CERN Trajectory Measurement System PUPE-FPGA Specification

Dr Andrew C McCormick, Alpha Data

Version 1.0: 18/10/2007

Introduction

This PUPE FPGA document specifies the general structure and operation of the PUPE FPGA firmware and the register interface accessible by the software. It is based upon the technical specification presented in [1] and tender document [2].

Overview

The detailed operation of the TMS cycles is described elsewhere [1,3]. Each FPGA contains 3 independent pick-up processing units. Each of these modules performs the following functions: phase locking (to particle bunches or a reference signal); baseline restoration and integration of the sum signal (Σ) and horizontal and vertical difference signals (Δx,Δy); recording of the integrated signals in a 256MB circular buffer; capturing of RAW signal data for diagnostic purposes.

The operation of the baseline restoration and integration is controlled through programmable phase tables, indicating when to start and stop integration tasks within each orbit.

The cycle goes through a number of states, with state changes indicated by a number of timing signals. The phase table and other synchronisation parameters depend on this state, and so a programmable state table is provided to select the correct phase table and the other parameters for that state.

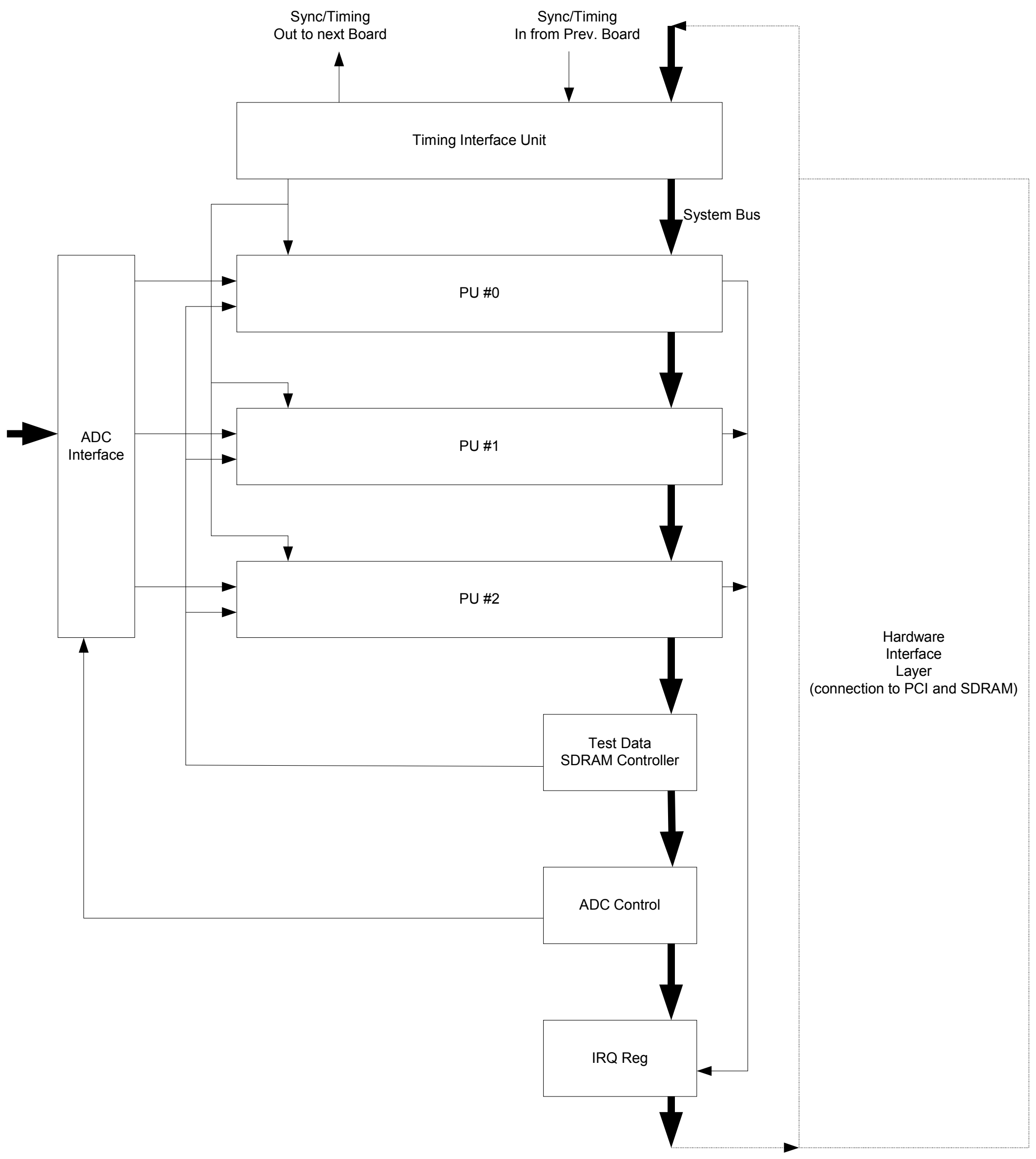
The timing signals coming into the module can be overridden by software to allow testing of the individual unit.

