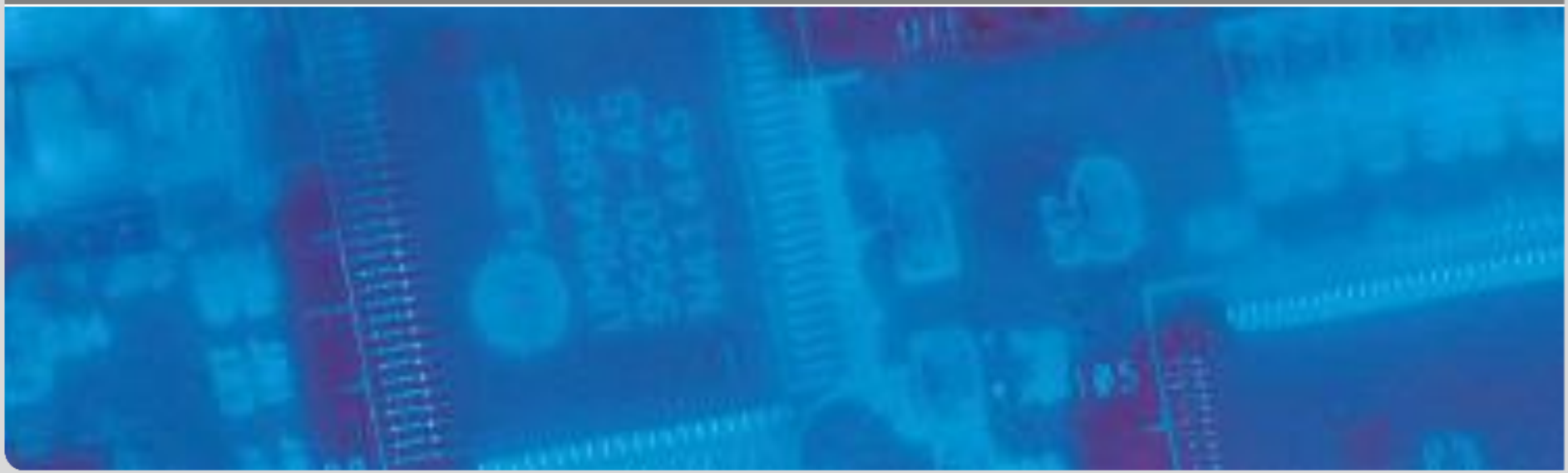
Karlsruhe Institute of Technology

Institut für Technik der Informationsverarbeitung

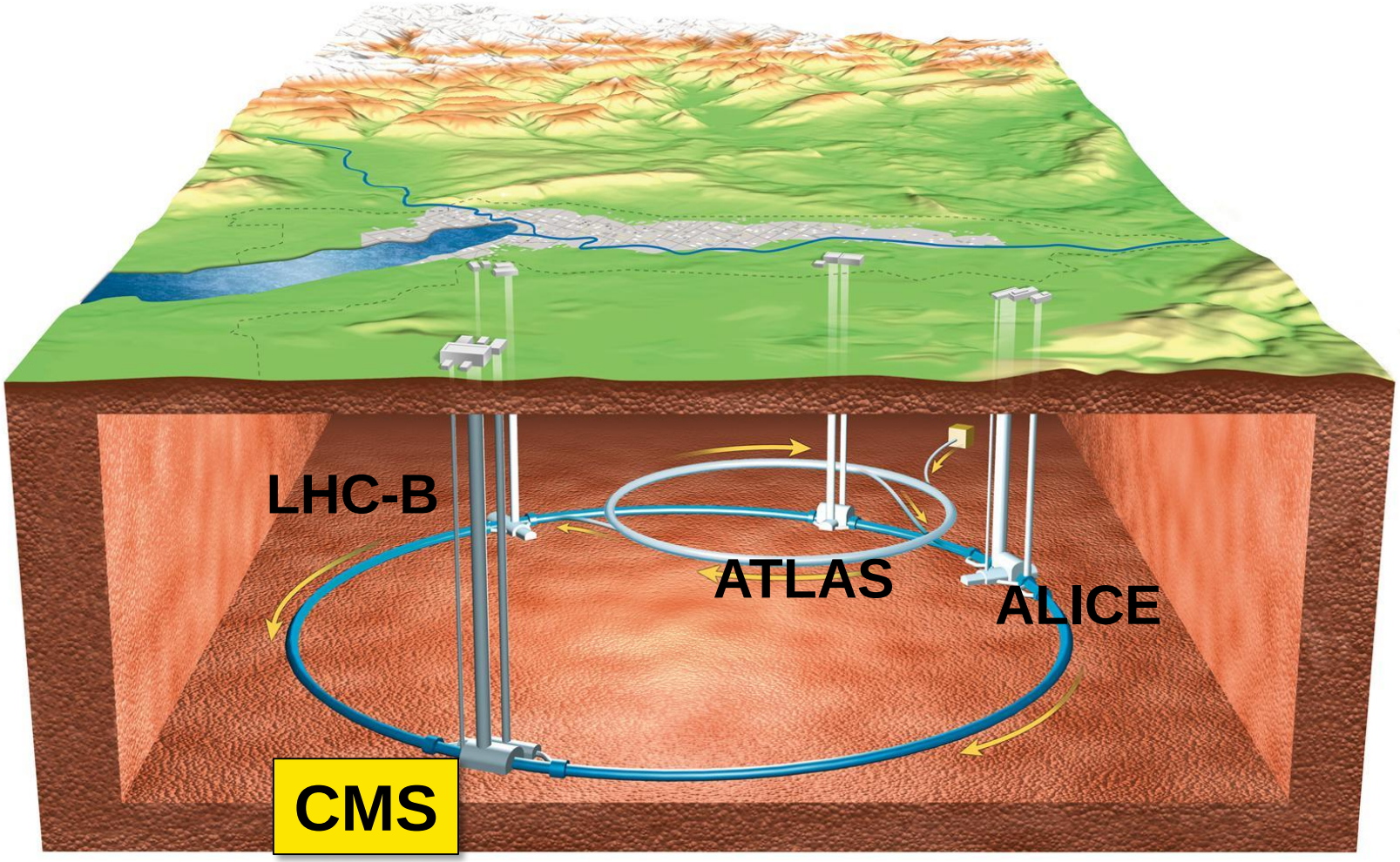
A Content Adapted FPGA Memory Architecture with Pattern Recognition Capability for L1 Track Triggering in the LHC Environment

Tanja Harbaum, Mahmoud Seboui, Matthias Balzer, Jürgen Becker and Marc Weber

Institut für Technik der Informationsverarbeitung (ITIV)

