FERMILAB-PUB-08-124-E

FPGA Curved Track Fitters and a Multiplierless Fitter Scheme

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

***Abstract*—The standard least-squares curved track fitting process is tailored for FPGA implementation so that only integer multiplications and additions are needed. To further eliminate multiplication, coefficients in the fitting matrices are carefully chosen so that only shift and accumulation operations are used in the process. Comparison in an example application shows that the fitting errors of the multiplierless implementation are less than 4% bigger than the fitting errors of the exact least-squares algorithm. The implementation is suitable for low-cost, low-power applications in high energy physics detector trigger systems.**

***Index Terms*—Trigger, Track Fitting, FPGA Firmware, FPGA Computing**

I. Introduction

In high-energy physics experiments, track fitting is traditionally considered a software task implemented in the higher level trigger and analysis stages. It is now possible to do track fitting in FPGA based lower level trigger systems. A successful example of track finding and fitting in high rate hadron collider trigger applications is the Fermilab CDF Silicon Vertex Trigger system [1-3].

Although implementing a fitting algorithm in today's large FPGA's is possible, without the judicious use of resources, cost and power consumption quickly become major concerns. Power-hungry operations requiring significant silicon resources, like the multiplications and divisions in many algorithms, can be replaced by less resource-intensive operations such as shifts, additions and subtractions. Such replacements can dramatically reduce the number of FPGA logic elements and thereby power consumption, justifying a minimal loss in precision.

In this paper, we describe a curved track fitter suitable for FPGA implementation that uses only integer multiplications and additions. The fitter is based on a standard least-squares algorithm with modifications to the matrix coefficients. A multiplierless version of the fitter is also discussed in which only shift and addition operations are used. The fitter is designed to match the data fetching speed so that it can be used to process a flowing data stream in trigger and DAQ systems.

Our FPGA curved track fitter was originally developed for the level 1 pixel trigger of the Fermilab BTeV experiment [4][5]. The pixel detector consists of measurement planes as shown in Fig. 1.

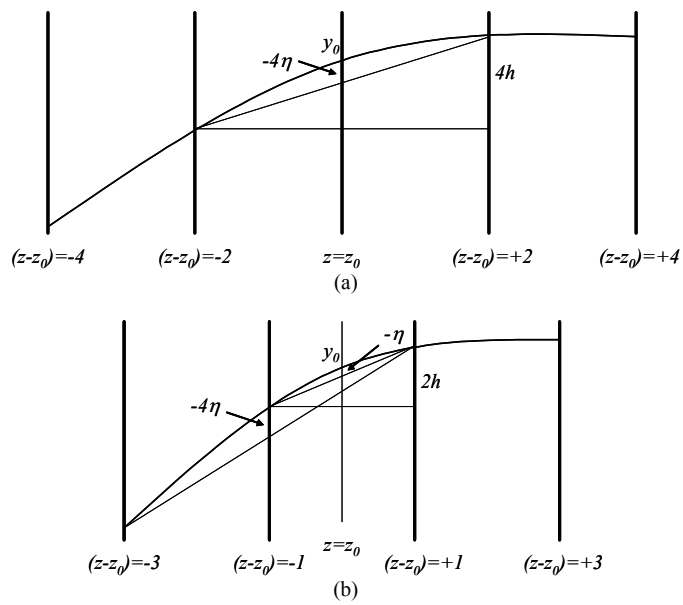

Fig. 1. Tracks in the BTeV detector: (a) The configuration for tracks with an odd number of hits. (b) The configuration for tracks with an even number of hits.

Raw hit data from the pixel planes are first sent through several stages for time stamp ordering, combining into hit clusters, and generating hit coordinates before sending to FPGA based segment trackers. Possible detector misalignments are also corrected while generating the coordinates. In the segment tracker, hits from three neighboring planes are grouped into inner and outer triplets representing, respectively, the beginning and ending segments of complete tracks. One possible implementation of the triplet finding algorithm is the "Tiny Triplet Finder" (TTF)[6]. Once the inner and outer triplets are found, they are then matched to form complete tracks in order to identify all hit coordinates belonging to a track. This complete set of coordinates can then be used by the track fitter to estimate the track parameters. In the original baseline design of the BTeV trigger system, the track fitting is done in a CPU-based higher level trigger farm, partially because the hits from the full detector were not available until reaching the CPU farm. With the latest

Manuscript received August 31, 2007, revised February 18, 2008. This work was supported in part by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

J. Wu, M. Wang, E. Gottschalk and Z. Shi are with Fermi National Accelerator Laboratory, Batavia, IL 60510 USA (phone: 630-840-8911; fax: 630-840-2950; e-mail: jywu168@ fnal.gov).

proposed change to the baseline architecture [7], events are built parasitically in the early stages. Hits from the full detector are, therefore, available to the FPGA segment tracker, making it possible to perform track fitting at this stage of the trigger.

