

# Beamforming for Antenna Arrays

## BEE2 vs DSP Processors

Vinayak Nagpal, Terry Filiba

May 11, 2007

### Abstract

We present an architecture for highly scalable and reusable time domain beam formers using general-purpose FPGA boards. We also present a comparable architecture for the same using DSP processors. The FPGA design is based on the BEE2[2] platform. System generality and reusability is the focus of our design, which assumes very little about the target antenna array and avoids any sort of tight coupling between subsystems. Our FPGA prototype, which we call BEEformer is being deployed at the Allen Telescope Array (ATA)[8] in July 2007.

At the core of our system is a single all-purpose design, which becomes a recurring motif replicated on all FPGAs across the system. On a single BEE2 board we provide a phasing engine for signals having up to 200MHz bandwidth from 32 antennas while forming 3 simultaneous beams. Multiple boards are connected together to further scale the system arbitrarily in bandwidth, polarization channels, number of simultaneous beams and up to a maximum of 256 antennas, which too is only a soft limit. The connection topology can be modified for any desired configuration on the fly with software control. All communication between subsystems is asynchronous and uses hot-swappable high-speed serial links using industry standard protocols[7]. The globally asynchronous packet processing approach also obviates the need for frequency locked clock distribution.

In addition to the above, we have conceptually worked out an architecture for the same instrument using the TMS320C6455, a state of the art DSP processor[1]. DSP processors provide an attractive alternative to FPGAs due to the ease of programming and simulation. We evaluated the performance and data bandwidth of the chip to draw comparisons between the DSP vs. FPGA approaches.

## 1 Introduction

Large antenna arrays are being used for radio astronomy, positioning (Geodetic VLBI) as well as communication applications (e.g. NASA's Deep Space Network (DSN) array to communicate with distant space probes). Very often these arrays need to operate in phased array or beam forming mode. The size of arrays being built today are approaching hundreds of antennas (Allen Telescope Array (ATA), Atacama Large Millimeter Array (ALMA) etc.) and receiver bandwidths have scaled into the tens of GHz ranges (DSN, ATA etc.). Even though correlators are algorithmically far more complex than beamformers, the calibration requirements of beamforming make their design very challenging[6][3][4][5]. Beamformers of this scale haven't been built before. In this project, we have worked on a digital architecture for beamforming an arbitrary number of antennas with a large number of simultaneous beams. We have tried to include safeguards to simplify instrument calibration, but rigorous calibration of the instrument has not yet been addressed and is considered outside the scope of this work.

Phased array processors are usually classified into frequency domain and time domain with both approaches having their merits. Architecturally frequency domain beam formers are more robust; signals from all antennas are frequency channelized and any delay compensation is done by complex multiply operations to get desired phase shift in any bin. The system designer has more leeway in terms of not having to calibrate many instrumental delays because instrumental delay effects are easy to measure and correct in the frequency domain. The obvious disadvantage of frequency domain beam forming is seen when a time domain phased signal is needed at the output. Converting a frequency spectrum back to time samples is both computationally expensive and also re-introduces all the instrument delay calibration problems that had relaxed because of spectral processing. Time domain beam formers still remain popular despite the challenges involved in their design because most systems require a time domain signal output.

In section 2 we describe the system level architecture of BEEmformer in limited detail. In section 3 we describe a conceptual system equivalent to BEEmformer built using DSP processors. Though we haven't done extensive simulations to prove the correctness of our concept, we have drawn performance estimates analytically to compare with BEEmformer. The conclusions are discussed in section 4.

## 2 BEEmformer

### 2.1 Instrument specifications

The ATA requires its beamformer to be able to handle at least 100MHz chunks of bandwidth at a time. The demonstration prototype should support 30 antennas dual polarization and scale to 150 in the final instrument. We need 16 simultaneous beams in the final instrument and at least 3 in the prototype. The beamformer must be able to interface with the ATA IF processing engine which is based on iBOB and iADC boards. Each iBOB in the IF processor processes 4 antennas single polarization data. Sampling frequency is 800Msamples/sec and after digital down-conversion complex samples are available per antenna at the rate of 100MHz. The physical interface is through XAUI links with a maser locked 1 pulse per second out of band framing. System output should be available over XAUI links as well.