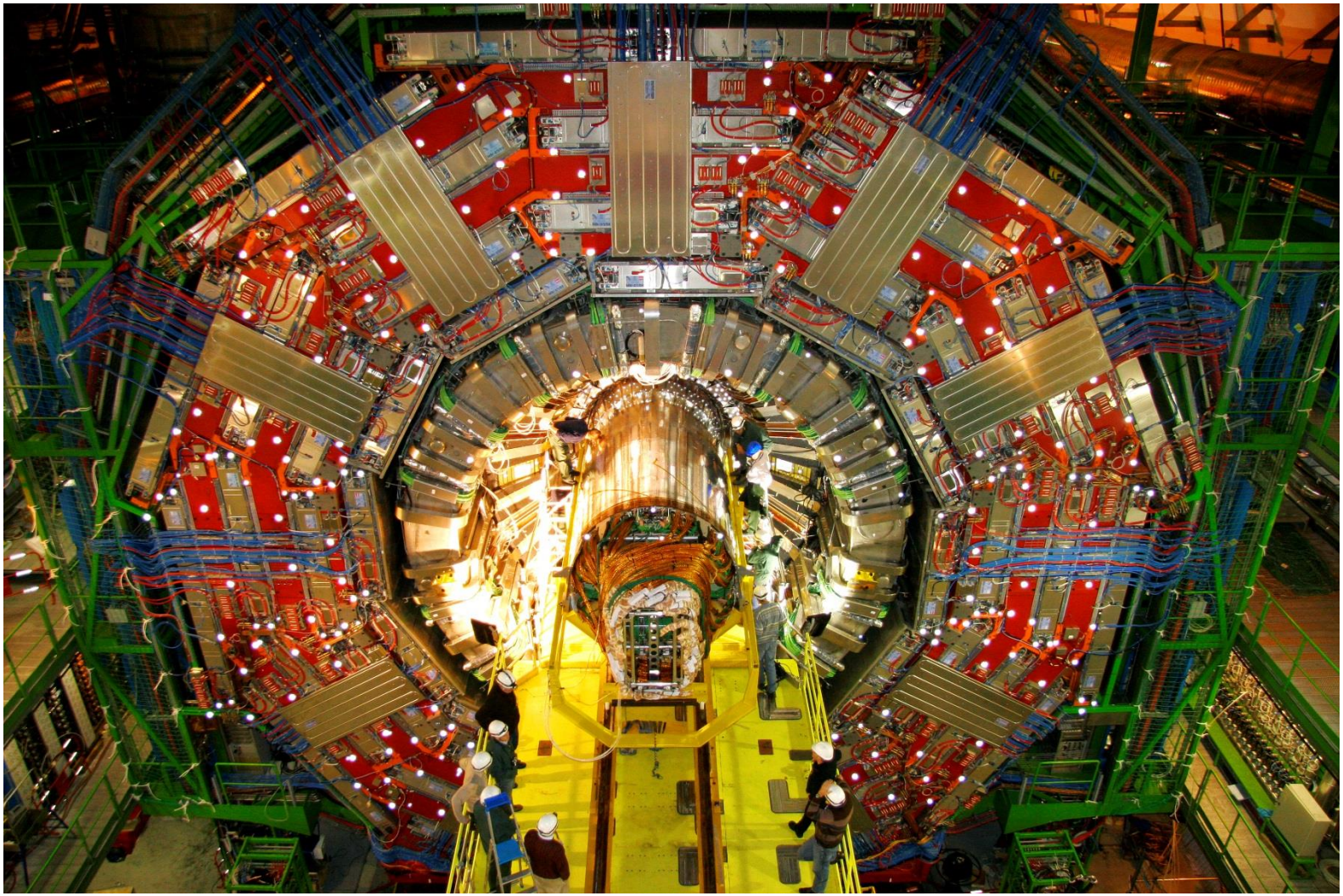


Large Hadron Collider - experiments



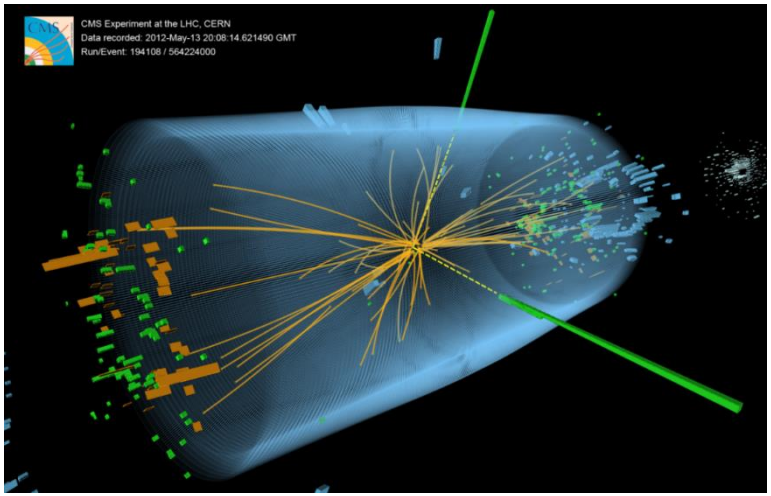
Compact Muon Solenoid

CMS detector

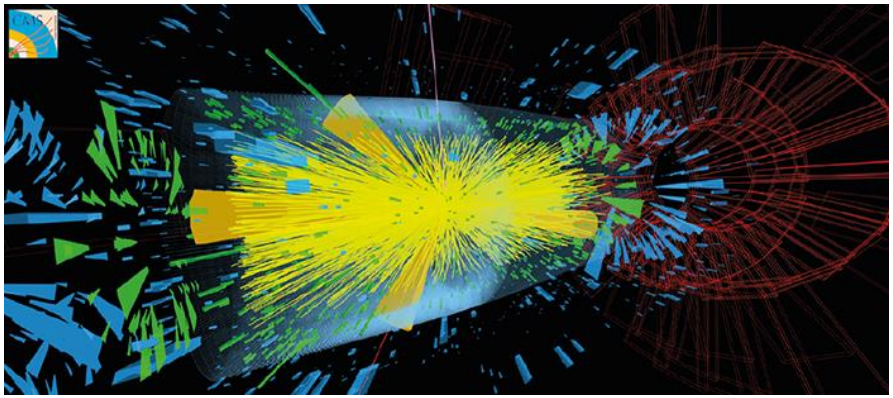


Motivation: LHC Upgrade Phase 2

LHC today

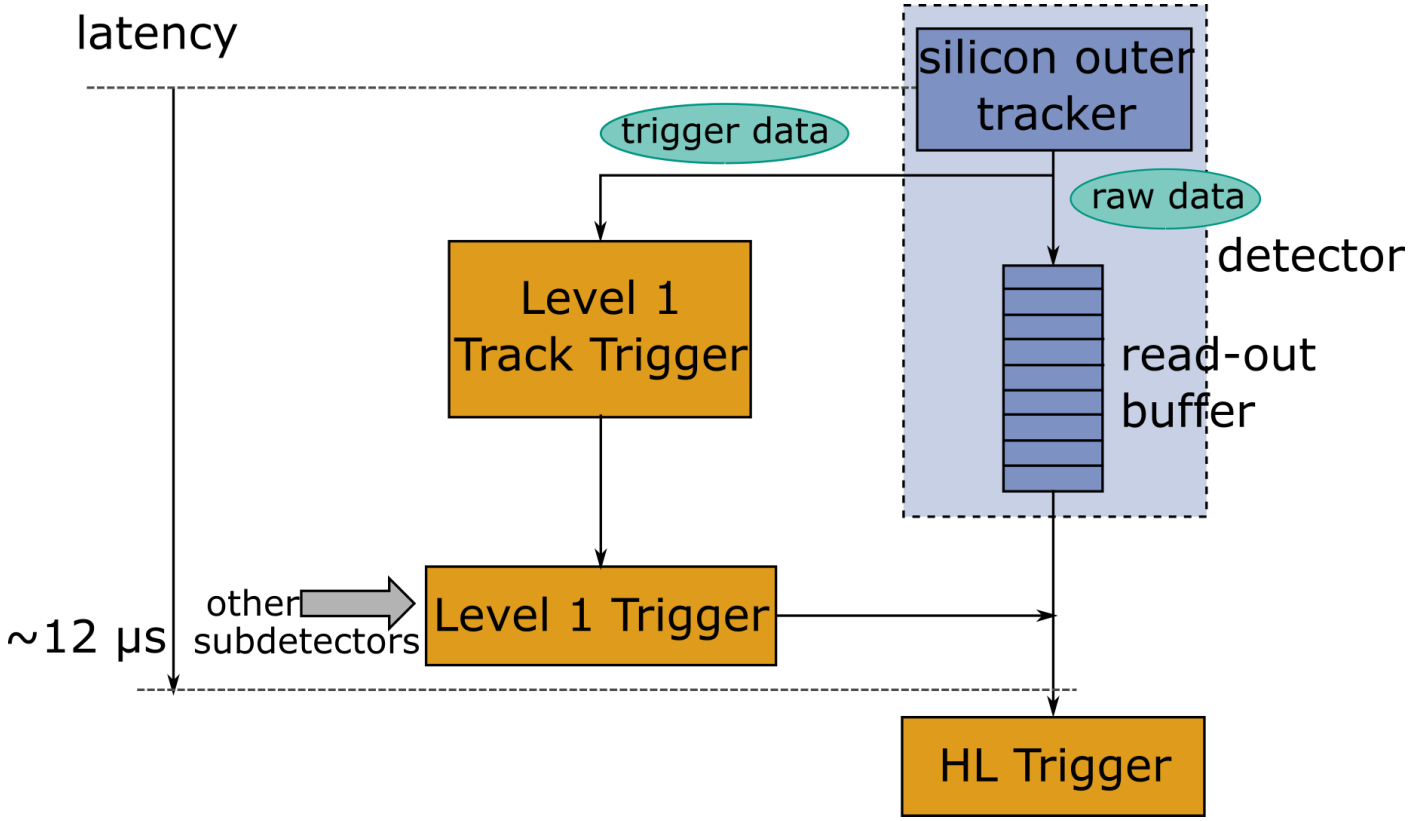


HL-HLC ~ 2025



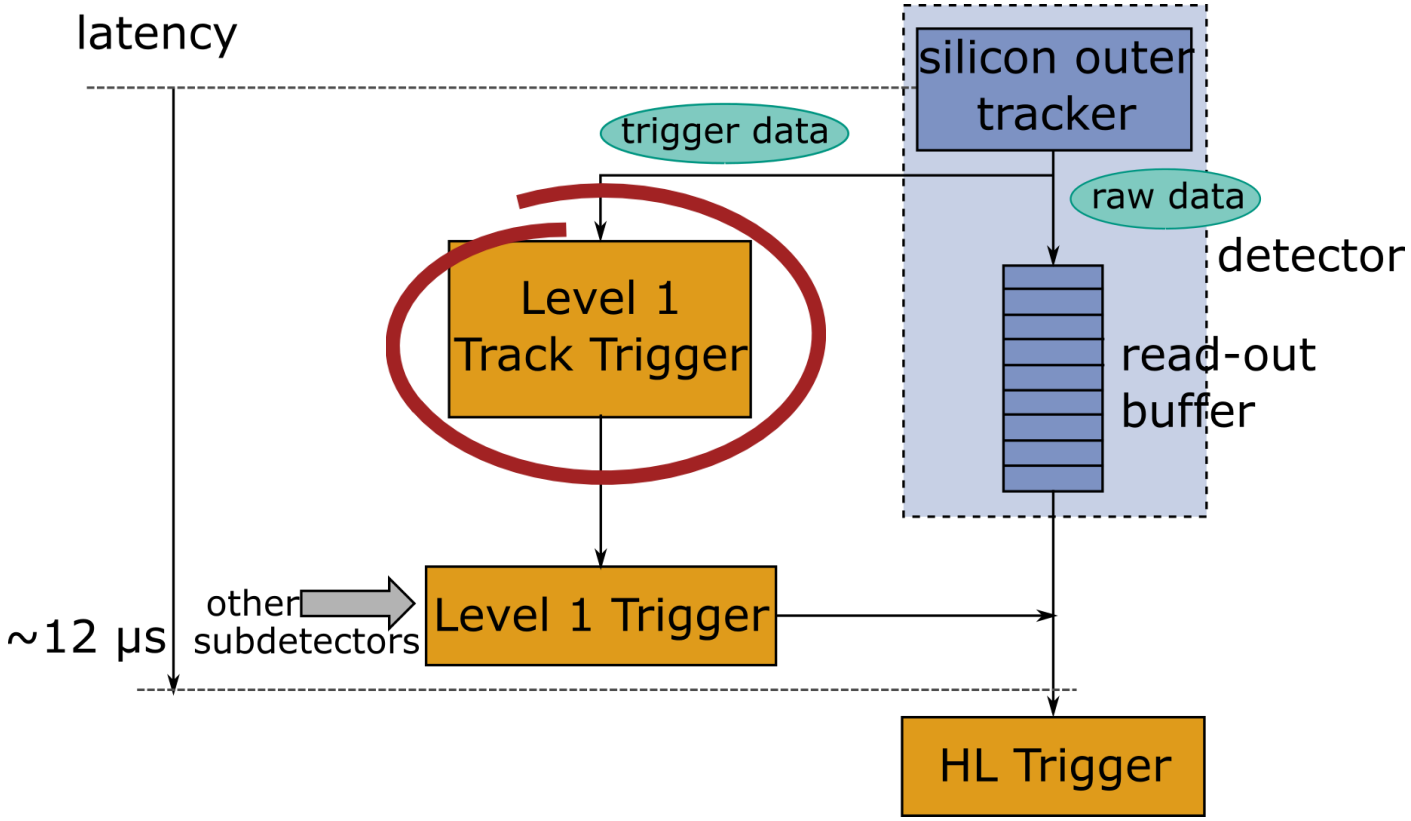
- increasing luminosity by a factor of 10
- 10000 tracks per bunch crossing
- input data rate ~ 100 Tb/s

CMS trigger system



- Reduce data by factor 400 (online)
- Trigger decision latency ~ 12 µs

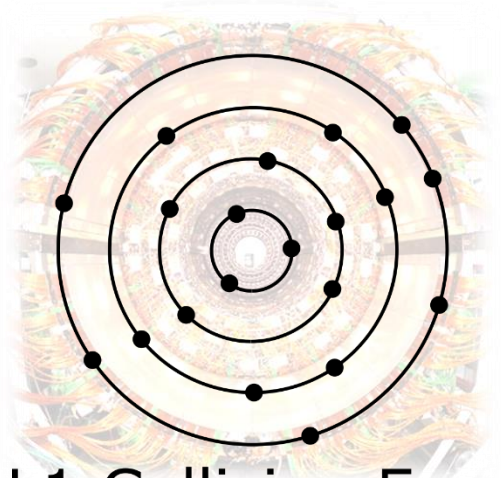
CMS trigger system



- reduce data by factor 400 (online)
- trigger decision latency ~ 12 µs

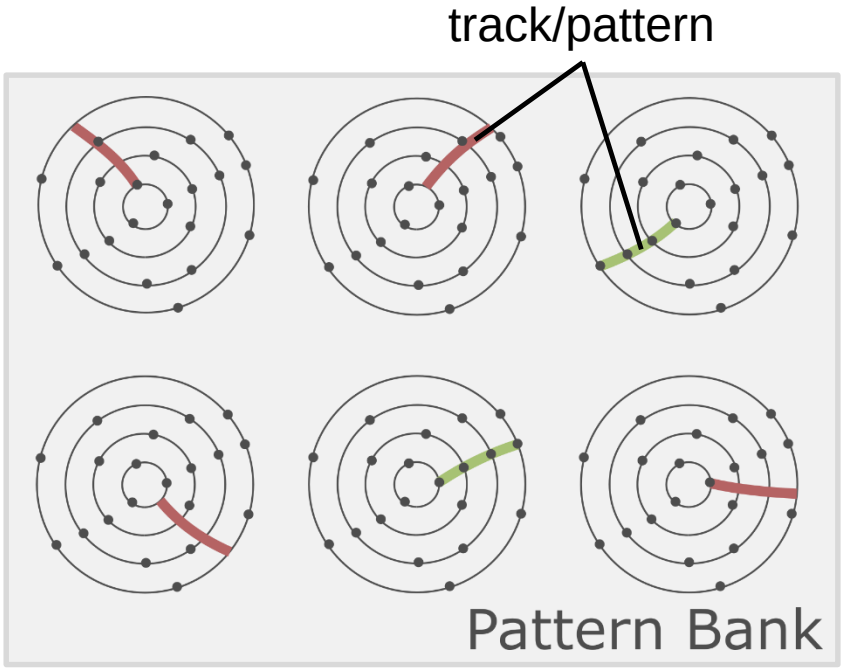
Pattern recognition – idea

- we know what interesting particle tracks look like
- generated pattern bank of interesting tracks
  - one pattern describes a particle track through the detector
- comparison event ↔ pattern bank



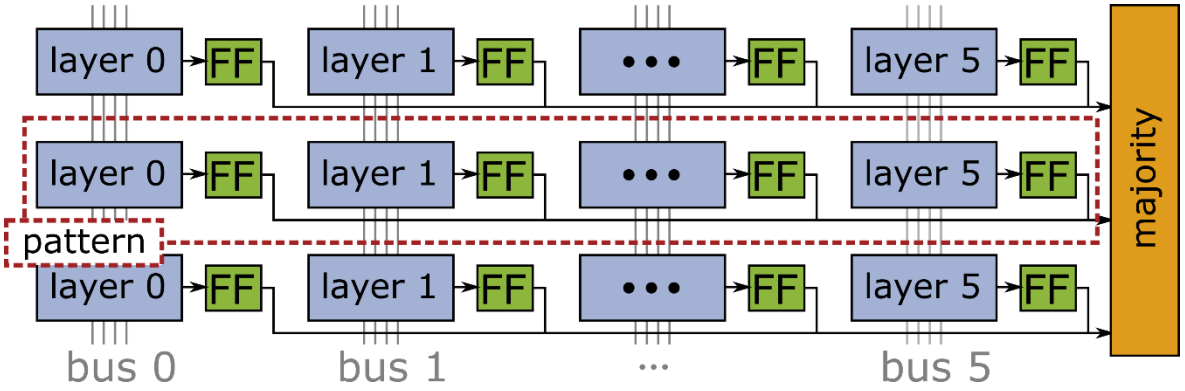
L1 Collision Event

compared to



Current hardware system

- different approaches for the new L1 Track Trigger
  - Two FPGA-based and one that uses FPGA + ASIC-based AM
- Associative Memory architecture (AM chip)
  - provides hit result within one clock cycle

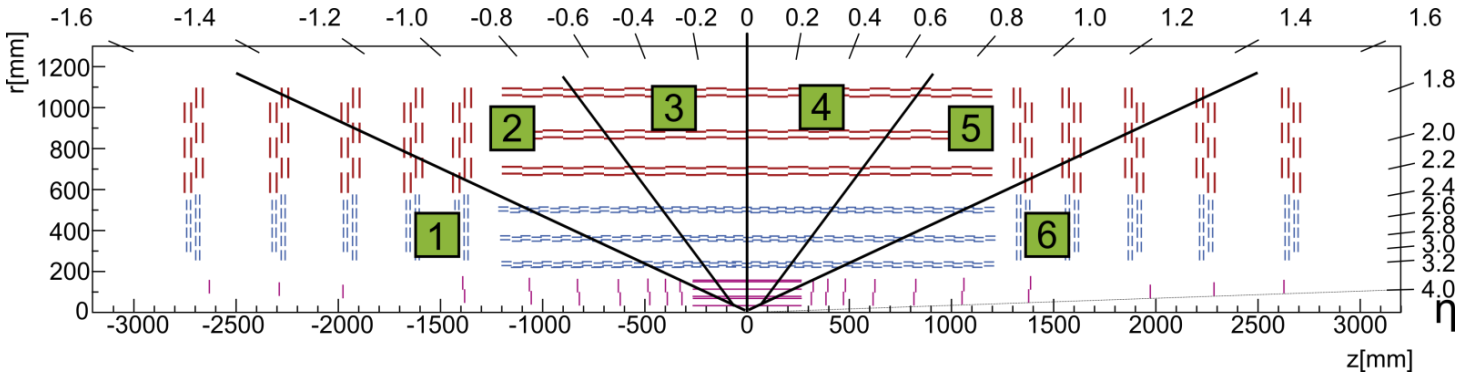


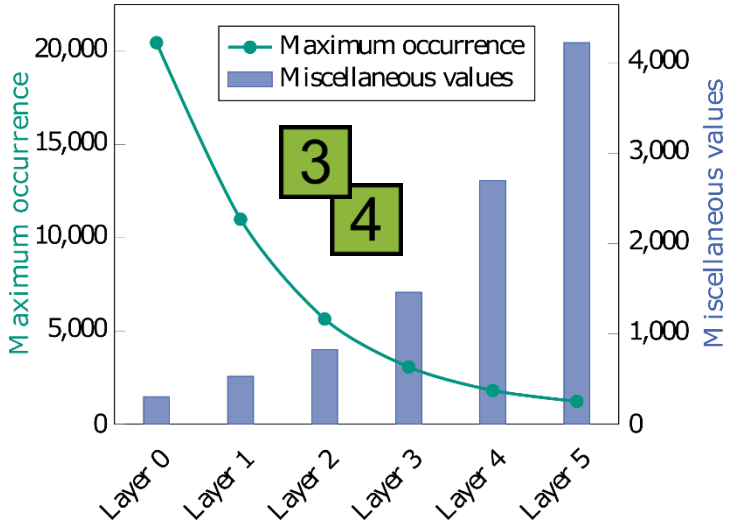
- implementation on an FPGA is not efficient
  - Large overhead

