Degradable Tracking System based on Hardware Multi-Model Estimators

Kiriakos Kiriakidis, Brien Croteau,
Tracie Severson, and Erick Rodriguez-Seda
Department of Weapons, Robotics, and Control Engineering
Department of Cyber Science, United States Naval Academy
Email: {kiriakid,croteau,severson,rodrigue}@usna.edu

Ryan Robucci, Riadul Islam, and Saad Rahman
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Email: {robucci,riaduli,rahman2}@umbc.edu

2022 Resilience Week (RWS) | 978-1-6654-8819-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/RWS55399.2022.9984042

***Abstract*—Sensing systems onboard unmanned vehicles operate in an environment of constrained computational resources. A cyber-attack may primarily aim to degrade these computing devices and, ultimately, incapacitate the sensing system itself. To prepare a prototype tracking system for degradation, this paper proposes distributed hardware implementation of a Multiple Model estimator on two FPGA units and, after an attack, adaptation of the estimator by leveraging Dynamic Partial Reconfiguration of the single surviving FPGA. The method ensures that the most likely models of the estimator are loaded on to the fabric of the surviving FPGA with minimal interruption.**

I. Introduction

The computational resources of Cyber-Physical Systems (CPSs) are high-value targets for cyber-attacks. A CPS with all of its physical components intact would be unable to operate, should its computing ability be diminished. After a cyber-attack causes a percentage of its computing pool to be lost, the question of continuing operation from a degraded state relying only on its surviving computational resources arises naturally.

This research addresses the above question by developing a software-hardware co-design approach to adaptive CPSs: a synthesis of dynamically reconfigurable hardware and scalable algorithms directly programmed on the hardware. The specific CPS example studied here is a system that comprises radar detectors and computing platforms to support algorithms for tracking dynamic objects. Similar systems are found in products from automotive to maritime applications.

The authors design the tracking system so that it is prepared for degradation. Having lost computational resources to cyber-attack, the system undergoes reconfiguration in order to continue to operate in the physical domain albeit with degraded performance. Preparation for degradation requires algorithms whose structure is dynamically scalable and adaptable to results in the field. Figure 1 illustrates an example of system diminution and re-tasking of computing hardware to support hardware-accelerated degradable algorithms.

The authors develop a hardware-software co-design of a distributed computer system comprising FPGA units where a Multiple Model (MM) estimator resides. When a cyber-attack impacts one or more FPGA units, Dynamic Partial Reconfiguration (DPR) is used to reprogram parts of the surviving units with the most likely among the multiple models subject to the computational resources remaining available. Along with achieving continuity in tracking, the method also offsets the cost of DPR by mitigating the drop in tracking performance.

For maneuvering targets, MM estimators are a class of filters with the ability to estimate several modes of the target's dynamic behavior [1]. Their key characteristic is to construct the likelihood of each mode by running the models that generate each of the assumed modes simultaneously. By limiting the number of modes, MM estimators trade off performance for less computational complexity. Nevertheless, practical issues with processing in real-time were recognized early on and solutions involving parallel implementations were sought out [2].

Although development tools increasingly support *dynamic partial reconfiguration*, making system-level assessment and run-time decisions at the system-level for reconfiguration towards system adaptation is an open topic for research. Algorithms for scheduling of compute tasks are usually designed independent of data processing and control algorithms, and without regard at run-time to processing algorithm performance. Reconfiguration in FPGAs is penalized by a reconfiguration overhead time, but it allows dynamic competencies by changing only parts of an FPGA design that would be otherwise statically configured as accelerators for specific competencies such as real-time feedback control, implementation of filtering, machine-learning classifiers or support of on-line learning.

A. Related Works

Researchers identified the need for DPR several decades ago due to the large reconfiguration time of an FPGA [3]. Today both the leading FPGA manufacturers (i.e., Xilinx and Intel) support DPR. In the early years of this unique technology, Xilinx used static and active partial reconfiguration [4], [5]. The static case allows users to modify a portion of the FPGA while the entire fabric remains inactive. In the latter case, the users can modify the configuration of an active FPGA. In this research, the active option is selected for adaptive run-time decision-making.

