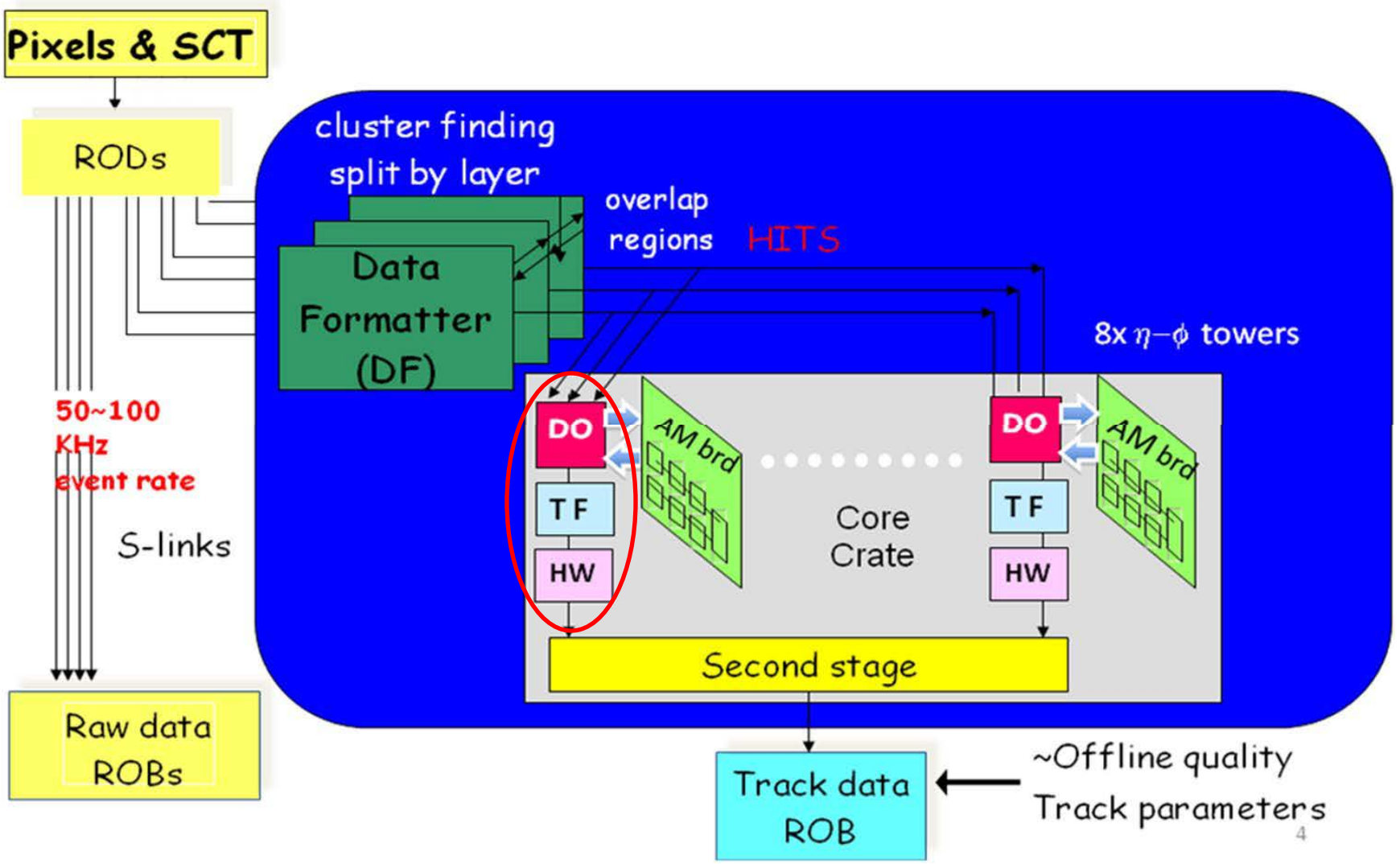
Data Organizer

Yangyang Cheng

03/21/2013

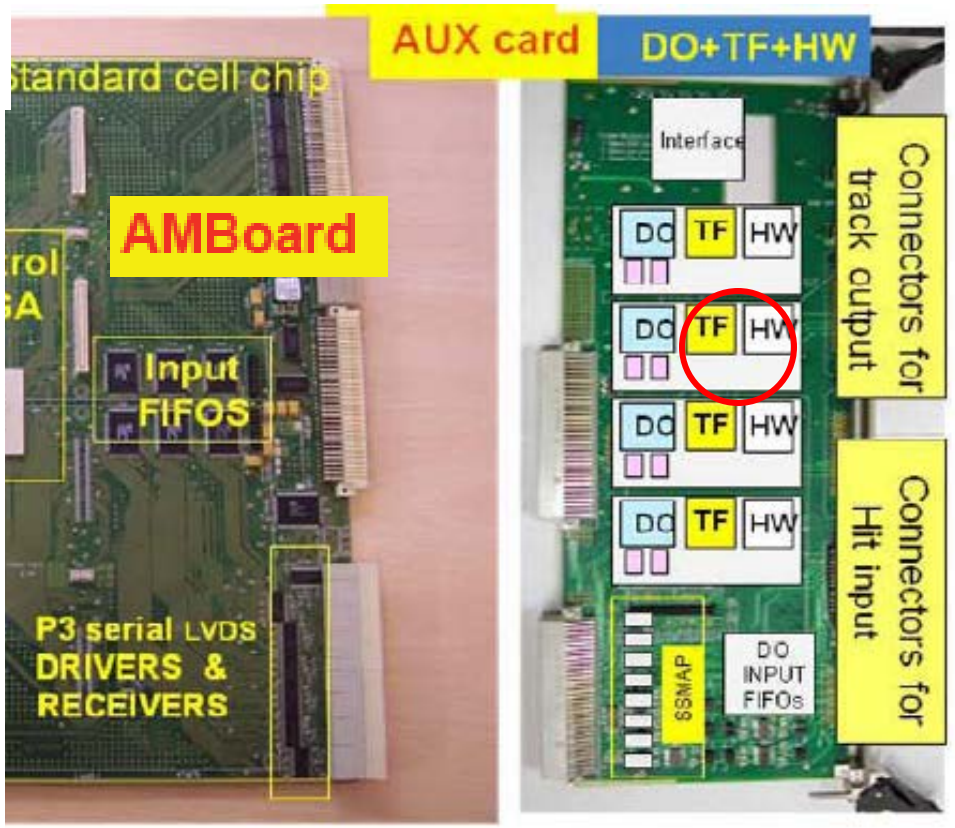
Hardware System Overview



AUX card & Data Organizer

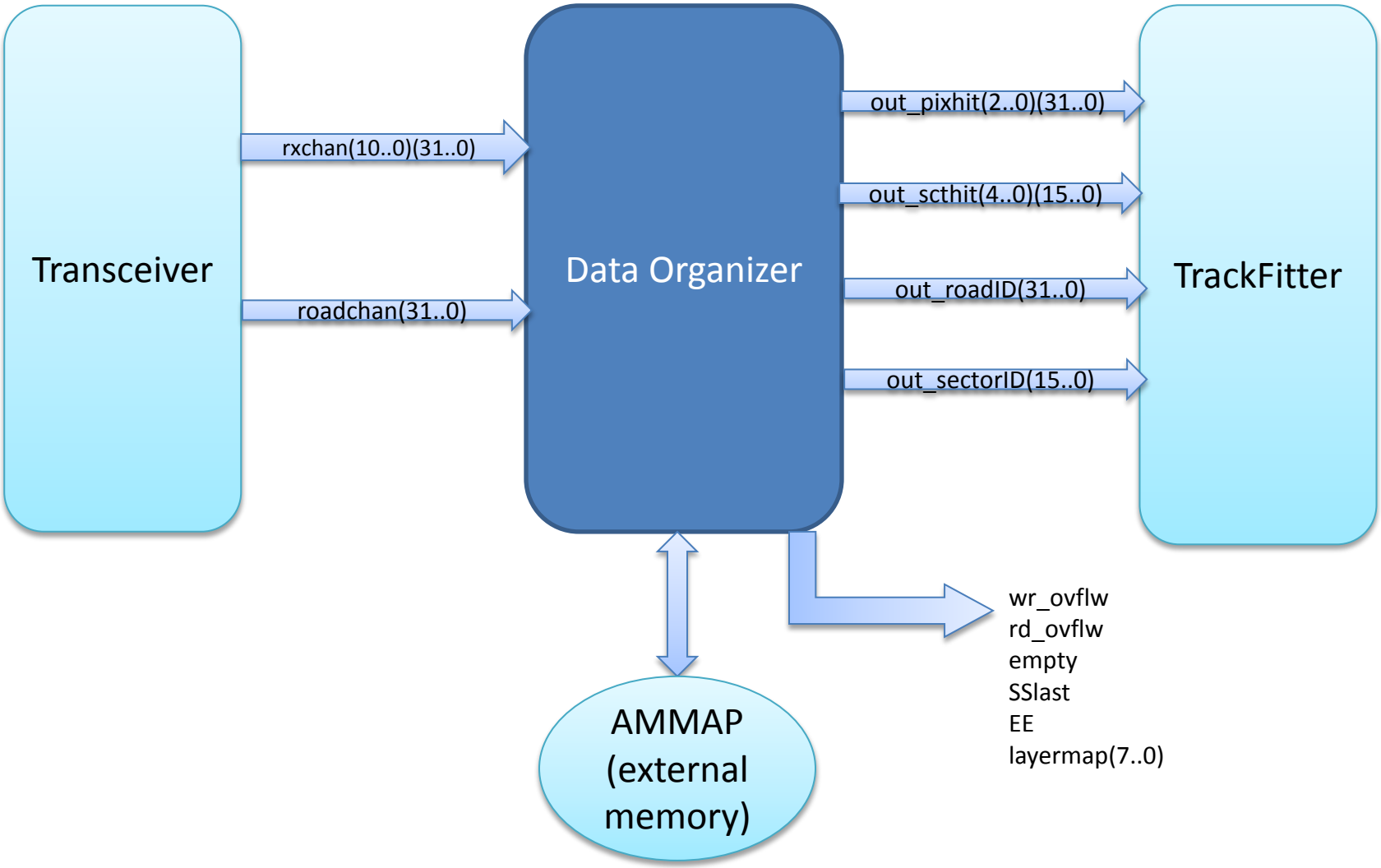
AUX Responsibilities

- Receive hits from Data Formatter
- Organize hits into Super Strips (SS)
- Send SS to AM Board
- Receive IDs for matched roads from AM board
- Reject bad roads
- Send hits from promising roads to Second Stage board

