Low-Complexity Architecture for Cyber-Physical Systems Model Identification

Charan Kumar Vala, Mark French, Amit Acharyya, and Bashir M. Al-Hashimi, *Fellow, IEEE*

***Abstract*—We propose a low complexity architecture for cyber-physical system (CPS) model identification based on multiple-model adaptive estimation (MMAE) algorithms. The complexity reduction is achieved by reducing the number of multiplications in the filter banks of the MMAE algorithm present in the cyber component of the CPS. The architecture has been implemented using FPGA for 16, 32, 64 filter banks as part of position and velocity estimations of autonomous auto-mobile application. It has been found up to 78% reduction in multiplications is possible, which translates to the reduction of 39% LUTs, 13% FFs, 27% DSPs, and 43% power reduction when compared with the conventional architecture (without multiplications reduction) at 100MHz operating frequency. Furthermore, the proposed architecture is able to identify accurate model of auto-mobile application just within 510ns, in the presence of external disturbances and abrupt changes.**

***Index Terms*—Cyber-Physical Systems, Model Identification, MMAE, MMAC, Bank of Kalman Filters, FPGA.**

I. Introduction

MODEL identification has numerous cyber-physical system (CPS) based applications [1], [2] including autonomous auto-mobiles, adaptive estimation [3], [4], intelligent adaptive plant control [1], fault detection-isolation [5], [6]. CPS systems are often characterised by high degrees of uncertainty, and hence practical adaptive control is likely to be important for good closed loop response. This paper considers a hardware architecture for the parallel implementation of multiple model adaptive estimation (MMAE) schemes for model identification. MMAE forms a component of Multiple Model Adaptive Control (MMAC) schemes, see [7] for a complete modular analysis. Within MMAC, MMAE algorithms to identity the physical plant so that the control architecture can dynamically switch in appropriate controllers in real time. MMAE based algorithms significantly improve performance compared to contemporary designs [8], [9], particularly in the presence of uncertainties including external disturbances and abrupt changes. However, significant computational demands and massive number of filter banks required by the MMAE algorithm [7] preclude its use in resource constrained applications using embedded computing platforms with low cost or low power requirements. To the best of our knowledge, there is no reported hardware architecture for MMAE algorithms. Therefore, in this paper, we propose for the first time a low complexity hardware architecture for the computationally intensive MMAE algorithm for the CPS model identification.

The rest of the paper is organized as follows. Section-II provides the details of the proposed architecture and section-III discusses the experimental results and finally section-IV concludes the discussion.

Manuscript received August 2, 2018; revised October 25, 2018; accepted November 10, 2018. This work was supported by EPSRC, U.K., as a part of PRiME Project under Grant EP/K034448/1 and Grant EP/N509747/1. This brief was recommended by Associate Editor J. Wu. (Corresponding author: Charan Kumar Vala.) C. K. Vala, M. French, and B. M. Al-Hashimi are with the School of Electronics and Computer Science, University of Southampton, Southampton, U.K. SO171BJ (e-mail: c.k.v@ecs.soton.ac.uk; mcf@ecs.soton.ac.uk; bmah@ecs.soton.ac.uk). A. Acharyya is with the Department of EE, Indian Institute of Technology Hyderabad, Hyderabad 500050, India (e-mail: amitacharyya@iith.ac.in). Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org. Digital Object Identifier 10.1109/TCSII.2018.2881481 DOI: 10.5258/SOTON/D0712

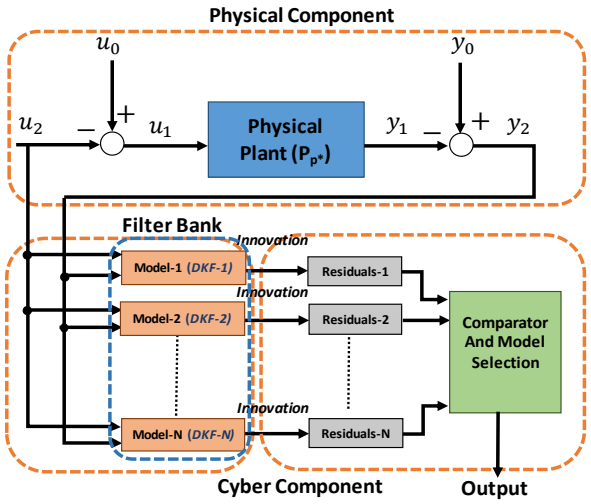

Fig. 1: Proposed architecture for MMAE based model identification. Each model represents the deterministic Kalman filter with particular state space matrix.

