

FTK Data Organizer Firmware Design Summary

1. Introduction

2. Structure

- a. Input
- b. Output
- c. External memory interface
- d. Double DOs
- e. Parallel Structure
- f. On-chip memories
 - i. HCM (HitCountMem)
 - ii. HLP (HitListPointer)
 - iii. HLM (HitListMem)

3. Logic Functionality

- a. Write Mode
 - i. Input from transceiver
 - ii. MLDATA
 - iii. MLADD
 - iv. HLCtrl_WR
- b. Read Mode
 - i. AMMapInt
 - ii. MLRDDATA/MLSECTORID/MLLayerMap
 - iii. MLRDADD
 - iv. HLCtrl_RD
 - v. Output to TF
- c. State Machine
 - i. DO1
 - ii. DO2
 - iii. DO_top

4. Resource Usage

- a. Memory Usage
- b. Logic Usage
- c. Total Resource Usage

5. Timing and other issues

1. Introduction

a. FTK and the ATLAS detector

The ATLAS detector is a multi-layered, general purpose particle detector on the Large Hadron Collider. The FastTracKer(FTK) reconstructs the particles' trajectory through its inner detectors, namely the Pixel(pix) and SCT detectors.

For a visual overview of FTK hardware system, see **Slide2**.

b. Hit:

The locations of the particles' path through the layers of ATLAS inner detectors (3 pix layers and 5 sct layers), i.e. hits, are stored in a series of coordinates to identify the positions.

The Hits are stored as a 32-bit word for the PIX layers, and 16-bit word for the SCT layers.

c. SSID:

The inner detectors are sliced up into small sections called SuperStrips (SS). The position of the SS is translated into a 14-bit word called the SSID. During one particle colliding-splashing-passing event, a specific SS could have no hits, 1 hit, or multiple hits, hence the hits can be grouped according to their SSID.

d. Don't Care (DC):

As mentioned above, we group and match hits according to their SSID. Sometimes we may not want to match the entire 14-bits, so we "Don't Care" the last few bits of the SSID. In the current design, we may use up to 3LSB of the SSID as DC bits.

In latter paragraphs, the entire 14-bit SSID could be referred to as fullSSID, while the 11MSB of the SSID (without the 3LSB DC part) is called the AMSSID. If unspecified, SSID refers to the full 14bits.

e. RoadID

We have a giant LUT called the AMMap. RoadIDs are 21bit words that the AMMap use to fetch the corresponding SSIDs for an event.

f. Data Organizer(DO):

The DO stores incoming hits from the Data Formatter and stores them according to their SSID. For a given RoadID, the RoadID word is translated into SSIDs by the AMMap, and the DO retrieves the stored hits corresponding to the SSIDs.

Through smart firmware design, we would like to achieve

- Rapid storage and retrieval of hits according to the SSIDs
- Efficient resource usage

2. Structure

For an overview of the AUX card, see **Slide3**.

For an overview of I/O for the DO, see **Slide4**.

The DO firmware use a list of pre-defined signal types listed in DO_deflib.vhd.

a. Input (excluding external memory interface):

- top_clk : global clock with a frequency of 200MHz. The maximum frequency allowed by the firmware is currently closer to 140 MHz for the DO alone, and 90MHz for the whole system (which does not quite meet our goals).
- top_reset : global reset
- top_inrxchan[11][32] : input from transceiver; 32 bit words containing Hits & SSIDs. The transceiver sends Hits & SSIDs on 12 parallel streams, 2 streams per pix layer and 1 stream per sct layer
 - pix layer:
rxchan0 to 5 carry input for the 3 pix layers.
Each pix layer is split into two streams, (0,1), (2,3), (4,5): layers0,2,4 carry information for pix hits with odd SSIDs; layer1,3,5 carry information for pix hits with even SSIDs.

The pix hit is a 32bit word, so due to width constraints we cannot bunch pix hit&SSID into the same word. Instead they are sent out consecutively, with bit(29) reserved to indicate whether that word is a hit or SSID:

bit(29)=0, hit word, all 32 bits

bit(29)=1, SSID, 14MSB

▪ sct layer:

rxchan 6 to 10 carry input for the 5 sct layers respectively.

The sct hit is a 16bit word, so we can group the hit together with the SSID. bit(15..0), hit word; bit(29..16), SSID.

