# Hybrid Clutter Canceler with Feedback Guided Predicative Filtering on a Heterogeneous Parallel Processing System

Jeffrey M. Brinkhus[1], Joseph A. Jezak[2], Charles Berdanier[3], Steven P. Levitan[1] and Donald M. Chiarulli[4]

[1]*Electical and Computer Engineering, University of Pittsburgh,*
[2]*Computer Engineering Graduate Program, University of Pittsburgh*
[3]*Electrical Engineering, Wright State University,*
[4]*Computer Science Department, University of Pittsburgh*

***ABSTRACT***

*In this paper we describe a novel clutter cancellation platform based on a two stage approach that combines a feedback guided predictive front-end hybrid clutter canceller with high performance back-end filtering and target detection. The front-end architecture is based on an FPGA implementation of a Kalman filter that predicts target locations in real time and removes the target signals from the incoming data prior to hybrid cancellation. The back-end is user configurable and exploits high performance GPU and multi-core parallel hardware to simultaneously compute multiple clutter suppression and target detection algorithms coupled to an intelligent selection strategy for selecting the most accurate result. These target locations are fed back to the FPGA Kalman filter periodically to update the target predictions.*

## INTRODUCTION

Clutter suppression and target recognition are by nature a heterogeneous computing problem. Like many statistical signal processing problems, no particular algorithm works best under all possible input conditions[1]. Thus, from a computational standpoint, there is no single parallel processing architecture that can satisfy all of the requirements of the various radar signal processing algorithms. Given these constraints, the best design of a radar signal processing system will combine multiple signal processing algorithms running in parallel on a tightly integrated heterogeneous parallel processing platform. On this platform, each algorithm can be matched to the optimal parallel processing architecture.

This paper describes a radar signal-processing platform based on this approach. The components of the platform are multi-core CPUs, graphical processing units (GPUs) and Field Programmable Gate Arrays (FPGAs). Each has a different architecture and is optimized for a particular parallel programming model. Each has a unique memory access and internal data communications structure. Using current COTS technology, all of these platforms can be integrated into a single heterogeneous parallel processing system at moderate expense and with much greater scalability than previously possible. For example, the test platform described in this paper, and shown in Figure 1, includes a high performance data acquisition system with an integrated FPGA for front-end filtering. This board is interfaced via an 8x PCIe interface to a commercial chassis that hosts dual 12-core processor chips, 64GB of RAM, and dual 512 processor GPUs.

This research has been partially supported by the USAF/AFRL under MYRIAD task order #520-10-10

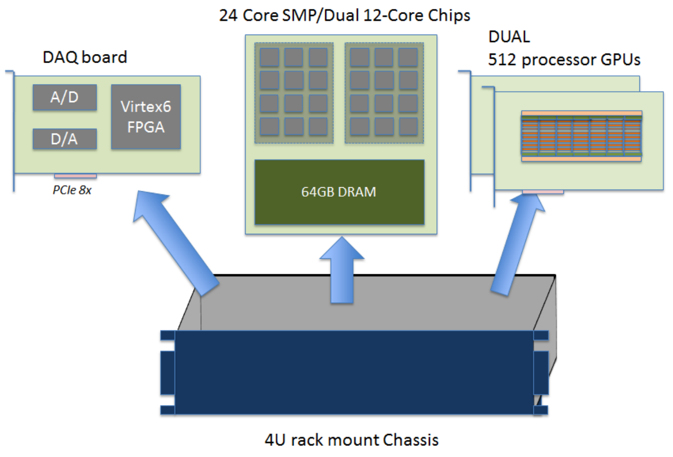


Figure 1: Heterogeneous Parallel Processing Platform

In the next three sections we describe the design of the RADAR DSP platform followed by the FPGA implementation of the real-time Kalman filter in section III. This is followed by a MATLAB model of the end-to-end system operation.

