Design of an FPGA based Algorithm for Real-Time Solutions of Statistics-Based Positioning

Don DeWitt, Robert S. Miyaoka, *Member IEEE*, Xiaoli Li, *Student Member IEEE*, Cate Lockhart, Tom K. Lewellen, *Fellow IEEE,* Scott Hauck*, Senior Member IEEE*

***Abstract*—We report on the implementation of an algorithm and hardware platform to allow real-time processing of the previously described Statistics-Based Positioning (SBP) method for continuous miniature crystal element (cMiCE) detectors. The SBP method allows an intrinsic spatial resolution of ~1.4 mm FWHM to be achieved using our cMiCE design. Previous SBP solutions have required a post-processing procedure due to the computation and memory intensive nature of SBP. This new implementation takes advantage of a combination of algebraic simplifications, conversion to fixed-point math, and a hierarchal search technique to greatly accelerate the algorithm. For the presented seven stage, 127x127 bin LUT implementation, these algorithm improvements result in a reduction from >$7x10^6$ floating-point operations per event for an exhaustive search to <$5x10^3$ integer operations per event. Simulations show nearly identical FWHM positioning resolution for this accelerated SBP solution, and positioning differences of <0.1mm from the exhaustive search solution. A pipelined Field Programmable Gate Array (FPGA) implementation of this optimized algorithm is able to process events in excess of 250K 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. These methods allow SBP results to be calculated in real-time and to be presented to the image generation components in real-time. A prototype hardware implementation has been tested, limited to 4 stages due to memory limitations of the initial prototyping board. A custom board is currently under development to allow implementation of the full seven stage algorithm.**

***Index Terms*—Continuous crystal, PET detector, FPGA.**

I. Introduction

Discrete crystal detector modules have traditionally been used to achieve high spatial resolution for small animal (pre-clinical) PET scanners. However, the cost goes up rapidly as crystal cross-section gets smaller. We have previously presented a continuous miniature crystal element (cMiCE) detector as a lower cost alternative, with a statistics based positioning (SBP) algorithm used for event positioning from list mode data [1-3]. We are currently developing a full pre-clinical scanner based on the cMiCE detector module. However, the processing and memory requirements of performing the event positioning from the list mode data results in an undesirable delay between the end of data collection and the positioning of the event data. Processor arrays are able to reduce this time; however, a goal of this work is to provide a real-time implementation. In this work, several methods of algorithm optimizations, along with a pipelined FPGA implementation, results in a system capable of generating SBP results in real-time (i.e., SBP event positioning of over 250K events per second per detector).

An FPGA is a special purpose integrated circuit (IC) designed to perform complex interface and logic processing in a small footprint [4]. The FPGA is a reconfigurable hardware platform consisting of a large number of simple logic elements with a matrix of interconnects that can be selectively enabled or disabled under the control of a downloadable configuration file. This allows the FPGA a flexibility that is not available with microprocessors.

The flexibility of the FPGA will allow many options to be considered in implementing the SBP method. The proposed system requires a method of generating solutions at a much higher rate than available with the post-processing method currently used. Some of the hardware options to be explored include parallel processing and hardware acceleration of the individual computations. Software options include optimizing

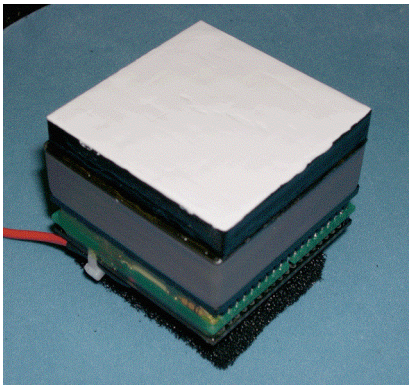

Figure 1. Picture of cMiCE detector with 8 mm thick LYSO crystal and white paint.

Manuscript received November 7, 2008. This work was supported in part by NIH-NIBIB grant: R21/R33 EB001563 and Altera Corporation.

Don DeWitt, PE is now doing independent consulting work.

Robert S. Miyaoka is with the University of Washington Department of Radiology, Seattle, WA 98195 USA (telephone: 206-543-2084, e-mail: rmiyaoka@u.washington.edu).

