Trigger Systems

Reconfigurable Computing in Experimental Physics

Andrea Triossi

October 2009, CHR working week, University of Ulm

Outlines

Trigger systems overview

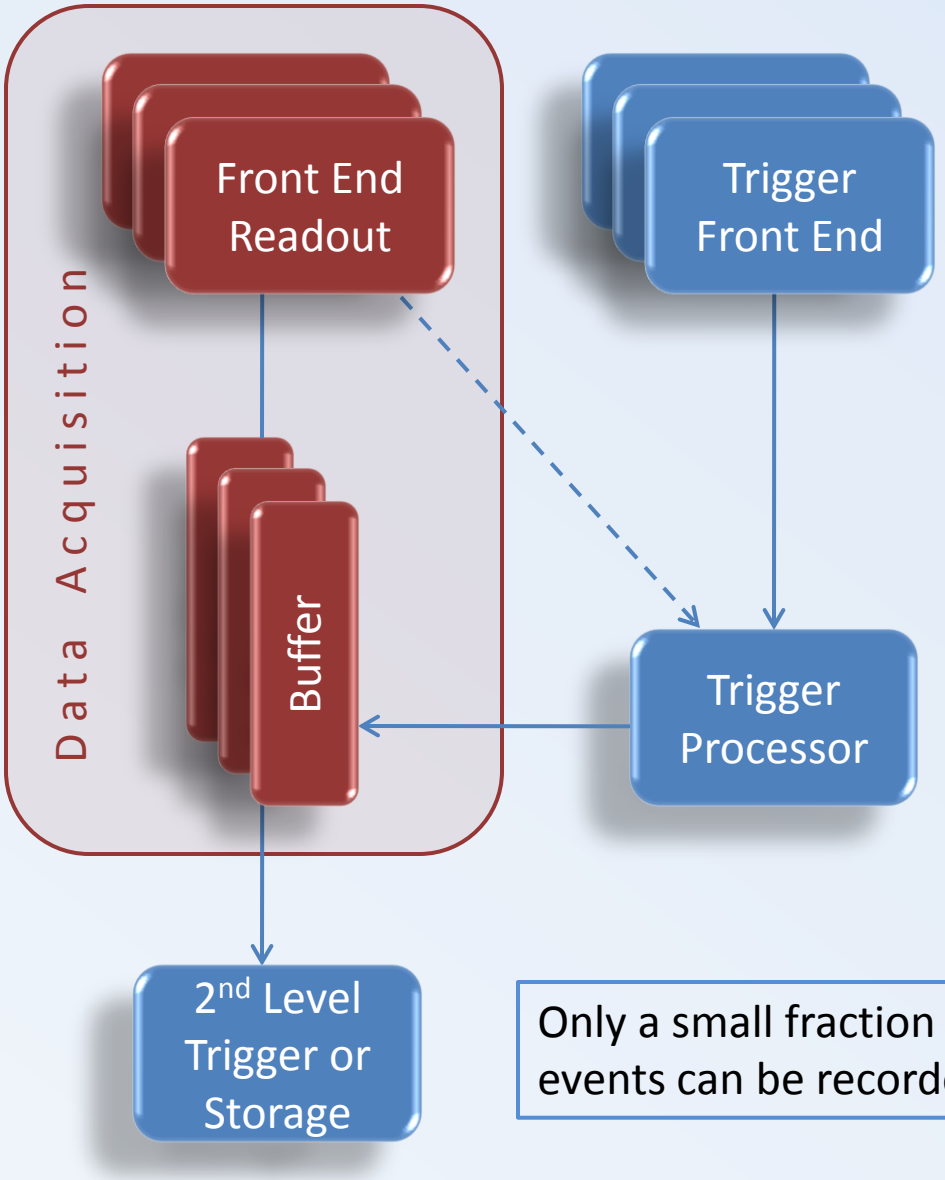
Trigger systems examples

- Compact Muon Solenoid (CMS) @ CERN
- Advanced Gamma Tracking Array (AGATA) @ GSI

