Design of a Real Time FPGA-based Three Dimensional Positioning Algorithm

Nathan G. Johnson-Williams

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2010

Program Authorized to Offer Degree:
Department of Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a Master's Thesis by

Nathan G. Johnson-Williams

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Scott A. Hauck

_____

Robert S. Miyaoka

_____

Thomas K. Lewellen

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature _____

Date _____

University of Washington

Abstract

Design of a Real Time FPGA-based
Three Dimensional Positioning Algorithm

Nathan G. Johnson-Williams

Chair of the Supervisory Committee:
Professor Scott A. Hauck
Electrical Engineering

Abstract- We report on the implementation and hardware platform of a real time Statistic-Based Processing (SBP) method with depth of interaction processing for continuous miniature crystal element (cMiCE) detectors using a sensor on the entrance design. Our group previously reported on a Field Programmable Gate Array (FPGA) SBP implementation that provided a two dimensional (2D) solution of the detector's intrinsic spatial resolution. This new implementation extends that work to take advantage of three dimensional (3D) look up tables to provide a 3D positioning solution that improves intrinsic spatial resolution. Resolution is most improved along the edges of the crystal, an area where the 2D algorithms performance suffers. The algorithm allows an intrinsic spatial resolution of ~0.90 mm FWHM in X and Y and a resolution of ~2.00 mm FWHM in Z (i.e., the depth of the crystal) based upon DETECT2000 simulation results that include the effects of Compton scatter in the crystal. A pipelined FPGA implementation is able to process events in excess of 220k events per second, which is greater than the maximum expected coincidence rate for an individual detector. In contrast to all detectors being processed at a centralized host, as in the current system, a separate FPGA is available at each detector, thus dividing the computational load. A prototype design has been implemented and tested using a reduced word size due to the memory limitations of our commercial prototyping board.

Table of Contents

Page

List of Figures

Page

No table of figures entries found. List of Tables

Page

Acknowledgments

The author wishes to express appreciation for the support of Zecotech, Altera, DOE grant DE-FG02-08ER64676, and NIH grants NIBIB EB001563 and EB002117 for the funding that made this project possible.

1 Positron Emission Tomography

Positron Emission Tomography (PET) is a medical imaging technique used to produce a three dimensional image of biological processes. A short-lived isotope is used as a radioactive tracer that allows a radiologist to locate the tracer inside the subject by detecting its positron emissions. The radioactive tracer is attached to a molecule of interest and placed in the subject's bloodstream, allowing the radiologists to follow and locate the molecule as it is processed by the body. Unlike CT or MRI, which gives an image of what is inside the body, PET produces an image of what is happening inside the body. Different molecules can be used depending on the biological process of interest. For example, Fluorodeoxyglucose, an analogue of glucose, is used to provide an image of metabolic activity by detecting the concentration of the tracer at various locations.

A key part to PET operation is the tracer. The tracer is what allows a radiologist to follow the molecule of interest without disturbing the natural processes. The tracer principle is based on the fact that radioactive molecules behave in the same way as nonradioactive molecules in an organism's biological processes. However, it is not the tracer itself that allows the molecule to be followed. As the isotope stabilizes it emits both positrons and neutrinos, the emitted positrons can be traced back to the source molecule.

As the tracer decays the positrons, the antiparticle of electrons, are emitted from the molecule. Within a short distance the positron encounters an electron and annihilates, producing two high energy (511 kv) photons. The two photons move at the speed of light in opposite directions, nearly 180° apart. The photons travel in a straight line though the body of the subject (somewhat attenuated by bone or tissue). The photons leaving the body are then detected by a ring of detectors that surround the subject [Figure 1].

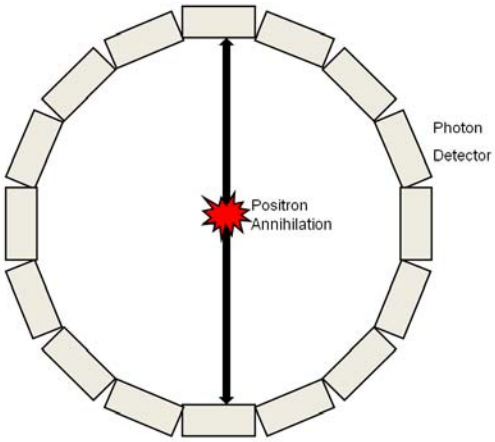

Figure 1. PET scanner detector arrangement

Since the high energy photons are not directly detectable by most devices, the detector used is a scintillator device. The detector module consists of two parts: the scintillator crystal and a light detector. A scintillator crystal is a luminescent material, when struck with a photon it absorbs its energy and reemits the energy as visible light. The visible light pattern is conducted through the translucent crystal with ideally little attenuation. The light response pattern at the back of the crystal is then measured by a light detector. Two common light detectors used in PET are Photomultiplier Tube (PMT) and photodiodes.

As the photon is detected, careful timing circuitry is used to measure when the photon was detected. Two detections at opposite sides of the detector ring within the same time frame is an indication that the photons detected were caused by the same positron emission. These two photons are known as a photon pair. Once such a photon pair is detected, a line of response (LOR) is drawn, localizing the position of the positron emission to somewhere along that line.

After many events have been collected an image can be created from the information derived from the combination of the many LORs. Typically a PET scan will use the information from many thousands to millions of events to create an image [Figure 2].

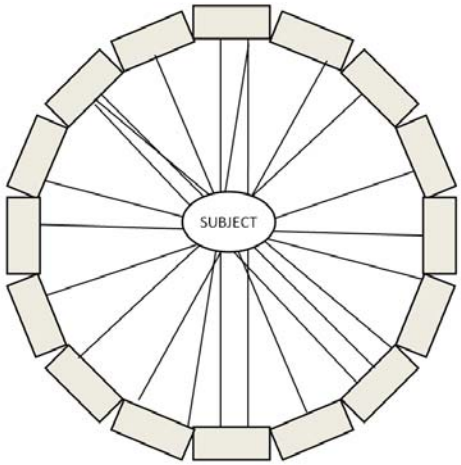

Figure 2. Example of line of response image recreation

There have been two approaches used in creating detectors at the University of Washington. One approach, called dMiCE, constructs the scintillator crystal out of a matrix of many smaller crystals. An opaque or semi opaque material is placed in between each small crystal, either eliminating or reducing the amount of light that is conducted from one crystal to another. This narrows the focus of the light response [Figure 3]. The other, cMiCE, uses a continuous crystal, which gives a broader distribution of light as the conduction of the light response pattern is only limited by the edge of the scintillator [Figure 4]. In both cases the scintillator crystal is connected to an array of light detectors.

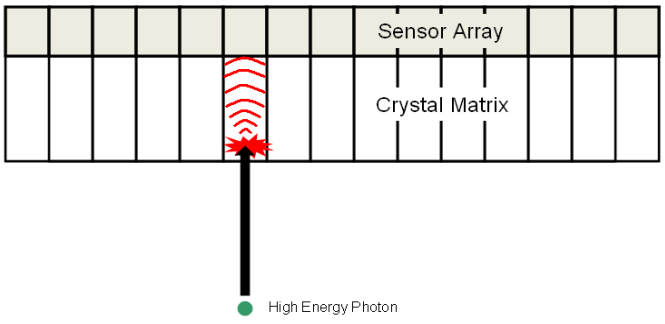

Figure 3. dMiCE detector module light response

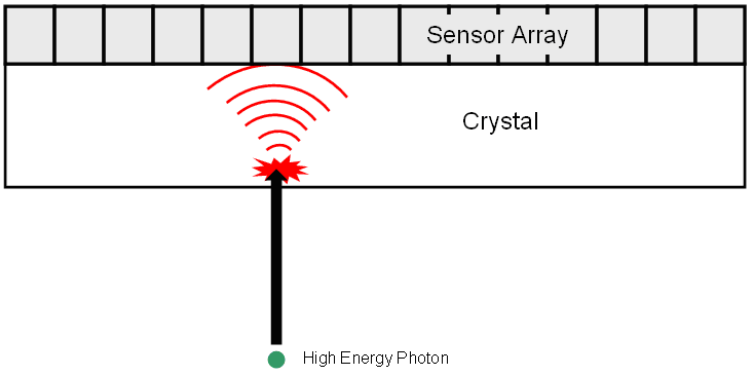

Figure 4. cMiCE detector module light response

The dMice approach has the advantage of easier positioning and accuracy. The light is focused on a single sensor element directly below the particular crystal element. The position of the event can be easily found by finding the highest reading sensor. The primary disadvantage of the dMiCE approach is cost. Each crystal in the matrix that makes up the scintillator crystal must be hand polished and assembled in the matrix [Figure 5]. The process of building a dMiCE module is both labor intensive and time consuming. Additionally each crystal in the matrix must be matched with a sensor element and for high resolution this requires many sensors in the array.

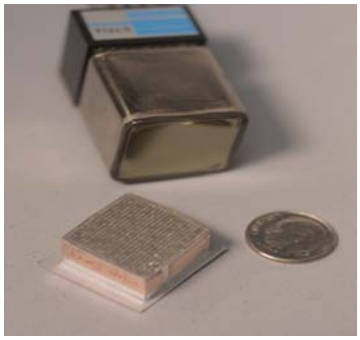

Figure 5. dMiCE detector module scintillator crystal matrix

In an effort to save cost, the cMiCE module was designed. In place of a small crystal matrix, a continuous monolithic crystal is used. In using a monolithic crystal the light response is not focused and many sensors are affected in place of just one. Since many sensor elements will be affected by an event it was found that a higher density of sensors wasn't necessary. However, determining the position of an event becomes much more

complex as the position of the event must be derived based on the light response pattern of the sensor array,

Most recently a new cMiCE module has been developed. The new module features a thicker 15mm crystal and sensors placed on the entrance position [Figure 6]. The new sensor position has the advantage of placing the sensor array at the surface of crystal, where most of the events occur. In addition to the new sensor array placement a new higher density array of photodiodes are used [1].

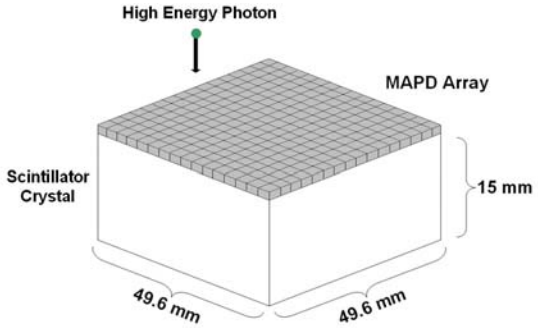

Figure 6. Updated cMiCE detector module

2 Statistical Based Positioning

Statistical Based Position (SBP) is a positioning algorithm used to determine the most likely location an event occurred based on comparing the sensor array output to sensor outputs from events at known locations [2,3]. Each crystal module is characterized, measuring the sensor array response for many possible positions [4,5,6,7]. At each position the mean and variance of each sensor output is stored. A sensor output caused by an event at an unknown location can then be compared with the mean and variance of sensor outputs from known locations, calculating the likelihood [Equation 1] that the event occurred at that location. The location of the event is determined by comparing the likelihood of all tested positions and selecting the one with the greatest likelihood.

$$\ln(P(\text{event}, X, Y)) = -\sum_{\text{row}=1}^{8}\sum_{\text{col}=1}^{8}\left[\frac{\left[\text{event}_{\text{row,col}} - \left(\mu_{X,Y}\right)_{\text{row,col}}\right]^2}{2\cdot\left[\left(\sigma_{X,Y}\right)_{\text{row,col}}\right]^2} + \ln\left[\left(\sigma_{X,Y}\right)_{\text{row,col}}\right]\right]$$

Equation 1: Liklihood equation

For 2D SBP, the module is characterized for 127x127 positions, with 127x127 representing the X and Y dimensions. Each XY position is on a uniform grid with each point separated by 0.3875mm. In total the module is characterized for 16,129 different positions.