Xiaoli Li is with the University of Washington Department of Physics, Seattle, WA 98195 USA.

Cate Lockhart is with the University of Washington Department of Radiology, Seattle, WA 98195 USA.

Tom K. Lewellen is with the University of Washington Department of Radiology, Seattle, WA 98195 USA.

Scott Hauck is with the University of Washington Department of Electrical Engineering, Seattle, WA 91895 USA.

the algorithm performing the computations and conversion from floating-point to an integer or fixed-point implementation to reduce the computational complexity. These methods will be explored in the following sections.

II. Materials and Methods

A. cMiCE Detector Module

A cMiCE detector module is pictured in Fig. 1. It is composed of a 50 mm by 50 mm by 8 mm thick slab of LYSO (Saint Gobain, Newbury, Ohio) coupled to a 64-channel multi-anode PMT (H8500, Hamamatsu Photonics K.K., Japan). One of the 50 mm by 50 mm surfaces was polished. All other surfaces were roughened. The polished side was coupled to the PMT using Bicron BC-630 optical grease. The face of the crystal opposite the PMT was painted white. The side surfaces of the crystal were painted black to reduce light reflections off the sides.

B. Statistics-based Positioning Method (SBP)

Suppose the distributions of observing PMT outputs M = M_1, M_2, …, M_n for scintillation position x, are independent normal distributions with mean $\{\mu(x)\}$ and standard deviation $\{\sigma(x)\}$.

The likelihood function for making any single observation m_i from distribution M_i given x is:

$$L[m_i \mid x] = \prod_{i=1}^{n} \frac{1}{\sigma_i(x)\sqrt{2\pi}} \exp\left(-\frac{(m_i - \mu_i(x))^2}{2\sigma_i^2(x)} \right) \qquad (1).$$

The maximum likelihood estimator of the event position x is given by:

$$\hat{x} = \arg\min_x \left[\sum_i \frac{\left(m_i - \mu_i(x)\right)^2}{2\sigma_i^2(x)} + \ln\left(\sigma_i(x)\right) \right] \qquad (2).$$

The SBP method requires that the light response function versus interaction location be characterized for the detector. Two SBP look-up tables (LUTs) corresponding to the mean and variance of the light probability density function (PDF) versus (x,y) position are created during the characterization process. The LUTs are 127x127, corresponding to ~0.4 mm by ~0.4 mm pixels for binning. For more detailed explanations about the SBP implementation refer to Joung [1].

C. Equation Simplification

The proposed method to implement SBP within an FPGA relies upon multiple levels of optimization. The first to be employed is reducing the number of computations required for each iteration of the likelihood equation, the central component of the SBP method. The log likelihood function for each signal channel can be represented as in Equation 3.

$$\frac{(m_i - \mu_i(x))^2}{2\sigma_i^2(x)} + \ln(\sigma_i(x)) \qquad (3)$$

The objective is to manipulate this function to minimize those numerical operations which are expensive to perform in terms of time or resources. With the computational resources available in this project the following guidelines are employed:

- Adds and subtracts are efficient.
- Multiply is relatively efficient, but should be kept to a minimum.
- Squares are special cases of multiply, but should also be minimized.
- Division is very inefficient because it requires an iterative approach.
- Transcendental functions should be avoided at all costs.

Each iteration of Equation 3 consists of seven operations:

1. Subtract (E - μ).
2. Square the result.
3. Square σ.
4. Multiply the result by 2.
5. Divide the result of step 2 by the result of step 4.
6. Find the natural log of σ.
7. Add the result of step 7 to the result of step 5.

The transcendental function of step 6 increases the complexity of this method well beyond the seven operations, and needs to be eliminated. Also, the division should be replaced by a multiplication by a pre-computed inverse if possible. Following the guidelines given above leads to the simplification shown in Equation 4.

