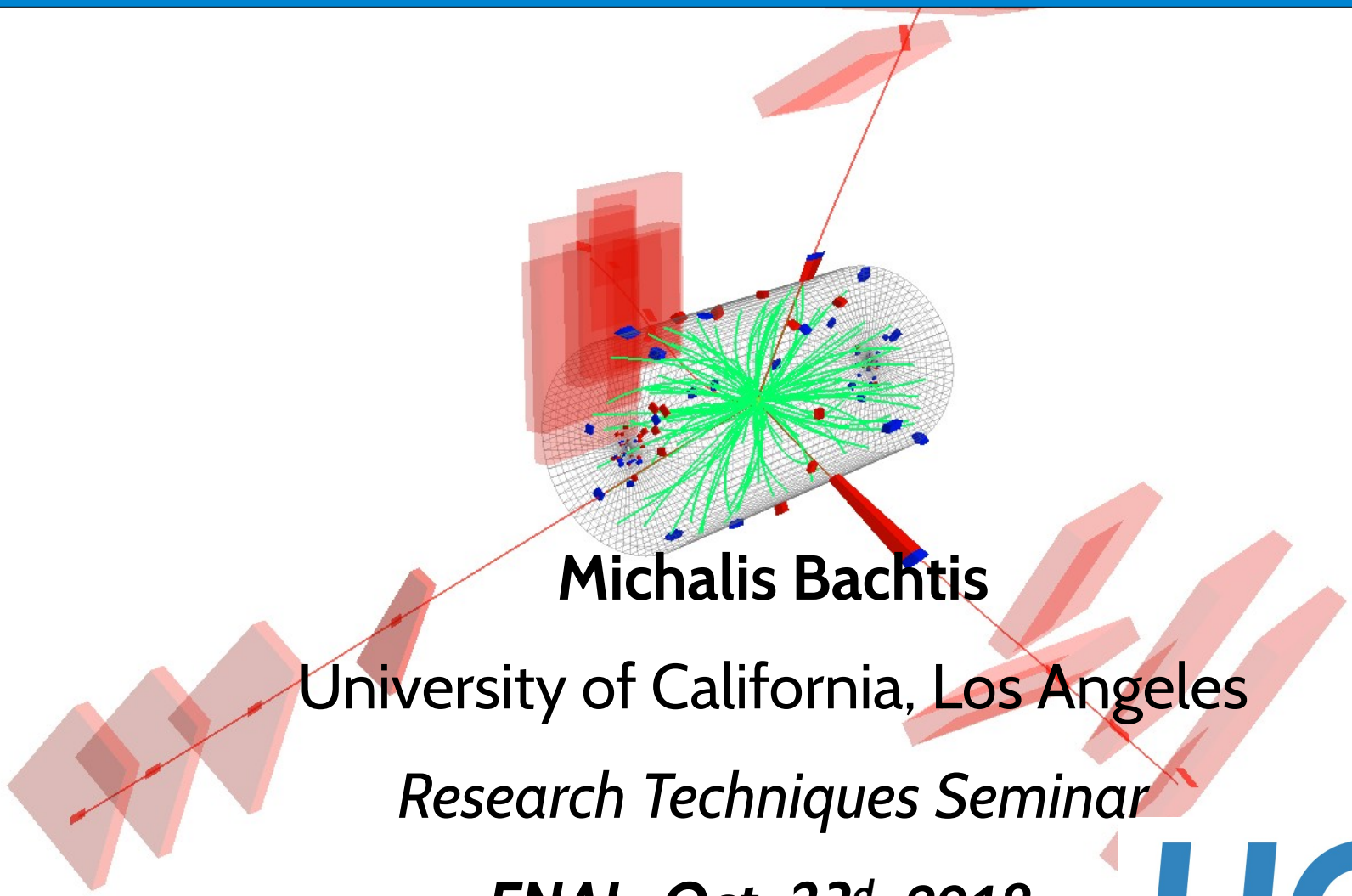
A Kalman filter for the CMS Muon Trigger for Run III and HL-LHC



Michalis Bachtis

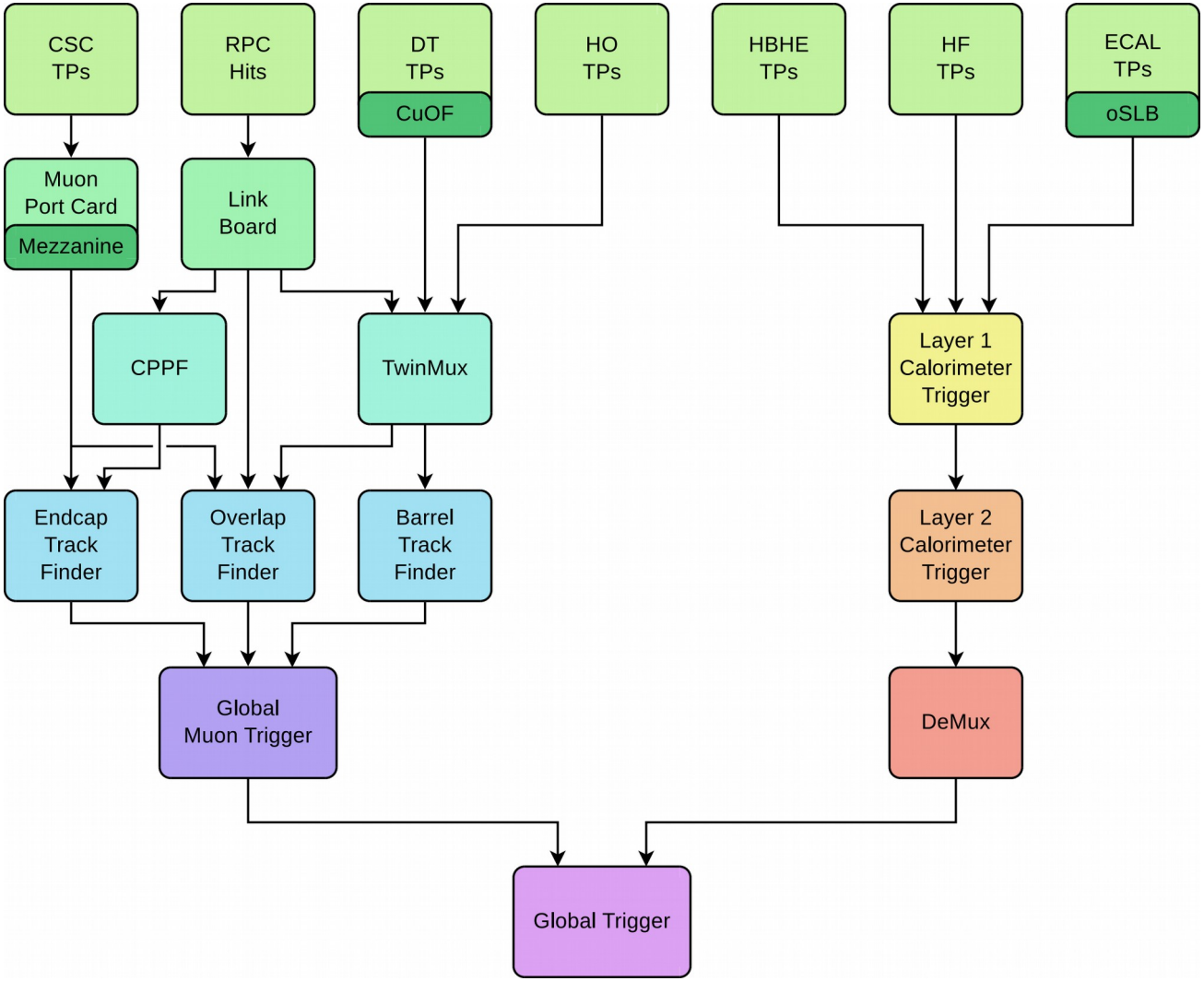
University of California, Los Angeles

Research Techniques Seminar

FNAL, Oct. 23d 2018

UCLA

The CMS L1 Trigger System



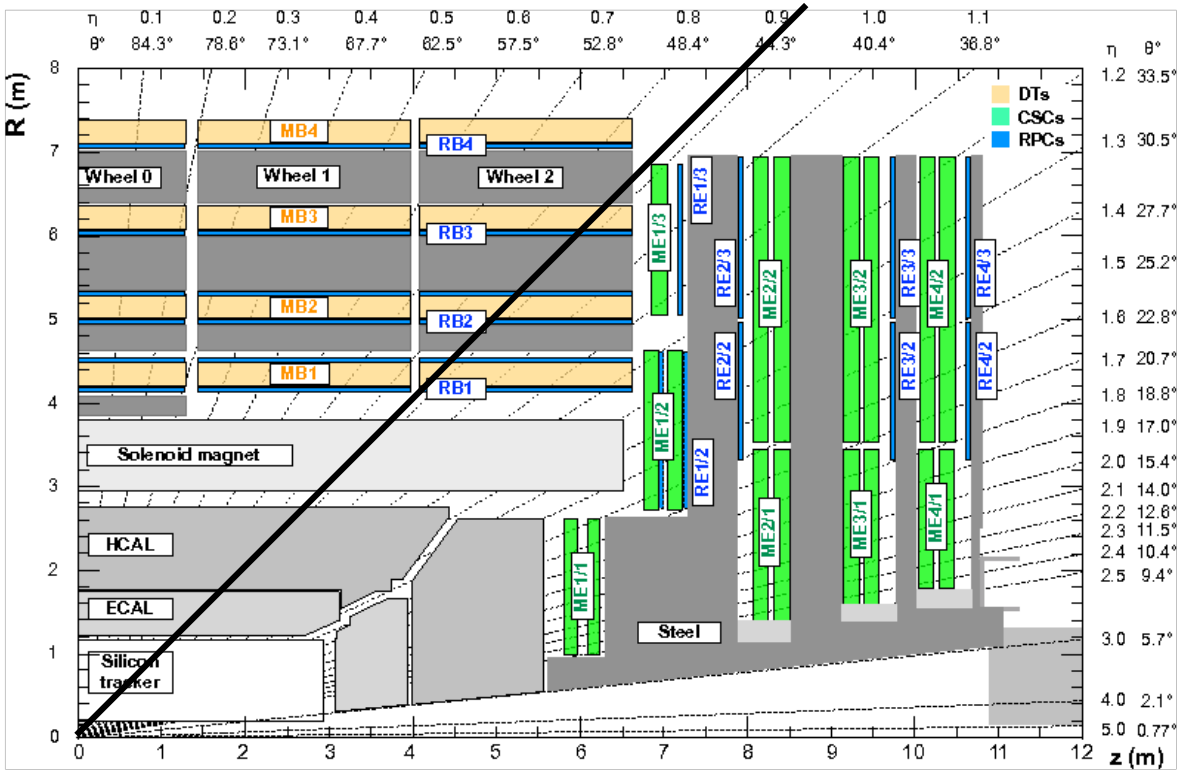
- Receives data from Calorimeter and Muon Detectors at a rate of 40 MHz and outputs data at 100 kHz
- Creates physics objects in HW (electron/photons, muons, jets, Missing energy) Decision taken only at the Global Trigger

