


Project Specification

**Project Name: ATLAS Calorimeter First Level Trigger-
Serialiser FPGA**

Version 1.2

July 2000

Approval	Name	Signature	Date
Project Manager	V. Perera		24/07/2000

Distribution for all updates:

Project Manager: V. Perera
Customer: A. R Gillman
Group Leader responsible for Project: R. J. Halsall
Project Managers of related projects: R. Staley
Account Manager: W. J. Haynes

**Changes to version 1.0 are listed in Appendix D - Project Change Order 1.
Changes to version 1.1 are listed in Appendix D - Project Change Order 2.**

1. Scope

This document defines the specification for the serialiser FPGA which provides the interface between the Pre-processor modules and the Cluster processor modules via the link de-serialiser chips.

2. Related projects and documents

2.1 ATLAS TDR at <http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/TDR/tdr.html>

2.2 CPM Specification at <http://hepwww.rl.ac.uk/Atlas-L1/Modules/Modules.html/>

2.3 ROD Specification at <http://hepwww.rl.ac.uk/Atlas-L1/Modules/Modules.html/>

2.4 Generic Test Module Specification at <http://hepnts1.rl.ac.uk/Atlas-L1/Modules/>

2.5 National Semiconductor Bus LVDS Serdes DS92LV1021, DS92LV1212

3. Technical Aspects

3.1 Requirements

A serialising device is required for the first-level trigger processor to convert the parallel data received from a link de-serialiser chip to 160 Mbit/s data streams for the cluster-processor ASIC (or FPGA), as well as for module-to-module data transfers via the backplane (see figure 1). Also within the same serialiser FPGA the received data should be captured and transferred to the DAQ via the read-out driver (ROD). Test signals for the clock calibration logic should also be implemented on this FPGA for setting up the data and clocks on the cluster processing chip.

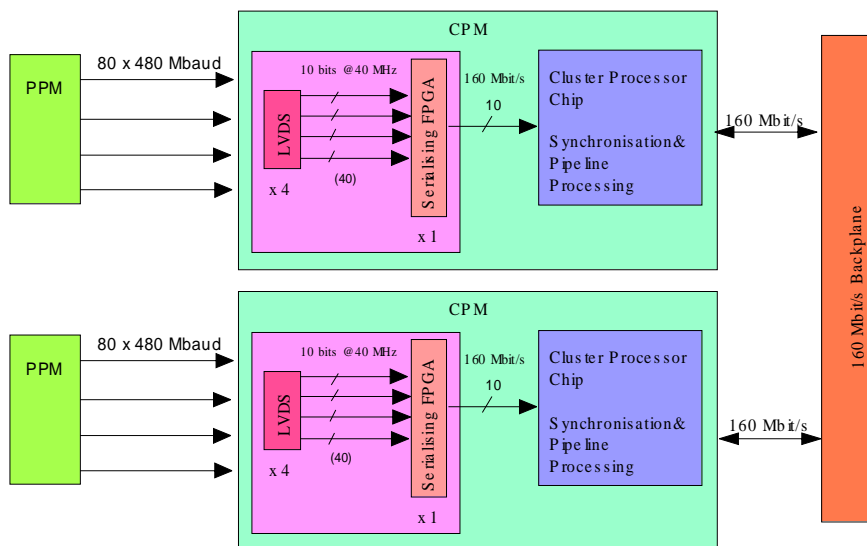


Figure 1. Data Chain from the Pre-processor to the CPMs

3.2 FPGA Specification

The first level calorimeter electron/photon and tau/hadron trigger processor will receive serial data from the pre-processor modules (PPM) using LVDS de-serialiser chips (10 bits/chip every 25 ns). As shown in figure 1, the serialiser FPGA will receive data from four BC-multiplexed channels, a total of 40 bits every 25ns. The 40 bits are split into two 20-bit fields, each of which is serialised on to five 160 Mbit/s serial links in four-bit nibbles. The 20 bit data field consists of two 8-bit trigger towers and their corresponding flag bits and 2 error checking bits. The serialiser FPGA will convert the two 8-bit words on to four 160 Mbit/s serial lines and the other four bits (two error bits + two flag bits) on to another 160 Mbit/s serial line. Following is a brief summary of the serialiser FPGA functionality (see block diagram below):

- (1) Receive four 10-bit data words every 25 ns.
- (2) Provide timing adjustment between pairs of 10-bit data words before serialising
- (3) Synchronise to system clock
- (4) Serialise each four-bit nibble at 160 Mbit/s.
- (5) Provide a four-way fan-out for each nibble
- (6) Provide 128-deep dual-port memory to capture the input data.
- (7) Allow the above memory to be used for testing and diagnostics (playback).
- (8) Provide 128-deep read-out buffer FIFO.
- (9) Transfer data from the dual port memory on to the FIFO on receipt of Enable Readout signal
- (10) Transfer the data out of the FIFO to the read-out shift register
- (11) Shift read-out data out of the FPGA @ 40 MHz
- (12) Implement the calibration sequence for calibrating the clock and data on the CP chip.
- (13) Provide JTAG scan path
- (14) Provide slow control access to all registers and memories.
- (15) Provide TTC interface to receive broadcast test commands (playback start/stop)
- (16) BC de-multiplex the readout data, with a slow-control option to bypass this process.
- (17) Carryout parity check on incoming data (data + flag bit) before capturing to dual-port RAM, and to provide scalars to count parity errors
- (18) Monitor and count link losses.
- (19) Pass data through the real-time data path with a latency of no more than 2.5 clock ticks. (See section 3.3.1 for a full definition of the latency).