## RADAR DSP PLATFORM

This platform presents an opportunity to rethink some of the fundamental ideas about algorithms and information flow for clutter suppression and target recognition. The most basic strategy is to run an ensemble of different clutter suppression and target recognition algorithms in parallel on the same data and then select the best result based on an a-priori (or learned) statistical criterion. However, more sophisticated strategies can be designed that take better advantage of the heterogeneity of the platform. These strategies include multiple time bases and multiple feedback paths across the heterogeneous processors in the platform. The radar-processing platform discussed in this paper is an example. In this implementation, a "front-end" system, operating at the pulse repetition frequency, processes incoming data through a hybrid clutter cancellation loop that includes a hardware filter implementation in an FPGA. Simultaneously, a "back end" loop runs the basic strategy described above of running multiple algorithms in parallel on data from the front-end. These results are then used to feed back update information to the front-end hardware filter.

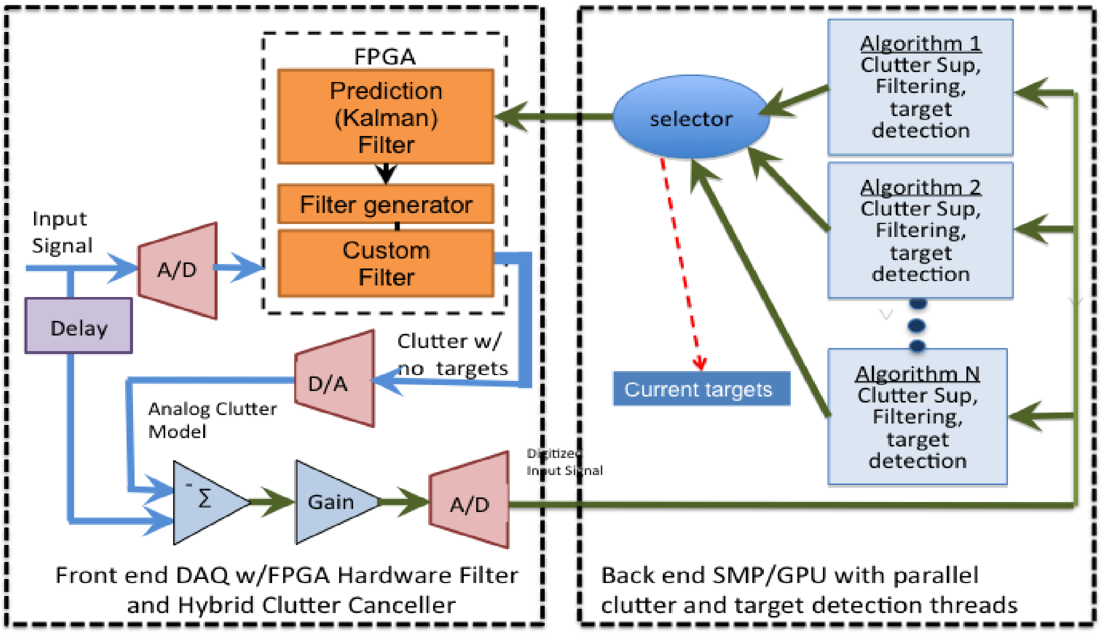

FPGA
Prediction (Kalman) Filter
Filter generator
Custom Filter
Input Signal
A/D
Delay
Clutter w/ no targets
D/A
Analog Clutter Model
- Σ
Gain
A/D
Digitized Input Signal
Front end DAQ w/FPGA Hardware Filter and Hybrid Clutter Canceller
selector
Current targets
Algorithm 1 Clutter Sup, Filtering, target detection
Algorithm 2 Clutter Sup, Filtering, target detection
Algorithm N Clutter Sup, Filtering, target detection
Back end SMP/GPU with parallel clutter and target detection threads

Figure 2: System Block Diagram

This approach is shown in more detail in the block diagram of Figure 2. The left side of the diagram shows the front-end data flow in the adaptive hybrid clutter canceller [2]. Feedback in this loop, shown in the blue line, is controlled in real time by a FGPA-based Extended Kalman filter that predicts target locations in each PRI and modifies the incoming clutter signal to eliminate the target signatures. Thus, the analog clutter suppression is more effective as it applies different levels of cancellation to clutter and target signals. In the second feedback loop, shown in green in Figure 2, the signal from the hybrid clutter canceller is transferred to the host parallel processor which runs parallel implementations of various clutter suppression and target detection algorithms. The targets that are selected from the outputs of these processes are fed back to the Kalman filter and incorporated into the real-time prediction model.

