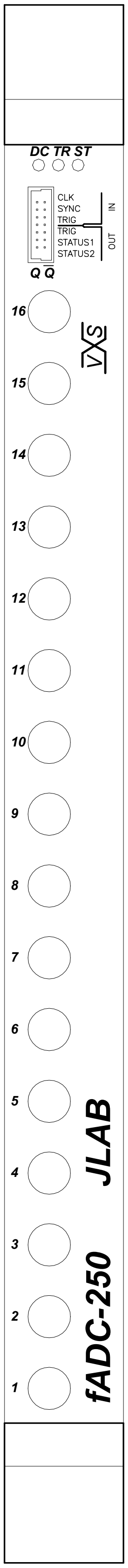
FADC250 User's Manual

Table of Contents

How to use this document

This document is a collage of different documentation produced by the Jefferson Lab data acquisition group regarding the FADC250 module. This collection pulls together these diverse sources to provide a single point of reference information regarding this module. For those who want to exploit the full capabilities of the module, it is recommended to start at the beginning and review the module specifications (section 2 below), and then proceed to the instructions for how to use the module (section 3 below). The Data Format section gives more details regarding the status values encoded in the data block header words and how the data from the module should be interpreted, which varies according to the different modes of operation. The internals of the board firmware gives a full picture of how the module behaves in each operating mode, and will be useful to those who are interested in extending its capabilities. The IEEE article at the end gives a general overview of the module, and can serve as a reference in case doubts arise whether a design feature has actually been implemented (see Results).

This collection has been assembled by Richard Jones, University of Connecticut, with full credit given to the actual authors of the contents, whose names are listed as authors of the IEEE article.



fADC250

VME64x Flash ADC Module Specifications

Signal Inputs	Number	16 S Version (50 Ohm, LEMO)*
	Range	-0.5V, -1V & -2V. User Selectable
	Offset	±10% FS per channel via DACs
Clock	Sampling	250 MSPS, Differential
	Jitter	1 pS (10-bit ADC), 350 fS (12-bit ADC)
	Source	Internal and External
Control Inputs/Outputs	Clock	IN – Diff., LVPECL (Front Panel & Backplane)
	Trigger	IN, OUT - Differential (Front Panel & Backplane)
	Status 1	OUT – Differential (Front Panel & Backplane)
	Status 2	OUT – Differential (Front Panel & Backplane)
	Sync	OUT – Differential (Front Panel & Backplane)
	Trigger SW	Software Strobe (Internal)
Conversion Characteristics	Resolution	10-bit (8 and 12-bit by chip replacement)
	INL	± 0.8 LSB
	DNL	± 0.5 LSB
	SNR	56.8 dB @ 100 MHz Input
	Data Latency	32 nS
Trigger Latency	8 µS	
Data Memory	8 µS	
Data Processing	Sparcification	
	Windowing	
	Charge, Pedestal, Peak	
	Time (Over Threshold, Relative to trigger)	
	Output (Backplane, VXS)	
Interface	VME64x – 2eVME Data Transfer Cycles (40, 80, 160 & 320 MB/sec) with VXS-P0	
Packaging	6U VME64x	
Power	+3.3V, +5V, +12V, -12V	

Using the FADC250 Module (V19 - 11/25/13)

1.1 Controlling the Module

Communication with the module is by standard VME bus protocols. All registers and memory locations are defined to be 4-byte entities. The VME slave module has three distinct address ranges.

A24 – The base address of this range is set by a 12-element DIP switch on the board. It occupies 4 Kbytes of VME address space, organized in 1 K 32-bit words. Relative to the base address, this space is divided as follows:

000-0FF – Register space to control and monitor the module (64 long words)

100-1FF – ADC processing registers (64 long words)

200-2FF – HITSUM processing registers (64 long words)

300-3FF – SCALER registers (64 long words)

400-4FF – SYSTEM TEST registers (64 long words)

500-FFF – Reserved (704 long words)

In addition to registers that are directly mapped to a VME address (Primary Address), the module supports Secondary Addressing in the A24 address space. These registers are accessed through an address mapping register (Secondary Address Register). Each secondary address is associated with a primary address. A Primary Address may have up to 64 K secondary addresses associated with it. A VME cycle loads the mapping register with data which is the internal (secondary) address of the target register. A VME cycle with the associated primary address accesses (read/write) the chosen internal register. Important registers are assigned primary addresses, allowing them to be directly accessible in a single VME cycle. Setup tables are assigned secondary addresses. This allows for a large internal address space, while maintaining a small VME footprint.

A32 - The base address of this range is programmed into register ADR32. It occupies 8 Mbytes of VME address space, organized in 2 M 32-bit words. A read of any address in this range will yield the next FADC data word from the module. Even though the module is logically a FIFO, the expanded address range allows the VME master to increment the address during block transfers. This address range can participate in single cycle, 32-bit block, and 64-bit block reads. The only valid write to this address range is the data value 0x80000000 which re-enables the module to generate interrupts (after one has occurred). The address range must be enabled by setting ADR32[0] = 1.

A32 - The lower and upper limits of this address range are programmed into register ADR_MB. This common address range for a set of FADC modules in the crate is used to implement the Multiblock protocol. By means of token passing FADC data may be read out from multiple FADC modules using a single logical block read. The board possessing the token will respond to a read cycle in this address range with the next FADC data word from that module. The token is passed along a private daisy chain line to the next module when it has transferred all data from a programmed number of events (register BLOCK SIZE). The address range must be enabled: set ADR_MB[0] = 1.

1.3 Module Registers

VERSION – board/firmware revision (0x0)

[7…0] – (R) – firmware revision

[15…8] – (R) – board revision

[31…16] – (R) – board type (“FADC”)

CSR – Control/Status (0x4)

0 – (R) – Event Accepted

1 – (R) – Block of Events Accepted

2 – (R) – Block of Events ready for readout

3 – (R) – BERR Status (1 = BERR asserted)

4 – (R) – Token Status (1 = module has token)

[5…10] – (reserved)

11 – (R) – Data FIFO Empty Flag Asserted

12 – (R) – Data FIFO Almost Empty Flag Asserted

13 – (R) – Data FIFO Half Full Flag Asserted

14 – (R) – Data FIFO Almost Full Flag Asserted

15 – (R) – Data FIFO Full Flag Asserted

[16…19] – (reserved)

20 – (W) – Pulse Soft **Trigger 2** (if CTRL[7] = 1 and CTRL[6..4] = 5)
(delayed **Trigger 1** follows; delay in TRIG21_DELAY register)
(R) – Trigger 2 -> Trigger 1 sequence active

21 – (W) – Pulse Clear Module – soft reset + clear data pipelines
(R) – Clear Module process active

22 – (W) – ENABLE SCALERS INTO DATA STREAM with FORCED BLOCK TRAILER INSERTION (write ‘1’ to bits 22, 23)

23 – (W) – FORCE BLOCK TRAILER INSERTION – will be successful only if there are NO triggers waiting to be processed

24 – (R) – Last FORCE BLOCK TRAILER INSERTION Successful

25 – (R) – Last FORCE BLOCK TRAILER INSERTION Failed

26 – (R) – Local Bus Time Out – target AK or DK timed out (5 us);

27 – (R/W) – Local Bus Error – target protocol violation;
(write ‘1’ clears latched bits 26, 27)

28 – (W) – Pulse Soft **Sync Reset** (if CTRL[11] = 1 and CTRL[10..8] = 6)

29 – (W) – Pulse Soft **Trigger 1** (if CTRL[7] = 1 and CTRL[6..4] = 6)

30 – (W) – Pulse Soft Reset – initialize counters, state machines, memory

31 – (W) – Pulse Hard Reset – initialize module to power-up state

CTRL1 – Control 1 (0x8)

[1…0] – (R/W) – Sampling Clock Source Select
0 = Internal Clock
1 = Front Panel connector
2 = P0 connector (VXS)
3 = P0 connector (VXS)

2 – (not used)

3 – (R/W) – Enable Internal Clock

[6…4] – (R/W) – Trigger Source Select

- 0 = Front Panel Connector (**Trigger 1**)
- 1 = Front Panel Connector (**Trigger 1**; synchronized)
- 2 = P0 Connector (VXS) (**Trigger1**, **Trigger 2**)
- 3 = P0 Connector (VXS) (**Trigger1**, **Trigger 2**; synchronized)
- 4 – (not used)
- 5 – Software Generated (**Trigger 2** + delayed **Trigger 1**)
- 6 = Software Generated (**Trigger 1**)
- 7 = Module Internal Logic

7 – (R/W) – Enable Soft Trigger

[10…8] – (R/W) – Sync Reset Source Select

- 0 = Front Panel Connector
- 1 = Front Panel Connector (synchronized)
- 2 = P0 Connector (VXS)
- 3 = P0 Connector (VXS) (synchronized)
- 4 – (not used)
- 5 – (not used)
- 6 = Software Generated
- 7 = no source

11 – (R/W) – Enable Soft Sync Reset

12 – (R/W) – Select Live Internal Trigger to Output

13 – (R/W) – Enable Front Panel Trigger Output

14 – (R/W) – Enable P0 (VXS) Trigger Output

[15…17] – (reserved)

18 – (R/W) – Enable Event Level Interrupt

19 – (reserved)

20 – (R/W) – Enable BERR response

21 – (R/W) – Enable Multiblock protocol

22 – (R/W) – FIRST board in Multiblock system

23 – (R/W) – LAST board in Multiblock system

24 – (reserved)

25 – (R/W) – Enable Debug Mode

[26…27] – (reserved)

28 – (R/W) – Multiblock Token passed on P0

29 – (R/W) – Multiblock Token passed on P2

30 – (reserved)

31 – (R/W) – System Test Mode (0 = normal, 1 = test mode enabled)

CTRL2 – Control 2 (0xC)

0 – (R/W) – GO (allow data transfer from external FIFOs to input FIFOs)

1 – (R/W) – Enable **Trigger** (**1** & **2**) to Module (source = CTRL1[6…4])

2 – (R/W) – Enable **Sync Reset** to Module (source = CTRL1[10…8])

3 – (R/W) – Enable Internal Trigger Logic

4 – (R/W) – Enable Streaming mode (NO event build)

[5…7] – (reserved)

8 – (R/W) – Enable Test Event Generation (for debug)

[9…15] – (reserved)

Bits 16 – 31 are functional only in Debug Mode (CTRL1[25] = 1)

16 – (reserved)

17 – (R/W) – Transfer data: build FIFO → output FIFO

[18…31] – (reserved)

BLOCK SIZE (0x10)

[15…0] – (R/W) – number of events in a BLOCK.
Stored Event Count ≥ BLOCK SIZE → CSR[3] = 1.

[31…16] – (reserved)

INTERRUPT (0x14)

[7…0] – (R/W) – Interrupt ID (vector)

[10…8] – (R/W) – Interrupt Level [2..0]. Valid values = 1,..,7.

11 - 15 – (reserved)

[20…16] – (R) – Geographic Address (slot number) in VME64x chassis.

21 – 22 – (reserved)

23 – (R) – Parity Error in Geographic Address.

24 – 31 – (reserved)

ADR32 – Address for data access (0x18)

0 – (R/W) – Enable 32-bit address decoding

1 – 6 – (reserved – read as 0)

[15…7] – (R/W) – Base Address for 32-bit addressing mode (8 Mbyte total)

ADR_MB – Multiblock Address for data access (0x1C)

0 – (R/W) – Enable Multiblock address decoding

1 – 6 – (reserved – read as 0)

[15…7] – (R/W) – Lower Limit address (ADR_MIN) for Multiblock access

16 – 22 – (reserved – read as 0)

[31…23] – (R/W) – Upper Limit address (ADR_MAX) for Multiblock access

The board that has the TOKEN will respond with data when the VME address satisfies the following condition:

$$\text{ADR\_MIN} \leq \text{Address} < \text{ADR\_MAX}.$$

SEC_ADR – Secondary Address (0x20)

[15…0] – (R/W) – Secondary Address for 24-bit addressing mode

16 – (R/W) – Enable auto-increment mode (secondary address increments by 1 after each access of the associated primary address)

DELAY – Trigger/Sync_Reset Delay (0x24)

[21…16] – (R/W) – Sync reset delay

[5…0] – (R/W) – Trigger delay

INTERNAL TRIGGER CONTROL (0x28)

[23…16] – (R/W) – trigger width (4 ns per count)

[7…0] – (R/W) – trigger hold off delay (4 ns per count)

RESET CONTROL (0x2C)

0 – (W) – Hard reset – Control FPGA

1 – (W) – Hard reset – ADC processing FPGA

[2…3] – (reserved)

4 – (W) – Soft reset – Control FPGA

5 – (W) – Soft reset – ADC processing FPGA

[6…7] – (reserved)

8 – (W) – Reset – ADC data FIFO

[9…10] – (reserved)

10 – (W) – Reset – HITSUM FIFO

11 – (W) – Reset – DAC (all channels)

12 – (W) – Reset – EXTERNAL RAM Read & Write Address Pointers

[13…15] – (reserved)

16 – (W) – Take Token – return token to 1st board of multiboard set

[17…31] – (reserved)

<u>TRIGGER COUNT</u> (0x30)

[31…0] – (R) – total trigger count

31 – (W) – reset count

<u>EVENT COUNT</u> (0x34)

[23…0] – (R) – number of events on board (non-zero → CSR[0] = 1).

[31…24] – (reserved)

<u>BLOCK COUNT</u> – (0x38)

[31…20] – reserved

[19…0] – (R) - number of event BLOCKS on board (non-zero → CSR[2] = 1).

<u>BLOCK FIFO COUNT</u> – (0x3C)

[31…6] – reserved

[5…0] – (R) - number of entries in BLOCK WORD COUNT FIFO

<u>BLOCK WORD COUNT FIFO</u> – (64 deep FIFO) (0x40)

[31…25] – reserved (read as ‘0’)

24 – (R) – count not valid (word count FIFO empty)

[23…20] – reserved (read as ‘0’)

[19…0] – (R) - number of words in next event BLOCK

<u>INTERNAL TRIGGER COUNT</u> (0x44)

[31…0] – (R) – internal live trigger count

31 – (W) – reset count

<u>EXTERNAL RAM WORD COUNT</u> (0x48)

[31…22] – reserved (read as ‘0’)

21 – (R) – RAM empty

20 – (R) – RAM full (1,048,576 eight byte words)

[19…0] – (R) – data word count (eight byte words)

<u>DATA FLOW STATUS</u> (0x4C) (for debug)

<u>DAC 1_2 – DAC channels 1,2</u> (0x50)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16] – (R/W) – DAC value channel 1

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 2

<u>DAC 3_4 – DAC channels 3,4</u> (0x54)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 3

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 4

<u>DAC 5_6 – DAC channels 5,6</u> (0x58)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 5

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 6

<u>DAC 7_8 – DAC channels 7,8</u> (0x5C)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 7

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 8

<u>DAC 9_10 – DAC channels 9,10</u> (0x60)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 9

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 10

DAC 11_12 – DAC channels 11,12 (0x64)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 11

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 12

DAC 13_14 – DAC channels 13,14 (0x68)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...18]– (R/W) – DAC value channel 13

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 14

DAC 15_16 – DAC channels 15,16 (0x6C)

31 – (reserved)

[30…28] – (reserved – read as 0)

[27...16] – (R/W) – DAC value channel 15

15 – (reserved)

[14…12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 16

<u>STATUS 1 – Input Buffer Status </u> (0x70)

31 – (R) – data buffer ready for input

30 – (R) – data buffer input paused

29 – (R) – reserved (read as ‘0’)

28 – (R) – data buffer empty

27 – (R) – data buffer full

[26…16] – (R) – data buffer word count

[15…0] – (reserved)

<u>STATUS 2 – Build Buffer Status </u> (0x74)

[31…29] – reserved (read as ‘0’)

28 – (R) – data buffer ‘A’ empty

27 – (R) – data buffer ‘A’ full

[26…16] – (R) – data buffer ‘A’ word count

[15…13] – reserved (read as ‘0’)

12 – (R) – data buffer ‘B’ empty

11 – (R) – data buffer ‘B’ full

[10…0] – (R) – data buffer ‘B’ word count

<u>STATUS 3 – Output Buffer Status </u> (0x78)

[31…30] – reserved (read as ‘0’)

29 – (R) – data buffer ‘A’ empty

28 – (R) – data buffer ‘A’ full

[27…16] – (R) – data buffer ‘A’ word count

[15…14] – reserved (read as ‘0’)

13 – (R) – data buffer ‘B’ empty

12 – (R) – data buffer ‘B’ full

[11…0] – (R) – data buffer ‘B’ word count

STATUS 4 – (spare) (0x7C)

[31…0] – reserved

AUXILIARY 1 – (spare) (0x80)

[31…0] – reserved

AUXILIARY 2 – (spare) (0x84)

[31…0] – reserved

TRIG21 DELAY (0x88)

[31…12] – reserved

[11…0] – (R/W) – Delay from soft TRIG2 to generated TRIG1 (4 ns/count)

RAM Address Register (0x8C) – The RAM is organized as two 36-bit words with a common address. Auxiliary VME access (R/W) to the RAM is provided through a pair of 32 bit data registers (RAM 1, RAM 2). Note that bits 35 – 32 of each RAM word are not accessible through VME. During data flow operations, these bits carry event marker tags (header, trailer).

31 – increment address after access (R/W) of RAM 1 Data Register

30 – increment address after access (R/W) of RAM 2 Data Register

[29…21] – reserved (read as 0)

[19…0] – RAM address

RAM 1 Data Register (0x90)

[31…0] – RAM data word bits 67 – 36 (32 bits)

RAM 2 Data Register (0x94)

[31…0] – RAM data word bits 31 – 0 (32 bits)

(PROM Registers 1 and 2 are used for FPGA configuration over VME.)

PROM Register 1 (0x98)

31 – READY – (R) – configuration state machine is available to accept command (i.e. no configuration process is currently executing).

[30…8] – reserved (read as 0)

[7…0] – configuration OPCODE

PROM Register 2 (0x9C)

[31…0] – PROM ID – (R) response to specific OPCODE write to PROM reg 1.

BERR Module Count (0xA0)

[31…0] – BERR count (driven by module to terminate data transmission)

BERR Total Count (0xA4)

[31…0] – BERR count (as detected on bus)

Auxiliary Scaler 1 (0xA8)

[31…0] – Total word count from ADC Processing FPGA

Auxiliary Scaler 2 (0xAC)

[31…0] – (reserved)

Auxiliary Scaler 3 (0xB0)

[31…0] – Event header word count from ADC Processing FPGA

TRIGGER 2 SCALER (0xB4)

[31…0] – (R) – **Trigger 2** count

31 – (W) – write ‘1’ to reset count

Auxiliary Scaler 5 (0xB8)

[31…0] – Event trailer word count from ADC Processing FPGA

SYNC RESET SCALER (0xBC)

[31…0] – (R) – **Sync Reset** count

31 – (W) – write ‘1’ to reset count

Module Busy Level (0xC0)

[31] – Force module busy

[30…20] – reserved

[19…0] – Busy level (eight byte words)
(External RAM word count > Busy level → module busy = 1)

Generate Event Header Word (0xC4) (for debug)

[31…0] – (W) – Event Header Word

Generate Event Data Word (0xC8) (for debug)

[31...0] – (W) – Event Data Word

Generate Event Trailer Word (0xCC) (for debug)

[31...0] – (W) – Event Trailer Word

MGT STATUS (0xD0)

0 – (R) – lane 1 up (GTX1)

1 – (R) – lane 2 up (GTX1)

2 – (R) – channel up (GTX1)

3 – (R) – hard error (GTX1)

4 – (R) – soft error (GTX1)

5 – (R) – lane 1 up (GTX2)

6 – (R) – lane 2 up (GTX2)

7 – (R) – channel up (GTX2)

8 – (R) – hard error (GTX2)

9 – (R) – soft error (GTX2)

10 – (R) – SUM DATA VALID

11 – (R) – MGT RESET ASSERTED

[31...12] – (R) - Reserved

MGT CONTROL (0xD4)

0 – **RELEASE MGT RESET** (0 = reset MGT, 1 = release reset)

1 – Data Type to CTP (0 = counting sequence, 1 = front-end data)

2 – Enable Data Alignment on Sync Reset occurrence

[31...3] – Reserved

<u>RESERVED (2 registers)</u> (0xD8 – 0xDC)

<u>SCALER CONTROL</u> (0xE0) – See SCALERS (0x300 – 0x340)

0 – (R/W) – Enable all scalers to count (1 = enable, 0 = disable)

1 – (W) – Latch all scalers. Write ‘1’ to simultaneously transfer all 17 scaler counts to registers for readout.