Introduction

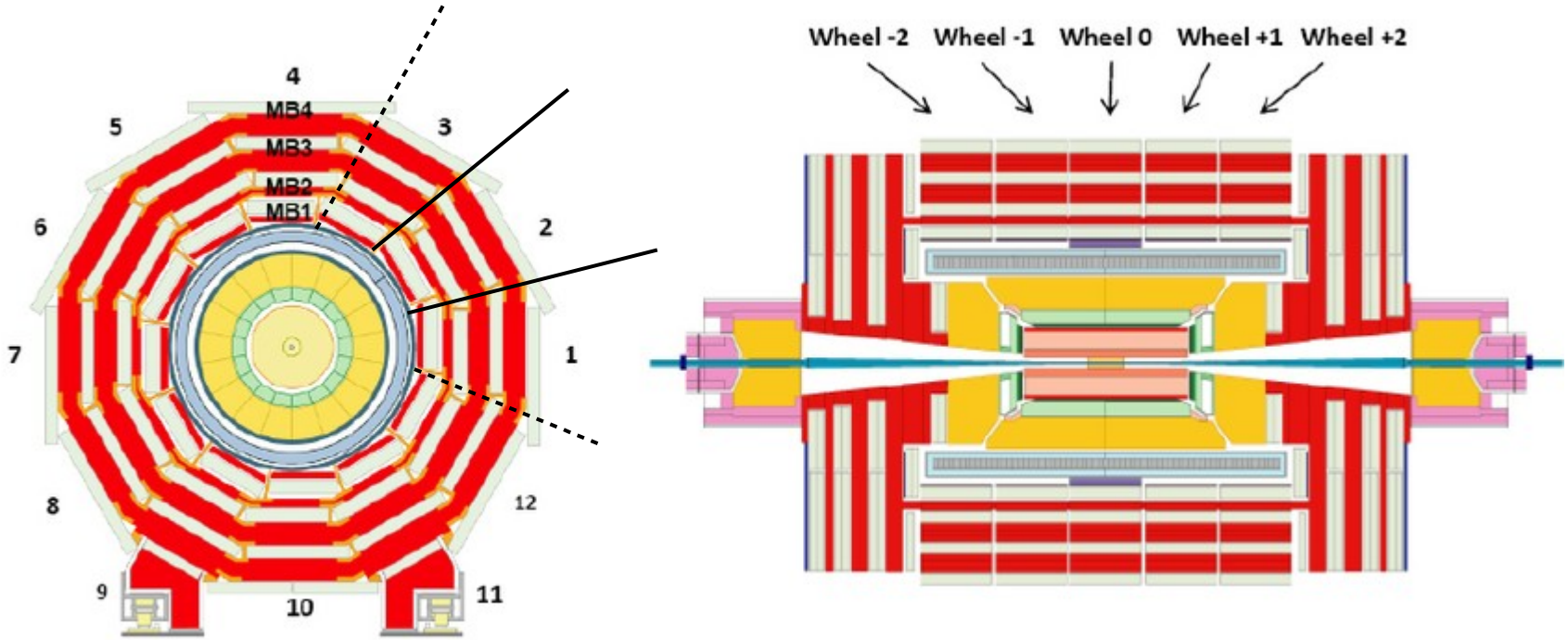
- Muon Trigger: Workhorse of the CMS Physics program
  - 75% of CMS publications rely in muon triggers to get the data on tape
    - Four out of the "big five" flagship Higgs analyses
- Muon Trigger Requirements for the HL-LHC upgrade
  - Trigger muons from electroweak processes with an efficiency> 95%
  - Enable displaced muon triggers to gain access to new physics models

- UCLA Focus:
  - Barrel Muon Trigger Track finding for HL-LHC
  - Already involved in current Barrel Muon Trigger with the European groups

Barrel Region

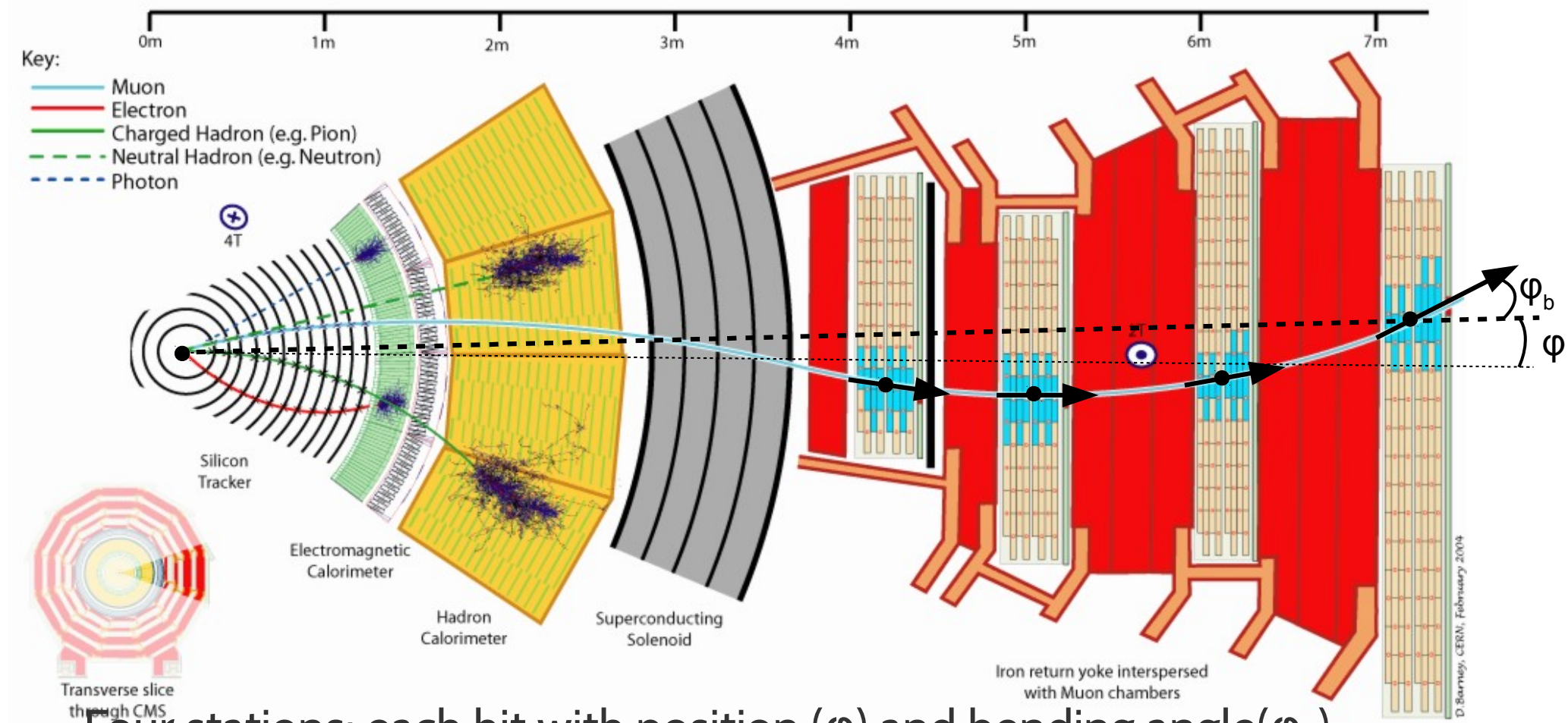


Current Barrel Muon Track Finder



- Consists of 12 processors featuring a Virtex 7 690T FPGA
- Each processor receives data from 5 wheels in θ x 3 sectors in φ
  - Left and right sectors shared in each processor to account for boundaries
- Each processor forwards 3 muons to the Global Muon Trigger (GMT)
  - GMT cleans overlap between sectors and track finders and forwards data to the Global trigger where decision is made

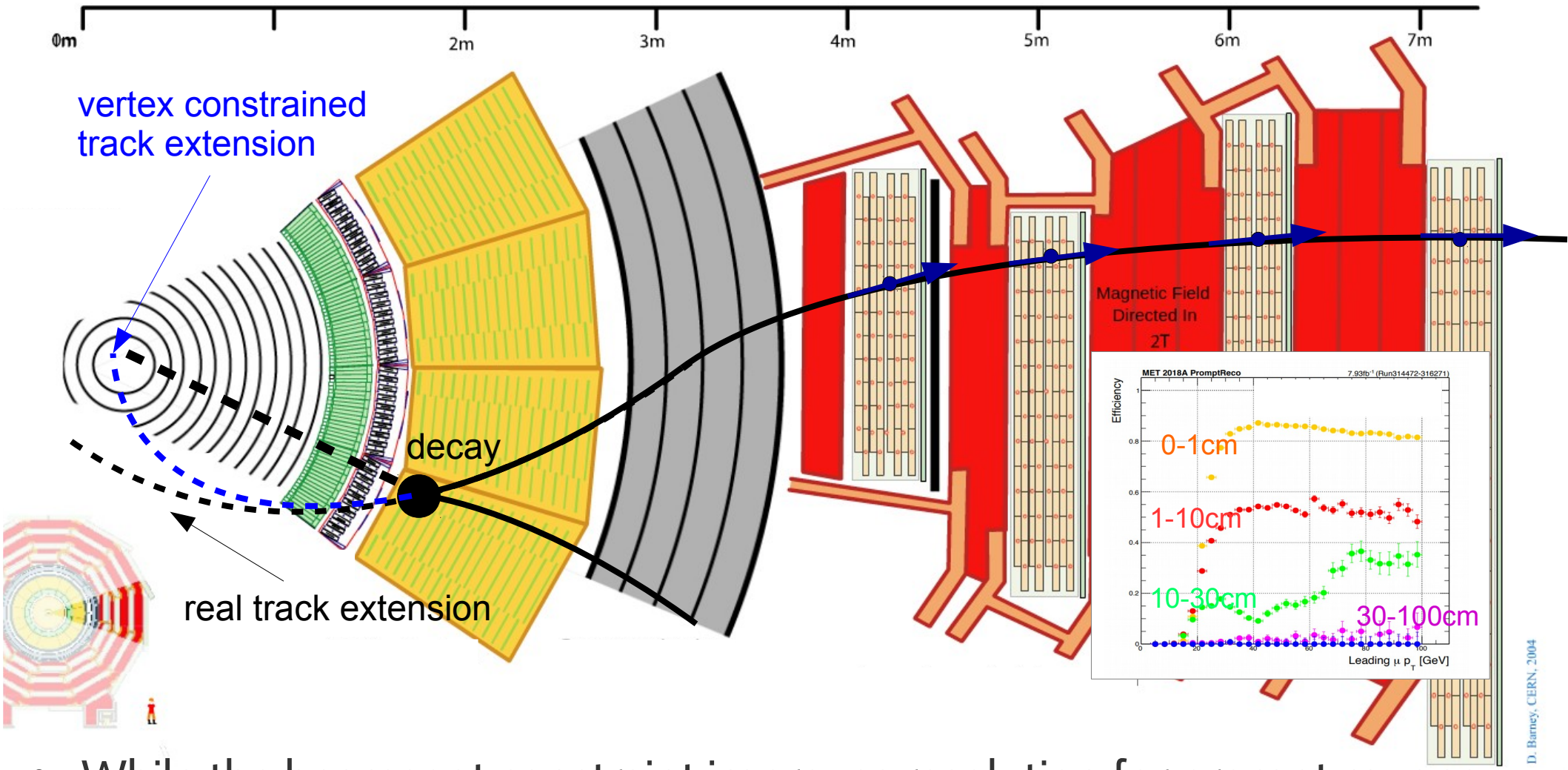
Tracking in the CMS barrel muon system



- Four stations: each hit with position (φ) and bending angle(φ_b)
  - 22 bits per station (maximum 88 bits per track)
- Current track finder assigns momentum using the $\Delta\varphi$ between two stations + assuming the track comes from the center of the detector
- Beamspot constraint improves resolution

