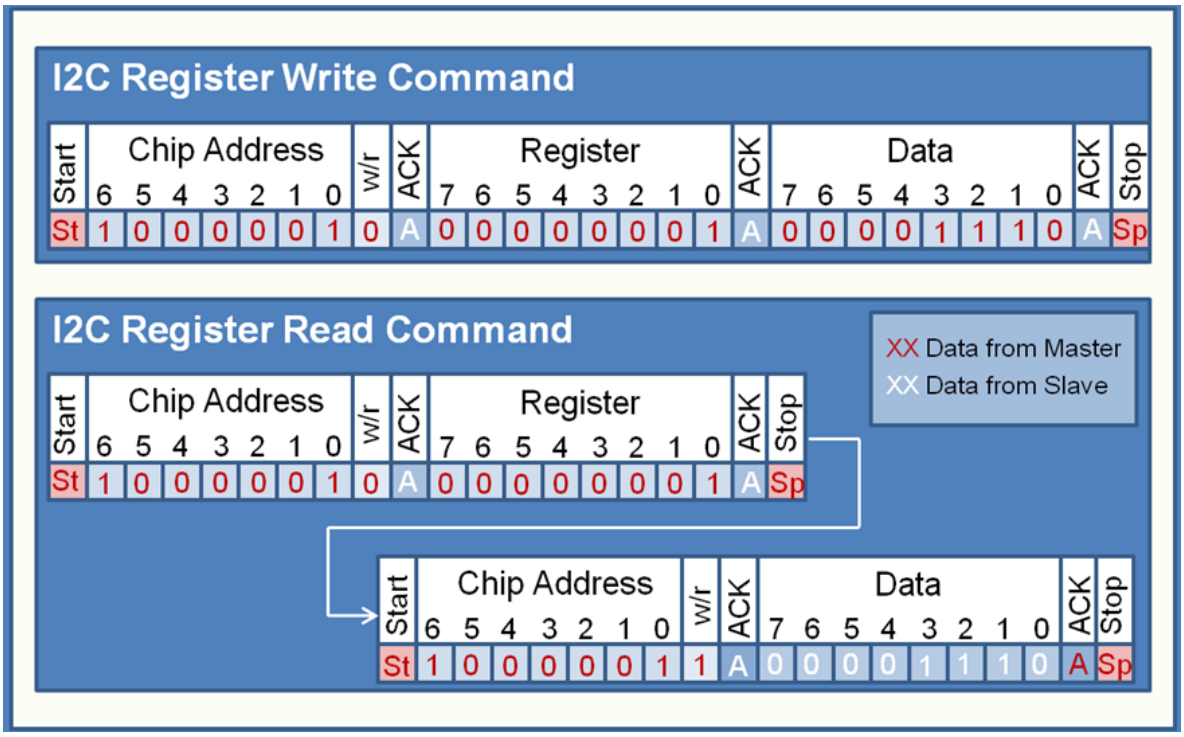

Figure 27 — I²C Write and Read commands.

Repeated Start Condition: The interface will respond to repeated start conditions without the need for a stop condition in between. Each time a start condition is detected, the interface resets itself to expect a chip address as the next command.

Hang-up Reset: If the interface hangs up, preventing commands being sent from the master, it will automatically reset. This hang-up condition is recognised by the SDA output being held low for 9 consecutive clock cycles. The reset is held active for one I²C clock cycle, starting on the rising edge of the 10th clock.

Hard Reset

An active high CMOS input signal (*RESET*) is provided to reset the chip to its *power-on* default state. Some on-chip logic employs a synchronous reset, so to ensure a complete reset, the Hard Reset should overlap a rising clock edge. Figure 28 shows a suggested timing for this reset, and figure 29 shows the regions of the CBC3.1 that are affected.

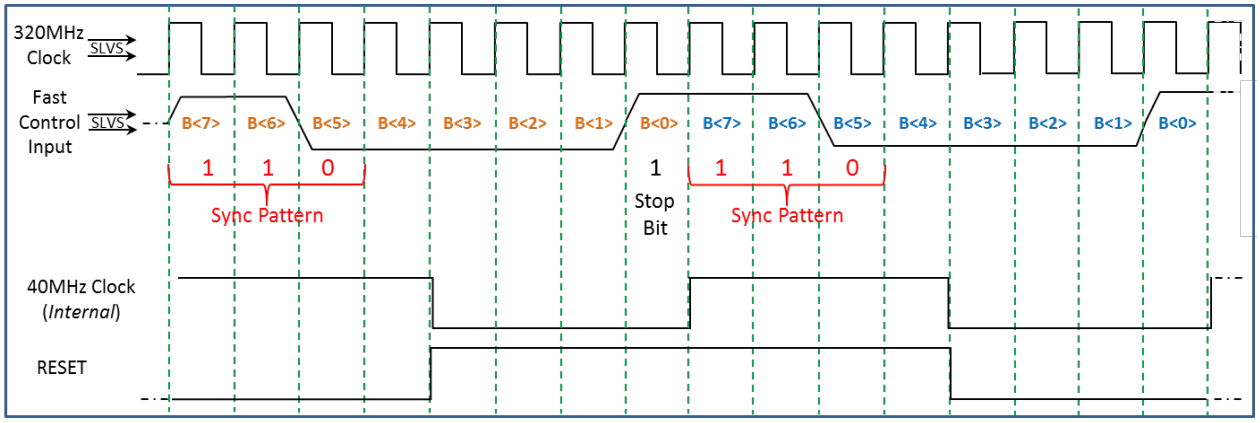

Figure 28 — Suggested timing for RESET.

► **NOTE:** In order to restore the analogue control voltages of the 40MHz DLL to their default levels, a RESET pulse width of at least 400ns is required. This is an initial condition and is not equivalent to placing the DLL into a locked state. Upon removal of RESET, the control voltages will track to different values as the DLL tracks into its delay locked state. Once lock is achieved, sending a one clock cycle wide RESET pulse will cause the control voltages to transition a little way towards their default state (disrupting the DLL's lock), but the pulse is too short for the default state to be reached, and they should quickly track back to the locked state values on removal of RESET.

► **NOTE:** Any application of the RESET will disturb the DLL's lock, and the user must allow time for lock to be re-established. This is likely to be particularly true in the case of a long (400ns) RESET pulse width, where the control voltages are likely to move further away from the locked state than with short RESET pulses.

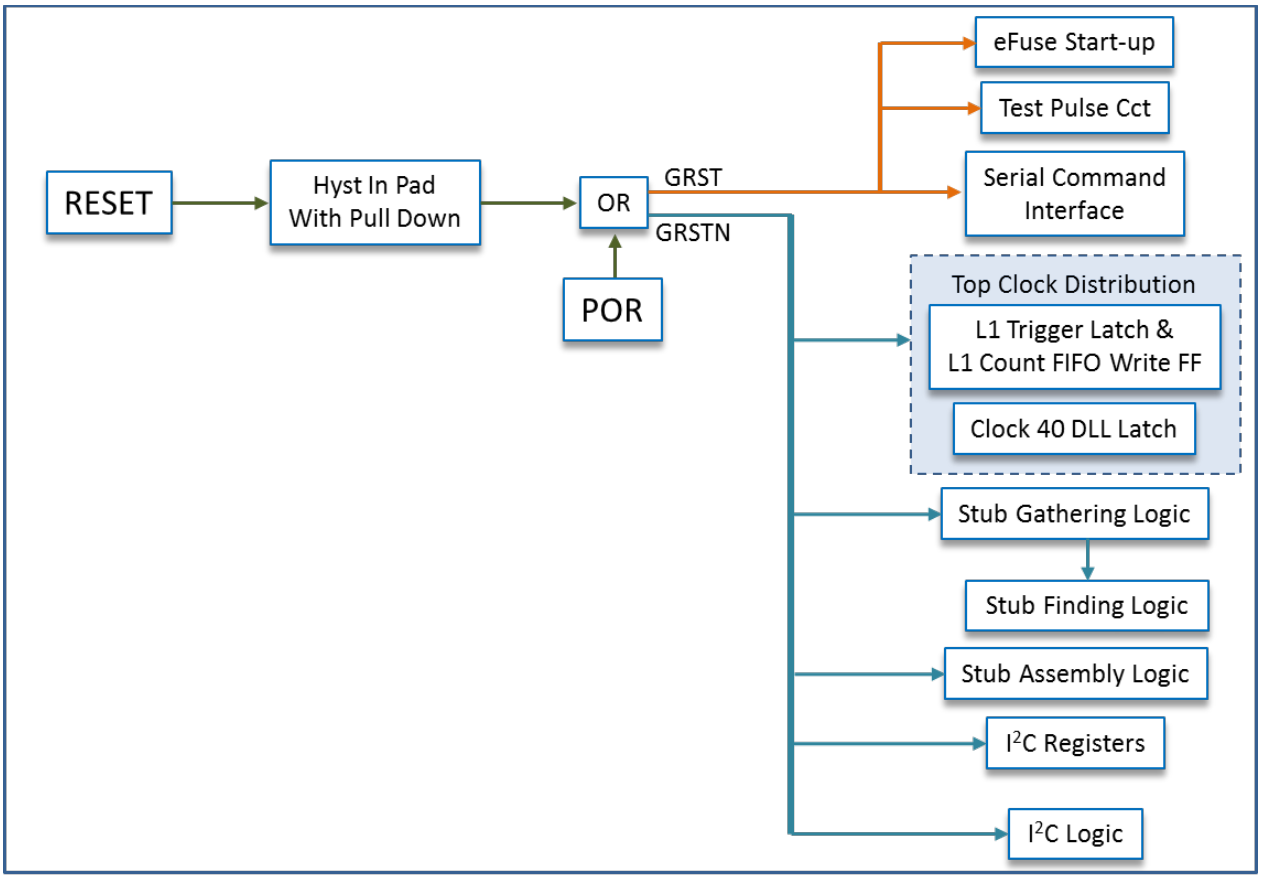

Figure 29 — Map of circuits affected by the hard reset.

320MHz Input Clock & Fast Command Interface

There is a single 320MHz clock input to the CBC3.1, and a single serial input for the Fast Control Interface, operating at 320 Mb/s. These are both SLVS type (*Scalable Low Voltage Signaling*) differential inputs. There is no termination on-chip, so a 100Ω termination resistor (per SLVS input) must be placed on the module substrate near to the differential pads on the chip. A 100Ω termination at the receiving end creates a differential voltage of 200mV approximately centered on 200mV. See figure 30.

Fast Data Outputs

For the *Stub Data* and *Triggered Data* there are six fast data outputs running at 320 Mb/s. These are all SLVS type (*Scalable Low Voltage Signaling*) differential outputs. These outputs have a programmable drive current with a default setting of 2mA. There is no termination on-chip, so a 100Ω termination resistor (per SLVS output) must be placed on the module substrate near to the receiver. A 100Ω termination at the receiving end creates a differential voltage of 200mV approximately centered on 200mV. See figure 30.

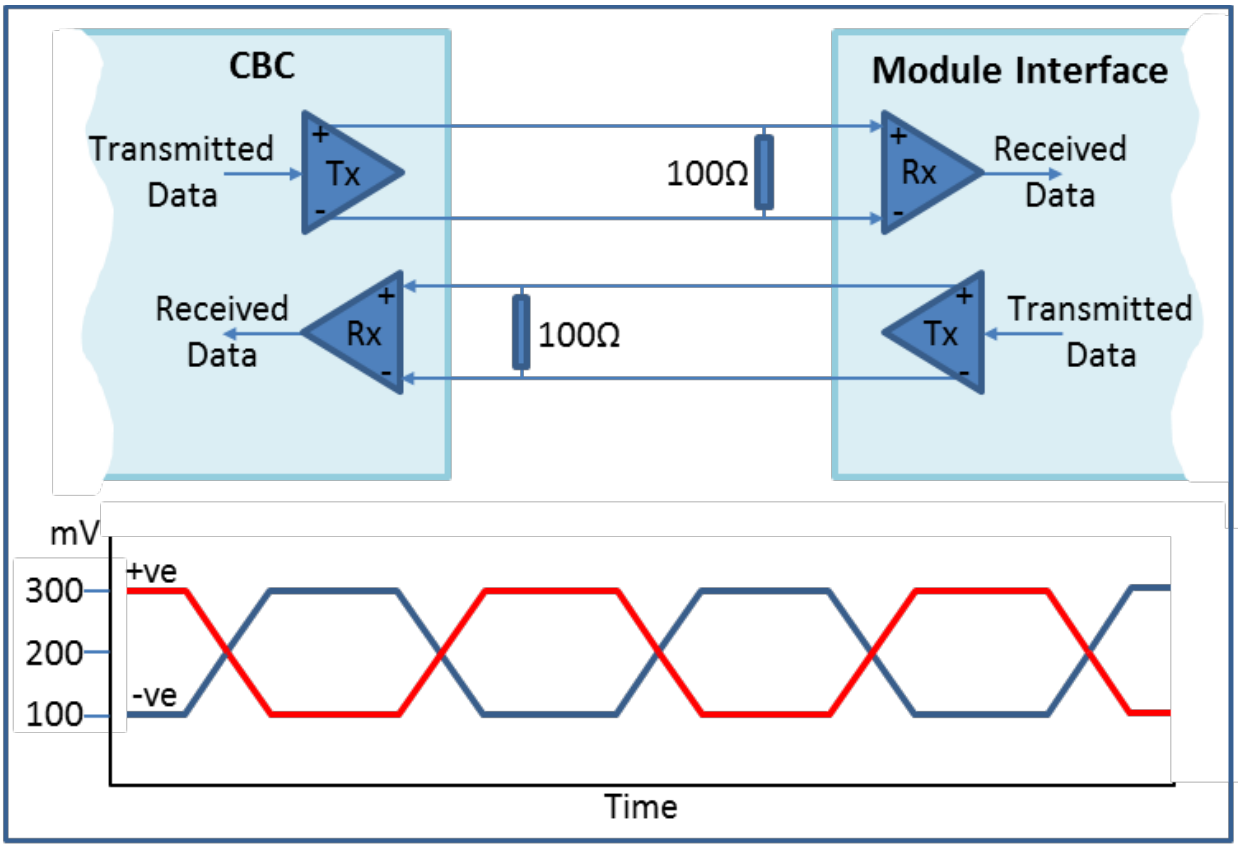

Figure 30 — SLVS I/O Format.

