

# **CFT/CPS Axial Daughterboard**

# **DFEA**

## **Technical Design Report**

ver 1.00

29 July 2003

Jamieson Olsen  
DZERO Experiment

## 1 INTRODUCTION

The DFEA daughterboard accepts data from the Central Fiber Tracker axial detector and the Central Pre-Shower axial detectors. Together these detectors contain over 40,000 individual fibers. Each fiber is sent to an Analog Front End [AFE] board which discriminates the fiber data; the resulting bits are sent through the MIXER system to organize the data into 80 sectors of 496 bits each. This information is sent to a Digital Front End [DFE] Motherboard, which contains two DFEA daughterboards. There are a total of 80 DFEA daughterboards, one for each sector of the detectors.

Each DFEA daughterboard unpacks the raw data and compares it against a pre-defined set of track equations. To keep the device size down the track equations are grouped by their transverse momentum into four groups: MAX, HIGH, MEDIUM, and LOW. Each group of track equations goes into its own Xilinx Virtex Field Programmable Gate Array [FPGA]. Within each FPGA the six highest Pt tracks are found and sent back to a fifth (“backend”) FPGA for further processing. Additionally, the track finder FPGAs find clusters in the CPS axial data and report those to the backend FPGA.

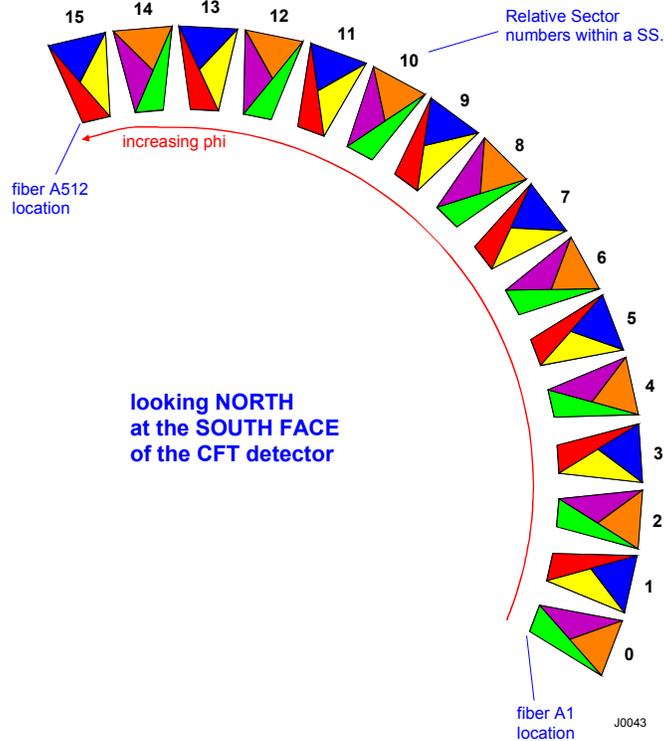
The backend FPGA is responsible for matching CFT tracks to CPS clusters (and vice versa), formatting the output records in accordance with the **DFE protocols**<sup>1</sup>. L1 records are very simple and contain counts of the found tracks and whether or not those tracks match to clusters. L2 records contain the actual tracks and clusters themselves. The backend FPGA stores up to 64 events worth of tracks and clusters into a pipeline, so that when a L1\_ACCEPT control signal occurs the DFEA jumps back to the appropriate event and generates L2 records containing the tracks and clusters.

A new event arrives at the DFEA every 132ns.

## 2 CFT/CPS AXIAL DETECTOR GEOMETRY

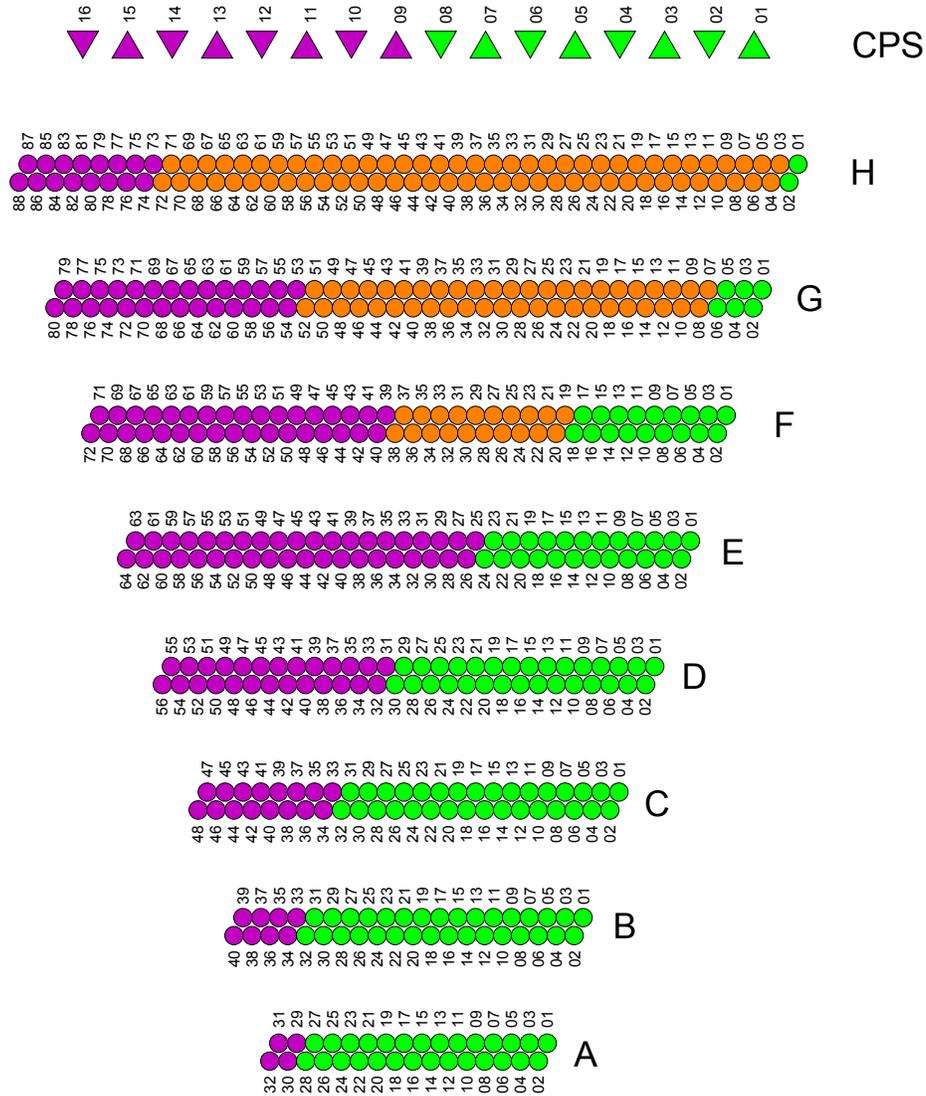
The detector is comprised of 80 sectors numbered 0-79. Sector 0 starts at  $\phi = 0$ .  $\phi$  increases in the counter-clockwise direction when the detector is viewed from the south, looking north along the negative z axis. Sector numbering increases with  $\phi$ .

There are five super-sectors, each containing 16 sectors, as shown below.

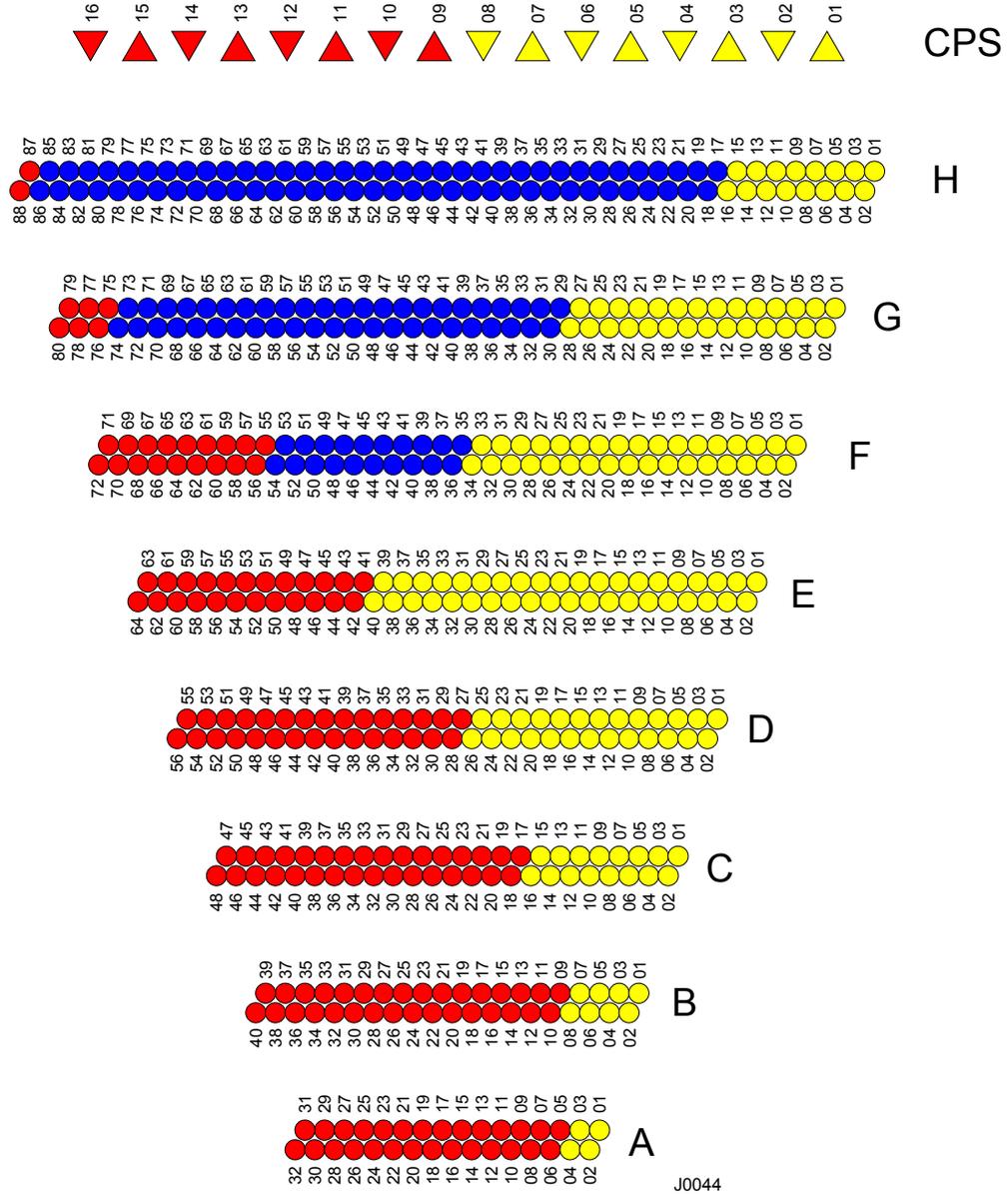


Each sector has 480 CFT siglet fibers and 16 CPS axial strips. Three LVDS links supply the data for one sector. If the relative sector number is odd the links are GREEN, PURPLE, and ORANGE. If the relative sector is even the links are RED, BLUE and Yellow.

Here's a detailed picture of an EVEN relative sector:

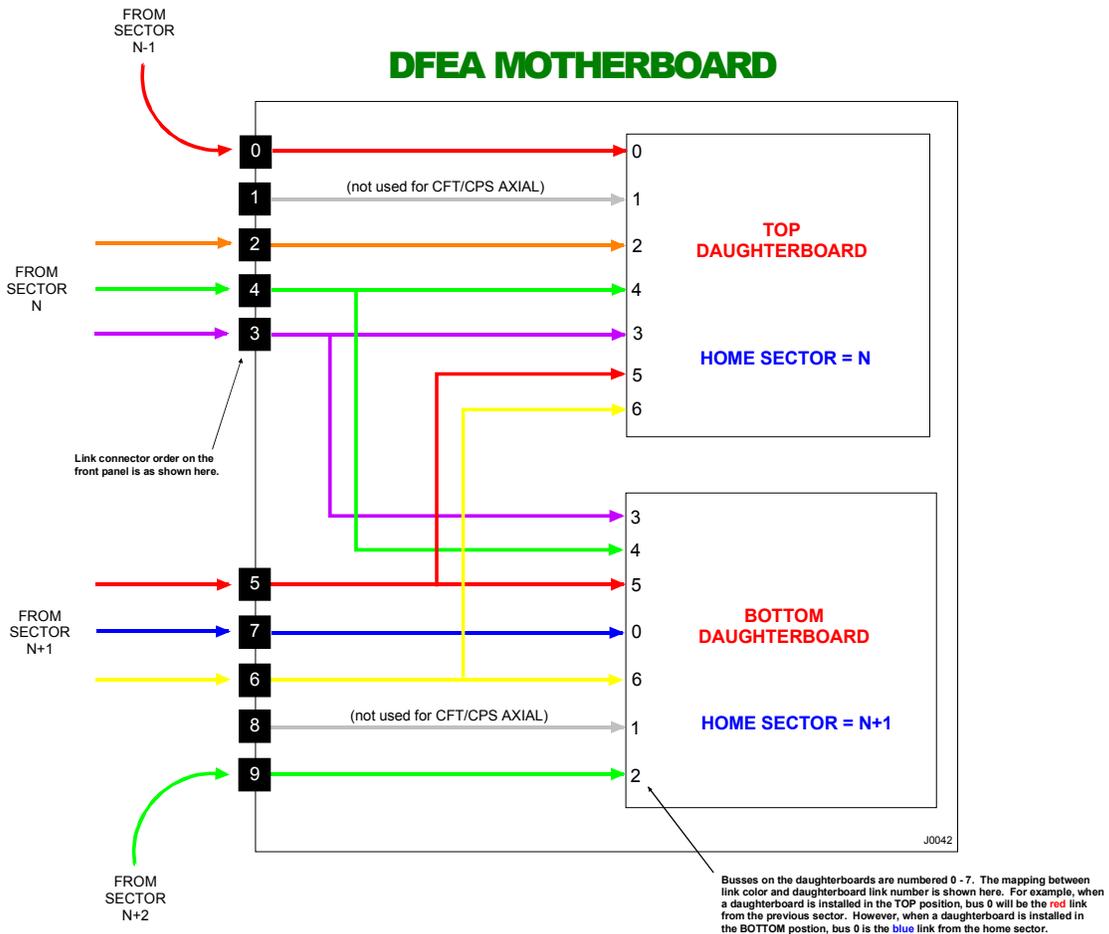


And the ODD relative sectors look like this:



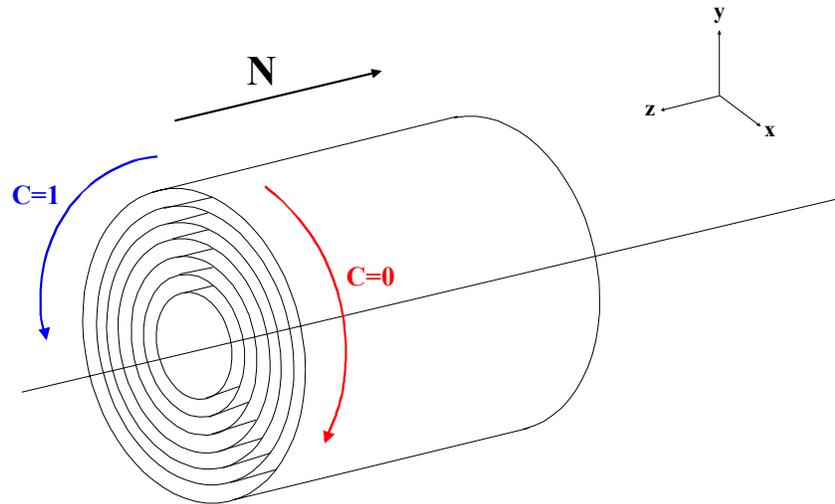
## 2.1 DFE MOTHERBOARD SHARING

Each home sector must see a portion of its adjacent sector's fibers. Some of this sharing is accomplished using redundant GREEN and RED Mixer outputs. The rest of the fiber sharing occurs on the DFE Motherboard. For example, if the DFEA daughterboard is in the top position it's home sector fibers are delivered on ORANGE, GREEN, and PURPLE links. However, it needs to see a portion of the previous sector; these fibers are delivered by the RED link coming in from the Mixer. It also needs to see a portion of the next sector; these fibers are delivered by the RED and YELLOW link copies.