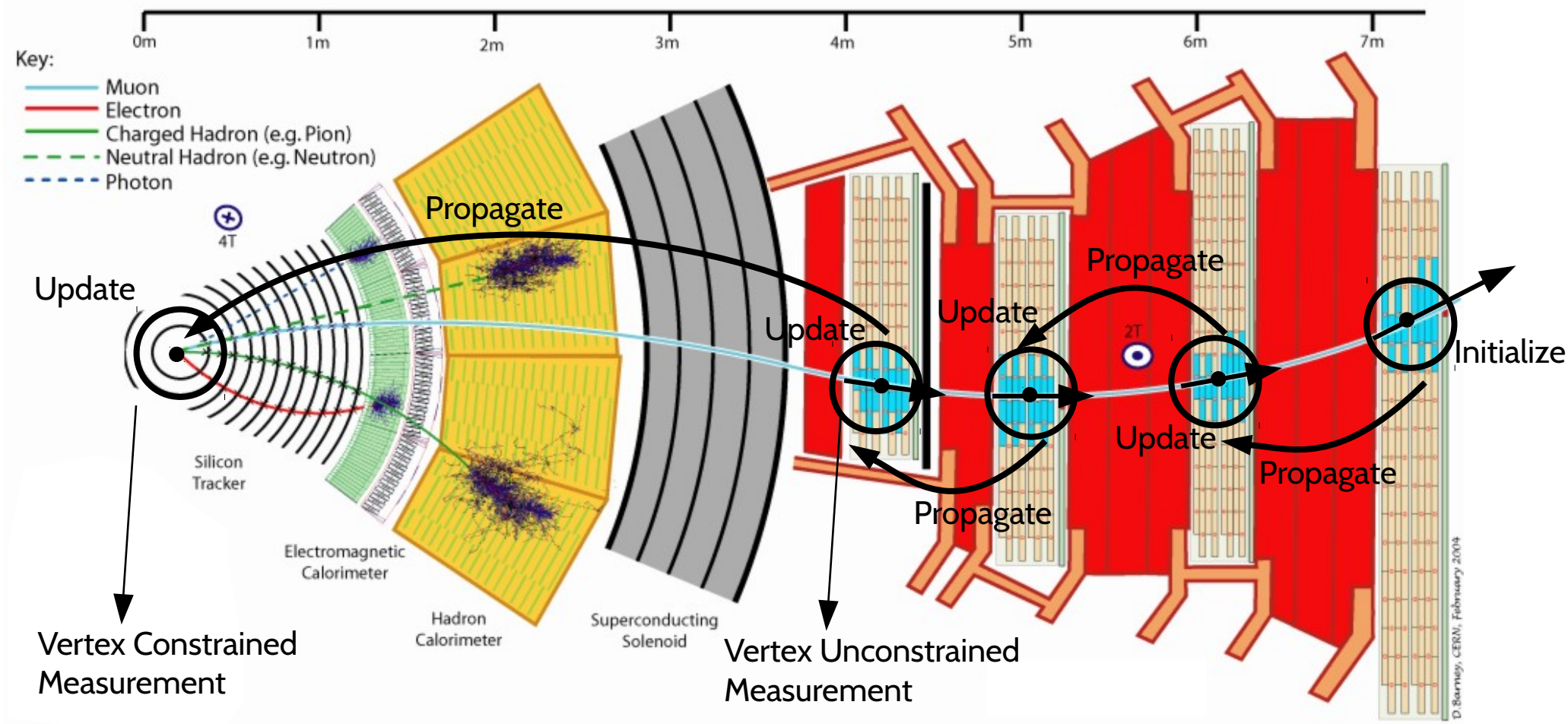
$$\frac{\sigma_{p_T}}{p_T} \sim \frac{1}{BL^2}$$

Beamspot constraint and displaced muons



- While the beamspot constraint improves resolution for prompt muons, it is sub-optimal for displaced particles
  - Momentum mis-measurement results in trigger inefficiency
- Displaced particle models very popular lately → Need displaced triggers

A Kalman Filter for BMTF



- Sequential algorithm: (mathematically equivalent to a χ^2 fit)
  - Propagate track inwards from station to station and match with a stub
  - Update track parameters and continue
- After reaching station 1 → save measurement without vertex constraint
- Propagate to vertex and update → vertex constrained measurement
- Challenge for an FPGA implementation → **Matrix algebra**

Track propagation from station to station

- Track parameters in the muon system: $\mathbf{x}_n = (\varphi, \varphi_b, k = q/p_T)$
  - Uncertainties on parameters expressed by a 3x3 covariance matrix P
- Assuming a perfect helix (P_T=0.3BR) and approximating with a parabola (of constant 1/(2R)) and converting to the track parameter space, we get a linear propagation relation:

Propagation Matrix F

$$\begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_n = \boxed{\begin{pmatrix} 1 & 0 & 0 \\ a & 1 & b \\ c & 0 & 1-b \end{pmatrix}} \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_{n-1}$$

Propagation of uncertainties

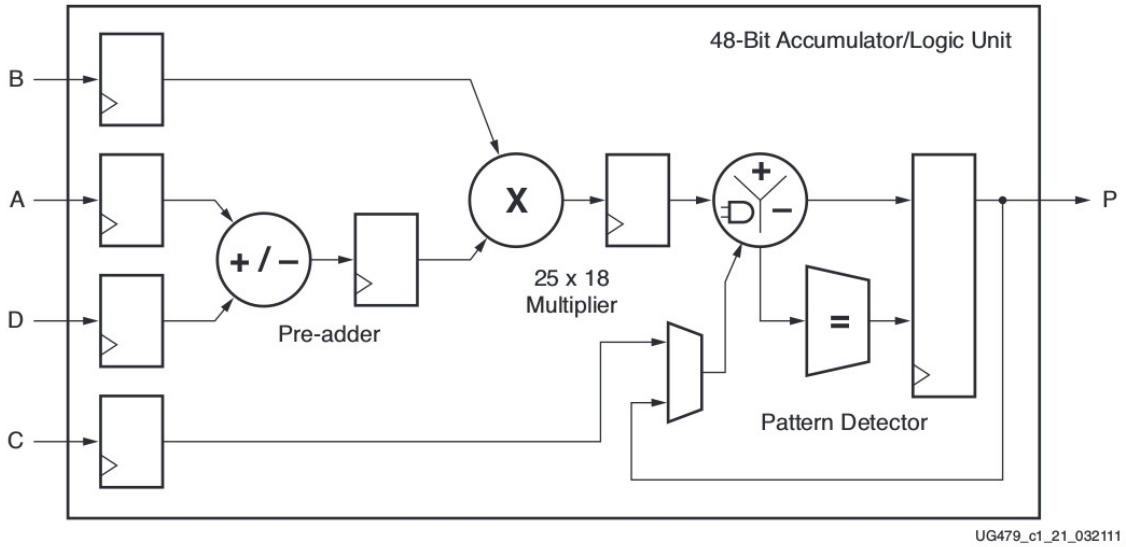
$$P_n = \boxed{F P_{n-1} F^T} + \boxed{Q(k, x/X_0)}$$

Multiple scattering

- Constants a,b,c depend on B-field and dimensions and are different from station to station
- Energy loss neglected for station-to-station propagation and added inclusively at the end

Performing linear algebra in FPGAs

- Use of "traditional" logic (LUTs and FFs) requires many multiplications that are expensive in FPGA resources
- Modern FPGAs include DSP cores unused in the current trigger system
  - Presence of DSP in modern FPGAs motivated by filtering applications, machine learning, and military stuff



- The operation x+y*z can be mapped to a DSP core reducing substantially the resources
  - Kalman filter matrix algebra mapped to DSPs

Adding hits to the track

- Each detector stub is a measurement $z_n = (\varphi, \varphi_b)$
- A residual is formed :

Associated Stub — Matrix H — Propagated track

$$r_n = \begin{pmatrix} \phi^{\text{stub}} \\ \phi_b^{\text{stub}} \end{pmatrix}_n - \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} k \\ \phi \\ \phi_b \end{pmatrix}_n$$

- The cov. matrix is transformed to measurement space and position resolution is accounted for

Transformation of P — Chamber Hit resolution

$$S = HP_nH^T + R$$

- Finally a weight (Kalman Gain) is estimated and the state is updated

Kalman Gain — matrix inversion...

$$x_n^{\text{upd}} = x_n + PH^TS^{-1}r_n$$

Implementing track update in the FPGA

- A lot of matrix operations including a 2x2 matrix inversion
  - Requiring division which is expensive in latency and resources
- To implement the update, we look into the physics
- What does the Kalman gain depend on?
  - It depends on the p (and therefore k) of the track at the station of the update
    - Since multiple scattering is depending on p
  - It depends on the hit-pattern of the already reconstructed track
    - A track that is updated at station 1 and already has hits in stations 2,3,4 will have a smaller weight at station 1 than a track that has only one hit in station 4
- Therefore instead of the matrix algebra we pre-calculate the Kalman gain as a function of k for each hit pattern and read it from a ROM
  - The FPGA used has memory blocks (BRAM) that can be used in read-only mode allowing 4K addresses and 9 bit storage
    - Therefore we compress k in 12 bits, and we define the gain in 9 bits using one BRAM for each update

Propagation to the beamspot and constraint

- To propagate to the beamspot, the track passes from the magnet coil, the calorimeters, and the tracker.
  - Energy loss must be taken into account
- Assuming muons in the 50 GeV range, an offset in p is applied

$$p' = p + \epsilon$$

- Expressing it as a function of curvature we get:

Implemented as LUT in Block RAM

$$k' = \frac{1}{p'} = \frac{1}{p+\epsilon} = \frac{k}{1+\epsilon k} = k - \boxed{\frac{\epsilon k^2}{1+\epsilon k}}$$

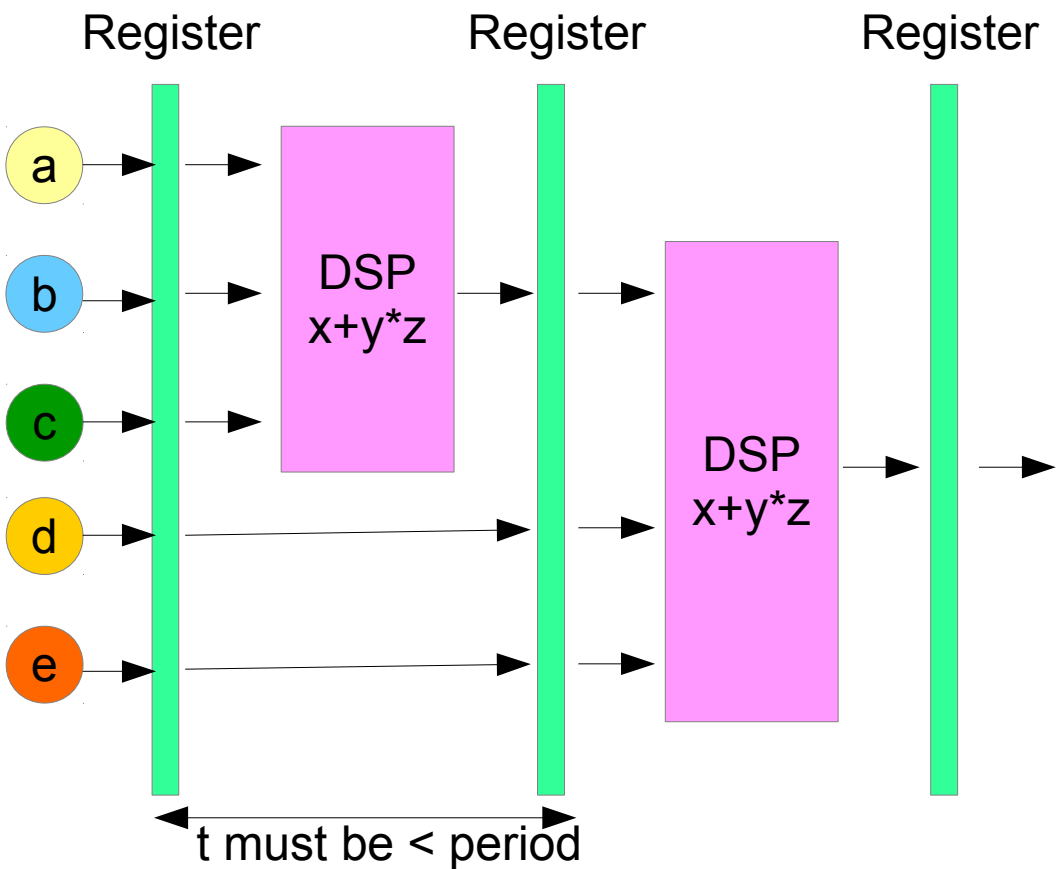
- An approximate impact parameter is propagated as

$$d_{xy} = \phi_b - ck$$

- A kalman update is performed assuming the "measured" value of impact parameter is zero

