Mu2e II Workshop TDAQ report

2020-12-09

G. Pezzullo (Yale), A. Gioiosa (Pisa)

Intro

- Updates from the GPU front
- Updates from the HLS study group
- Proposal for a new FPGA algorithm (from Jin-Yuan Wu)

Updates from the GPU front

- A large amount of literature was shared by
- Antonio and Giani chatted with Lamanna (Pisa) and he gave us good tips for starting
- He pointed us a tool that allows to estimate performance gain from parallelization with minimal changes to the code: https://www.openacc.org/tools
- We want to try using the KinKal package for testing it: https://github.com/KFTrack/KinKal

HLS updates

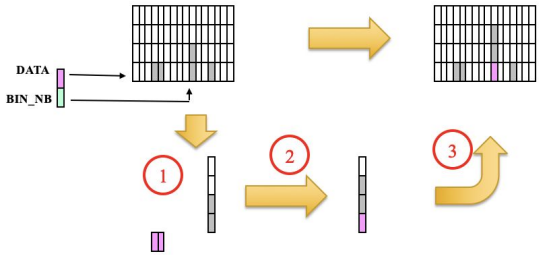
- Richie and Giani chatted Ryan who helped us quite a bit:
  - A lot of material
  - Platform where to work

- Now we need to coordinate how to split the development of the first reco algorithms

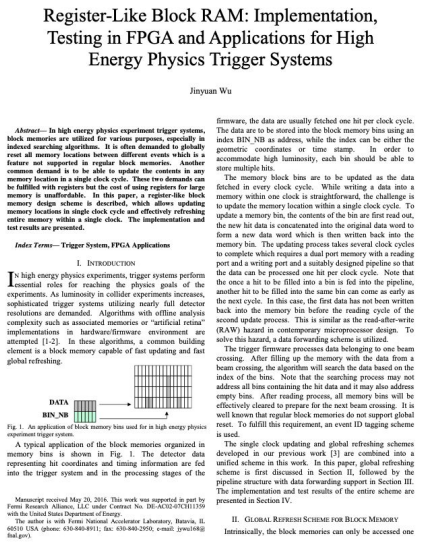
FPGA building blocks

- A resource saving scheme to implement large number of updatable cells. (e.g. in event buffer, clustering, histogram..)
- Several building blocks useful for tracking trigger implemented in FPGA can be used to implement track seeding engine for Mu2e-II
- References: Ref1, Ref2

Register-Like Block RAM



- The register-like block RAM use regular RAM in FPGA to implement large number updatable memory locations.
- It can be used for: event buffer, clustering, histogram, Retina cells, Hough Transform cells.



Register-Like Block RAM: Implementation, Testing in FPGA and Applications for High Energy Physics Trigger Systems

Jinyuan Wu

Abstract— **In high energy physics experiment trigger systems, block memories are utilized for various purposes, especially in indexed searching algorithms. It is often demanded to globally reset all memory locations between different events which is a feature not supported in regular block memories. Another common demand is to be able to update the contents in any memory location in a single clock cycle. These two demands can be fulfilled with registers but the cost of using registers for large memory is unaffordable. In this paper, a register-like block memory design scheme is described, which allows updating memory locations in single clock cycle and effectively refreshing entire memory within a single clock. The implementation and test results are presented.**

Index Terms— **Trigger System, FPGA Applications**

I. Introduction

In high energy physics experiments, trigger systems perform essential roles for reaching the physics goals of the experiments. As luminosity in collider experiments increases, sophisticated trigger systems utilizing nearly full detector resolutions are demanded. Algorithms with offline analysis complexity such as associated memories or "artificial retina" implementations in hardware/firmware environment are attempted [1-2]. In these algorithms, a common building element is a block memory capable of fast updating and fast global refreshing.

Fig. 1. An application of block memory bins used for in high energy physics experiment trigger system.

A typical application of the block memories organized in memory bins is shown in Fig. 1. The detector data representing hit coordinates and timing information are fed into the trigger system and in the processing stages of the firmware, the data are usually fetched one hit per clock cycle. The data are to be stored into the block memory bins using an index BIN_NB as address, while the index can be either the geometric coordinates or time stamp. In order to accommodate high luminosity, each bin should be able to store multiple hits.

The memory block bins are to be updated as the data fetched in every clock cycle. While writing a data into a memory within one clock is straightforward, the challenge is to update the memory location within a single clock cycle. To update a memory bin, the contents of the bin are first read out, the new hit data is concatenated into the original data word to form a new data word which is then written back into the memory bin. The updating process takes several clock cycles to complete which requires a dual port memory with a reading port and a writing port and a suitably designed pipeline so that the data can be processed one hit per clock cycle. Note that the once a hit to be filled into a bin is fed into the pipeline, another hit to be filled into the same bin can come as early as the next cycle. In this case, the first data has not been written back into the memory bin before the reading cycle of the second update process. This is similar as the read-after-write (RAW) hazard in contemporary microprocessor design. To solve this hazard, a data forwarding scheme is utilized.