### 2.2 Top Level Architecture

Fig. 1 shows the system level architecture of BEEmformer. 1 BEE2 supports 32 antennas single polarization data from 8 iBOBs of the ATA IF processor. It can process 3 beams with support for additional beams by cascading. As shown in the figure the total input data rate per BEE2 is  $6.4\text{Gbps} \times 8 + 3 \times 3.2\text{Gbps} = 60.8\text{Gbps}$ .

#### 2.2.1 Communication between boards

Beamformers are very unforgiving of any data sample slips or undetermined timing variations in clocks or communication links. Any non-determinable delay could correspond to a  $\pi$  radian phase shift and cause destructive phasing, thereby severely damaging the system performance at that frequency. To ensure predictability, usually beamformers are designed with globally synchronous approaches with maser locked clock trees going to every board in the system. We however have tried to use a globally asynchronous but locally synchronous design paradigm, to allow for greater scaling. The iBOBs and iADC based IF processor are fed by a tree of maser locked clocks to make sure all samplers are frequency locked. The sampled and down-converted data are communicated to the BEEmformer over asynchronous serial links and the BEE2 boards

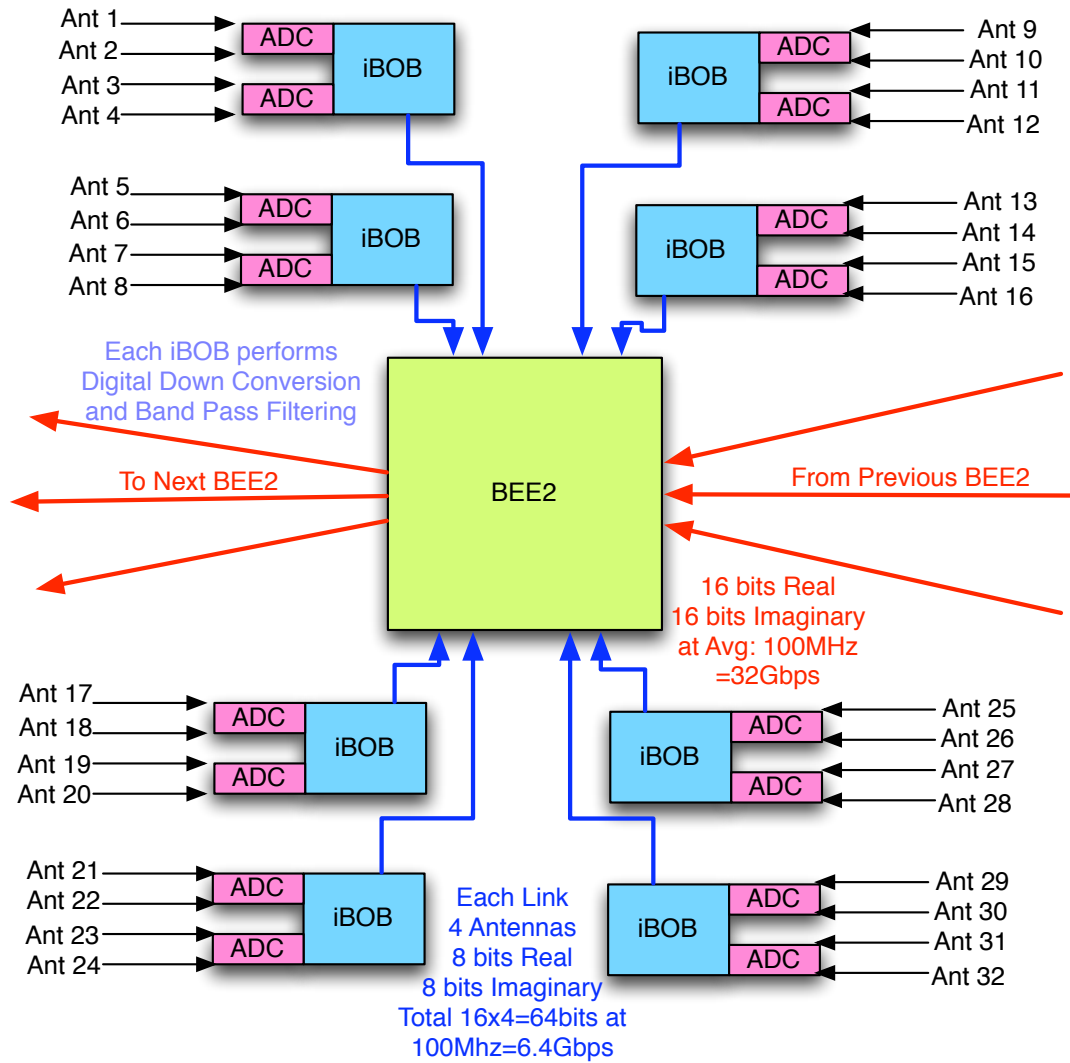


Figure 1: System level architecture.

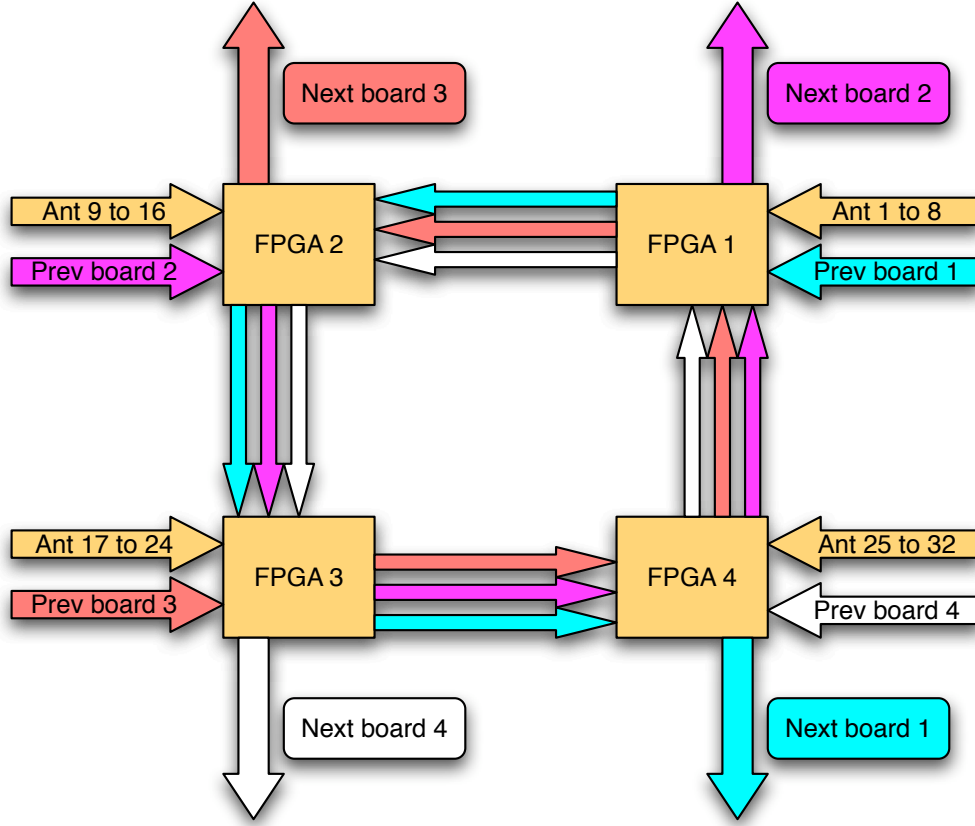


Figure 2: Board Level Architecture: Every FPGA plays the role of START for one beam, END for one beam and MID for the third beam.

derive their clock from local crystal oscillators. If data transmit and receive rates on the serial links are the same, any frequency offsets between the maser and crystal can cause the XAUI FIFOs to overflow temporarily destroying the sample number and time continuity. To avoid this scenario we run the BEE2 boards much faster than they need to. The BEE2 boards are fed a 200MHz clock from the crystal as opposed to the iBOB transmit rate of 100MHz, which ensures that any frequency drifts do not overflow the link FIFOs. The signal processing chain is designed to work with data bursts rather than a continuous stream to accommodate this unique architecture.