Formal description

- Property Specification Language (PSL)
- Automatic Synthesis

Trigger in Physics Experiments



Purpose:
identify the events that should be recorded for later analysis

Key information are used to perform fast and approximate calculation

Typically low level trigger runs on custom electronics

Only a small fraction of events can be recorded

Trigger implementation

General requirements

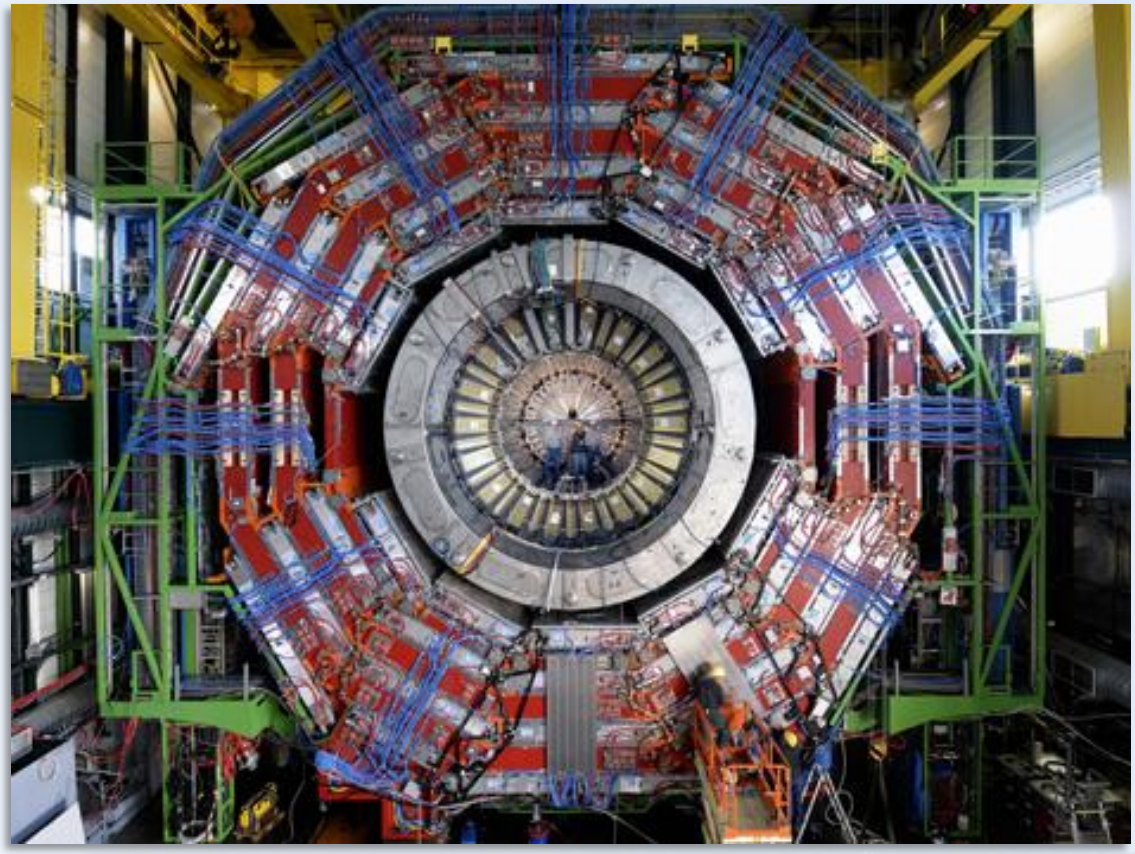
- Low latency: few μs
- Output rate 100 KHz ÷ 1MHz
- Large degree of parallelism
- Dead time free operation
- Flexibility
- Pattern matching algorithms
- High number of channels

How to implement it?

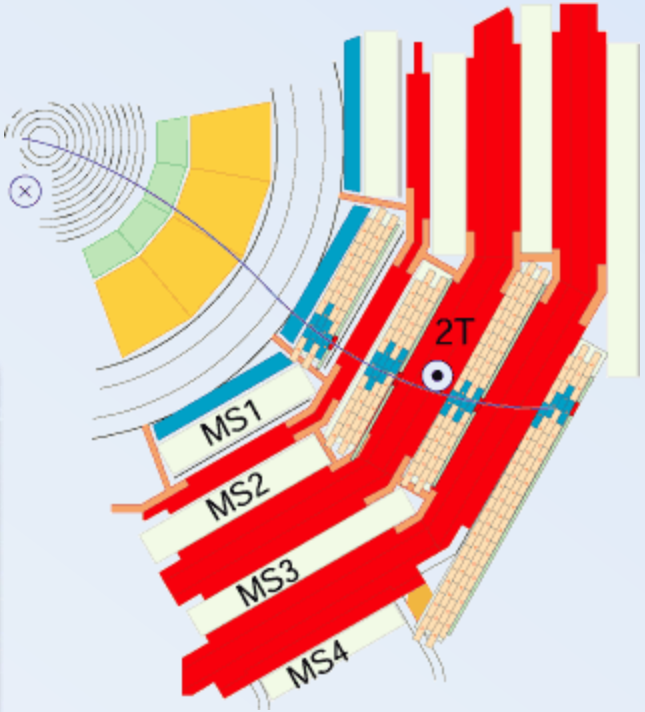
Temporal computing
- Slow
- Flexible
- Need instruction
- Low cost

