Institut für Technik der
Informationsverarbeitung

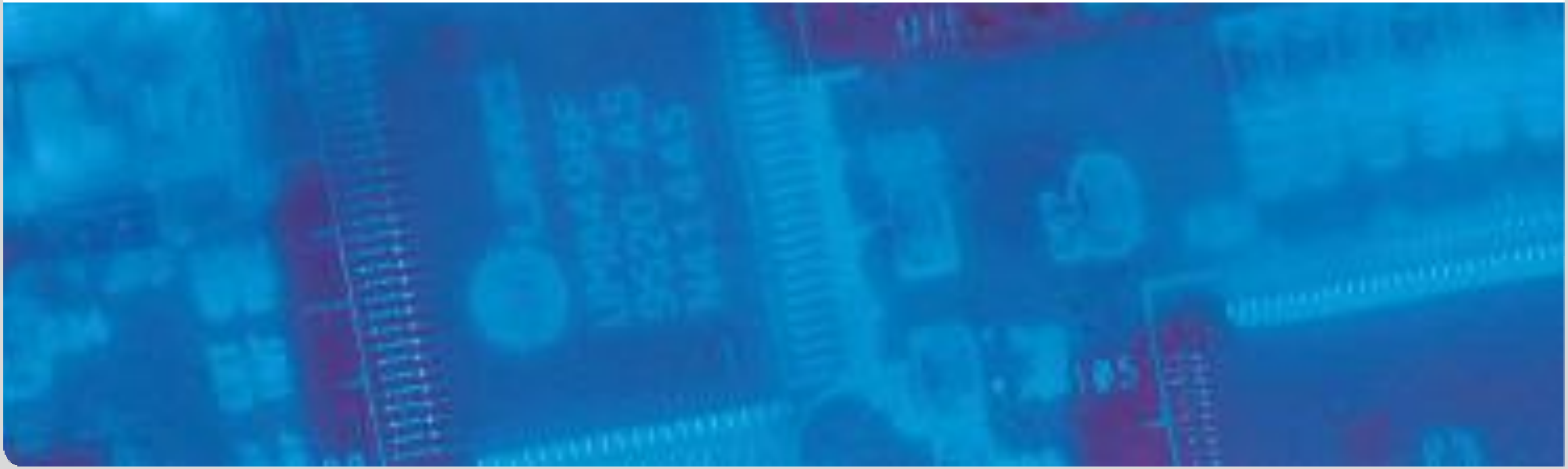
Management
Prof. Dr.-Ing. Dr. h.c. J. Becker
Prof. Dr.-Ing. Eric Sax
Prof. Dr. rer. nat. W. Stork

FPGA implementation of an Associative Memory

Institut für Prozessdatenverarbeitung
und Elektronik (**IPE**)
Management
Prof. Dr. rer. nat. M. Weber

Tanja Harbaum

Institut für Technik der Informationsverarbeitung (ITIV)

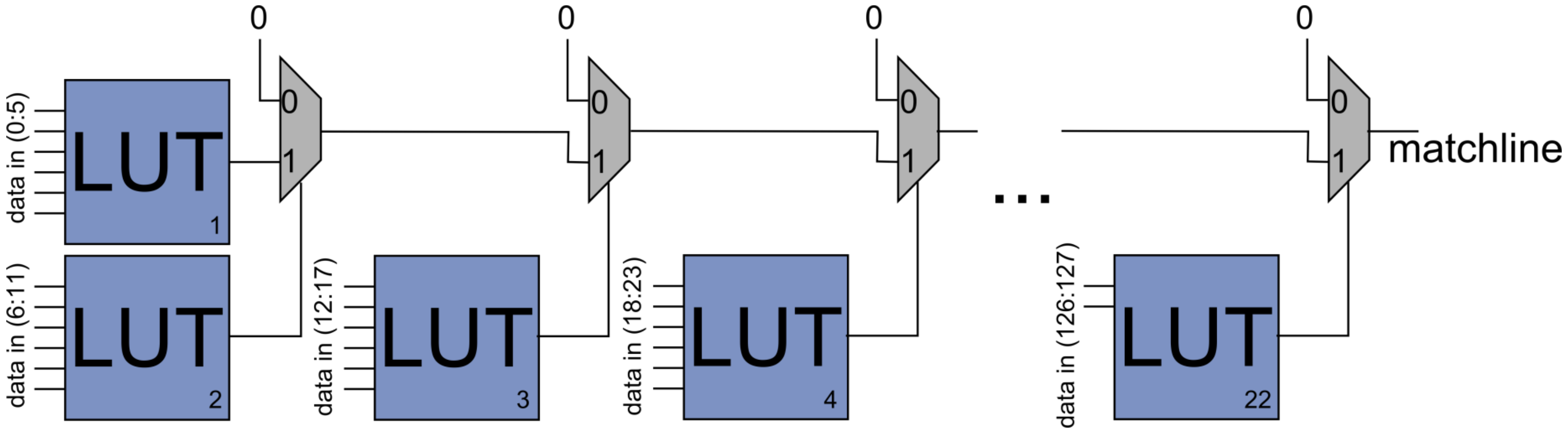


Summary – last talk

- Associative Memory architecture
  - provides hit result within one clock cycle
- FPGA architecture
  - less memory
  - many programmable logic units (Look Up Tables - LUTs)
  - efficient implementation of a memory architecture is not possible

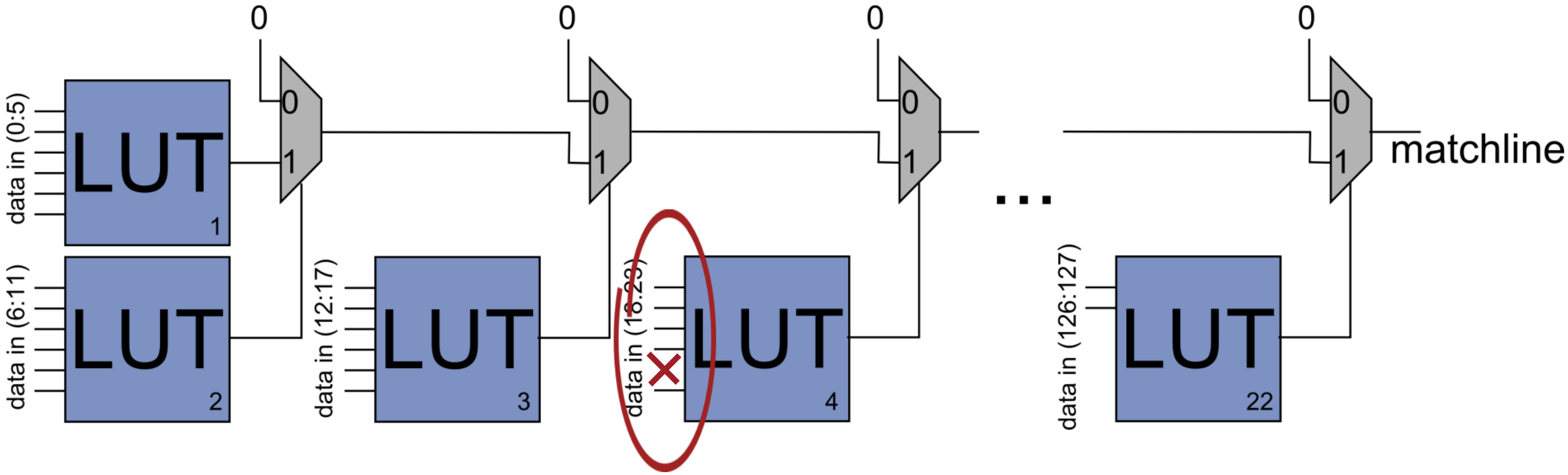
➡ use logic instead of memory

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

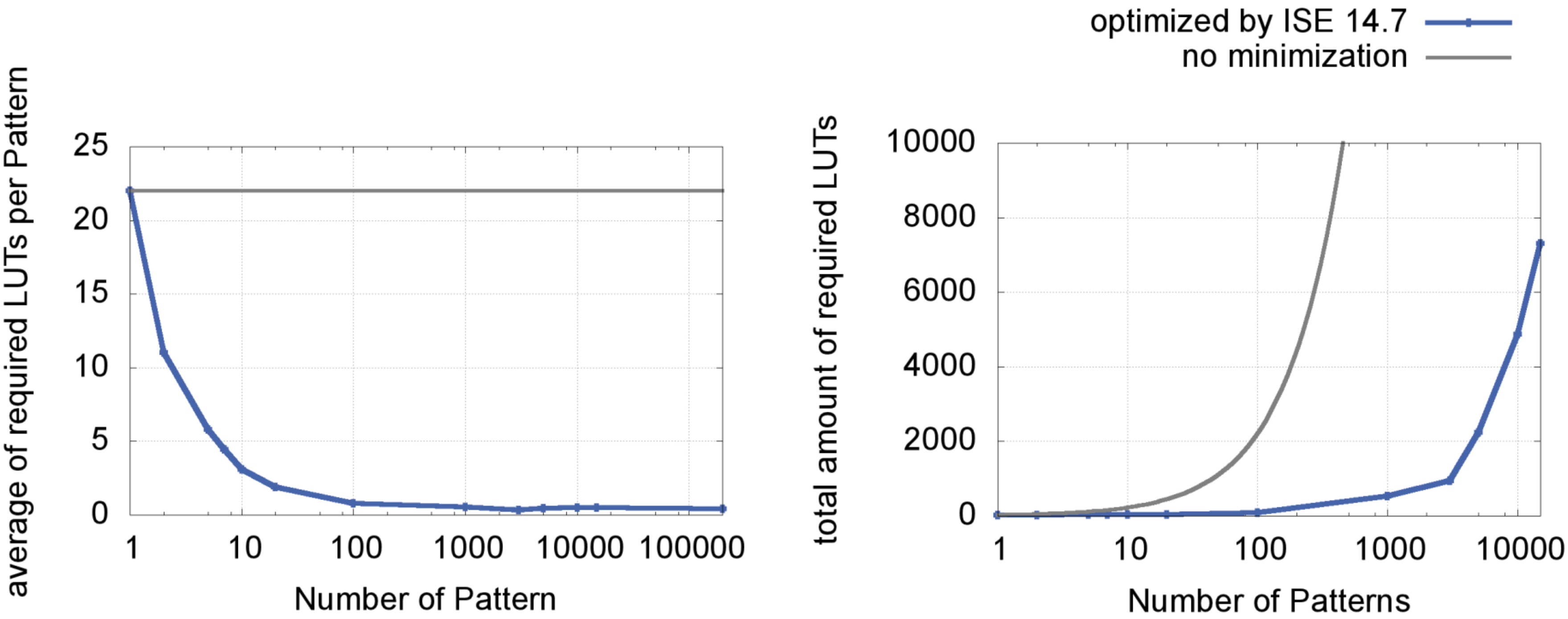
FPGA approach – minimization by logic



- 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

FPGA approach – first results

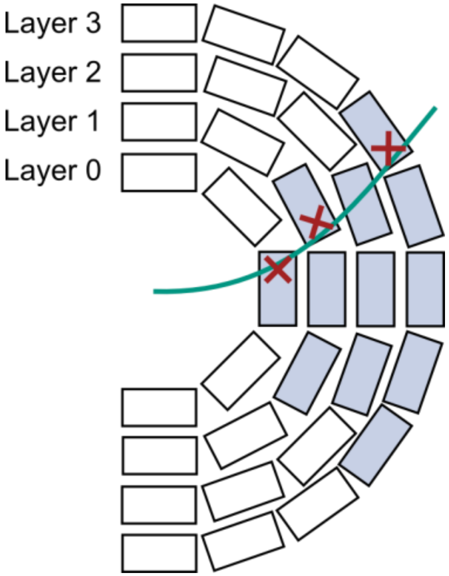


- gain saturates – 0.5 LUT per pattern in average
- depends on the composition of the pattern bank

extensive minimization by logic is possible

Comparison with AM chip

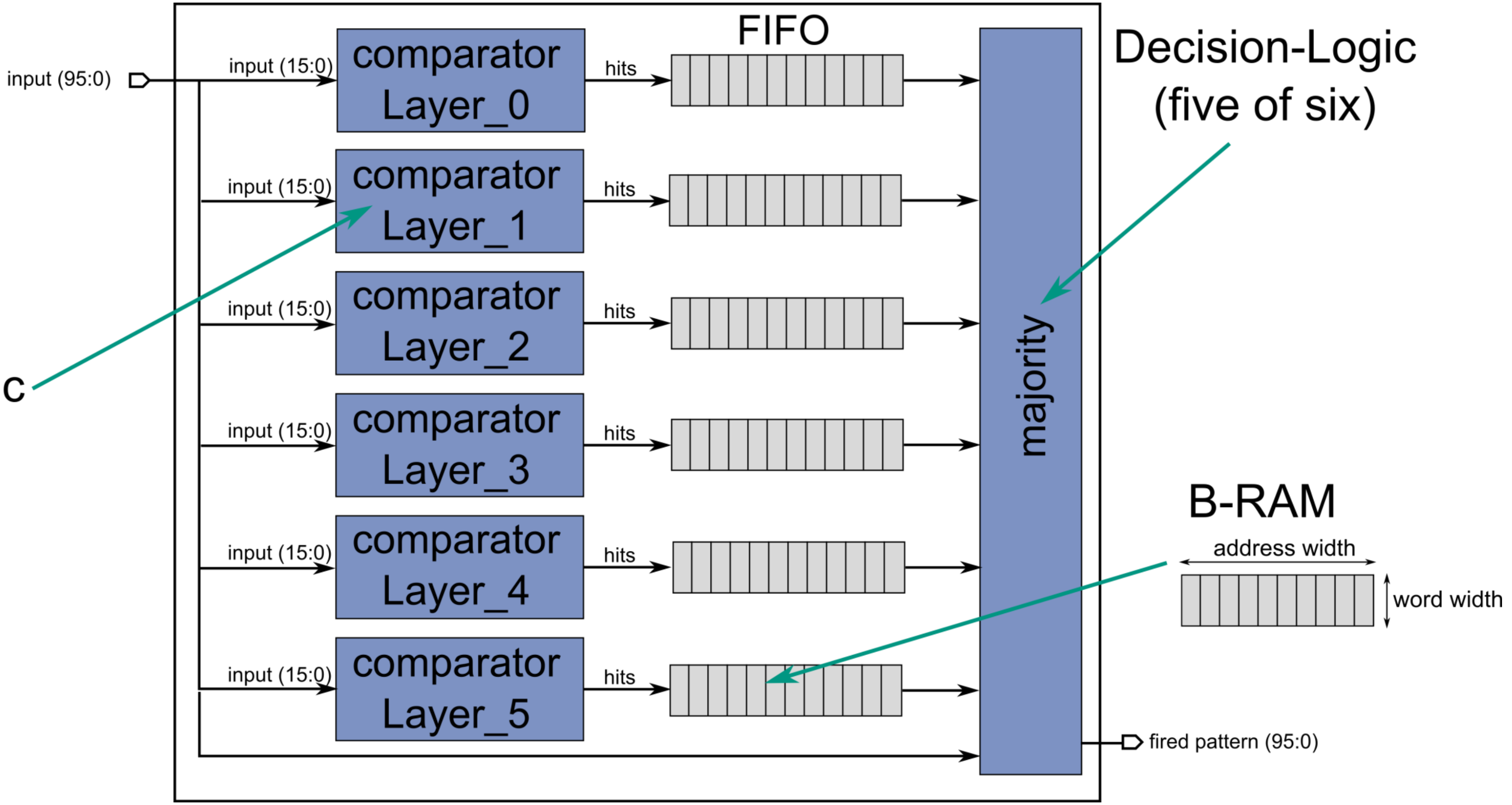
- AM chip offers additional features
- writeable memory
  - synthesize the FPGA



