

## SEED FILTER & TRACK FITTING

<u>Davide Cieri</u>\* ESR11 VIII INFIERI Workshop @ Fermilab, US • 19 October 2016

\*Supported by the EU FP7-PE0PLE-2012-ITN project nr. 317446, INFIERI, "Intelligent Fast Interconnected and Efficient Devices for Frontier Exploitation in Research and Industry"



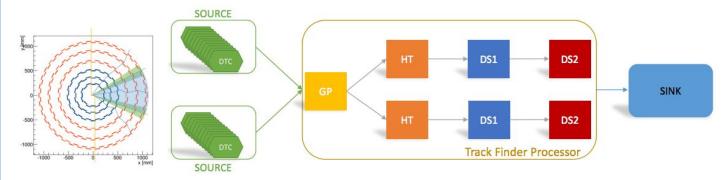






# OUTLINE ##

davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016



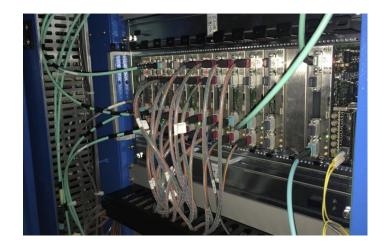
### New Demonstrator 2.0 mapping, to set up the downstream chain

### **Upstream (Luigi's talk):**

- Data Source
- Geographic Processor (GP)
- Hough Transform (HT)

### **Downstream:**

- RZ Seed Filter
- Track Fitter
- Duplicate Track Removal (Luis' Talk)



# DOWNSTREAM OPTIONS:

### **Seed Filter + Fitter**

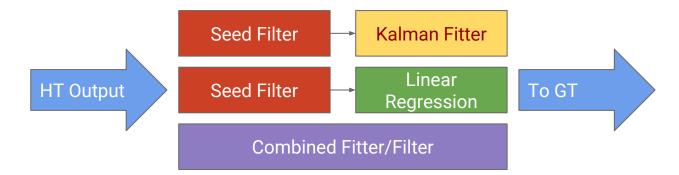
Filtering stage using the z coordinate followed by a fitter (Kalman or Linear Regression)

## **Combinatorial Kalman Fitter**

4 parameters fit, filtering stubs at the same time

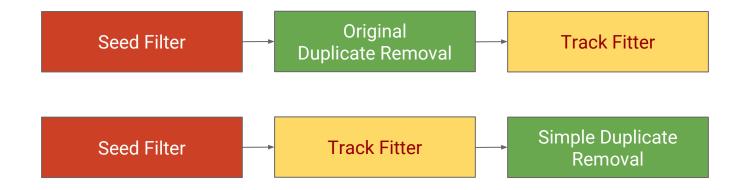
### **Linear Regression Fitter**

Fits a straight line through a 2 dimensional data set by minimizing the squared distances between line and data in one dimension



## **OPTIONS**

- Two possible duplicate removal algorithms can be run in different parts of the chain
  - Original Duplicate Removal: placed after the Seed Filter
  - Simple Duplicate Removal: placed at the end of the chain
- Follow Luis' talk for more details





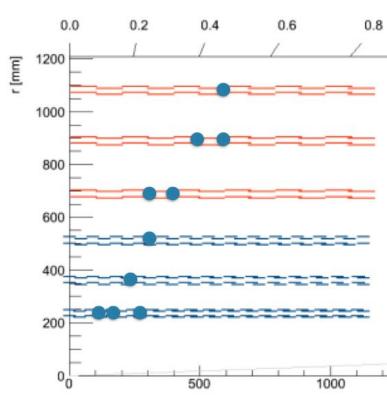
## RZ SEED FILTER

## ALGORITHM OVERVIEW #



davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016 Seed Filter identifies and eliminates stubs assigned to tracks by the HT that lie tens of centimeters from the track in the rz plane