Inter-chip Hit Signals

As with the CBC2, signals must be passed between chips for the correlation to be performed. Unlike the CBC2, these signals originate from the *Hit Detect* circuit and not the output of the Cluster Width Discrimination logic. This requires extra logic, but reduces the number of inter-chip signals required. A total of 60 pads are required for the inter-chip signals. Each CBC3.1 outputs the hits from the twelve highest numbered channels assigned to the *Correlation Layer* (Top sensor), along with the two highest numbered channels assigned to the *Seed Layer* (Bottom sensor). Consequently each chip also inputs these same signals from its neighbour for comparison with hits from its own lower numbered channels. Similarly, each CBC3.1 outputs the hits from the thirteen lowest numbered channels assigned to the *Correlation layer*, along with the three lowest numbered channels assigned to the *Seed layer*. Each chip will input these same signals from its neighbour for comparison with hits from its own higher numbered channels. Figure 31 illustrates the inter-chip signals.

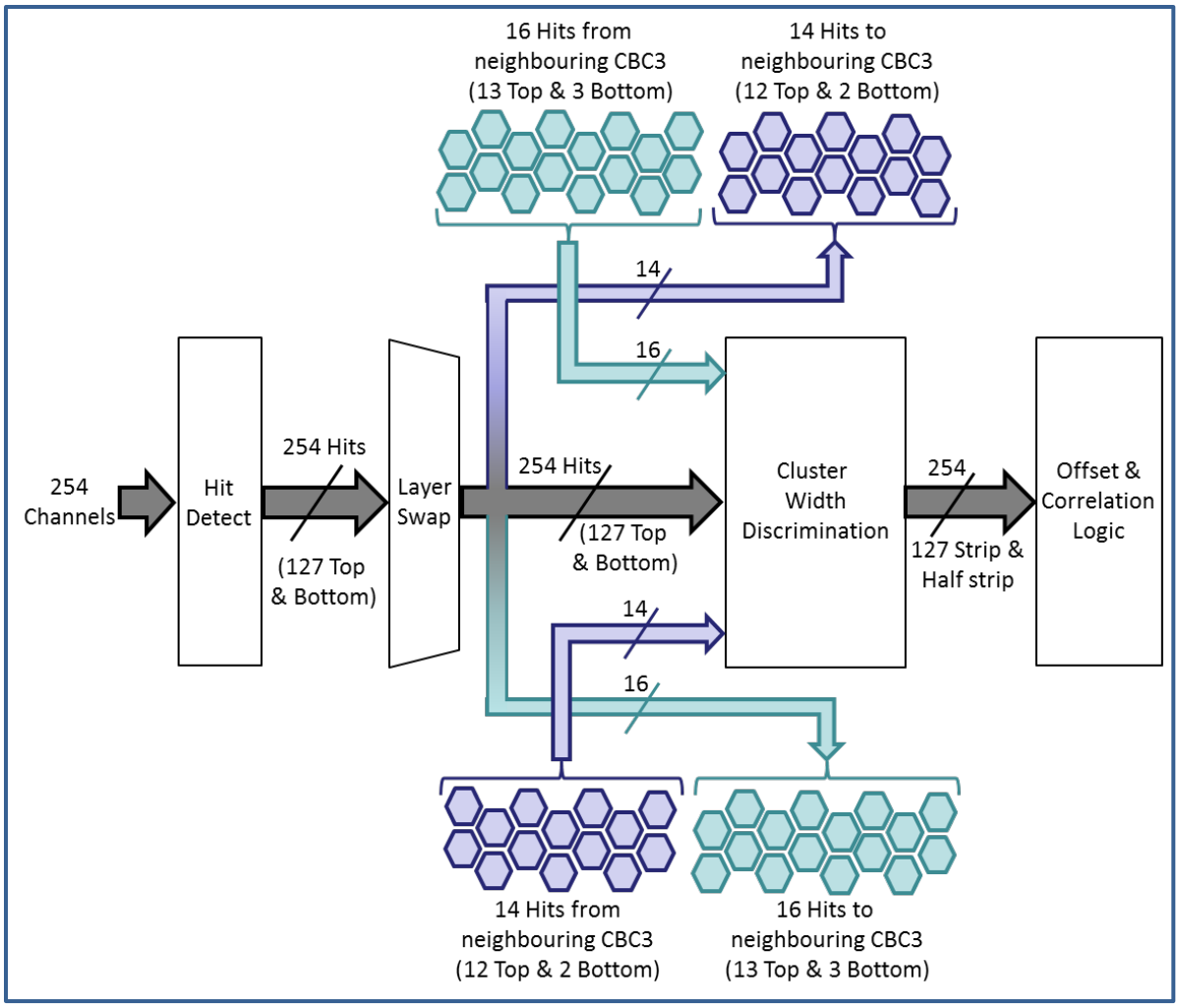

Figure 31 — Illustration of the inter-chip signals required for finding stubs.

These are single-ended CMOS inputs and outputs. The inputs are pull-down type so that off-chip termination is not required for the CBC3.1s at the module ends. The pull-down resistor value is 1.2MΩ and there is a 250Ω ESD resistor in series, as shown in figure 32.

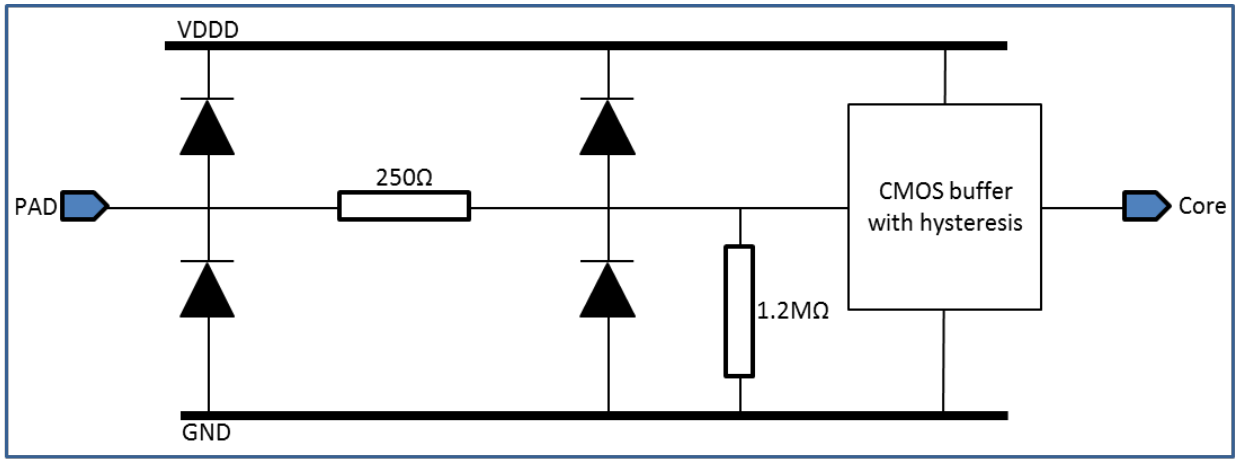

Figure 32 — Illustration of the inter-chip input pad structure.

Programmable e-Fuses

There are a number of electronically programmable fuses (*e-Fuses*) included on the CBC3.1 ASIC.

Chip ID

One set of 19 e-Fuses are used to program each CBC3.1 with a unique identifier code. This programming will be carried out at the time of wafer level testing using probe pads specific to the purpose. The value to be programmed is loaded into an on-chip register using the I^2C interface.

Bandgap Trimming

The CBC3.1 employs a PMOS transistor referenced bandgap circuit instead of the diode referenced version used on earlier CBC variants. This circuit is more robust to radiation effects, but the circuit is susceptible to manufacturing process variations and it is therefore necessary to include on-chip trimming of the reference. This will be a one-time operation performed at the time of wafer level test. A set of 6 e-Fuses are available to trim the bandgap for each CBC3.1. The value to be programmed is loaded into an on-chip register using the I^2C interface.

e-Fuse Programming

The fuses are programmed by passing a high current through the fuse element for a prescribed time period. The level and duration of the fuse blow current is critical. A 3.3V supply input (*Burn33*) is provided as the current source for the fuse programming transistor. This transistor is switched on/off by an on-chip circuit designed to shift levels between the externally supplied 1.2V programming logic signal (*FuseProgPulse*) and the 3.3V programming transistor.

Thick oxide transistors are used for the programming transistors, so the level of the source voltage can be between 3.3V and 3.7V ($\pm$100mV). The width of the programming pulse, and hence the time period that the programming current is applied for, is determined by off-chip logic. This pulse width should be between 0.18ms and 1ms, with a pulse rise time of less than 200ns. The programming current will be between 10mA and 13.5mA

Both the *Burn33* and the *FuseProgPulse* pads are intended for use during the wafer-level test phase only, and as such there are no C4 bump-bond pads to connect these inputs on the module. An 88kΩ pull-down resistor is present on the *FuseProgPulse* input to tie the signal inactive when not in use. The *Burn33* input is resistively tied to 1.2V internally when the chip is bump-bonded.

Test Features

Charge Pulse Generator

The CBC3.1 ASIC retains the *Charge Pulse Generator* circuit used on the CBC2 design, a block diagram of which is shown in figure 33.

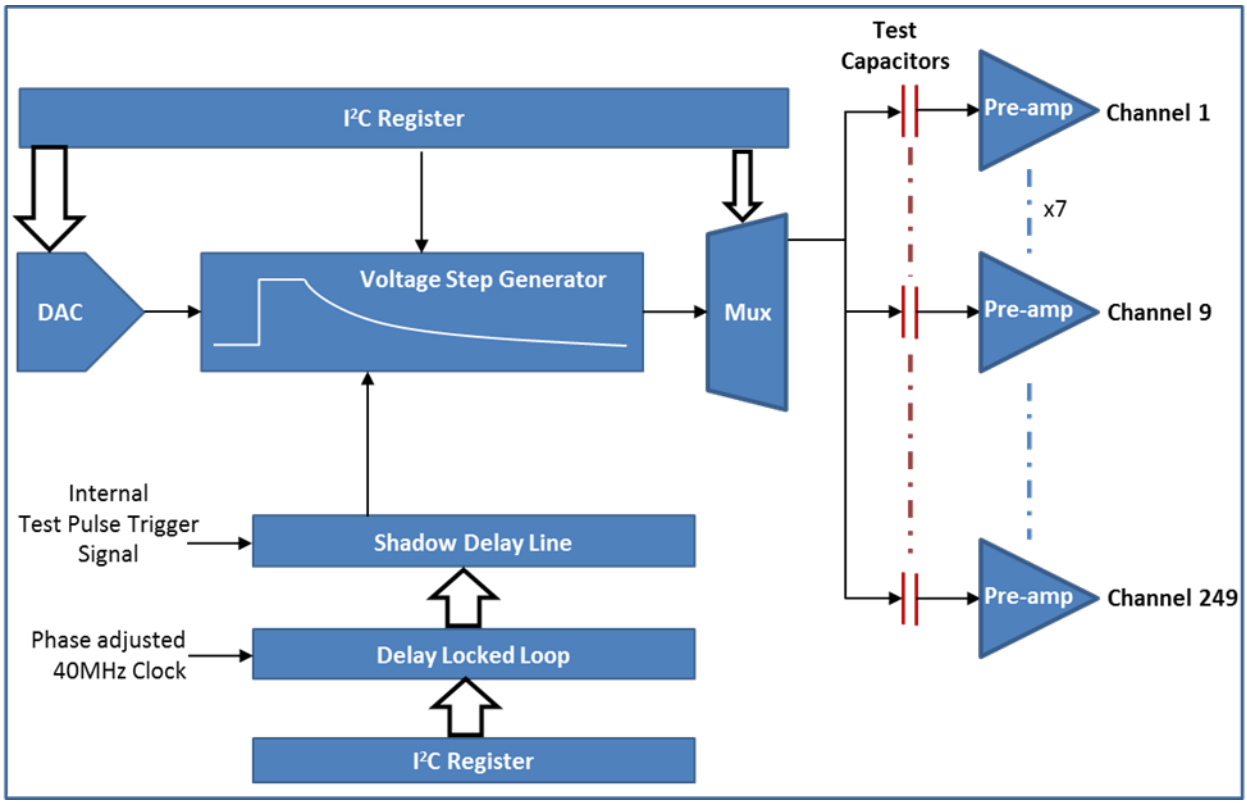

Figure 33 — Block diagram of the Charge Pulse Generator

This circuit is used to mimic the charge signal from the sensor by applying a voltage step on a small test capacitor connected to each pre-amplifier input. The circuit consists of a programmable delay line, coupled to a *Voltage Step Generator*. The voltage step is initiated by sending a *Test Pulse Trigger* command via the *Fast Control Interface*. This produces an internal signal lasting for one 40MHz clock cycle, which in turn produces a *Test Pulse Request* signal lasting one 40MHz clock cycle. It is the rising edge of the *Test Pulse Request* signal which triggers the voltage step (see figure 34). The voltage step is therefore initiated at least one 40MHz clock cycle after the sending of the *Test Pulse Trigger* command.

Timing of the *TestPulseRequest* signal can be adjusted in 1ns subdivisions of a 40MHz clock cycle using a dedicated DLL. This then determines the time within a clock cycle at which the voltage step is applied to the test capacitors on the pre-amplifier inputs. The variance of each subdivision is specified as being less than +/- 5ps. This is a separate DLL to the one used for adjustment of the 40MHz clock.