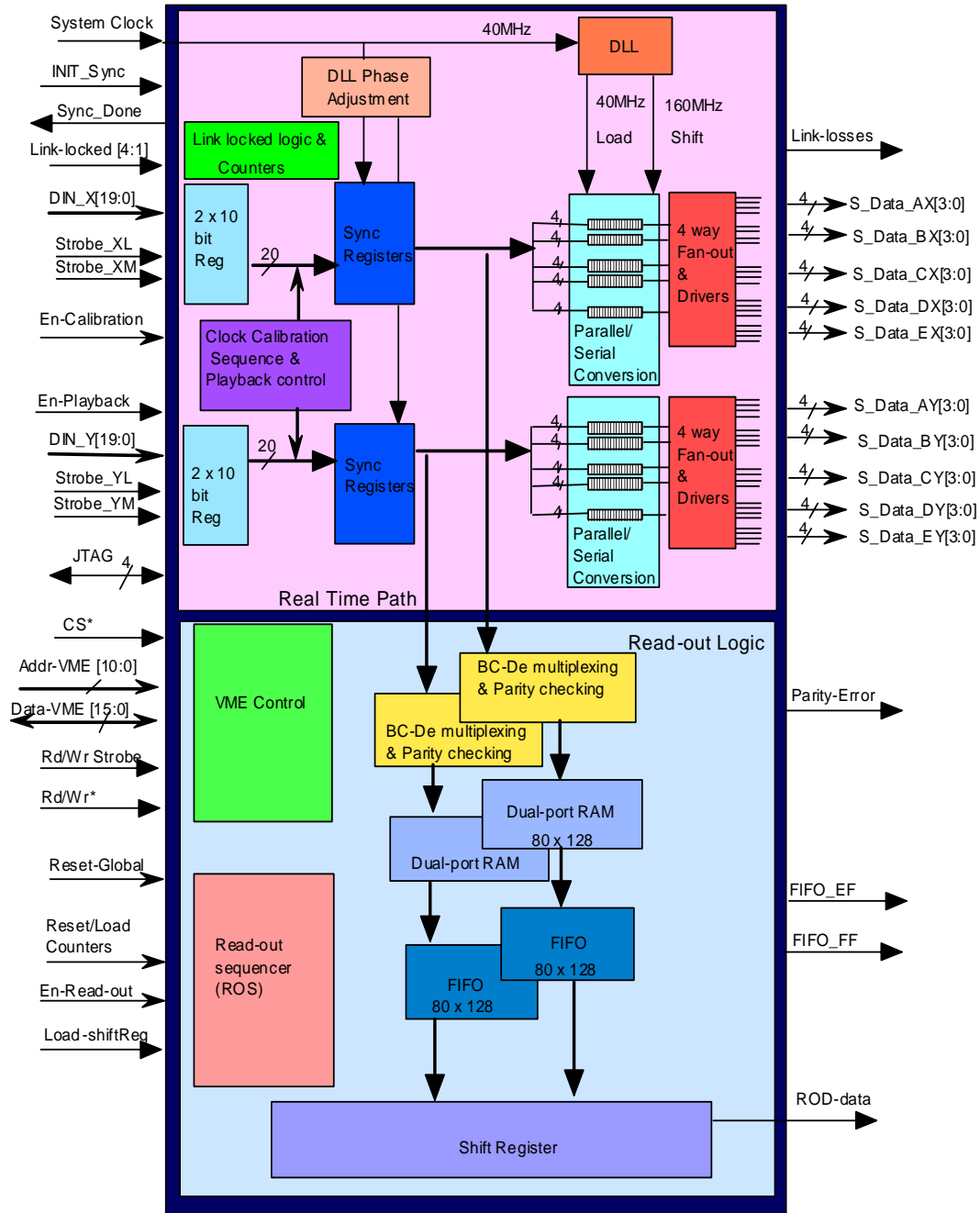


Figure 2. Serialiser Block Diagram

3.3 Logic Blocks

3.3.1 Real Time Data Path

The Serialiser FPGA receives data from four LVDS de-serialiser chips. Each chip provides 10 bits of parallel data and a recovered 40 MHz clock, which is used by the Serialiser FPGA to clock in the data. The data are then aligned to the 40 MHz LHC clock using an intermediary clock, CLK_SYNC. Figure 3 shows the logic required for one 10-bit data channel. More details are given in appendix C.

Since two 10-bit words are serialised on to five 160 Mbit/s serial lines, the data from a pair of channels must arrive within an acceptable time window. Due to the timing spread between the serialiser–de-serialiser chip-sets [2.4] additional timing adjustment may be required. The flip-flop (*) shown in figure 3 provides this extra adjustment by allowing early signals to be delayed by one clock tick.

After the data have been synchronised the parallel-to-serial converter takes a 4-bit nibble of data and serialises it to 160 Mbit/s, and provides a four-way fan-out. The data which pass through the real-time path are also copied to a dual port RAM. This RAM can be readout to the DAQ (3.3.3)

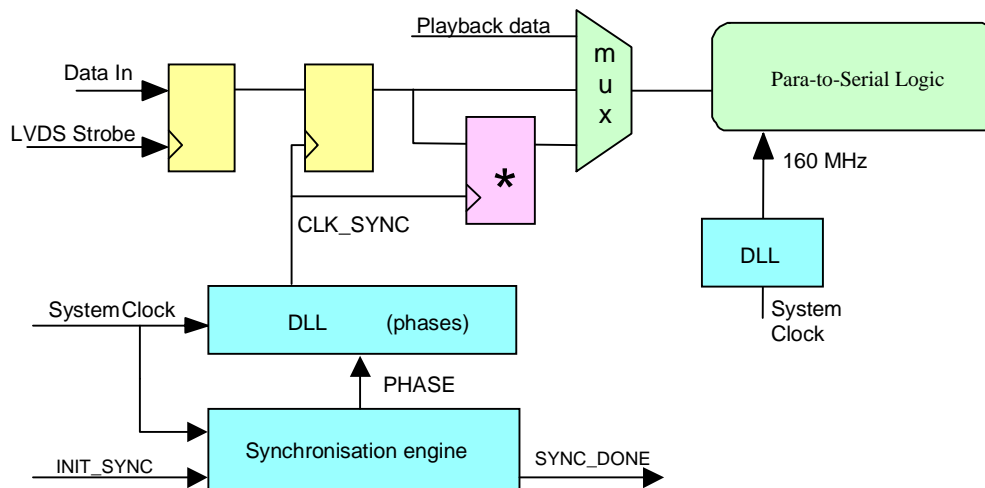


Figure 3. Data Capture and Synchronisation Logic

The latency of the real-time data path is defined as the period between the receipt of the LVDS clock, which clocks the incoming data, and the output of the fourth data bit in each serialised nibble. This latency should be no more than 2.5 LHC clock ticks, assuming no optional delays are added. (These delays are added to re-synchronise early data channels and therefore do not contribute to the total latency of the trigger processor). See figure 4.

Link ready logic will monitor the link ready signals from the 480 Mbit/s serialiser chips on the CPM and will scale them using counters. If there were any link losses this will be indicated with an output (Link-losses) on the device for external monitoring and establishing unreliable serial links from the pre-processor to the CPM.

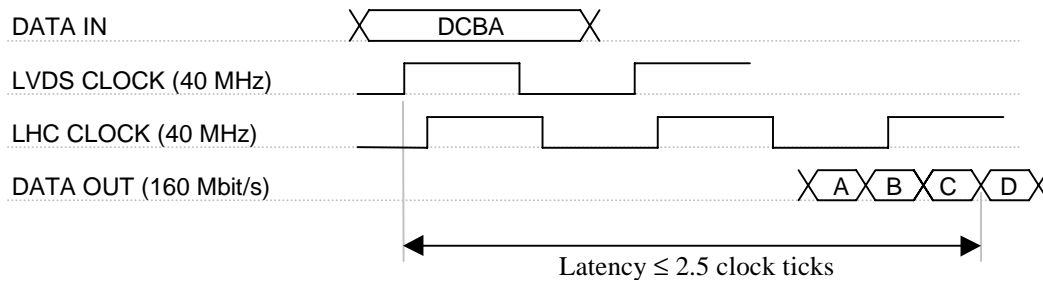


Figure 4. The latency of the real-time data path. Four parallel bits of input data are shown being converted to a serial output.

3.3.2 Delay Locked Loop (DLL)

The 160 MHz clock required for the 160 Mbit/s serialiser block is derived from the 40 MHz system clock using delay locked loops.

