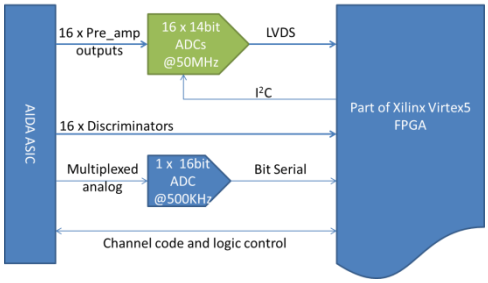
FEE64 VHDL documentation: VHDL file arrangement

1.0 How the VHDL modules fit together with the Mux analog and discriminator data path.

Signals are written in *italics* and VHDL files and modules are written as ***Bold_italics***

ASIC to FPGA. Figure 1



1.1 ASIC mux analog data to PPC memory

1.1.1 The signals from the ASICs connect to ***Read_out_DMA_GWR*** . For every rising edge of the DRDY signals or for every rising edge of a Discriminator or SYNC or Pause or Resume a timestamp plus activity pattern is stored in the *time_fifo*. The *time_fifo* is 512 locations of 136 bits with programmable empty at 200 and programmable full at 500.
At the rising edge of the ASIC clock (500kHz), following a delay of 50nS, the state of the DRDY signals are tested and held. There follows a programmable delay (set via the user interface) after which the Serial readout of the ADC connected to the ASIC multiplex output is started.

- The Serial ADC conversion is handled by ***ASIC_handle_rdo*** with one per ASIC. The ADC is set to convert and the channel number is registered. The state machine that handles the ADC uses a 125MHz clock. Thus the ADC serial data clock is 125MHz/2. The three ADC data signals pass through DDR registers. The data from the ADC conversion with the channel number are put into the *rdo_queue* fifo. This is 512 locations of 21 bits. With separate Write and Read clocks.

1.1.2 When a time pattern is detected at the output of the *time_fifo* the *readout* state machine tests the activity bits in the time pattern (one per ASIC DRDY, one per info and one per discriminator => 72 bits). For the ASIC DRDY the ADC data is collected from the relevant *rdo_queue* fifo and along with the timestamp stored in the time pattern is written to the *last_stage* fifo , 512 locations of 72 bits with programmable full at 400, programmable empty at 16 and separate Read and Write clocks. The Discriminator bits of the time pattern are tested in ASIC groups of 16. The time pattern is fully processed before the programmable full flag of the output fifo is tested.

1.1.3 The *last_stage_fifo* Read flags are monitored by the module ***mux_rdo_dma_GWR*** and when there are more than 16 entries it assembles the 72 bit data into 64 bits and writes it into the *Burst_fifo* in pairs to form 128 bit data for DMA into the PPC DDR memory. The DMA uses burst writes

of 16 128bit words. The PPC peripheral *master_drive_v1_00_b* monitors the flags of the *Burst_fifo* and writes a burst to the DDR memory using a local_link bus that is separate from the processor memory bus. The PPC sets the memory start address and the number of locations to use. Two memory blocks are used (half the buffer size each). The module keeps a track of the current address and adds 256 after each burst is completed. (16 x 128bit => 256 bytes). The PPC sets a high water mark (HWM). When a byte count passes this value a flag is set for the PPC and the second PPC DDR memory block is then filled, providing that it isn't waiting to be emptied. If no block is available the module loops checking and increments a 32 bit waiting counter.

1.2 What happens when data cannot flow to the PPC memory.

1.2.1 If the *time_fifo* reaches its programmable full (500) flag setting then a Pause info data item is written into the *time_fifo* so the time when the system stops is recorded. The pause signal is sent to the state machines that handle the ASIC mux readout, the discriminators and the correlation scalar.

1.2.2 When the *time_fifo* subsequently reaches its programmable empty flag setting (200) the pause is removed and resume is set for one clock allowing a resume data item to be written into the *time_fifo*. Thus recording the time when the system became available to collect data again.

