The DSP 6202 processor board for ATLAS calorimeters

S. Böttcher, J. Parsons, S. Simion, W. Sippach
Columbia University / Nevis Labs

1 Introduction

The DSP processing unit (PU) is part of the ATLAS calorimeter read-out driver (ROD), responsible for receiving the digitized samples from the FEB, processing these data, and making the processed data available to the DAQ [1]. The data processing includes the linear filtering of the digitized samples and computation of the energy, time, and pulse shape quality for all channels. The PU is also responsible for data monitoring for every channel.

This note describes the PU prototype realized with a TMS C6202 CPU. A general description of the board is given, followed by more detailed information on how this board can be operated (hardware guide). The software issues, including how the physics algorithms are implemented on this particular board, are not discussed here. Instead, these will be addressed in a separate document.

2 General architecture

The PU (Figure 1) is organized around a 250 MHz TMS320C6202 DSP with 256K Bytes of internal program memory and 128K Bytes of data memory. Input data are provided through the input FPGA in charge of receiving the FEB data and TTC information, data format consistency checking, and some data rearrangement. The events are then buffered into a dual-port memory, available to the DSP as read-only external asynchronous RAM. This dual-port RAM also serves for initializing the DSP internal program and data memory, at reset. After event processing, the DSP writes the output data in the final format, to the output FIFO, ready to be read by the ROD motherboard.

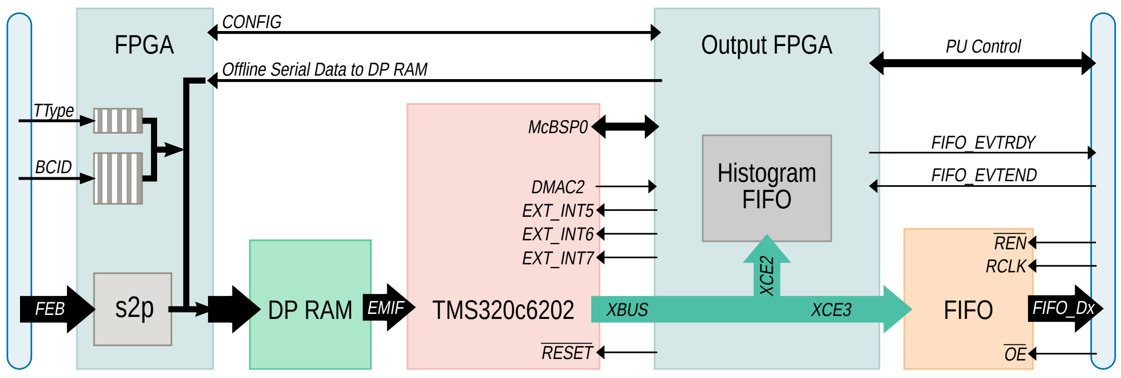

Figure 1: C6202 DSP block diagram.

The board is controlled by the output FPGA, which implements the PU interfacing to the ROD motherboard. The output FPGA is responsible for booting the PU board by configuring the input FPGA, then loading the DSP code. It also provides a second FIFO (for histogramming purposes) which can be written to by the DSP, and read from by the ROD motherboard. Finally, the output FPGA provides a two-way serial communication with the DSP, and a few other control signals.

Table 1: PU Board Specifications

CPU Type	TMS320C6202 @ 250 MHz
Dual-port RAM Type	CYC7C057V-12 ns 32K × 36 (used as 32K × 32)
Input FPGA	ALTERA FLEX 10K30E
Output FPGA	ALTERA FLEX 10K30E
FIFO Type	IDT72V3680 16K × 36 (used as 16K × 32)
FIFO almost-full offset	1023 words (hardware setting)