By using two feedback loops and two time bases, the system works somewhat differently than conventional clutter suppression. Specifically, the real-time loop cancels the clutter based on data for the current pulse repetition interval (PRI) with the moving targets removed; the analog subtraction suppresses only the clutter. The back-end loop further processes this result to generate and update the target model used by the Kalman filter. With no targets, the Kalman filter will suppress all stationary clutter sources, leaving only moving targets and non-stationary clutter in the back-end signal. Thus target acquisition happens when the back-end algorithms detect a moving target. This can be made even more effective if the back-end algorithms incorporate non-stationary clutter and jamming models.

## FPGA-Based Real-Time Kalman Filter

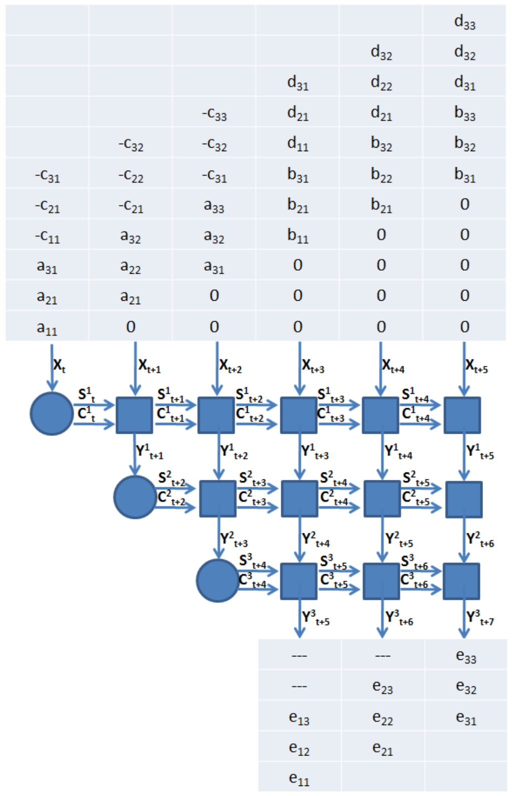
In this section we describe the algorithm and implementation details of the Kalman filter in the FPGA hardware. The Kalman filter tracks and predicts position, velocity, and acceleration for each of up to 25 targets and generates a new prediction on each PRI.

### *Systolic Array Based Parallel Algorithm*

The greatest challenge for efficiently implementing the Kalman filter algorithm in an FPGA is performing the matrix inversion operation required by the update equations. Our implementation is based on a design by Bigdeli et al [3]. This algorithm takes advantage of the Schur complement[4], $D + CA^{-1}B$, which can be calculated using the compound matrix $M = \begin{matrix} A & B \\ -C & D \end{matrix}$ using the Faddeev algorithm[4]. This computation is performed through matrix triangulation and annulment and has a pipelined systolic array solution.

The pipelined systolic array architecture for a single computation of a 3x3 Schur complement using the Faddeev algorithm is shown in Figure 3. In this architecture, two types of computational cells, boundary cells (shown as circles) and internal cells (shown as squares), are arranged as a right-angled trapezoid with boundary cells placed at the left hand diagonal. Inputs A, B, -C, and D, are clocked into the array diagonally from the top as shown, one row per clock cycle. After an internal latency of 2N clock cycles, where N=3 for the figure, the resultant Schur complement array begins to be clocked out, one diagonal set if elements per clock cycle over the next 2N cycles.