an independent R&D

➡ use logic instead of memory: Analyze data to be stored

Analyzed pattern bank



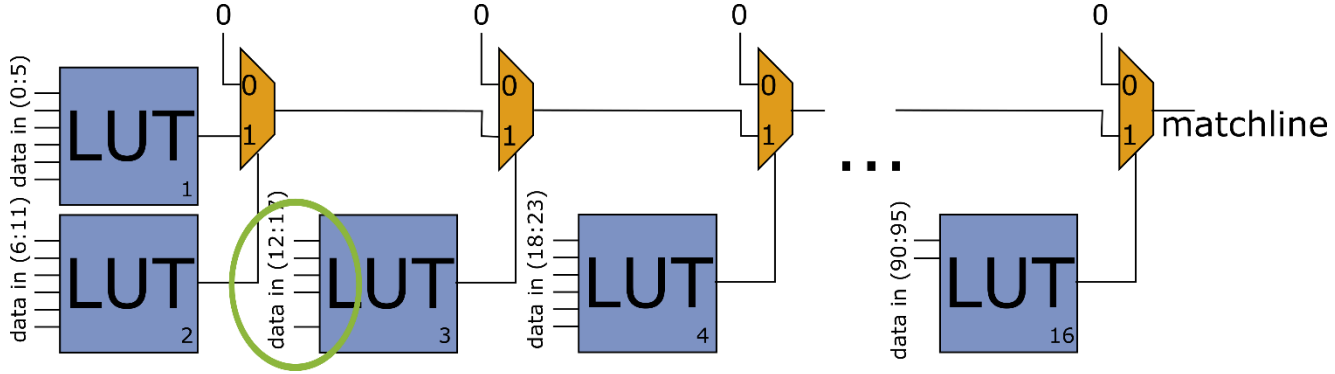


- Layer 0
  - ~300 miscellaneous values (sensors)
  - >20000 matches per input possible
- Layer 5
  - >4200 miscellaneous values
  - max. 1230 matches per input

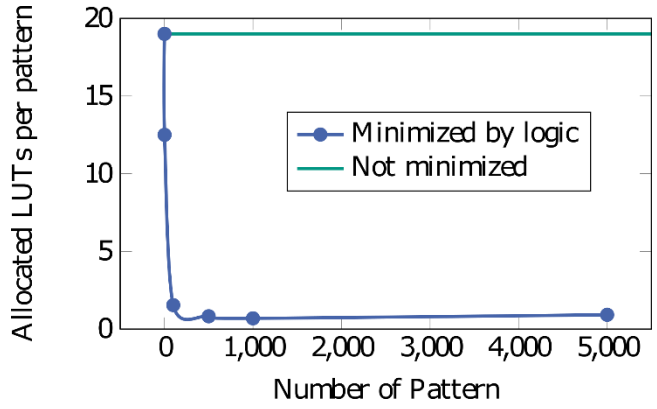
➡ Minimization by logic

FPGA approach

- LUT structure for two patterns



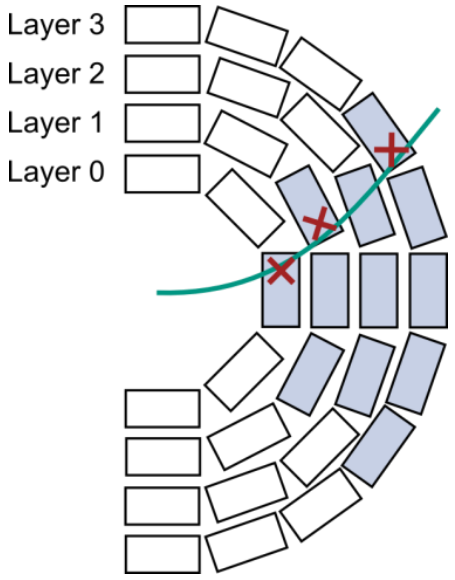
- first results
  - decision: is the track stored in the pattern bank?



Extensive minimization by logic is possible

Comparison with AM chip

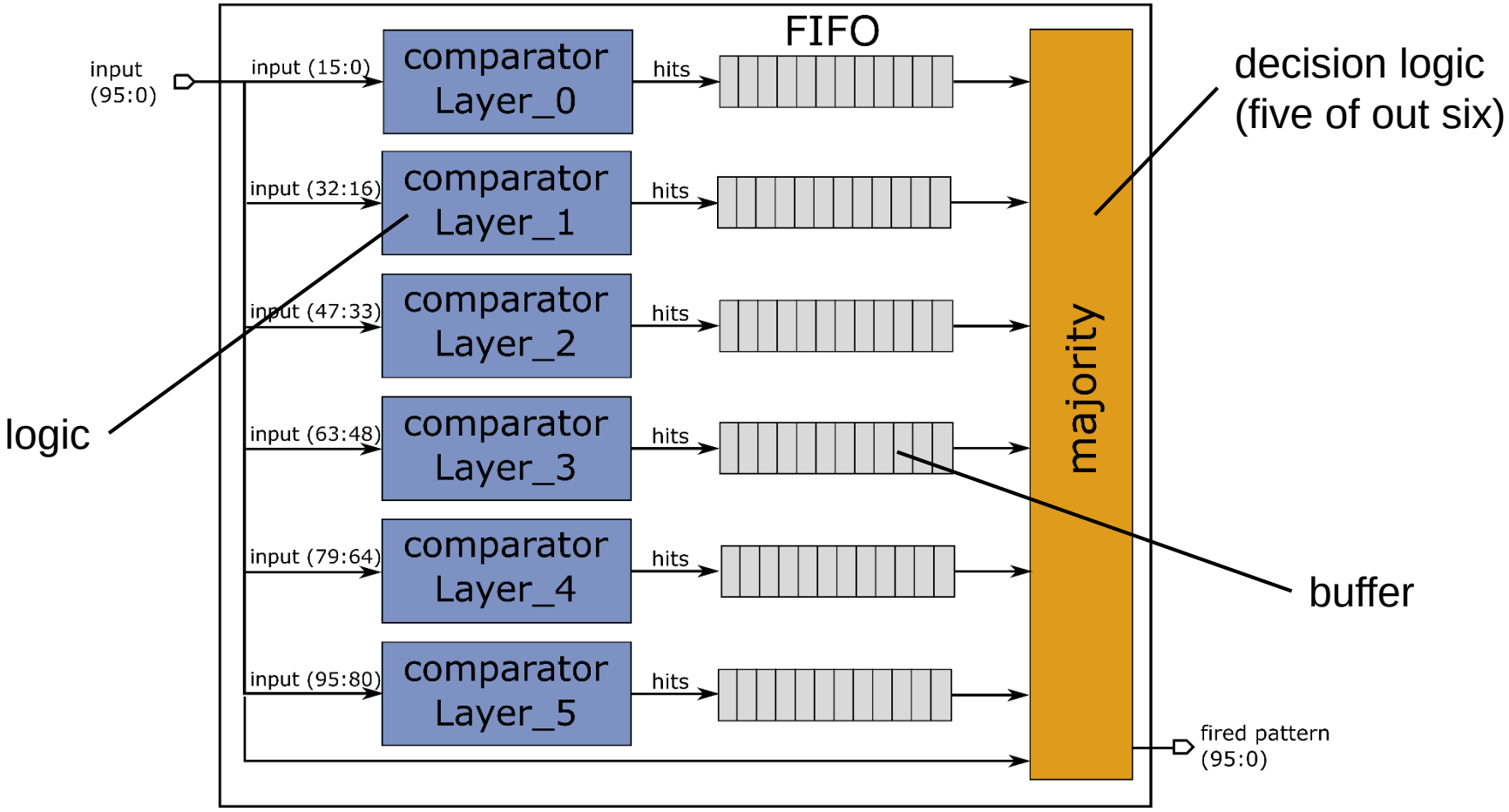
- AM chip offers additional features
- writeable memory
  - synthesize the FPGA



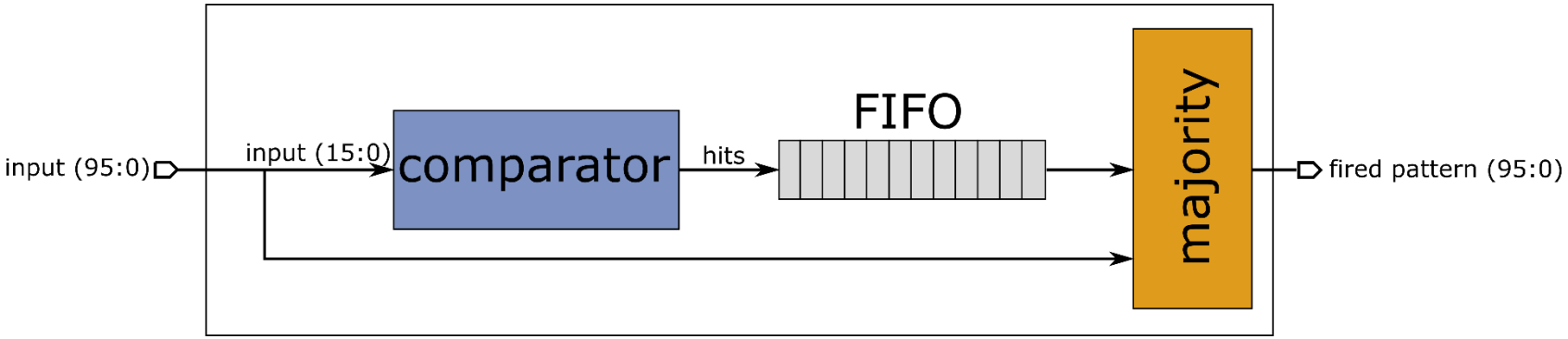
- handle failure layers
  - split pattern into layers (96 Bits → 6*16 Bits)

→ Layer-based approach

Layer based approach

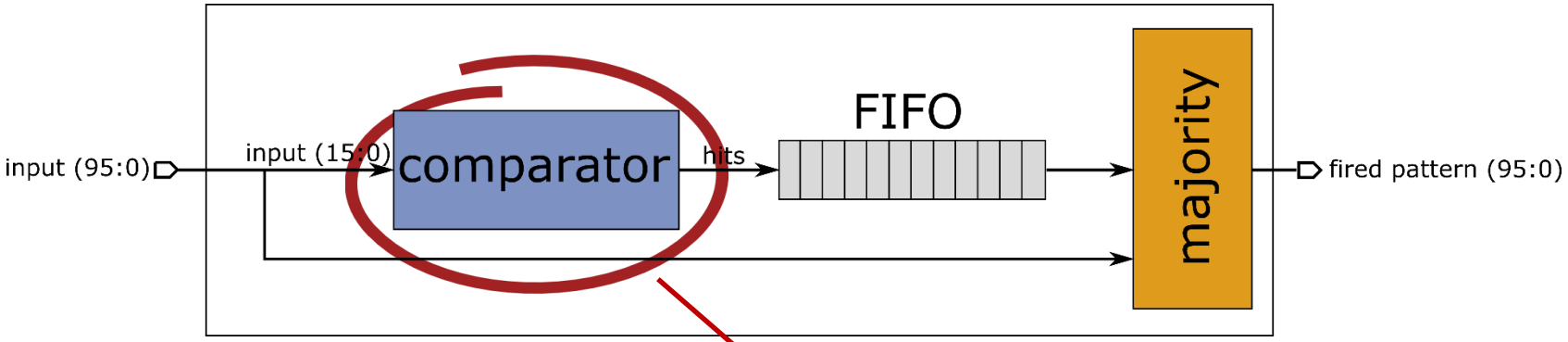


Layer-based approach – one layer



- comparator unit
  - input: 16 Bits
  - output: n Bits fired pattern number
- FIFO unit
  - buffers fired pattern
- majority unit
  - decision (five out of six)
  - output: fired pattern