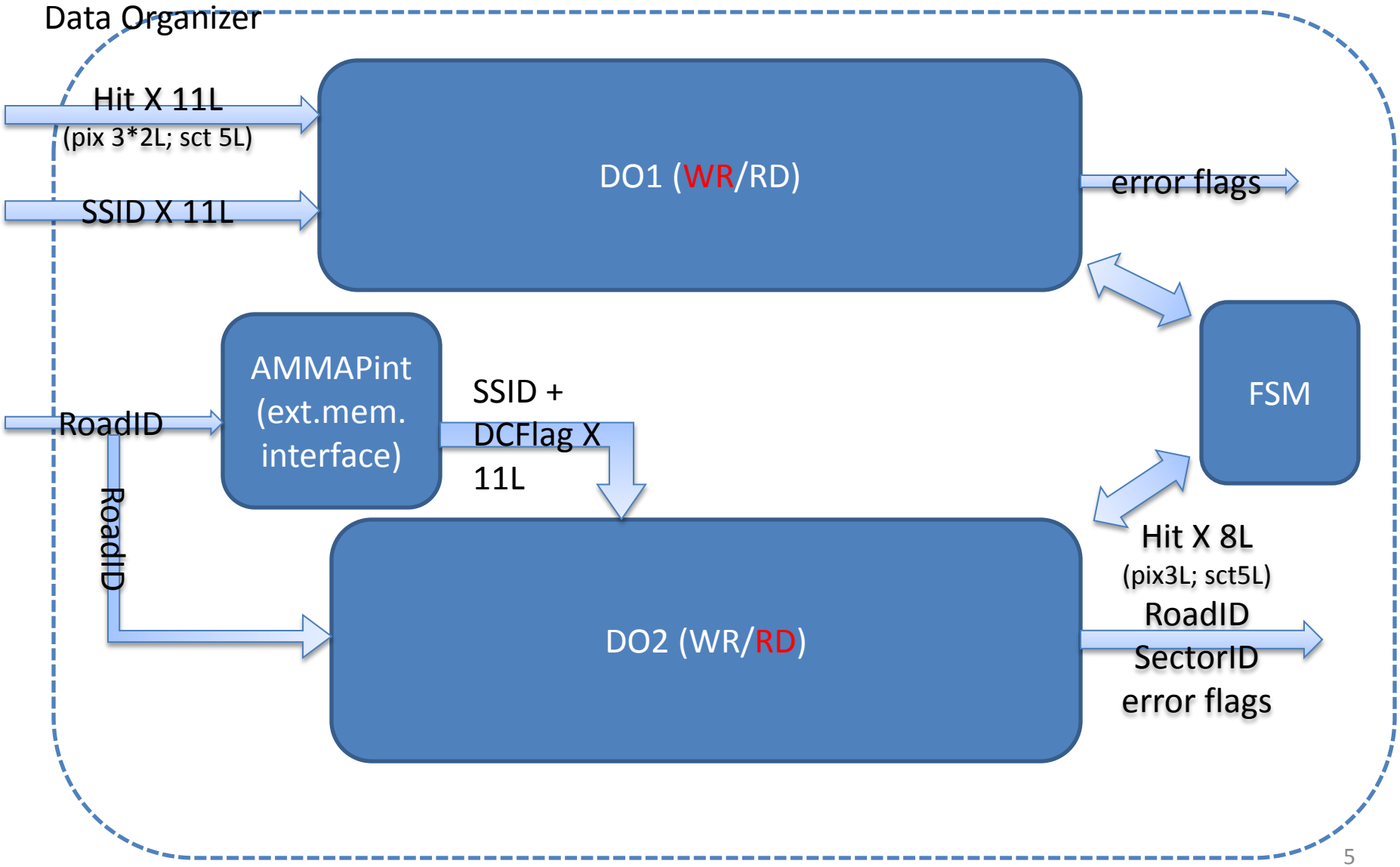


Data Organizer: A smart database built on the fly so that Hits can be immediately stored/retrieved according to its SSID.