## 2.2 CFT TRACK CURVATURE

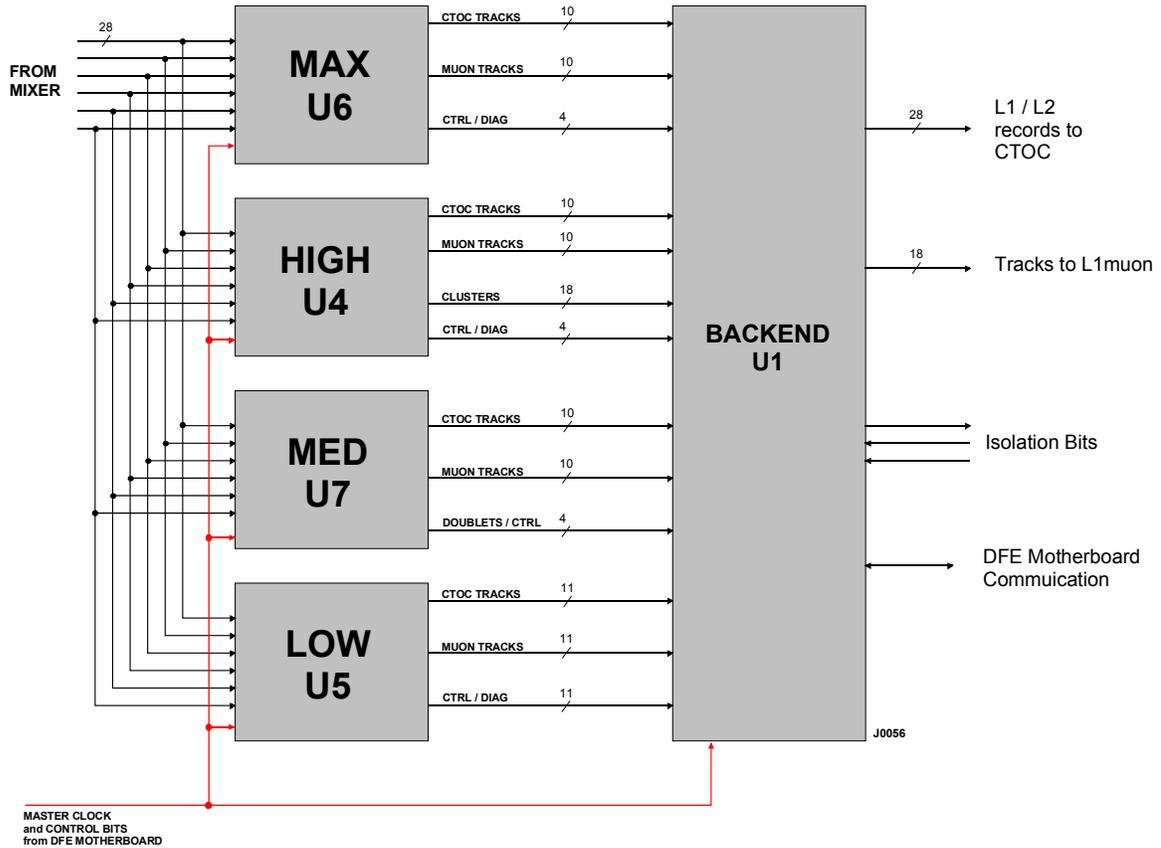
Tracks curving in the increasing phi direction have their Curvature bit (c) set to 1. This convention is used throughout the DFEA firmware and applies to all CTOC and L1muon tracks.



In this diagram sector 0 is on the EAST side of detector, along the +x axis.

### 3 DFEA DATAFLOW

The picture below shows data paths in the DFEA daughterboard. The Mixer sends data to each DFEA over six LVDS channel-links; each link is 28-bits wide and contains raw CFT/CPS bits and various control signals. The aggregate data rate into each DFEA is 8.9Gbps.



The LVDS channel links terminate on the DFE motherboard. The motherboard de-serializes the data back into a 28-bit wide bus and drives these busses up onto the daughterboard. The motherboard also extracts a 53MHz clock and some of the system control signals (i.e. SYNC\_GAP, FIRST\_XING, etc.) from one of the channel links (the default is input #2) and conditions these before driving them up onto the daughterboard.

Each track finder has two track output busses. One bus has tracks destined to go to the L1muon hardware; the other has tracks which will be used for building up CTOC L1/L2 records. The L1muon track serialization algorithm in each track finder FPGA is “faster but lossy” compared to the CTOC track algorithm. This was done to reduce the DFEA latency for L1muon – more on the differences in the algorithm later.

## 4 TRACK FINDER FPGA FUNCTIONS

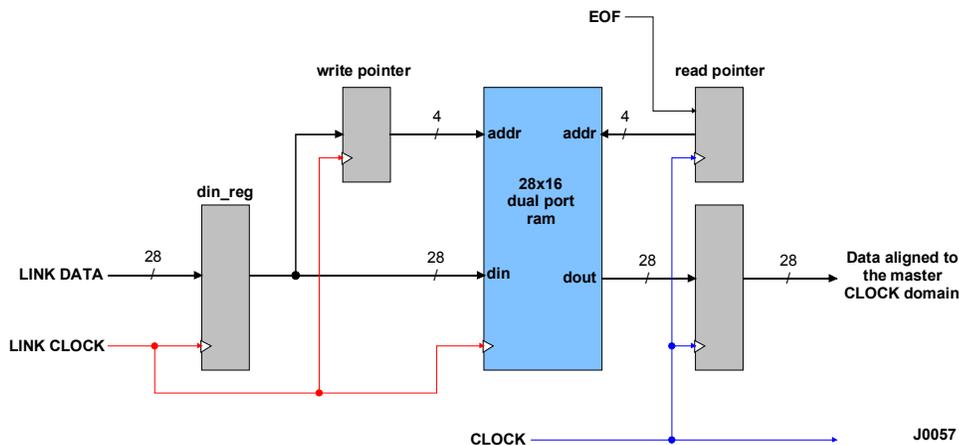
Four FPGAs are used to accept data and control signals from the Mixer system. The primary purpose of these chips to find the six highest Pt tracks in each set of equations. As previously mentioned the track equations are grouped into four Pt bins: MAX, HIGH, MED, and LOW. The lower the Pt range, the more resources are needed to find tracks. MAX, HIGH and MED track finders fit into Xilinx Virtex 400 devices; while the LOW track finder requires a Virtex 600. Additionally, some of the track finder FPGAs have other functions such as counting the number of fibers hit and finding CPS Axial Clusters. The track finder chips are always running in L1 mode – they do not care if the system is in L2 mode, or if a CFT\_RESET has been issued, etc. Every 132ns a new set of data arrives and they look for tracks.

### 4.1.1 FRONT END

The front end of the track finder FPGAs has two main purposes: to synchronize links input links into one clock domain and to unpack the CFT/CPS axial bits into a format that the track equations can use.

#### 4.1.1.1 Link Synchronization

Each LVDS link receiver generates its own clock; the timing of the receiver's 28-bit output bus is referenced to this clock. For a given DFEA there are 6 LVDS input links (three from the home sector and three from the neighbour sectors). Each link is multiplexed by 7 – therefore each LVDS link can supply a maximum  $28 \times 7 = 196$  bits for raw CFT/CPS data and control bits. On each link the least significant bit is reserved for the End of Frame (EOF) marker. This bit position is zero except for the 7<sup>th</sup> (last) timeslice where it is set. This bit is used to synchronize the links.



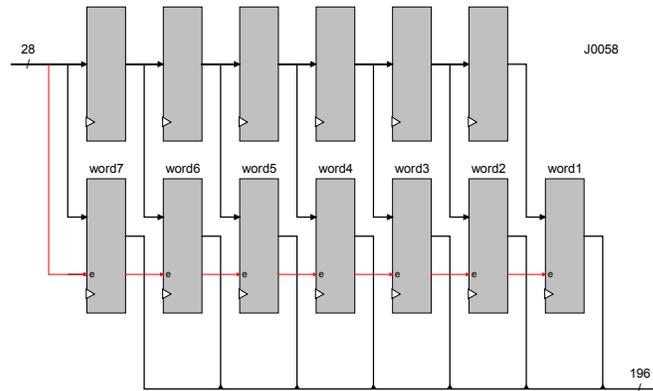
The figure above shows a single link deskew circuit. A small dual port ram is used to cross clock domains. On the link side the 28-bit data bus is registered using the link clock. If the least significant bit is set then `din_reg(27..0)` contains the last word of a frame. Then on the next link clock this word will be clocked into the ram and the write pointer will be reset to 0. This insures that the first word of a frame is always stored in ram address 0.

Reading from the ram is done in the master clock domain. A read pointer pulls data out of the ram, and the output data is registered before making the synchronized data available to the rest of the chip. The key is keeping the read pointer a trailing a few counts behind the write pointer at all times.

The DFEA motherboard has a small CPLD that looks at input links 2 and 7. It strips control bits off of one of these links, delays them appropriately, and distributes them to all FPGAs on the two DFEA daughterboards. The track finder chips only care about one control bit called EOF. It is a copy of link2's (or link7's) EOF bit delayed by a couple of clock ticks. This control bit is aligned with the master clock. When the track finder chips see this master EOF they reset their read pointers on the next rising edge of the master clock. Thus all link deskew modules read pointers are always identical and they're always trailing the write pointers by a couple of clock ticks. The maximum LVDS link skew this scheme can tolerate is  $\pm$  one or two clock ticks.

#### 4.1.1.2 Link Unpacking

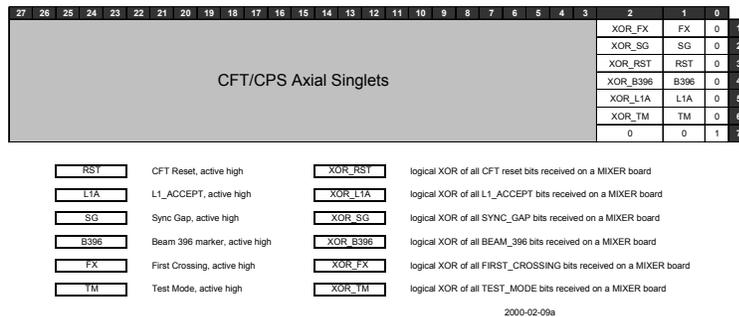
Now that the link data is in the master clock domain it needs to be flattened into one long array of 196 bits. This is done because the track finder algorithm is largely parallel and needs to see all of the bits in the home sector and neighbouring sectors at the same time.



The above picture shows the unpacker for one link. The top row of shift registers accepts data at 53Mhz directly from the link deskew circuit output. When the 7<sup>th</sup> word of the frame is present on the deskew circuit output bus the bottom row of flops loads all 196 bits and holds them for 132ns before the next load occurs. The least significant bit (0) in word1 becomes bit 0 in the flattened array; the most significant bit (27) in word7 becomes the most significant bit in the flattened array. Since all unpackers are running in the master clock domain, all unpackers update their outputs at the same time.

### 4.1.1.3 Control bit format

Input links 2 and 7 contain not only the EOF marker, they also provide other system control bits such as FirstCrossing (FX), SyncGap (SG), CFT\_RESET (CR), L1\_ACCEPT (L1A) and TEST\_MODE (TM). There is one spare bit that is currently unused. These control bits originate from the serial command link (SCL) receiver. From there they go to the sequencer controller, then to the sequencer. The sequencer sends the control bits to the AFE, which sends them to the mixer along with the raw CFT/CPS discriminator bits. The mixer reorganizes the data bits and ORs the control bits from the different links. The mixer then sends the data to the DFEA motherboards. The mixer also generates an XOR of the control bits, which in theory the DFEA could look at to see if some of the control bits were in disagreement. Currently the DFEA ignores the XORed control bits. Control bit format for links 2 and 7 is shown below:



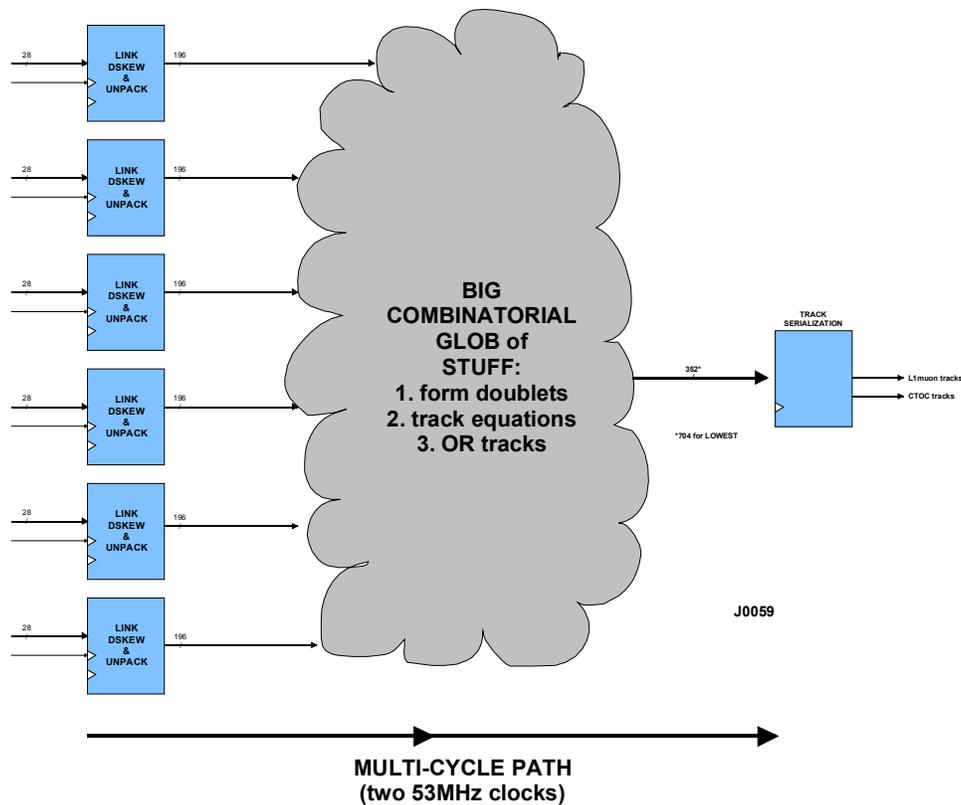
## 4.1.2 TRACK EQUATIONS

A CFT track is defined as a doublet “hit” on each of 8 CFT axial layers. Tracks are characterized by their phi value (H layer doublet index) and Pt value (degree of curvature). Individual tracks with similar Phi and Pt values are grouped together. In hardware terms each track equation is an 8-wide AND; similar tracks are ORed together. Typically the number of ORed track equations is between 3 and 16, depending on the Pt value.