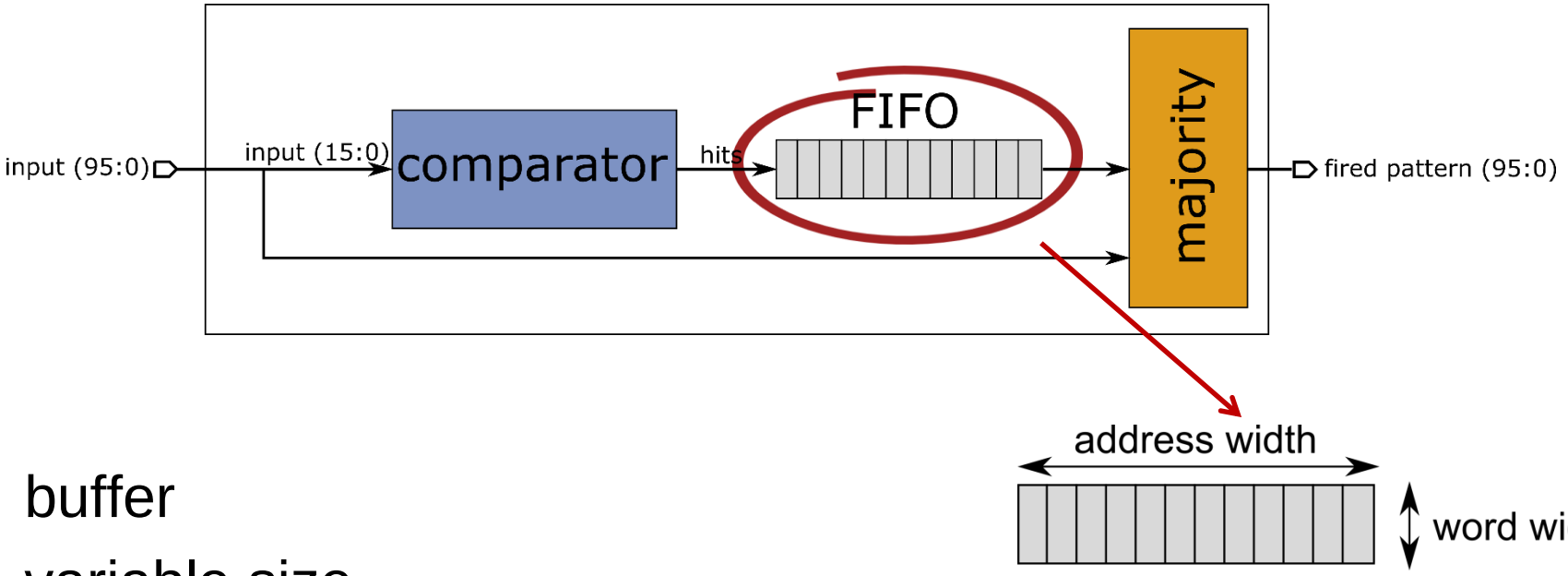
Layer based approach – comparator



- pure logic (state machine)
  - pre-computed
  - depends on stored pattern
- use only lookup tables
- minimize by vendor tools
- hits are sorted strictly

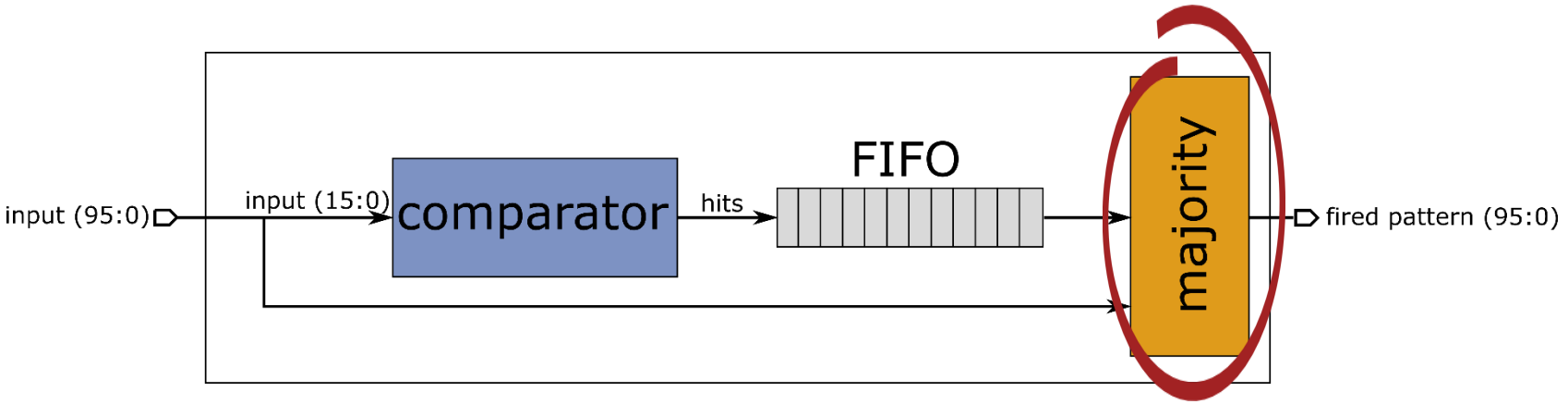
```
if (input="0010000010000111") then
    hit0<="00000000";
end if;

if (input="0001100010000001") then
    hit0<=" 00000001";
    hit1<=" 10001001";
    hit2<=" 10100011";
end if;
```

Layer based approach – FIFO



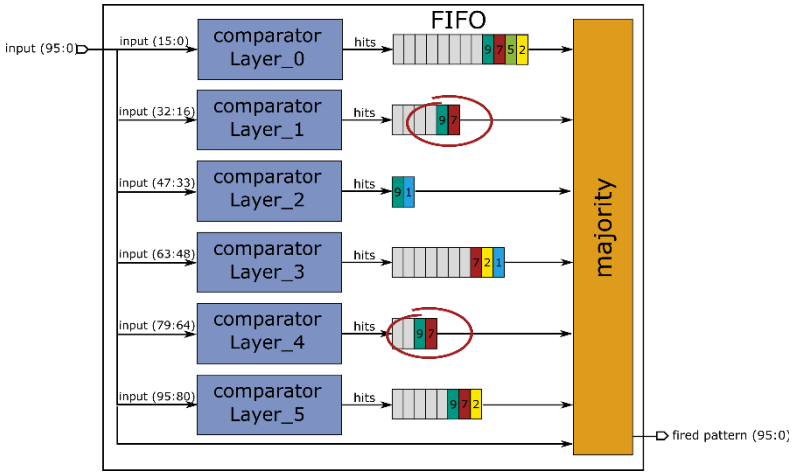
- buffer
- variable size
  - address width: $\log_2$(number of max. possible hits)
  - word width: $\log_2$(number of stored patterns)
- due to the latency: bunch of FIFOs
- content: strictly sorted fired hits

Layer-based approach – majority unit



- selection of fired pattern
  - pure logic
- read out strinctly sorted hits

  9 > 7 > 5 > 2 > 1
- search two largest numbers
  - rest will be discarded



Layer based approach – Implementation results

- Vivado, x7v2000t, 400 entries per FIFO

Number of pattern	Number of LUTs	LUTs per pattern	Frequency in MHz	Latency in μs
1,023	6,983	6.8	238.83	1.67
4,095	29,432	7.1	220.4	1.81
8,191	59,655	7.3	209.12	1.92
16,383	134,750	8.2	146.26	2.74
32,767	307,195	9.4	150.29	2.66

- Also a combination of two designs is possible
  - Reduce overhead

⇒ possibility to store 100,000 patterns per FPGA

Conclusion

- A new memory architecture was developed
  - combines features of content addressable memory and an efficient utilization of FPGA resources
  - stored data is minimized by logic
- Layer based design is running
  - meets strong time constraints
- FPGA implementation still looks feasible
  - has to improve further to be an alternative
- Future work
  - implementation of pipelined structure
    - handle several hits per layer
  - improvement of running design
    - assembling of several consecutive patterns into one package

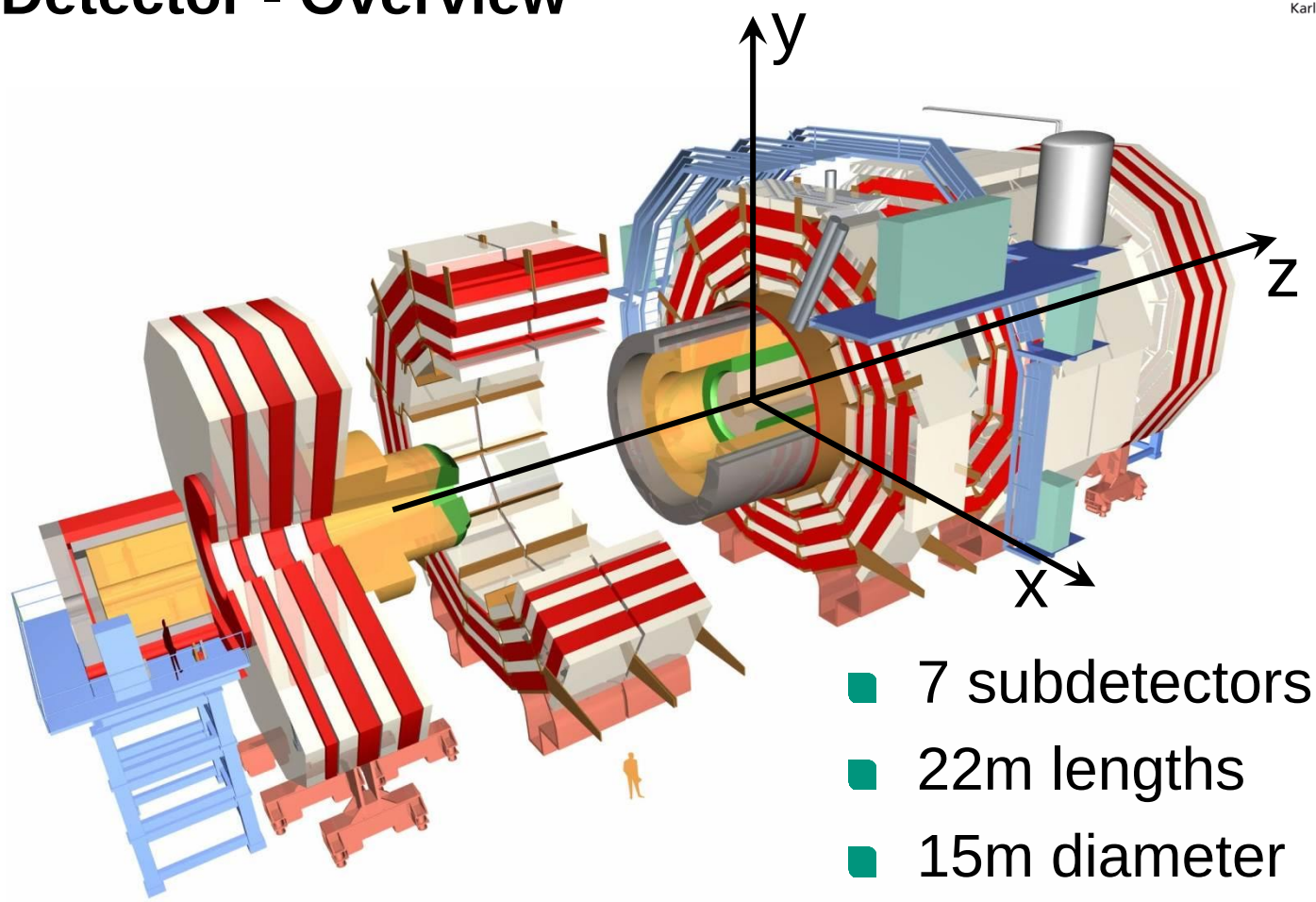
Karlsruhe Institute of Technology

Thank you for your attention.

Dipl. Inform. Tanja Harbaum
Institut für Technik der Informationsverarbeitung
harbaum@kit.edu

