CDF/DOC/TRIGGER/4340
Version 5.2
December 12, 1997

Specifications and Design of the Level 1 Calorimeter Trigger PreFRED Modules for the Run II Upgrade.

Alexis Amadon, Mircea Bogdan, Peter Wilson
University of Chicago

1 Introduction

The Level 1 (L1) Calorimeter PreFRED sub-system is part of the Global L1 decision crate[1]. It consists of 2 printed circuit boards or modules whose name and functions are the following[2] :

1. **SUMET**: sums up the energy deposited in all 24 calorimeter wedges and apply the ΣE_t and $\not\!\!E_t$ trigger thresholds

2. **TOWTRG**: summarize the tower-over-threshold triggers

The trigger bits issued from these two modules are sent out simultaneously to FRED[3] every 132 ns. In this note, we specify the I/O signals of these modules, and present a detailed design of the PC board. An integrated solution allows us to use the same board design for the two modules. The top level diagram of the board is shown in Figure 1. **Custom_P3** represents the P3 backplane through which all the input signals from the CRATESUMs[7] and the output trigger bits to FRED pass. **PreFRED_logic** is where the functions specific to the PreFRED board are processed. It represents the core of the design. **DAQ_Interface** is the interface with the Data Acquisition System and Level 2. This is where the output data is temporarily stored while waiting for L1 and L2 accept/reject signals. **VME_Interface** is the interface with VME. It deals with the low-level signals required for 32-bit word access and block transfer. **Front_Panel** encompasses the connection to L2 and the set of LEDs used on the front panel. **Control** deals with VME address decoding, clock and strobe signal generation both for VME transactions and run operations, bunch counting, and error handling.

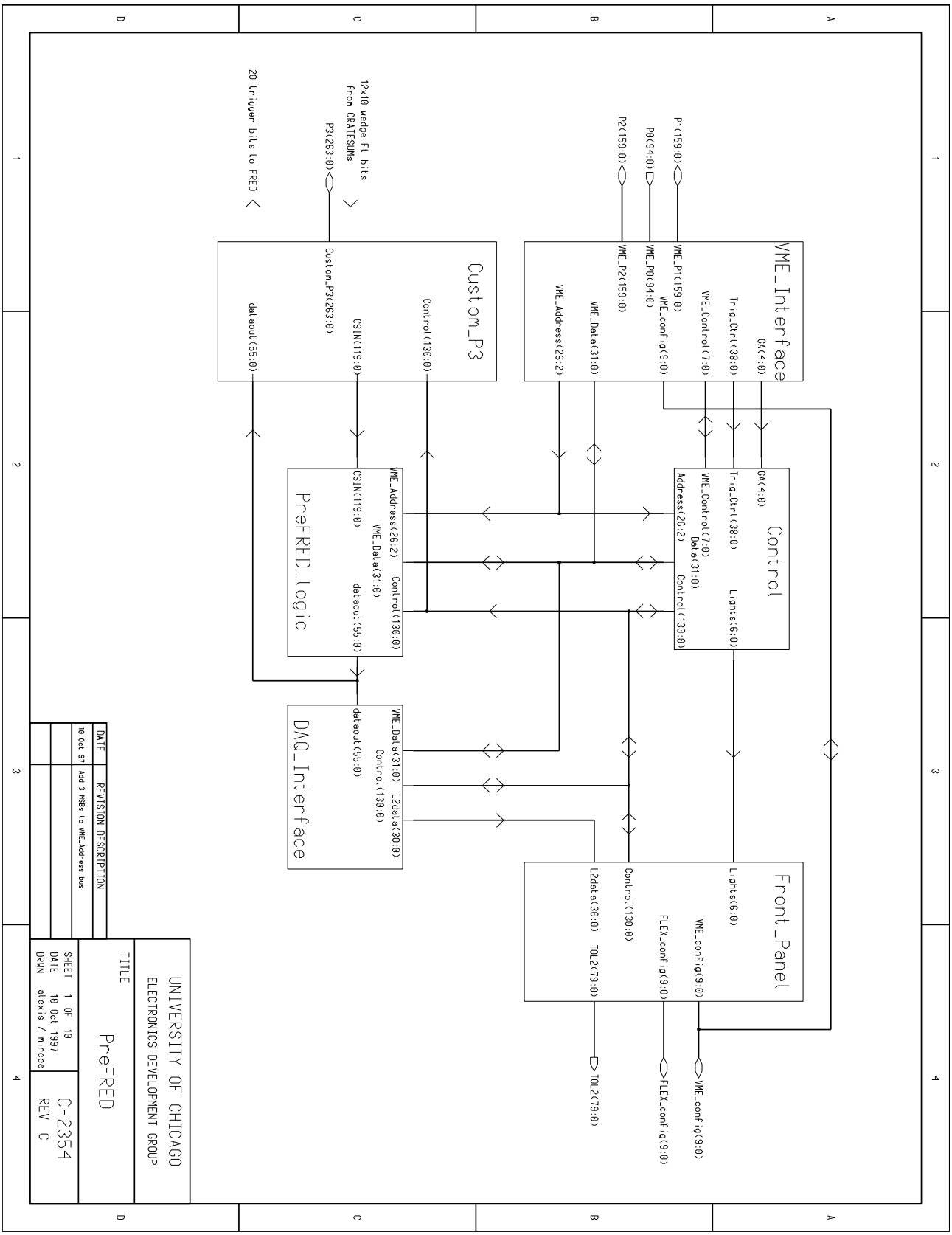


Figure 1: Top level diagram of the common board used for the PreFRED modules. See text for a brief explanation of the functional boxes. See appendix for a schematic breakdown of each one of them.

2 SUMET

2.1 Specifications

The SUMET Module receives 2 10-bit transverse energy sum words ($E_t(\phi)$) from each of the 12 CRATESUM boards. Each word is associated with a ϕ-wedge and has a 0-512 GeV range with a 0.5 GeV least count. A total of 240 differential signals are brought in with 12 cables.

The output to FRED consists of 4 bits representing the outcome of 4 programmable thresholds applied to ΣE_t or $\not\!\!E_t$ (typically 2 thresholds for $\not\!\!E_t$ and 2 thresholds for ΣE_t, reconfigurable to 3 or 4 thresholds for $\not\!\!E_t$ and 1 or 0 threshold for ΣE_t). The trigger bits are sent out to the J3 backplane of the crate, so the FRED module[3] can quickly access them. They are accompanied by an active high Bunch 0 tag bit for synchronization diagnosis purposes.

The PreFRED thresholds will have to span a certain range and granularity. A minimum number of bits are thus required for ΣE_t and $\not\!\!E_t$ before the thresholds are applied. The dynamic range proposed for $\not\!\!E_t$ and ΣE_t are 0-256 GeV and 0-2048 GeV, respectively, with a 1 GeV least count for both.

The output to FRED should be sent in phase with that of the other PreFRED Modules. The latency for the data throughput should not exceed 3 CDF clock cycles, excluding any output alignment waiting time.

ΣE_t, $\Sigma E_x = \Sigma E_t \cos\phi$ and $\Sigma E_y = \Sigma E_t \sin\phi$ are sent to L2 in case of a L1 Accept (L1A). In case of a L2 Accept (L2A), these words are also to be sent to DAQ along with a bunch counter value[4], the 4 trigger bits, $\not\!\!E_t^2$, and the ΣE_t word from the previous **CDF_clk** cycle. The latter is needed in offline analysis in order to detect splash events that leave unwanted transverse energies in the next crossing. These words will first have to feed a FIFO until a L1A or L1R (Reject) signal is received from the Trigger Supervisor Interface (TSI) for the corresponding event. On a L1A, as for the other modules, TSI transfers a common 2-bit buffer address which tells which of 4 **DAQ buffers** the data should be directed to on the board. On a L1R, the data is read out of the FIFO without being transferred anywhere. Subsequently, if L2 rejects the event, no signal is sent, and the DAQ buffer can be overwritten on a subsequent L1A. If L2 accepts the event, the contents of that buffer are read out over VME using block transfers by the DAQ.

Should one or more $E_t(\phi)$ input words be saturated (all bits set to 1), the SUMET output words should also be saturated.

2.2 General design

2.2.1 The re-use of the L1 DIRAC Receiver Aux-Card and backplane

As mentioned earlier, 240 differential signals have to be fed into the SUMET board every 132 ns (note that this is the same as for the TOWTRG module). For these, we use the input connecting system developed for the DIRAC board[5]. The CRATESUM data will come in two stages within a CDF clock cycle: since each CRATESUM has to send E_t for two adjacent detector wedges, it seems natural to have it done in two sequential steps of 66 ns, on the same input data lines. A total of 12 even wedge E_t's are sent in the first phase, and the 12 odd wedge E_t's are sent in the second phase. The definition of even and odd wedges is arbitrary, and for reasons that will become clear in the next paragraph, Figure 2 shows the wedge indexing chosen to perform the summing algorithm. This scheme reduces the number

of incoming wires by a factor of two and enables the use of 12 20-pin connectors instead of 12 high-density 40-pin ones. We use the Aux-cards[6] developed for DIRAC to transform the differential input data signals into individual bits directly usable by the SUMET logic.

2.2.2 Wedge E_t summing, $\not\!\!E_t^2$ processing

The operations to be performed by the SUMET module are the sums ΣE_t, $\Sigma E_x = \Sigma E_t \cos\phi$, $\Sigma E_y = \Sigma E_t \sin\phi$, $\not\!\!E_t^2 = (\Sigma E_x)^2 + (\Sigma E_y)^2$, and the comparison of ΣE_t and $\not\!\!E_t^2$ with their trigger thresholds.

ΣE_t, ΣE_x, and ΣE_y are performed in parallel. Before being added, the E_t's have to be weighted by the cosine and sine of their wedge azimuthal angle in the case of ΣE_x and ΣE_y. However, to minimize the number of weighting operations, the associated multiplications are done after summing the sets of four wedges which share the same weighting factor in absolute value. Let us use the notation i_j to address the jth (even (0) or odd (1)) wedge in sector i, following Figure 2's definition. For instance, the E_t's from wedges 0_0, 5_0, 6_0 and 11_0 will first be added/subtracted together, then their signed sum will be multiplied by the cosine/sine of the 0_0 wedge angle before being added further with the other weighted partial sums. The individual steps of that process are detailed in Figure 3: even and odd wedge data are pipelined in the stage prior to phi-weighting; the incoming E_t's are added or subtracted depending on whether the result will feed a cosine or a sine multiplication. The equality $\cos\phi = \sin(\frac{\pi}{2} - \phi)$ is taken advantage of such that only 6 different weighting factors are needed for the 24 wedges. Immediately before the ϕ-weighting, multiplexers are used to switch to even or odd wedges depending on the clock phase. Once the phi-weighting is performed for both even and odd wedges, the resulting signed partial sums are simply added together until ΣE_x and ΣE_y are yielded. Meanwhile, ΣE_t is computed in parallel, with a cascade of unsigned adders which starts with the top six of the first column of adders in the diagram (the subsequent adders in the ΣE_t chain have been omitted from the diagram for clarity).