### 2.3 Board Level Architecture

Fig. 2 shows the design at the board level. Each FPGA plays the role of the first FPGA of 1 beam (START), the last FPGA of another beam (END) and an intermediate FPGA of a third beam (MID). For all 3 beams each FPGA processes 8 antennas, it receives 16 bits per sample (8 real, 8 imaginary) for 4 antennas per XAUI link from 2 links coming from 2 different iBOBs. It sums up these 8 antennas via programmable delay lines to get a 32 bit sum (16 real, 16 imaginary) the

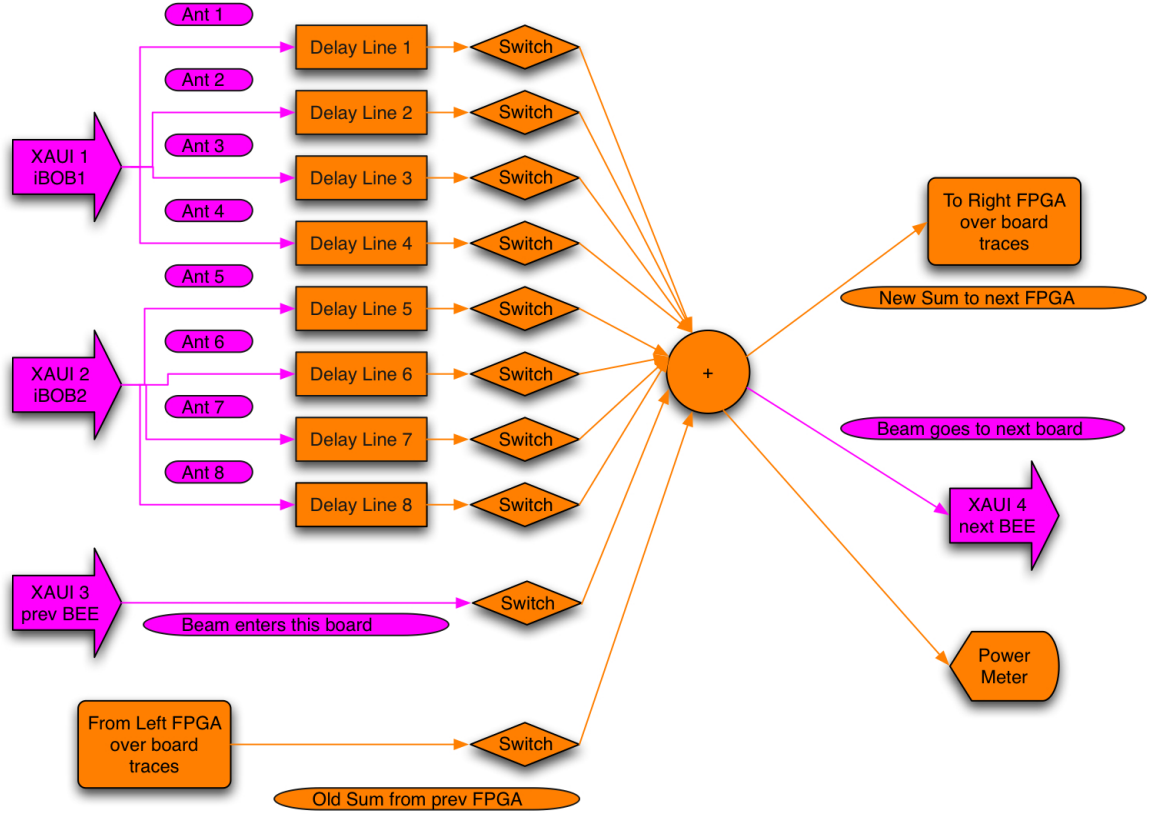


Figure 3: Chip Level Architecture: The orange blocks are replicated three times.

