Supercomputing Systems

H1 Second Level Fast Track Trigger

Feasibiltiy Study

Project Manager: David Müller
Reviewer(s): Markus Herrli

Revision list:

Version	Date	Author	Remarks	Visa
0.0	31.1.00	dm	Preliminary	—
1.0	7.3.00		Updated Business / administration part moved to project book	dm
1.1	10.3.00		Reviewed	mh
1.2	13.3.00		Few corrections	dm

1 Index

2 Introduction

2.1 Purpose of this document

The overall system design of the Fast Track Trigger (FTT) has been described in [1], [2]. The main characteristics of the proposed L2 system will be confirmed or introduced here to clearly define the system which SCS consideres to be feasible. This document is far more detailed than feasibility studies usually are.

Wherever possible, this document does not touch purely 'physical' questions (e.g. what binning to choose for the L2 linking) but concentrates on technical limitations.

The specification and the design description will be based on this study.

2.2 Status of this document

The understanding of this document may be difficult without knowledge of the H1 trigger system as well as the information given in previous documents about the FTT and the datasheets listed in the references.

2.3 Status of the project

- Feasibility of the L2 system is considered to be proven for the 48 DSP solution.
- Interfaces to other trigger systems, especially L3, are not yet finally decided on. Different scenarios have been discussed, a final decision will be necessary with the specification of the L2 FTT.
- The efficiency of the proposed linking algorithm should be checked with data.

3 Constraints

3.1 General Comments

All constraints embedding the L2 FTT into the H1 system are listed here. Any constraint missing in the following list risks not to be met by the design.

3.2 Timing

Operation of the L2 trigger starts after the L1 keep signal. An L2 trigger decision has to be delivered to the central trigger after 19.8 us.

3.3 Multiplicities

3.3.1 Track segments

Track segments are received from the four layers of the L1 system. Three inner layers consist of 6 L1 cards each, on the fourth layer, 12 L1 cards are needed. No assumptions on the distribution of the track segments relative to the L1 cards are done (i.e. it does not matter whether they are uniformly distributed or whether all segments originate from one single L1 card).

A maximum of 128 valid track segments is delivered per layer. The ordering of the track segments is arbitrary; if more then 128 track segments are delivered, only the first 128 segments will be processed on L2, ignoring the rest of the data entirely.

3.3.2 Fitted tracks

The L2 FTT is designed to reconstruct up to 48 CJC tracks per event. If more tracks are found in the linking step, these will not be fitted by the DSPs during the L2 time. An 'overflow' signal can be sent from the linker to the decision board, where a specific trigger response can be generated in these cases.

3.4 Interfaces

3.4.1 Track Segments from L1

On the physical layer, the LVDS standard using the DS90CR483 / DS90CR484 serializer/deserializer chipset by National Semiconductors will be used. The chipset allows to transmit 48 bit wide data plus clock at 33 to 112 MHz. For the FTT, the speed will be 50 MHz, yielding 2.4 Gbit/s transmission rate. LVDS is not recommended for data transfer over distances of more then 5 meters. If such transmission distances turn out to be necessary, fibre optical transmission would be a better choice.

SCS proposes to use the 14526-EZHB-200-0QC type cable and the 10226-1210VE board connectors as recommended by 3M. A very similar cable by the same manufacturer is successfully used by SCS with a different LVDS chipset.

Data transmission between the different L2 FTT cards will be done using the same physical layer with a different protocol.

In order to allow the PLL of the receiver to stabilize, the transmitter cannot be switched of completely when no valid data is transmitted. Valid data has therefore to be clearly identified (a valid 'zero' must differ from 'no valid data'). The following usage of the 48 bits has been proposed:

Usage	# bits	Decoded by
Cell number + Data valid	5 bit	HW + FPGA
Mask number	17 bit	FPGA / DSP
z (after division on L1)	8 bit	FPGA / DSP
Kappa, Phi for L2 linking	16 bit	FPGA
TOTAL	46 bit + 2 Spares	

table 1: Data format for track segments from L1.

Note that the layer number does not have to be communicated to L2 since it is defined by the input port of the L2 linker board. For the transmission of the segments from linker to fitter boards, the layer number will of course be necessary, however the 16 bit for (Kappa, Phi) are not necessary any more at that stage.

Valid cell-numbers range from 1 to 30. 'data not valid' could therefore be defined as all 5 bits of the cell number being '0'. The non-existing cell -number 31 would be used to transmit system data from the L1 cards to L2 (e.g. 'end-of-event'). A more convenient definition of 'data valid' would be, to take one of the presently unused spare bits.

Note that 'data not valid' has to be decoded by hardwired logic on the input cards and therefore

- cannot be changed by software;
- has to be identical for the data arriving from L1 and the data exchanged between the different L2 cards.

No error detection such as parity-check or CRC has been requested.

3.4.2 Input data from other H1 subsystems

It is assumed that input from additional L1 systems is needed for the calculations on the DSPs. The timing of this data is not critical as almost the entire linking step has to be finished before fitting starts. The format of the input data has not to be defined fully at the present time. If it is provided using the same LVDS chipset as the L1 FTT, no additional hardware is needed on L2 FTT. If the data is sent using any other standard, an adapter card fitting onto the L2 FTT piggyback input connector would have to be built. The specific needs of such an adapter card in terms of power supply, clock signals etc. which have to be provided by the main board, will have to be fixed in the detailed specification of the L2 FTT.

Similarly, any data output can be realized from the standard L2 FTT card, adding additional piggy back cards.

3.4.3 I/O from/to the central trigger.

Even though it seems feasible to run the L2 FTT without input signals from the CTL (deducing e.g. the L1 keep signal from the data delivered by the L1 FTT), it seems to be useful, to foresee a few digital inputs e.g. to finish all tasks and to soft-reset the system by the 'AClr' signals, or to reset internal counters at run start. The physical definition of these inputs will be fixed in the specification.