The size of the systolic processor depends on the size of the input arrays. Figure 3 shows a 3x3 implementation with six columns and three rows. In our implementation the state array holds nine variables, corresponding to the x, y, and z components of position, velocity, and acceleration, and thus requires a 9x9 implementation with 18 columns and nine rows. Moreover, the Kalman filter algorithm requires multiple iterations of the Schur complement. Each iteration calculates one of the Kalman filter equations manipulated into the Schur complement form $D + CA^{-1}B$, as shown in Table 1.



$d_{33}$
$d_{32}$ $d_{32}$
$d_{31}$ $d_{22}$ $d_{31}$
$-c_{33}$ $d_{21}$ $d_{21}$ $b_{33}$
$-c_{32}$ $-c_{32}$ $d_{11}$ $b_{32}$ $b_{32}$
$-c_{31}$ $-c_{22}$ $-c_{31}$ $b_{31}$ $b_{22}$ $b_{31}$
$-c_{21}$ $-c_{21}$ $a_{33}$ $b_{21}$ $b_{21}$ 0
$-c_{11}$ $a_{32}$ $a_{32}$ $b_{11}$ 0 0
$a_{31}$ $a_{22}$ $a_{31}$ 0 0 0
$a_{21}$ $a_{21}$ 0 0 0 0
$a_{11}$ 0 0 0 0 0
$X_t$ $X_{t+1}$ $X_{t+2}$ $X_{t+3}$ $X_{t+4}$ $X_{t+5}$
$S^1_t$ $C^1_t$ $S^1_{t+1}$ $C^1_{t+1}$ $S^1_{t+2}$ $C^1_{t+2}$ $S^1_{t+3}$ $C^1_{t+3}$ $S^1_{t+4}$ $C^1_{t+4}$
$Y^1_{t+1}$ $Y^1_{t+2}$ $Y^1_{t+3}$ $Y^1_{t+4}$ $Y^1_{t+5}$
$S^2_{t+2}$ $C^2_{t+2}$ $S^2_{t+3}$ $C^2_{t+3}$ $S^2_{t+4}$ $C^2_{t+4}$ $S^2_{t+5}$ $C^2_{t+5}$
$Y^2_{t+3}$ $Y^2_{t+4}$ $Y^2_{t+5}$ $Y^2_{t+6}$
$S^3_{t+4}$ $C^3_{t+4}$ $S^3_{t+5}$ $C^3_{t+5}$ $S^3_{t+6}$ $C^3_{t+6}$
$Y^3_{t+5}$ $Y^3_{t+6}$ $Y^3_{t+7}$
--- --- $e_{33}$
--- $e_{23}$ $e_{32}$
$e_{13}$ $e_{22}$ $e_{31}$
$e_{12}$ $e_{21}$
$e_{11}$

Figure 3: A 3x3 Systolic Array Example

In the most general implementation of the Kalman filter there are nine prediction and feedback equations[5]. However, in this implementation, the inputs to the Kalman filter (target location, velocity, and acceleration) are in the same form as the internal state. Thus, the observation model array, 'H,' from the generalized 9-equation model is identity, making one matrix multiplication unnecessary. Moreover, there is no control-input model, 'B,' or control vector, 'u,' required in our version which also removes one matrix multiplication from the general implementation. Taking this into consideration, one of nine equations is completely eliminated, leaving eight that require computation. Two of these equations are reduced to a simple matrix addition that can be performed in one clock cycle by a one-dimensional array of adders. Thus, the number of Kalman equations that need to be mapped to the Schur complement is reduced from nine to six.