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

→ Layer-based approach

Layer-based approach

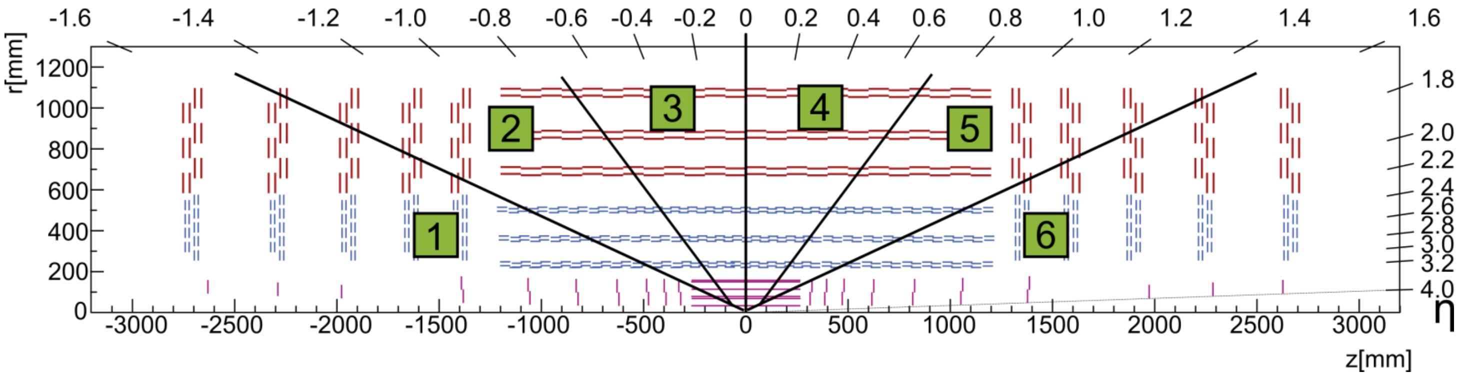


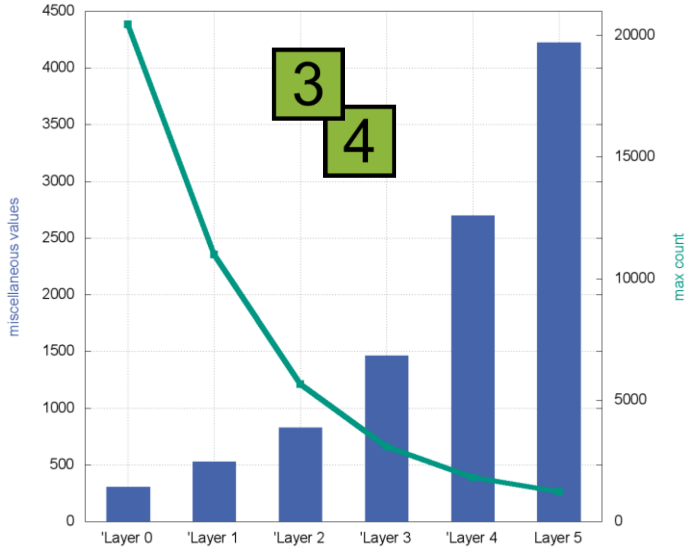
logic

Decision-Logic (five of six)

B-RAM

Analyzed pattern bank - first results

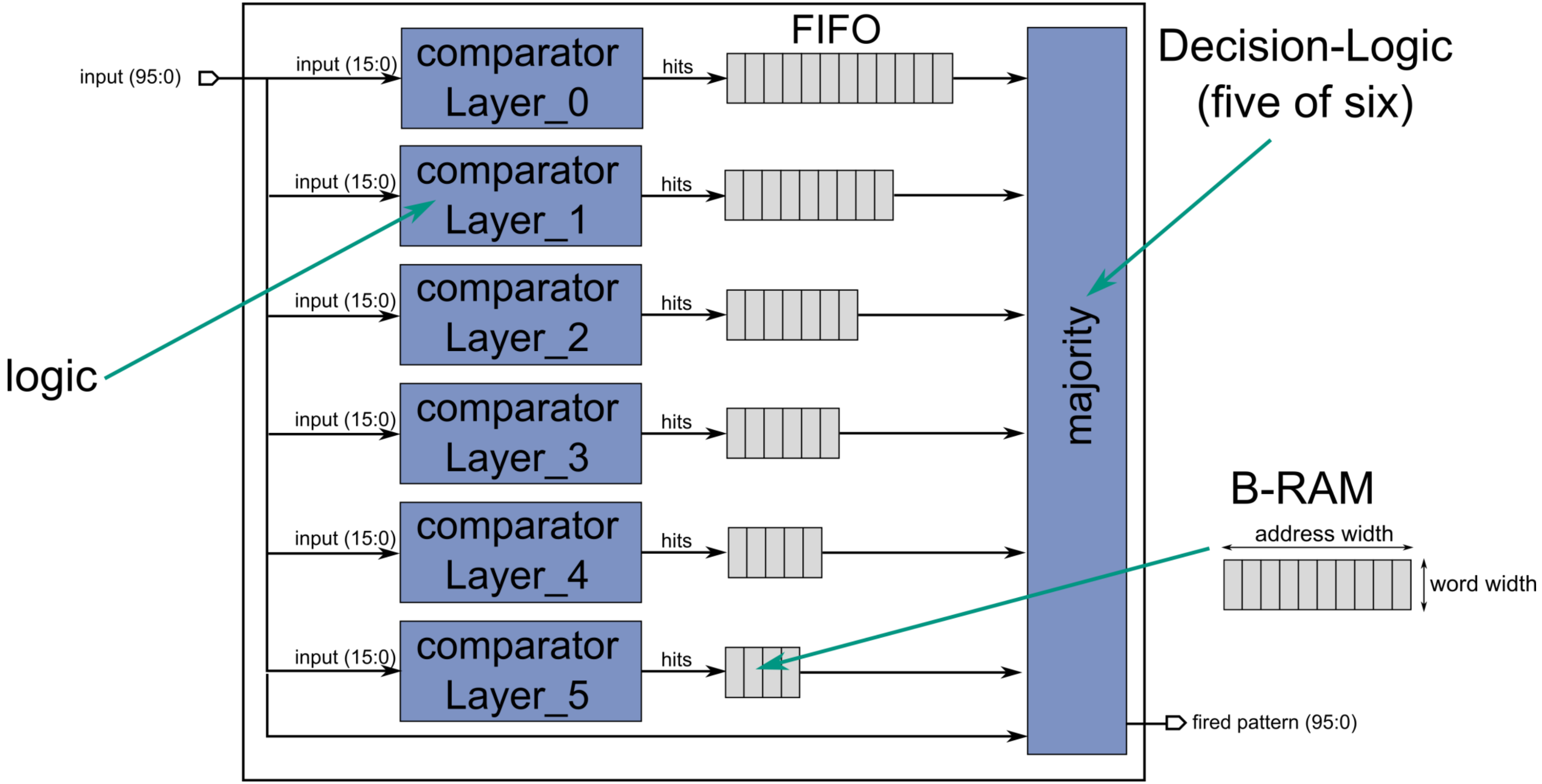




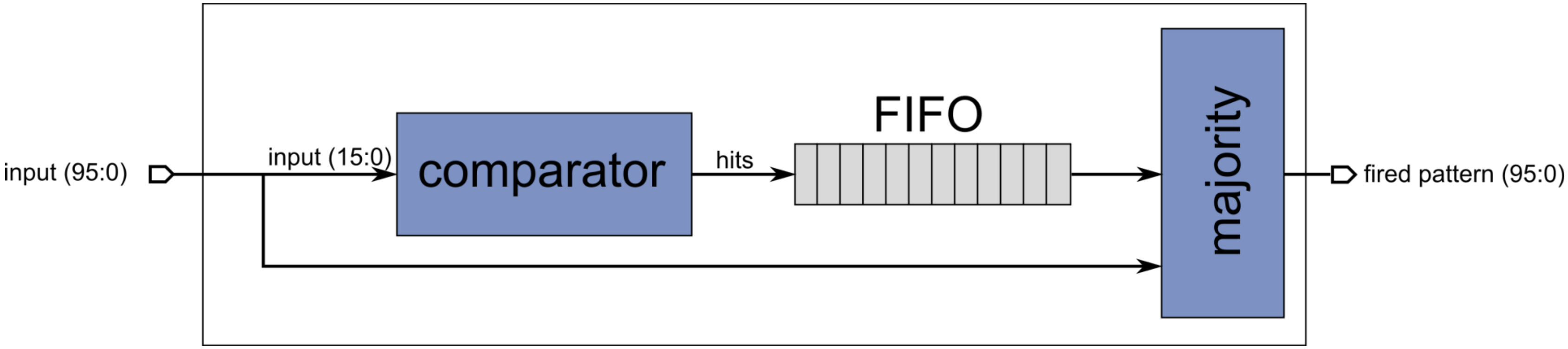
- Layer 0
  - ~300 miscellaneous values
  - >20000 hits per input possible
- Layer 5
  - >4200 miscellaneous values
  - max. 1230 hits per input

→ affects length of FIFOs

Layer-based approach

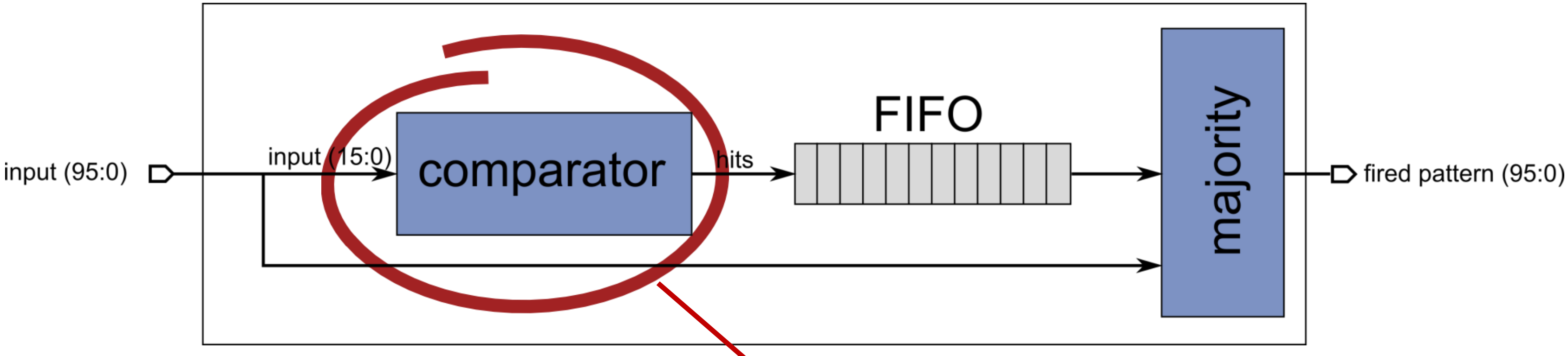


Layer-based approach – one layer



- comparator
  - input: 16 Bits
  - output: n Bits fired pattern number
- FIFO
  - buffers fired pattern
- majority matrix
  - decision (five of six, four of six, …)

Layer-based approach – comparator

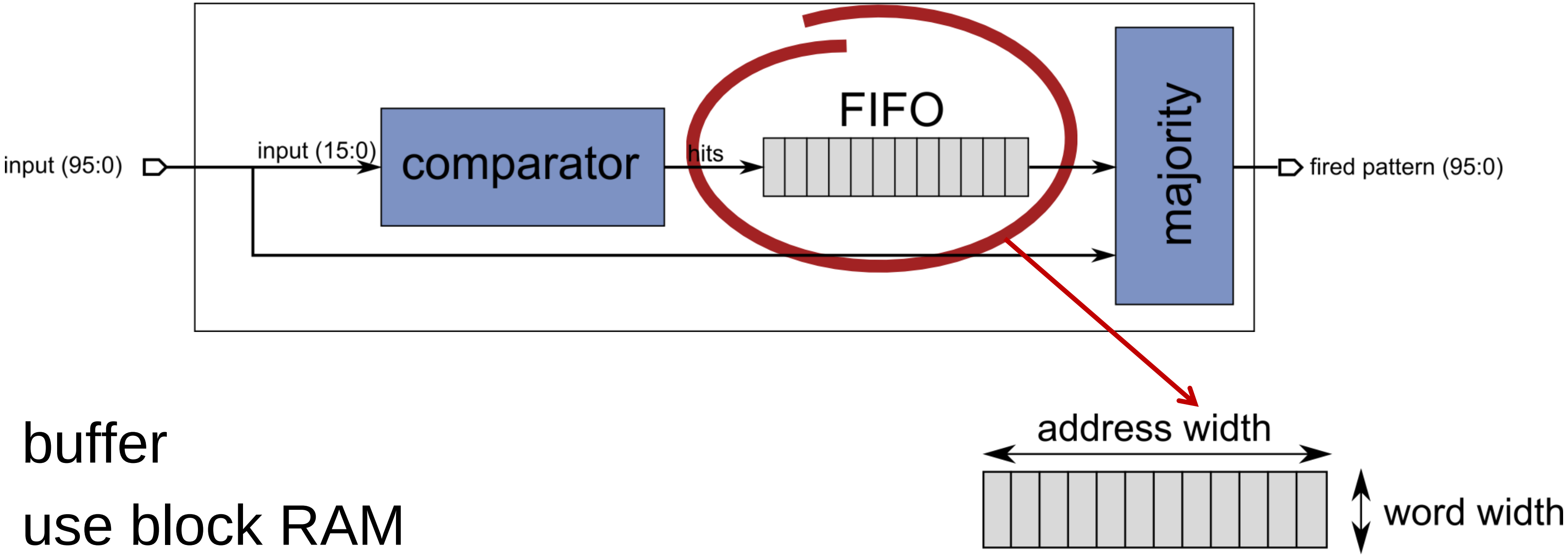


- pure logic (state machine)
    - pre-computed
    - depends on stored pattern
- use only lookup tables
- minimize by vendor tools