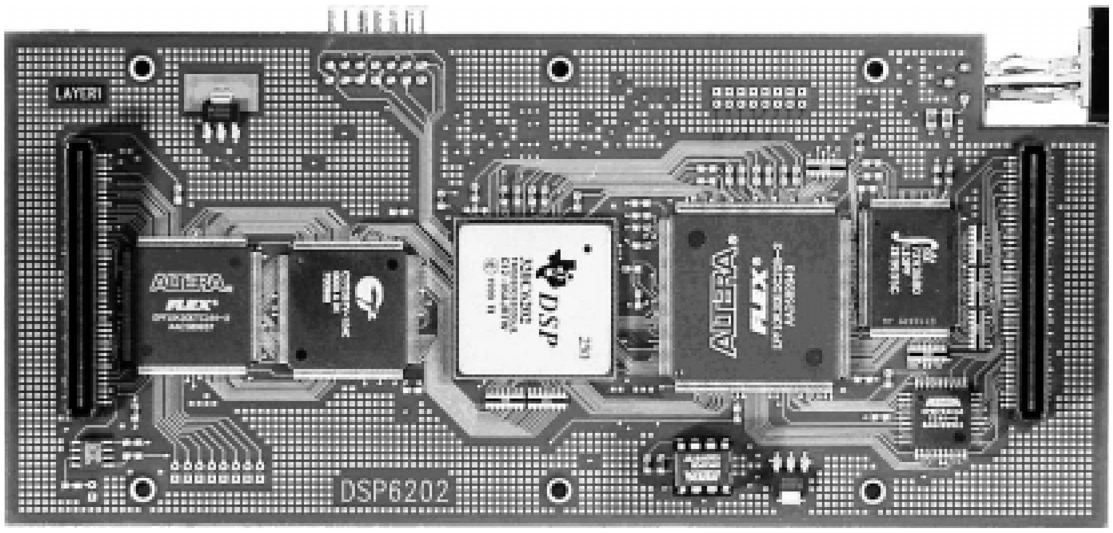

Figure 2: C6202 DSP board (component side, heatsink not shown).

3 6202 Peripherals

This section describes only those peripherals which are relevant for the prototype board.

3.1 External RAM

The dual-port memory is the only device connected to the External Memory Interface (EMIF). The right port of this device is available to the DSP as read-only asynchronous RAM, and can be accessed directly via LOADs or by DMA transfers. Note that all four CEn spaces will in fact address the same physical device. The EMIF timings are normally configured (immediately or soon after boot) to the fastest settings supported by the dual-port RAM.

Please refer to Section 10.3 of the Peripherals Guide [5], describing the memory map. On this board, the DSP boots in MAP1 mode.

3.2 Expansion Bus

The expansion bus (XBus) is configured in synchronous FIFO mode. It provides the high throughput necessary for event output, as well as for histogramming and for test purposes. The XBus can only be written by DMA transfers; it is not possible to use STOREs. The XCE3 space corresponds to the event output FIFO, while XCE2 corresponds to a second FIFO inside the output FPGA. In addition to the XBus signals, the following synchronization signals can be used:

- The rising edge of DMAC2 informs that an event has been written to the XCE3 space.
- INT6 is the almost-full flag $\overline{\text{PAF}}$ from the event output FIFO. The DMA transfers must not exceed 1024 words if this signal is used for synchronization.
- INT5 is the half-full flag from the FPGA histogram FIFO. This flag is actually bit 8 of the histogram FIFO wordcount. Multiframe DMA transfers must be used for transfers exceeding 256 words.

3.2.1 XBus and DMA configuration for writing to the event FIFO

Before writing events to the output FIFO, a few registers must be set up appropriately, as shown in Table 2. These are the XBus global control register, DMA primary control register, DMA secondary control register, and DMA destination address register. In addition, the DMA source address register must be set up before each transfer. The DMA channel 2 should in principle be used for writing out events, since the DMAC2 signal increments the FPGA event counter.

Table 2: Software configuration of the XBus and DMA for event output.

Register	Address	Set value	Notes
XBus global control (XBGC)	0188 0000 h	6000 h	Clock rate = CPU / 4 (XFRAT=2)
DMA destination address	0184 001c h	7000 0000h	XCE3 address.
DMA source address	0184 0014 h	as needed	
DMA transfer count	0184 0024 h	as needed	32-bit wordcount
DMA secondary control	0184 000c h	001c 0008 h	DMAC reflects FRAME COND (DMAC=4) . Use level-triggered frame synchronization (FSIG=1, RSPOL=1).
DMA primary control	0184 0004 h	0401 8011h	Start the transfer (START=1) with frame synchronization (FS=1) on EXT_INT6 (RSYNC=6). Do not increment the destination address (DST DIR=0)

Using these settings, the transfer does not proceed if the event FIFO is almost full (less than 1024 words free). You must not issue a second DMA transfer before the previous transfer has completed. This can be checked by reading the STATUS field of the DMA primary control register. For details, please refer to Chapters 5 and 8 of the Peripherals Guide [5], and to the DMA Performance Application Report [7].