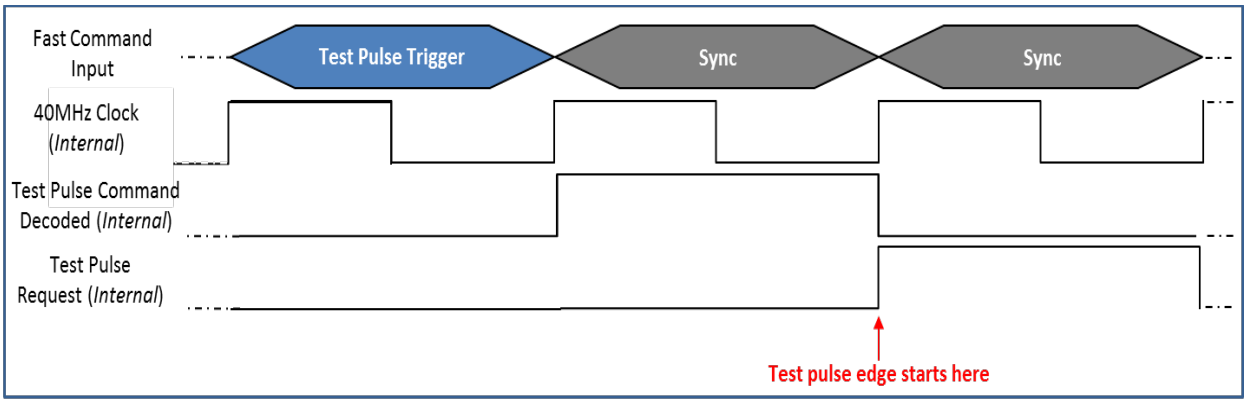

Figure 34 — Test pulse timing.

The *Voltage Step Generator* uses a programmable 8 bit DAC to adjust the amplitude of the voltage step. This DAC uses the analogue supply voltage as its reference, so a 1.1V analogue supply will provide 4.3mV resolution, which with a 20fF test capacitor gives a 0.086fC charge pulse resolution and a full range of 22fC. Although the input charge is generated by applying the voltage step to a fixed capacitance, these capacitors have a total tolerance (3σ) of 25% due to process dependent variation, which will affect the accuracy of the voltage to charge conversion. The amplitude of the charge signal is programmed by setting the DAC value in a register that is written to using the I^2C interface. The voltage step polarity, and hence the charge polarity can also be programmed this way.

Following the application of a test pulse, the *Voltage Step Generator* must be reinitialized ready for the next voltage step when required. The initialization will begin automatically four 40MHz clock cycles after the *Test Pulse Trigger* command is detected, and lasts for approximately 10μs. No further *Test Pulse Trigger* commands should be sent during this time.

Due to the large number of channels, it is not practical to stimulate single channels, so the output of the *Voltage Step Generator* is multiplexed onto one of eight tracks that deliver the voltage step to the group of test capacitors connected to that line. Each line supplies groups of thirty two capacitors arranged as shown in figure 35 over page. This allows channels on the Seed and Correlation layers to be stimulated at the same time. The group of channels to be stimulated is selected by programming 3 bits of a register written to using the I^2C interface. By setting another register bit, the user can also choose whether to ground the test inputs of all of the unused test capacitors.

In order that the *Charge Pulse Generator* timing tracks the phase adjustments made to the 40MHz clock for BX timing, the *Charge Pulse Generator* DLL uses the phase adjustable 40MHz Clock for its timing. At start up, the *Charge Pulse Generator* DLL circuit requires 40 to 50 clock cycles to lock to the 40MHz clock. Once lock is established the *Test Pulse Trigger* command can then be sent at the user's discretion.

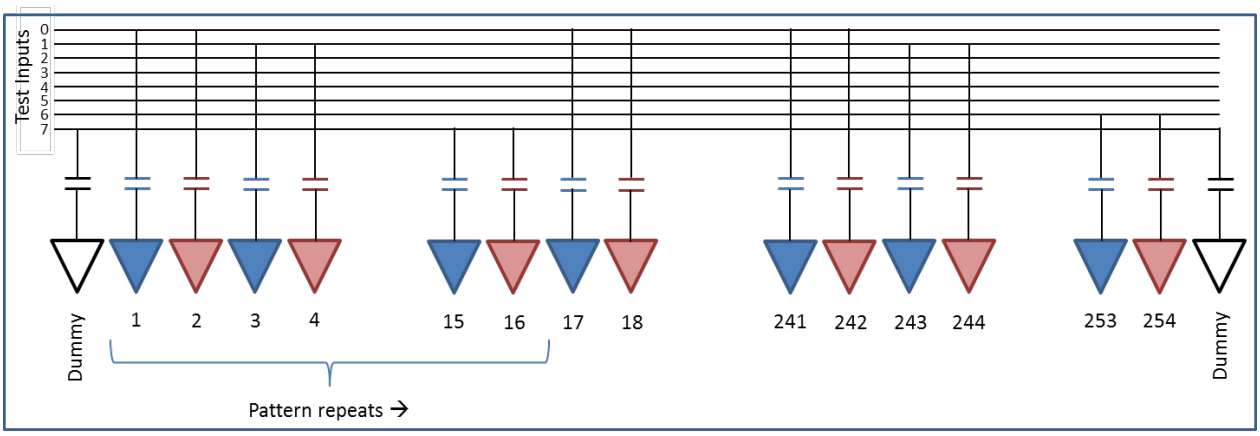

Figure 35— Test pulse capacitor arrangement

It is possible to operate the CBC3.1 ASIC at lower clock speeds simply by reducing the speed of the 320MHz clock and consequently the Fast Control Input. This will have the effect of reducing the 40MHz clock domain to one eighth of whatever clock frequency is used in place of the 320MHz clock. However neither of the DLL circuits will work if the 40MHz clock domain drops below 35MHz.

To allow for *Charge Pulse Generator* circuit operation with a 40MHz clock input frequency in place of the default 320MHz clock, the *Charge Pulse Generator* circuit's DLL can be clocked directly with the 320MHz input. This mode is selected by setting the *TestPulseClockSelect* bit in one of the I^2C programmable registers (table 32).

Analogue Bias Multiplexer

The CBC3.1 includes a 17:1 analogue multiplexer to allow monitoring of the analogue bias signals (table 18). Selection of the bias signal to be interrogated is achieved by writing the desired five bit address to a register via the I^2C interface. The multiplexer directly connects the on-chip bias to the pad without buffering.

40MHz Clock Output

A tristate test output is available for observing the 40MHz clock. The output of the DLL is buffered and then gated through pass-transistors to both bump and wire bond pads. When selected, the output of the DLL can be monitored on the output pads, otherwise the output floats. Depending on the programming of the DLL, this output can either be the original 40MHz clock as generated by the Fast Command Interface, or a phase shifted variant as used by the Hit Detect and other front end logic.

Nearest Neighbour IO Test Features

CBC2 and CBC3 lacked the ability to test the connectivity of the Nearest Neighbour IO pads when testing single chips. This meant that wafer level tests could not check 100% of the chip functionality, leaving an uncertainty about the yield, with the potential to result in bad modules. To overcome this deficiency, additional I^2C registers have been added along with logic, enabling these inputs and outputs (IO) to be validated.

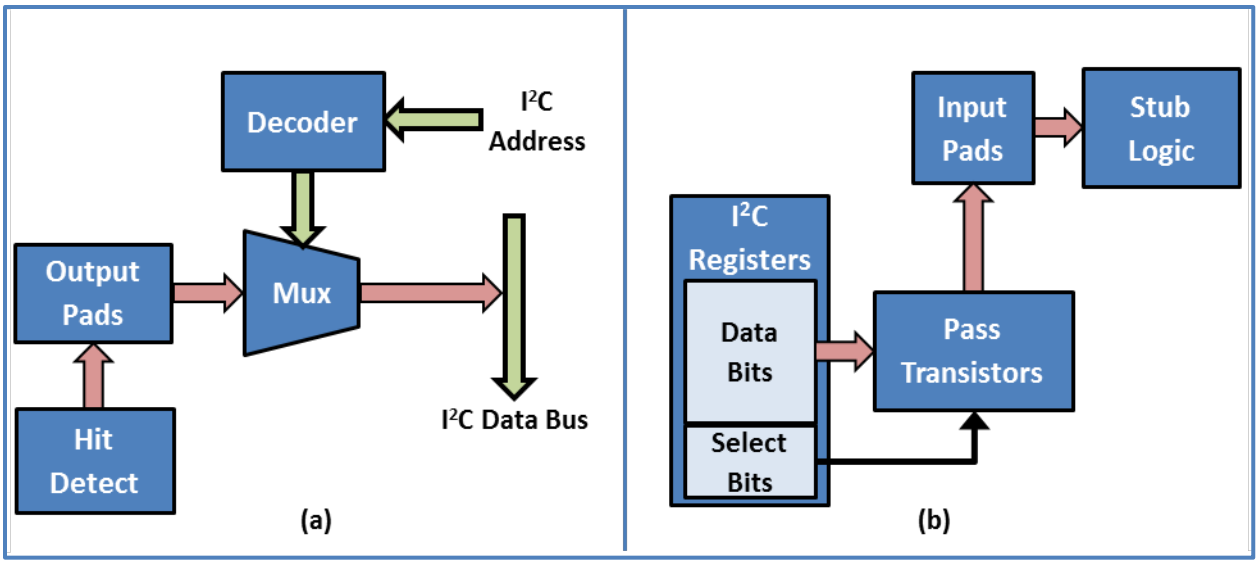

Figure 36 — (a) Test feature for outputs to neighbouring CBC3.1, (b) Test feature for inputs from neighbouring CBC3.1

Figure 36 (a) and (b) illustrate the two test features. In (a), the output lines from the relevant Hit Detect circuits are connected to the Nearest Neighbour output pads, which in-turn are connected to a multiplexer. The latter is controlled by the I^2C addressing scheme such that the outputs will appear on the I^2C Data Bus when an I^2C read is sent to that I^2C address. The scheme does not use I^2C registers and there is no latching of the outputs, other than that provided by the Hit Detect circuits. The outputs are taken directly from the pads (after the output drivers) and buffered onto the I^2C Data Bus.

In (b), the outputs from an I^2C register can be configured to drive the Nearest Neighbour input pads. Pass-transistors connect the register outputs directly to the input pads such that the data will pass through the input pad drivers on its way to the Stub Finding Logic, in the same way as the signals from the nearest neighbours would. The pass-transistors are enabled by two bits in the I^2C register. Their default state is 0, which disables the pass-transistors and disconnects the register outputs from the pads. The AND of the two bits is used for enabling the pass-transistors, as a protection against Single Event Upset (SEU). The pass-transistors are made deliberately resistive to counter any radiation induced leakage current issues in the switches.

I^2C registers 81 to 84 provide control/access for the test features of the Nearest Neighbour IO pads TopOut_B<12:1>, TopOut_A<2:1>, TopIn_B<13:1>, and TopIn_A<3:1>.

I^2C registers 85 to 88 provide control/access for the test features of the Nearest Neighbour IO pads BotOut_B<13:1>,Bot_A<3:1>, BotIn_B<12:1>, and BotIn_A<2:1>.

Powering

Unlike the CBC2, there is no DC-DC converter on the CBC3.1, however the scheme of separating the analogue and digital power domains by generating the analogue VDD using a Low Dropout voltage regulator circuit (*LDO*), is still employed. The *LDO* is powered from the same power supply used for the digital power domain VDDD, and as such, the supply to the analogue power domain will be lower than the digital. The chip has been designed to be fully functional for VDDD values of 1.2V +/- 10%. The actual value of the analogue supply rail (VDDA) will depend on the choice of the bandgap voltage (VDDA is twice the bandgap voltage) which can be programmed and fixed using efuses at wafer probe time. The analogue front-end has been designed to function correctly with an analogue supply as low as at 1V.

The supply to, and the analogue output from the LDO, are connected to dedicated pads that are independent of the VDDD pads. Although the supply pads will in practice be connected to the same module power layer as the VDDD pads, keeping the pads separate allows for greater flexibility with the design.

The digital power consumption for VDDD = 1.2V has been measured to be approximately 33mA, equivalent to 160µW/ channel. Total power for the analogue and digital sections will depend on the user programmable bias register settings, but should be nominally equivalent to 510µW/channel.

Operating Temperature

Text to be provided

I²C Programmable Registers

Page Function

Analogue bias settings and the various function options of the CBC3.1 are configured by programming a series of 8 Bit registers via the I²C interface. The ASIC requires more than the 256 register addresses normally available with an I²C 8 Bit address, so a paging system is employed to increase the address space. With this system addresses can be used twice, with a page Bit determining which of two registers

is actually being addressed. Register address 0 is considered page free so that it can always be written to. This register includes a bit location (Bit 7) that can be set to 1 or 0 to determine which of the two pages are being addressed. The default is 0 for page 1 which includes all of the main bias configuration registers. This system gives the ASIC an address space with 511 addresses.

The page set in register address 0 will remain active until it is changed by another write to register address 0. A reset will return register address 0 to its default condition.

► **NOTE**: Bit 7 is the first to be loaded in time sequence during I^2C transfer. This is usually the MSB where groups of bits form command words.

Register 0 — Comparator Settings, Beta Multiplier Reset Polarity, Trigger Latency & Page selection

Address: b00000000		
Bit	**Function**	**Default**
7	"Page" Bit (0 writes to page 1)	0
6	Comparator polarity (0=holes, 1=electrons)	1
5	Comparator hysteresis Bit 3 (MSB)	1
4	Comparator hysteresis Bit 2	1
3	Comparator hysteresis Bit 1	1
2	Comparator hysteresis Bit 0 (LSB)	1
1	Beta Multiplier Reset Polarity	0
0	Bit 8 (MSB) of the Trigger Latency (9 bits in total)	0

Table 3 — Register Address 0 default settings

Due to its special status as the register used to define the page being addressed, Address 0 is the only register not assigned a page.

While the ability of the pre-amplifier to be programmed for either polarity of input signal has been removed, the ability to swap the polarity of the comparator circuits was retained as a precaution. This is a global function applied to all comparators on a chip.

The Comparator Hysteresis bits control the amount of hysteresis used in the comparator circuits, and are by default set to give the maximum hysteresis. This is a global function applied to all comparators on a chip.

Bit 1 allows the reset signal to the Beta Multiplier to be inverted. This has no practical use in normal operation and was included as a precaution.

Page1 — Register 1: Trigger Latency

Address: b00000001		
Bit	**Function**	**Default**
7	Trigger Latency Bit 7 (see *Register 0* for MSB)	1
6	Trigger Latency Bit 6	1
5	Trigger Latency Bit 5	0
4	Trigger Latency Bit 4	0
3	Trigger Latency Bit 3	1
2	Trigger Latency Bit 2	0
1	Trigger Latency Bit 1	0
0	Trigger Latency Bit 0 (LSB)	0

Table 4 — Trigger Latency default settings

These bits are used to set the delay between the Write and Read pointers of the Pipeline SRAM. A default value of 200 bunch crossings is hardwired, but the actual value for operation will be determined by the delays in the final trigger system. Including bit 0 of Register 0, there is a 9 bit value offering 512 possible latency settings.