 Collects pairs of stubs, which belong to different PS layers

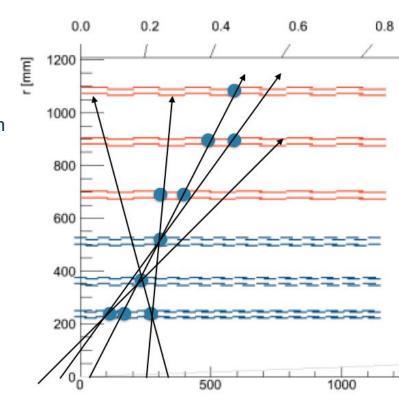


## ALGORITHM OVERVIEW



davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

- 1. Collects pairs of stubs, which belong to different **PS layers**
- Computes lines passing through those stubs (seeds)

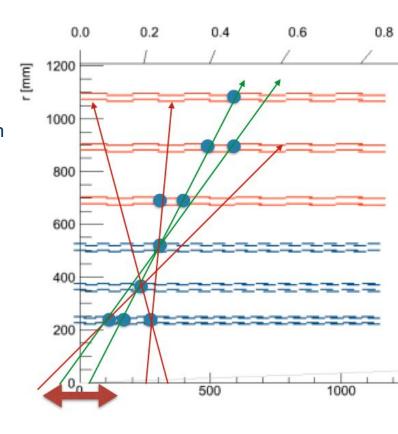


# ALGORITHM

**OVERVIEW** 

davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

- Collects pairs of stubs, which belong to different PS layers
- 2. Computes lines passing through those stubs (**seeds**)
- Discards seeds that would correspond to tracks out of the **beam spot**

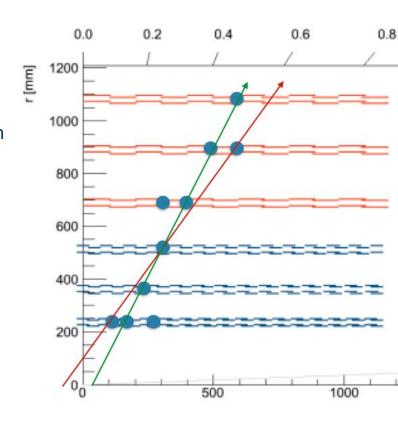


# ALGORITHM

davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

**OVERVIEW** 

- 1. Collects pairs of stubs, which belong to different **PS layers**
- 2. Computes lines passing through those stubs (**seeds**)
- Discards seeds that would correspond to tracks out of the **beam spot**
- Extrapolates surviving seeds to other tracker layers, rejecting stubs not compatible with the line

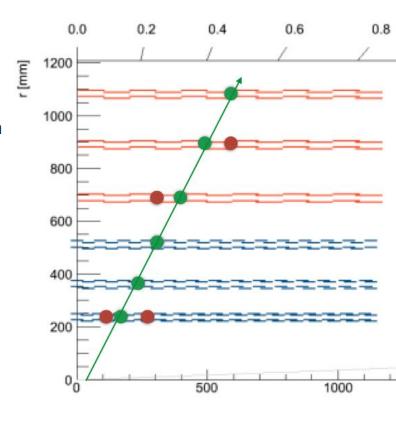


### ALGORITHM OVERVIEW



davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

- Collects pairs of stubs, which belong to different PS layers
- 2. Computes lines passing through those stubs (**seeds**)
- Discards seeds that would correspond to tracks out of the **beam spot**
- Extrapolates surviving seeds to other tracker layers, rejecting stubs not compatible with the line
- Only tracks that still contain enough stubs in different layers are kept