ΣE_x and ΣE_y are available simultaneously; then their absolute value is computed before they are squared and added, yielding $\not\!\!E_t^2$. In a last computing step, ΣE_t and $\not\!\!E_t^2$ are compared with thresholds to issue the 4 trigger bits.

2.2.3 L1 FIFO and FRED alignment pipeline

The bunch counter value, ΣE_t, ΣE_x, ΣE_y, $\not\!\!E_t^2$, and the four trigger bits are time-aligned and sent to FIFOs without delay. These so-called **L1 FIFOs** store those output data while waiting for a L1A or L1R. The implementation of the L1 FIFOs and DAQ buffer management is similar to the one developed for DIRAC, so as to keep a consistent production between the Calorimetry L1 Trigger boards. See the DAQ Interface section below for details.

As for the output to FRED, in order for the trigger bits and the B0 tag bit coming from the Controller (see later section) to be aligned with that of the other PreFRED modules, a pipeline of variable length is implemented in which those bits are duplicated and stored while waiting for a certain delay. That waiting time, referred as **fred_delay**[5:0], is programmable and consists of a coarse delay in units of CDF clock cycles as well as an adjustable fine delay that will allow alignment of the PreFRED outputs within 22 ns. The **fred_delay** word is part of the control words that will have to be entered via VME. Its format breaks down into 2 distinct sets of 3 bits each. The 3 MSB (most significant bits) represent the coarse delay in units of 132 ns. The 3 LSB (least significant bits) are the phase-selecting fine delay in

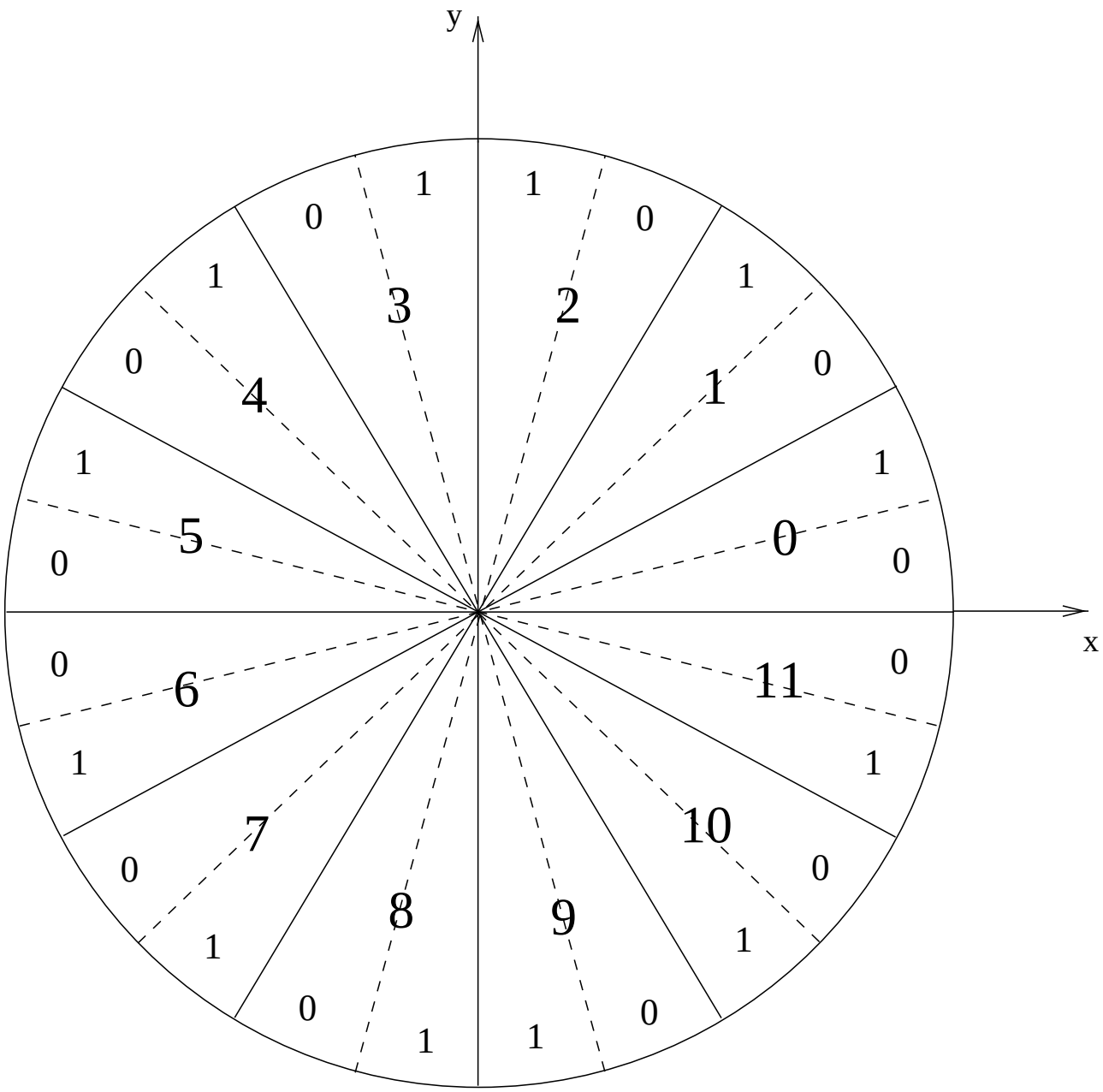


Figure 2: Numbering of the 24 calorimeter wedges for the summing process: two adjacent wedges are combined to form a sector. Each sector is associated with a CRATESUM board; their number appears in the inner part of the circle. The numbers at the outer part of the circle indicate whether the wedge is even (0) or odd (1). Note that each quadrant is a mirror image of its quadrant neighbors. In this framework, **even** means that the data from the associated wedges arrive first at the SUMET (or TOWTRG) board; **odd** means associated data arrive in the second 66 ns step.

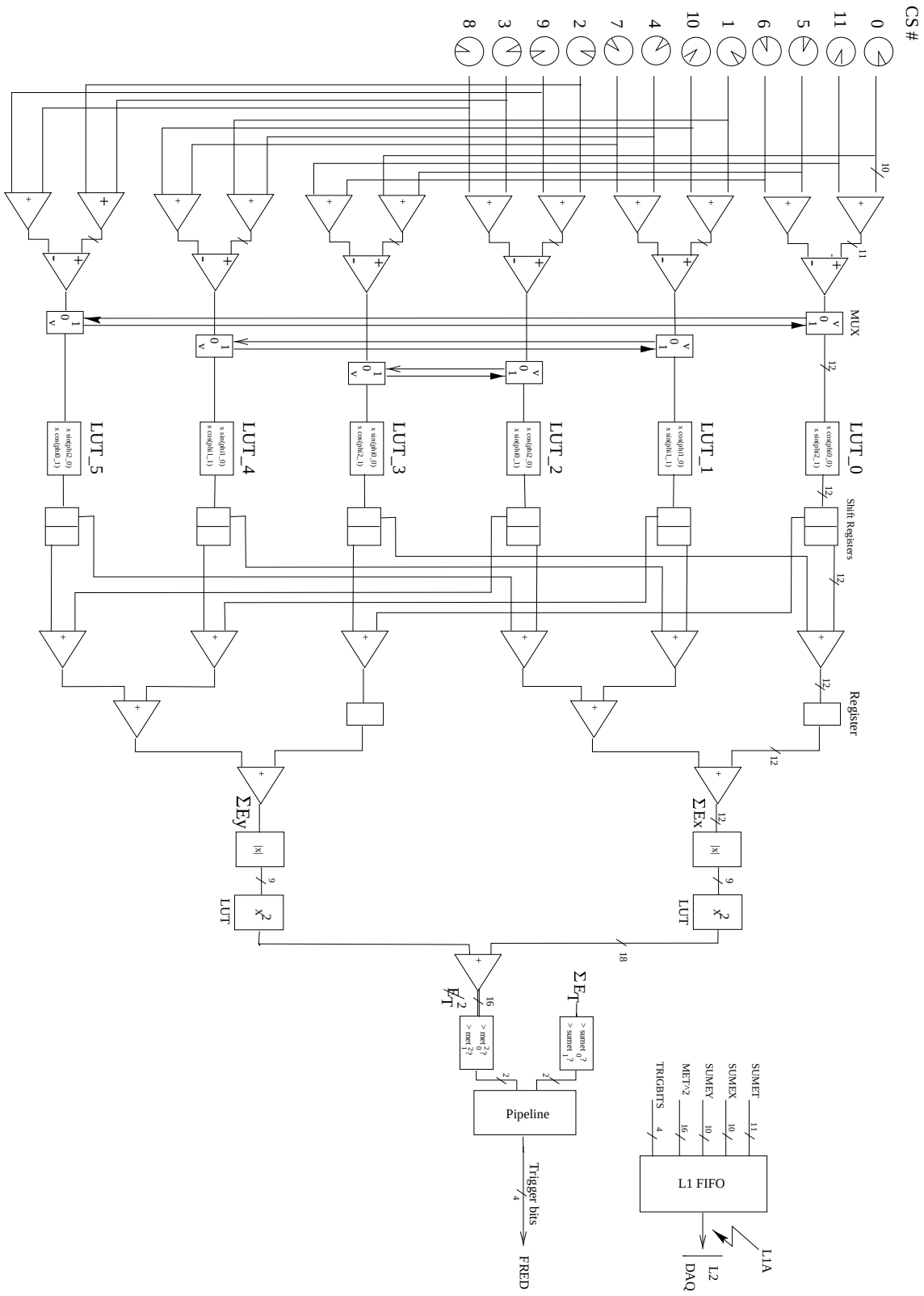


Figure 3: Functional diagram of the SUMET module. The actual chain that leads to ΣE_t has been omitted for clarity. The sectors in the circles represent the transverse detector sections to which the incoming data belong (a sector covers an even and an odd wedge). These data change every 66 ns. They are multiplexed as they arrive at the address ports of the ϕ-weighting Look-Up-Tables (LUTs). The MUX symbols represent the switching process occuring every 66 ns: the **0** (**1**) port connects even (odd) wedge data to the MUX output; **v** symbolizes a connection to the VME address bus, used when loading the LUTs. The data are demultiplexed right after the LUTs.