2 – (W) – Reset all scalers. Write ‘1’ to simultaneously reset all 17 scaler counts to zero.

[3 – 31] – (reserved)

<u>BOARD SERIAL NUMBER 0</u> (0xE4)

[31…24] – (R) – board serial number byte 0

[23…16] – (R) – board serial number byte 1

[15…8] – (R) – board serial number byte 2

[7…0] – (R) – board serial number byte 3

<u>BOARD SERIAL NUMBER 1</u> (0xE8)

[31…24] – (R) – board serial number byte 4

[23…16] – (R) – board serial number byte 5

[15…8] – (R) – board serial number byte 6

[7…0] – (R) – board serial number byte 7

<u>BOARD SERIAL NUMBER 2</u> (0xEC)

[31…24] – (R) – board serial number byte 8

[23…16] – (R) – board serial number byte 9

[15…8] – (R) – board serial number byte 10

[7…0] – (R) – board serial number byte 11

SCALER INSERTION INTERVAL (0xF0) - Data from the SCALERS defined below (0x300 – 0x340) may be inserted into the readout data stream at regular event count intervals. The interval is specified in multiples of the event BLOCK SIZE. When the interval is ZERO (the default condition), there is NO insertion of scaler data into the data stream. When programmed for a non-zero interval, the current scaler values are appended to the last event of the appropriate BLOCK of events. The current Trigger 1 count is also inserted as the 18th scaler. Note that the scalers are **NOT** reset after their values are captured.

Example: Interval = 10 means that every 10th block of events will have the integrated scaler data appended to it.

(See the document FADC V2 Data Format for information on identifying scaler data words in an event.)

The scalers may ALSO be inserted into the data stream when a FORCE BLOCK TRAILER is done by the user. A simultaneous write of ‘1’ to bit 22 and bit 23 of the CSR (0x4) accomplishes this. The scaler values are those at the time of the last trigger’s occurrence.

[15…0] - (R/W) – N (in BLOCKS of events); every Nth block of events has integrated scaler data appended to the last event in the block.

[31…16] – (reserved)

SPARE (3 registers) (0xF4 – 0xFC)

SCALER Registers (0x300 – 0x340) (R)

SCALER[0] – (0x300) - input channel 0 count

SCALER[1] – (0x304) - input channel 1 count

SCALER[2] – (0x308) - input channel 2 count

SCALER[3] – (0x30C) - input channel 3 count

SCALER[4] – (0x310) - input channel 4 count

SCALER[5] – (0x314) - input channel 5 count

SCALER[6] – (0x318) - input channel 6 count

SCALER[7] – (0x31C) - input channel 7 count

SCALER[8] – (0x320) - input channel 8 count

SCALER[9] – (0x324) - input channel 9 count

SCALER[10] – (0x328) - input channel 10 count

SCALER[11] – (0x32C) - input channel 11 count

SCALER[12] – (0x330) - input channel 12 count

SCALER[13] – (0x334) - input channel 13 count

SCALER[14] – (0x338) - input channel 14 count

SCALER[15] – (0x33C) - input channel 15 count

TIME COUNT – (0x340) - timer (each count represents 2048 ns)

<u>System Test Registers</u> (0x400 – 0x410)

<u>TEST BIT REGISTER</u> (0x400)

0 – (R/W) – trigger_out_p0 (1 = asserted, 0 = not asserted)

1 – (R/W) – busy_out_p0 (1 = asserted, 0 = not asserted)

2 – (R/W) – sdlink_out_p0 (1 = asserted, 0 = not asserted)

3 – (R/W) – token_out_p0 (1 = asserted, 0 = not asserted)

[4 – 7] – (R/W) – spare out test bits

8 – (R) – status_b_in_p0 state (1 = asserted, 0 = not asserted)

9 – (R) – token_in_p0 state (1 = asserted, 0 = not asserted)

[10 - 14] – (R) – reserved (read as ‘0’)

15 – (R) – clock_250 counter status (1 = counting, 0 = not counting)

[16 - 31] – (R) – reserved (read as ‘0’)

<u>CLOCK_250 COUNT REGISTER</u> (0x404)

0 – (W) – Write ‘0’ resets the counter. Write ‘1’ initiates 20us counting interval.

[31 - 0] – (R) – CLK_250 counter value. (Should be 5000 after count interval.)

<u>SYNC_IN_P0 COUNT REGISTER</u> (0x408)

0 – (W) – Write ‘0’ resets the counter.

[31 - 0] – (R) – SYNC_IN_P0 counter value.

<u>TRIG1_IN_P0 COUNT REGISTER</u> (0x40C)

0 – (W) – Write ‘0’ resets the counter.

[31 - 0] – (R) – TRIG1_IN_P0 counter value.

TRIG2_IN_P0 COUNT REGISTER (0x410)

0 – (W) – Write ‘0’ resets the counter.

[31 - 0] – (R) – TRIG2_IN_P0 counter value.

ADC PROCESSING FPGA ADDRESS MAP:

Control Bus Memory Map for FADC FPGA

Name **[VME ADDRESS]**	Width (Bits)	Quantity	Access	Primary Address (Secondary Address)	Function
STATUS0 **[0x100]**	16	1	R	0x0000 (---)	Bits 14 to 0: Code Version Bit 15: 1= Command can be sent to AD9230
STATUS1 **[0x104]**	16	1	R	0x0001 (---)	TRIGGER NUMBER BIT 15 to 0
STATUS2 **[0x108]**	16	1	R	0x0002 (---)	Tbd. Read 0
CONFIG 1 **[0x10C]**	16	1	R/W	0x0003 (---)	Bit 0-2 (process mode): 000 → Select option1 001 → Select option2 010 → Select option3 011 → Select option4 111 → Run option1 then option4 for each trigger Bit 3: 1:Run Bit 6-5 : Number of Pulses in Mode 1 and 2 Bit 7: Test Mode (play Back).
CONFIG 2 **[0x110]**			R/W	0x0004 (---)	When 1 ADC values = 0 Bit 0 → ADC 0 Bit 1 → ADC 1 Bit 2 → ADC 2 Bit 3 → ADC 3 Bit 4 → ADC 4 Bit 5 → ADC 5 Bit 6 → ADC 6 Bit 7 → ADC 7 Bit 8 → ADC 8 Bit 9 → ADC 9 Bit 10→ ADC 10

					Bit 11→ ADC 11 Bit 12→ ADC 12 Bit 13→ ADC 13 Bit 14→ ADC 14 Bit 15→ ADC 15
CONFIG 4 **[0x114]**	16	1		0x0005 (---)	7 => rising edge write to AD9230 ADC 6 => 1 write to all ADC. Bits 3..0 are don't care 5 => 0 write to AD9230 1 read from AD9230 3..0 => Select ADC to write to
CONFIG 5 **[0x118]**	16	1		0x0006 (---)	15..8 => Registers inside AD9230 7..0 => Data to write to register.
PTW **[0x11C]**	9	1	R/W	0x0007 (---)	Number of ADC sample to include in trigger window. PTW = Trigger Window (ns) * 250 MHz. **Minimum is 6**. **Always report Even Number. For odd PTW number, discard the last sample reported.**
PL **[0x120]**	11	1		0x0008 (---)	Number of sample back from trigger point. PL = Trigger Window(ns) * 250MHz
NSB **[0x124]**	12	1		0x0009 (---)	Number of sample before trigger point to include in data processing. This include the trigger Point. **Minimum is 2 in all mode.**
NSA **[0x128]**	13	1		0x000A (---)	Number of sample after trigger point to include in data processing. **Minimum is (6 in mode 2)and (3 in mode 0 and 1). Number of**

					sample report is 1 more for odd and 2 more for even NSA number.
TET **[0x12C – 0x148]** (2 channels per word) (see **Note 1** below)	12	16		0x000B - 0x001A (---)	Trigger Energy Threshold.
PTW DAT BUF LAST ADR **[0x14C]**	12	1		0x001B (---)	Last Address of the Secondary Buffer. See calculation below
PTW MAX BUF **[0x150]**	8	1		0x001C (---)	The maximum number of unprocessed PTW blocks that can be stored in Secondary Buffer. See Calculation below.
Test Wave Form **[0x154]**	16	1		0x001D (---)	Write to PPG. Read should immediately follow write.
ADC0 Pedestal Subtract **[0x158]**	16	1	R/W	0x001E	Subtract from ADC0 Count before Summing
ADC1 Pedestal Subtract **[0x15C]**	16	1	R/W	0x001F	Subtract from ADC1 Count before Summing
ADC2 Pedestal Subtract **[0x160]**	16	1	R/W	0x0020	Subtract from ADC2 Count before Summing
ADC3 Pedestal Subtract **[0x164]**	16	1	R/W	0x0021	Subtract from ADC3 Count before Summing
ADC4 Pedestal Subtract **[0x168]**	16	1	R/W	0x0022	Subtract from ADC4 Count before Summing
ADC5 Pedestal Subtract **[0x16C]**	16	1	R/W	0x0023	Subtract from ADC5 Count before Summing
ADC6 Pedestal Subtract **[0x170]**	16	1	R/W	0x0024	Subtract from ADC6 Count before Summing
ADC7 Pedestal Subtract **[0x174]**	16	1	R/W	0x0025	Subtract from ADC7 Count before Summing

ADC8 Pedestal Subtract **[0x178]**	16	1	R/W	0x0026	Subtract from ADC8 Count before Summing
ADC9 Pedestal Subtract **[0x17C]**	16	1	R/W	0x0027	Subtract from ADC9 Count before Summing
ADC10 Pedestal Subtract **[0x180]**	16	1	R/W	0x0028	Subtract from ADC10 Count before Summing
ADC11 Pedestal Subtract **[0x184]**	16	1	R/W	0x0029	Subtract from ADC11 Count before Summing
ADC12 Pedestal Subtract **[0x188]**	16	1	R/W	0x002A	Subtract from ADC12 Count before Summing
ADC13 Pedestal Subtract **[0x18C]**	16	1	R/W	0x002B	Subtract from ADC13 Count before Summing
ADC14 Pedestal Subtract **[0x190]**	16	1	R/W	0x002C	Subtract from ADC14 Count before Summing
ADC15 Pedestal Subtract **[0x194]**	16	1	R/W	0x002D	Subtract from ADC15 Count before Summing

PTW MAX BUF = INT(2016 / (PTW + 8) * 250000000)

Where:

2016 → Number of address of Secondary Buffer

PTW → Trigger Window width in nano-second

PTW DAT BUF LAST ADR = PTW MAX BUF * (PTW + 6)- 1;

Where:

6 → 4 address for Time Stamp and 2 address for Trigger Number

NumberOfBytePerTrigger → PTW * 250 MHz.

NOTE 1: Trigger Energy Threshold (TET)

0x12C – Channel 1 & Channel 2

[31…28] – not used
[27…16] – channel 1 threshold
[15…12] – not used
[27…16] – channel 2 threshold

0x130 – Channel 3 & Channel 4

[31…28] – not used
[27…16] – channel 3 threshold
[15…12] – not used
[27…16] – channel 4 threshold

. . . .

0x148 – Channel 15 & Channel 16

[31…28] – not used
[27…16] – channel 15 threshold
[15…12] – not used
[27…16] – channel 16 threshold

FADC250 Data Format

(Ed Jastrzembski – updated **11/4/13**)

We have identified the multiple types of data produced by the JLab 250 MHz Flash ADC module and defined a 32-bit word data format for readout over VME. This format is consistent with the 12 GeV standard defined for JLab custom data acquisition modules.

Data Word Categories

Data words from the module are divided into two categories: Data Type Defining (bit 31 = 1) and Data Type Continuation (bit 31 = 0). Data Type Defining words contain a 4-bit data type tag (bits 30 - 27) along with a type dependent data payload (bits 26 - 0). Data Type Continuation words provide additional data payload (bits 30 – 0) for the *last defined data type*. Continuation words permit data payloads to span multiple words and allow for efficient packing of raw ADC samples. Any number of Data Type Continuation words may follow a Data Type Defining word. The scaler data type is an exception. It specifies the number of 32-bit data words that follow.

Data Type List

0 – block header
1 – block trailer
2 – event header
3 – trigger time
4 – window raw data
5 – (reserved)
6 – pulse raw data
7 – pulse integral
8 – pulse time
9 – 11 – (reserved)
12 – scaler data
13 – (reserved)
14 – data not valid (empty module)
15 – filler (non-data) word

Pulse Data

To reduce the amount of data generated at high trigger rates, the module has the capability of identifying pulses within the trigger window and reporting computed quantities (pulse integral, pulse time) instead of raw data samples or simple sums of raw samples. The algorithm for identifying a pulse may be complex and application dependent. Individual pulses may have characteristics that compromise the accuracy of the computed quantities. For example, a pulse that extends beyond the measurement window may significantly affect the computed integral value, while the computed time value (which relies on the leading edge) may be accurate. On the other hand, a pulse whose leading edge starts outside the measurement window may have an inaccurate time, but an acceptable integral value. To identify these or other conditions, we include a two-bit *quality factor* when reporting computed quantities for pulses. The algorithm designer may use the four classes to identify specific conditions that may be associated with the measurement, or give an overall rating of confidence in the measurement based of all characteristics of the pulse. Of course, the algorithm can ultimately choose to not report a particular measurement for a pulse.

The data format allows for up to four identified pulses in the trigger window for each channel. The pulse number links together pulse data types 6, 7, and 8.

Data Types

Block Header (0) – indicates the beginning of a block of events. (High-speed readout of a board or set of boards is done in blocks of events.)

(31) = 1
(30 – 27) = 0
(26 – 22) = slot number (set by VME64x backplane)
(21 – 18) = module ID ('1' for FADC250)
(17 – 8) = event block number (used to align blocks when building events)
(7 – 0) = number of events in block

Block Trailer (1) – indicates the end of a block of events. The data words in a block are bracketed by the block header and trailer.

(31) = 1
(30 – 27) = 1
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = total number of words in block of events

Event Header (2) – indicates the start an event. The included trigger number is useful to ensure proper alignment of event fragments when building events. The 22-bit trigger number is not a limitation, as it will be used to distinguish events within event blocks, or among events that are concurrently being built or transported.

(31) = 1
(30 – 27) = 2
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = event number (trigger number)

Trigger Time (3) – time of trigger occurrence relative to the most recent global reset. Time in the ADC data processing chip is measured by a 48-bit counter that is clocked by the 250 MHz system clock. The global reset signal is distributed to every ADC processing chip. The assertion of the global reset clears the counters and inhibits counting. The de-assertion of global reset enables counting and thus sets t = 0 for the component. The trigger time is necessary to ensure system synchronization and is useful in aligning event fragments when building events. With careful clock, trigger, and global reset distribution it may be possible to achieve identical trigger times from all components of the system. However, even if t = 0 is not the same for all components, *changes* in trigger times can be monitored to ensure system synchronization is maintained. The six bytes of the trigger time

$$\text{Time} = T_A\ T_B\ T_C\ T_D\ T_E\ T_F$$

are reported in two words (Type Defining + Type Continuation):

Word 1:

(31) = 1
(30 – 27) = 3
(26 – 24) = reserved (read as 0)
(23 – 16) = T_D
(15 – 8) = T_E
(7 – 0) = T_F

Word 2:

(31) = 0
(30 – 24) = reserved (read as 0)
(23 – 16) = T_A
(15 – 8) = T_B
(7 – 0) = T_C

Window Raw Data (4) – raw ADC data samples for the trigger window. The first word identifies the channel number and window width. Multiple continuation words contain two samples each. The earlier sample is stored in the most significant half of the continuation word. Strict time ordering of the samples is maintained in the order of the continuation words. A *sample not valid* flag may be set for any sample; e.g. the last reported sample is not valid when the window consists of an odd number of samples.

Word 1:

(31) = 1
(30 – 27) = 4
(26 – 23) = channel number (0 – 15)
(22 – 12) = reserved (read as 0)
(11 – 0) = window width (in number of samples)

Words 2 - N:

(31) = 0
(30) = reserved (read as 0)
(29) = sample x not valid
(28 – 16) = ADC sample x (includes overflow bit)
(15 – 14) = reserved (read as 0)
(13) = sample x + 1 not valid
(12 – 0) = ADC sample x + 1 (includes overflow bit)

Pulse Raw Data (6) – raw ADC data samples for an identified pulse. Raw data from an interval of the trigger window that includes the pulse is provided. The first word indicates the channel number, pulse number, and first sample number. The first sample number is relative to the beginning of the trigger window (sample 0). Up to 4 pulses may be identified for each channel. Multiple continuation words contain two raw samples each. The earlier sample is stored in the most significant half of the continuation word. Strict time ordering of the samples is maintained in the order of the continuation words. A *sample not valid* flag may be set for any sample; for example, the last reported sample is tagged as not valid when the pulse interval consists of an odd number of samples.

Word 1:

(31) = 1
(30 – 27) = 6
(26 – 23) = channel number (0 – 15)
(22 – 21) = pulse number (0 – 3)
(20 – 10) = reserved (read as 0)
(9 – 0) = first sample number for pulse

Words 2 - N:

(31) = 0
(30) = reserved (read as 0)
(29) = sample x not valid
(28 – 16) = ADC sample x (includes overflow bit)
(15 - 14) = reserved (read as 0)
(13) = sample x + 1 not valid
(12 – 0) = ADC sample x + 1 (includes overflow bit)

Pulse Integral (7) – integral of an identified pulse within the trigger window. The pulse integral may be a simple sum of raw data samples over the pulse duration, or the result of a complex fit to pulse shape. Pedestal subtraction may be included.

(31) = 1
(30 – 27) = 7
(26 – 23) = channel number (0 – 15)
(22 – 21) = pulse number (0 – 3)
(20 – 19) = measurement quality factor (0 – 3)
(18 – 0) = pulse integral

Pulse Time (8) – time associated with an identified pulse within the trigger window.

(31) = 1
(30 – 27) = 8
(26 – 23) = channel number (0 – 15)
(22 – 21) = pulse number (0 – 3)
(20 – 19) = measurement quality factor (0 – 3)
(18 - 16) = reserved (read as 0)
(15 – 0) = pulse time

Scaler Header (12) – indicates the beginning of a block of scaler data words. The number of scaler data words that will immediately follow it is provided in the header. The scaler data words are 32 bits wide and so have no bits available to identify them. Currently there are 18 scaler words reported: 16 from individual channels, a timer, and a trigger count. The scalers and time represent values recorded at the indicated trigger count. Scaler data must be enabled into the data stream by the user.

(31) = 1
(30 – 27) = 12
(26 – 6) = reserved (read as 0)
(5 – 0) = number of scaler data words to follow (18 = current)

Data Not Valid (14) – module has no valid data available for read out.

(31) = 1
(30 – 27) = 14
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = undefined

Filler Word (15) – non-data word appended to the block of events. Forces the total number of 32-bit words read out of a module to be a multiple of 2 or 4 when 64-bit VME transfers are used. **This word should be ignored**.