There are also a number of modules external to the pick-up processing units within the FPGA:

A timing interface module is used at the FPGA level, to control the timing bus between FPGAs in the system, since the timing signals come into one FPGA and must be broadcast to all the others.

A test memory bank can be used to inject synthetic test signals in place of the ADC inputs. This memory bank is also used as scratch memory for the data pattern engine.

FPGA level control is also provided for the 9 channel ADC inputs and for handling interrupts.



Sync/Timing Out to next Board
Sync/Timing In from Prev. Board
Timing Interface Unit
System Bus
PU #0
ADC Interface
PU #1
PU #2
Hardware Interface Layer (connection to PCI and SDRAM)
Test Data SDRAM Controller
ADC Control
IRQ Reg

Figure 1: FPGA Top Level Design

Figure 1 shows the main elements in the FPGA design including the 3 processing units (PU's). The interfacing across the PCI bus and the interfacing with the SDRAM use an existing design library which provides simple implementation of registers, PCI accessible dual port Block RAM , PCI accessible shared access SDRAM as well as interrupt handling capabilities.

Processing Units

Each individual processing unit performs a number of tasks:

Timing and input data can be set to use either real data or test values, programmable by software.

The PLL-SPU unit performs the main processing tasks, locking onto the phase and integrating the signals.

The data logger, records the outputs of the PLL-SPU unit and writes them to SDRAM. It also updates a cycle timing table every 1ms with references to the SDRAM cycle data table, to aid the software in finding the required data.
The data logger also acts as an embedded logic analyser, capturing the RAW data input signals and control signals for diagnostic purposes.

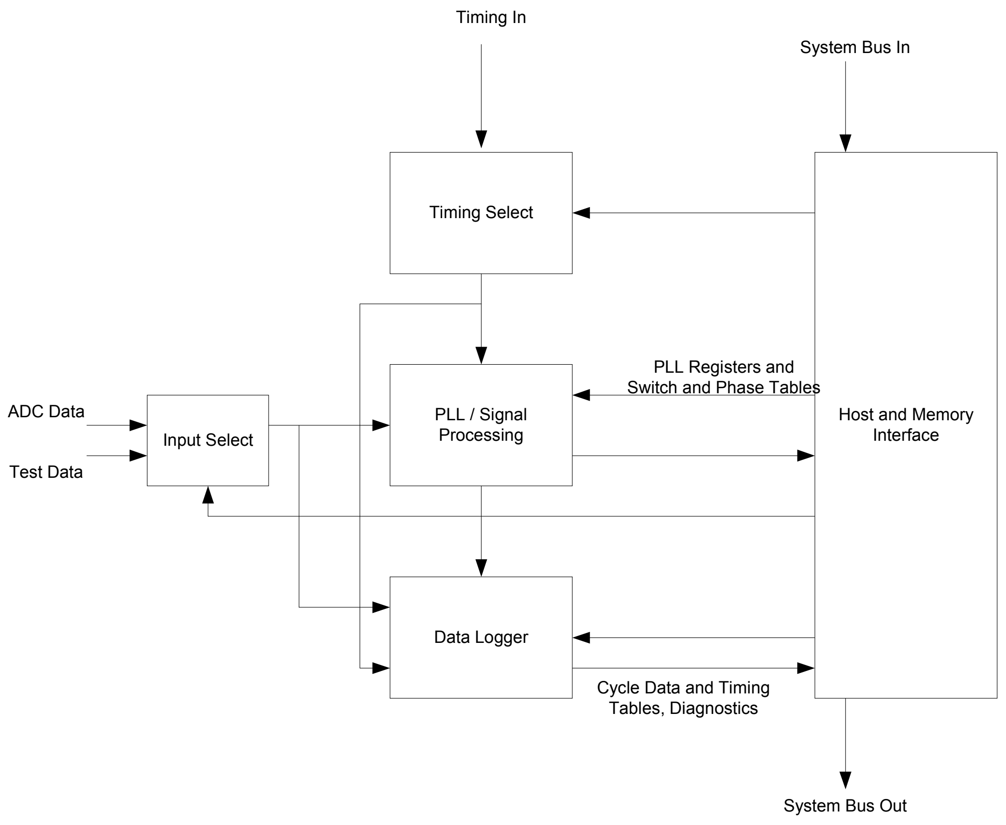


Figure 2: PU Core

Figure 2 shows the structure of the processing unit. A VHDL library is used to provide an abstract interface between the application and the PCI bus/SDRAM.

