



# Progress on software implementation of RD53B data decoding

Maurice Garcia-Sciveres, Timon Heim, Hongtao Yang

Lawrence Berkeley National Lab and University of California

Weekly Instrumentation Meeting

June 12, 2020

- RD53B data decoding task is now at a critical stage as the chip will arrive soon
- Today focus on updates since our previous discussions ([April 24](#), [May 15](#))
  - Validation of decoding algorithm using bit stream from chip simulations
  - More analyses on bit stream content & decoding performance
  - Integration of decoding in YARR

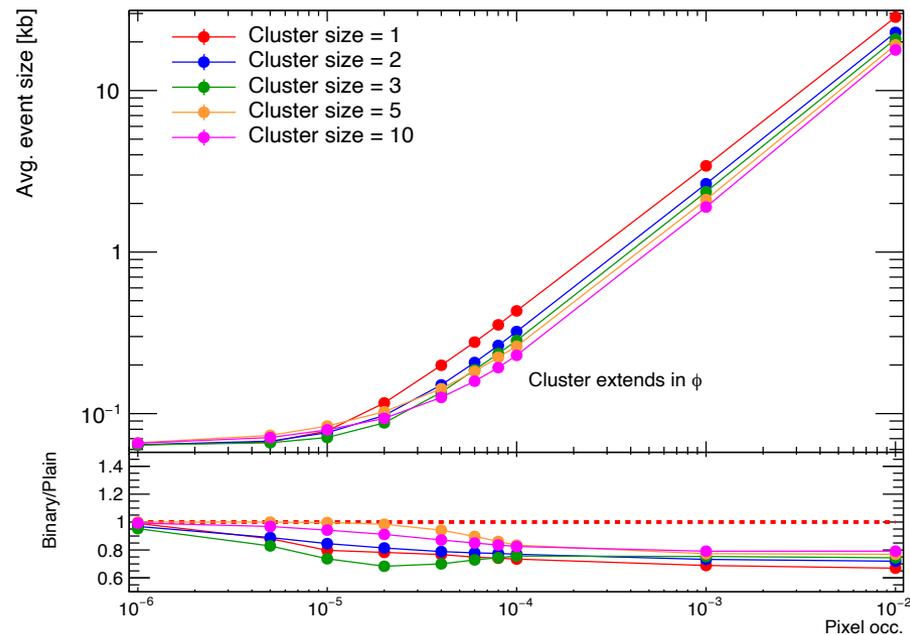
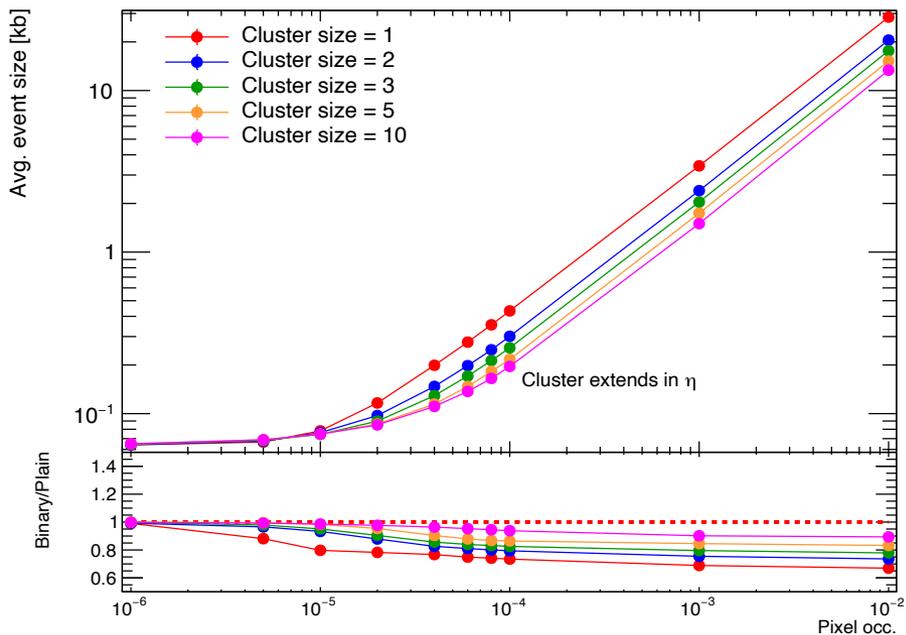
A quarter row