(31) = 1
(30 – 27) = 15
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = undefined

FIRMWARE for FADC250 Ver2 ADC FPGA

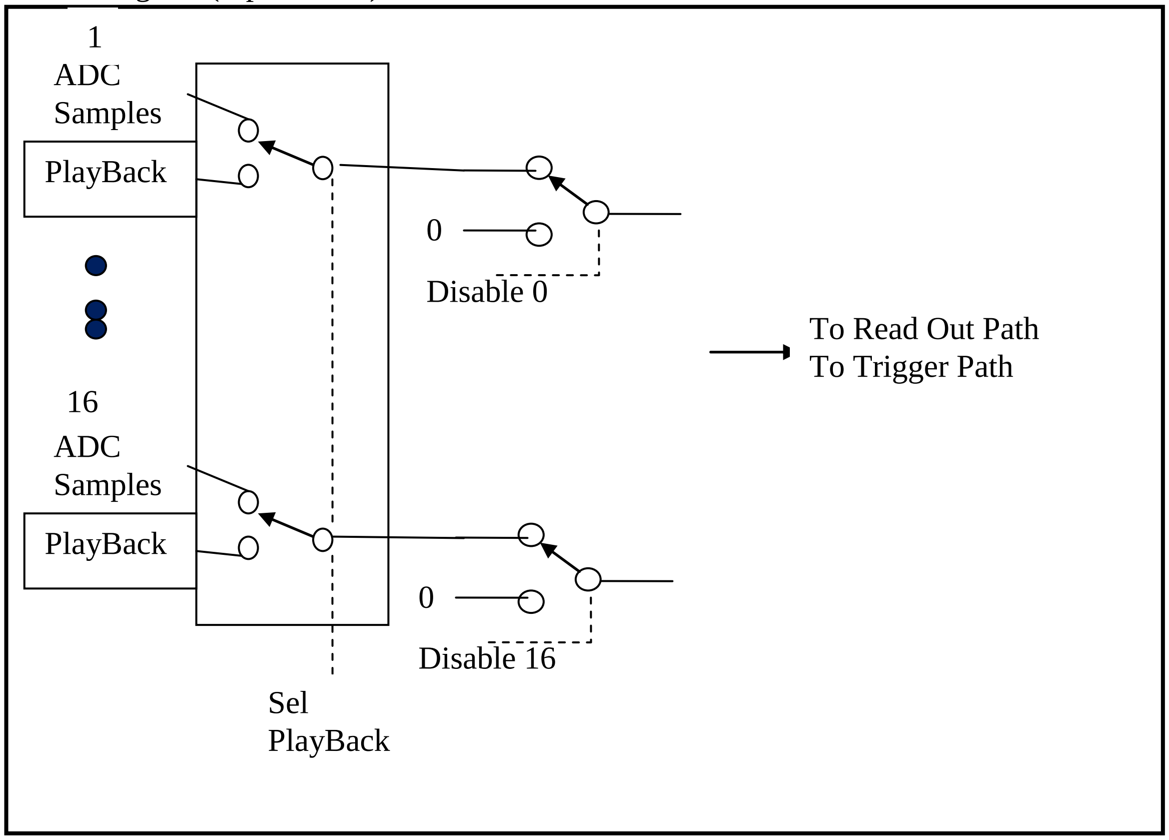
Table of Content:

ADC FPGA Functional Description

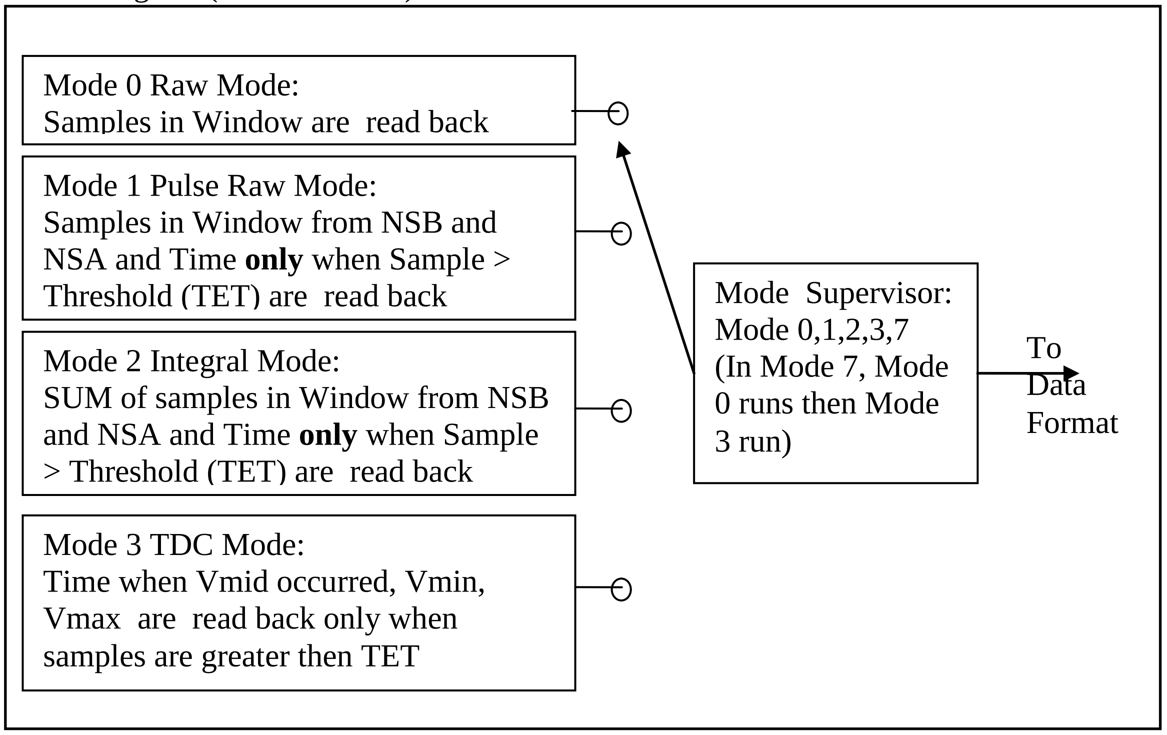
Overview:

The ADC FPGA receives 12-bit data words streaming at 250 MHz from 16 ADC. It performs **Channel Data Processing** for each ADC, computes **Energy Sum** of all ADC, and generates **Acceptance Pulse** for each ADC. The data selected in Channel Data Processing and results of Energy Sum are passed to CTRL FPGA to be sent to VME host and CTP respectively. The code is modular such that processing algorithms can easily be added or deleted.

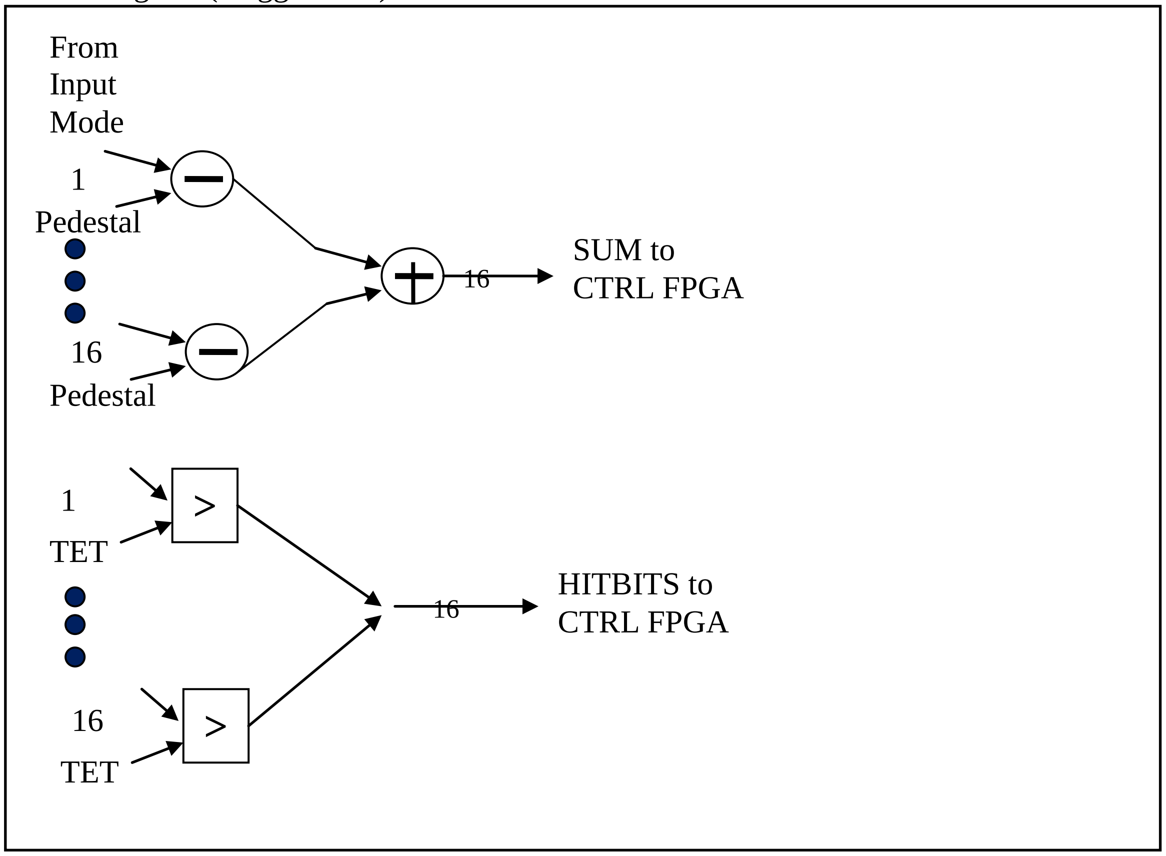
Block Diagram (Input Mode):



Block Diagram (Read Out Path):


Mode 0 Raw Mode:
Samples in Window are read back
Mode 1 Pulse Raw Mode:
Samples in Window from NSB and NSA and Time **only** when Sample > Threshold (TET) are read back
Mode 2 Integral Mode:
SUM of samples in Window from NSB and NSA and Time **only** when Sample > Threshold (TET) are read back
Mode 3 TDC Mode:
Time when Vmid occurred, Vmin, Vmax are read back only when samples are greater then TET
Mode Supervisor:
Mode 0,1,2,3,7
(In Mode 7, Mode 0 runs then Mode 3 run)
To Data Format

Block Diagram (Trigger Path):



Reset:

Hard Reset: Reset Everything except Time Stamp and ADC IC
Soft Reset: Reset Everything except Time Stamp, Registers, and ADC IC
Sync : Only reset Time Stamp.
ADC IC is reset through register bit.

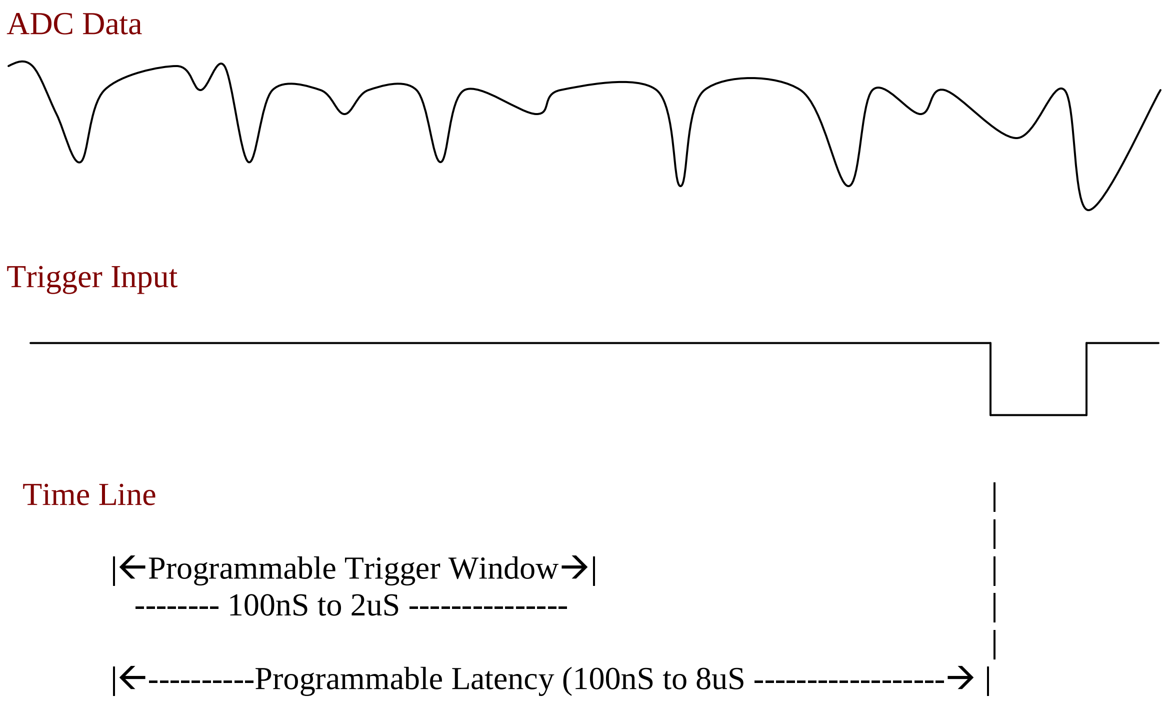
Pedestal Subtraction:

Samples received from the ADC are immediately subtracted from a programmable pedestal value in the Trigger Data Path. The result is not allowed to go below zero. Each of the ADC has a separate pedestal value.

Programmable Pulse Generator (PPG):

Input to Channel Data Processing can either come from ADC after pedestal subtraction or the Programmable Pulse Generator (PPG). Users can load simulated PMT data into the PPG via VME host. When a trigger occurs in test mode, the stored data is read and apply to Channel Data Processing. There are 16 PPG, one for each ADC channel and each PPG can hold 32 samples.

1. **Channel Data Processing:**


ADC Data
Trigger Input
Time Line
|←Programmable Trigger Window→|
-------- 100nS to 2uS ---------------
|←----------Programmable Latency (100nS to 8uS ------------------→ |

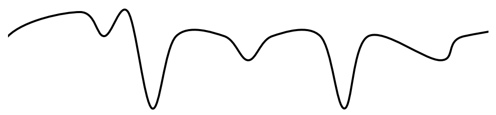
Data from ADC are stored continuously in circular buffer until Trigger input becomes active (low). The data that was stored from the time that the Trigger occurs back to the time specified by Programmable Latency within the Programmable Trigger Window are processed.

There are three main options to which these data are processed. The options are selectable by the user via VME register setting and two Trigger Inputs.

While data are being processed, ADC FPGA will continue storing incoming ADC data with no loss of data. Programmable Trigger Window (PTW) and Programmable Latency(PL) are common to all 8 ADC channels.

<u>Mode 0 (Raw Mode):</u>
Data within the Programmable Trigger Window [PTW] is passed with no further processing to the VME Host.

Option 1 Raw Mode Data to VME Host Illustration:



Trigger Input

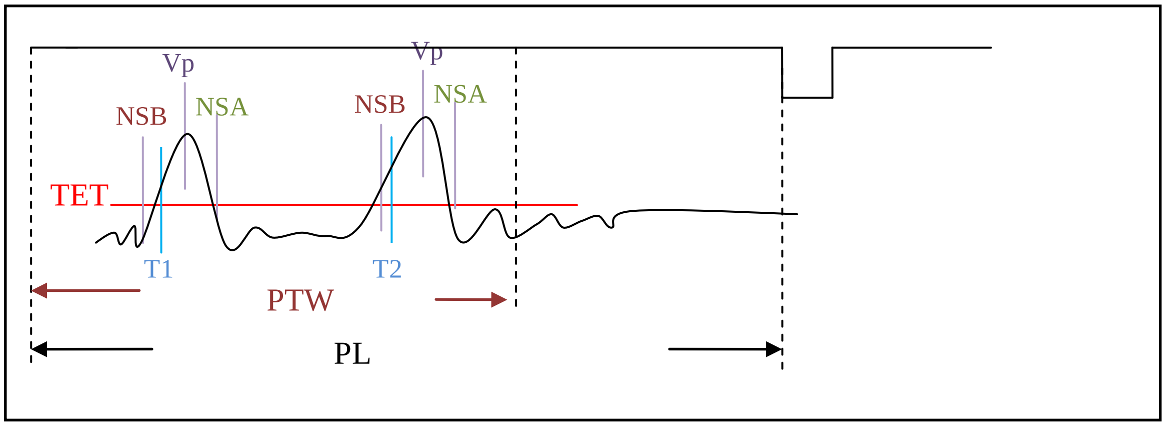
Time Line

```
                                                                        |
                                                                        |
      |←Programmable Trigger Window→|                                   |
                                                                        |
      |←----------Programmable Latency   -----------------------------------→ |
```

Mode 1 (Pulse Mode) :

When an ADC sample has a value that is greater than Programmable Trigger Energy Threshold (TET), the number of samples before (NSB) the Maximum value (Vp) and the number of samples after (NSA) Vp are sent to VME Host. NSB and NSA are programmable. T1 and T2 are described in TDC Algorithm.
TET is 12 bits and unique to each ADC channel.
NSB has a maximum value of 1024
NSA has a maximum value of 1024

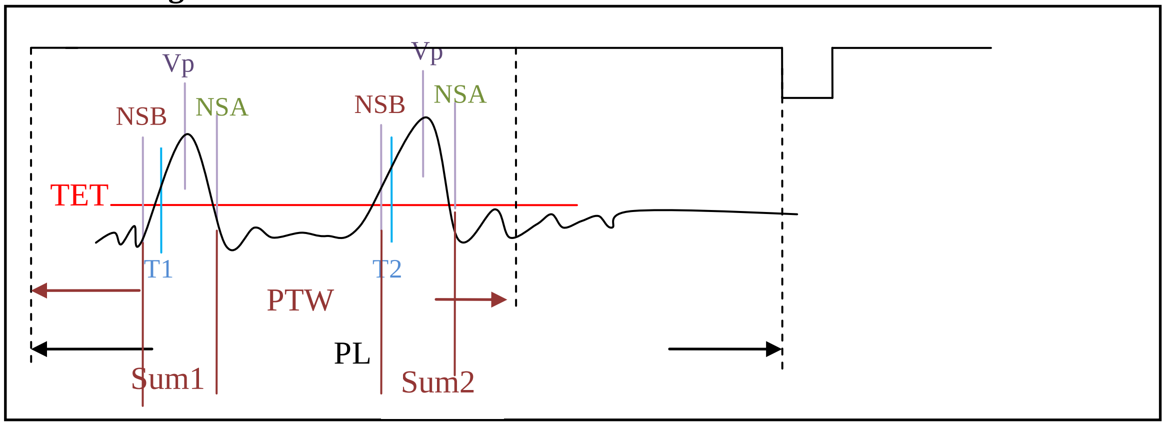
Mode 1 Pulse Mode Data to VME Host Illustration:



Mode 2 Integral Mode:

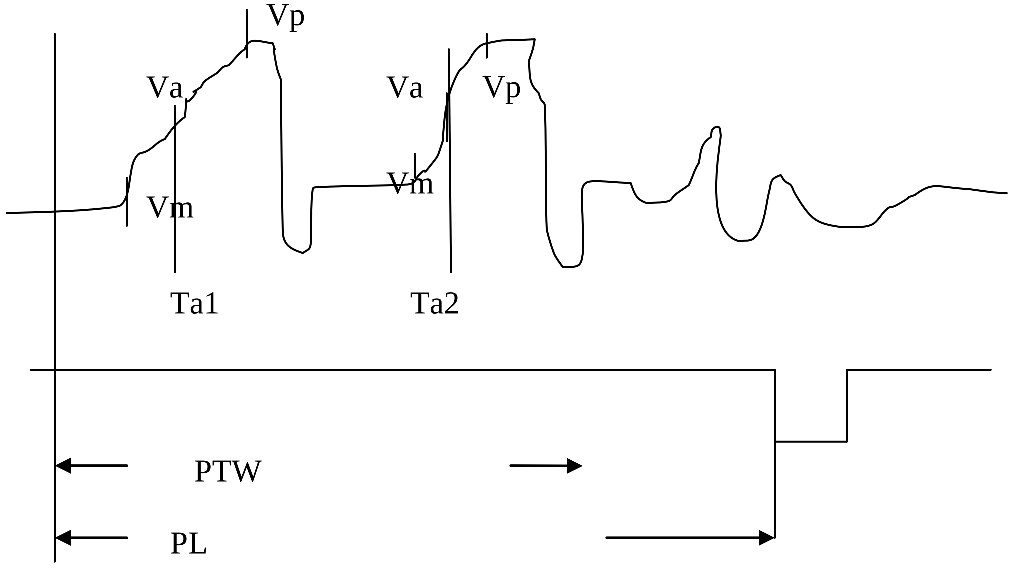
Data within NSB and NSA of Option 2 Raw Mode are summed around T1 and T2. PNS defines the number of samples before and after T1 and T2 include in Sum 1 and Sum 2 respectively**.** Only Sum 1, T1, Sum 2, and T2are passed to VME FPGA. T1 and T2 are described in TDC Algorithm.

Mode 2 Integral ModevData to VME Host Illustration:



Mode 3 TDC Algorithm:

The TDC algorithm calculates time of the mid value (Va) of a pulse relative to the beginning of the look back window. Va is the value between the smallest and the peak value (Vp) of the pulse. The smallest value (Vm) is the beginning of the pulse. The time consists of coarse time and fine time. The coarse time is the number of clock the sample value before the mid value and the fine time is the interpolating value of mid value away from next sample. The coarse value is 10 bits and the fine value is 6 bit. The resolution of LSB is 1/(CLK * 64). For a 250MHz the resolution is 62.5 pS. For example for a 20MHz clock, a pulse time (Ta) value of 110 means the mid-point of the pulse occurred at 6.875nS (62.5pS * 110) from the beginning of look back window.