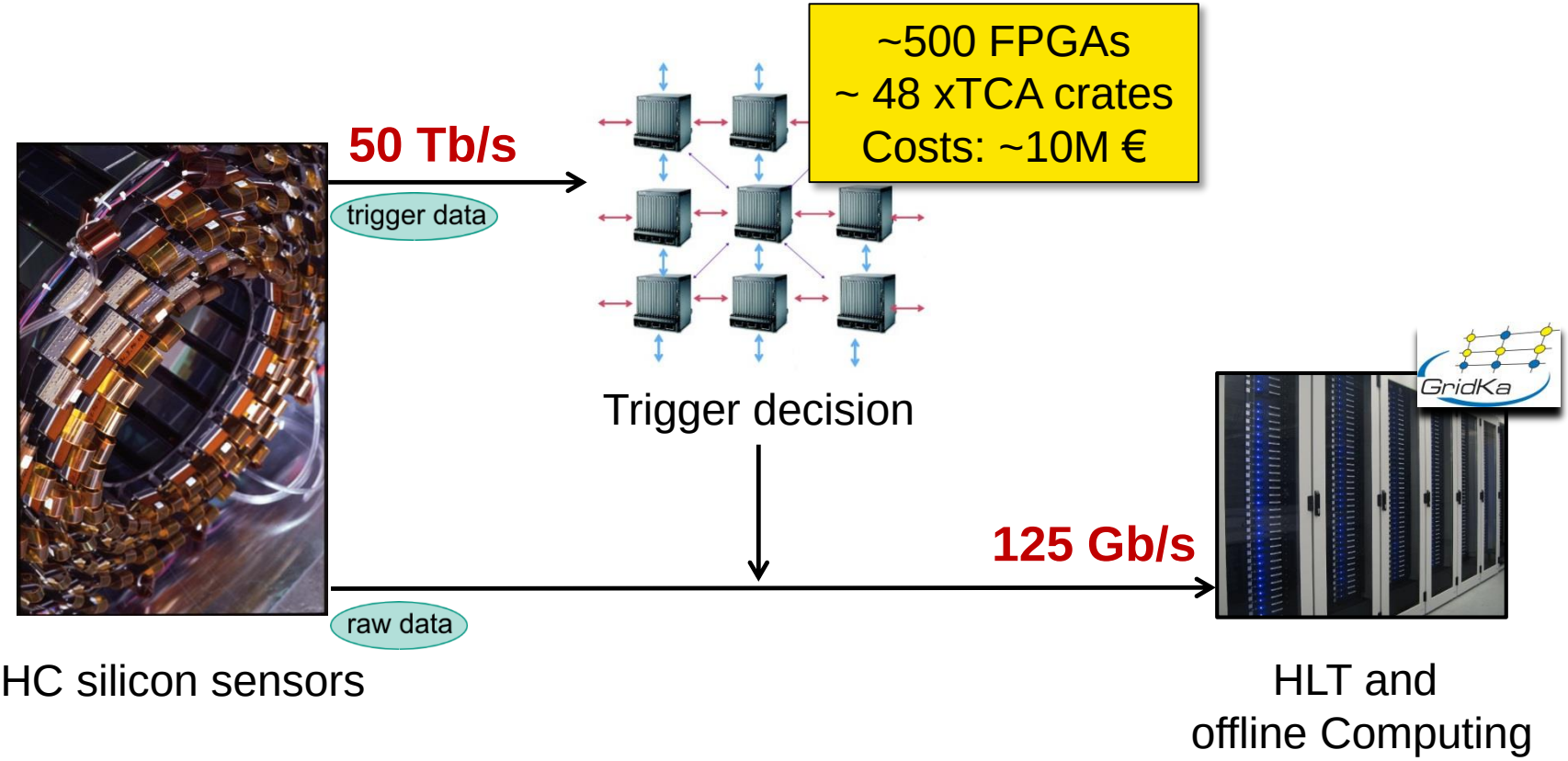
BACKUP SLIDES

CMS Detector - Overview



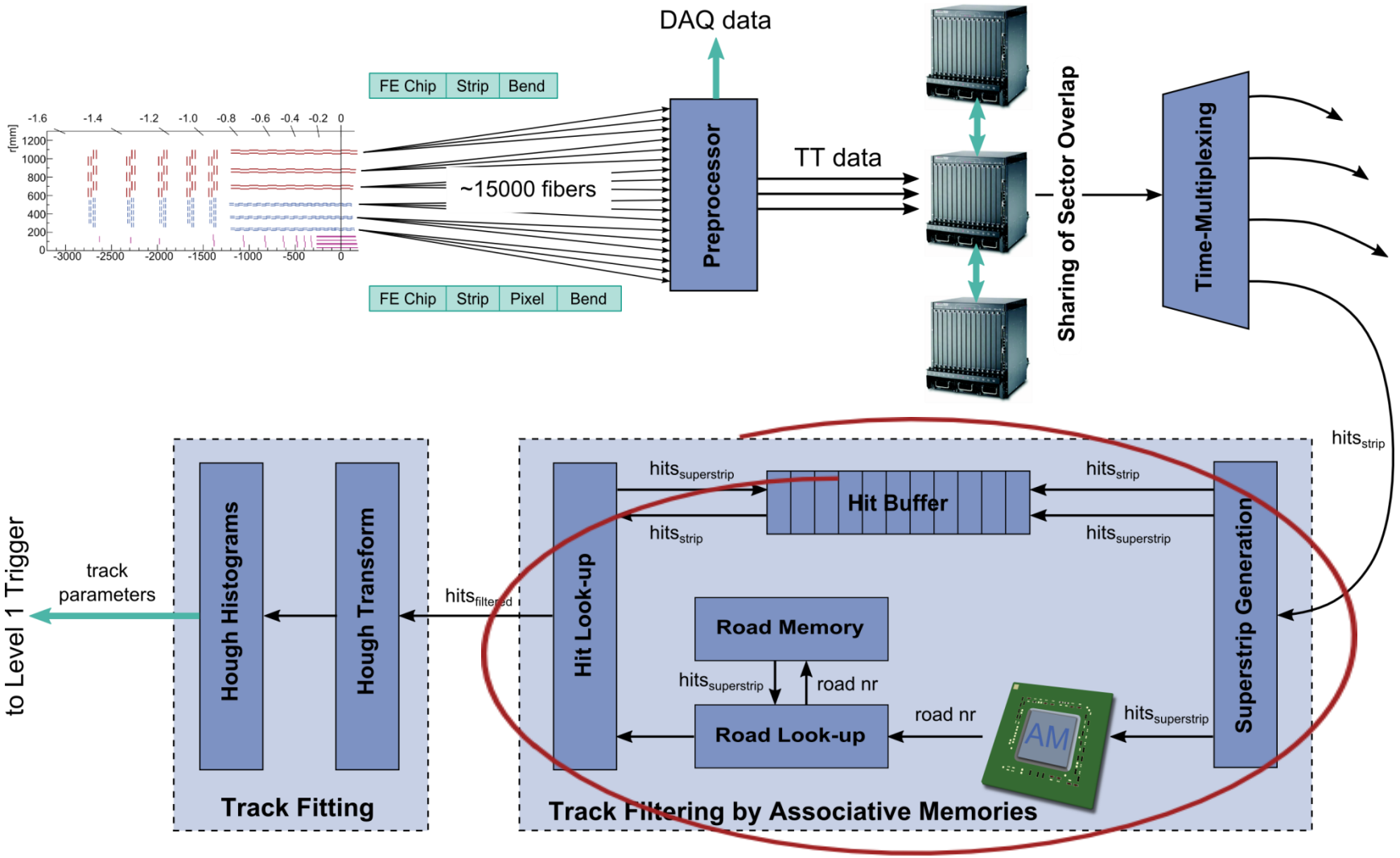
- 7 subdetectors
- 22m lengths
- 15m diameter
- 12.500 tons weight
- ~100 million channels

CMS Experiment - Challenges



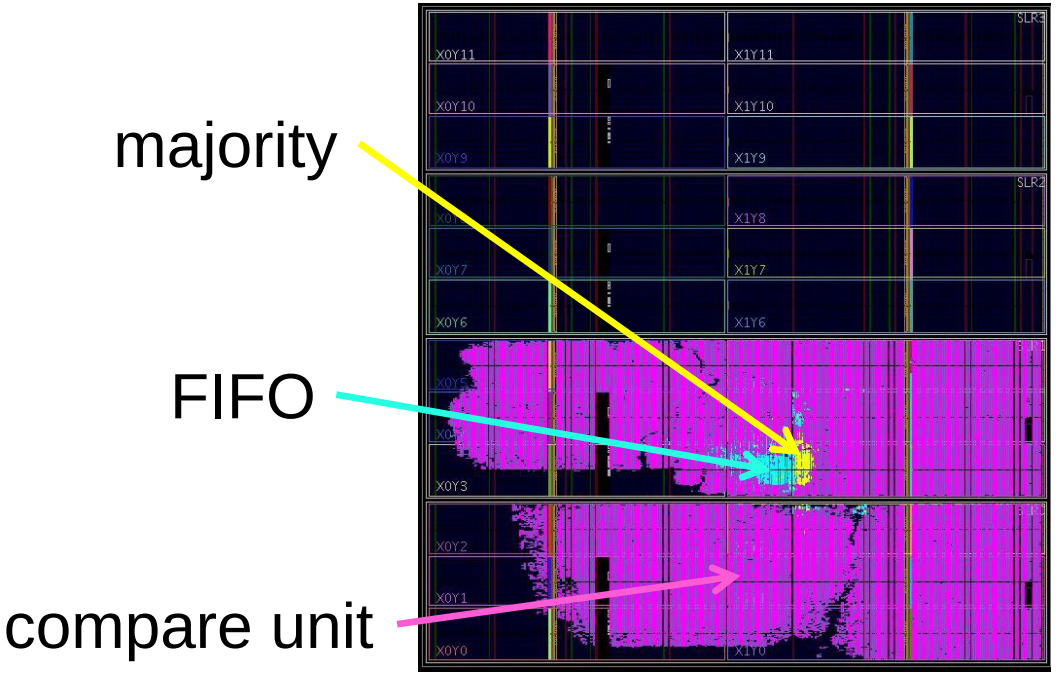
- Reduce data by factor 400 (online)
- Trigger decision latency ~ 12 µs

CMS Track Trigger



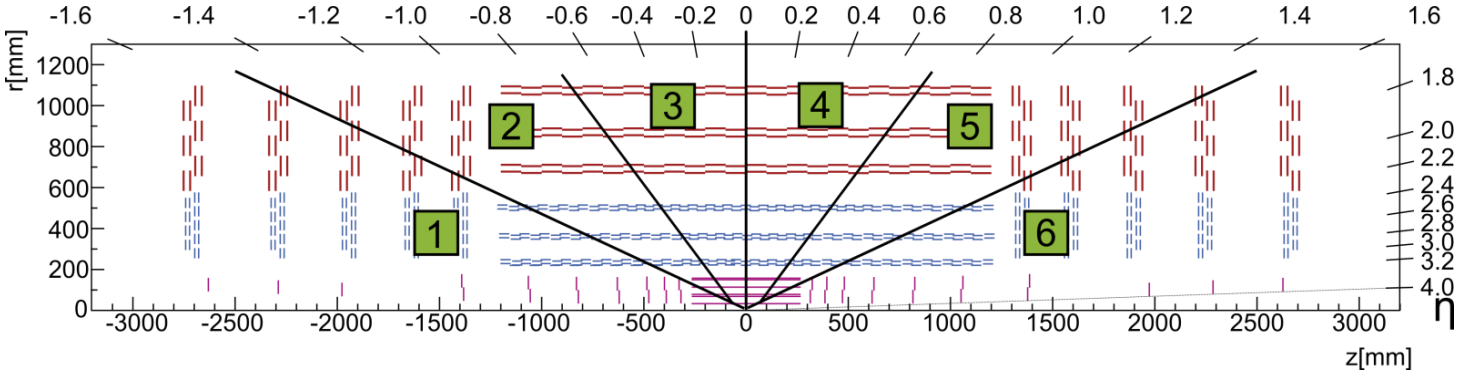
Layer-based approach – first results

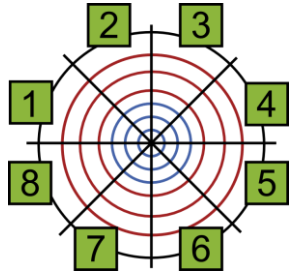
- design for 50k patterns is running
  - 142.857MHz clock cycle
  - 50 FIFOs with max. 400 entries
  - <35% Utilization of an huge up-to-date FPGA (x7v2000t)
  - output: pattern (not road number)



CMS Detector – 48 Trigger Towers

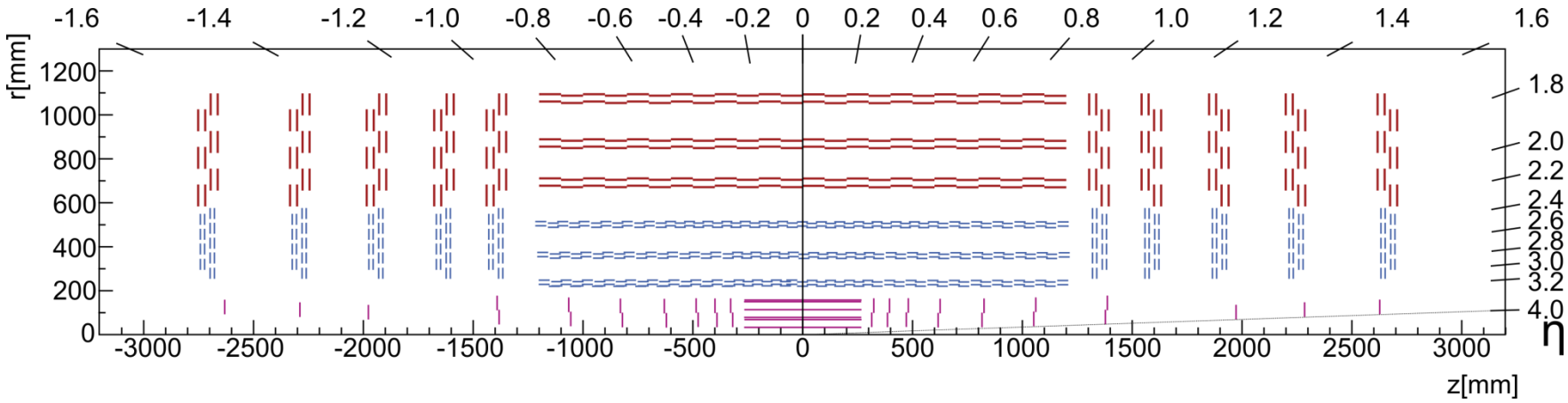
- 6 Sectors (z-η)Plane X 8 Sectors (r-ϕ)Plane





- 48 Trigger sections
  - Up to 500 records per event possible
  - 400 – 600 Gb/s per Trigger Tower
- 50 Tb/s data from sensors
  - ~15.000 fibers (modules) each 3.25 Gb/s

CMS Detector - Barrel-Endcaps Geometry

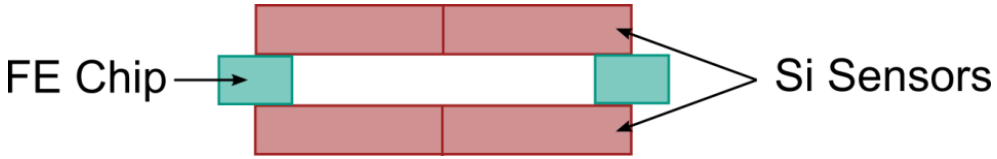


- Inner Pixel
  - 4 BPIX + 10 FBIX
- Outer Tracker
  - 7084 PS Modules (1 macro-Pixel + 1 Strip Sensor)
  - 8424 2S Modules (2 Strip Sensors)