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17

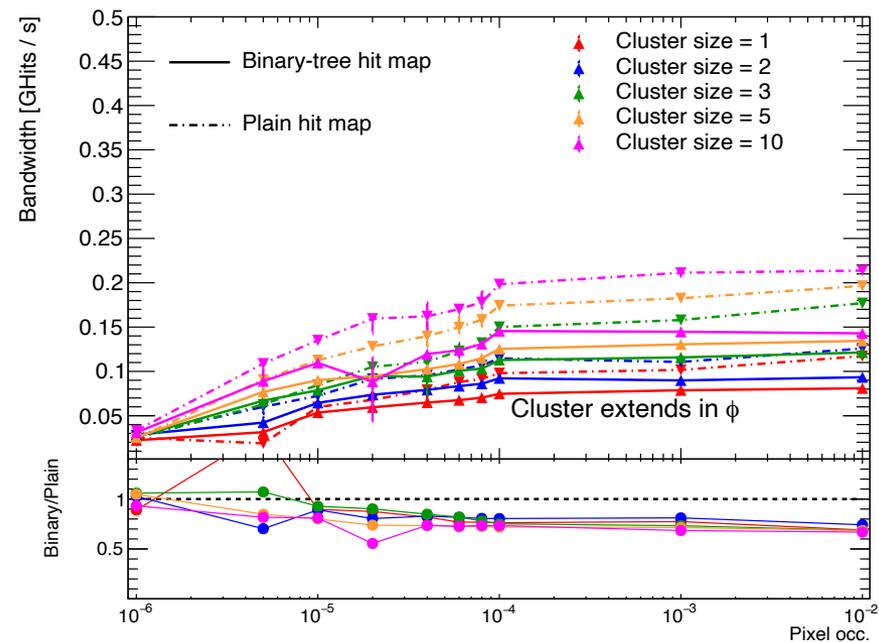
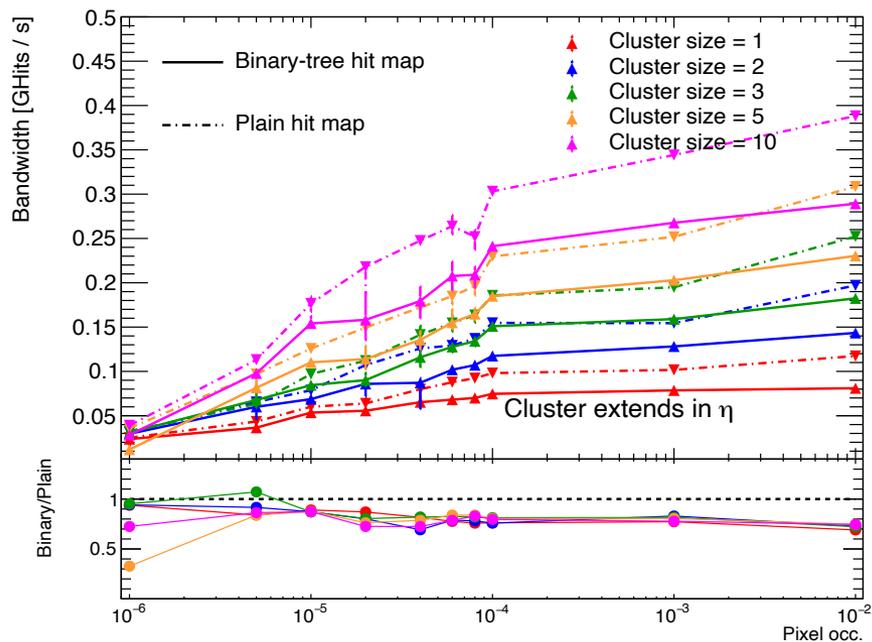
Binary-tree: 11.10.0(1).11.11.0(1)10.0(1)10  
Row 0 Row 1