PLL-SPU

The core processing occurs in the PLL-SPU (phase locked loop and signal processing unit).

The PLL component will follow the specification in [1], fig 12 as closely as possible. The baseline restoration and integration will follow the specification [1] fig.9 and example VHDL [4] as closely as possible.

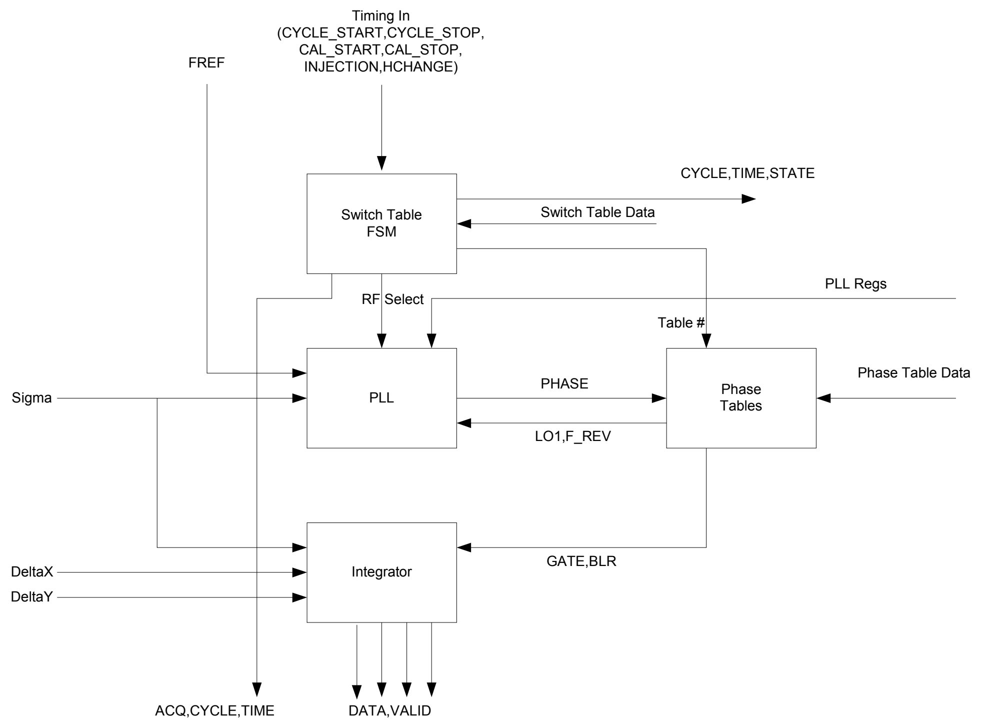


Figure 3: PLL-SPU

Figure 3, shows the structure of the PLL-SPU block.

The phase table block uses a single 8192 x 8 bit block RAM to store up to 16 phase tables. The read address is generated by concatenating the phase (least significant 9 bits) and the table number (most significant 4 bits), to output the LO,F_REV,GATE and BLR signals.

The switch table FSM controls the PLL based on the timing information signals. The FSM has 16 states of which 14 are programmable. States 0xE and 0xF are reserved for Error and Finished/Idle states. The state number also corresponds to the Phase Table number.

The state machine is initially set to state 0xF, where it waits until the CYCLE_START event occurs, up which it switches to state 0x0. If the machine ends up in state 0xE, a register write can move it to state 0xF, to clear the error.
Other state transitions depend on the contents of the switch table.

The switch table consists of 14 elements of width 28 bits. For each table element bits 0-7 are used as control bits, e.g. bit 0 sets the ACQ signal to enable the acquisition of data, bit 1 sets the RF select bit in the PLL for filter 1, bit 2 sets the RF select bit in the PLL for filter 2, bit 3 Selects between the filter 1 or filter 2, to feedback into the PLL. Bits 4 and 5 are used to select the LO pulse source. By default LO1 and LO2 are generated from the phase table, however by setting these bits, the MSB of the PLL is used as LO1 or LO2. This bit is not phase shifted so allows all PLLs to lock to the same FREF signal.