Because the number of tracks is so large, they were partitioned into four Pt “bins”: Max (2400 tracks), High (2400 tracks), Medium (3300 tracks), and Low (8300 tracks). Each found track has a two bit field to denote which Pt bin it came from. Max=00, High=01, Medium=10 and Low=11.

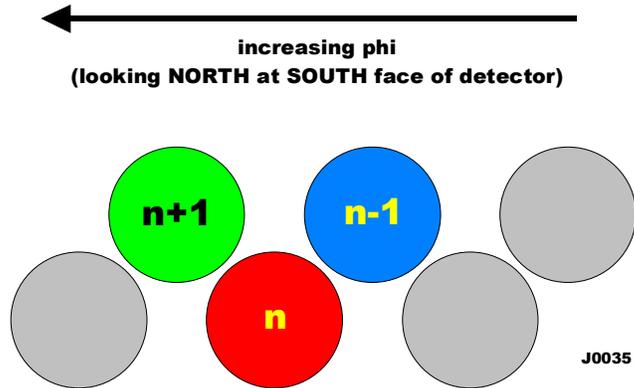
Within each Pt bin tracks are grouped into four “sub-bins”. The Low design needs more Pt resolution, so in that case there are eight “sub-bins”. When a track is found the sub-bin information goes into a 3-bit field called Extended Pt. The lower the extended Pt number, the higher the Pt value within each Pt bin.

A majority of the logic in the track finder FPGAs is devoted to the track equations. This operation must be done in parallel to keep the DFEA latency to a minimum. The result is that the equations become one large “glob” of purely asynchronous combinatorial logic. (FPGA architecture is really designed for synchronous circuits, as a result this section of the design takes a long time in place-and-route.) Because this glob of logic is so large, it takes two 53MHz clock cycles to get through it.



### 4.1.2.1 Doublet Formation

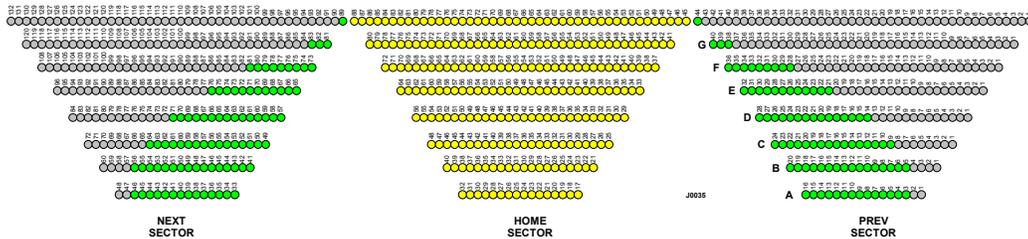
The raw CFT information supplied to the DFEA is individual fiber hits, called singlets. In order to reduce the size of the equations singlets are converted to doublets. Three singlets are involved in the doublet calculation:



$$\text{doublet}[k] = [\text{NOT singlet}(n-1) \text{ AND singlet}(n)] \text{ OR singlet}(n+1)$$

Using doublets in the track equations also helps eliminate fake tracks caused by particles lighting up adjacent fibers in the detector.

The track equations expect the doublets to be grouped by layer and indexed starting from 1 in the previous sector. This is where the fiber sharing across sector boundaries comes into play. The home sector needs to see a portion of the previous and next sectors' doublets. The picture below shows the home sector and the previous and next sector doublets. In order to see tracks with Pt down to 1.5 GeV the home sector must see all of its doublets, plus the colored doublets in the neighbouring sectors:



Each link coming from the Mixer arranges the CFT and CPS data differently, thus each DFEA doublet former is unique. To avoid mistakes, a Microsoft Access database was created to automate this procedure. The database (and Visual Basic macros) produce the unique DFEA doublet unpacker VHDL files.

### 4.1.2.2 Matrix Mapping

The output of the track equation logic is a 352-bit wide bus. Each bit in this bus corresponds to a found track. In the Max, High, and Medium track finders there are four extended Pt bins, and each extended Pt bin has a positive and negative curvature. Thus the 352-bit bus can be thought of as a two dimensional array or matrix where the columns are phi values (0-43) and the rows represent extended Pt and curve bits:

43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	C	Pt	ExpPt
43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	00	000
87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	0	00	000
131	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	1	00	001
175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	0	00	001
219	218	217	216	215	214	213	212	211	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	1	00	010
263	262	261	260	259	258	257	256	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230	229	228	227	226	225	224	223	222	221	220	0	00	010
307	306	305	304	303	302	301	300	299	298	297	296	295	294	293	292	291	290	289	288	287	286	285	284	283	282	281	280	279	278	277	276	275	274	273	272	271	270	269	268	267	266	265	264	1	00	011
351	350	349	348	347	346	345	344	343	342	341	340	339	338	337	336	335	334	333	332	331	330	329	328	327	326	325	324	323	322	321	320	319	318	317	316	315	314	313	312	311	310	309	308	0	00	011

Maximum Matrix, 352 Elements

43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	C	Pt	ExpPt
43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	01	000
87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	0	01	000
131	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	1	01	001
175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	0	01	001
219	218	217	216	215	214	213	212	211	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	1	01	010
263	262	261	260	259	258	257	256	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230	229	228	227	226	225	224	223	222	221	220	0	01	010
307	306	305	304	303	302	301	300	299	298	297	296	295	294	293	292	291	290	289	288	287	286	285	284	283	282	281	280	279	278	277	276	275	274	273	272	271	270	269	268	267	266	265	264	1	01	011
351	350	349	348	347	346	345	344	343	342	341	340	339	338	337	336	335	334	333	332	331	330	329	328	327	326	325	324	323	322	321	320	319	318	317	316	315	314	313	312	311	310	309	308	0	01	011

HIGH Matrix, 352 Elements

43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	C	Pt	ExpPt
43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	10	000
87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	0	10	000
131	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	1	10	001
175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	0	10	001
219	218	217	216	215	214	213	212	211	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	1	10	010
263	262	261	260	259	258	257	256	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230	229	228	227	226	225	224	223	222	221	220	0	10	010
307	306	305	304	303	302	301	300	299	298	297	296	295	294	293	292	291	290	289	288	287	286	285	284	283	282	281	280	279	278	277	276	275	274	273	272	271	270	269	268	267	266	265	264	1	10	011
351	350	349	348	347	346	345	344	343	342	341	340	339	338	337	336	335	334	333	332	331	330	329	328	327	326	325	324	323	322	321	320	319	318	317	316	315	314	313	312	311	310	309	308	0	10	011

MEDIUM Matrix, 352 Elements

The Low track finder has more equations and requires finer extended Pt resolution. Its output bus is 704 bits wide, arranged as a 44x16 matrix. There are eight extended Pt bins:

43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	C	Pt	ExpPt
43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	11	000
87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	0	11	000
131	130	129	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	1	11	001
175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	0	11	001
219	218	217	216	215	214	213	212	211	210	209	208	207	206	205	204	203	202	201	200	199	198	197	196	195	194	193	192																			

### 4.1.3 CFT TRACK SERIALIZATION

Now that the CFT tracks are arranged in a matrix, the next step is to select the six highest Pt tracks in each Pt bin and pass them on to the backend chip (U1) for processing. A priority encoder cannot be used on the matrix itself because of the large number of bits – the logic resources would be huge. The DFEA track serializer does the priority encoding in two steps: coarse and fine. The serialization is done synchronously in a pipeline, so that logic is not duplicated for each track – the result is very small and runs easily at 53MHz. The “coarse” priority encode involves subdividing the matrix into 1x22 blocks:

44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
BLOCK 1											BLOCK 0 (HIGHEST PRIORITY)																																
BLOCK 3											BLOCK 2																																
BLOCK 5											BLOCK 4																																
BLOCK 7											BLOCK 6																																
BLOCK 9											BLOCK 8																																
BLOCK 11											BLOCK 10																																
BLOCK 13											BLOCK 12																																
BLOCK 15 (LOWEST PRIORITY)											BLOCK 14																																

Block Structure for MAX, HIGH, and MEDIUM Matrix

44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
BLOCK 1											BLOCK 0 (HIGHEST PRIORITY)																																
BLOCK 3											BLOCK 2																																
BLOCK 5											BLOCK 4																																
BLOCK 7											BLOCK 6																																
BLOCK 9											BLOCK 8																																
BLOCK 11											BLOCK 10																																
BLOCK 13											BLOCK 12																																
BLOCK 15											BLOCK 14																																
BLOCK 17											BLOCK 16																																
BLOCK 19											BLOCK 18																																
BLOCK 21											BLOCK 20																																
BLOCK 23											BLOCK 22																																
BLOCK 25											BLOCK 24																																
BLOCK 27											BLOCK 26																																
BLOCK 29											BLOCK 28																																
BLOCK 31 (LOWEST PRIORITY)											BLOCK 30																																

Block Structure for LOW Matrix

The six most important blocks are extracted from the matrix one at a time and passed down a pipeline. The pipeline knows what block number it was given, so it can determine the Pt, Extended Pt, Curve, and Phi of any tracks it may find inside the block. The “fine” priority encoder looks inside each block for individual tracks. First the block is presented to two priority encoders. One priority encoder scans from left to right across the block, and the other priority encoder scans the other way across the block. If there are two or more tracks in the block the two priority encoders will return the leftmost and rightmost tracks in the block.

#### 4.1.3.1 L1CTOC vs. L1muon Serialization Algorithms

In the case where the leftmost track and the rightmost track in a block are different, the pipeline expands and reports the leftmost track first, followed by the rightmost track. Then it proceeds to process the next block. This takes a few clock cycles for FIFO memory overhead to handle this pipeline expansion and contraction reliably.

L1Muon needs tracks very quickly, and every clock cycle counts. Rather than wait a few more clock cycles to resolve the leftmost/rightmost tracks, L1Muon always grabs the leftmost track and ignores the rightmost track in a block. This leads to lost tracks, but the latency is less and that’s the most important thing. This is why there are two track busses coming out of each CFT track finder FPGA: one is for “early” L1muon tracks, and the other bus is for the “regular” tracks destined for the CTOC/STOV/STSX boards.

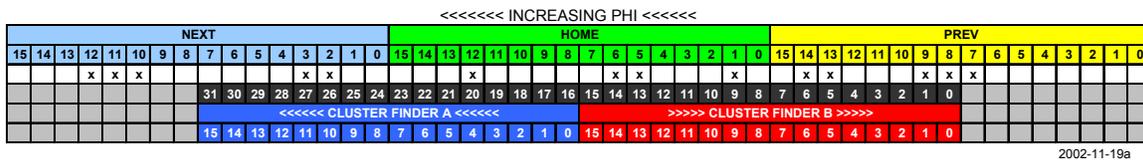
## 4.2 OTHER FUNCTIONS

In addition to finding CFT Axial tracks all four track finder FPGAs have secondary functions. The Maximum FPGA (U6) contains input link diagnostics; the High FPGA (U4) contains the input link diagnostics and the CPS Axial cluster finder; and the Medium FPGA (U7) contains logic to count the number of “hit” doublets; the Low FPGA (U5) contains input link diagnostics. All of this extra information is passed back to the backend FPGA (U1) for processing into L1/L2 records. This section describes these extra functions.

### 4.2.1 CPS AXIAL CLUSTER FINDER

Like the CFT fibers, the CPS fibers run parallel to the beam axis; however the number of CPS axial fibers is much smaller – only 16 per sector. CPS Axial fibers are split into North and South segments, however these ORed back together on the AFE board – as far as the DFE is concerned there is only one bit per CPS axial strip. Like the CFT, the CPS Axial cluster finder can “see” data from each of the neighbouring sectors: there are 16 CPS strips in the home sector plus 8 CPS strips in each of the neighboring sectors.

In the High FPGA (U4) there is a module built into the front end that extracts the CPS axial bits from the incoming data. The bits are flattened into an array of 32 bits, shown in the black box below:



These 32 bits are all of the CPS axial bits that the home sector can see. Now the clusters must be identified. Starting at one end of the array and scanning across it introduces a phi bias, so the solution is to build two separate cluster finder machines that start at the middle of the home sector and scan outward in either direction. Each cluster finder reports the first four clusters that it sees. Clusters are described by their centroid and width. Cluster finder A scans outwards in the increasing phi direction; cluster finder B scans outward in the decreasing phi direction. If a wide cluster is located in the center of the home sector it will be reported as two smaller clusters – the overlap is *not* resolved in the DFEA.

Each cluster finder can report clusters up to 16 bits wide. However, the width of the cluster finder output bus limits the centroid field to 4 bits. So in the example above cluster finder B reports four clusters, and finder A reports two clusters:

Centroid	Width	Centroid	Width
4	1	13*	2
10*	2	9	1
		5*	2
		0*	2

\* If a cluster is an even number of CPS strips wide the centroid is reported as the lower CPS strip.

The backend FPGA (U1) needs to have the cluster centroids listed as a 5-bit number (0-31), so when A clusters arrive it adds 16 to the centroid field.