Input/Output



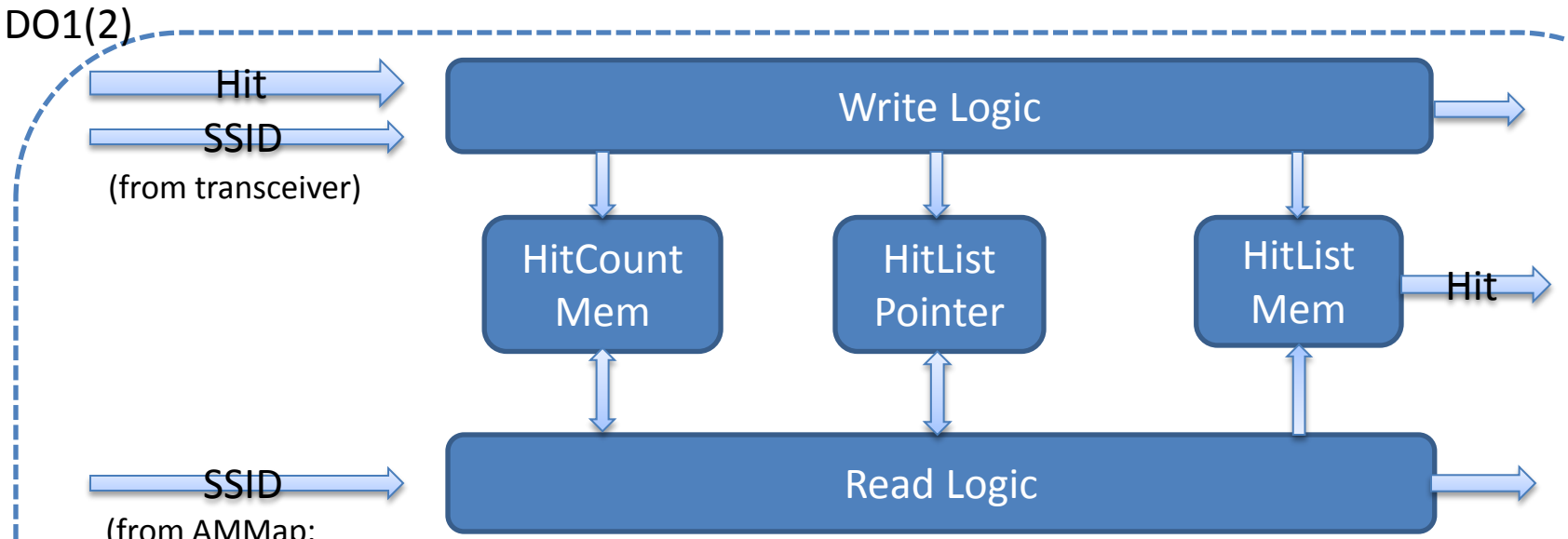
Inside the DO Block



WRITE/READ mode

- Write Mode:
  - Stores incoming hits according to its SSID
- Read Mode:
  - Sends out pre-stored hits corresponding to incoming RoadID; RoadID is translated into SSID by AMMap; output hits grouped by SSID
- DC (Don't Care)
  - up to 3 LSB of SSID is ignored in comparison/matching
- Hit information w.r.t. corresponding SSID is stored/retrieved in 3 separate parts
  - **HCM**(HitCountMem): stores HitCnt (the number of hits with specific SSID)
  - **HLP**(HitListPointer): stores the address for the 1st hit with specific SSID (SSID used as HLP address) in the HLM
  - **HLM**(HitListMem): stores hits sequentially; hits with same SSID stored consecutively (hits with same SSID will be coming consecutively on input stream)

