An FPGA Implementation of Statistical Based Positioning for Positron Emission Tomography

Donald Q. DeWitt

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

Master of Science in Electrical Engineering

University of Washington

2008

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 doctoral dissertation by

Donald Q. DeWitt

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:

University of Washington

Abstract

An FPGA Implementation
of Statistical Based Positioning
for Positron Emission Tomography

Donald Q. DeWitt

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

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 implementation, these algorithm improvements result in a reduction from $>7\text{x}10^6$ floating-point operations per event for an exhaustive search to $<5\text{x}10^3$ integer operations per event. Simulations show nearly identical FWHM positioning resolution for this accelerated SBP solution, and 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 the maximum expected rate of ~250k coincidence events per second for a 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, further dividing the computational load. These methods allow SBP results to be calculated in real-time and to be presented to the image generation components with virtually no delay. A prototype hardware implementation has been tested, limited to 4 stages due to memory limitations of this system. A circuit is currently under development to allow implementation of the full seven stage algorithm.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The author wishes to express appreciation for the support of the National Institute of Health (NIH) National Institute of Biomedical Imaging and Bioengineering (NIBIB), R21/R33-EB0001563 for the funding that made this project possible.

1 Positron Emission Tomography

Positron Emission Tomography (PET) is a powerful medical imaging technique that is able to form three-dimensional images of the level of metabolic activity in the subject. This is especially useful for studying biochemical processes involving blood flow and glucose metabolism.

The PET imaging process begins with the creation of a short-lived radioisotope, such as Fluorine-18. This atom is attached to a metabolically active molecule of interest, e.g. deoxyglucose (an analog of glucose), to produce fluorodeoxyglucose, and then introduced into the subject. An intervening waiting period allows metabolic processes to concentrate the radiotracer in areas of high metabolic activity. As the radioisotope decays, positrons are produced. Within a short distance, the positron will encounter an electron, and annihilate. This process produces pairs of high-energy (511-keV) photons, Figure 1, which are emitted from the point of the annihilation at the speed of light.

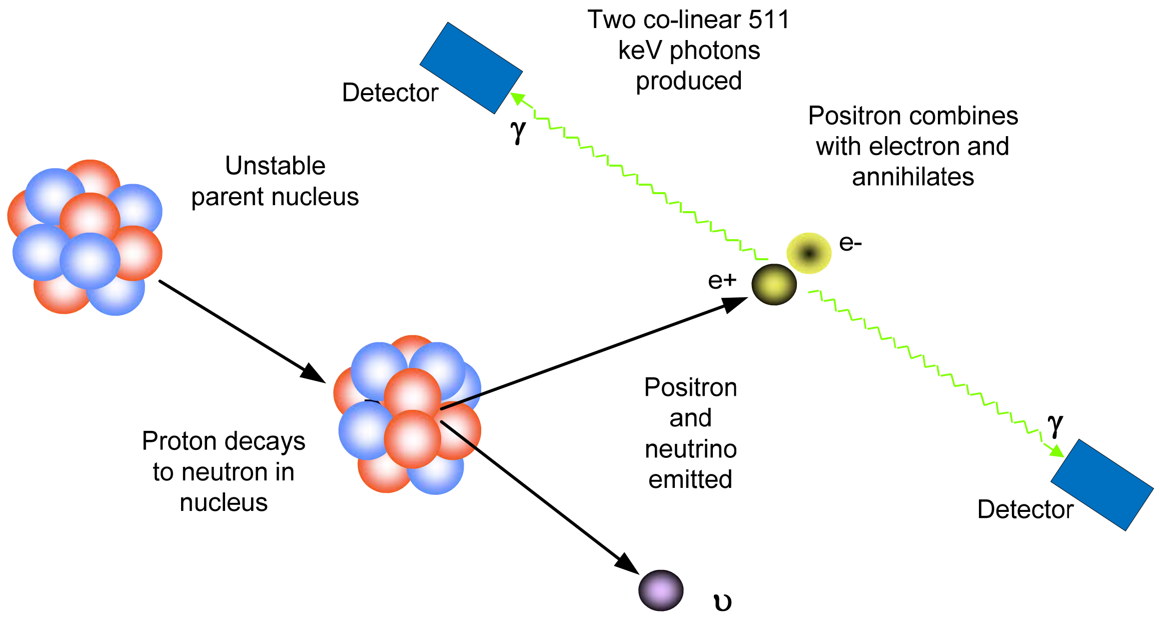

Two co-linear 511 keV photons produced
Detector
γ
Positron combines with electron and annihilates
Unstable parent nucleus
e-
e+
Proton decays to neutron in nucleus
Positron and neutrino emitted
γ
Detector
υ

Figure 1. Positron decay and detection.
(Image courtesy Michael Haselman.)

Conservation of momentum, and the relatively low net momentum of the causative positron and electron, requires these two photons to be emitted at almost exactly 180 degrees to one another. Detectors are arranged in a circular array surrounding, and roughly centered on the subject. Since both photons are traveling at the same speed, the speed of light, and they have traveled nearly the same distance, precision timing circuits can be used to differentiate these photons from most randomly occurring non-paired events (i.e., single events). Single events occur when only one out of the two annihilation photons are detected by the tomograph. Coincidence detection allows data from single events to be discarded without further processing. When the coincidence detector determines that two photons are paired, they are assumed to have originated from the same source, and to have originated somewhere along the line connecting the two points of detection. For all events that are accepted by the coincidence detector, a "line of response" (LOR) is determined, Figure 2.

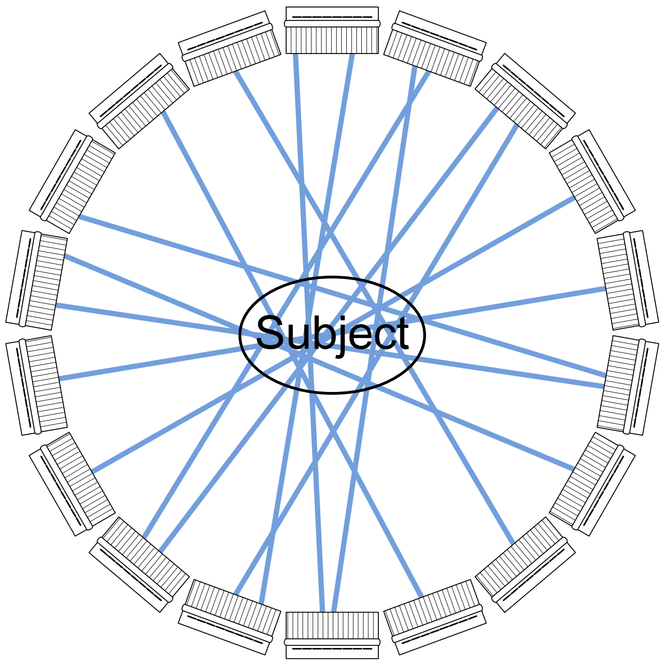


Figure 2. Lines Of Response.

As the emissions continue, a large number of individual LORs can be determined, with many thousands to millions of LORs being collected during a typical PET procedure. Computer reconstruction techniques are then used to create a map of the intensity of emissions from areas within the subject. A complete PET system is shown in Figure 3.

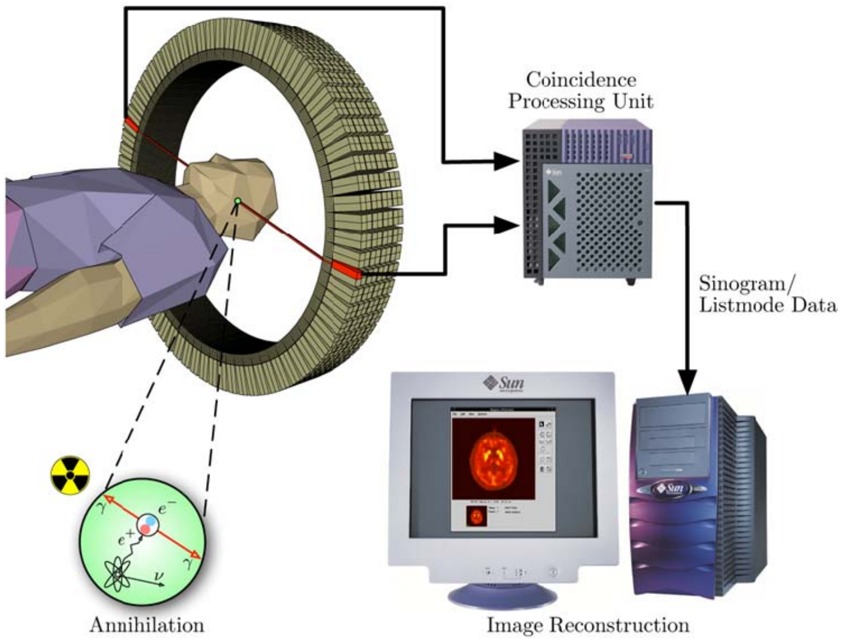


Figure 3. PET Imaging system.
(Image courtesy Jens Langner.)

A reconstructed PET image represents the concentration of the isotope caused by biological processes in the subject, as shown in Figure 4. It is apparent that PET images have poorer spatial resolution than anatomic imaging techniques such as Computer Tomography (CT) or Magnetic Resonance Imaging (MRI), as shown in Figure 5. To improve the interpretation of the metabolic activity visible in a PET image, PET images are often overlaid on higher resolution CT or MRI images.

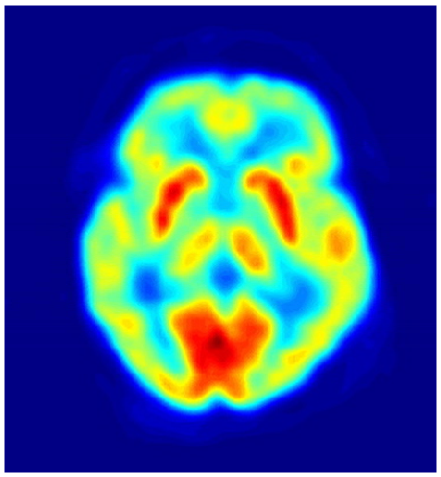


Figure 4. PET Image.
(Image courtesy Jens Langner.)

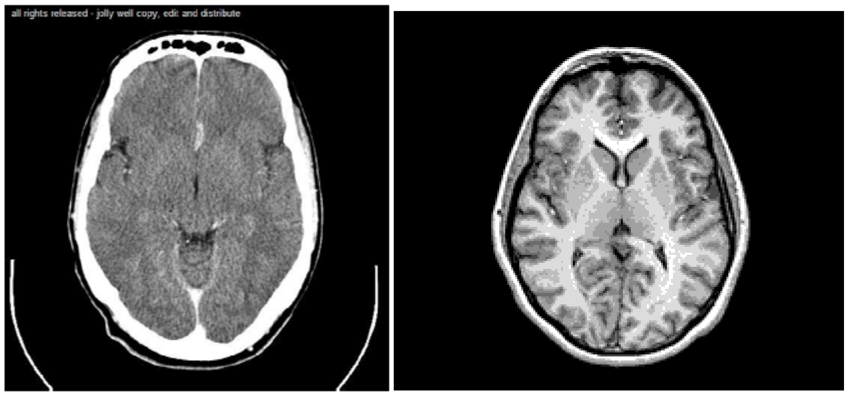


Figure 5. Computer Tomography (CT) and Magnetic Resonance Images (MRI). (Image courtesy Mikael Häggström.)

All of these medical imaging techniques are improving due to the results of research in these areas. Much of the current activity in PET scanner improvements are centered on improving image resolution, reducing imaging time, and improving the accuracy of the reconstructed image.