### 4.2.1.1 Cluster Finder Logic

Each cluster finder deals with an array of 16 bits. Scanning across it one bit at a time would require 16 clocks, which would increase the DFEA latency. Alternatively looking at all 16 bits in parallel is very fast, but it requires too much logic and the operation would be largely asynchronous – which makes the design less stable. The cluster finder circuit used here is a very small completely synchronous pipeline. The first step is to do an edge detect on the raw data:

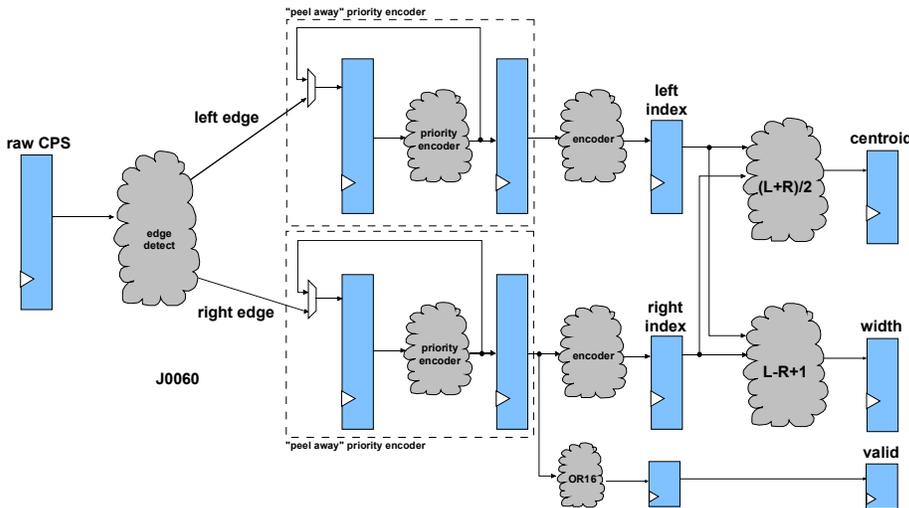
<<<<< SCAN DIRECTION <<<<<

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			1	1	1						1				
					1						1				
			1								1				

raw CPS hits  
right edge  
left edge

2002-11-19a

Once per crossing the left and right edge information is loaded into a pair of “peel away” priority encoders. These modules operate just like a regular priority encoder, but with each clock tick the most significant bit is destroyed, potentially revealing more less-significant bits underneath. The output of these encoder modules is one-hot, so it needs to be binary encoded into a four bit number. At this point there are two 4-bit numbers representing the left and right edges of the cluster. With every clock tick these registers change to reflect the next set of edges in the raw data. To calculate the centroid, simply add the numbers together and divide by two (bit shift right). To calculate the width, subtract the right index from the left and add 1. Below is a diagram of the cluster finder pipeline.



This pipeline is capable of finding more than four clusters, however the backend FPGA only accepts the first four clusters from each finder module. The pipeline also zero compresses the data – all found clusters are reported in order with no gaps in between.

### 4.2.1.2 Cluster Bus Format

An 18-bit cluster bus connects U4 to the backend FPGA U1 on the DFEA.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
v	Width				Centroid				v	Width				Centroid			

## 4.2.2 OCCUPANCY/DOUBLET COUNT

The Medium FPGA (U7) counts up the number of hit doublets in the home sector. This number can range from 0 to 239 and is reported as a unsigned 8-bit number. The backend FPGA uses this count in L1 record it produces.

Home sector doublets are grouped into six strings of 40 bits. Each 40-bit string is partitioned into ten 4-bit chunks. The bits in each chunk are counted (using a small LUT). The ten 4-bit counts are added using multiple cascaded adders until a 6-bit sum is generated. The six 6-bit sums are registered, and sent to an accumulator serially. The final result is an 8-bit count of the number of “hit” doublets in the home sector.

There are only three bits available to send the doublet count back to the backend FPGA, so the count is sent back two bits at a time, requiring four clock cycles. The most significant bits are sent first.

## 4.2.3 INPUT LINK DIAGNOSTICS

With the exception of the Medium FPGA, all track finder FPGAs have link diagnostic modules built in. The Maximum FPGA has a module called *sync\_detect* that looks for the EOF bit on the LSB of each input link. If the EOF bit is not present on one out of seven clock ticks *sync\_detect* sends a status bit back to the backend FPGA.

The Mixer has a test mode where it sends a walking 1's pattern to the DFEAs. Each DFEA daughterboard sees six links from the Mixer; each DFEA input link has a small module that looks for the Mixer test pattern. If the received pattern has an error the corresponding checker module sends a status bit back to the backend FPGA. Space is limited, so these pattern checker modules are scattered across the Max, High and Low track finder FPGAs.

The backend FPGA continuously collects these seven status bits and sends them down to the DFE motherboard, where they are pulse stretched and made accessible to the 1553 network via the DFE backplane and the DFE crate controller. All of these modules continuously monitor the DFEA input links and send status bits back to the backend FPGA.

Using this feature a very powerful test mode was developed where the Mixer turns on the test pattern on one output link, and the associated DFEA board is queried to see that it received that pattern error free. This way the 320 links between the Mixer and DFEA can be checked out in minutes.

### 4.3 OUTPUT FORMAT AND TIMING

The busses connecting the track finder FPGAs to the backend FPGA were originally designed for a single 16-bit track bus and a general purpose 8-bit status bus. Due to the fact that two individual track busses are needed for each track finder FPGA, the actual bus assignments differ quite a bit from the busses as they appear on the schematics. The tables below show the bus lines and their new assignments:

MAXIMUM (U6)		HIGH (U4)		MEDIUM (U7)		LOWEST (U5)	
max_track(15)	valid	high_track(15)	valid	med_track(15)	valid	low_track(15)	valid
max_track(14)	curve	high_track(14)	curve	med_track(14)	curve	low_track(14)	curve
max_track(13)	ext_pt(1)	high_track(13)	ext_pt(1)	med_track(13)	ext_pt(1)	low_track(13)	ext_pt(2)
max_track(12)	ext_pt(0)	high_track(12)	ext_pt(0)	med_track(12)	ext_pt(0)	low_track(12)	ext_pt(1)
max_track(11)	phi(5)	high_track(11)	phi(5)	med_track(11)	phi(5)	low_track(11)	ext_pt(0)
max_track(10)	phi(4)	high_track(10)	phi(4)	med_track(10)	phi(4)	low_track(10)	phi(5)
max_track(9)	phi(3)	high_track(9)	phi(3)	med_track(9)	phi(3)	low_track(9)	phi(4)
max_track(8)	phi(2)	high_track(8)	phi(2)	med_track(8)	phi(2)	low_track(8)	phi(3)
max_track(7)	phi(1)	high_track(7)	phi(1)	med_track(7)	phi(1)	low_track(7)	phi(2)
max_track(6)	phi(0)	high_track(6)	phi(0)	med_track(6)	phi(0)	low_track(6)	phi(1)
max_track(5)	valid	high_track(5)	valid	med_track(5)	valid	low_track(5)	phi(0)
max_track(4)	curve	high_track(4)	curve	med_track(4)	curve	low_track(4)	valid
max_track(3)	ext_pt(1)	high_track(3)	ext_pt(1)	med_track(3)	ext_pt(1)	low_track(3)	curve
max_track(2)	ext_pt(0)	high_track(2)	ext_pt(0)	med_track(2)	ext_pt(0)	low_track(2)	ext_pt(1)
max_track(1)	phi(5)	high_track(1)	phi(5)	med_track(1)	phi(5)	low_track(1)	ext_pt(1)
max_track(0)	phi(4)	high_track(0)	phi(4)	med_track(0)	phi(4)	low_track(0)	ext_pt(0)
max_stat(7)	phi(3)	high_stat(7)	phi(3)	med_stat(7)	phi(3)	low_stat(7)	phi(5)
max_stat(6)	phi(2)	high_stat(6)	phi(2)	med_stat(6)	phi(2)	low_stat(6)	phi(4)
max_stat(5)	phi(1)	high_stat(5)	phi(1)	med_stat(5)	phi(1)	low_stat(5)	phi(3)
max_stat(4)	phi(0)	high_stat(4)	phi(0)	med_stat(4)	phi(0)	low_stat(4)	phi(2)
max_stat(3)	link_sync_err	high_stat(3)	link3_perr	med_stat(3)	reserved for doublet count	low_stat(3)	phi(1)
max_stat(2)	link0_perr	high_stat(2)	link4_perr	med_stat(2)		low_stat(2)	phi(0)
max_stat(1)	link2_perr	high_stat(1)	link5_perr	med_stat(1)		low_stat(1)	link6_perr
max_stat(0)	start	high_stat(0)	start	med_stat(0)	start	low_stat(0)	start

2002-08-05a

The yellow cells are the L1CTOC track bus, and the green cells are the L1muon track bus. There are no changes to U4's 18-bit cluster bus. The orange cells are input link status bits.

The start bit is set to mark the first L1muon track. The backend chip uses the start bit as its timing reference when collecting all of the data from the four track finder FPGAs.

L1muon CFT tracks come out first; with a total latency of 14 clocks (263ns). The L1CTOC tracks begin to come out 4 clock cycles later, with a latency of 338ns. The first cluster comes out 1 clock cycle after the start bit. Doublets are shifted out two bits at a time (MSb first) on a 3-bit bus. The MSb of the 3-bit doublet bus is set for four clock cycles to indicate that the lower two bits contain a valid doublet count.

All output busses are synchronous to the DFE board master clock. The table below shows the relative timing of the output busses as referenced to the start bit. The yellow cells show the data for one event.

Start	1							1								1
L1muon tracks	T1	T2	T3	T4	T5	T6		T1	T2	T3	T4	T5	T6		T1	T2
L1ctoc tracks		T4	T5	T6		T1	T2	T3	T4	T5	T6		T1	T2	T3	T4
Clusters			C1	C2	C3	C4			C1	C2	C3	C4				
Doublets	D3	D4					D1	D2	D3	D4					D1	D2

2002-08-05

Tracks and clusters are always zero compressed into the beginning time slots. For example, if only three tracks were found they would show up in slots T1, T2, and T3.

## 5 BACKEND FPGA [U1]

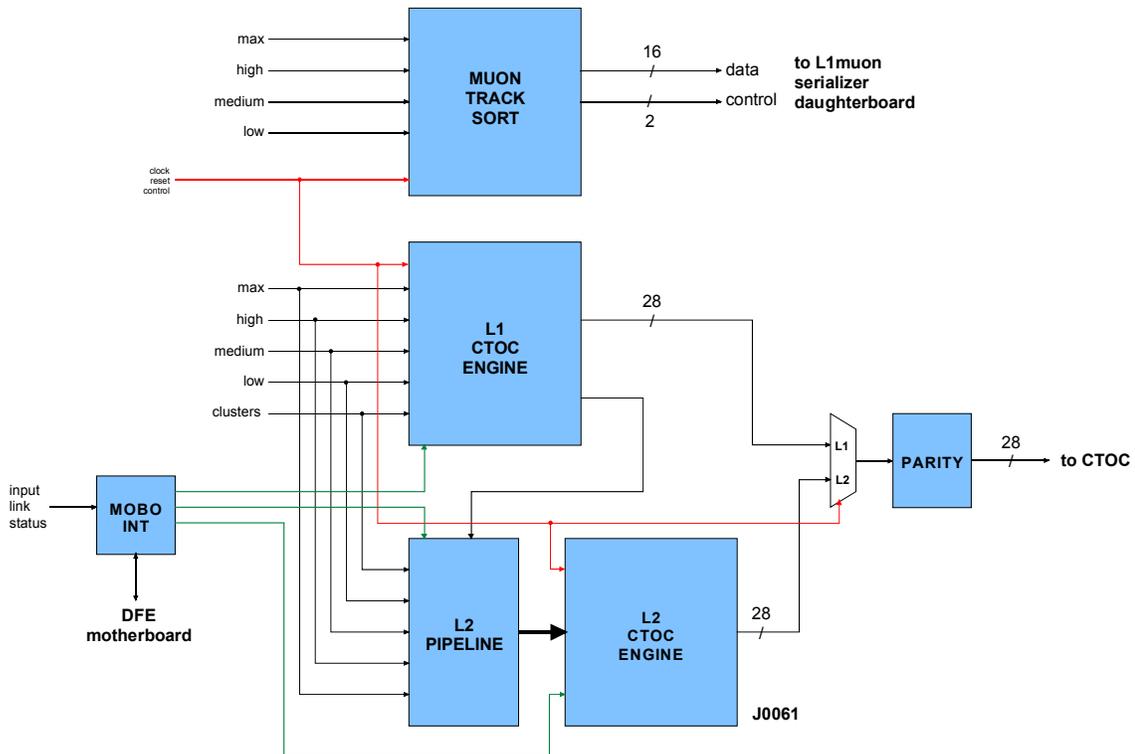
### 5.1 INTRODUCTION

The backend FPGA is the smallest device on the DFEA daughterboard (XCV300-4BG432) but it is also the most complex. It collects all of the data from the CFT track finders once per crossing and it uses that data to produce the following:

- **L1Muon tracks.** The six highest Pt tracks are selected, zero compressed, and sent to L1muon. This is the most critical path, so latency must be kept to an absolute minimum.
- **L1CTOC record.** CFT tracks and CPS clusters are matched, and the number of tracks in each Pt bin are counted and put into a L1 record. Also adds up the total Pt of all found tracks, and includes the occupancy in the output record. Each DFEA queries the its neighbouring sectors to see if there is an isolated track.
- **L2CTOC records.** As raw tracks and clusters arrive at this FPGA they are stored in a deep pipeline. When this FPGA detects that a L1\_ACCEPT occurred it jumps back N events in the pipeline and extracts the raw tracks and clusters. There are two separate L2 records sent the the CTOC: L2CFT (list of tracks with matching cluster information) and L2CPS (list of clusters with matching track information).

In addition to these normal functions the backend FPGA also provides several types of fake L1 records to the CTOC. It also sends input link diagnostic bits from the track finder FPGAs down to the DFE motherboard. This FPGA handles all communication with the DFE motherboard.

### 5.2 BLOCK DIAGRAM OF THE BACKEND FPGA

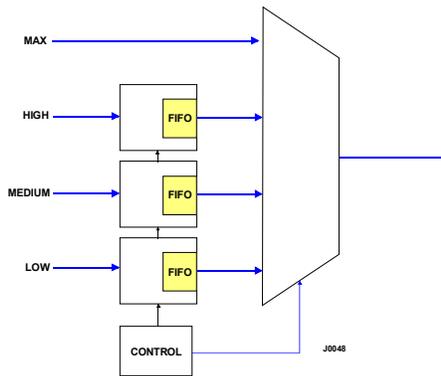


### 5.3 L1 MUON TRACK SORT LOGIC

The purpose of the L1 Muon Track Sorter is to collect the CFT axial tracks from the track finder FPGAs and select the six highest Pt tracks and output them as quickly as possible to meet L1 muon timing requirements.

Four track busses (Max, High, Medium, and Low) deliver up to six tracks each to the sorter module. On each bus the tracks are compressed into the first *n* timeslices with no gaps in between. Each track finder FPGA presents its highest Pt track first – the rest of the tracks are listed in order of decreasing Pt.