The advantages of DPR include the reduction of the size of FPGA resources to perform a given computation, consequently reducing power consumption and cost. Besides, DPR provides flexibility in the choices of algorithms available to

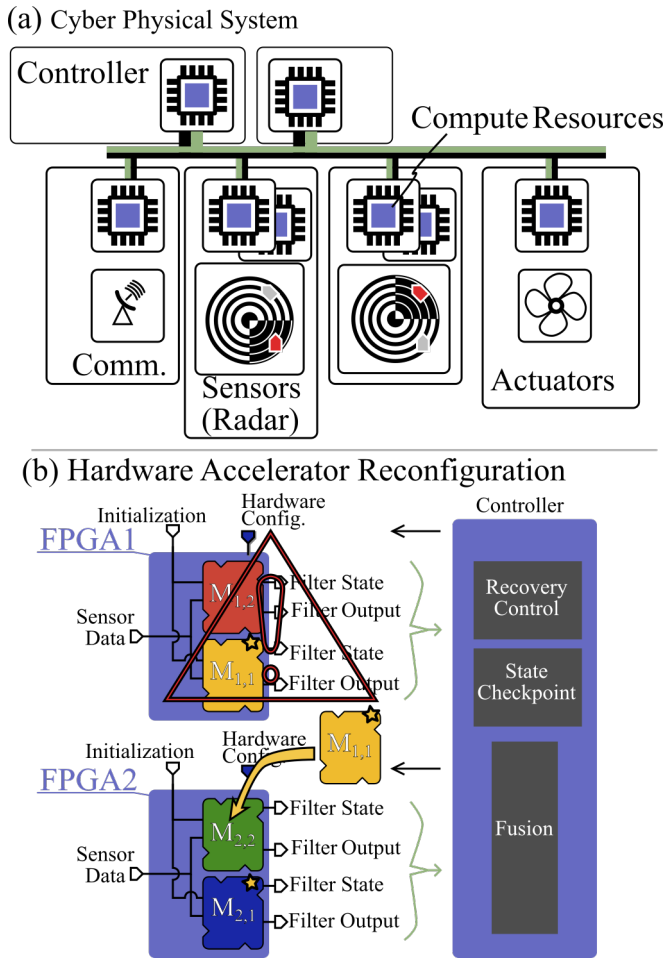


Fig. 1: Degradable performance hardware and algorithms subject to loss of computing hardware. (a) Resilient CPS architecture with distributed algorithms and reconfigurable computing hardware. (b) Under normal operation, preparation for degradation involves selection of best performing models (starred). Upon degradation of hardware, the scalable algorithms and hardware implementations are adapted to the surviving computational resources. Partial reconfiguration enables hardware retasking to support needs arising from lost hardware.

an application, improving FPGA-based fault tolerance and design security. In recent years, researchers use FPGA DPR to improve the cyber-physical system reliability and lifetime considering hardware trojan attacks [6]. Other researchers effectively used DPR by swapping equivalent secured hardware accelerators [7].

High demand in terms of computational resources is intrinsic to Kalman Filters (KFs) and has steered real-time applications toward parallel implementation of the algorithm. Pioneering works include systolic array processors [8]. The regularity of processor cells makes the systolic architecture suitable for VLSI implementation. The advent of FPGAs offered a malleable platform to design hardware architectures for KFs [9]. Later works pursued systolic structures to facilitate the matrix operations and, particularly, matrix inversion in update step of the filter [10], [11], [12].

Conventional real-time embedded systems improve resiliency by introducing check-pointing (i.e., temporal redundancy) [13], [14] and resource replication (i.e., spatial redundancy) [15], [16]. An intelligent scheduler combined both time and active space redundancy considering multiple faults and introduced a globally optimized check-pointing technique using conventional tabu-search algorithms [16].