Although the FPGA fitter described in this paper was originally developed for a low level trigger system, it can also find application in higher level trigger and analysis stages. One could easily imagine its implementation in an FPGA based reconfigurable co-processor. Such a fast and fairly precise track fitter occupying a small footprint will be of great value to applications requiring the fast and efficient filtering of interesting events.

In Appendix A, we discuss why the errors produced by the approximations in the multiplierless algorithm are negligible. According to Equation (A14) derived from Theorem 1, the multiplierless algorithm or other approximations can be used in any function-fitting processes based on least-squares method without increasing the fitting errors significantly.

II. Principle

A. Computations for Track Fitting

In the detector with magnetic field along x-axis, a track is projected to both the non-bend and bend views and its equations can be written approximately:

$$\begin{aligned} x &= x_0 + l(z - z_0) \\ y &= y_0 + h(z - z_0) + \eta(z - z_0)^2 \end{aligned} \tag{1}$$

A track has either an odd or an even number of hits. The center of the track is chosen with $z=z_0$ as shown in Fig. 1. The parameters x_0 and y_0 are offsets of the track and l and h are slopes at track center. The parameter η represents the track momentum. With a set of coordinate measurements x_i and y_i, the parameters of the tracks can be found with the following linear combinations.

$$\begin{aligned} x_0 &= \sum_i a_i x_i / \sum_i a_i \quad l = \sum_i b_i x_i / \sum_i b_i (z_i - z_0) \\ y_0 &= \sum_i c_i y_i / \sum_i c_i \quad h = \sum_i d_i y_i / \sum_i d_i (z_i - z_0) \\ \eta &= \sum_i e_i y_i / \sum_i e_i (z_i - z_0)^2 \end{aligned} \tag{2}$$

The coefficients can be chosen nearly freely as long as the following constraints are satisfied:

$$\sum_i a_i (z_i - z_0) = 0 \tag{3}$$

$$\sum_i b_i = 0 \tag{4}$$

$$\sum_i c_i (z_i - z_0) = 0 \quad \sum_i c_i (z_i - z_0)^2 = 0 \tag{5}$$

$$\sum_i d_i = 0 \quad \sum_i d_i (z_i - z_0)^2 = 0 \tag{6}$$

$$\sum_i e_i = 0 \quad \sum_i e_i (z_i - z_0) = 0 \tag{7}$$

For example, the least-square fit is an algorithm that yields minimum fitting error. The bend view parameters of the least-square fitting are linear combinations given in the following with coefficients satisfying the constraints above:

$$\begin{aligned} y_0 &= (A\sum_i y_i - B\sum_i (z_i - z_0)^2 y_i)/(nA - B^2) \\ h &= \sum_i (z_i - z_0) y_i / B \\ \eta &= (n\sum_i (z_i - z_0)^2 y_i - B\sum_i y_i)/(nA - B^2) \\ A &= \sum_i (z_i - z_0)^4 \quad B = \sum_i (z_i - z_0)^2 \end{aligned} \tag{8}$$

For other choices of the coefficients in the linear combination, the errors of the fitting are bigger, but each choice is still considered as a valid fitting algorithm.

B. Non-Division Integer-Only Operations for FPGA

In general, the computations above need floating point multiplications and divisions. To simplify the process so that it can be done in an FPGA with low resource usage, the advantage of invariance of Equations (2) through (7) under re-scaling is taken. Each set of coefficients can be multiplied with a common factor so that the sums in the denominators in Equation (2) become 32, 512 or 4096. The constraints for the coefficients can be summarized as following:

$$\sum_i a[i] = 32 \qquad \sum_i a[i](z_i - z_0) = 0 \tag{9}$$

$$\sum_i b[i] = 0 \qquad \sum_i b[i](z_i - z_0) = 512 \tag{10}$$

$$\sum_i c[i] = 32 \quad \sum_i c[i](z_i - z_0) = 0 \quad \sum_i c[i](z_i - z_0)^2 = 0 \tag{11}$$

$$\sum_i d[i] = 0 \quad \sum_i d[i](z_i - z_0) = 512 \quad \sum_i d[i](z_i - z_0)^2 = 0 \tag{12}$$

$$\sum_i e[i] = 0 \quad \sum_i e[i](z_i - z_0) = 0 \quad \sum_i e[i](z_i - z_0)^2 = 4096 \tag{13}$$

Then the linear combinations for calculating the track parameters can be rewritten:

$$\begin{aligned} xx_{32} &= \sum_i a[i]x[i] \approx 32x_0 \quad ll_{512} = \sum_i b[i]x[i] \approx 512l \\ yy_{32} &= \sum_i c[i]y[i] \approx 32y_0 \quad hh_{512} = \sum_i d[i]y[i] \approx 512h \\ eta_{4096} &= \sum_i e[i]y[i] \approx 4096\eta \end{aligned} \tag{14}$$