II. Proposed Architecture

In general CPS is an integration of computation with physical processes [1], [2] and it is represented by two components: Physical and Cyber. The proposed architecture shown in Fig. 1, physical component comprises the physical plant (P_{p^*}) and Cyber-component comprises the hardware architecture of the MMAE based model identification algorithm. The Cyber-component measures the signals $(u_2, y_2)^T$ from the physical component and processes these signals for identifying the model of the physical plant.

A. Physical Component

Physical plants (P_{p^*}) are represented as discrete-time linear time-invariant (LTI) system in the form

$$\begin{aligned} X(k+1) &= AX(k) + Bu_1 \\ y_1(k) &= HX(k) \end{aligned} \tag{1}$$

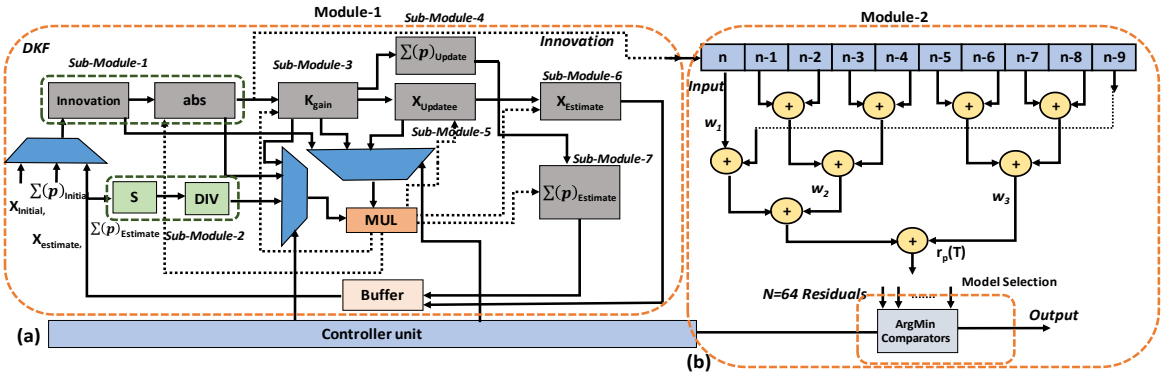

Fig. 2: Proposed Multiple model estimation based model identification architecture (a) Deterministic Kalman filter architecture, (b) Residuals, ArgMin, Comparator and Model selection.

Where $(A, B, H) \in \mathbb{R}^{nxn}$ x $\mathbb{R}^{nx1}$ x $\mathbb{R}^{1xn}$ are state-space matrices, state X is $\in \mathbb{R}^{nx1}$, and $(u_2, y_2)^T$ are the measured signals, $(u_0, y_0)^T$ are the external disturbances and $(u_1, y_1)^T$ are the original signals at the input and output side of a physical plant (P_p) respectively.