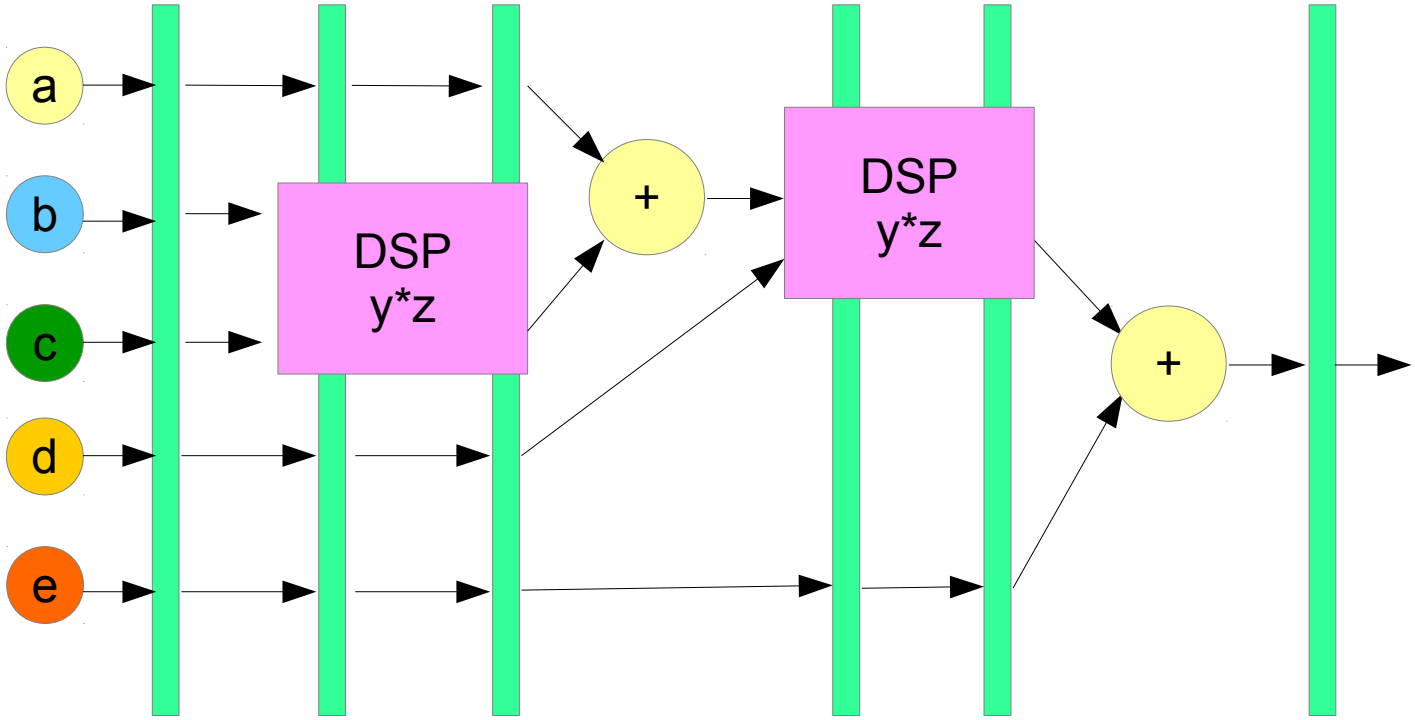
Firmware implementation tools

- Firmware written in Vivado high level synthesis (HLS) -C based
- Lessons learned
  - Works great if the code is simple. No for loops used, no complicated if statements
  - The best strategy is to code assuming coding in VHDL/Verilog just not worying about the registers
- Example of synthesis : Assume one wants to implement (a+b*c)*d + e and synthesizes with a very slow clock speed



Higher clock speed

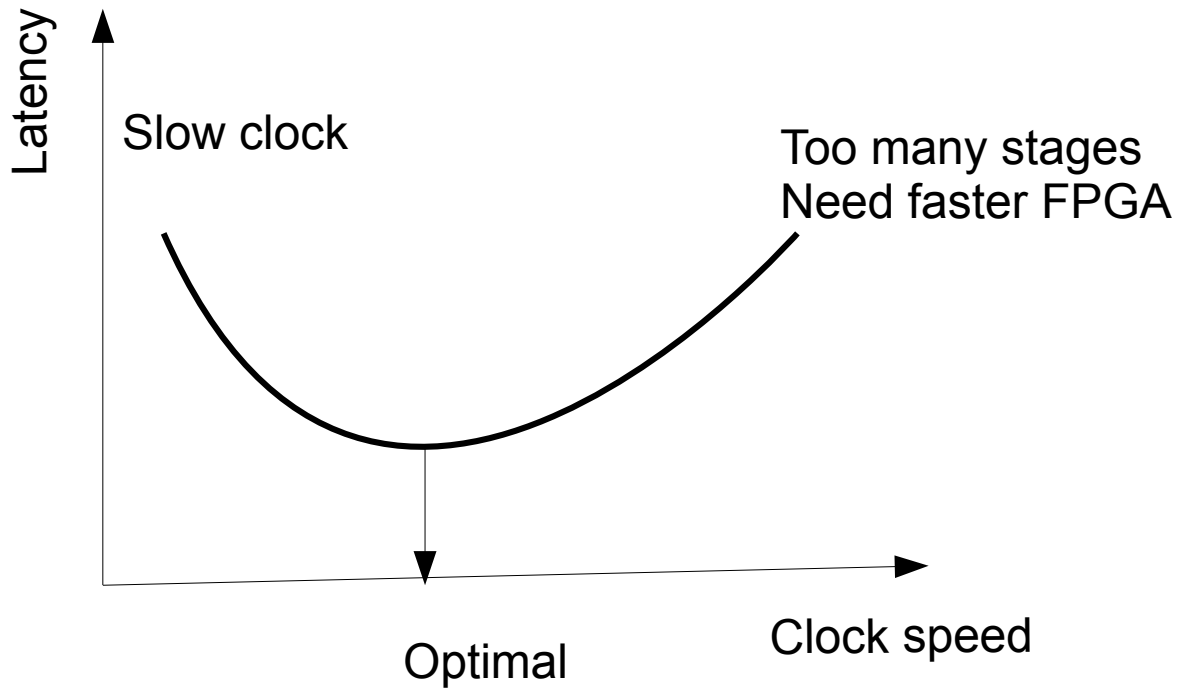
- If you ask for higher clock speed, HLS will place registers to optimize the timing paths. The same example would look like this:



- In this case HLS will use more pipeline stages to keep the timing path in the limits
- Not a perfect procedure yet but saves **infinite amount of development time**

Optimizing for logic

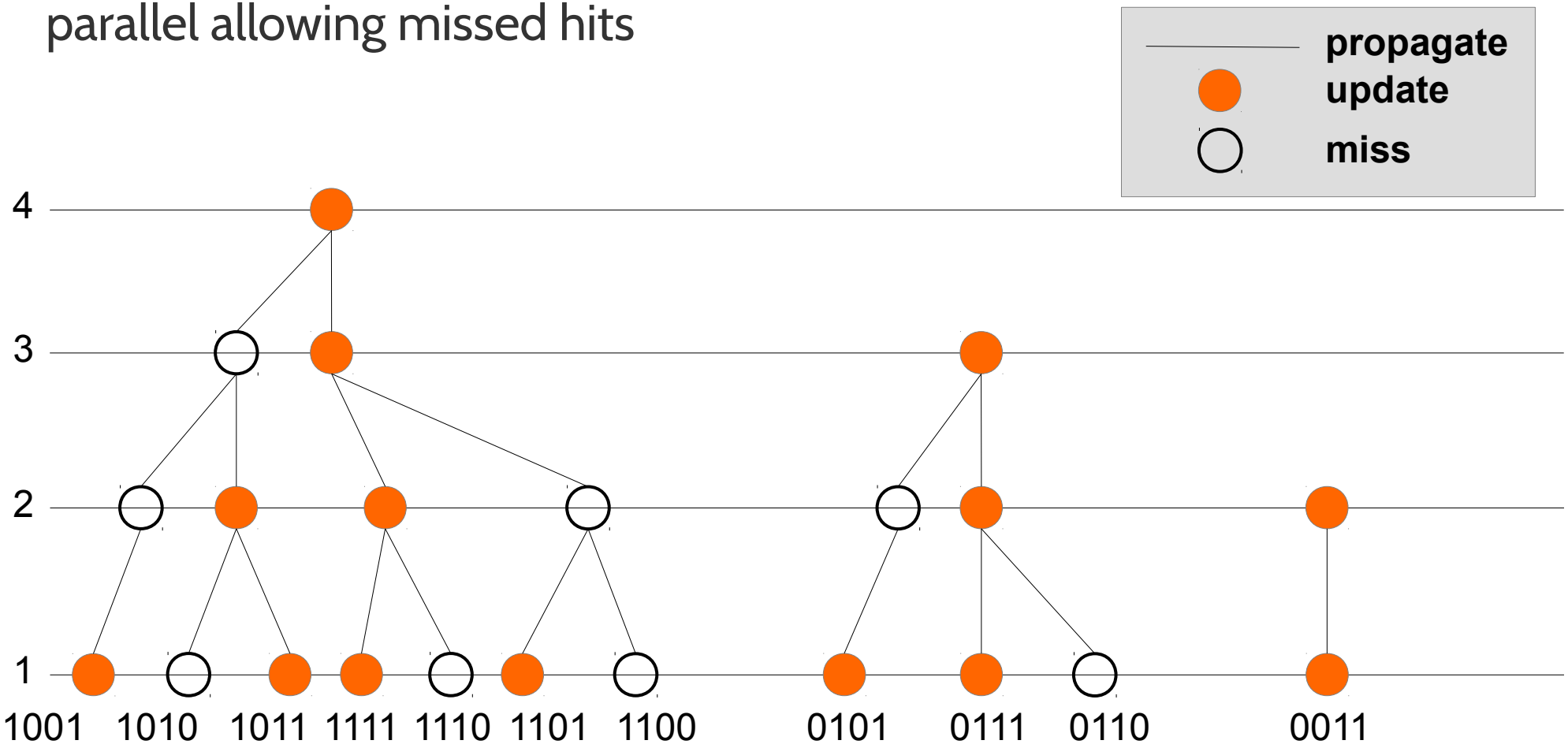
- Repeating synthesis for different clock speeds result in the optimal latency working point for the FPGA and the algorithm used:



- As the clock gets faster, latency decreases
- At some point because of the FPGA speed too many stages are added to keep the timing paths in the limits so latency increases
- In the BMTF Kalman algorithm: Optimal at 200MHz
  - We use 160 MHz to keep the same clock with the current running algorithm