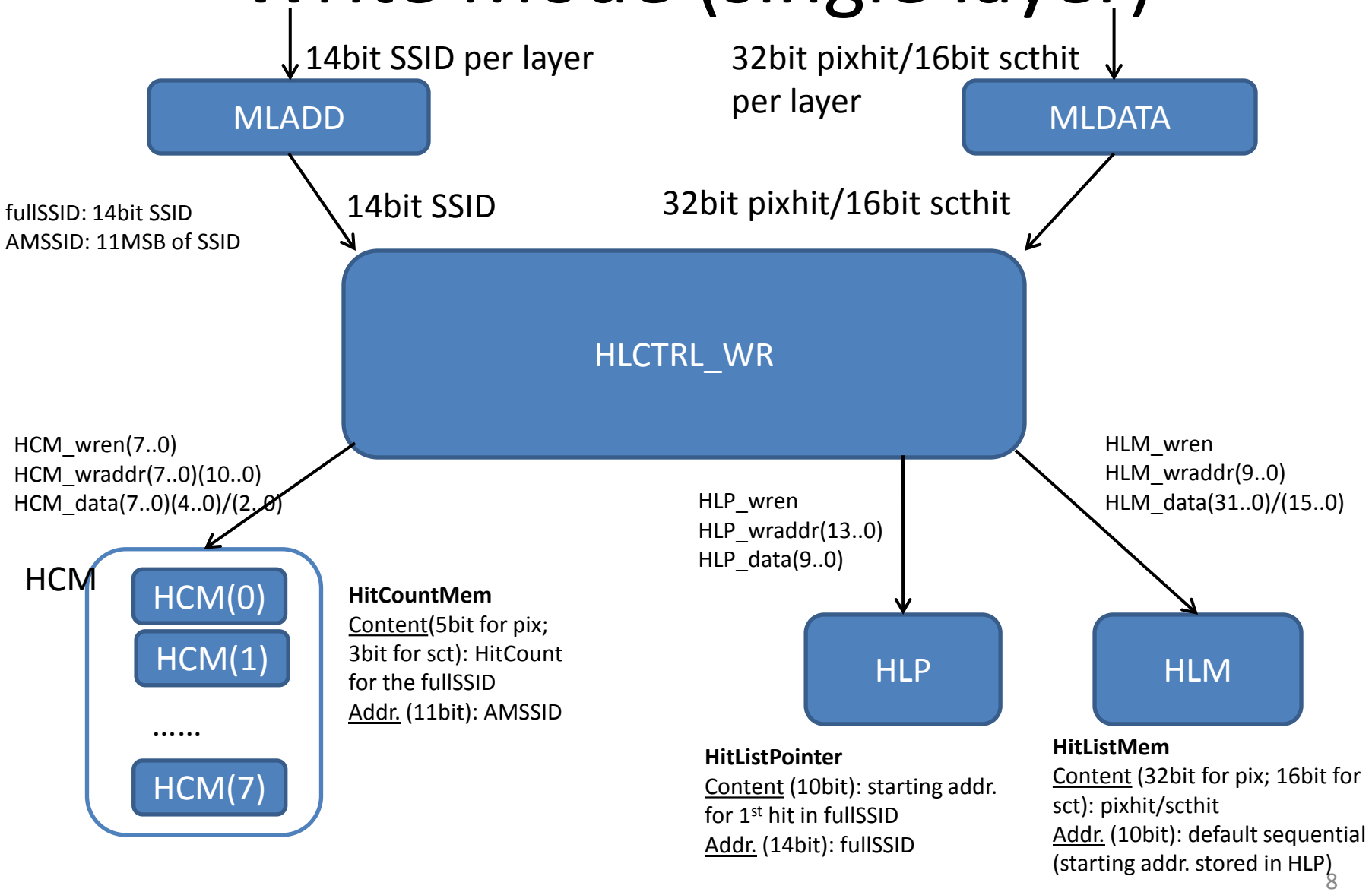
Parallel Structure