Plain: 00011000.00001100  
Row 1 Row 0

- Decoding algorithm now validated with bit streams from chip simulation
  - Valuations done for both compressed and uncompressed hit maps
- Core column (ccol) index starting from 1. Quarter row (qrow) index from 0
  - Column index = (ccol - 1) × 8. Row index = qrow × 2
- Uncompressed 16-bit hit map: **large indices come first**, i.e. NS...(17) (16)...(10)(07)(06)...(00)
- Compressed binary tree hit map: first split into two rows, then split into columns. **Small indices come first**
- ToT: **small indices come first**, i.e. NS...(00)(01)...(07)(10)(11)...(17)



- Now extend the lower bound to  $10^{-6}$ , and interpolated more points in  $[10^{-5}, 10^{-4}]$ 
  - Average event size flat at low occupancy, then scales linearly at high occupancy
- Using binary-tree hit map results in smaller event size compared with uncompressed 16-bit hit map
- Event size reduces as cluster size increases. Small difference between growing cluster in eta or phi



- Use number of hits instead of number of bits to quantify bandwidth (no event making, which is current bottleneck)
  - Using uncompressed 16-bit hit map still leads to better bandwidth
    - Compressed hit map is roughly 2 times slower to decode compared with uncompressed using current algorithm
  - More clear difference between eta and phi directions

- Process 100k events with cluster size = 10, occ =  $10^{-5}$ , which takes largest reduction in bandwidth from event making
- Compare the following scenarios
  1. Std YARR event making
  2. Change std::list to std::vector
  - 3. Reserve vector space in advance**
  4. Switch to simple c++ array
  5. No event making

```
#elif MAKEEVENT == 2
  uint16_t m_curOut[100000][2000][3];
  uint64_t m_nEvent, m_nHit[100000];
#endif
```

scenarios	1	2	3	4	5
Processing time [ms]	16.6	13.3	2.7	1.5	0.5
Improvement wrt 1	—	20%	84%	91%	97%