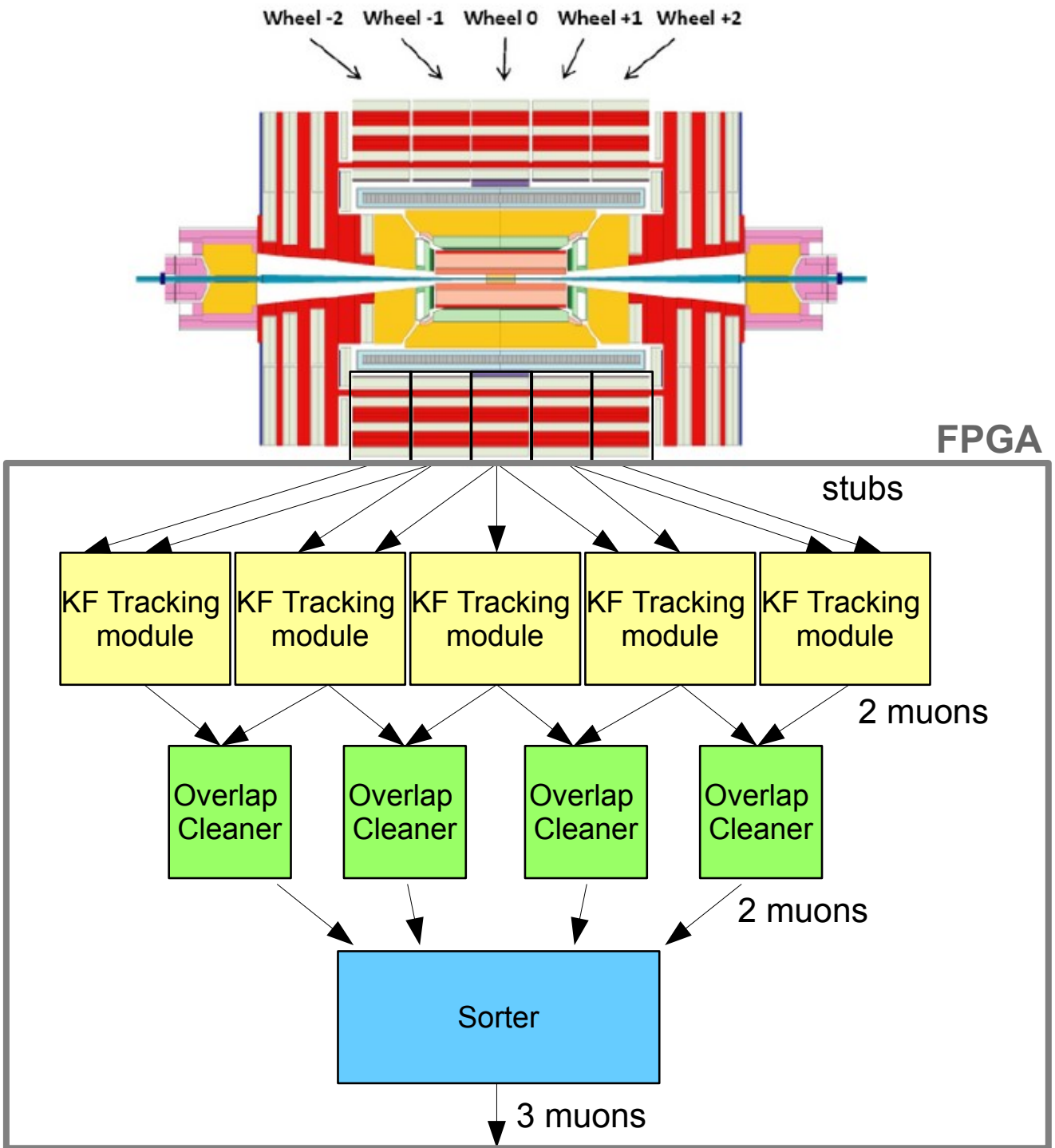
The Tracking module: Parallilization

- We exploit parallel processing to implement all possible combinations in parallel allowing missed hits



- At least 2 hits are needed to define a track
- We chose the best among tracks with overlapping stubs by an approximate χ^2

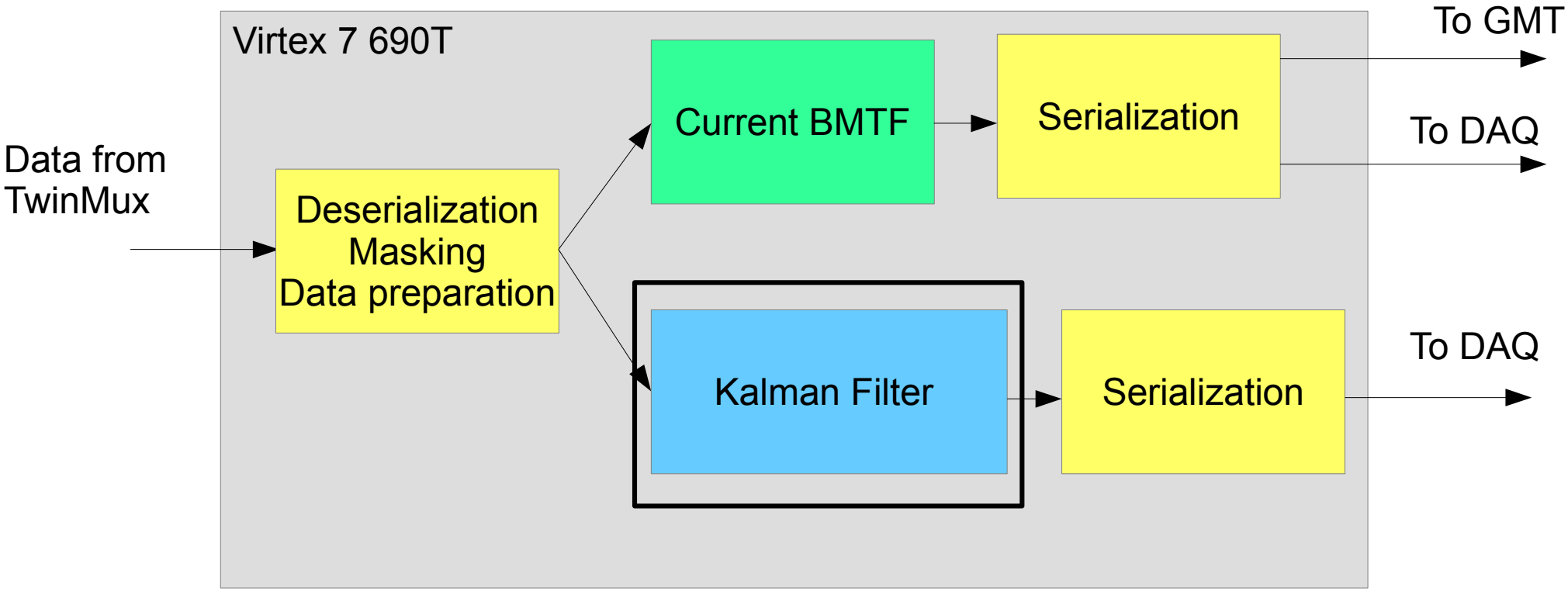
Total firmware implemented in the chip



Performance in Virtex 7 FPGA

- Resource utilization very low:
  - 16% of the logic elements
  - 25% of the DSP cores
- Latency of 26 clocks at 160 MHz
- **Implying we are taking the hits, fit all track combinatorics, perform overlap cleaning and sort them in 162.5 ns or 6.5 bunch crossings!**

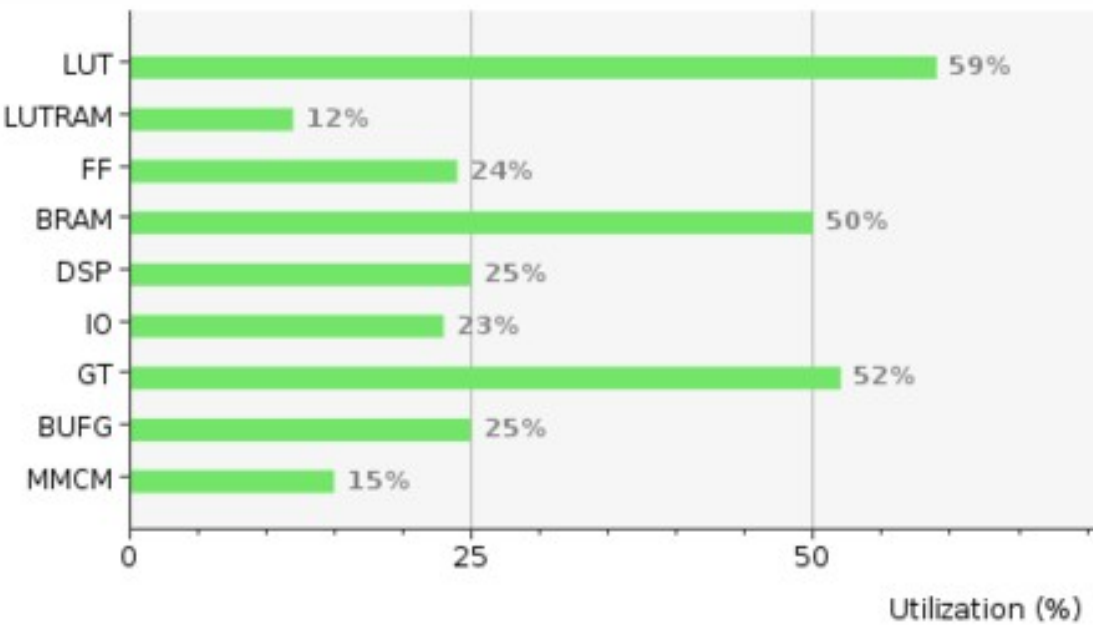
Deployment in CMS data taking



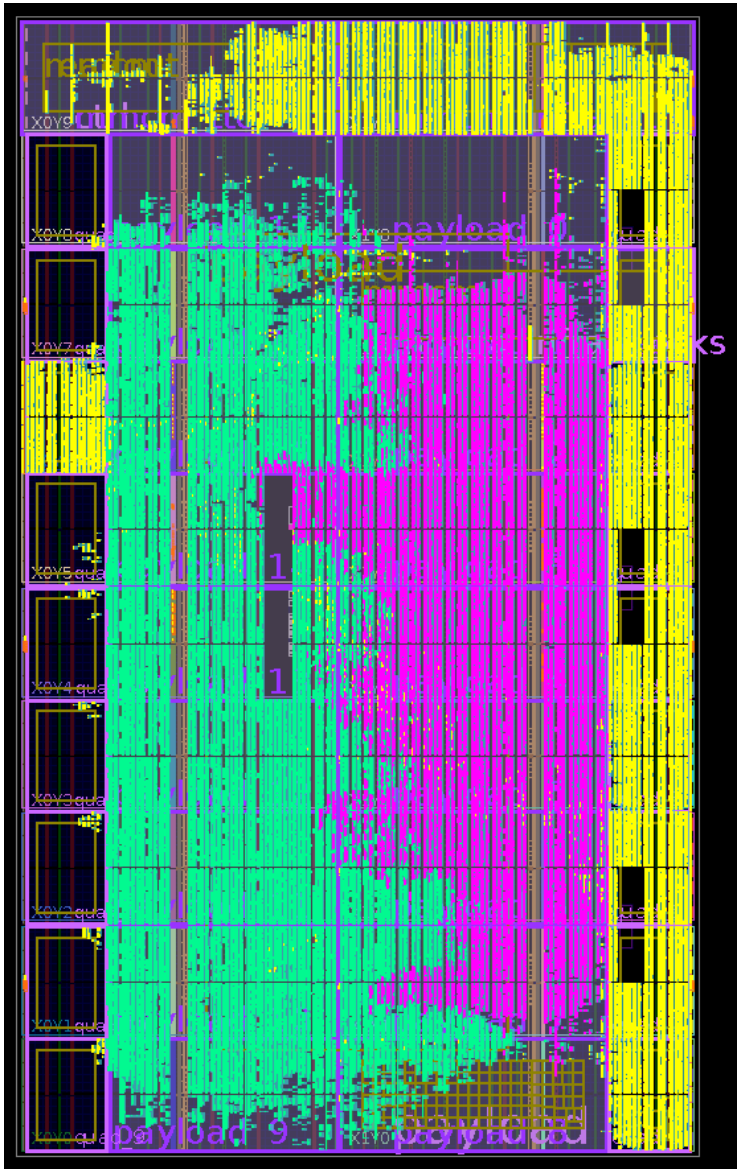
- To commission the Kalman filter in data, we implemented **both** algorithms in the same chip
- We trigger with the current trigger but we read-out the Kalman muons for each triggered event in CMS
  - To study the firmware performance and compare data with the emulator

Firmware results for both track finders

- About 60% of the chip
  - Current BMTF , Kalman, links and infrastructure framework
- Experiencing easy synthesis
  - Good timing closure
- Latency of the full system with the Kalman trigger → 9.50 BX



- Kalman filter smaller in size than the current track finder!