If we let $A=\mu_i(x)$, $B=1/(1.414\sigma_i(x))$ and $C=\ln(\sigma_i(x))$, Equation 3 can be rewritten as Equation 4.

$$((x - A) * B)^2 + C \qquad (4)$$

In this simplification, the three variables A, B and C replace the original μ and σ in the data tables. Variable A is simply equal to μ. Variables B and C can be pre-calculated by the host before being stored in the data tables used for this calculation, saving the time and resources of both the arithmetic operations and the transcendental function. With these simplifications, only four operations are required:

1. Subtract (E - A).
2. Multiply the results by B.
3. Square the results.
4. Add C.

This greatly reduces the computational complexity of each iteration of the likelihood equation and will be substituted for the original equation in all of the following uses.

D. Twenty-One channel solution

The original SBP solution summed the log likelihood equation for each of the 64 channels of the 8x8 sensor. However, it was found that using channels far from the peak reduced positioning accuracy. Using the 21 channels surrounding the peak energy, as shown in Figure 2, led to

better positioning performance. In addition to improving positioning, this results in a ~3:1 reduction in the number of operations needed for each solution.

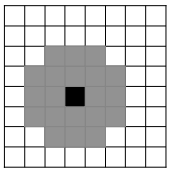

Figure 2. Twenty-one channel masking, centered on the peak energy channel.

E. Conversion to Fixed-Point

Floating-point calculations are computationally expensive in an FPGA implementation. Additionally, floating-point values occupy 4 or more bytes of memory storage per value, creating an excessive storage requirement. Converting to integer representations, while retaining the required resolution, necessitates multiplying by a power of two before the conversion to move fractional parts of the value into the resulting integer. Proper management of these multiplication steps allows the resulting calculations to correctly represent the relative values of the original results. The advantages of using integer values are that a smaller number of bits are required for storage, and integer math operations can be used for all calculations and comparisons.

F. Hierarchal Search

An exhaustive search method has the advantage of being easy to implement and is guaranteed to always find the absolute minimum solution. It has the disadvantage of being very inefficient, both in terms of the number of iterations, and the number of memory accesses to find a solution. Observation of the solution set, figure 3, made it apparent that a structured search could be used to converge to a solution with many fewer iterations.

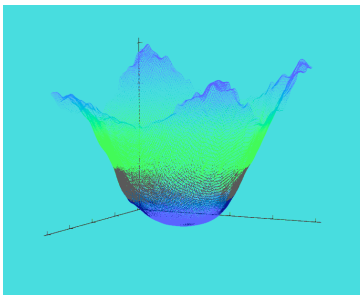

Figure 3. The complete solution set for a representative event.

Of the structured searches explored [5], it was found that a hierarchal search provided the best combination of reduced iterations and memory usage, allowing memory to be partitioned in a way that was conducive to a pipelined implementation. This search functions by finding the solutions at a number of evenly distributed samples points. The minimum result from these samples identifies the region holding the ultimate solution and a new, smaller solution space is sampled, centered on the previous best solution. This process continues iteratively until the solution space is a single pixel. An advantage of this method is that the early stages of the search only require access to a small number of the table values. This allows separate tables for each stage of the search, without the large storage requirements of multiple copies of the complete tables.

This algorithm yields a large reduction in the number of points that need to be sampled. Variations in the number of sampled points at each iteration were simulated, ranging from a 2x2 array of samples at each iteration of the search, resulting in an 8 iteration search in a 127x127 solution space (best case) to 5x5 samples resulting in a three iteration search. These require a total of 2*2*8=32 sampled points for the 2x2 search to 5*5*3=75 points for a 5x5 search.

The optimal solution was found to be a hierarchal search with 3x3 samples at each iteration, dividing the solution space to approximately one-quarter of its previous size. An example of this search on a 63x63 solution space is shown in Figure 4. This example shows the five iterations necessary to fully search the 63x63 space. The solution space of each successive stage is shown by the increasingly smaller box. Within each solution space, the nine X-Y points to be tested are shown, centered on the best solution from the preceding stage. The search concludes with a three by three search at a spacing of one pixel.