- Integrate decoding as part of RD53B data processor in YARR ([merge request](#))
- Unit testing of decoding implemented using chip simulation bit stream
  - Require reproducing each hit (in total 35137)
- Also prepared a helper tool to visualize decoding

```

10000010100110110110011011001010000100111100010101101101101001011
1 00000101 001101 1 0 10011011 0 0 10 10 0001 001111 0 0 01010110 11 10 10 0 10 10 11
NS Tag ccol islast isnext qrow Hit map ToT0 ccol islast isnext qrow Hit map
13 155 0(1) 0(1) 10 10 00000000000001000 1 15 86 11 10 10 0(1) 10 10 11 0100000011000000

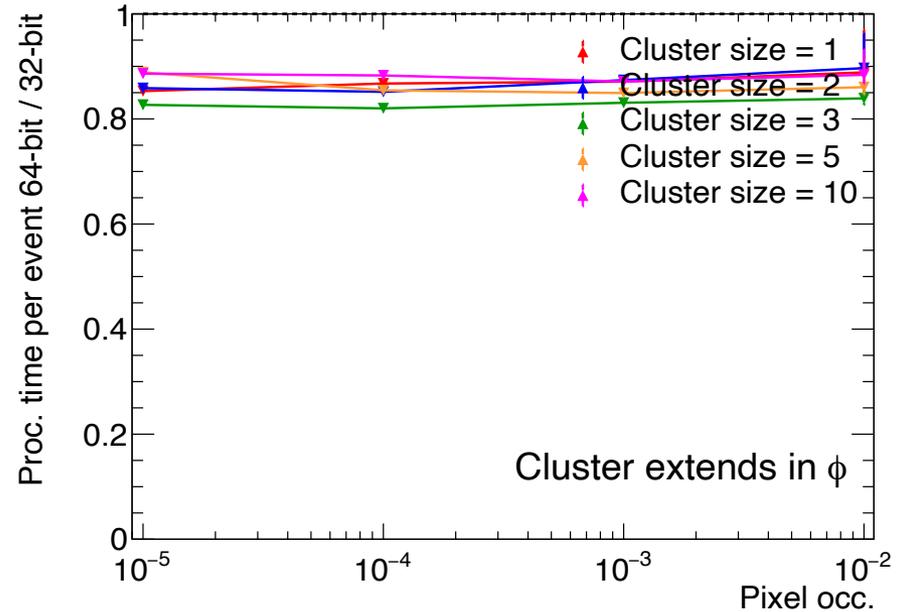
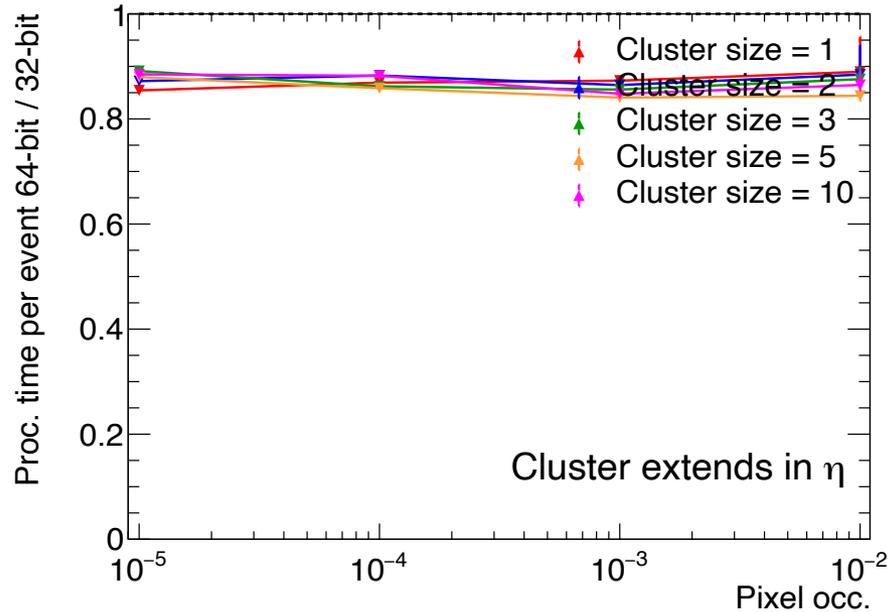
000100010001011101010000100010100000011111010010100111110100100
0 0010 0010 0010 1 1 10 10 10 0 0010 001010 0 0 00011111 0 10 0 10 1001 1 1 11 10 10 0 10 0
NS ToT0 ToT1 ToT2 islast isnext Hit map ToT0 ccol islast isnext qrow Hit map ToT0 islast isnext Remain
2 2 2 10 10 10 0(1) 0100000000000000 2 10 31 0(1) 10 0(1) 10 0000000000100000 9 11 10 10 0(1) 10

0101001100101000000011010111110101011010010110011001100110011
0 (0) 10 1001 1001 010000 0 0 01101011 11 11 0 10 10 10 11 0 10 0 10 1100 1100 1100 1100 1 1
NS Hit map ToT0 ToT1 ccol islast isnext qrow Hit map ToT0 ToT1 ToT2 ToT3 islast isnext
0(1) 10 0100000000100000 9 9 16 107 11 11 0(1) 10 10 10 11 0(1) 10 0(1) 10 0010100000011000 12 12 12 12

01001010110000100100011011000100100010011111000110100100010001000
0 10 0 10 10 1100 001001 0 0 01101100 0 10 0 10 0010 0 1 11 10 0 0 11 0 10 0 10 0010 0010 00
NS Hit map ToT0 ccol islast isnext qrow Hit map ToT0 islast isnext Hit map ToT0 ToT1 Remain
10 0(1) 10 10 0000100000000000 12 9 108 0(1) 10 0(1) 10 0000000000100000 2 11 10 0(1) 0(1) 11 0(1) 10 0(1) 10 0001000000011000 2 2
    
```