- EE[11] : this is the End-of-Event flag to indicate the last hit of the event has been received by the DO. In the final version EE would be part of the Hit Word package, and handled by a multi-layer synchronization structure. Right now before the data format is finalized, it is temporarily set as an 11bit input linked to one of the idle transceiver outputs, with 1 bit for each layer of hit input.
- top_inRoadchan[32] : input from transceiver; 32 bit word containing RoadID (24LSB) and HitMap(8MSB). The RoadID word is translated into SSIDs per layer by the AMMap; the HitMap indicates whether or not the specific layer has been hit (1 with hit; 0 no hit) for the 3 pix layers+ 5 sct layers.

b. Output (excluding external memory interface)

- top_hlm8L_outpixhit[3][32]: output to TrackFitter; Hits for the 3 pix layers sent out on 3 parallel streams respectively.
- top_hlm8L_outscthit[5][16]: output to TrackFitter; Hits for the 5 sct layers sent out on 5 parallel streams respectively.
- top_outRoadID[24]: output to TrackFitter from transceiver
- top_outsectorID[16]: output to TrackFitter from AMMap
- top_outlayermap[8]: output to TrackFitter from transceiver, indicating whether that layer has been hit
- top_hlc8L_SSlast[8]: state machine signal, indicating whether the last SSID from that event for that layer has been processed(i.e. last hit from that SSID sent out to TF); currently connected to temporary dummy outputs ; may be used later as part of control/enable signals for the whole AUX system.

- top_hlc8L_empty[8]: state machine signal, indicating whether that SSID has no hits (i.e. empty); currently connected to temporary dummy outputs ; may be used later as part of control/enable signals for the whole AUX system.
- top_hlcrd8L_ovflw[8]: state machine signal, indicating hit overflow (counter exceeding the HitCnt for that SSID in error) in the READ mode; currently connected to temporary dummy outputs ; may be used later as part of control/enable signals for the whole AUX system.
- top_hlcwr11L_ovflw[11]: state machine signal, indicating hit overflow (more hits than the maximum the HCM width allows) in the WRITE mode; currently connected to temporary dummy outputs ; may be used later as part of control/enable signals for the whole AUX system.
 - neither overflow cases should happen; in current compilation both sets of registers are stuck to GND
- top_hitee11L[11]: state machine signal, End-of-Event flag passed through the DO structure; currently connected to temporary dummy outputs ; may be used later as part of control/enable signals for the whole AUX system

c. External Memory Interface (AMMAP_int)

We use using RLDRAMII: MT49H16M36-25 as the external memory that stores the AMMap. The interface is generated using QuartusII 12.1 for Arria V.

See **Slide11**. (additional details to come)

d. Double DOs

As mentioned before, the DO has two states, WRITE and READ. For each DO, the firmware is duplicated, with one in WRITE mode on event n and the other in READ mode on event $(n-1)$. In WRITE mode, hits are written into the DO. In READ mode, hits are extracted for each road number sent by the AMB. When WRITE mode is finished, the state is switched to READ mode for the same event. When READ mode is finished, the state is switched to WRITE mode for a new event. A FSM controls which state either DO is in (details later). The AMMap interface sends out SSIDs and other information to the DO that is in the READ mode. A block diagram for the structure is shown on **Slide5**.

e. Parallel Structure

Inside each copy of DO, since for the WRITE mode, the incoming Hits and SSIDs are sent from the transceiver on parallel streams, one for each layer, and for the READ mode, the AMMAP translates the ROADID into SSIDs on parallel streams, again one for each layer, and the final output to the TF is also on parallel streams by layer, the DO has a fully parallel structure, i.e. the same (or similar if you consider the slight difference in data length between pix and sct layers) logic structure (including control logic and on-chip memories) repeated over 11 layers ($3*2=6$ pix, and 5sct).

A block diagram for one layer of DO structure is shown on **Slide7**.

f. On-chip memories

To achieve the DO's functionality of rapid storage and retrieval with efficient resource usage, the information of Hits w.r.t. their SSIDs are stored in 3 types RAM structures on-chip, the HitCountMem(HCM), the HitListPointer(HLP), and the HitListMem(HLM).

1) HitCountMem(HCM)

The HCM stores the number of hits in a given SSID. In the case of using DC bits (we use 3 DCbits, i.e. the 3LSB of SSID) in SSID matching, and because we do not have enough time to clear all HCMs at the beginning of each event, we came up with the following scheme for the HCM:

for each layer, there is a set of 8 identical HCMs, each 11bit deep (using the 11bit AMSSID, i.e. 11MSB of SSID, with the last 3 bits of SSID being DCbits). The width is the number of bits used to store the HitCNT, for which we use 5bits for PIX layers, and 3 bits for SCT layers.