Bits 8-31 are used for 6 state transitions, indicating the state to change to when one of 5 timing events occurs:

Bits 8-11 : CYCLE_STOP
Bits 12-15 : CAL_STOP
Bits 16-19 : CAL_START
Bits 20-23 : INJECTION
Bits 24-27 : HCHANGE
Bits 31-28 : EVENT_DELAYED

EVENT_DELAYED is a special event which occurs 16 revolutions after every other event. It can be ignored by setting bits 31:28 to the current switch state. Alternatively it can be used to allow a second switch setting change after an event.

A typical switch table set up could be:

State	ED	HC	INJ	CaStrt	CaStp	CyStp	PLL LO2	PLL LO1	FS	RF2	RF1	ACQ	Hex Value
0	0	E	3	1	E	E	0	0	0	0	0	0	0E31EE00
1	1	E	E	E	2	E	0	0	0	0	0	1	1EEE2E01
2	2	E	3	E	E	E	0	0	0	0	0	0	2E3EEE00
3	3	4	E	E	E	F	0	0	0	0	1	1	34EEEF03
4	4	5	E	E	E	F	0	0	0	0	1	1	45EEEF03
5	5	6	E	E	E	F	0	0	0	0	1	1	56EEEF03
6	6	E	E	E	E	F	0	0	0	0	1	1	6EEEEF03
E	X	X	X	X	X	X	X	X	0	0	0	0	00000000
F	X	X	X	X	X	X	X	X	0	0	0	0	00000000

This allows the calibration to run with the phase information in table 1. After injection (whether Calibration occurs or not), phase table #3 is used. H-Changes can then move on to using phase tables, 4,5 and 6. Out of sequence timing events send the state machine to the Error state.

Bunch Mean Value Computation

Mean values of the beam and selectable bunches will also be computed at the bunch rate. 2 bunch mean bits will be added to the phase table to use as enable strobes to push data into the filters used to obtain the mean values. These will be set at the end of the gate pulse of any bunch to be added to the mean computation. Every 1ms, the output of these filters will be sampled and stored in a bunch mean table.

FPGA Memory Map

The FPGA occupies 4MB of address space in the users application. This 4MB window can be accessed directly, or through the use of Slave Mode DMA. Direct access is recommended for accessing register spaces. Slave mode DMA is recommended for any accessed to the memory regions.

The basic memory map consists of 3 areas:
System Control Registers : controlling the memory paging and interrupts
Application Registers : for use by the PUPE modules

2 MB Memory window : for accessing SDRAM or Block RAM banks

System Memory Map:

Address	Name	R/W	Description
0x000000	IMEM_REG	RW	Block RAM Index
0x000008	LOCKED	RO	DCM Locked Status
0x000010	ADDR_REG	RW	PAGE Register for SDRAM access
0x000018	MEM_REG	RW	SDRAM Bank Select Register
0x000020	IER	RW	Interrupt Enable Register
0x000028	ISR	RW	Interrupt Status Register
0x000030	MEM_RAS	RW	SDRAM Read Address Scaling
0x000038	MEM_RAO	RW	SDRAM Read Address Offset
0x000040 -0x000058	MEM_GNTx	WO	Memory Grant Registers
0x000800 -0x0009FF	Application Registers	RW	See Application Memory map
0x200000 -0x3FFFFF	Memory Window	RW	2MB window for accessing SDRAM or Block RAM

System Registers

0x000000 IMEM_REG
This register is used to select the bank of block RAM in the application to map into the Memory Window. The application may implement many banks of block RAM and make them accessible to the PCI bus. This register selects the appropriate bank, which will be accessible if MEM_REG 0x0000018 is set to 0

0x000008 LOCKED
This register indicates if the internal FPGA DCMs for clocking the SDRAMs have achieved lock status.

0x000010 ADDR_REG
This register is used to select a 2MB page in the SDRAM memory bank The significant bits are from bits 19 to 24.

0x000018 MEM_REG
This register is used to lock an SDRAM bank for PCI access. Setting bit 3 switches the memory window from internal to external memory. Bits 0 to 2 select the bank (0,1,2 or 3). Read only bits 8 to 11 indicate if the internal FPGA PU process has been granted access to the memory, and bits 16 to 19 indicate if the internal PU process is requesting access to these memories. Note any bit set between 8 and 11 indicates that PCI access of the bank is prohibited.

0x000020 IER
The interrupt enable register allows up to 32 interrupts sources to be enabled.

0x000028 ISR
The interrupt status register indicates which of 32 interrupt sources generated the interrupt. The bit is cleared by writing a 1.

0x000030 MEM_RAS

0x000038 MEM_RAO
SDRAM Memory address scaling and offset. These 2 registers combine with the ADDR_REG to allow scatter gather reads to the SDRAM, using the Data Pattern Engine. These registers affect the way the local bus address is translated into a SDRAM memory address. Each register is only 5 bits allowing values up to 32.

