

Space Time Adaptive Processing Estimates for IBM/Sony/Toshiba Cell Broadband Engine Processor

Mr. Luke Cico (lcico@mc.com)

Mr. Jon Greene (jgreene@mc.com)

Mercury Computer Systems

199 Riverneck Road, Chelmsford, MA 01824

+1 978 256 0052

Abstract

Advances in Commercial Off-The-Shelf (COTS) embedded computing technologies have yielded impressive gains in computational throughput over the past 5-10 years. Adaptive sensor array systems utilizing real-time teraflop class machines are in wide deployment today. Gains in processor density have generally been achieved by steady improvements in semiconductor optical lithographic processes along with less frequent innovations in processor chip architectures. It is likely that the real-time embedded community is entering an era where processor architectural innovations will be bearing most of the burden for producing processing density gains as lithographic processes approach fundamental physical limitations. More and more 'systems on a chip' are emerging to address these trends. An exciting example of this technology trend is IBM's new Cell Broadband Engine (CBE) architecture which offers massive SIMD compute power on multiple computational units interconnected via a high bandwidth internal fabric. This paper explores the application of a computationally intensive adaptive nulling problem on the CBE architecture.

The reference application is the RT_STAP benchmark developed by MITRE Corporation and serves as a fair representation of the processing requirements of an adaptive nulling problem for a modern radar system. The benchmark represents a processing mode with 22 spatial channels sampled at 5MHz and performs 3rd order post doppler adaptive nulling along with pulse compression and doppler filtering operations. Total computational throughput requirements for this mode are 39 GFLOPS based on a processing interval of 32.5 milliseconds. Expectations based on this initial analysis are that computational throughput gains of 10X are possible and, furthermore, that throughput per watt will improve by several factors over what is achieved with the current generation of high performance floating point general purpose programmable processors.

1. Space-Time Adaptive Processing (STAP) Computational Requirements

This paper will explore the application of a particular class of radar algorithm primitives on the IBM/Sony/Toshiba Cell Broadband Engine (CBE) processor. The primitives form the core mathematical procedures used to perform adaptive cancellation techniques in phased array sensor systems. These techniques are employed to dynamically adjust a radar sensor's antenna pattern response to greatly reduce the power of interfering signal sources. These signal sources can include either signals intentionally introduced into the environment (Jammers) or unintentional ground clutter returns on airborne radar platforms.

This class of processing is referred to as Space Time Adaptive Processing (STAP) techniques, given their ability to adaptively filter spatially and temporally correlated interference. If these interfering sources are not mitigated, the received signal power from these sources can greatly reduce the ability to detect targets of interest. The processing is performed in high performance digital computers requiring 10s to 100s of GFLOPS (billions of Floating Point Operations per Second) of computational throughput. In order to satisfy the real-time requirements of the radar system and the latency requirements of the back end tracking processes, the radar signal processor must typically perform all the computations in time frames ranging from single digit milliseconds to several tens of milliseconds.