additional bits are kept to allow upto 256 antennas in the array. In addition to these 8 antennas the beam START gets a 32 bit sum (16 Re, 16 Im) from the previous BEE2 in the daisy chain over a XAUI link. MID and END use another 32 bit sum from the previous FPGA on the same board over inter-chip links. END outputs its final sum to the next BEE2 in the chain over XAUI. START and MID relay their final sums to the next FPGA over inter-chip links. This architecture is used because it makes optimal use of the 4 XAUI links per FPGA available on the BEE2. 2 XAUIs are used to get data from iBOBs, 1 for getting data from previous BEE2 and 1 for sending data to next BEE2. Every board handles 3 simultaneous beams so START and END of 3 beams are spread across to 3 separate FPGAs.

## 2.4 Chip Level Architecture

Fig. 3 shows the BEEmformer architecture at the level of 1 FPGA. The orange blocks are replicated thrice. The delay line consists of a FIFO based coarse delay and an FIR based fine interpolation delay. The FIR filter has demand loading of coefficients to allow delay tracking as shown in Fig.4.

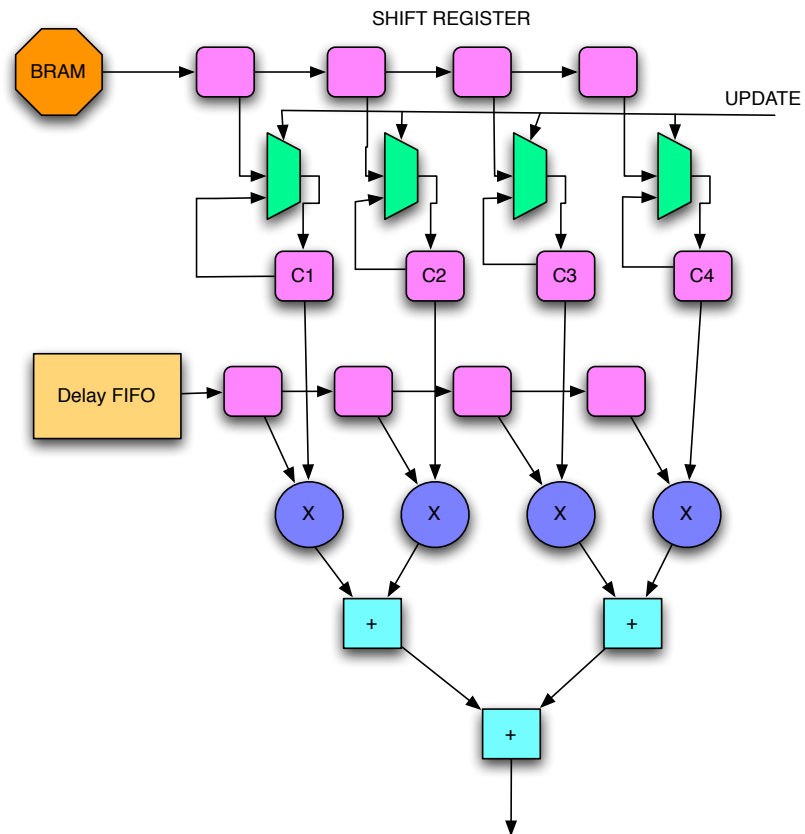


Figure 4: Delay Architecture: 4 tap FIR filter with demand loading of coefficients and double buffering to allow smooth changes.

## 2.5 BEEemformer Status

An initial prototype of BEEemformer was designed by us this semester, it is currently being finalized by ATA engineers and scheduled for testing and deployment in late June 2007.

## 3 DSP Processor based Beamformer

FPGA chips provide a good platform for designing beam formers, but incur a high development cost. Development time could be lowered by using a microprocessor and programming the algorithm in a software language. In order to provide good performance, it is necessary to use an architecture that can sustain the output both in processing and in I/O bandwidth without being too costly.