The default value for MEM_RAS is 1, and MEM_RAO is 0. The SDRAM read address (64-bit granularity) is calculated from the following:

```
page := ADDR_REG >> 19;
laddr := (FPGA_ADDR-0x200000) >> 3;

SDRAM_ADDR := (page*2^18+laddr)*MEM_RAS+MEM_RAO
```

0x000040 MEM_GNT0
0x000048 MEM_GNT1
0x000050 MEM_GNT2
0x000058 MEM_GNT3

These 4 write only registers control the Request/Grant lines for the SDRAM bank dual port access. By default they are set to auto-grant mode (i.e. if the PU side code requests the memory, it always gets it). Write 0 to disable access to a bank (e.g. before performing a local bus read), write 1 to grant access to a bank and write 2 to enable auto-grant mode. These registers are overridden by the MEM_REG. Setting bit 3 of MEM_REG disables the memory grant line on the FPGA side of the memory controller.

Application Register Banks

The application registers will be split into 4 banks, 1 for registers to control the whole FPGA, and 3 identical banks of registers for each PU.

FPGA Registers

Address	Name	R/W	Description
0x000800	FIRMWARE	RO	Firmware ID code “CN” + version numbers
0x000808	ADC	RW	ADC Control Register
0x000810	TIMING_IO	RW	Timing I/O Register
0x000818	TESTCTRL	RW	Test Data Control Register
0x000820	TESTLEN	RW	Test Data Pattern Length

0x000800 FIRMWARE

This register contains the firmware identification tag and firmware version number. This is 4 Bytes : “C”,“N”, Major Version, Minor Version. The version numbers are 8 bit integers.

0x000808 ADC

This register controls the 9 channel ADC interface module. Bits 0:8 select the Analogue Channel source for the 9 channels. Setting the bit enables input from the ADC, clearing the bit selects test input from the test SDRAM bank.

Bits 0,3 and 6 also enable the 3 banks of 3 ADC devices.

Bit 9 is connected to the RESET line of the synthesiser control logic. This should be pulsed high to stabilise the clocks.

Bit 10 enables the orange SW LED
Bit 11 enables the red SW LED

0x000810 TIMING_IO

This register controls the input and output options of the timing IO, allowing the FPGA to accept the master timing input signals, and echo them onto the local timing bus.

Bits 1:6 can be used to set the control signals:

cycle_start : bit 1
cycle_stop : bit 2
cal_start : bit 3
cal_stop : bit 4
injection : bit 5
hchange : bit 6

Bits 8:15 are used to select the test signals, rather than external input signals.

ten_mhz : bit 8
cycle_start : bit 9
cycle_stop : bit 10
cal_start : bit 11
cal_stop : bit 12
injection : bit 13
hchange : bit 14
fref : bit 15

The test 10MHz clock is generated internally in the FPGA.
The test FREF signals is generated using a programmable counter, with FREF toggled every N * 8ns. Where N cycles is set by bits 16:27 of the TIMING_IO register.

Bits 28:30 are used to select the timing bus IO.
master_bplane : bit 28
When set to 0, this Tri-states the Back-Plane Timing IO pins and uses these as inputs.
When set to 1, the Master or Test timing signals are output on the Back Plane.
master_header : bit 29
When set to 0, this Tri-states the Header Timing IO pins and uses these as inputs.
When set to 1, the Master or Test timing signals are output on the Back Plane.
use_bplane : bit 30 is used to select either the Back Plane IO or the Header IO as the PUPE local timing bus. Even when Master mode is enabled, the timing bus inputs are delayed bythe IOB input delay so that they are synchronised across all PUPE boards.

0x000818 TESTCTRL

This register enables the operation of the test pattern buffer, allowing SDRAM bank 3 to be used as a source of data in place of the ADC inputs.

Bits 24:0 indicate the start address of the test pattern.
Bit 25 Enables the test pattern reading from the SDRAM

0x000820 TESTLEN

This register indicates the size of the test data set in bank 3, and indicates when it needs to repeat its cycle.