Page1 — Register 2: Beta Multiplier & SLVS Output Pad settings

Address: b00000010		
Bit	**Function**	**Default**
7	Beta Multiplier Control Bit 3 (MSB)	0
6	Beta Multiplier Control Bit 2	0
5	Beta Multiplier Control Bit 1	0
4	Beta Multiplier Control Bit 0 (LSB)	1
3	SLVS Current Bit 3 (MSB) – Active Low	1
2	SLVS Current Bit 2 – Active Low	0
1	SLVS Current Bit 1 – Active Low	0
0	SLVS Current Bit 0 (LSB) – Active Low	0

Table 5 — Beta Multiplier & SLVS Output Pad settings

Bits 7:4 are used to set the current in the Beta Multiplier circuit. This defines the current in the feedback of the Post-amp circuit. The programmable range of current is 11nA (0001) to 143nA (1111) at typical process corner and nominal 1V analogue supply voltage. A value of 0000 will produce only 84.4pA.

Bits 3:0 are used to set the current used by the SLVS outputs. The bits are active low with 0xF = Minimum current = 1.5mA and 0x0 = Maximum current = 2.5mA. The default setting gives 2mA.

Page1 — Register 3: Preamp Input Branch Current (Ipre1) settings

Address: b00000011		
Bit	**Function**	**Default**
7	Preamp input Branch Current (MSB)	0
6	Preamp input Branch Current	0
5	Preamp input Branch Current	0
4	Preamp input Branch Current	0
3	Preamp input Branch Current	1
2	Preamp input Branch Current	0
1	Preamp input Branch Current	1
0	Preamp input Branch Current (LSB)	0

Table 6 — Preamp input Branch Current (Ipre1)

The programmable range of current is 0 to 105µA, with a default value of 34µA.

► **NOTE:** The actual value of the current flowing in the preamp input transistor is approximately 6 times the value programmed in this register.

Page1 — Register 4: Preamp Cascode Branch Current (Ipre2) settings

Address: b00000100		
Bit	**Function**	**Default**
7	Preamp Cascode Branch Current (MSB)	0
6	Preamp Cascode Branch Current	0
5	Preamp Cascode Branch Current	1
4	Preamp Cascode Branch Current	0
3	Preamp Cascode Branch Current	1
2	Preamp Cascode Branch Current	1
1	Preamp Cascode Branch Current	1
0	Preamp Cascode Branch Current Bit 0 (LSB)	0

Table 7 — Preamp Cascode Branch Current (Ipre2)

The programmable range of current is 0 to 45µA, with a default value of 10µA.

Page1 — Register 5: Preamp Source Follower Bias (Ipsf) settings

Address: b00000101		
Bit	**Function**	**Default**
7	Preamp Source Follower Bias Current (MSB)	0
6	Preamp Source Follower Bias Current	1
5	Preamp Source Follower Bias Current	1
4	Preamp Source Follower Bias Current	1
3	Preamp Source Follower Bias Current	1
2	Preamp Source Follower Bias Current	0
1	Preamp Source Follower Bias Current	1
0	Preamp Source Follower Bias Current (LSB)	0

Table 8 — Preamp Source Follower Bias (Ipsf)

The programmable range of current is 0 to 37µA, with a default of 22µA.

Page1 — Register 6: Postamp Bias Current (Ipa) settings

Address: b00000110		
Bit	**Function**	**Default**
7	Postamp Bias Current (MSB)	0
6	Postamp Bias Current	1
5	Postamp Bias Current	1
4	Postamp Bias Current	0
3	Postamp Bias Current	1
2	Postamp Bias Current	0
1	Postamp Bias Current	1
0	Postamp Bias Current (LSB)	0

Table 9 — Postamp Bias Current (Ipa)

The programmable range of current is 0 to 46µA, with a default of 21µA.

Page1 — Register 7: Postamp Offset Adjustment (Ipaos) settings

Address: b0000011		
Bit	**Function**	**Default**
7	Postamp Offset adjust Bias Current (MSB)	0
6	Postamp Offset adjust Bias Current	1
5	Postamp Offset adjust Bias Current	0
4	Postamp Offset adjust Bias Current	0
3	Postamp Offset adjust Bias Current	1
2	Postamp Offset adjust Bias Current	0
1	Postamp Offset adjust Bias Current	1
0	Postamp Offset adjust Bias Current (LSB)	1

Table 10 — Postamp Offset Adjustment Bias Current (Ipaos)

The programmable range of current is 0 to 24µA, with a default of 8µA.

This current is the reference current for the circuit used to trim the output voltage level from the post amplifier circuits.

Page1 — Register 9: Comparator Bias Current (Icomp) setting

Address b00001001		
Bit	**Function**	**Default**
7	Comparator Bias Current (MSB)	0
6	Comparator Bias Current	0
5	Comparator Bias Current	1
4	Comparator Bias Current	0
3	Comparator Bias Current	0
2	Comparator Bias Current	0
1	Comparator Bias Current	1
0	Comparator Bias Current (LSB)	1

Table 11 — Comparator Bias Current (Icomp)

The programmable range of current is 0 to 20µA, with a default of 3.2µA.

Page1 — Register 11: Postamp Reference Voltage (VPLUS1/VPLUS2) setting

Address b00001011		
Bit	**Function**	**Default**
7	VPLUS2 (MSB)	0
6	VPLUS2	1
5	VPLUS2	1
4	VPLUS2 (LSB)	1
3	VPLUS1 (MSB)	0
2	VPLUS 1	1
1	VPLUS1	1
0	VPLUS 1(LSB)	1

Table 12 — Postamp Reference Voltages (Vplus1 & 2)

VPLUS 1 is a global reference to the positive input of the post amplifiers. VPLUS 2 is used to generate the feedback control voltage for the post amplifier feedback network, which is then distributed globally to the post amplifiers. These two voltages are meant to be equal and are derived from two identical 4 bit resistor ladder DACs. The programmable range of voltage is 281mV to 750mV, with a default of 500mV.

Page1 — Register 12: HIP Suppression, Serialiser Handshake Retiming & SLVS Disable setting

Address b00001100		
Bit	**Function**	**Default**
7	HIP count Bit 2 (MSB)	0
6	HIP count Bit 1	0
5	HIP count Bit 0 (LSB)	0
4	HIP Suppress	1
3	HIP source	0
2	Select_Clk40_Ref	0
1	Select_Clk40_DLL	0
0	Output SLVS Enable/Disable	0

Table 13 — HIP Suppression options & SLVS output enable/disable

The HIP Count (bits 7:5) is used by the *Hit Detect* circuit to define the acceptable length of time that the Hit Detect output can remain high following a single over-threshold event in the comparator. At the end of the count, if the comparator output has not returned below threshold, then the output of the *Hit*

Detect circuit is forced to zero. The count rate is 40MHz to match the bunch crossing rate. The default count (0) will suppress hits when the feature is enabled. HIP Suppression is enabled by setting bit 4 to 0.

Bit 3 selects whether the HIP suppression acts on either the *40MHz Sampled* output, or the *Logical OR* output (The OR of the *40MHz Sampled* output with the *Fixed Pulse Width* output). The default setting is the *40MHz Sampled* output.

Bit 2 selects which edge of the reference 40MHz clock is used in the retiming circuit for the handshake control signal that indicates to the *Output Data Serialiser* that there is data to be read from the *Buffer SRAM*. This should be set to 1 if timing issues occur when the delayed version of the 40MHz clock is close to the phase of the reference 40MHz clock.

Bit 1 selects which edge of the delayed 40MHz clock is used in the retiming circuit for the handshake control signal from the *Output Data Serialiser* to the *Buffer SRAM*, when reading available data. This should be set to 1 if timing issues occur when the delayed version of the 40MHz clock is close to anti-phase with the reference 40MHz clock.

Bit 0 selects whether the SLVS outputs are enabled/disabled. The default is enabled.

Page1 — Register 13: Test Pulse Amplitude setting

Address b00001101		
Bit	**Function**	**Default**
7	Test Pulse Potentiometer Node Select (MSB)	0
6	Test Pulse Potentiometer Node Select	0
5	Test Pulse Potentiometer Node Select	0
4	Test Pulse Potentiometer Node Select	0
3	Test Pulse Potentiometer Node Select	0
2	Test Pulse Potentiometer Node Select	0
1	Test Pulse Potentiometer Node Select	0
0	Test Pulse Potentiometer Node Select (LSB)	0

Table 14 — Test Pulse Amplitude setting

This register is used to select the value of the voltage step applied by the Test Pulse Generator. The default setting gives a voltage output equal to VDDA. Setting Bit 7 to 1 gives half range.

Page1 — Register 14: Test Pulse Delay & Channel Group Selection

Address b00001110		
Bit	**Function**	**Default**
7	Test Delay Select Bit 0 (LSB)	0
6	Test Delay Select Bit 1	0
5	Test Delay Select Bit 2	0
4	Test Delay Select Bit 3	0
3	Test Delay Select Bit 4 (MSB)	0
2	Select Channel Group To Test Bit 0 (LSB)	0
1	Select Channel Group To Test Bit 1	0
0	Select Channel Group To Test Bit 2 (MSB)	0

Table 15 — Test Pulse Delay & Channel Group Selection

The Test Delay Select bits (Bits 7:3) determine the time delay for the test pulse, the active edge of which can be delayed relative to the 40MHz clock. The valid code range is from 00000 (1ns delay) through to 11001 (25ns delay), and codes outside of this range will result in no test pulse output.

Bits 2:0 select which group of channels is to receive the test pulse. The default setting of 000 connects the lowest numbered channel group. Table 16 shows the channels selected for each possible setting.

Channel Group Select	Stimulated Channels
000	242, 241, 226, 225, 210, 209, 194, 193, 178, 177, 162, 161, 146, 145, 130, 129, 114, 113, 98, 97, 82, 81, 66, 65, 50, 49, 34, 33, 18, 17, 2, 1
001	244, 243, 228, 227, 212, 211, 196, 195, 180, 179, 164, 163, 148, 147, 132, 131, 116, 115, 100, 99, 84, 83, 68, 67, 52, 51, 36, 35, 20, 19, 4, 3
010	246, 245, 230, 229, 214, 213, 198, 197, 182, 181, 166, 165, 150, 149, 134, 133, 118, 117, 102, 101, 86, 85, 70, 69, 54, 53, 38, 37, 22, 21, 6, 5
011	248, 247, 232, 231, 216, 215, 200, 199, 184, 183, 168, 167, 152, 151, 136, 135, 120, 119, 104, 103, 88, 87, 72, 71, 56, 55, 40, 39, 24, 23, 8, 7
100	250, 249, 234, 233, 218, 217, 202, 201, 186, 185, 170, 169, 154, 153, 138, 137, 122, 121, 106, 105, 90, 89, 74, 73, 58, 57, 42, 41, 26, 25, 10, 9
101	252, 251, 236, 235, 220, 219, 204, 203, 188, 187, 172, 171, 156, 155, 140, 139, 124, 123, 108, 107, 92, 91, 76, 75, 60, 59, 44, 43, 28, 27, 12, 11
110	254, 253, 238, 237, 222, 221, 206, 205, 190, 189, 174, 173, 158, 157, 142, 141, 126, 125, 110, 109, 94, 93, 78, 77, 62, 61, 46, 45, 30, 29, 14, 13
111	Top Dummy, 240, 239, 224, 223, 208, 207, 192, 191, 176, 175, 160, 159, 144, 143, 128, 127, 112, 111, 96, 95, 80, 79, 64, 63, 48, 47, 32, 31, 16, 15, Bottom Dummy

Table 16 — Test Pulse Channel Selection

Page1 — Register 15: Test Pulse Control & Analogue Mux setting

Address b00001111		
Bit	**Function**	**Default**
7	Test Pulse Polarity Select (= 1 for positive edge)	0
6	Enable Test Pulse (= 1 to enable)	0
5	Ground the Test Pulse capacitor plates of non-selected Test Pulse groups (=1 to ground capacitors)	0
4	Analogue Mux Select Bit 4 (MSB)	0
3	Analogue Mux Select Bit 3	0
2	Analogue Mux Select Bit 2	0
1	Analogue Mux Select Bit 1	0
0	Analogue Mux Select Bit 0 (LSB)	0

Table 17 — Test Pulse Control & Analogue Mux

Bits 4:0 are used to select the bias voltage to be seen on the output of the *Analogue Multiplexer* circuit. The 5-bit codes and their corresponding bias voltages are shown in table 18 below.

Analogue Multiplexer output selection		
CODE	**BIAS**	**Function**
00000	Floating	No output
00001	Ipa	Postamp bias
00010	Ipre2	Preamp Cascode Branch bias
00011	Cal_I	Calibration Circuit delay control circuit bias
00100	Ibias	Master reference current for the Bandgap circuit
00101	Vcth	Comparator threshold voltage
00110	VBGbias	Master voltage bias from the voltage bandgap circuit
00111	VBG_LDO	Bias voltage to the LDO
01000	Vpafb	Postamp Feedback bias
01001	Nc50	Comparator current balancing reference
01010	Ipre1	Preamp Input Branch bias
01011	Ipsf	Preamp Source Follower bias
01100	Ipaos	Postamp Offset Adjust bias
01101	Icomp	Comparator amplifier bias
01110	Ihyst	Comparator hysteresis bias
01111	CAL_Vcasc	Calibration Circuit Cascode bias
10000	VPLUS2	Postamp feedback control voltage
10001	VPLUS1	Postamp reference voltage

Table 18 — Valid Analogue Multiplexer codes and their corresponding bias voltages.

Bits 7:5 relate to the *Test Pulse Generator* circuit. Bit 7 chooses the polarity of the test pulse. This feature was retained even though the Pre-amp is no longer designed for both polarity of signal. Bit 6

enables the *Test Pulse Generator* circuit. As this is a test feature that will not be used in normal operation, it is disabled as a default. Bit 5 determines whether the Test Pulse side of the Test Capacitors is grounded or allowed to float (default) when they are not connected to the *Test Pulse Generator* circuit. This feature was made available in case the floating option results in excess noise.