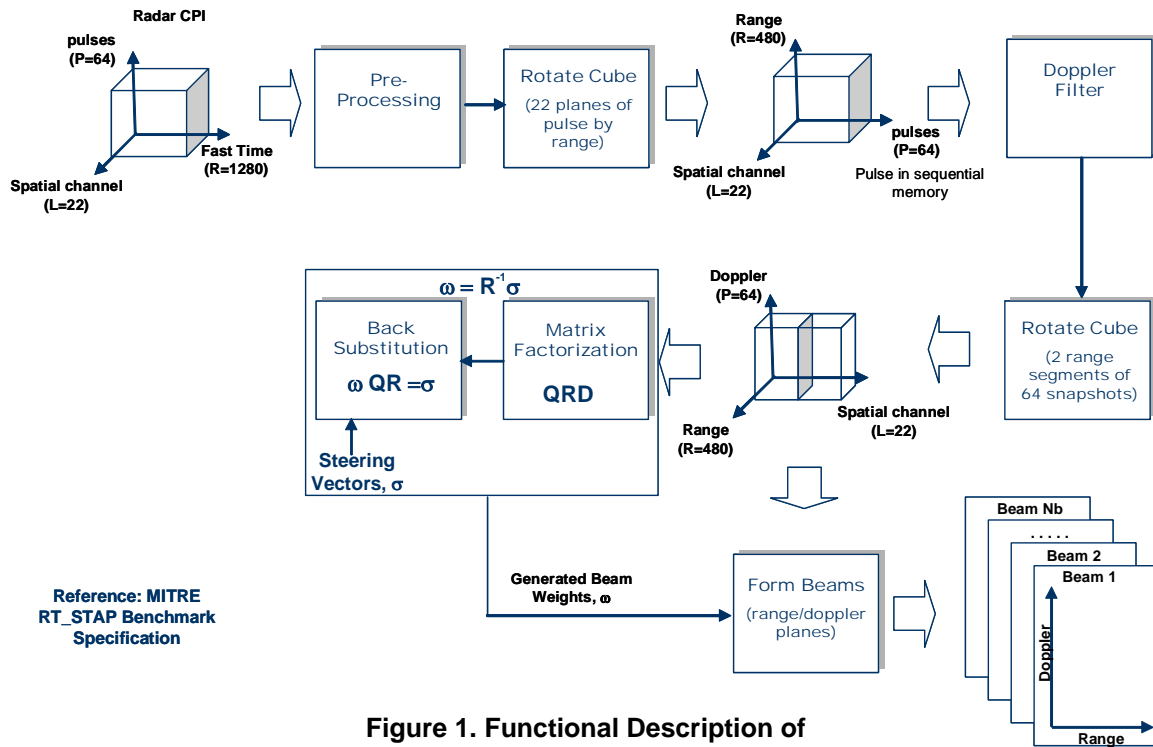


Figure 1. Functional Description of STAP processing stages.

A functional description of the processing steps of a representative STAP mode is shown in figure 1 and consists of the computational stages described below.

Preprocessing:

- Demodulation of the received antenna signals to base-band and formation of quadrature signal components,
- Down-sampling of the input signal after the application of anti-aliasing filters (FIR),
- Correlation of the received radar pulses with a replica of the transmitted waveform in order to increase the Signal To Noise Ratio (SNR) to maximize the likelihood of echo detection. This step will also increase the resolution of the radar’s range measurement.

Doppler Filtering:

- Doppler filtering to measure a target’s velocity with respect to the radar.

Weight Computation:

- Computation of the beam weights to apply to the received antenna signals in order to spatially and temporally filter unwanted signals. A separate weight solution is calculated for each doppler bin and for several blocks of non overlapping range regions,

Beamforming:

- Application of the computed weights on the received antenna pulses in order to spatially filter signals arriving from directions not in the direction of the target of interest.

The sensor data arrives in the digital processor as a sequence of time sampled pulses from each of the receive elements in the antenna array. A pulse corresponds to a single transmit and receive interval of operation of the radar. A number of pulses are collected to serve as the temporal samples for the doppler spectral filtering operation. The total data set received and to be processed therefore corresponds to a 3 dimensional set of data consisting of spatial channel (antenna), radar pulse and, finally, time sample in a pulse. The orientation of the data set in computer memory is determined by the natural order of the collected ADC samples from each receive channel. Time samples are collected in sequential memory positions, followed by pulses followed then by channel.

Each stage requires, at least conceptually if not actually, a realignment of the data cube to position the data samples in memory for the next stage of processing. For example, the

pre-processing stage operates on the sequential time samples in each pulse (forming range samples at the output of the process) while the next stage, doppler filtering, will operate along the pulse dimension (forming doppler frequency bins at the output of the process). Mathematically, these operations are transposes or rotations of the 3 dimensional object as it progresses through the processing stages.

The front end receiver oscillators and digital converters must maintain phase coherency from pulse to pulse over an entire data collection interval (Coherent Processing Interval or CPI). This absolute phase coherency requirement is necessary in order to perform the doppler processing. Channel to channel phase and amplitude imbalances are corrected with a calibration process in each spatial channel in order to eliminate errors in the formation of the beam patterns in the beamforming process.

Pre-Processing Primitive Definition and FLOP count per Coherent Processing Interval (CPI)

The pre-processing stage operates on the time samples received in each pulse. Each pulse is processed independently for each channel.