The track sorter is essentially a large 4:1 bus MUX with FIFOs on some of the inputs:



If there is a track on the MAX bus it is always passed to the output. If tracks are present on any other bus they are stored in the appropriate FIFO. If there is no track on the MAX bus the control logic looks to see if there are any tracks stored in the High FIFO. If a track is found there it is passed to the output. If there is no track in the High FIFO then the Medium and Low FIFOs are tried next. In this manner the tracks from multiple track finders are “zero compressed” and serialized in order of descending Pt. Control logic insures that the FIFOs are flushed at the appropriate time so that no old tracks show up in the next event. In the following example eight tracks are found:

Timeslice	MAX	HIGH	MEDIUM	LOW
1	T1	T4	T5	T7
2	T2		T6	T8
3	T3			
4				
5				
6				
7				

In the first three timeslices the MAX bus always has a track, thus T1, T2, and T3 are passed to the output. On the fourth timeslice MAX doesn't have a track, so the control logic looks at the High FIFO, which contains T4. That track is passed to the output. On the next clock tick T5 is selected from the Medium FIFO. On timeslice six T6 is selected from the Medium FIFO. T7 and T8 are not reported.

The track finder latency is 2 clock cycles. The total DFEA latency from first data in to first L1muon track out is 16 clock cyles or approximately 300ns.

The L1muon track sorter logic also drives two control signals: PE (Parity Enable) and TE (Transmit Enable) to the SLDB (serial link daughterboard). PE is always asserted on the 7<sup>th</sup> timeslice – it serves as the End of Record marker for L1muon.

The TE control bit is a function of the trigger framework. TE is always dropped during SyncGap. If the DFEA sees the CFT\_RESET control bit it immediately drops TE until the first crossing after CFT\_RESET is not asserted. A small state machine in the L1muon track sorter module checks SyncGap and CFT\_RESET and drives the TE output bit appropriately. The control bits and the 16-bit data bus are synchronous to the DFE motherboard master 53MHz clock.

### 5.3.1 L1MUON OUTPUT BUS FORMAT

The data bus to L1muon is 16 bits wide, organized like this:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
v	0	0	0	c	pt	extpt				phi					

2002-11-19a

Where **V** is the valid bit, set to indicate a valid CFT track.

**C** is the curvature of the track

**Pt** is the Pt bin (00=Max, 01=High, 10=Med, 11=Low)

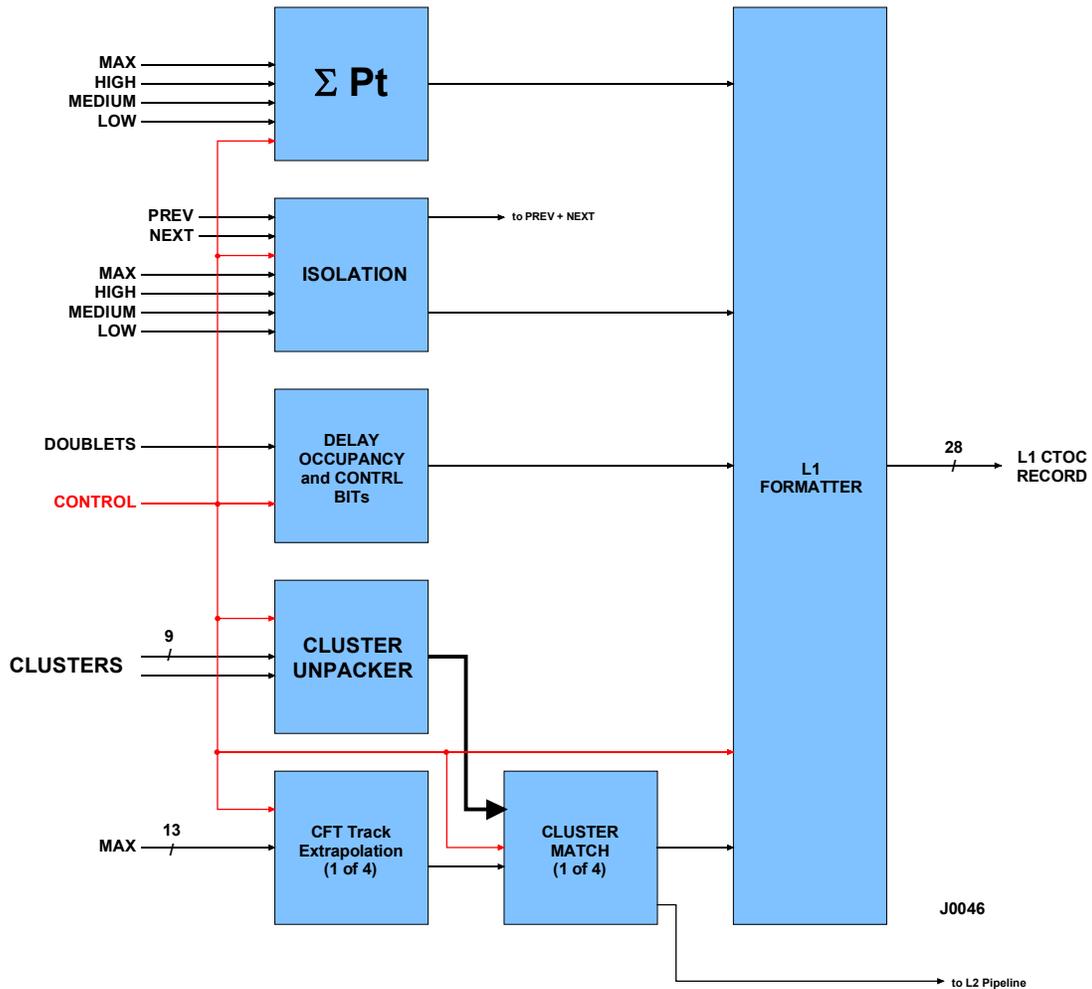
**ExtPt** is the Extended Pt field (0-7)

and **Phi** is the H-layer doublet index in the home sector (0-43)

### 5.4 L1CTOC ENGINE

The L1 CTOC engine does the most work in this FPGA. It is responsible for producing counts of the number of tracks in each Pt bin. It also checks to see if a CFT track matches to a CPS cluster. Neighbouring sectors are queried to see if they have any found tracks and this information is used to determine if the home sector has an isolated track. The backend FPGA *always* gets new data from the track finder FPGAs. It doesn't care if the system is in SyncGap or in L2 readout or if there's beam, etc. – the Track Finder FPGAs always runs and the L1CTOC Engine always accepts their output data and produces L1CTOC output records.

### 5.4.1 L1CTOC ENGINE BLOCK DIAGRAM



Wherever possible, the L1 Engine keeps the data serial. In other words, CFT tracks are never “flattened” and viewed all at the same time – this leads to code bloat. Rather, the six tracks in each Pt bin are sent serially down a pipeline, and the answer falls out some clock cycles later. In terms of logic resources this is very efficient because a single pipeline operates on six tracks – thus is 1/6 the size of a parallel implementation.

The **track extrapolation/cluster match** modules form the longest pipeline in the engine. All of the other modules formulate their answers much sooner but hold delay their output so that it is aligned with the data falling out of the end of **cluster match**. Data pours into the **L1 Formatter** serially; the pieces of data are aligned in time such that the formatter doesn’t have to move any data from one timeslice to another to form the L1CTOC output record. The **L1 Formatter** dumps out the complete L1CTOC record in 7 timeslices. Fake L1CTOC records are contained in the **L1 Formatter** and these can be turned on and off remotely.

### 5.4.2 MATCHING TRACKS TO CLUSTERS

The most complex operation in the L1 Engine is matching tracks to clusters. To do this, a pipeline is constructed so that as each CFT track passes through the pipeline it is compared

against up to eight found clusters. To do this the clusters must be flattened so that they can be accessed in parallel.

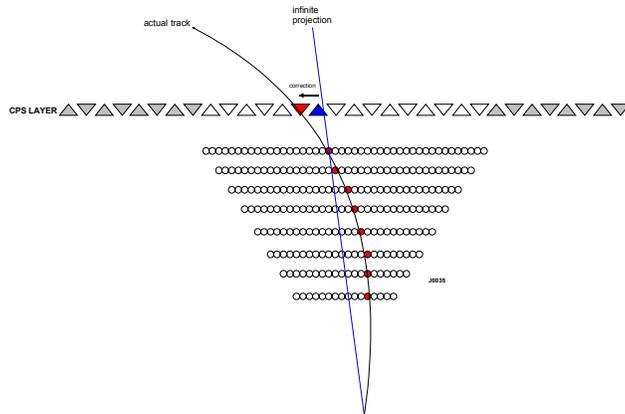
**5.4.2.1 Cluster Unpacking**

The cluster unpacker accepts up to eight clusters directly from the cluster finder modules A and B in the Medium FPGA. Cluster finder A delivers clusters in increasing phi order, while cluster finder B delivers them in order of decreasing phi. The first cluster delivered by cluster finder A becomes cluster0, the next cluster1, up to cluster3. The first cluster delivered by cluster finder B becomes cluster7, the next cluster6, etc. Now the clusters are ordered in terms of increasing phi, however there will be gaps in the list if less than 8 clusters are found.

The cluster unpacker flattens the clusters into eight busses of 6 bits each. The most significant bit of each bus is a valid cluster marker, and the lower 5 bits are the cluster centroid position. The cluster width information is discarded here. The outputs of the cluster unpacker are stable for seven clock ticks.

**5.4.2.2 Track extrapolation to CPS layer**

In order to determine which cluster matches best to each track the CFT track must be projected up onto the CPS layer. This extrapolation process has two steps: infinite projection and correction.



The blue line represents the infinite projection – a straight line drawn from the origin through the H layer doublet of the track. In hardware this is lookup table, shown on the left:

phi	CPS	phi	CPS
0-1	8	22-24	16
2-4	9	25-27	17
5-7	10	28-29	18
8-10	11	30-32	19
11-13	12	33-35	20
14-15	13	36-38	21
16-18	14	39-41	22
19-21	15	42-43	23

2002-01-31a

Pt Bin	Correction	
	C=1	C=0
Max	0	0
High	1	-1
Medium	2	-2
Low	3	-3

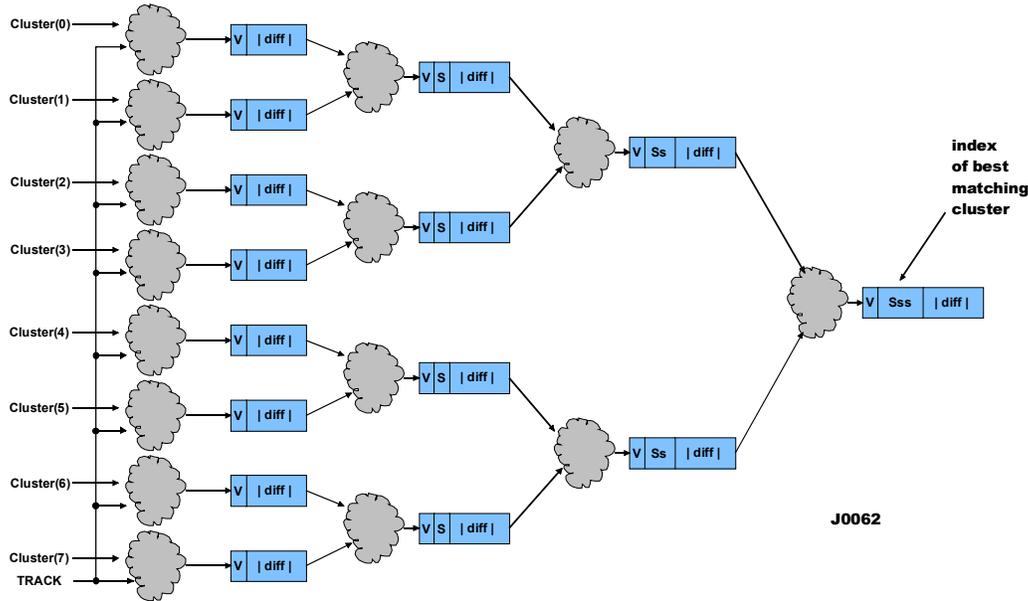
2002-01-31a

The final step is to do the correction. This shift is a function of the track’s curvature and Pt bin. The CPS fiber index given in the first step is incremented or decremented using another lookup table shown on the right.

As each track leaves the extrapolation module it has been converted into a CPS axial phi value, a curve bit, and a valid bit. This information is passed on to the next stage of the pipeline, where the best matching cluster is determined.

### 5.4.2.3 Cluster Match Module

The cluster match module sees all of the found clusters at the same time, but accepts CFT tracks from the extrapolator one at a time. The module is another pipeline, four stages long:



The first stage is where the absolute difference between the extrapolated CFT track and the CPS centroid is calculated. If the cluster is valid and the track is valid, a valid bit is passed on to the next stage along with the difference value.

At the next stage of the pipeline the two differences are compared, and the cluster with the lower of the two differences is passed on. If the two differences are valid and equal the cluster with the lower index is passed on. In addition to passing on the Valid flag and the difference, this stage tacks on an 'S' bit to indicate which cluster index was selected (0=top, 1=bottom).

The next two stages work the same way, each adding another 'S' bit. At the end of the pipeline the binary code 'Sss' is the index of the cluster who's centroid is the closest to the extrapolated CFT track.

### 5.4.2.4 Cluster Matching Windows

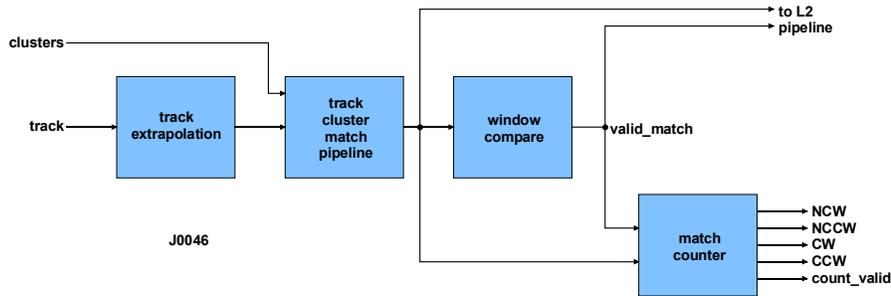
The last step of the track and cluster matching logic compares the difference between the extrapolated track the closest CPS centroid. If the difference is less than or equal to a constant the cluster is considered matched to the track. The track/cluster matching windows are sized depending on the Pt bin:

Max: 2 CPS strips  
 High: 4 CPS strips  
 Med: 6 CPS strips  
 Low: 8 CPS strips

At this point there is a list of up to 24 tracks, each track containing a new field that contains the index of the best matching cluster. An additional bit called **valid\_match** indicates that there is a valid match between the track and the cluster index. This information is passed directly onto the L2 Pipeline for later readout in the event of a L1\_ACCEPT.