Spatial computing
- Fast
- Not flexible
- No instruction
- Expensive

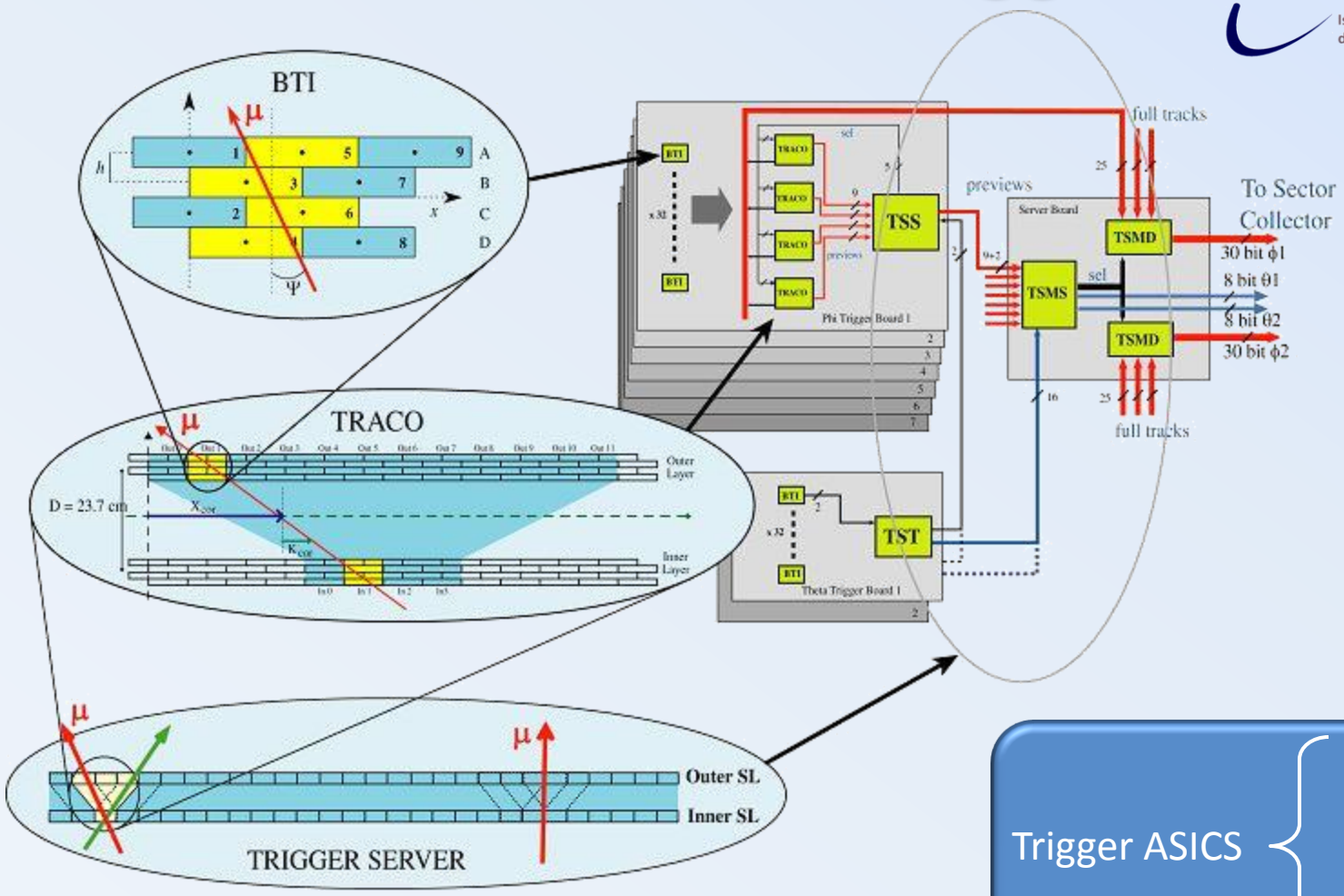
CMS: Muon Drift Tube



Size:	21 m long
	15 m wide
	15 m high
Weight:	12 500 tones
Location:	100 m underground
	inside LHC tunnel