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

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

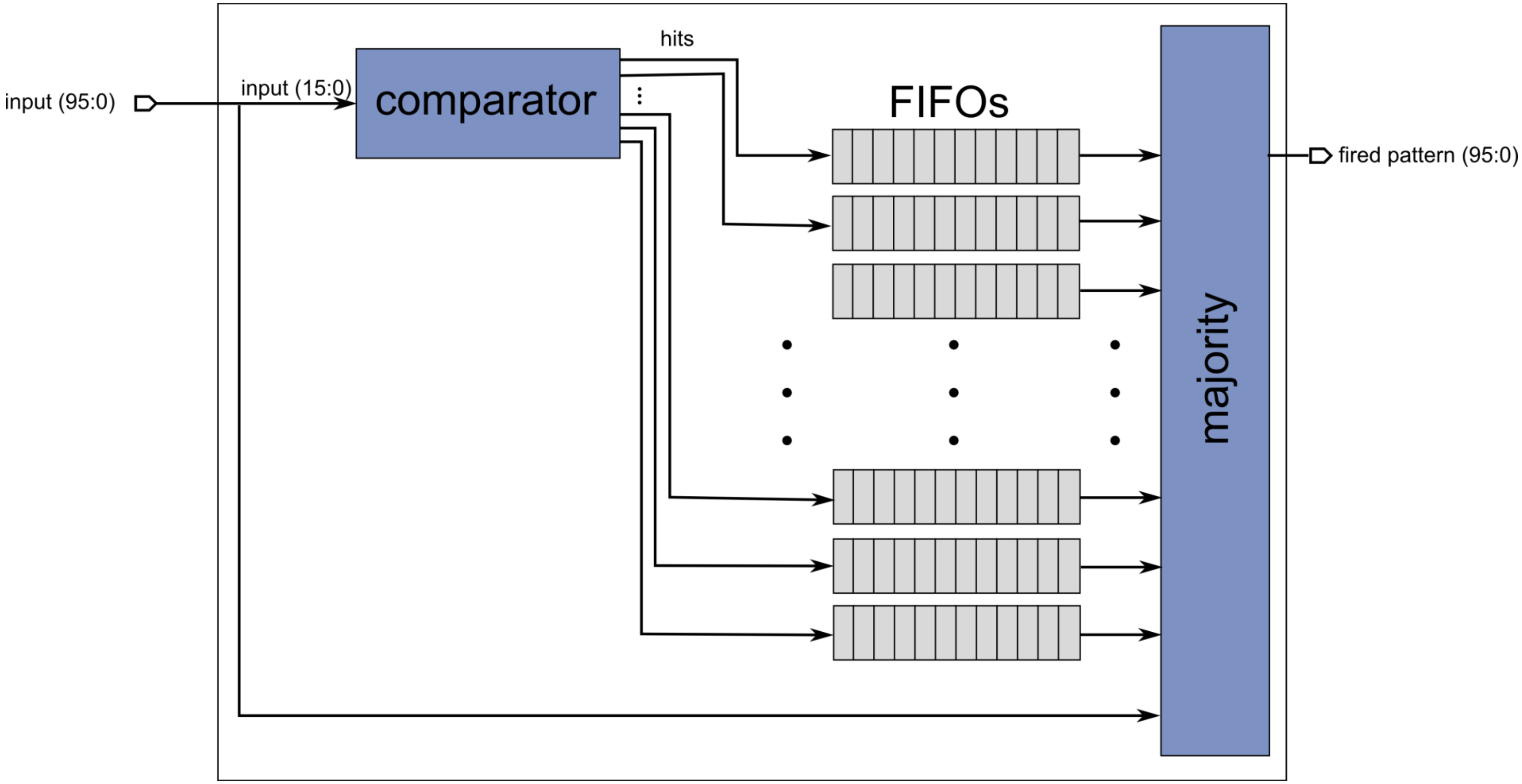
Layer based approach – FIFO



- buffer
- use block RAM
- variable size
  - address width: $\log_2$(number of max. possible hits)
  - word width: $\log_2$(number of stored patterns)

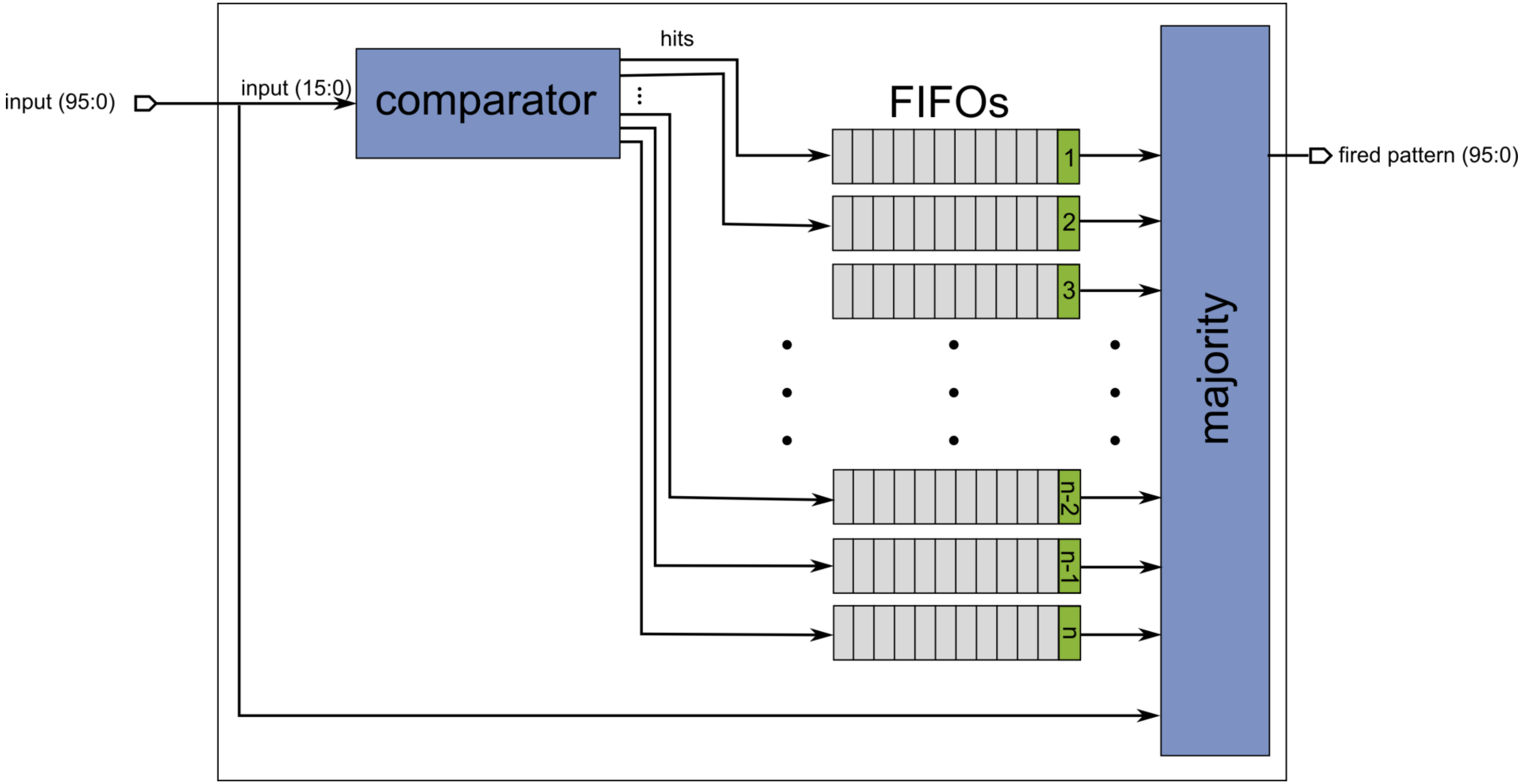
Single comparator-FIFO unit – detailed view

- one FIFO unit consists of several small FIFOs
  - due to the latency



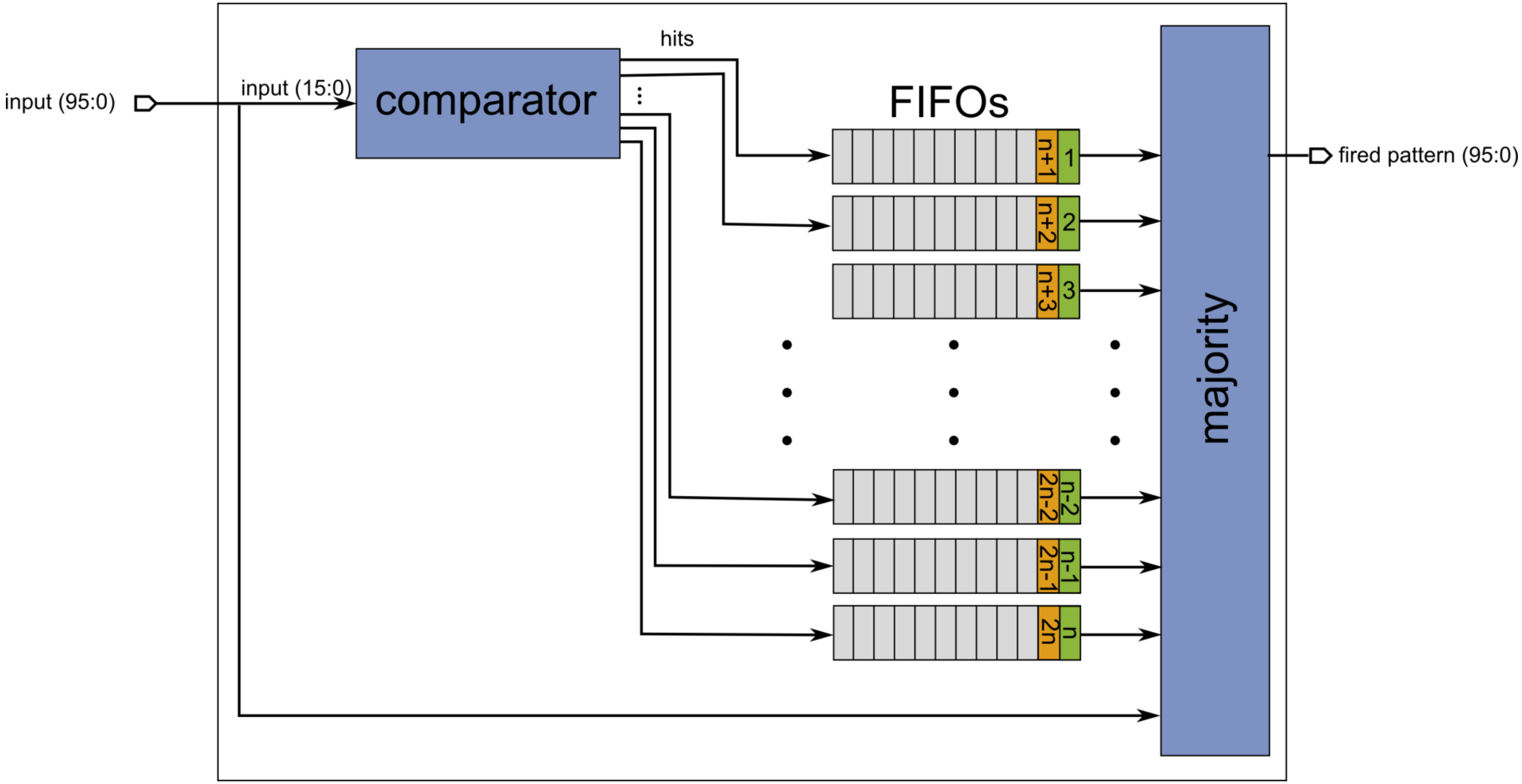
Single comparator-FIFO unit – working principle

- filling FIFOs in parallel order



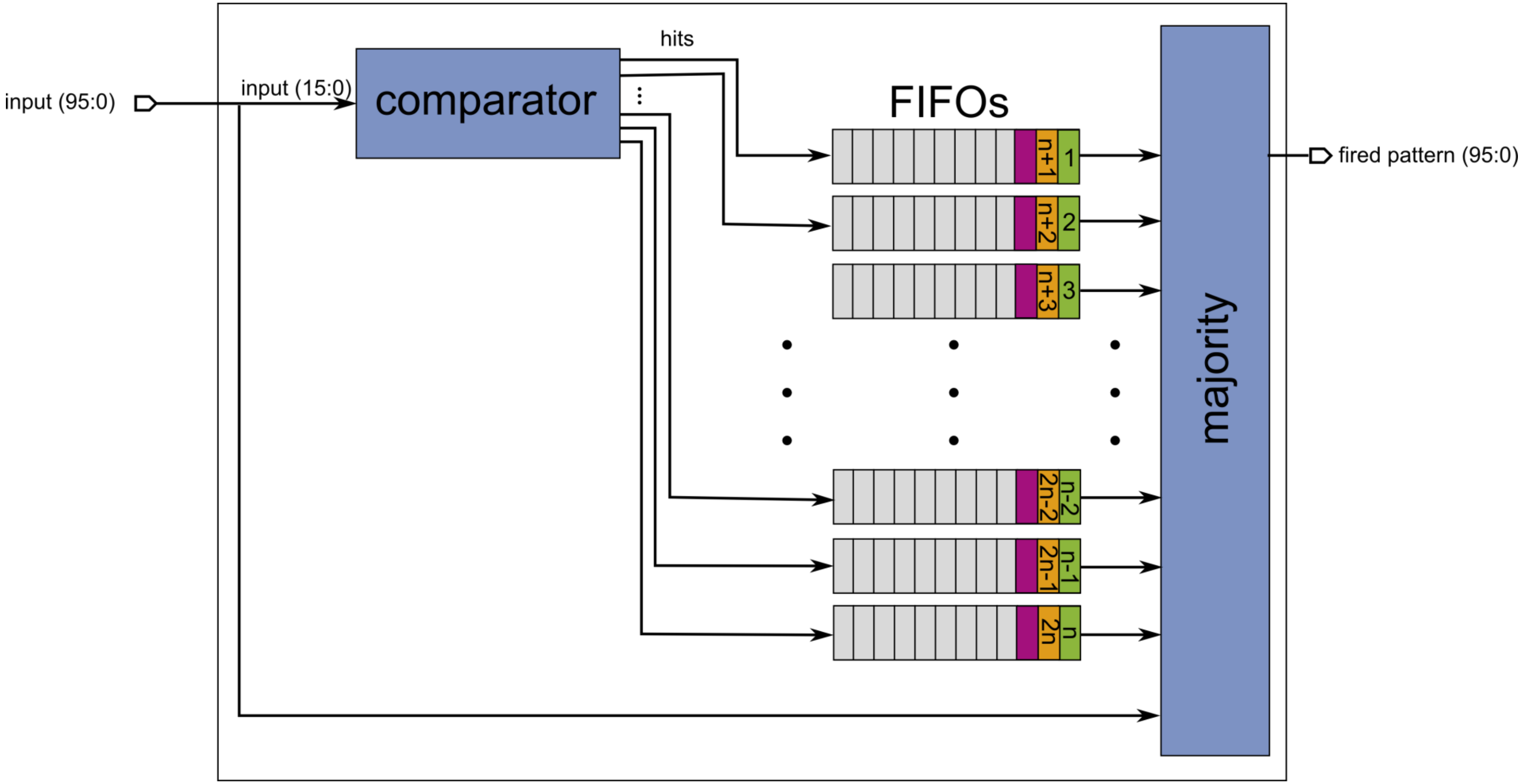
Single comparator-FIFO unit – working principle