1.1 Resolution of PET Imaging Systems

The most apparent limitation of the PET imaging system is the lower resolution when compared to MRI or CT scans. Several factors are responsible for this.

- The recombination distance for the original positron, which may travel up to several millimeters before annihilating and creating the high-energy photons.
- The opposing photons do not travel exactly 180 degrees opposite to each other due to the contribution of the momentum of the electron and positron at annihilation. Scattering can also contribute to this source of error.
- Smoothing filters applied during the image reconstruction process to control the statistical noise in the acquired data.
- Resolution limitations in the components of the photon detection and processing systems.

The first three factors will not be addressed here. The last factor is the result of geometric and electronic aspects of the physical detection system, along with the methods used to translate the resulting electronic signals into position data. This project seeks to develop a real-time processing solution for an advanced positioning algorithm that will support high resolution PET imaging.

1.2 Positron Detection

The operation of the PET system depends on the accurate detection of "events", consisting of the photons emitted from the subject as a result of the positron decay. These high-energy (511-keV) photons (gamma particles) are not directly detectable by most electronic photon detectors. To allow these photons to be electronically sensed, an energy conversion layer is introduced, known as a scintillator. This consists of a material that is able to intercept the high-energy photons and then re-emit visible light. The scintillator crystal is transparent to these light photons, allowing these photons to pass out of the scintillator where they can be detected electronically. The conversion process results in a high-energy photon being converted into a large number of visible light photons that are free to spread as they continue to the electronic sensor, as shown in Figure 6.

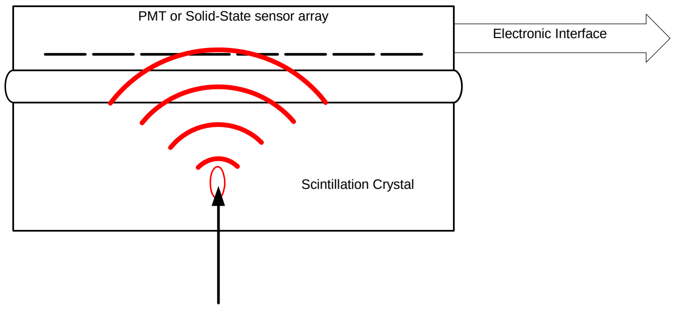


Figure 6. Dispersed output from a continuous scintillator crystal.

This arrangement is known as a monolithic crystal detector, suggesting that the crystal is one continuous piece. In this configuration, it is apparent that the visible light photons will not strike a single sensor element in the sensor array, but instead will spread across a number of sensing elements. In our implementation, which we call the cMiCE (for continuous miniature crystal element) detector, the monolithic crystal is coupled to a 64-channel multi-anode photomultiplier tube (PMT). A cMICE detector is pictured in Figure 7. A representative distribution of energies detected for one event by the 8X8 PMT is shown in Figure 8.

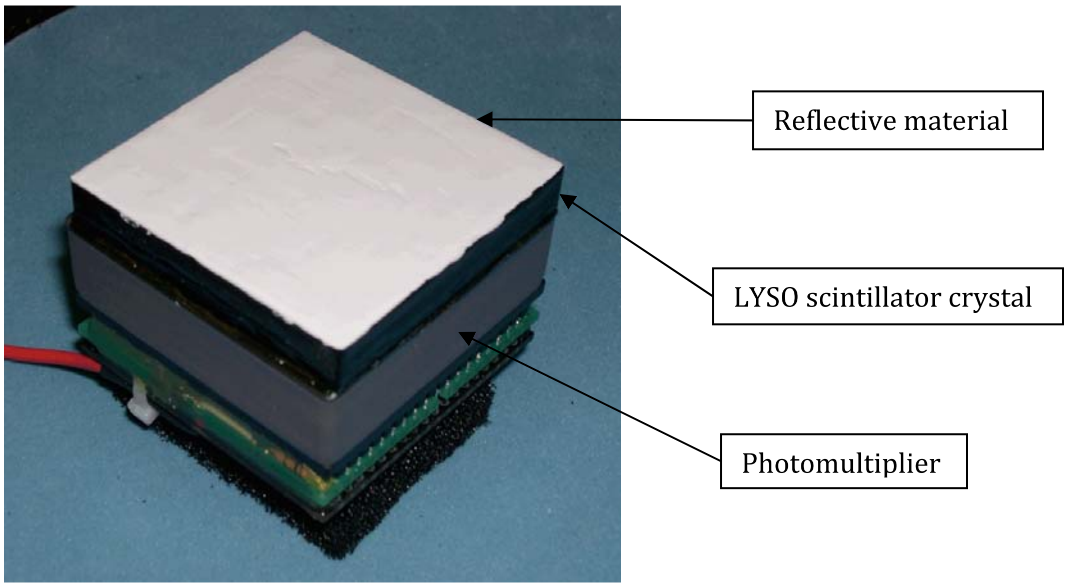


Figure 7. Picture of cMiCE detector with 6 mm thick LYSO crystal and white paint. (Image courtesy Robert Miyoaka.)

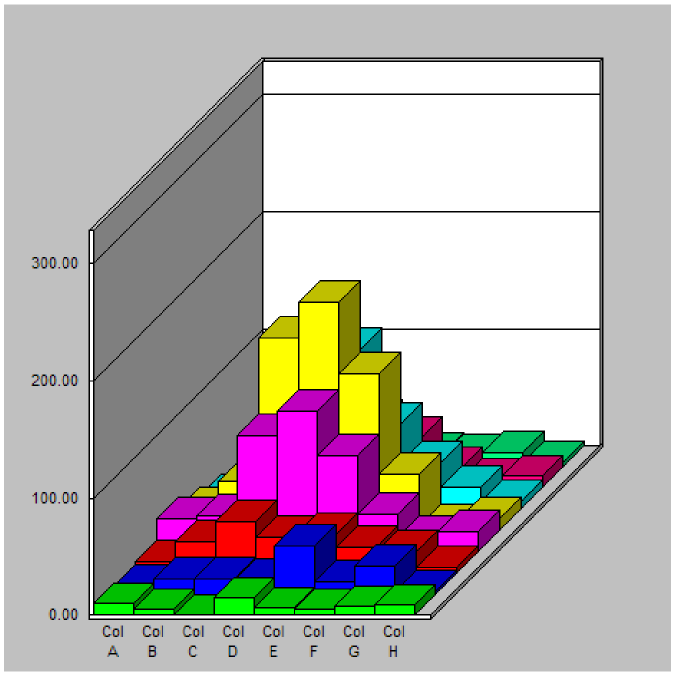


Figure 8. A representative event, showing the response of an 8x8 PMT array.

Determining the sensor element where the peak energy was sensed only allows an 8x8 resolution to be obtained from a standard 8x8 sensor element. Improvement beyond the 8x8 resolution can be obtained using curve-fitting and interpolative methods. A detection problem that is poorly addressed with these methods is known as “edge-effect”.

The quality of the signal from the elements along the edges of the electronic sensor are affected by differences in the response of the scintillator caused by the abrupt edge of the crystal. This edge creates reflections and truncation of the light response seen by the sensor. Additionally, the electronic sensor itself can have a different response along its edges, especially to oblique signals. This can cause the resulting position information to be distorted along these edges and can cause large portions of the sensor area to be relatively ineffective (i.e., positioning in those regions is severely biased).

One method currently being used to increase the resolution and the effective area of the sensor involves installing a large number of small scintillator crystals in a reflective matrix. This configuration is known as a discrete crystal array [2], Figure 9. This causes the visible light photons to be constrained to a smaller area of the electronic sensor, making the determination of the location of the event more accurate than using the same algorithm in the continuous crystal. This also reduces the edge effect, allowing a larger useful imaging area of the electronic sensor.

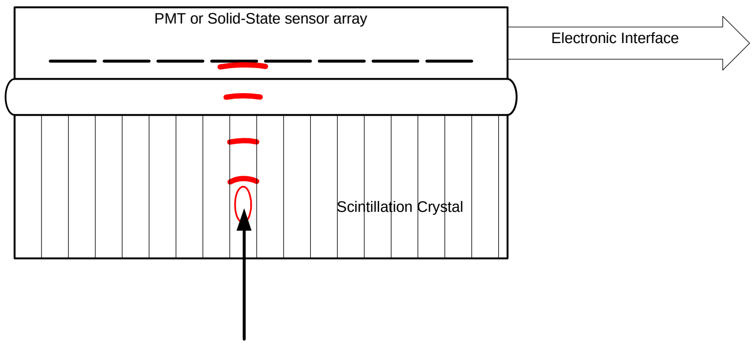


Figure 9. Concentrated output from a “discrete” scintillator crystal array.

Although the discrete array does correct some of the problems present in the continuous array, new problems arise. As can be seen in a photo of an actual scintillator array in Figure 10, this consists of hundreds of sub-millimeter rectangular crystals arranged in a reflective grid. Each of these crystal slivers have to be cut to shape and polished, then placed by hand in the reflective matrix. This is a labor intensive and costly process requiring many hours per sensor for each of the sensor positions in the system. The physical interface between the discrete array and the electronic sensor is also more problematic than the continuous array. The fabrication of the sensors in this system constitutes a significant increase to the overall system costs.

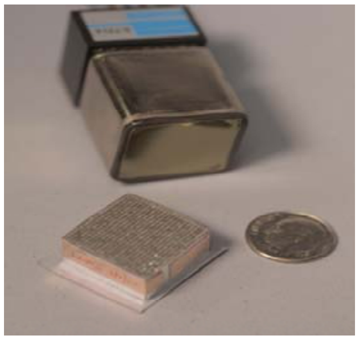


Figure 10. Example of a discrete scintillator crystal [3].

A secondary problem with the discrete array is that the reflective matrix occupies a part of the exposed area of the sensor without contributing to the scintillator area. Additionally, the crystal segments require a clearance to allow them to be assembled into the reflective array. Collectively, this results in a significant decrease in overall sensitivity. To allow collection of the same quantity of events in a discrete system, an increase in exposure is required. This can only be accomplished by either increasing the radioisotope dose, or increasing the exposure time, both of which are undesirable options.

2 Statistics Based Position, SBP

Another method of increasing the resolution of the photon sensors is known as Statistics Based Positioning (SBP) [1]. SBP is able to improve the overall detection characteristics of a continuous crystal detector. This allows the labor intensive construction process of the discrete crystal array to be eliminated. Using a continuous crystal also increases the effective area of the sensor in that the active area of the crystal is increased to 100% by eliminating the reflective matrix and clearance space.

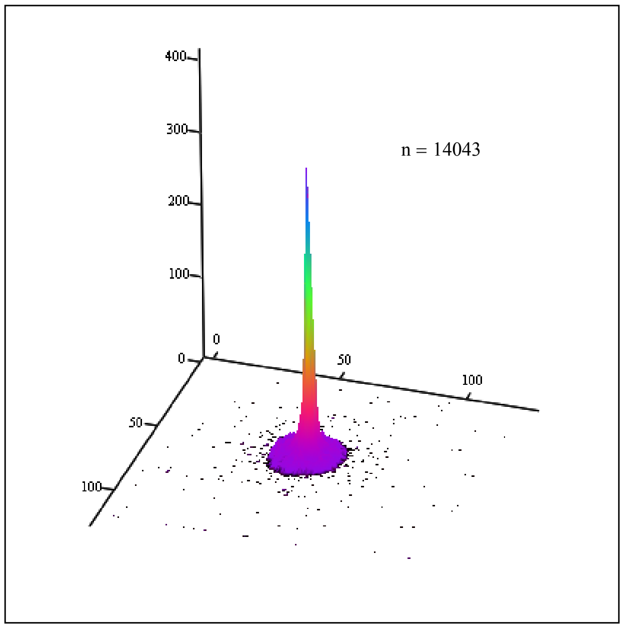
2.1 Description of the SBP algorithm

In a SBP system, each sensor array requires an initial characterization step. This involves positioning a source in the field of view, collecting data from each of the electronic sensors in the array, and saving the light response characteristics for each X-Y location in a table for future reference. The two statistical characteristics that are stored are the mean (μ) and variance (σ^2) of the light distribution function of each sensor for a given X-Y location. Later, when data is collected from an unknown location, it is compared with the previously collected data table using Equation 1, and the coordinates of the characterization data that statistically most closely matches the unknown data, i.e., having the maximum value for Equation 1, are taken to be the position of the unknown source. Using this method, position resolution much finer than the spacing of the individual detectors within the 8x8 arrays is achieved.