The trigger firmware processes data belonging to one beam crossing. After filling up the memory with the data from a beam crossing, the algorithm will search the data based on the index of the bins. Note that the searching process may not address all bins containing the hit data and it may also address empty bins. After reading process, all memory bins will be effectively cleared to prepare for the next beam crossing. It is well known that regular block memories do not support global reset. To fulfill this requirement, an event ID tagging scheme is used.

The single clock updating and global refreshing schemes developed in our previous work [3] are combined into a unified scheme in this work. In this paper, global refreshing scheme is first discussed in Section II, followed by the pipeline structure with data forwarding support in Section III. The implementation and test results of the entire scheme are presented in Section IV.

II. Global Refresh Scheme for Block Memory

Intrinsically, the block memories can only be accessed one

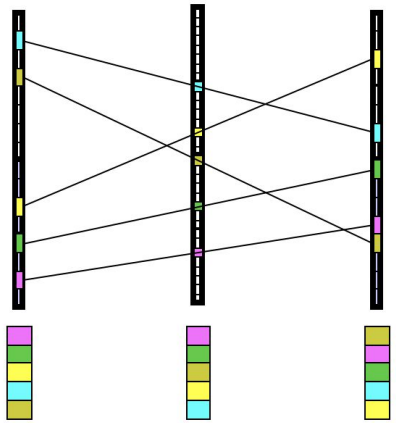
Manuscript received May 20, 2016. This work was supported in part by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

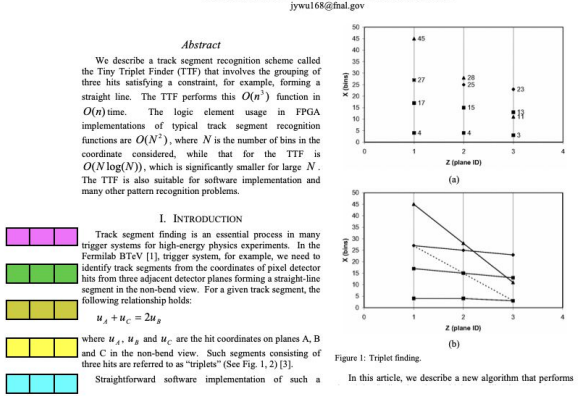
The author is with Fermi National Accelerator Laboratory, Batavia, IL 60510 USA (phone: 630-840-4911; fax: 630-840-2950; e-mail: jywu168@fnal.gov).

The tiny triplet finder

- Single clock matching three or more hits with two free parameters. (e.g., tracks in r-z plane, tracks in r-phi with small impact parameters)
- Tiny Triplet Finder can be used if the hit multiplicity per plane is not very high

The Tiny Triplet Finder





Tiny Triplet Finder (TTF) – A track segment recognition scheme and its FPGA implementation developed in the BTeV level 1 trigger system

Jinyuan Wu, Z. Shi, M. Wang, H. Garcia and E. Gottschalk

Fermi National Accelerator Laboratory, Batavia, IL 60510, USA
jywu168@fnal.gov

Abstract

We describe a track segment recognition scheme called the Tiny Triplet Finder (TTF) that involves the grouping of three hits satisfying a constraint, for example, forming a straight line. The TTF performs this $O(n^2)$ function in $O(n)$ time. The logic element usage in FPGA implementations of typical track segment recognition functions are $O(N^2)$, where N is the number of bins in the coordinate considered, while that for the TTF is $O(N\log(N))$, which is significantly smaller for large N. The TTF is also suitable for software implementation and many other pattern recognition problems.

I. Introduction

Track segment finding is an essential process in many trigger systems for high-energy physics experiments. In the Fermilab BTeV [1], trigger system, for example, we need to identify track segments from the coordinates of pixel detector hits from three adjacent detector planes forming a straight-line segment in the non-bend view. For a given track segment, the following relationship holds:

$$u_A + u_C = 2u_B$$

where u_A, u_B and u_C are the hit coordinates on planes A, B and C in the non-bend view. Such segments consisting of three hits are referred to as "triplets" (See Fig. 1, 2) [3].