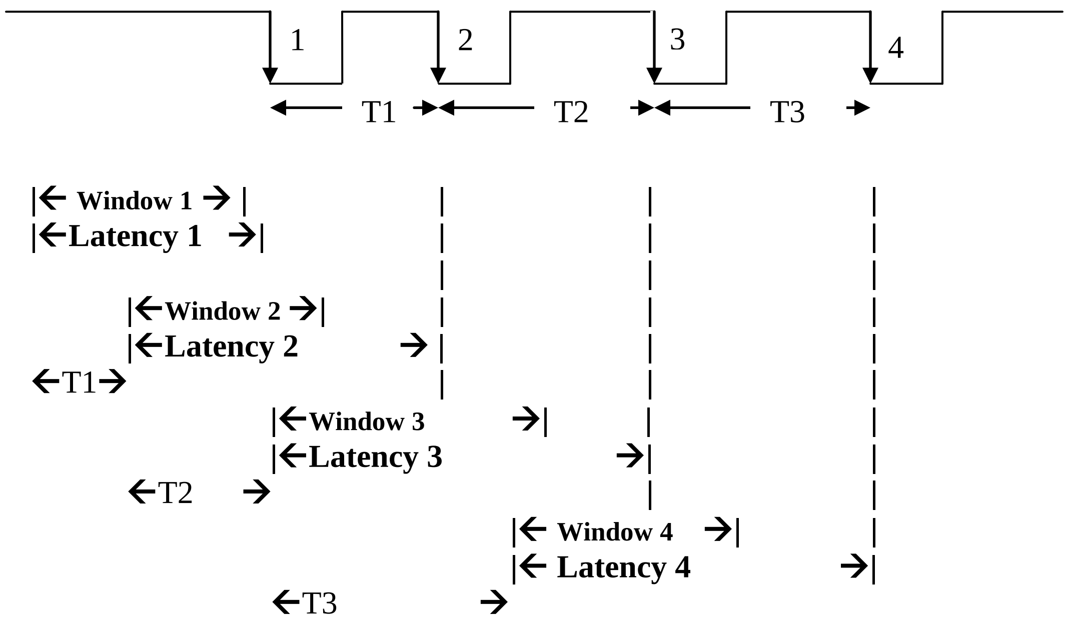


Requirements for TDC Algorithm:

i) There must be at least 5 samples (background) before pulse. Four of these samples are used to determine the pedestal (Vnoise) floor. The minimum value of the pulse is the first value that is greater than Vnoise.

<u>Trigger Input Buffer:</u>
In the event that the Trigger Input rate is faster than the data processing time, the processing algorithm has to be able to process 100 consecutives triggers with no loss in time lines.   If a trigger cannot be processed due to an overflow condition, the VME FPGA will be notified: “no data for trigger. If T1, T2, or T3 is less than 50 Ns, the trigger will not be recorded.

Successive Trigger Input Illustration:



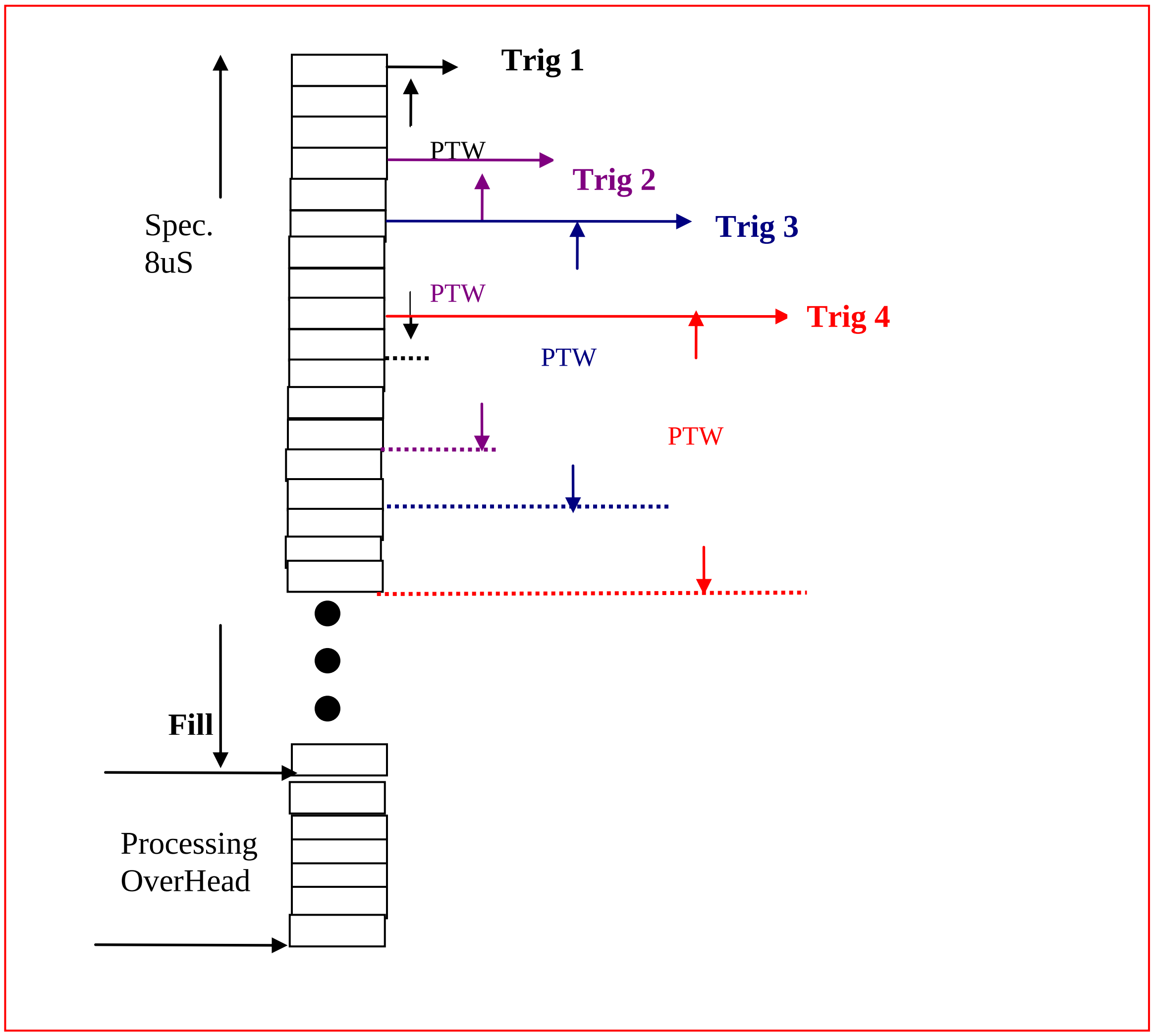
<u>Trigger Options:</u>

The type of processing mode is determined by two trigger inputs and the two bits of a VME register setting. The Trigger Processing Mode table below shows the possible processing mode.

Trigger Processing Mode:

VME Bits	Trigger Inputs Trig2 | Trig1	Modes
00	00	Idle
	01	Raw, Integral
	10	Integral
	11	Scaler Read Back
01	00	Idle
	01	Pulse, Integral
	10	Integral
	11	Scaler Read Back
10	Xx	Idle
11	Xx	Idle

Memory Model for Successive Trigger Input Illustration:



2. Energy Sum:

Data from ADC are added and the 16 bits-sum is sent to CTRL FPGA. Three stages pipeline adders are implemented to allow 250 MHz clocking. Sum valid signal accompanied the Sum.

The 16 bit energy sum is transferred from the CTRL FPGA on two full duplex gigabit transceiver ports. The transceivers are configured to operate at 2.5Gb/s per lane and will communicate directly to the VXS switch “A” slot.

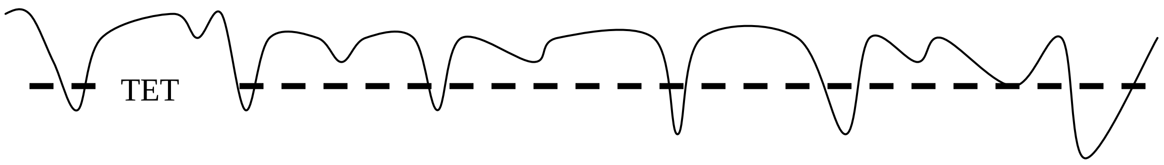
There is probably more information that can be written here to define the configuration of the transceivers and explain the data format of the energy sum.

3. <u>HITBITS:</u>

When counts from an ADC channel are greater than threshold, the corresponding Hit Bit for that channel is high. The HitBits are processed by TRIG_PROC_TOP to form coincident trigger.

HitBits Illustration:

ADC data

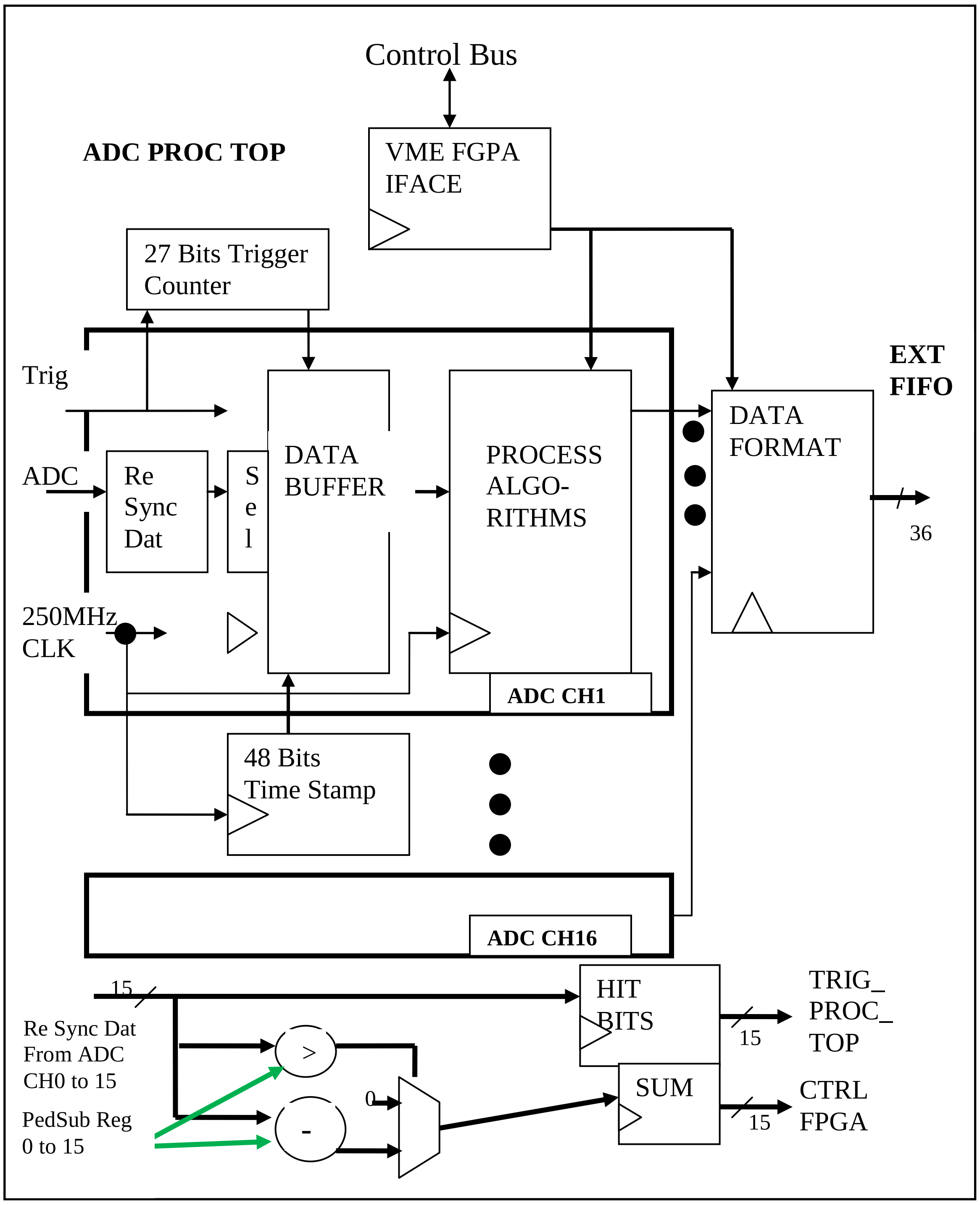


Hit Bit

```
__    _________   ____________  ______________ _________   _____________   __
  |  |         | |            ||              ||         |  |             |   |
  |  |         | |            ||              ||         |  |             |   |
  -----         ---            --              --         ----             -------
```

Conceptual Architecture Diagram

Overview:


Control Bus
ADC PROC TOP
VME FGPA IFACE
27 Bits Trigger Counter
EXT FIFO
Trig
DATA FORMAT
ADC
Re Sync Dat
S e l
DATA BUFFER
PROCESS ALGO-RITHMS
36
250MHz CLK
ADC CH1
48 Bits Time Stamp
ADC CH16
15
HIT BITS
TRIG_ PROC_ TOP
Re Sync Dat From ADC CH0 to 15
>
0
-
SUM
15
CTRL FPGA
15
PedSub Reg 0 to 15

Data from each ADC is resynchronized with FPGA main CLK. The outputs of the Resync are inputs of Data Buffer, Pedestal Substraction, and Hit Bits circuits. Each ADC Channel has Resync, Data Buffer and Processing Circuits. The Data Buffer stores Resync Data, Trigger Number, and Time Stamp. Processing Circuit processes data from Data Buffer. Format read results from each ADC channels (0-15) Processing Circuit and mux it to external FIFO. For each of the ADC, Pedestal Subtraction Circuit subtracts a programmable constant from ADC sample if the sample is greater than the constant. If the sample is smaller than the constant, zero is output. The output of the Pedestal Subtraction Circuit is fed to the Sum circuit. Sum circuit adds the pedestal-subtracted-samples from each Pedestal Subtraction Circuit on a clock by clock basis. Bit Bits circuit compare Resync data to TET and produce a low active signal when Resync data is above TET.

The architecture supports Processing Modularity. Processing algorithms are independent of the other functions.

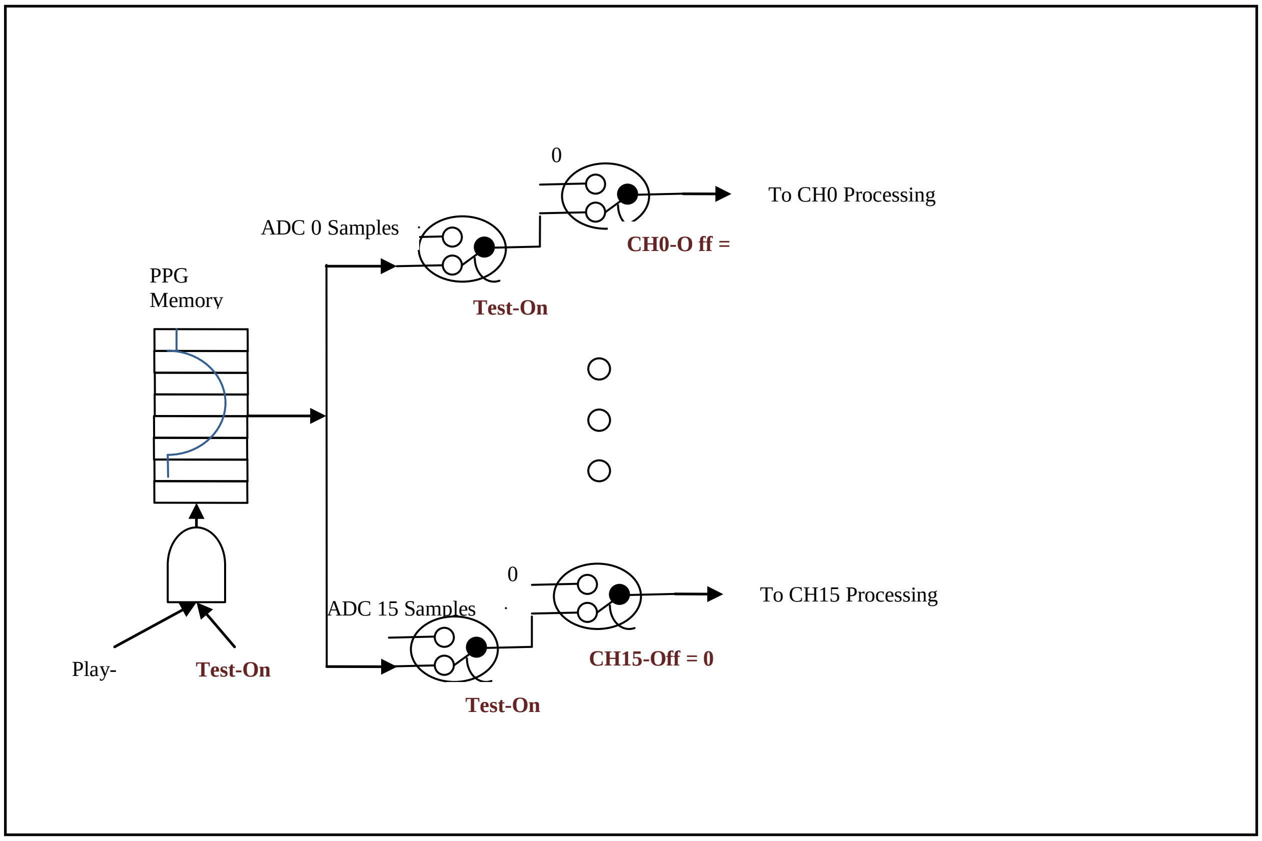
Sel block was added on March 3, 2008 to accommodate both 10 bits and 12 bits FADC boards This feature also allows individual ADC Channel values (counts) to be set to zero (effectively disable the ADC). CONF register (see below) configures these options.
On March 15, Sel block is expanded to include Programmable Pulse Generator.

Programmable Pulse Generator (PPG):
The PPG generates pulses by reading out digitized values of pulses (samples) stored in a memory. The memory has 32 locations. Each location has 16 bits and can hold one sample. Thirteen of the bits simulated the 12 data and 1 overflow bits output of the ADC implemented on the FAD250 board. The 16th bit facilitates writing and reading the memory. Samples are written to the PPG memory when VME write to address (0x0211 and 0x0011) and bit 16th is a one. The address automatically incremented after a sample is written. The last two samples written are required to have bit 16 zeroed (Sample value = 0x0000). After the last samples (0x0000) the address reset to the location of the first sample. Bits 14 and 15 are don't care bits. VME can verify the data is written by immediately read back the data (write follow by read).
Data are read out of PPG memory when Play-Back and Test-On are both logical one. The first location that will be read out is set by a register (Read-Out-Start-Address). Subsequent locations are read out at 4nS interval until Play-Back returns to logical zero. The read back cycles to Read-Out-Start-Address when bit 16th is a zero.

AUX_IO(1) is used as Play-Back.
Bit 7 of CONFIG register is used as Test-On
Bit 8-15 of CONFIG register is used as CHx-OFF

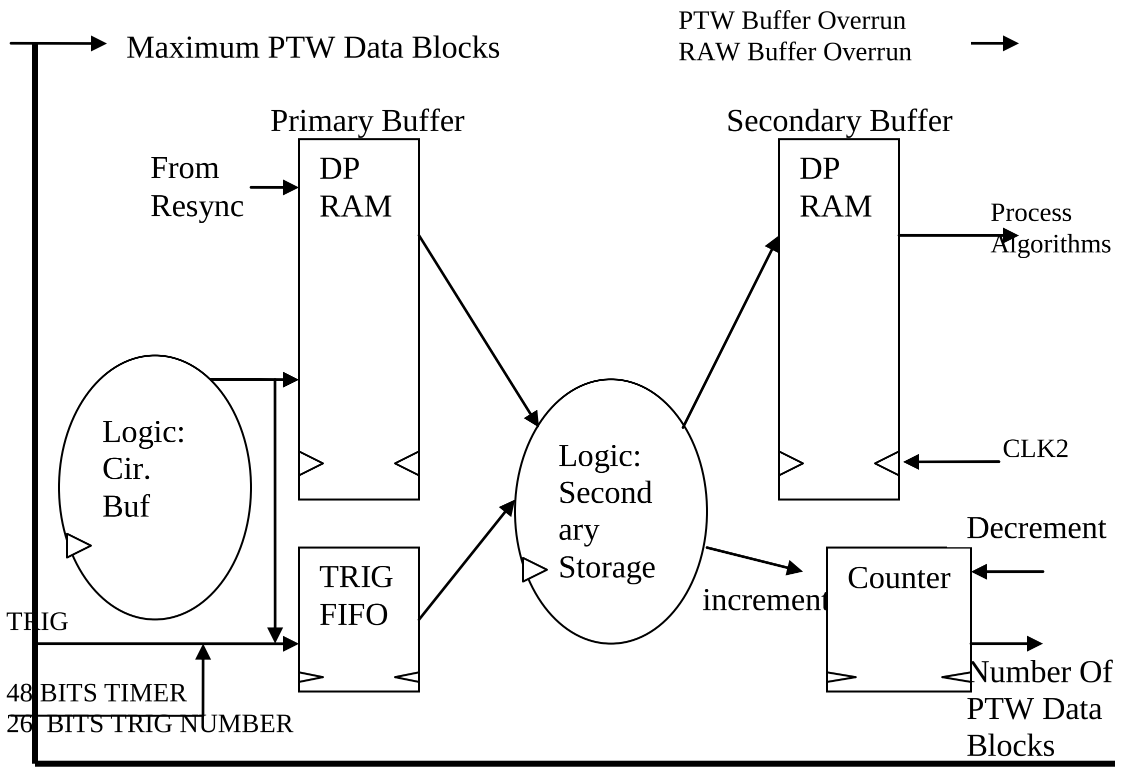
For the FADC250 Version 1 the PPG can only hold 16 samples. The last two samples have to be written with 0x8000.