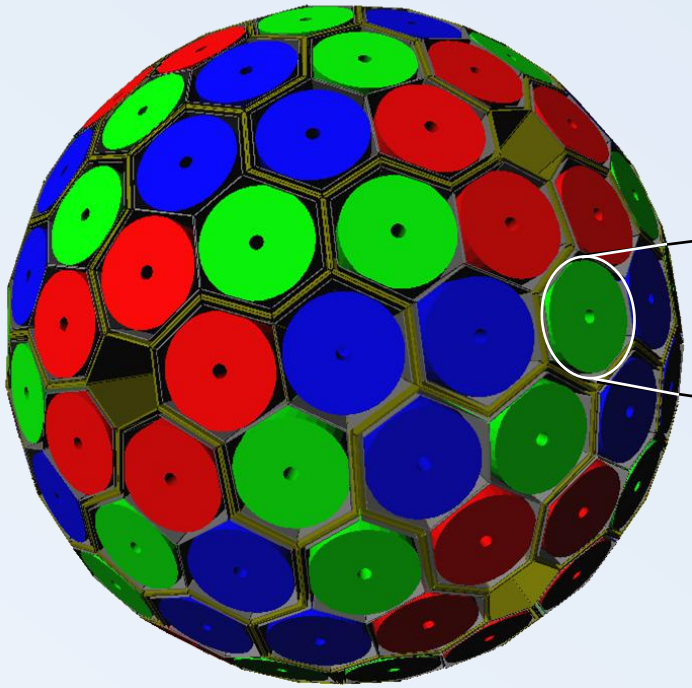
CMS: Muon Front End Trigger



Trigger ASICS		
	BTI	50000
	TRACO	4400
	TSS	1100
	TSM	240

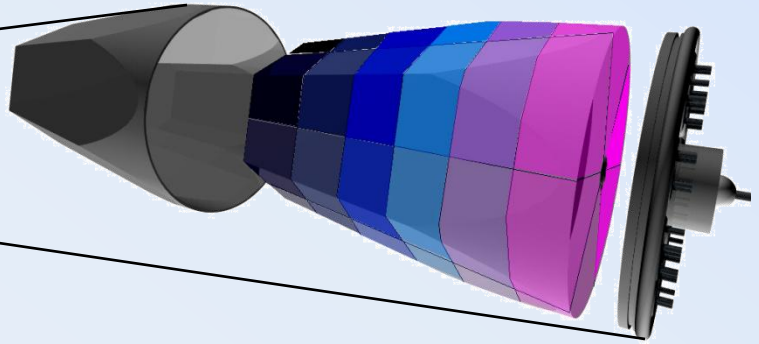
Track selection based on the best track quality and higher transverse momentum