B. Contributions

To the best of the authors' knowledge, the adaptive response to cyber-attack using DPR of FPGAs and scalable MM estimators has not been investigated in the literature. In terms of theoretical contribution, the result of the proposed synthesis will be larger than the sum of its parts. Without DPR, the surviving FPGA boards would carry on running resident models irrespective of the probability of these models given the data. On the other end of the spectrum, without the MM estimation algorithm prioritizing models for the surviving FPGAs, DPR would only be able to transfer the bit stream of a last resort alternative such as the maximum maneuver KF decreasing tracking performance [17].

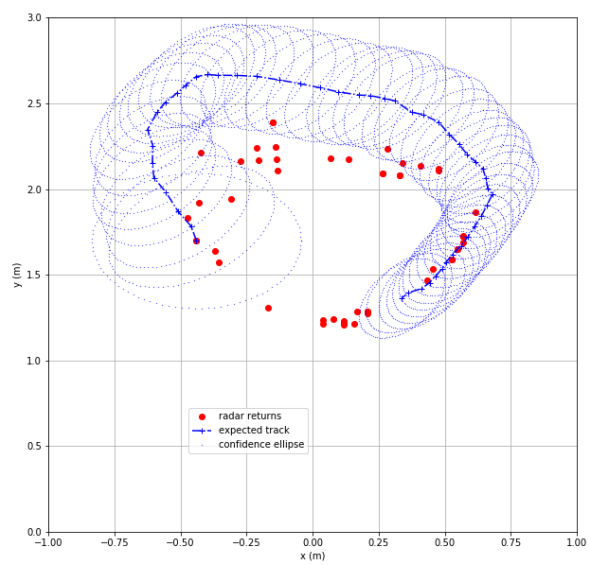


Fig. 2: The track of an RC boat sailing clockwise on a circular path inside the test pool. The millimeter wave radar is located poolside at the origin. The tracking algorithm processes radar returns (red dots) and generates the expected position of the boat (blue plus marks) and confidence ellipse (shown for half-sigma).

II. Tracking Algorithm Design

The tracking algorithm considered here processes radar returns and calculates the probability distributions of kinematic properties of a target. A typical track from experiments with an RC boat and millimeter wave radar (discussed later in the paper) is depicted in Fig. 2.

To meet the co-design requirement for a dynamically scalable and adaptable algorithm the specific class of MM estimators was selected [1]. The essence of MM estimators is a bank of elemental filters that are run as an ensemble. With the hypothesis that the target's motion changes modes (e.g., from constant acceleration to constant velocity) each elemental filter is matched to the model of a particular modal behavior.

For the purpose of this paper, the elemental filters are run in parallel and the likelihoods of their measurement residuals are calculated for each sample. Bayes's theorem is employed to estimate the posterior probability of each filter by blending their likelihoods. This is the process of fusion in Fig. 1 and, in turn, it yields a ranking of the filters from most to least probable.

A. Target Motion Models

The following sampled-data state equation can represent two fundamental modal behaviors of the target's motion relative to the radar array:

$$X_k = F_k X_{k-1} + V_k, \;\; k = 1, 2, \ldots, \tag{1}$$

For motion on the horizontal plane, the state vector comprises the kinematic quantities of the target, e.g.,

$$X_k = \left[\begin{array}{ccc|ccc} x_k & \dot{x}_k & \ddot{x}_k & y_k & \dot{y}_k & \ddot{y}_k \end{array} \right]$$

where x_k and y_k the target's Cartesian coordinates as indicated in Fig. 2. For the two fundamental modes considered, the assumed mode of behavior will determine the size of the state vector. The first fundamental mode is the nearly Constant Velocity (CV) mode, which is generated by Eq. (1) with $4{\times}4$ block-diagonal transition matrix

$$F_k = \left[\begin{array}{cc} A_k & \\ & A_k \end{array} \right], \text{ where } A_k = \left[\begin{array}{cc} 1 & \delta_k \\ 0 & 1 \end{array} \right]$$

and δ_k is the sample period. The second mode is the near Constant Acceleration (CA) mode, which is also generated by Eq. (1) with $6{\times}6$ transition matrix, F_k, where

$$A_k = \left[\begin{array}{ccc} 1 & \delta_k & 0.5\delta_k^2 \\ 0 & 1 & \delta_k \\ 0 & 0 & 1 \end{array} \right].$$

The dimensions of the process noise vector V_k and its covariance matrix Q are scaled accordingly.