3.3.3 Read out Logic

Figure 5 shows a block diagram of the logic associated with the readout of the Serialiser FPGA. The input data are extracted from the real-time data path after synchronisation (see figure 2). These data are BC-de-multiplexed and written to a dual-port memory, which also captures the status of the link-ready signals and parity-error flags associated with the data (see section 3.3.3.2). The data for one bunch crossing, after BC-de-multiplexing, is referred to as a ‘slice’.

The purpose of the dual-port RAM is to hold a history of data. Slices of data are written to the RAM and read from it at addresses given by the Wr Address Counter and Rd Address Counter respectively. These 7-bit counters increment every bunch crossing and roll over to zero; a slice is thus stored for 128 bunch crossings before being overwritten. The Rd Address Counter is offset from the Wr Address Counter by the value held in the Offset register. This offset should be set up to compensate for the latency of the trigger system from the time at which a data slice is written into the dual-port RAM, to the time at which the En-readout signal for that slice is received by the Serialiser. The Rd Address Counter and Wr Address counter are reset and reloaded, respectively, by the Reset/Load Counters signal.

The read out of the Serialiser is controlled by the En-readout and Load-ShiftReg signals, which are generated on the CPM by the read-out controller (ROC) and are sampled by the Serialiser synchronously with the rising edge of the system clock. If, on this clock edge, En-readout is high, the data slice pointed to by the Rd Address Counter is copied from the dual-port RAM to the FIFO buffer. The Serialiser output FIFO-EF indicates the status of the FIFO buffer: if FIFO-EF flag is low the FIFO contains data.

If, on a rising clock edge, Load-ShiftReg is high, one data slice from the FIFO is transferred to the shift register and shifted out of the Serialiser at 40 MHz. This shift register is permanently enabled, and once the valid 80 bits of data are shifted out zeros will follow.

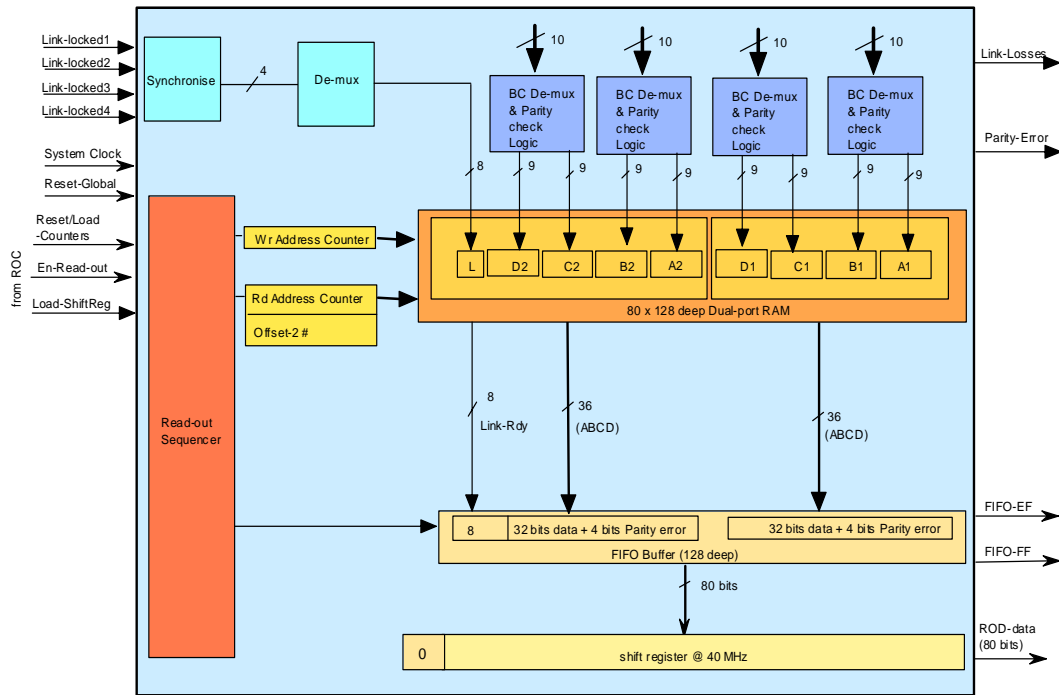


Figure 5. Read-Out Associated Logic on the FPGA

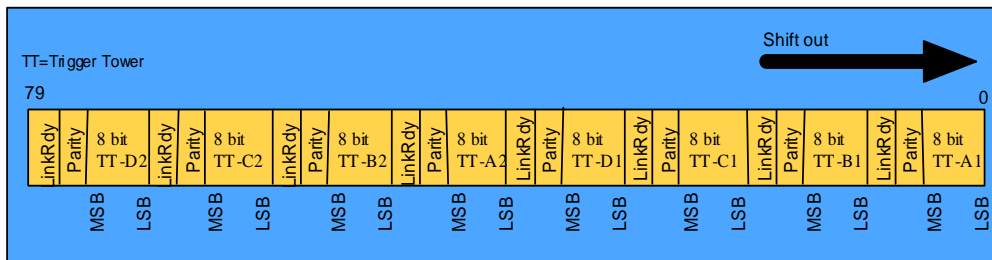


Figure 6. One slice of readout data from the Serialiser

The format of the readout data output by the Serialiser is shown in Figure 6. The following sections describe the BC-de-multiplexing and parity-checking logic implemented by the Serialiser, and the requirements that the Serialiser places on the CPM ROC.

3.3.3.1 BC De-multiplexing and Parity Check Logic

The data supplied to the BC-de-multiplexing logic is extracted from the real-time data path before parallel-to-serial conversion. Four blocks of identical BC de-multiplexing logic are required to de-multiplex the 40 bits of parallel data received by the serialiser.

Each block of de-multiplexing logic examines 10 bits of data, consisting of an eight-bit field of calorimeter data and the associated flag and error bits. If non-zero calorimeter data are found, two consecutive bunch crossings are processed and the data are assigned to a trigger tower and a bunch crossing according to their flag bits. Those trigger towers that are not assigned data for a bunch crossing are set to zero, with error bits set to indicate no error. BC de-multiplexing transforms each 10-bit field into an 18-bit field (2

x [8-bit data + parity]), containing data for two trigger towers. See Appendix A for a VHDL model of this process. It is possible to run the pre-processor with no BC multiplexing, in which case the BC-de-multiplexing on the serialiser will also be switched off. This will be done by setting the appropriate bits in the control register (3.7.2.1). This will have no effect on the volume of data read-out however, since 80 bits will still be transferred, with some fields being set to zero.

This block also carries out parity checking on the received bits. This should be done in parallel to the BC-de-multiplexing logic. If this logic finds any of the trigger tower data in error, then the corresponding 'parity error' bit will be set in the dual-port RAM for all towers and time slices potentially affected by the parity error. In addition to setting the parity error field in the dual-port RAM, the parity errors will be scaled using counters. If any of the counters are >0 then this will be indicated on a device pin (Parity-Error) so that the CPM can monitor this signal before it needs to read the parity error counters.