To provide the L2 FTT trigger elements to the CTL, an additional piggy back card will be necessary.

3.4.4 Data output to the L3 system

At present it is planned, to feed off-the-shelf VME processor cards with the L2 output data using LVDS interface cards on the processor cards PCI-Mezzanine-Card (PMC) extension connectors. PMC cards are connected to the processor by a PCI bus. As these local PCI busses could be used in parallel on all cards, the bandwidth should be sufficient.

It is not yet clear, whether a well suited LVDS mezzanine card is available on the market.[1] The cost for developing such a card especially for the L3 FTT should be avoided.

A protocol for all LVDS data within the different L2 and L3 cards will have to be defined.

3.4.5 Data output to the other L2 systems

Output to L2NN / L2TT can be done on any fitter card using additional piggyback cards. The data format needed here is not yet defined.

---

[1] A PCI Mezzanine card (PMC) which would solve the problem is beeing developped by SBS (see [7]). The final design and price are obviously not yet fixed - access to the LVDS pins of an ALTERA 20K300E is however promised. Data transmission from L2 to L3 would be possible this way, if the LVDS outputs of the IO Controller on the L2 decision board are used. LVDS connectors would have to be placed next to the FPGA. Whether the LVDS serializer chips used on L1/L2 could be used as well is unclear at the moment.

3.4.6 VME interface

The usage of a VME rather than a CompactPCI interface seems to be unavoidable in order to fit into the H1 readout system. The interface must allow to configure the FPGAs, to boot the DSPs, to download constants to the DSPs (e.g. the vertex location) and to read data from FPGA and DSP after an event is accepted. Readout is done only after an event is kept by L2, download of constants only at runstart.

In addition to the VME interface, the possibility to configure the FPGA via serial interfaces (JTAG) will be implemented anyway, in order to allow easy configuration in the testing / commissioning phase. Adding Eprom based memories to this serial bus, automatic configuration of the FPGAs at powerup is possible without VME access. The merger cards, where no readout is needed and changes in the FPGA code are unlikely, could therefore be placed in a dummy crate, providing power supply but no VME bus.

4 Solution

4.1 Hardware

4.1.1 Overview

An overview of the data flow is given in figure 1. The L2 FTT consists of four types of cards:

- The MERGER card is needed to collect the data from the different L1 cards of each layer. The main characteristics are:
  - 6 LVDS inputs from the L1 cards.
  - 1 LVDS output to the linker board.
  - An FPGA to multiplex all input data to the output.
- The LINKER card collects all track segments and scans all layers for coincidences in the kappa-phi plane. The coordinates of the linked track segments are communicated to the fitter cards.
  - 5 LVDS inputs from the merger cards.

- 2 (as may as possible) LVDS outputs to the fitter cards (daisy-chains).
- An FPGA for the linking task.
- VME interface for readout.

- Each FITTER card carries 4 DSPs to calculate the parameters of the tracks. The accuracy of the kappa-phi determination can be increased significantly on this step. The parameters of the fitted tracks are transmitted via daisy-chain.
  - 1 LVDS input from the daisy-chain.
  - 1 LVDS output to the daisy-chain.
  - 1 Input for the z-Vtx histogram from the MWPC trigger (on one / a few cards only).
  - Outputs to L2TT / L2NN (on one / a few cards only).
- The DECISION card is similar to the fitters. Output of all fitter cards is collected here and an L2 decision based on this information is formed by DSPs. Calculation of e.g. momentum sums starts as soon as the first track data is available.
  - 2 (as may as possible) LVDS inputs for the daisy-chains.
  - 1 Output to the CTL L2.
  - 1 (or more) outputs (APEX LVDS?) to L3.

All cards have a large number of IO connections which must be handled by an FPGA. The functionality of the FPGA is very simple, except for the linker. DSPs are necessary only on the fitter/decision boards. In spite of these differences it is still advisable, to use the same card layout for all three purposes: initial costs for design, layout and production would be much higher if different designs are used. The economizations due to smaller prints and cheaper components in the case of three different designs play a much smaller role because no more than approximately 20 boards are needed in total.

Using one single layout, the choice on whether to populate all boards identically (making bookkeeping with FPGA software and with spares simpler) or not (saving cost for large FPGAs and DSPs), is with the customer.

The boards will carry 2 connectors for piggyback cards on each side. Every piggyback card can provide

- two LVDS inputs or
- one LVDS input and one output or
- IO channels to interface with other trigger systems.

Each connector will provide 48 parallel IO connections, attached to the IO-Controller FPGA.