B. Measurement Model

The polar coordinates of the target's range r_k and azimuth θ_k are measured based on the radar returns shown in Fig. 2. The Cartesian coordinates of the returns are calculated as follows:

$$x_k = r_k \sin(\theta_k), \;\; y_k = r_k \cos(\theta_k).$$

These measurements are related to the state vector through the measurement matrix H as follows:

$$Z_k = H X_k + W_k$$

The measurement noise is W_k and its covariance matrix R is determined by coordinate transformation [18].

C. Elemental Filters and Hardware Acceleration

The elemental filters in the MM ensemble are based on Eq. (1). The matrix operations of the conventional KF algorithm are programmed directly on hardware using Faddeeva's elimination [19]. This numerical technique employs Gaussian elimination without back substitution (return course) and can be very efficient. The prediction step of the elemental filter is

$$\hat{X}_k^{\#} = F_k \hat{X}_{k-1}, \;\; P_k^{\#} = F_k P_{k-1} F_k^T + Q$$

where $\hat{X}_k^{\#}$ and $P_k^{\#}$ are the predicted state and error covariance, respectively. After measurement, the correction step is

$$\begin{aligned} E_k &= Z_k - H\hat{X}_k^{\#}, & S_k &= H P_k^{\#} H^T + R, \\ K_k &= P_k^{\#} H^T S_k^{-1}, & \hat{X}_k &= \hat{X}_k^{\#} + K_k E_k, \\ P_k &= (I - K_k H) P_k^{\#}, \end{aligned}$$

where E_k and S_k are the measurement residual and its covariance, respectively, and K_k is the filter gain.

D. Ensemble Adaptation using Model Probabilities and DPR

Elemental filters are differentiated by the following two attributes: 1) the mode of behavior modeled, resulting in matrix structure CV or CA, and 2) the magnitude assumed for the process noise covariance Q. For example, a bank of four may include two CV elemental filters with input covariance matrices $Q_1^{CV} = 0.75$ and $Q_2^{CV} = 1$ and two CA ones with $Q_1^{CA} = 0.05$ and $Q_2^{CA} = 0.005$. Since these filters share the same data, the measurement noise covariance R is common among all four.

Given the data Z_k, the posterior probability that a model in the ensemble matches the motion of the target is calculated from the corresponding filter's measurement residual E_k and covariance matrix S_k. For the ensemble of four above, Fig. 3 depicts an evolution pattern of model probabilities. Suppose the two CV models and the CA ones are respectively programmed on two FPGA platforms as shown on the left in Fig. 4. If a cyber-attack damaged the FPGA of the most probable models, the loss of those models would trigger the adaptation of the software-hardware configuration. Figure 4 illustrates the successive DPR stages of the surviving FPGA in order to adapt the ensemble while ensuring continuity of target tracking.

III. Hardware Design for DPR

The elemental filters intended for MM estimators were designed in Verilog Hardware Description Language (HDL) [20], and the functional and timing verification were carried out by Vivado Design Suite 2021.1. The design was then implemented and tested in the Xilinx Xc7z020clg484-1 FPGA

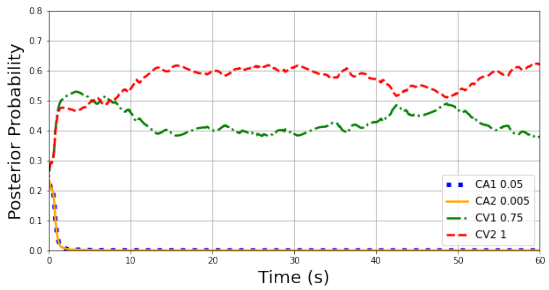


Fig. 3: Model probabilities for an ensemble comprising two different CV and CA elemental filters.