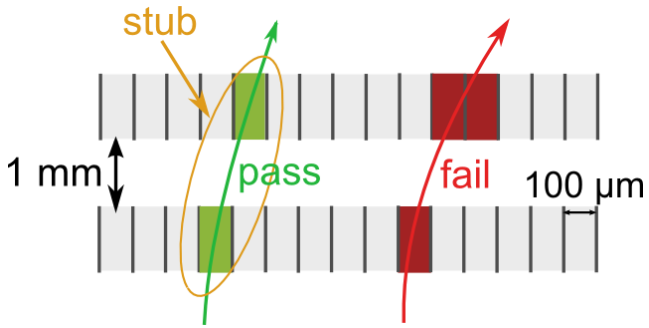
Part used at L1

CMS Detector – Outer Tracker Modules

- p_T Module Concept

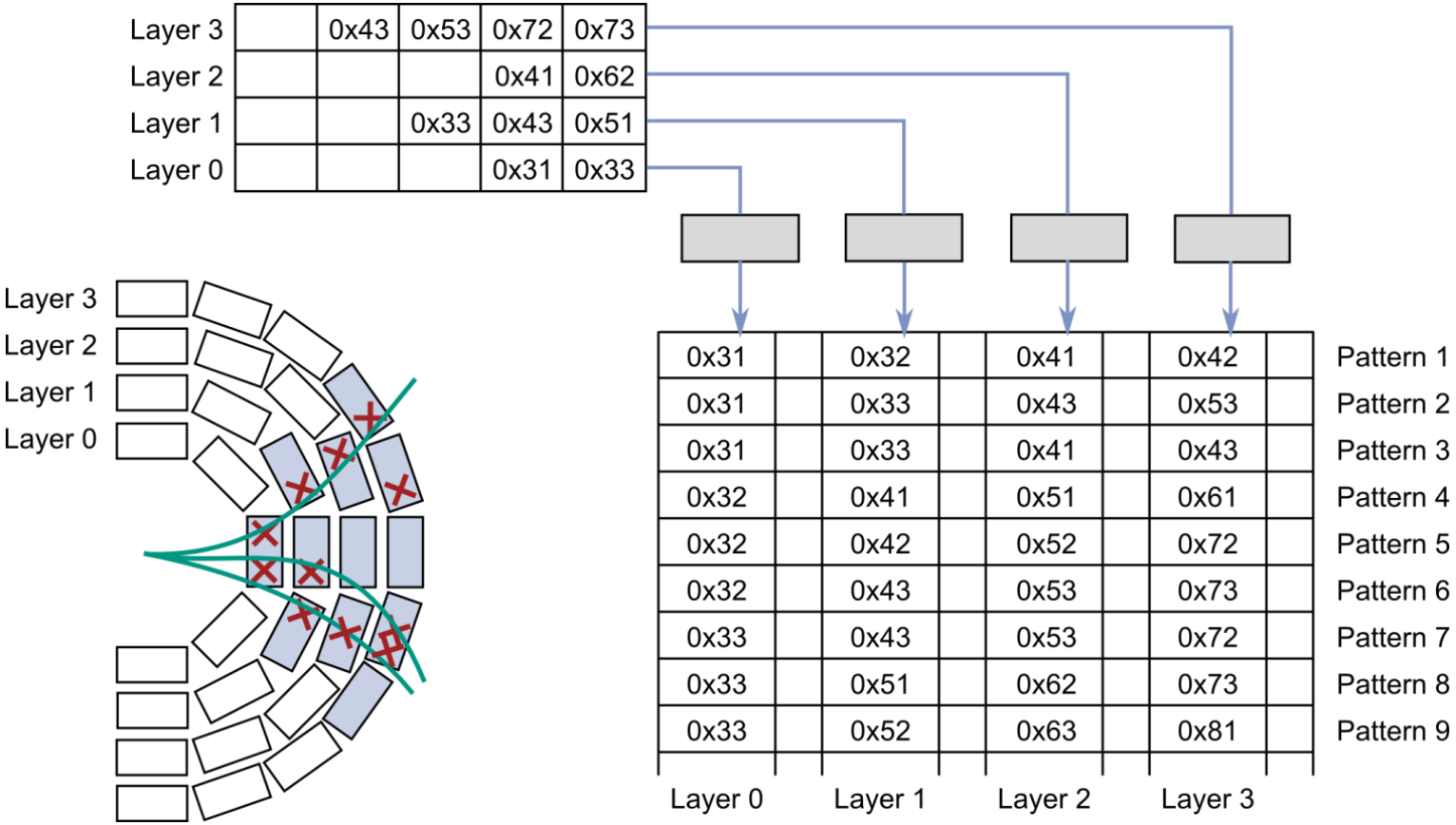


- Stub – Pair of Clusters in the two Sensors

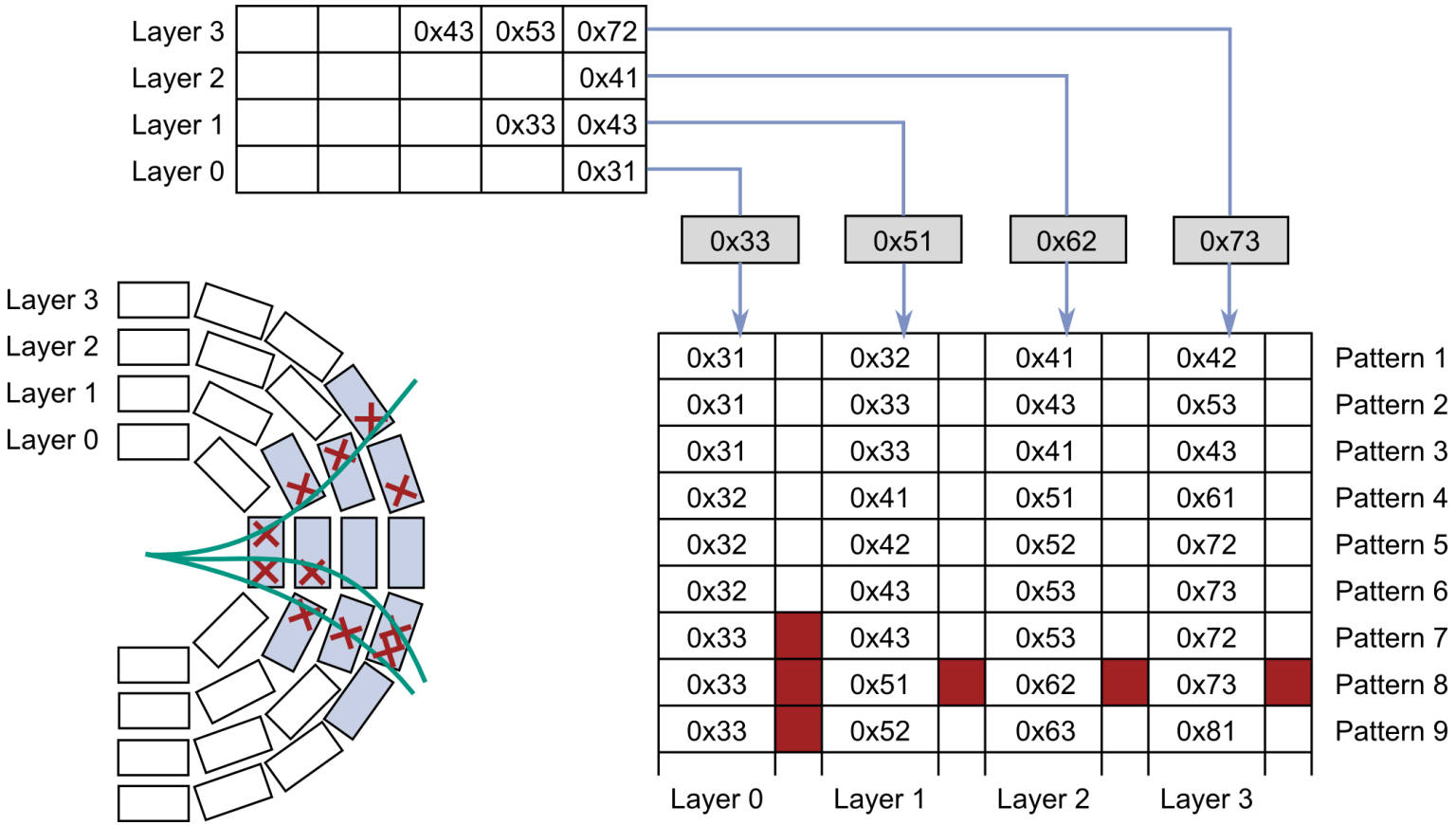


➡ First rough p_T Cut at the Module Level

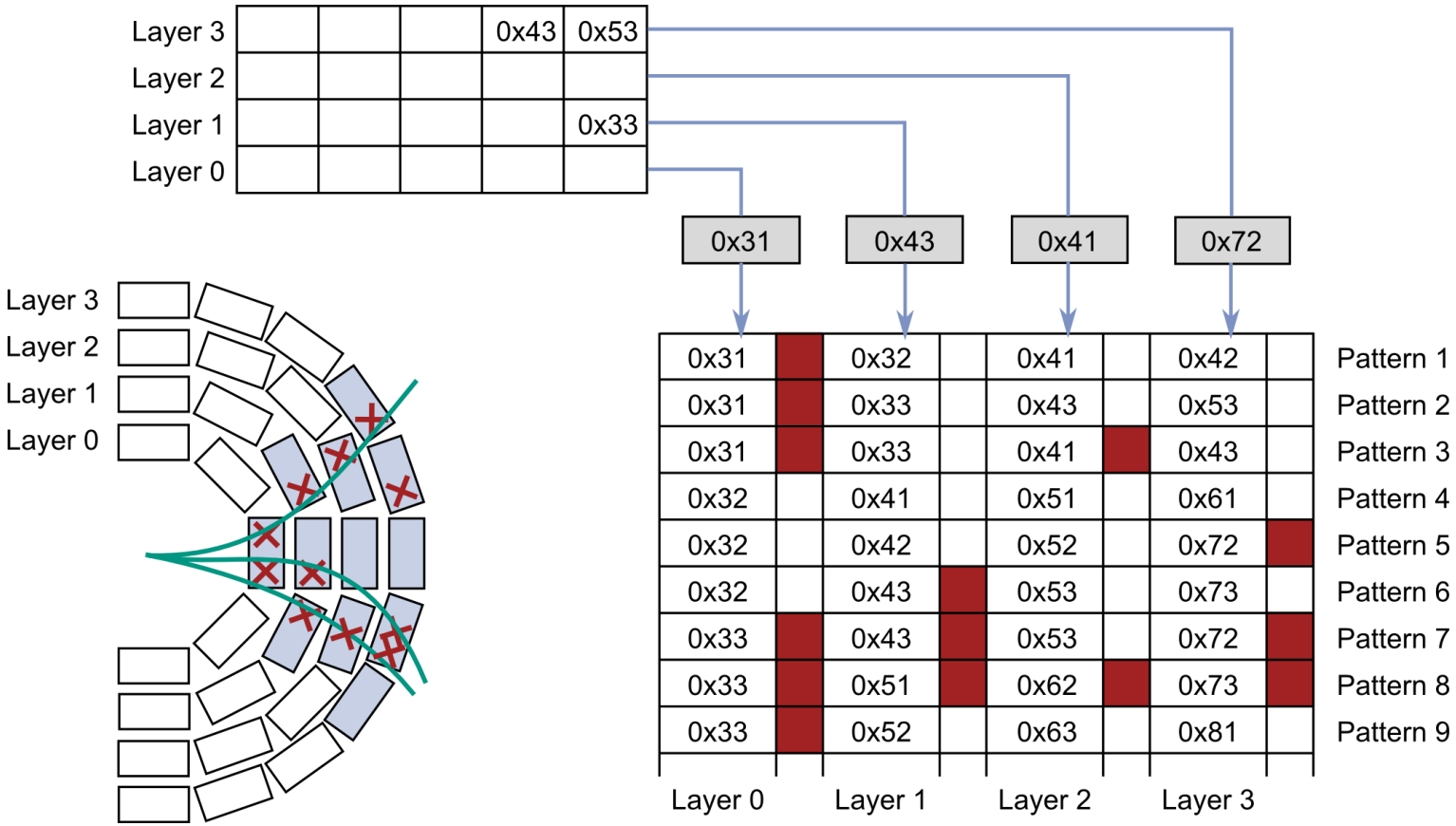
AM Chip – Operating Mode



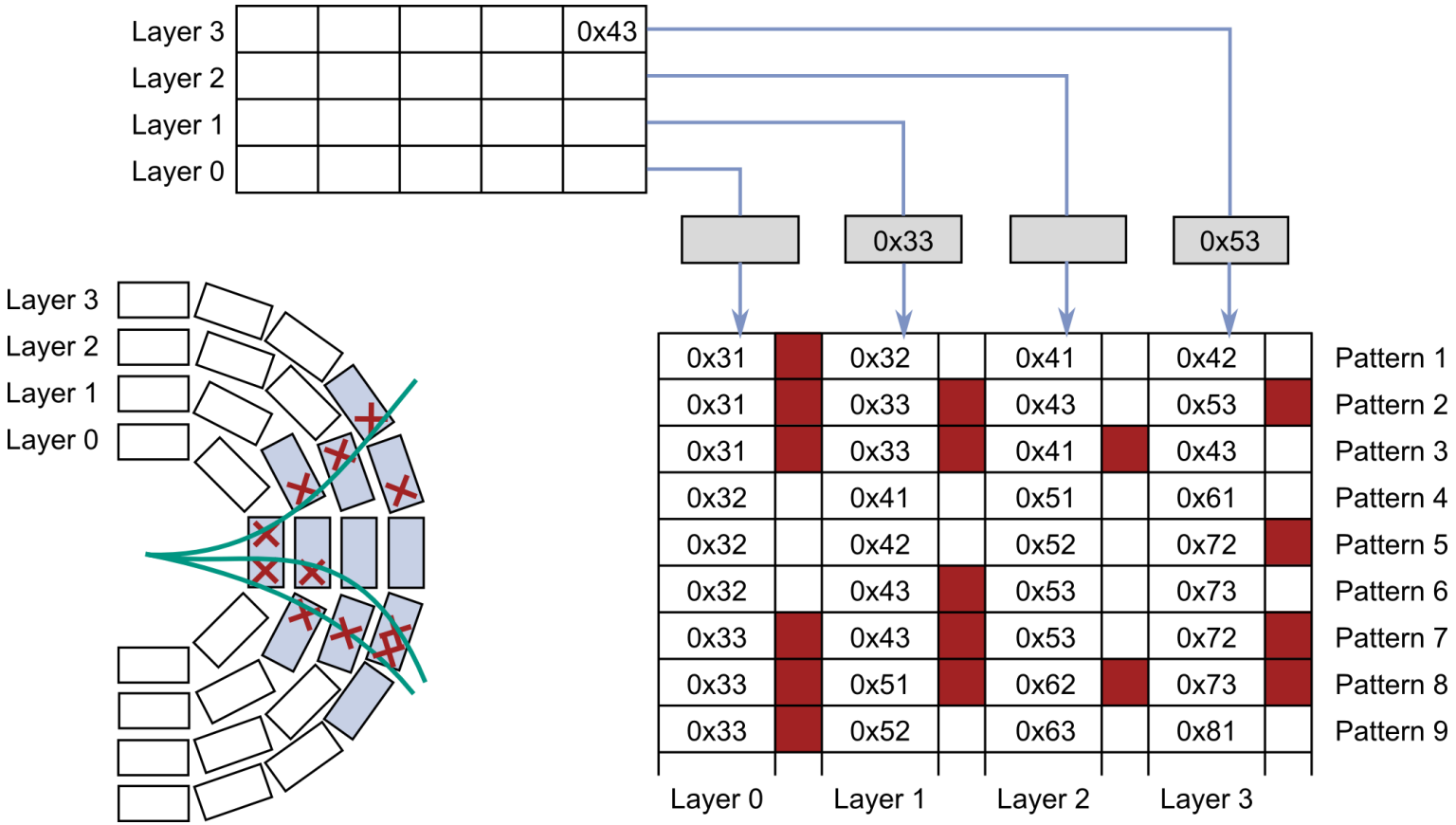
AM Chip – Operating Mode