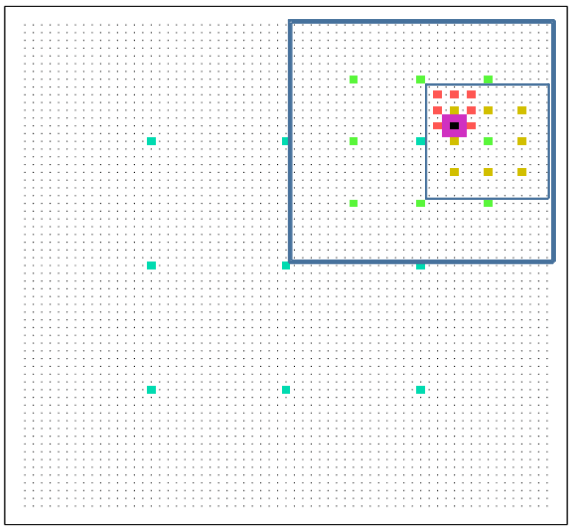

Figure 4. Five iterations of a 3x3 search on a 63x63 solution space.

The regularity of this search allows each level of the search to consist of an almost identical process, where each iteration starts with a center point for the search and the spacing between points to be tested at this iteration. For the 3x3 search shown, 9 sets of characterization data table values are needed for the first stage, one for each position to be tested. The next iteration will consist of potential sample positions at one-half the spacing, with sets of points centered on each of the previous stage's points, with the first stage defaulting to being centered on the center of the entire data table (see figure 4). This results in each stages' data table having approximately

four times the number of testable locations as the previous stage, but still only a fraction of the overall table, until the last stage is reached. The last stage is the only stage that needs access to the entire table. Each iteration will begin with the best solution from the previous stage as the center point of its search and will test all points adjacent to it at the current resolution. As each stage passes its best result to the next stage, eventually the last stage is reached where the spacing between adjacent pixels is one. The results of this stage represent the best solution available for the data table used.

Structuring the search in this way allows points sampled during each iteration to be independent of calculations currently underway in other iterations. This can allow calculations for different events to be pipelined, allowing multiple stages to be processed simultaneously.

A second advantage of this method is that the storage requirement for each stage's data table is reduced for each iteration prior to the last. Ideally, for ***n*** samples, the new solution space would contain ***n*** times as many possible solutions as the previous iteration. For our example with nine samples, we would hope for nine times as many potential sample points at the next iteration. However, experiments determined that when the final solution was located approximately midway between two sampled points, this solution was not always nearest to the best sampled point. The cause of this was found to be that the solution set is not completely symmetric and is subject to some solution noise. To solve this problem, each successive iteration must operate on a subset that is significantly larger than ***n*** times the size of the previous iteration.

Our solution is to have the solution space dimension at each stage to be an integer power of 2, minus 1 (2^n-1). Constraining each level to have an odd number of values allows each solution space to have a pixel in the exact center, with other X-Y locations spaced at a power of two. A representation of a simple four stage system is shown in Figure 5. The top row of figures shows the spacing of the locations that can be tested at each stage. The bottom row of figures shows how these values are stored using a compressed addressing scheme where the unused values are not included in the table for that stage. This allows the data table at all stages except for the final stage to be stored in a much smaller memory space. The storage of the odd number of values is padded with blank values to allow separate X and Y addresses to access the desired coordinate without requiring an address calculation. The proper value is addressed by simply appending the binary representation of the X location to the Y location. For example, if the desired X coordinate is 11_b and the Y coordinate is 01_b, the address of the desired value is at 1101_b. As the solution moves from one stage to the next, address translations to match the next stage are accomplished by multiplying the X and Y coordinates by two. This is accomplished by appending a zero to the binary address values. For the example above, 11_b->110_b and 01_b->010_b, so the resulting address of the same coordinate in the next stages addressing scheme would be 110010_b. This allows the binary system to do in-stage addressing and next-stage address translations in zero time.

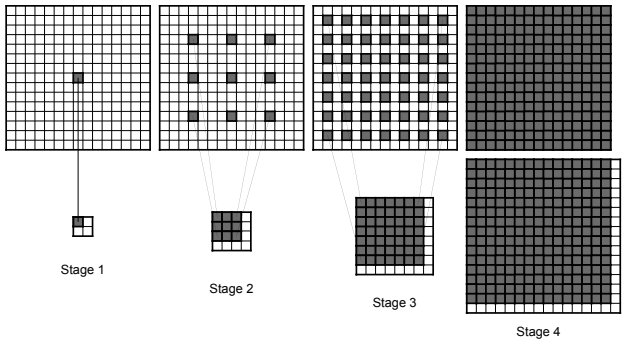

Figure 5. Compressed storage for data tables.