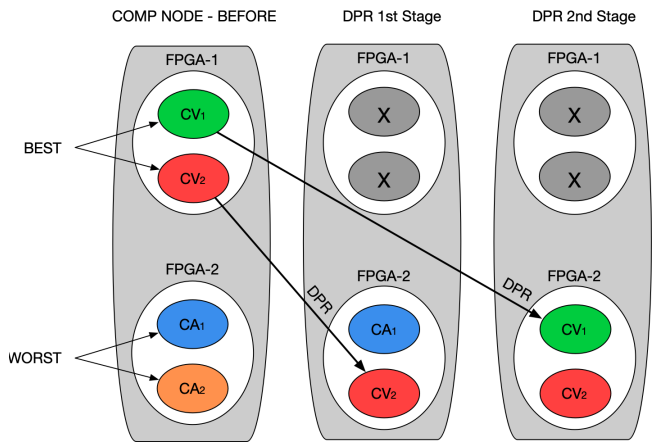


Fig. 4: Graphic showing the partial reprogramming stages after an event where the two best preforming filters are lost (worst-case failure). To adapt the ensemble while preserving continuity of tracking, only one elemental filter is reprogrammed on the surviving FPGA at each DPR stage. First, the worst filter is replaced with the current best performing one and, then, in the following stage the second-best filter is loaded.

(Zedboard) using both synthetic and real data. Figure 5 depicts the hardware setup consisting of the FPGA (HW accelerator), Raspberry Pi 4 Model B 2 GB (controller), and 60-GHz Texas Instrument (TI) IWR6843 Antenna on Package (AoP) Evaluation Module (EVM) mmW radar (sensor). The labels here refer to the functions identified in Figure 1.

Firstly, target information is preprocessed and transmitted to the Raspberry PI as point cloud data in the spherical coordinate system via Universal Asynchronous Receiver Transmitter (UART) at a baud rate of 921600 bits/s. Raspberry Pi then stores target data as a .csv file and transmits it to the FPGA for post-processing using Serial Peripheral Interface (SPI) at 4 MHz. Finally, the FPGA receives the point cloud data and then sends the filtered data back to the controller to be fused together to determine the most likely target location estimate.

In case of degraded operation, the corresponding filter configuration bitstreams are read from an onboard memory (SD card) and programmed using the Processor Configuration Access Port (PCAP) which changes part of the FPGA configuration in 290 ms while other parts of the FPGA continue processing data. Table I presents the resource utilization for implementing elemental filters from the Vivado Post-implementation report running at 100 MHz system clock.

TABLE I: HW resource utilization

Models	Input size	LUTs	FFs	DSPs
CV	4x32	28.21%	16.00%	14.55%
CA	6x32	63.14%	29.42%	21.82%

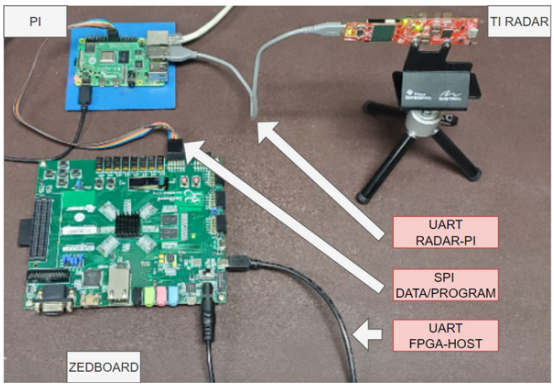


Fig. 5: Hardware set-up for filter performance evaluation. Unmarked wires are for power. TI mmW radar is powered by USB-UART cable.

IV. Experimental Demonstration

A. Experimental Testbed

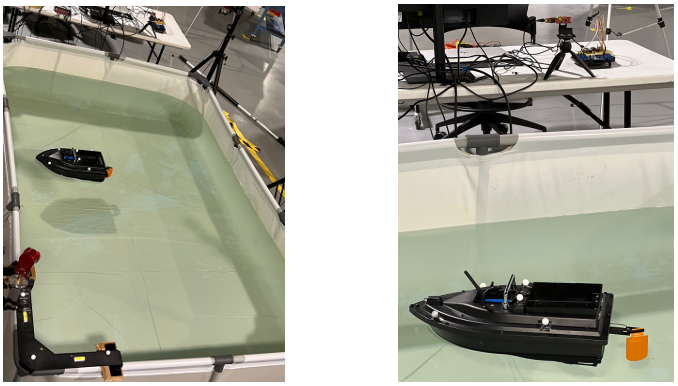
The search and surveillance testbed in Fig. 6a comprises of an RC boat which represented a scaled-down version of an unmanned surface vehicle (USV), an above ground pool, and a Milimeter Wave (mmW) radar sensor that provides updating tracking information on the RC boat's range, azimuth and elevation as well Doppler speed and SNR. An OptiTrack Motion Capture System provided position and velocity information on the USV and served as the source of truth data.