Table 1: Format of the DAQ output data. The first three words in the table also go to L2 on a L1A. * Signed words are actually absolute values appended with a sign bit at the MSB location; they are not two's complements. In the case where these words are saturated, i.e. their bits are all set to 1's, the sign bit has no significance and should be ignored.

Output word	# of bits	type	LSB	Full scale
ΣE_t	11	unsigned	1 GeV	2048 GeV
ΣE_x	10	signed *	0.5 GeV	256 GeV
ΣE_y	10	signed *	0.5 GeV	256 GeV
$\not\!\!E_t^2$	16	unsigned	1 GeV^2	(256 GeV)^2
trigbit	4	each bit set to 1 if above threshold		

units of 22 ns. Once the Global L1 system is integrated and the alignment delays between modules are perfectly known, **fred_delay** will be frozen to a constant value.

2.2.4 Output format

Table 1 summarizes the output to DAQ/L2 and their associated format.

During the adding process, an overflow bit is kept at every relevant stage. It is ORed with all other overflow bits involved in the ΣE_t ($\not\!\!E_t^2$) calculation. In the end, the ΣE_t, $|\Sigma E_x|$, $|\Sigma E_y|$ and $\not\!\!E_t$ words are ORed with their respective overflow bits so the results come out saturated in case of an overflow. The trigger bits are then set to 1, so as to accept the saturating event.

2.2.5 Connection to L2

Upon a L1A, ΣE_t, ΣE_x, and ΣE_y will be sent out from the L1 FIFOs to L2 via a front panel differential connector. These words will be the only data signals passing through the front panel of the board. In addition, the 2-bit DAQ buffer address associated with the event and received from the TSI will be sent in parallel with them, to tag the data for L2. In addition, an active high strobe signal, **FP_str**, will be sent along, to let L2 know when the data are valid on the cable. This strobe turns high at the same time as the data become valid, and goes back low 45 ns afterwards. Thus the L2 system should use the trailing edge of that strobe to latch the data into input registers. Thus a total of 34 differential bits are sent to L2. They go through a single 80-pin RPS-80-LM Honda connector (see pin assignment for that connector in Table 2). Some pins are connected to ground.

2.2.6 Connection to the J3 backplane

As mentioned earlier, every 66 ns, the Aux-card will receive 120 input signals from the CRATESUMs, feeding the SUMET module through the J3 backplane. The backplane is the same as the one used for the L1 Calorimetry crate, which has 24 point-to-point connections between each DIRAC and its associated CRATESUM[6]. In a similar way, the PreFRED modules will use subsets of these connections to send their trigger bits to FRED. In the SUMET case, only 4 output connections will be used. The input/output pin assignments on the backplane are the same for every PreFRED module; they are shown in Table 3.

Table 2: Pin assignments for the SUMET front panel connector, i.e. **output to L2.** Format is "pin number: data". SUMET = ΣE_t, SUMEX = ΣE_x, SUMEY = ΣE_y, L1BA = DAQ buffer address. For each one of those words, bit 0 represents the least significant bit. SUMEX9 and SUMEY9 are sign bits.

1: SUMET0+	41: SUMEX6+
2: SUMET0-	42: SUMEX6-
3: SUMET1+	43: SUMEX7+
4: SUMET1-	44: SUMEX7-
5: SUMET2+	45: GND
6: SUMET2-	46: SUMEX8+
7: GND	47: SUMEX8-
8: SUMET3+	48: SUMEX9+
9: SUMET3-	49: SUMEX9-
10: SUMET4+	50: SUMEY0+
11: SUMET4-	51: SUMEY0-
12: SUMET5+	52: GND
13: SUMET5-	53: SUMEY1+
14: GND	54: SUMEY1-
15: SUMET6+	55: SUMEY2+
16: SUMET6-	56: SUMEY2-
17: SUMET7+	57: SUMEY3+
18: SUMET7-	58: SUMEY3-
19: SUMET8+	59: GND
20: SUMET8-	60: SUMEY4+
21: GND	61: SUMEY4-
22: SUMET9+	62: SUMEY5+
23: SUMET9-	63: SUMEY5-
24: SUMET10+	64: SUMEY6+
25: SUMET10-	65: SUMEY6-
26: SUMEX0+	66: GND
27: SUMEX0-	67: SUMEY7+
28: GND	68: SUMEY7-
29: SUMEX1+	69: SUMEY8+
30: SUMEX1-	70: SUMEY8-
31: SUMEX2+	71: SUMEY9+
32: SUMEX2-	72: SUMEY9-
33: SUMEX3+	73: GND
34: SUMEX3-	74: FP_str+
35: GND	75: FP_str-
36: SUMEX4+	76: L1BA0+
37: SUMEX4-	77: L1BA0-
38: SUMEX5+	78: L1BA1+
39: SUMEX5-	79: L1BA1-
40: GND	80: GND

Table 3: Pin assignments for the PreFRED-J3 Backplane connection. **TRIGBIT** = output trigger bits to FRED, **CSIN** = input from CRATESUMs. In the case of the SUMET board, only the four least significant TRIGBITs are used, and the CSIN pins transmit the wedge transverse energies. In the case of the TOWTRG module, CSIN brings in 20 trigger bits. The index associated with CSIN represents the channel number as depicted in Figure 2. The numbers in parentheses are the bit numbers with 0 as least significant. **AUX_SUM_EN** is connected to ground on the board so as to forbid the summing of channels enabled for DIRAC. The Receiver Aux-card makes no connection to the TRIGBIT pins.

	Row						
Pin #	z	a	b	c	d	e	f
1	GND	GND	TRIGBIT(0)	TRIGBIT(1)	TRIGBIT(2)	TRIGBIT(3)	GND
2	GND	N/C	TRIGBIT(4)	TRIGBIT(5)	GND	TRIGBIT(6)	GND
3	GND	N/C	TRIGBIT(7)	GND	TRIGBIT(8)	TRIGBIT(9)	GND
4	GND	N/C	TRIGBIT(10)	TRIGBIT(11)	GND	TRIGBIT(12)	GND
5	GND	GND	TRIGBIT(13)	GND	TRIGBIT(14)	TRIGBIT(15)	GND
6	GND	N/C	CSIN0(0)	CSIN0(1)	CSIN0(2)	CSIN0(3)	GND
7	GND	N/C	CSIN0(4)	CSIN0(5)	GND	CSIN0(6)	GND
8	GND	N/C	CSIN0(7)	CSIN0(8)	CSIN0(9)	CSIN1(0)	GND
9	GND	GND	CSIN1(1)	GND	CSIN1(2)	CSIN1(3)	GND
10	GND	N/C	CSIN1(4)	CSIN1(5)	CSIN1(6)	CSIN1(7)	GND
11	GND	N/C	CSIN1(8)	CSIN1(9)	GND	CSIN2(0)	GND
12	GND	N/C	CSIN2(1)	CSIN2(2)	CSIN2(3)	CSIN2(4)	GND
13	GND	GND	CSIN2(5)	GND	CSIN2(6)	CSIN2(7)	GND
14	GND	N/C	CSIN2(8)	CSIN2(9)	CSIN3(0)	CSIN3(1)	GND
15	GND	N/C	CSIN3(2)	CSIN3(3)	GND	CSIN3(4)	GND
16	GND	N/C	CSIN3(5)	CSIN3(6)	CSIN3(7)	CSIN3(8)	GND
17	GND	GND	CSIN3(9)	GND	CSIN4(0)	CSIN4(1)	GND
18	GND	N/C	CSIN4(2)	CSIN4(3)	CSIN4(4)	CSIN4(5)	GND
19	GND	N/C	CSIN4(6)	CSIN4(7)	GND	CSIN4(8)	GND
20	GND	N/C	CSIN4(9)	CSIN5(0)	CSIN5(1)	CSIN5(2)	GND
21	GND	GND	CSIN5(3)	GND	CSIN5(4)	CSIN5(5)	GND
22	GND	N/C	CSIN5(6)	CSIN5(7)	CSIN5(8)	CSIN5(9)	GND
23	GND	N/C	CSIN6(0)	CSIN6(1)	GND	CSIN6(2)	GND
24	GND	N/C	CSIN6(3)	CSIN6(4)	CSIN6(5)	CSIN6(6)	GND
25	GND	GND	CSIN6(7)	GND	CSIN6(8)	CSIN6(9)	GND
26	GND	N/C	CSIN7(0)	CSIN7(1)	CSIN7(2)	CSIN7(3)	GND
27	GND	N/C	CSIN7(4)	CSIN7(5)	GND	CSIN7(6)	GND
28	GND	N/C	CSIN7(7)	CSIN7(8)	CSIN7(9)	CSIN8(0)	GND
29	GND	GND	CSIN8(1)	GND	CSIN8(2)	CSIN8(3)	GND
30	GND	N/C	CSIN8(4)	CSIN8(5)	CSIN8(6)	CSIN8(7)	GND
31	GND	N/C	CSIN8(8)	CSIN8(9)	GND	CSIN9(0)	GND
32	GND	N/C	CSIN9(1)	CSIN9(2)	CSIN9(3)	CSIN9(4)	GND
33	GND	GND	CSIN9(5)	GND	CSIN9(6)	CSIN9(7)	GND
Connector Key (equivalent to 3 pins)							
34	GND	N/C	CSIN9(8)	CSIN9(9)	CSIN10(0)	CSIN10(1)	GND
35	GND	N/C	CSIN10(2)	CSIN10(3)	GND	CSIN10(4)	GND
36	GND	N/C	CSIN10(5)	CSIN10(6)	CSIN10(7)	CSIN10(8)	GND
37	GND	GND	CSIN10(9)	GND	CSIN11(0)	CSIN11(1)	GND
38	GND	N/C	CSIN11(2)	CSIN11(3)	CSIN11(4)	CSIN11(5)	GND
39	GND	N/C	CSIN11(6)	CSIN11(7)	GND	CSIN11(8)	GND
40	GND	N/C	CSIN11(9)	AUX_SUM_EN	TRIGBIT(16)	TRIGBIT(17)	GND
41	GND	GND	TRIGBIT(18)	AUX_SPARE	TRIGBIT(19)	b0_delayed	GND
42	GND	N/C	bp_spare(0)	bp_spare(1)	GND	bp_spare(2)	GND
43	GND	N/C	N/C	AUX_ENABLE	N/C	N/C	GND
44	GND	N/C	N/C	N/C	N/C	N/C	GND

2.3 Implementation

On the schematics in Figure 1 and in Appendix, the SUMET function is implemented in the **PreFRED_logic** block.