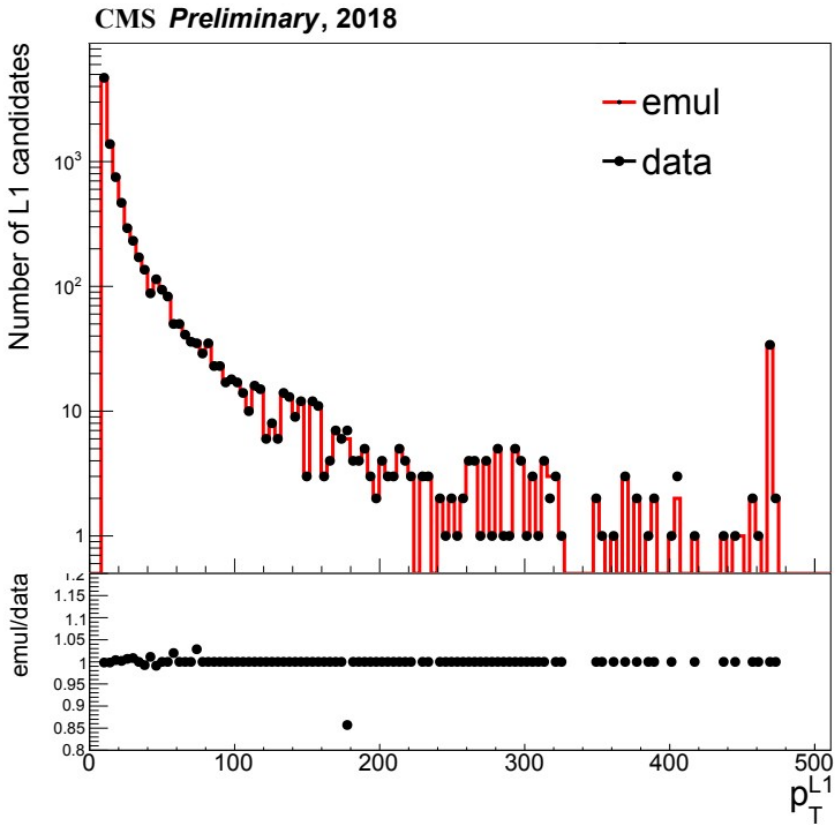


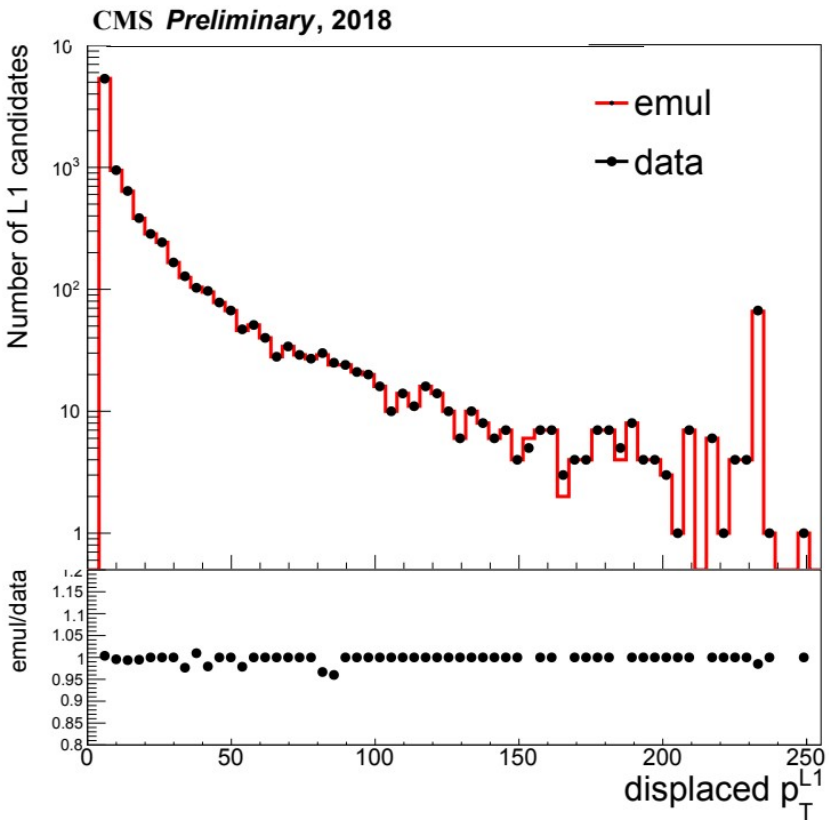
Kalman-BMTF

BMTF

Infrastructure

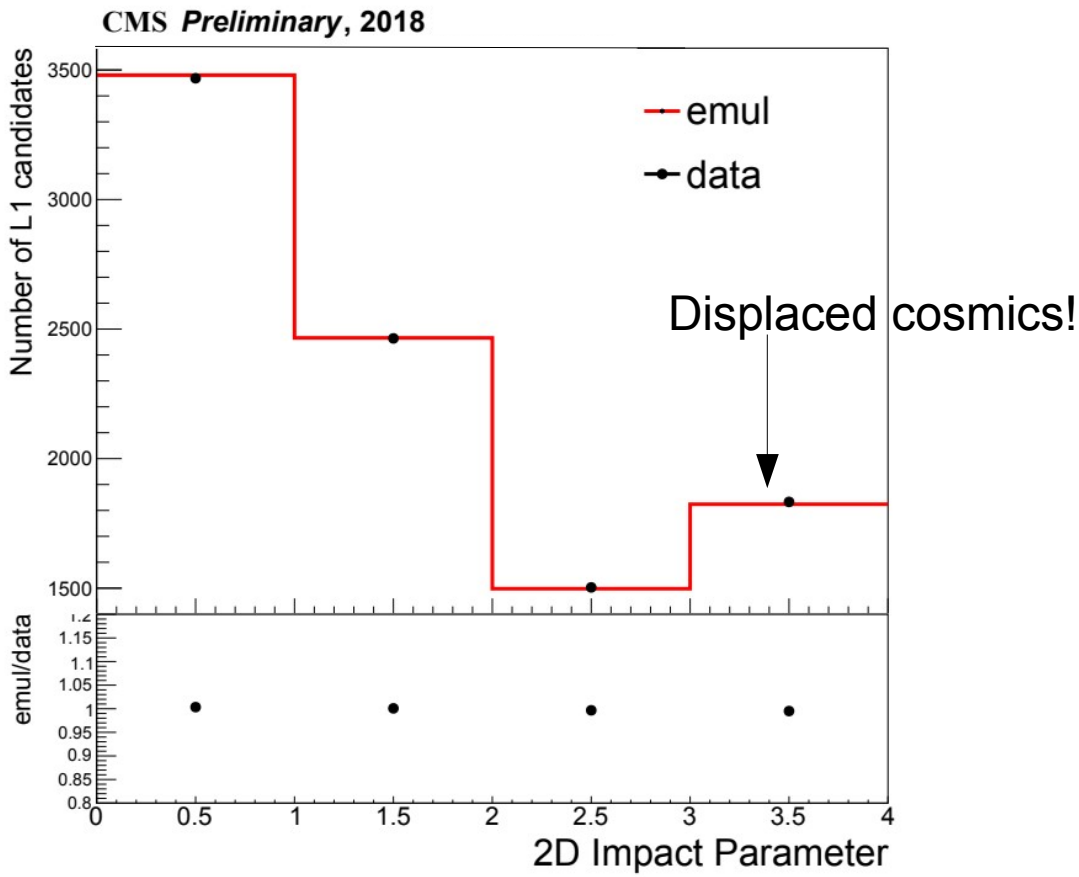
Results from CMS data taking in 2018 (I/II)





- The performance of the hardware is evaluated by comparing the output of the boards with an emulator built in software
- Agreement better than 99%
  - Residual disagreement → fixed to be done in the emulator !
- Algorithm has been tested as the default trigger and is ready for Run III

Results from CMS data taking in 2018 (II/II)

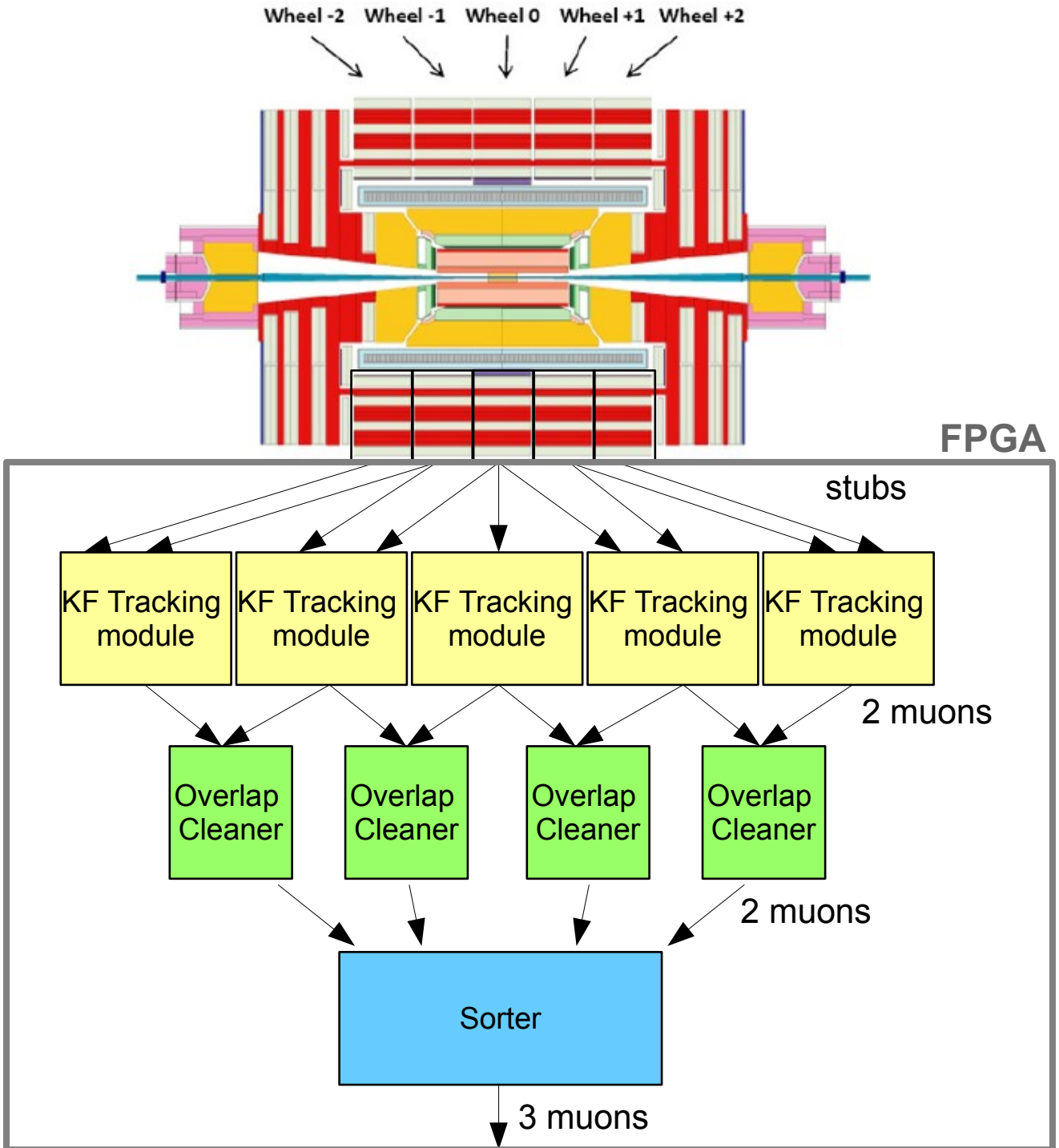


- Impact parameter output in 2 bits
  - Since only two bits were available in the output frame that goes to 10Gbps transceivers
- Looking at cosmic ray data we can see high impact parameter muons from cosmics that hit CMS far from the beamline