Page1 — Register 16: CAL_I bias setting

Address 00010000		
Bit	**Function**	**Default**
7	CAL_I (MSB)	0
6	CAL_I	1
5	CAL_I	1
4	CAL_I	0
3	CAL_I	0
2	CAL_I	1
1	CAL_I	0
0	CAL_I (LSB)	0

Table 19 — CAL_I bias settings

This sets the bias current for the analogue delay control circuit used in the *Test Pulse Generator's* DLL. The default current is set to 5µA with a full range of 12.7µA available.

Page1 — Register 17: CAL_Vcasc bias setting

Address b00010001		
Bit	**Function**	**Default**
7	CAL_Vcasc (MSB)	0
6	CAL_Vcasc	0
5	CAL_Vcasc	1
4	CAL_Vcasc	1
3	CAL_Vcasc	1
2	CAL_Vcasc	1
1	CAL_Vcasc	1
0	CAL_Vcasc (LSB)	1

Table 20 — CAL_Vcasc bias settings

This sets the bias voltage for the cascade transistor in the analogue delay control circuit used in the *Test Pulse Generator's* DLL. The default voltage is set to 0.4V, with a full range of 0.8V available.

Page1 — Register 18: Pipeline & Stub Logic input selection, & Pt Width setting

Address b00010010		
Bit	**Function**	**Default**
7	PLSel bit 1 (MSB) – Select output from hit detect to pipeline	0
6	PLSel bit 0 (LSB)	0
5	STSel bit 1 (MSB) – Select input to Stub Logic	0
4	STSel bit 0 (LSB)	0
3	Pt_Width bit 3 (MSB) – Strip Window	0
2	Pt_Width bit 2	0
1	Pt_Width bit 1	1
0	Pt_Width bit 0 (LSB)	1

Table 21 — Pipeline & Stub logic input selection & Pt width setting

Bits 7:6 are used to select which of the four available *Hit Detect* circuit outputs are used as the input to the *Pipeline SRAM*. Table 22 shows the code assignments.

Bits 5:4 are used to select which of the four available *Hit Detect* circuit outputs are used as the input to the *Stub Finding Logic*. Table 22 shows the code assignments.

CODE	Hit Detect output to Pipeline	Hit Detect output to SFL
00	40MHz Sampled Output	40MHz Sampled Output
01	Logical OR Output	Logical OR Output
10	HIP Suppressed Output	HIP Suppressed Output
11	Fixed Pulse Width Output	Fixed Pulse Width Output

Table 22 — Selection codes for the Pipeline & Stub Logic input choice

Page1 — Register 19: Correlation Window Offset setting for regions 3 & 4

Address 00010011		
Bit	**Function**	**Default**
7	Offset4<3> (MSB)	0
6	Offset4<2>	0
5	Offset4<1>	0
4	Offset4<0> (LSB)	0
3	Offset3<3> (MSB)	0
2	Offset3<2>	0
1	Offset3<1>	0
0	Offset3<0> (LSB)	0

Table 23 — Correlation Window Offset setting for regions 3 & 4

Bits 7:4 are used to adjust the offset of the *Correlation Window* for the highest channel numbered region. Likewise bits 3:0 are used to adjust the offset of the *Correlation Window* for the next lower channel numbered region. The offset has an adjustable range of +/- 3 sensor strips, with a half-strip adjustment resolution. The code scheme uses two's compliment to represent negative offset, as shown in table 24.

► **NOTE:** Positive offset is in the direction of the higher number channels and negative is in the direction of the lower number channels.

CODE	OFFSET	CODE	OFFSET
0000 (Default)	No Offset	1111	-1 half-strip
0001	+1 half-strip	1110	-2 half-strips
0010	+2 half-strips	1101	-3 half-strips
0011	+3 half-strips	1100	-4 half-strips
0100	+4 half-strips	1011	-5 half-strips
0101	+5 half-strips	1010	-6 half-strips
0110	+6 half-strips		

Table 24 — Valid codes for Correlation Window Offset programming.

Page1 — Register 20: Correlation Window Offset setting for regions 1 & 2

Address b00010100		
Bit	**Function**	**Default**
7	Offset2<3> (MSB)	0
6	Offset2<2>	0
5	Offset2<1>	0
4	Offset2<0> (LSB)	0
3	Offset1<3> (MSB)	0
2	Offset1<2>	0
1	Offset1<1>	0
0	Offset1<0> (LSB)	0

Table 25 — Correlation Window Offset setting for regions 1 & 2

Bits 3:0 are used to adjust the offset of the *Correlation Window* for the lowest channel numbered region. Likewise bits 7:4 are used to adjust the offset of the *Correlation Window* for the next higher channel numbered region. The offset has an adjustable range of +/- 3 sensor strips, with a half-strip adjustment resolution. The code scheme uses two's compliment to represent negative offset, as shown in table 24.

Page1 — Register 21: Bandgap Fuse Register setting

Address b00010101		
Bit	**Function**	**Default**
7	Override – Overrides the fuse setting with the register bits 0-5	0
6	Sel – select whether to read out fuse or register (1=fuse)	0
5	BG Fuse Bit 5 – set to 1 to blow.	0
4	BG Fuse Bit 4 – set to 1 to blow.	0
3	BG Fuse Bit 3 – set to 1 to blow.	0
2	BG Fuse Bit 2 – set to 1 to blow.	0
1	BG Fuse Bit 1 – set to 1 to blow.	0
0	BG Fuse Bit 0 – set to 1 to blow.	0

Table 26 — Bandgap Fuse Register settings

Bit 7 allows the user to override the trim value that was configured with the eFuses during wafer level testing.

Bit 6 allows the user to read back either the trim value set by the eFuses, or the value that was written to the register via the I^2C interface. The default is to read back the register default setting or the last written register contents.

Bits 5:0 represent the trim value for the Bandgap. The value written to these bits will be programmed into the eFuses during the trimming phase of the wafer-level testing. The programmed value affects the the two voltage outputs from the bandgap, namely *VBG_LDO* and *VBGbias*. Code 000000 (default) gives the highest output voltage in both cases. Each code is equivalent to a 1.7mV step in the voltage. For *VBG_LDO* the programmable range is 560mV:451mV, and for *VBGbias* the programmable range is from 694mV:586mV.

Page1 — Register 22: Chip ID eFuse Register setting (Part 1)

Address b00010110		
Bit	**Function**	**Default**
7	Chip ID<7>	0
6	Chip ID<6>	0
5	Chip ID<5>	0
4	Chip ID<4>	0
3	Chip ID<3>	0
2	Chip ID<2>	0
1	Chip ID<1>	0
0	Chip ID<0> LSB	0

Table 27 — Chip ID eFuse programming register (Least significant 8 bits)

Bits 7:0 comprise the least significant 8 bits of the 19 bit word that is used to program the eFuses that identify the ASIC. This programming will be carried out at the time of wafer-level testing.

Page1 — Register 23: Chip ID eFuse Register setting (Part 2)

Address b00010111		
Bit	**Function**	**Default**
7	Chip ID<15>	0
6	Chip ID<14>	0
5	Chip ID<13>	0
4	Chip ID<13>	0
3	Chip ID<11>	0
2	Chip ID<10>	0
1	Chip ID<9>	0
0	Chip ID<8>	0

Table 28 — Chip ID eFuse programming register (Next most significant 8 bits)

Bits 7:0 comprise the next most significant 8 bits of the 19 bit word that is used to program the eFuses that identify the ASIC.

Page1 — Register 24: Chip ID eFuse Register setting (Part 3)

Address b00011000		
Bit	**Function**	**Default**
7	Unused	0
6	Unused	0
5	Unused	0
4	Unused	0
3	Selects whether the efuse value is read out. (1=fuse)	0
2	Chip ID<18> MSB	0
1	Chip ID<17>	0
0	Chip ID<16>	0

Table 29 — Chip ID eFuse programming register (Most significant 3 bits)

Bits 2:0 comprise the most significant 3 bits of the 19 bit word that is used to program the eFuses that identify the ASIC.

Bit 3 enables the user to read back the eFuse values for the Chip ID after programming. The default is to read back the register default setting or the last written register contents.

Page1 — Register 27: Layer Swap control & Cluster Width setting

Address b00011011		
Bit	**Function**	**Default**
7	Not used	0
6	Not used	0
5	Not used	0
4	Not used	0
3	Layer swap	0
2	Cwidth<2>	0
1	Cwidth<1>	0
0	Cwidth<0>	0

Table 30 — Layer Swap control & Cluster Width settings

Bit 3 is used to change which channels are used as the Seed and Correlation layers for the purposes of *Stub* finding. The default setting uses the ODD number channels (253:1) for the *Seed Layer* and the EVEN numbered channels (254:2) for the *Correlation Layer*.

Bits 2:0 are used to set the rejection width for *Clusters* in the *Seed Layer*. The effect of each code is shown in table 31 below.

Cluster Width setting (Cwidth<2:0>)	Result
000	No rejection based on cluster width. Strip passed to Stub Gathering Logic
001	Cluster width of 1: Only passes clusters of 1 hit strip. Rejects >1
010	Cluster width of 2: Only passes clusters of 1 or 2 hit strips. Rejects >2
011	Cluster width of 3: Only passes clusters of 1, 2 or 3 hit strips. Rejects >3
100	Cluster width of 4: Only passes clusters of 1, 2, 3 or 4 hit strips. Rejects >4
101	No rejection based on cluster width. Strip passed to Stub Gathering Logic
110	
111	

Table 31 — Effect of Cluster Width settings

Page1 — Register 28: 40MHz Clock Delay, Test Pulse Clock, 40MHz Clock Output and OR254

Address 00011100		
Bit	**Function**	**Default**
7	TPG clock select (0=40MHZ)	0
6	ENOR254 – enable OR254 output	0
5	Enable 40MHz clock test output	0
4	40MHz Clock Delay Select Bit 0 (LSB)	0
3	40MHz Clock Delay Select Bit 1	0
2	40MHz Clock Delay Select Bit 2	0
1	40MHz Clock Delay Select Bit 3	0
0	40MHz Clock Delay Select Bit 4 (MSB)	0

Table 32 — 40MHz Clock Domain options and OR254 test feature

Bit 7 is used to select the clock used for the test pattern generation circuit. The default is zero which bypasses the DLL and uses the direct clock input to the chip. This feature is intended for use only during wafer probe testing when a clock speed of 40MHz is used instead of 320MHz. The test pattern generation circuit will only operate using a 40MHz clock and cannot be run using the 320MHz clock. Setting the bit to 1 uses the 40MHz clock derived from the DLL circuit, which is normal operation.

Bit 6 is used to enable the logical OR of the 254 channel outputs. This is a test feature and the default state is disabled.

Bit 5 is used to enable a test output for the 40MHz clock generated on-chip. The default is tri-state.

Bits (4:0) are used to program the delay between the on chip derived 40MHz clock and the phase shifted 40MHz clock. The delay is programmable in 1ns increments from 0 to 25ns with the default set as 0ns.

► **NOTE**: There is a slight difference in the path between the 40MHz test output and the 0ns delayed 40MHz clock.

Page1 — Register 29: Fast Command Interface and Error Flags

Address 00011101		
Bit	**Function**	**Default**
7	FCI delay<1>	0
6	FCI delay<0>	0
5	Not used	0
4	Buffer RAM overflow (read only)	0
3	Latency error (read only)	0
2	Sync lost (read only)	0
1	Sync stat (read only)	0
0	Bad code (read only)	0

Table 33 — Fast Command Interface and Error Flags

Bits (7:6) are used to select a delay to the serial command word by either one or two 320MHz clock cycles. This is used to coarsely adjust the timing of the 40MHz clock derived from the serial command. The default is no delay. Values 00, 01 (one clock cycle delay) and 11 (two clock cycles delay) are valid. The value 10 is not valid.

Bits (4:0) are error flags and are read only. Bit 4 indicates an overflow in the buffer RAM in the event that more than 32 triggers are received before the L1 data can be read. Bit 3 indicates a mismatch between the programmed latency and the difference between the pipeline read and write counters.

Bits (2:0) are error flags for the fast command interface. Bit 2 indicates that synchronization has been lost at some time and does not imply that this is still the case. Bit one indicates the current synchronization status only. Bit zero indicates whether an invalid code has been received. This may be due to synchronization issues or simply the user sending an incorrect command code.

Page1 — Register 32 to 62: Mask Channels 1-248

Addresses 00100000 to 00111110		
Bit	**Function**	**Default**
7	Mask Channel<8> … <248>	1
6	Mask Channel<7> … <247>	1
5	Mask Channel<6> … <246>	1
4	Mask Channel<5> … <245>	1
3	Mask Channel<4> … <244>	1
2	Mask Channel<3> … <243>	1
1	Mask Channel<2> … <242>	1
0	Mask Channel<1> … <241>	1

Table 34 — Channel Masking registers

These registers are used to either mask channels that are behaving badly or for test purposes. One bit is used per channel, with 8 bits per register requiring 31 register for channels 1-248. In the default case all of channels are unmasked.

Page1 — Register 63: Mask Channels 249-254

Address 00111111		
Bit	**Function**	**Default**
7	Unused	1
6	Unused	1
5	Mask Channel<254>	1
4	Mask Channel<253>	1
3	Mask Channel<252>	1
2	Mask Channel<251>	1
1	Mask Channel<250>	1
0	Mask Channel<249>	1

Table 35 — The last Channel Masking register

Register 63 is used for masking the six uppermost channels. Due to the number of channels being limited to 254 not all of the bits in this register are used. In the default case all of channels are unmasked.