2.3.1 Adding, squaring, threshold comparison, alignment pipeline: the FPGA integration

The new generation of Programmable Logic Devices with Embedded Memory allows a great deal of flexibility, integration, and registered speed performance. By including a maximum number of logic functions within such a device, we effectively reduce the number of parts needed for the board, increase the speed performance and enable future re-design of the logic by simply reprogramming or reconfiguring the device. Such an implementation has been achieved at the simulation stage with the Altera 503-pin EPF10K100GC503-3DX PGA chip: except for the ϕ-weighting and the L1 FIFOs, all of the logic in Figure 3 has been fitted into that chip, later referred as the **Data Processor**, with the Altera development software MaxPLUS2. The functions implemented include the cascade adders, the absolute value and squaring functions, the threshold comparators and the alignment pipeline. They were coded with the Altera High-Level Development Language (a language similar to VHDL, only simpler) and its parameterized "megafunctions". From that source code, the Altera Development Environment fits all required logic within the single FLEX10K100 chip. The 120 input bits coming from the CRATESUMs, the 72 bits going out to the 6 ϕ-weighting devices (see next paragraph), the 72 bits coming back in from these devices, as well as the 55 total output bits are well accomodated by the 406 I/O pins available on the FLEX10K100.

At the end of the adder cascade (see Figure 3), the MSB of the overall ΣE_x or ΣE_y word (FS = 512 GeV) is considered as an overflow bit while its LSB is dropped, yielding a 9-bit word with a 256 GeV full scale and a 0.5 GeV least count. This word feeds the address port of a 512×18 embedded memory which acts as a look-up table to produce the square of ΣE_x or ΣE_y. Then $(\Sigma E_x)^2$ and $(\Sigma E_y)^2$ are added, yielding the $\not\!\!E_t^2$ result as well as a last overflow bit to be ORed with the $\not\!\!E_t$ overflow sequence. In the process, the two LSB are dropped from $\not\!\!E_t^2$ so that the least count for the result becomes 1 GeV^2. [1] Thus for the compare-to-threshold stage, simple 16-bit-wide comparators are used on the $\not\!\!E_t^2$ path. Similar comparators deal with ΣE_t. Since the system has to be reconfigurable to accomodate as many as 3 or 4 $\not\!\!E_t^2$ thresholds (and 1 or 0 ΣE_t thresholds), 4×16 bits are reserved in the FPGA for the thresholds' allocation. For the ΣE_t thresholds, only 11 out of 16 bits are used. The thresholds must be down-loaded before the beginning of a run. This is done across a VME 32-bit data port connected to registers within the FPGA. Two thresholds are loaded at a time, so that only two VME write cycles are required for the 4 thresholds.

The FRED alignment pipeline is implemented with a shift register whose length is set to a maximum of 8 4-trigger-bit words, i.e. about 1 μs delay. Via a multiplexer, the coarse delay word **fred_delay**[5:3] selects which output of the shift register is to be used as an output to FRED. Moreover, this 4-bit output is latched with a clock selected among a set of 6 132-ns clocks whose rising edges are separated by 22 ns. This is implemented with a clock multiplexer whose selector is the fine delay word **fred_delay**[2:0].

[1] No reduction of the $\not\!\!E_t^2$ 16-bit word to a smaller 8-bit $\not\!\!E_t$ is foreseen, as it would involve a large computing latency.

Configuring the FLEX10K FPGAs The FLEX10K100 FPGA is programmed with volatile memories within it. After the power is turned off, it needs to be reprogrammed or "reconfigured". Dedicated PROMs (2 daisy-chained Altera EPF1s) on the SUMET board will automatically reconfigure the FPGA as soon as the power is turned on. The same PROMs are used to configure the **Controller**, a FLEX10K30 (see below). In the early life of the board, instead of configuring with PROMs, a FLEX download cable[10] linking an Altera programming computer to the board front panel will be used in order to test various programs within the FLEX10K devices. When a program, loaded with the FLEX cable, has been tested and validated, it can be transferred to new PROMs which then have to replace the old ones on the board. On the front panel, a mechanical switch associated with LEDs selects which of the PROMs or the download cable is used for configuration (see **Front Panel** schematics). The state of the switch, whose signal name is **src_config**, can be remotely read out by VME, as the MSB (8th bit) of the Controller program version number (see Table 7). In run mode, this switch should be positioned to the PROM path (**src_config** = 0), so that whenever a power outage occurs, the PROMs are immediately ready to reconfigure the FLEX10K devices.

2.3.2 ϕ-dependent weighting: the SRAM LUTs

The ϕ-dependent weighting function cannot easily be implemented in such an FPGA due to the large combinatorial multiplication process and the subsequent extra latency it would induce. This is why fast static random access memomies (SRAMs) are used as look-up tables (LUTs) to perform the multiplications. After the first stage of adders and before the ϕ-weighting, the transverse energies are signed and coded on 12 bits, with LSB = 0.5 GeV and full scale FS = $\pm$1024 GeV. To preserve a certain resolution after weighting by the cosine/sine factors, the SRAMs' 12-bit data words are rescaled with LSB = 0.25 GeV and FS = $\pm$512 GeV.[2] Should the result of a multiplication be greater than 512 GeV, the 11 LSB of the LUT data bus are all set to 1's.

So 6 SRAMs are required with a 12-bit wide addressing and at least 12 bits for the output (4K$\times$12 minimum requirement). Asynchronous 16K$\times$16 fast SRAMs with read cycle times of less than 15 ns (Motorola MCM62996) were selected to fulfill that purpose. These memories are loaded via VME at initialization (after power is turned on): the connections of their address and data ports are switched to the VME address and data ports within the **Data Processor**. The width of the VME data bus allows the reading/writing of two such memories at the same time (see details in Appendix, section 5.1.2).

2.3.3 Timing diagram and latency

A timing diagram of the various signals going through the **Data Processor** has been obtained from a simulation using the Altera development package (please request it from the authors if needed). There is no rest-time between the processing of separate events: the whole sequence of events is pipelined and each event comes in two 66-ns data bunches. On the diagram, the 12 **etin** lines represent the incoming 10-bit signals coming from the CRATESUMs. In this particular instance, time $t_0 = 594$ ns indicates the arrival of the first data bunch into the FPGA chip [3]. Incoming data for the first event extend to $t_0 + 132$ ns.

[2] A simple Monte Carlo simulation shows the $\not{E}_t$ numerical resolution deteriorates by less than 0.5 GeV in the scheme presented here.

[3] Prior to t_0, the loading of the thresholds is performed via VME.

etin_0 is merely the output of the input register associated with the incoming E_{t0}. The first adding stage yields 6 partial sums which are latched with the 66 ns clock as they are sent out to the LUTs as **lut_address.** The 6 weighted partial sums which come back into the Altera chip from the LUTs are called **lut_data.** They are also latched, and the associated register output is **sumodd**. The input named **clk_132ns**[2:0] is a 132-ns-period square clock echoed with 0, 22, and 44 ns shifts. By using it and its complements, registers can be strobed with a 22 ns granularity. **clk_132ns0** is synchronized with the incoming data from CRATESUMs.

ΣE_t is made available for output as the truncated 11-bit **sumet_** while ΣE_x (**sume0**) and ΣE_y (**sume1**) are extracted from their absolute values and re-concatanated with their sign bits. **metsq** is $\not{E}_t^2$. The 4 trigger bits appear as **trigbits**. All those output data are aligned into a 132-ns clocked register, and **trigbits** feeds a programmable length pipeline (shift-register) whose output **tofred** is latched by one of the **clk_132ns**[2:0] (or their complements) according to the fine delay set in **fred_delay** (see section 2.2.3). **fred_delay** itself is an input to the **Data Processor** and is maintained by the **Control** block. In the example shown on the diagram, "**fred-delay** = 5" causes the output to FRED to be delayed by about 120 ns after their time of first availability (5×22 ns + 20 ns prior to that delay for setting up the variable delay process).[4]

2.4 Peripheral functions

As mentioned earlier, the **PreFRED_logic** is the core of the design. All the other blocks in the top level diagram are basically interfaces with the outside world.

2.4.1 The DAQ Interface

The DAQ Interface consists of a set 8 9-bit-word asynchronous FIFOs and 19 4×4 RAM DAQ buffers (see schematics in Appendix). The FIFOs can store as many as 256 words. This depth should be useful in diagnostics mode, when testing the board with numerous fake events. The L1 FIFOs gather the output words while waiting for a L1 accept, which triggers the dispatch of the data to L2 and the DAQ buffers. The output words are **dataout**[55:0] from the **Data Processor**, which includes ΣE_t, ΣE_x, ΣE_y, $\not{E}_t^2$ and the trigger bits, the bunch number[4] and the board ID from the control block. When the FIFOs are read out upon a L1A/L1R, the 11-bit ΣE_t word feeds a register which yields the SUMET from the previous crossing to the DAQ buffers, in case of a L1A. In turn, the DAQ buffers are read out via VME subsequent to a L2 accept. This is done with a block transfer which, in the SUMET case, contains four 32-bit words. These words are strobed with signals generated by the **Controller**, referred as **_L2B_R**[3:0]. Their content and format are:

- **1st DAQ word :** [31:8] = board ID; [7:0] = bunch number
- **2nd DAQ word :** [31:24] = unused; [23:22] = $\not{E}_t^2$ trigger bits, [21:20] = ΣE_t trigger bits, [19:16] = FRED-delayed trigger bits (used for diagnostics); [15:0] = $\not{E}_t^2$
- **3rd DAQ word :** [31] = FRED-delayed B0 (used for diagnostics); [30:21] = ΣE_y, [20:11] = ΣE_x, [10:0] = ΣE_t
- **4th DAQ word :** [31:11] = unused; [10:0] = ΣE_t from previous crossing

[4]By ignoring **tofred** and merely considering **trigbits**, the 4 trigger bits could be issued out to FRED without delay, shortly after the specified 400 ns overall latency requirement.