The processing primitives and floating point operation (FLOP) counts for each function in the pre-processing stage are:

- **Video IQ formation:** A real to real vector multiply for the in-phase component and another real to real vector multiply for the quadrature-phase component. This operation requires a total of $\mathbf{P}*\mathbf{L}*(2*\mathbf{T})$ floating point operations for all pulses (\mathbf{P}), spatial channels (\mathbf{L}) and un-decimated time samples (\mathbf{T}) in each pulse.
- **Down-sampling:** A real value FIR filter applied to the in-phase component and the same real-value FIR filter applied to the quadrature-component of the time samples. The number of floating point operations required for this function is $2*\mathbf{P}*\mathbf{L}*\mathbf{R}*(2*\mathbf{N}_{fr}-1)$ for all pulses (\mathbf{P}), all channels (\mathbf{L}) and all decimated filter output samples (\mathbf{R}). The number of filter taps is \mathbf{N}_{fr} .
- **Pulse Compression:** Each pulse is segmented into three equal length regions on which an FFT overlap and save operation is performed. Each region is zero padded to 256 points. The zero padded sequences are processed through a 256 point radix-2 FFT, a 256 point complex vector multiplication and a 256 point radix-2 inverse FFT. The number of floating point operations required is $3*\mathbf{P}*\mathbf{L}*\mathbf{N}_{fr}*(10\log_2(\mathbf{N}_{fr})+6)$ for all pulses (\mathbf{P}) and all channels (\mathbf{L}) and $\mathbf{N}_{fr}=256$ samples.

Doppler Filtering Primitive Definition and FLOP count per Coherent Processing Interval (CPI)

Doppler processing consists of weighted DFTs along the pulse dimension of the cube. Prior to the FFT computation a weighting function (real valued) is applied to the input samples to control spectral leakage of the FFT operation. The constituent operations are, therefore, two real vector multiplications (windowing) and a \mathbf{P} -sample radix-2 FFT. The total number of floating point operations for this stage is $\mathbf{L}*\mathbf{R}*\mathbf{P}*(5\log_2(\mathbf{P})+2)$ for all channels (\mathbf{L}) and all range samples (\mathbf{R}).

Adaptive Weight Computation and FLOP count per Coherent Processing Interval (CPI)

The computation of the adaptive weights is the most numerically intensive algorithm in this STAP mode. A full treatment of STAP techniques can be found in references [1] and [2] and is not repeated here.

The solution of the optimal set of beam weights to perform cancellation of the interference is computed by solving

$$\mathbf{C} \boldsymbol{\omega} = \boldsymbol{\sigma} \quad [1]$$

for the beam-weight vectors, $\boldsymbol{\omega}$,

$$\boldsymbol{\omega} = \mathbf{C}^{-1}\boldsymbol{\sigma}$$

given the space-time steering vectors, $\boldsymbol{\sigma}$, and the space-time covariance matrix, \mathbf{C} . In practice, the interference covariance matrix \mathbf{C} is unknown and must be estimated by computing a sample of the matrix over a number of independent observations of the space-time samples,

$$\mathbf{C} \sim 1/M \mathbf{X}^H \mathbf{X} \quad [2]$$

where \mathbf{X} is the observed data samples used to form the statistics of the interference. The complex conjugate transpose (Hermetian) of this matrix is denoted by \mathbf{X}^H .

Rather than inverting the covariance matrix, numerical techniques can be used to reduce the above expression to a form suitable for computationally simpler methods. In particular, the data matrix can be factored into two matrices

$$\mathbf{X} = \mathbf{Q}\mathbf{R} \quad [3]$$

where \mathbf{Q} is a unitary matrix and \mathbf{R} is an upper triangular matrix. Combining equations 1 and 2 produces,

$$\mathbf{X}^H \mathbf{X} \boldsymbol{\omega} = \mathbf{M}\boldsymbol{\sigma}$$

$$(\mathbf{R}^H \mathbf{Q}^H) (\mathbf{Q} \mathbf{R}) \boldsymbol{\omega} = \mathbf{M}\boldsymbol{\sigma}.$$

By noting that the product of the unitary matrix \mathbf{Q} by its conjugate transpose (Hermetian operator) is equal to the identity matrix,

$$\mathbf{Q} \mathbf{Q}^H = \mathbf{I}$$

produces the final result,

$$\mathbf{R}^H \mathbf{R} \boldsymbol{\omega} = \mathbf{M}\boldsymbol{\sigma}. \quad [4]$$

A Modified Gram Schmidt QR factorization routine is used to compute the upper triangular matrix, \mathbf{R} .