The number of different possible positions makes this method computationally intensive. To be absolutely certain that the position selected for a given event is the most likely, the likelihood of all 16,129 positions must be calculated and compared. If performing these operations one at a time, as with a PC, this approach could take a long time to process. Exhaustive searches, such as the one just described, executed on a PC can typically take hours for a data set of 10,000 events.

Several variations of the SBP method were designed which aimed to reduce the computational intensity of finding a solution. While an exhaustive search, which calculates the likelihood of each and every point, produced the most correct SBP solution, it was incredibly computationally intensive. A "Localized Exhaustive" was designed which assumed the event occurred near the highest reading sensor and performed a comparison of all points near that sensor. This greatly reduced the number of computations required to produce a solution. A structured hierarchical search was also designed which compares points in several iterations with each search becoming progressively finer grained and focused until a solution was determined.

2.1 The need for 3D SBP

The 2D SPB algorithm solves for the most likely X and Y position on the crystal. However, the photon does not always strike the surface of the crystal or at a consistent depth. Although the photon generally strikes very near the surface of the crystal, in some cases it may travel a ways into the crystal before striking a crystal molecule. The most prominent repercussion of this occurrence is an inaccuracy in the line of response. Assuming that the photon strikes near the surface may work for most cases, but in instances in which the photon actually traveled farther into the crystal this can result in a significantly incorrect line of response [Figure 7].

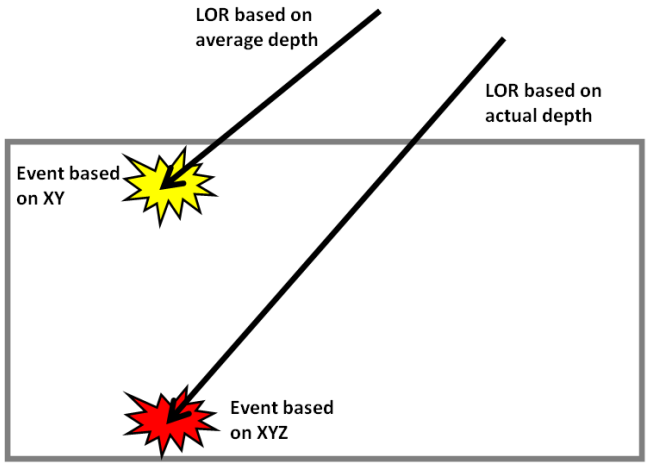

Figure 7. Illustration of LOR error based on depth of interaction

In characterizing a detector module for 2D SBP all events at a given XY position are included regardless of depth. However, the light response pattern is very different for events that occur near the sensor array compared to those that occur farther away. Since the characterization values are based upon events that occur at all depths, and most events occur near the surface of the crystal, this biases the characterization tables toward the light response pattern for events that occur near the surface of the crystal. In the case of an actual event occurring far from the surface of the crystal the characterization values are inaccurate as they are more strongly based on events that occurred at a different depth. Characterizing the detector module for three dimensions allows more accurate characterization values as the values are more closely related to the situation in which they will be used. The more accurate characterization values give more accurate likelihood calculations.

The new detector module, which has a 15mm deep crystal compared to the older 8mm crystal, is another reason the 3D algorithm is developed. While the 2D algorithm performed well with the 5mm crystal, it did not with the 15mm crystal. With a 5mm deep crystal there was at most only a 5mm difference in the depth of events (photons that do not strike the crystal when passing through are not detected). The 5mm difference meant that there was not as much variance in the light response pattern, allowing the characterization values to remain reasonably close for all situations. Furthermore if the

event did occur near the back of the crystal the telemetry of the line of response was not as greatly affect. With the new 15mm deep crystal, these considerations do not hold true.

3 Determining Algorithm Accuracy

When comparing many different methods of the same task it is important to have a common metric to judge the results. A commonly used metric in Radiology is Full Width at Half Max (FWHM) in mm, which measures the width of a distribution of solutions for a data set at one half of the maximum value [Figure 8]. This gives the designer an idea of how accurate the algorithm is in determining the correct position. A small FWHM indicates that the algorithm most commonly selects the correct position or one very nearby; a larger FWHM indicates a broader distribution of solutions.

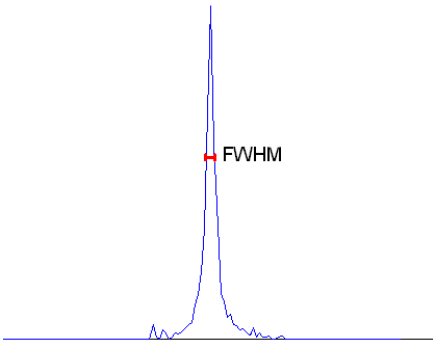

Figure 8. Example measurement of FWHM

There are, however, many issues against the use of FWHM. In many cases the use of only FWHM as a benchmark can be misleading. Figure 9 shows an example of this. If only using FWHM, the distribution on the left and the right would be judged as equals. When viewing the distributions it can be plainly seen that the distribution on the left is clearly better. One can partially address this issue by measuring the full width at other amplitudes as well: 25% of max, 75% of max, etc.

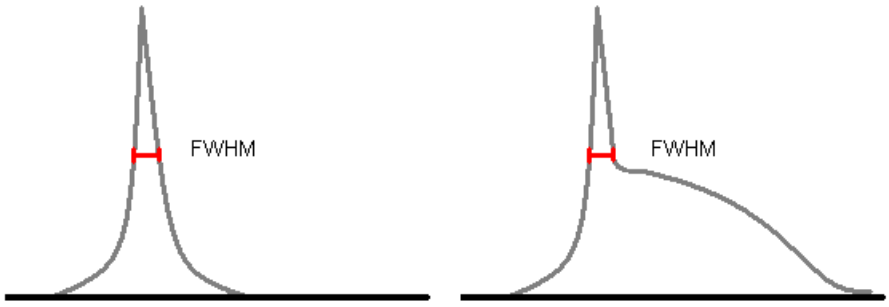

Figure 9. Example of confounding FWHM results

Due to the issues using FWHM, another metric used is average error [Equation 2]. Average error gives an indication of the severity and frequency of the incorrectly derived positions. A small average error indicates that the algorithm produces very good results, while a larger average error indicates poor performance. Average error does have down sides as well. Unlike FWHM, the average error is affected by outliers. A seldom occurring incorrect solution that is far from the correct position can have a great effect on the average error. The same outlier would have no effect on FWHM. Additionally, average error does not give any indication whether the value is based on frequent near misses or occasional outliers.

$$AverageError = \frac{1}{n}\sum_{i=1}^{n}\left|CorrectPosition_i - Solution_i\right|$$

Equation 2: Average Error

The combination of FWHM and average error gives a decent approximation of the distribution. Average error gives an indication of the general accuracy, while FHWM gives an indication of the distribution of the solutions. Comparing the FWHM and average error of different algorithm variations serves as the evaluation criteria when testing algorithms described in this paper

The use of these methods as a benchmark is highly dependent on the availability of data sets which feature multiple occurrences of an event at the same position. There are two methods of collecting such data: simulation and experimental results. In collecting experimental results a fine beam of photons is precisely aimed at the detector module

and the sensor readings are read and stored into a computer. Simulation results are created using software packages designed for the development of radiology systems [5,6].

Simulation results ensure that the event always occurs at the same location under the same set of circumstances; experimental results cannot. Since much of the benchmarking is a comparison against a known solution, it is important to eliminate other variables outside of the algorithm used. For this reason the development of the search algorithms used simulated data.

As a baseline for the development of a real-time SBP algorithm, an exhaustive search is used [Table 1]. The exhaustive search compares every possible point, and then selects the point with the greatest likelihood for the solution. Throughout this work the aim is to increase the throughput of the SBP method while maintaining accuracy similar to that of the exhaustive search method.

Method	FWHM-XY (mm)	FHWM-Z (mm)	Average Error - XY (mm)	Average Error - Z (mm)
Exhaustive SBP	0.9	1.88	1.67	1.74

Table 1: Accuracy of exhaustive SBP method

4 Field Programmable Gate Arrays

A Field Programmable Gate Array (FPGA) is an integrated circuit that consists of programmable logic blocks and programmable routing. An FPGA allows a range of hardware to be executed on it by configuring the logic blocks and programmable routing to execute the desired operations. Along with simple logic units, FPGAs can also have embedded memory and arithmetic units. The hardware nature of using an FPGA can offer near ASIC performance while maintaining the flexibility of a microprocessor.

With the high software execution time of the positioning algorithm and the iterative nature of a hierarchical approach, the SBP algorithm proves to be a good candidate for hardware implementation. An FPGA will allow the hardware to be developed at a significantly reduced cost over the development and fabrication of an ASIC. The

reconfigurable ability of an FPGA will also allow the hardware designed for this PET system to be reused if the detector module or some other portion of the project changes.

Throughout the development of the 3D algorithm, a commercially available development board is used to aid design and establish hardware limitations. The development board used is the DSP development kit, Stratix II edition. As well as featuring the larger S180 variant of the Stratix II FPGA, the development board features several other useful components. Of particular interest are 32MB of SDRAM, 16MB of Flash, and 1MB of SRAM.

A custom board is currently under development which is being designed for this specific application. However, at the time of design for this algorithm the specifications regarding memory capacity and other hardware limitations have not been finalized. For this reason the work presented in this thesis is based on the hardware limitations established by the development board.

5 Two Dimensional SBP Implementation

In Don Dewitt's master's thesis an FPGA was used to implement a 2D high-throughput variation of the SBP algorithm [8]. A structured hierarchical search was used as the basis of this approach to drastically reduce the number of calculations required in producing a solution. The first step in implementation was modifying the core equation [Equation 1] to be better suited for hardware. This process was performed in two steps: converting to fixed point numbers and reorganizing the equation for easier hardware implementation.

Fixed point numbers are significantly less computationally intensive to calculate than their floating point counterparts. The advantage of floating point numbers is the greater range of values which they can hold. Since the range of the values used in the SBP calculations are very small, the full range of floating point numbers is not necessary. The values are converted to fixed point numbers by multiplying by powers of two until the necessary precision has been obtained, while the remaining bits are truncated. When choosing word size for the each value it is desired to keep the word as small as possible while still achieving the necessary precision. A small word size means less hardware

required to perform the calculation and less memory to store the value. For the 2D implementation a 16 bit word size was used for the all characterization values.