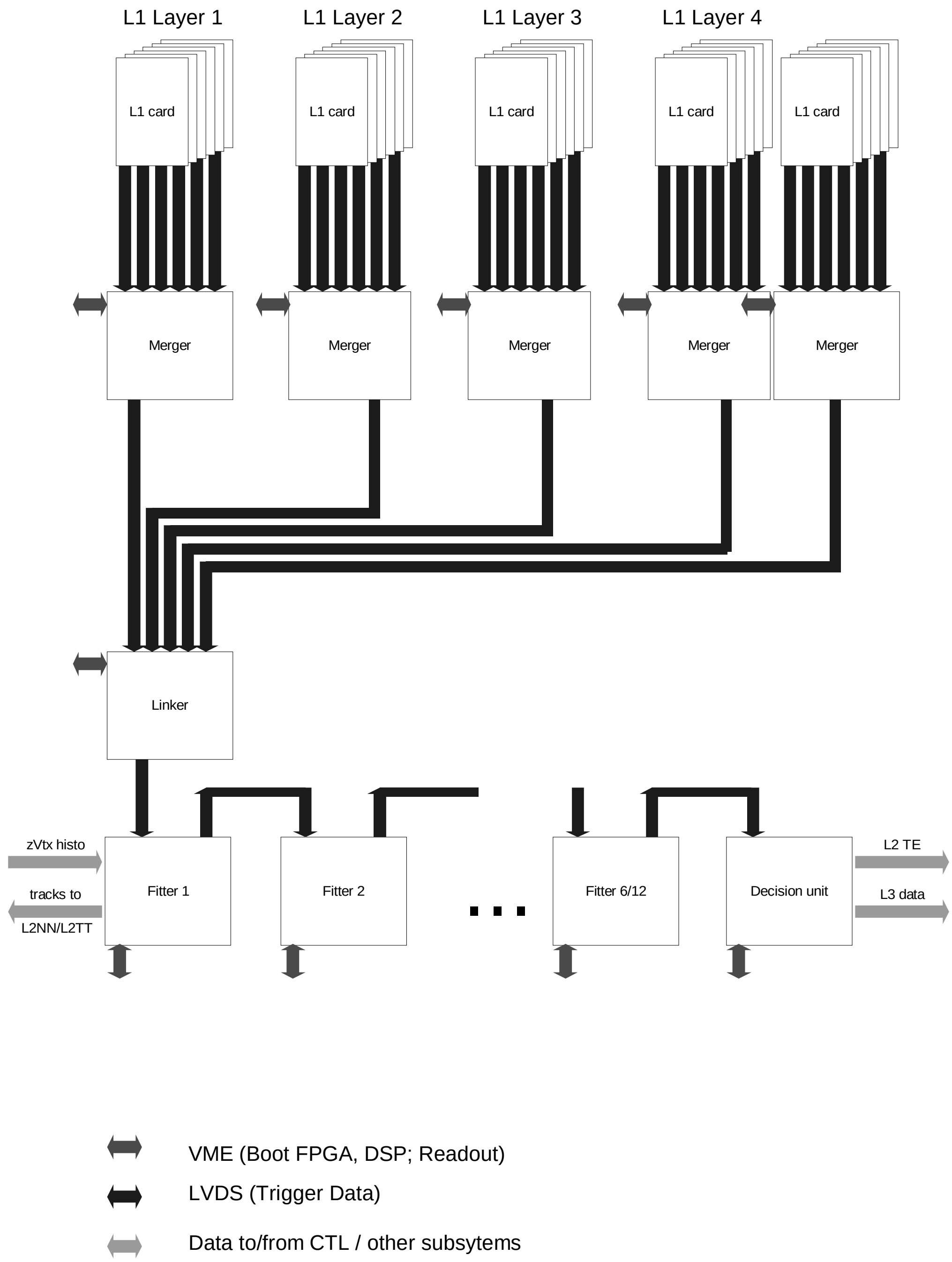

figure 1: Overall view of the L2 FTT data flow using only one daisy-chain for the fitter boards. For the 48 DSP system, a second daisy-chain will be built in parallel in order to reduce the latency.

4.1.2 Multi-purpose L2 FTT card

An overview of the functionalities of the main board is given in figure 2. The task of interfacing the IO piggybacks and the DSPs has been split on to two FPGAs in order not to have to go to the very largest devices available (number of available IO pins). The VME interface will most probably be implemented in a third, small FPGA with its configuration permanently stored in a EPROM based memory