The unit of measured coordinates x[i] and y[i] are chosen so that they are integers. For example, the channel numbers or their integer multiples in the silicon pixel detector can be used as the hit coordinates. The coefficients in the linear combinations are also chosen to be integers and the results of the linear combinations: xx_{32}, ll_{512}, yy_{32}, hh_{512} and eta_{4096} are also integers, representing the corresponding parameters scaled by factors of 32, 512 and 4096, respectively. Note that divisions are not needed anymore. The only computations needed in Equation (14) are integer multiplications and accumulation.

The unit in the z direction is chosen so that the separation between two detector planes is 2. With this unit, the possible values of (z_i - z_0) are also integers, which are even when the number of hits is odd with the middle plane being 0 and are odd when the number of hits is even with the two center detector planes being -1 and +1.

With this simple scaling, the fitting process can be

implemented in FPGA devices with integer multipliers such as the Altera Cyclone II [8] family devices. Bigger scaling factors such as 8192, 16384 etc. can also be chosen if better calculation precision is required.

C. Eliminating Full Multiplications

To reduce computations further, the coefficients in the linear combinations are limited to the "weight-two" or "two-bit" integers, e.g., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, which are 2^m+2^n or 2^m-2^n with positive integers m and n. An example of choosing e[i] for tracks with odd numbers of hits is shown in Table I.

TABLE I
COEFFICIENTS FOR THE FPGA TRACK FITTER (CURVATURE, ODD HITS)

	Half-length of the Track													
	16		14		12		10		8		6		4	
z-z_0	e_i	e[i]	e_i	e[i]	e_i	e[i]	e_i	e[i]	e_i	e[i]	e_i	e[i]	e_i	e[i]
-16	5.3	6												
-14	3.3	2	7.5	8										
-12	1.6	2	4.3	4	11.3	12								
-10	0.1	0	1.6	2	5.6	5	17.9	18						
-8	-1.1	0	-0.7	-2	1.0	1	7.2	7	31.0	31				
-6	-2.0	-3	-2.4	-2	-2.6	-4	-1.2	-1	7.8	8	61.0	56		
-4	-2.6	-3	-3.6	-5	-5.1	-5	-7.2	-8	-8.9	-9	0.0	12	146.3	144
-2	-3.0	-3	-4.4	-4	-6.6	-5	-10.7	-9	-18.8	-20	-36.6	-40	-73.1	-64
0	-3.2	-2	-4.6	-2	-7.2	-8	-11.9	-14	-22.2	-20	-48.8	-56	-146.3	-160
2	-3.0	-3	-4.4	-4	-6.6	-5	-10.7	-9	-18.8	-20	-36.6	-40	-73.1	-64
4	-2.6	-3	-3.6	-5	-5.1	-5	-7.2	-8	-8.9	-9	0.0	12	146.3	144
6	-2.0	-3	-2.4	-2	-2.6	-4	-1.2	-1	7.8	8	61.0	56		
8	-1.1	0	-0.7	-2	1.0	1	7.2	7	31.0	31				
10	0.1	0	1.6	2	5.6	5	17.9	18						
12	1.6	2	4.3	4	11.3	12								
14	3.3	2	7.5	8										
16	5.3	6												
Error	2.91	3.02	3.05	3.15	3.22	3.26	3.41	3.43	3.65	3.65	3.93	3.99	4.28	4.29

The columns of e_i in Table I represent coefficients derived from the least-squares fitting. The e[i] coefficients are chosen in a spread sheet, guided by the e_i coefficients. Since the parameterization of the track in Equation (1) is chosen with symmetry around z_0, the coefficients for the least-squares fitting are also symmetric. Some symmetric properties of the coefficients are sufficient conditions for certain constraints in Equations (9) though (13). For example, the symmetric property $e_i = e_{-i}$ exists and with this property, the constraint $\Sigma e_i(z_i-z_0)=0$ is satisfied automatically. Appropriate symmetries are programmed in the spread sheet cells so that the constraints and scaling requirement are satisfied with minimum hand editing. In our work, the coefficient selection is semi-automatic. Clearly it is not too difficult to write a fully automatic program to choose the coefficients.