In a fully systolic implementation of all eight steps, individual arrays for each of the Schur complement stages can be arranged end-to-end such that the output of one is immediately clocked into the input of the next. The combination forms new systolic array of twice the depth that computes the two equations with no internal buffering between the stages. An analysis of the data flow and data dependencies in the Kalman equations reveals the specific arrangement of the stages that is the fully systolic implementation shown in the graph shown in Figure 4.

| Pipeline Stage | Filter Operation | Systolic Mappings | |
|---|---|---|---|
| 1 | Predict State Estimate $x = F * x$ | A = I<br>B = x | C = F<br>D = 0 |
| 2 | Predict Estimate Covariance (a) $FP = F * P$ | A = I<br>B = P | C = F<br>D = 0 |
| 3 | Predict Estimate Covariance (b) $P = FP * F' + Q$ | A =I<br>B = F' | C = FP<br>D = Q |
| 4 | Residual Covariance $S = P + R$ | A = P<br>B = R | |
| 5 | Kalman Gain $K = P * S^{-1}$ | A = S<br>B = I | C = P<br>D = 0 |
| 6 | Update Estimate Covariance $P = -K * P + P$ | A = I<br>B = P | C = -K<br>D = P |
| 7 | Residual $y = z + -x$ | A = z<br>B = -x | |
| 8 | Update State Estimate $x = K * y + x$ | A = I<br>B = $y$ | C = K<br>D = x |

Table 1: Kalman Filter Equations for optimized 6+2 implementation and Systolic Mappings for the Faddeev Algorithm

In this figure, each of the numbered round blue blocks represents a Schur complement stage corresponding to the equation number from Table 1. The round red blocks are the adder arrays for the simplified equations, 4 and 7. In cases where data dependences span one or more stages of the pipelined systolic computation, the square blocks represent buffer arrays that delay the variable shown for one stage of clock cycles.

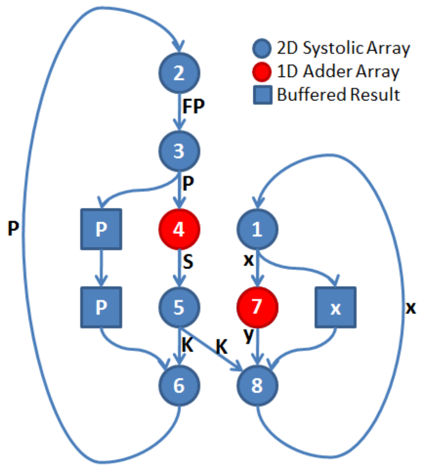


Figure 4: Kalman Filter 8-Equation Pipeline Timing Diagram.

### *Resource and Timing Analysis*

By analyzing the graph shown in figure 4 we can estimate the total logic required to implement the filter as well as the throughput and latency of the computations. Our first analysis compares the logic required to the available resources in the Xilinx Virtex 6 family [6] of FPGAs. FPGAs in this family have special logic blocks (DSP slices) that are optimized for the high speed multiply-accumulate operations required by each systolic cell. Our implementation maps one systolic cell to each DSP slice. Figure 5 is a plot of the number of systolic cells required for a full implementation versus the number of elements in the state array (N). The horizontal lines on this plot are the DSP slice capacity of several representative FPGAs in the Virtex 6 family. Based on this analysis, an N=9 element state (position, velocity, and acceleration) implementation will fit comfortably into any of the larger FPGAs in the Virtex 6 family.

Moving now to the timing analysis, by measuring the longest delay path through the graph shown in figure 4, the end-to-end latency of each target prediction can be estimated. Each Schur complement stage in the graph requires 2N systolic

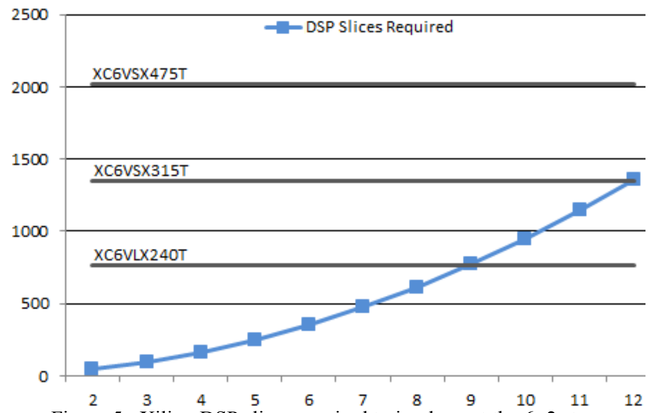


Figure 5: Xilinx DSP slices required to implement the 6+2 stage pipelined Kalman filter versus size of the state vector. Horizontal lines show the capacity of various Virtex-6 devices.