Certain operations are easier to perform in hardware than others. Addition and subtraction are both very easy and efficient to implement, capable of one execution per clock cycle. Multiplication is somewhat more computationally intensive. Division is very computationally intensive and should be avoided. Transcendental functions such as trigonometric functions, logarithms and exponents are all very computationally intensive and should be avoided where possible.

Since many values used in the calculation are stored in a lookup table, there is the possibility of precalculating some values before storing them. Doing so reduces the number of operations that must be performed on the FPGA, increasing throughput. There are several cases in the basic equation [Equation 1] where this is possible. The standard deviation value used in the equation is involved in two computationally intensive operations: a multiplication and a natural log. Since these operations are performed on only the standard deviation value and not any external independent variable they can be precomputed. In place of being stored as simply the standard deviation they are stored as $\frac{1}{\sqrt{2}\times\sigma}$ and ln(σ). While this will essentially store the variance value twice, eliminating a natural log operation in hardware was found to be a worthwhile trade off.

The implementation uses a hierarchical search to reduce the number of computations required to produce a solution. The hierarchical search is composed of a number of stages beginning with a coarse grained search stage, focusing and progressing through finer grained search stages until a solution is found. Each search stage compares nine different points equally distributed across the search area. The likelihood of each point is calculated and the most likely of the nine points is selected as the center point for the next more refined search [Figure 10]. Comparing 9 different points in each search stage allows the hierarchical SBP algorithm to solve for the positioning of a 127x127 grid in six stages. In total, 54 data point computations are required, which is a drastic reduction from the 16,129 data point calculations needed for an exhaustive search.

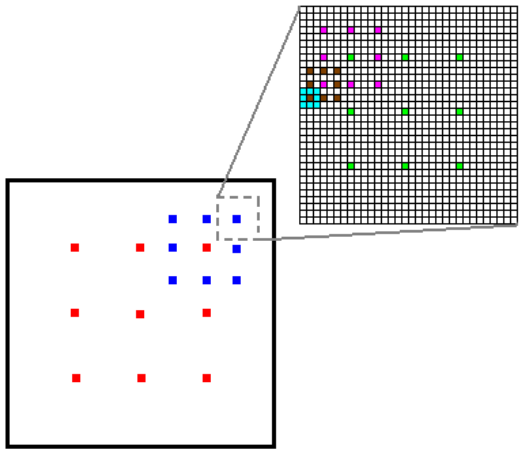

Figure 10. Example of 2D hierarchical search

Testing showed that a 9 data point search per stage was a good balance between accuracy and throughput. In each search, the 9 points were distributed so that they were equally spaced apart. The distribution also allowed sufficient overlap in the paths that the search could take. In the 9 point implementation the search can backtrack in the finer search stages to the point directly next to the nearest other selection in a coarse stage.

Each search stage is independent, and once given a starting point the stage can execute without any other input from the other stages. Since each stage can perform its search independent of the others, the six stages which make up the search were pipelined. Pipelining allowed a significant improvement in throughput. In place of providing a solution every 54 calculations, pipelining allowed the algorithm to provide a solution every 9 calculations, the throughput of a single stage [Figure 11].

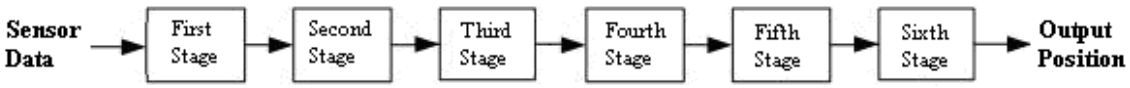

Figure 11. Outline of hardware implementation of 2D search

In order to allow each stage to operate independently, each stage requires its own set of characterization value tables. However, since the search is hierarchical not every stage requires a full set of tables. To save memory each stage is given a table that only contains values that it may be required to access. The first stage was given a table with 9 data points, while the second stage was given a table with 36, etc.

6 Design of a Three Dimensional Statistical Positioning Algorithm

Many of the same ideas and approaches from the 2D implementation were carried over to the new 3D implementation. The same equation reductions from the 2D implementation were used in the 3D implementation to reduce the computational intensity of finding the likelihood of a data point. As a basic approach, the hierarchical search method was also used. Since the 3D implementation adds an extra dimension, and the detector module is slightly different, some changes were needed.

The new photodetector array is much more dense. In place of the 8x8 array, a 16x16 array is used. Handling each sensor value individually, as done with the 8x8 array, would increase bandwidth requirements between the detector and the FPGA to a point that was believed to be unfeasible. As a result of this, row-column summing is used. In row-column summing the sum of each sensor in each row and column is used in place of individual sensor values. For the 16x16 array this corresponds in a reduction of from 256 values to 32 (16 rows and 16 columns). In SBP, these row and column values can be used in place of the individual sensor values [Equation 3].

$$\ln(P(event,X,Y,Z)) = -\sum_{row=1}^{16}\left[\frac{\left[event_{row}-(\mu_{X,Y,Z})_{row}\right]^2}{2\left[(\sigma_{X,Y,Z})_{row}\right]^2}+\ln\left[(\sigma_{X,Y,Z})_{row}\right]\right] - \sum_{column=1}^{16}\left[\frac{\left[event_{column}-(\mu_{X,Y,Z})_{column}\right]^2}{2\left[(\sigma_{X,Y,Z})_{column}\right]^2}+\ln\left[(\sigma_{X,Y,Z})_{column}\right]\right] \quad \text{E}$$

Equation 3. Likelihood for row column summing

For 3D SBP, the module is characterized for 127x127x15 positions, with 127x127 representing the X and Y dimensions, and each X and Y position is characterized for 15 depths. As with the 2D implementation, each XY position is on a grid with each point

separated by .3875mm. The 15 depths are not equally distributed over the 15mm deep crystal; depth is characterized at a higher resolution near the surface, where events are most likely to occur [Table 2, Figure 12]. In total the module is characterized for 241,935 different positions, 15 times more than 127x127 2D implementation.

	Start Depth (mm)	End Depth (mm)
Level 1	0.0000	0.8275
Level 2	0.8275	1.4131
Level 3	1.4131	2.0290
Level 4	2.0290	2.6784
Level 5	2.6784	3.3653
Level 6	3.3653	4.0943
Level 7	4.0943	4.8709
Level 8	4.8709	5.7016
Level 9	5.7016	6.5948
Level 10	6.5948	7.5604
Level 11	7.5604	8.6113
Level 12	8.6113	9.7640
Level 13	9.7640	11.0404
Level 14	11.0404	12.4703
Level 15	12.4703	15.0000

Table 2. Depth voxel spacing

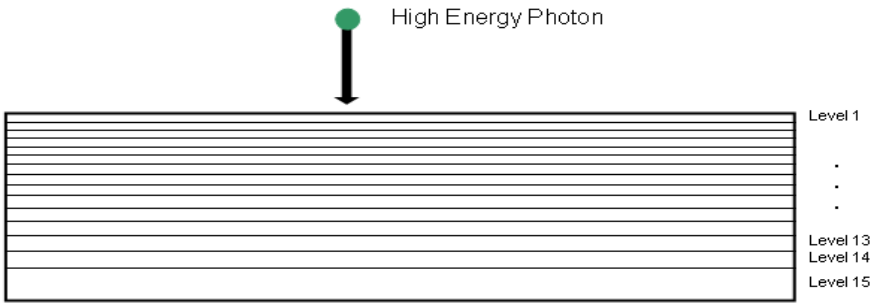

Figure 12. Spacing of DOI levels

The simplest way to extend the 2D hierarchical search to a 3D hierarchical search is to expand some of the stages in the previous 2D algorithm to 3D stages. The 3D search treats the depth of interaction as an independent variable (just like the X and Y dimensions), and adds the depth variable to the search algorithm. A 3D stage uses a volumetric search, performing the same basic comparison operation as a 2D stage but comparing a distribution of points across the X, Y, and Z dimensions in place of just the X

and Y dimensions [Figure 13]. The 3D stage finds the best fit combination of X, Y, and, Z within the stage's distribution of points.

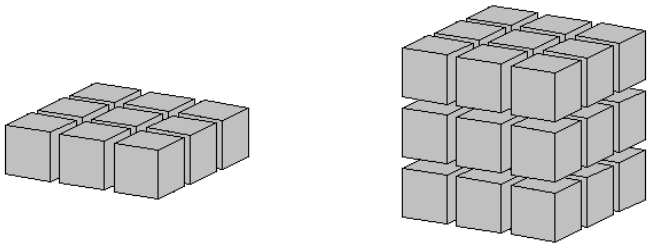

Figure 13: Grouping of voxels in a 2D search (left) and grouping of voxels in a 3D search (right)

Functionally, a 3D stage works similar to a 2D stage with one major difference: a 3D stage uses a three dimensional characterization table in place of a two dimensional characterization table. In a 3D table, values are dependent on all three dimensions rather than in a 2D table where X and Y values are averaged according to the expected distribution of depth. The 3D table allows a 3D stage to examine independent points across all three dimensions and select the best fit point in X, Y, and Z. The computation process in a 3D stage is the same as in a 2D stage; the likelihood of each point is calculated with the best fit becoming the center point in the more refined search carried out in the next stage. Since a 3D search compares the same XY points of a 2D search at multiple depths, a 3D search will consider a greater total number of points than the comparable 2D search.

For the given system, the depth of interaction is defined by 15 depths. As with the XY planar search, the depth search can be divided into a hierarchical search, reducing the number of computations required to produce a solution. Dividing the 15 levels equally into a hierarchical search would allow the depth solution to be found in only 3 stages, with each stage comparing three depths [Figure 14]. Since the existing pipeline has six stages, only half of the stages will be converted to 3D stages. Stages not converted to a three dimensional search will remain two dimensional to complete the XY search. With only half of the search stages required to be volumetric, the order of those stages in the pipeline becomes a variable for optimization of accuracy and memory requirements.

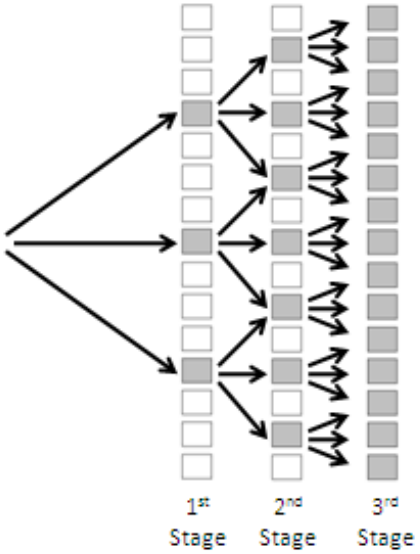

Figure 14: Outline of hierarchical search to find depth among 15 levels

Planar searches following any volumetric stage(s) in the pipeline will also use three dimensional characterization tables. The depth of interaction found in the preceding stages is used as a constant with the 3D characterization table producing, a 2D dimensional characterization table matched to the calculated depth of interaction. The matched table offers the advantage that characterization values from the table are based on a specific depth, which is more accurate than a 2D table with values based on an average depth.