3.2.2 DMA configuration for writing to the histogram FIFO

To set up a multiframe DMA transfer, please follow the indications in Table 3 below. Frames are synchronized using the half-full flag of the FIFO, available as EXT_INT5. In this example, we assume that the DMA channel 3 is used.

Table 3: Software configuration of the DMA for histogram output.

Register	Address	Set value	Notes
DMA destination address	0184 005c h	6000 0000h	XCE2 address.
DMA source address	0184 0054 h	as needed	
DMA transfer count	0184 0064 h	as needed	Wordcount for the first frame (must not exceed 256) and number of frames.
DMA global counter reload	0184 0028 h	100h	Wordcount for all subsequent frames.
DMA secondary control	0184 004c h	0018 0008 h	Use level-triggered frame synchronization (FSIG=1, RSPOL=1).
DMA primary control	0184 0044 h	0401 4011h	Start the transfer (START=1) with frame synchronization (FS=1) on EXT_INT5 (RSYNC=5). Do not increment the destination address (DST DIR=0)

3.3 Serial Port and other control signals

The C6202 has three Multichannel Buffered Serial Ports (McBSPs), of which McBSP0 is used for two-way communication with the output FPGA. To ensure compatibility with the FPGA, this port shall be configured, in that order, as described in Table 4.

Table 4: Software configuration by the DSP of the McBSP0 serial port

Register	Address	Set value	Notes
Receive control (RCR)	018c 000c h	0001 00a0 h	XDATDLY = 1
Transmit control (XCR)	018c 0010 h	0001 00a0 h	
Sample rate generator (SRGR)	018c 0014 h	2020 0008 h	Clock rate = CPU / 6
Pin control (PCR)	018c 0024 h	0000 0a02 h	Change the polarity of CLKX0 so that data are driven by the falling edge of the clock.
Serial port control (SPCR)	018c 0008 h	00c1 0001 h	Enable transmitter and receiver.

The two other serial ports are used as general control signals: DX1, FSR1, FSX1, DX2, FSR2, FSX2, CLKR2. Each one of these pins is connected to the output FPGA controller, and therefore **the assignment of these signals as inputs or outputs must match the configuration of the output FPGA**.

The present hardware configuration of these signals is described in Table 5.

Please refer to Chapter 11 of the Peripherals Guide [5] for what concerns the software configuration of the DSP serial ports. In particular, refer to Table 11-22 in the above document for what concerns the McBSP configuration for general I/O. Note that if you do not wish to use these signals, their default configuration at reset does not conflict with the hardware assignment.

Do not configure the FSR1, FSR2, or FSX2 as DSP outputs!

Table 5: Configuration of McBSP1 and McBSP2 signals in the FPGA controller

Signal	Driven by	Description
CLKR2	DSP	This is the BUSY flag to the LVL1 trigger.
DX1	DSP	INIT signal to the input FPGA
FSX1	DSP	Copied to Test Point 3 of the FPGA controller (scope debugging).
FSR1	FPGA	General-purpose signal set by VME via the PU control register.
FSR2	FPGA	General-purpose signal set by VME via the PU control register.
FSX2	FPGA	Reserved for future use
DX2	Not connected	Reserved for future use

4 Board power-up and initialization sequence

The output FPGA controller is configured from the on-board PROM. The PU initialization must be completed via VME, by configuring the input FPGA and loading the DSP code.

During and after power-up, the output FPGA maintains the DSP $\overline{\text{RESET}}$ signal asserted.