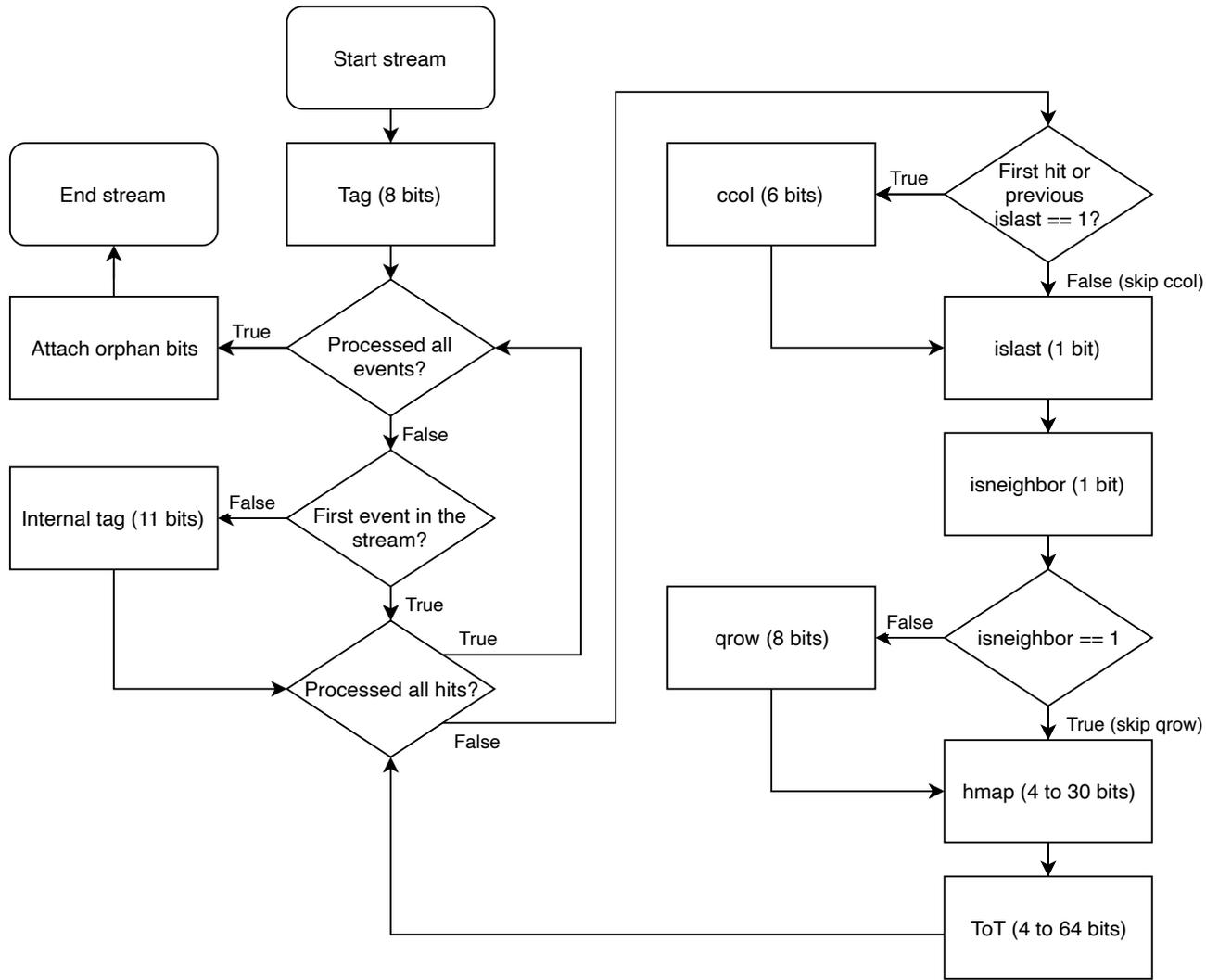
- Implement software emulator for RD53B
- Get software ready for RD53B chip testing
- Prepare more detailed analyses of bit stream, e.g. fraction of bits used for addressing, ToT, hit map etc.