- filling FIFOs in parallel order



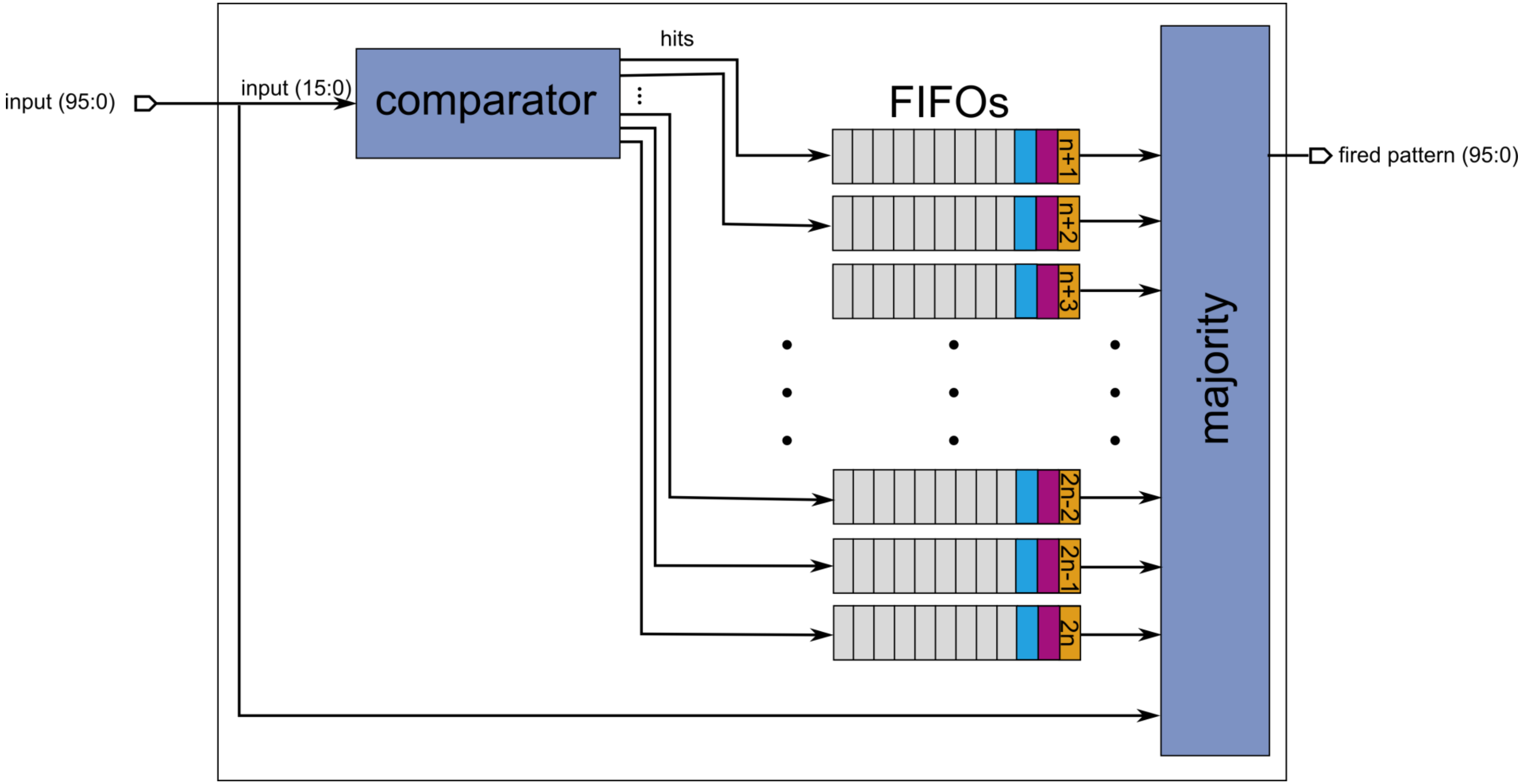
Single comparator-FIFO unit – working principle