1.2.3 During the period pause is active missed DRDY and correlation scalar activity is counted.

2.0 How the ASIC Pre-amp outputs are captured and transferred to PPC memory. (Waveforms)

2.1 Initialisation of the 14bit 50MHz ADCs

2.1.1 On power-up the first program to run, which eventually loads Linux, runs the firmware, ***Preamp_ADC_Dec11***, to calibrate the serial data link from the ADC to the FPGA. The state machines will finish with a status of 0xF7 for success. Any other status will be reported as a failure to calibrate. For further information about this data link check the ADC data sheet. The ADCs are calibrated again during SETUP. If this fails then individual FEEs can be calibrated from the FADC browser page.

2.2 Operation of the Waveform capture (***Wave_capture_and_store_no_dma)***.

2.2.1 The ADC digital output range is from 0 to 16383 with the zero point at 0x2000 or 8192. The input from the ASIC pre-amplifier is inverted once in the ASIC before buffering out to the ADCs. Thus a positive input will appear as a negative waveform to the ADC. Positive signals are seen as values from 8192 to 16383 and negative signals from 8192 to 0.

2.2.2 Providing a calibration has been successful, the ADC 14 bit data is stored in a 1024 location circular buffer, *cap_analyser*. Storage is controlled by a state machine, in ***cap_chan_led***, which handles the trigger, capture length and pre-trigger delay.

2.2.3 The trigger (***led_true.vhd***) is a level detector which looks for a rising or falling pass through a value. The value is set by the user and the rising or falling choice is set by the polarity choice by the user of the signal.

2.2.4 The flow through the trigger state machine can be described as follows :-

a) If *Pause* = 0 & *enable* = 1 then if *over_threshold* = 0 OR *force_capture* = 1 then goto b) Else loop

b) If *over_threshold* = 1 OR *force_capture* = 1 then set *my_trigger* and *led_busy* to 1 and goto c) Else loop

c) Set *my_trigger* to 0. If *led_re_arm* = 1 then goto d) Else loop

d) Set *led_busy* to 0. If *over_threshold* = 0 & *force_capture* = 0 then goto a). Else if *was_force* = 1 then goto e) else loop

e) If *force_cap* = 1 then loop Else goto a)

- *Over_threshold* = ((*polarity* = 0 & *adc_dout* > *threshold*) OR (*polarity* = 1 & *adc_dout* < *threshold*))
- *Pause* is from the stream readout state machine.
- *Enable* and *force_capture* are set by the user actions

2.2.5 Control of the circular buffer waits for the pre-trigger number of memory locations to pass. Then the trigger state machine is enabled and data is stored continuously until a trigger is detected. The number of locations to be filled is then calculated (*capture_size – pre_trigger*) and counted down. When the required number of post trigger locations are filled the storage stops and the waveform readout state machine is signalled.(*capt_ready*). When readout of the data is complete this is signalled by *readout_done.* The trigger state machine is re-enabled and the capture process restarts.

2.2.6 The ADC operates at 50MHz so storage occurs at 20 nS intervals. The user can select to store at a different rate by selecting the number of 50MHz clocks occur between storage.

2.2.7 Storage into the buffer is managed with the 50MHz ADC clock and read back from the buffer is managed with the system 100MHz clock.

2.3 Operation of the waveform readout to the Linux processor memory. (***q8_transfer_simple_GWR)***

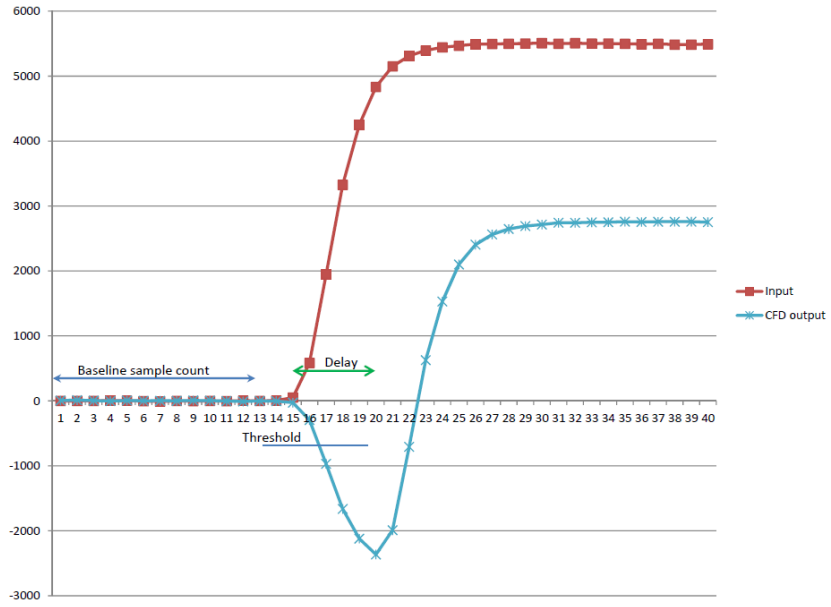
2.3.1 All 64 channels of waveform capture are readout using one state machine. The waveforms are preceded by a data item describing the content and a timestamp. Waveform data is forwarded to a DMA state machine as 64 bit words comprising four sequential 14 bit ADC words using the fifo *wcap_tfr_fifo*. The two unused MSBs of each 14 bit ADC word are set to 0.

