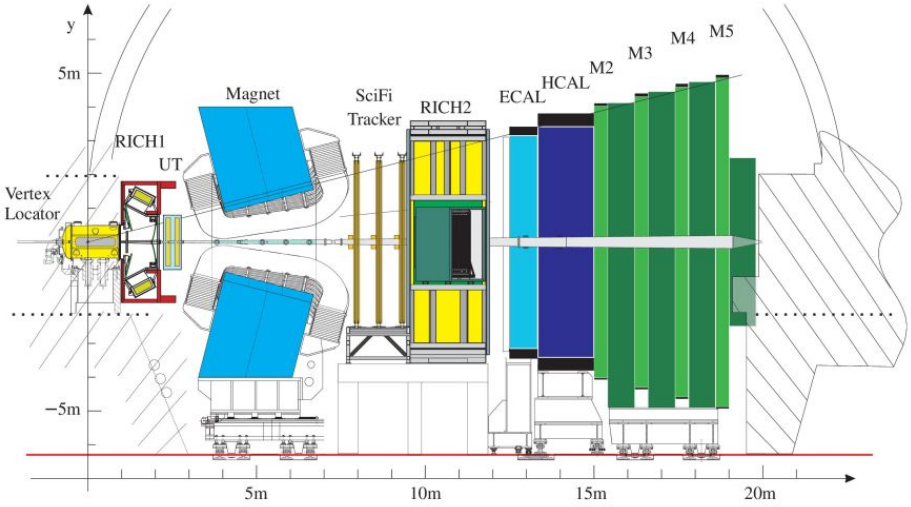
Algoritmo di clustering bidimensionale su FPGA per il rivelatore di vertice (VELO) di LHCb

Giovanni Bassi, Luca Giambastiani, Federico Lazzari,
Michael J. Morello, Tommaso Pajero, Giovanni Punzi

106° Congresso Nazionale SIF - 14-18 Settembre 2020

Contesto: LHCb-Upgrade

- LHCb sta attualmente “aggiornando” il rivelatore: **LHCb-Upgrade**.
- LHCb-Upgrade raccoglierà dati durante Run 3 (23 fb^{-1}) e Run 4 (50 fb^{-1}) di LHC a una **luminosità** di $2x10^{33}$ $cm^{-2}s^{-1}$, **×5 rispetto ai precedenti Run** (Run 1 e 2, ≈ 9 fb^{-1}).
- Vertici/evento <v> = 7.6 (1.1 nel Run 2).
- Tutti i **rivelatori letti a 40 MHz** (rate collisioni dei bunch in LHC). **Trigger** di livello zero (**L0**), che riduceva il rate di lettura dei rivelatori a 1 MHz, **non è più usato**.
- **Flusso di dati da 40 Tb/s gestito dall’*High Level Trigger*** (HLT).



Il progetto RETINA

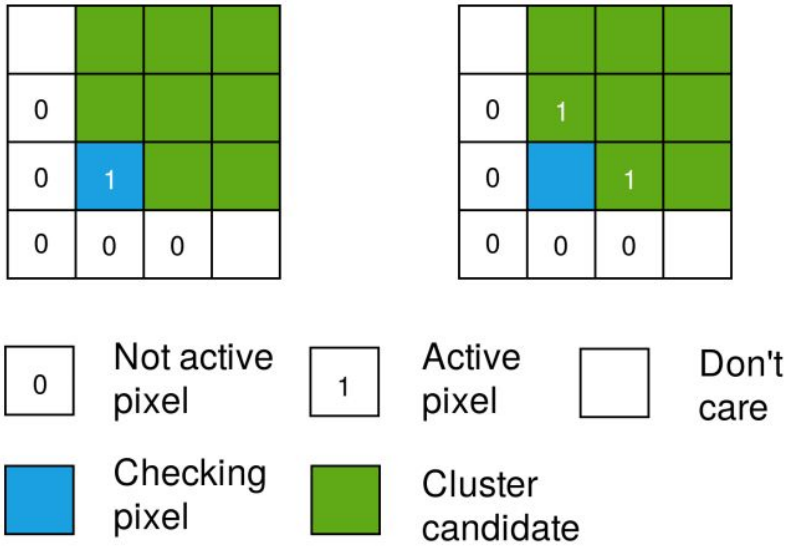
- Crescita delle risorse necessarie alla gestione dei dati maggiore rispetto alle risorse disponibili sul mercato: necessità di **processare i dati in tempo reale** al livello più basso per ridurne il flusso.
- L'uso di **acceleratori** (tra cui FPGA e GPU), dispositivi specializzati nell'esecuzione veloce di compiti altamente parallelizzabili, è già attualmente considerato una **soluzione praticabile ed economica**.
- Progetto **LHCb-RETINA**: R&D per il **tracking su FPGA in tempo reale a 30 MHz**. Obiettivo: installazione tracking per il rivelatore di vertice (VELO) per la presa dati del Run 4 (2027-30).