The principle downside to the volumetric search stage approach is the increased number of computations required. As outlined above, each volumetric search in the pipeline performs a 3x3x3 search, 3 times as many calculations as the original 2D search. With memory bandwidth limitations restricting stages with off chip memory to operating on one point at a time, the 3D stage has a critical path of 27 calculations. Replacing a 2D stage with a 3D stage increases the critical path length in the pipeline to that of the 3D stage, reducing throughput by a factor of three. This limitation does not exist if all three volumetric searches can use on-chip memory (which has much higher bandwidth limitations).

The memory requirement for a 3D search is also drastically increased. Each volumetric search and each 2D stage following a volumetric search requires a three dimensional table. The size increase of a 3D table over a 2D table is dependant on the number of levels included. For example, a 3D table with all 15 depths requires 15 times the memory of the corresponding 2D table. Since the depth search is organized as a hierarchical search, not every volumetric search requires a full 3D table. The table for the first of the three volumetric stages is increased by a factor three, the second by a factor of seven, and the third by a factor of fifteen [Figure 14]. Any 2D stage following a 3D stage is required to have at least as many depths as the preceding 3D stage. With the 2D sixth stage table already requiring close to 3MB, converting to 3D tables is a significant increase in capacity requirements.

To evaluate the tradeoffs, the algorithm was tested with the volumetric search stages placed in different locations in the pipeline to study the effects of volumetric search placement compared to system accuracy [Table 3]. The results show that placing the depth search too early or too late in the pipeline is detrimental to system accuracy across all dimensions. When volumetric stages were placed early, the X and Y resolution proved to be too coarse/inaccurate to provide a good basis for a depth search. When placed late, results indicated that many of the calculations early in the pipeline were inaccurate due to a lack of information on the actual depth of interaction. Placing the volumetric searches toward the middle of the pipeline allowed a sufficiently accurate XY basis for calculating the depth while producing a depth solution early enough to increase the accuracy of the finer-grained XY searches. The best combination tested places a volumetric search third, fourth, and fifth in the pipeline.

Location of Volumetric Searches	FWHM-XY (mm)	FHWM-Z (mm)	Average Error-XY (mm)	Average Error-Z (mm)
Stages 1,2,3	1.03	3.49	1.78	2.52
Stages 2,3,4	0.95	1.95	1.68	1.78
Stages 3,4,5	0.91	1.87	1.68	1.74
Stages 4,5,6	0.92	1.88	1.68	1.73

Table 3: Simulation of different volumetric search locations

Placing a volumetric search third, fourth, and fifth in the pipeline produces very good results; however there are several hardware limitations which prohibit its use. The added memory requirement of the volumetric searches would exceed the capacity of the on-chip memory currently available, which would move the later volumetric stage's table off-chip where memory bandwidth is much more limited. Total memory capacity is also a problem since the sixth stage table is required to be fifteen times larger. The number of computations required for each volumetric search is also prohibitive. Given the current hardware design, there is insufficient memory bandwidth to perform the 27 calculations required for a volumetric search and meet the throughput requirements of the system. The existing hardware will only permit the 27 calculations necessary in a volumetric stage to be computed in approximately 500 clock cycles, yielding a throughput of 140,000 events per second. This is 0.7x the required throughput.

To increase the throughput of the system, the volumetric search can be split into two independent stages: an XY planar search followed by a linear Z search. Using this method the algorithm will alternate between a 3x3 planar search and a 3 depth linear search. The two stages combined do not perform the same operation as the volumetric search; in place of comparing all possible points (X,Y,Z) the algorithm first solves for XY, and then Z [Figure 15]. This change dramatically increases the throughput of the algorithm, reducing the 27 operations to 9 operations.

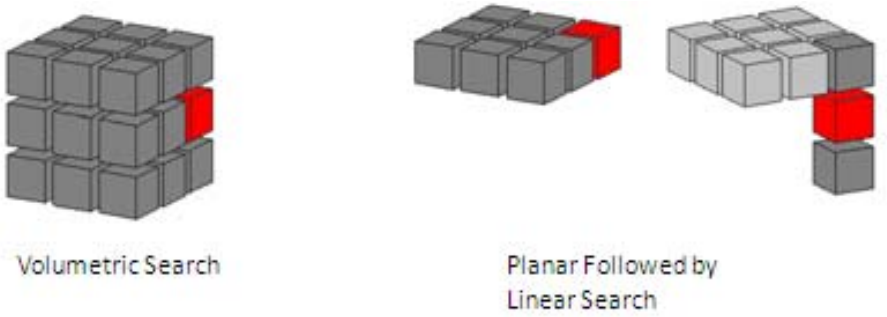

Figure 15. Outline of a planar/ linear search stage

Both the planar search and the linear search will use 3D characterization tables. The planar search will use the depth information found in the preceding stages as a constant, solving for the best fit XY at the given depth. The linear search will use the XY information found in the previous stage as a constant, solving for the best depth fit at the given XY.

The additional 3 independent depth search stages significantly increases hardware requirements over the single volumetric search stage. By having both the planar and linear searches act independently as separate stages, the memory bandwidth and capacity requirements are significantly increased compared to the volumetric search, as each stage would require its own characterization tables and perform their calculation independently.

At the cost of some throughput, the increase in required memory bandwidth and capacity requirements can be addressed. The independent planar search and linear search can be grouped as a single stage performing two operations. In this configuration the stage first performs a planar search, followed by the linear search. This allows the planar search and linear search to share one memory table. The cost of this is an increase in the critical path caused by the addition of the 3 linear search calculations to the planar stage. Compared to the original volumetric search approach first described, this variation allows an algorithm with no additional memory bandwidth or memory capacity requirements, but with a greater than 2X increase in throughput.

Using the same test setup as the volumetric search algorithm, several orderings of the planar/linear stages were compared [Table 4]. As was seen in Table 1 there is a significant quality gain when performing the depth search after the first stage and little to no gain in beginning the depth search after the second stage. The best option is placing a hybrid search stage second, third and fourth in the pipeline. Compared with the best option from table 1, the hybrid search stage performs similarly, with slightly improved depth resolution and slightly poorer XY resolution.

Location of Linear/Planar Stages	FWHM-XY (mm)	FHWM-Z (mm)	Average Error-XY (mm)	Average Error-Z (mm)
Stages 1,2,3	1.20	3.52	1.78	2.51
Stages 2,3,4	0.95	1.95	1.68	1.77
Stages 3,4,5	0.90	1.87	1.68	1.73
Stages 4,5,6	1.02	1.90	1.70	1.72

Table 4: Simulation of different planar/linear search locations

Another option available to increase throughput is placing the depth search by itself in a single stage, performing the entire depth search at once. Using the previously outlined

hierarchical depth search, the DOI can be found in a single stage in 9 calculations. Given that a hierarchical search can find the depth in 9 calculations, placing the depth search in a single stage maintains the original 2D search's critical path length of 9 calculations. This change offers several other benefits without much cost. By compressing all of the depth searches into a single stage the memory tables needed for the coarser grained depth searches can be eliminated, reducing the capacity requirement of on-chip memory.

Table 5 shows the simulated results of the algorithm with the linear depth search placed at different locations in the pipeline. Compared with the results from table 4 it can be seen that performing the entire depth search in a single stage has little effect on the accuracy of the system. Based on the results shown in Table 5 there is no benefit in performing the depth search after the fourth stage of planar XY search. Since the depth search characterization table size increases by a factor of 4 for each planar search preceding it, the depth search is ideally placed after the fourth XY search, where a high degree of accuracy is achieved without requiring a larger memory than necessary. However, due to the size of the 15 depth table that would be required for a depth search after the fourth stage, this is not possible. Moving the depth search back a stage reduces the required table size to an amount that can fit on the available on-chip memory, allowing the algorithm to be implemented on the development board.

Location of Linear Depth Search	FWHM-XY (mm)	FHWM-Z (mm)	Average Error-XY (mm)	Average Error-Z (mm)
After Stage 1	1.25	3.82	1.98	4.95
After Stage 2	1.00	2.36	1.71	2.25
After Stage 3	0.90	2.02	1.68	1.81
After Stage 4	0.91	1.89	1.68	1.75
After Stage 5	1.14	1.92	1.71	1.77
After Stage 6	1.09	1.91	1.77	1.74

Table 5: Simulation of different single stage linear depth search locations

A common problem to all the previously described methods is the large memory capacity required. The full data tables for the later stages of the algorithm are quite large, with the Stage 6 15-depth table requiring 45MB of storage space. The total memory requirement for the system exceeds 60MB, an amount greater than the development board capacity.

While not required, an XY planar search can benefit from the use of a three dimensional characterization table over a two dimensional one. A three dimensional characterization table allows a better match between sensor outputs and table values based on the known depth rather than an average or assumed depth. However, not all 15 depths are required to take advantage of this. In the case of stage 4, 5 and 6 the full data table is not required, and a simplified three dimensional table can be used which has a reduced number of levels. This reduction saves memory at the potential cost of lower accuracy.

Table 6 shows the algorithm accuracy based on the number of depths used in each stage following the depth search stage. A comparison between single depth tables (number of depths = 1) and multi-depth tables illustrates the advantage of using three dimensional tables for the final stages of the XY planar search. Also illustrated in the table is a demonstration that the full 15 depth tables are not needed to see an increase in accuracy when using 3D tables for a planar search.

Number of Depths in Stage 4 Table	Number of Depths in Stage 5 Table	Number of Depths in Stage 6 Table	FWHM-XY (mm)	Average Error-XY (mm)
1	1	1	1.09	1.77
1	1	2	1.19	1.73
1	1	4	1.15	1.72
1	1	8	1.11	1.71
1	1	15	1.13	1.71
1	2	2	1.00	1.72
1	**2**	**4**	**1.02**	**1.70**
1	2	8	1.04	1.69
1	2	15	1.06	1.69
1	4	4	0.99	1.70
1	4	8	1.01	1.68
1	4	15	1.00	1.68
1	8	8	1.01	1.68
1	8	15	0.98	1.68
1	15	15	0.96	1.68
2	2	2	1.00	1.72
2	2	4	0.97	1.70
2	2	8	1.00	1.69
2	2	15	0.99	1.68
2	4	4	0.95	1.69
2	4	8	0.93	1.68
2	4	15	0.92	1.68
2	8	8	0.92	1.67
2	8	15	0.92	1.67
2	15	15	0.90	1.67
4	4	4	0.95	1.69
4	4	8	0.95	1.68
4	4	15	0.91	1.68
4	8	8	0.93	1.67
4	8	15	0.90	1.67
4	15	15	0.90	1.67
8	8	8	0.92	1.67
8	8	15	0.90	1.67
8	15	15	0.90	1.67
15	15	15	0.90	1.67

Table 6. Simulation of accuracy as a function of table depth in final stages

In general, the trend shows that the greater number of levels that can be used, the greater the accuracy [Figures 16, 17]. However, the results indicate that the number of depths used is subject to diminishing returns, where significant improvements in accuracy correlate with larger tables up to a certain point when accuracy is not gained as readily.