B. Cyber-Component

The cyber-component represents the hardware architecture of MMAE algorithm where major functional building blocks are the bank of Kalman filters, residuals and model selection. In the MMAE Algorithm-1, a model plant-set defined such that the true plant P_{p^*} lies with a set of N 'candidate' plants $P_{p^*} \in \{P_1, ..., P_N\}$, $1 \leq p \leq N$. Model identification is performed on the true plant P_{p^*} for robust estimation. In this regard, each model-plant runs one computationally intensive Deterministic Kalman Filter (DKF) in the presence of external disturbances. Thus the complete model identification comprises of N DKF modules where each DKF processes $(u_2, y_2)^T$ at every time instance to provide an estimated outcome. Then the difference between the estimated and the measured output is computed (known as $Innovation$) at every time instance for each model-plant. Next step is the computation of the weighted sum of m-number of such $innovations$ per model plant known as $Residuals$. The minimum $Residual$ among such N model-plants represents the best match of the physical plant P_p (known as Model Selection). The fore-mentioned algorithm is shown in the form of pseudo-code in Algorithm-1. The total number of multiplications involved in MMAE algorithm are equal to the $(1+4n+3n^2+2n^3)N$ where N, n represents the total number of models and states respectively. The detailed comparison of number of multiplications involved in MMAE algorithm and proposed hardware architecture are given in Table-1. Most of the multiplications conferred in DKF. This is evident that, N-DKF modules contains significant number of multiplications and this brief proposes an architecture to eliminate most of these, there by making it low-complexity.

The hardware mapping of MMAE algorithm is shown in Fig. 2. It comprise two modules one containing DKF (line no: 1-22 in Algorithm-1) with $Innovation$ computation (line no: 5 in Algorithm-1) and the other one involving $Residual$ computation (line no: 23, 24 in Algorithm-1) along with model selection (line no: 25-29 in Algorithm-1, where T_k truncation up to time step k). These modules are pipelined to achieve higher throughput. The controller block is designed to monitor the data flow between the intra sub-modules of module-1 and module-2.

TABLE I: Comparison between the MMAE algorithm (Shown in Algorithm-1) and proposed hardware modules in terms of the number of multiplications. Where Sub-module 1-7 are associated with DKF, N= Number of models, n= number of states to be estimated, No. of mul=Number of multiplications, H.W.A= Hardware architecture.

Sub-Module	No. of mul in MMAE algorithm	No. of mul in proposed H.W.A
(1) $Innovation$	Nn	0
(2) S	Nn^2	0
(3) $Kalman\ Gain$	Nn^2	Nn
(4) X_{update}	Nn	Nn
(5) P_{update}	Nn^3	Nn^2
(6) $X_{Estimate}$	$N(n^2+n)$	0
(7) $\sum(P)_{Estimate}$	$N(n^3+n)$	0
(8) $Residuals$	N	0
$Total$	$(1+4n+3n^2+2n^3)N$	$(2n+n^2)N$

1) Deterministic Kalman Filter (DKF): The inputs of DKF module are $A(i)$, $B(i)$, $H(i)$ with initial conditions $X_{Initial}$ (state variables), $\sum(P)_{Initial}$ (covariance) and y_2. The DKF module computes innovation, Kalman gain, update and estimation of state variable (X), covariance $\sum(P)$. As shown in Algorithm-1 from line 1-22, all these computations involves in matrix multiplications, inversion, additions and subtractions. Since $A(i)$, $B(i)$, $H(i)$, are constant matrices, the computation of innovation, $X_{Estimate}$, $\sum(P)_{Estimate}$ are performed by using addition and shifting operations. Thereby the use of multipliers eliminated completely from the computation of these steps. The Kalman gain, X_{update}, $\sum(P)_{update}$ computations majorly depend on the $innovation$, $X_{Estimate}$, $\sum(P)_{Estimate}$ matrices, these matrices changes at every iteration. Therefore these computations should be performed by using the multipliers. The term BB^T conferred in the $\sum(P)_{Estimate}$ is pre-computed and reused in every iteration of DKF, by that we are able reduce significant number of multipliers. The detailed reduction in number of multiplications at each step of MMAE are given Table-1. The original MMAE algorithm requires $(1+4n+3n^2+2n^3)N$ number of multiplications. However in the proposed archi-