Page1 — Register 64 to 78: Bend register<0:14>

Addresses 01000000 - 01001110																
Cluster centre	**Reg No.**	**Page 1 ADDRESS**	**Input Code**					**Default Output Code**								
								7	6	5	4	3	2	1	0
-7	0	01000000	1	0	0	1	0					1	0	0	1
-6 ½	0	01000000	1	0	0	1	1	1	0	0	1				
-6	1	01000001	1	0	1	0	0					1	0	1	0
-5 ½	1	01000001	1	0	1	0	1	1	0	1	0				
-5	2	01000010	1	0	1	1	0					1	0	1	1
-4 ½	2	01000010	1	0	1	1	1	1	0	1	1				
-4	3	01000011	1	1	0	0	0					1	1	0	0
-3 ½	3	01000011	1	1	0	0	1	1	1	0	0				
-3	4	01000100	1	1	0	1	0					1	1	0	1
-2 ½	4	01000100	1	1	0	1	1	1	1	0	1				
-2	5	01000101	1	1	1	0	0					1	1	1	0
-1 ½	5	01000101	1	1	1	0	1	1	1	1	0				
-1	6	01000110	1	1	1	1	0					1	1	1	1
-½	6	01000110	1	1	1	1	1	1	1	1	1				
CENTRE	7	01000111	0	0	0	0	0					0	0	0	0
+½	7	01000111	0	0	0	0	1	0	0	0	0				
+1	8	01001000	0	0	0	1	0					0	0	0	1
+1 ½	8	01001000	0	0	0	1	1	0	0	0	1				
+2	9	01001001	0	0	1	0	0					0	0	1	0
+2 ½	9	01001001	0	0	1	0	1	0	0	1	0				
+3	10	01001010	0	0	1	1	0					0	0	1	1
+3 ½	10	01001010	0	0	1	1	1	0	0	1	1				
+4	11	01001011	0	1	0	0	0					0	1	0	0
+4 ½	11	01001011	0	1	0	0	1	0	1	0	0				
+5	12	01001100	0	1	0	1	0					0	1	0	1
+5 ½	12	01001100	0	1	0	1	1	0	1	0	1				
+6	13	01001101	0	1	1	0	0					0	1	1	0
+6 ½	13	01001101	0	1	1	0	1	0	1	1	0				
+7	14	01001110	0	1	1	1	0					0	1	1	1
Invalid	14	01001110	1	0	0	0	0	1	0	0	0				

Table 36 — Bend Lookup Table mapping

Registers 64 to 78 are used to define the four bit bend information that is output in response to 5 bit bend codes generated by the stub finding logic.

The default register settings are equal to the 4 most significant bits of the corresponding 5 bit bend code. The 4 bit equivalent of the strip 5 bit bend codes is set using the 4 least significant bits of the registers. The 4 bit equivalent of the half-strip 5-bit bend codes is set using the 4 most significant bits of the register.

Page1 — Register 79: VCTH (Threshold Voltage part 1)

Address 01001111		
Bit	**Function**	**Default**
7	VCTH <7>	0
6	VCTH <6>	0
5	VCTH <5>	0
4	VCTH <4>	0
3	VCTH <3>	0
2	VCTH <2>	0
1	VCTH <1>	0
0	VCTH <0>	0

Table 37 — Comparator threshold setting (Part 1)

Register 79 contains part of the 10 bit DAC code used to set the comparator threshold. The eight least significant bits of the 10 bit word are set to 0 as default.

Page1 — Register 80: VCTH (Threshold Voltage part 2)

Address 01010000		
Bit	**Function**	**Default**
7	Unused	0
6	Unused	0
5	Unused	0
4	Unused	0
3	Unused	0
2	Unused	0
1	VCTH <9>	1
0	VCTH <8>	0

Table 38 — Comparator threshold setting (Part 2)

Register 80 contains part of the 10 bit DAC code used to set the comparator threshold. The most significant bit of the 10 bit word is set to 1 as default; this gives a threshold setting in the middle of the range.

*Page1 — Registers 81 to 84: **TOP** Nearest Neighbour IO Test Registers*

Address 01010001			
Bit	**Function**	**Access**	**Default**
7	Test input for Nearest Neighbour Input: TopIn_B<5>	Read/Write	0
6	Test input for Nearest Neighbour Input: TopIn_B<4>	Read/Write	0
5	Test input for Nearest Neighbour Input: TopIn_B<3>	Read/Write	0
4	Test input for Nearest Neighbour Input: TopIn_B<2>	Read/Write	0
3	Test input for Nearest Neighbour Input: TopIn_B<1>	Read/Write	0
2	Test input for Nearest Neighbour Input: TopIn_A<3>	Read/Write	0
1	Test input for Nearest Neighbour Input: TopIn_A<3>	Read/Write	0
0	Test input for Nearest Neighbour Input: TopIn_A<3>	Read/Write	0

Table 39 — TOP Nearest Neighbour IO Test Input Register 81

Address 01010010			
Bit	**Function**	**Access**	**Default**
7	Test input for Nearest Neighbour Input: TopIn_B<13>	Read/Write	0
6	Test input for Nearest Neighbour Input: TopIn_B<12>	Read/Write	0
5	Test input for Nearest Neighbour Input: TopIn_B<11>	Read/Write	0
4	Test input for Nearest Neighbour Input: TopIn_B<10>	Read/Write	0
3	Test input for Nearest Neighbour Input: TopIn_B<9>	Read/Write	0
2	Test input for Nearest Neighbour Input: TopIn_B<8>	Read/Write	0
1	Test input for Nearest Neighbour Input: TopIn_B<7>	Read/Write	0
0	Test input for Nearest Neighbour Input: TopIn_B<6>	Read/Write	0

Table 40 — TOP Nearest Neighbour IO Test Input Register 82

Registers 82 and 81 provide the input pattern stimulus representing Nearest Neighbour signals coming into the top of the CBC3.1. Their default value is 0. Patterns of 1's and 0's can be programmed as stimulus for the Stub Finding Logic either for single chip or wafer level tests. They are activated by setting both control bits (1:0) of register 83 to a 1, otherwise they have no effect on the Stub Finding Logic and signals from off chip are required.

Address 01010011			
Bit	**Function**	**Access**	**Default**
7	Test output for Nearest Neighbour Output: TopOut_B<4>	Read Only	0
6	Test output for Nearest Neighbour Output: TopOut_B<3>	Read Only	0
5	Test output for Nearest Neighbour Output: TopOut_B<2>	Read Only	0
4	Test output for Nearest Neighbour Output: TopOut_B<1>	Read Only	0
3	Test output for Nearest Neighbour Output: TopOut_A<2>	Read Only	0
2	Test output for Nearest Neighbour Output: TopOut_A<1>	Read Only	0
1	Test Mode Select: Multiplexes inputs from pads or test register. Both set to either 1 or 0 at the same time as protection against SEU	Read/Write	0
0		Read/Write	0

Table 41 — TOP Nearest Neighbour IO Test Output Register 83, and Test Mode Select

Address 01010100			
Bit	**Function**	**Access**	**Default**
7	Test output for Nearest Neighbour Output: TopOut_B<12>	Read Only	0
6	Test output for Nearest Neighbour Output: TopOut_B<11>	Read Only	0
5	Test output for Nearest Neighbour Output: TopOut_B<10>	Read Only	0
4	Test output for Nearest Neighbour Output: TopOut_B<9>	Read Only	0
3	Test output for Nearest Neighbour Output: TopOut_A<8>	Read Only	0
2	Test output for Nearest Neighbour Output: TopOut_A<7>	Read Only	0
1	Test output for Nearest Neighbour Output: TopOut_A<6>	Read Only	0
0	Test output for Nearest Neighbour Output: TopOut_A<5>	Read Only	0

Table 42 — TOP Nearest Neighbour IO Test Output Register 84

Registers 83 and 84 provide access to the outputs from the top channels feeding the Nearest Neighbour IO pads going out from the top of the CBC3.1. Their default value is 0. Register 84 bits (7:0) and register 83 bits (7:2) are read only, for the purpose of reading back via I2C the values that appear on the Nearest Neighbour outputs.

Bits (1:0) of register 83 are used to control whether the test input registers are used as inputs to the Stub Finding Logic in place of the signals from the Nearest Neighbour IO pads. Default is 0, which connects the pads and not the register outputs to the logic.

*Page1 — Registers 85 to 88: **BOTTOM** Nearest Neighbour IO Test Registers*

Address 01010101			
Bit	**Function**	**Access**	**Default**
7	Test input for Nearest Neighbour Input: BotIn_B<6>	Read/Write	0
6	Test input for Nearest Neighbour Input: BotIn_B<5>	Read/Write	0
5	Test input for Nearest Neighbour Input: BotIn_B<4>	Read/Write	0
4	Test input for Nearest Neighbour Input: BotIn_B<3>	Read/Write	0
3	Test input for Nearest Neighbour Input: BotIn_B<2>	Read/Write	0
2	Test input for Nearest Neighbour Input: BotIn_B<1>	Read/Write	0
1	Test input for Nearest Neighbour Input: BotIn_A<2>	Read/Write	0
0	Test input for Nearest Neighbour Input: BotIn_A<1>	Read/Write	0

Table 43 — BOTTOM Nearest Neighbour IO Test Input Register 85

Address 01010110			
Bit	**Function**	**Access**	**Default**
7	Test Mode Select: Multiplexes inputs from pads or test register. Both set to either 1 or 0 at the same time as protection against SEU	Read/Write	0
6		Read/Write	0
5	Test input for Nearest Neighbour Input: BotIn_B<12>	Read/Write	0
4	Test input for Nearest Neighbour Input: BotIn_B<11>	Read/Write	0
3	Test input for Nearest Neighbour Input: BotIn_B<10>	Read/Write	0
2	Test input for Nearest Neighbour Input: BotIn_B<9>	Read/Write	0
1	Test input for Nearest Neighbour Input: BotIn_A<8>	Read/Write	0
0	Test input for Nearest Neighbour Input: BotIn_A<7>	Read/Write	0

Table 44 — BOTTOM Nearest Neighbour IO Test Input Register 86, and Test Mode Select

Registers 85 and 86 provide the input pattern stimulus representing Nearest Neighbour signals coming into the bottom of the CBC3.1. Their default value is 0. Patterns of 1's and 0's can be programmed as stimulus for the Stub Finding Logic either for single chip or wafer level tests.

Bits (7:6) of register 86 are used to control whether the test input registers are used as inputs to the Stub Finding Logic in place of the signals from the Nearest Neighbour IO pads. Default is 0, which connects the pads and not the registers. Setting both control bits to a 1, connects the register outputs to the logic.

Address 01010111			
Bit	**Function**	**Access**	**Default**
7	Test output for Nearest Neighbour Output: BotOut_B<5>	Read Only	0
6	Test output for Nearest Neighbour Output: BotOut_B<4>	Read Only	0
5	Test output for Nearest Neighbour Output: BotOut_B<3>	Read Only	0
4	Test output for Nearest Neighbour Output: BotOut_B<2>	Read Only	0
3	Test output for Nearest Neighbour Output: BotOut_B<1>	Read Only	0
2	Test output for Nearest Neighbour Output: BotOut_A<3>	Read Only	0
1	Test output for Nearest Neighbour Output: BotOut_A<2>	Read Only	0
0	Test output for Nearest Neighbour Output: BotOut_A<1>	Read Only	0

Table 45 — BOTTOM Nearest Neighbour IO Test Output Register 87

Address 01011000			
Bit	**Function**	**Access**	**Default**
7	Test output for Nearest Neighbour Output: BotOut_B<13>	Read Only	0
6	Test output for Nearest Neighbour Output: BotOut_B<12>	Read Only	0
5	Test output for Nearest Neighbour Output: BotOut_B<11>	Read Only	0
4	Test output for Nearest Neighbour Output: BotOut_B<10>	Read Only	0
3	Test output for Nearest Neighbour Output: BotOut_B<9>	Read Only	0
2	Test output for Nearest Neighbour Output: BotOut_B<8>	Read Only	0
1	Test output for Nearest Neighbour Output: BotOut_B<7>	Read Only	0
0	Test output for Nearest Neighbour Output: BotOut_B<6>	Read Only	0

Table 46 — BOTTOM Nearest Neighbour IO Test Output Register 88

Registers 87 and 88 provide access to the outputs from the top channels feeding the Nearest Neighbour IO pads going out from the bottom of the CBC3.1. Their default value is 0. Both registers are read only, for the purpose of reading back via I2C the values that appear on the Nearest Neighbour outputs.

Page2 — Register 1 to 254: Channel 1-254 Offset

Address 00000001 to 11111110		
Bit	**Function**	**Default**
7	Channel Offset <7>	1
6	Channel Offset <6>	0
5	Channel Offset <5>	0
4	Channel Offset <4>	0
3	Channel Offset <3>	0
2	Channel Offset <2>	0
1	Channel Offset <1>	0
0	Channel Offset <0>	0

Table 47 — Channel Offset Adjustment

On page 2 of the I^2C register map, registers 1 to 254 are used to adjust the offset of each channel. The offset adjustment for each channel has an eight bit resolution requiring one I^2C register per channel. The default value for each register is set to decimal 128, which is the centre of the range.

Page2 — Register 255: Dummy Channel Offset

Address 11111111		
Bit	**Function**	**Default**
7	Channel Offset Dummy <7>	1
6	Channel Offset Dummy <6>	0
5	Channel Offset Dummy <5>	0
4	Channel Offset Dummy <4>	0
3	Channel Offset Dummy <3>	0
2	Channel Offset Dummy <2>	0
1	Channel Offset Dummy <1>	0
0	Channel Offset Dummy <0>	0

Table 48 — Dummy Channel Offset Adjustment

Also on page 2 of the I^2C register map, register 255 is used to adjust the offset for the dummy channel. The default value for the register is set to decimal 128, which is the centre of the range.