3.3.3.2 Requirements of the Serialiser for the CPM ROC

The readout controller (ROC) logic on the CPM controls the readout of the Serialiser. The following signals are used:

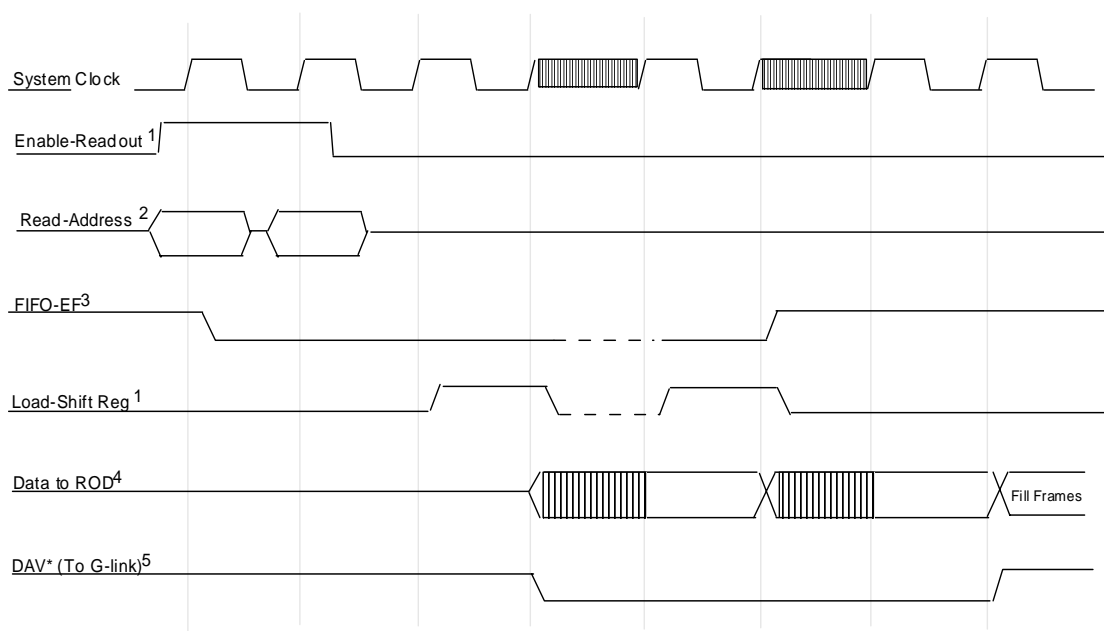
- En-readout: input to the Serialiser from the ROC
- Load-shift: input to the Serialiser from the ROC
- Reset/Load Counters: input to the Serialiser from the ROC.
- FIFO-EF: output from the Serialiser to the ROC.

The Serialiser samples the En-readout, Load-ShiftReg and Reset/Load Counter signals synchronously with the rising edge of the board clock. On receipt of an L1A the ROC should read out the Serialiser using the following procedure.

1. Assert the En-readout signal to the Serialiser. For every bunch crossing during which the En-readout signal is asserted, one slice of data will be transferred within the Serialiser from the dual-port RAM to the FIFO buffer. It is possible to read out multiple data slices for a single L1A, up to a maximum of five. This can be controlled by the duration of the En-readout signal. To read out a five-slice event, for example, the En-readout signal should be held active for five bunch crossings.
2. Monitor the FIFO-EF output from the Serialiser. When this is low there are data in the Serialiser FIFO and the Load-ShiftReg signal may be asserted (see below).
3. Assert Load-ShiftReg for one bunch crossing. This causes one slice of data to be read from the FIFO and loaded into the shift register. The slice is then clocked serially out of the Serialiser at 40 MHz.
4. Wait for 80 bunch crossings. This is the time taken to output one slice of data from the Serialiser. En-readout may be asserted again during this period (which should happen if another L1A arrives), but Load-ShiftReg should not, as this would overwrite the data contained in the shift register.
5. For the readout of a multiple-slice event, repeat steps 2. to 4. as many times as required (which should correspond to the duration of the En-readout signal issued).

Figure 7 shows a timing diagram for the signals needed to control the read out of the Serialiser.

In addition to the above process, the ROC also supplies the Reset/Load Counters signal to the Serialiser. This clears the Wr Address Counter and loads the Rd Address Counter with the value contained in the Offset register. To avoid discontinuities in the data written to the dual port RAM, the Wr Address Counter should only be cleared when it is due to roll over to zero. This will also correspond to the point at which the Rd Address Counter is due to reach its Offset value. In normal operation, therefore, the Reset/Load Counters signal will have no effect; it will only effect counters that have lost synchronisation.



- Note: 1. Input signals to the Serialiser FPGA from the ROC logic on the CPM
 2. Internally generated on the Serialiser FPGA
 3. Output from Serialiser FPGA to the ROC
 4. Data output to ROD from the G-Link via the ROC
 5. Data available signal from the ROC to G-link to frame the DAQ data

Figure 7. Timing diagram shows the signalling requirements for controlling the readout sequence for transferring data for a two-slice event

3.3.4 Play Back Mode

The dual port RAM will be used as a source of test data to test the rest of the data chain without the need for data coming from the pre-processor, by playing back the data from the dual-port RAM on to the serialiser block. The playback function can be initiated via VME or TTC broadcast command for simultaneous start.

3.3.5 Clock Calibration

Unlike the G-links and the LVDS links, where the clock is encoded with the data and is recovered at the receiving end, the 160 Mbit/s data transmitted from the serialiser FPGA does not have a clock-encoding scheme. Therefore the CP chip has to carryout a clock calibration procedure to select the 160 MHz clock phase to capture the incoming serial data at 160 Mbit/s correctly. This block on the serialiser FPGA will generate the

sequence of A5s (hex) to carryout the clock calibration procedure on the CP chips. This can be enabled via VME or TTC broadcast command.

3.4 Technology

Xilinx Virtex-E family XCV100-E FPGAs will be used

3.4.1 Input Output Technology

3.4.2.1 All inputs and outputs will be low voltage CMOS (LVCMOS) compatible.

3.4.2.2 Set-up and hold times:

set-up time (t_{su}) = 3ns, hold time (t_h) = 3 ns

3.5 Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Supply Voltage core	Vddc	- 5%	1.8	+ 5%	V
Supply Voltage I/O	VddI/O	- 5%	3.3	+ 5%	V
Output Drive Low/High	I_{OL}/I_{OH}			24	mA
Output Low Voltage	V_{OL}	GND		0.6	V
Output High Voltage	V_{OH}	2		3.3	V
Input Low Voltage	V_{IL}	GND		0.8	V
Input High Voltage	V_{IH}	2.0		3.3	V
System Clock	CLK	-	-	40	MHz
Junction Temperature	T_J			85	°C
Power (estimate)	P(3V3)			150	mW
Power (estimate)	P(1V8)			300	mW