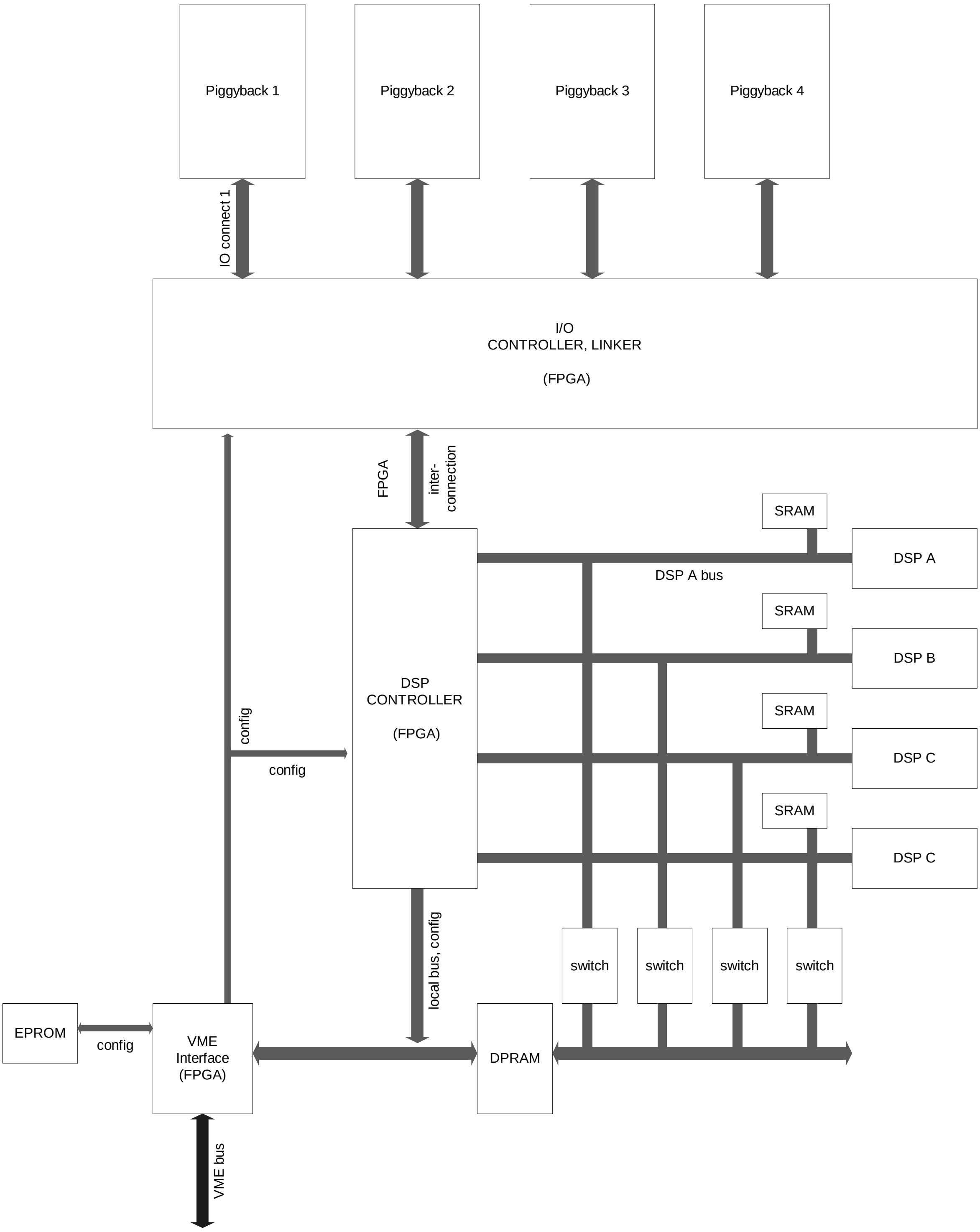

figure 2: Block schematic of the L2 FTT multi-purpose card.

4.1.3 Starting up

After power up, the VME controller will configure quickly. Download of the application dependent code for DSP controller and IO controller is then possible via VME. After initializing the FPGAs, the controlling PC writes the DSP code into the dual port RAM. On completion of this, a command is written to the DSP controller which starts one DSP after the other: the according bus switch is enabled and the reset signal is released. The DSP will load its code from the DPRAM. When all DSPs are running, user constants (e.g. LookUp Tables) can be loaded via VME into the DPRAM from where they are copied by the DSPs into the SRAMs.

4.1.4 IO Controller

Depending on the card function, this FPGA has to fulfill completely different tasks:

Merger: Multiplex data coming from 6 data streams to 1 output stream. One or two input streams must be read at a time, the output stream operated at the same time. The code needed is very simple; FIFOs, Multiplexer; 50 MHz.

Linker: Receive data from all 4 layers in parallel, write to RAM and CAM at the same time. As soon as all data is received, the linking process starts and linked track segments are sent to the daisy-chains. The code is very complex and must reach high speed (100 MHz); see 4.2 for more about the algorithm.

Fitter: Receive data from the daisy-chain, retransmit with minimal delay; recognize data needed by the DSPs of the board, send fit results into the daisy-chain.

Decision card: Receive data from fitters, communicate to DSP controller. Transmit trigger decision to CTL; transmit data to L3, using piggyback card or alternatively LVDS outputs of FPGA.

The number of IO pins needed is identical for all card:

Purpose	Pin count	
Configuration	5	5
IO	4 x [48(data)+12(control)]	240
FPGA interconnection	48(data) + 8 (control)	56
Dig IO, LEDs, Test	25	25
TOTAL		326

table 2: Pin count IO controller.

The ALTERA 20K200E type FPGA in the 484 pin fineline ball grid array (FBGA) case provides 376 user IO pins which would perfectly match our needs here for mergers and fitters. The slowest speed grade (-3) will most probably be sufficient to reach the required 50 MHz on mergers/fitters.

On the other side, the layout-compatible 20K400E to 20K1000E devices in the 672 pin FBGA case can be used for the linker board. The fastest speed grade will certainly make sense there. If such a device is placed, the FPGA area is increased from 22x22 to 26x26 pins. LVDS IO pins are located in the outermost two rows i.e.. exactly in the area covered in addition now. It is therefore possible, to have these pins connected to optional LVDS connectors.

4.1.5 Piggyback IO connector / cards

Four identical connectors will be mounted on the main board, two on the top side, two on the bottom. 48 data bits must be transmitted bidirectionally at 50 MHz. Another 48 pins are therefore needed for grounding on that connector. In addition a few control signals (select IO channels, signals from the FIFOs, clock) are necessary. In total a 150 pin connector will be sufficient.

As several other IO piggybacks may be needed in order to integrate the FTT into H1, the risk of plugging a wrong piggyback card to a main board cannot be neglected. If an input card is placed on a board where the FPGA is programmed to interface to an output card, serious damage of the FPGA as well as of the input card is to be expected. A coding system pulling e.g. 4 pins differently to GND / +3.3V on each type of piggyback card would allow the FPGA to check at startup whether the correct cards are mounted and to keep the IO pins tristated in case of a mismatch.

Every connector can carry one piggyback card which must be smaller than half a 6U VME card (233 mm / 2 ~= 110 mm). The proposed connectors for the LVDS signals are 38 mm wide (including case), in addition some space (2 mm) is needed between two of them. Large adapter cards could be mounted if the neighboring piggyback connector is not used.

The usage of piggyback cards on both sides of the main board will increase the horizontal space needed by the cards - most probably beyond the width of single VME slots thus reducing the number of cards which can be placed per crate.

4.1.6 Local Bus, FPGA Interconnection

During configuration, both FPGAs are accessed by the VME controller via their dedicated configuration pins.

The local bus is controlled by the VME controller, providing 32 data bits and 16 bit address space. VME controller, DSP controller and DPRAM are treated as memory banks, adding 3 additional bits (chip selects).

The FPGA interconnection is 24 bits wide in every direction and operated at 100 MHz corresponding to the full data throughput of 2.4 Gbit/s of the daisy chain.

4.1.7 DSP Controller

This FPGA provides the interface for the four DSPs and is connected to the VME bus via the local bus. It receives data from the IO controller via the FPGA interconnect and sends the DSP output back there. In addition this FPGA will be responsible to store the data needed for the readout or to put it via a DSP to the DPRAM.

Again, pin count shows that at least the 20K200E device is necessary. From experiences with ALTERA 10K devices, we can predict that the available logic cells will be sufficient and the slowest 20KE devices will be fast enough for that task.