In the best case of high accuracy with a reduced table size, 4 depths are used in stage 4, 8 in stage 5, and 15 in stage 6. This corresponds to the following memory requirements: 768KB for the 4th stage, 6MB for the 5th stage, and 45MB for the 6th stage.

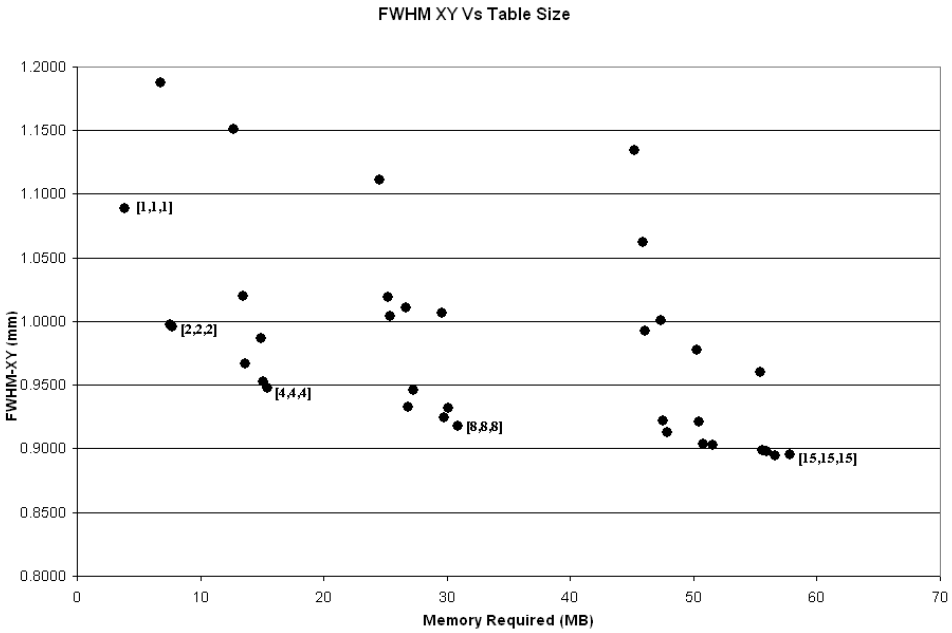

Figure 16. FWHM as a function of table size [4th table depth, 5th table depth, 6th table depth]

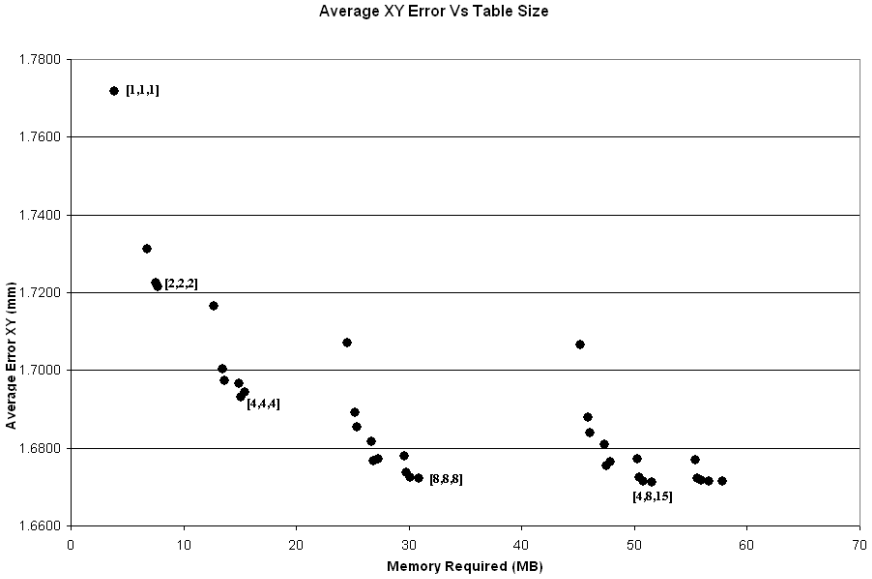

Figure 17. Average error as a function of table size [4th table depth, 5th table depth, 6th table depth]

However, the development board cannot support such a large memory requirement. The general trend of the table is that the more depths used, the greater the accuracy. Following this general trend, the best combination will be the most detailed characterization tables that can fit on the development board. After allocating some of the on-chip memory for the first three stages and the depth stage, roughly 0.2MB remains for the fourth stage. This is sufficient space for a single depth 4th stage table. The off chip SRAM has 32KB of addressing, allowing at most a 2 depth 5th stage table. The 500KB of addressing for the SDRAM memory is sufficient to allow an 8 depth 6th stage table. Although an 8 depth table can be used for the sixth stage, a 4 depth table produces better results.

The recommended number of depths used for the development board implementation uses 1 depths for stage 4, 2 depths for stage 5, and 4 depths for stage 6 (highlighted in Table 5). While this solution is not ideal, it is a significant improvement over using a single depth for the final stages and also represents a significant savings in memory. Reducing stage 4 from a 15 depth table to a 1 depth table saves 2.6MB, while the reduction in stages 5 and 6 save 9.75MB and 24MB respectively. The reductions amount to a total 58% lower memory capacity requirement for the system.

Combining observations from the previous tests gives the basic framework of the three dimensional algorithm [Figure 18]. The first three stages are XY planar searches, similar to ones used in the original two dimensional algorithm. The depth of interaction is found in a linear search after the first three stages and the depth is then used as part of the planar searches in stages 4, 5 and 6. The depth remains unchanged through the last three stages of the pipeline.

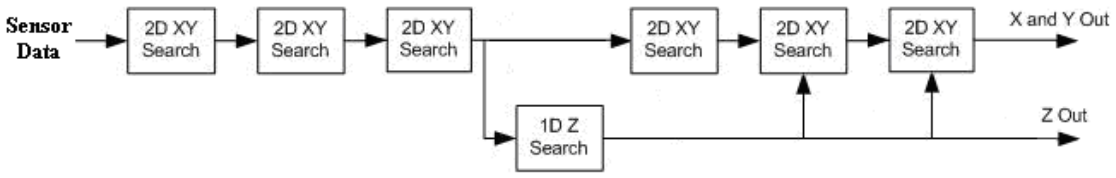

Figure 18. Basic framework of 3D SBP hierarchical search

7 Other Methods of Finding the Depth of Interaction

In place of extending the SBP method to find the depth of interaction, several other methods were investigated as part of the algorithm development. The primary advantage to using a different approach is avoiding the characterization tables and the necessary memory bandwidth and capacity associated with using SBP. In the 2D implementation FPGA resources were already stretched, and finding the depth of interaction without requiring additional memory was an alluring prospect. It was found that based on the pattern or values from the sensor array there was several different ways to calculate the depth of interaction. In general events that occurred near the surface of the crystal, close to the sensor array, gave very high readings over a few sensor rows and columns. Events that were near the back of the crystal, far from the sensor array, gave lower sensor readings, but more rows and columns saw the affect of the event. Observation of those properties led to a study of the feasibility of such an algorithm.

The simplest approach looked at the highest reading sensor row and column. On the basis that events near the surface of the crystal gave high sensor readings and events towards the back of the crystal gave very low sensor readings it was proposed that it could be used to determine the depth of interaction. Using a large sample of simulated data that covered the entire crystal, the maximum row and column value were averaged for each depth. Using these numbers, thresholds were defined that would allow the depth to be found based on maximum row and column value.

Testing showed that finding the depth based on a row and column value worked well in some situations, but over the entire data set did not [Table 7]. In particular points near the edges did not perform well.

Region of Crystal	FWHM-Z (mm)	Average Error (mm)
Center	3.50	2.48
Edge	3.39	2.86
Corner	3.67	3.58
Entire	3.47	2.75

Table 7. Accuracy of finding DOI based on peak sensor row and column

Based on the poor performance of finding the depth from the maximum row/column in certain areas of the crystal, a new variation was tested. The area of the crystal was divided into several sections: center, edge and corner. For each of these sections a large sample of simulated data was used to find the average maximum row and column for each depth. The aim of this approach is to provide more accurate depth thresholds based on the location of the event on the crystal. In implementing this approach the depth of interaction is found after a few stages of 2D SBP search, using the found position to determine which depth table to use. Testing of this approach showed some improvement, but could not match the results of 3D SBP [Table 8].

Region of Crystal	FWHM-Z (mm)	Average Error (mm)
Center	3.40	2.86
Edge	3.40	2.77
Corner	3.51	2.60
Entire	3.41	2.80

Table 8. Accuracy of finding DOI based on the sum of multiple sensor rows and columns

For each depth, the variance of the sensor value for the highest reading row and column indicated that basing depth on a single row and column would never provide very accurate results. In an attempt to reduce the inaccuracy of this approach it was investigated whether using multiple rows and columns would improve results. The approach is largely the same; however, in place of a single sensor row and column, the sum of multiple rows and columns is used. The aim of this approach is that outliers are averaged across multiple rows and columns and have a less profound impact.

Testing was used to find the optimal number of rows and columns used for this method. Testing found that accuracy actually decreased when using the sum of many rows and columns [Table 9]. Furthermore, more than 8 rows and columns could not be used as the sum approached a constant for all depths. Based on these results and those found in table 8, finding depth as function of peak sensor(s) will not work well for this application.

Number of Rows/Columns	FWHM-Z (mm)	Average Error (mm)
4	3.55	2.80
6	5.44	5.93
8	4.28	3.33

Table 9. Accuracy of finding DOI based on sum of multiple rows and columns

Rather than using sensor values themselves, another approach used the number of sensor rows and columns that read above a certain threshold. In this approach the depth as a function of how many sensor rows and columns are affected by an event was examined. As seen earlier in this section, events that occur very near the sensor array only affect a small number of sensors, while events that occur farther from the sensor array affect a much greater number. Using a large sample of simulated results a threshold was found which was high enough not to be affected by noise yet low enough to be used for events which occur far from the sensor array. The threshold was used to determine whether a sensor row or column was affected. One problem with this approach is the number of sensors above the threshold when an event is near an edge does not follow the same pattern as at the center of the crystal [Figure 15].

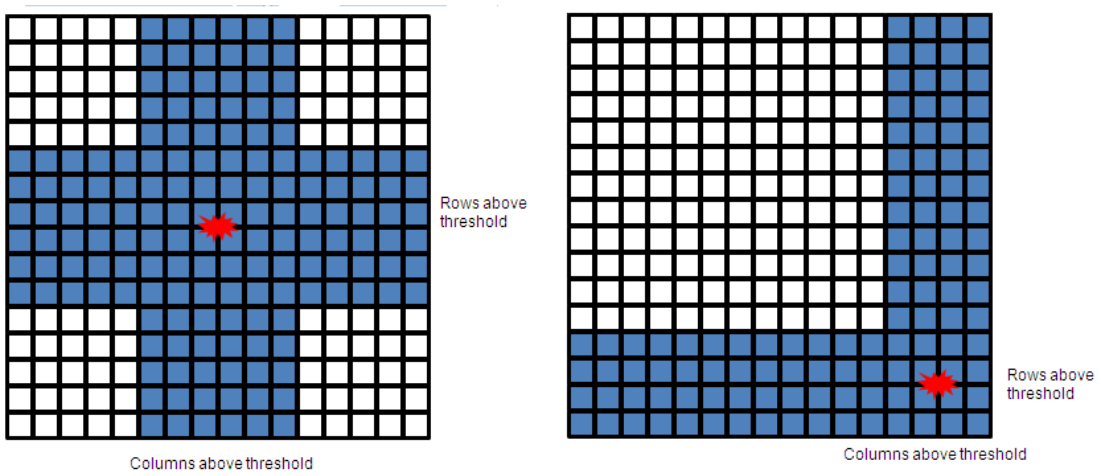

Figure 19: Rows and columns above threshold for different event locations

To counter this effect, the crystal was again divided in different sections: center, edge, and corner. Each section was then tested to find how many rows and columns were above the threshold for each corresponding depth. Basing the depth of interaction on the number of sensors above a certain threshold worked for events in the center of the crystal, however,

it did not work well for positions near the edge. Even more exhaustive attempts at fitting tables to each situation on the edge of the crystal did not significantly improve performance.