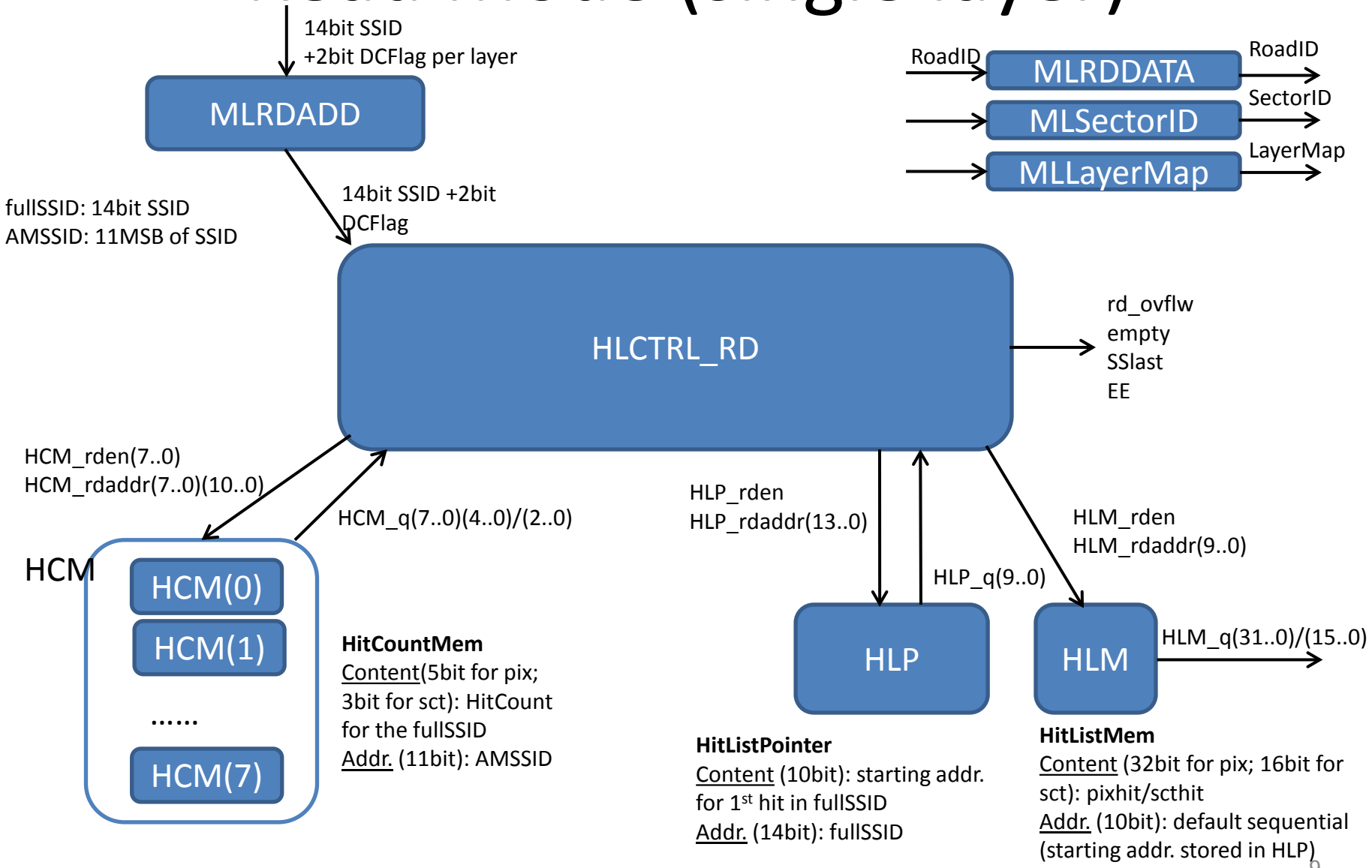
The above structure duplicated for 11 parallel layers