The relative errors contributed by the parameter η for both algorithms are calculated. The relative error here is defined as transverse reconstruction RMS error after projecting the track by half-length (L/2) from the first or last hit of the track, with the unit of the RMS error of the y[i] measurements. Assume the errors of the y[i] measurements δy_i are independent and they have the same RMS value δy, then the error of the calculating parameter η can be estimated:

$$\delta\eta = \delta y \sqrt{\sum_i (e[i])^2} / 4096 \tag{15}$$

The assumption of independence of the measurement errors may not hold if most tracks are high momentum and parallel. But for typical events with various track angles, it is a good approximation. The transverse reconstruction RMS error δY after projecting the track by half-length from first or last hit of the track, i.e., $(z-z_0) = 2(L/2)$, can be calculated:

$$\delta Y = (z - z_0)^2 \delta\eta = 4(L/2)^2 \delta y \sqrt{\sum_i (e[i])^2} / 4096 \tag{16}$$

More precisely, the values in row "Error" of Table I are defined as ($\delta Y/\delta y$). The choice of error-scaling here reflects the typical application in our detector. When charged tracks come from the interaction point with small angles, they travel long distances before hitting the detector planes and produce long tracks with many hits. On the other hand, shorter tracks with few hits are those coming from interaction points with larger angles and hit the first detector planes after traveling short distances. Therefore, the projections needed for tracks are often proportional to their lengths recorded in the detector.

Note that the "projection" here should just be viewed as a rescaling process to bring the fitting errors of a parameter to a convenient unit. It is an estimate for a single parameter only. If, for example, the vertex measurement error is needed, the errors for all parameters should be considered simultaneously since they could be correlated.

Similarly, the coefficients e[i] and relative errors for tracks with even numbers of hits can be calculated as shown in our previous document [9].

From Table I, it can be seen that the errors increase for the multiplierless algorithm compared with the least-squares algorithm. However, the multiplierless fitting algorithm increases the track reconstruction errors only slightly (less than 4%).

III. FITTING ERRORS AND DISCUSSIONS

The coefficients of the fitting matrix for the other two parameters of a curved track yy_{32} and hh_{512} are also calculated using similar procedures. The relative errors, defined as ($\delta Y/\delta y$), contributed by the three parameters with different track lengths for both least-squares and FPGA multiplierless algorithms are plotted in Fig. 2.

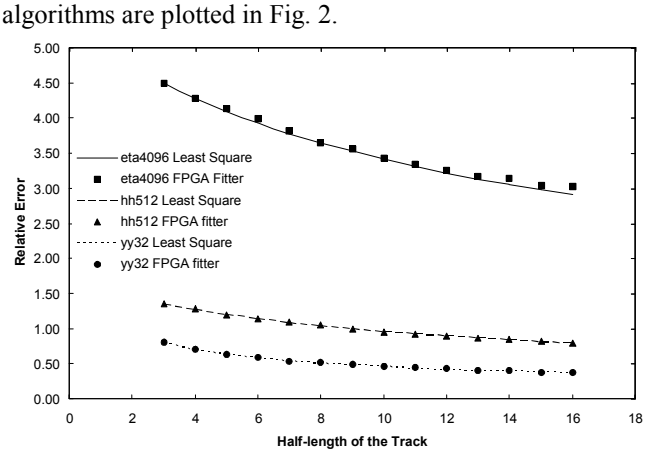

Fig. 2 Relative errors between the least-squares fit and the FPGA multiplierless fit.

The multiplierless algorithm used in the FPGA increases the fitting errors for all parameters as expected. However, the

differences are very small. In other words, the fitting errors are relatively insensitive to the variations of the coefficients of the fitting matrix.

It can be seen from Fig. 2 that the relative error contributed by the curvature parameter (eta_{4096}) is significantly higher than the errors contributed by the offset and slope parameters. We will focus on the curvature parameter in our discussion.

In general, fitting of longer tracks yields smaller errors due to two reasons: longer lever arms and more measurement points. In order to compare these two effects, relative errors contributed by the curvature parameter calculated with several fitting schemes are plotted in Fig. 3.

In addition to the least-squares algorithm and the FPGA approximation shown in Table I, two other "3-point" algorithms are also studied. One of the 3-point schemes calculates eta_{4096} using hits of the first three detector planes at the beginning of a track. In this case, the lever arm is a fix length, despite the extra length provided by additional planes. This scheme produces the largest errors. When the track is projected over long distances, the errors increase rapidly. The calculation method of this scheme is the simplest. However, the result from this scheme is only useful as a coarse estimate of track momentum. It causes large errors due to short lever arm when the tracks are to be projected in long distance.

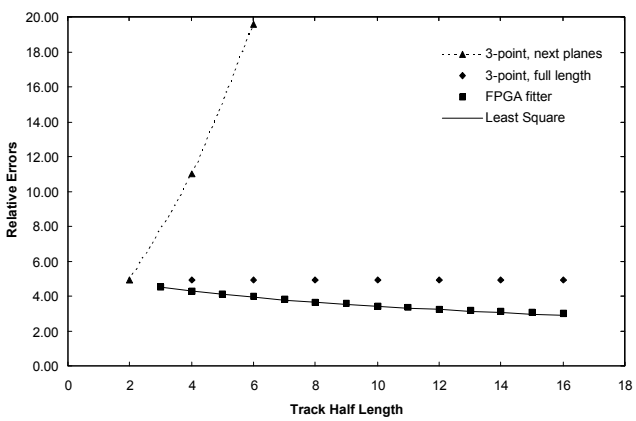

Fig. 3 Relative errors of several track fitting schemes.

