Upgrade of the CMS Barrel Muon Track Finder for HL-LHC featuring a Kalman Filter algorithm and an ATCA Host Processor with Ultrascale+ FPGAs

G. Karathanasis on behalf of the CMS Collaboration
Tracking in the CMS barrel muon system

- Four stations: each hit with position ($\phi$) and bending angle ($\phi_b$)
  - 22 bits per station (maximum 88 bits per track)
- Vertex constraint improves resolution
- Momentum measured with a memory LUT using info from two stations
- Can we do better?

G. Karathanasis, TWEPP 2018
- The vertex constraint (applied by default) is sub-optimal for displaced particles
  - Results in mis-measurement of the momentum and loss in trigger efficiency
- Providing a high performance displaced muon trigger extends the CMS physics program!
  - Physics Motivation: Very compressed mass SUSY, Stealth SUSY, numerous exotic models with weak couplings.
A Kalman Filter for BMTF

- Sequential algorithm: (mathematically equivalent to a $\chi^2$ fit)
  - Propagate track from station to station and match with a stub
  - Update track parameters and continue

- After reaching station 1 → save measurement without vertex constraint
  - Except the "standard" track parameters, K-BMTF also gives $p_T$ at 1st station and dxy (Impact parameter)

- Propagate to vertex and update → vertex constrained measurement

G. Karathanasis, TWEPP 2018
Challenge in implementation

- Lots of matrix algebra:

\[ x_n = \begin{pmatrix} k \\ \phi_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ a & b & c \\ d & 0 & d \end{pmatrix} \begin{pmatrix} k \\ \phi_{n-1} \end{pmatrix} \]

- Matrices: \( F, Q, H, \) and \( P, \phi, \) with multiple scattering.

\[ Z_k = \begin{pmatrix} \phi_s \\ \phi_{bs} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} k \\ \phi_b \end{pmatrix} \]

\[ \begin{cases} 
    y_n = z - H x_n \\
    S = HPH^T + R \\
    K = PH^T S^{-1} \\
    x = x_n + K y_n
\end{cases} \]

- Matrix multiplication, matrix inversion
- Latency budget very tight in current system
Matrix algebra in DSP cores

\[ x_n = \begin{pmatrix} k \\ \phi \\ b \end{pmatrix}_n = \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & b \\ c & 0 & d \end{pmatrix} \begin{pmatrix} k \\ \phi \\ b \end{pmatrix}_{n-1} \]

- Map expressions such as \( a \times b \) or \( a \times b + c \) in DSP cores.
- Can be internally pipelined based on clock speed
- Saves large amount of FPGA logic
  - Since multipliers come for free on conventional logic (LUTs, FFs)
Precalculate and lookup the gain

- We perform physics approximation
  - Kalman gain $K$ depends only on the momentum of the track and the track hit pattern in previous stations
  - We precalculate it and store it in a ROM for each track
  - Resolution chosen so that it fits in a single Block RAM of the Virtex 7

- Update of the track parameters $\rightarrow$
  - Implemented in a single DSP core
Parallel processing of combinations

- We create all tracks with at least two stubs (11 tracks total) in parallel from station 4 to station 1 (x2 for two muons per sector)

- We chose the best among tracks with overlapping stubs by an approximate $\chi^2$

- We make all tracks with at least 2 hits → As sensitive as BMTF in single point (chamber) failures
High Level Synthesis

• 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 worrying about the registers

• Example of synthesis: Assume one wants to implement 
\[(a+b*c)*d + e\] and synthesizes with a very slow clock speed

- **Diagram:**
  - Input variables: \(a, b, c, d, e\)
  - Operations: \(x+y*z\)
  - Output: \(x+y*z\)
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 within 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 high speed of FPGA, 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
Deployment in Point 5

- To commission the Kalman filter in parallel, we implemented both algorithms in the same chip.
- The trigger is provided by the legacy system, while the new algorithm is read-out.
  - Study the performance on data
Firmware results for both track finders

- About 60% of the chip
  - Current BMTF, Kalman BMTF, links and infrastructure framework

- Experiencing easy synthesis
  - Good timing closure

- Latency of the full system with the Kalman trigger → 9.5 BX (max. allowed 10.5 BX)
Results from CMS data taking

- Good agreement between data and emulator (over 99%):
  - Agreement helps understand inner workings of firmware
  - Good for addressing discrepancies
  - No bugs in firmware, it does exactly what it should do!
Hardware platforms for Phase II

- Very low utilization for such algorithms after exploiting fully the DSP cores
- Studying platforms for Phase II with cost effective FPGAs
- Effort from Greek groups see S. Mallios poster
- Effort from UCLA: Common ATCA baseboard + Mezzanine with FPGA+optics

ATCA Baseboard in assembler

Mezzanine design in progress
ZYNQ Ultrascale+ 19 EG
44 GTH 16G transceivers
28 GTY 28G transceivers
Summary

• A Kalman muon tracking algorithm was implemented for the CMS Barrel Muon Track Finder
  • Using High level synthesis
  • Use of DSP cores reduces utilization substantially
• Algorithm commissioned in parallel in CMS data
  • Both Phase I algorithm and the new algorithm implemented at the same chip.
  • Utilization of about 60% for both algorithms
• Results from CMS data taking show excellent agreement between data and emulator
  • Proving the whole machinery works
• Several platforms are studied for Phase II using cost effective FPGAs
  • Kintex Ultracale (+) or equivalent

PHASE 2 algorithm already running at CMS!!!
Contributors

U. Athens:
- G.Karathanasis, I Papavergou, P.Sphicas, C.Vellidis

UCLA:
- M.Bachtis, D. Hamilton, T.Lam, S.Regnard

U. Ioannina:
- C.Foudas, P.Katsoulis, S.Mallios
Variables

CMS Preliminary, 2018 Cosmic Data

Number of L1 candidates

emul

data

CMS Preliminary, 2018 Cosmic Data

Number of L1 candidates

emul

data

2D Impact Parameter

G. Karathanasis, TWEPP 2018