As the solution progresses, the only information passed from one stage to the next is the center point on which to base that stage's search. Each stage has access to data table information at twice the resolution as the previous stage, but will only be accessing values directly adjacent to the center value, as shown in Figure 6. This is done by sequentially adding then subtracting one from the values of the center X and Y positions passed from the previous stage.

In this example, it can be seen that the spacing between tested locations reduces from four in the 15x15 space, to two in the 7x7 space, to one pixel as the last 3x3 stage is reached.

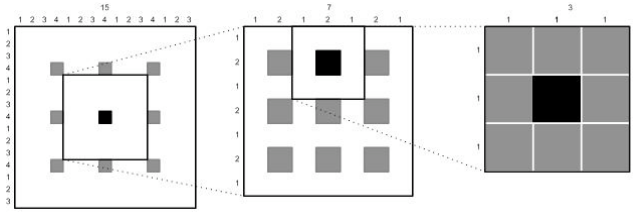

Figure 6. Diagram of the last three iterations of a hierarchal search.

The one pixel spacing of the last stage results in an exhaustive search of this reduced space.

G. Pipelined Architecture

To attain the desired processing rates for this project, multiple events must be simultaneously processed. Although parallel processing paths could easily be formed in the FPGA, multiple copies of the complete data tables would be required to support this architecture. Using the search described above, a pipelined architecture is possible. In this design, each stage of the hierarchal search is implemented with dedicated hardware that has exclusive access to a data table appropriately sized for the stage. As each event's processing at one stage is completed, the intermediate results are passed as a starting point to the successive stage, allowing the next event to immediately begin processing. This results in multiple events simultaneously being at various stages of their processing without requiring multiple complete copies of the data tables.

H. Memory Division

The characterization tables for each sensor must be available for solving the likelihood function. The most direct approach would be to store the tables in the FPGA's on-chip