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

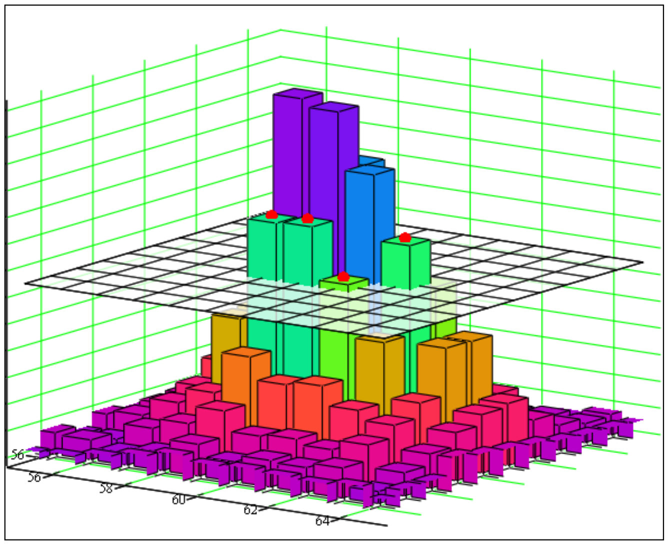
Equation 1. The equation to determine the likelihood that an event occurred at a sampled X,Y position.

A histogram of the results of 14,043 co-located events, processed using the SBP algorithm with a 127x127 X-Y characterization data-set is shown in Figure 11. A horizontally expanded view of this peak is shown in Figure 12. A standard that is customarily used to judge the quality of this result is known as "Full-Width at Half-Maximum" (FWHM). This is the value of the width of the cross-section at a height of half of the maximum value, represented by the horizontal plane intersecting the peak in Figure 12.



hxy$_{fp}$

Figure 11. Histogram plot of SBP results for a representative fixed point source.



h_w, h_0, h_1

Figure 12. Expanded view of histogram peak.

Calculating the FWHM for discrete data points as in this histogram leads to results that are difficult to interpret, so an alternate representation was used. In this version, Figure 13, the value of the histogram at each pixel has been connected with a straight line to the adjacent pixels.

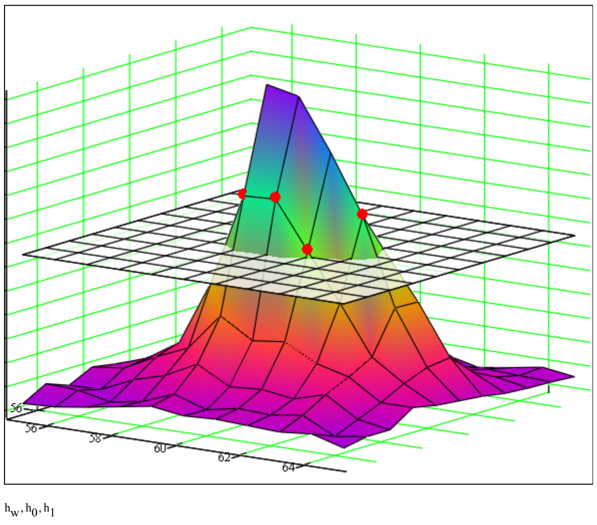


Figure 13. Alternate representation of histogram data.

In Figure 14, the X-Y positions of the intersections of these lines and the plane are calculated (hollow blue dots), and the vertexes of the convex hull are connected (blue line). This is then divided into triangles (black lines), and the areas of the triangles are summed. The “Full-Width” is taken as the diameter of the circle that has the same total area as the polygon. To allow greater flexibility than a fixed FWHM calculation, the plane in this figure will be allowed to sweep from the top to bottom of the histogram and a series of intersections are calculated. As the plane is swept across the height of the histogram, a profile of the “Full-Width” at each increment is plotted on the graph shown in Figure 15. From this graph, the “Full-Width at Half-Maximum” can be taken as the X coordinate of the point that intercepts the horizontal line drawn at half of the maximum Y value.

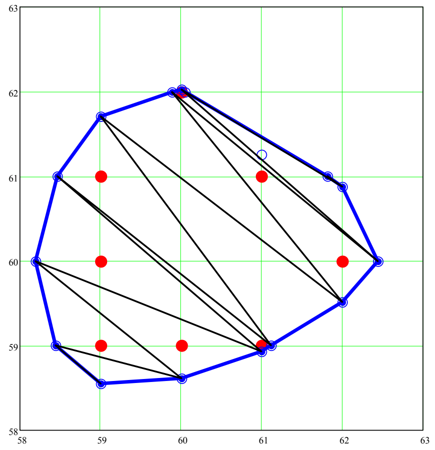


Figure 14. Convex hull and area calculation.

For this representative data, the FWHM is shown to be approximately 1.4 mm (approx. 0.4 mm/pixel). For the 127x127 data-set used to find this result, this is equivalent to an increase in resolution of approximately 4 times the physical resolution of the 8x8 sensor array.

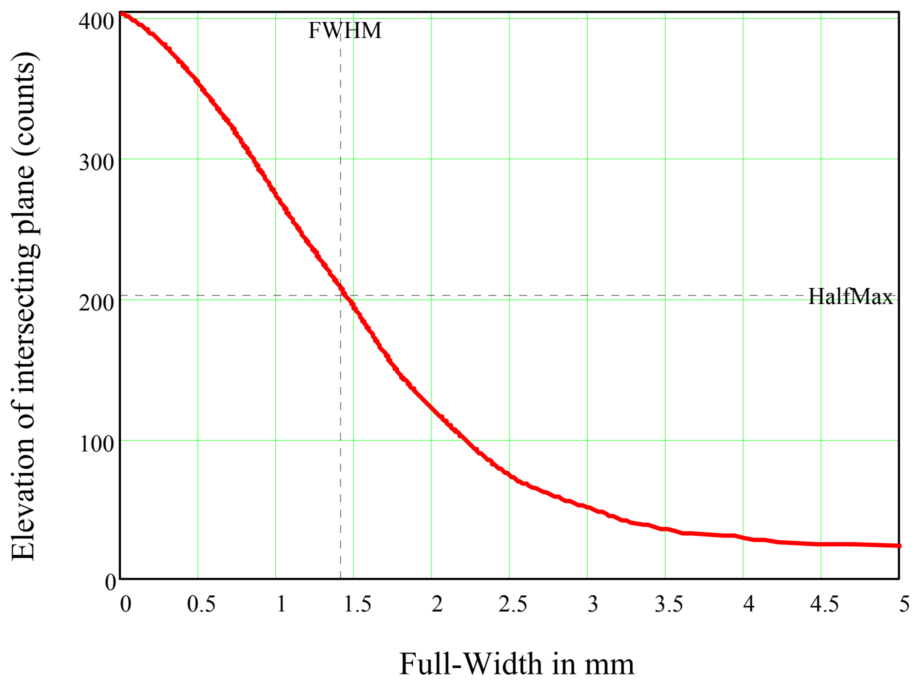


Figure 15. Full-Width profile of dataset shown in Figure 11.

2.2 Current implementations of SBP

Current systems implementing SBP collect and store raw data from the sensors during the period of the PET scan. This can involve the collection of data for many millions of events. When the data collection is complete, this information is transferred to a powerful processing station. The process of determining the positions of each event and then composing the image can take several hours. Processor arrays can speed this up, but image formation times are a problem with the SPB method due to the computationally intensive nature of this process.

The first implementations of SBP used an exhaustive search to find the best statistical fit of the data for each unknown point. In the exhaustive search, the likelihood function is computed for every potential location in the characterization data-set. This resulted in a simple algorithm that was always able to find an optimum solution but put a heavy load on processing resources. The solution for each event's possible position requires 8x8 statistical computations, each containing approximately 6 floating-point operations. The desired 127x127 possible positions for each sensor results in $6 \times 8 \times 8 \times 127 \times 127 \approx 6$ Million Floating Point Operations (MFLOP) per detected event, for a PET scan that may contain millions of events. This resulted in a computation time that was unworkable.

In an effort to shorten the computation time, a method termed "Localized Exhaustive" was employed. In this method, it is assumed that the solution is near the location of the detector channel with the maximum response. An exhaustive search can safely ignore all values far away from the location of this channel. This can increase the search speed to approximately 1/30 the time required for the exhaustive search, depending on the margin included around the maximum channel. This results in a much shorter processing time than the exhaustive search, but is still long enough to limit the usefulness of the SBP method.

Since the beginning of this project, a structured search algorithm has been developed for the post-processing environment. This method works using a combination of the localized search to limit the area that is searched, followed by a sampling hierarchal search, similar to that which will be described below for this project. This has greatly reduced the delay between collecting the data and presenting the positioned results.

A remaining limitation of the current post-processing system is the intensive use of communication channels on the PET system. Each of the sensors in current systems has a communication channel to the host computer. When an event occurs, the information for each of the 64 channels in each sensor is transferred to the host via the communication channel. This results in a large volume of information being transferred on the system's communication channels for each event, whose end effect is to convey a single X-Y position. This is a very inefficient use of the communication channels that could be improved greatly if the X-Y position could be determined at the sensor and only the X-Y information would be transferred. This would result in approximately a 64 to 2 decrease in the communication bandwidth used. Using an FPGA for each individual sensor would also allow the information from each of the sensor systems to be processed separately, dividing the processing load across multiple devices.

3 Field Programmable Gate Arrays (FPGA)

The requirements of the SBP solution in a PET system are a good match for the capabilities of a “Field Programmable Gate Array” (FPGA). An FPGA is a special purpose integrated circuit designed to perform complex interface and logic processing in a small footprint. The natural tendency is to view the FPGA as being similar to a microprocessor. It is supplied in a similar package, and the name even includes the word “Program”. The same editors can often be used to write code very similar to that which is run on microprocessors, and many of the same results can be achieved. However, the FPGA is fundamentally different from a micro-processor. The microprocessor consists of a complex, but fixed, hardware configuration and performs operations under the control of a program fetched from memory. The FPGA, in contrast, is actually 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. Although this difference may seem minor, it allows the FPGA a flexibility that is not available with microprocessors.

An analogy of an FPGA is the well-known electronic bread-board. With this one tool, a wide variety of different circuit configurations can be created by inserting inter-connecting wires between electronic components. If a modification to the current circuit or even a completely different circuit is needed, this can easily be done by rearrangement of the wires. In this analogy, a microprocessor would be seen as a printed circuit board. A more complex circuit may be more efficiently implemented in this technology, but changes are virtually impossible to apply without reworking the entire manufacturing process.

An FPGA is the Integrated Circuit (IC) implementation of a circuit breadboard. This IC breadboard can consist of tens to hundreds of thousands of logic elements and the interconnects required to connect them in almost any way imaginable. Although the total complexity may not be as high as some microprocessors, the ability to easily re-wire the circuit is an important part of the flexibility of the FPGA. This allows the FPGA to be configured to precisely match the requirements of the system being designed. Many current FPGAs, in addition to a large array of simple logic circuits, have included higher functions such as arithmetic units and memory blocks. Some FPGAs even have low to medium complexity processors included as part of the matrix of interconnectable components.