2.4.2 The VME Interface

The **VME Interface** uses an original design documented by John Wahl for the DIRAC card[9]. The various functions are basically the same, only implemented differently: except for separate data transceivers, they are now integrated into a single Altera PLD, an EPM7128_160-2 device (see schematics in Appendix). They include the following:

- **Address Modifier (AM) masking**: ensures the type of transaction requested is an extended (32-bit word) data access or extended block transfer (BLT). Both supervisory and non-privileged accesses are allowed by the fact that **AM**[2] is ignored[11].

- **Geographical Address (GA) masking**: compares the 5 MSB of the VME address bus (**A**) with the geographical address given by the location of the board in the crate (i.e. the slot number, which comes active low on the backplane and therefore has to be inverted). If both addresses match, the PreFRED module is selected for a VME transaction.

- **VME control signal generation** (formely implemented in a VME2000 chip): the VME-specific signals used by PreFRED are gathered in the **VME_Control** bus; they are described in Table 5. Those generated by the VME Interface from primitive P1 signals are **_modsel**, **_vme_data_str** and **vme_write.** A list of the primitive P1 signals is given in Table 4 (for a detailed description of these signals, see reference [11]). **_modsel** is activated when the address modifier and geographical address have been checked and when the Address Strobe (**_AS**) is active. To avoid glitches on **_modsel**, the leading edge of a 10-ns delayed **_AS** is actually used to latch the result of the **AM**, **_LWORD** and **A**[31:27] masking. **_modsel** remains active as long as the address strobe or the data strobe (**_DS0** or **_DS1**) is active, whichever lasts longer. **_vme_data_str** is activated by **_modsel** and **_DS0** or **_DS1**, as long as no acknowledge or bus error signal is active. It remains active as long as **DS0** or **DS1** is active. The other signals generated by the VME Interface are **_DTACK** and **_BERR**, which are derived respectively from **_ACK** and **_vme_error** coming from the **Controller** (see section below). Both **_DTACK** and **_BERR** are open-collector outputs.

- **VME address latching**: a "local" **VME_Address**[26:2] is latched to the PreFRED board upon **_modsel**'s activation. **VME_Address** is actually the value of a counter (see next item) loaded with **A**[26:2] using an active-high 10-ns strobe whose rising edge matches **_modsel**'s falling edge.

- **BLT handling**: if a BLT is requested (**AM**[1] = 1), **VME_Address** is incremented on the rising edge of the **_vme_data_str** signal delayed by some 10 ns. As many as 64 word transfers are supported by VME for a BLT.

2.4.3 Control

The control block includes the "house-keeping" functions. See schematics in Appendix.

The board ID The control block contains a set of 19 switches which allows assignment of a hard-coded ID number corresponding to the board type and serial number. This ID number is appended to the Geographical Address gotten from the VME interface. Thus a

Table 4: **VME_Controls** bus (P1 signals). Signals beginning with an underscore are active low. Missing bit numbers are unused by PreFRED.

Bit #	Name of signal	Description
0	sysclk	VME-specific 60 ns clock going to the **Controller**, currently unused
1	_DS1	Data strobe 1
2	_DS0	Data strobe 0
3	_vme_write	Read/write signal; 0 = write; 1 = read
4	_DTACK	Acknowledge signal
5	_AS	Address strobe
6	_IACK	Interrupt acknowledge cycle signal
15	_BERR	Bus error
17	_LWORD	Long word selector, should always be 0 for 32 bit transfers

Table 5: **VME_Control** bus. All signals are **VME_Interface** output except **_ACK** and **_vme_error** which are generated by the **Controller**. Signals beginning with an underscore are active low.

Bit #	Name of signal	Description
0	_modsel	Activates the PreFRED module for a VME transaction
1	_vme_data_str	active-low VME data strobe, comes along with the VME address
2	vme_data_str	same VME data strobe, only inverted to make it active-high
3		unused
4	_ACK	Acknowledge signal returned by the PreFRED Controller
5	_vme_error	Transaction error bit returned by the Controller; currently always set to 1
6	vme_write	Write (1) or read (0) request from VME
7	sysclk	VME-specific 60 ns clock going to the Controller, currently unused

24-bit word identifies the board uniquely in the VME address map. This word is sent along with the 8-bit bunch number as the header word at the beginning of a DAQ transfer.

The CDF-specific P2 signals The **Trig_Ctrl** bus includes the specific CDF signals which come from the P2 backplane. The list of signals used by PreFRED follows. Except for the clock itself, these signals are latched with **CDF_clk** as they arrive at the board.

- **CDF_clk**, 132-ns master clock, comes as a differential signal and is converted to a single signal by an AT&T 1241 MF line receiver
- **_HALT**, active low[5], stops the writing/reading of the L1 FIFOs and the writing to the DAQ buffers
- **_RESET**, resets the L1 FIFOs and the error register (see below), only works if **_HALT** is asserted
- **_B0**, tells the board that the current crossing corresponds to Bunch 0. This signal is used to reset the bunch counter[4] after an offset delay associated with the time location of the PreFRED module in the overall L1 pipeline. This delay, named **B0_offset**[5:0], is programmable in units of **CDF_clk** cycles; like the other programmable delays on the board, it is set via VME during initialization and will be frozen at the time of integration.
- **_L1A, _L1R**, L1 accept and reject signals, either one of which comes every 132 ns after the halt/reset sequence and subsequent offset delay. If L1A is asserted, the data are passed on from the L1 FIFOs to the DAQ buffers and L2
- **_L1BA**[1:0], DAQ buffer write-address sent along with a L1 accept, tells the DAQ Interface which of the 4 buffers is to be overwritten

Trig_Ctrl also includes an output line from the **Controller**, the board error signal **_cdf_error**. This error bit is asserted (set to 0) when any one of the 8 L1 FIFOs has raised its "Full" Flag (**_FF**[7:0]). It is then latched by **CDF_clk** and can only be negated again by the **_RESET_** strobe.

The clock generator The **clock generator** issues all of the primary clock signals required for the board: it receives **CDF_clk** from the P2 backplane and first symmetrizes it (50%/50% duty cycle) without changing its phase. Subsequently, it tap-delays it by 22 ns increments to create a **cdf_132ns**[5:0] bus. One of these 6 clocks is selected by the **Controller** to be synchronous with the incoming data from the CRATESUMs. The choice of this clock, thereafter named **CS_132ns**, is determined by the programmable delay **CS_delay**[2:0] set via VME during initialization. Like **fred_delay**, this delay will be fixed once and for all at the time of integration. From **CS_132ns**, the **clock generator** produces a 66 ns period clock, **clk_66ns**, whose rising edge is aligned with that of **CS_132ns**. **clk_66ns** is used to latch the **Data Processor** input registers, i.e. the CRATESUMs' incoming data. It also clocks out the data to the ϕ-weighting LUTs and clocks them back into the **Data Processor**. **CS_132ns** is also 22 ns tap-delayed into **clk_132ns**[2:0] to provide the **Data Processor** with a fine time granularity when latching registers. **clk_132ns0** is identical to **CS_132ns**.

[5]In this note, all signals whose name begins with an underscore are active low

Table 6: VME address map for the SUMET module. Note that A1 and A0 are always 0's as the full VME address refers to byte locations and CDF uses 32-bit (4-byte) data transfers. A26:A24 are also 0's.

Transaction target	Port location	Access type	# of data bits	VME address (A23:A0)
Control registers	Controller	R/W	10×8	0x000024 : 0x000000
Module ID	Controller	RO	32×8	0x10007C : 0x100000
ϕ-weighting LUTs	Data Processor	R/W	$3 \times 8K \times 24$	0x417FFC : 0x400000
Trigger thresholds	Data Processor	R/W	2×32	0x500004 : 0x500000
dataout	Data Processor	RO	2×32	0x60FFFC : 0x600000
L1 FIFOs	Data Processor	WO	256×32	0x7003FC : 0x700000
DAQ buffer 0	DAQ buffer 0	RO	4×32	0x80000C : 0x800000
DAQ buffer 1	DAQ buffer 1	RO	4×32	0x90000C : 0x900000
DAQ buffer 2	DAQ buffer 2	RO	4×32	0xA0000C : 0xA00000
DAQ buffer 3	DAQ buffer 3	RO	4×32	0xB0000C : 0xB00000

On the VME side, the Data Strobe signal is made active high by the **clock generator** and delayed by 25 ns taps. This then allows the **Controller** to select appropriate time signals for various VME accesses on the board.

The Controller The **Controller** is the main chip in the control block. It is an Altera EPF10K30BC356-3 FPGA, of the same family as the **Data Processor**. It takes care of several functions. Primarily, it handles the **control of VME transactions**. In particular, it generates the acknowledge signal **_ACK** expected by the VME Interface as a hand shake subsequent to a Master's request. **_ACK** is asserted some time after a Data Strobe, the delay depending on the transaction requested. It is typically about 125 ns, except when reading the output of the **Data Processor** (**dataout**, see below), which requires an extra processing latency. The **_vme_error** signal is also returned by the **Controller**. In the current implementation, it is always negated. The VME address decoding is also handled by the **Controller** as part of VME transaction control. This involves the VME address map listed in Table 6.

The type of VME access allowed in that table (reading/writing) applies only when in load mode (load mode is on/off when run mode is off/on). The run/load mode is determined by the status of a control register writable only via VME. The list and description of the control registers are given in Table 7.

In Table 6, the module ID is a string of 32 ASCII characters used to identify an individual module and/or map the location of all such modules in a crate. It is different from the board ID discusssed earlier. Its format is specified in [8].

Since the data bus of each ϕ-weighting LUT is only 12 bits wide, the LUTs are accessed by VME in pairs. The 24 LSB of the VME data bus are used. As for the LUT address width, as was seen in an earlier section, 12 bits are used while in run mode; however a 13th bit is added to the SRAM address bus so as to allow extra throughput configurations when in diagnostics mode. This accounts for the "$3 \times 8K \times 24$" in Table 6, 3 referring to the number of LUT pairs. Then the partial address **A**[16:15] = 0, 1, or 2, points to one of the LUT pairs, in the top-to-bottom order in which they appear in Figure 3. For instance **A**[16:15] = 0 refers to both LUT_0 and LUT_1, with the partial VME data bus **D**[11:0] (**D**[23:12]) mapping the LUT_0 (LUT_1) data bus.