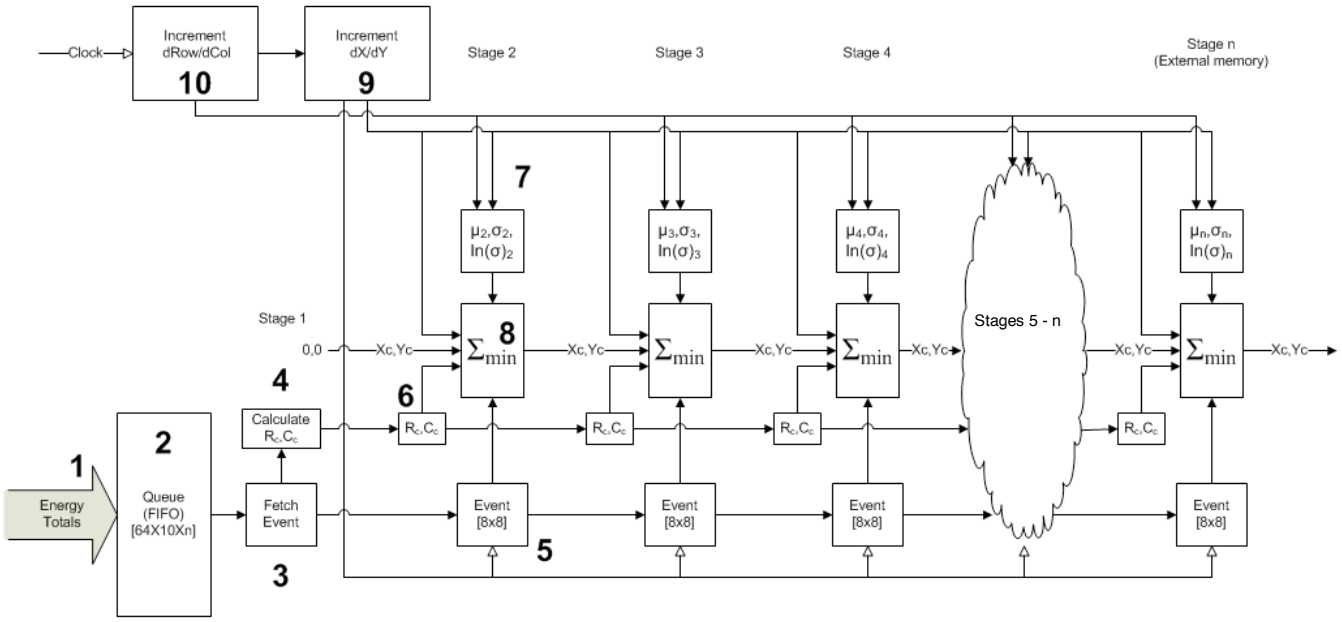

Figure 7. Block diagram of FPGA implementation of SBP. The numbered components are defined in section J.

SRAM memory. Processing the events would be simplified compared to accessing off-chip synchronous dynamic (SD) or double data rate (DDR) memory. The difficulty with this method is the large size of the tables. Each of the three tables requires 16-bit values for each of the 8x8 channels of sensed data, for 127 possible positions in each of the X and Y axes. This equates to [3 Tables] x [2 byte values] x [8x8 channels] x [127x127 possible X-Y locations] or >6 Mbytes. This is well above the approximately 300 kB to 1.1 MB available in members of the family of FPGA devices chosen for this project (i.e., Altera Stratix II or III devices, Altera, San Jose, CA). Further, future versions of this system are planned that will require multiple sets of tables to support three dimensional positioning of a detected event. If it were possible to create the current solution using only internal memory, this project's work would not be reusable for future designs that will certainly require access to external memory.

The design employs a mix of FPGA internal memory and high-speed external memory. This allows all the small data tables required by the early stages of the process to have individual data and address busses, allowing parallel access by the different stages. The largest data tables are kept in external memories where adequate storage is available. Because most of the processing is done in the multiple earlier stages, memory bandwidth requirements are significantly reduced to the external memories.

I. Input Buffering

Although the average event rate is expected to be 250K events per second or less, because of the random nature of positron decay, instantaneous peak rates as high as 2 times this value are expected. An input FIFO buffer allows data collection at up to the peak rate, and delivery to the SBP algorithm at no more than the designed rate.

J. Prototype Implementation

A block-diagram of the system design is shown in Figure 7. The numbered components of this system are:

1. An interface to the incoming data. In the prototype system this is previously collected data that is downloaded, stored in local memory and sequenced into the system to simulate actual data collection.
2. A FIFO buffer. This allows the randomly occurring samples to be collected at up to the peak rate, but processed at their average rate. This will be necessary in the target system because the events occur randomly and may momentarily occur at faster rates than the SBP algorithm can process them. This buffer allows them to be stored and dispensed to the algorithm at a lower average rate.
3. The Fetch Event block gets the next available event from the FIFO and structures it for use in the SBP calculations.
4. This block determines the row and column of the sensor with the peak energy. This is made available to the SBP algorithm to allow the 21-cell version of the SBP algorithm to calculate the proper channels to process for the corresponding event.
5. The values of an event are stored locally at each stage so that they can be accessed for the calculations required at that stage. As processing of a particular stage completes, the values are passed along to the next stage. Double-buffering is used to allow each stage to transfer the values to the next stage before either stage has completed its current calculations. This allows the transfer of data to overlap the calculations.
6. Row center (Rc), Column center (Cc). The values of the row and column centers for the 21-cell solution are also stored locally at each stage.
7. The characterization tables for each stage holds the data required for the SBP algorithm.