The above ground pool is 10ft by 6.5ft filled to a depth of approximately 3ft. The RC controller received motor commands directly from an LPC 1768 ARM microcontroller that interfaced with a PC running MATLAB and sent those commands to the RC boat to control its speed. The RC boat was tracked by the mmW radar sensor. The sensor module enabled access to point-cloud data and power via a USB interface to the PC. A close-up view of the RC boat and sensor is depicted in Fig. 6b.

B. Experimental Approach

To demonstrate the adaptation of the hardaware MM estimator by DPR and according to the most likely target models, a number of experiments were conducted to generate data rich in both CV and CA modes of behavior.

Due to limitations in pool size, it was necessary to drive the RC boat in a circular motion to avoid making contact with the pool walls. Each experiment was run for a minimum



(a) Experimental set-up

(b) Close-up view showing RC boat and sensor

Fig. 6: Search and surveillance testbed.

of 60 seconds with the first 10 seconds of each run reserved for calibration of the mmW sensor. Therefore, while the RC boat was in the field of view of the radar sensor at the start of the experiment, it was motionless in the water. After the initial 10 seconds, the boat received motor commands commensurate with the desired motion profile (either CV or CA).

In the CV experiment, the goal was to drive the motion of the RC boat so that it rotated clockwise at a constant speed. In the CA experiment, the goal was to drive the RC boat such that it initially rotated quickly in the clockwise direction, then slowly decreased its rotation rate until it stopped turning, and then after a period of time increased its rotation rate until it reached the maximum RPMs.

C. Experimental Results

Radar data from one experiment were collected and analyzed using a software version of the KF's FPGA hardware implementation described in Section III. The MM estimator's complete ensemble comprises four different filters (two CA and two CV ones) as discussed in Section II. In Figure 3, the posterior probability of each filter over time is shown. For these data, the CV filters were determined to be more likely based on how well their filter predictions were matching the incoming radar data. Note that the rank order of filters is dynamic driven by the data.

Figure 7 shows the first seven seconds of the boat's motion where it makes one circle in the water. This agrees with the results from Figure 3 that show the CV models should do a better job of tracking the target boat.

Figure 8 and Table II use a numerical metric to compare the filter performance. The Mean Squared Error (MSE) was calculated between each filter estimated position output, x_f, at each time step and the raw radar location input, x_r.

$$MSE = \frac{1}{T_{end} - T_{st}} \sum_{i=T_{st}}^{T_{end}} [x_r(i) - x_f(i)]^2 \qquad (2)$$

Figure 8 shows that, according to Eq. 2, the MSE metric for two CV filters are lowest. If a cyber-attack were to take place at 5 seconds and prevented the use of half of the computing resources that currently hold the best two performing filters (CV2 and CV1) as depicted in Figure 4, then tracking performance would suffer having to rely on the two worst performing filters (CA1 and CA2). The second row of Table II shows the MSE values over the time period from 5-120 seconds using just CA1 and CA2. Assuming a nominal reprogramming time of 3 seconds, the best filter (CV2) can be reloaded by 8 seconds and the third row shows how the MSE values would improve for the rest of the experiment. Finally, if desired, another DPR step can be performed to load the second best model (CV1), which yields better performance as shown in the last row.