The HitCNT for a specific full 14bit SSID will be stored in one of the 8 HCMs, i.e. HCM(i), the index i corresponds to the integer value of the 3LSB of its SSID (i.e. DCbits).

2) HitListMemory(HLM)

The HLM stores the hits, grouped by their SSID. (Note: the Data Formatter DF, which is upstream of the AUX card and sends out the hits, will group the hits according to their SSID, so all hits from a specific SSID will be sent out by the DF/received by the AUX consecutively.) To make maximum usage of memory space, the hits are stored consecutively on the HLM, which is 10bit deep, and

32bit wide for pix layers (32bit pix hits) and 16bit wide for sct layers (16bit sct hits).

3) HitListPointer(HLP)

Because the hits in the HLM are stored sequentially, a separate memory, HLP is created to store the address of the first hit with a specific SSID in the HLM. Hence the HLP is 14bit deep, corresponding to the 14bit SSID, and 10bit wide, corresponding to the address size of the HLM.

A summary table of the on-chip memories and their resource usage is shown on Slide14.

3. Logic Functionality

The basic functionality and structure of DO has been briefly described in previous parts. A summary can be viewed on **Slide6**.

The following paragraphs describe the WRITE mode and READ mode in further detail. Unless otherwise noted, the description is for ONE LAYER of DO structure, processing one hit input stream.

a. WRITE mode

Stores incoming hits according to its SSID.
Data/Logic flow is as follows:

1) Input from transceiver

2) MLDATA

3) MLADD

Input from the transceiver are identified as either Hits or SSIDs. MidLatch(ML)s are registers that sends out the input when the latch request is on; they are put in for synchronization purposes and to streamline the data flow. Hits are sent to MLDATA; SSIDs to MLADD; where they are synchronized and sent out in pairs on the next rising clock edge.

4) HLCtrl_WR

As mentioned above, we do not clear the HitCntMem after each event; in the case of DC bits, there is the possibility of mismatching the AM SS (SSID – DCbits) from a previous event.

We came up with the scheme of using 8 identical HCMs, each 11bits deep. The first hit of an AM SS is identified as the first hit with 11MSb of the SSID different from 11MSb of the SSID (i.e. AMSSID) of the previous hit. In the case of the first hit for that AMSSID, each address (addressed by that AMSSID) in all 8 HCMs is cleared.

To avoid clearing the same set of 8 HCM addresses multiple times within the same event, we use an $2^{11}=2048$ bit word DCReset to flag whether or not that set of HCM addresses have been cleared within the event.

- DC Reset
 - 2^{11} bits, each bit is a Reset flag for each HCM address respectively
 - Entire DCReset word set to “0” at beginning of event
 - Corresponding bit set to “1” when the content in the HCM with respective addr. is cleared
 - Only clear the corresponding HCM when its bit inside the DCReset word is “0”

So each HCM content could only be reset max once per event.

For each 14bit SSID,

AMSSID(10..0) <= SSID(13..3)

- AMSSID(10..0) translated to an integer AMSS_index range 0 to 2^{11} to index which bit in the DC_Reset word this AMSSID(i.e. HCM address) correspond to.

DCBits(2..0) <= SSID(2..0)

- DCBits(2..0) translated to an integer DCIndex range 0 to 7 to index which one of the 8 HCMs to write to.

The HLCtrl_WR can be roughly described as two parts.

- i. comparing fullSSID and calculate HitCnt

HLCtrl_WR compares incoming SSID (WR_tempSSID) with previous SSID (WR_savedSSID), and increments the HitCnt (TempHitCnt) when it's the same; if different, it means it's a new SSID and the HitCnt starts counting from 0 again.

Also in this part, the hit is being written to the HLM consecutively. In the case of a new SSID, the wraddr to the HLM is saved and written to the HLP with the address as the new SSID.

ii. comparing AMSSID and decide which HCM to write to

Incoming AMSSID[10..0] (WR_tempAMSSID) compared with previous AMSSID (WR_savedAMSSID) , simultaneously as fullSSID[13..0] is being processed as in part i above.

- if Same
Write TempHitCnt to HCM[DC_Index]
- if Different
Check DC_Reset[AMSS_Index]
 - » if DC_Reset[AMSS_Index]=0
Reset all 8 HCM at addr=AMSSID
Write TempHitCnt (should be 1) to HCM[DC_Index]
 - » if DC_Reset[AMSS_index]=1
Write TempHitCnt (should be 1) to HCM[DC_Index]

A block diagram of the WRITE mode is shown in **Slide8. Slide15&16** contain a brief description of the control logic as detailed above.