Il clustering RETINA

- Il tracking richiede l'uso di cluster pre-elaborati: necessità di un algoritmo di **cluster-finding che processi i dati prima di essere elaborati dall'acceleratore**.
- Obiettivo: eseguire il **clustering del VELO sugli FPGA delle schede di readout** di LHCb (PCIe40) già nel Run 3.
- Tale sistema di clustering è anche una prima prova di un approccio in cui si fanno **elaborazioni sofisticate dei dati a un livello molto basso del DAQ, già nel readout**.
- Aumento del **throughput di HLT** stimato: **+8%** per nodo della HLT farm. Riduzione di **banda** stimata: **-20%**.
- Un algoritmo di clustering per FPGA deve essere strutturato e scritto in maniera intrinsecamente diversa da un algoritmo sequenziale per CPU.
- **Sviluppato** interamente **in VHDL** per limitare l'uso di **risorse**, massimizzando al contempo **parallelismo e throughput**.

L’algoritmo di clustering

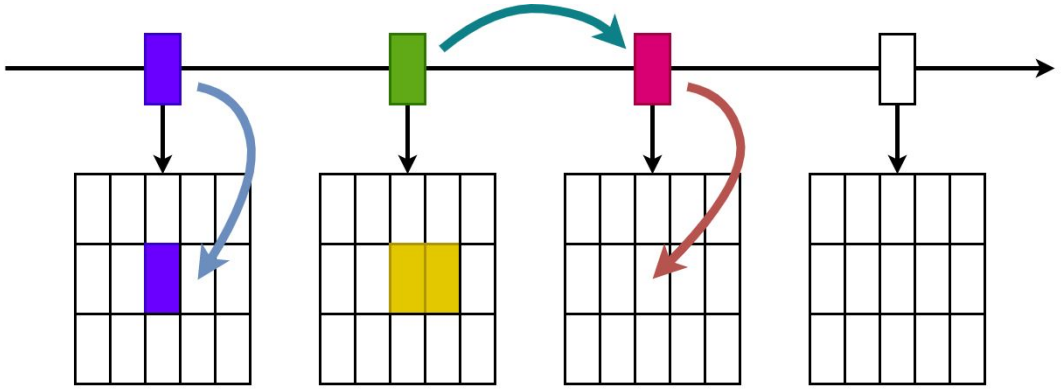
- **VELO**: rivelatore a **pixel di silicio**. 208 sensori ognuno formato da 256 × 768 pixel. Circa **40 milioni di pixel in totale**.
- L’algoritmo **ricerca i cluster tramite pattern di pixel** pre-determinati (*L-pattern*). Ogni pixel esegue la ricerca in parallelo con gli altri.
- In caso di “**matching**” il **cluster** considerato è **contenuto nella corrispondente sotto-matrice 3×3**.
- Il **centro** viene **calcolato con una look-up-table** (LUT) con una precisione di ⅛ di pixel.



L’algoritmo di clustering

- Pixel analizzati tramite una **catena di 20 matrici che mappano un intero sensore**. La posizione di una matrice all’interno del sensore è determinata dai primi pixel che vi entrano. Ogni matrice viene riempita solo con pixel adiacenti.

- Blu pixel appartengono alla matrice: la riempiono.
- Verde pixel non appartengono alla matrice: vanno avanti nella catena.
- Rosso pixel che non sono entrati in nessuna matrice riempiono la prima vuota.



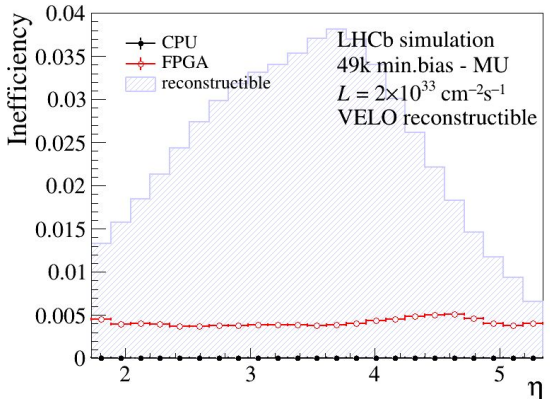
- Algoritmo implementato e testato su una scheda di prototipazione. Raggiunto **throughput di 38.9 MHz**, oltre i 30 MHz minimi richiesti.
- Utilizza il **21% della logica** dell’FPGA.