computation steps, where N is the number of elements in the state array. Each node in the systolic array requires three clock cycles to compute the required multiply-add operation. Addition stages each require one clock cycle. Thus the longest delay path in figure 4 corresponds to 4 Schur stages and one addition stage or (4*2N*3)+1 clock cycles. Figure 6 is plot of this longest delay path length in clock cycles versus N. For the N=9 implementation described here, the end-to-end latency of each computation will be 217 clock cycles.

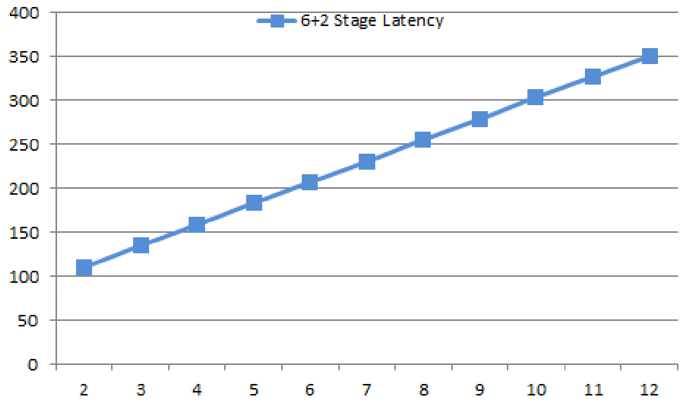


Figure 6: End-to-end Latency of the 6+2 stage Kalman filter systolic processor implementation versus number of elements in the state vector

Throughput is the number of computations (target predictions) completed by the systolic processor per second. As a fully pipelined system, this systolic processor will generate a new N-variable target prediction each (2N*3) clock cycles. This is the time that it takes to do a single Schur complement stage in the Kalman filter pipeline.

For Xilinx Virtex 6 FPGA technology, best case clock rates reach up to 600MHz. As a conservative estimate, we assume an actual clock rate for the array of 450MHz. At this clock frequency, the target prediction throughput of a system with a nine variable state array will be 450MHz/(54 cycles/target update) = 8.3 million target updates/second.

Combining the latency and throughput results demonstrates the advantage of the pipelined architecture in the context of multiple target tracking. If tracking a single target, the latency and throughput of the system will be the same, equal to the end-to-end pipeline latency period. For multiple targets, up to four target computations can be active at any time, the throughput rate for successive predictions for 25 targets is the pipeline throughput rate divided by 25 or 332KHz. In other words, a new prediction for all 25 target tracks can be generated every 3.0μsec.

## FULL SYSTEM DEMONSTRATION IN MATLAB

To verify this approach, we have built an end-to-end simulation model in Matlab and tested the system for several radar architectures and target environments. The radar model is based on the phased array toolkit initially released with the 2011a version of Matlab. In this example we modeled a mono-static, single channel, isotropic S-band radar with three targets.

| | |
|---|---|
| Frequency | 3GHz |
| Channels | 1 |
| Pulse width | 3e-8 m |
| PRF | 300 KHz |
| Pulses | 128 |
| Max Range | 5 Km |
| Range Res | 50 m |
| Rx Gain | 25 |
| Noise BW | 3.3 GHz |
| Target 1 range<br>Target 2 range<br>Target 3 range | 4.05 Km<br>7.1 Km<br>7.8 Km |