- filling FIFOs in parallel order

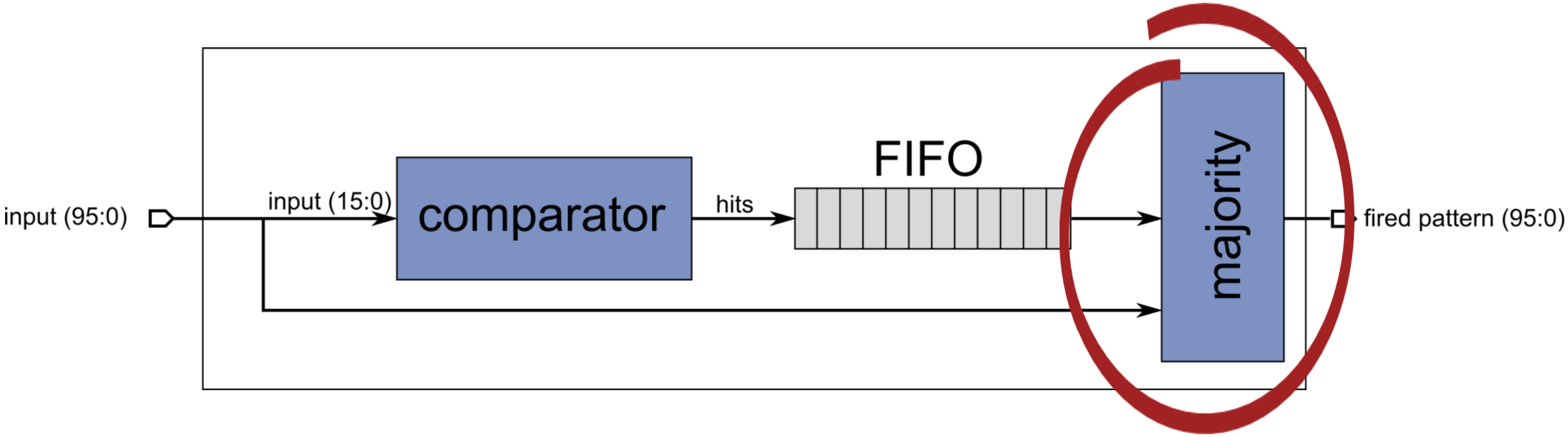


Single comparator-FIFO unit – working principle

- filling and emptying FIFOs at the same time



Layer-based approach – majority unit

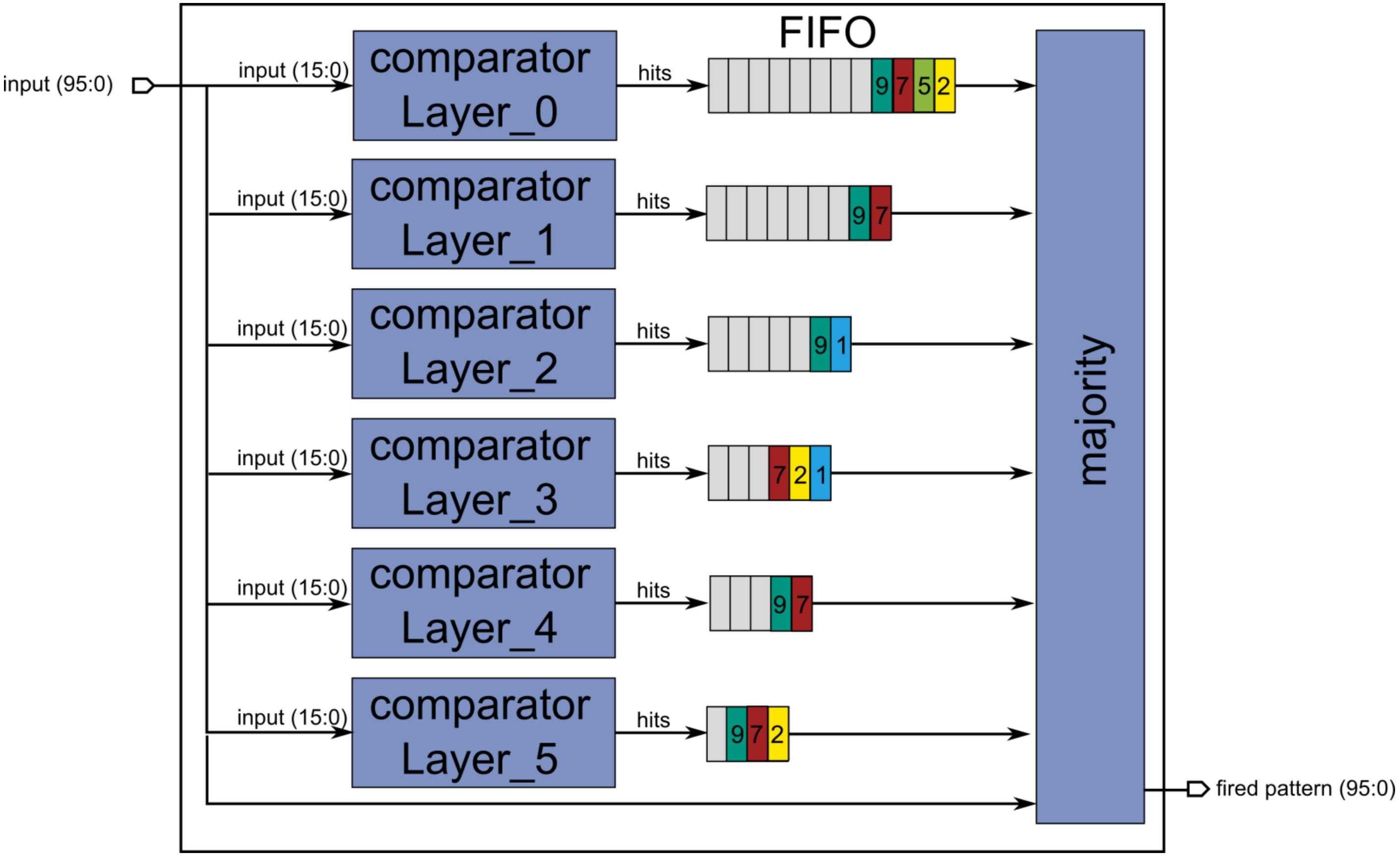


- selection of fired pattern
  - pure logic

Majority unit – working principle

- filled FIFOs: strict order of hits

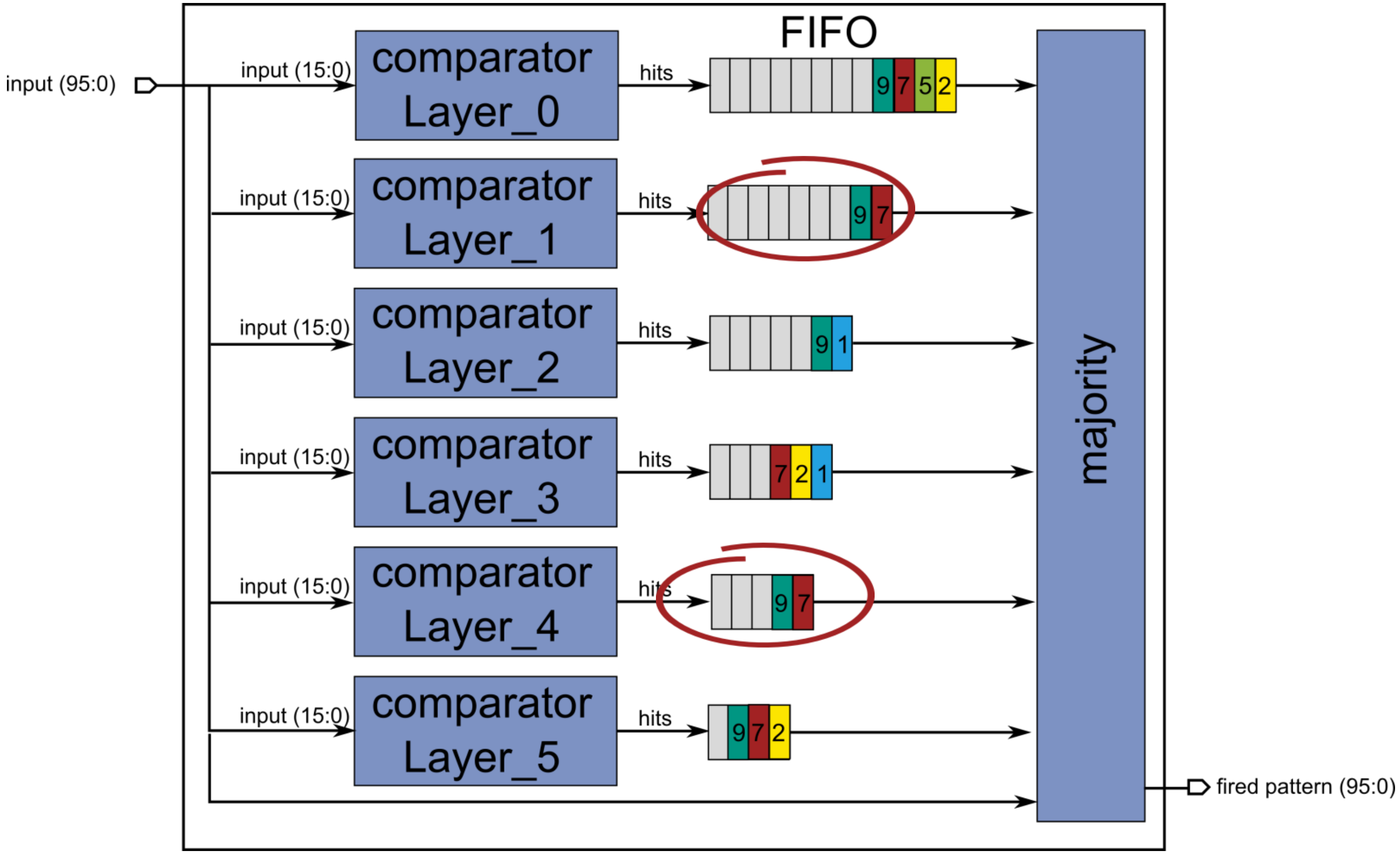
9 > 7 > 5 > 2 > 1



Majority unit – working principle

- 5/6 decision: identify two largest numbers

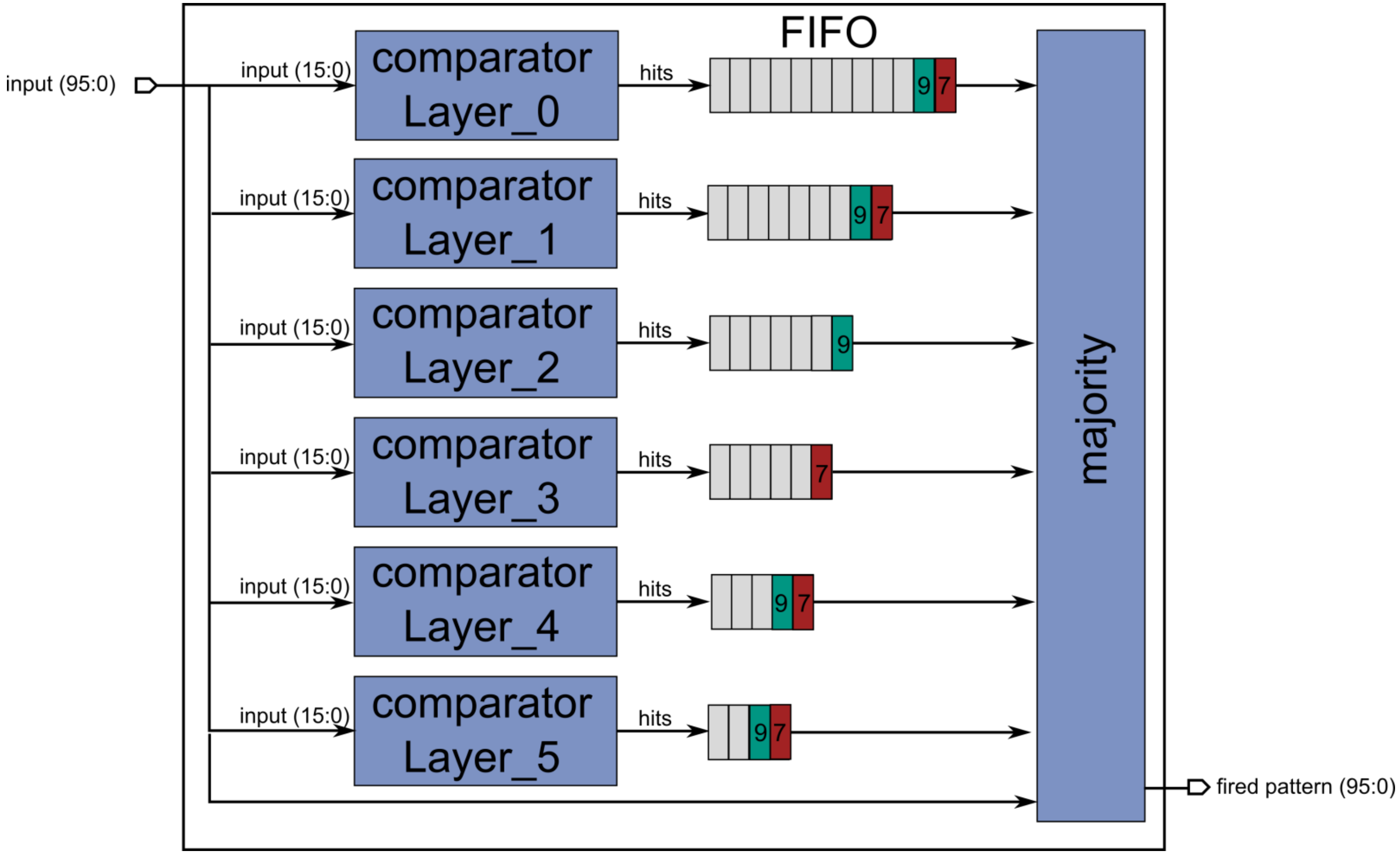
9 > 7 > 5 > 2 > 1



Majority unit – working principle

- throw away all smaller entries

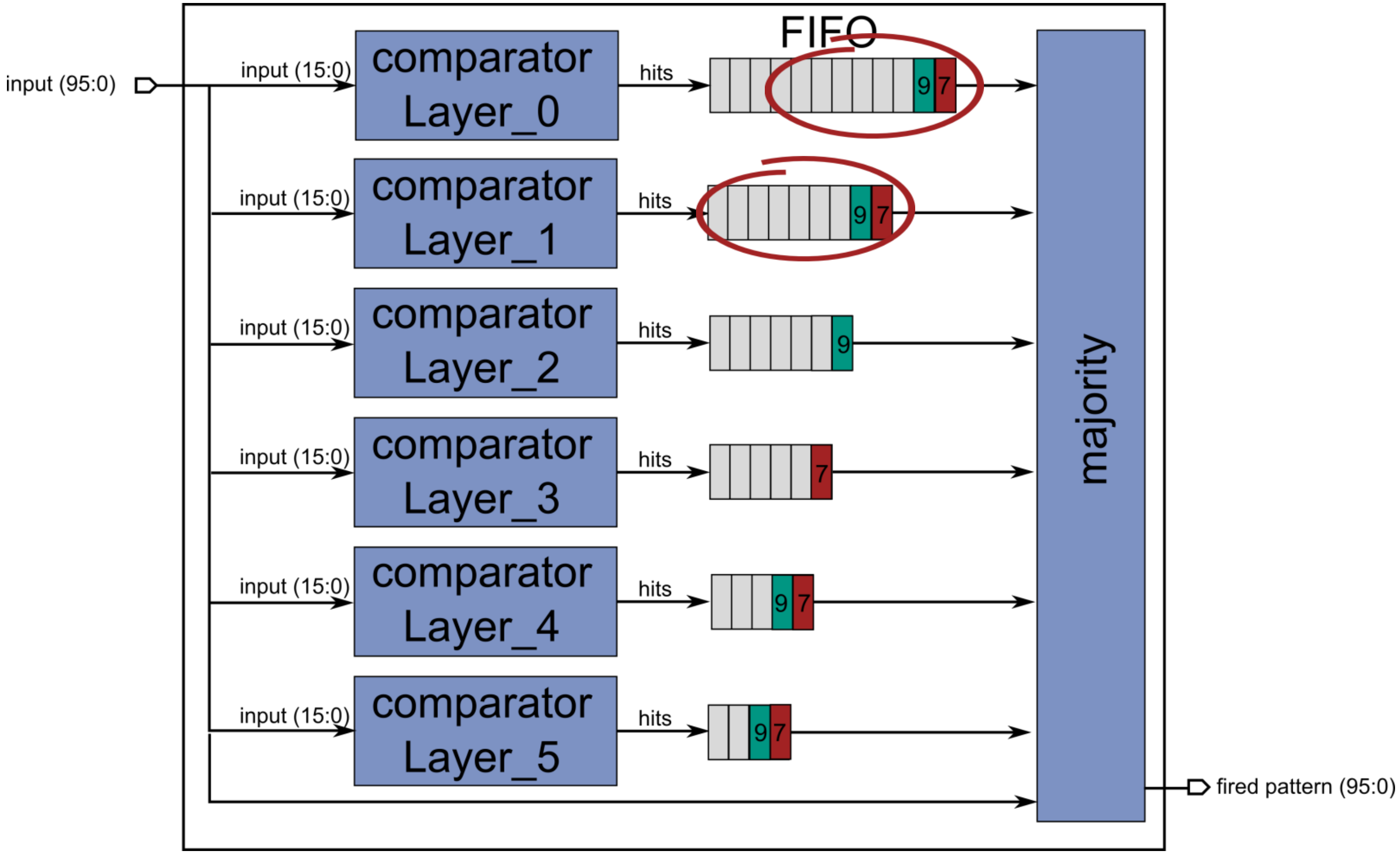
9 > 7 > 5 > 2 > 1



Majority unit – working principle

- 5/6 decision: identify two largest numbers
  - they are equal -> look at all entries