8. Algorithm refinement

As described in the fourth section, the algorithm produces very good results in a drastically reduced number of calculations. However, in its current state the algorithm is not able to meet the throughput goal of the project. The next series of modifications deal with improving throughput while maintaining accuracy and improving accuracy near the edges of the crystal.

8.1 Throughput Improvement

As part of the 2D hierarchical search, the algorithm compares 9 different points in a stage. After selecting the point with the maximum likelihood, that point becomes the center point of the new, more refined 9 point search. In the previous implementations, the point was recalculated as part of the new nine point search. By restructuring the implementation and passing the likelihood of the selected point to the next stage, that point no longer has to be recalculated. Using this approach the number of data point calculations required in a 2D search stage is reduced from 9 to 8, resulting in an improved throughput of 12.5%. In order to implement this, an extra stage is added at the beginning of the pipeline. This stage simply calculates the likelihood of the center point and passes it to the next stage.

The first variation is based on observations of the sensor array outputs. When viewing the row-column sum of the sensor array it is apparent that not every row and column is affected by an event. For an event which occurs very near the sensor array, on average only 5 rows and 5 columns show a significant output [Figure 20 left]. For events very far from the sensor array, the nearest 11 rows and 11 columns show a significant output [Figure 20 right]. As originally outlined the likelihood of each event is the sum of the

likelihood of each row and each column. Reducing the number of rows and columns included reduces the computational intensity of calculating each data point's likelihood.

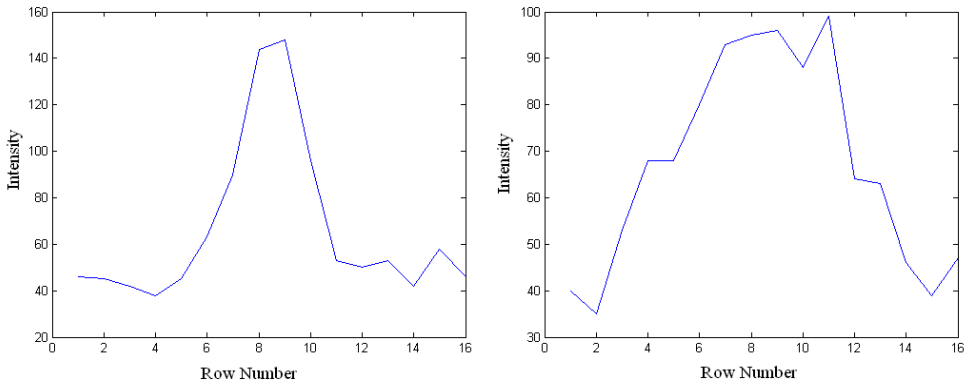

Figure 20. Row outputs for an event near the sensor array (left) and for an event far from the sensor array (right)

In order to reduce the number of row and column likelihood calculations, the equation was modified to only calculate a certain number of rows on either side of the highest reading row, and columns on either side of the highest reading column [Figure 21]. To ensure that this modification would not have a detrimental effect on system accuracy several variations of this approach were tested and compared to the results of using all rows and columns. Each variation uses a different number of rows and columns, with the goal of finding the least number of row and columns needed that produces a comparable result.

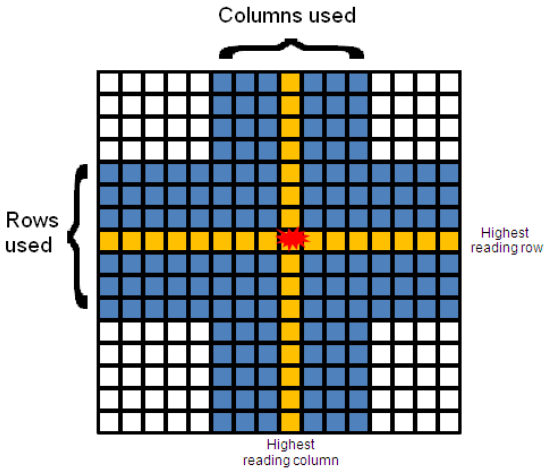

Figure 21. Example of selecting highest reading row/column and nearest six row/columns

Testing showed that up to a certain point accuracy is only slightly affected by reducing the number of rows and columns used in finding the likelihood [Table 10]. This is due to the primarily noise signals in channels which occur far from the event. Since each row and column carries equal weight, removing rows and columns which mainly contain noise removes a lot of noise from the system. This is most apparent in events that occur near the corners of the crystal, as the ratio of relevant signal channels to noisy signal channels is much lower.

Number of Row Columns Used	FWHM-XY (mm)	FHWM-Z (mm)	Average Error-XY (mm)	Average Error-Z (mm)
2	2.43	2.99	3.98	2.17
6	1.07	2.53	2.73	2.00
10	0.99	2.21	2.05	1.98
14	0.99	2.11	1.82	1.90
18	0.99	2.02	1.74	1.85
22	**0.97**	**2.07**	**1.73**	**1.83**
26	0.97	2.01	1.70	1.82
30	0.99	2.02	1.69	1.82
All	1.00	2.02	1.69	1.81

Table 10: Results with reduced number of row columns used

Although using all rows and columns produces the best results, reducing the number of rows and columns used offers a significant improvement in throughput with a relatively small penalty in accuracy. The general trend, as seen in table 10, indicates that the more rows and columns used the higher accuracy. Observing this trend, the best variation is the one that uses the greatest number of rows and columns yet still allows the algorithm to meet its throughput goals.

Based on the results of testing, using the nearest 11 rows and 11 columns (22 total) was selected as a balance between throughput and accuracy. This modification has a profound impact on the algorithms throughput. Using every row and column to find a data point's likelihood involved 288 calculations. With the preceding modifications this number is reduced to 198 calculations for a point in the center of the crystal. A point near the corner of the crystal can be reduced to as few as 108 calculations, since even fewer rows and columns are used when an event occurs along an edge [Figure 22]. Averaging for an even

distribution of events across the crystal, the algorithm requires an expected 160 calculations per data point, a reduction of 44% over using all 32 rows and columns.

Of concern in the previous variation was the situation that occurred when the highest reading row or column was located along the edge of the crystal. In this situation, events located on the edge of the crystal would often have few or no rows/columns on one side of the highest reading row/column. In that case the calculation is performed with less information than other parts of the crystal. Two possible solutions to this problem were investigated: when near the edge of the crystal, expand the area viewed to include the desired numbers of rows and columns, or count certain rows and columns twice to make up for lost information. However, testing showed that these modifications were actually detrimental to accuracy or had little effect. The algorithm was implemented so that in the event that the highest reading row or column was located along an edge only sensor row/columns within the specified distance where used. In this arrangement likelihood calculations of points along the edge could use fewer total row/columns than a point in the center of the crystal.

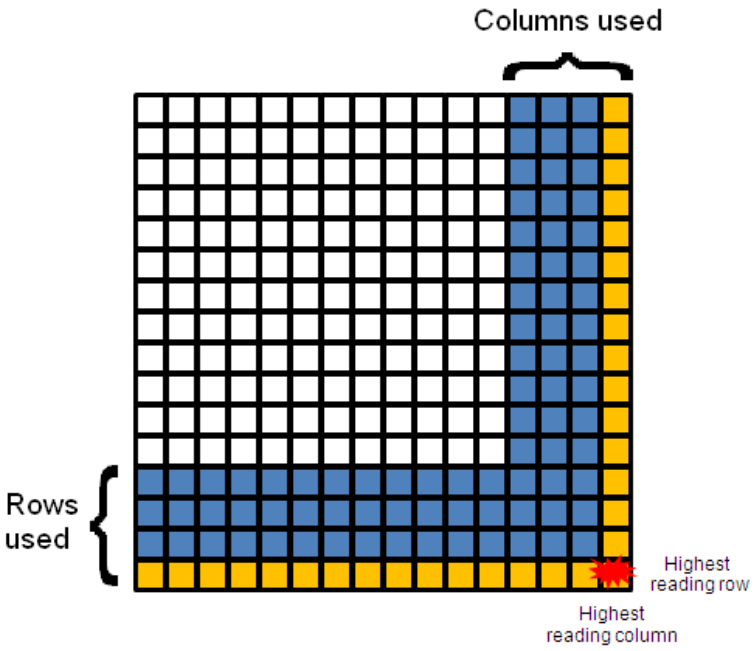

Figure 22 Example of selecting highest reading row/column and nearest six row/columns in corner of crystal

8.2 Reduction of Multiple Peaking

In the ideal case the solution set forms a hyperbolic parabaloid with a smooth and consistent gradient to the most likely solution. In this ideal case a hierarchical search will consistently produce the same result as an exhaustive search, as any point closer to the absolute minimum has a greater likelihood, allowing the algorithm to always converge to the correct solution. However, in practice the solution set is rarely ideal and often has local minima or an uneven gradient.

There are many reasons behind the unbalanced gradient and local minima: reflections from the edges of the crystal, drift in the gain of the MAPDs, and noise in the system. Reflections from the edge of the crystal are most detrimental to system accuracy. In this case light can reflect from the edge of the crystal, giving the appearance of an event in a different location. Drift in the gain of the MAPD is another cause; in this case the sensor readings behave differently than what was expected when the module was characterized. Lastly, as with any system, noise can have a detrimental effect to the system's accuracy.

The previously described effects can produce local minima, which are very detrimental to system accuracy. A local minimum is a portion of the solution set which breaks from the general gradient of the set and, when viewed in a limited scope, appears to be the absolute minimum of the system. As a result of a local minimum, a hierarchical search can incorrectly select a point farther from the absolute minima in coarse grained stages and eventually converge to the local minima as a solution in place of the true minimum. Generally, the local minima that can trap the hierarchical search are located near the true minimum. Local minima and noise that have a great effect on system accuracy have the highest occurrence on the outer edges of the crystal where light reflections from the edge of the crystal have the greatest impact on the light response function. Examining the results of data sets across a range of positions on the crystal show that events occurring within 5mm of the crystal edge produce significantly poorer results than positions located closer to the center of the crystal.