Although the FPGA chosen for this project does not include an embedded processor, a processor can be constructed and included as part of the overall design by interconnecting a large number of the existing logic elements according to a pre-designed configuration.

The FPGA family chosen for this project is the Altera Stratix II. A representative floor-plan of this device is shown in Figure 16. This diagram shows that the most prevalent component of the FPGA is a large array of simple logic elements. Special purpose arithmetic units and a variety of different sized memory blocks are interspersed within this array. Interfaces for external memories and other devices such as high-speed Analog to Digital Converters (ADC) are also included in this particular device.

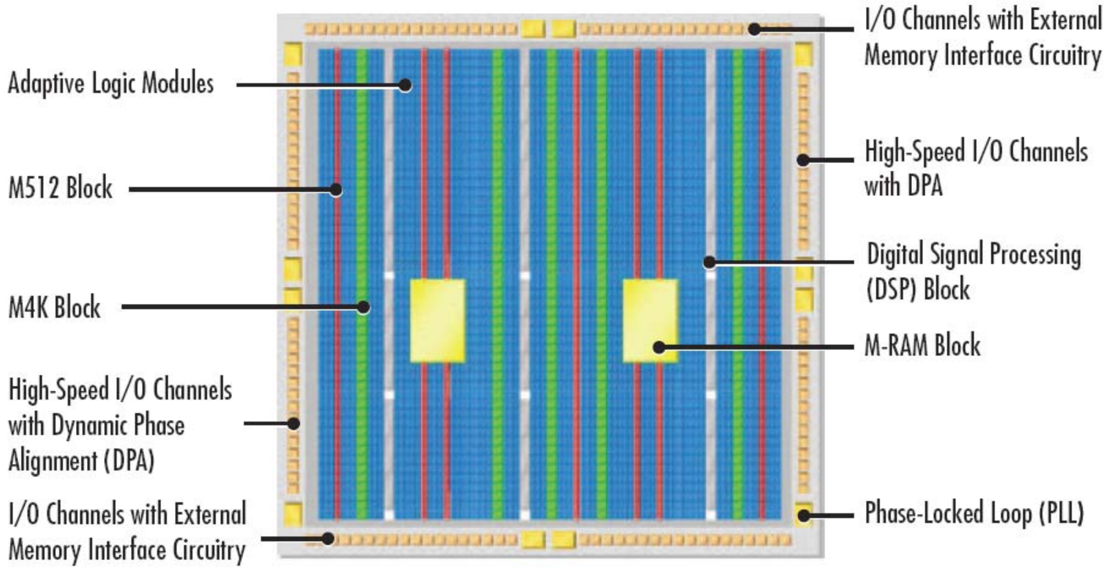

I/O Channels with External Memory Interface Circuitry
Adaptive Logic Modules
M512 Block
High-Speed I/O Channels with DPA
Digital Signal Processing (DSP) Block
M4K Block
M-RAM Block
High-Speed I/O Channels with Dynamic Phase Alignment (DPA)
Phase-Locked Loop (PLL)
I/O Channels with External Memory Interface Circuitry

Figure 16. Floorplan of Stratix II EP2S60 FPGA.
(Image courtesy Altera Corporation)

Not shown on this map are the layers of interconnects that overlay this matrix. The general arrangement of these layers is a dense mesh of horizontal and vertical conductors. At many of the junctions of these interconnects, configurable switches allow signals to either continue through, or to be routed to conductors in a perpendicular direction. Connections between individual components within the array can be formed by selectively enabling or disabling these switches. Development software allows the configuration of these switches to be controlled in a way that causes complex logical functions, and in a similar way an entire system, to be built up from the array of simpler components. Much of the current research on FPGAs concerns the structure and function of these interconnects and switches to provide greater potential to design software. Advancements in design software take advantage of this potential by allowing the high-level design of complex circuits. This is done by allowing Hardware Description Languages (HDL) and schematic design to be efficiently converted and mapped to the array of interconnects and switches present on the FPGA.

4 Design of the new SBP system

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 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.

4.1 Requirements

A problem with some methods is the latency between the collection of data and the availability to the user of a finished image. A poorly designed post processing method can take as much as several hours to complete.

The time from exposure to the image being available needs to be minutes or less to allow SBP to be valuable in a research or diagnostic environment. This time reduction will be one of the requirements of the new system. As one of the methods of decreasing the latency, each event will be processed as it happens, rather than collecting the entire data set before beginning the processing step. This will allow the SBP processing to proceed in parallel to the data collection phase.

Although the time to process each event may exceed the time between some pairs of events, the average rate of processing must keep pace with the average rate of events generated by the system. Otherwise, events will overtake the rate of solutions, requiring that event information be stored to be processed at some time in the future. This quickly degenerates into a similar system as the post processing method, where events are stored and solved after the data collection is completed. In order to solve for event positions at a rate that equals the generation of new events, multiple parallel computation paths must be provided. This requirement will ensure that the SBP solution is available almost immediately after the last event data is collected.

At dosages currently used in this system, photon pulses are collected by each of the sensor arrays at a peak average rate of approximately one per 100 cycles of the 70 MHz system clock. Of these, the coincidence detector rejects approximately two out of each three detected pulses, resulting in approximately one positron event every 300 clock cycles for each of the sensor arrays. This is the rate that the new SBP system must be able to process incoming events.

4.2 Hardware Optimizations

4.2.1 Multi-Processing

In order to meet the processing rate required to match the incoming data rate from the sensors, multiple parallel processing paths will be required. The first level of parallel processing will be implemented by providing each sensor with its own FPGA. This will provide the system with multiple times the processing power of a centralized processing solution.

4.2.2 Memory Organization

High-speed access to the characterization data tables is fundamental to the success of any design for SBP. We explored several possibilities in developing this project's solution.

4.2.2.1 Using Only Internal FPGA Memory

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 SRAM memory. Processing the events would be simplified compared to accessing off-chip SD or 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 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]. This would require over 6 Mbytes, well above the approximately 300 kB to 1.1 MB available in members of the family of FPGA devices chosen for this project. By interpolating some values and mirroring values based on the symmetry of the device geometry, a significant reduction in the table size could be obtained. The overall memory requirement could be reduced to as low as approximately 1 Mbyte. This is slightly below the maximum FPGA memory available, but there are several reasons that this is still not adequate.

- Although the total number of memory bits slightly exceeds the requirements, memory allocation is in distinct sized blocks. This block size is generally not an even divisor of the memory size required for each table. This causes some memory to be unused and unavailable for use by other components in the system.
- There are other operations to be performed by this system. Significant additional memory resources will be required to support these functions.
- The response of each channel of the electronic sensor is not constant across the face of the sensor. This would make the mirroring of values based on symmetry impossible without first equalizing the response of the channels. This is not planned for the current system.
- 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 in this future version that will certainly require access to external memory.

For these reasons, the option of using only on-chip memory was discarded.

4.2.2.2 Shared Data Tables

Each FPGA in this system will be processing the data-stream from one sensor array. The entire system will consist of a ring of multiple sensor arrays, each with a separate FPGA-memory system. One possibility for this system would have each device containing only a portion of the data tables. The incoming data-stream from each sensor array would be sorted to determine which subset of the data table would be required to solve for the position of this event. If the local FPGA was not the one that contained the proper subset of the tables, the data, along with the identification of the device where the data originated, would be communicated to the proper FPGA system by means of existing communication channels. This FPGA would then process the data using its subset of the tables. When the solution was complete, it would be communicated to the host, along with the identity of the originating device.

This solution suffers from the difficulty that the μ and σ characterization tables for each of the sensor arrays are not identical. The manufacturer includes a calibration table with each sensor array that specifies a scalar for each of the 64 separate channels due to manufacturing variations. These can vary from 100% to as low as approximately 30%. The differences between sensor arrays require individual characterization tables to be created. Since the characterization tables for each sensor array are different, it is not possible to divide a shared large table across multiple devices.

As in the previous approach, the possibility exists to normalize the data generated by the 64 channels of the sensor by dividing by the values in the calibration data. This would still only account for one aspect of the difference between the sensor arrays. Each electronic sensor array is mechanically attached to a scintillator crystal that converts the incoming 511 keV photons to visible light photons that can then be detected by the sensor array. Fine details in the manufacture, shape, edges, and alignment of this crystal with the face of the sensor array, along with the characteristics of the attachment epoxy, and other mounting details, can affect the overall sensitivity of the individual channels. This information cannot reasonably be controlled. These differences alone would be enough to cause the tables to be different for each detector module.

This configuration would often require that 64 channels of information be transmitted across existing communication channels to another processing station. However, the communication channels are already experiencing problems with too much data traffic. Indeed, one of the requirements of this system is to allow the 64 channels of data to be condensed to a single X,Y location value before being transmitted. The addition of sensor data being transmitted between processing stations would increase the traffic beyond the system capabilities.

For these reasons, the option of using shared tables was also discarded.

4.2.2.3 Using only External Memory

The large size of the data tables required for each detector force the use of off-chip memory to implement this system. Although using off-chip memory eliminates several problems, it creates others. Events occur with an average separation of 300 clock cycles at each sensor. For each incoming event, a large number of memory accesses will be required to find each solution. For each solution, values from all three tables for each of the 8x8 channels must be read. Assuming a 48-bit wide value for the table entry at each X-Y location, and 300 clock cycles available for each solution, [127x127][8x8]*48/300 ≈ 160k memory bits per clock cycle are required for the current exhaustive search implementation of SBP. Even considering multiple reads per clock cycle, this would require an impossibly wide data path.

It is apparent that neither a fully internal nor a fully external memory implementation will allow an FPGA to solve the current SBP algorithms. In the following sections we will present an alternative approach. This approach will use mathematical optimizations, a more efficient algorithm, and will take advantage of features available through using an FPGA. The result is a system that will use SBP to accurately report the position of events in real-time.

4.3 SBP Equation modifications

4.3.1 Mathematical Simplifications

In order to use SBP to create real-time position solutions, a more efficient solution must be developed. Two simplifications are available in the original equation, Equation 1. Since we are looking for relative values, i.e., the maximum of solutions of this function, and the natural logarithm is a monotonically increasing function, the natural log of the result can be eliminated. Additionally, rather than looking for the maximum of the negative value of the function, we will remove the negative sign from the right side of the equation and look for the minimum. This results in Equation 2.

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

Equation 2. Simplified Likelihood equation.

The next step is to reduce the complexity of each summation. For each of the rows and columns of the detector, and at each potential X-Y location, the value of Equation 3 must be determined. The variable E represents the energy of one channel of the detector, μ and σ represent the corresponding values for that channel at the X-Y value currently being summed.

$$R = \frac{(E - \mu)^2}{2 \cdot \sigma^2} + \ln(\sigma)$$

Equation 3. A single value of the SBP summation.

The objective is to manipulate the function to minimize those numerical operations which are expensive in terms of time or resources to perform. 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 results by 2.
5. Divide the results of #2 by the results of #4.
6. Find the natural log of σ.
7. Add the results to the results of #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 precomputed inverse if possible. Following the guidelines given above leads to the simplification shown in Equation 4.

$$R = \frac{(E-\mu)^2}{2\cdot\sigma^2} + \ln(\sigma) = \left[(E-\mu)\cdot\frac{1}{\sqrt{2}\cdot\sigma}\right]^2 + \ln(\sigma)$$

$$\text{let :} \qquad A = \mu \qquad B = \frac{1}{\sqrt{2}\cdot\sigma} \qquad C = \ln(\sigma)$$

$$R = [(E-A)\cdot B]^2 + C$$

Equation 4. Simplification of SBP formula.