PU Registers (one bank per PU)

3 Banks of Registers are located at addresses
0x880-0x8E0 PU #0
0x900-0x960 PU #1
0x980-0x9E0 PU #2

The registers for PU #0 are as follows

Address	Name	R/W	Description
0x000880	CONTROL	RW	PU General Control and Status register
0x000888	CYCLE	RW	Cycle number
0x000890	TIME	RO	Time in ms from start of cycle
0x000898	TIME_TBLADDR	RO	Last write address in timing table
0x0008A0	PLL_FREQUENCY	RW	PLL Reference orbit frequency
0x0008A8	PLL_FREQDELAY	RW	PLL frequency load delay
0x0008B0	PLL_PHASEDELAY	RW	PLL phase delay
0x0008B8	PLL_GAIN	RW	PLL gain
0x0008C0	DDS_FREQ_MIN	RW	PLL DDS minimum frequency
0x0008C8	DDS_FREQ_MAX	RW	PLL DDS maximum frequency
0x0008D0	DIAG_CTRL	RW	Diagnostics Control/Status
0x0008D8	DIAG_TRIGGER	RW	Diagnostics Trigger
0x0008E0	DIAG_DELAY	RW	Diagnostics Capture Delay
0x0008E8	TEST	RW	Timing Test

The same registers for PU #1 and PU #2 are mapped into the ranges 0x900-0x968 and 0x980-0x9E8 respectively.

0x000880, 0x000900, 0x000980 CONTROL

This register provides general purpose control bits (e.g. reset and enable bits) and status bits (e.g. switch state information)

Bit	Name	Description
0	Init	
1	Loop Control	
2	DDS Frequency Limit Enable	
4:7	Switch State	
8:11	PLL Use Neighbours	Allows the 3 PLLs to combine mixed signals to improve lock time. Bits 8 and 9 are used to enable summation with neighbours for filter 1. Bits 10 and 11 enable summation with neighbours for filter 2.

0x000888, 0x000908, 0x000988 CYCLE

This register indicates the current cycle number. This is incremented every time a Start Cycle Timing Event occurs.

0x000890, 0x000910, 0x000990 TIME

This register indicates the time in ms from the last Start Cycle event.

0x000898, 0x000918, 0x000998 TIMETBL_ADDR

This register indicates the last address written to in the Cycle Timing Table.

0x0008A0, 0x000920, 0x0009A0 PLL_FREQUENCY

Initial value of orbit frequency. Should be set to (B->f)*2^32/125e6

0x0008A8, 0x000928, 0x0009A8 PLL_FREQDELAY

Delay (in ms) after CYCLE_START before setting PLL_FREQUNECY

0x0008B0, 0x000930, 0x0009B0 PLL_PHASEDELAY

Phase offset in clock cycles to compensate for PU's position in the ring

0x0008B8, 0x000938, 0x0009B8 PLL_GAIN

Gain of PLL

0x0008C0, 0x000940, 0x0009C0 DDS_FREQ_MIN

Minimum frequency parameter for DDS

0x0008C8, 0x000948, 0x0009C8 DDS_FREQ_MAX

Maximum frequency parameter for DDS

0x0008D0, 0x000950, 0x0009D0 DIAG_CTRL

Diagnostics control and status register. This register contains control bits for arming the trigger, selecting the sample rate, and selecting a start time.

Bit	Name	Description
0	Arm	Set to Arm the Diagnostics Data Capture
1	Capturing	Indicates that data are being recorded
2:3	Source	Selects one of 4 –64 bit sources for the data and trigger
4	Store Trigger	Store trigger rather than upper 8 bits of data
5	AND/OR	Selects AND/OR or trigger operation. 0 => Start if OR_reduce(trigger_reg AND trigger) 1 => Start if not AND_reduce(trigger_reg OR trigger)
6	Data/Trigger	User either 8 bit trigger or 32 bits of data to trigger data capture
7	Wait CLK	Wait until CTIME counter reaches start time
12:8	CLK Select	If bit 12 is clear, divide 125 MHz clk by 1,2,5,…100000 depending on bits 11:8, to generate a clock enable strobe for recording the diagnostic signals. If bit 12 is set, bit 8 selects fref (1) or a 1 ms pulse (0) as the clock enable strobe.

0x0008D8, 0x000958, 0x0009D8 DIAG_TRIGGER

Diagnostics trigger register. The least significant 8 bits of this register are used to set the trigger condition which is tested against the 8 trigger signals, if bit 6 of DIAG_CTRL is clear. If bit 6 is set, all 32 bits are compared with the least significant 32 bits of the selected diagnostics bus.