The other 3-point scheme calculates the eta_{4096} using the first, the middle and the last hits of a track. In this situation, advantage of the full track length is taken. However, the information provided by the redundant measurements of the other points in the track is not used. The relative errors when the track is projected over half-length are nearly a constant.

The full-length 3-point scheme may appear to be simpler in computation than the multiplierless fitting scheme, but actually this is not the case. To calculate eta_{4096} using the 3-point scheme, a lookup table and a floating point multiplication are needed in order to bring results for different track lengths into a unified scale. In the multiplierless fitting scheme, the scale unification is achieved through choosing the two-bit coefficients.

IV. FPGA Implementation

FPGA curved track fitters both with and without multipliers were compiled and simulated in an Altera Cyclone II [8] device (EP2C5T144C6). The block diagram of the multiplierless version is shown in Fig. 4.

When a data train of track hit coordinates arrives, the number of hits in the track, "Nalg" is first extracted from the header. This number is used to construct the address for the Constant MEM so that the set of the constants corresponding to the given track length is chosen. The 16-bit hit coordinates x(i) and y(i) are clocked at 100 MHz through the fitter, one clock cycle per pair of the coordinates (i.e., the total throughput rate is 400 MB/s).

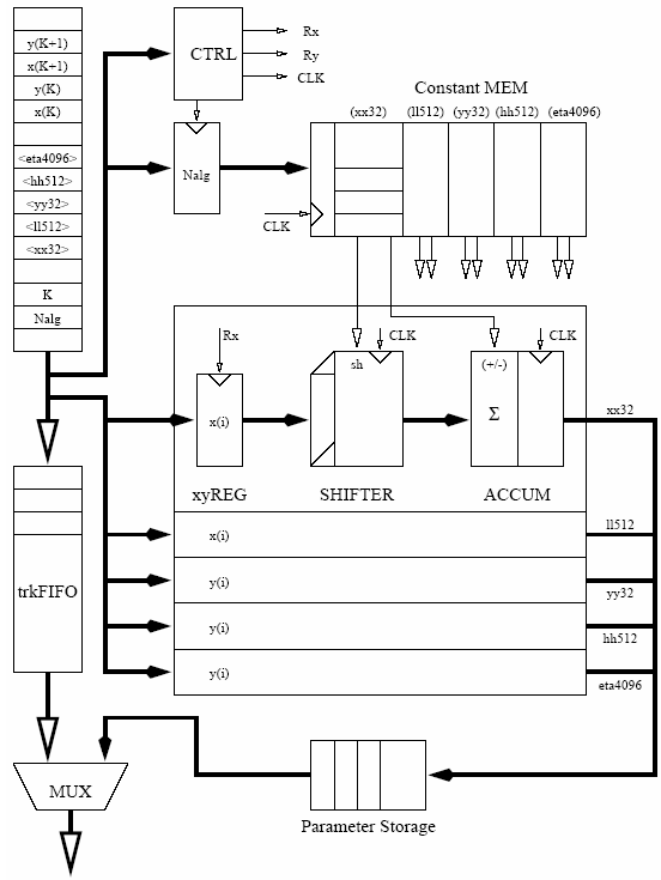

Fig. 4 The multiplierless FPGA track fitter.

For each parameter, an accumulator is used to calculate the linear combination. The coordinate data is shifted through a logarithmic shifter by pre-defined numbers of bits that are stored in the constant memory. The shifted version of the coordinate is added to or subtracted from the accumulator.

The Constant MEM, SHIFTER and accumulator are clocked at 200 MHz. Therefore, each coordinate is shifted and added/subtracted twice. This is equivalent to multiplying the coordinate by a two-bit integer and accumulating for the linear combination.