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 precalculated by the host before being stored in the data tables used for this calculations, 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.

4.3.2 Twenty-One Channel Solution

In the 8x8 solution to the likelihood equation, it was found that the channels with low values added little to the solution. A solution that ignores the channels that are far from the channel with the peak energy yields results that do not introduce any significant errors to the solution set (this result will be shown further below). However, solving and summing only the values closest to the peak could significantly reduce the processing load. Experiments showed that using only the 21 values closest to the peak, as shown in Figure 17, was adequate for results that closely matched those found with the exhaustive solution.

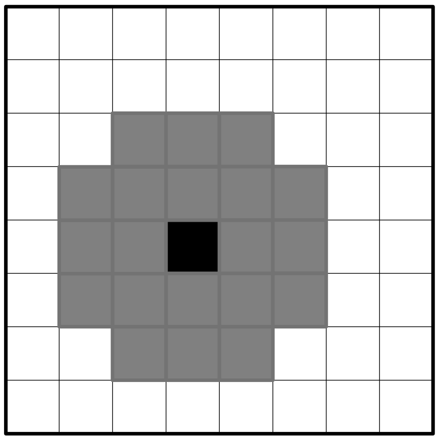


Figure 17. Likelihood function using only 21 channels adjacent to the peak energy.

4.3.3 Fixed-Point Mathematics

Another of the simplifications that are needed to optimize the FPGA implementation of the SBP algorithm is the conversion of the solution of Equation 4 from floating-point math to fixed-point math. Along with this conversion comes some degree of rounding or truncating of both the values stored in the data tables, and the values of intermediate calculations. Offline resources can be used for rounding values to be saved in the data tables, but truncating would be most efficient for the intermediate calculations within the FPGA. Both of these will have an effect on the final result that depends on the degree of rounding and truncating applied to the values. In order to get some perspective on the number of bits required to maintain the accuracy of the result, another round of simulations were run, with varying degrees of rounding/truncating applied. The results were compared against the floating-point solutions.

Graphs of these results are shown in Figure 18 through Figure 28. These graphs show the RMS error between floating-point results and the results after rounding and/or truncating. These results were found using floating-point rounding and truncating functions so that continuous graphs could be produced. The results are graphed relative to the power-of-two that the rounding or truncating represents, i.e. a power-of-two rounding represented by -3 is rounding to the nearest 1/8th and +3 is rounding to the nearest 8. This is done because of the ease of rounding or truncating to a power of two in binary systems. Fractional powers of two are included to allow relatively smooth graphs to be created. Ultimately, only whole powers of two will be used in rounding the values used in this project. The rounding or truncating that can be applied to a value without causing an unacceptable error in the results, along with knowledge of the highest expected value for that variable, defines the dynamic range for that value, and thus the number of bits required to adequately represent it.

The first three graphs show the results of rounding each of the three data table variables stored in memory, with 21-cell and 64-cell solutions shown on the same graph. The next eight graphs show the results of rounding/truncating the intermediate results, first for 64-cell solutions, then for 21-cell solutions. It was found by visually observing the results that an RMS error of 5 or less was unnoticeable in the resulting image quality. A target error of between 1 and 2 was selected as a maximum allowed error to insure that the results would not be noticeably affected.

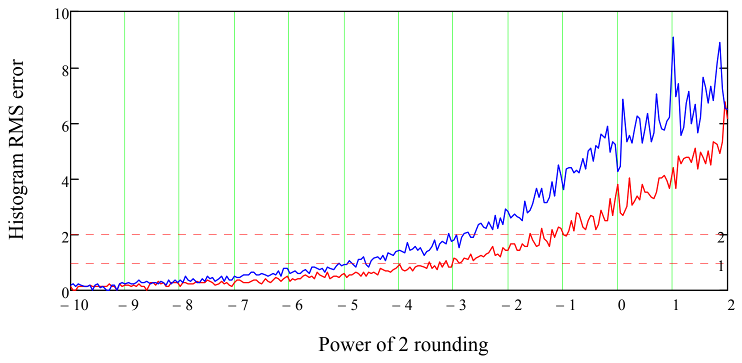


Figure 18. Effect of rounding A for 21/64 cell calculations.

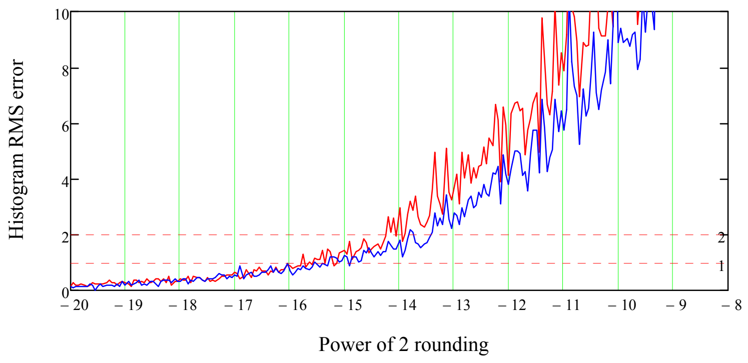


Figure 19. Effect of rounding B for 21/64 cell calculations.

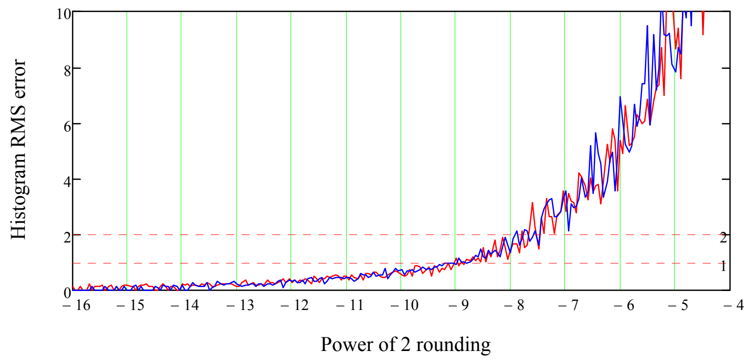


Figure 20. Effect of rounding C for 21/64 cell calculations.

The following graphs show the results of truncating/rounding intermediate calculations

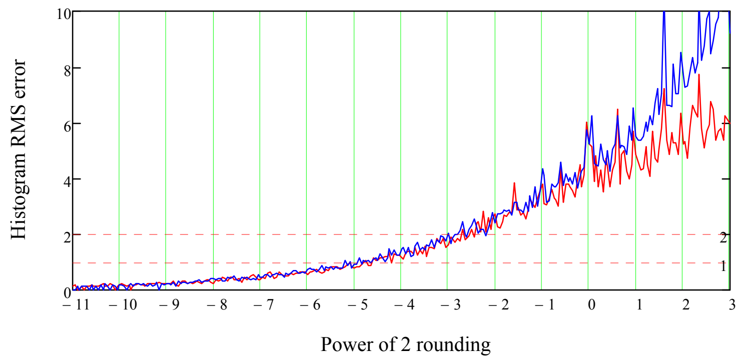


Figure 21. Effect of truncating/rounding the 64-cell results of (x-A).

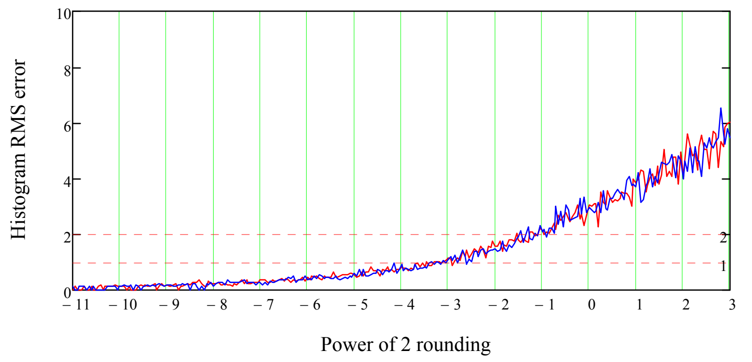


Figure 22. Effect of truncating/rounding the 21-cell results of (x-A).

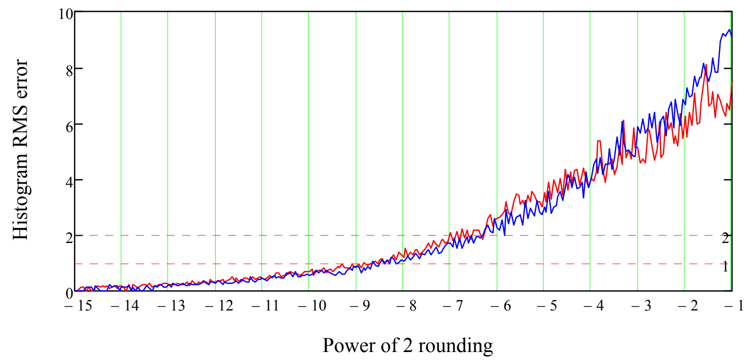


Figure 23. Effect of truncating/rounding the 64-cell results of (x-A)*B.

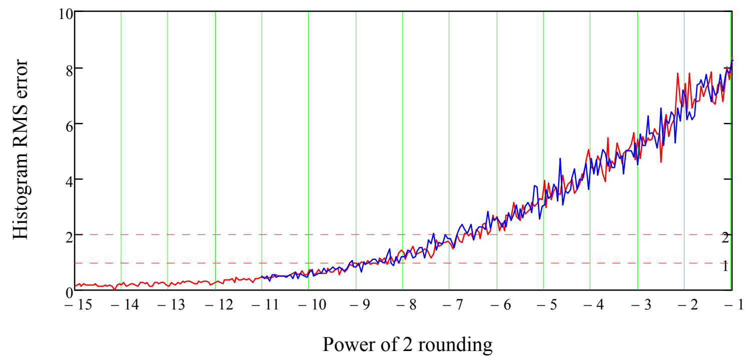


Figure 24. Effect of truncating/rounding the 21-cell results of (x-A)*B.

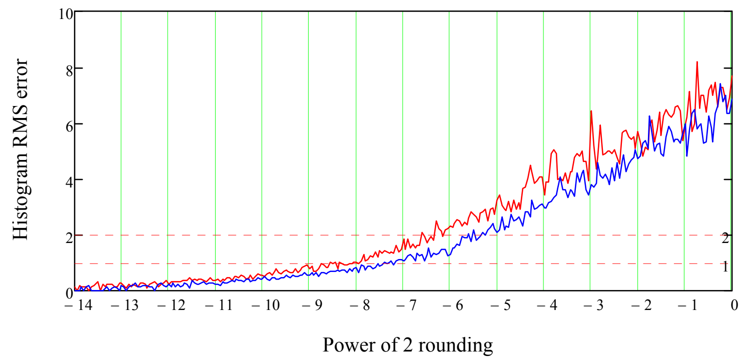


Figure 25. Effect of truncating/rounding the 64-cell results of ((x-A)*B)^2.

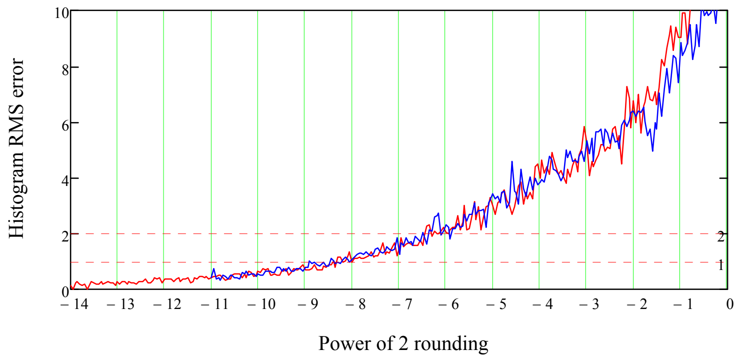


Figure 26. Effect of truncating/rounding the 21-cell results of ((x-A)*B)^2.