# Backup



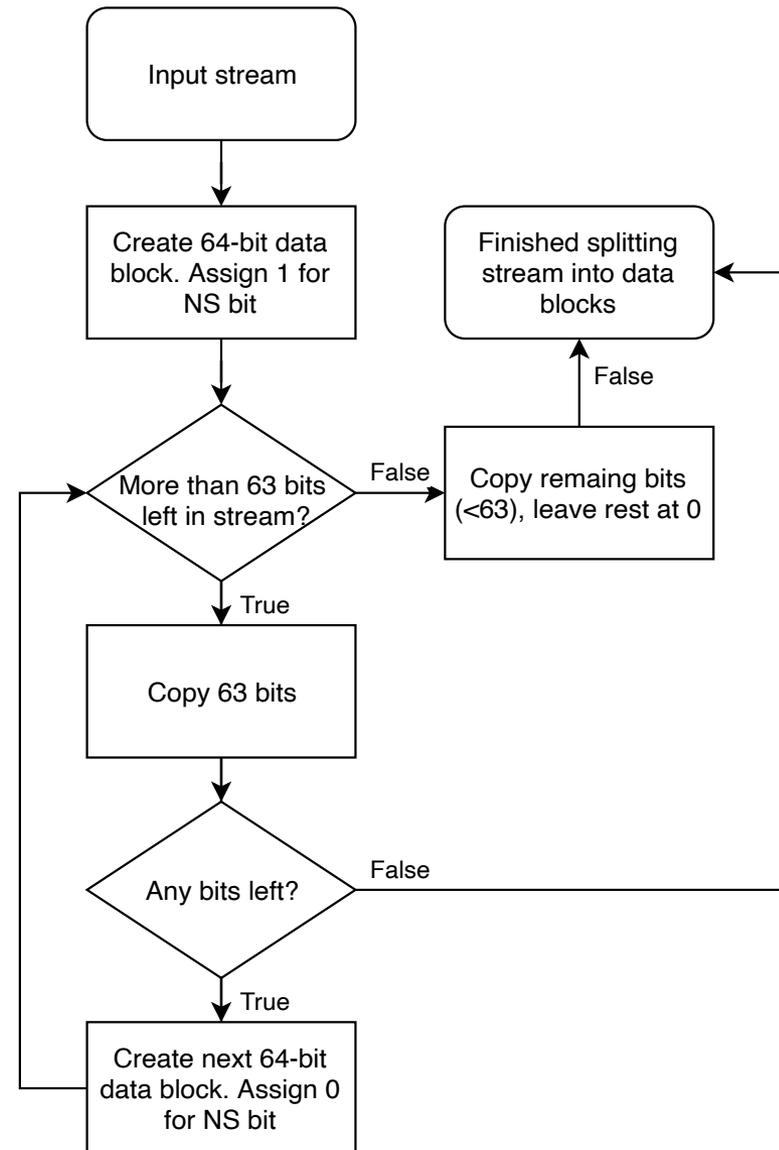
- 1 block of RD53B data stream consists of 64 bits
- YARR use 32-bit words to save the stream: need to break down each block into two words
- Use 64-bit word (thus one word for one block) leads to 10~20% improvement of processing time