A simple simulation is shown in Fig. 5. Hit coordinates of a track with 5 hits are sent through the fitter. The coordinates X and Y represent a track with the following parameters: x_0 = 1002, l = 0.5, y_0 = 4, h = 2 and η = 7. The signal HDR is an indicator of the header word and Y (=5) at the header is interpreted as the number of hits in the track. The coordinates are then fed into the fitter and are accumulated. Shortly after

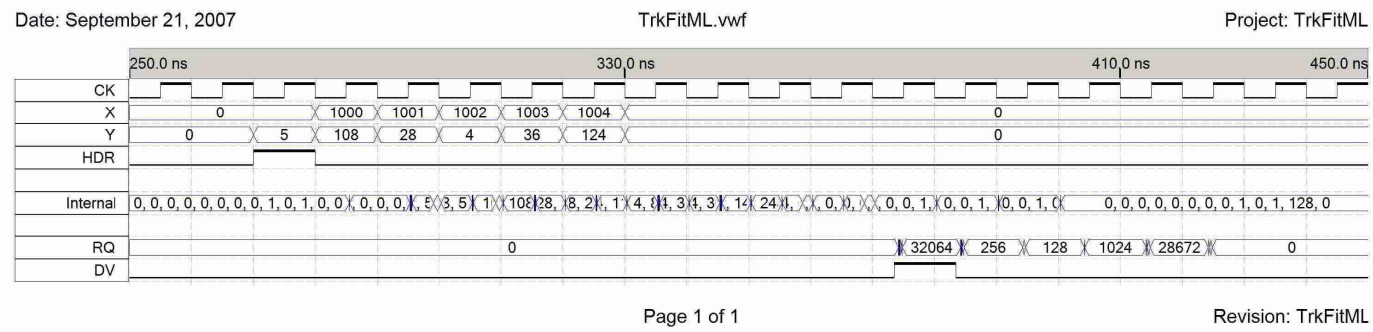

Fig. 5. Simulation of the FPGA track fitter (multiplierless version)

all coordinates are input, the calculated track parameters xx32, ll512, yy32, hh512 and eta4096 are output from the bus RQ in sequence, with the data valid signal DV indicating the first parameter xx32. It can be seen that the outputs are the expected results: xx32 = 32064 = 32*1002, ll512 = 256 = 512*0.5, yy32 = 128 = 32*4, hh512 = 1024 = 512*2 and eta4096 = 28672 = 4096*7.

The time between the inputting of the last hit coordinates and the outputting of first track parameter is about 45 ns, or 9 clock cycles at 200 MHz. However, this is not the "deadtime" of the fitter. The fitter is designed in a pipelined fashion which allows a new set of track data to follow the previous set of data immediately.

Silicon resource usages of the FPGA curved track fitters are given in Table II.

TABLE II
SILICON RESOURCES USAGE OF CURVED TRACK FITTERS IN FPGA

Device: EP2C5T144C6 Price: $19.20 (September 2007)		
	With Multiplier	Multiplierless Version
Logic Elements	634/4608 (14%)	728/4608 (16%)
M4K RAM	6/26 (23%)	3/26 (12%)
18-bit Multipliers	3/13 (23%)	0

Both fitters (with and without multipliers) process a pair of coordinates, 16-bit each, at 100 MHz. Each parameter is kept in a 32-bit accumulator. In the version with multipliers, the multipliers are clocked at 200 MHz, which allows two parameters to share a multiplier. So the fitter using 3 multipliers is capable of calculating up to 6 linear combinations, although only 5 are used.

In implementing the logarithmic shifter for the multiplierless version, advantage of the global loading feature available in the Cyclone II FPGA family is taken so that a 3-to-1 multiplexer can be implemented in one logic element (typically can only implement a 2-to-1 MUX). This way, a logarithmic shifter capable of shifting 0 to 8 bits can be built in 2 stages. A 16-in-24-out shifter uses 44 logic elements with this design.

V. CONCLUSION

Track fitting in an FPGA both with and without multipliers has been discussed. The fitting errors in the approximate multiplierless algorithm is only slightly larger than the errors of the least-squares fitting algorithm.

Multipliers are now available in more and more FPGA devices today and it seems that eliminating multiplications is not as critical as several years ago. Intrinsically, however, multiplication is a power and resource consuming operation and a multiplierless algorithm becomes especially beneficial when the firmware developed in an FPGA is to be ported to an application-specific integration circuit (ASIC).

Generally speaking, more computations yield better quality of the results. From the 3-point schemes, to the multiplierless FPGA fitter, to the least-squares algorithm, the fitting errors reduce as the number of total operations increases. However, after a certain point, the quality of the results does not improve as rapidly as before. It is common that a large amount of computations brings only a small improvement in the mathematically perfect algorithms. So it is possible to find algorithms with a reasonable amount of computations that produce sufficiently good results, as illustrated in this paper.

ACKNOWLEDGEMENT

The authors would wish to express thanks to Bob DeMaat of Fermilab and Prof. Gilbert Strang of Massachusetts Institute of Technology for their helpful input.

APPENDIX A

In an attempt to understand the insensitivity of the fitting errors in the multiplierless algorithm, we study several topics on least-squares fitting. Some variables are redefined from our earlier document [9] to follow the conventions in mathematics references.