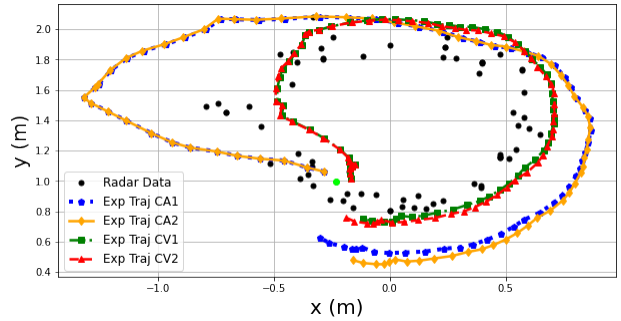


Fig. 7: Filter estimated position outputs for the four different filters for a segment of ship motion. Black dots show the raw radar returns.

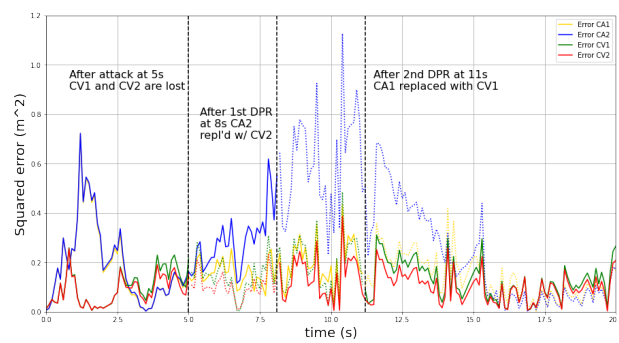


Fig. 8: Excerpt of the Mean Squared Error plot that shows the sequential reprogramming steps and the relative performance of the four different filters.

TABLE II: Mean Squared Error Comparison of Filter Outputs

	CA1	CA2	CV1	CV2
No attack	0.1238	0.2057	0.1110	0.0865
After attack at 5 s	0.1206	0.2067	-	-
After first DPR at 8 s	0.1193	-	-	0.0858
After second DPR at 11 s	-	-	0.1073	0.0838

V. Conclusion

This paper has developed a software-hardware co-design approach to adaptive CPSs in the event of cyber-attack. During

normal operation, target data from mmW radar sensor are processed by a scalable MM estimator. One key feature of the design is that the algorithm is distributed among FPGA computing units, whose number is determined by resource utilization. Each FPGA is programmed with different elemental KFs that, as an ensemble, model the target's distinct modes of behavior. Posterior probabilities of each behavioral mode are calculated on separate hardware and the elemental KFs are ranked from the likeliest one down given the target data. By tagging the best filters in each FPGA the CPS is prepared to adapt in the event of cyber-attack.

The specific cyber-attack considered renders one of the FPGA computing units inoperative. The surviving FPGA is reconfigured to adapt the MM estimator by using the most likely elemental KFs to scale down its ensemble. Partial reconfiguration is executed in stages to allow for continuity of target tracking by running some of the less likely KFs while the FPGA fabric is being re-programmed. Model descriptions and selection information must be shared outside each unit periodically in advance of lost hardware so as to be prepared for reconfiguration. The authors have addressed other types of cyber-attacks on CPSs, such as false data injection, in prior work [21], [22].

Because of the reduced computing capacity available after a cyber-attack, the system's overall performance is expected to degrade, but if the model selection process is successful it will not degrade as much as if there were no choice about which two of the four possible models remained in operation on the working computing unit. There is a smaller penalty paid to perform the partial reconfiguration of just one portion of the FPGA fabric compared to the larger penalty would be required if the entire board was to be reprogrammed instead. The demonstration illustrated that any time penalty could be offset by the increased performance after the most appropriate models are loaded onto the surviving computing node.

Future efforts with this project will include the use of an optical sensor to provide ground truth data to compare the filter output results instead of using the input data itself. A larger tank of water will be employed which will allow for more varied boat motion including straight line, constant turn rate, and constant acceleration profiles which will show more diverse performance between the candidate models. Additionally, the final goal of this work will be to include all the computing hardware on the boat itself and be able to process the data in real time including actual DPR of the FPGA fabric. The present work has demonstrated the feasibility of the proposed framework for adaptive hardware MM estimators by DPR.

References

[1] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 103–123, 1998.

[2] A. Averbuch, S. Itzikowitz, and T. Kapon, "Parallel implementation of multiple model tracking algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 2, pp. 242–252, 1991.