AGATA Experiment



4π array of HPGe detectors
for in-beam γ-ray spectroscopy

x 180

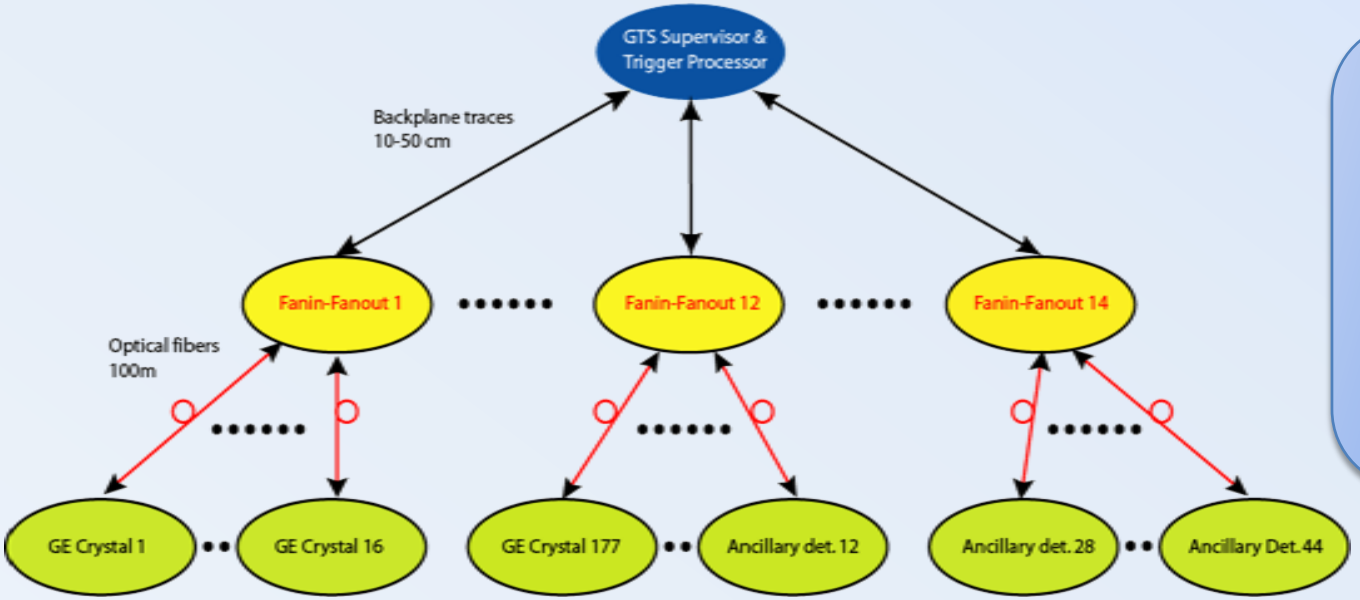


36 fold segmented crystal
+ central core contact
(6660 channels)

Digital electronics and sophisticated Pulse Shape Analysis algorithms

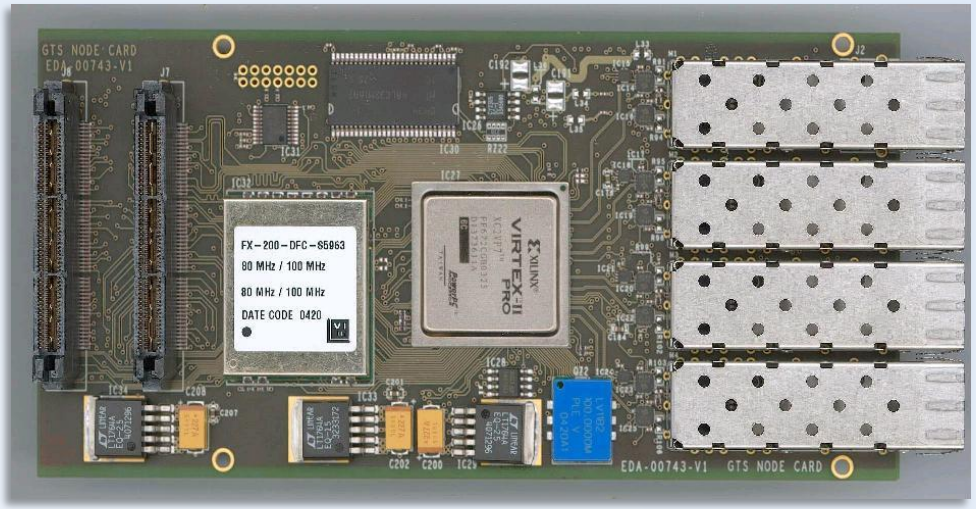
Operation of Ge detectors in position sensitive mode ⟶ γ-ray tracking

AGATA: Global Trigger System



The tree root node acts as a custom programmable trigger processor which continuously scans all incoming trigger requests for a match of the rules

User defined trigger rules range from simple multiplicity conditions in independent partitions to more complex delayed coincidences involving two or more partitions



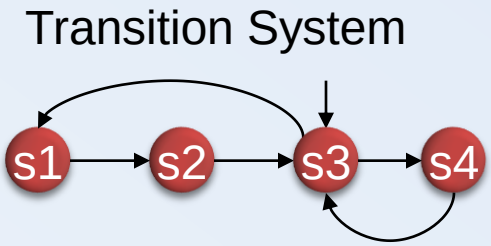
Formal description of trigger

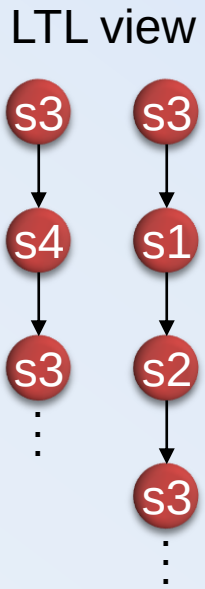
- To search for multiplicity conditions in independent partitions during a fixed time window or delayed coincidences involving more partitions
- Integration on multiple channels with threshold comparison
- To look in a small database of predefined patterns for a complete or partial matching with the recorded values