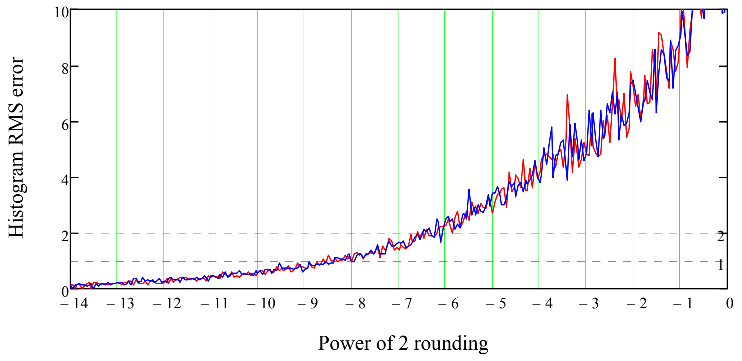


Figure 27. Effect of truncating/rounding the 64-cell results of ((x-A)*B)^2+C.

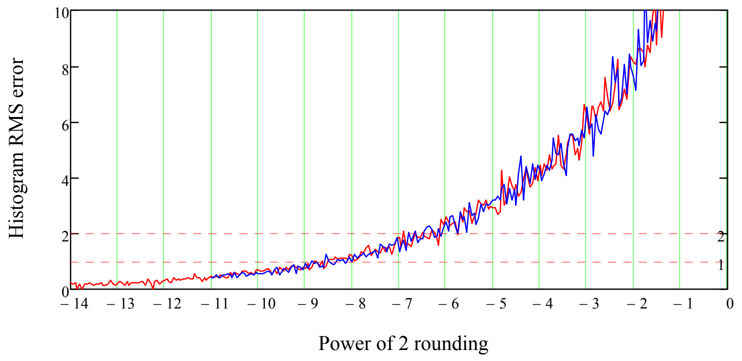


Figure 28. Effect of truncating/rounding the 21-cell results of ((x-A)*B)^2+C.

These results, coupled with tests showing the maximum values present for each of these values and computation steps, show that the three table values will have sufficient accuracy when stored as 16-bit values, and that 24 bits should be adequate for intermediate values in the calculation of Equation 4.

4.4 Algorithms Considered

Selection of the proper algorithm to search for the minimum solution is central to making the most effective use of processor and memory resources. Several different algorithms were explored to determine if sufficient optimizations were possible to allow the SBP method to be implemented effectively in the FPGA.

4.4.1 Exhaustive Search

The brute-force exhaustive search algorithm, as used in the current post-processing solution, simply calculates the value of all possible solutions in the solution set, and then does an exhaustive search of the results to find the minimum value. This method suffers from the large number of results that do not contribute to finding the optimal solution. It also requires access to the whole table of pre-computed values. These limitations prohibit this method from being used for an efficient solution. However, because this method is guaranteed to always find the absolute minimum value despite the structure of the solution set, it is useful as a gold standard for comparing against the results of other tested methods.

4.4.1.1 Localized Exhaustive Search

This improvement to the Exhaustive Search relies on the fact that the final X-Y position is going to be near the sensor channel that had the highest response. This allows the area of the exhaustive search to be limited to 1/(8*8), plus a surrounding margin, resulting in approximately 1/32 the original search area. Although this is a significant reduction in the computational load, it does nothing to the memory space requirement and does not adequately reduce the bandwidth limitations present in the exhaustive search.

4.4.2 Structured Searches

Observation of the entire solution set for a particular event suggests another family of algorithms could be used to substantially reduce the processing load. A representative solution set for an event near the center of the sensor array is shown in Figure 29. This figure makes it apparent that some sort of structured search algorithm could be used.

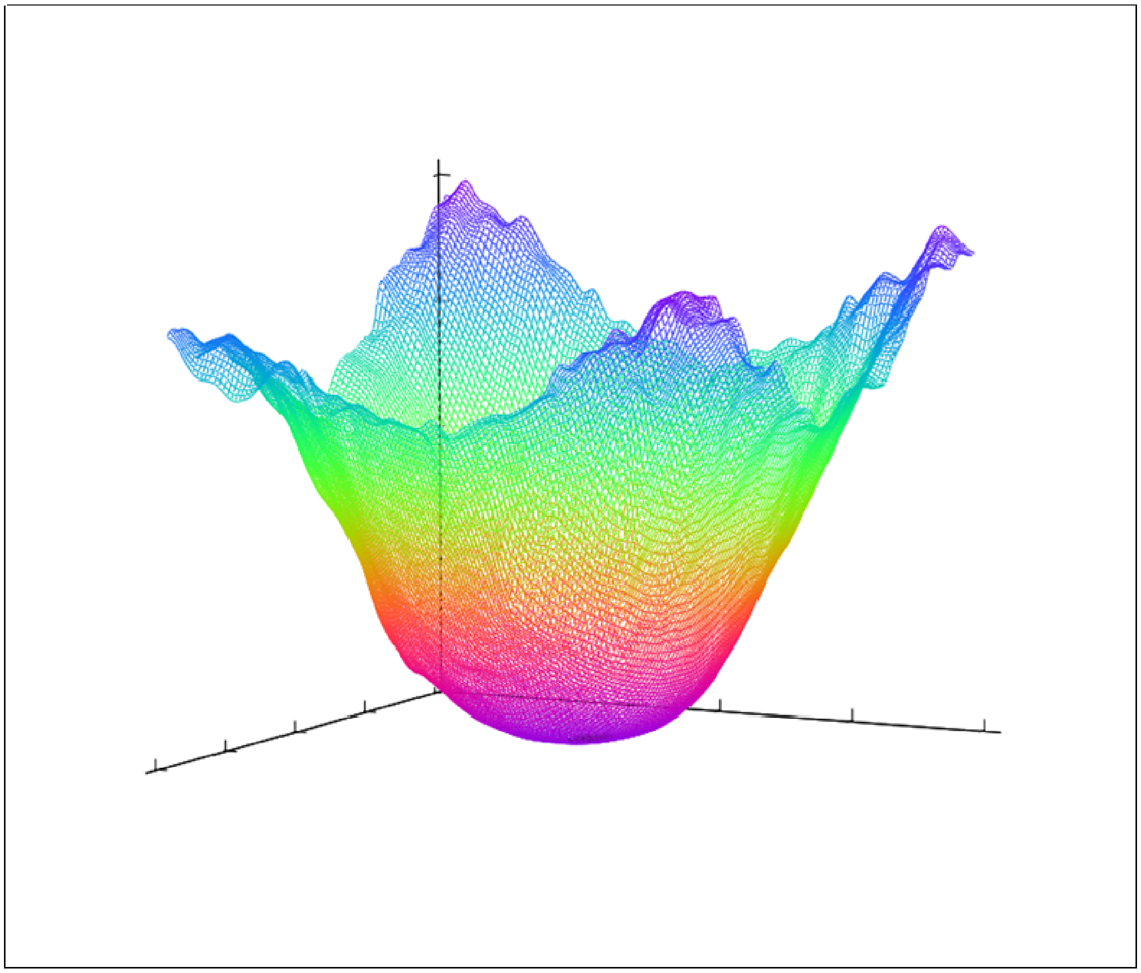


Figure 29. The complete solution set for a point near the center of the solution space.

4.4.2.1 Gradient Based Search

The first structured search algorithm tested relies on the gradient at a sampled point. In this search, the gradient at a point of interest is determined by sampling the value of adjacent points. If the gradient shows that adjacent points are not of equal values, the direction designated by the line connecting the maximum to the minimum value is taken as the direction toward the minimum overall value, as shown in Figure 30. In the tested algorithms, a 2-D search was implemented, with the X and Y components being determined simultaneously. In this algorithm, the two points shown in Figure 30 are replaced with from four to nine points to properly determine the 2-D direction of the gradient.

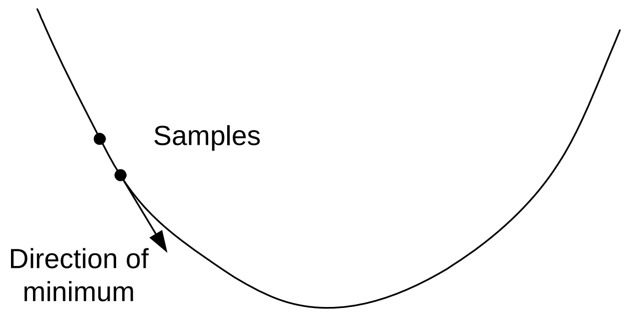


Figure 30. Gradient based search.

Once the direction of the gradient is determined, another sample is taken in the direction of the minimum and the process is repeated. There are variations on this algorithm ranging from a gradient "walk", where the next sampled point is the next adjacent point in the direction of the minimum, to a binary search where the first new sample is a large distance, with each new sample a progressively smaller distance. In either case, when the gradient is reduced to zero, the solution has been reached.

These algorithms have the potential to reduce the number of sampled points to a relatively small fraction of the total solution space. Gradient walk implementations reduced the search of a 127x127 solution set to approximately 150 sampled points, while gradient/binary searches reduced the sampled points to a minimum of 50.

All implementations of this method suffer from the tendency to be susceptible to local undulations in the curvature of the solution set, as can be observed along the periphery of Figure 29. This can cause some "hunting" for a solution, especially in the area of the minima of the solution set, where the values are very nearly equal in value. This requires the number of iterations to be increased from the theoretical minimum to insure that solutions were reached even for less than ideal solution sets. Experiments with representative data showed that this generally required the minimum number of sampled points to be increased to about 1.5 times the values mentioned above. The gradient-based algorithms also do nothing to reduce the memory requirements for storage of the data table.

4.4.2.2 Hierarchal Search

The next algorithm tried will be termed the hierarchal search as a notational shortcut. Although many searches, including the gradient/binary search described above, could be considered hierarchal, the use of this term in this context will be described below.

The hierarchal search starts with sampling points at multiple locations, uniformly scattered across the solution space. The sample with the lowest value is assumed to be closer to the final solution than any point with a higher value. The lowest valued sampled point is then used as the center of a new solution space that is somewhat smaller than the previous solution space. When the size of the solution space is reduced to one, the solution has been reached. Variations in this search can range from sampling a small number of points at each iteration, and continuing for a large number of iterations, to sampling a large number of points, requiring a smaller number of iterations to reach a solution.

This algorithm shows a higher level of improvement 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.

4.5 Algorithm Simulation

Since the hierarchal search showed the most improvement in the number of calculations required, and was easily adaptable to a pipelined architecture, it was chosen as the preferred solution for this project. The search described above was coded in "C" before committing to a hardware implementation. This allowed some initial testing of the hierarchal search to be explored as described below.

4.5.1 Search Accuracy

To test the accuracy of this hierarchal search algorithm, 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 31.