### 5.4.3 COUNTING TRACKS

The L1 CTOC record does not contain any tracks, but merely the counts of the number of tracks in each Pt bin. For each Pt bin four counts are generated: CCW tracks with no cluster match, CW tracks with no cluster match, CCW tracks with a cluster match, and CW tracks with a cluster match. A module called *match\_counter* catches the tracks as they fall out of the track/cluster match pipeline. It also considers the **valid\_match** bit calculated by the window comparator.



Based on this information *match\_counter* produces the four counts for each Pt bin. These 16 counts are passed on the L1 Formatter module.

### 5.4.4 COUNTING CLUSTERS

A simple module counts the number of valid clusters as they arrive in the backend FPGA. The count is truncated to two bits, as specified by the L1CTOC protocols. If three or more clusters are found this count equals 3. Note that neither the cluster finders nor the cluster count logic differentiates between clusters found in the home sector and clusters found outside the home sector. For example, if only two clusters are found, but they're both in the Previous sector, the L1CTOC record "#PSC" field will equal two.

### 5.4.5 SUM OF PT CALCULATION

As each track is clocked into the L1 Engine a small lookup table converts the Pt and Extended Pt fields into an 8-bit integer representing the absolute Pt of the track. The lookup table is defined as:

Pt	Ext Pt	GeV	Pt	Ext Pt	GeV
MAX	000	80	MED	010	3
	001	27		011	3
	010	16		000	3
	011	12		001	3
HIGH	000	9	LOW	010	2
	001	7		011	2
	010	6		100	1
	011	5		101	0
MED	000	4	110	0	
	001	4	111	0	

2002-02-08b

After going through the lookup table the GeV value of each track is passed onto an accumulator, where the sum of all the 8-bit GeV values is calculated. The final result is an 8-bit number representing the total absolute Pt of the tracks found in the home sector. Note that there is no

overflow protection here – if the total sum is 257 it will be reported as 0x01. The L1CTOC protocols only allow for six bits to express the sum of Pt in a trigger sector, so the six most significant bits of the 8-bit sum are inserted into the “SUM ABS [PT]” field (in other words, the sum of Pt is divided by four).

### 5.4.6 ISOLATED TRACK DETERMINATION

Dedicated busses on the DFE backplane allow a DFEA daughterboard to send a bit to its two neighbours. Likewise each DFEA can read a bit from each of its two neighbours. These shared bits are used to determine isolated tracks. If a DFEA has one or more tracks it informs both of its neighbours. If a DFEA is only one track and neither of its neighbours have any tracks this is considered to be an isolated track, and it is reported as such in the L1 records sent to the CTOC.

When the DFEA determines that it has one or more tracks it sets its **home\_trk** output and keeps asserting it for a whole 132ns crossing. This allows sufficient time for the signal to propagate from the DFEA through the motherboard, through the backplane (and any cables connecting the two DFEA crates). Four clock cycles after asserting **home\_trk** the isolation module samples the **prev\_trk** and **next\_trk** lines coming from the adjacent sectors.

### 5.4.7 FORMATTING THE L1 RECORD

All of the modules feeding the L1Formatter delay and stretch their outputs so that everything is aligned in time as it arrives at the L1Formatter inputs. Most of the data is stable for almost a whole 132ns crossing. A start bit tells the L1Formatter that its inputs are valid; then it begins collecting data and building up the L1CTOC frames in parallel. Then all seven frames are loaded into a 7x28 bit shift register and sent out. The parity bits are not calculated now – that is done in the final stage immediately before the output of the FPGA. Two control lines select which type of record will be loaded and shifted out of the L1Format module:

27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	normal event [00]			
1	1	1		PM		OCT		CR	FX	0	0	0	0	0	0	REL SECTOR	0	0	1	0	1	1	1	1	1	1	1				
0	0	0		0	0	0	0	0	0	0	0	MAX CCW	MAX CW	0	0	MAX NCCW	MAX NCW	0	0												
0	0	0		0	0	0	0	0	0	0	0	HIGH CCW	HIGH CW	0	0	HIGH NCCW	HIGH NCW	0	0												
0	0	0		0	0	0	0	0	0	0	0	MED CCW	MED CW	0	0	MED NCCW	MED NCW	#PSC													
0	0	0		0	0	0	0	0	0	0	0	LOW CCW	LOW CW	0	0	LOW NCCW	LOW NCW	0	0												
0	0	0		SUM OF PT				0	0	OCCUPANCY DOUBLET HITS				ISOLATED TRACK																	
0	0	0	0																												
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	null event [01]			
1	1	1		PM	0	0	0	CR	FX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0	0	0	0		SUM OF PT				0	0	OCCUPANCY DOUBLET HITS				ISOLATED TRACK																
0	0	0	0																												
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	sparse event [10]			
1	1	1		PM	0	0	0	CR	FX	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0				
0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0				
0	0	0		SUM OF PT				0	0	OCCUPANCY DOUBLET HITS				ISOLATED TRACK																	
0	0	0	0																												
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	test event [11]			
1	1	1	1																			0	0	1	0	1	1				
0	0	0																													
0	0	0																													
0	0	0																													
0	0	0																													
0	0	0																													
0	0	0	0																												

The L1Formatter module accepts data and builds up the L1CTOC record frames continuously regardless of what mode the trigger framework is in. A multiplexor selects if a L1 or L2 record will be passed onto the parity module and then to the CTOC board.

## 5.5 L2 CTOC ENGINE

When the DFEA sees a L1\_ACCEPT control signal the backend FPGA switches modes. It jumps back  $n$  events and produces two L2 records. These records contain more stuff than the L1 records: up to 24 tracks, up to 8 clusters, turn and crossing information, etc. The L2CTOC engine builds up these two records when activated.

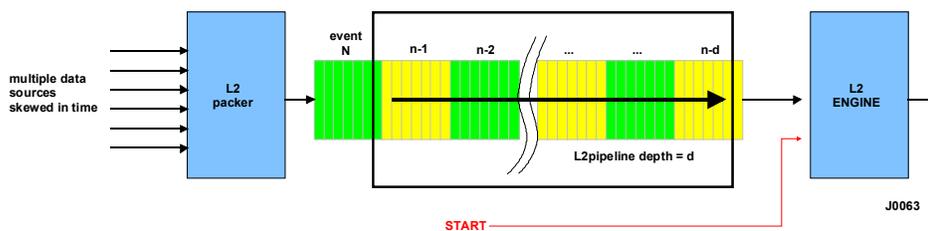
Building the L2CFT record is done first. It contains a list of the found tracks sorted in order of decreasing Pt. Each track frame contains information about the matching cluster (if applicable). Many different tracks may point to the same cluster.

While the L2CFT record is being assembled additional logic is working on resolving the cluster to track matching. Since many tracks may match to the same cluster the logic must determine the best track to match to each cluster. When this indirection is resolved the L2CPS record is produced and sent to the CTOC.

Record processing in the L2engine has a constant latency. The output begins at the same time regardless of how many tracks or clusters were found.

### 5.5.1 L2 PIPELINE OPERATION

As information arrives at the backend FPGA it must be saved in case it is needed for a L2 record later on. The L2pipeline is the storage element that handles this operation. Raw data (and some of the results from the L1engine) are delayed so that they are all aligned in time before being clocked into the L2pipeline.



The L2pipeline operates like a very long shift register: events clocked into the input of the pipeline fall off the back  $n$  events later. The pipeline never stops, even when a L1\_ACCEPT occurs.

In reality the pipeline is constructed out of an 80-bit x 512 word dual port SRAM. However, the complexity of manipulating the read and write pointers is hidden from the rest of the design. The depth of the pipeline can be dynamically (and remotely) adjusted from 1 to 64 events deep.

### 5.5.1.1 L2 Pipeline Packing

Each event in the pipeline takes up seven words. The turn and crossing information is loaded into the pipeline with the raw data – insuring that the data and the turn/crossing information are strongly tied together (useful for setting the correct pipeline depth).

Before the data can be clocked into the pipeline it must be aligned in time. Some of the stuff that must go into the pipeline comes early – for example the raw tracks and clusters. Other data is available much later – such as the list of matched tracks and clusters coming from the L1 engine. The early data is delayed using Xilinx SRL16 primitives, which are tiny 1x16 adjustable-depth shift registers that fit into a single Virtex “slice”. All of the data is delayed such that it fits into seven clock ticks like this:

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35								
0	# of tracks	# of tracks	# of tracks	# of tracks															ISO	P&M																																	
1	V	C	Ext Pt		Phi					CM	Cnum	V	C						Ext Pt		Phi																																
2	V	C	Ext Pt		Phi					CM	Cnum	V	C						Ext Pt		Phi																																
3	V	C	Ext Pt		Phi					CM	Cnum	V	C						Ext Pt		Phi																																
4	V	C	Ext Pt		Phi					CM	Cnum	V	C						Ext Pt		Phi																																
5	V	C	Ext Pt		Phi					CM	Cnum	V	C						Ext Pt		Phi																																
6	V	C	Ext Pt		Phi					CM	Cnum	V	C						Ext Pt		Phi																																
7																																																					

	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
0																		V		Centroid		Width	V		Centroid		Width																												
1	V	C	Ext Pt		Phi													V		Centroid		Width	V		Centroid		Width																												
2	V	C	Ext Pt		Phi													V		Centroid		Width	V		Centroid		Width																												
3	V	C	Ext Pt		Phi													V		Centroid		Width	V		Centroid		Width																												
4	V	C	Ext Pt		Phi																																																		
5	V	C	Ext Pt		Phi																																																		
6	V	C	Ext Pt		Phi																																																		
7																																																							

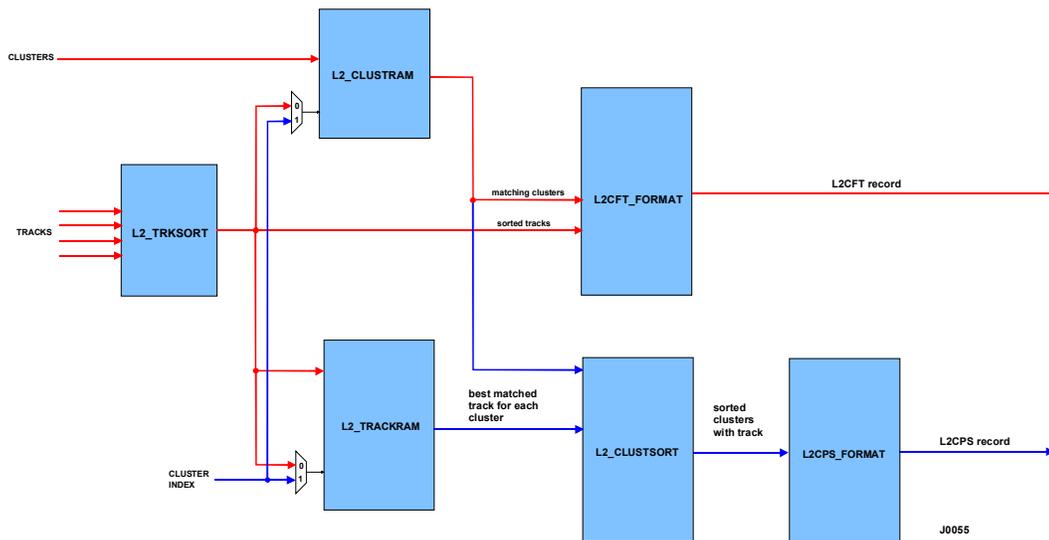
Maximum
High
Medium
Low
Cluster
Misc

2002-03-22a

Where **CM** is a bit that indicates that the track matched to a cluster, and **CNUM** is the index of the matching cluster (0-7). The number of tracks in each Pt bin is also calculated before the tracks are clocked into the pipeline. The order in which the data is positioned in the pipeline has been optimized for the L2engine.

## 5.5.2 L2 ENGINE OPERATION

The L2engine is always standing by, ready to catch the event as it flies out of the L2pipeline. When a L1\_ACCEPT signal is detected a **START** control bit is passed to the L2engine just as the desired event begins falling out of the pipeline. The L2engine then grabs the data, chews on it and spits out the L2CFT and L2CPS records. The L2engine can be ready for another **START** in about seven 132ns crossings.



The L2engine consists of two sorting modules, two small RAMs, and two formatter modules. The red lines in the block diagram illustrate the data paths exercised when the L2engine is producing the L2CFT record:

1. L2\_TRACKRAM is erased
2. Clusters are loaded into the L2\_CLUSTRAM
3. Tracks are sorted by L2\_TRKSORT and output one at a time.
4. Each track's CNUM field is used as address pointer into L2\_CLUSTRAM. L2\_CLUSTRAM outputs the matching cluster centroid, width, and valid cluster bit.
5. Tracks and their matching clusters are clocked into L2CFT\_FORMAT, which produces the L2CFT record.

As each track is leaving L2\_TRKSORT, the L2\_TRACKRAM watches each track go by. If L2\_TRACKRAM sees that a valid track matches cluster  $x$  it checks to see if its memory location  $x$  is empty. If it is the current track is stored in L2\_TRACKRAM location  $x$ . Since tracks leave L2\_TRKSORT ordered by decreasing Pt, L2\_TRACKRAM now contains the highest Pt track that matches to each cluster. Now the L2\_CPS process can begin:

1. The muxes in front of the two RAMs switch over and are driven by a 3-bit counter.
2. The cluster index counts from 0 to 7.
3. Cluster information is output from the L2\_CLUSTRAM, and the highest Pt matching track information is output from the L2\_TRACKRAM.
4. At this point clusters are not zero compressed, and also some clusters may have centroids outside of the home sector. The cluster and matching track data is filtered and compressed in L2\_CLUSTSORT.

- The final list of clusters is sent to the L2CPS\_FORMAT module and the L2CPS record is assembled.

**5.5.2.1 Track and Cluster Ram Initialization**

Before L2\_TRKSORT begins outputting sorted tracks the two RAMs must be initialized. The cluster ram contained in L2\_CLUSTRAM is loaded with clusters from the pipeline:

	8	7	6	5	4	3	2	1	0
0	V	Centroid						Width	
1	V	Centroid						Width	
2	V	Centroid						Width	
3	V	Centroid						Width	
4	V	Centroid						Width	
5	V	Centroid						Width	
6	V	Centroid						Width	
7	V	Centroid						Width	

2002-11-22a

Internally L2\_CLUSTRAM is actually two small RAMs (they're loaded in parallel from the pipeline), but to the rest of the L2engine it appears as the 9x8 RAM shown above. Centroid values are 0-31, where centroids 0-7 are in the previous sector, 8-23 are in the home sector, and 24-31 are in the next sector. The cluster index serves as the memory address. Clusters in this ram are ordered in terms of increasing centroid phi, but there may be gaps. Also, some clusters centroids may exist outside of the home sector.