PPG Mode:



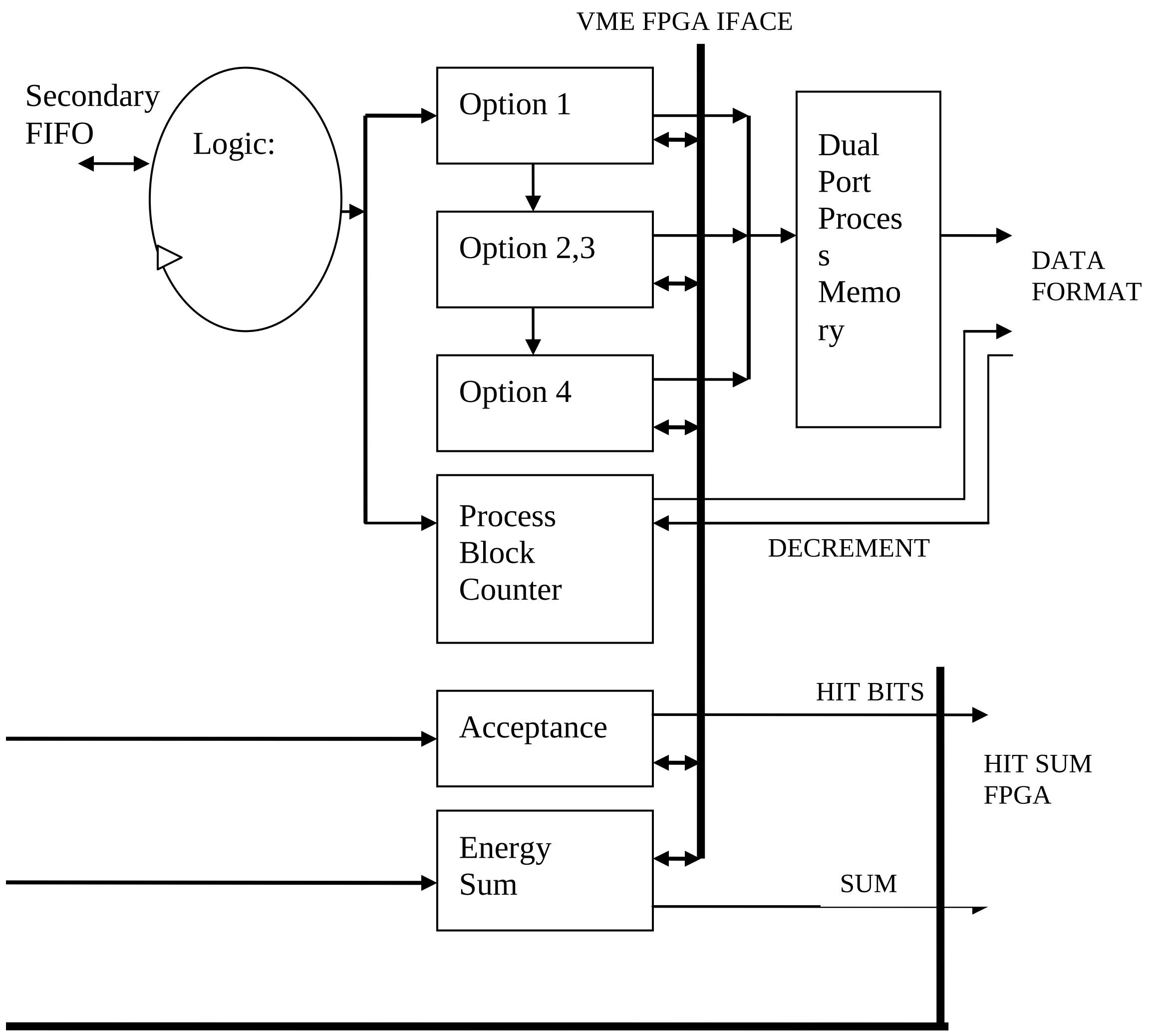
Test On is bit 6 of Configuration register. CHx-Off are bits 8-15 of configuration register.

Data Buffer:



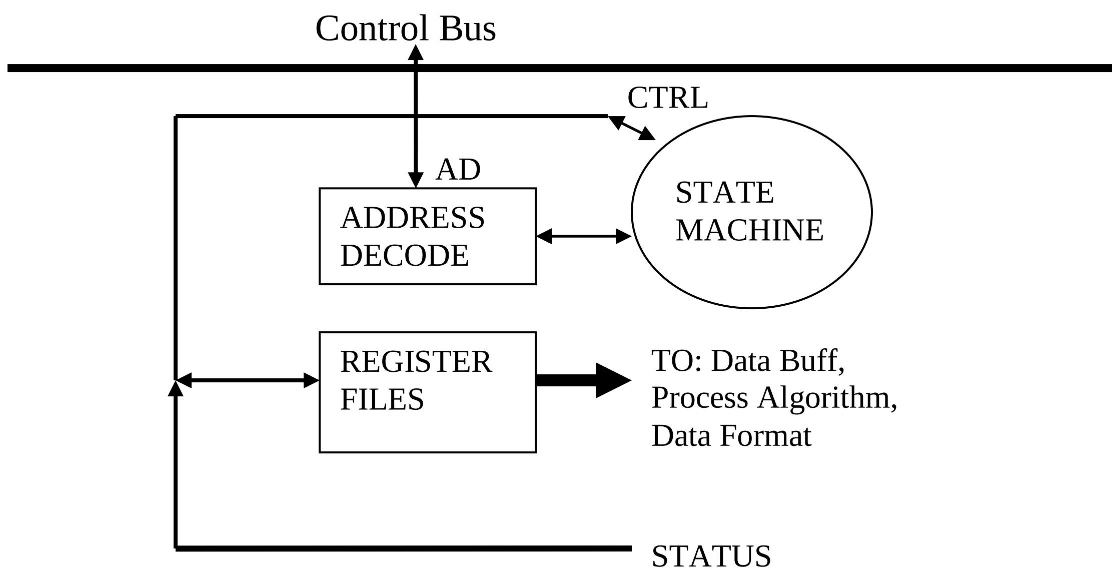
1. Synchronize data from ADC to 250 MHz FPGA CLK
2. Store ADC data to Primary Buffer. Implement Primary Buffer as ring (circular) buffer.
3. When a Trigger occurs, the trigger is stored along with the values of the 48 Bits Timer and the Pointer of the Primary Buffer in a FIFO.
4. For each trigger, data within Programmable Trigger Window are copied from Primary Buffer to Secondary Buffer with time stamps and markers necessary for further processing. After the block is copied, Number of PTW Data Blocks increments by one.
5. After a block is read and process, Decrement should be pulsed to decrease th Number of PTW Data Blocks by one.
6. Each ADC channel has its own Data Buffer.
7. When the Trigger Rate is faster then the time needed to copy ADC Data from Primary Buffer to Secondary Buffer, RAW BUFFER OVERRUN is set and remain set until RESET_N or SOFT_RESET_N goes low.
8. When Number of PTW Data Block is equaled Maximum PTW Data Blocks setted by the host, PTW Buffer Overrun sets and remains set until RESET_N or SOFT_RESET_N goes low.
9. Utilized 700 LUT, six 18000-bits RAM blocks. Max Clock is 252 MHz.

Process Algorithms:



1. Read data from Secondary Buffer.
2. Parse data to Processing Algorithms
3. Process all three options of Data Channel Processing.
4. Create Acceptance (Hit bit) pulse
5. Compute Energy Sum

VME FPGA IFACE:



VHDL Hierarchy

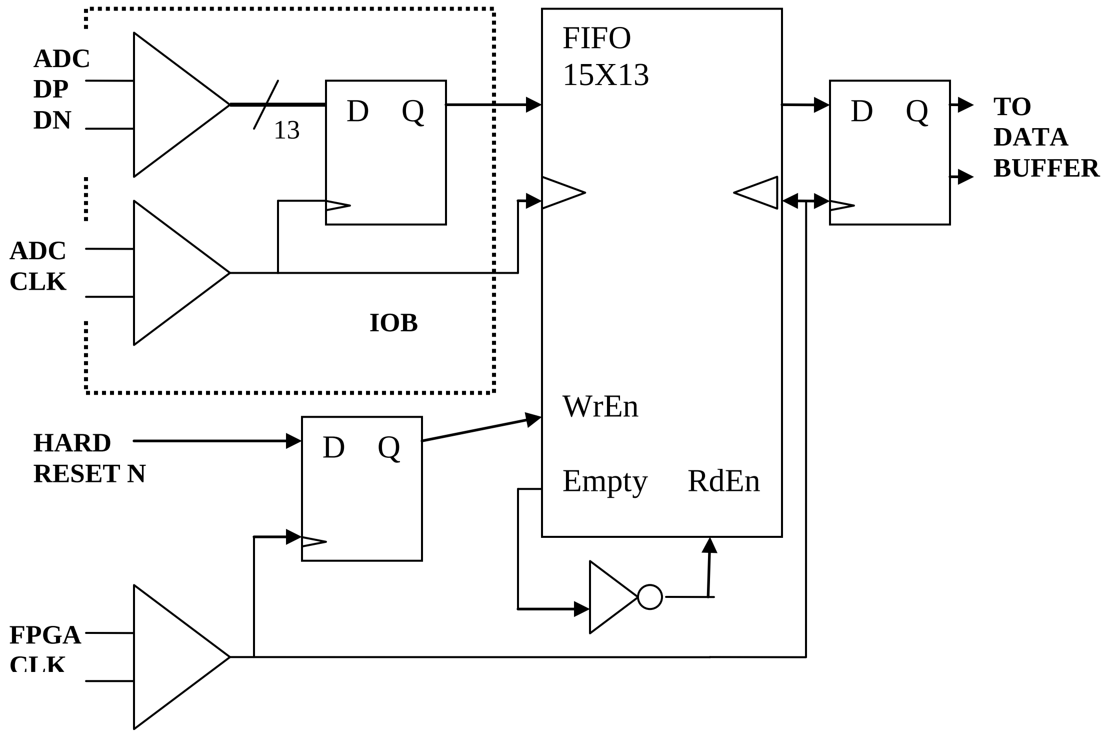
1. **ADC_PROC_TOP**
    a. SYNC_ADC_IN_VER2
        i. IODELAY
    b. PlayBack_WV_Ver2
        i. DPRAM_16_1024 (UMEM)
    c. Data_Buffer_AllCh_Ver2_TOP
        i. Data_Buffer_Top (UADCx)
            1. DP_RAM1_TOP (2Kx13) (URAW_BUFFER_IN)
            2. FIFO_1 (UTrigger_Buffer)
            3. DP_RAM2_TOP(2Kx17) (UPTW_DATA_BUF)
            4. PTWCPSM
    d. TimeStamp_TOP
        i. Time_stamp (Xilinx core gen)
    e. Trigger_Number_TOP
        i. Trigger_number
    f. Processing_All_Ver2_Top
        i. PROCESSING_TOP (CHx_PROCESSING)
            1. DP_RAM3_TOP (2Kx18) (UPROCESS_BUF)
            2. PROCESSM
            3. fifo_12_64 (UProcAdrHist)
            4. TDC_TOP
                a. Linear_Interpolation (ULI)
                    i. Divide_18By12
                        1. DIVIDESM
                b. TDCSM
        ii. PROALLSM
    g. DataFormat_VER2_TOP
        i. DATFORSM
    h. SUM_VER2_TOP (USUM_TOP)
    i. Hit_Bit_All_ver2_Top (UHIT_BITS_ALL_TOP)
        i. HIT_BITS_TOP (UHIT_x)
        ii. IOREG_8bits
2. **HOST_ADCFPGA_VER2_TOP**
    a. HSHOSTSM
3. **TRIG_PROC_TOP**
    a. HITBITS_TOP
        i. ONE_SHOT
        ii. ONE_SHOT_LONG (UTHIT_WIDTH)
        iii. DELAY_16bits_max32clk (UDELAY_16bits)
        iv. HIT_WINDOW
            1. Dpram_65k_1
        v. OVERLAP_WINDOW
    b. SUM_TRIG
    c. EXT_FIFO_WRITE
        i. TRIGGER_SYNC
            1. Fifo_4

    ii. EXTFIWRSM

d. HITSUMTOP2 (not connected at FADC250_V2_TOP)

VHDL Block Diagram

ADC Input ReSync



Each ADC has 12 bits data, an overflow, and an ADCCLK. The ADC Input Resync captures ADC’s data and overflow bits with ADC’s output clock to a 15 deep (smallest allow by ISE) by 13 bits FIFO. The FIFO allows the FPGA main CLK to be independent of ADC clock. The FPGA main CLK clocks the data out of FIFO and send to the Data Buffer Block.
The advantage of using ADC’s own CLK to capture its data is the elimination of timing variations from ADC to ADC. Moreover, the FIFO Empty signal is used as FIFO Read Enable to allow variation in ADC start up time.

Data Buffer

Primary Memory Map

Address Location	Content
0	ADC Data 0
1	ADC Data 1
2	ADC Data 2
3	ADC Data 3
4	ADC Data 4
:	:
:	:
:	:
4078	ADC Data 4078
4079	ADC Data 4079
4080	ADC Data 4080
0	ADC Data 4081
1	ADC Data 4082
2	ADC Data 4083
3	ADC Data 4084
:	:
:	:
:	:

Primary Memory stores ADC data as it comes in. At the end of buffer, the storing re-circulates and overwrites previous data.

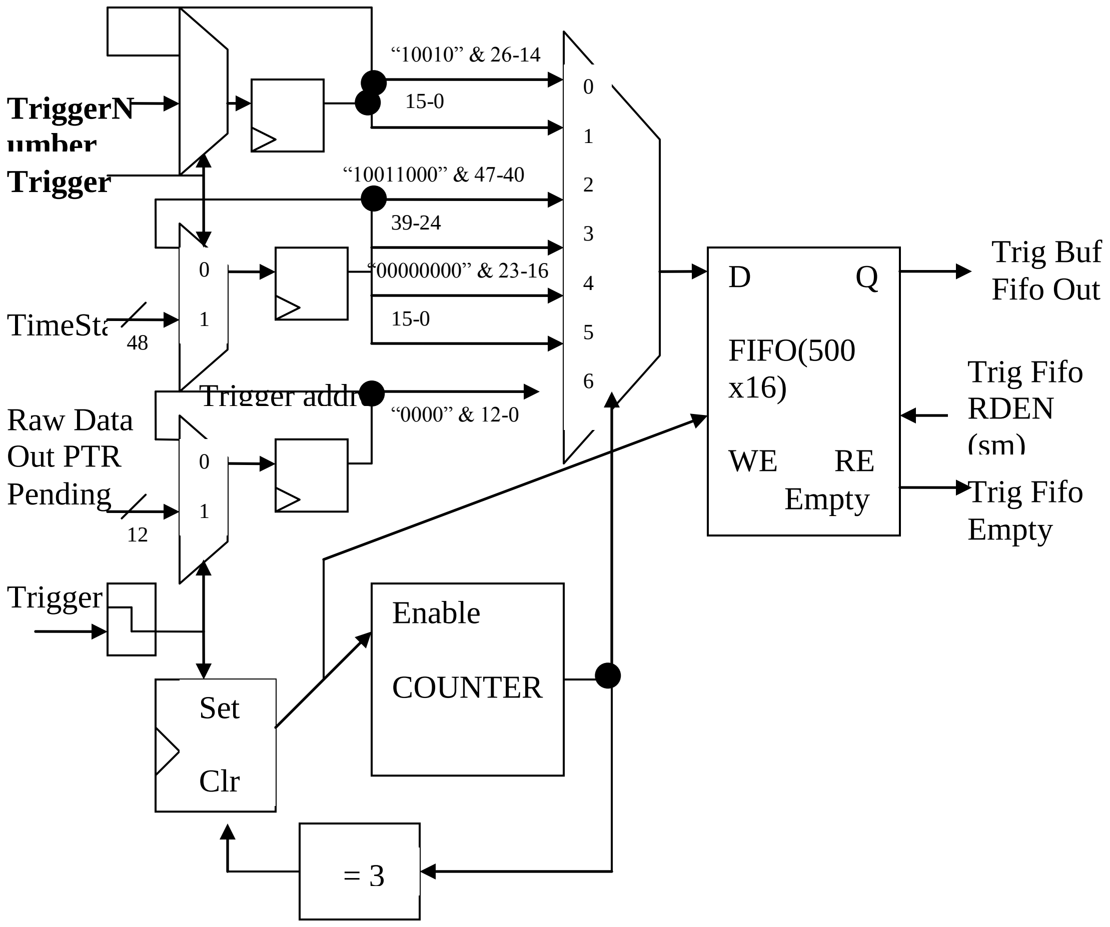
Data Buffer
Secondary Memory Map

Memory location from beginning of PTW	**Content**
0	PTW **0** “10010” Trigger Number bits 26-16
1	Trigger Number bits 15-0
2	“10011000” Time Stamp bits 47-40
3	Time Stamp bits 39-24
4	“00000000” Time Stamp bits 23-16
5	Time Stamp bits 15-0
6	PTW **0** data 0
:	PTW **0** data 1
:	:
N-4	“111” PTW **0** data N-4. “111” indicate almost last data
N-3	PTW **0** data N-3
N-2	PTW **0** data N-2
N-1	PTW **0** data N-1
N	PTW **0** last data
N+1	PTW **1** “10010” Trigger Number bits 26-16
N+2	Trigger Number bits 15-0
N+3	“10011000” Time Stamp bits 47-40
N+4	Time Stamp bits 39-24
N+5	“00000000” Time Stamp bits 23-16
N+6	Time Stamp bits 15-0
N+7	PTW **1** data 0
	PTW **1** data 1
	:
M-4	“111” PTW **1** data M-4. “111” indicate almost last data
M-3	PTW **1** data M-3
M-2	PTW **1** data M-2
M-1	PTW **1** data M-1
M	PTW **1** last data
M+1	PTW **2** “10010” Trigger Number bits 26-16
M+2	Trigger Number bits 15-0
M+3	“10011000” Time Stamp bits 47-40
M+4	Time Stamp bits 39-24
M+5	“00000000” Time Stamp bits 23-16
M+6	Time Stamp bits 15-0
M+7	PTW **2** data 0
	PTW **2** data 1
	:

O-4	“111” PTW **1** data O-4. “111” indicate almost last data
O-3	PTW **2** data O-3
O-2	PTW **2** data O-2
O-1	PTW **2** data O-1
O	PTW **2** last data