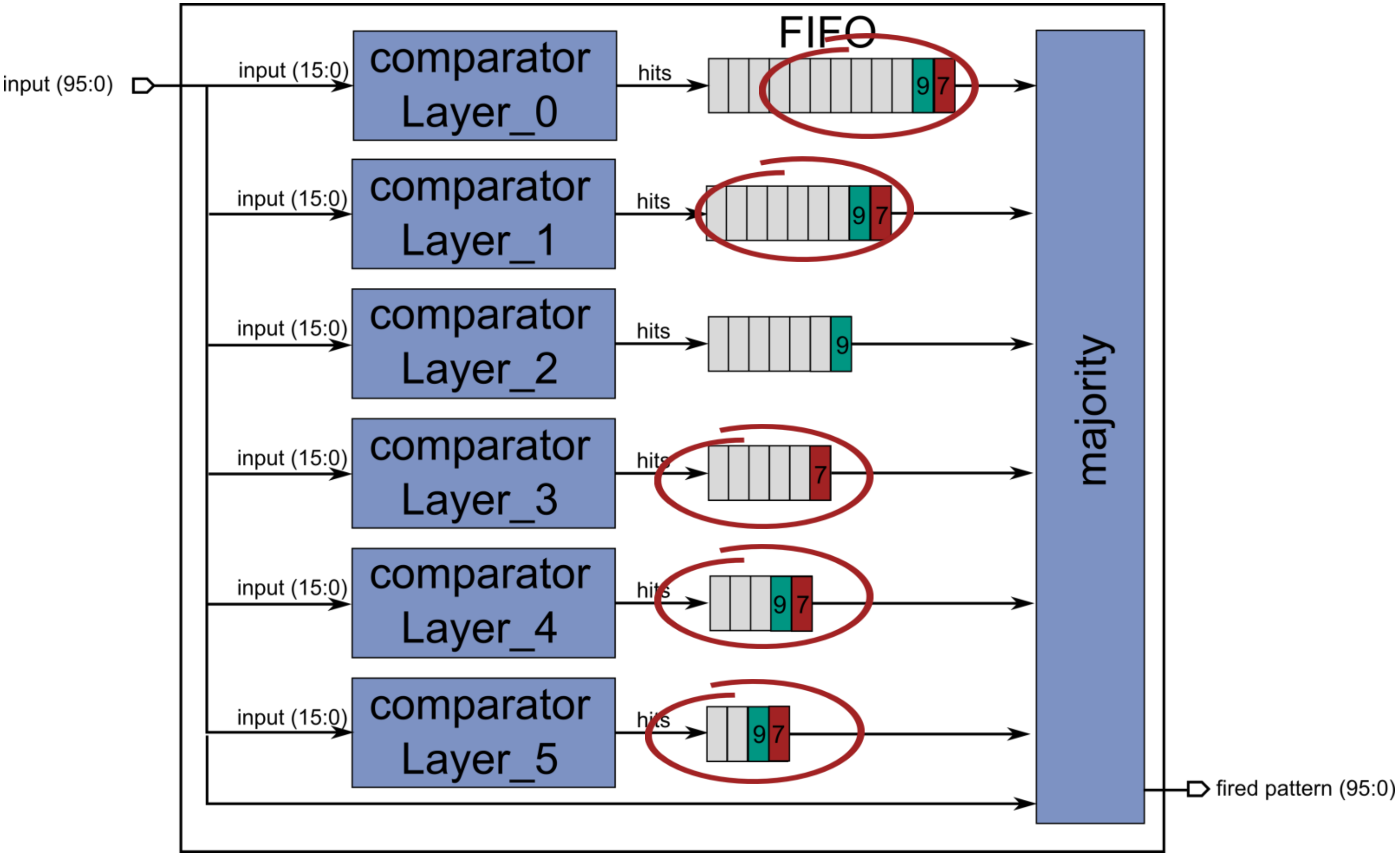
9 > 7 > 5 > 2 > 1



Majority unit – working principle

- 5/6 decision: fired pattern

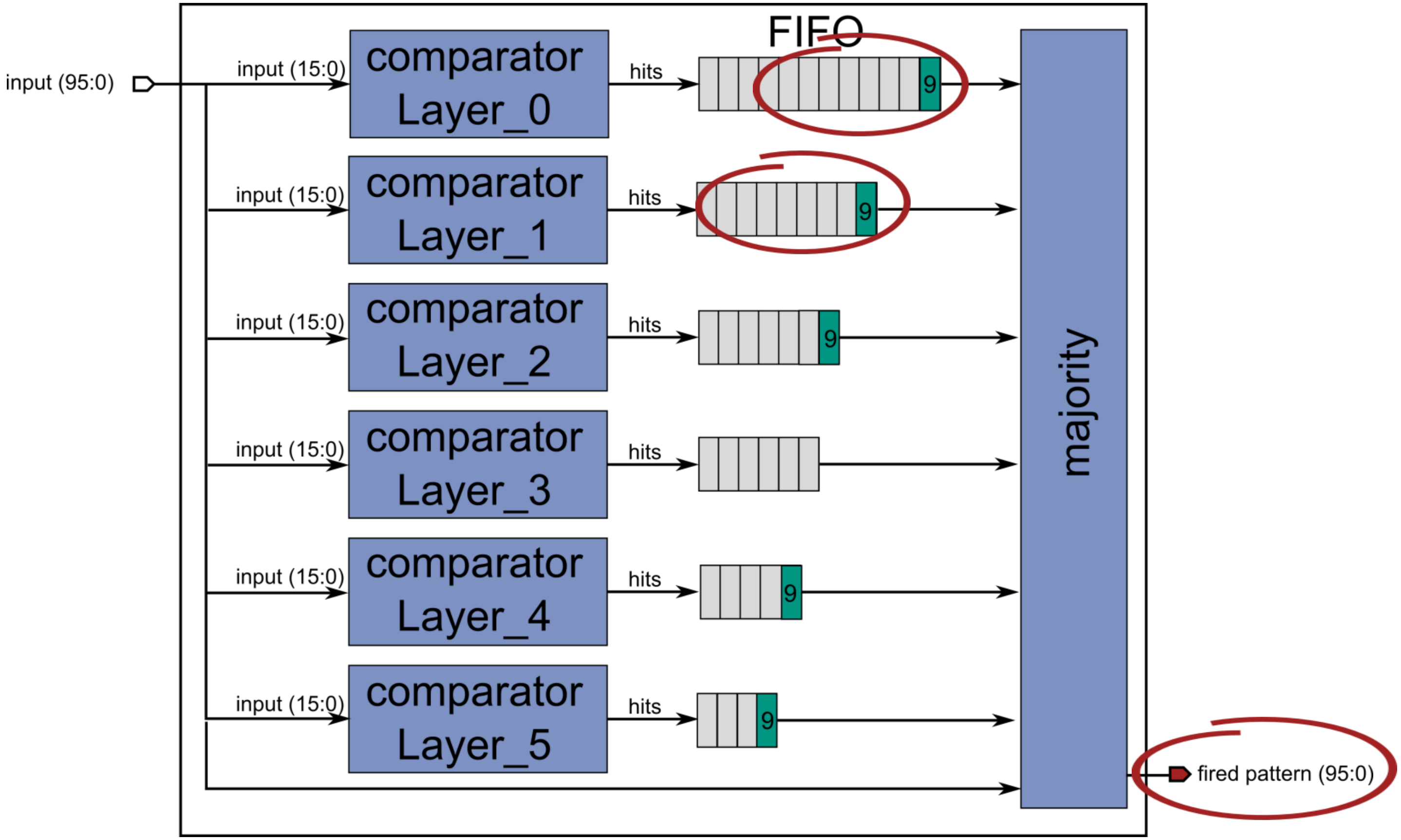
9 > 7 > 5 > 2 > 1



Majority unit – working principle

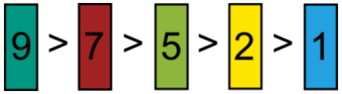
9 > 7 > 5 > 2 > 1

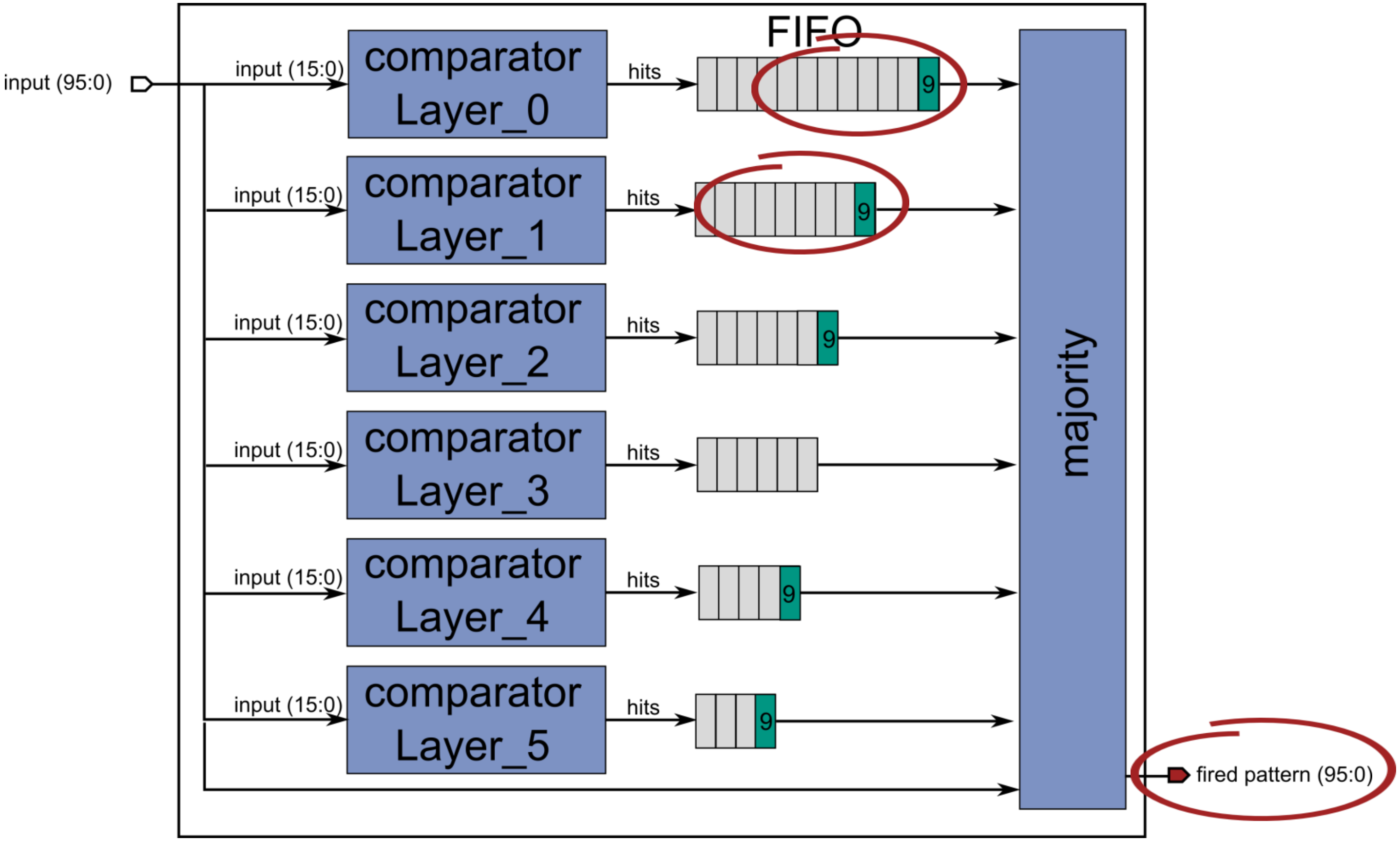
- read out fired pattern
- 5/6 decision: identify two largest numbers



Majority unit – working principle

- 5/6 decision: identify two largest numbers
  - they are equal -> look at all entries

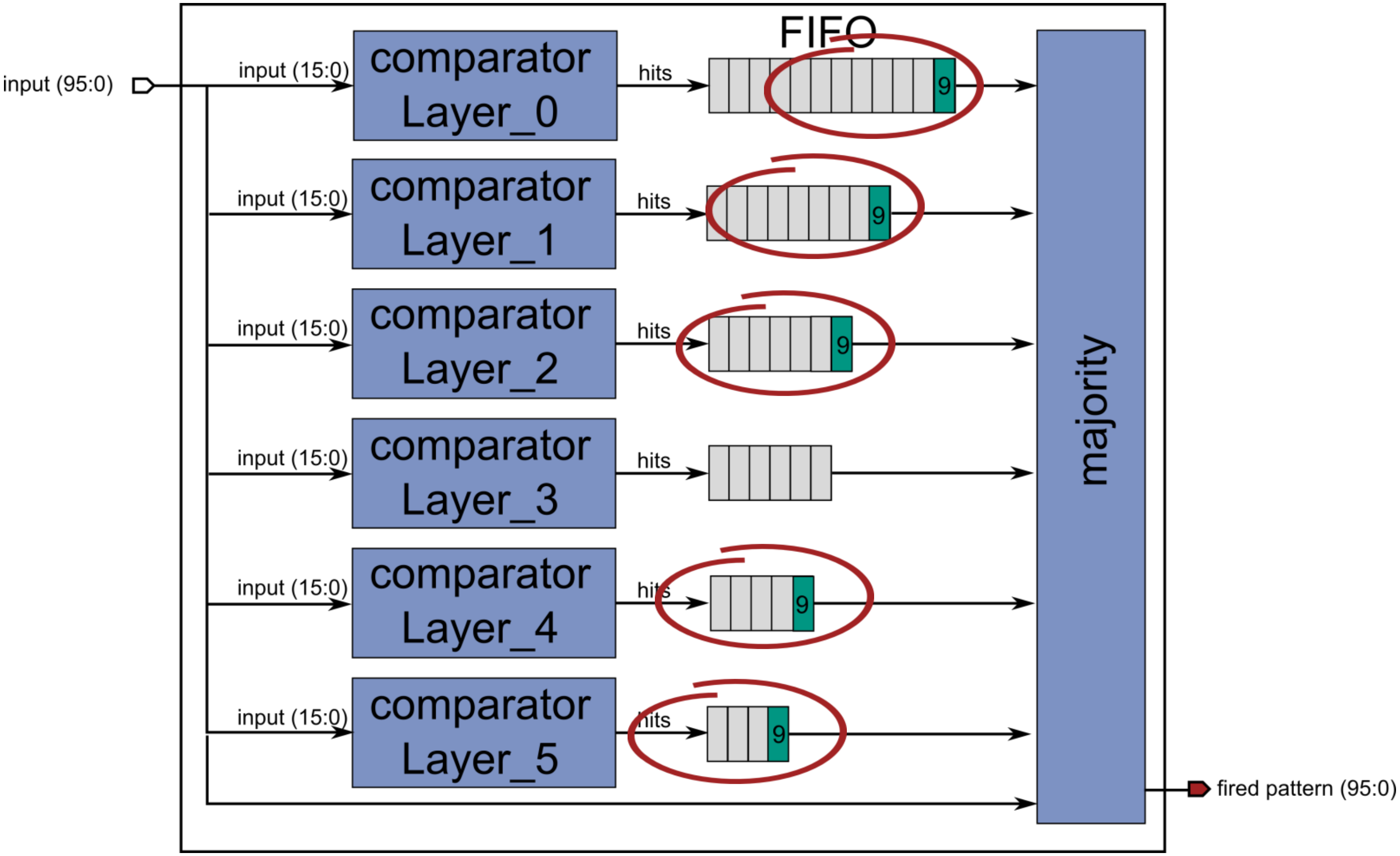




Majority unit – working principle

- 5/6 decision: fired pattern

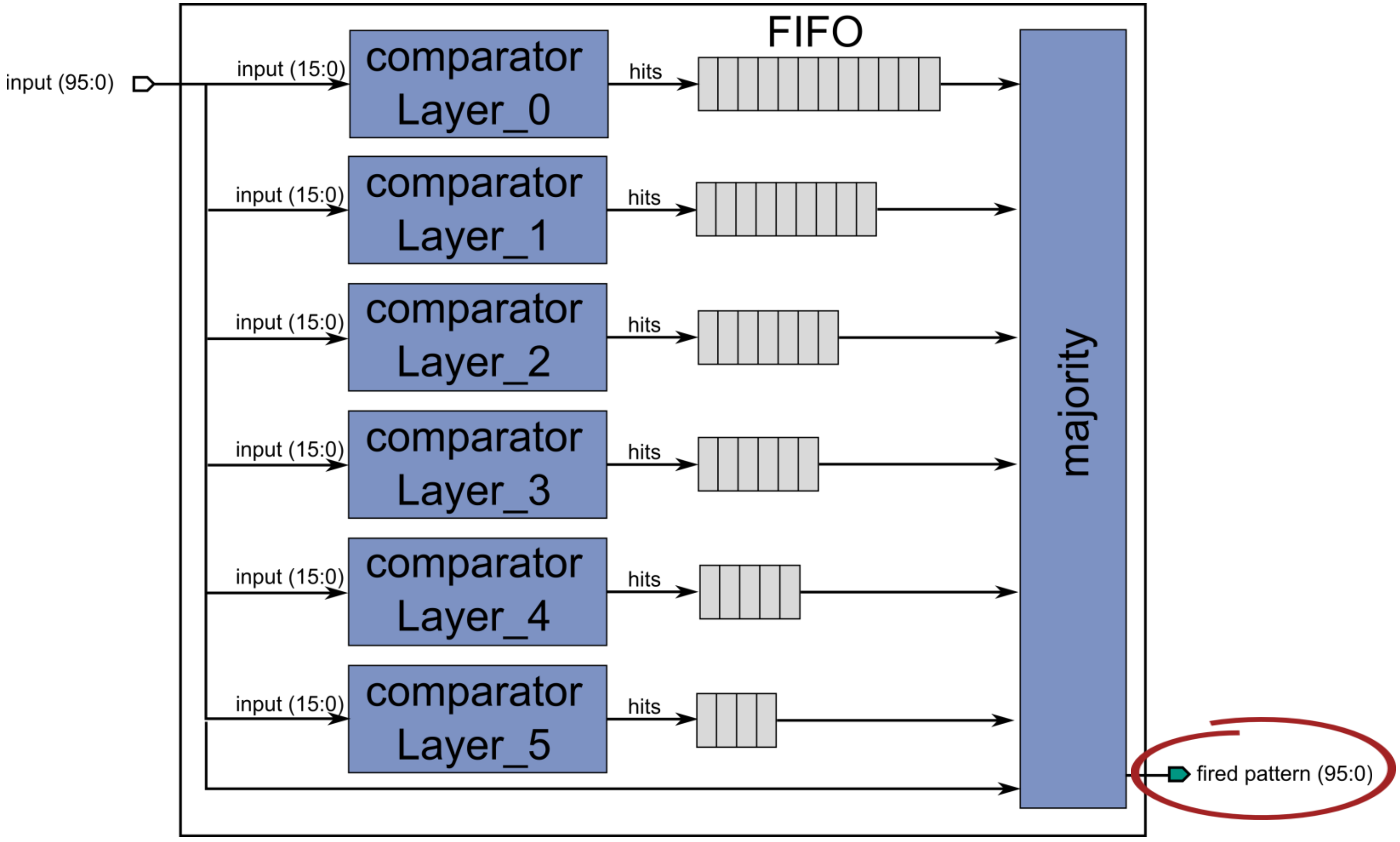
9 > 7 > 5 > 2 > 1



Majority unit – working principle

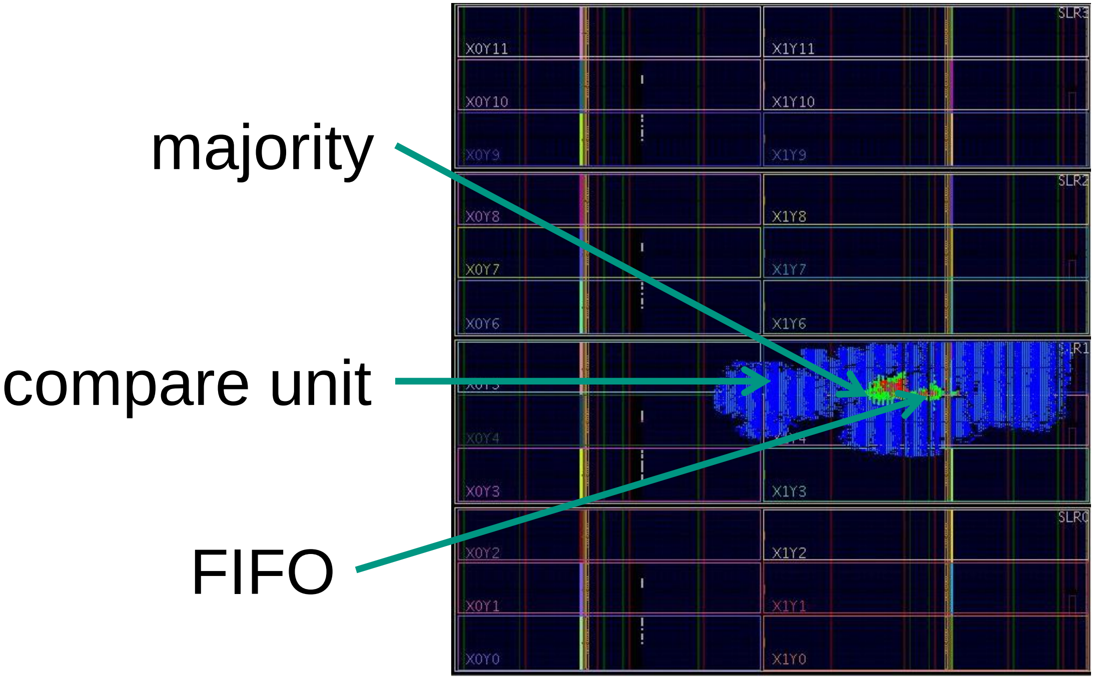
- read out fired pattern

9 > 7 > 5 > 2 > 1



Layer-based approach – first results

- design for 10k patterns is running
  - 180MHz clock cycle
  - 26 FIFOs with max. 400 entries
  - <10% Utilization of an huge up-to-date FPGA (x7v2000t)
  - output: pattern (not road number)