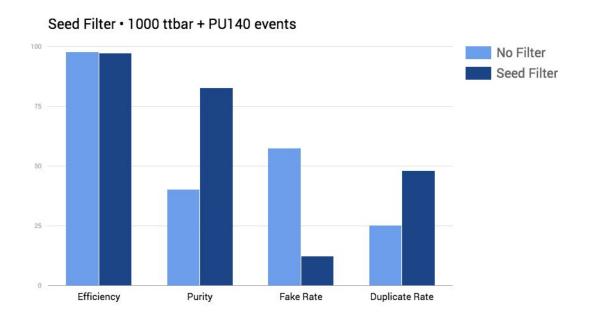


## ALGORITHM RESULTS



### davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

## The Seed Filter reduces the number of track candidates out of the Hough Transform by ~60%



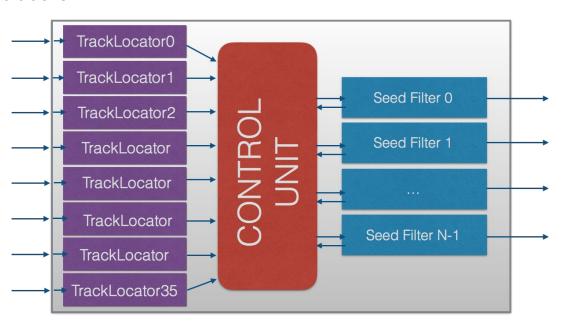
## FIRMWARE DESIGN



davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

- Tracks from HT are firstly stored in the Track Locator
- The Control Unit reads stubs from the track locator and sent them to the first available Seed Filter block
- Seed Filters run the algo and buffer out stubs belonging to survived tracks

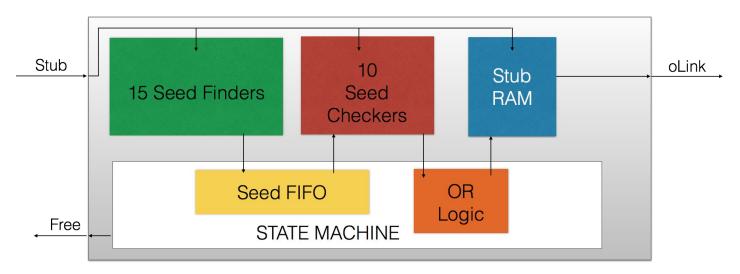
Virtex 7 690
72 optical I/O up to 12.5Gbps
MTCA, total optical b/w 0.9Tbps



## FIRMWARE DESIGN

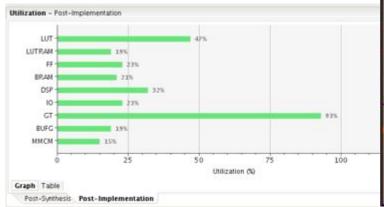


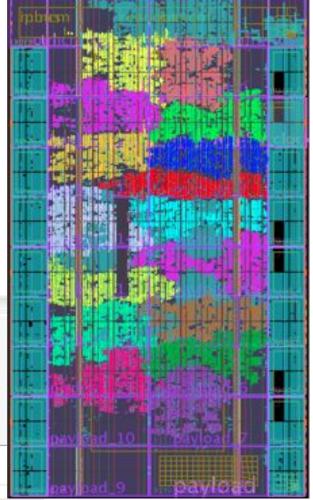
- A **Seed Filter** block processes a track at a time
- Seed Finder finds seeds (up to 15) from pairs of stubs in PS modules
- Using **DSP calculations**, seeds not compatible with the beam spot length are rejected
- Survival seeds (up to 10) are sent to the **Seed Checkers**, where compatibility with the stubs in the other layers is verified
- Stubs associated with good seeds are read out from the Stub RAM



## FIRMWARE RESOURCES

davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016 Seed Filter firmware implementation with 36 track locators and 18 Seed Filter modules

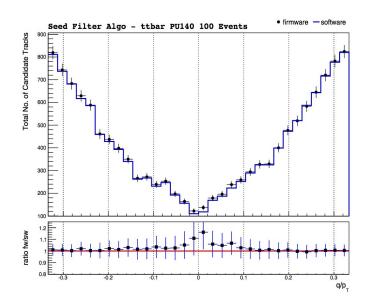


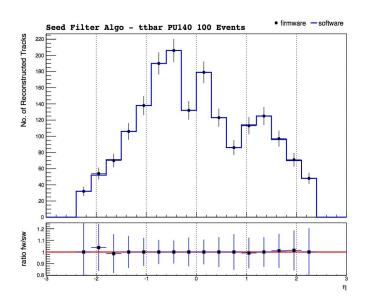


## FIRMWARE RESULTS



- Excellent agreement between tracks produced by seed filter running on MP7 hardware vs. predictions from analysis software (99.8%)
- Average First In First Out Latency = ~292 ns
- Average First Out Last Out Latency = ~439 ns (far below the 875 ns TM period limit)