The resulting matrix is used in equation [4] to solve for the weight vector, ω , using back and forward substitutions.

The total number of floating point operations required for the QR operation is $K * M * (20 * L^3 * Q^3)$, where $K=64$ is the total number of doppler bins, $M=2$ is the number of range training intervals, $L=22$ is the number of channels and $Q=3$ is number of temporal samples used.

Beamforming Primitive Definition and flop count per Coherent Processing Interval (CPI)

The weights computed for each doppler/range interval are applied to the data set. The operation is a complex matrix multiplication of the space-time samples in each range/doppler bin by the optimized beam weights solved in equation [1].

The total number of floating point operations required for this operation is $K * R * (8 * L * Q)$, where $K=64$ is the total number of Doppler bins, $R=480$ is the number of range training intervals, $L=22$ is the number of channels and $Q=3$ is number of temporal samples used.

2. CBE Architecture Overview

The CBE processor developed by IBM, Sony and Toshiba is, at the conceptual level, a heterogeneous multi-computer on a single chip. The functional units of the chip as shown in figure 2 consist of:

- **PPE:** A general purpose PowerPC processor unit (PPU) with 32K of instruction and 32K of data L1 cache, an on-die 512K L2 cache and an MMU. The PowerPC has a 128-bit SIMD Vector Multimedia Extension unit.

- **SPE:** Synergistic Processing Elements are the main computational engines of the chip. There are a total of 8 of these units per chip. Each unit consists of a 128-bit SIMD (Single Instruction, Multiple Data) engine (termed the SPU), a Memory Flow Controller (MFC) containing DMA engines and interface logic to the internal ring, and a Memory Management Unit (MMU). Each SPU has 256K of local storage memory (LS) which is partitioned by the application for use as storage for both user data buffers and instructions (primitives and user text).
- **XDR Controller:** A memory controller to external high performance XDR memory is included on-die.
- **Coherent Interface:** A high performance coherent interconnect for communication with another CBE processor for a dual SMP node configuration.
- **I/O Interface:** An interface to a South Bridge device for external chip IO, bulk memory and peripheral device interconnect.
- **EIB:** Element Interconnect Bus is a high-bandwidth ring that connects all the functional units on chip. The ring is capable of moving up to 96 bytes per cycle.

This analysis is based on the assumption that the CBE is operating at a frequency of 3 GHz. The EIB bus runs at half the 3GHz core rate. At this operating point of 3GHz, the aggregate peak computational (maximum theoretical) throughput of all 8 SPU SIMD engines is **192 GFLOPS** based on a single cycle multiply accumulate. The chip manufacturers have stated that higher clock frequencies are possible but 3GHz would appear to be the likely operating point for a CBE in a deployed defense application.