The complete initialization sequence is the following:

1. For a warm reinitialization, the DSP $\overline{\text{RESET}}$ signal should first be asserted.
2. The input FPGA is configured as specified in the ALTERA application note.
3. The input FPGA is put in *Offline* mode (as opposed to *Event processing* mode). In this way, the dual-port RAM can be initialized via VME.
4. Load the output FPGA address register with the absolute address in the dual-port RAM where the data is to be stored.
5. Send the data as 32-bit words by writing to the corresponding register in the output FPGA. The address register is incremented automatically.
6. Go back to step 4 to initialize another block of memory, if necessary.
7. The DSP $\overline{\text{RESET}}$ signal is deasserted. The DSP now initiates its ROM boot process. The program located in the dual-port RAM is copied to address 0 by the DMA controller. This is done with a single-frame block transfer of 64K bytes. After completion of the transfer, the DSP starts executing from address 0.
8. If the DSP code and data needed for event processing occupy more than 64K bytes, a second phase of the boot process is needed. Subsequent transfers will be initiated by the DSP (by software) to download the data or code from the upper 64K of the dual-port RAM.
9. If the DSP code and data occupy more than the 128K bytes of the dual-port RAM, a third phase of the boot process is needed, involving some VME handshaking via the

DSP serial port or via the general control signals: The host CPU refills the dual-port RAM and notifies the DSP that more data is available. The DSP then copies the data from the dual-port RAM to its internal memory and acknowledges the transfer.

5 The VME interface

The PU interface to the ROD motherboard is implemented by the output FPGA controller. Its main functionalities are described below.

5.1 Configuration of the input FPGA

The output FPGA controller drives the nCONFIG, DATA0 and DCLK pins and monitors the nSTATUS and CONF_DONE pins of the input FPGA.

5.2 Offline data path to the input FPGA and the DP RAM

The output FPGA controller drives the Offline signal to the input FPGA. When this signal is high, data can be sent via VME to the input FPGA or to the dual-port RAM. This *Offline* mode effectively inhibits FEB and TTC data processing by the input FPGA.

5.3 C6202 interface

The output FPGA controller drives the DSP $\overline{\text{RESET}}$ pin. This low-active signal is asserted at power-up, and can be controlled via VME.

The output FPGA controller provides a two-way interface to the DSP via the McBSP0 port. Unfortunately, currently there is no buffering at the FPGA receive end; therefore the DSP must not send any new data until the previous data word has been read via VME. Recent data always overwrite previous data at the FPGA receive end.

The output FPGA controller monitors the general-purpose DX1 and FSX1 DSP signals.

5.4 FIFO interface

The output FPGA controller receives data from the XCE2 space, into the internal histogram FIFO. This FIFO is then read out via VME, through the PU 8-bit wide data bus. The internal histogram FIFO word count can be read via VME as the `f_bwc` field of the status register.

Data from the XCE3 space is written directly into the external event FIFO. The event FIFO is read out directly via the 32-bit wide event data bus.

The `CW_nRSTBUF` and `CW_nRSTFIFO` bits of the FPGA control register must be set for the corresponding data path to be enabled. Clearing the `nRSTFIFO` bit resets the external event FIFO. Clearing the `nRSTBUF` bit resets the internal histogram FIFO.

The output FPGA controller monitors the event FIFO empty flag $\overline{\text{EF}}$ and the event FIFO almost-full flag $\overline{\text{PAF}}$. The $\overline{\text{PAF}}$ flag is also connected to the INT6 pin of the DSP.

5.5 Event counter

The ouput FPGA controller implements the Event-Ready logic via a counter. The rising edge of DMAC1 increments the event counter, while the OC_EvtEnd signal decrements the counter. The output signal FIFO_EvtRdy is active when the event counter is non-zero.

A software pulse sent via the control register, can be used to reset both the event FIFO and the counter. The event counter can be read via VME as the `f_nevt` field of the status register.

Table 6: PU VME Register Mapping

PU_Add[4..0]		R/W	Function
0	TI_CONTROL	RW	Control Register
1	TI_SWRITE	W	Serial word to McBSP0
2	TI_DP_ADDR	W	DP RAM Address Register
3	TI_DP_DATA	W	DP RAM Data Register
4	TI_ALTERA_CONFIG_DATA	W	The least significant 8 bits are sent as configuration data to the input FPGA
6	TI_STATUS	R	Status Register
7	TI_HIST_DATA	R	Data from histogram FIFO
8	TI_SREAD	R	Serial word from McBSP0
9	TI_RESET	W	Software PU reset (dataless function)