The resource usage summary for HLCtrl_WR is as follows: (shown for pix layer; sct layer similar)

```

+-----+
; Analysis & Synthesis Resource Usage Summary ;
HLCtrl_WR (pix layer)
+-----+-----+
; Resource ; Usage ;
+-----+-----+
; Estimate of Logic utilization (ALMs needed) ; 1875 ;
; ; ;
; Combinational ALUT usage for logic ; 2899 ;
; -- 7 input functions ; 8 ;
; -- 6 input functions ; 755 ;
; -- 5 input functions ; 2090 ;
; -- 4 input functions ; 4 ;
; -- <=3 input functions ; 42 ;
; ; ;
; Dedicated logic registers ; 2384 ;
; ; ;
; I/O pins ; 256 ;
; Total DSP Blocks ; 0 ;
; Maximum fan-out ; 2384 ;
; Total fan-out ; 24797 ;
; Average fan-out ; 4.28 ;
+-----+-----+

```

b. Read Mode

Sends out pre-stored hits corresponding to incoming RoadID; RoadID is translated into SSID by AMMap; output hits grouped by SSID

Data/Logic flow is as follows:

- 1) AMMap_int**
- 2) MLRDDATA/MLSectorID/MLLayerMap**
- 3) MLRDADD**

The DO receives the RoadID from the transceiver. On one hand, the RoadID is passed on through the MLRDDATA and sent out directly to the TF as part of the data stream; on the other hand, it is sent to the AMMap (stored on external memory) through the megawizard generated external memory interface (AMMAP_int).

The AMMap translates the RoadID into SSID+DCFlag for each of the 8 layers (3pix+5sct), sent out on parallel streams, where the 3 pix layers are again split to 6 by even/odd SSIDs, creating once again 11 parallel layers. The SSIDs and DCFlags are sent through MLRDADD to be latched at the next rising clock edge.

- DCFlag is a 2bit word that indicates how many DCbits are used for that layer, hence 0 up to 3 DCbits can be used.
 - » DCFlag, "00", "01", "10", "11"
 - » DCCnt= 0, 1, 2, 3 (No. of DCBits used)
 - » DCCombNum = 1, 2, 4, 8 (No. of possible SSIDs)
- DCComb(2..0): possible 3LSB of SSID with DC

In the meantime, the AMMap also sends out LayerMap (which indicates whether or not that specific layer has hits), SectorID; together with ROADID, these 3 pieces of information are passed through the respective MidLatches to be sent out to the TF downstream.

4) HLCtrl_RD

First, it calculates the possible DCComb(2..0) according to the 3LSB of SSID and the DCFlag

- For i range 0 to DCCombNum-1 LOOP
- If DCFlag = "00", DCCombNum=1
tempDCComb(2..0)= SSID(2..0)

- If DCFlag = "01", DCCombNum=2
tempDCComb(2..0)= SSID(2..1) + i (1 LSB)
 - If DCFlag = "10", DCCombNum=4
tempDCComb(2..0)=SSID(2) + i (2 LSB)
 - If DCFlag = "11", DCCombNum=8
tempDCComb(2..0)= i (3 LSB)
- tempDCIndex = to_integer(tempDCComb)
DCIndex(tempDCIndex)=1, others=0

hence, the possible DCComb are stored in DCIndex(7..0) by its integer value.

Then, it loops over the possible DCCombs

- For i range 0 to 7 LOOP
 - if DCIndex(i) = 0
 - not valid DCComb, check next DCComb, i = i+1
 - if DCIndex(i) = 1
 - valid DCComb
calculate the corresponding fullSSID with this DCComb
 - » DCComb(2..0) = std_logic_vector(to_unsigned(i,3))
 - » SSIDComb(13..0) = SSID(13..3)+ DCComb(2..0)
 - Read HitCntMem
 - » Read out HitCnt corresponding to this fullSSID from HCM(i)
 - hcm_rdaddr(i) = SSID(13..3)
 - hcm_q(i) = HitCnt
 - Read HitListPointer
 - » Read out baseaddr i.e. addr. for the 1st hit with this fullSSID in the HLM from HLP
 - hlp_rdaddr = SSID(13..0)
 - hlp_q = hlm_baseaddr
 - Read HitListMem
 - » Read out hits from HLM, consecutively starting from the baseaddr. retrieved from HLP.
 - hlm_rdaddr = hlm_baseaddr + tempCNT
 - for tempCNT range 0 to HitCNT-1 LOOP
 - if tempCNT = HitCNT, HitLast=1
last hit with this fullSSID read out
 - if HitLast=1, check next DCComb, i = i+1
 - if i = 7, finished with current AMSSID, SSLast=1

- If SSLast=1, fetch new SSID

5) Output to TF

Hits grouped by their SSIDs are sent out consecutively on each of the 8 layer parallel streams (the initial 3*2 pix layers are combined into 3 output pix layers, since only one in each pair would have output in either case of even or odd SSID for that pix layer).