Studio prestazioni di fisica

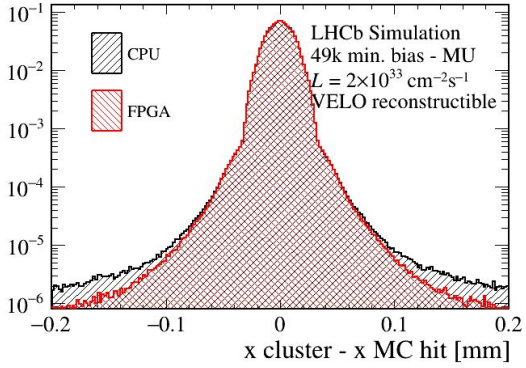
- Necessari **studi sulle prestazioni di fisica** dell'algoritmo FPGA per verificare la qualità dei cluster prodotti **rispetto all'algoritmo** baseline (per **CPU**).
- Utilizzata la simulazione di LHCb a cui è stato aggiunto un "emulatore" software (C++) del clustering firmware che simula tutte le sue caratteristiche di basso livello.
- Quantità analizzate:
    - Efficienza e qualità cluster ricostruiti;
    - Efficienza e qualità tracking di HLT a partire dai cluster;
    - Efficienza e qualità ricostruzione vertici primari;
    - Risoluzione impulso e IP.

Efficienza di clustering

- Efficienza: $\frac{N_{linked\_MC\_clu}}{N_{MC\_clu}}$
- Efficienze vicine al 100% per entrambi gli algoritmi. **Inefficienza media** dell'algoritmo FPGA **sotto 0.5%**.

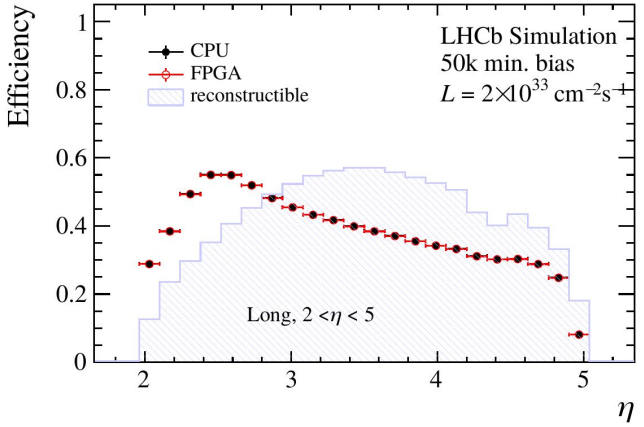


- Residui posizione cluster ricostruiti: $x_{clu} - x_{MC}$
- Non si notano bias.
- **Risoluzione** di circa **11.5 µm** per entrambi gli algoritmi (dimensione pixel 55x55 **µm²**).

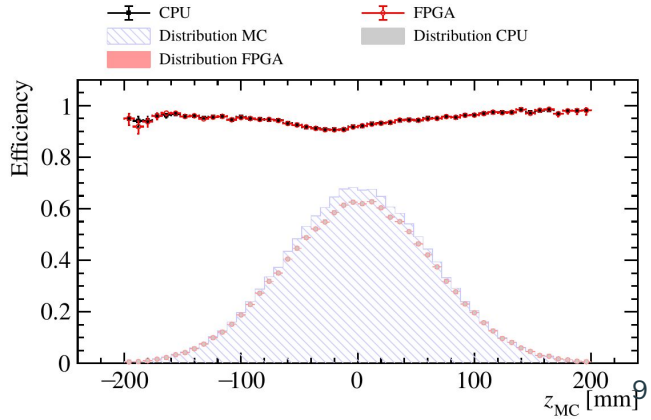


Tracking e PV

- Efficienza tracking: $\frac{N_{MC\_matched}}{N_{MC\_reconstructible}}$
- Le **efficienze di tracking** ottenute dai cluster FPGA e CPU sono **equivalenti**.



- Efficienza ricostruzione PV: $\frac{N_{MC\_matched}}{N_{MC\_reconstructible}}$
- Le **efficienze e risoluzioni nella ricostruzione dei vertici primari** ottenute dai cluster FPGA e CPU sono praticamente **indistinguibili**.



Conclusioni

- **Prima implementazione** di un algoritmo (altamente parallelo) di **clustering 2D su FPGA** capace di processare collisioni di LHC a velocità superiori a **30 MHz**.
- **Prestazioni di fisica equivalenti** a quelle dell'**algoritmo** baseline su **CPU**.
- La realizzazione di questo progetto rappresenta un primo salto nel futuro verso un **utilizzo sempre maggiore di calcolo eterogeneo con FPGA** in HEP:
    - dimostra che un elaborazione eterogenea su FPGA è sia vantaggiosa che possibile in pratica.
    - rende possibile un tracking in tempo reale nel futuro.