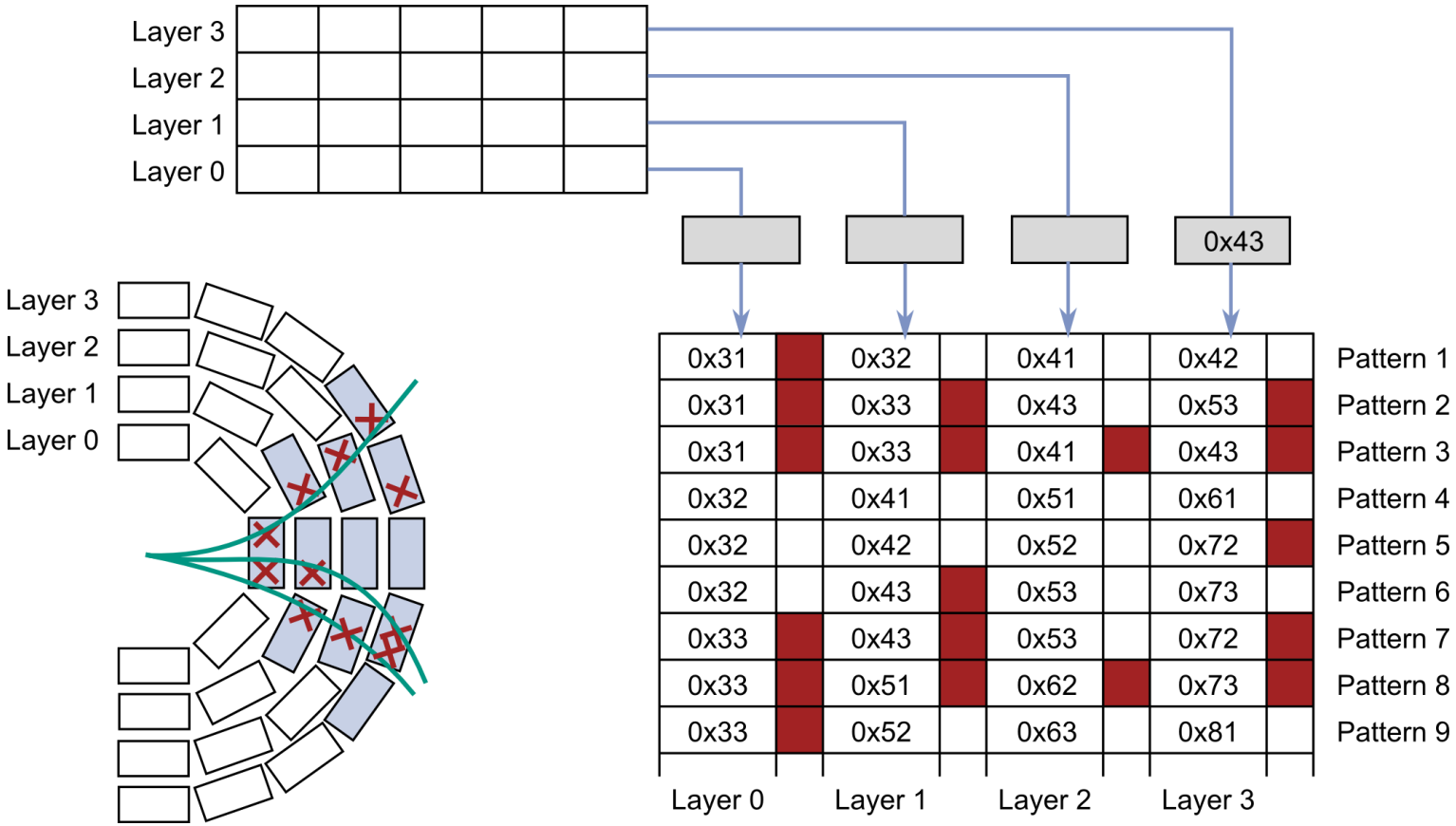
AM Chip – Operating Mode



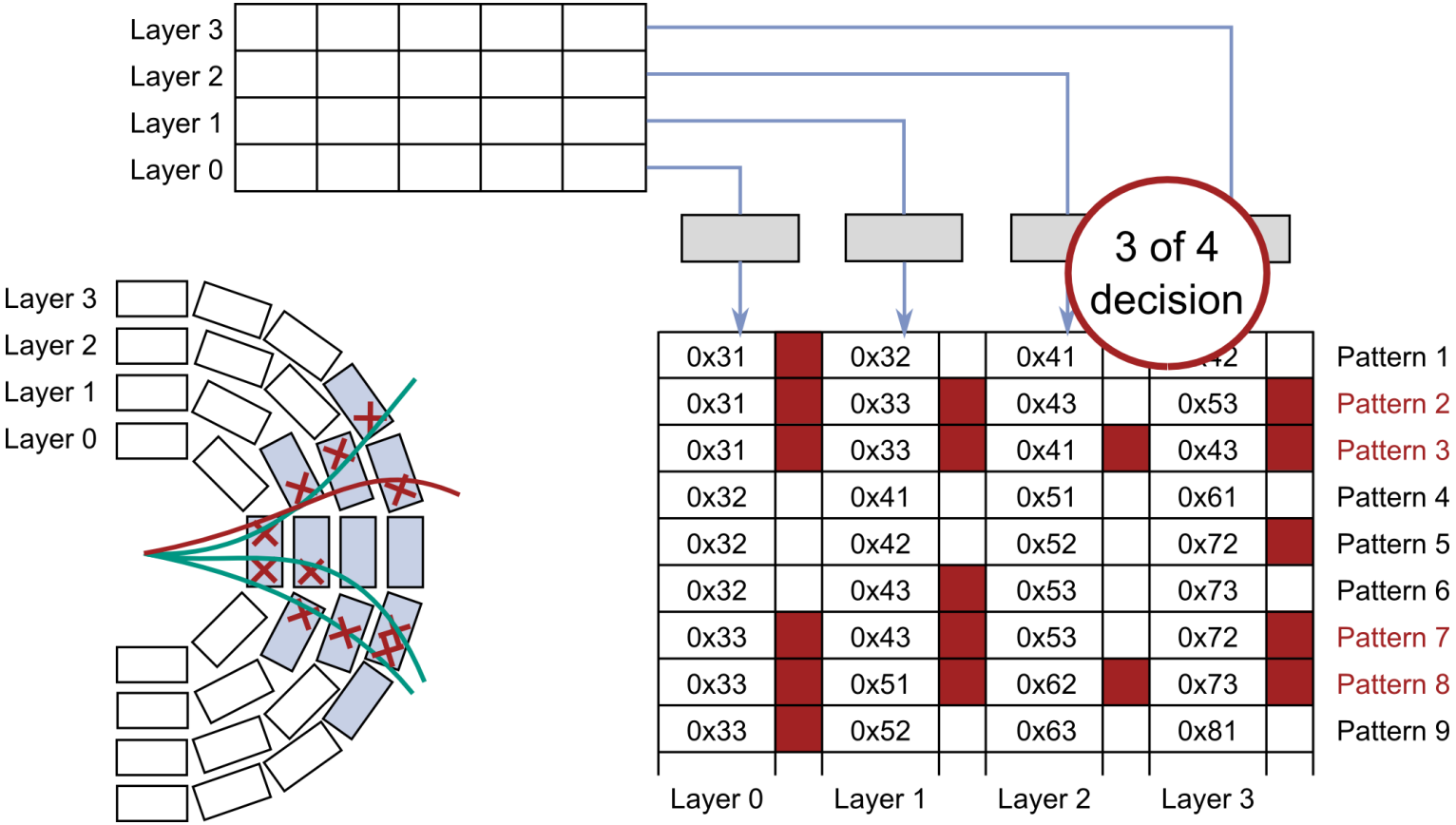
AM Chip – Operating Mode



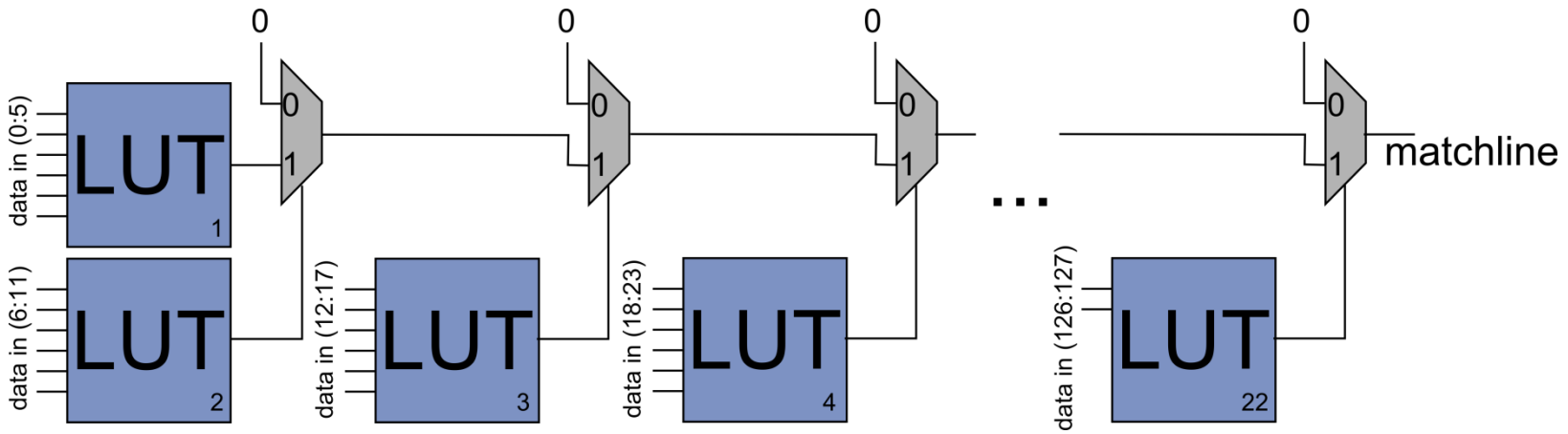
AM Chip – Operating Mode



AM Chip – Operating Mode

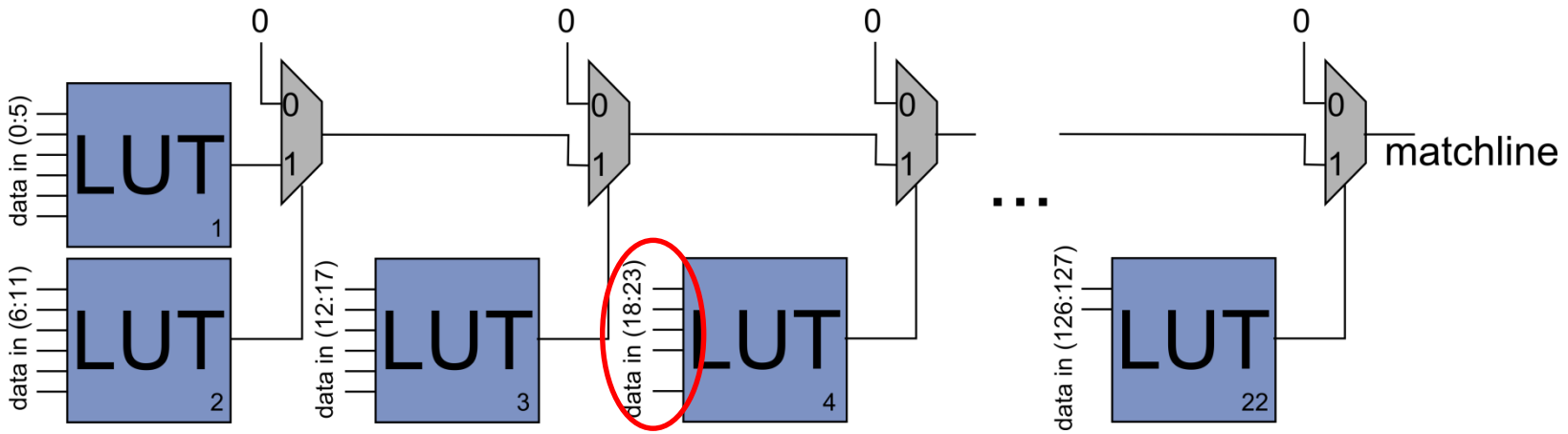


FPGA Approach – Minimization by Logic



- LUT structure for one pattern
  - 22 LUTs
  - Pure combinatorial logic – no clock cycle
  - Plus one 128 bit register to store the input