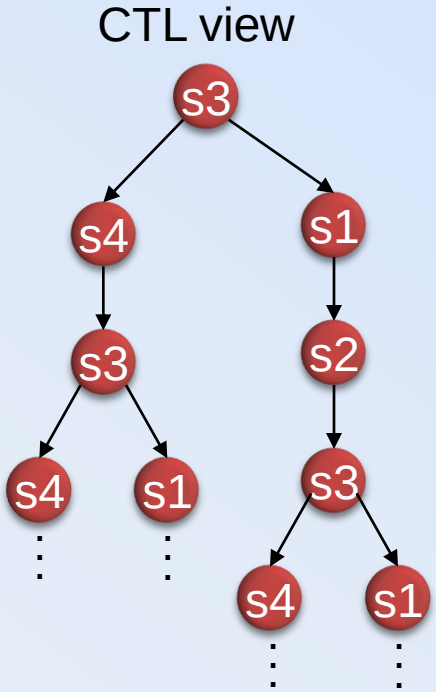
Temporal Logic

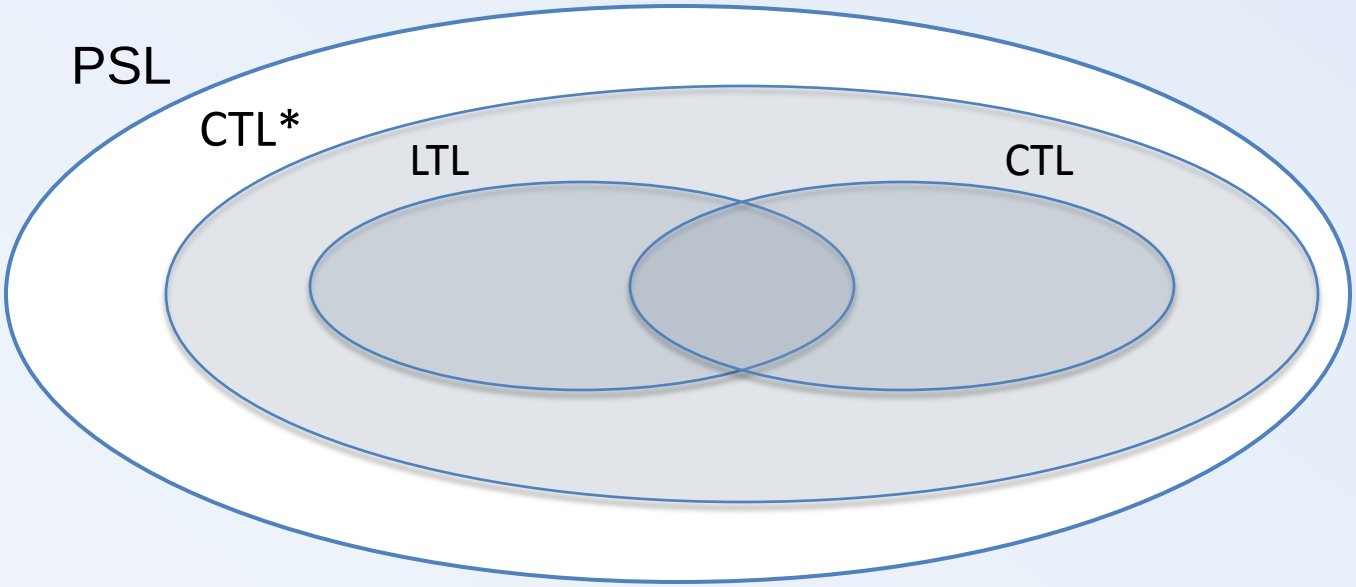
Modal Logic
- Necessity → formula will be always true in the future
- Possibility → formula will be eventually true in the future