Pads

Overview

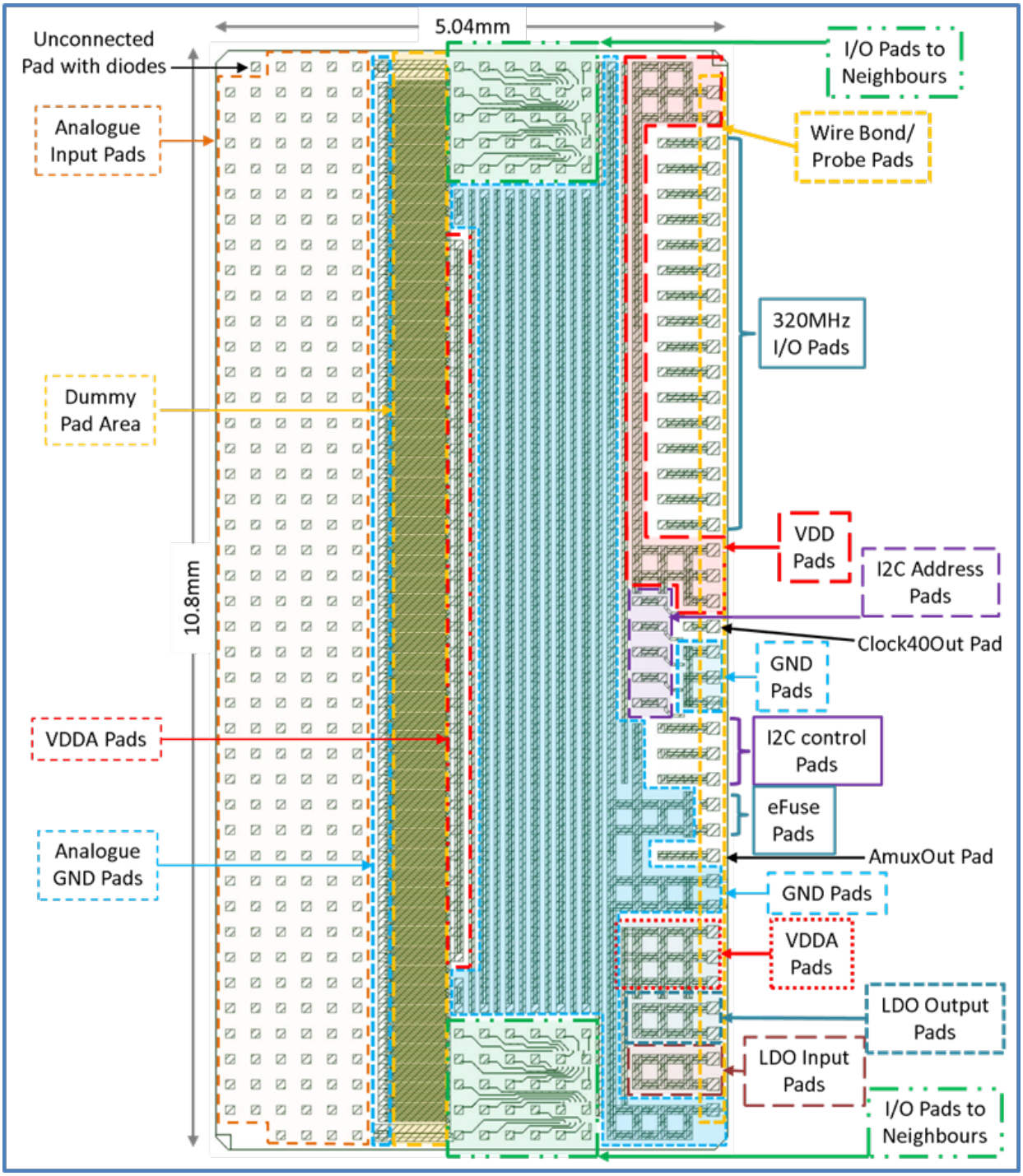

Figure 37 — CBC3.1 Pad layout – Overview (pads facing upwards)

As with the CBC2 and CBC3, there are 254 analogue inputs arranged as an array of pads up to six columns deep by 43 rows, with no active circuitry placed under the pads in order to avoid noise injection. Due to the sensitive nature of these inputs, the pads are separated from the back-end pads by a pad-free region equivalent to two columns of pads. During the bump-ball processing stages there is the option to add two columns of dummy bump-balls in this region, with these balls to be grounded on the hybrid. These Dummy bump-balls will not connect electrically to the ASIC.

The bump pads are intended for C4 lead-free bump bonding with 250um pitch.

The column of pads furthest from the analogue inputs are wire-bondable pads for single ASIC characterisation tests and wafer-probing.

Pad Allocation

Figures 38 and 39 show how the pads are allocated to the I/O. The top and bottom halves of the ASIC are shown separately for greater clarity.

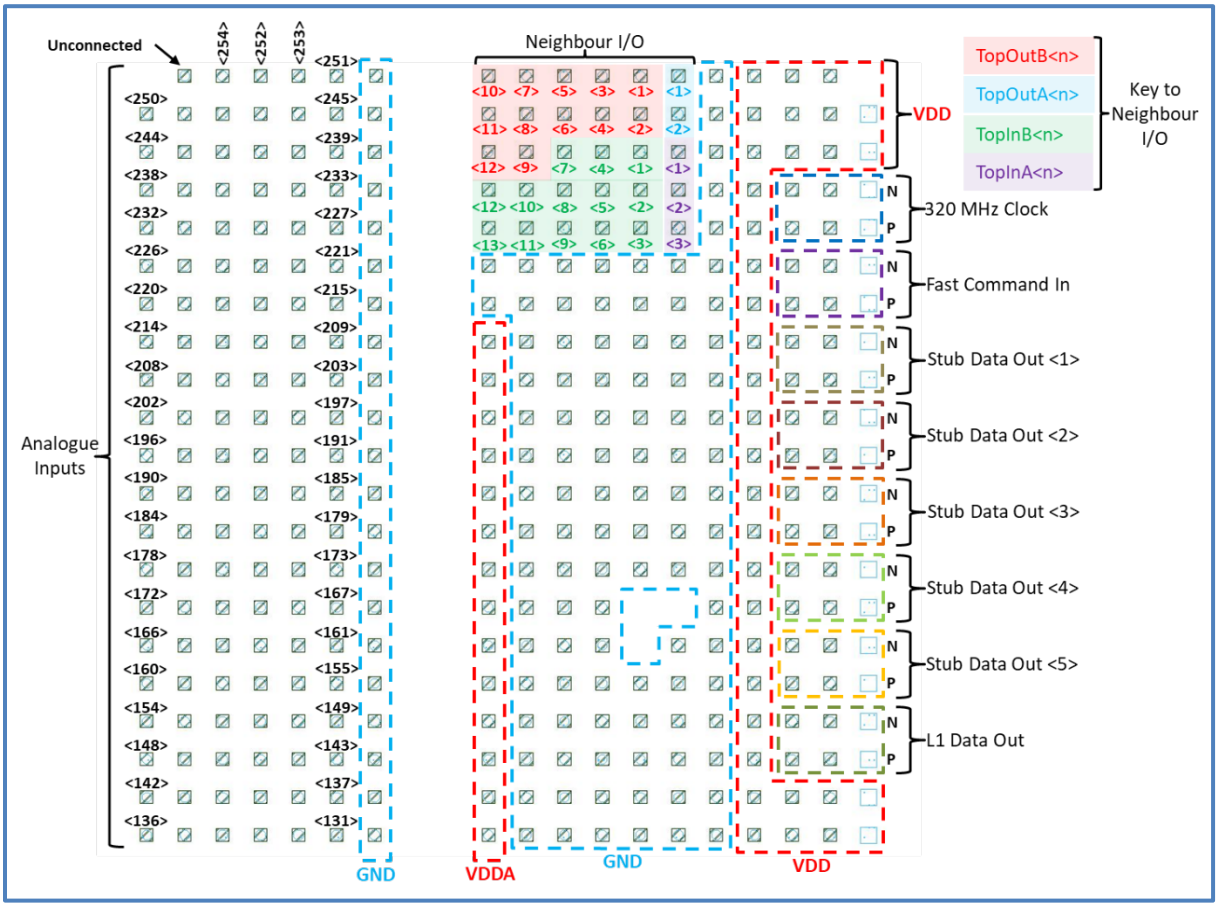

Figure 38 — CBC3 Pad layout – Top half of chip (pads facing upwards)

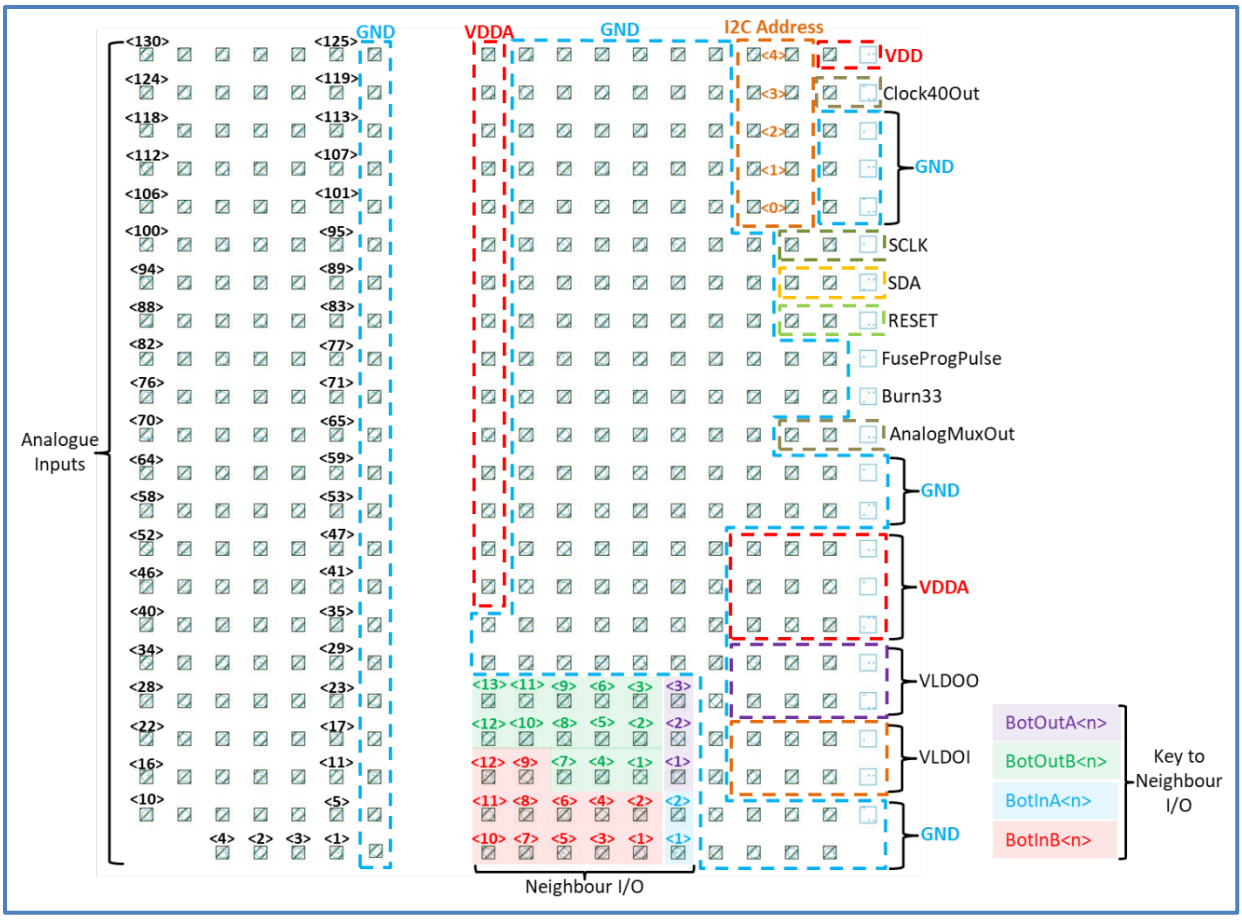

Figure 39 — CBC3.1 Pad layout – Bottom half of chip (pads facing upwards)

CBC3.1 differs from CBC3 in having two extra ground pads at the top and bottom ends of the column of ground pads to the right of the analogue input pads. These replace dummy ground pads and form a complete wall of ground pads between the sensitive analogue inputs and the digital back end of the ASIC.

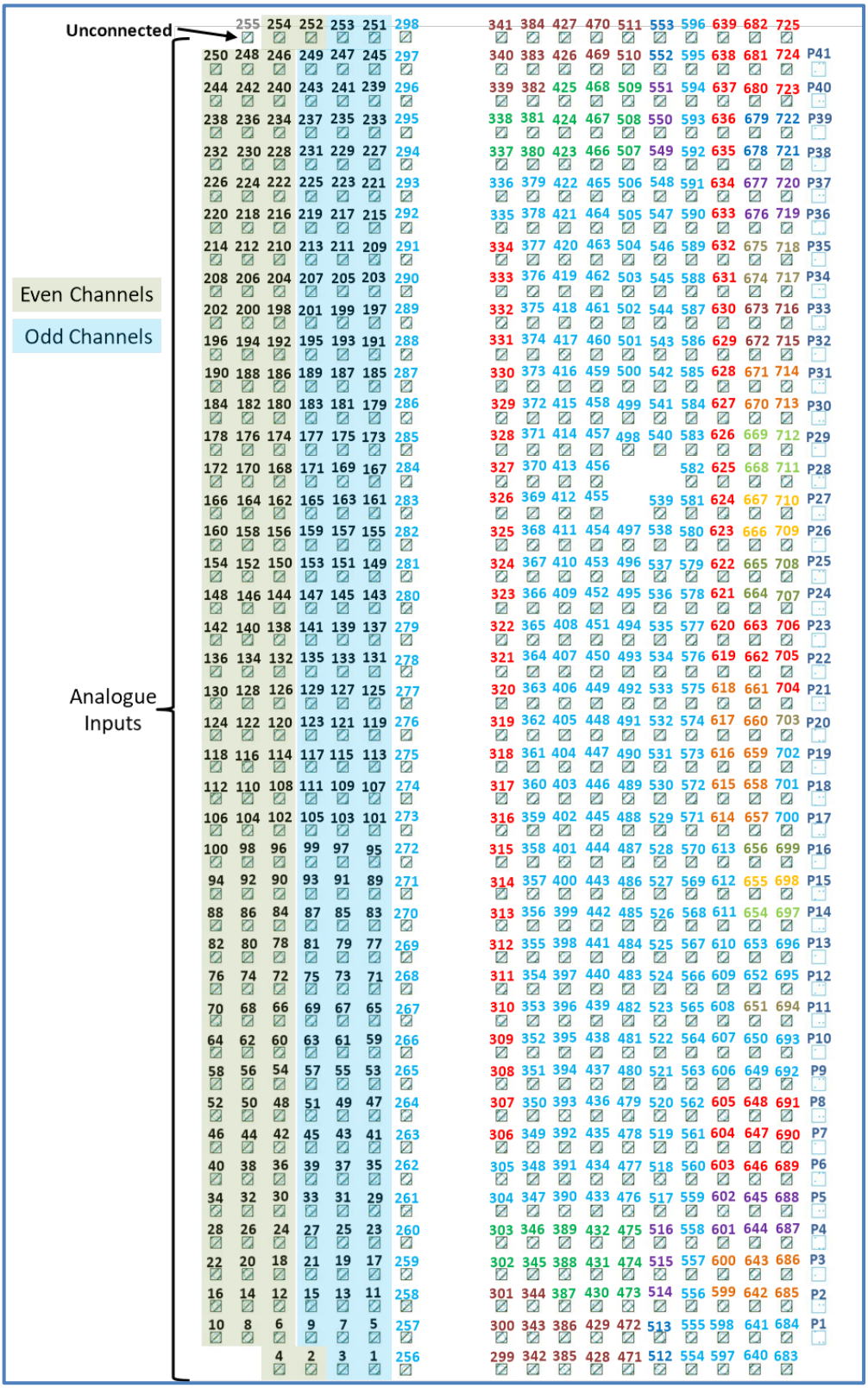

Figure 40 — CBC3.1 Pad layout – Pad Numbering (pads facing upwards)