The L2\_TRKRAM is erased prior to any tracks coming out of the L2\_TRKSORT module. This RAM looks like this:

	7	6	5	4	3	2	1	0
0	V	CM	C	Pt	ExtPt			
1	V	CM	C	Pt	ExtPt			
2	V	CM	C	Pt	ExtPt			
3	V	CM	C	Pt	ExtPt			
4	V	CM	C	Pt	ExtPt			
5	V	CM	C	Pt	ExtPt			
6	V	CM	C	Pt	ExtPt			
7	V	CM	C	Pt	ExtPt			

2002-11-22a

Where V is a valid track bit, C is the curvature of the track, and CM indicates that the track matches to a cluster.

**5.5.2.2 Sorting Tracks**

L2\_TRKSORT is very similar to the Muon Track sort described earlier in this document. It accepts up to 24 tracks on four busses from the L2pipeline. Within each Pt bin up to six tracks are delivered serially to L2\_TRKSORT. Tracks leave L2\_TRKSORT strictly ordered by decreasing Pt. Tracks are also zero compressed.

**5.5.2.3 Formatting the L2CFT Record**

Tracks and their matching cluster information are presented to the L2CFT\_Formatter serially. To build up the L2CFT header the formatter must know how many tracks are going to be reported. This is the reason why the four [# of tracks] fields were stored in the L2pipeline. While the tracks are being sorted these four numbers are quickly added together and made available to the L2CFT formatter. Likewise, the Turn and Crossing, P&M, and ISO information is registered as it falls out of the pipeline – it is kept around for use by the L2CFT and L2CPS formatters.

The L2CFT formatter basically has to build up the record header and then insert the tracks and cluster data as it arrives. There is very little manipulation of the data, with one exception: if the

track matches a cluster outside the of the home sector, the R bit is set (see the [Protocols](#)). The R bit is generated from the matching cluster’s centroid. Parity bits are not added here, as a module exists later on to do this.

The entire L2CFT record is zero compressed, and has a maximum length of 27 frames (24 tracks + 2 header + parity frame).

The L2CFT formatter module also selects between normal and fake L2CFT records. The fake L2CFT record is defined as:

27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	1	1	h	1	0	real oct	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0
0	0	0	h	real turn number										real tick number														
0	0	0	h	0	0	1	1	real RelSect	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	
0	0	0	h	0	0	0	0	real RelSect	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	1	0	0	
0	0	0	h	1	1	0	1	real RelSect	1	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	0	0	0
0	0	0	h	0	0	0	0	real RelSect	0	0	1	1	1	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0
0	0	0	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v

dfea\_u1\_vectors.xls

There are four tracks in this fake record:

- MAX track extpt=0, phi=7, curve=1, tight match to cluster centroid=3
- HIGH track extpt=1, phi=13, curve=0, no cluster match
- MED track extpt=3, phi=40, curve=1, tight match to cluster centroid=13
- LOW track extpt=2, phi=38, curve=1, no cluster match

The latency for the fake L2CFT record is exactly the same as the normal L2CFT record latency, so fake and normal L2CFT records will arrive at the CTOC/STOV/STSX inputs at the same time.

### 5.5.2.4 Matching Clusters to Tracks

L2\_TRKRAM observes every track as it leaves L2\_TRKSORT. L2\_TRKRAM uses the track’s CNUM field as an address pointer into its RAM. If that memory location is empty, the current track information (V, CM, C, Pt, and ExtPt) is stored in the track ram at the location given by CNUM. Because tracks are sorted by Pt, this process insures that the trackram will be filled with the highest Pt track that matches to each cluster.

### 5.5.2.5 Selecting and Zero Compressing Clusters

After the last track has passed out of the L2\_TRKSORT module both RAMs now contain all of the information needed to build up the L2CPS record. However additional sorting and filtering is needed for two reasons: 1) there may be gaps in the cluster ram, and the L2CPS record must not contain gaps; 2) *Clusters with centroids outside of the home sector will not to be reported in the L2CPS record.*

Until this filtering and sorting is complete the number of objects in the L2CPS record cannot be determined. This is done in the L2\_CLUSTSORT module. L2\_CLUSTSORT takes over control of the two RAMs and pulls out the cluster and matching track data serially, starting with cluster index 0. After L2\_CLUSTSORT has seen all 8 potential (clusters and track data) it informs L2CPS\_FORMAT of the number of clusters to be reported. As L2CPS\_FORMAT is building up its header L2\_CLUSTSORT begins to send the cluster and track data to it.

The entire L2CPS record is zero compressed and has a maximum length of 11 frames (8 clusters + 2 header + parity frame).

Just like the L2CFT formatter, the L2CPS formatter can substitute a fake L2 record in for the real data. This fake L2CPS record looks like this:

27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	h	1	0	real oct			0	1	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0
0	0	0	h	real turn number												real tick number											
0	0	0	h	0	0	0	1	1	0	0	1	1	0	1	0	0	0	0	0	0	real RelSect	0	0	0	0	0	
0	0	0	h	0	1	1	0	1	0	1	1	1	0	1	1	0	0	1	1	0	real RelSect	0	0	0	0	0	
0	0	0		v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v

There are two clusters, and each cluster matches to a track. This fake L2CPS record is designed to complement the fake L2CFT record. **Note that the fake L2CPS and fake L2CFT records cannot be enabled at the same time – refer to the bus control register section of this document for information on how to turn these L2 fake records on and off.**

**5.5.2.6 Tick and Turn Counters**

The tick and turn numbers are based off two control bits coming from the trigger framework. Those bits are CFT\_RESET (aka SCL\_INIT, CR) and First\_Crossing (aka FX). CFT\_RESET is a very long duration pulse – on the order of a couple seconds. Because CFT\_RESET is not very precise, the counters are started on the First\_Crossing after CFT\_RESET disappears. The turn number is a 16-bit counter that is reset to 0x0001; it increments every time it sees First\_Crossing. The crossing number is an 8-bit counter that is reset to 0x07 when First\_Crossing is seen, it is a modulo 159 counter, so the maximum count is 158 or 0x9E. These two counters are used only for L2 readout in the DFEA.

**5.6 MOTHERBOARD INTERFACE**

The DFE motherboard supplies the DFEA daughterboard with buffered copies of the LVDS link data, a master clock, and control signals (which are derived from link 2 or 7). Additionally the DFE motherboard acts as a conduit for FPGA configuration data coming from the DFE crate controller (DFEC). The motherboard also collects status information from the DFEA boards and passes it back to the DFEC, which makes that data available to the rest of the system. The DFE motherboard also has the ability to write bytes to the DFEA backend FPGA (U1) – this is used to set the L2 pipeline depth, set the home sector, and force the DFEA into other diagnostic modes.

**5.6.1 FPGA CONFIGURATION**

The five FPGAs on the DFEA daughterboard are arranged in a “slave serial” configuration scheme. The order of devices in this chain is as follows:

1. U6 (Max) XCV400-4BG432C
2. U4 (High) XCV400-4BG432C
3. U7 (Medium) XCV400-4BG432C
4. U5 (Low) XCV600-4BG432C
5. U1 (Backend) XCV300-4BG432C

The design files are compiled individually, but then the BIT files are concatenated in the PROMGEN utility to produce a single HEX file, one HEX file per DFEA daughterboard.

### 5.6.2 DFEA MOTHERBOARD BACKPLANE REGISTERS

The DFEC communicates with the DFE motherboard though a very simple synchronous “A16D8” bus. The timing of this bus is specified in Appendix A.

Each DFEA Motherboard has eight registers that the DFEC may write to or read from. These are listed below:

	Write								Read							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
base+0	Board Control								Board Status							
+1	Device Control								Device Status							
+2	Device Config Data								Device Config Data							
+3	Device Firmware ID								Device Firmware ID							
+4	LED Control Register								LED Control Register							
+5				CH				Page Select								
+6								L2 Pipeline Depth								
+7								Home Sector Register								
+8	Bus Control Register															

2003-03-03a

The first three registers work as described in the DFE Crate Controller TDR<sup>2</sup>. These registers are used for device initialization and to set the current device.

#### 5.6.2.1 Device Firmware ID Register

The Device Firmware ID Register simply stores an 8-bit value which may be used to identify which firmware version is currently loaded into the DFEA. Use of this register is optional. The following macro shows how to set the Firmware ID registers for the top and bottom DFEA boards:

```
set DFE 12
set device 0
cmd writebyte 3 0x99      # set top DFEA firmware ID
set device 1
cmd writebyte 3 0x74      # set bottom DFEA firmware ID
```

This next macro shows how to read the Firmware ID registers for a DFEA motherboard:

```
set DFE 12
set device 0
cmd readbyte 3
set device 1
cmd readbyte 3
```

#### 5.6.2.2 LED Control Register

The LED control register is a diagnostic feature to make sure that the backplane reads and writes are working properly. It is not intended to be used in normal operation. This register is readable and writeable. If the MSb of this register is set, then the lower seven bits [6:0] drive the front panel leds like this:

master clock detect	[6]	[ ]	+3.3V power OK
top DB configured	[5]	[4]	bottom DB configured
backplane DTACK	[3]	[2]	First Crossing
L1 Accept	[1]	[0]	Sync Gap

#### 5.6.2.3 Extra Backplane Registers

The registers at +5 through +8 affect the current device; device 0 is the top DFEA and device 1 is the bottom DFEA. The current device is set by writing to the Device Control Register (see 2000-01-28a for details). These extra registers are used to pass additional parameters to the DFEA daughterboards after they have been initialized. These registers can be written at any time after initialization and the changes will take effect immediately.

#### 5.6.2.4 L2 Pipeline Depth

The L2 Pipeline Depth Register controls the depth of the DFEA's L2 pipeline, expressed in terms to 132ns crossings. The power-up default value is 36 crossings (resetting the DFE motherboard will set this register to 36). Values from 0 to 63 are valid. The following macro shows how to set the top DFEA pipeline depth to 35:

```
set DFE 12
set device 0
cmd writebyte 6 0x23
```

#### 5.6.2.5 Home Sector Register

The Sector Register tells the DFEA what its home sector number is. Values from 0-79 (0x00 – 0x4F) are allowed. Based on this number the “Octant” and “Relative Sector” fields in the L1 and L2 records are calculated. The power-up default value for this register is 0x00 (resetting DFE motherboard will set this register to 0x00). The following macro sets the bottom DFEA home sector to 49:

```
set DFE 12
set device 1
cmd writebyte 7 0x31
```

#### 5.6.2.6 Bus Control Register

Each DFEA daughterboard has two outputs: one goes to L1muon and the other goes to the CTOC/STOV/STSX boards. This 8-bit register allows the user to control what information is sent over these busses. The power-up default value for this register is 0x00 (resetting the DFE motherboard will zero out this register).

7	6	5	4	3	2	1	0
L2 Control		L1 Control		CTOC/STOV/STSX Bus Control		L1muon Bus Control	

##### L1muon Bus Control [bits(1..0)]

```
00 = normal
01 = test pattern
10 = fake max track in crossing 10
11 = reserved
```

##### CTOC/STOV/STSX Bus Control [bits(3..2)]

```
00 = output bus enabled
01 = output bus disabled
10 = 28-bit johnson counter pattern
11 = reserved
```

*If the CTOC/STOV/STSX bus is enabled (00), then the following four bits apply:*

##### L1 Control [bits(5..4)]

```
00 = normal data mode
01 = empty pattern
```

10 = sparse event  
11 = link test pattern

**L2 Control [bits (7..6)]**

00 = normal data mode  
01 = prohibit L2 transmission (ignores L1\_ACCEPT)  
10 = enable L2CFT test record  
11 = enable L2CPS test record

**The following examples show how to use the bus control register...**

1. Setup the top DFEA to send normal data to L1muon and send the special link test pattern to CTOC/STOV/STSX while ignoring L1 Accepts. The bus control register will look like:

01 11 00 00 = 0x70

set DFE 14  
set device 0  
cmd writebyte 8 0x70

2. Setup the bottom DFEA to completely shutdown the link going to CTOC/STOV/STSX.

xx xx 01 00 = 0x04

set DFE 6  
set device 1  
cmd writebyte 8 0x04

3. Setup the top DFEA to send normal L2CFT records but fake L2CPS records.

11 00 00 00 = 0xC0

set DFE 7  
set device 0  
cmd writebyte 8 0xC0

### 5.6.2.7 Status Page Register

The purpose of the status page register is to control what status information is displayed in the slow monitor user bits (7..0). Each DFEA daughterboard supports up to 16 bytes of status information organized as 16 8-bit “pages”:

page	S7	S6	S5	S4	S3	S2	S1	S0
0						Found Track	Sync Error	TFW Locked
1	Home Sector (0-79)							
2	Bus Control Register							
3	L2 Pipeline Depth (0-63)							
4			Patt-err6	Patt-err5	Patt-err4	Patt-err3	Patt-err2	Patt-err0
5	0	1	0	1	0	1	0	1
6	1	0	1	0	1	0	1	0
7					L1_ACCEPT	CFT Reset	Sync Gap	First Xing
8							SG Error	FX Error
9								
A								
B								
C								
D								
E								
F								

2003-03-03a

Each DFEA daughterboard has its own set of 16 status pages. The DFEA motherboard can only display one of the 32 possible status pages at a time. To view a status page one must set the current device to the top or bottom daughterboard and then write to the status page register to select the page of interest.

Pages 1 through 3 echo back the DFEA parameters that were set by writing to the DFE backplane registers. Some of the other status bits are listed below:

**TFW Locked.** If this bit is set it means that the DFEA’s tick/turn event counter has locked onto the trigger framework control signals and is incrementing normally.

**Sync Error.** If this bit is set it means that the End of Frame markers on the DFEA’s input links did not line up, even after synchronization.

**Found Track.** This bit is set whenever the DFEA finds a valid CFT track

**L1\_ACCEPT.** This bit is set whenever the DFEA observes a L1\_ACCEPT control bit.

**CFT Reset.** This bit is set whenever the DFEA observes a CFT Reset (aka SCL Init) control bit.

**Sync Gap.** This bit is set whenever the DFEA observes a SYNC\_GAP control bit.

**SG Error.** This bit is set whenever the DFEA determines that the SYNC\_GAP was an incorrect length.

**First Xing.** This bit is set whenever the DFEA observes a FIRST\_CROSSING control bit.

**FX Error.** This bit is set whenever the DFEA determines that the FX control bit arrived at an unexpected time.

**Status Page 4:** These bits are set if the DFEA detects an error in the Mixer Output test pattern on any of its six input links. These bits are only used when running the Mixer to DFEA link test.