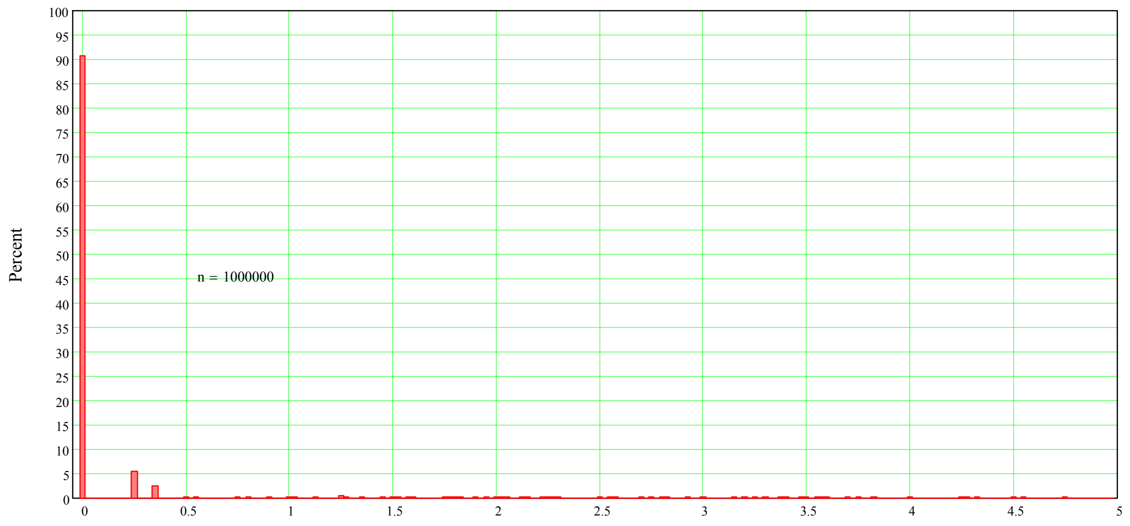


Figure 31. Distance error histogram (mm) between exhaustive and hierarchal searches.

The results show that over 90% of the events show no error and over 99% are within 1.4 pixels (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 in Figure 11. 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.

5 System Implementation

5.1 Development system

5.1.1 Development Hardware

Two levels of implementation are required for this project. Since the final hardware will not be completed for some time, a working prototype system was used to proof the hardware and design concept. The system used for implementing the prototype is the Altera DSP Development Board, Stratix II edition, Figure 32. This development board consists of the Altera EP2S60 with 2.5M-bits of internal RAM, 256k x 32-bit on-board SRAM and 4M x 64-bit SDRAM. Approximately 80 I/O pins are brought out on header pins and are available for interfacing to external circuitry.

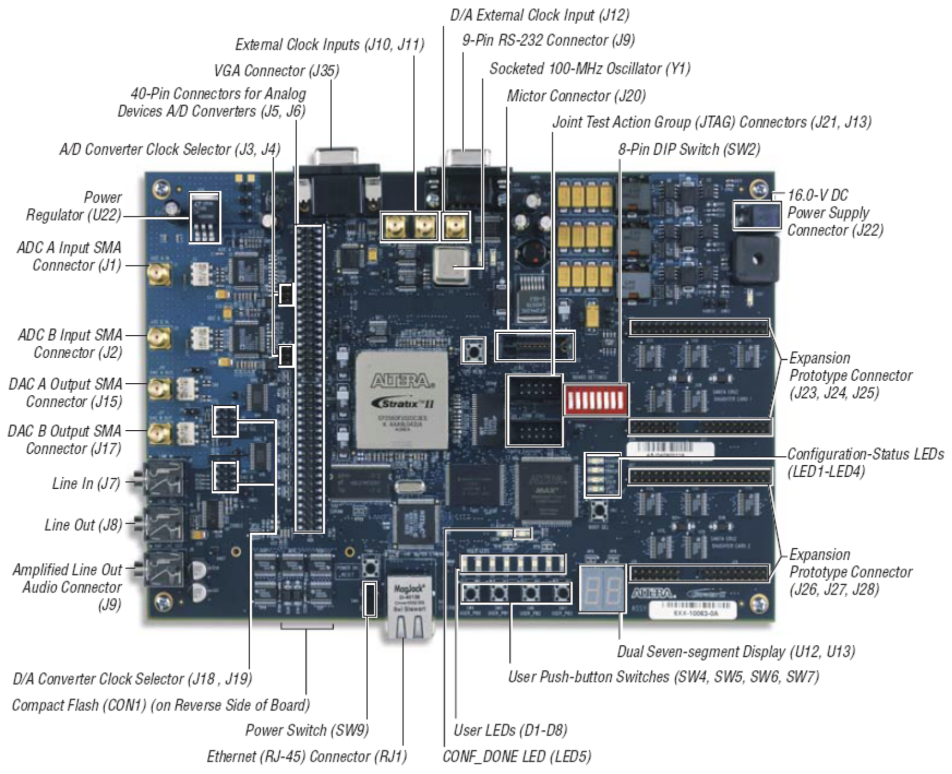

D/A External Clock Input (J12)
External Clock Inputs (J10, J11)
9-Pin RS-232 Connector (J9)
VGA Connector (J35)
Socketed 100-MHz Oscillator (Y1)
40-Pin Connectors for Analog Devices A/D Converters (J5, J6)
Mictor Connector (J20)
Joint Test Action Group (JTAG) Connectors (J21, J13)
A/D Converter Clock Selector (J3, J4)
8-Pin DIP Switch (SW2)
Power Regulator (U22)
16.0-V DC Power Supply Connector (J22)
ADC A Input SMA Connector (J1)
ADC B Input SMA Connector (J2)
Expansion Prototype Connector (J23, J24, J25)
DAC A Output SMA Connector (J15)
DAC B Output SMA Connector (J17)
Configuration-Status LEDs (LED1-LED4)
Line In (J7)
Line Out (J8)
Expansion Prototype Connector (J26, J27, J28)
Amplified Line Out Audio Connector (J9)
Dual Seven-segment Display (U12, U13)
User Push-button Switches (SW4, SW5, SW6, SW7)
D/A Converter Clock Selector (J18 , J19)
Compact Flash (CON1) (on Reverse Side of Board)
Power Switch (SW9)
User LEDs (D1-D8)
Ethernet (RJ-45) Connector (RJ1)
CONF_DONE LED (LED5)

Figure 32. Stratix II Development Kit.
(Image Courtesy Altera Corporation.)

This development system will allow all components of the SBP system to be tested, but will not allow a full implementation because of the following limitations:

- In the PET system, sensor data is converted to digital values with multiple high-speed analog to digital converters (ADC). This hardware is not available in the prototype system, so pre-collected data will be transferred to the prototype board via a serial communication protocol.
- The prototype has a smaller internal memory space than will be available on the target system. This will be accommodated by limiting the number of stages present in the development system. A larger system will allow including additional stages.
- The external memory on the development system is configured differently than what will be present on the final system. This will be accommodated by using a different memory driver. The data presented to the SPB algorithm will be identical to that available in the final system.

5.1.2 Development Software

The development software available for the Altera system is known as Quartus. This is an integrated development environment that includes a block diagram design interface, an editor that is aware of the syntax of the Hardware Description Language (HDL) (i.e., an HDL-aware editor), and multiple levels of simulation for validating logical design and insuring all timing constraints are met.

5.1.3 Method for Implementation in FPGA

One of the primary concerns for this implementation of SBP is access to the data tables. For the most efficient design, proper consideration must be given to the binary nature of the FPGA. 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. The reasons for these choices will be detailed below.

An example of this search on a 63x63 solution space is shown in Figure 33. 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.

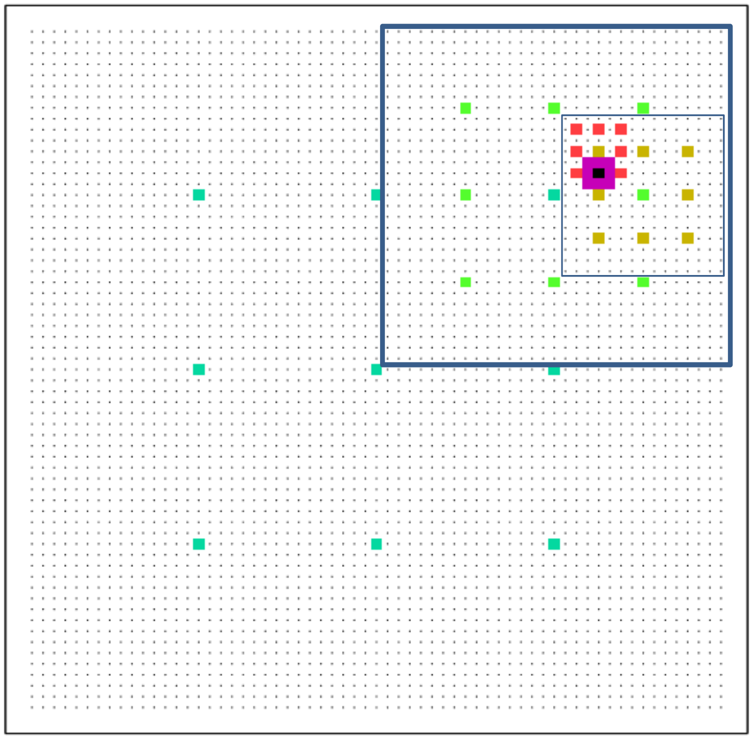


Figure 33. Five iterations of a 3x3 search on a 63x63 solution space.

The regularity of this search allows each level of the search to consist of an almost identical process, where each iteration consists of 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. 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 at each iteration to be independent of calculations currently underway in the other iterations. This can allow calculations for different events to be underway at each of the stages simultaneously, a process known as pipelining.

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.

This project's 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 34. 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.

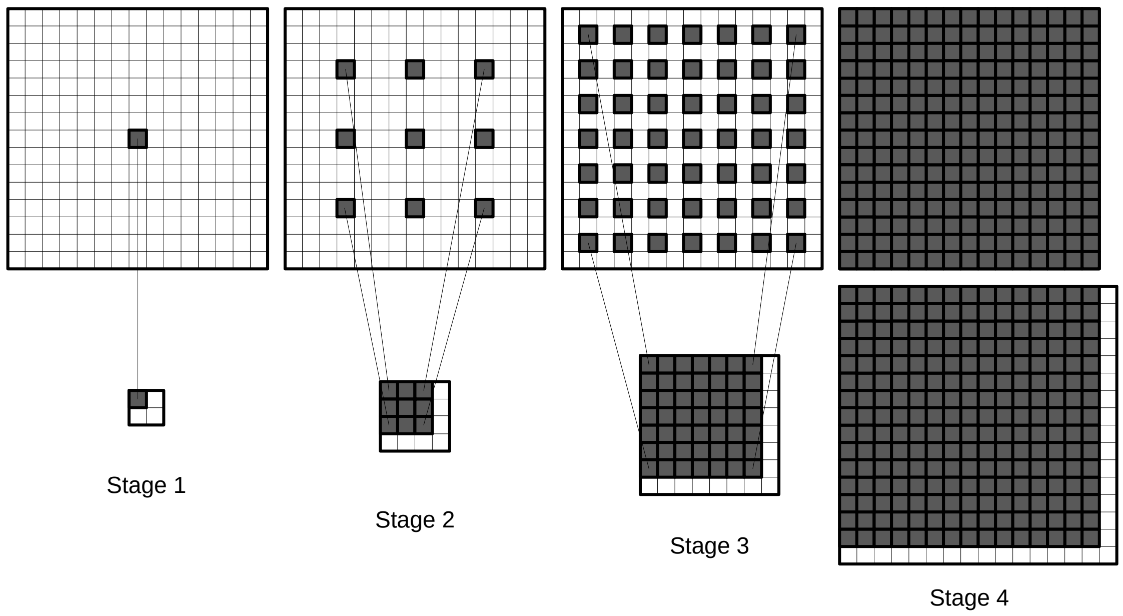


Figure 34. Compressed storage for data tables.

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 35. This is done by sequentially adding then subtracting one from the values of the center X and Y positions passed from the previous stage.