A high occurrence of the same local minima leads to a tendency to produce incorrect solutions in the same location with a higher than normal frequency. When viewing a large

number of samples for a data point with local minima, the distribution of the results shows multiple peaks: often the correct solution and a peak occurring at each location of a local minimum [Figure 23].

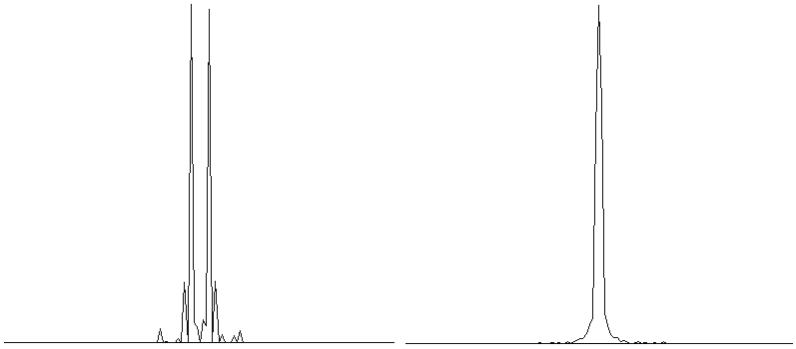

Figure 23: Illustration of multiple peaking (left) compared to the desired solution (right)

Additional peaks, even those close to the correct solution, have a great effect on the usefulness of the PET system. When constructing an image from positioned data it is impossible to determine the difference between a peak caused by a correct event or by a local minimum. Incorrectly identifying a peak caused by a local minimum as a true event leads to image ghosting: an object that appears in the produced image but does not actually exist. Inversely, ignoring closely grouped peaks that might be caused by local minima but are actually different data points can eliminate valuable information. Since image construction cannot differentiate between the two, it is important to eliminate the problem at the source and ensure that the hierarchical search is nominally affected by local minima in the solution set.

As stated above, examination of data sets with local minima shows that local minima have a tendency to occur near the absolute minima. Since the local minima have a tendency to be located near the true minima, a broader fine grain search becomes a viable correction option [Figure 24]. A broader 5x5 search increases the range of the search by 0.3875mm in all directions. Expanding the final search stage from a 3x3 to a 5x5 reduces the occurrence of multiple peaking by 41% and decreases average error by 0.011 mm over the entire range of positions. A 7x7 search increases the range of the search by 0.775 mm. While a 7x7 doesn't further decrease the rate of dual peaking, expanding the final search

stage to a 7x7 search reduces the average error by an additional 0.001mm compared to the 5x5. When expanded to a 9x9 search there is little benefit over a 7x7, with only a slight decrease in average error.

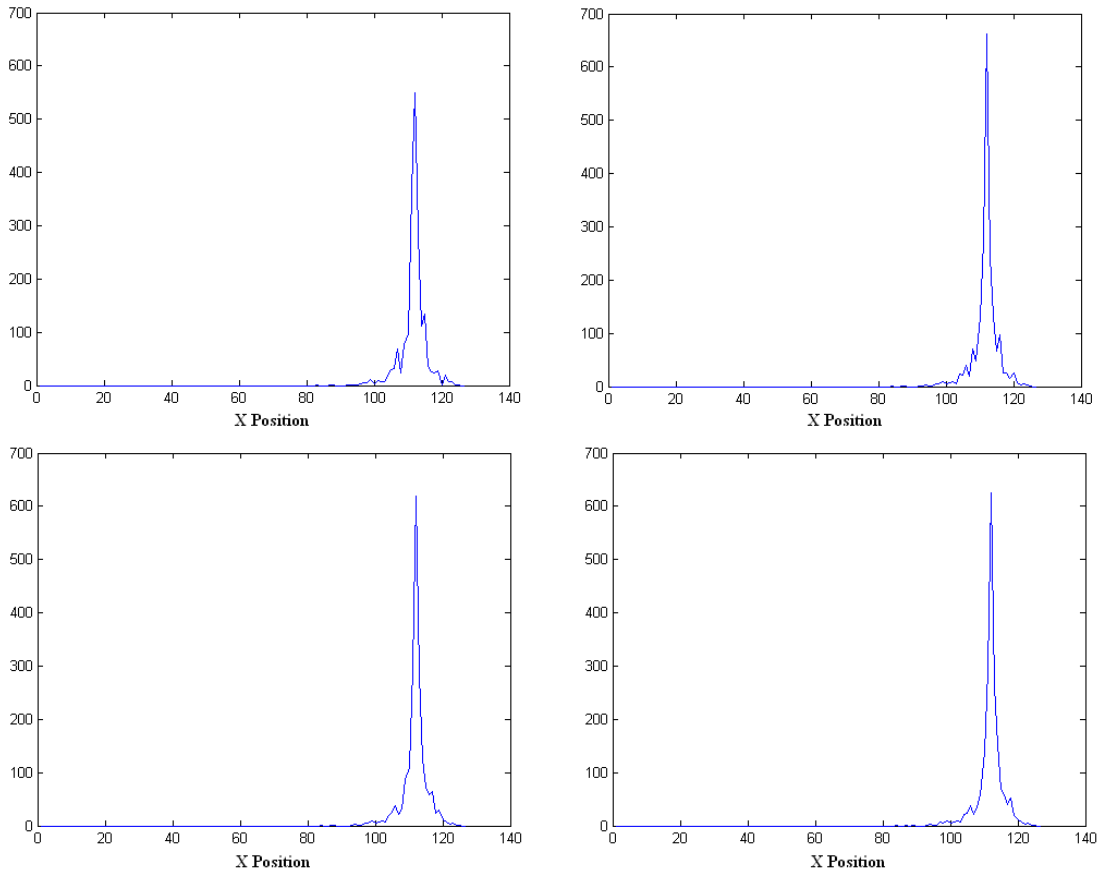

Figure 24: Example of X distribution for 2000 samples of a point near the corner of the crystal. Different final stage search breadths are used for each distribution: 3x3 (top left), 5x5 (top right), 7x7 (bottom left), 9x9 (bottom right).

Further analysis shows that positions located in the corner of the crystal are more greatly affected than those that are near only one edge. Nearly every data point within 5mm of the corner had instances of multiple peaking. If the data point is located along a single edge the occurrence is generally much lower and less severe. When focusing on an area 5mm from a corner a 5x5 search reduces the instances of dual peaking by 36%, a 7x7 reduces the rate by 68%, and a 9x9 reduces the rate by a comparable amount. Similar performance gains are also seen in average error: a 5x5 reduces average error by

0.09mm, a 7x7 reduces average error by 0.13mm, and a 9x9 reduces average error by 0.15mm.

The cost of the expanded search is increased computational requirements. Compared to a 3x3 search, a 5x5 search requires 2.7 times as many calculations. A 7x7 search requires 5.4 times as many calculations as a 3x3, and a 9x9 search requires 9 times as many calculations. A broader final search stage, while addressing the multiple peak issue, has a great effect on system throughput. Replacing the final stage with a broader search reduces throughput by 2.7x for a 5x5, 5.4x for a 7x7, and 9x for a 9x9. Since 9x9 search requires a significantly greater number of calculations, with a only slight benefit over the 7x7, it is not considered.

The majority of crystal positions receive little to no benefit from using an expanded search. Examining system accuracy and occurrences of multiple peaks shows that only positions within 5mm of the crystal edge benefit by a broader search. Approximately 37% of the crystal area falls within 5mm of an edge. Given an equal distribution of events across the crystal this corresponds to 37% requiring a broader search. If only events within 5mm of the edge of the crystal invoke a 7x7, throughput is reduced to .38X compared to a 3x3 final stage. Further improvements can be made using a 5x5 search for data located near only one edge of the crystal and a 7x7 search for data located in the corner of the crystal [Figure 24]. This modification results in an increase of throughput of 1.5X over only using a 7x7 search, reducing throughput to .57X when compared to the 3x3 search.

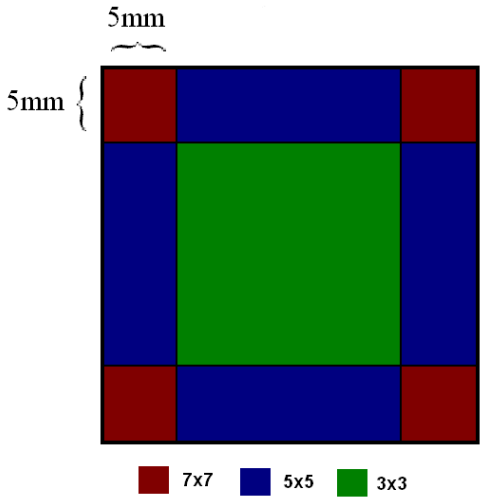

Figure 25. Search breadth regions

A 7x7 search is computational intensive, requiring more than five times the calculations of a 3x3 search. While only a portion of data points are expected to invoke a 7x7 search, the computational intensity of those select data points have a profound impact on the overall throughput of the system. While multiple final stage data tables allowing parallel calculations would be ideal, the size of a full table makes it impractical. Alternatively, in the event of a 7x7 search the pipeline is stalled while the 7x7 search performs the extra calculations; in this situation the characterization tables of the previous stage are unused for much of the time. In an effort to increase the throughput of a 7x7 search the fifth stage table can be used to perform some of the calculations. 9 of the points in a 7x7 search are available in the fifth stage table; performing those calculations in parallel increases the throughput of a 7x7 search by 22%. Overall this change results in an expected 1.05X speed up averaged across all data points compared to using the sixth stage table by itself.

The decision of search breadth is based on the position found in the preceding stage. When the preceding stage indicates that the location of the event is with 5mm of the edge of the crystal the algorithm is adjusted accordingly. If the previous stage indicates that the position is near the edge the algorithm enters the 5x5 search state, and if the previous stage indicates a corner position the algorithm enters the 7x7 search state. Note that the state of the final stage has no effect on the operations of the preceding stages.

While broader searches help reduce the total number of instances of multiple peaking, it cannot completely eliminate them. For some data points, multiple peaks exist in the exhaustive solution set. In these instances the position with the greatest likelihood produces the incorrect result. For these cases, even a perfectly accurate hierarchical method will result in multiple peaks.

9 Hardware

9.1 Memory Management

When implementing the algorithm in an FPGA, the limiting factor has proven to be memory resources. The 2D implementation used 90% of available on-chip memory while using only 5% and 7% of logic and routing resources respectively. Previous implementations provide an example of the importance of efficient use of memory resources. Early versions of the two dimensional algorithm used a reduced word size as it was believed that there was not enough memory available to completely implement the 2D algorithm on the available development board. After reorganization and manual allocation it was found that the 2D algorithm could be completely implemented on a development board.

Approximately a third of the available memory on the Stratix II FPGA is composed of 4Kb or 512b blocks. These blocks can be used independently or grouped together to make larger memory blocks. The 4Kb and 512b blocks were primarily grouped together to form the characterization tables for the first three stages. The proceeding stage table uses the large MRAM blocks to store characterization values. Each MRAM block has a capacity of 512Kb. The Mu, Sg, and Ln values each require 2 MRAM blocks to reach the necessary capacity. MRAM blocks are utilized in the same way as the smaller 4Kb and 512b blocks.