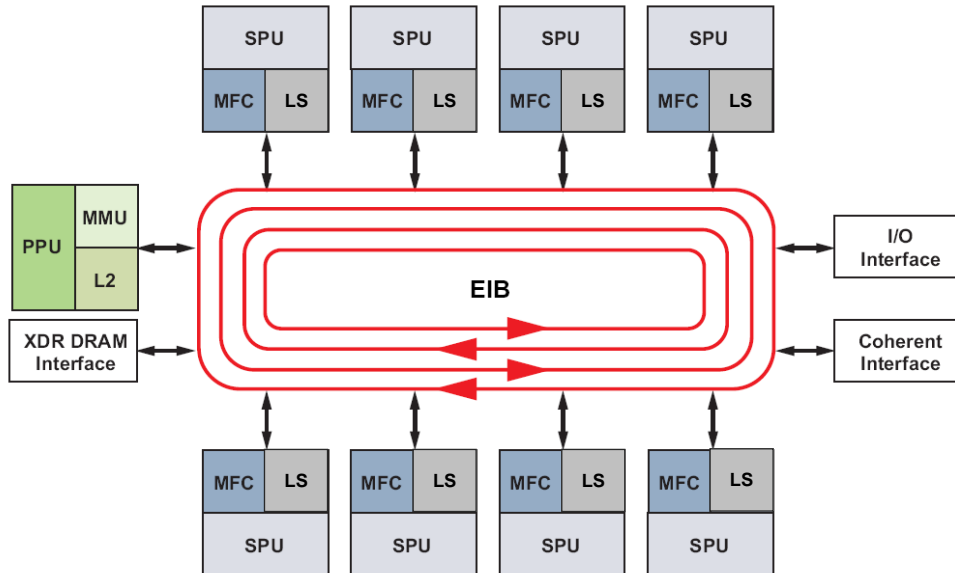


Figure 2 - Cell Architecture overview highlighting PowerPC processor, 8 SIMD engines, XDR memory and external interfaces. All components connected via high bandwidth EIB ring.

XDR memory bandwidth is stated to be 25.6 GB/s peak theoretical with an expected average sustained rate of approximately 19 GB/s for load and store operations between SPU Local Store (LS) and XDR Main Store (MS). Although this is considerable main memory bandwidth performance, it must be noted that this memory is shared by all 8 SPU SIMD engines, PPU processor and the IO interfaces. For SPU engine accesses alone, the ratio of peak SIMD GFLOPS to MS memory bandwidth is 192 GFLOPS to 25.6 GB/s (peak in both cases). Application designers must pay careful attention to this architectural fact, especially for IO or border-line IO bound algorithms, as the memory bandwidth may be a limiting factor to the sustainable performance of the chip.

3. Data Flow Analysis

This section will describe the management of the data flow between the SPU Local Storage and XDR main storage in support of the computational operations described in the preceding sections.

The management of the data flow must work within the constraints imposed by the memory hierarchy of the chip and the SPU DMA controller capabilities. The resulting guidelines for the application designer are:

- The size of the Local Store is 256K and must contain all input buffers, output buffers, temporary buffers, coefficient buffers, DMA descriptors, user text (code) and code for primitives. The working assumption is no more than 175-200 KB can be reserved for data. The rest is for executable code.
- The SPU DMA controllers support a list capability. The list capability can be used to read or write (via DMA) data stored sequentially in LS memory to non-sequential positions in XDR MS memory.
- Applications need to be designed for optimal data locality and data reuse in the LS memory to minimize trips back and forth to XDR MS.
- Maximum SPU IO bandwidth is achieved with DMA transfer sizes of 128 bytes. A further requirement for achieving maximum bandwidth is that the relative offsets between the local address and the remote address be the same. Non-equal relative offsets between the local and remote memory addresses will result in degraded bandwidth performance.

For the first stage (Pre-Processing) the data cube is properly oriented in MS for access by the SPU as received by the ADCs. The data set is partitioned such that each SPU processes a unique set of 8 pulses for all channels. The individual data vectors are relatively small, consisting of $T=1920$ 32-bit real input samples (4

bytes per) and $R=480$ 64-bit complex output samples and can be easily accommodated in the local store memory.

The demodulation process requires a unique set of complex coefficient samples for each pulse processed. These coefficients correspond to samples of a complex sinusoidal mixer at the IF processing frequency of the radar and are coherent from pulse to pulse. Although the footprint in LS of each set of coefficients doesn't add significantly to the total LS buffer requirements, the processing order of data sets must be structured to decrease the amount of reads to XDR for the coefficients. The correct way to handle this is for each SPU to access pulse sets across the channel dimension of the cube. In other words, process the same pulse ID for all channels before processing the next pulse. By partitioning and distributing by pulse to SPUs and accessing pulses across channel, the demodulation coefficients need only be read once from XDR. Of course an alternative strategy to loading the coefficients would be to generate them on the SPUs when needed, or to maintain a single set of coefficients and adjust the phase from pulse to pulse.