Pad No.	Pad Name	Pad Function
1-254	Ain<254:1>	Inputs to analogue front end – Diode ESD protection only
255	N/C	This pad is for bump asymmetry and only connects to protection diodes.
256 - 298	GND	0V Ground connections
299	BotInB<10>	Hit inputs from the bottom neighbour chip Correlation layer – Active High 1.2V CMOS
300	BotInB<11>	
301	BotInB<12>	
302	BotOutB<12>	Hit outputs from Correlation layer to bottom neighbour chip – Active High 1.2V CMOS
303	BotOutB<13>	
304 - 305	GND	0V Ground connections
306 - 334	VDDA	Analogue Power connections for the analogue front end circuits
335 - 336	GND	0V Ground connections
337	TopInB<13>	Hit inputs from top neighbour chip Correlation layer – Active High 1.2V CMOS
338	TopInB<12>	
339	TopOutB<12>	Hit output from Correlation layer to top neighbour chip – Active High 1.2V CMOS
340	TopOutB<11>	
341	TopOutB<10>	
342	BotInB<7>	Hit inputs from bottom neighbour chip Correlation layer – Active High 1.2V CMOS
343	BotInB<8>	
344	BotInB<9>	
345	BotOutB<10>	Hit outputs from Correlation layer to bottom neighbour chip – Active High 1.2V CMOS
346	BotOutB<11>	
347 - 379	GND	0V Ground connections
380	TopInB<11>	Hit inputs from top neighbour chip Correlation layer – Active High 1.2V CMOS
381	TopInB<10>	
382	TopOutB<9>	Hit outputs from Correlation layer to top neighbour chip – Active High 1.2V CMOS
383	TopOutB<8>	
384	TopOutB<7>	
385	BotInB<5>	Hit inputs from bottom neighbour chip Correlation layer – Active High 1.2V CMOS
386	BotInB<6>	
387	BotOutB<7>	Hit outputs from Correlation layer to bottom neighbour chip – Active High 1.2V CMOS
388	BotOutB<8>	
389	BotOutB<9>	
390 - 422	GND	0V Ground connections
423	TopInB<9>	Hit inputs from top neighbour chip Correlation layer – Active High 1.2V CMOS
424	TopInB<8>	
425	TopInB<7>	
426	TopOutB<6>	Hit outputs from Correlation layer to top neighbour chip – Active High 1.2V CMOS
427	TopOutB<5>	
428	BotInB<3>	Hit inputs from bottom neighbour chip Correlation layer – Active High 1.2V CMOS
429	BotInB<4>	
430	BotOutB<4>	Hit outputs from Correlation layer to bottom neighbour chip – Active High 1.2V CMOS
431	BotOutB<5>	
432	BotOutB<6>	

Table 49 — CBC3.1 Bump Pad definitions (part 1)

Pad No.	Pad Name	Pad Function
433 - 465	GND	0V Ground connections
466	TopInB<6>	Hit inputs from top neighbour chip Correlation layer – Active High 1.2V CMOS
467	TopInB<5>	
468	TopInB<4>	
469	TopOutB<4>	Hit outputs from Correlation layer to top neighbour chip – Active High 1.2V CMOS
470	TopOutB<3>	
471	BotInB<1>	Hit inputs from bottom neighbour chip Correlation layer – Active High 1.2V CMOS
472	BotInB<2>	
473	BotOutB<1>	Hit outputs from Correlation layer to bottom neighbour chip – Active High 1.2V CMOS
474	BotOutB<2>	
475	BotOutB<3>	
476 - 506	GND	0V Ground connections
507	TopInB<3>	Hit inputs from top neighbour chip Correlation layer – Active High 1.2V CMOS
508	TopInB<2>	
509	TopInB<1>	
510	TopOutB<2>	Hit outputs from Correlation layer to top neighbour chip – Active High 1.2V CMOS
511	TopOutB<1>	
512	BotInA<1>	Hit inputs from bottom neighbour chip Seed Layer – Active High 1.2V CMOS
513	BotInA<2>	
514	BotOutA<1>	Hit outputs from Seed Layer to bottom neighbour chip – Active High 1.2V CMOS
515	BotOutA<2>	
516	BotOutA<3>	
517 - 548	GND	0V Ground connections
549	TopInA<3>	Hit inputs from top neighbouring chip Seed Layer – Active High 1.2V CMOS
550	TopInA<2>	
551	TopInA<1>	
552	TopOutA<2>	Hit outputs from Seed Layer to top neighbour chip – Active High 1.2V CMOS
553	TopOutA<1>	
554 - 598	GND	0V Ground connections
599 - 600	VLDOI	Input voltage for the LDO – VDD
601 - 602	VLDOO	Output voltage from the LDO – VDDA
603 - 605	VDDA	Output voltage from the LDO (~1.1V for a 1.2V LDO input voltage)
606 - 613	GND	0V Ground connections
614	I²C Address 0	Inputs for setting bits 0 to 4 of the unique I²C Address – set to either GND or VDD by connection on the module. ►NOTE: Each pad has an 88kΩ pull-up resistor to VDD included on the chip.
615	I²C Address 1	
616	I²C Address 2	
617	I²C Address 3	
618	I²C Address 4	
619 - 639	VDD	1.2V Power connections
640 - 641	GND	0V Ground connections
642 - 643	VLDOI	Input voltage for the LDO – VDD
644 - 645	VLDOO	Output voltage from the LDO – VDDA
646 - 648	VDDA	Analogue power from the LDO (~1.1V for a 1.2V LDO input voltage)
649 - 650	GND	0V Ground connections

Table 50 — CBC3.1 Bump Pad definitions (part 2)

Pad No.	Pad Name	Pad Function
651	AnalogMuxOut	Output from analogue multiplexer test feature for bias monitoring
652 - 653	GND	0V Ground connections
654	RESET	Initialising reset to circuits in fig. 29 – Active High 1.2V CMOS
655	SDA	I²C Input/Output – open drain requires pull-up resistor (~330Ω)
656	SCLK	I2C Slow Control Clock – 1.2V CMOS
657	I²C Address 0	Inputs for setting bits 0 to 4 of the unique I²C Address – set to either GND or VDD by connection on the module. ►NOTE: Each pad has an 88kΩ pull-up resistor to VDD included on the chip.
658	I²C Address 1	
659	I²C Address 2	
660	I²C Address 3	
661	I²C Address 4	
662 - 663	VDDD	1.2V Power connections
664 - 665	L1 Data Out	SLVS output for the L1 data stream (664 = +ve terminal 665 = -ve)
666 - 667	Stub Data Out 5	SLVS outputs for the Stub data stream Even number pads are the SLVS +ve terminal Odd number pads are the SLVS -ve terminal (See figure 20 for the data packet configuration)
668 - 669	Stub Data Out 4	
670 - 671	Stub Data Out 3	
672 - 673	Stub Data Out 2	
674 - 675	Stub Data Out 1	
676 - 677	FastCommandIn	SLVS input for the fast command (676 = +ve terminal & 677 = -ve)
678 - 679	320MHz Clock	SLVS input for the 320MHz Clock (678 = +ve terminal & 679 = -ve)
680 - 682	VDD	1.2V Power connections
683 - 684	GND	0V Ground connections
685 - 686	VLDOI	Input voltage for the LDO – VDD
687 - 688	VLDOO	Output voltage from the LDO – VDDA
689 - 691	VDDA	Analogue power from the LDO (~1.1V for a 1.2V LDO input voltage)
692 - 693	GND	0V Ground connections
694	AnalogMuxOut	Output of the analogue multiplexer test feature for bias monitoring
695 - 696	GND	0V Ground connections
697	RESET	Initialising reset to circuits in fig. 29 – Active High 1.2V CMOS
698	SDA	I²C Input/Output – open drain requires pull-up resistor (~330Ω)
699	SCLK	I²C Slow Control Clock – 1.2V CMOS
700 - 702	GND	0V Ground connections
703	Clock 40 Out	Test output for the on-chip generated 40MHz clock – 1.2V CMOS
704 - 706	VDD	1.2V Power connections
707 - 708	L1 Data Out	SLVS output for the L1 data stream (707 = +ve terminal, 708 = -ve)
709 - 710	Stub Data Out 5	SLVS outputs for the Stub data stream Even number pads are the SLVS -ve terminal Odd number pads are the SLVS +ve terminal (See figure 20 for the data packet configuration)
711 - 712	Stub Data Out 4	
713 - 714	Stub Data Out 3	
715 - 716	Stub Data Out 2	
717 - 718	Stub Data Out 1	
719 - 720	FastCommandIn	SLVS input for the fast command (719 = +ve terminal, 720 = -ve)
721 - 722	320MHz Clock	SLVS input for the 320MHz Clock (721 = +ve terminal & 722 = -ve)
723 - 725	VDD	1.2V Power connections

Table 51 — CBC3.1 Bump Pad definitions (part 3)

Pad No.	Pad Name	Pad Function
P1	GND	0V Ground connections
P2 - P3	VLDOI	Input voltage for the LDO – VDD
P4 - P5	VLDOO	Output voltage from the LDO – VDDA
P6 - P8	VDDA	Analogue power from the LDO (~1.1V for a 1.2V LDO input voltage)
P9 - P10	GND	0V Ground connections
P11	AnalogMuxOut	Output of the analogue multiplexer test feature for bias monitoring
P12	Burn33	Power supply to the Fuse Programming circuit (Typically 3.3V)
P13	FuseProgPulse	Control signal for the fuse programming – Active High 1.2V CMOS
P14	RESET	Initialising reset to circuits in fig. 29 – Active High 1.2V CMOS
P15	SDA	I^2C Input/Output – open drain requires pull-up resistor (~330Ω)
P16	SCLK	I^2C Slow Control Clock – 1.2V CMOS
P17 - P19	GND	0V Ground connections
P20	Clock 40 Out	Test output for the on-chip generated 40MHz clock – 1.2V CMOS
P21 - P23	VDD	1.2V Power connections
P24 - P25	L1 Data Out	SLVS output for the L1 data stream (P24 = +ve terminal, P25 = -ve)
P26 - P27	Stub Data Out 5	SLVS outputs for the Stub data stream Even number pads are the SLVS +ve terminal Odd number pads are the SLVS -ve terminal (See figure 20 for the data packet configuration)
P28 - P29	Stub Data Out 4	
P30 - P31	Stub Data Out 3	
P32 - P33	Stub Data Out 2	
P34 - P35	Stub Data Out 1	
P36 - P37	FastCommandIn	SLVS input for the fast command (P36 = +ve terminal & P37 = -ve)
P38 - P39	320MHz Clock	SLVS input for the 320MHz Clock (P38 = +ve terminal & P39 = -ve)
P40 –P41	VDD	1.2V Power connections

Table 52 — CBC3.1 Probe/Wire Bond Pad definitions

Further Reading

The CBC design has been the subject of many publications, which are listed here for the user's interest.

1. **"The CBC3 readout ASIC for CMS 2S-modules"**
   K. Uchida et al..
   To be published in NIM.

2. **"Test beam demonstration of silicon microstrip modules with transverse momentum discrimination for the future CMS tracking detector"**
   W. Adam et al. [CMS Tracker Collaboration].
   DOI:10.1088/1748-0221/13/03/P03003
   JINST 13, P03003 (2018).

3. **"CBC3: a CMS microstrip readout ASIC with logic for track-trigger modules at HL-LHC"**
   M. L. Prydderch et al..
   DOI:10.22323/1.313.0001
   PoS TWEPP -17, 001 (2018).

4. **"Development of the readout electronics for the high luminosity upgrade of the CMS outer strip tracker"**
   D. Braga, Imperial College Ph.D thesis (2016)
   https://spiral.imperial.ac.uk:8443/handle/10044/1/33725

5. **CBC2: A CMS microstrip readout ASIC with logic for track-trigger modules at HL-LHC"**
   G. Hall et al..
   DOI:10.1016/j.nima.2014.04.056
   Nucl. Instrum. Meth. A 765, 214 (2014).

6. **"Characterization of the CBC2 readout ASIC for the CMS strip-tracker high-luminosity upgrade"**
   D. Braga, G. Hall, L. Jones, P. Murray, M. Pesaresi, M. Prydderch and M. Raymond.
   DOI:10.1088/1748-0221/9/03/C03001
   JINST 9, C03001 (2014).

7. **"CBC2: A microstrip readout ASIC with coincidence logic for trigger primitives at HL-LHC"**
   D. Braga, G. Hall, L. Jones, P. Murray, M. Pesaresi, M. Prydderch and M. Raymond.
   DOI:10.1088/1748-0221/7/10/C10003
   JINST 7, C10003 (2012).

8. **"The CBC microstrip readout chip for CMS at the high luminosity LHC"**
   W. Ferguson et al..
   DOI:10.1088/1748-0221/7/08/C08006
   JINST 7, C08006 (2012).

9. **"The CMS binary chip for microstrip tracker readout at the SLHC"**
   M. Raymond et al..
   DOI:10.1088/1748-0221/7/01/C01033
   JINST 7, C01033 (2012).

NOTES