Temporal logic

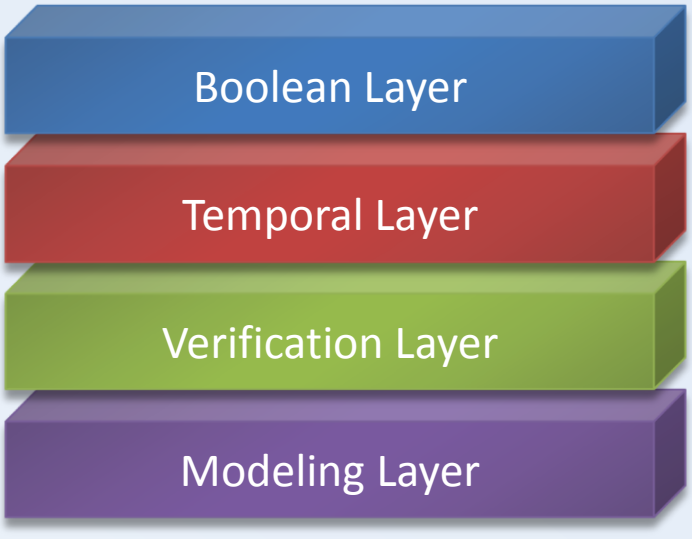








Property Specification Language

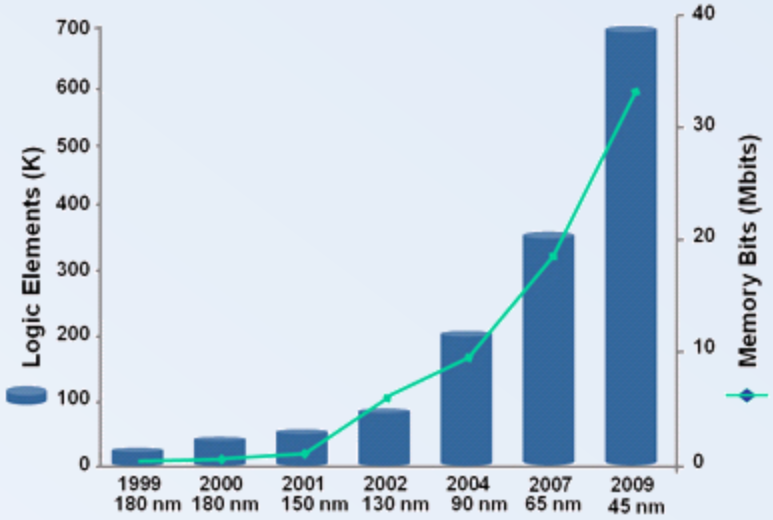


SystemVerilog – Verilog
VHDL – SystemC – GDL

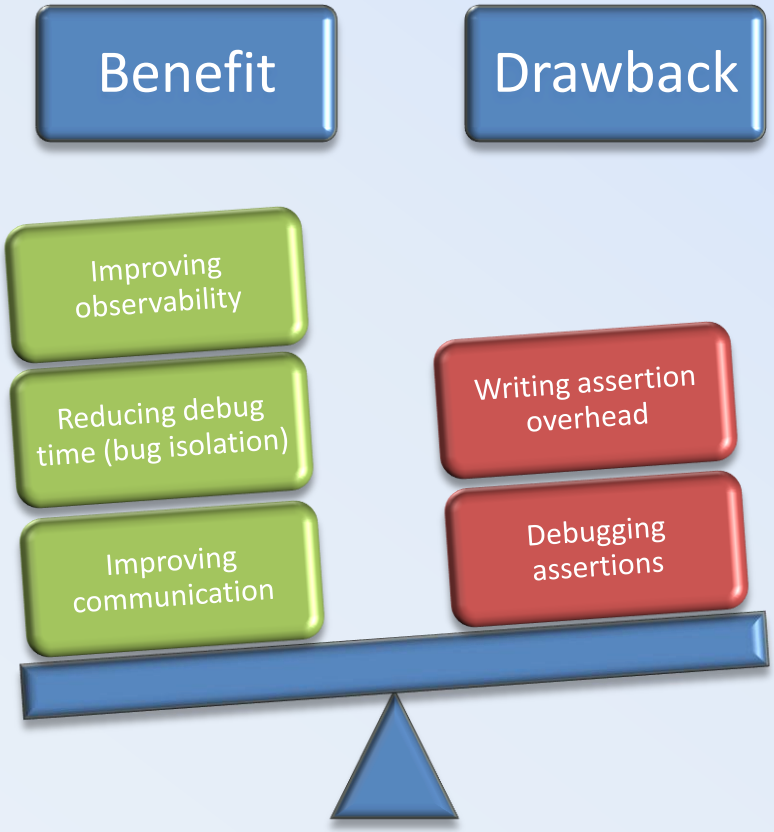
Functional specification → RTL implementation

Functional verification → Simulation
Functional verification → Formal verification