Going into the future: Towards HL-LHC

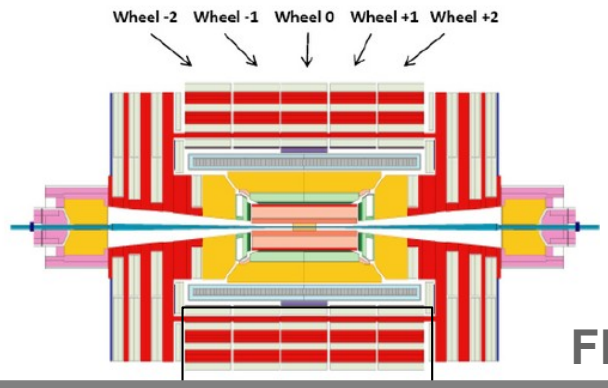
- Current FPGA in CMS → Virtex 7 (28nm)
- New generation FPGAs (Ultrascale+) at 16nm allow more arithmetic operations in the same clock period.
- The current Kalman muon trigger is latency constrained.
  - Extra latency budget in HL-LHC
- How better can we do with modern chips?

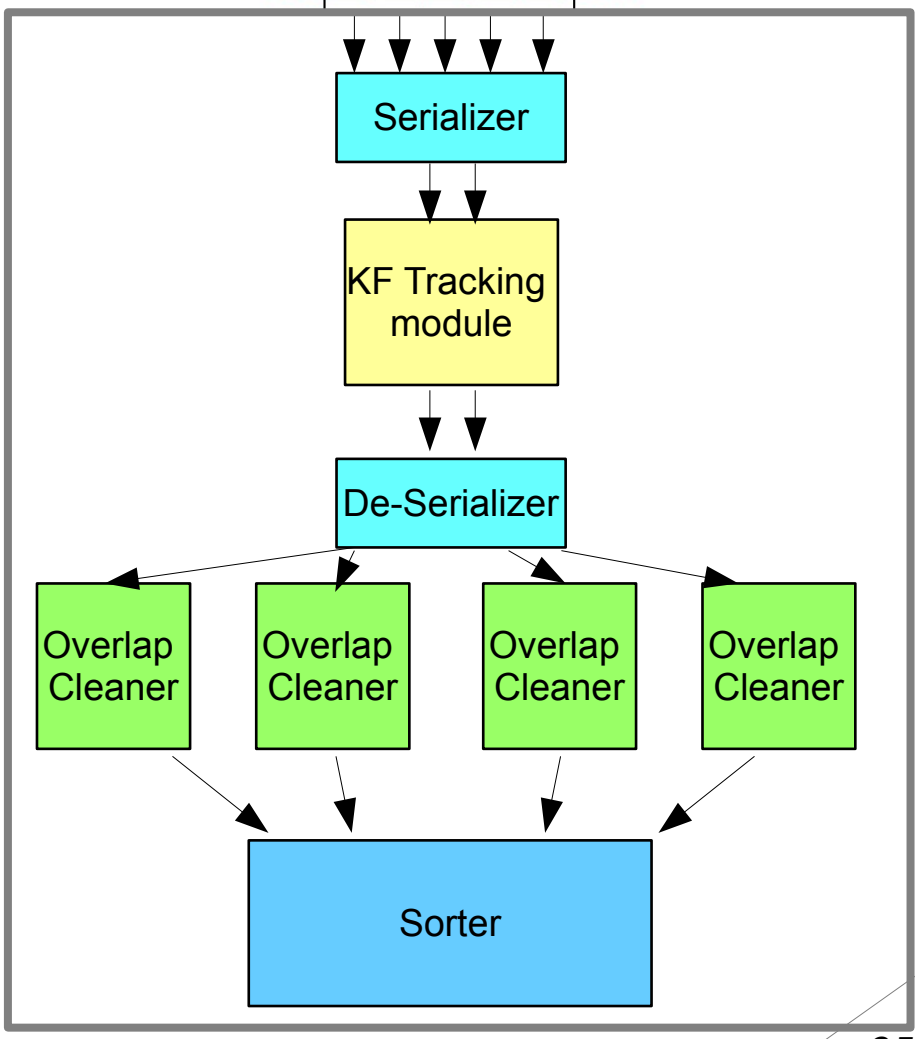
Recall the firmware instantiated in the current chip



Reusing logic

- Logic can be re-used to process a lot of data before the data from the next collision arrive
- In this case, one tracking module is instantiated
  - if the clock speed is at least 5x40MHz = 200 MHz
  - Small latency overhead for (de)serialization
- Using a small low effective modern FPGA (Kintex Ultascale+)
  - 3% of logic elements
  - 8% of DSP
  - Latency of 100ns or 4 BX



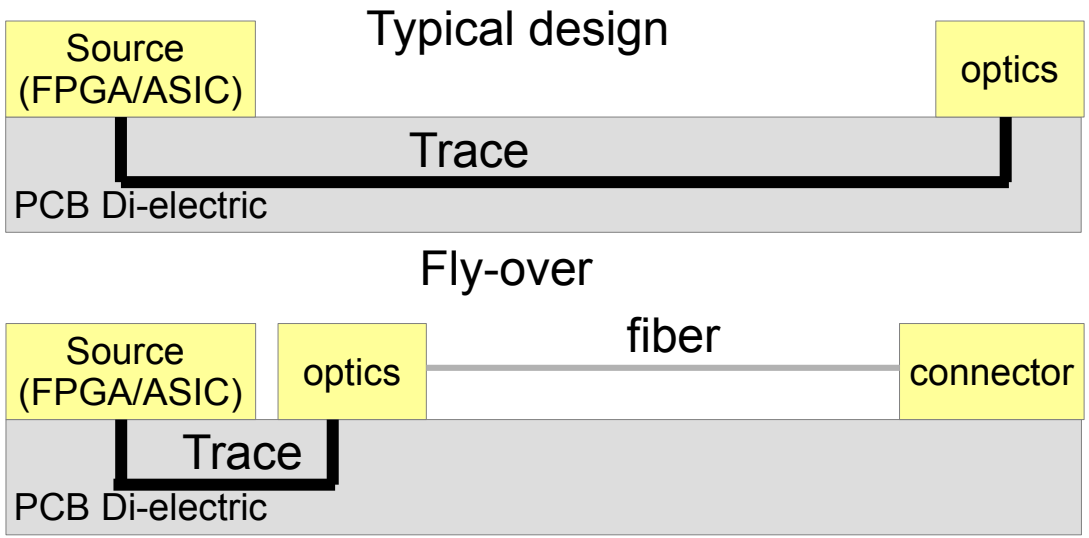


Instrumentation R&D towards HL-LHC

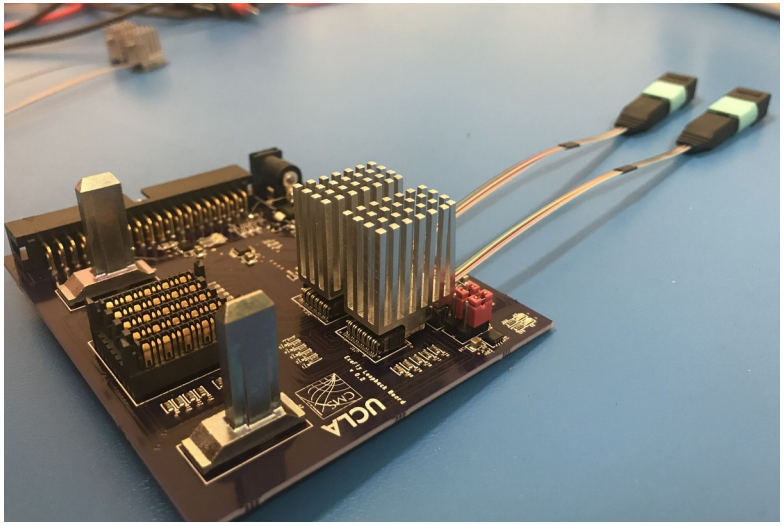
- What we have learned from the Kalman filter exercise:
  - Exploiting DSP cores reduces resource usage by large amount
  - Re-using logic and going to high clock speed results in even lower utilization
  - Given the resource usage in modern FPGAs we can use cost effective chips and still fill only 5-10% of them
    - A Kintex Ultrascale+ (small chip) has a price of 3-4K$
    - A Virtex Ultrascale+ (large chip) has a price of ~ 15K$
- What do we (probably) need for Phase II
  - We have a lot of data to process → We need more or higher speed links
    - Current links in CMS running at 10Gbps
  - We need to exploit as much as possible the latest trends in technology

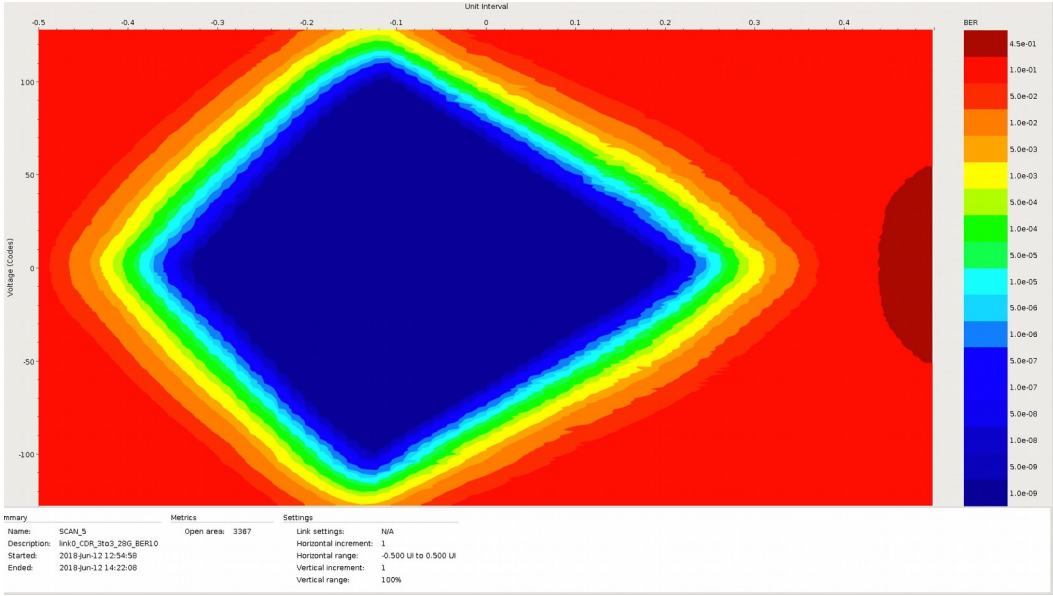
The challenge in high speed data transmission

- Dielectric losses too large at high frequencies (1-3 dB/inch)
- Deploy flyover technology to minimize trace length in PCBs