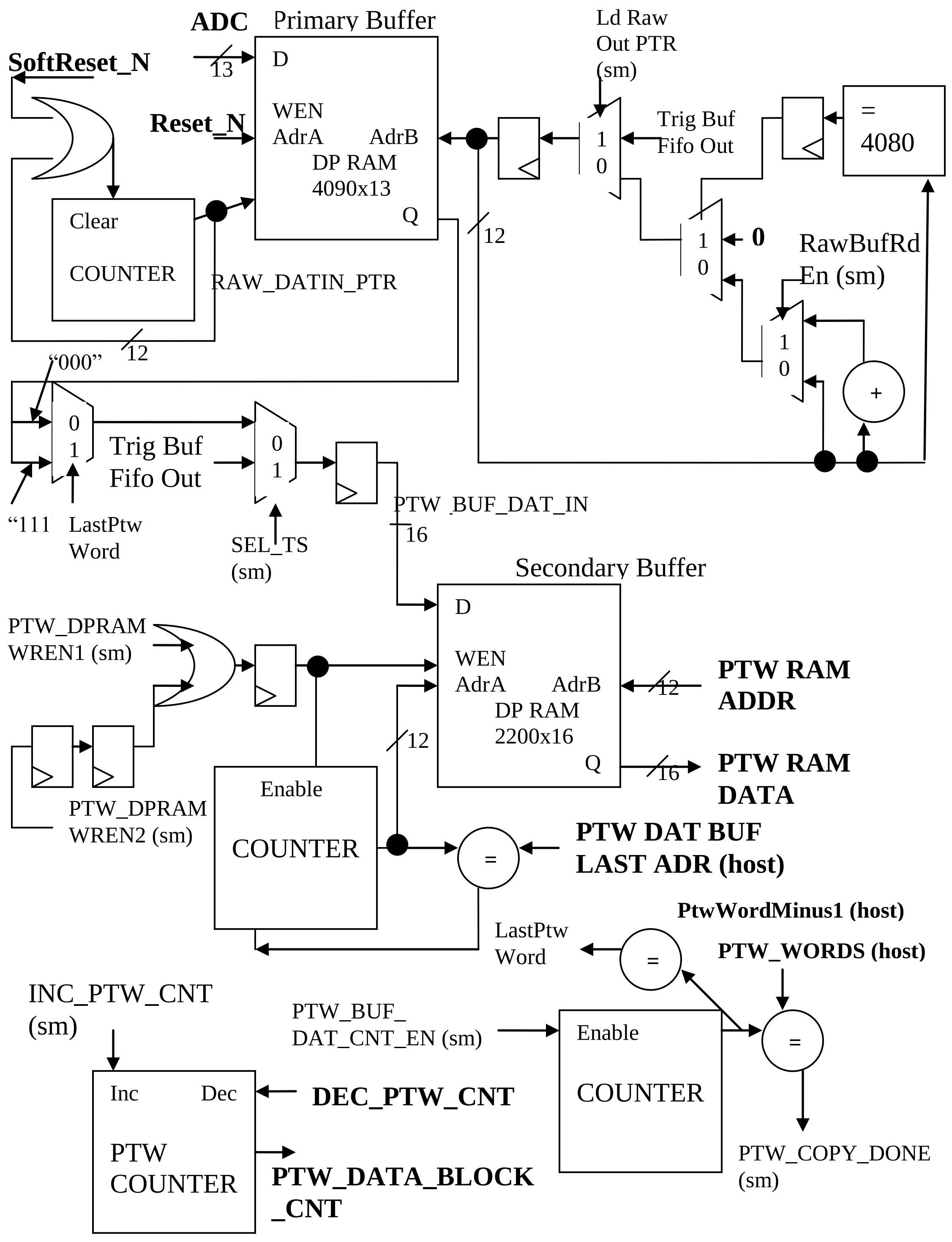
When a trigger occurs, a number of ADC data words (=PTW*25MHz) is copied from Primary to Secondary Buffer. The time at which the trigger occurred and the Trigger Number of Bits is included. Since the Number of ADC data words effects where the buffer ended and to minimize gate count, the location of the end of the buffers is provided by the Host Interface block. The Secondary Buffer Size is 2040 to accommodate 4 successive triggers of 2uS PTW (500 locations per trigger).

Trigger Buffer



When a trigger occurs, the time stamp and the pointer that points to beginning of Programmable Trigger Window (Raw Data Out PTR Pending) is store to 16 bit FIFO. The 48-bits time stamp is stored in 4 consecutive locations with LSB stored first. Bits 11-0 is padded with "1100" to signify the beginning of PTW window and Time Stamp Words. Bits 23-12, 35-24, and 47-36 are padded with "0100" to signify Time Stamp. After the first word is stored, TrigFifoEmpty goes high and kick off the State Machine to copy time stamp from FIFO to Secondary Dual-Port memory. Data in the PTW stored in the Primary Buffer starting at Trigger Address are copied to Secondary Buffer.

Data Buffer:
Primary and Secondary Buffer



After power up, data from ADC is stored in Ring Buffer continuously. When Trigger is in Trigger Buffer, the Time Stamp is copied from the Trigger Buffer to the Secondary Buffer. The Primary Address when the trigger occurred is retrieved from the Trigger Fifo to be used as the starting Primary address to copy ADC data over. A counter is keeping track of the number of ADC words copied. When the counter equaled the PTW words the copied process stop. Another counter that keeps track of the number of triggers that are in the Secondary Buffer ready for Process algorithm. When a block of trigger is process, this counter is decrement by the Process algorithm.

The Secondary Buffer storage is such that the starting address of each block of trigger data is determine by the PTW but it is fixed with PTW. For example, if PTW is 2uS, the starting address are 0, 504, 1008, 1512.   The data formats from low to high address are

```
“1000” “TS bits 47-36”
“1000” “TS bits 35-24”
“1000” “TS bits 23-12”
“1000” “TS bits 11-0”
“010”   “ TriggerNumber bits 26-14”
“01”    “ TriggerNumber bits 13-0”
“000”  “ADC data”
  :
  :
“001” “Last ADC data in PTW”
```

PTW Counter is coded such that when decrement commands and increment commands occurs exactly at the same time, decrement occurs before increment.

Data Buffer:

STATUS

Data Processing:
Memory Map

Data Processing Memory Assignment for Mode 0

Memory location from beginning of PTW	**Content (WITH EVENT)**
0	**“00” “10010” Trigger Number bits 26-16**
1	**“00” Trigger Number bits 15-0**
2	**“00” “10011000” Time Stamp bits 47-40**
3	**“00” Time Stamp bits 39-24**
4	**“00” “00000000” Time Stamp bits 23-16**
5	**“00” Time Stamp bits 15-0**
6	**“00” PTW data 0**
7	**“00” PTW data 1**
8	**“00” PTW data 2**
9	**“00” PTW data 3**
etc	**etc**
N+7	***”11” “FFFF” : end of PTW**
Memory location from beginning of PTW	**Content (WITHOUT EVENT)**
0	**“00” “10010” Trigger Number bits 26-16**
1	**“00” Trigger Number bits 15-0**
2	**“00” “10011000” Time Stamp bits 47-40**
3	**“00” Time Stamp bits 39-24**
4	**“01 “00000000” Time Stamp bits 23-16**
5	**“01” Time Stamp bits 15-0**
6	**“01” “0000”**
7	**“01” “0000”**
:	**:**
:	**:**
N+6	**N**
N+7	**“11” “0000” : end of PTW**

N = PTW

Data Processing Memory Assignment for Mode 1:

Memory location from beginning of PTW	Content (WITH EVENT)
0	“00” “10010” Trigger Number bits 26-16
1	“00” Trigger Number bits 15-0
2	“00” “10011000” Time Stamp bits 47-40
3	“00” Time Stamp bits 39-24
4	“00” “00000000” Time Stamp bits 23-16
5	“00” Time Stamp bits 15-0
6	“10” “0000” Pulse Number “00” SampleNumber from Thredhold bits 9-0
7	“00” PTW pulse 0 data 0
8	“00” PTW pulse 0 data 1
N	“00” PTW pulse 0 data last
N+1	“10” “0000” Pulse Number “01” SampleNumber from Thredhold bits 9-0
N+2	PTW pulse 1 data 0
N+3	PTW pulse 1 data 1
M	PTW pulse 1 data last
M+1	“10” “0000” Pulse Number “10” SampleNumber from Thredhold bits 9-0
M+2	PTW pulse 2 data 0
M+3	PTW pulse 2 data 1
O	PTW pulse 2 data last
O+1	“10” “0000 Pulse Number “11” SampleNumber from Thredhold bits 9-0
O+2	PTW pulse 3 data 0
O+3	PTW pulse 3 data 1
P	PTW pulse 3 data last
P+1+7	“11” “0000” : end of PTW
Memory location from beginning of PTW	**Content (WITHOUT EVENT)**
0	“00” “10010” Trigger Number bits 26-16
1	“00” Trigger Number bits 15-0
2	“00” “10011000” Time Stamp bits 47-40
3	“00” Time Stamp bits 39-24
4	“01” “00000000” Time Stamp bits 23-16
5	“01” Time Stamp bits 15-0
6	x”10000”
7	x”10000”
8	“11” “0000” : end of PTW

Data Processing Memory Assignment for Mode 2:

Memory location from beginning of PTW	Content (WITH EVENT)
0	“00” “10010” Trigger Number bits 26-16
1	“00” Trigger Number bits 15-0
2	“00” “10011000” Time Stamp bits 47-40
3	“00” Time Stamp bits 39-24
4	“00” “00000000” Time Stamp bits 23-16
5	“00” Time Stamp bits 15-0
6	“10” “0000” Pulse Number “00” SampleNumber from Thredhold bits 9-0
7	“00” Pulse 0 Sum bits 18-3
8	“00” “0000000000000” Pulse 0 Sum bits 2-0
9	“10” “0000” Pulse Number “01” SampleNumber from Thredhold bits 9-0
10	“00” Pulse 1 Sum bits 18-3
11	“00” “0000000000000” Pulse 1 Sum bits 2-0
12	“10” “0000” Pulse Number “10” SampleNumber from Thredhold bits 9-0
13	“00” Pulse 2 Sum bits ~~18-3~~ 20-5
14	“00” “0000000000000” Pulse 2 Sum bits ~~2-0~~ 4-0
15	“10” “0000” Pulse Number “11” SampleNumber from Thredhold bits 9-0
16	“00” Pulse 3 Sum bits 18-3
17	“00” “0000000000000” Pulse 3 Sum bits 2-0
18	“11” “0000” : end of PTW
Memory location from beginning of PTW	**Content (WITHOUT EVENT)**
0	“00” “10010” Trigger Number bits 26-16
1	“00” Trigger Number bits 15-0
2	“00” “10011000” Time Stamp bits 47-40
3	“00” Time Stamp bits 39-24
4	“01” “00000000” Time Stamp bits 23-16
5	“01” Time Stamp bits 15-0
6	x”10000”
7	x”10000”
8	“11” “0000” : end of PTW

Data Processing Memory Assignment for Mode 3:

Memory location from beginning of PTW	**Content (WITH EVENT)**
0	**“00” “10010” Trigger Number bits 26-16**
1	**“00” Trigger Number bits 15-0**
2	**“00” “10011000” Time Stamp bits 47-40**
3	**“00” Time Stamp bits 39-24**
4	**“00” “00000000” Time Stamp bits 23-16**
5	**“00” Time Stamp bits 15-0**
6	**“10” “0000” Pulse Number “00” SampleNumber from Thredhold bits 9-0**
7	**“00” Tfine(5..0) Vmin (11..4)**
8	**Vmin(3..0) Vp (11..0)**
9	**“10” “0000” Pulse Number “01” SampleNumber from Thredhold bits 9-0**
10	**“00” Tfine(5..0) Vmin (11..4)**
11	**“00” Vmin(3..0) Vp (11..0)**
12	**“10” “0000” Pulse Number “10” SampleNumber from Thredhold bits 9-0**
13	**“00” Tfine(5..0) Vmin (11..4)**
14	**“00” Vmin(3..0) Vp (11..0)**
15	**“10” “0000” Pulse Number “11” SampleNumber from Thredhold bits 9-0**
16	**“00” Tfine(5..0) Vmin (11..4)**
17	**“00” Vmin(3..0) Vp (11..0)**
18	**“11” “0000” : end of PTW**
Memory location from beginning of PTW	**Content (WITHOUT EVENT)**
0	**“00” “10010” Trigger Number bits 26-16**
1	**“00” Trigger Number bits 15-0**
2	**“00” “10011000” Time Stamp bits 47-40**
3	**“00” Time Stamp bits 39-24**
4	**“01” “00000000” Time Stamp bits 23-16**
5	**“01” Time Stamp bits 15-0**
6	**x”10000”**
7	**x”10000”**
8	**“11” “0000” : end of PTW**

Data Processing:

Data Processing for all mode involves scanning the entire secondary buffer. If there is no pulses (data that cross thredshold), x"FFF0" is written to processing memory (PTW) data locations. Trigger Number and Time Stamp info are copied from secondary buffer to processing memory. X"FFF0" signal DataFormat block to prevent data from written to external FIFO. This feature only writes ADC channel that has data that cross thresdhold (TET).

Data Processing consists of 4 state machines, counters, and pointers. The 4 State Machines include Main and one for each of the 3 Processing Options. When there is ADC data to process, Main State Machine read Time Stamps and Trigger Number from Secondary Buffer and write to Data Processing Buffer. It then calls on one of the other three state machines to process the Option that is in effect.

The state machine for option 1 does the following:

1. Copies PTW * 20MHz number of words from Secondary Data Buffer to Data Processing.
2. Increment number of process counter by one

The state machine for option 2 does the following:

1. Read ADC data from Secondary Data Buffer. Start PULSE_TIMER to tick mark the data read.
2. If ADC data is above Trigger Threshold, it writes PULSE_TIMER to Process Buffer. Then it copies NSB and NSA number of words from Secondary Data Buffer to Data Processing Buffer as follow:
    a. If the number of words read before threshold is greater than NSB load RD_PTW_PTR with address that is NSB before threshold. If the number of words read (WORD_AFTER_TS_CNT) is less than NSB, load the RD_PTW_PTR with address of WORD_AFTER_TS_CNT word back from threshold.
    b. Start NSB_CNT.
    c. When NSB_CNT = NSB if WORD_AFTER_TS_CNT > NSB **or** NSB_CNT = WORD_AFTER_TS_CNT if WORD_AFTER_TS_CNT < NSB start NSA_CNT.
    d. When NSA_CNT = NSA, it stop reading Secondary Buffers.
3. Repeat Step 1 and 2 until number (PTW * 250MHz) numbers of words have been read.
4. Write "FFFF" to signal the end of PTW.
5. Increment number of process counter by one

The state machine for option 2 does the following:

1. Read ADC data from Secondary Data Buffer.
2. If ADC data is above Trigger Threshold, it unable accumulated sum circuit to add ADC value from NSB to NSA ADC words.
3. Write accumulated sum to Secondary Data Buffer
4. Repeat Step 1, 2, and 3 until number number PTW * 20MHz numbers of words have been read.
5. Write "FFFF" to signal the end of PTW.

6. Increment number of process counter by one

In mode 2 and 3, when the number of words read before the ADC value exceeds the Trigger Threshold is less then NSB, only that many word are processed.

Each state machine is responsible to change and reset the counters that pertained to the option.

The counters and their functions are listed below.

1. WORD_AFTER_TS_CNT: keep track of words read from beginning of PTW to the ADC sample that exceeds the Trigger Threshold. If WORD_AFTER_TS_CNT is less then NSB when this Threshold exceeded occurs, the NSB_PTR_ENOUGH pointer is used as starting address. Only WORD_AFTER_TS_CNT number of word before Threshold is processed.
2. TS_CNT: keep track of the number of time stamp and trigger number words read from the Secondary Buffer. Main state machine uses this to stop copying time stamp and trigger number words.
3. PTW_WORDS_CNT: keep track of the number of word in PTW has been read out. It is cleared when it is equaled to number of "PTW words + 4 Time Stamp words + 2 Trigger Number words".
4. NSB_CNT: Keep track of the number of words before Threshold has read and process.
5. NSA_CNT: Keep track of the number of words after Threshold has read and processed.
6. PULSE_TIMER: Tick mark ADC data read from Secondary Buffer from beginning of PTW.
7. PULSE_NUMBER: Keep track of the number of pulses in PTW.
8. HOST_BLOCK_CNT: Keep track of the number of PTW ready to transfer to host. The host decrement this counter after the host read one PTW.

The pointers and theirs functions are listed below:

1. NSB_PTR_ENOUGH: This pointer is used as starting address if the number of words read from PTW beginning to Threshold is greater than NSB value. A number of NSB words is processed.
2. NSB_PTR_NOT_ENOUGH: This pointer is used as starting address if the number of words read from PTW beginning to Threshold is less than NSB value. Only WORD_AFTER_TS_CNT number of word is processed.

Counters that also serve as pointers are listed below:

1. RD_PTW_PTR: This is the address to the Secondary Buffer. It is load with either NSB_PTR_ENOUGH or NSB_PTR_NOT_ENOUGH and increment under state machine control. It is cleared (restart at address 0) when PTW_WORDS_CNT is equaled to "number of PTW words + 4 Time Stamp words + 2 Trigger Number words".

TDC Algorithm Overview:

The TDC algorithm calculates time of the mid value of a pulse relative to the beginning of the look back window. The mid value is the value between the smallest and the peak value of the pulse. The smallest value is the beginning of the pulse. The time consists of coarse time and fine time. The coarse time is the number of clock the sample value before the mid value and the fine time is the interpolating value of mid value away from next sample. The coarse value is 10 bits and the fine value is 6 bit. The resolution of LSB is 1/(CLK * 64). For a 250MHz the resolution is 62.5 pS. For example for a 20MHz clock, a pulse time value of 110 means the mid-point of the pulse occurred at 6.875nS (62.5pS * 110) from the beginning of look back window.

Requirements for TDC Algorithm:

ii) There must be at least 5 samples (background) before pulse. Four of these samples are used to determine the pedestal (Vnoise) floor. The minimum value of the pulse is the first value that is Vnoise.

TDC Algorithm for Mode 3:

1) Search for Vaverage
   a) Latch starting PTW_RAM ADR
   b) Read four samples. Vnoise = Average of 4 samples. Increment sample count by 4.
   c) Vmin = Vnoise. Increment sample count.
   d) Read until Vram < Vram_delay. Vpeak = Vram if Vram is greater than TET. Increment sample count.
   e) Store PTW_RAM ADR for Vpeak.
   f) Vaverage = (Vpeak – Vmin) / 2
2) Search for sample before (Vba) and sample after (Vaa) Vaverage
   a) Restore starting PTW_RAM ADR. Increment Pulse Timer whenever the address is incremented.
   b) Read until Vram > Vmin. Vba = Vram
   c) Read one more for Vaa
   d) Calculated Tfine
3) Write Pulse Number and Pulse Timer to Processing RAM.
4) Increment Pulse Number
5) Restore PTW_RAM ADR for Vpeak. Load sample count to Pulse Timer.
6) Read until Vram < Vmin. End of first pulse. Increment Pulse Timer whenever the address is increment. End processing whenever PT_RAM data is ended.
7) Go to step 1.

Data Format :

Data format read data from Data Processing Memory, put the data in proper format as described in FADC Data Format, and write to external FIFO to host. The data format falls into 5 categories: Event_Header, Time_Stamp, Window_Raw_Word1, Pulse_Raw_Word1, Window_Pulse_Raw_Words_2_to_N, Pulse_Integral and Event_Trailer. The words are 36 bits wide.