3.6 FPGA Pin Definition and Package

3.6.1 Pin Definition

Name	Pin	Type	Signal
Addr-VME[0] Addr-VME[1] Addr-VME[2] Addr-VME[3] Addr-VME[4] Addr-VME[5] Addr-VME[7] Addr-VME[8] Addr-VME[9] Addr-VME[10]		I	11 bit address lines for accessing memory and control registers
CS*		I	Chip select signal, enables read/write access to the memory and registers via slow controls.
Data-VME[0] Data-VME[1] Data-VME[2] Data-VME[3] Data-VME[4] Data-VME[5] Data-VME[6] Data-VME[7] Data-VME[8] Data-VME[9] Data-VME[10] Data-VME[11] Data-VME[12] Data-VME[13] Data-VME[14] Data-VME[15]		I/O	Bi-directional data bus for slow controls.
DIN_XL[0]		I	Parity bit for LVDS link XL
DIN_XL[1]		I	BC flag for LVDS link XL
DIN_XL[2] DIN_XL[3] DIN_XL[4] DIN_XL[5] DIN_XL[6] DIN_XL[7] DIN_XL[8] DIN_XL[9]		I	8-bit data for LVDS link XL

DIN_XM[0]		I	Parity bit for LVDS link XM
DIN_XM[1]		I	BC flag for LVDS link XM
DIN_XM[2]		I	8-bit data for LVDS link XM
DIN_XM[3]			
DIN_XM[4]			
DIN_XM[5]			
DIN_XM[6]			
DIN_XM[7]			
DIN_XM[8]			
DIN_XM[9]			
DIN_YL[0]		I	Parity bit for LVDS link YL
DIN_YL[1]		I	BC flag for LVDS link YL
DIN_YL[2]		I	8-bit data for LVDS link YL
DIN_YL[3]			
DIN_YL[4]			
DIN_YL[5]			
DIN_YL[6]			
DIN_YL[7]			
DIN_YL[8]			
DIN_YL[9]			
DIN_YM[0]		I	Parity bit for LVDS link YM
DIN_YM[1]		I	BC flag for LVDS link YM
DIN_YM[2]		I	8-bit data for LVDS link YM
DIN_YM[3]			
DIN_YM[4]			
DIN_YM[5]			
DIN_YM[6]			
DIN_YM[7]			
DIN_YM[8]			
DIN_YM[9]			
En-Calibration		I	Enables the calibration data pattern
En-Playback		I	Enables playback mode
En-Read-out		I	Initiates the readout sequence
FIFO-EF		O	FIFO empty flag (high if empty)
FIFO-FF		O	FIFO Full flag (high if full)
INIT_SYNC		I	Initialise self-synchronisation logic
Link-locked1		I	Indicates that the link is locked
Link-locked2		I	Indicates that the link is locked
Link-locked3		I	Indicates that the link is locked
Link-locked4		I	Indicates that the link is locked
Link-losses		O	Indicates that one or more links are down
Load-ShiftReg		I	Signal from ROC to transfer data from the FIFO to shift register
Parity-Error		O	Indicates parity errors detected

Reset/Load Counters		I	Resets the dual-port RAM write counter and loads the read counter with the L1A offset
Reset-Global		I	Resets all registers except for the control registers.
Rd/Wr-Strobe		I	Dual port memory read and write strobe.
Rd/Wr*		I	Dual port memory read and write enable signal.
Rod-Data		O	Serialised read-out data
S_DATA_AX0 S_DATA_AX1 S_DATA_AX2 S_DATA_AX3 S_DATA_BX0 S_DATA_BX1 S_DATA_BX2 S_DATA_BX3 S_DATA_CX0 S_DATA_CX1 S_DATA_CX2 S_DATA_CX3 S_DATA_DX0 S_DATA_DX1 S_DATA_DX2 S_DATA_DX3 S_DATA_EX0 S_DATA_EX1 S_DATA_EX2 S_DATA_EX3		O	160 Mbit/s serial outputs. A,B,C,D are data, and E is control (flags +parity)
S_DATA_AY0 S_DATA_AY1 S_DATA_AY2 S_DATA_AY3 S_DATA_BY0 S_DATA_BY1 S_DATA_BY2 S_DATA_BY3 S_DATA_CY0 S_DATA_CY1 S_DATA_CY2 S_DATA_CY3 S_DATA_DY0 S_DATA_DY1 S_DATA_DY2 S_DATA_DY3 S_DATA_EY0 S_DATA_EY1 S_DATA_EY2 S_DATA_EY3		O	160 Mbit/s serial outputs. A,B,C,D are data, and E is control (flags +parity)

Spare-Roc1		I/O	Spare pin connected to ROC
Strobe_XL		I	Strobe from the de-serialiser.
Strobe_XM		I	Strobe from the de-serialiser.
Strobe_YL		I	Strobe from the de-serialiser.
Strobe_YM		I	Strobe from the de-serialiser.
Sync_Done		O	Status of self-synchronisation logic
System_Clock		I	40 MHz system clock
TD0		O	JTAG output
TDI		I	JTAG input
TCK		I	JTAG Clock
TMS		I	JTAG Mode select
Vdd-1		Supply	+ 3.3 Volts supply for periphery
Gnd-1		Supply	Ground periphery
Vdd-2		Supply	+ 1.8 Volts supply for core
Gnd-2		Supply	Ground core

NOTE: I = input; O = output; I/O = bi-directional, * = active low

Total Number of I/O = 137

3.6.2 Package

The package will be either PQ/HQ240 (32 mm x 32 mm) or FG256 (17 mm x 17 mm)

3.7 Programming Model

3.7.1 Guidelines

These are to aid the software control of the Serialiser FPGA.

1. All registers can be read by the computer, hence there are no ‘write only’ registers.
 - 1.1 All Status Registers shall be Read-Only registers.
 - 1.2 All Control Registers shall be Read/Write registers.
2. If the computer tries to read or write a value that the device itself is able to modify at the same time, the data integrity cannot be guaranteed.

3.7.2 Memory Map

The FPGA is addressed using a chip select (CS), the 11-bit address bus and the 16-bit data bus.

Address	Type	Bits	Name	Description
0	RO	16	Version_Register	Firmware version number
1	WR	16	Control Register	BC-Mux off, Sending A, Sending B
2	RO	16	Status Register	See below
3	RW	8	Rd Address_Offset	Offset from Wr Address
4	RO	7	FIFO Write Counter	Write counter of FIFO
5	RO	7	FIFO Read Counter	Read counter of FIFO
6	RO	8	Clk Phase Register	See below
7	RW	4	Delay Register	See below
8	RO	8	Link Ready Counter	Count of BCs with link not ready for LVDS channel XL
9	RO	8	Link Ready Counter	Count of BCs with link not ready for LVDS channel XM
A	RO	8	Link Ready Counter	Count of BCs with link not ready for LVDS channel YL
B	RO	8	Link Ready Counter	Count of BCs with link not ready for LVDS channel YM
C	RO	8	Parity Error Counter	
D	RO	8	Parity Error Counter	
E	RO	8	Parity Error Counter	
F	RO	8	Parity Error Counter	
80–FF	RW	16	128 × 80 Dual Port RAM	Dual-port RAM bits 0–15
100–17F	RW	16	128 × 80 Dual Port RAM	Dual-port RAM bits 16–31
180–1FF	RW	16	128 × 80 Dual Port RAM	Dual-port RAM bits 32–47
200–27F	RW	16	128 × 80 Dual Port RAM	Dual-port RAM bits 48–63
280–2FF	RW	16	128 × 80 Dual Port RAM	Dual-port RAM bits 64–79
300–37F	RW	16	128 × 80 FIFO	Access to FIFO, bits 0–15
380–3FF	RW	16	128 × 80 FIFO	Access to FIFO, bits 16–31
400–47F	RW	16	128 × 80 FIFO	Access to FIFO, bits 32–47
480–4FF	RW	16	128 × 80 FIFO	Access to FIFO, bits 48–63
500–57F	RW	16	128 × 80 FIFO	Access to FIFO, bits 64–79