Table 7: SUMET control registers. These are 8-bit words which map the 8 MSB of the VME 32-bit data bus. Some words don't use all 8 bits, then the MSB can be used as spare bits; e.g. **CS_delay**[2:0] maps **VME_data**[26:24]. The "Access" type applies in load mode.

Word[bit]	Name	Access	VME Address	Description
0[5:0]	B0_offset[5:0]	R/W	0x000000	Bunch 0 offset delay in 132 ns units, must include the PreFRED Data Processor latency (i.e. > 3 **CDF_clk** cycles)
1[2:0]	CS_delay[2:0]	R/W	0x000004	Delay introduced after the **CDF_clk** rise to clock the CRATESUM input data into the Data Processor, in 22 ns units
2[5:0]	fred_delay[5:0]	R/W	0x000008	FRED alignment delay, encompasses a coarse and fine granularity (see format in section 2.2.3)
3[0]	run	R/W	0x00000C	Defines the mode of operation of the board: 1=run, 0=load
3[1]	reset	R/W	"	When asserted (=1) in load mode, resets the L1 FIFOs and the board error bit
3[2]	BP_trigbits_en	R/W	"	When =1 in load mode, enables output to backplane
3[3]	FP_str	R/W	"	When =1 in load mode, turns the Front Panel strobe light on
3[4]	FIFO_R	R/W	"	When =1 in load mode, generates a strobe that reads out the L1 FIFOs
3[5]	L1B_W	R/W	"	When =1 in load mode, generates a strobe that writes to a DAQ buffer
3[7:6]	L1BA[1:0]	R/W	"	In load mode, DAQ buffer address to write to when **L1B_W** is asserted
4[7:0]	FF_mask[7:0]	R/W	0x000010	Masks full FIFO flag so as to be able to run with a dysfunctioning L1 FIFO
5[0]	aux_enable	R/W	0x000014	When =1 in load mode, enables input from Aux-card
5[1]	aux_spare	R/W	"	In load mode, spare bit which connects to the Aux-card
5[7:2]		R/W	"	Spare bits
6[7:0]	FF[7:0]	RO	0x000018	Full status flag for each L1 FIFO (active high); allows to find dysfunctioning L1 FIFO when board error bit comes up
7[7:0]	EF[7:0]	RO	0x00001C	Empty status flag for each L1 FIFO (active high)
8[7:0]	ctrl_version	RO	0x000020	Version number of the Controller configuring program
8[7]	src_config	RO	"	Source configuring the FLEX10K devices: 0 = PROMs, 1 = download cable
8[6:4]	PreFRED type	RO	"	0 = SUMET, 1 = TOWTRG, 2 = MUON, 3 = TRACKING
9[7:0]	proc_version	RO	0x000024	Version number of the Data Processor configuring program
9[6:4]	PreFRED type	RO	"	0 = SUMET, 1 = TOWTRG, 2 = MUON, 3 = TRACKING

The 16-bit trigger thresholds are also loaded in pairs, which explains the "2×32" in Table 6. $\mathbf{A}[2] = 0$ (1) refers to the 2 ΣE_t ($\not\!\!E_t^2$) thresholds.

dataout consists of 2 sets of output words as they directly come out of the **Data Processor**. They can be read in diagnostics mode in order to test the static functioning of the **Data Processor** independently of the DAQ interface chain. The 2 word sets correspond to the 2nd and 3rd DAQ words (see list given in the DAQ Interface section above). They are addressed respectively by $\mathbf{A}[15] = 0$ and $\mathbf{A}[15] = 1$. In this mode, $\mathbf{A}[14{:}2]$ are used to produce 12 fake 10-bit $E_t(\phi)$ input data words. These stay unchanged until the end of the VME transaction, which occurs after the **Data Processor** has had enough time to process the data and present the expected **dataout** value on its output port (> 400 ns).

Also used in diagnostics mode, the writing to the L1 FIFOs from the same **Data Processor** output port permits testing of the function of each L1 FIFO. The VME data bus then carries some 32-bit pattern which is simultaneously written to all FIFOs. As for the DAQ buffers, their reading occurs following the sequence described in the DAQ Interface section. Writing to them with VME is not directly possible. It can only be done by reading the L1 FIFOs. A step-by-step read/write procedure has to be used to test the standalone DAQ Interface chain. It involves dedicated control registers (**FIFO_R, L1B_W, L1BA**); see details in Table 7.

While in run mode, the **Controller** authorizes only the following transactions:

- Writing of the "run" control register (1 = run mode, 0 = load mode)
- Reading of DAQ buffers
- Reading of control registers and module ID words
- Reading of trigger thresholds

In addition to VME transaction control, the **Controller** deals with bunch counting[4] and handles the halt/reset sequence at the beginning or in the middle of a run. In particular, it delays the **_B0** signal with **B0_offset** (see the **CDF-specific P2 signals** paragraph), producing the **b0_delayed** signal passed on to the Data Processor for alignment with the trigger bits sent to FRED. The Controller generates as well the strobe signal **_L1_FIFO_W** which controls the writing of **dataout** to the L1 FIFOs. This strobe comes every 132 ns, once the first delayed B0 signal is detected after a HALT/RESET sequence. It is synchronized with **dataout** and the bunch number, both latched by **CS_132ns**. The **Controller** also issues the **_L1FIFO_R** (FIFO read out) and **_L1B_W** (DAQ buffer write) strobes. Moreover, it deals with error management, which merely consists in ORing the 4 L1 FIFO "full" flags to produce an **_error** signal on P2. At last the **Controller** issues control signals to the P3 backplane (Aux Card control, FRED trigger bit output enable) and to the Front Panel (strobe that validates the data sent to L2, **FP_str**, and LED control). 8 LED lights exist on the Front Panel, 5 of which are controlled by the **Controller** (see their meaning on Front Panel schematics in appendix).

In the appendix, we give in Table 8 the list of the various control signals that are transmitted between the **Controller** and the rest of the board. These lines are gathered in what is named the **Control** bus on the schematics.

3 TOWTRG

3.1 Specifications

The TOWTRG Module receives 20-bit tower-trigger summary words from the 12 CRATE-SUM boards[7] and condenses them into a single 20-bit detector summary word to be sent out to FRED and DAQ. The 20-bit input tower-trigger summary words from the CRATE-SUMs will, a priori, include 10 single object trigger bits (e.g. 1 electron with $E_t > 20$ GeV, 1 photon with $E_t > 25$ GeV...) and 3 di-object pairs of bits (e.g. at least 2 electrons with $E_t > 10$ GeV...) which tell whether 0, 1, 2 or more objects with the passed a di-object threshold in the associated wedge pair. In the case where more than 2 objects fulfill a di-object trigger requirement, the result of the 2-bit adder is set to 3. The output tower-trigger summary word should reflect the same pattern. For the di-object triggers, out of the two resulting bits, only the most significant bit will tell FRED whether at least two objects of the same nature were found with the corresponding threshold. The lower significant bit will also be used by FRED for combined di-object triggers like e-μ. Thus in this scheme, 16 bits out of 20 are used by the system, which leaves 4 bits for additional information if required in a future development (tri-object trigger?).

The timing requirement is that all steps leading to the TOWTRG summary should be performed within one clock cycle (132 ns). However, the output must be sent in phase with the other PreFRED modules.

3.2 Design and Implementation

The logic design of the board is straightforward; the tower trigger summary will be performed in the same fashion as in CRATESUM with the DIRAC data: the single object trigger bits will have to be ORed whereas the di-object bits will simply be added two by two. Then an overflow in the adding process means more than 3 objects pass the di-object threshold, in which case the 2-bit result is set to 3.

This logic is simple enough that it can easily fit in the same type of FPGA (FLEX10K100) as that used for SUMET. Moreover, the number of input bits ($2\times$ 120) is identical to that of the SUMET board; the number of output bits (20) is lower than for SUMET (55). This enables the same board design for the 2 modules, as well as the use of the same kind of Aux cards (DIRAC's) to receive the incoming data. Only the FPGA program would have to differ. Thus, as for SUMET, the data is received in 66 ns sequences from the CRATESUMs: each 20-bit summary word is cut and sent in two sequential packets across the same kind of cables as that used to feed the SUMET Aux-card. The first sequence should be made of 10 bits that include the 6 di-object trigger bits as the di-object summing process takes longer than the ORing of the single trigger bits. The 2nd sequence would then be made of the remaining 10 single object trigger bits.

The SRAM look-up tables used in SUMET seem to be irrelevant in the TOWTRG scheme, as the data stream does not have to go through them a priori. However, they could be used for a $\Delta\phi$ correlation in a dijet or dielectron trigger.

Eventually, the 20 resulting bits are sent out of the FPGA to the L1 FIFO and to FRED, after a PreFRED alignment delay implemented in the same way as in SUMET. The only difference here is the larger number of bits to be sent to FRED, which is taken into account in the joint board design.

3.2.1 Connection to L2

No connection to L2 is required for TOWTRG, although the PreFRED Front Panel connectors used by the SUMET module will be available to transfer the 20 TOWTRG output trigger bits if found necessary.

3.2.2 Connection to the J3 backplane

As in the case of SUMET, every 66 ns, the Aux-card will receive 120 input signals from the CRATESUMs, feeding the TOWTRG module through the J3 backplane. As for the output to FRED in the TOWTRG case, 20 point-to-point connections will be used on the backplane. The pin assignments for the TOWTRG connection to the backplane (Table 3) show the locations of the input trigger bits from the CRATESUMs as well as the output trigger summary bits.

4 Conclusion

Both the SUMET module and the TOWTRG module will be implemented using an identical PC board with different programs loaded in their FPGAs. In both instances, the data will come from the CRATESUM boards in two sequences per event. The re-use of the DIRAC-specified Aux-Card and backplane is then appropriate to bring the incoming data to the board. Moreover the output trigger bits to FRED, whether they come from SUMET or TOWTRG, are implemented with point to point connections on the same L1 P3 backplane, as was specified to link the DIRACs to the CRATESUMs.

A total of 6 16K$\times$16 SRAM chips are needed for the SUMET ϕ-dependent weighting while only one Altera FPGA is required for the rest of the processing (FLEX10K100). In addition, the board has a VME interface (EPLD 7128), a **Controller** (FLEX10K30), and a DAQ interface. The latter uses the same FIFOs and DAQ buffers as DIRAC.