Consider fitting m measurement points with a set of n ($n<m$) functions using various algorithms:

$$y = x_1 a_1(z) + x_2 a_2(z) + x_3 a_3(z) + \cdots + x_n a_n(z) \tag{A1}$$

In fitting with a parabola, e.g., the functions $a_1(z)$ $a_2(z)$, $a_3(z)$ are chosen to be 1, z, z^2. The parameters x_1, x_2 to x_n (Note: They are not coordinates.) are to be evaluated through fitting the measurement data (z_1, y_1), (z_2, y_2) to (z_m, y_m). The z-coordinates here are known and can have any values. We use an m-row by n-column matrix $A = \{a_{ij}\} = \{a_j(z_i)\}$ to denote the function values at the corresponding measurement points. We also define vectors $y = (y_1, y_2, \ldots y_m)^T$ and $x = (x_1, x_2, \ldots x_n)^T$ so that an over-constrained linear equation system can be written

in matrix format:

$$y = Ax \tag{A2}$$

One set of parameters x that produces minimum fitting errors can be found using least-squares method. We skip the detailed derivation process of least-squares fitting and directly write down the final result [10][11]:

$$\hat{x} = (A^T A)^{-1} A^T y \tag{A3}$$

From the total least-squares (TLS) theory, the relative variations of the final fitted parameters is upper bounded by the relative perturbations of the measurement data y and the A matrix, scaled by the condition number of matrix A.

$$\frac{\|\delta x\|}{\|x\|} \le K(A)\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta y\|}{\|y\|}\right) \tag{A4}$$

The condition number $K(A)$ of matrix A is defined as the ratio of the biggest and smallest singular values of A derived from the singular value decomposition (SVD) of A. We use the 2-norm ||*|| throughout this document. In order to obtain a robust set of parameter x, it is necessary that matrix A is well-conditioned, i.e., the condition number $K(A)$ is not too much bigger than 1, which we shall assume.

In multiplierless algorithm design work, however, a tighter and parameter-by-parameter error estimate is more desirable due to following reasons:

(1) The elements in x usually have different units, such as offset, slope and so on. In order to calculate the norm $\|x\|$, the elements must be redefined so that they have a unified unit.

(2) In multiplierless applications, the perturbations in A normally can be ignored as in ordinary least-squares (OLS). But there are perturbations from calculation and the perturbations usually can be limited within some subspaces (which we will see is an advantage) and their effects are small.

In the remaining part of this section, we first define a matrix called algorithm matrix which represents a generic fitting algorithm. Then we will show that the least-squares algorithm is a special case of the generic algorithms. We will study the orthogonal property of the algorithm difference matrices and their application in estimating fitting errors.

Definition 1: The algorithm matrix $G = \{g_{ji}\}$ is defined to be an n-row by m-column left inverse matrix of A:

$$GA = I \quad \sum_{i=1}^{m} g_{ji} a_{ik} = \delta_{jk} \quad j,k = 1,2,...n \tag{A5}$$

With an algorithm matrix G, the values calculated from the following linear combinations are called a set of fitting parameters under algorithm G:

$$x = Gy \quad x_j = \sum_{i=1}^{m} g_{ji} y_i \quad j = 1...n \tag{A6}$$

With constraints (A5), the parameters calculated in (A6) reduce to the "true" values if the fitting model (A1) is correct and there are no measurement errors. In fact, if for a set of parameters x_0 (the "true" values), the linear equation system $y=Ax_0$ holds for all rows, then $x = Gy = GAx_0 = I\,x_0 = x_0$.

In general the left inverse G of m x n matrix A is not unique. For each parameter x_j, there are m values of g_{ji} that satisfy n constraints which permits different values of g_{ji} to be chosen. Each set of g_{ji} values corresponds to a particular fitting algorithm. For example, given four measurement points, there are many ways to calculate the parameter η and two of them are shown in Fig. 1(b). It can be shown that the least-squares fitting algorithm is a special case of the generic algorithms.

We use $\hat{G}$ to represent the algorithm matrix of the least-squares fitting. Comparing with (A6), it can be seen that:

$$\hat{G} = (A^T A)^{-1} A^T \tag{A7}$$

Clearly $\hat{G}$ satisfies the left inverse constraints given in (A5):

$$\hat{G}A = (A^T A)^{-1} A^T A = I \tag{A8}$$

We choose the least-squares algorithm as a reference. All the other generic fitting algorithms are viewed as a perturbation from the least-squares algorithm:

$$G = \hat{G} + \Delta G \quad g_{ji} = \hat{g}_{ji} + \Delta g_{ji} \tag{A9}$$

Theorem 1 (The Algorithm Deviation Orthogonality Theorem, ADOT): The row space of the difference matrix ΔG of two algorithms is orthogonal to the column space of A:

$$\Delta G A = 0 \quad \sum_{i=1}^{m} (\Delta g_{ji}) a_{ik} = 0 \quad j,k = 1,2...n \tag{A10}$$

Simply right multiply (A9) with A and use (A5) and (A8) to get equation (A10). Note that this theorem is true for the difference of any two algorithms but we will use ΔG to denote only the difference between an arbitrary algorithm and the least-squares algorithm.