3.7.2.1 Control Register

Bit	Description
0	BC-Mux on/off (default on)
1	Mux off sending A or B (default A)
2	Reset Link error counters #
3	Reset Parity error counters #

Read back zero

3.7.2.2 Status Register

Bit	Description
0	FIFO empty flag
1	FIFO full flag
2	lock status of 160 MHz clock DLLs
3	lock status of synchronisation DLL
4	status of self-synchronisation logic
5	calibration status
6	data-in status
7	BC de-multiplexing status
8	playback status

3.7.2.3 Clock Phase Register

Bit	Description
0–1	Phase of synchronisation clock for LVDS channel XL
2–3	Phase of synchronisation clock for LVDS channel XM
4–5	Phase of synchronisation clock for LVDS channel YL
6–7	Phase of synchronisation clock for LVDS channel YM

3.7.2.4 Delay Register

Bit	Description
0	Enables addition of extra delay (1 BC) to LVDS channel XL
1	Enables addition of extra delay (1 BC) to LVDS channel XM
2	Enables addition of extra delay (1 BC) to LVDS channel YL
3	Enables addition of extra delay (1 BC) to LVDS channel YM

3.8 Handling Precautions

The device will be static sensitive, hence must be handled with proper care.

3.9 Design Methodology

The designs will be fully synchronous, and will be described in VHDL at register-transfer-level (RTL) then synthesised to an appropriate FPGA

3.10 CAE

CAE tools such as; Cadence, Synopsis, VHDL, Renoir, Leonardo and Modelsim will be used.

3.11 Design Verification

Functional, post synthesis, post layout, timing verifications and design rule checks will be carried out.

3.12 Testability

To make it easy to test the FPGA, the following will be considered and implemented where possible. If any of these cannot be implemented, then the reasons should be documented and an alternative approach taken.

3.12.1 Power up and Reset Strategy

The following issues will be considered in the design:

1. Asynchronous reset
2. Provision for internal/external power-up reset
3. Provision for resetting individual logic blocks for debugging purposes where necessary.

3.13 JTAG Boundary Scan

JTAG port will be provided for use with board level boundary scan testing.

3.14 Testing

3.14.1 Strategy

Full speed tests will be carried out using a purpose built test (Generic Test Module)

3.14.2 Test equipment

- (1) VME crate
- (2) 16/32 bit VME interface
- (3) Computer to run software
- (4) Generic Test Module (GTM)
- (6) Logic analyser
- (7) Oscilloscope
- (8) VME Extenders

3.15 Storage, Shipping & Installation

The FPGAs will be of-the-shelf components and will be purchased and will be stored in the R25 anti-static storage area if required by the customer.

3.16 Maintenance and further orders

During the prototype stage support will be given to the users to implement, and modify the FPGA firmware when necessary.

3.17 Software

A test engineer from the System Support Group will develop the LabView test software.

4. Project Management

4.1 Personnel

		RAL Ext.	RAL Location
Customer:	A. R. Gillman	5521	R1, 1.54
Project Manager:	V. Perera	5692	R68, 2.31
Project Engineer	I. Brawn	6816	R68, 2.09

4.2 Deliverables

4.2.1 To the Customer:

1. Serialiser FPGA firmware
2. Test report

4.2.2 From the Customer:

Software to drive the GTM may be required

4.3 Project plan (Milestones)

- | | |
|------------------|---------------|
| 1. PDR | February/1999 |
| 2. FDR | Q1/2000 |
| 3. Start Testing | Q2/2000 |
| 4. CR | Q2/2001 |

4.4 Design Reviews

The customer must be present at the PDR and the CR. If the customer wishes, he may attend the FDR.

The progress of the project will be reported on a monthly basis in the ATLAS Calorimeter First-level Trigger project monitor form.

4.5 Training

Training will be carried out as required on the job.

4.6 Costs and finance

All manufacturing, assembly and component costs will be charged to FK40000.

4.7 Intellectual Property Rights (IPR) and Confidentiality

All background and foreground Intellectual Property Rights in this project will remain with CLRC. The customer will have unrestricted rights to items listed under deliverables (4.2). If the customer requires other data, then an appropriate protective agreement should be in place before releasing such data.

4.8 Safety

General laboratory safety codes apply.

4.9 Environmental impact

None

4.9.1 Disposal

RAL will dispose of the devices at end of their life.

4.9.2 EMC

Since these are components, they will be outside the scope of the EMC regulations. However the modules incorporating these devices must function as designed, without malfunction or unacceptable degradation of performance due to electromagnetic interference (EMI) within their intended operational environment, the electronic modules shall comply with specifications intended to ensure electromagnetic compatibility.

Appendix A — BC de-multiplexing

The VHDL code below shows how the 10 bits of data received from one LVDS link, and the link-ready signal, are processed by the BC de-multiplexing logic. The Serialiser requires a total of four instances of this logic.

```
-----
architecture RTL of BC_DEMUX is
-----
- ipb 20-8-99
--

signal BCF      : std_logic;      -- bunch-crossing(BC) flag from BC n
signal BCF_NEXT : std_logic;      --           "           from n+1
signal BCF_PREV : std_logic;      --           "           from n-1
signal DATA    : std_logic_vector(8 downto 0); -- data from BC n
signal DATA_NEXT : std_logic_vector(8 downto 0);
signal PHASE    : std_logic; -- keeps track of where we are in data pair

constant NO_DATA : std_logic_vector := "100000000"; -- no data, odd parity
constant READY   : std_logic       := '1';         -- link ready state

signal LR_NEXT   : std_logic;      -- Link Ready from BC n+1
signal LR        : std_logic;      -- Link Ready from BC n

-----
-
begin

DATA_NEXT <= DATA_IN(8 downto 0);
BCF_NEXT  <= DATA_IN(9);
LR_NEXT   <= LINK_RDY;

A_PROC: process (PHASE, BCF, BCF_PREV, DATA)
-- combinatorial logic to determine output on A channel
--
begin
DATA_A      <= NO_DATA;
LINK_RDY_A <= READY;
if ((PHASE = '0' and BCF = '0')
    or (PHASE = '1' and BCF = '1' and BCF_PREV = '1')) then
DATA_A      <= DATA;
LINK_RDY_A <= LR;
end if;
end process;

B_PROC: process (PHASE, BCF, BCF_PREV, BCF_NEXT, DATA, DATA_NEXT)
-- combinatorial logic to determine output on B channel
--
begin
DATA_B      <= NO_DATA;
LINK_RDY_B <= READY;
if ((PHASE = '0' and BCF = '1')
    or (PHASE = '1' and BCF = '1' and BCF_PREV = '0')) then
DATA_B      <= DATA;
LINK_RDY_B <= LR;
elsif (PHASE = '0' and BCF = '0' and BCF_NEXT = '0') then
DATA_B      <= DATA_NEXT;
LINK_RDY_B <= LR_NEXT;
end if;
end process;