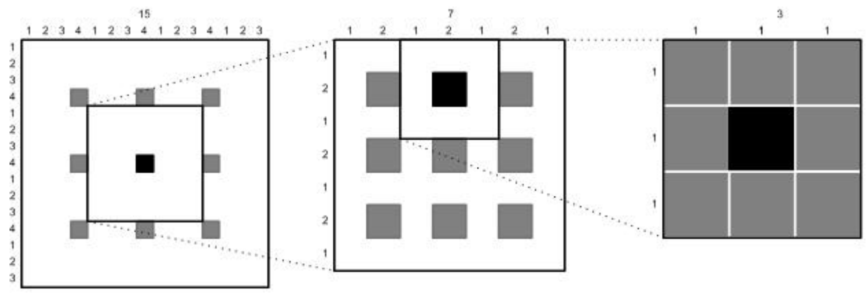


Figure 35. Diagram of the last three iterations of a hierarchal search.

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

To ease future modification to this system, both the sequence of row/column offsets to perform the 21-cell solution, and the sequence of X-Y offsets were done in a lookup table. The bit patterns for the required control signals for each clock cycle of one complete calculation are stored sequentially in an addressable memory on the FPGA. A counter circuit cycles through the addresses in this memory, one location per each clock cycle. As a location is addressed, values representing the row and column offsets are added to the row and column value of the peak energy channel to determine which of the 64 table entries to sum at each of the tested X-Y locations. After each X-Y location is calculated, the X-Y position is advanced to the next one to be tested and the 21-cell solution is repeated. As each of the 9 X-Y locations is tested, they are compared against previous calculations and the minimum result is identified. At the completion of the nine tests, control signals from this lookup table transfer the required information to the next stage and the cycle repeats.

5.2 Prototype Implementation

A block-diagram of the system design is shown in Figure 36.

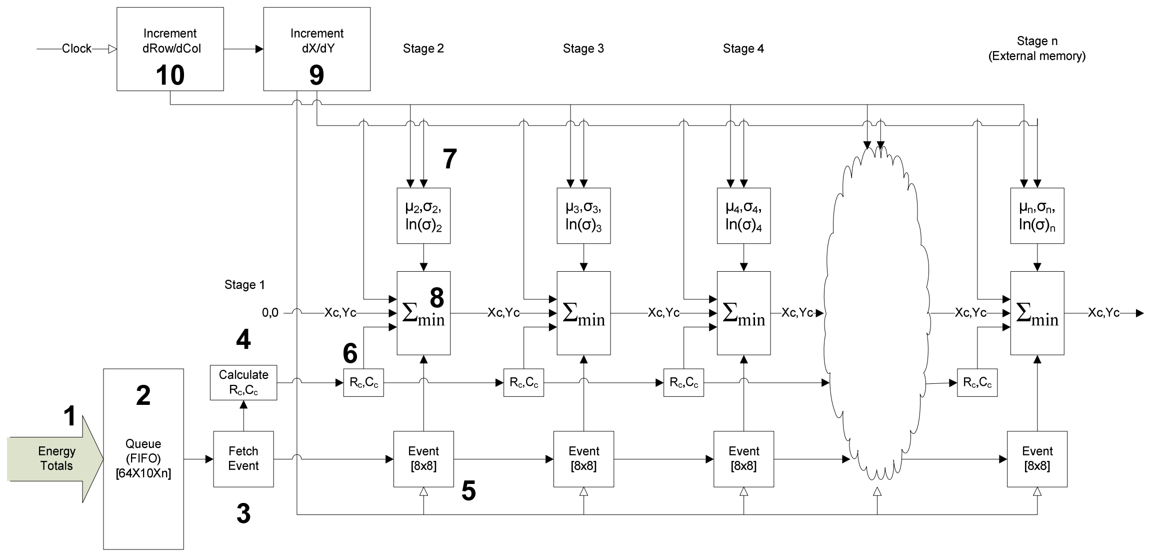


Figure 36. Block Diagram of FPGA implementation of SBP.

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 much 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 the 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 completes 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/col and last row/col 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 for the fact 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.

Because the memory available in the prototype is not sufficient for a seven stage system, only four stages could be created and tested. The results of this system were compared, on an operation-by-operation basis with the results of an integer-based simulation written in "C". The prototype was found to properly compute X_c and Y_c locations at each stage and transfer this information to the next stage. Each stage requires 21*9+3=192 clock cycles. This represents a 21-cell solution for each of 9 X-Y locations at each stage, plus 3 cycles for final calculation of the coordinates of the minimum value and transferring the results to the next stage. Although each event will take 192 clocks at each of seven stages, for a total of 1344 clock cycles, as each stage completes its calculation it is immediately available for the next sequential event. This results in 7 events being solved simultaneously, with each event at a different stage of its solution, a process known as pipelining. This allows the system throughput to be based on the 192 clock-cycle count, well below the 300 clock-cycle target.

5.3 Performance

5.3.1 Accuracy

In order to determine the overall accuracy of a seven stage system, the four stage “C” simulation described above was extended to seven stages and the results compared with the original floating-point results. A “Full-Width” comparison of floating-point and fixed-point results for a representative data-set is shown in Figure 37, showing a negligible difference between the two methods. A summary of the results for fifteen available datasets are in Table 1, showing the difference in the FWHM between the floating-point reference calculation and the results from the optimized fixed-point calculations. Data collected near the edge of the detectors, dataset 1 and 2, showed the largest errors and the largest differences from the reference floating-point calculations. This represents the difficulty that both methods have as the solution approaches the edge of the detector. Points further from the edge showed little to no differences.

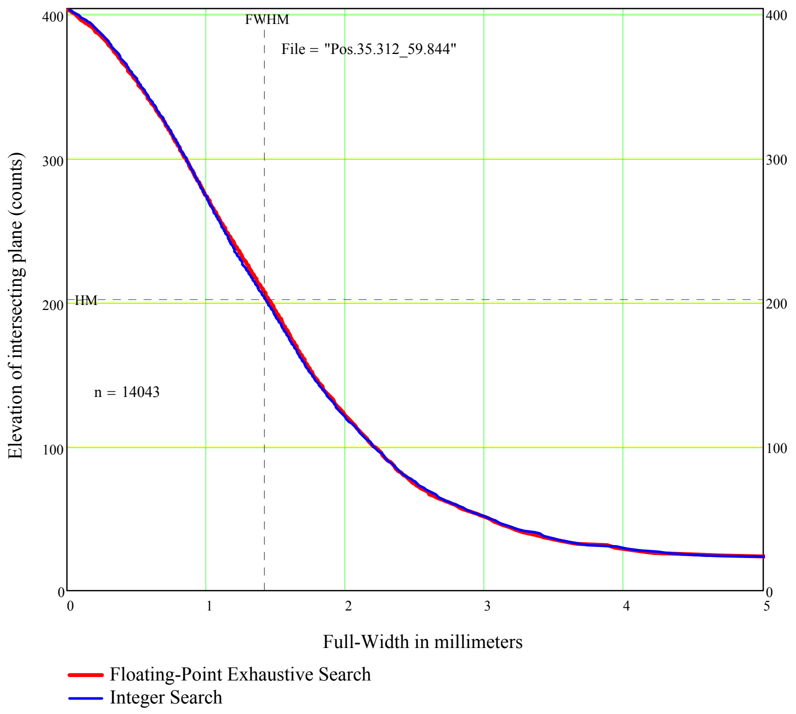
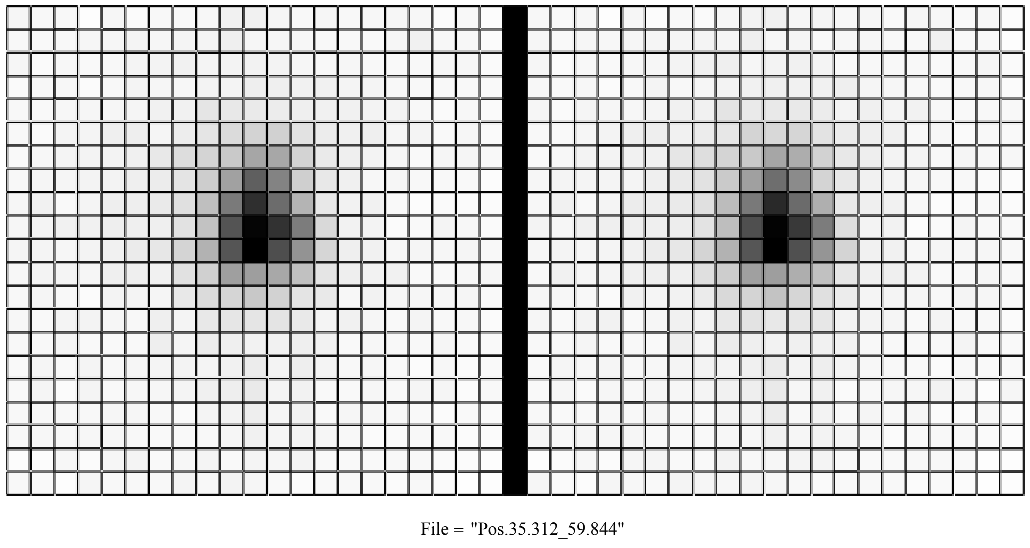


Figure 37. Full-width profiles (with FWHM) of floating-point and fixed point results.

Table 1. Comparison of reference and optimized calculations FWHM.

		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 final test of the quality of the results is a representation of the image produced by this data-set shown in Figure 38. This is a greatly magnified version of the resulting point image, showing the individual pixels. The slight variation between the full-width profiles shown in Figure 37 at around 2-3 pixels from the center are virtually unobservable in this image. Other data-sets showed similar results.



h

Figure 38. Resulting image for floating-point (left) and fixed-point (right) calculations. Full-width profiles shown in Figure 37.

5.3.2 Throughput and Latency

All timing results assume an FPGA clock frequency of 70MHz. This frequency is dictated by other components of the system.

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 being begun, and one completed, 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.

5.4 FPGA Resources Used

The constraint that proved to be most limiting was the use of FPGA internal memory for storage of the data tables. As shown in Table 2, the number of stages that can fit in internal FPGA memory depends on the size of the device model used. For the maximum values, the figures are those reported by the Quartus development software for that device. Values for fewer stages are extrapolated based on the relative size of each stage. Although the SL340 has memory available for up to six stages, memory partitioning by the Quartus compiler required that the tables be broken down into an array of smaller segments to allow the tables to be efficiently mapped into the internal memory blocks. For both values marked with an asterisk (*), memory usage by other functions implemented on this FPGA may not allow this configuration.

Table 2. FPGA internal memory usage.

[redacted]

(* see text)

The FPGA logic components used by this design turned out to be only a minor factor. Table 3 shows that less than 10% of the available logic is used in all but the largest systems (i.e., largest number of search stages) on the smallest processor.

Table 3. FPGA logic used.

[redacted]

6 Conclusion

It has been shown that the SBP algorithm can greatly increase the resolution of the PET data collection process, and can do so on a much easier to produce and less expensive continuous scintillation crystal. The disadvantage of this solution has been the greatly increased times between data collection and presentation of the finished image because of the computational complexity of this method. 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 SPB 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 et al., “cMiCE:a high resolution animal PET using continuous LSO with a statistics based positioning scheme,” *Nuclear Science Symposium Conference Record, 2001 IEEE*, 2001, pp. 1137-1141 vol.2

[2] Kisung Lee et al., “Detector characteristics of the micro crystal element scanner (MiCES),” *Nuclear Science, IEEE Transactions on*, vol. 52, 2005, pp. 1428-1433.

[3] R. Miyaoka et al., “Toward the development of a microcrystal element scanner (MiCES): quickPET II,” *Nuclear Science Symposium Conference Record, 2003 IEEE*, 2003, pp. 2242-2246 Vol.4