The 8 trigger signals are:

ten_mhz_clk
cycle_start
cycle_stop
cal_start
cal_stop
injection
hchange
fref

These form an 8 bit bus with bits (7 downto 0):
fref & hchange & injection & cal_stop & cal_start & cycle_stop & cycle_start & ten_mhz_clk;

The 4x 64 bit diagnostics buses are:

Bus 0 - bits (63 downto 0)
diagnostics(63 downto 56) <= frev_in & pll_msb & phase_table_msb & LO_pulse1 & event_delayed_switch & injection_switch & hchange_switch & LO_pulse2;
diagnostics(55 downto 48) <= (hchange_switch and phase_table_msb_falling) & sel_filter & rf_select2 & rf_select1 & phase_table_data(7 downto 6) & GATE_pulse & BLR_pulse;
diagnostics(47 downto 32) <= switch_state & sigma(13 downto 2);
diagnostics(31 downto 26) <= hchange & injection & cal_stop & cal_start & cycle_stop & cycle_start;
diagnostics(25 downto 0) <= dds_freq(31 downto 6);

Bus 1 – bits(127 downto 64)
diagnostics(127 downto 120) <= frev_in & pll_msb & phase_table_msb & LO_pulse1 & sigma(13 downto 11) & LO_pulse2;
diagnostics(115 downto 102) <= mult_out1;
diagnostics(101 downto 88) <= mult_out2;
diagnostics(87 downto 64) <= f_error(23 downto 0);

Bus 2 – bits (191 downto 128)
diagnostics(191 downto 184) <= frev_in & pll_msb & phase_table_msb & LO_pulse1 & sigma(13 downto 11) & LO_pulse2;
diagnostics(183) <= GATE_pulse;
diagnostics(182 downto 160) <= b0(22 downto 0);
diagnostics(159 downto 128) <= SXT(result0_tmp,32);

Bus 3 – bits (255 downto 192)
diagnostics(255 downto 248) <= frev_in & pll_msb & phase_table_msb & LO_pulse1 & BLR_data_out_0(23) & BLR_data_out_0(12 downto 11)& LO_pulse2;
diagnostics(247) <= BLR_pulse;
diagnostics(246) <= GATE_pulse;
diagnostics(245 downto 232) <= sigma;
diagnostics(231 downto 218) <= deltax;
diagnostics(217 downto 204) <= deltay;
diagnostics(198 downto 193) <= hchange & injection & cal_stop & cal_start & cycle_stop & cycle_start;

0x0008E0, 0x000960, 0x0009E0 DIAG_CTRL

This 32 bit register contains a count value to delay the diagnostics capture by a programmable number of clock cycles after the trigger condition.

0x0008E8, 0x000968, 0x0009E8 TEST

Test Register, used to write timing signals to the individual PU.

Bits 1:6 can be used to set the control signals:

cycle_start : bit 1

cycle_stop : bit 2
cal_start : bit 3
cal_stop : bit 4
injection : bit 5
hchange : bit 6

Bits 8:15 are used to select the test signals, rather than external input signals.

cycle_start : bit 9
cycle_stop : bit 10
cal_start : bit 11
cal_stop : bit 12
injection : bit 13
hchange : bit 14
fref : bit 15

The test FREF signals is generated using a programmable counter, and is held high for 8 clock cycles, out of every N. Where N cycles is set by bits 16:31 of the TIMING register.

SDRAM Bank Allocation

Bank	Addresses	Function
0	0x0000000-0xFDFFFFF	PU #0 Cycle Data Table
0	0xFE00000-0xFFFFFFF	PU #0 Bunch Mean Tables
1	0x0000000-0xFDFFFFF	PU #1 Cycle Data Table
1	0xFE00000-0xFFFFFFF	PU #1 Bunch Mean Tables
2	0x0000000-0xFDFFFFF	PU #2 Cycle Data Table
2	0xFE00000-0xFFFFFFF	PU #3 Bunch Mean Tables
3	0x0000000-0xFFFFFFF	Test Pattern Buffer

Cycle data are stored in 64 bit blocks containing 3x16 bit values in banks 0,1 and 2. The format is:

Bits	63:48	47:32	31:16	15:0
Function	CTIME	Delta Y	Delta X	Sigma

Bunch mean data are stored in 25 tables, the first table being the sum of all data and tables 2 to 25 containing data for each individual bunch in a rotation. Each table contains 4096 records of 128 bits. The format for these records are:

Bits	127:96	95:64	63:32	31:0
Function	Bunch Count	Delta Y Sum	Delta X Sum	Sigma Sum

The sums are accumulated on “Mean Filter Clock Enable 0” pulses, with a count of the number of pulses also recorded. For table 1, all pulses are active. For other tables, the pulse is ANDed with a one-hot shift register which is shifted every pulse and reset every rotation.

Test pattern data are stored in a more compact form, to improve memory access efficiency and to guarantee 1 sample per clock cycle operation. The format uses 12 bit integer for Sigma values, and 10 bits for Delta X and Delta Y. These values are sign extended to 14 bits.