8. The SBP calculations to determine the likelihood values are performed here. As each X-Y position is tested, if the sum is less than previously tested positions the X-Y value is saved. When the last X-Y position is tested, the X-Y position of the position with the lowest sum is made available to the next stage as the center X-Y position (X_c-Y_c) of the positions to be tested.
9. Increment dX, and dY. This sequences each step of the algorithm through the proper X and Y locations for each X and Y position to be tested. The dX (delta X) is added to the X_c (X center) to determine proper position on which to perform the likelihood calculation. The Y values are computed in the same way. Additional timing signals are generated here to signify occurrences of the first X-Y and last X-Y tested to coordinate resetting values at the beginning of calculations and transferring values at the end of calculations.
10. Increment dRow and dCol. This sequences the likelihood calculation through all the rows and columns of the 21-cell solution. Similar to the X-Y locations, the delta values for the row and column are added to the center location of the 21 cells to be included in the calculation. Additional timing signals are generated here to signify occurrences of the first row/column and last row/column to coordinate resetting values at the beginning of each iteration of the likelihood function.

When the computations at each stage are complete the event data, the row and column centers, and the newly computed X and Y centers are passed to the next stage. Each stage operates identically to the previous stage except that the data tables are larger at each successive stage. The number of potential X-Y positions that will be tested for each event at each stage are the same, but they can have a larger number of potential X_c and Y_c positions on which to center this stage of the search. This is accommodated by appending a single bit to the least significant end of the X_c and Y_c values as they are transferred to the next stage. This also explains the apparent lack of a first stage. The first stage data table would consist of only one location, so the results of this computation are known without requiring any computations. The address of this value (0,0) are passed to stage two and used as the starting point for that stage. The result at each stage is the X-Y location of the best fit to the characterization data to the precision achieved at that stage. The result of the final stage is the value that represents the highest precision available in this implementation.

IV. Results

To test the accuracy of our design implementation, simulations were run where the results of a large number of sampled locations were compared to the results of the exhaustive search on an event-by-event basis. To summarize the cumulative effect of these comparisons, a histogram showing the distance error between the positions calculated using the exhaustive search and the same positions calculated using the hierarchal search was prepared. The worst case results were for a 2x2 search. These results are shown in Figure 8.

The results show that over 90% of the events show no error and over 99% are within 1.4 pixels (actually 0.3 mm), representing all the pixels immediately adjacent to the reference pixel. Note that in this comparison, diagonal pixels are represented by the Euclidean distance. A large portion of the <1% remaining errors are the results of the solutions of background noise, seen as the scattering of points across the surface. Many of these events have solution sets that do not allow an accurate solution regardless of the method used. These errors do not noticeably impact the quality of the resulting image.

A four stage system has been tested in hardware and a seven stage system is currently being developed. An integer based simulation extending the results of the four stage hardware was developed to compare solutions with the reference floating-point SBP solutions. Positioning results using floating point and fixed-point calculations, illustrated in figure 9, show no visible difference in performance. The results for fifteen tested experimental data sets are shown in Table 1. Data collected near the edge of the detectors (i.e., data set 1 and 2) showed the largest differences from the reference floating-point calculations. Points further from the edge showed little to no error.

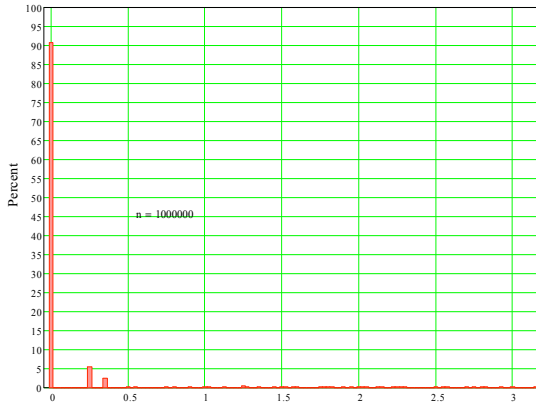

Figure 8. Histogram of distance errors (mm) between exhaustive and hierarchal searches.

Table 1. Comparison of reference (original SBP implementation) and optimized calculation (FPGA implementation) FWHM values.