2.3.2 Similarly to the method used in the ASIC mux data readout the trigger signals from each channel are written into a fifo as a pattern, when any one or more is true, along with the 64 bit timestamp and 3 bits for the SYNC/PAUSE/RESUME info data items. The fifo, *wcap_tfifo_GWR,* which is 512 x 115 with programmable full flag at 400 and programmable empty flag at 200.

2.3.3 When there is data in the *wcap_tfifo_GWR* then the pattern is readout, and the waveforms indicated are transferred with a suitable header to the DMA state machine using the *wcap_tfr_fifo.* When transfer of the waveform data

into the fifo is complete the channel is signalled with *readout_done* which causes the channel to re-enable its trigger and start to capture the next waveform.

2.3.4 As the data is written into the transfer fifo, *wcap_tfr_fifo*, it is also written to the maths unit, ***Wave_proc,*** for calculation of the Time Vernier value. The last two ADC values are ignored and replaced by the Time Vernier value and a status word to indicate success or otherwise of the maths unit. Refer to EDOC969 for further information.

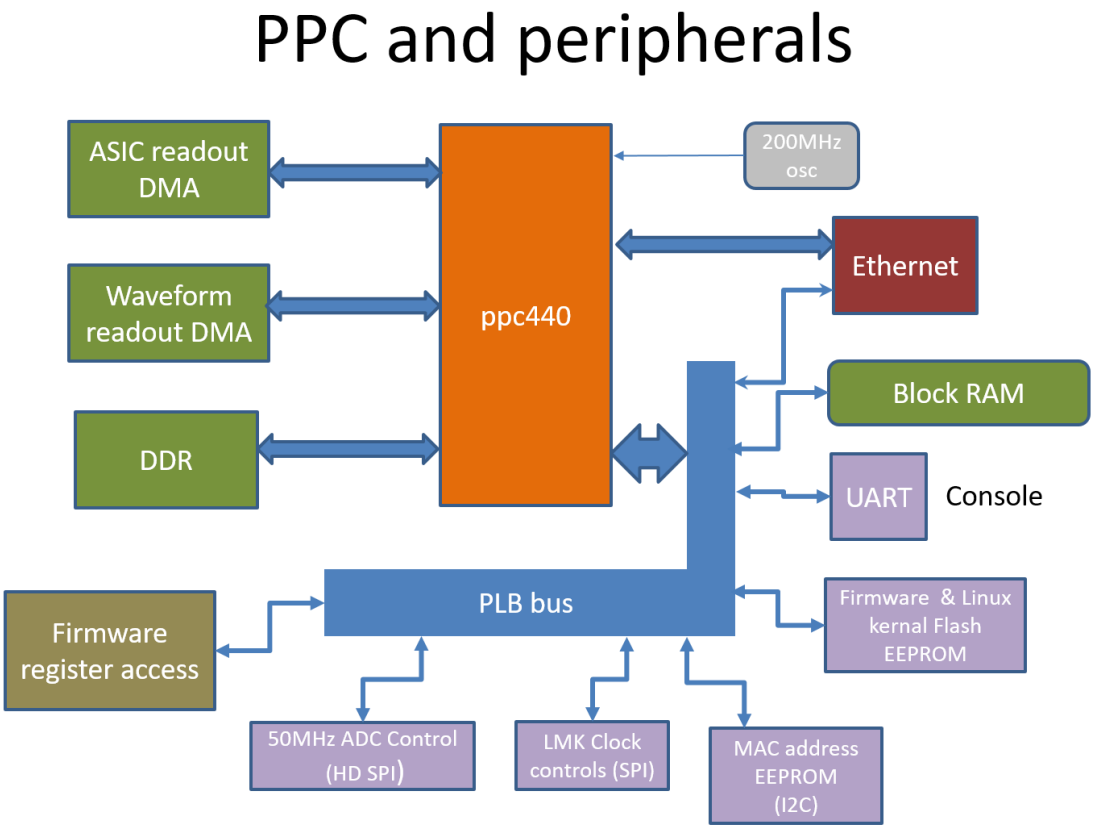


Extracted from EDOC969

2.3.5 For an explanation of the data transfer to Linux PPC memory see section 1.1.3. The mechanism is the same. The VHDL module involved is ***Wcap_dma***.

3.0 The Power PC and peripherals.

3.1 Overview of the ***system*** module



The PPC was created using the XPS ide and comprises the following parts

a) Two DMA channels
b) DDR memory controller
c) Firmware register access port
d) Hard Ethernet with DMA
e) Block RAM for Boot time program
f) UART for console interface
g) Flash EEPROM for firmware code and Linux Kernal
h) SPI for LMK3200 clock distribution controls
i) Wide SPI for 50MHz ADC control
j) I2C for the MAC address EEPROM
k) Master Clock Generator
l) Reset
m) Interrupt router
n) JTAG port

3.2 Discussion of each part.

3.2.1 DMA channel. The channel connects direct to the PPC 128bit data bus and operates in burst mode using cycle-steal technique so as not to affect the processor speed. The firmware fills a FIFO to contain a burst for transfer to the DDR memory. Software controls the addresses used, the high water mark and start/stop. The firmware operates the channel. The Xilinx provided module has been modified to improve performance for this instance.

3.2.2 DDR memory controller. This interfaces the DDR memory to the PPC, handles the refresh cycles and organises the addressing across the DRAMs using the row and column method. The peripheral is provided by Xilinx and has been programmed for the particular DDR devices in use. Entries in the .ucf file define some register positions, timedelays and pin allocations.