The Mixer supplies 8 links to each DFEA motherboard. On the DFE motherboard some links are shared between the two DFEA Daughterboards (see 1999-01-05A, page 4). Furthermore, each DFEA daughterboards re-numbers the input links as follows:

DFEM Input Link	TOP DFEA	BOT DFEA
0	0	
1		
2	2	
3	3	3
4	4	4
5	5	5
6	6	6
7		0
8		
9		2

2003-03-03a

An example: the Mixer turns on the test pattern for DFE Motherboard input number 2. The bottom DFEA does not see this input link, but the top DFEA does and reports the status of this link as patt\_err2 (status page 4, bit 1).

Another example: The Mixer turns on the test pattern for DFE motherboard input number 4. Both the top and bottom DFEAs report the status of this link as patt\_err4 (status page 4, bit 3).

### 5.6.2.8 Clearing the History Registers

The **orange cells** shown in the status page table operate differently than the rest of the status bits – once the bit is set it stays set until the Clear History bit is asserted. Setting bit four of the Page Register zeros out these bits. This enables the slow monitor system to observe intermittent glitches without having to sample a status bit continuously (as was the case with previous firmware). To reliably assert the Clear History bit, toggle it 0-1-0 like this:

```
cmd writebyte 5 0x00
cmd writebyte 5 0x10
cmd writebyte 5 0x00
```

The “CH” control bit is not tied to any particular status page. Toggling the “CH” bit resets all of the history status bits for the current device.

### 5.6.3 DFEA SLOW MONITOR STATUS WORD

In addition to the DFE backplane registers, there is a slow monitor bus that continuously moves small amounts of status information from the DFEA motherboards to the DFEC. Each DFE Motherboard produces a 16-bit status word. The DFEA Motherboards define these 16-bits like this:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFEA = 1011				CLK SEL		RDY	REG	status page (7..0)							

2003-01-06

- DFEA board type code is 0xB or 1011.
- Clock Select bits describe what the DFEA motherboard is using for its master clock source. (see 2000-01-28a for info).
- Ready is set when both the Top and Bot DFEA boards have been initialized.
- REG: This bit goes LOW if either DFEA voltage regulator is having a problem holding the FPGA core voltage at 2.5V  $\pm$ 5%.
- Status Page byte is determined by the current device number and the page register.

### 5.6.4 SOME POST-INITIALIZATION MACRO EXAMPLES

1) Set the top DFEA to sector 34, L2 pipeline depth=37, and ignore L1\_Accept signals. Verify that the changes were made.

```
set DFE 8
set device 0
cmd writebyte 6 0x25 # set L2 pipeline depth
cmd writebyte 7 0x22 # set home sector
cmd writebyte 8 0x40 # ignore L1_Accept

cmd writebyte 5 0x03 # set status page 3 (L2 depth)
get userbits 8 # verify that lower 8 bits = 0x25

cmd writebyte 5 0x01 # set status page 1 (home sector)
get userbits 8 # verify that lower 8 bits = 0x22

cmd writebyte 5 0x02 # set status page 2 (bus control)
get userbits 8 # verify that lower 8 bits = 0x40
```

2) Clear the history for the top and bottom DFEA and read out status page 0 for each DFEA.

```
set DFE 10

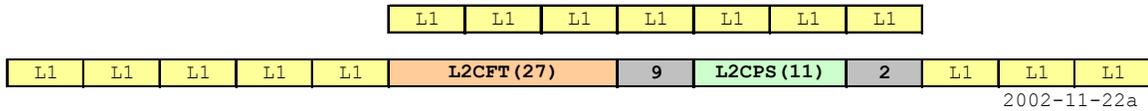
set device 0
cmd writebyte 5 0x00
cmd writebyte 5 0x10
cmd writebyte 5 0x00
get userbits 10 # lower 8 bits = top DFEA status page 0

set device 1
cmd writebyte 5 0x00
cmd writebyte 5 0x10
cmd writebyte 5 0x00
get userbits 10 # lower 8 bits = bottom DFEA status page 0
```

### 5.7 PARITY GENERATION AND L1/L2 RECORD SEQUENCING

The parity generator is a small module that is responsible for adding the horizontal and vertical parity bits to L1 and L2 records. The parity module accepts the records one frame at a time. The module is “smart” in that it can tell what kind of record is coming in, so it knows where the vertical parity frame should be inserted. L1 records are always seven frames, but L2 records may be as short as three frames and as long as 27 frames long. The vertical parity frame always follows immediately after the last L2 data frame, or after the L2 header if there is no data in the L2 record.

A Mux immediately in front of the parity generator selects the record source: L1, L2CFT, or L2CPS. Normally L1 records are passed on to the parity module and the out to the CTOC. But when a L1\_ACCEPT happens the mux is switched first to L2CFT and then to L2CPS and then back to L1 after the L2CPS timeslot has passed. Sequencing logic at the top level determines when to switch this mux so that the transitions are clean and always fall on L1 record boundaries:



## 6 DFEA LATENCY

Latency is measured first frame in to first frame out. Due to the deskew circuitry at the front end of the DFEA, these numbers can vary by up to one 18.8ns clock cycle.

- L1muon = 14 + 2 clocks (~300ns)
- L1CTOC = 14 + 21 clocks (~660ns)
- L2CFT = 14 + 7 clocks (~400ns)
- L2CPS = 14 + 7 + 36 clocks (~1070ns)

## 7 POWER REQUIREMENTS

Power Dissipation is approximately 6.5A per DFEA motherboard (two DFEA daughterboards, clocks running at 53MHz). 3.3V @ 6.5A = 21.5W per DFEA motherboard.

The linear +2.5V regulator on each DFEA can safely regulate to 3A, dissipating 2.4W. Current estimates are that the DFEA FPGAs draw 2 A of +2.5V.

## 8 FPGA RESOURCES

### U1 "Backend" FPGA (XCV300)

Number of External GCLKIOBs	2 out of 4	50%
Number of External IOBs	175 out of 316	55%
Number of LOCed External IOBs	166 out of 175	94%
Number of BLOCKRAMs	10 out of 16	62%
Number of SLICES	1683 out of 3072	54%
Number of GCLKs	2 out of 4	50%

### U6 Maximum Track Finder (XCV400)

Number of External GCLKIOBs	4 out of 4	100%
Number of External IOBs	150 out of 316	47%
Number of LOCed External IOBs	150 out of 150	100%
Number of SLICES	2551 out of 4800	53%
Number of GCLKs	4 out of 4	100%

### U4 High Track Finder (XCV400)

Number of External GCLKIOBs	4 out of 4	100%
Number of External IOBs	185 out of 316	58%
Number of LOCed External IOBs	185 out of 185	100%
Number of SLICES	3060 out of 4800	63%
Number of GCLKs	4 out of 4	100%

### U7 Medium Track Finder (XCV400)

Number of External GCLKIOBs	4 out of 4	100%
Number of External IOBs	143 out of 316	45%
Number of LOCed External IOBs	143 out of 143	100%
Number of SLICES	3351 out of 4800	69%
Number of GCLKs	4 out of 4	100%

### U5 Low Track Finder (XCV600)

Number of External GCLKIOBs	4 out of 4	100%
Number of External IOBs	169 out of 316	53%
Number of LOCed External IOBs	169 out of 169	100%
Number of SLICES	5721 out of 6912	82%
Number of GCLKs	4 out of 4	100%

## 9 DFEA DAUGHTERBOARD MECHANICAL

The OrCAD Layout file contains the exact DFEA mechanical dimensions for the connectors, mounting holes, and components. Pin numbers are also listed in this file.

DFEA Schematics and Layout files are available online at the DFE Hardware Documentation Home Page<sup>2</sup>.

## 10 REFERENCES

1. DFE Protocols maintained by Levan Babukhadia [blevan@fnal.gov](mailto:blevan@fnal.gov)  
<http://d0server1.fnal.gov/projects/VHDL/General/>
2. DFE Hardware Documentation Home Page:  
<http://www-d0online.fnal.gov/www/groups/cft/CTT/online/docs/>

## 11 REVISION HISTORY

- |      |   |
|------|---|
| 0.5  | 22 Jan 2003. Pre-release additions. FPGA resources, backplane registers, slow monitoring, latency.  |
| 0.51 | 23 Jan 2003. Change the L1 test pattern outputs. "busy" fake event replaced with "test" event.  |
| 0.52 | 28 Jan 2003. Wrong information in DFEA motherboard feature register/firmware revision register.   |
| 0.60 | 07 Mar 2003. Overhauled the backplane registers and slow monitor user bits. added macro examples.   |
| 0.61 | 13 Mar 2003. Changes to status pages.   |
| 0.62 | 25 Apr 2003 Changes to fake L2CFT and fake L2CPS records. add example macros showing how to put the DFEA into various fake modes.                 |
| 1.00 | 29 Jul 2003. Roll in eng note 1999-01-05a. Fix curvature definitions. Clean up references. Engineering note 2000-03-28a absorbed into Appendix A. |

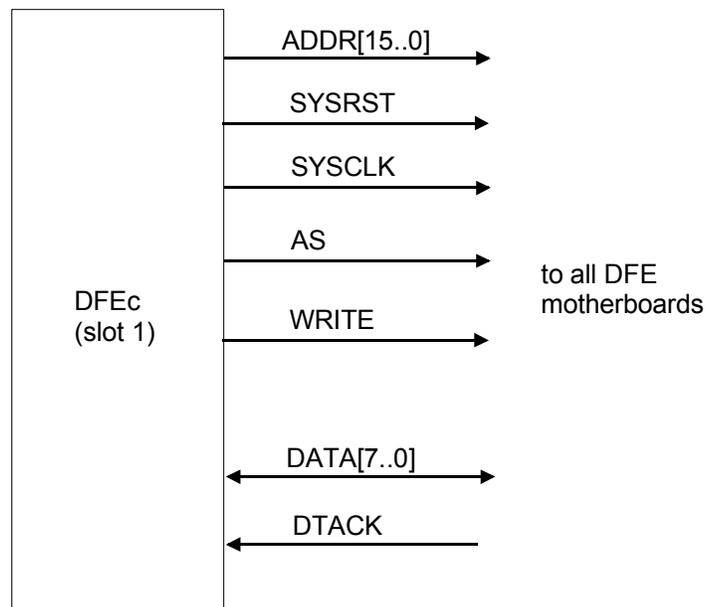
## APPENDIX A:

### DFE Backplane Protocol and Timing

In addition to the slow monitoring serial bus, the DFE backplane supports a higher speed address/data bus used primarily for transferring FPGA configuration data from the DFE controller to the motherboards. In some documentation this bus is described as a VME A16D8 bus, but this is no longer the case.

Even the simplest standard VME bus is overkill, hence a very simple custom bus is used instead. The protocol of this simple bus is described here.

### Bus Signals

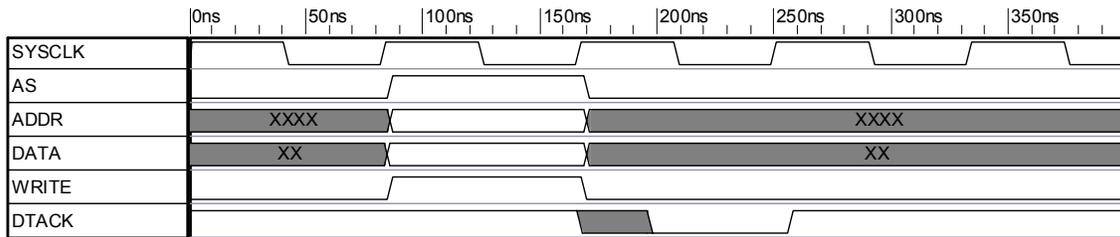


J0003

SYSRST is active low and forces all of the motherboards into a power up reset state if it is asserted for more than 2 seconds. Note: the slow monitor bus is not shown here. Check out my engineering note "20000131A" for more info on that bus. When WRITE is high the controller is writing data to a motherboard. DTACK is active low, open drain. AS is not really an address strobe, but here it indicates that the address on the bus is valid on the next rising edge of SYSCLK. SYSCLK is a free-running clock with a maximum frequency of 12MHz. (However, the controller has the ability to kill SYSCLK if necessary.)

## Write Cycle

Unlike VME, the DFE address/data bus is SYNCHRONOUS. Everything is registered to a free-running SYSCLK. A Write cycle requires two clocks: the first clock writes the data into a motherboard, the second cycle is the motherboard responding with a DTACK pulse to indicate that the write was successful. A single write cycle is shown below:



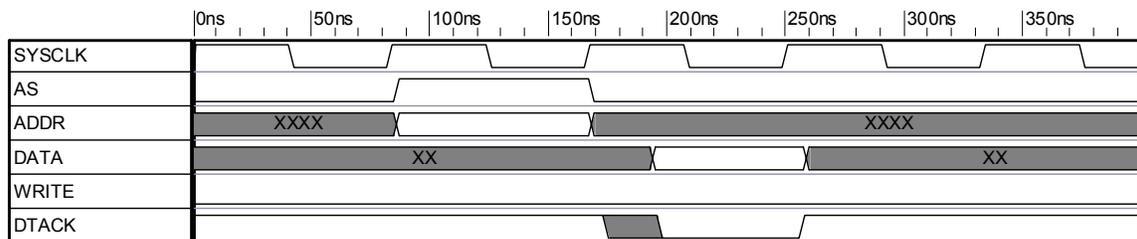
On the rising edge of SYSCLK the address and data lines are latched by the motherboard. If the address is valid the motherboard immediately pulls DTACK low for one clock cycle. DTACK may take some time to fall to a low level, especially in cases where the subrack is filled with motherboards; this is not a problem since SYSCLK is fairly slow (~10MHz). DTACK must be low at least 30 ns before the next rising edge of SYSCLK.

Multi-cast and broadcast writes are not supported; each motherboard must be written separately. (The one exception is when SYSRST is held low, then all of the motherboards RESET together.)

“Express Style” or burst mode writing is not supported. Additionally, a one cycle wait state must follow each write cycle. Since three SYSCLK cycles (write, dtack, wait) are required for a write cycle, the maximum throughput of the backplane bus is SYSCLK/3.

## Read Cycle

Each read cycle requires two SYSCLKs. The first SYSCLK writes the address into the motherboards. If the address is valid then the motherboard immediately asserts the data byte onto the data bus, as well as pulls DTACK low. DTACK and the byte on the data bus must be stable at least 30 ns before the next rising edge of SYSCLK.



Back to back reads are not supported; a one cycle wait state must occur between successive read cycles.

## Performance

The goal is to download an entire subrack of DFE motherboards in under a minute. Since the configuration file can be as large as 70MB, a total effective throughput rate of slightly faster than 1MB/s is required.

If polling was used during the download, each byte would require 6 SYSCLK cycles (WRITE, DTACK, WAIT, READ, DTACK, WAIT). If SYSCLK was running at 12MHz, the throughput with polling would be 2MB/s, thus the subrack could be configured in a little over 30seconds. Note that polling after each byte may not be necessary: polling after every N writes may be sufficient in many situations.

Note that the download times assume that all of the configuration data is contained in the Flash memory on the controller board. 1553 download times are not considered here.