		Reference Calculation	Optimized Calculation	
data-set	n	FWHM (mm)	FWHM (mm)	delta (mm
1	16793	2.16	1.93	-0.23
2	18796	1.90	2.33	0.43
3	19540	1.41	1.43	0.02
4	19560	1.28	1.31	0.03
5	18921	1.19	1.10	-0.09
6	13699	1.46	1.48	0.02
7	14136	1.25	1.43	0.18
8	13100	1.35	1.33	-0.02
9	13223	1.19	1.23	0.04
10	14209	1.37	1.30	-0.07
11	12584	1.32	1.35	0.03
12	14043	1.46	1.43	-0.03
13	13545	1.54	1.54	0.00
14	12920	1.56	1.59	0.03
15	13690	1.25	1.27	0.02
Average		1.45	1.47	0.02

The throughput of the system is defined as the rate at which new solutions are completed. In this pipelined design, it is

equal to the time for one stage to complete. For the current design, each stage requires nine summations of the 21-channel solution, each being able to complete in one clock cycle. Additionally, three clock cycles are required for determining the minimum of the nine summations and transferring this value to the next stage. This results in a new event starting, and one completing, every 192 clock cycles. At the designed clock rate of 70 MHz, this results in a throughput of >360k events per second.

The latency is defined as the time from the acceptance of the data to the presentation of the processed SBP position to the host. Disregarding the time required to pass the detector data through the FIFO and the time to pass the results to the host, this is equivalent to the combined time of all the stages. For a complete seven stage system, this would be seven times 192 clock cycles at 70 MHz, or approximately 20 µS.

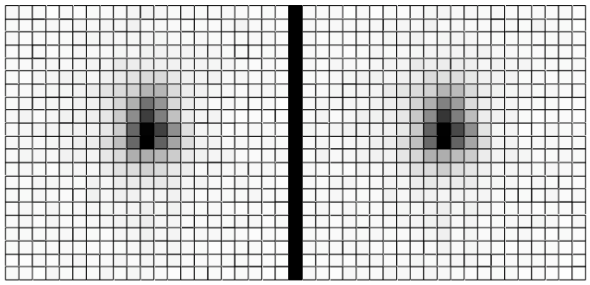

Figure 9. Resulting point source image for floating point (left) and fixed-point (right) calculations.

V. Conclusions

This study showed that optimizations to the SBP algorithm to allow real-time implementation on an FPGA can be made without adversely affecting the accuracy of the results. These optimizations included algebraic manipulation of the SBP equation to allow higher speed computations, reduction of the number of cells required for each solution, conversion to fixed-point math, and the use of a structured search algorithm.

The hierarchal search method developed as part of this study, in addition to greatly increasing the search speed over current implementations, allows the memory requirements of the SBP method to be distributed across multiple stages, and across both internal and external memories. This allows a multi-stage pipeline solution to be used. This was a significant factor in increasing the speed of the SBP solution.

Collectively, these optimizations allow an accelerated SBP solution to be found at a rate that exceeds the maximum expected average event rate, to an accuracy that compares satisfactorily with existing methods. This project has shown that it is possible to use an FPGA implementation of SBP to facilitate real-time processing of event data for a PET system. This is a major step forward in the development of a research preclinical PET system being built at the University of Washington.

References

[1] J Joung, R.S. Miyaoka, T.K. Lewellen, "CMiCE: A high resolution animal PET using continuous LSO with a statistics based positioning scheme," *Nucl. Instrum. Meth. Phys. Res. A*, vol. 489, no. 1-3, pp. 584-589, Aug. 2002.

[2] T. Ling, K. Lee, R.S. Miyaoka, "Performance comparisons of continuous miniature crystal element (cMiCE) detectors," *IEEE Trans. Nucl. Sci.,* vol. 53, pp. 2513-2518, 2006.

[3] T. Ling, T.K. Lewellen, R.S. Miyaoka, "Depth of interaction decoding for a continuous crystal detector," *Phys. Med. Biol.*, vol. 52, pp. 2213-2228, April 2007.

[4] S. Hauck, A. DeHon (editors), "Reconfigurable Computing: The Theory and Practice of FPGA-based Computation", Morgan Kaufmann/Elsevier, 2008.

[5] D. DeWitt, "An FPGA Implementation of Statistical Based Positioning for Positron Emission Tomography", Master's thesis, Department of Electrical Engineering, University of Washington, 2008.