Event_Header indicates the start of an event and bits are assigned as follow:
(35-34) = 0
(33-32) = 1
(31) = 1
(30-27) = 2
(26-0) = trigger number

➔ x"19 trigger number"

Trigger Time (Time_Stamp) indicates time of trigger occurrence relative to the most recent global reset. The six bytes (48 bits) of trigger time Ta Tb Tc Td Te Tf are format in two 32-bits words:

Word1:
(35-34) = 0
(33-32) = 0
(31) = 1
(30-27) = 3
(26-24) = 0
(23-16) = Ta
(15-8) = Tb
(7-0) = Tc

➔ x"0980 time stamp hi

Word2:
(35-34) = 0
(33-32) = 0
(31) = 0
(30-24) = 0
(23-16) = Td
(15-8) = Te
(7-0) = Tf

➔ x"0000 time stamp lo

Window Raw Word1 indicates the beginning of Window Raw Data.
(35-34) = 0
(33-32) = 0
(31) = 1
(30-27) = 4
(26-23) = Channel number (0-7)

(22-12) = 0
(11-0) = Window Width (PTW) (in number of samples).
➔ x"0A ChannelNumber 00 numberOfSamples"

```
3322 2222 2222 1111 1111 1198 7654 3210
1098 7654 3210 9876 5432 10
----------------------------------------------------
1010 0Cha n000 0000 0000 Ptw-  ---- ----
```

Pulse Raw Word1 indicates the beginning of Pulse Raw Data.
(35-34) = 0
(33-32) = 0
(31) = 1
(30-27) = 6
(26-23) = Channel number (0-7)
(22-21) = pulse number (0-3)
(20-10) = 0
(9-0) = time from beginning of PTW that the pulse crossed thredshold
➔ x"0B ChannelNumber 00 TIME"

```
3322 2222 2222 1111 1111 1198 7654 3210
1098 7654 3210 9876 5432 10
----------------------------------------------------
1011 0Cha nP#0 0000 0000 00Ti  me-- ----
```

Remaining words for Pulse Raw Data and Window Raw Data have the same format.
(35-34) = 0
(33-32) = 0
(31) = 0
(30) = 0
(29) = 1 indicates sample x not valid
(28-16) = ADC sample x (includes overflow bit)
(15-14) = 0
(13) = 1 indicates sample x+1 not valid.
(12-0) = ADC sample x+1 (includes overflow bits).

```
3322 2222 2222 1111 1111 1198 7654 3210
1098 7654 3210 9876 5432 10
----------------------------------------------------
00xA dcSa mple   ----    00xA dcSa mple ----
```

Pulse Time (8) – time associated with an identified pulse within the trigger window.
(31) = 1
(30 – 27) = 8
(26 – 23) = channel number (0 – 15)

(22 – 21) = pulse number (0 – 3)
(20 – 19) = measurement quality factor (0 – 3)
(18 - 16) = reserved (read as 0)
(15 – 6) = coarse pulse time
  (5 – 0) = fine pulse time

```
3322 2222 2222 1111 1111 1198 7654 3210
1098 7654 3210 9876 5432 10
-----------------------------------------------------
1100 0Cha nP#0 0000 Puls  eTim e
```

Pulse Integral (7) – integral of an identified pulse within the trigger window. The pulse integral may be a simple sum of raw data samples over the pulse duration, or the result of a complex fit to pulse shape. Pedestal subtraction may be included.

(31) = 1
(30 – 27) = 7
(26 – 23) = channel number (0 – 15)
(22 – 21) = pulse number (0 – 3)
~~(20 – 19) = measurement quality factor (0 – 3)~~
~~(18 – 0) = pulse integral~~
(20-0) = pulse integral

```
3322 2222 2222 1111 1111 1198 7654 3210
1098 7654 3210 9876 5432 10
-----------------------------------------------------
1011 1Cha nP#0 0Pul  seIn    tegr  al
```

Pulse Vmin Vpeak (10) –   ADC count for minimum and peak value of a pulse. This is too be used off line to apply correction to Pulse Time in TDC mode.

(31) = 1
(30 – 27) = 10
(26 – 23) = channel number (0 – 15)
(22 – 21) = pulse number (0 – 3)
(20 – 12) = Vmin
(11 – 0)   = Vpeak

```
3322 2222 2222 1111 1111 1198 7654 3210
1098 7654 3210 9876 5432 10
-----------------------------------------------------
1101 0Cha nP#v minn nnn  vpea kkkk kkkk
   D
```

Event Trailer: Indicate the end of an event.

```
EVENT_TRAILER = "0010" & X"E8000000";
```

Example:

Raw Data (mode0) :

x”19_____”   Event Header
x”98_____ “ Time Stamp upper 24 bits.
x”_______”   Time Stamp lower 24 bits.
x”A_____” Channel Number, Window Width (PTW)
x”______”   Raw Data
x”2E8000000” End of Event

Pulse Data (mode 1):

x”19_____”   Event Header
x”98_____ “ Time Stamp upper 24 bits.
x”_______”   Time Stamp lower 24 bits.
x”B______” ChanNum(26-23), PulseNumb(22-21),Time from beginning of PTW that the pulse crossed thredshold(9-0).
x”_______” 2 pulses (12-0) (28-16) per 36 bits words.
x”2E8000000” End of Event

Pulse Sum (mode 2):

x”19_____”   Event Header
x”98_____ “ Time Stamp upper 24 bits.
x”_______”   Time Stamp lower 24 bits.
x”C______” Pulse time, ChanNum(26-23), PulseNumb(22-21),Time(15-0)
x”B8_____” Channel Numbe(26-23)r, Pulse Number(22-21), Pulse Integral (18-0)
x”2E8000000” End of Event

TDC (mode 3):

x”19_____”   Event Header
x”98_____ “ Time Stamp upper 24 bits.
x”_______”   Time Stamp lower 24 bits.
x”C______” Pulse time, ChanNum(26-23), PulseNumb(22-21),Time(15-0)
x”D______” ChanNum(26-23), PulseNumb(22-21),Vm(20-12),Vp(11-0)
x”2E8000000” End of Event

Raw Data and TDC (mode 7)

x”19_____”   Event Header
x”98_____ “ Time Stamp upper 24 bits.
x”_______”   Time Stamp lower 24 bits.
x”A_____” Channel Number, Window Width (PTW)
x”______”   Raw Data
x”C______” Pulse time
x”D______” VminVpeak
x”2E8000000” End of Event

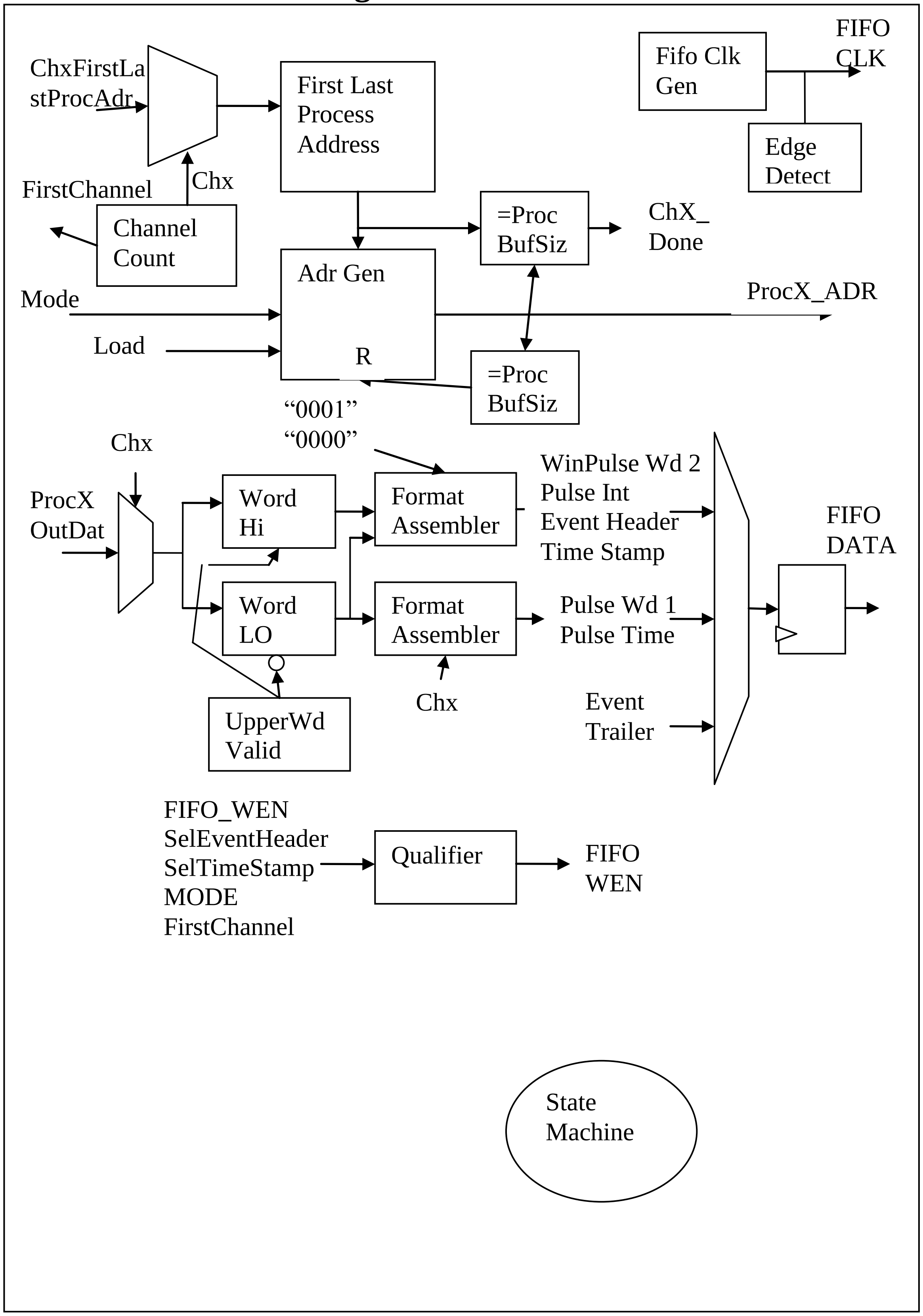
Data Format VHDL:

The VHDL code read data streams from processing block, format them per document "FADC Data Format" by Ed Jastrzembski.
When all HOST_BLOCKx_CNT is greater then one, DATFORSM begins the write out algorithm. The algorithm is as follow:
   1) Pop the starting and last address of the data in the processing buffer.
   2) Load the starting adddress of Channel 0 to Address counter. Start FIFO clock. Inc Address counter on rising edge of FIFO clock.
      Output Address to PROCx_ADR
   3) Read data from PROCx_OUTDAT. Assemble them into Event Header, TimeStamp1, and TimeStamp2 and writes to FIFO.
   4) In mode 0, the Address is stop after TimeStamp2 address (5) to allow time to insert Window Raw Data Word 1 which contains Channel Number and Window Width.
   5) In mode 1 and mode 2, the Address is stop after Pulse Number and SampleNumber from Threshold Address (6), to allow time to assemble Pulse Raw Data Word 1 which contains Channel Number and
      first sample number for pulse or Pulse Time which contains Channel Number and pulse time.
   6) The data are read and write in pairs until the Address counter equal last address of the processing buffer. The channel are incremnent and repeats step 1 through 6.
   7) After the last channel is finish, Event Trailer is written to FIFO.

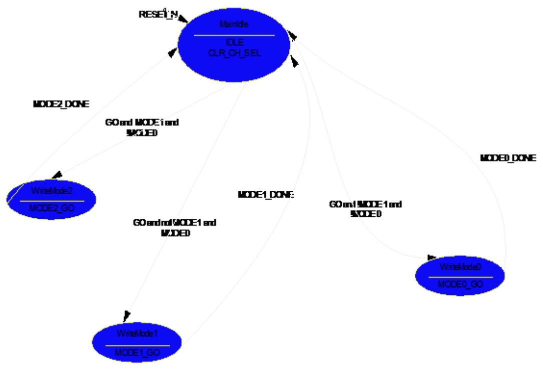
Because of the different in the data format between the modes: 0,1,and 2, each mode has its own state machine.
In mode 2, there might be extra words (for some setting of NSA and NSB) in the processing buffer after the last integral, the statemachine does not write this to FIFO.
In mode 0 and 1, for even number of data, the number of data written to FIFO is 2 more, for odd number of data, the number of data written to FIFO is one more.

Data Streams from Processing for diferent modes:
In mode 0: EventHeader, TimeStamp1, TimeStamp2, WindowRaw(not from processing), Deven Dodd,..., TimeEnd
In mode 1: EventHeader, TimeStamp1, TimeStamp2, PulseRaw(upper 16 from processing, not lower 16), Deven Dodd,..., TimeEnd
In mode 1: EventHeader, TimeStamp1, TimeStamp2, PulseRaw(upper 16 from processing, not lower 16), Integral, TimeEnd

Data Format VHDL Diagram



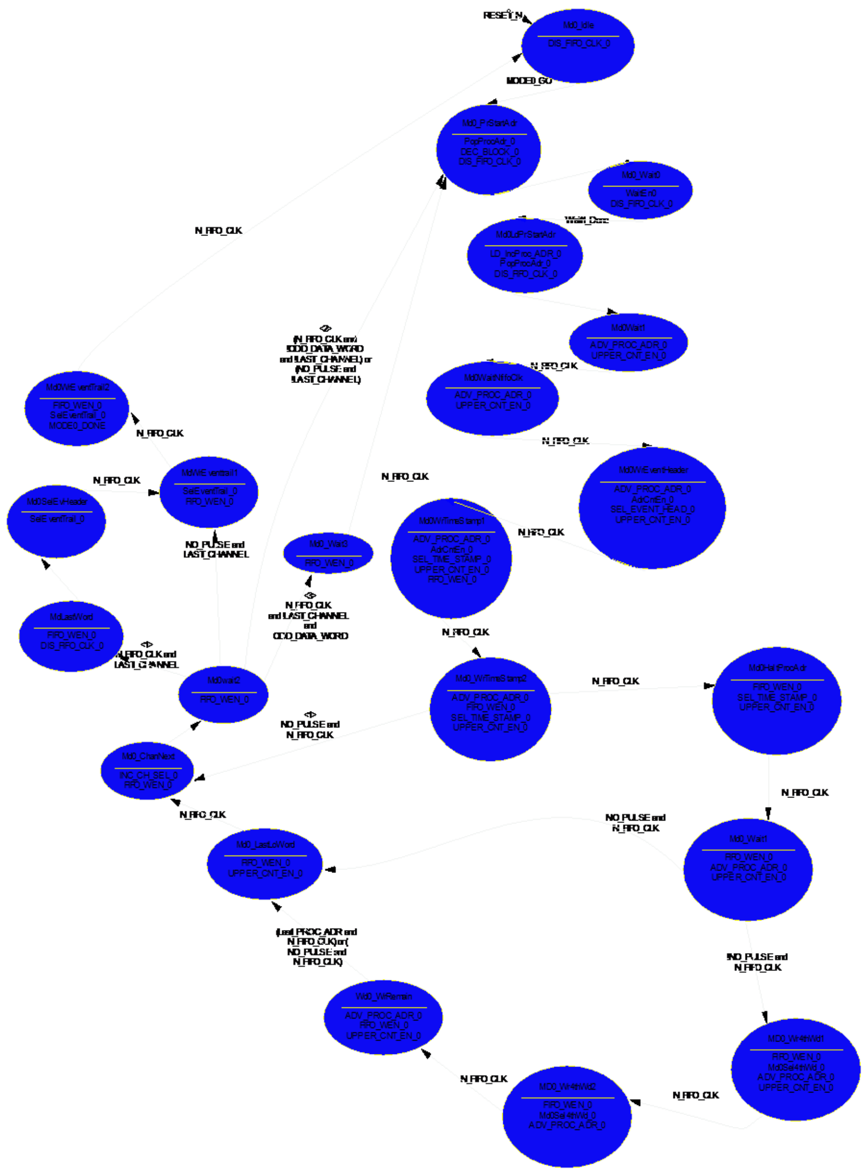
Data Format State Machine Main



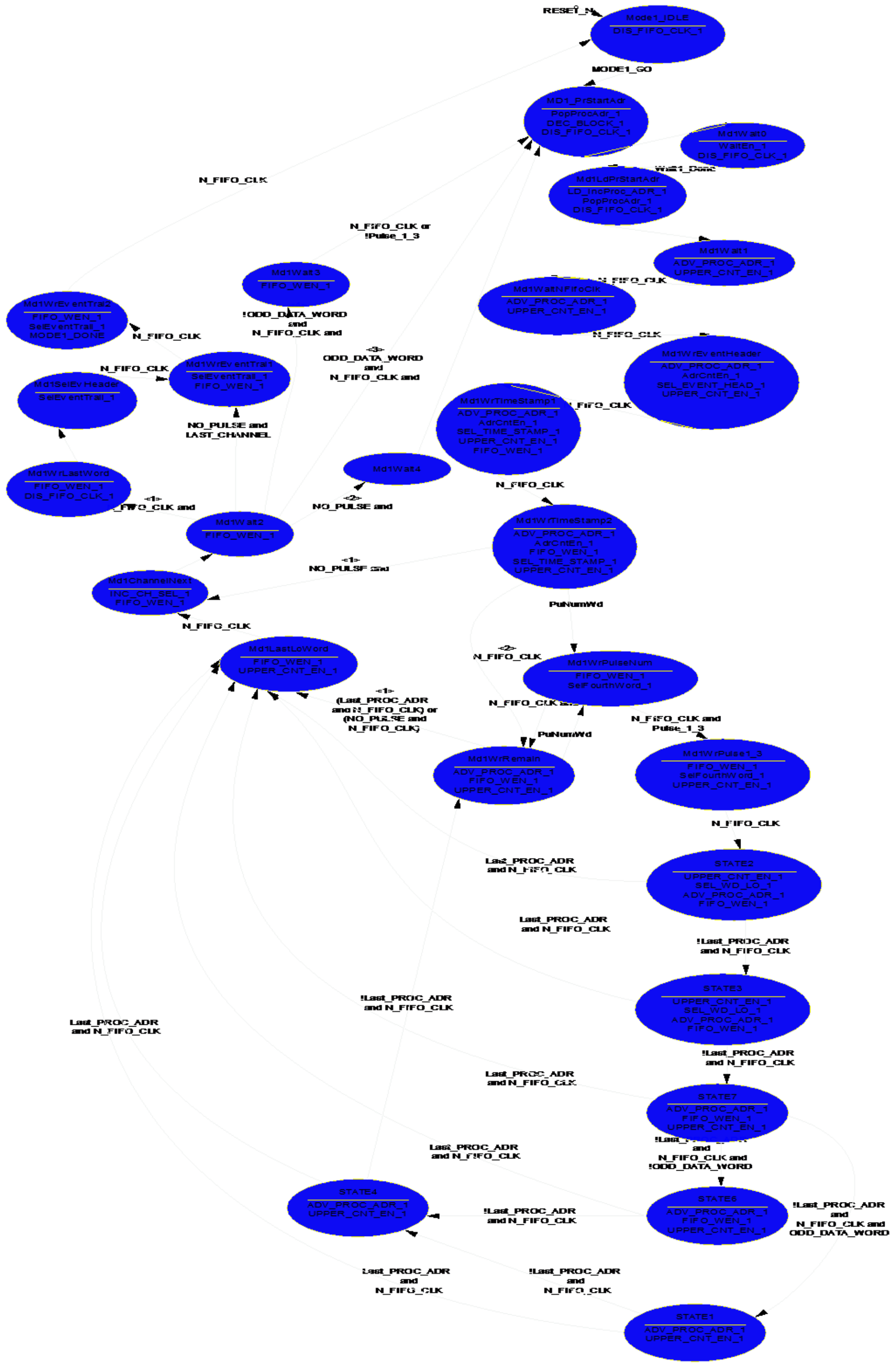
Main State Machine does the following:

1. Call State Machine for Mode 0,1,or 2.

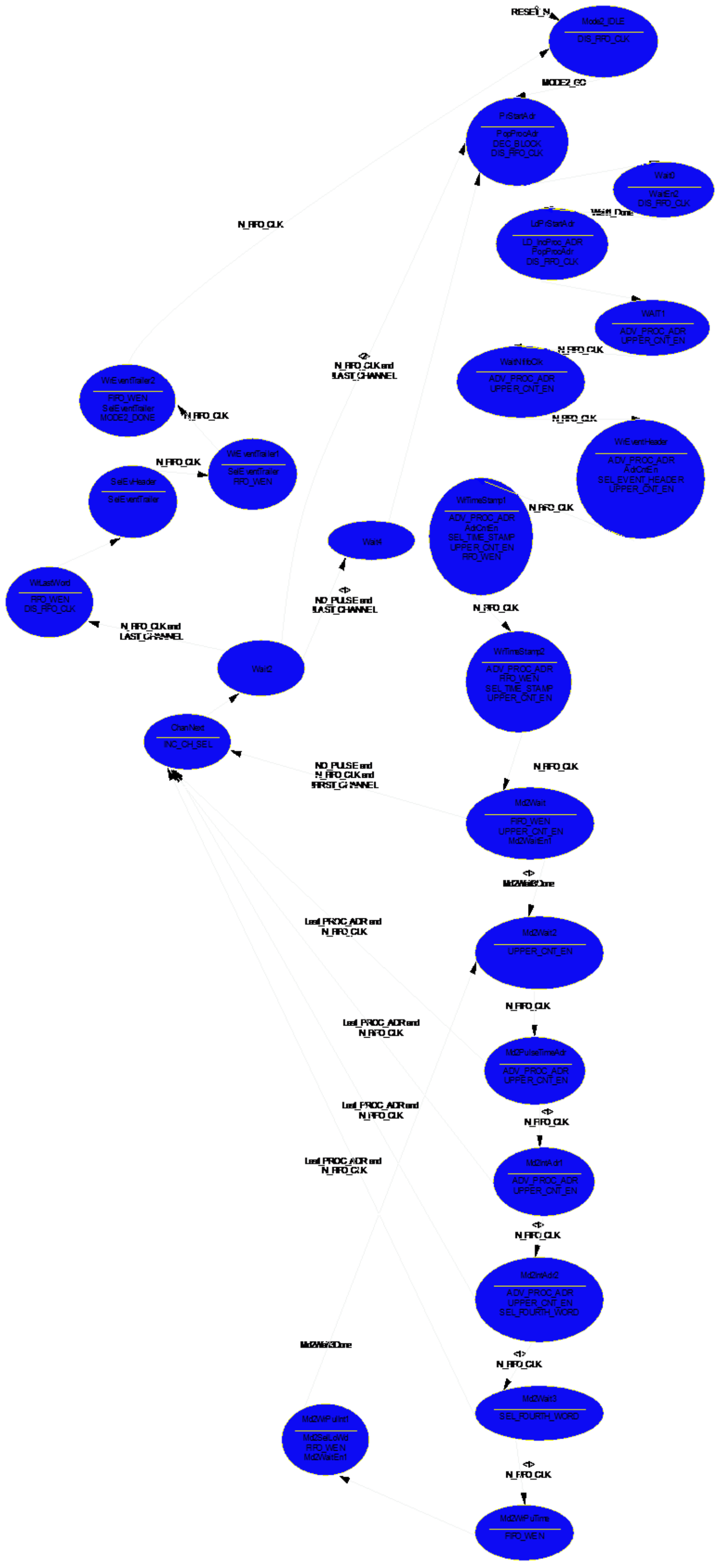
Data Format State Machine For Mode 0



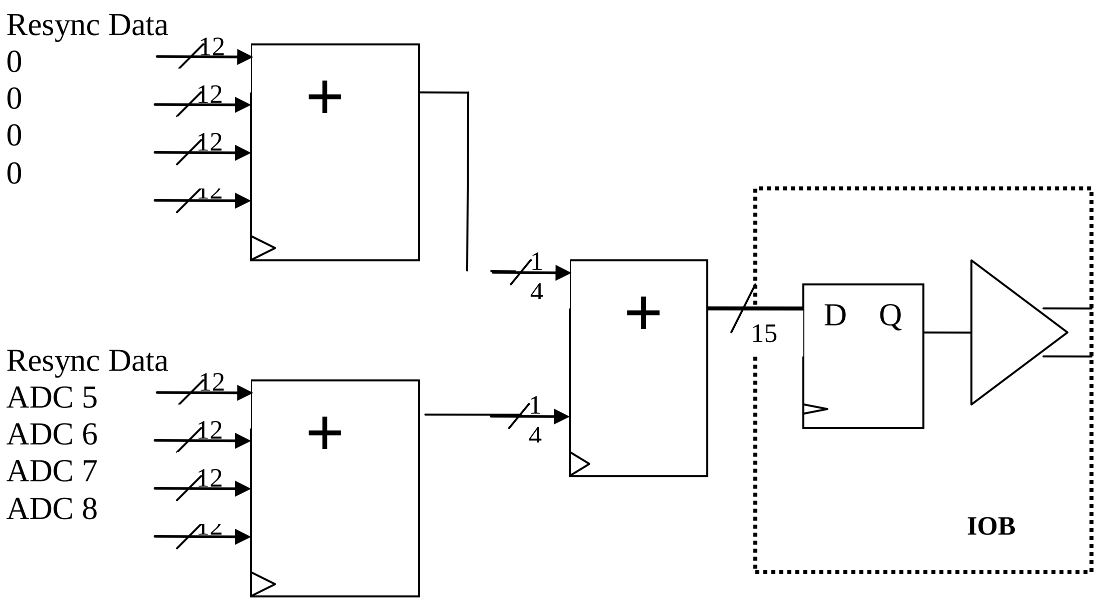
Data Format State Machine RD For Mode 1



Data Format State Machine RD For Mode 2



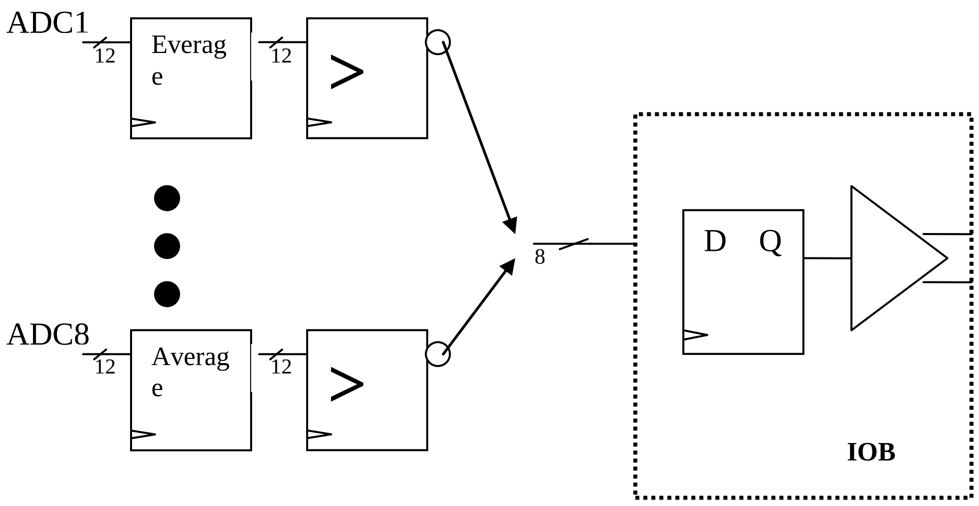
SUM



Resync Data
0
0
0
0
12
12
12
12
+
Resync Data
ADC 5
ADC 6
ADC 7
ADC 8
12
12
12
12
+
1
4
1
4
+
15
D
Q
IOB

HIT BITS

-



VME FPGA IFACE:

Control Bus Memory Map for FADC FPGA

Name	Width (Bits)	Quantity	Access	Primary Address (Secondary Address)	Function
STATUS0	16	1	R	0x0000 (---)	Bits 14 to 0: Code Version Bit 15: 1= Command can be sent to AD9230
STATUS1	16	1	R	0x0001 (---)	TRIGGER NUMBER BIT 15 to 0
STATUS2	16	1	R	0x0002 (---)	Tbd. Read 0
CONFIG 1	16	1	R/W	0x0003 (---)	Bit 0-2 (process mode): 000 → Select Mode 0 001 → Select Mode 1 010 → Select Mode 2 011 → Select Mode 3 111 → Run Mode 0 then Mode 3 for each trigger Bit 3: 1:Run Bit 5-4 : Number of Pulses in Mode 1 and 2 Bit 7: Test Mode (play Back).
CONFIG 2			R/W	0x0004 (---)	When 1 ADC values = 0 Bit 0 → ADC 0 Bit 1 → ADC 1 Bit 2 → ADC 2 Bit 3 → ADC 3 Bit 4 → ADC 4 Bit 5 → ADC 5 Bit 6 → ADC 6 Bit 7 → ADC 7 Bit 8 → ADC 8 Bit 9 → ADC 9 Bit 10→ ADC 10 Bit 11→ ADC 11 Bit 12→ ADC 12

					Bit 13→ ADC 13 Bit 14→ ADC 14 Bit 15→ ADC 15
CONFIG 4	16	1		0x0005	7 => rising edge write to AD9230 ADC 6 => 1 write to all ADC 5 => 0 write to AD9230 1 read from AD9230 4 => 1 Reset ADC 3..0 => Select ADC to write to
CONFIG 5	16	1		0x0006	15..8 => Registers inside AD9230 7..0 => Data to write to register.
PTW	9	1	R/W	0x0007 (---)	Number of ADC sample to include in trigger window. PTW = Trigger Window (ns) * 250 MHz. **Minimum is 6**. **Always report Even Number. For odd PTW number, discard the last sample reported.**
PL	11	1		0x0008 (---)	Number of sample back from trigger point. PL = Trigger Window(ns) * 250MHz
NSB	12	1		0x0009 (---)	Number of sample before trigger point to include in data processing. This include the trigger Point. **Minimum is 2 in all mode.**
NSA	13	1		0x000A (---)	Number of sample after trigger point to include in data processing. **Minimum is (6 in mode 2)and (3 in mode 0 and 1). Number of sample report is 1 more for odd and 2**

					more for even NSA number.
TET	12	16		0x000B - 0x001A	Trigger Energy Thredhold.
PTW DAT BUF LAST ADR	12	1		0x001B	Last Address of the Secondary Buffer. See calculation below
PTW MAX BUF	8	1		0x001C	The maximum number of unprocessed PTW blocks that can be stored in Secondary Buffer. See Calculation below.
Test Wave Form	16	1		0x001D	Write to PPG. Read should immediately follow write.
ADC0 Pedestal Subtract	16	1	R/W	0x001E	Subtract from ADC0 Count before Summing
ADC1 Pedestal Subtract	16	1	R/W	0x001F	Subtract from ADC1 Count before Summing
ADC2 Pedestal Subtract	16	1	R/W	0x0020	Subtract from ADC2 Count before Summing
ADC3 Pedestal Subtract	16	1	R/W	0x0021	Subtract from ADC3 Count before Summing
ADC4 Pedestal Subtract	16	1	R/W	0x0022	Subtract from ADC4 Count before Summing
ADC5 Pedestal Subtract	16	1	R/W	0x0023	Subtract from ADC5 Count before Summing
ADC6 Pedestal Subtract	16	1	R/W	0x0024	Subtract from ADC6 Count before Summing
ADC7 Pedestal Subtract	16	1	R/W	0x0025	Subtract from ADC7 Count before Summing
ADC8 Pedestal Subtract	16	1	R/W	0x0026	Subtract from ADC8 Count before Summing
ADC9 Pedestal Subtract	16	1	R/W	0x0027	Subtract from ADC9 Count before Summing
ADC10 Pedestal Subtract	16	1	R/W	0x0028	Subtract from ADC10 Count before Summing
ADC11 Pedestal Subtract	16	1	R/W	0x0029	Subtract from ADC11 Count before Summing
ADC12 Pedestal Subtract	16	1	R/W	0x002A	Subtract from ADC12 Count before Summing
ADC13 Pedestal Subtract	16	1	R/W	0x002B	Subtract from ADC13 Count before Summing
ADC14 Pedestal Subtract	16	1	R/W	0x002C	Subtract from ADC14 Count before Summing
ADC15 Pedestal	16	1	R/W	0x002D	Subtract from ADC15

Subtract					Count before Summing
STATUS3	16	1	R	0x0400 (---)	00
CONFIG6	16	1	R/W	0x0401 (---)	00→ Table mode 10→ Window mode 01→ Boolean Overlap 11→ undefined Bit 2: 0→ T_HIT to Ctrl FPGA, Hitpattern to FIFO 1→ T_SUM to Ctrl FPGA, Sumpattern to FIFO Bitt3: 1→ select Hit Bit with programmable positive pulse width to P2. 0→ select Sum to P2 Bit4 0→ unable Table overlap and Trigger mode 1→ read back hit pattern selection table. Disable Table overlap and Trigger mode.
HITBITS_WIDTH	8	16	R/W	0x0402 (0 – 0x000F)	(7..0) Hit Bits One Shot Pulse Width. Actual width is one clk longer.
HITS_DLY	16	1	R/W	0x0403	Actual delay is 7 clock longer for all values. Exmple: 0→ 7, 1→ 8, 2→ 9 etc. Delay is from input of FX20.
Live Trig Out WIDTH	8	1	R/W	0x0404	Pulse width of LiveTrig Output. Actual width is 1 clock longer.
TRIGGER HITBITS	16	1	R/W	0x0405	In Window Mode. Select Hit Bit(s) that can

					activate(s) window. The Bit(s) that activate the Window is include in the Trigger Hit Pattern.
WINDOW WIDTH	16	1	R/W	0x0406	In Window Mode. Select the duration of window. Width is 2 clock longer.
BOOLEAN OVERLAP QUALIFIED BITS	16	1	R/W	0x0407	In Boolean Overlap Mode. Select Hit Bits to be active in this mode
HIT PATTERN SELECTION TABLE DATA	16	65536	R/W	0x0408	Write to 65536x1 Hit Pattern Selection Table. Each word contains data for 1 location. The address are auto-increment.
SUM/HITBIT External FIFO	16		R	0x0409	Read HITBITS
SUM Threshold	16	1	R/W	0x40A	Write SUM Threshold Register. T_SUM goes high when BSUM > register value

PTW MAX BUF = INT(2016 / (PTW + 8) * 250000000)
Where:
2016 → Number of address of Secondary Buffer
PTW → Trigger Window width in nano-second

PTW DAT BUF LAST ADR = PTW MAX BUF * (PTW + 6)- 1;
Where:
6 → 4 address for Time Stamp and 2 address for Trigger Number
NumberOfBytePerTrigger → PTW * 250 MHz.

A VME64x, 16-Channel, Pipelined 250 MSPS Flash ADC With Switched Serial (VXS) Extension

F.J. Barbosa, E. Jastrzembski, H. Dong, J. Wilson, C. Cuevas, D.J. Abbott
Thomas Jefferson National Accelerator Facility, Newport News, Virginia[1]

Abstract - We have designed a 250 MSPS pipelined flash ADC (Analog-to-Digital Converter) which will be employed at the Jefferson Lab's four experimental physics halls. This high speed and high density flash ADC conforms to VITA-41 VME64x switched serial (VXS) standard.

I. Introduction

A high speed and high density flash ADC module has been designed for use in nuclear physics experiments at Jefferson Lab. The Continuous Electron Beam Accelerator at Jefferson Lab currently delivers 6 GeV electrons to three experimental end stations. As part of a planned energy upgrade to 12 GeV electrons, a new experimental end station (Hall D) will be built and instrumented. A high speed and pipelined flash ADC will be used to provide information about the energy deposited on a detector element or group, timing, hit and trigger information.

The trigger information output from each of the modules on a VXS crate is routed via the backplane to a switch slot. A total of 18 flash ADCs per crate feed serial data to the switch slot at an aggregate rate of 6 Gb/s which is then sent to the experiment global VXS trigger crate.

Figure 1 shows some of the components of a typical VXS crate to be used at Jefferson Lab[2]: fADCs, an energy sum output switch card, a switch card for clock and timing distribution and a backplane.

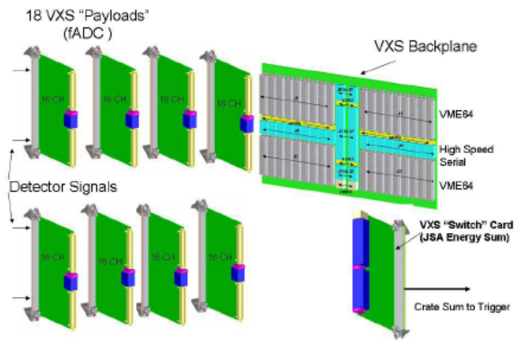


Figure 1: VXS Crate Components

II. The Flash ADC Module

The flash ADC module has been designed to accept 8-, 10- or 12-bit ADC chips. The choice of the chip will depend on the application resolution requirements and cost. The architecture of the fADC is shown in figure 2.

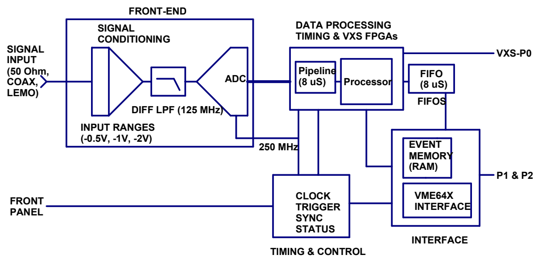


Figure2: Simplified fADC Architecture

There are three user-selectable ranges available for each of the inputs and differential signal conditioning scales the input signals to within the dynamic range of the ADCs. A single-pole low-pass filter limits the signal bandwidth to the Nyquist band of the converter or 125 MHz. Individual channel offsets are effected by means of DACs under VME control. For the 10-bit version of the flash ADC, the INL and DNL are expected to be ± 0.8 LSB and ± 0.5 LSB, respectively.

The digitized LVDS data and clock signals are then fed to a set of Virtex 4 LX25 FPGAs for data processing and temporary storage at the full 250 MSPS rate. ADC data is stored in RAM for event building and readout via VME. Additionally, these FPGAs output 16-bit energy sum words and channel hit information.

A Virtex 4 FX20 FPGA effects the board energy sum for output through the VXS connector via the RocketIO Multi-Gigabit Transceivers (MGT).

A fourth STRATIX II FPGA handles the control, trigger and interface functions. The VME 2eSST data transfer cycles can reach up to 320 MB/s.

The depth of the pipeline is 8 us and windowing and trigger latency are user programmable. In addition, sparcification, charge, pedestal and peak values may be obtained, as well as, hit information with user-defined thresholds and trigger processing.

In order to take full advantage of this high speed ADC, a differential PECL clock with jitter specification of less than 2 ps is timed properly across the various clock domains.

III. Results

The fADC prototype board is shown in figure 3. This 14-layer high density board employs fine pitch components on both sides and takes advantage of impedance matched micro-strip lines. The FPGAs' packages are of the BGA type and the ADC chips use QFN packages for improved thermal performance.

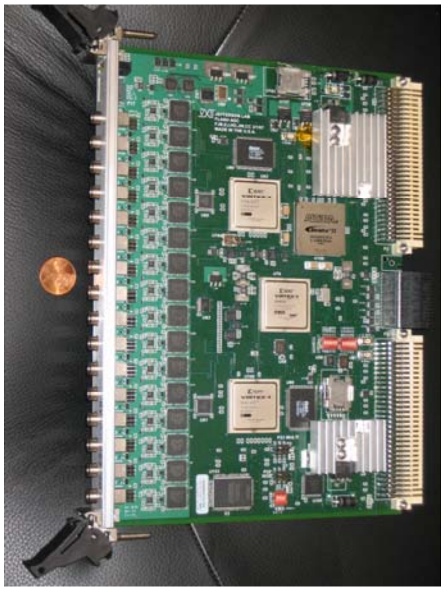


Figure 3: The 250 MSPS fADC

The 250 MHz differential PECL clock is shown in Figure 4. Its jitter was measured to be less than 2 ps.

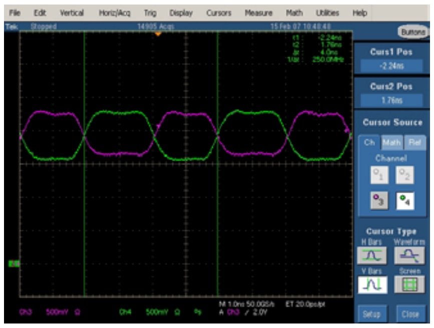


Figure 4: The 250 MHz Sampling Clock

The front end signal response to a 10 ns wide pulse is shown in figure 5. The pulses at the top of the picture are the differential inputs to the ADC chip after being conditioned and filtered by the input low pass filter; the pulse at the bottom is the difference of the above signals and represents what the ADC chip actually digitizes. The front end bandwidth was measured to be 110 MHz.

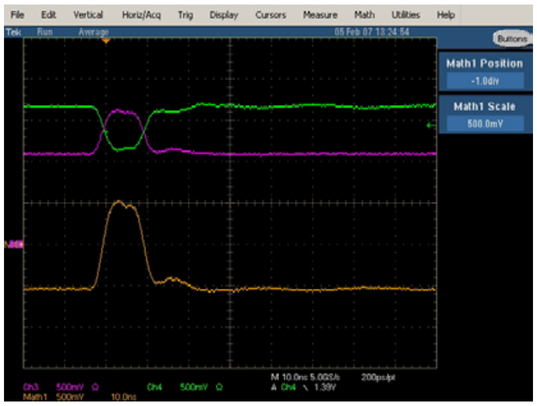


Figure 5: Front End Pulse Response

IV. Conclusion

We have designed and assembled a high density 250 MSPS flash ADC for use in physics experiments. This prototype, which is currently undergoing tests, will be used to further study the data transfer characteristics of the VXS standard conforming to VITA-41 and to implement algorithms in the study of detector signals for energy deposition, timing and triggering in realistic conditions expected in the experimental end stations at Jefferson Lab.

V. References

[1] H. Dong, C. Cuevas, D. Curry, E. Jastrzembski, F. Barbosa, J. Wilson, M. Taylor, " VXS Switch Card for High Density Data Acquisition System", IEEE SoutheastCon 2007, Richmond, VA, March 22-25, 2007.

VI. Acknowledgements

Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177. The U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce this manuscript for U.S. Government purposes.