3.2.3 Firmware register access port. This is generated from a Xilinx EPC module. All that is required is to interface the data and address bus used for register access in the peripheral with the PPC PLB (peripheral bus) and handle the handshakes that allow transfer of data across the clock boundaries. The firmware uses 32 bit address and data. All transactions are acknowledged after a fixed delay in this module. There is no error reporting for bad addresses or slow response. The firmware is expected to provide valid data in the allocated time.

3.2.4 Hard Ethernet with DMA. This is part of the silicon in the Virtex5. The instantiation is handled by licenced code. This peripheral has DMA access to the DDR. The Ethernet is operated at 1GHz.

3.2.5 Block RAM for Boot time program. A pre-loaded RAM block situated at the top of the memory map. It contains the software to extract the Linux kernel from the flash EEPROM, decompress it to the DDR and start it. The Kernal then loads further code via the Ethernet.

3.2.6 UART for console interface. An RS-232 style interface which is used by the PPC to provide a console for the boot time messages and the Linux operating system.

3.2.7 Flash EEPROM for firmware code and Linux Kernal. The Xilinx Flash EEPROM is partitioned into four zones. Two are for firmware and two are for the Linux Kernal. Only one of each is used. Software can re-program this memory to allow firmware updates.

3.2.8 SPI for LMK3200 clock distribution controls. There are two PLL and clock distribution devices. LMK3200. They require setup after power-on and subsequently synchronisation to ensure the 50MHz clocks are operating from the same rising edge of the input clock.

3.2.9 Wide SPI for 50MHz ADC control. The eight devices each with eight channels are setup for the data readout serial communication via this link. The devices require 16bit data rather than the standard 8bit. This peripheral was altered from the Xilinx provided to accommodate this.

3.2.10 I2C for the MAC address EEPROM. This is a Xilinx provided peripheral with settings for the EEPROM used.

3.2.11 Master Clock Generator. This Xilinx provided utility generates the clocks required by the peripherals and the processor from the 200MHz crystal oscillator on the FEE64 board.

3.2.12 Reset. This Xilinx utility block handles the power-on reset sequence and internal resets if required.

3.2.13 Interrupt router. Xilinx utility block.

3.2.14 JTAG port.

3.2.15