We believe the SUMET/TOWTRG board is general and flexible enough that it can also be used to implement the other PREFRED modules, with just different programs in their FPGAs.

References

[1] A. Amadon et al. *Conceptual Design of the Global L1 Trigger Decision Crate for the Run II Upgrade*, CDF Note 4339.

[2] H. Frisch et al. *Conceptual Design of the L1 Calorimeter Trigger for the Run II Upgrade*, CDF Note 2909.

[3] G. Feild *Conceptual Design of CDF FRED for Run II*, Yale University report, 09/97, obtainable at http://hepwww.physics.yale.edu/www_info/yale_cdf/l1crate.html.

[4] K. Ohl, J. Wahl, P. Wilson *Specification of DAQ/Trigger Synchronization Checking Protocol and VME board Header Word*, CDF Note 3145.

[5] J. Wahl *DIRAC - The Final Specification*, CDF Note 4225.

[6] P. Wilson *J3 Backplane and Transaction Module (L1AUX) for the L1 Calorimeter Trigger Upgrade*, TGN-92.

[7] D. Toback *CrateSum, Upgrade for Run II.*

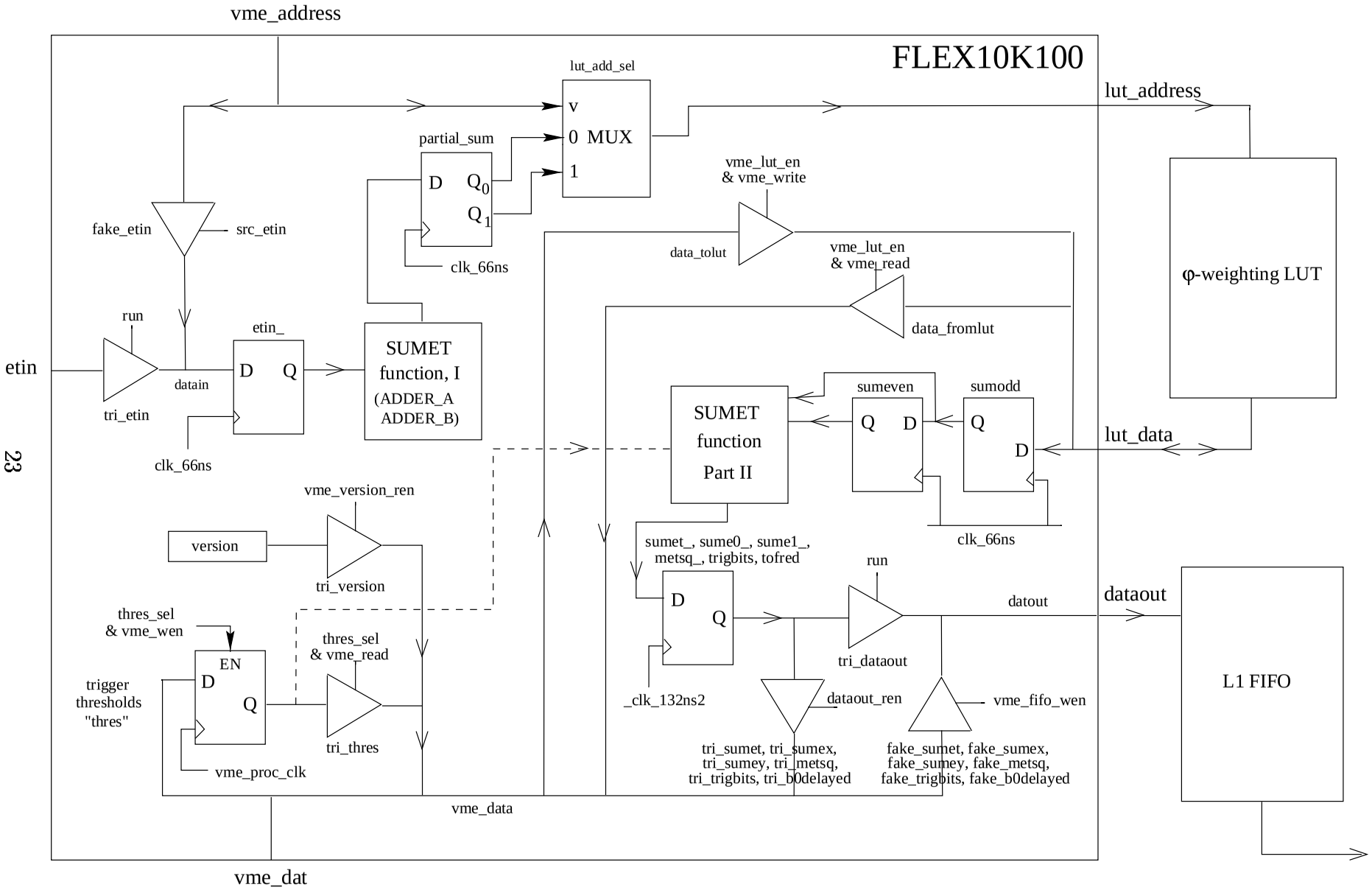
[8] T. Shaw and G. Sullivan *A Standard Front-End and Trigger VMEbus Based Readout Crate for the CDF Upgrade - The CDF Readout Crate*, CDF Note 2388.

[9] J. Wahl *The Anatomy of a DIRAC*, CDF Note 4231.

[10] Altera *Configuring FLEX 10K Devices*, Application Note 59.

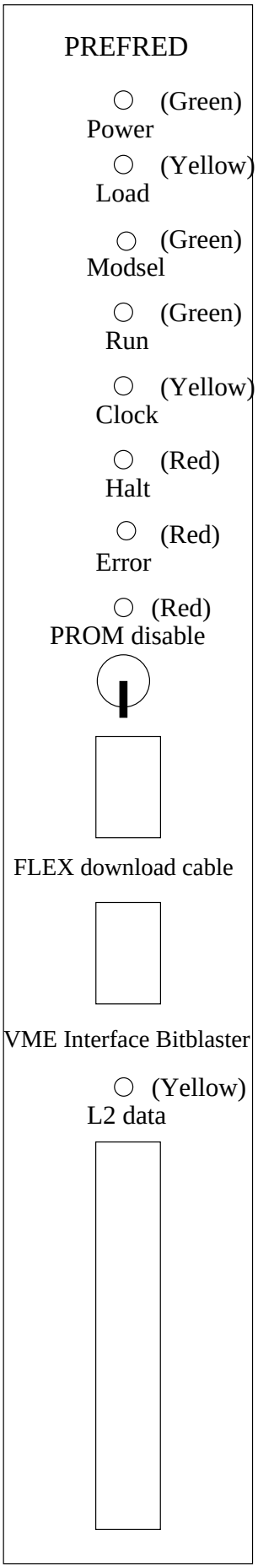
[11] W. Peterson *The VMEbus Handbook, Third Edition*, VFEA International Trade Association, 1993.

5 Appendix



SUMET DATA PROCESSOR INTERFACE TRI-STATE BUFFERS & I/O REGISTERS

PREFRED Front Panel



3 mm LEDs

E101MD1AV2BE SPDT C&K switch

10-pin Header connector

10-pin Header connector

3 mm LED

RPS-80-LM Honda connector

Table 8: The SUMET Control bus. All signals originate from the Control block except the FIFO full/empty status flags and src_config. Signals beginning with an underscore are active low. Missing bit numbers are spares.

Bit #	Name of signal	Description
0	run	Run (1) or load (0) mode status flag
1	lut_add_msb	13th bit of all ϕ-weighting LUTs' address (used for diagnostics); =0 in run mode
2	_reset_	Reset for the L1 FIFOs; when in load mode, activated by a control register
3	vme_proc_clk	Clock that loads the trigger thresholds into the Data Processor at initialization
7:4	_L2B_R[3:0]	DAQ buffer read strobe (subsequent to a L2 accept in run mode); detailed in DAQ Interface section
8	_L1B_W	DAQ buffer write strobe subsequent to a L1 accept; when in load mode, generated by a control register
10:9	L1BA[1:0]	DAQ buffer write address subsequent to a L1 accept; when in load mode, set by control registers
15:14	L2BA[1:0]	DAQ buffer read address (subsequent to a L2 accept in run mode)
16	L1FIFO_R	L1 FIFO read strobe subsequent to a L1A or L1R; when in load mode, generated by a control register
17	L1FIFO_W	L1 FIFO write strobe; when in load mode, can be generated by a VME write request
25:18	_EF[7:0]	Empty status flag for each of the 8 L1 FIFOs
26	src_config	indicates the source that configures the FLEX10K devices: 0 = EPC1 PROMs, 1 = FLEX download cable
27	_bp_trigbits_en	Enable signal for the backplane drivers that send the trigger bits to FRED; =0 in run mode
28	FP_str	Front Panel strobe sent along with valid data to L2, subsequent to a L1 accept
29	vme_fifo_wen	VME write request to all the L1 FIFOs at once (used in diagnostics); spawns a **_L1FIFO_W** strobe
30	src_etin	$E_t(\phi)$ source feeding the Data Processor: 0 = P3 backplane, 1 = VME address bus (fake $E_t(\phi)$)
31	aux_enable	Enables input signals from Aux card; =1 in run mode
34:32	vme_lut_en[2:0]	VME access selector to 1 of 3 pairs of ϕ-weighting LUTs; see Controller section
38	vme_write	Write (1) or read (0) request from VME
39	clk_66ns	Clock with a 66 ns period which is synchronized with the incoming data from the CRATESUMs
41:40	thres_sel[1:0]	VME access selector to 1 of 2 pairs of trigger thresholds; see Controller section
49:44	fred_delay[5:0]	FRED alignment delay; see section 2.2.3
57:50	bunch_cnt[7:0]	Bunch number from bunch counter; see [4]
60:58	_lut_write[2:0]	Write strobe to 1 of 3 pairs of ϕ-weighting LUTs; see Controller section
67:65	clk_132ns[2:0]	Clock with a 132 ns period, echoed twice with 22 ns shifts; see Clock Generator section
69:68	dataout_ren[1:0]	VME read request to the Data Processor output; see Controller section
70	vme_version_ren	VME read request to the Data Processor program version number
71	aux_spare	Spare line running from the Controller to the backplane to control the Aux Card
77:72	config[5:0]	Lines used to configure the FLEX10K devices using a daisy-chain scheme[10]
85:78	_FF[7:0]	Full status flag for each of the 8 L1 FIFOs
86	b0_delayed	B0 signal delayed by B0_offset (active high)
89:87	bp_spare[2:0]	spare bits going from the Controller to the backplane P3 pins that are connected to FRED

5.1 Specifications for loading and testing the SUMET module

5.1.1 Configuring the programmable Altera devices

There are three programmable Altera devices on the PreFRED board:

- Two FLEX10K FPGAs, configured either by PROMs upon power startup or by a download cable connected to the Front Panel (see section 2.3.1).
- One MAX7000S EPLD, which keeps its program upon power outage and therefore does not need reconfiguration at power startup. However, to allow maximum flexibility, that VME interface chip is also connected to the Front Panel for In-System-Programmability (ISP), which allows to download a new program within it from a bit-blaster controlled by a personal computer. This accounts for a 10-bit Front Panel connector in addition to the one used for the FLEX10Ks. It is expected that this ISP connector will never be used, or used only in the very early stage of the board debugging. Immediately after the FLEX10Ks are configured, they are automatically initialized, which means their registers are reset while their I/Os are enabled. Then the devices are available for operation (user-mode). The whole configuration/initialization process takes a negligible amount of time (less than 200 ms); it is not monitored.