Purpose	Pin count	
DSP IF	4 x [32(data) + 16(address) + 10(ctrl)]	232
FPGA interconnection	48(data) + 8 (control)	56
Local BUS	32(data) + 16(address) + 8(ctrl)	56
Test, LEDs…	25	25
TOTAL		369

table 3: Pin count DSP controller.

4.1.8 VME Interface

The VME standard has historically grown and is quite complex. Only a subset of the functionalities of the standard is needed and will be implemented:

- Write to the L2 FTT card; Depending on the address the data might be (see section 4.5 as well)
    - FPGA Configuration
    - Commands to the FPGA
    - DSP Code (via DPRAM)
    - DSP Constants (via FPGA or DPRAM)
- Read from the L2 FTT card; Easy access is possible, if all readable data is kept ready in the VME Interface FPGA. An ALTERA 10K30E device could for instance provide 6 memory blocks of 512 byte each. Larger amounts of data would have to be stored in the DPRAM or in the DSP controller, making VME access across the interfacing FPGA more complex.
- No interrupts or bus requests will be possible from the L2 FTT cards.

Purpose	Pin count	
VME bus	32(data) + 16(address) + 8(ctrl)	56
Local bus	32(data) + 16(address) + 8(ctrl)	56
Configuration	5	5
LEDs, Test	10	10
TOTAL		127

table 4: Pin count VME controller. The 208 pin PQFT package (147 user IOs) or the 256 pin FineLine BGA package (176 user IOs) of the 10K30E FPGA could be used.

4.1.9 DSP

Four DSPs of the TMS320C6x01 series by Texas Instruments will be placed on the fitter boards. If floating point arithmetic is needed, x stands for 7, otherwise the fixed point version 6201 can be used. The two DSPs are completely pin-compatible such that the final choice can be postponed until the devices have to be ordered. The more recent C6x02 series would have the benefit of higher computing power.

However the pinout is different and the introduction of the floating point version of that device on the market is not yet scheduled.

The DSPs are connected through the 'extended memory interface' (EMIF) to the DSP controller, the SDRAM and to the DPRAM. As multi processor busses are poorly supported by that type of DSP, each of them is connected by a separated bus. Data transfer is done by DMA. SDRAM can be accessed at 83-100 MHz (depending on the core frequency of the DSP), the FPGA at around 35 MHz; data transfer from the DPRAM is possible at even lower speed only.

Additional serial communication channels, interrupt and flag lines allow communication with the FPGA (e.g. 'new track segments available' or 'fitting procedure finished').

4.1.9.1 SRAM

SRAM can be added to the DSP busses. Possible choices are:

- SDRAM (Synchronous Dynamic RAM):
  - Pro: 8 MByte are easily possible, access at 100 MHz is possible
  - Con: ~15 cycles are necessary to open pages (latency for random access)
- SRAM (Static RAM (asynchronous))
  - Pro: low latency, access at ~50-80 MHz
  - Cons: more expensive, typ. up to 1 MByte

The RAM would e.g. allow to store lookup tables where track segment numbers can be translated to (x,y) coordinates. Access to the RAM is independent of the status of the other DSPs whereas sharing a lookup table on the DPRAM would require additional arbitration and be much slower.

4.1.10 Power Supply

Supply of +5V and +12V will be provided via the backplane. For the L2 FTT most likely 3.3V, 2.5V and 1.8V DC will be needed in addition. The necessary circuitry will be placed on an additional piggyback PCB, placed at the rear end. The power supply card will be equipped with an additional connector which can be used to provide the input if the system is not installed in a VME crate.

The total power consumption of a fully equipped board is estimated to be in the order of 15 W.

4.1.11 Data transfer within the L2 cards

Transfer from the mergers to the linker board is simple. The time to transmit the 128 segments per layer one after the other is included in the timing considerations.

The linker board will have to transmit all linked segments to the fitters. During 528 search cycles @100 MHz (5.28 us), a maximum of 48 links of 4 sements (20 ns transmission time each) – which correspond to 48 x 4 x20 ns = 3.84 us – are found. As the 48 links are distributed more or less randomly over the 528 searched seeds, the 72% load on the link seem to be too high. If two or even three daisy chains are used in parallel, the load can be reduced accordingly and becomes less critical. If too many links are found with the last searched seeds, some additional latency due to the link will still arise. No detailed studies have been carried out on that topic and no time is attributed to that effect in the timing overview!

At the end of the fitting, the data volume is reduced rather the increased (2 to 4 segments are transformed to e.g. one word for kappa/phi and maybe a second word for theta). The load on the link between fitters and decision board will therefore not be critical.

4.2 Linking

4.2.1 Description of the algorithm

4.2.1.1 Requirements

Different algorithms have been discussed. The following requirements had to be met by all of them:

1. Receive data at full speed (segments @50 MHz on every layer, all layers in parallel).

2. Evaluate a search seed every clock cycle. This requirement dictates the use of the 'encoded' operation of the CAMs: the location where a match is found is encoded in 5 bits, if more then one hit are found at the time, a nonsense bit pattern will be delivered instead.

3. If locking certain bins is necessary, it should be done only once per layer of search seeds (no flushing of pipelines for every single link possible; avoid dependencies between segments following each other in the search).

4. The algorithm should depend as little as possible on the ordering of the track segments which are received at random.

The algorithm which is expected to be best has partially been modeled in VHDL and fitted onto a 20K100EQC240-1 device (fastest speed grade of a chip which can only hold some parts of the logics needed).

4.2.1.2 Overview algorithm

The linking is performed in a 2dimentional array, we use the convention to plot phi on the horizontal, kappa on the vertical axis. Every L1 layer contributes such a 2d array, which contains logical variables (segment present or empty). If several segments of one layer fall into the same bin, an arbitrary decision, which one to take will be unavoidable.