## TRACK FITTING KALMAN FILTER

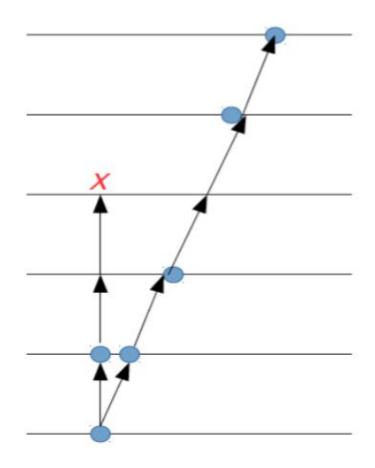
### KALMAN FILTER ALGO



davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

### Kalman Filter Algorithm steps

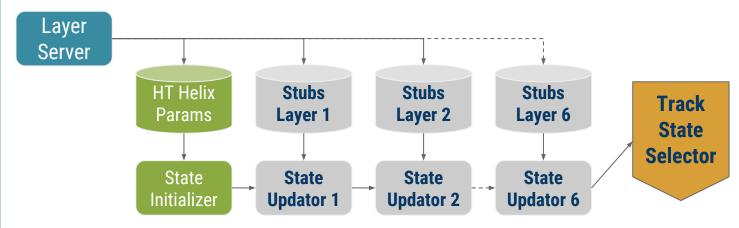
- Initial trajectory estimate from "HT + seed filter" assigned semi-infinite uncertainty
- One stub from each layer in turn is added to trajectory, with helix parameters updated with each new stub
- If several stubs are present in one tracker layer, several `track states' are created, one for each of these stubs
- If stubs are incorrect, the extrapolated `track state' will typically find no subsequent compatible stubs, so be killed



### KF FIRMWARE DESIGN



- Each Kalman Filter block processes data from 6 sectors (6 blocks/octant)
- Layer Server reads HT data from all 6 sectors simultaneously
  - It sends HT estimate of tracks helix params to "HT Helix Params" block
  - Stubs assigned to a track by HT are sent to "Stub Layer N" block if they are in tracker layer N. (where  $N \le 6$ )
- 1. State initializer gets initial estimate from HT Helix Params
- 2. **State Updator** *i* updates parameters using layer *i* stubs
- If a track has more than a single state, Track State Selector choose the best one





## KF FIRMWARE STATUS



davide.cieri@stfc.ac.uk VIII INFIERI Workshop FermiLab • 19/10/2016

- Kalman Filter partially written using a high-level language based on Java
  - MaxCompiler https://www.maxeler.com/products/software/maxcompiler
- Firmware completed expected for Track State Selector
- Resource usage: ~49% DSPs in Virtex 7 FPGA
- Behaviour tested via ModelSim using data from ttbar+PU140 events