Table 7: PU Control Register: RW, VME address 0x0 or TI_CONTROL in dsp.h

Bit	Mnemonic	Function
0	CW_nRSTDSP	This bit controls the state of the DSP $\overline{\text{RESET}}$ pin.
1	CW_nRSTFIFO	Setting this bit enables data to the XCE3 space to be written to the external event FIFO.
2	CW_nRSTBUF	Setting this bit enables data to the XCE2 space to be written to the histogram FIFO.
3	CW_OFFLINE	Offline flag to the input FPGA.
4	CW_nCONFIG	This bit controls the configuration pin of the input FPGA.
5	CW_FSR1	This bit controls the state of the DSP FSR1 and EXT_INT7 pins.
6	CW_FSR2	This bit controls the state of the DSP FSR2 pin.

Table 8: PU Status Register: RO, VME address 0x06 or TI_STATUS in ti_dsp.h

Bit	Mnemonic	Function
0..8	f_bwc	Histogram FIFO wordcount.
9	f_bff	Histogram FIFO Full flag.
10	f_bef	Histogram FIFO Empty flag.

Table 8: PU Status Register: RO, VME address 0x06 or TI_STATUS in ti_dsp.h

11	f_fifoff	Event FIFO Full flag (the opposite of FIFO $\overline{FF}$ flag).
12	f_fifoaf	Event FIFO almost-full flag (the opposite of FIFO $\overline{PAF}$ flag).
13	f_fifoef	Event FIFO Empty flag (the opposite of FIFO $\overline{EF}$ flag).
14	f_srdy	This bit is set if serial data from McBSP0 is available.
15	f_sovf	This bit is set when the serial port buffer in the output FPGA is overwritten by new data from McBSP0.
16..23	f_nevt	Output FPGA event counter.
24	f_dspbsy	State of the CLKR2 pin (general-purpose DSP output / BUSY flag to LVL1)
25	f_dx1	State of the DSP DX1 pin (general-purpose DSP output)
26	f_fsx1	State of the DSP FSX1 pin (general-purpose DSP output)
27..29	f_rsvd	Reserved for future use
30	f_nstatus	State of the input FPGA configuration pin.
31	f_conf_done	State of the input FPGA configuration pin.

6 The input FPGA data interface

During normal event processing, the input FPGA receives the FEB data and the TTC trigger information. The data are then formatted and written to the dual-port RAM. In *Offline* mode, the input FPGA receives 32-bit serial data from the output FPGA controller, and writes the data to the dual-port RAM (if A15 is 0) or to its internal *data-formatting table* (if A15 is 1).

6.1 The dual-port RAM map during event processing

The 128K Bytes of the dual-port RAM are divided into 128 contiguous event blocks of 1K Bytes. It is therefore assumed that the complete formatted data from one event (64 channels) holds entirely within one 1K Byte event block. The input FPGA Altera code may be modified to accommodate a different block size, if necessary.

The last word of the dual-port RAM (address 0x7FFF) contains the TTC event block counter and the FEB event block counter. The TTC counter occupies the upper 16 bits and the FEB counter occupies the lower 16 bits.

The DSP must **poll** on these counters when waiting for incoming data; no other signal is delivered to the DSP upon reception of a TTC trigger or FEB event.

6.2 The TTC interface

The prompt BCID information, and the delayed Trigger Type, are decoded and buffered separately into two FIFOs. When the TTC information is complete (i.e. the Trigger Type FIFO is not empty), both FIFOs are read, and the 32-bit TTC word is written to the dual-port RAM, at offset 0xFE inside the corresponding event block. The TTC event counter (i.e. block counter) is then incremented.

Currently the input FPGA does not check the TTC BCID and the FEB BCID for equality. This can be done by the DSP.

6.3 The FEB interface

The FEB data are de-serialized. Reception of the FEB event header (BEVT) triggers the decoding of the FEB event. The FEB control words CTL1, CTL2, and CTL3, and the capacitor addresses RADDn, are written to the dual-port RAM without any modification; while the ADC values and the gain bits are written separately. The writing takes place at the offsets prescribed in the *data-formatting table*.

All data words (except BEVT and EEVT) are checked for odd parity. If one or more errors are detected, corresponding flags are set in the error status word.