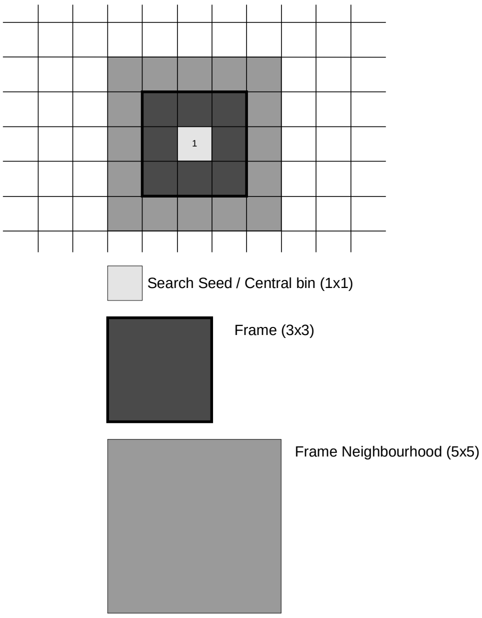

figure 3: Naming convention for linking.

The search goes through all segments of all four layers. Every single search starts from the bin, the track segment lies in - the 'search seed'. Allowing a certain uncertainty in the kappa/phi-determination, this segment can be linked with any other segments within the same 3x3 bin 'frame' centered around the seed. A 'link-quality' can be calculated for the central bin, weighting hits on the other layers e.g. as shown in table 5.

bin	weight factor
same bin	4
left, right, top, bottom bin	2
upper-left, upper-right, lower-left. lower-right	1

table 5

A simple algorithm would now just apply a threshold to the 'link-quality' to decide whether the link should be considered sufficient or not. However there is the possibility, that we would have found an even better link, if the search started from one of the neighboring bins. The entire search / link-quality calculation is therefore done for all bins inside the 3x3 frame - which makes it necessary to search through all bins in the 5x5 bin 'frame neighborhood'. If a higher link quality is found in a neighboring bin, the link is rejected; the segments yielding to the higher link-quality in the neighboring bin will be found later anyway and must not be considered any further at the moment.

An additional veto rejects the link, if segments from a lower level are present in the central bin, because that link has already been found when the seeds from the lower level were searched.

To execute this entire analysis once per clock cycle, 25 CAMs per layer have to be searched in parallel. Bin numbers have therefore to be divided into 25 groups, such that every possible frame neighborhood contains one bin of every group - dividing both indices of the bin by 5 delivers a group-id (the remainders) and a bin-sub-number (the integer quotient).

4.2.1.3 Filling CAMs

Filling is completely parallel and independent for every layer. Bin numbers have to be divided by 5 and the according bin-sub-numbers have to be given to the 'cam controller' of the according group. That logical unit ('entity') has to perform the following operations on the CAM:

1. Read (search) whether that data is already present in the CAM.
2. Evaluate the result of the previous step.
3. Write step 1 (input data, if this is not yet present, illegal data otherwise).
4. Write step 2.

For one layer this procedure has been fitted onto a 20K100EQC240-1 FPGA. In case of a match in step 1, all bits of the input data were set to '1' at step 2. With one single step between search and write, only 60 MHz could be reached this way. If the match flag was ignored, 175 MHz were possible. The intended functionality can certainly be reached at 100 MHz or more if the pipeline were extended by a second evaluation step, giving a data-throughput of one word every 50 ns. A simple alternative which would require 4 steps and run at high speed is, to input the 'match flag' of the CAM to an otherwise unused bit of the data which is written into the CAM - leaving all other bits unchanged, no matter whether data is valid or not. In the searches this stored 'match found' bit would simply have to be required to be low.[2]

Data arrives from L1 at 50 MHz - however every word has to be written only into one out of 25 CAMs. A FIFO on the FPGA would therefore allow to stall the writing operation whenever two segments following each other are to be written into the same CAM. As long as the segments are to be written to different bins, the FIFO can be read at 100 MHz until it is empty again (no new data from L1 available yet). The only way, delay could arise from this filling procedure is, if the last segments of a layer are all to be written to the same CAM (delaying by 20 ns or 30 ns per segment).

Less then 300 logic cells have been necessary to implement that model (including 'write' and 'read' operation).

4.2.1.4 Searching links

Searching through the CAMs (32 bit wide, 32 word deep) is possible at up to 175 MHz, one search per clock cycle. This has been tested together with the writing sequence for the 25 CAMs of one layer.

For the evaluation, a second test program was written. Starting point was an array 5x5 of logicals ('segment found') for each layer, the expected as output from the CAMs. The following 5 steps have been defined for the evaluation pipeline:

1. sum number of hits of all layers of each bin (resulting number 0..4; for each of the 25 bins in parallel)

[2] Misfortunately this option could not be tested any more at SCS because the borrowed memory of the PC was already removed again. It will be done in the implementation phase of the project

2. For each of the 9 bins in the 'frame', sub-sums of 3 bins forming one row (including weighting) have been built. (3x9 times in parallel).

3. Sum the three sub-sums to the full sum ('link-quality'). (9 times).

4. Compare central bin to each of its 8 neighbors ('neighbor > central bin',8 times in parallel).

   (+ compare central bin to threshold for 'good link' + check whether link has already been found before; not simulated)

5. ORe result of comparisons.

That algorithm consumed 679 logic elements but no ESBs. 139 MHz were predicted for the 20K100E-1 device. Together with the determination of correct bin-sub-addresses and searching through the CAMs and vetoing in case of segments in lower levels, a total pipeline length of 10 steps seams feasible.

4.2.1.5 Locking

With the present scheme, no locking of bins as discussed earlier is necessary any more.

4.2.1.6 Binning

The binning of the 2d histogram will have to be optimized using detector data. From the technical side, the only limitation is, that no more then 2^{31} bins are possible (CAMs are 32 bit wide, 1 bit is used as 'data valid' flag).