```

```

PHASE_GEN: process (CLK)
-- Data arrive in pairs in consecutive BCs. PHASE
-- keeps track whether we are processing first word
-- or second word/no data.
--
begin
  if (CLK'event and CLK = '1') then
    if (DATA_NEXT = NO_DATA) then --ignore parity
      PHASE <= '1';
    else
      PHASE <= not PHASE;
    end if;
  end if;
end process;

REG_PROC: process (RESET, CLK, BCF_NEXT, BCF, BCF_PREV, DATA_NEXT)
-- a pipeline memory: holds a history of DATA and BCF
-- which the combinatorial logic (A/B_PROC) needs to
-- examine.
--
begin
  if (RESET = '1') then
    BCF_PREV <= '0';
    BCF <= '0';
    DATA <= NO_DATA;
    LR <= READY;
  elsif (CLK'event and CLK = '1') then
    BCF_PREV <= BCF;
    BCF <= BCF_NEXT;
    DATA <= DATA_NEXT;
    LR <= LR_NEXT;
  end if;
end process;

end RTL;

```

Appendix B — Data format

The Serialiser FPGA receives four channels of 10-bit input data at 40 MHz and outputs 10 channels of serial data at 160 MHz. Some re-organisation of the data is involved in this process.

Each of the four channels received by the Serialiser consists of eight bits of calorimeter data plus two control bits (parity and BC flag). Before serial-to-parallel conversion these data are split into nibbles, with calorimeter data from one input channel being split across two nibbles, and the control bits from pairs of input channels being grouped together. Each nibble is then serialised and output. See figure B1.

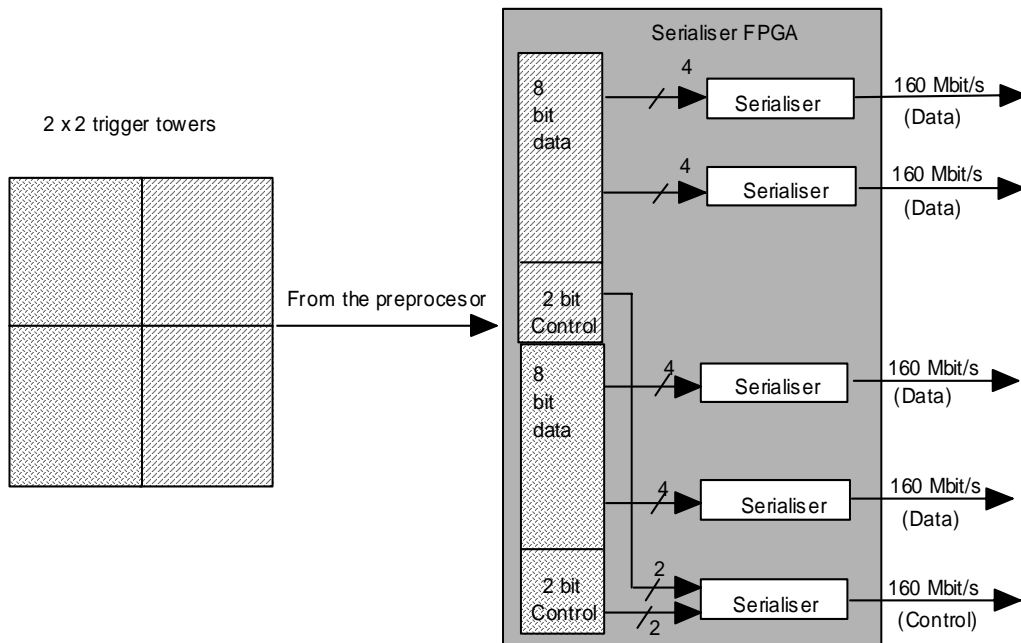


Figure B1. The reformatting of the data received by the Serialiser for a pair of input channels.

Appendix C — Data Synchronisation

From the LVDS de-serialiser chips the Serialiser FPGA receives four channels of 10-bit data. Each channel is accompanied by a 40 MHz clock and this is used to clock the data into the Serialiser. The processing of the data, however, is performed using the LHC clock, and to avoid possible set-up or hold violations on the data passing between these two timing regimes, an intermediary clock is used, CLK_SYNC. See figure C1.

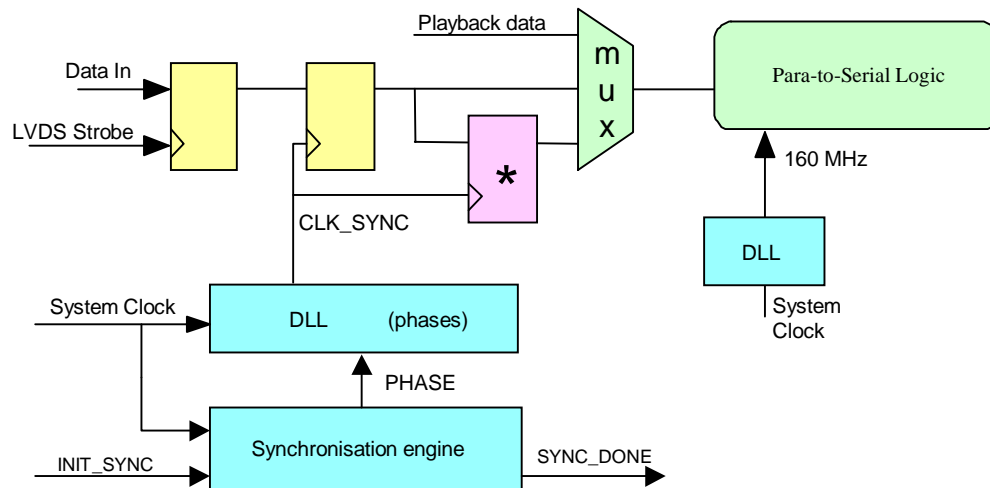


Figure C1. Data Capture and Synchronisation Logic.

CLK_SYNC can be defined individually for each data channel to have a phase of 0°, 90°, 180° or 270° with respect to the LHC clock. The user can set these phases via slow controls, or the Serialiser can be set instructed to determine the necessary phases automatically.