The gain bits for a given channel must preserve their value for all samples during an event[1]. Otherwise, a flag is set in the error status word.

6.4 Configurable data format

Since the input event buffering uses a dual-port RAM, the format of the FEB data to the DSP is configurable to a large extent. This flexibility is useful for a fully efficient DSP code.

The different types of information are described in the table below. The data are always packed in 32-bit words. Each one of the words in the left column can be written or not to the dual-port RAM. If a word is written, the RAM write address is also user-defined, via a data-formatting table which is downloaded to the input FPGA.

Table 9: Input FPGA intermediate format building blocks

Word	Bits 31…16 refer to:				Bits 15…0 refer to:			
CTL1 (4)	channels i+39 … i+32				channels i+7 … i			
CTL2 (4)								
CTL3 (4)								
RADD (4, NSAMP)								
ADC (32, NSAMP)	channel i+32				channel i			
GAIN (4, NSAMP)	ch39	…	ch33	ch32	ch7	…	ch1	ch0
ERR_FLAG	Input FPGA error status word.							

The software library provides an easy way to configure the event format in the dual-port RAM using the data-formatting table.

[1] If more than one gain are read out, the gain bits for each gain must preserve their value across all samples.

6.5 The input FPGA error status word

The input FPGA detects various errors while receiving FEB data. These errors cause the corresponding bits to be set in the event status word, as shown in Table 10.

Table 10: Input FPGA event error status word

Bit	Error condition
0	BEVT (0xffff) missing in any of the eight groups of channels.
1	CTL1 mismatch
2	CTL2 mismatch
3	RADD mismatch
4	Gain mismatch: the gain bits are not preserved across the time samples, for some channel(s).
5	CTL3 mismatch
6,7	Parity error count: 0=no error, 1=one error, 2=two errors, 3=more than two errors.

7 Power consumption

The C6202 board consumes 4 W when idle, and 5.8 W during event processing. Event processing includes: computation of energy, time, and χ^2 for 64 channels at 200 kHz LVL1 trigger rate, and the associated I/O; and filling of energy and time histograms for each channel, for all events.

8 Upgrade path

The processor board described in this document is beleived to be able to provide the CPU and I/O bandwidth necessary to process and monitor 64 channels at 100 kHz trigger rate. However, some straightforward steps can be taken to boost the performance of this board, which may be useful for flexibility (e.g. if the DSP code can written in C instead of assembler) or, alternatively, for doubling the number of channels which can be processed.

Therefore, the following changes are considered:

- Replacement of the dual-port RAM with a 10 ns version (instead of 12 ns), which would allow to operate safely at 1/3 of the CPU clock speed (12 ns access time), thereby reducing the input transfer time by 20%.
- Replacement of the output event FIFO with a 7.5 ns version (instead of 10 ns), which would allow to operate at 1/2 (instead of 1/4) of the CPU clock speed.
- Replacement of the C6202 processor with a C6203, which runs at 300 MHz and has 512K Bytes of data memory.

9 References

1 ATLAS Liquid Argon Calorimeter TDR, CERN/LHCC 96-41.

2 Liquid Argon Calorimeters Read-Out Drivers web page http://atlasinfo.cern.ch/Atlas/GROUPS/LIQARGON/ROD/largrod.html

3 TMS320C62X Family Application Notes http://www.ti.com/sc/docs/apps/dsp/tms320c62x.html

4 TMS320C6000 CPU and Instruction Set Reference Guide SPRU189E

5 TMS320C6000 Peripherals Reference Guide SPRU190C

6 TMS320C620X/TMS3206701 DMA and CPU: Data Access Performance SPRA614

7 TMS320C6000 Expansion Bus to External Synchronous FIFO Interface SPRA547

8 TMS320C6000 Technical Brief SPRU197D

9 TMS320C6202 Data Sheet SPRS104a

10 TMS320C6000 Optimizing C Compiler User's Guide SPRU187G

11 TMS320C6000 Programmer's Guide SPRU198D

12 TMS320C6000 Tools: Vector Table and Boot ROM Creation SPRA544A

13 TMS320C6000 Assembly Language Tools User's Guide SPRU186G