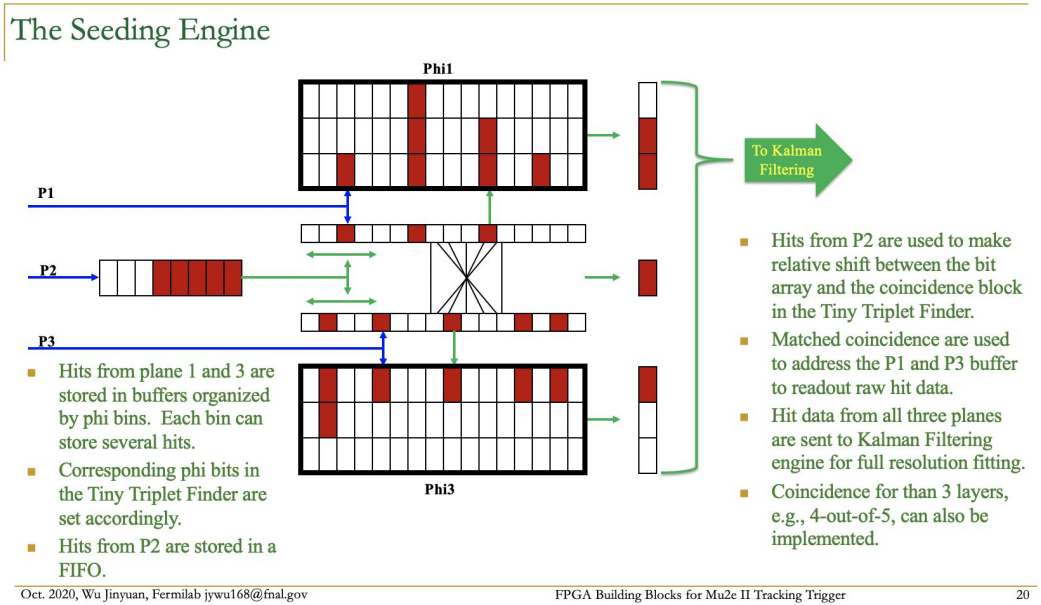
Straightforward software implementation of such a

Figure 1: Triplet finding.

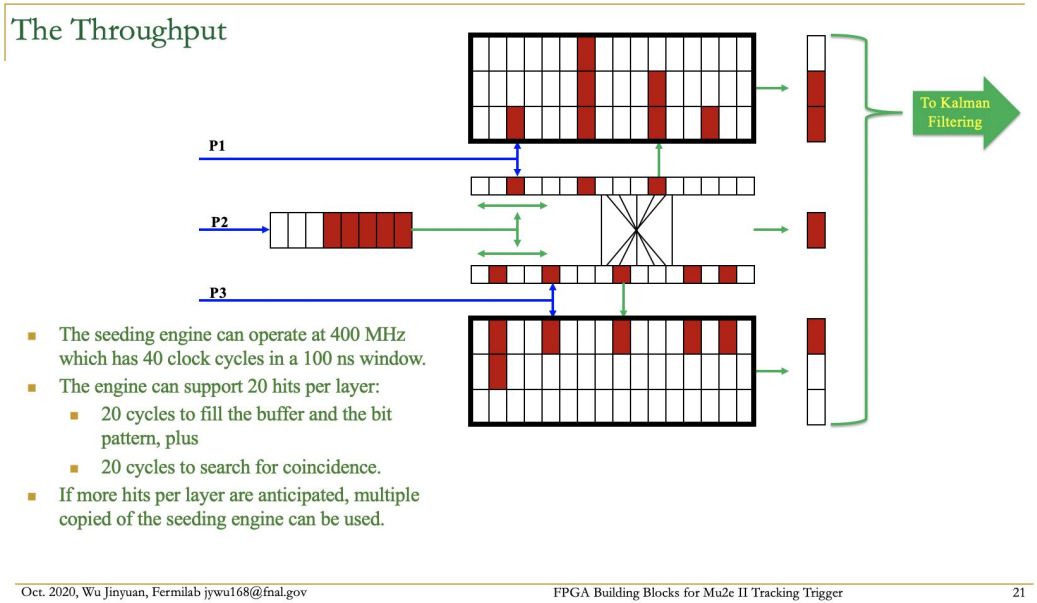
In this article, we describe a new algorithm that performs

- The triplets are group of at least 3 (but can be more than 3) hits that satisfy the first constraint.
- Triplet finder reorganize hits for further track fitting.
- The Tiny Triplet Finder is a resource saving scheme for triplet finding.

A track-seed engine for Mu2e-II



A track-seed engine for Mu2e-II



Summary

- We are exploring several options to perform the track pattern recognition
    - FPGA/HLS
    - GPU
    - FPGA (tiny triplet finder)
- We need to provide some simulated data to Yuan for making some benchmark studies
- We are planning a new TDAQ Mu2e-II meeting in January 2021, after winter brake
- We will plan a new workshop when we will have some result from GPU, HLS studies, and when we will have a new tdaq schema proposed for CRV