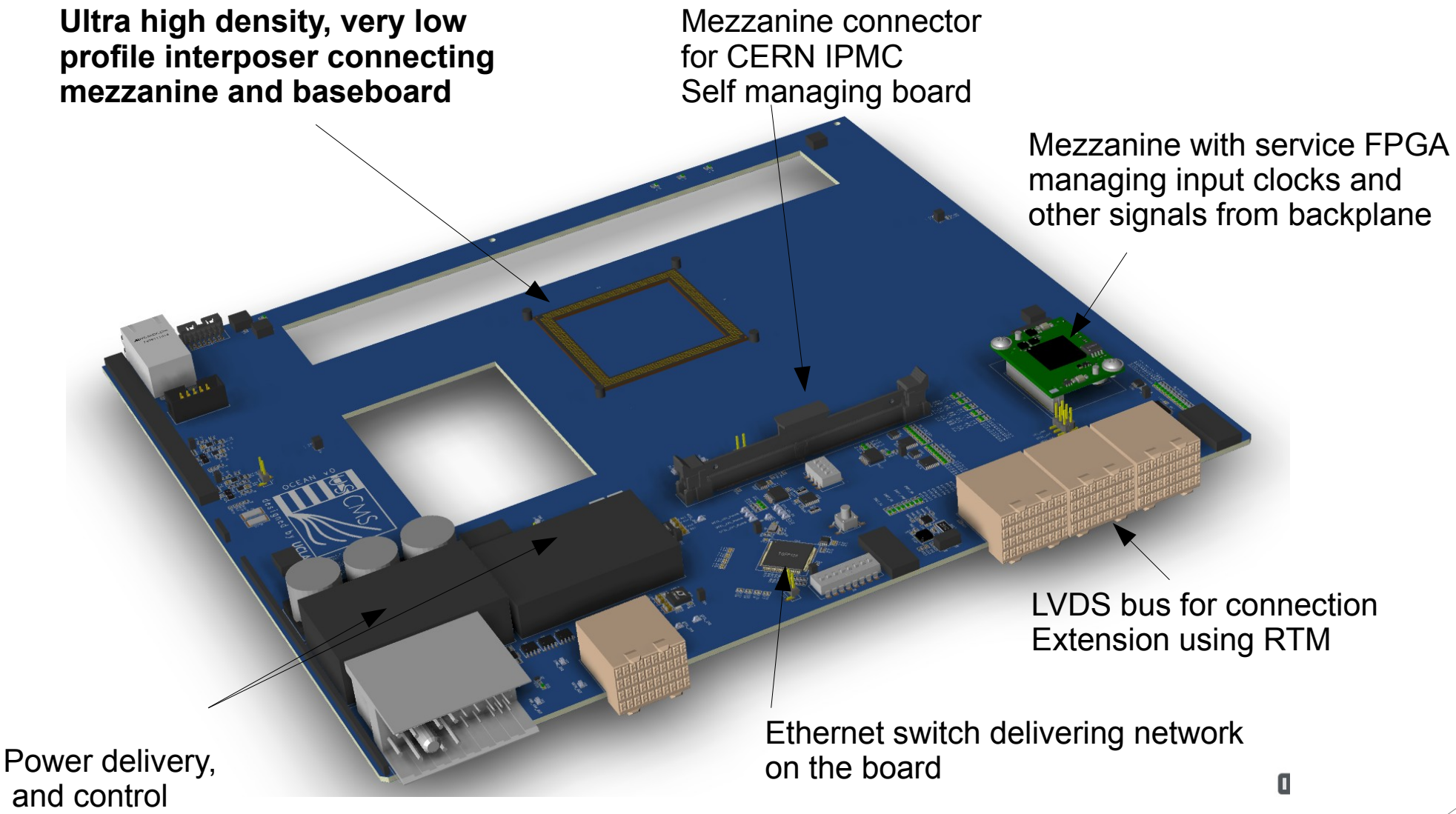
UCLA Test Board with high speed di-electric shows lossless transmission at 28Gbps





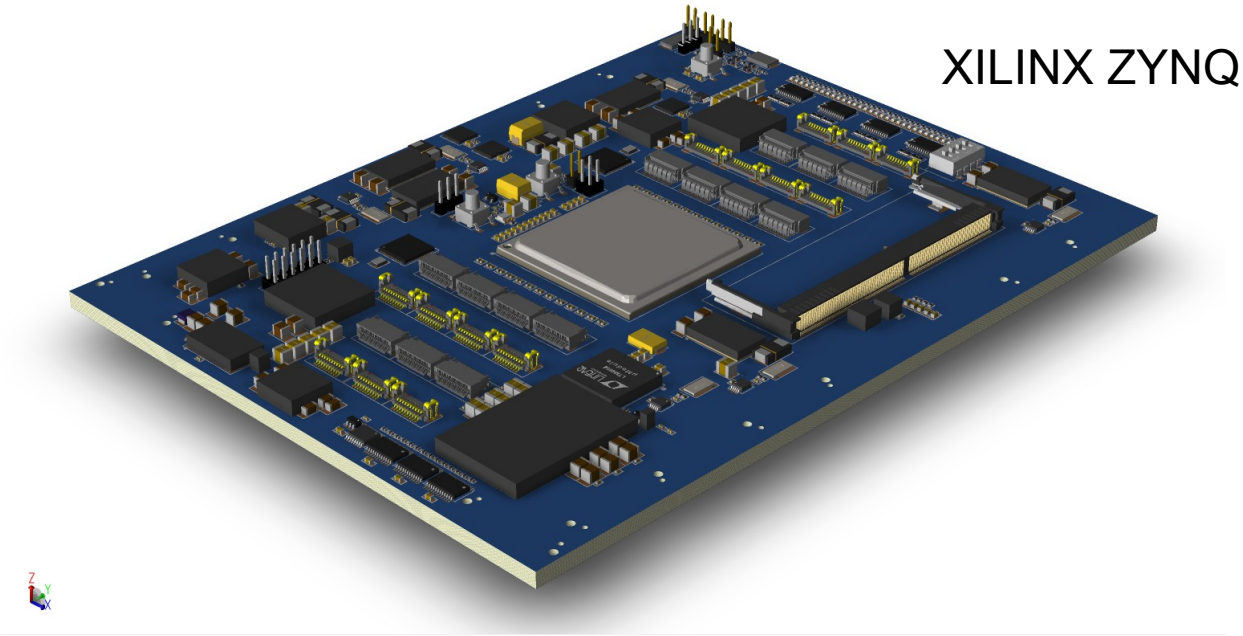
Ocean: A demonstrator board for Phase II

- CMS is moving to the ATCA form factor (28x32cm)
- We are working on a modular design of a baseboard with ATCA services and a mezzanine board with FPGA and high speed fly-over optics



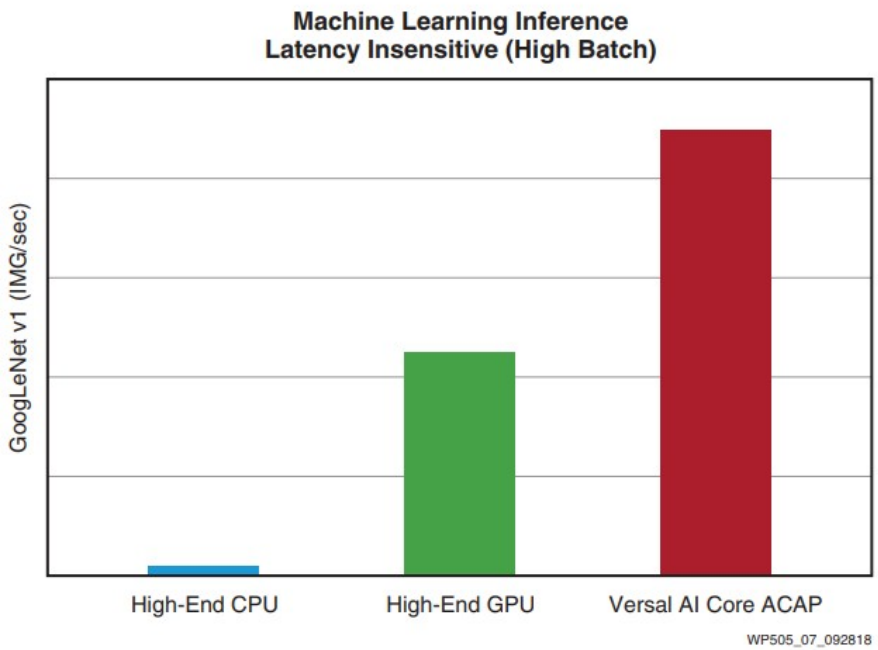
In assembler – Looking forward to the first prototype in the next 10 days

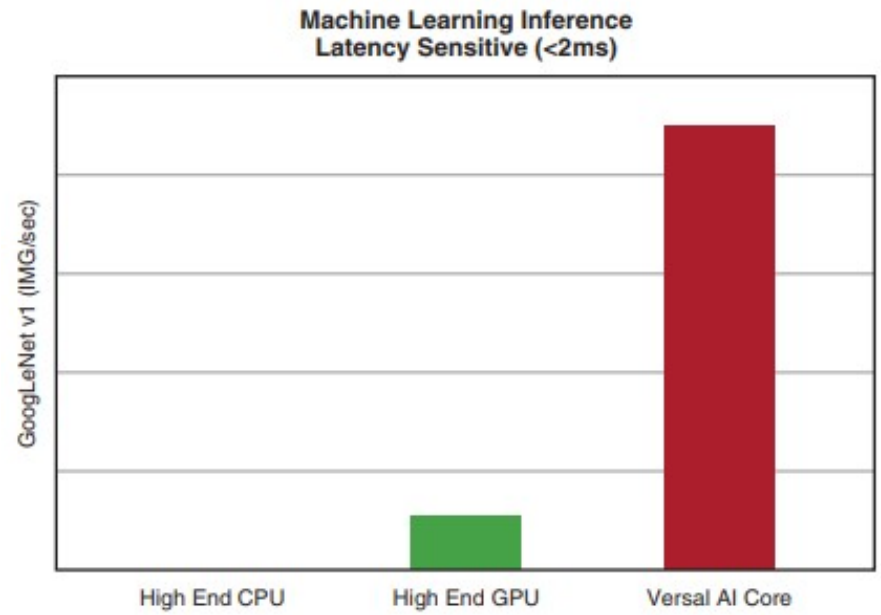
A demonstrator mezzanine with a SoC



- Mezzanine with a System on Chip (SoC) and high speed optics
  - SoC includes FPGA , quad core processor , real time processor and GPU
  - 44 links at 16Gbps and 28 links at 28 Gbps
  - Interconnect between the memory /cache and the FPGA logic
- Possible applications other than conventional trigger implemetations
  - Processor memory can be accessed from the FPGA allowing look up tables and patterns up to 32 GB assisting FPGA logic
  - High speed data streaming to the processor to do data analysis in hardware
  - Heterogeneous computing → Use processor as the main system and FPGA to accelerate software

Where is computing going?





- XILINX just announced their 7nm technology (Versal)
  - Exactly what we are working with but much bigger FPGA logic and links
- Focused on data-center applications/accelerating software
- At 7nm you get much more FPGA logic and many more links at the same package (also links at 56 -112 Gbps)
- **Can we exploit these devices?**

Summary

- A Kalman filter algorithm was implemented for the CMS L1 Barrel Muon Trigger
  - Already commissioned and ready for Run III and HL-LHC
    - Last run was taken last night with the Kalman algorithm triggering by default and sending data to the HLT
  - Very low resource utilization and latency even in current FPGAs
- R&D in new chips and high speed optical links
  - 28 Gbps demonstrated. Waiting for 56 Gbps optics
  - A processor in the main processing FPGA seems to become the standard – the Ocean blade is a first demonstrator
    - We are exited to explore this new world
- There are a lot of intermediate stages between a very low latency system (algorithm in FPGA) and a long latency system (CPU)
  - Heterogeneous computing becomes the standard and we should exploit it in the LHC