[3] G. Donohoe and J. Lyke, "Adaptive computing for space," in *42nd Midwest Symposium on Circuits and Systems (Cat. No.99CH36356)*, vol. 1, 1999, pp. 126–129 vol. 1.

[4] C. Kao, "Benefits of partial reconfiguration," pp. 65–67, 2005.

[5] C. Carmichael, "Correcting single-event upsets through virtex partial configuration," 2000.

[6] K. Guha, A. Majumder, D. Saha, and A. Chakrabarti, "Dynamic power-aware scheduling of real-time tasks for FPGA-based cyber physical systems against power draining hardware trojan attacks." in *The Journal of Supercomputing*, vol. 76, 2020, pp. 8972–9009.

[7] R. S. Ram, M. L. C. Prabhaker, K. Suresh, K. Subramaniam, and M. Venkatesan, "Dynamic partial reconfiguration enchanced with security system for reduced area and low power consumption," *Microprocessors and Microsystems*, vol. 76, p. 103088, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933119306180

[8] H.-G. Yeh, "Systolic implementation on kalman filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, pp. 1514–1517, 1988.

[9] C.-R. Lee and Z. Salcic, "High-performance FPGA-based implementation of Kalman Filter," *Microprocessors and Microsystems*, vol. 21, no. 4, pp. 257–265, 1997.

[10] Y. T. Lai, A. Bigdeli, and M. Biglari-Abhari, "An optimised systolic array-based matrix inversion for rapid prototyping of Kalman Filters in FPGA's," in *2004 12th European Signal Processing Conference*, 2004, pp. 2035–2038.

[11] D. Pritsker, "Hybrid implementation of Extended Kalman Filter on an FPGA," in *2015 IEEE Radar Conference (RadarCon)*, 2015, pp. 0077–0082.

[12] A. Mills, P. H. Jones, and J. Zambreno, "Parameterizable FPGA-Based Kalman Filter Coprocessor Using Piecewise Affine Modeling," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2016, pp. 139–147.

[13] M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali, and J. Henkel, "Two-state checkpointing for energy-efficient fault tolerance in hard real-time systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 7, pp. 2426–2437, 2016.

[14] M. Khludova, "Probabilistic evaluation of checkpoint-based fault tolerance in real-time systems," in *2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, 2019, pp. 1–4.

[15] Y. Xie, L. Li, M. Kandemir, N. Vijaykrishnan, and M. Irwin, "Reliability-aware co-synthesis for embedded systems," in *Proceedings. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2004.*, 2004, pp. 41–50.

[16] P. Pop, V. Izosimov, P. Eles, and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant embedded systems with checkpointing and replication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 3, pp. 389–402, 2009.

[17] Y. Bar-Shalom and D. W. Blair (Eds.), *Multitarget-Multisensor Tracking: Applications and Advances (Vol. III)*. Norwood, MA: Artech House Inc., 2000.

[18] Y. He, J. Xiu, and X. Guan, *Radar data processing with applications*. Singapore: Publishing House of Electronics Industry: Wiley, 2016.

[19] V. N. Faddeeva, *Computational Methods of Linear Algebra*. New York, NY: Dover Publications Inc., 1959.

[20] "IEEE standard for verilog hardware description language," *IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001)*, pp. 1–590, 2006.

[21] B. Croteau, D. Krishnankutty, K. Kiriakidis, T. Severson, C. Patel, R. Robucci, E. Rodriguez-Seda, and N. Banerjee, "Cross-level detection framework for attacks on cyber-physical systems," *Journal of Hardware and Systems Security*, vol. 1, no. 4, pp. 356–369, Dec 2017. [Online]. Available: https://doi.org/10.1007/s41635-017-0027-9

[22] T. A. Severson, B. Croteau, E. J. Rodríguez-Seda, K. Kiriakidis, R. Robucci, and C. Patel, "A resilient framework for sensor-based attacks on cyber–physical systems using trust-based consensus and self-triggered control," *Control Engineering Practice*, vol. 101, p. 104509, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066120301246