majority

compare unit

FIFO

Layer-based approach – first results

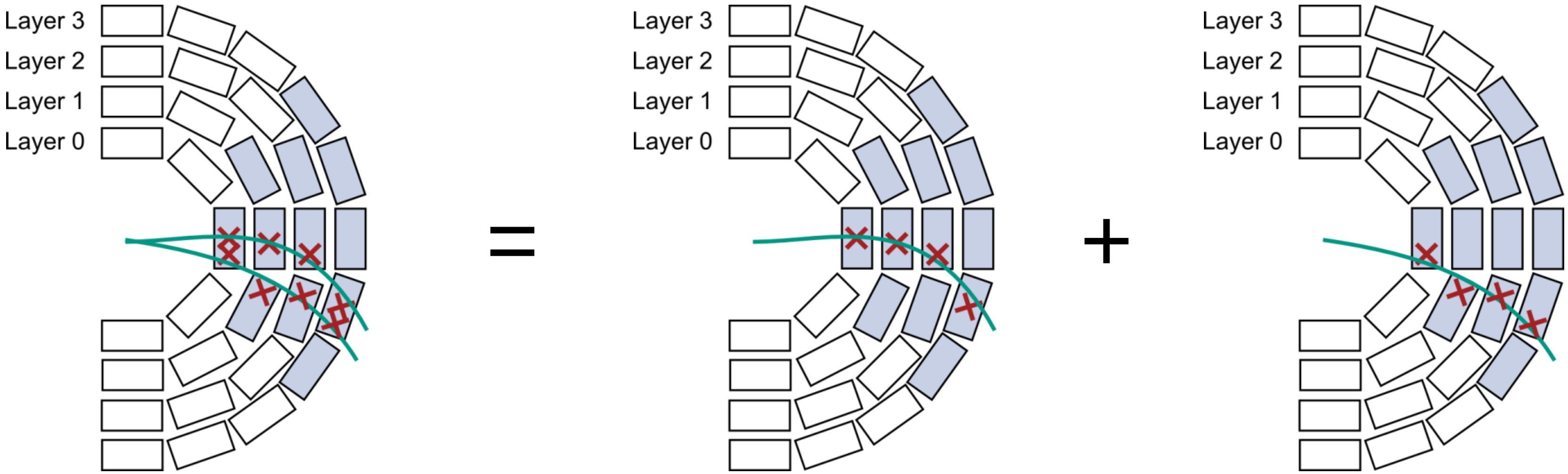
- 400 entries per FIFO

Number of pattern	Number of LUTs	LUTs per pattern	Resources of x7v2000t
10000	75000	7.5	6,2%
15000	110000	7.3	9%
25000	195000	7.8	16%

- possibility to store 150000 patterns per FPGA

Comparison with AM chip

- handle multiple hits per layer

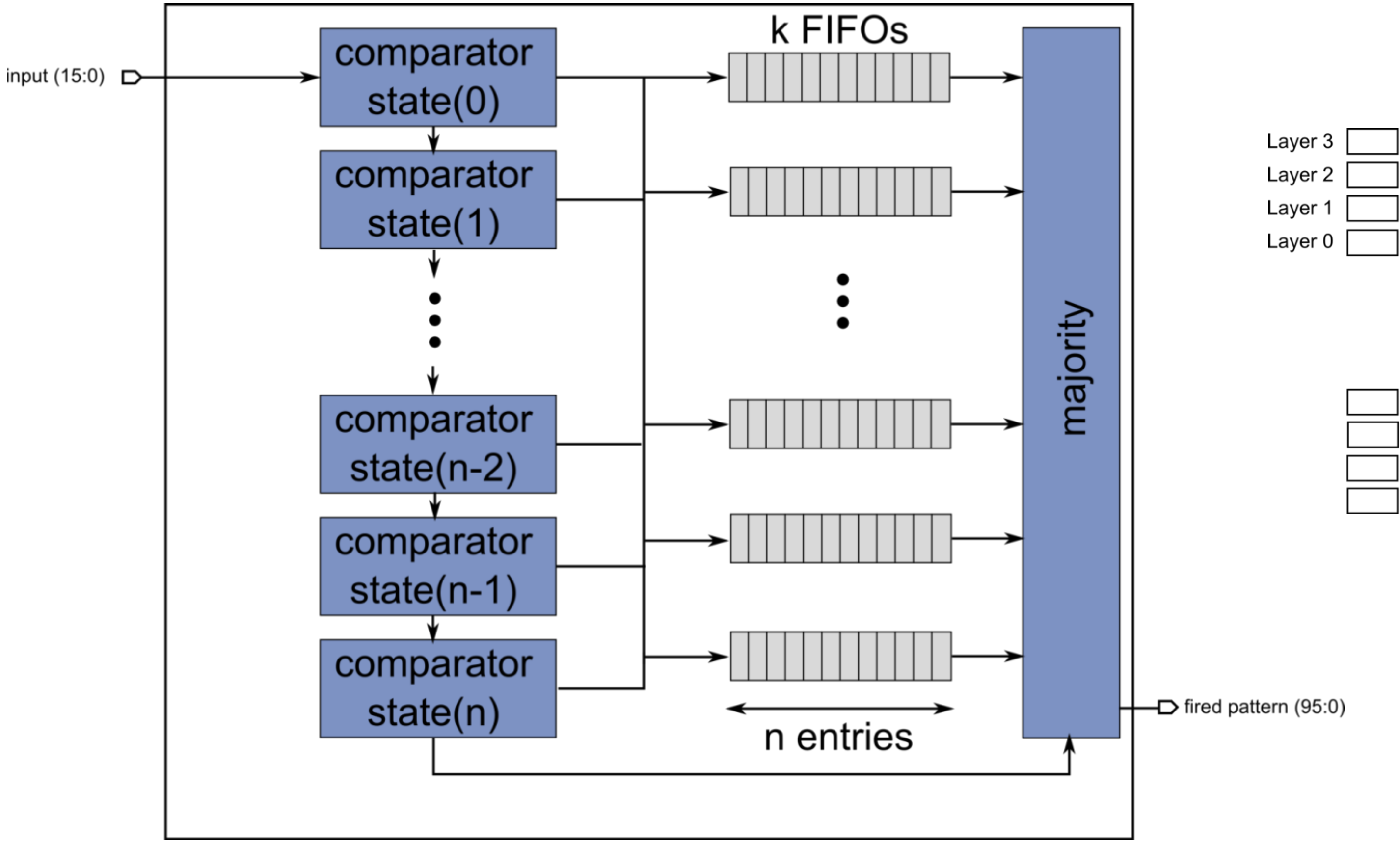


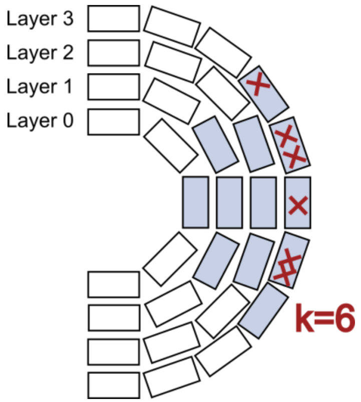
- test all combinations of hits is not practicable

→ pipelined structure of comparator unit

Pipelined structure for one layer

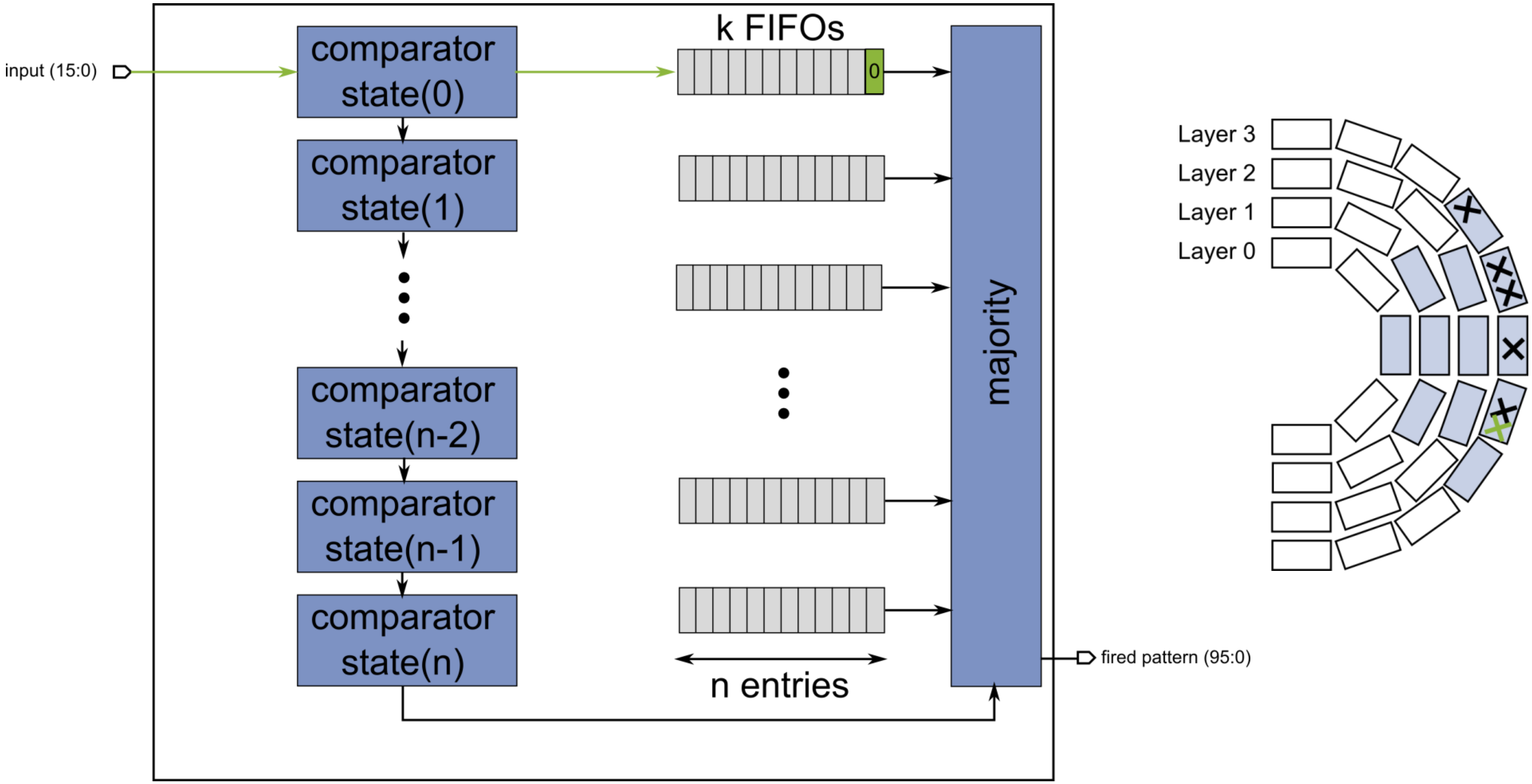
- k hits per layer
- n pattern per hit





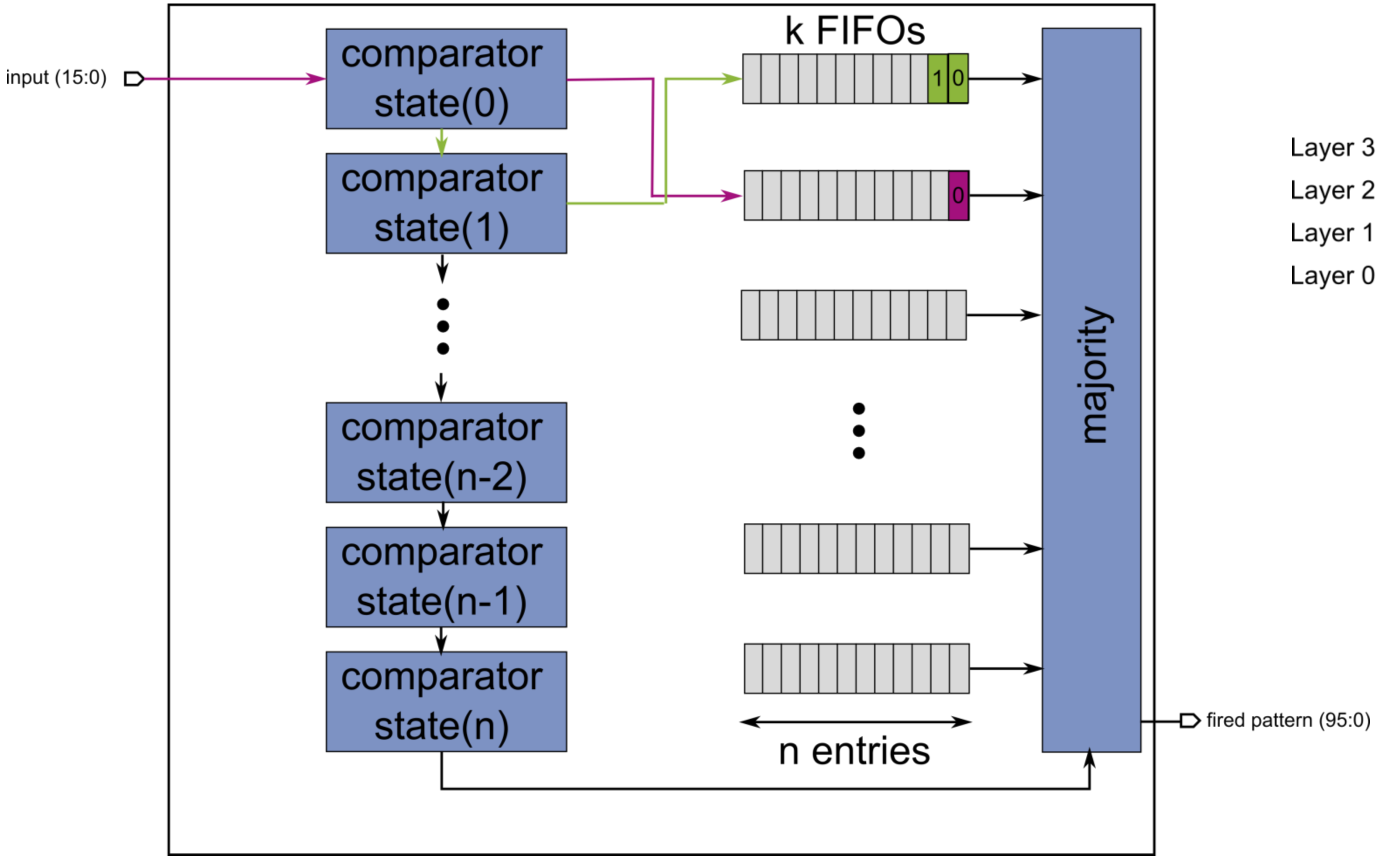
Pipelined structure – working principle

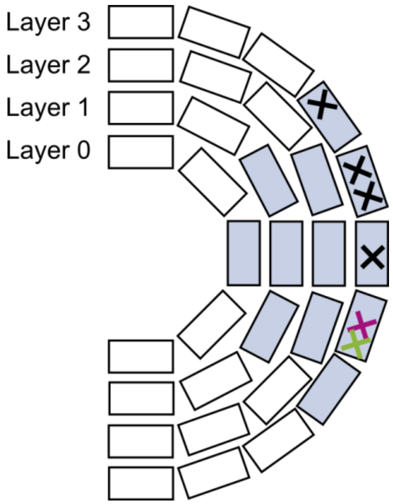
- read in first hit
  - compute first entry for hit one