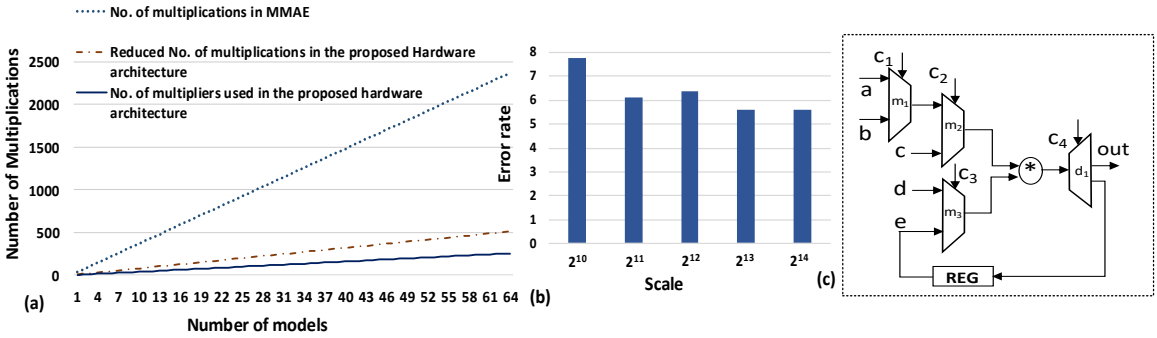

Fig. 3: Multiplications reduction using the proposed hardware architecture: comparison for 1 to 64 models (Note: Number of multipliers used in proposed architecture are even less than the number of multiplications required), (b) Error rate comparison for various scaling factors and (c) Reusable multiplier architecture

Algorithm 1 model adaptive estimation (MMAE) based model identification

1: **Kalman Filter:**
2: INPUT: $State\ Space\ Matrix\ of\ LTI\ System\ A(i),\ B(i),\ H(i),$ $X_{Initial}, \sum(P)_{Initial}\ and\ y_2$
3: **for** i =1 to N **do**
4: **Sub-module 1**
5: $Innovation = y_2 - H.X_{estimate}(i)$
6: $abs(Innovation(i))$
7: **Sub-module 2**
8: $S(i) = H.\sum(P)(i).H^T$
9: $S^{-1}(i) = Scale/S(i)$
10: **Call Residuals**
11: **Kalman Gain:**
12: **Sub-module 3**
13: $K(i) = \sum(P).H^T.S^{-1}(i)$
14: **Sub-module 4**
15: $X_{update}(i) = X_{estimate}(i) + K(i).Innovation(i)/Scale$
16: **Sub-module 5**
17: $\sum(P)_{update}(i) = \sum(P)(i) - K(i).H.\sum(P)(i)/scale$
18: **Sub-module 6**
19: $X_{estimate}(i) = A(i).X_{update}(i) + B(i).U$
20: **Sub-module 7**
21: $\sum(P)_{estimate}(i) = A(i).\sum(P)_{update}(i).A^T(i) + B(i).B^T(i)$
22: **end for**
23: **Residuals:**
24: $r_p(k)(i) = \sum_{t=T-l}^{T} \|Innovation\|_{[HPH^T+1]^{-1}}, T \geq l$
25: **Model Selection:**
26: **for** j =1 to N **do**
27: $P_{qf} = \underset{p\in P}{\operatorname{argmin}}(\min_{(u_1^p, y_1^p)^T \in T_k M_p} \|T_k(u_2, y_2)^T + T_k(u_1^p, y_1^p)^T\|)$
28: **end for**
29: OUTPUT: Identified Model

tecture, by using shifting, addition and pre-computation, the total number of multiplications reduced to $(2n + n^2)N$ and detailed comparison shown in Fig. 3(a). For $N = 64$ with $n = 2$ the total number of required multiplication observed in MMAE algorithm are 2368 and the proposed hardware architecture reduced them to 512 and approximately 78% reduction is observed. Due to the data dependency between the sub-modules of DKF, the total operation of DKF is performed in 7 clock cycles, enabling us to reuse the multipliers and thereby total number of multipliers are further reduced from $(2n + n^2)N$ to n^2N. In this hardware architecture for 64 number of models with $n = 2$, we have used only 256 multipliers. This yields 50% reduction in terms of multipliers, on the hardware architecture.