Assertion-Based Verification

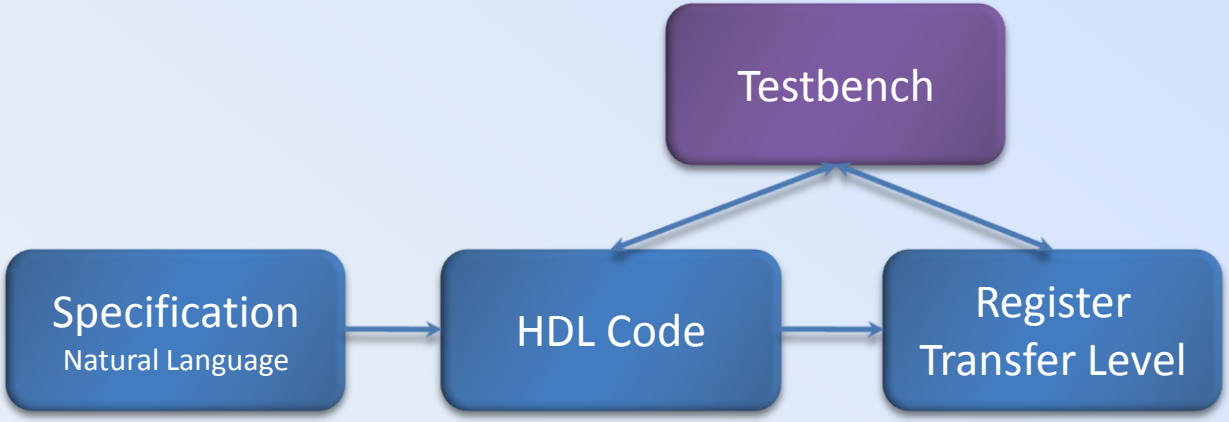


Testbench > Design
Lack of suitable debugging tools

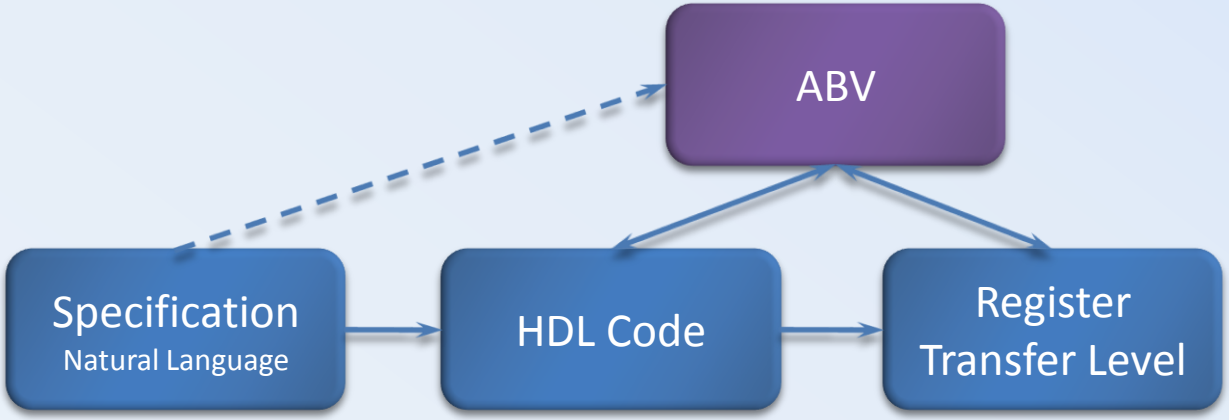


Design Flows

Simulation



Model Checking



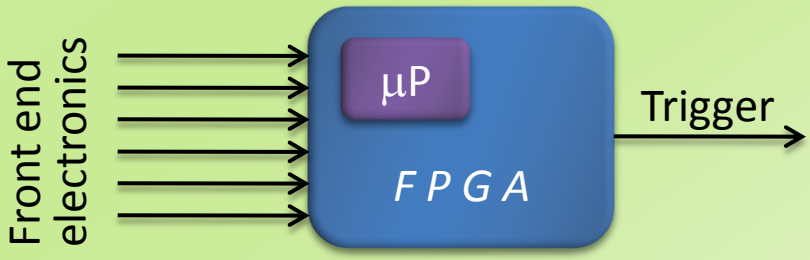
Automatic Synthesis

Constraint Handling Rules

Selecting trigger rules

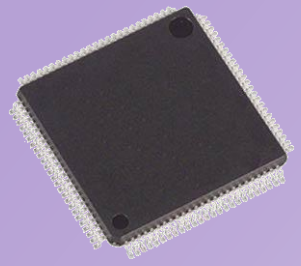
FPGA tags signals:
`new_event(T,P,E)`

CHR program runs on embedded microprocessor



Hardware elements

```
new_event(T,P,E), buf(X) <=> c_event(T,P,E,1), buf(X+1).
c_event(T1,_,_,X1) \ c_event(T2,P2,E2,X2) <=> X1=X2,
     T2<T1 | c_event(T2,P2,E2,X2+1).
len(L) \ buf(_), c_event(_,_,_,X) <=> X>L | buf(X-1).
```

驰 ==>  ?