Table 2: Radar data cube and model parameters

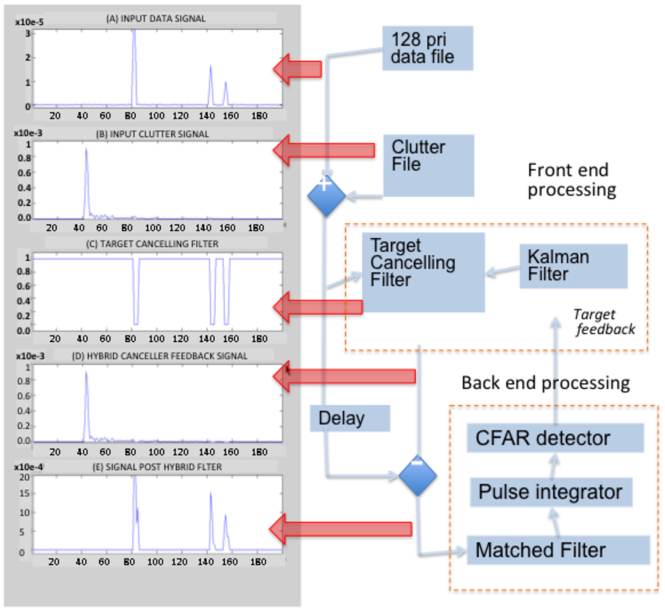


Figure 7: Block diagram of full system Matlab model (left) and result waveforms at various processing stages (right)

The radar model and target parameters for the test data cube in this demonstration are listed in Table 2. Figure 7 shows a block diagram of the radar processing model with result waveforms shown at various processing stages plotted as signal amplitude versus range bin. The top waveform, labeled (A), is the raw data generated by the radar system model configured with three targets at the ranges listed in table 2. The next trace, labeled (B), is a clutter waveform that is added to the raw waveform in the model to create a test waveform with a 20db signal to clutter/noise ratio. Based on predicted target locations from the Kalman filter, a custom "target cancelling" filter is generated in the hybrid clutter canceller loop and shown on the trace labeled (C). When this filter is applied to the combined target pulse and clutter input waveform, the result is the waveform labeled (D). This is the data plus clutter waveform with the targets removed that is converted back to analog and subtracted from the delayed raw input signal. The result is waveform (E), the original waveform with the large ground clutter spike removed by the hybrid clutter canceller. The clutter suppression and target detection algorithms on the host processor read this waveform and extract the target information. This information is feed back to the Kalman filter as measured data for updating the state variables used the subsequent prediction.

## Summary and Conclusions

The significance of this system design is in its versatility. By decoupling of the front-end time base, there are few limits on the scalability of the back-end processing loop. One can envisage large numbers of multi-processors and/or GPUs, each running different clutter suppression and target recognition algorithms, parallelized and optimized for different clutter environments. The best result from these algorithms can be dynamically selected to be fed back to the front-end filters. This capability will be particularly important as we move to machine learning and other cognitive approaches for radar DSP. This type of parallel computation with selection of the best result has been the model of machine learning based decision systems in other domains and one can only expect that the same will be true for radar DSP[7].

## References

[1] Ali H. Sayed ; Fundamentals of Adaptive Filtering; 2003, Wiley-IEEE Press, ISBN 9780471461265

[2] Brown D., Weiner D., and Wicks M; Hybrid Clutter Cancellation Method and System for Improved Radar Performance; US Patent # 5,061,934, November 1990.

[3] Abbas Bigdeli, Morteza Biglari-Abhari, Zoran Salcic, and Yat Tin Lai, "A New Pipelined Systolic Array-Based Architecture for Matrix Inversion in FPGAs with Kalman Filter Case Study," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 89186, 12 pages, 2006. doi:10.1155/ASP/2006/89186

[4] Haynsworth, E. V., "On the Schur Complement", *Basel Mathematical Notes*, #BNB 20, 17 pages, June 1968.

[5] Greg Welch and Gary Bishop; An Introduction to the Kalman Filter; Technical Report # TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill; http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[6] Xilinx Inc.; Virtex 6 Family Overview; http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf

[7] Haykin, S.; Cognitive radar: a way of the future; IEEE Signal Processing Magazine, Vol 23 Issue:1; Jan. 2006