Stubin		MIX WX II X X X (5/h00) (5/h00) 0 (13/h0000) (13/h0000) (11/h000) 0 0 0 FALSE FALSE FALSE
■	-No Data-	WILWI ((() (18h00000)
Proto State Creator - Chain 0		
B ♣ Stubin	-No Data-	WILD WILL M. (5 (5.100) (5.100) 0 (13.10000) (13.10000) (11.1000) 0 0 00 FALSE FALSE
■ - StateOut		WMX LIX ( (1shogoo)
State-to-Stub Association - Chain 0 - Layer 0 —		
□ 4 StateIn(0)		WXIW [[[X] 18700000] [18700000] [18700000] [18700000] [18700000] [18700000] [18700000] [18700000] [18700000] [18700000] [18700000]
□ → StateIn(1)		(18h00000)
■ StateOut		{ MODON( ) MONOY() (1shooooo) {1shooooo}
■ StubOut		{ LODOCOCCCC WCC (LX 5:h00) (5:h00) (13:h000) (13:h000) (13:h000) (0 0 0 FALSE FALSE FALSE
- Kalman Filter - Chain 0 - Layer 0		
■		( XXXXXI 18 h00000) (18 h00000)
■ StateOut(0)		(18 h00000) IXXXIIIX XX (18 h00000)
■ ♦ StateOut(1)		(18 hodooo)IXXXXIXXX XXXXXXXXXXXXXXXXXXXXXXXXX
— State-to-Stub Association - Chain 0 - Layer 1 —		
	-No Data-	(18h00000), XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
■	-No Data-	(18 hogos)
	-No Data- -No Data-	T DO [[1] IOU 00000000000000000000000000000000000
Kalman Filter - Chain 0 - Layer 1	-No Data-	1 DC 1813 (1000.000000000000000000000000000000
B → stateIn	-No Data-	( WXX WX (1 ( WX) WXX WXX WXX (1shooco) (1shooco) (1shooco) (1shooco) (1shooco) (1shooco) (1shooco) (1shooco)
StateOut(0)	-No Data-	(18h00000)   18h00000   18h000000   18h0000000   18h000000   18h0000000   18h000000   18h0000000   18h000000   18h0000000   18h00000000   18h00000000   18h00000000   18h00000000   18h
StateOut(1)	-No Data-	(18h00000 W. W. (1 1000) 000000 (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000)
— State-to-Stub Association - Chain 0 - Layer 2 —	110 0000	
□- StateIn(0)		(18h00000, WX W(1, XW(0)WWWWWWWW (18h0000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000) (18h00000)
■  StateIn(1)		(18 hogood, (XXXXI) (1 XXX
StateOut		{18h00000} XXXXI TXIX XXXXIXXX (( XXXXIX 1 XX X (18h0000) {18h00000} (18h00000) {18h00000} (18h00000) {18h00000} (18h00000) {18h00000} (18h00000) {18h00000} (18h00000) {18h00000}
StubOut		(5'h00) (5'h XXXXXXXIII XXXXXXXXXXXXXXXXXXXXXXXX
- Kalman Filter - Chain 0 - Layer 2		
		(18h00000)1
□ ♦ StateOut(0)		{18h00000}{18h00000. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
StateOut(1)		{18hogoo}}{18hogoo}
- State-to-Stub Association - Chain 0 - Layer 3 -		
StateIn(0)		(1shoqoo) (1shoqoo, XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
□ ♦ StateIn(1)		(18h00000, NOT) NOTE: KONTO (000000000000000000000000000000000000
□ ♦ StateOut □ ♦ StubOut	-No Data-	118 hococol ( (118 ho. W (118 ho. W (118 ho. W (118 hococo) (118 ho
	-No Data-	15/1001 15/100 [15/10 [15/10 MULTIN DI 1
Kalman Filter - Chain 0 - Layer 3     stateIn	-No Data-	(18h0000) (1X(18h0X(18:XX)   1   1   1   1   1   1   1   1   1
StateOut(0)	-No Data-	(18 no 0000) (18 no 000) (18 n
StateOut(1)	-No Data-	(18h0000)
State-to-Stub Association - Chain 0 - Layer 4 —	110 00.0	
■ ♣ StateIn(0)		{18h00000} {18h00000}({1sho N{1sh N(10)}))))
■ ◆ StateIn(1)		{18hogoob}{18hogoon, \$\( \) {18ho \$\( \)
StateOut		{12ho0000} {12ho0000} X{12h DXXXXDDDD []) [(X WXXX []) [(14h00000] X [] [X [] 12h00000] (12h00000) (12h00000) (12h00000) (12h00000) (12h00000) (12h00000) (12h00000) (12h00000) (12h00000) (12h000000) (12h0000000) (12h000000) (12h000000) (12h000000) (12h000000) (12h000000) (12h000000) (12h000000) (12h0000000) (12h0000000) (12h000000) (12h0000000) (12h000000) (12h000000) (12h000000) (12h0000000) (12h0000000) (12h0000000) (12h0000000) (12h0000000) (12h0000000) (12h0000000) (12h00000000) (12h0000000) (12h0000000000000) (12h00000000) (12h00000000000000000) (12h000000000000000000000000000000000000
■ StubOut		(5h00) (5h00) 0 (13h00 X(5h0 X(5h0 X(5h0 X(5h0 X(5h00)))) (X X(5h00) (5h X X X X X X X X X X X X X X X X X X
- Kalman Filter - Chain 0 - Layer 4		
■		{1shooooo} {1shooooo} { ](1sh )) [DDD[[00000]] [[ 00000]] [[ 000000] [ 1shooooo} [ 00000] [ 00000] [ 1shooooo] [ 1shoooooo] [ 1shoooooo] [ 1shooooo] [ 1shoooooo] [ 1shoooooo] [ 1shoooooo] [ 1shoooooooooooooooooooooooooooooooooooo
■ StateOut(0)		{18h00000}} {18h00000} {18h00000X{18hXXXXXX} (DXXXXX) } (DXXXXXXX X X X X X X X X X X X X X X X
StateOut(1)		{18h00000}
- State-to-Stub Association - Chain 0 - Layer 5 -		
■ ◆ StateIn(0)		(18h00000)
StateIn(1)  ■       StateIn(1)		(18hogoo)
□ 🍲 StateOut		{1shoqooo} {1shoqooo} {1shoqooo}(fish % (M)(COO)((COO))
■ ◆ StubOut		(5-hoo) (5-hoo) o (13-hooo) (13-hoo X(5-ho X(3-ho X(3-hoo)) X
- Kalman Filter - Chain 0 - Layer 5	No Color	73000000 (1000000
state(n	-No Data-	(18-hogoop) (18-ho
StateOut(0)	-No Data- -No Data-	128 noccool (18 no
D StateOut(1)	-No Data-	=15700000F (18700000F (1870000F (1870000F (18700000F (18700000F (18700000F (18700000F (18700000F (18700000F (1870000F (187000F (1870000F (187000