For this process the Serialiser should be sent a toggling data pattern, with all bits high for one clock cycle, alternating with all bits low. The automatic synchronisation process can then be initiated by pulsing the INIT_SYNC pin. This causes the Serialiser to test every phase of CLK_SYNC, for each channel, looking for errors in the toggling data pattern indicative of set-up/hold violations. The phase of CLK_SYNC for a channel is then set to that which produces the least number of errors (hopefully zero). Once this process is complete and the phase of CLK_SYNC has been defined for all four channels, the Serialiser pulls the SYNC_DONE pin high. The phases selected by the automatic synchronisation process can be inspected and overwritten via the slow controls. Figure C2 shows a flow diagram of the automatic synchronisation process.

Due to the time spread across the LVDS serialiser–de-serialiser chip sets, it is possible that data from the same bunch-crossing that travel across different links will arrive at the Serialiser to be clocked into different LHC clock cycles. To correct for this an optional flip-flop (*) is included in the real-time path to delay early signals if required. No mechanism is provided within the Serialiser for *automatically* determining whether or not this extra delay is needed for a particular channel. This decision must be taken at a global level and communicated to the Serialiser via the slow controls.

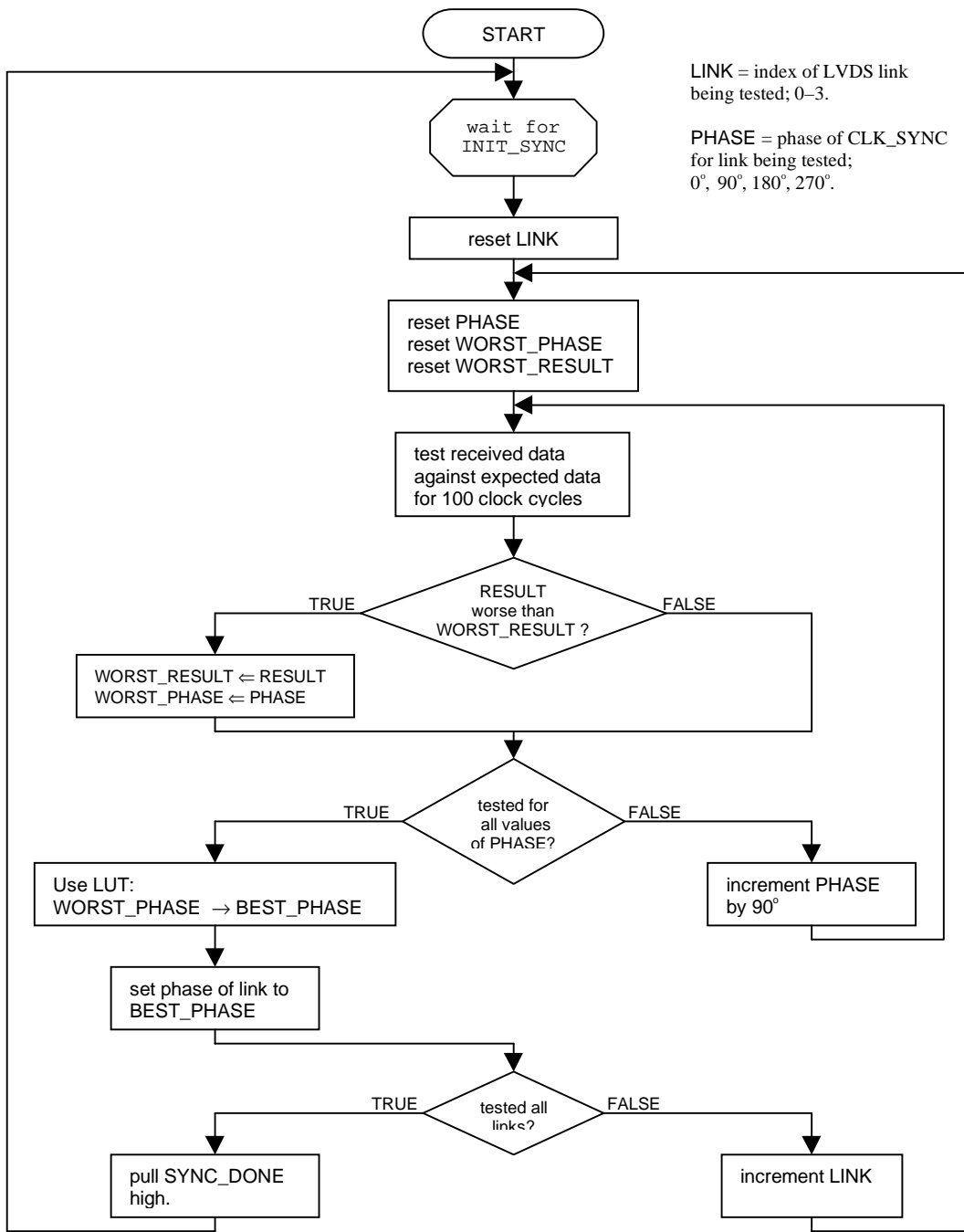


Figure C2. The clock synchronisation process.

Appendix D - Project Change Orders

Project Change Order

Serial Number:-001

Project Name:- ATLAS Calorimeter First Level Trigger –Serialiser FPGA

Date: 26/06/2000

Description of change:

1. Transferring the parity bit from the serialiser to the ROD has to change since now the parity calculation includes data as well as the flag bit.

The serial data format as seen by the ROD (ROD specification page 8) shows that we are only sending the 8 bit data plus the parity and the link ready signal from the serialiser to the ROD. With the proposal of Paul's (after specification version 1.0 was released) on how to protect the data, the parity is calculated including the flag. Therefore sending only the parity is no use for the DAQ end since it will be not possible to check for parity errors without the flag bit.

This will be handled by the serialiser as follows:

1. Calculate the parity on the serialiser in parallel with the BC De-mux logic (this is already done for the CP chip)
 2. In place of the parity bit use this location to indicate parity error (e.g. if bit is set then do not trust the data). This way DAQ does not require to calculate parity, either trust the data or not depending on the parity error bit. Doing it this way, there will be no changes to the data format to the ROD, hence no changes required to the ROD design.
 3. Provide scalars on the serialiser for counting the errors and an output pin to indicate an error if scalar is > 0 . - page 10
2. Another minor change was requested and agreed, to provide an output to indicate when a link loses lock (Logical OR of all link locked signals) - Page 5 last paragraph
3. Readout logic section has been re-written to clarify some issues. No changes to the functionality except an output (Data-in-Sreg) to ROC to indicate when data is available on the shift register. - Pages 6-10

Specification version 1.1 will be released with the above changes.

Customer approval when required (sign, date)

A. R. Gillman

Change implemented. Project Manager (sign, date)

V. J. O. Perera

Project Change Order

Serial Number:-002

Project Name:- ATLAS Calorimeter First Level Trigger –Serialiser FPGA

Date: 17/07/2000

Description of change:

Readout logic section has been modified to simplify the handshake between the ROC and the Serialiser, as well as the Read address which was centrally (ROC) generated is now moved on to the individual Serialiser FPGAs- Pages 6-9

Specification version 1.2 will be released with the above changes.

Customer approval when required (sign, date)

A. R. Gillman

Change implemented. Project Manager (sign, date)

V. J. O. Perera