As stated before, the RoadID, LayerMap, and SectorIDs are also sent out to the TF. The state machine signals, as mentioned in the 2nd section of this note, are temporarily sent out to dummy outputs before the data format/control is finalized for the entire design.

A block diagram of the READ mode is shown in **Slide9**. **Slide17&18** contain a brief description of the control logic as detailed above.

The resource usage summary for HLCtrl_WR is as follows: (shown for pix layer; sct layer similar)

```

+-----+
; Analysis & Synthesis Resource Usage Summary      ;
HLCtrl_RD (pix layer)
+-----+-----+
; Resource                ; Usage ;
+-----+-----+
; Estimate of Logic utilization (ALMs needed) ; 200 ;
; Combinational ALUT usage for logic        ; 80 ;
; -- 7 input functions                ; 2 ;
; -- 6 input functions                ; 16 ;
; -- 5 input functions                ; 20 ;
; -- 4 input functions                ; 11 ;
; -- <=3 input functions              ; 31 ;
; Dedicated logic registers            ; 371 ;
; I/O pins                            ; 196 ;
; Total DSP Blocks                    ; 0 ;
; Maximum fan-out                    ; 371 ;
; Total fan-out                      ; 1636 ;
; Average fan-out                    ; 1.94 ;
+-----+-----+

```

c. State Machine

The Entire DO (including the 2 duplicate copies inside) is controlled by a top_level FSM, which consists of 2 FSMs for each copy of the DO, i.e. DO1fsm and DO2fsm.

DO1fsm and DO2fsm are identical with one small difference. Both have 4 possible states:

- state_write,
 - writing in new hits
 - switch to WRready when hitEE=1 i.e. end of event, last hit in event
- state_WRready,
 - wait for the other DO to finishing reading
 - switch to READ when the other DO is in WRITE mode
- state_read,
 - reading out hits
 - switch to Rdready when last=1 or empty=1 i.e. all hits in that SSID read out
- state_RDready,
 - wait for the other DO to finish writing
 - switch to WRITE when the other DO is in READ mode

the only difference between the two being that DO1 start in state_write; DO2 start in state_RDready. The choice of DO1(2) is arbitrary in the beginning.

Summary of FSM, see **Slide10**.

4. Resource Usage

a. Memory Usage

Most of the DO memory usage are for the on-chip memories. We use megawizard generated RAMS for these memories. The sizes of them are tabulated in **Slide13**.

Note the final memory usage =

$$\begin{aligned} & [(HCMPIX*8*3*2Layers+HCMST*5Layers)+ \\ & (HLPPIX*3*2Layers+HLPST*5Layers)+ \\ & (HLMPIX*3*2Layers+HLMST*5Layers)]*2\text{copies of DO} \\ & + \text{mem. usage for ext. mem. interface} \end{aligned}$$

which brings to 5.6Mb in total, and 7Mb when implemented on-chip.

b. Logic Usage

Most of the DO logic usage are for the HLCtrl_WR and HLCtrl_RD functionalities. Note the usage for HLCtrl_WR is almost an order of magnitude larger than HLCtrl_RD: this is mostly due to the 2¹¹-bit long DC_Reset word we are using to flag whether or not that specific HCM address has been cleared.

A tabulated summary of the logic usage is shown on **Slide14**.

c. Total Resource Usage

See attached file.

5. Timing and Other Issues

(Work in progress.)

Right now the HLCtrl_WR logic can handle consecutive incoming hits at 200MHz clock rate; the HLCtrl_RD logic can handle consecutive incoming SSIDs at 200MHz clock rate (with a few relatively minor bugs to be worked out); the external memory interface, according to the Quartus Megawizard generated testbench, can read from the external memory at random at 200MHz clock rate. The full testbench simulation is still in progress.

The full DO, when compiled alone on-chip, operates at around 140MHz.

A few things to note:

- a. data format w.r.t. input from transceiver and output to DO
- b. enable/state machine/error flag signal propagation through the system
- c. synchronization within internal logic: making sure information to/from the on-chip memories are on the right clock edges
- d. multi-layer synchronization
- e. large logic usage may be problematic
- f. warnings/timing issues w.r.t. ext. mem. interface
- g. warnings/tim g issues w.r.t. the whole system