Pipelined structure – working principle

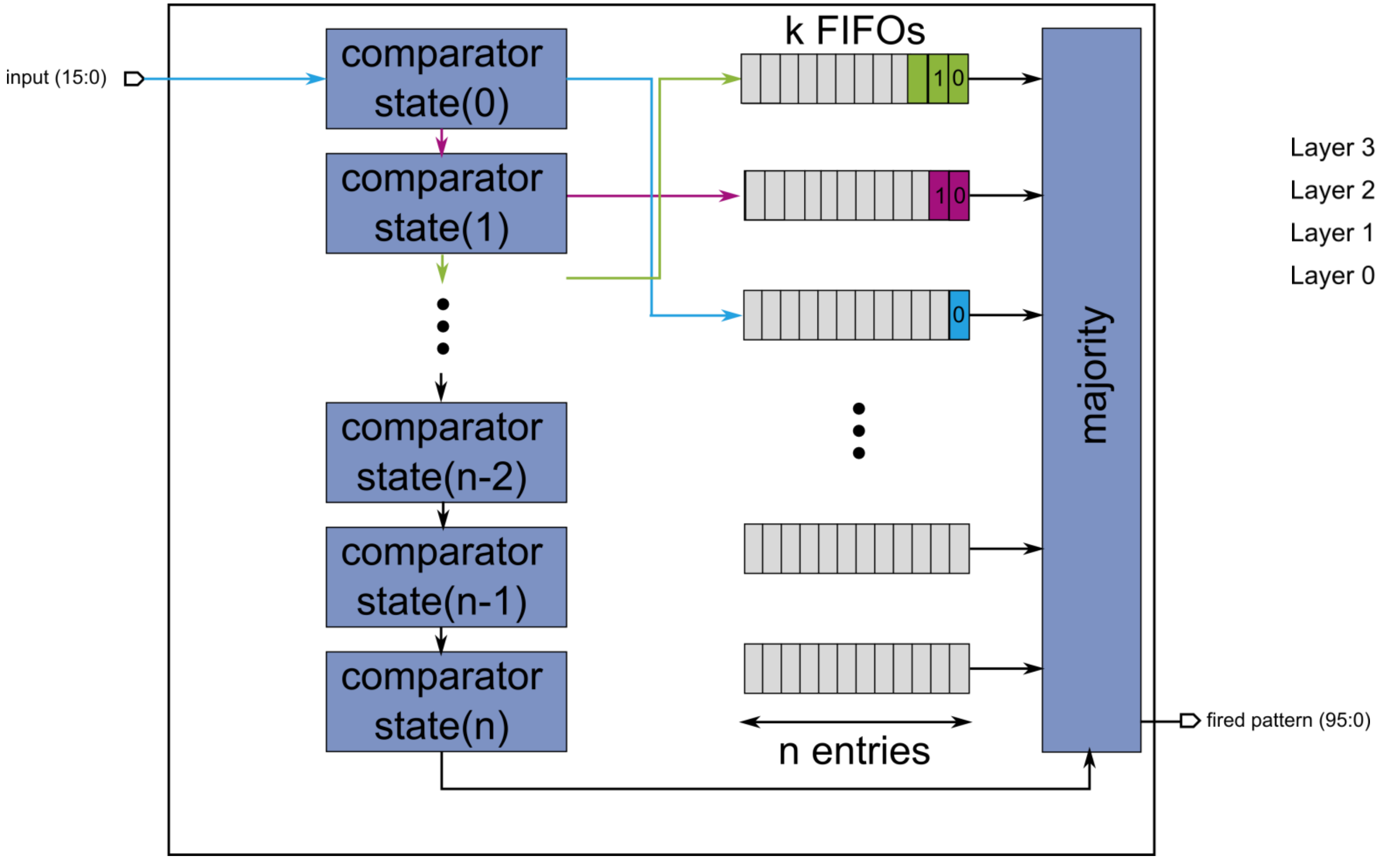
- read in second hit
  - compute first entry for hit two and second entry for hit one

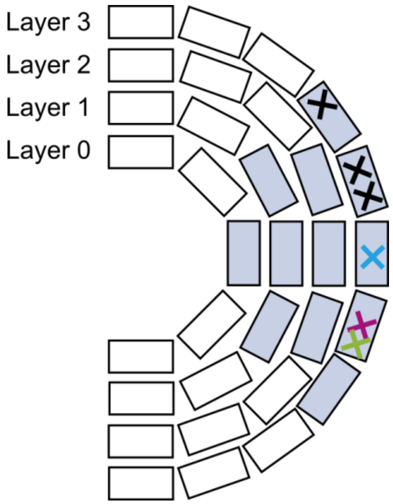




Pipelined structure – working principle

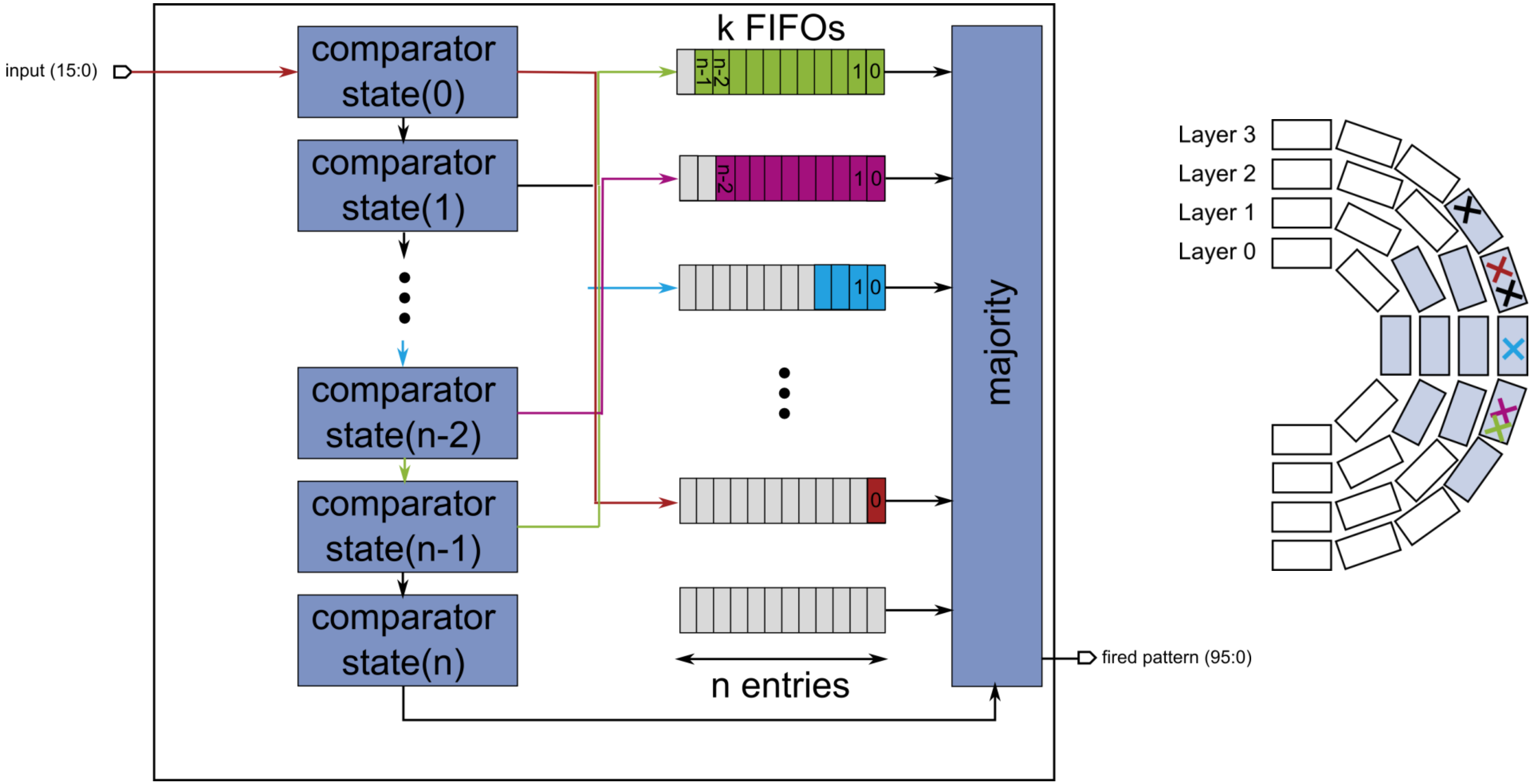
- read in third hit
  - compute: hit3-entry1, hit2-entry2, hit1-entry3





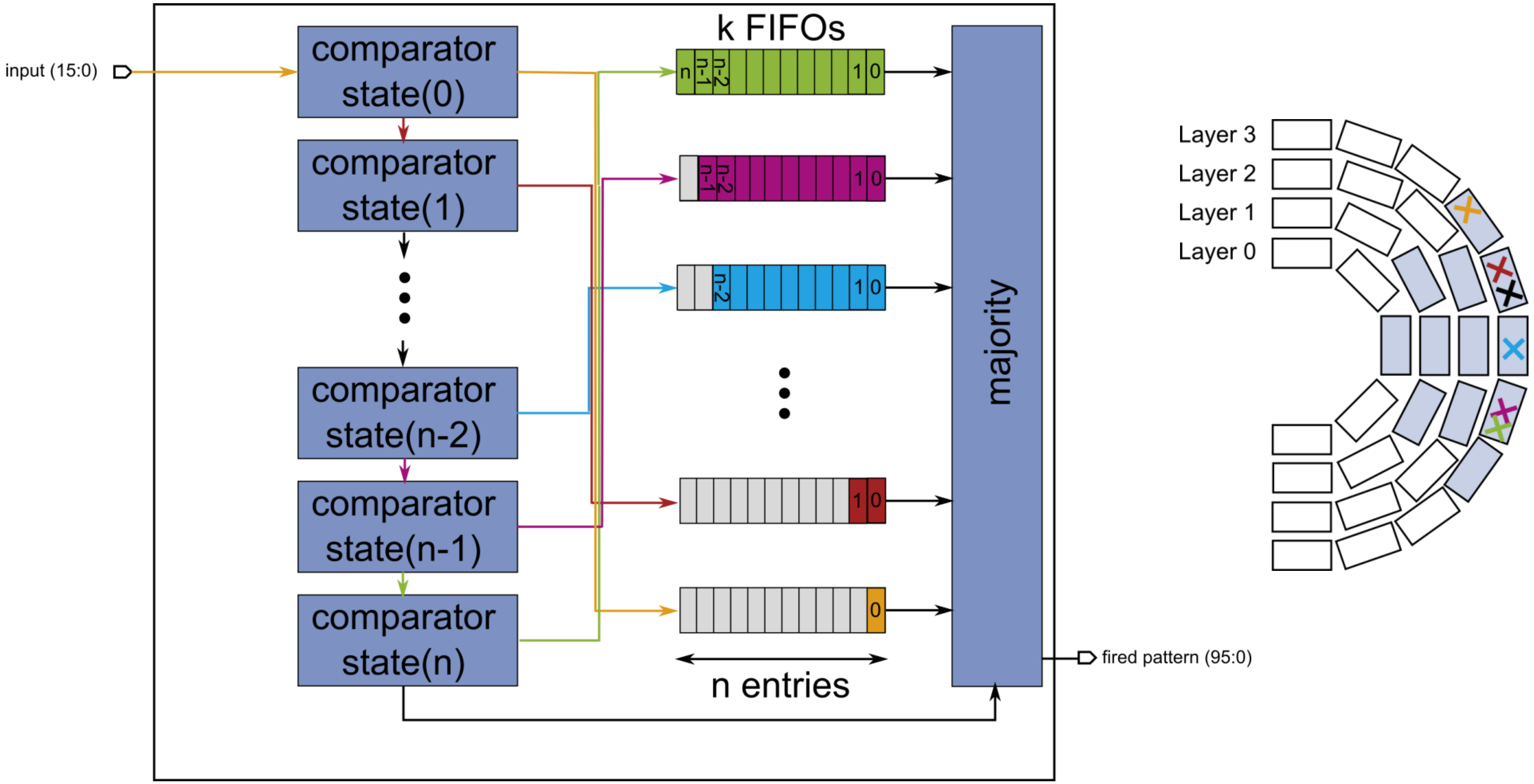
Pipelined structure – working principle

- read in hit(n-1)
  - compute: hit(n-1)-entry1, … , hit1-entry(n-1)



Pipelined structure – working principle

- read in hit(n)
    - compute: hit(n)-entry1, … , hit1-entry(n)



Summary of pipelined architecture

- first FIFO is filled after n cycles
  - n=number of entries per FIFO
- last FIFO is filled after (n+k) cycles
  - k=number of hits per layer
- after n cycles the majority unit starts computing
  - Worst case: majority unit idle k cycles
- computation ends after (2n+k) cycles
  - (n+k) cycles read in
  - n cycles read out

Conclusion

- first design is running
- FPGA implementation looks still feasible

- next steps
  - implementation of pipelined structure
  - improvement of running design
    - analysis of reordering Pattern Bank
    - define arrays of fired pattern numbers

- Ideas and remarks are welcome: harbaum@kit.edu