After range processing (Pre-Processing), the data is written back to XDR memory as range sample by pulse by channel. After down-sampling and pulse compression each pulse consists of 480 range samples. We know that in the next processing stage, Doppler Filtering, the direction of processing will be along the pulse dimension of the data cube. Furthermore, thinking ahead to this next stage of processing, we will partition by range sample and process a P-pulse by $(R/8)$ -range swath of samples in each channel in each SPU. In order to maintain DMA alignment and optimal transfer size requirements, the Pre-Processing stage will partition the $R=480$ range samples into eight swaths of $R/8 = 60$ range samples with a pad of 4 samples added to each range segment to maintain DMA alignment. The result of this padding is that the DMA transfer operations for every stage after pulse compression loses an additional 6.25% efficiency in bus performance.

At the completion of Doppler processing each SPU will stage the processed data back into XDR memory in preparation for re-partitioning and re-assignment of data segments for the adaptive weight computation and weight application. Again, the order in which the data is written back to XDR is determined by careful consideration of the necessary ordering requirements for efficient computation of the QR algorithm. This is done because the local store memory on each SPU is not sufficient in size to allow any local transposition of the rather large QR input data matrix. It is simply best to properly align the data at the output of a

Estimate Summary			
	Throughput (MFLOPS)	stage time as % of allowed exe time	Ratio of compute time to IO Time
Pre-Processing (video I/Q, pulse compression)	83,737	7.2%	2.73
Doppler Filtering and data reorganization	32,938	2.0%	0.46
Adaptive Weight Computation	106,913	31.2%	7.53
Adaptive Weight Application	11,139	4.5%	0.22

Figure 3 – Summary of performance estimates of STAP processing kernels on the IBM/Sony/Toshiba CBE processor

completed stage before loading the data for the next stage. As it turns out this is rather easy to do with the link list capability of the SPU DMA controller by writing back successive range blocks with address offsets in XDR that align the data in range-channel-doppler order (ie., range sample varies fastest and doppler sample varies slowest). Once again, though, the range segments are broken into eight swaths of 60 samples each with 4 samples of padding.

The QR work load is partitioned by doppler bin among the eight SPU units. In particular each SPU processes 8 contiguous doppler bins. Each SPU therefore computes a total of 16 QR solutions; one for each of the 8 doppler bins assigned in each of the 2 non-overlapping range intervals over which the training windows have been defined.

4. Computational Performance Estimates

This report has defined a mapping of a STAP algorithm to the CBE architecture. Performance modeling was performed by analyzing the data flow patterns required between the SPU local store memory and XDR main memory. The computational algorithms defined in section 1 were then modeled and estimates formed of the expected computational throughput for each stage of processing, as well as the memory loading on the XDR memory by the contending SPU SIMD engines. The estimates of the computational efficiency of the various kernels was formed by analyzing the CBE chip design documents and experience at Mercury in programming of a variety of SIMD engines including SPUs.

The estimated results for each stage are shown in figure 3. One can observe the following performance characteristics:

- The throughput of the most computationally intensive parts of the processing, in particular QR decomposition and Pre-Processing is quite promising with an estimated delivered throughput of 107 and 84 GFLOPS, respectively.
- Functions with a low ratio of floating point computations per sample of IO were not particularly efficient and yielded between 11 and 33 GFLOPS of performance.
- Overall the estimates predict that the throughput of the chip could be about 90GFLOPS. Since the RT_STAP benchmark that was used in the modeling example only requires 39GFLOPS of sustained performance the chip is less than 50% loaded.

Further analysis is being conducted to substantiate these results with measured data on CBE hardware.

References

[1] **Space Time Adaptive Processing for Airborne Radar**, J. Ward, Technical Report 1015, 13 December 1994, Massachusetts Institute of Technology Lincoln Laboratories, Lexington, MA
 [2] **RT_STAP Benchmark Report**, MTR96b21_rstap, MITRE Corporation, Bedford, MA
 [3] **Space Time Adaptive Processing Principles and Applications**, R. Klemm, Institution of Electrical Engineers Press
 [4] **IBM CBE Documents**, IBM Corporation, <http://www.306.ibm.com/chips/techlib/techlib.nsf/products/cell>, USA 2005