FPGA Approach – Minimization by Logic



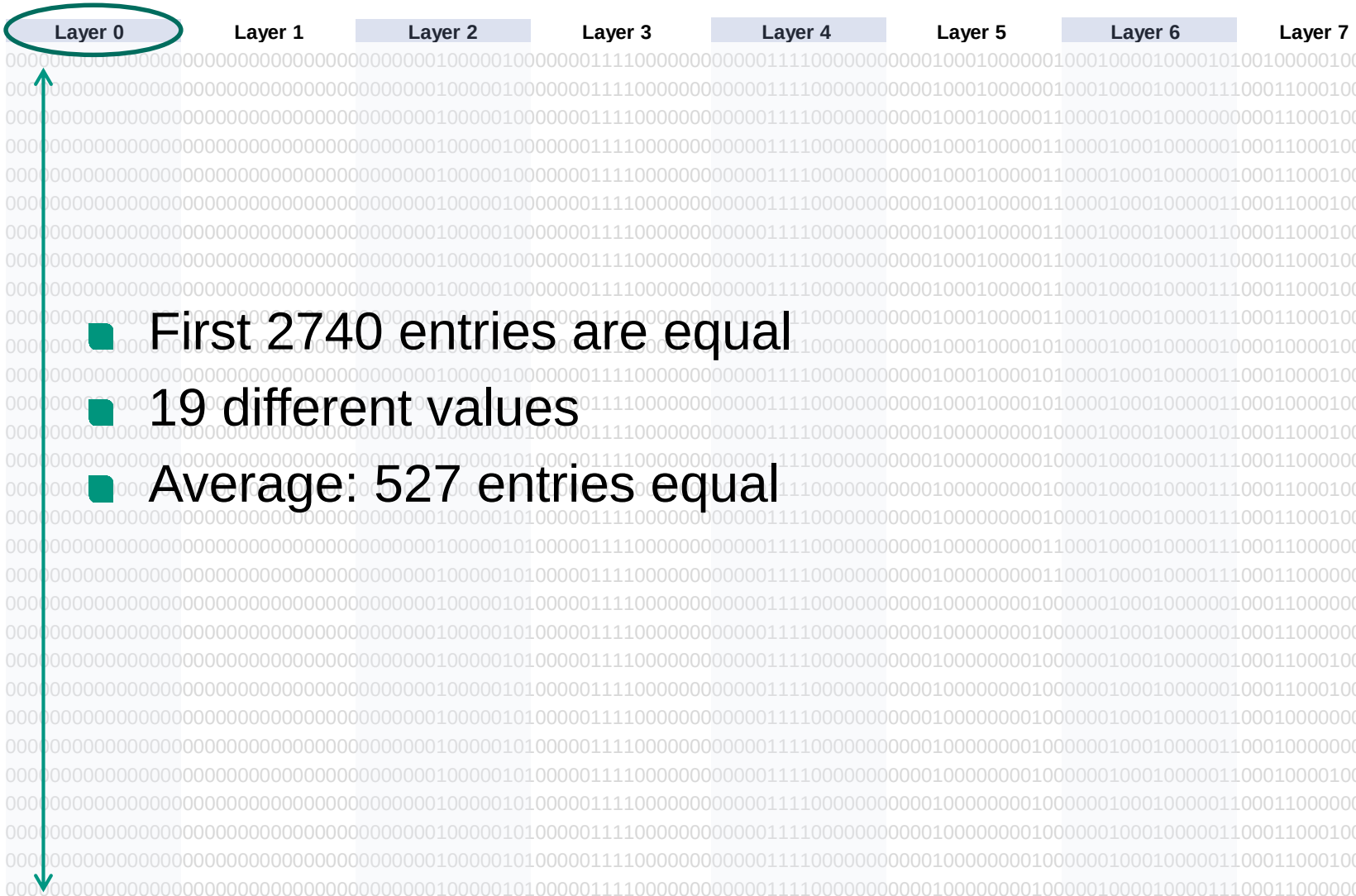
- LUT structure for two patterns
  - One bit differences
  - 22 LUTs
  - Pure combinatorial logic – no clock cycle
  - Plus one 128 bit register to store the input

➡ Two instead of one pattern contained in 22 LUTs

Pattern Bank

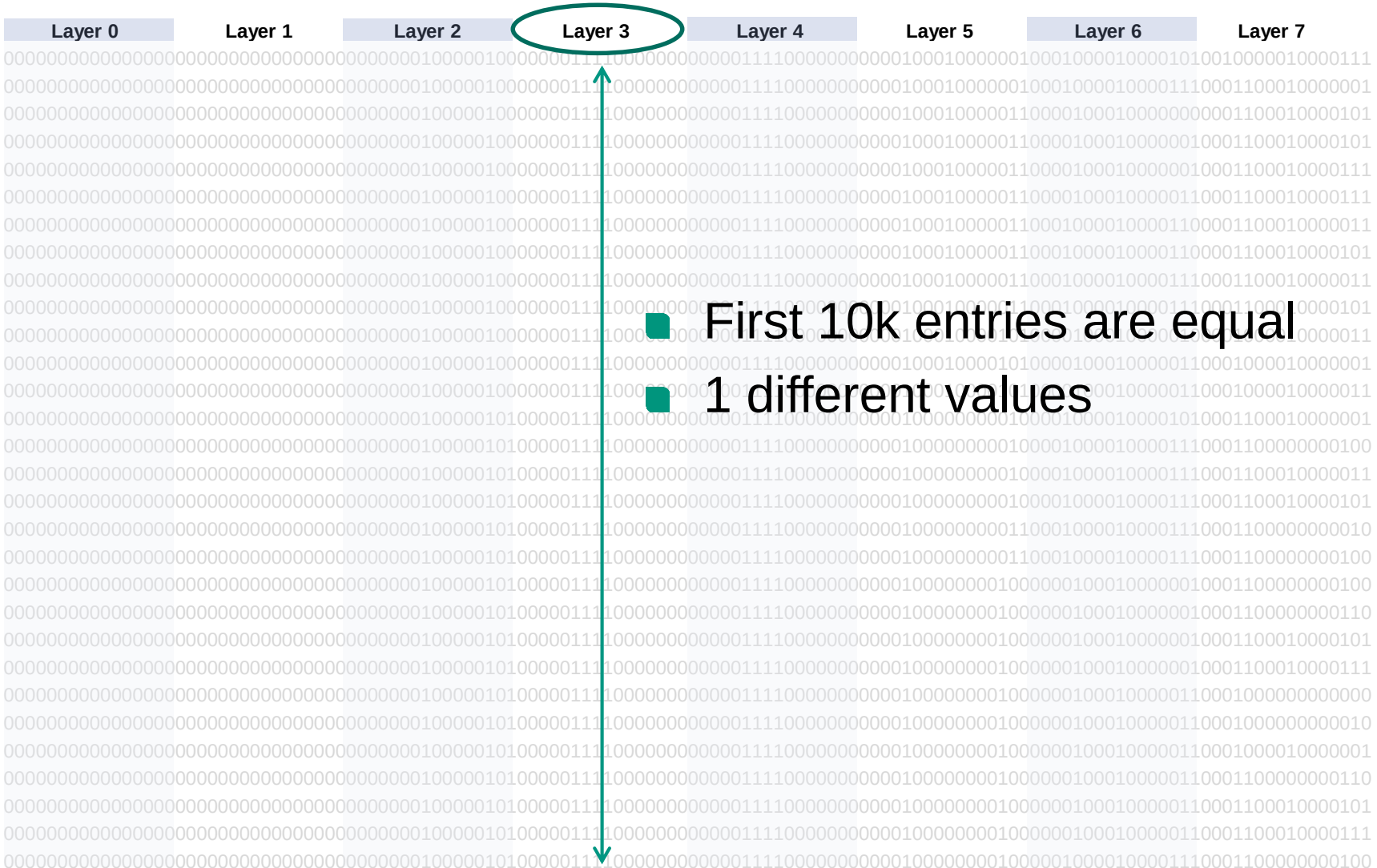
Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000100	0100010000101001	0000010000010111
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000100	0100011100011000	0100000100000001
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0000100000000011	0000100010000101
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000010000001	0001100010000101
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000010000001	0001100010000111
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000010000011	0001100010000111
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000011000011	0001100010000011
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000010000101	0001100010000101
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000011000011	0001100010000011
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000110	0001000011000111	0001100010000101
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000101	0000100000000011	0001000010000011
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000101	0000100000000011	0001000010000001
0000000000000000	0000000000000000	0000000100000100	0000001111000000	0000011110000000	0001000100000101	0000100000000011	0001000010000011
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0100010000101001	0001100010000001
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0100011100011000	0001100000000100
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0100011100011000	0001100010000011
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0100011100011000	0001100010000101
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001100	0100011100011000	0001100000000010
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001100	0100011100011000	0001100000000100
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100000000100
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100000000110
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100010000101
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100010000111
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000011000010001	0000000000000000
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0000100000000010
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001000010000001
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100000000110
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100010000101
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100010000111
0000000000000000	0000000000000000	0000000101000001	0000001111000000	0000011110000000	0001000000001000	0000010000010001	0001100000000100

Pattern Bank – Analysis 10k Pattern



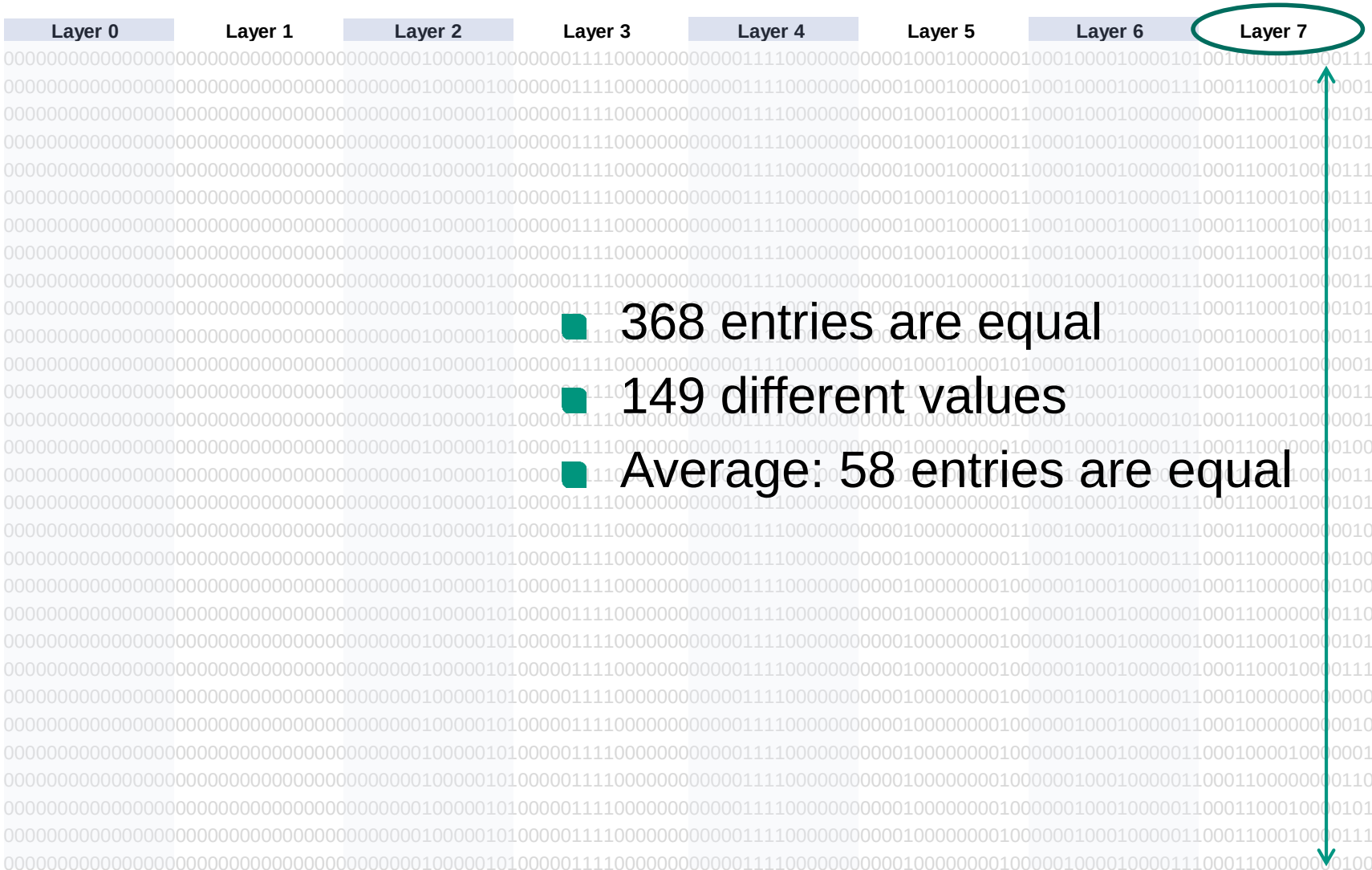
- First 2740 entries are equal
- 19 different values
- Average: 527 entries equal

Pattern Bank – Analysis 10k Pattern



- First 10k entries are equal
- 1 different values

Pattern Bank – Analysis 10k Pattern



- 368 entries are equal
- 149 different values
- Average: 58 entries are equal