### 3.1 DSP Chip

We have analyzed how the TMS320C6455, a fixed point DSP chip developed by Texas Instruments, can be used to build a beam former similar to the one that is described in Section 2. This processor provides a dual data path very long instruction word (VLIW) architecture. Each data path has 32 registers and four functional units. Two of the functional units execute logical, arithmetic, and branching instructions, one is used for accessing memory like loads and stores, and the last functional unit is dedicated to multiply-accumulate operations (MACs). Each multiply-accumulate unit is capable of performing four 8 bit by 8 bit multiply and accumulate operations per cycle. The chip can operate at 1GHz, affording 8 MACs each nanosecond.

The architecture provides many channels for I/O. The memory interface can move 64 bits of data at 133MHz, giving a data rate of over 8 Gbps. Interchip communication is supported with separate I/O links that run at 3.125 Gbps, full-duplex. Four of these are provided on each chip giving a communication bandwidth of over 12 Gbps.

### 3.2 DSP Design

In order to compare the platforms, it is necessary to see how many antennas a single DSP can process. This analysis describes what can be done on a single datapath, in order to minimize the amount of data that needs to be moved between the two register files. We explored what it would take to implement a system with single polarization 100MHz inputs and real FIR filters (with 6 taps and 8 bit data) to form a single beam using the TMS320C6455.

Approaching the design in a similar way as the FPGA architecture, many antennas could be processed in parallel. The whole sample delay is implemented by addressing memory with an offset. To interpolate between samples, the FIR filter is implemented in software. This approach is limited by the number of registers available in the datapath. Each antenna requires 12 registers total for storing the 6 FIR coefficients as well as the 6 partial sums from each FIR. Accounting for additional looping data, such as loop counters and data pointers, results in a maximum of 24 registers that can be dedicated to FIR information, or a maximum of 2 antennas that can be processed in parallel.

Each antenna will require 6 MACs to be performed each time a new sample comes in. A new sample is produced every 10ns. With this algorithm 12 MACs will be performed every 10 ns while the datapath is capable of supporting  $10\text{ns} * 4 \text{ MACs/ns} = 40\text{MACs}$ . Most of the cycles will be wasted waiting for the next value to be produced and the MAC units will only be used at 30% utilization.

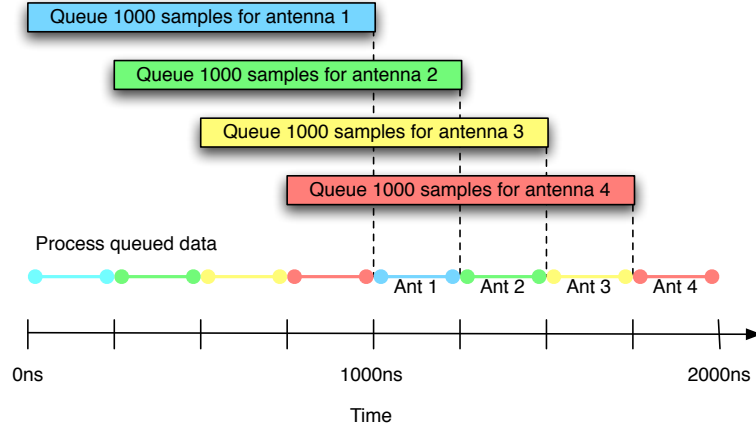


Figure 5: Time multiplexing 4 antennas on 1 datapath

```

for(i=0; i<1000; i++)
{
    output[i] += coefficient[5] * input [i];
    output[i+1] += coefficient[4] * input[i];
    output[i+2] += coefficient[3] * input[i];
    output[i+3] += coefficient[2] * input[i];
    output[i+4] += coefficient[1] * input[i];
    output[i+5] = coefficient[0] * input[i];
}

```

Figure 6: Simple C loop for processing FIR filters

Processing data that has already been queued in RAM can alleviate the need to wait for the next sample to arrive. With this approach, many antennas are time multiplexed onto a single datapath. A large number of samples from each antenna are queued and then processed at the fastest possible speed. Since the processor no longer needs to wait for data to become ready, this algorithm should increase the utilization of the processor and support more than two antennas on a single datapath.