The 64 bit format contains 2 samples:

Bits	63:54	53:44	43:32	31:22	21:12	11:0
Function	ΔY1	ΔX1	Σ1	ΔX0	ΔY0	Σ0

Block RAM Bank Allocation

Bank	Size (min 2kB)	PU	Bit Width	Function
0	32kB	0	64	Cycle Timing Table
1	2kB	0	64	Cycle Information Table
2	2kB	0	8	Timing Phase Table
3	2kB	0	32	Timing Switch Table
4	8kB	0	64	Diagnostics table
5	8kB	0	64	Bunch Mean Table #0
6	8kB	0	64	Bunch Mean Table #1

Block RAM bank allocation similar for PU #1 and PU#2, using bank ID's 8-15 and 16-23 respectively. All block RAMs are access as 64 bit wide memories on the PCI bus Smaller word widths occupying multiple addresses are mapped into single 64 bit wide memory locations. This makes it possible to create the Timing Phase table as an array of char and DMA it directly to the Block RAM.

The cycle timing table stores data in 64 bit words containing a 32 bit cycle number and a 32 bit address pointing into the cycle data table.

Bits	63:32	31:0
Function	Address	Cycle Time

The cycle information table, records all timing events, and the address pointer for the cycle timing table.

Bits	31:24	23:0
Function	Event	Address

4 cycles are stored in this table. Space is provided for 16 events per cycle. The 2 least significant bits of the cycle number are used as the most significant bits of the address, with the event counter incrementing from 0 to 15 on events, and reset to 0 on the CYCLE_START event.

The timing phase table consists of 16 tables with 512x8 bit entries.
Within each 8 bit value, the following bits are defined:
0 – LO1
1 – BLR
2 – GATE
3 – LO2
6 – Mean Filter Clock Enable 0
7 – Mean Filter Clock Enable 1

The timing switch table format has been described in the PLL-SPU section.
The Diagnostics Sigma, Delta X, Delta Y table is stored in the same format as the cycle data:

Bits	63:48	47:32	31:16	15:0
Function	Not used	Delta Y	Delta X	Sigma

The bunch mean tables are updated at the same time as the Cycle Timing Table, with the output of the bunch mean filters. The data is formatted as 3x16 bit values representing Sigma, Delta X and Delta Y.

Interrupts

The interrupt register has space for 32 interrupt lines. The following bits are used:

Bit	Function
0	PU #0 CYCLE_START
1	PU #0 CYCLE_STOP
2	PU #0 ERROR
3	PU #0 DIAGNOSTIC INFO CAPTURED
4	PU #0 SDRAM Write FIFO Half Full
5-7	PU #0 User Interrupts (Switch Table bits 5-7)
8	PU #1 CYCLE_START
9	PU #1 CYCLE_STOP
10	PU #1 ERROR
11	PU #1 DIAGNOSTIC INFO CAPTURED
12	PU #1 SDRAM Write FIFO Half Full
13-15	PU #1 User Interrupts (Switch Table bits 5-7)
16	PU #2 CYCLE_START
17	PU #2 CYCLE_STOP
18	PU #2 ERROR
19	PU #2 DIAGNOSTIC INFO CAPTURED
20	PU #2 SDRAM Write FIFO Half Full
21-23	PU #2 User Interrupts (Switch Table bits 5-7)

The SDRAM Write FIFO Half Full interrupt is to alert the host that it is locking the PUPE out of access to the SDRAM, and that there is a danger of data being lost. The host should release the SDRAM (stop the current memory read) and allow the PUPE to record data.

References

[1] IT-3384/AB: Technical Specification for a new trajectory measurement system for the CERN Proton Synchrotron
[2] Alpha Data's TMS Tender "pre-design-1.5".

[3] Alpha Data/Beam PUPE-API document 0.6 (Terry Barnaby)
[4] pll3.vhd (reference PLL VHDL design, Greg Kasprowicz CERN)

Synthesis

Synthesis is achieved using ISE tools (9.2), controlled using a make file (makefile_tms_pupe or makefile_tms_pupe_inc). Make files are provided in the synthesis directory and these run XST, NGDBUILD, MAP, PAR and BITGEN as appropriate. The ISE GUI is not used, and therefore no ISE project file is necessary. The XST project file and build scripts are: tms_pupe_xst.scr and tms_pupe_xst.prj. The project can be built from the command line using nmake –f makefile_tms_pupe

All synthesis scripts and files are located in the synthesis directory. The UCF file contains placement constraints for the SDRAM controller block memories. Without these constaints memory read errors can occur.

The other source code is divided into 3 main directories:
tms-processing contains the PUPE signal processing code core
pupe-wrapper contains the top level interfacing code
ddr2_memory_interface contains the core modules for the DDR2 memory interfaces