5.1.2 VME down-loading

Once the Altera devices are configured and initialized, programmable information needs to be downloaded to the board via VME. There are two types of information: that which should remain constant throughout the lifetime of the system (delays, ϕ-weighting factors), and that which is run-dependent (trigger thresholds). Any information which is "written" to the board should be systematically read back out to check that the VME master accessed the board properly. If The writing/reading failed, the problem may come from the VME Interface, the Controller, the Data Processor or the LUTs depending on the type of transaction requested.

Testing the VME Interface First the VME Interface should be tested by checking that strobe signals indeed circulate in and out of the VME Interface chip. In particular, by attempting a simple read transaction (e.g. that of the Module ID or program version number embedded in the Controller), one should make sure the **_modsel** and **_vme_data_str** signals go active low with a scope. If not, something may be wrong with the chip connections, or, if worse comes to worse, the VME Interface program may need to be changed using the ISP connector. Then the **_ACK** (and subsequent **_DTACK**) produced in response by the Controller should be checked. If **_ACK** is not returned by the Controller as an active low strobe some time after the **_vme_data_str** signal, something may be wrong with the Controller... One should also attempt to read some register from the board using a wrong geographical address or a wrong address modifier, and make sure **_modsel** or any other VME signal is not activated.

Down-loading the control registers All registers in the Altera chips are automatically reset upon start-up, which provides the expected startup default values for the control registers which do not represent programmable delays (i.e. words 3 thru 5 in Table 7). In particular, the run bit is low, which means the board starts up in load mode. The first step

towards making the board operational for run conditions is to load (and read back) the programmable delays (words 0 thru 2 in Table 7). This can be done via a VME block transfer, in which case all 6 writeable control words should be loaded appropriately, the same way as in the board simulation (see Mentor's SUMET board timing diagram). Different delays should eventually be tested in conjuction with an input test board, in particular **CS_delay**, **fred_delay**[5:3] and **fred_delay**[2:0], whose ranges are respectively 0-5, 0-7 and 0-5 (see below).

Testing the DAQ Interface Using the control bits listed in Table 7 (**reset, FIFO_R, L1B_W, L1BA...**), one can initiate a step-by-step reading/writing cycle through the DAQ Interface chain, in order to test the functioning of the L1 FIFOs and DAQ buffers. One first has to reset the FIFOs and operate 4 VME writings to them with, say, all data bits set to 1. Then one should read out the FIFOs and simultaneously write to the DAQ buffers by setting the control bits **FIFO_R, L1B_W** and by providing **L1BA**. This should be done for each of the 4 DAQ buffers. Then the buffers should in turn be read out using VME block transfers and their output matched with all 1's in the relevant fields (see used bits in list of DAQ words, section 2.4.1). The entire procedure should be repeated with all data bits set to 0 instead.

Down-loading the ϕ-weighting LUTs The 6 LUT SRAMs need to be loaded after each power outage. The 12 LSB of the common LUT port address are a mapping of **VME_Address**[13..2], while the port's MSB (bit 13) is **VME_Address**[14] in load mode. In run mode, that bit is always 0. Note that bit 12 of the LUT port is always grounded. The LUTs are loaded in pairs. **VME_Address**[16..15] determines which of the 3 pairs of LUTs is being accessed. Then the VME data bus is shared by the data ports of an even LUT and its consecutive odd LUT (see numbering scheme in Fig. 3). This implies that the LUT_2i (where i=0,1,2) and LUT_2i+1 data buses are concatenated so that the LUT_2i data bits map **VME_Data**[11..0] while the LUT_2i+1 data bits map **VME_data**[23:12]. Some VME-loading routine needs to be written to implement that concatenation. Given an address, the LUT data values per say are calculated through a rather straightforward algorithm which has already been written in FORTRAN, and which can be found on the UCCDF cluster, in cdf$user:[amadon.prefred]lut.for. Down-loading of the LUTs should be performed via successive VME block transfers, each one of which can load 64 24-bit words. It takes therefore 64 BLTs to load the data required for running into one pair of LUTs, or a total of 192 BLTs to load all data. All data should be automatically read back and checked after being loaded.

Down-loading the trigger thresholds The 16-bit trigger thresholds are loaded in pairs into the Data Processor (see Table 6). **VME_Address**[2] = 0 (1) refers to the 2 ΣE_t ($\not\!\!E_t^2$) thresholds. So these need to be concatenated by some VME routine. Again, those thresholds should automatically be read back and checked after being loaded. The first ΣE_t ($\not\!\!E_t^2$) threshold (LSBs of the **VME_data** bus) corresponds to the **TRIGBIT**[0] (**TRIGBIT**[2]) output to FRED. In the same way, the second ΣE_t ($\not\!\!E_t^2$) threshold (MSBs of the **VME_data** bus) corresponds to the **TRIGBIT**[1] (**TRIGBIT**[3]) output. Somehow, the VME routines will have to correlate the thresholds with the trigger bits. To avoid ambiguity, it is specified here that the "first" threshold is the lower one of the pair. Thus if **TRIGBIT**[1] (**TRIGBIT**[3]) is set to 1 by the Data Processor, so is **TRIGBIT**[0] (**TRIGBIT**[2]). Otherwise,

an error occured in the system. In run conditions, such a test may only be carried on at the software level (with the DAQ data), as this was not specified at the time the **_cdf_error** bit was defined.

Data Processor test with constant input: dataout direct VME readout Once the LUTs and thresholds have been loaded, a test of the Data Processor output (**dataout**) can be performed via VME, by faking constant inputs from the CRATESUMs with the **VME_Address** bus (see the part about **dataout** in the Controller section). The following example should be tried out: the **VME_Address** bits have been combined in such a manner that an input address **VME_Address**[23:0] = 0x600094 should yield bf VME_Data = 0xXXFFFFFF (because it makes $\not\!\!E_t^2$ reach its full scale[6]) and, which is more informative about whether the processor works as expected, **VME_Address**[23:0] = 0x608094 should yield bf VME_Data = 0x7FF54FBD, which informs us that $\Sigma E_t = 1981$, $\Sigma E_x = 681$ (i.e. really -169), and $\Sigma E_y = 1023$ (or -511) (the sign bit is interpreted as factor of 2^9 here). Another test is **VME_Address**[23:0] = 0x608004, which should yield bf VME_Data = 0x2A9FFB80, which decodes into $\Sigma E_t = 896$, $\Sigma E_x = 1023$ (or -511), and $\Sigma E_y = 340$. Also **VME_Address**[23:0] = 0x600004 yields bf VME_Data = 0xXXFFFFFF.

Testing the Front Panel and backplane P2 and P3 connectors

5.1.3 Test in run conditions

This can be accomplished first with a test card. Presently, the DIRAC test card sends fake input data every 132 ns. If we wanted to test the SUMET module in real conditions, we would have to make the input stream change every 66 ns. However, we will limit ourselves to the DIRAC test card, but should analyze the results based on two cases: either we consider the transition from one set of input data to the next to be associated with the arrival of a new event, or we consider it to be the transition between even and odd wedge data within the same event. In either dynamic setting, the **dataout** results should be stored in the FIFOs (up to 256 events) before being read out via DAQ buffer write & read VME cycles (the writing part has to be performed with the use of specific control registers, see above). To find out what the results should be, a routine exists which computes the expected values of ΣE_t, ΣE_x, ΣE_y and $\not\!\!E_t^2$ (including saturated results and other peculiar cases), given 2 x 12 10-bit input wedge transverse energies. It can be found on the UCCDF cluster, in cdf$user:[amadon.prefred]sumet.for, and may be incorporated in a larger routine which would determine the expected 2nd and 3rd DAQ words based on the mapping introduced in section 2.4.1. From there, checking the results from one event becomes straightforward. The next step would be to build a higher-level routine which would allow to repeat the checking procedure over many events whose input data may be randomly generated.

Selection of B0_offset First one should find out what **B0_offset** should be from external sources, whether working with a test card or with the CDF detector. Remember that **B0_offset** should include the SUMET latency, i.e. be at least greater than 3. One should first check that the bunch number is zero in the DAQ header word of the first event read out after the beginning of run, and subsequently incremented for the next events. Then one

[6]Also the ΣE_t thresholds are assumed to be smaller than 1981 in that particular instance

needs to check the synchronicity of that bunch number with the CRATESUM data results appearing in DAQ words 2 and 3. That can only be done after a proper determination of **CS_delay**.

Selection of CS_delay In the **CS_delay** determination procedure, **fred_delay** can be set to an arbitrary value. The 6 values of **CS_delay** (0 through 5) should be tried (via VME write transactions) before conclusions can be drawn as to which one to use. The one that yields the expected results with the smallest latency should be picked. However, a large number of events should be tested (see below) with the selected **CS_delay**, to make sure the associated clock (**CS_132ns**) rising edge does not come too early with respect to the setup time constraints of the Data Processor input registers. If frequent or occasional errors are found in the output stream, the next delay should be picked instead.

Selection of fred_delay After **B0_offset** and **CS_delay** have been determined, all 6 **fred_delay** values should be tested to check that we always get the same results, only shifted in time. The final choice will come at the time of integration, based on the processing latencies required by the other PreFRED paths.