4.2.1.7 Resource usage

25 CAMs per layer are necessary with the scheme described. As every segment is stored in only one of these CAMs, the length can be reduced to 32 word (even shorter CAMs have been discussed but there is no gain, if only 16 words are used since the remaining space cannot be used otherwise).

If no locking is performed, all four layers have to be written into CAMs which means that 100 ESBs are needed.

SRAMs are necessary on the FPGAs to store all track segments: 4 layers x 128 segments x 48 bit = 12 ESBs.

In addition a lookup table is necessary, to find the way back from the address of a hit in the CAM to the SRAM where the full information of all track segments is stored in the FPGA. 25(32) CAMs x 32 word = 800 (1024) addresses / layer are to be stored. As the 128 segments per layer require 7 bit addressing, the 256 x 8bit configuration of the SRAM on ESBs will be used. 4 layers x 1024 addresses x 8 bit = 16 ESBs.

In total 100(CAM) + 12 (Track segments) + 16 (LUT) = 128 ESBs are needed. Further ESBs will be needed for FIFOs and to implement some logic ('product terms', address decoders etc.). The 20K600E device with 152 ESBs seems to be sufficient. This device will contain 24320 logic elements - far beyond of what is to be expected based on the tests in 4.2.1.3 and 4.2.1.4.

4.2.1.8 Timing linking

Linker and DSPs can be operated asynchronously

Task	Steps	Steps	Freq.	Time
Receiving	128 + 2 (pipe)	130	50 MHz	2.60 us
Filling	6 (pipe) + 4 (stall, ?)	10	100 MHz	0.10 us
Search	4 x 128 (seeds) + 12	524	100 MHz	5.24 us
TOTAL				7.94 us

4.3 Fitting of Tracks

4.3.1 Algorithm implementation[3]

The non-iterative track fitting algorithm by Karimäki has been simulated for the DSP. All distances have been measured in cm such that high powers of numbers made floating point arithmetic necessary. All variables have been restricted to 'single precision' (32 bit). The following restrictions have been applied to the algorithm

- 3 to 5 points (vertex + 4 segments).
- track segment input as (phi, r).
- All variables depending on the nominal vertex position are already calucalted.
- no coordinate transformation (origin equals center of CJC i.e. radius of every group independent of phi).
- Taylor-expansion to the second order for sin/cos functions and for other terms where possible.
- Usage of DSP intrinsic function for divisions and square roots with reduced accuracy.

The accuracy reached is clearly sufficient, no more than 680 cycles are necessary on the TMS6701 processor. Some further optimization potential exists, if the machine-generated assembler code is hand-optimized - to expect factor of 2 is however not realistic. For the overall timing considerations, 6 us per fit are assumed. This time will have to include in addition to the r-phi fit the z-fit which is thought to be much less time-consuming as well as data-transfer on the fitter board: data received by the IO Controller must be passed to the DSP controller which immediately gives an interrupt to the waiting DSP. The DSP in turn starts the DMA to receive the data (DMA can even start before all data is present in the DSP controller, calculation on the DSP may already start when the first data is available). At the end, data has to be transferred from the DSP back to the IO controller.

[3] Thanks for the fruitful collaboration with C. Wissing, Uni Dortmund.

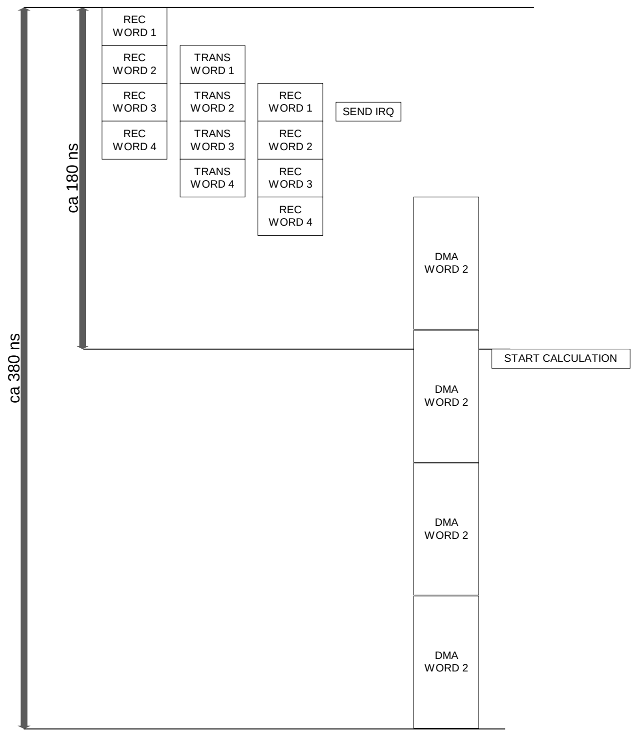

figure 4: Delay on the fitter board: four 48 bit words are received at 50 MHz on the IO controller (first column). Data is transmitted to the DSP controller (third column) where an interrupt is send to the DSP. After approximately 30 ns the DMA starts (fifth column), reading one 32 bit word every 35 ns. Calculation on the DSP starts as soon as the first track segment has arrived. As an alternative, only one word per track is read first (e.g. mask number) and the remaining data (z information) of all segments is read while the r-phi fit is being done. Results can be written back to the DSP controller as soon as they are available, transmission to the IO Controller starts with the last word. Another lead time in the order of 200 ns is to be expected.

4.3.2 Alternatives

Even though the proposed algorithm has been shown to be sufficiently fast and accurate, the following alternatives to optimizing the existing code could be investigated in addition:

- Choose better suited coordinate system such that fixed-point arithmetics are sufficient. Keep in mind, that this would not only mean to increase the clock frequency from 166 to 200 MHz, but that in addition the number of clock cycles needed to perform the task could be reduced.