With most of the on chip memory resources used for the preceding stages, there is not sufficient room to store the characterization tables for the fifth stage on chip. In place of on chip memory, off chip memory is used. The development board implementation used two SRAM memory chips. A memory controller was designed so that off chip memory could be used with few changes to the core algorithm. However, each chip only had a 16

bit word size (two chips in parallel made a 32 bit word), smaller than the necessary 48 bit word. The 2D implementation used spare memory resources to create an additional block of memory equal in size to one of the SRAM chips; this block was a combination of MRAM, 4Kb, and 512b blocks. In the 3D implementation this was not an option as much more on-chip memory resources were used as a result of the 3D tables needed for the depth search. In order for the 5th stage to function properly a reduced word size is used. The $\frac{1}{\sqrt{2}\times\sigma}$ and $\ln(\sigma)$ values are reduced from 16b to 8b, the mu value remains 16b.

The final stage also uses off chip memory to store its corresponding characterization tables. The development board implementation used two 32 bit word SDRAM chips available on the development board. A different memory controller was designed for the final stage due to additional requirements in operating an SDRAM. The SDRAM banks had sufficient bandwidth and word sizes that all necessary tables were able to fit on the available chips.

One problem faced while implementing the algorithm in an FPGA is loading the characterization tables onto the hardware itself. While some values can be stored in the 4Kb and 512b blocks as part of the FPGA configuration bitstream, this is not an option for the larger MRAMs or off chip memory. Additionally, memory on the Stratix family FPGA is volatile, once the FPGA is turned off values stored in the memory are lost. As a solution, a full set of tables was stored on the non-volatile flash memory available on the Stratix II development board using a separate FPGA configuration. An additional startup state was added to the algorithm. In the startup state the hardware cycles through every value stored in the flash memory, storing them in the appropriate stage's table. The startup characterization value reallocation takes approximately 5 seconds. Upon completion of the startup state the hardware enters into normal operation mode. Due to the low throughput of the flash memory it is not used beyond the startup state.

The DRAM memory on the development board is organized in rows, columns, and pages. This significantly complicates using DRAM. Opening or changing a row of memory consumes 3 clock cycles. Once a row has been opened the memory has random access across all columns in that row, with a new value available each clock cycle. In order to

maximize memory bandwidth, memory organization becomes important. For a given set of operations the number of row changes needs to be minimized to maximize throughput in the stage. In order to achieve this, the memory was organized so that points that are frequently grouped together are placed in the same row. Since the DRAM is used for the final stage this is fairly easy to achieve as points that are frequently accessed together are physically closely located and therefore share much of the same address.

The DRAM needs to be refreshed to maintain its stored values at least every 15μs, or once every 1050 clock cycles at 70MHz. This requirement was handled by grouping the refresh command with the open page command that occurs when processing the first point of a new event. Additionally the refresh command is issued when there is no event and the stage is idle. Overall the refresh command is issued at least every 450 clock cycles, the amount of time it takes to perform a broad corner search.

9.2 Control

The 2D implementation consisted of 6 stages, with each stage performing nine calculations. Thus each stage performed the same operation as the others at the same time. The hardware was designed to perform the same operation across all stages during each clock cycle. This was done by the creation of a central control module that issued instructions to each stage, with each stage receiving the same instructions. Since the algorithm always performed the same operations in the same order this was very efficient. However, this approach proved to be difficult if one stage needed to be modified and still maintain synchronization with the other stages.

The 3D implementation took a different approach. Each stage performs its necessary calculations independent of the other stages. Upon completion of the stages requisite calculations the stage sends a “completion” flag to a central controller. The central controller waits until all stages have sent their completion flag before issuing the “go ahead” command. The go ahead command synchronizes all the stages to allow efficient data transfer between them. Upon completion of the data transfer each stage then continues to process its data independently. Since some clock cycles are lost in handshaking between the central controller and the stages, this implementation is not as

efficient. However, it is much more flexible. One stage can be modified to perform a different operation without requiring changes to the other stages or to the central controller. This is particularly useful since the number of final stage calculations is determined dynamically.

The sensor values are passed from one stage to the other when processing the last data point of the search. The passed value is stored in a register for a clock cycle in the next stage before overwriting the value just used in calculating the last data point of the search. With this implementation no additional time is needed to transfer data from one stage to another as the transfer occurs during operation. Another advantage is reduced memory requirement, as only enough memory is required to hold a single events worth of sensor data in each stage.

10 Results

The presented 3D SBP algorithm uses a variation on the approach used for the 2D SBP hierarchical search. The new algorithm uses an 8 stage pipeline, with each stage providing a more fine grained solution. The first four stages in the pipeline are planar searches used to find a coarse grained XY solution. Using the course grained XY position, the depth of interaction is found in the proceeding stage. The solved depth of interaction is then used to refine the characteristic values in the final three planar XY stages.

2D versions of the hierarchical search and early 3D versions would often produce multiple peaks in distributions of points near the edges and corners of the crystal. In order to improve the accuracy of the 3D hierarchical search along the edges and corners of the detector module, the algorithm dynamically selects a search breadth based on the position of the event. A broader search compares more points across a greater range of the possible solutions, this allows the hierarchical search to escape the local minima that cause multiple peaks and select the absolute minimum. Along the edges of the crystal, additional effort is used and a broader range of data points are compared. For events near the corner an even more extensive search is used. These modifications, while improving the performance of the algorithm, come at the cost of additional computational effort. By dynamically selecting the search breadth at run time, the algorithm matches the

appropriate computational effort with the position of the event. The addition of the dynamic breadth search reduces the incidences of additional peaks by 63% over the standard 3x3 search.

The designed algorithm was implemented on a FPGA development board. Due to the restrictions of limited resources several concessions had to be made during the design process at the cost of either throughput or accuracy. While the algorithm has been designed to, as closely as possible, match the performance of an exhaustive search, custom hardware specifically designed for the task offers room for improvement.

Implementing the design in an FPGA allows significant improvement in throughput. Pipelining the stages allows a throughput between 8 and 48 likelihood calculations, depending on the position of the event. Pipelining within each stage and algorithm refinement allows the 8 to 48 likelihood calculations to be performed in 180 to 885 clock cycles. At a 70 MHz clock rate, the throughput is in between 388,888 and 79,000 events per second. Averaging for an expected equal distribution of events across the solution space gives an average throughput in excess of 220,000 events per second. With an expected rate of 200,000 events per second, this hardware implementation is capable of processing the data in real time.

The algorithm design was limited by both the memory capacity and bandwidth of the FPGA development board. A custom board is currently under design for this specific application with additional memory capacity and bandwidth. While the development board implementation provides acceptable levels of accuracy, the additional memory provides room for improvement. The new board under development features three very large banks of off chip memory.

Memory limitations of the development board required the depth search to be placed after the third XY stage, a location that provides good although not ideal results. The addition of the third bank of off-chip memory would allow the depth search to be placed in the optimal location between XY stage 4 and XY stage 5. This results in a significant improvement in depth accuracy as seen in Table 5. Additionally, the larger off chip memory banks allow more detailed characterization tables to be stored for stage 5 and

stage 6. With the larger banks the full 15 depth tables can be stored in both stage5 and stage 6, the development board only allowed a 2 depth stage 5 and a 4 depth stage 6.

Table 11 summarizes the accuracy of the three dimensional hierarchical statistical based algorithm. An exhaustive search was used as a benchmark for the algorithm development. Over all, the 3D hierarchical SBP algorithm provides very near accuracy to the more time consuming exhaustive method. The most noticeable difference between the accuracy of the exhaustive search and the accuracy of the algorithm as implemented on the development board is seen in finding the DOI. Additional accuracy can be anticipated with the use of the new custom board which will allow a greater freedom in algorithm design.

Method	**FWHM-XY (mm)**	**FHWM-Z (mm)**	**Average Error - XY (mm)**	**Average Error - Z (mm)**
Exhaustive SBP	0.9	1.88	1.67	1.74
3D Hierarchical SBP Development Board	0.91	2.02	1.72	1.82
3D Hierarchical SBP Custom Board	0.87	1.9	1.69	1.78

Table 11: Summary of results of 3D hierarchical SBP

References

[1] X. Li, et al., “A high resolution, monolithic crystal PET/MRI detector with DOI positioning capability *IEEE EMBS*,” pp. 2287-2290, 2008

[2] J Joung, et al., “CMice: a high resolution animal PET using continuous LSO with a statistics based positioning scheme,” *NIM. Phys. Res. A*, vol. 489, no. 1-3, pp. 584-589, Aug. 2002.

[3] T. Ling, et al., “Performance comparisons of continuous miniature crystal elements (cMiCE) detectors,” *IEEE TNS,* vol. 53, pp. 2513-2518, 2006.

[4] G. Tsang, et al., “A simulation to model position encoding multicrystal PET detectors,” *IEEE TNS*, vol. 42, pp. 2236-2243, 1995.

[5] G. F. Knoll, et al., “Light collection in scintillation detector composites for neutron detection,” *IEEE TNS*, vol. 35, pp. 872-875, 1988.

[6] S. Agostinelli, et al., "Geant4 – a simulation toolkit," *NIM Section A*, vol. 506, pp. 250-303, 2003.

[7] B. D. Rooney, et al., “Scintillator light yield nonproportionality: calculating photon response using measure electron response,” *IEEE TNS*, vol. 44, pp. 509-516, 1997.

[8] D. DeWitt, Master’s Thesis, Department of Electrical Engineering, University of Washington, 2008.

[9] Altera Corporation, “Stratix II Device Handbook.” Altera Corporation. 11/06/2009 <http://www.altera.com/literature/hb/stx2/stratix2_handbook.pdf>.

[10] Altera Corporation, “Stratix II EP2S180 DSP Development Board Reference Manual.” Altera Coporation. 11/06/2009 <http://www.altera.com/.../manual/mnl_stx2_pro_dsp_dev_kit_ep2s180.pdf>.

[11] Micron Technology, Inc., “Synchronous DRAM” Micron Technology, Inc. 11/06/2009 <http://download.micron.com/pdf/datasheets/.../128MbSDRAMx32_E.pdf>.

[12] Integrated Silicon Solution, Inc., “IS61LV25616AL” Integrated Silicon Solution, Inc. 11/06/2009 <http://www.issi.com/pdf/61LV25616AL.pdf>.

[13] Spansion. “S29GL128/256N MirrorBit Flash with Alternative BGA Pinout.” Spansion. 11/06/2009 <http://www.spansion.com/Support/Datasheets/s29gl128_256_sp_a2_e.pdf>.

[14] D. DeWitt, R. S. Miyaoka, X. Li, C. Lockhart, T. K. Lewellen, S. Hauck, "Design of an FPGA based Algorithm for Real-Time Solutions of Statistics-Based Positioning", to appear in *IEEE Transactions on Nuclear Science*.

[15] M. Haselman, D. DeWitt, T. K. Lewellen, R. Miyaoka, S. Hauck, "FPGA-Based Front-End Electronics for Positron Emission Tomography", *ACM/SIGDA Symposium on Field-Programmable Gate Arrays*, pp. 93-102, 2009.