At 100MHz, it will take 10,000ns for 1,000 samples to queue. To calculate one output sample, one new input sample must be loaded into a register and then 6 multiply accumulate operations to be executed. At 4 MACs a cycle, it will take, on average, 1.5 cycles, or ns, to calculate the FIR for a single sample, if the chip is processor limited. In order to maintain throughput, each output sample needs a corresponding 8 bit input sample loaded. With 8 Gbps throughput from RAM, two 8 bit values (one for each data path) can be loaded every 2 ns on average. In order to maintain throughput at this rate, it is necessary to preload the cache and software pipeline the loop. Fortunately, the compiler greatly aids in these optimization efforts.

To complete the rest of the beam forming, the delayed beams must be summed. First, the antennas processed on the chip must be summed together and then the 16 bit result should be sent along to another chip, the same way it summation is performed on the FPGA architecture.



To minimize additional cycles spent on adding, we will utilize the multiply accumulate from the beginning of the FIR. Referring to the loop in figure 6, calculating output[i+5] requires a multiply but no add. At this point the result from a previous antenna could be added to fully utilize the multiply accumulate. Moving the partial sums between the chips will make use of the interchip communication paths. Each chip will produce a 16 bit sample every 2 ns that needs to be transferred to an adjacent chip. This communication only requires 8 Gbps, which is much less than what is afforded by the on chip communication.

From this, it is clear that the system will be I/O limited. In the 2 ns it takes to load the data, only 6 out of a possible 8 multiplies will be calculated. This algorithm has significantly improved over the naive alternative since makes 75% utilization of the multipliers.

To process 1,000 samples of data it will take 2,000 ns plus additional latency to switch between antennas. When switching between antennas, it is necessary to load the new FIR coefficients and begin preloading the cache. In 10,000 ns, at most 4 antennas can be processed on a data path. Overall, the entire chip can process the real part of 8 antennas. To do the entire complex calculation, 2 chips are necessary for processing 8 antennas.

## 4 Conclusions

We have successfully demonstrated the suitability of the BEE2 platform for building highly scalable antenna array beam formers. In addition our comparison with DSP processors has shown us that though correlators map very well to FPGAs alone, beam formers can have alternative and equally viable DSP processor implementations.

## References

- [1] TMS320C6455 fixed-point digital signal processor. Technical report, Texas Instruments, 2007.
- [2] Chen Chang, John Wawrzynek, and Robert W. Brodersen. BEE2: A high-end reconfigurable computing system. *IEEE Design and Test*, 22(2), 2005.
- [3] R. D’Addario. ATA memo 27: Notes on delay and phase tracking for the ATA. Technical report, Radio Astronomy Labs, UC Berkeley, 2001.
- [4] R. D’Addario. ATA memo 43: Digital processing architecture of the ATA. Technical report, Radio Astronomy Labs, UC Berkeley, 2002.
- [5] G.R. Harp. ATA memo 51: Customized beam forming at the allen telescope array. Technical report, Radio Astronomy Labs, UC Berkeley, 2002.
- [6] Vinayak Nagpal. An FPGA based phased array processor for the sub-millimeter array, 2006.
- [7] Aaron Parsons, Donald Backer, Chen Chang, Daniel Chapman, Henry Chen, Patrick Crescini, Christina de Jesus, Chris Dick, Pierre Droz, David MacMahon, Kirsten Meder, Jeff Mock, Vinayak Nagpal, Borivoje Nikolic, Arash Parsa, Brian Richards, Andrew Siemion, John Wawrzynek, Dan Werthimer, and Melvyn Wright. Petaop/second FPGA signal processing for SETI and radio astronomy. *Proceedings of the Asilomar Conference on Signals, Systems, and Computers.*, 2006.
- [8] M.C.H. Wright. ATA memo 1: Astronomical imaging with the one hectare telescope. Technical report, Radio Astronomy Labs, UC Berkeley, 1998.