- A completely different algorithm might be used. As it is known from the linking step, which points are to be combined, the task is in principle much simpler than the one solved by the Karimäki algorithm (where it is not known a priory, which points to use and which ones to reject). It has not been checked, whether a chi-square minimization algorithm similar to the one described in [8] could be used. This algorithm would have the advantage that it mostly uses 'DSP like' functions (multiply-accumulate) and might be possible in fixed point.

4.4 Forming L2 Decisions

No code to form L2 decisions has been tested on the DSP platform. The task is considered to be non-critical since it consists mostly of summing up variables, which can be done very fast by the DSPs on the decision card, starting as soon as the first track is fitted. For the non-hidden calculations (last tracks + find decision + transmit result to CTL) 3 us (equivalent to 500 clock cycles of the 166MHz DSP core) are assumed.

4.5 L2 Timing

4.5.1 Transmission delays

The latency due to serializing the data when transmitting via LVDS enters twice in the transmission of data from L1 to L2 plus up to 13 times, when data is transmitted through one single daisy-chain from linker to decision board. The number of delays could be reduced considerably, if a tree structure rather than a daisy-chain is used.

The manufacturer quotes a latency of 1.5 clock cycles + 4.5 ns latency for the LVDS transmitter, 3 clock cycles + 4.5 ns for the receiver. At 50 MHz this sum up to 99 ns. With the FIFO, another three clock cycles are foreseen on the receiver side.

The merger board will be equipped with 3 dual input piggybacks to receive the data and one output piggyback card. The additional delay from registers on the FPGA will be 2 clock cycles. In total we have to expect 10 clock cycles of 20 ns; another 40 ns have been added for contingency in the following calculations.

In order to reduce the latency in the daisy-chain, it is possible to retransmit the data as soon as it is read out of the FIFO, without the need of feeding it through the FPGA. This would reduce the latency by two clock cycles, it is however only possible, if only one single piggyback card is involved - as soon as a tree-structure is used, all data has to be fed through the FPGA.

For the following calculation of the overall timing, two identical parallel daisy-chains are assumed.

4.5.2 Overall timing

The times required for the tasks as described before are given in the two following tables. All numbers are considered to be conservative estimates, it may be that the 24 DSP solution will turn out to be possible during the implementation of the system.

Task		CC	Freq. (MHz)	time (us)	finished at
Latency L1-L2	1	24	50	0.48	0.48
Linking: receive data	2	130	50	2.60	3.08
Linking: fill CAMs (pipe + stall)	3	10	100	0.10	3.18
Linking: check CAMs	4	528	100	5.28	8.46
Linking: fill dirty CAMs	5	0	100	0.00	8.46
Latency fitter tree	6	48	50	0.96	9.42
Fitting 1, including delays on fitter-board	7	1000	166	6.02	15.44
Fitting 2	8	1000	166	6.02	21.47
Sums / eval	9	500	166	3.01	24.48
link/fitting overlap	10	-72	100	-0.72	23.76
SUM				23.76	
SPARE TIME				-3.96	
TOTAL TIME				19.80	
TIME USED				120%	

table 6: Timing overview; Linking without locking, Fitter tree, 24 DSPs. Another 4 us would have to be gained, which may turn out to be possible during the implementation of the system, if fitting or evalutating can be optimized.

		CC	freq (MHz)	time (us)	finished at
Latency L1-L2	1	24	50	0.48	0.48
Linking: receive data	2	130	50	2.60	3.08
Linking: fill CAMs (pipe + stall)	3	10	100	0.10	3.18
Linking: check CAMs	4	528	100	5.28	8.46
Linking: fill dirty CAMs	5	0	100	0.00	8.46
Latency fitter tree	6	84	50	1.68	10.14
Fitting 1, including delays on fitter-board	7	1000	166	6.02	16.16
Fitting 2	8		166	0.00	16.16
Sums / eval	9	500	166	3.01	19.18
link/fitting overlap	10		100	0.00	19.18
SUM				19.18	
SPARE TIME				0.62	
TOTAL TIME				19.80	
TIME USED				97%	

table 8: Timing overview; Linking without locking, Fitter tree, 48 DSPs. All estimates are conservative, 0.62 us of spare time are available.

4.6 Controlling the L2 FTT

Being embedded into a complex system, different flows of control signals can be defined. Possible sources of commands are:

- VME
- CTL signals via dig. input on main boards
- LVDS

It is proposed, that every card receives a set of control signals from the CTL (e.g. Runstart, L1Keep, AClear) which initiate tasks on the boards. The completion (successful or not) could be communicated via LVDS. All these 'end-of-action' signals will be received by the L2 decision card where bookkeeping can be done. Errors detected this way can be read out via the VME interface of the decision card.

For commissioning, all cards can be placed in VME crates where they are directly accessible to a PC. Looping back IO signals, standalone testing of most components is possible.

4.7 Space requirements

The L2 FTT will be realized on boards fitting into standard 6U VME crates. Due to the piggyback cards mounted on both sides, most probably each card will cover more space than the width of a single slot, reducing the number of boards which can be placed in a crate.

5 List of objects

5.1 List of figures

5.2 List of tables

5.3 References

[1] A Fast Track Trigger with High Resolution for H1, 1999, PRC 99/06 (H1 Internal)

[2] Addendum to the Proposal [1], 1999, PRC 99/07 (H1 Internal note H1-09/99-576)

[3] http://www-h1.desy.de/idet/upgrade/trigger/ftt/welcome.html

[4] http://www.national.com/pf/DS/DS90CR483.html
http://www.national.com/ds/DS/DS90CR483.pdf

[5] http://www.altera.com/document/ds/apex.pdf Datasheet rev. 2.05 (Nov. 1999)

[6] http://www.ti.com/sc/docs/products/dsp/tms320c6701.html

[7] e-Mail by SBS, see attachment

[8] http://www-cdf.fnal.gov/upgrades/tdr/tdr_12_trig.html
http://mccdf3.pi.infn.it/SVT/svt_tdr.ps