# Encoding flowchart: building a stream

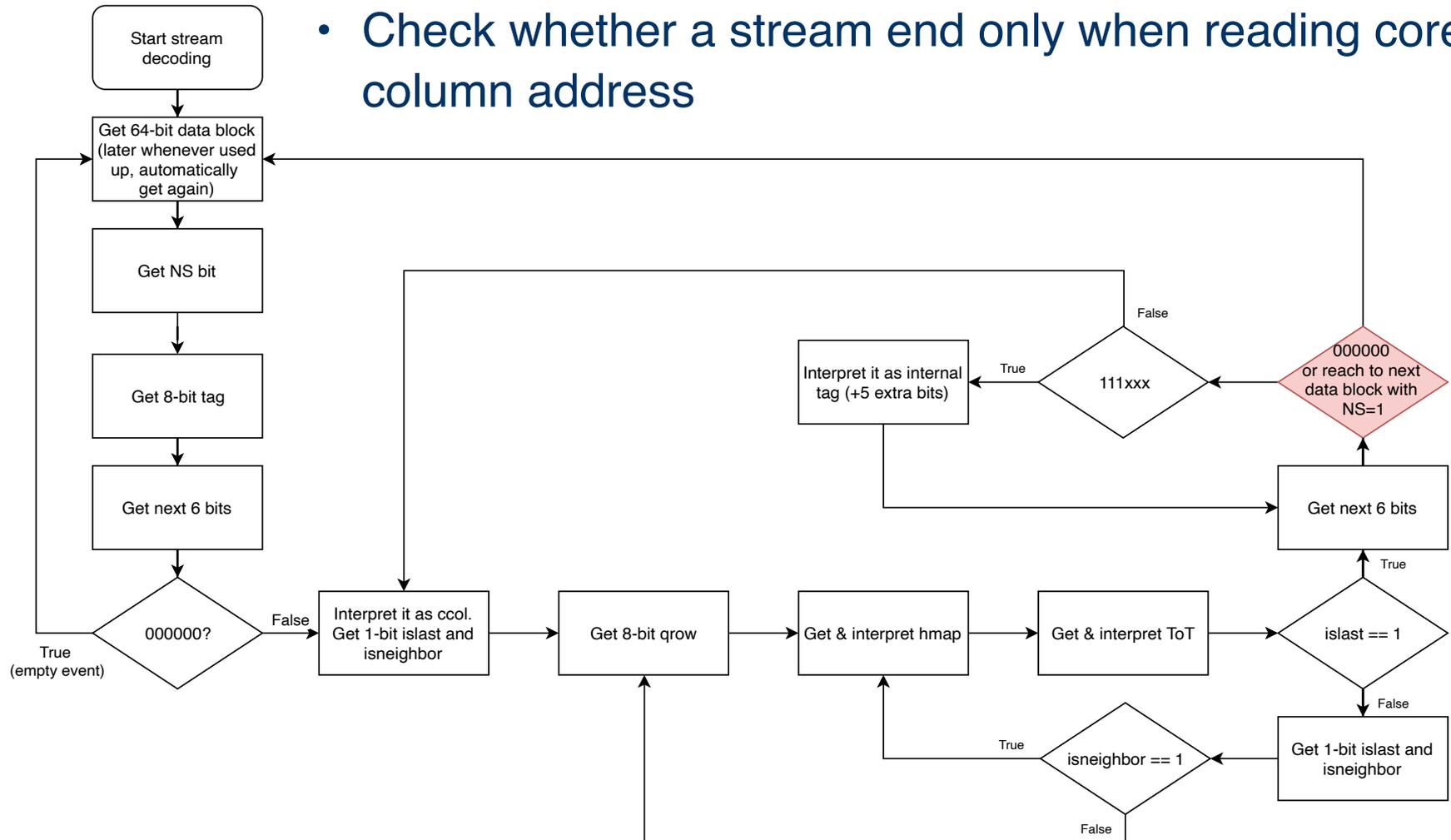


- Development largely based on Noemi's code. Neglecting chip ID

- The first bit of a 64-bit block will always be New Stream (NS) bit
  - The first block of the stream has NS=1
  - The trailing blocks, if any, have NS=0
  - Chip ID, if exist, will follow NS bit



- Check whether a stream end only when reading core column address



- Decoding algorithm validated with bit stream from RD53B chip simulation. Thanks to Attiq and Sara for the help!