- 3*2 pixel layers (3 layers in final output; doubled w.r.t. even/odd SSID)
- 5 SCT layers

Write Mode (single layer)



Read Mode (single layer)



Finite State Machine

- type FSMSTATES is (
  - 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

  );

- DO1 start in state_write (DO1fsm); DO2 start in state_Rdready (DO2fsm)

AMMapInt (ext. mem. interface)

- using RLDRAMII: MT49H16M36-25
- interface generated using QuartusII v12.1 for Arria V
  - studied through auto-gen. testbench/example
  - using the same pin assignments as example
- AMMap stored on ext. mem.
  - translate input RoadID into SSID+DCFlag per layer
- Multiple compilation warnings/timing issues
  - detailed in separate doc.

VHDL Code Index

- Dotop
  - DO_deflib (defines the signal types used)
  - fsmtop
    - DO1fsm; DO2fsm
  - AMMapInt
  - DO11L
    - MLPIX(SCT)DATA, MLADD
    - HLCWritePIX(SCT)
    - MLRDADD, MLRDDATA, MLSectorID, MLLayerMap
    - HLCReadPIX(SCT)
    - HCM, HLP, HLM (PIX/SCT)

Resource Usage: Memory

RAM name	Width	Depth	Size (M10K)
HLM_pix	32	10	4
HLM_sct	16	10	2
HLP_pix	10	14	20
HLP_sct	10	14	20
HCM_pix (*8)	5	11	1*8
HCM_sct (*8)	3	11	1*8

Total: 5.6Mb/25Mb needed; 7Mb/25Mb implemented
Usage expected/needed/implemented detailed in separate doc.

Resource Usage: Logic

	LC Comb.	LC Registers	ALMs needed	ALMs placed
HLCRD_pix	81	370	135.11	205.00
HLCRD_sct	71	347	116.73	190.17
HLCWR_pix	2898	2382	1857.50	1937.83
HLCWR_sct	2898	2332	1840.00	1890.33
AMMap_int	1027	1534	757.05	963.75
FSM	141	132	53.92	80.50
DO1(DO2)	33022	37003	23749.53	26853.27
DOtop	67214	77033	49034	55366

Total: 49,034/190,240 needed; 55366/190,240 implemented
Usage expected/needed/implemented detailed in separate doc.

HLCtrl_WR

- SSID(13..0)-> AMSSID(10..0)+DCBits(2..0)
  - DCBits(2..0) -> int DCIndex range 0 to 7
  - AMSSID(10..0) -> int AMSS_index range 0 to 2^11
- fullSSID
  - identify new SSID
  - count # of hits in old SSID : HitCnt
  - obtain starting addr. for each SSID by consecutive incrementation
- 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.

HLCtrl_WR (cont'd)

- AMSSID
  - incoming AMSSID(10..0) compared with previous AMSSID, simultaneously as fullSSID(13..0) is being processed (as described in previous slide)
    - Same
      - Write current HitCnt to HCM(DC_Index)
    - Different
      - Check DC_Reset(AMSS_Index)
        - » DC_Reset(AMSS_Index)=0
          - Reset all 8 HCM at addr=AMSSID
          - Write HitCnt (=1) to HCM(DC_Index)
        - » DC_Reset(AMSS_index)=1
          - Write HitCnt (=1) to HCM(DC_Index)

HLCtrl_RD

- DCFlag, 2 bit word
  - 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

- 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
    - **the possible DCComb are stored in DCIndex(7..0)**

HLCtrl_RD (cont'd)

- 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
      - DCComb(2..0) = std_logic_vector(to_unsigned(i,3))
      - SSIDComb(13..0) = SSID(13..3)+ DCComb(2..0)
    - Read HitCntMem
      - Read out HitCnt from HCM(i)
        - » hcm_rdaddr(i) = SSID(13..3)
        - » hcm_q(i) = HitCnt
    - Read HitListPointer
      - Read out baseaddr from HLP
        - » hlp_rdaddr = SSID(13..0)
        - » hlp_q = hlm_baseaddr
    - Read HitListMem
      - Read out hits from HLM
        - » hlm_rdaddr = hlm_baseaddr + tempCNT
        - » for tempCNT range 0 to HitCNT-1 **LOOP**
          - if tempCNT = HitCNT, HitLast=1
    - if HitLast=1, check next DCComb, i = i+1
  - if i = 7, finished with current AMSSID, SSLast=1
- If SSLast=1, fetch new SSID