Estimated Latency ~1 us





## TRACK FITTING LINEAR REGRESSION

### LR ALGO OVERVIEW



- Tracks with sufficient  $p_T$  should draw a straight line on both rφ and rz planes
- Fit helix parameters using **independent** straight line fit in the two planes with **Linear Regression** technique
  - ightharpoonup No use of hit resolution → Simpler math

$$r$$
- $\phi$  plane  $r$ - $z$  plane  $m = \frac{nr\phi - r\phi}{\text{denom}}$   $m = \frac{nrz - rz}{\text{denom}}$   $c = \frac{r^2\phi - rr\phi}{\text{denom}}$   $c = \frac{r^2z - rrz}{\text{denom}}$ 

with denom = 
$$n\overline{r^2} - \overline{r}^2$$
 and  $\overline{x} = \sum_{i=0}^{n-1} x_i$ 

# LR FIRMWARE OVERVIEW #



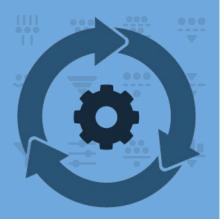
VIII INFIERI Workshop FermiLab • 19/10/2016

- Fully pipelined design, fitting each track in a fixed time
- A Linear Regression segment for each HT segment

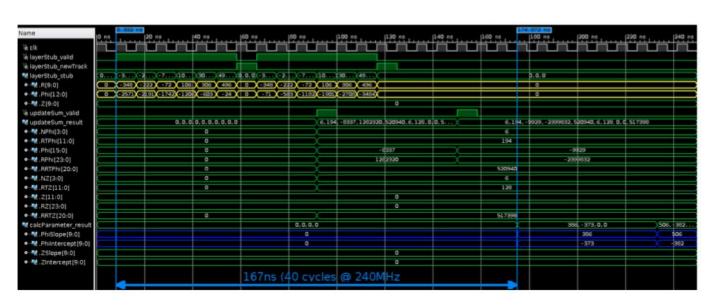


- Several Track Fit iteration in series, each capable to reject a bad stub from the track
- Latency proportional to number of iterations (currently ~1 us)

## LR FIRMWARE STATUS



- Firmware developed in just two months
  - Only missing component : Book Keeper
- Expecting utilisation of 100% DSPs
- Currently tested in modelsim with a set of 6 stubs





### **Seed Filter**

- Firmware completed and running in our demonstrator chain
- Results validated against simulation (99.8% matching)
- Few more ideas to improve latency, dups removal

### Kalman Filter

- Firmware almost completed (Track State Selector missed)
- Behaviour tested in ModelSim with ttbar+PU140 events
- Expected to be integrated in demonstrator chain by end of the month

### **Linear Regression**

- Book Keeper only missing block
- Algorithm tested using a set of 6 sample stubs
- Expected to run on real hardware next week

 Final decision on demonstrator chain layout will be taken after hw estimation of fitters latency



## THANKS!

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n. 317446







Imperial College London