Corollary 1: The row space of difference matrix ΔG of two algorithms is orthogonal to the row space of the algorithm matrix $\hat{G}$ of the least-squares fitting:

$$\Delta G \hat{G}^T = 0 \tag{A11}$$

Combining (A7) and (A10) gives (A11). In fact, (A7) describes that the row space of $\hat{G}$ is in the column space of A and according to ADOT, the row space of ΔG is orthogonal to any vectors in the column space of A including row vectors of $\hat{G}$.

Now consider the fitting error for each of the parameters x_1, x_2 to x_n. If the y-measurements are independent and they have the same standard deviation δy, the variations of the parameters can be written:

$$(\delta x_j)^2 = (\delta y)^2 \sum_{i=1}^{m} (g_{ji})^2 \tag{A12}$$

The square-sum in (A12) is actually the length of the row vector of the matrix G. It contains contributions from the least-squares algorithm and the additions from the perturbation:

$$\sum_{i=1}^{m} (g_{ji})^2 = \sum_{i=1}^{m} (\hat{g}_{ji} + \Delta g_{ji})^2 = \sum_{i=1}^{m} \hat{g}_{ji}^{\,2} + \sum_{i=1}^{m} 2\hat{g}_{ji}\Delta g_{ji} + \sum_{i=1}^{m} \Delta g_{ji}^{\,2} \tag{A13}$$

The middle sum term in (A13) vanishes because from (A11), the row spaces of ΔG and $\hat{G}$ are orthogonal. The variations of the fitting parameters due to measurement errors are simply composed with two parts:

$$(\delta x_j)^2 = (\delta y)^2 \left(\sum_{i=1}^{m} (\hat{g}_{ji})^2 + \sum_{i=1}^{m} (\Delta g_{ji})^2\right) \tag{A14}$$

The equation (A14) reflects the fact that the error of each parameter is a minimum when the least-squares fitting

algorithm is used. Around this point, the fitting errors are relatively insensitive to the perturbation of g_{ji} values allowing the user to choose different g_{ji} values to reduce the total computations without increasing the fitting errors significantly. If, for example, all g_{ji} values differ from values in $\hat{g}_{ji}$ by 10%, the error of each parameter increases by only about 0.5% from the minimum.

The intrinsic reason for this insensitivity is because the perturbation of G is limited in a subspace allowed by (A5). Unlike the perturbations generated by computational rounding errors which are small but unrestricted, the perturbations in multiplierless algorithms are allowed to be relatively large as long as (A5) is satisfied. From ADOT, the perturbations in multiplierless algorithms increase the fitting error only slightly.

REFERENCES

[1] B.Ashmanskas et al., "The CDF Silicon Vertex Trigger", Nucl. Instr. and Meth., vol. A518, pp. 532-536, 2004.

[2] J.Adelman et al., "The Silicon Vertex Trigger upgrade at CDF", Nucl. Instr. and Meth., vol. A572, pp. 361-364, 2007.

[3] S.Amerio et al., "The GigaFitter for fast tracking based on FPGA DSP arrays", in *Nuclear Science Symposium Conference Record, 2007. IEEE*, Volume 3, Oct. 26 2007 Page(s): 2115-2117.

[4] Kulyavtsev et al., BTeV proposal, Fermilab, May 2000, available: { http://www-btev.fnal.gov/DocDB/0000/000066/002/index.html}

[5] E.E. Gottschalk, BTeV detached vertex trigger, Nucl. Instrum. Meth. A 473 (2001) 167.

[6] J. Wu et al., "The application of tiny triplet finder (TTF) in BTeV pixel trigger", IEEE Tans. Nuclear Science, vol. 53, p 671, 2006.

[7] J. Wu et al., "Integrated Upstream Parasitic Event Building Architecture for BTeV Level 1 Pixel Trigger System", IEEE Tans. Nuclear Science, vol. 53, p 1039, 2006.

[8] Altera Corporation, "Cyclone II Device Handbook", (2007) available via: {http://www.altera.com/}

[9] J. Wu et al., "FPGA Curved Track Fitter with Very Low Resource Usage", in *Nuclear Science Symposium Conference Record, 2006. IEEE*, Volume 2, Oct. 29 2006-Nov. 1 2006 Page(s):1290 - 1295.

[10] Strang, G. "Introduction to Linear Algebra", 3rd ed. Wellesley, MA: Wellesley-Cambridge Press, March 2003

[11] Leon, S. "Linear Algebra with Applications", 5th ed., Upper Saddle River, NJ, 1994.