As shown in Fig. 2(a), DKF module comprises of seven sub-modules. In the first sub-module $Innovation$ and its absolute values are computed. This consists of subtraction and multiplication of two matrices (H and $X_{estimate}$) (line no: 5 in Algorithm 1). However, since H is constant and can be pre-computed, hence the multipliers are eliminated and entire matrix multiplication boiled down to simple additions and shifting. For example, considering h_1x_1 where $h_1 = 13$, multiplication can be removed by expressing the equation as

$$\begin{aligned} h_1x_1 = 13x_1 &= (8x_1) + (4x_1) + (1x_1). \\ &= 2^3x_1 + 2^2x_1 + 2^0x_1 \\ &= ls(x_1, 3) + ls(x_1, 2) + x_1 \end{aligned} \quad (2)$$

where $ls(a, b)$ signifies a left-shifted by b bits. In this example, the multiplier is replaced by just three adders and two shifters. These shifters can be implemented by simple hardware wiring.

In the second sub-module S and S^{-1} (line no: 8, 9 in algorithm-1) are computed. This consists of multiplication of three matrices ($H.\sum(P)(i).H^T$). This is also performed by reusing the same structure discussed in first sub module and explained further using equation-1. In general S^{-1} computation requires a division that increases the hardware complexity. For instance, in a Xilinx FPGA, fixed-point addition takes one cycle, whereas a single precision floating-point adder would require 14 cycles while using one order of magnitude more resources for the same number of bits. Therefore, here we have used a LUT based method. However, to retain the precision of this LUT based division, we up-scaled the numerator first and then use the denominator as the address to fetch the appropriate data from the LUT. Scaling factor was decided based on the empirical simulations running for various scaling factors and the error rates are shown in Fig. 3(b). For the scaling factor 2^{11}, 2^{12}, 2^{13}, and 2^{14} the error rates 7.78, 6.12, 6.36, 5.59, and 5.58 respectively observed. As the scaling factor increases, the size of LUT also increases, in this work optimally we have chosen 2^{13} as scaling factor.

TABLE II: Reusable multiplier operation. kg=Kalman gain, AI=abs(Innovation) and SM=Sub module OTM=Output to the sub-module $in_1 = \sum(P)H^T$ and $in_2 = H\sum(P)$

Inputs to multiplier					Control Bits				OTM
a	b	c	d	e	c_1	c_2	c_3	c_4	
S^{-1}	-	-	in_1	-	0	0	0	1	$SM3$
-	-	AI	-	kg	-	1	1	0	$SM4$
-	in_2	-	-	kg	1	0	1	0	$SM5$

In the third sub-module, by using n reusable multipliers, Kalman gain (line no: 13 in Algorithm-1) is computed and corresponding results were stored in registers. The architecture for reusable multiplier is shown Fig 3(b). Where a, b, c, d, e are the inputs of multiplexer (MUX) and c_1, c_2, c_3, c_4 are the control bits to the MUX. To compute Kalman gain, the inputs to reusable multiplier are S^{-1} and $\sum(P)H^T$, the outcome of multiplier is stored in a register and in the next cycle it is used ot compute the X_{update}. Fourth sub-module involves the computation of X_{update} (line no: 15 in Algorithm-1). This requires n multipliers and shifters for the computation of $K * Innovation/Scale$. However the multipliers used in third sub-nodule are reused here and right shifters are implemented by wiring. Outcome of these n multipliers is added to the $X_{estimate}$ to obtain X_{update}. It is to be noted the division with scale can performed by the right shifting of numerator. Fifth sub-module involves computation of $\sum(P)_{update}$ (line no: 17 in Algorithm-1). The term $K*H*\sum(P)/scale$ requires n^2 multipliers that is computed by re using the n multipliers from third sub-module and n other multipliers. Then computed results are subtracted from $\sum(P)$ to obtain final $\sum(P)_{update}$ value. Overall DKF module uses only n^2 multipliers and the data path of these multipliers supervised by the controller. The detailed operation of reusable multiplier is given in Table-II.

The sixth and seventh sub-modules involves computation of $X_{Estimate}$ and $\sum(P)_{Estimate}$ respectively. Since state space matrix A is constant, the term $\sum(P)_{update}(i).A^T(i)$ (line no: 21 in algorithm-1) is computed using the shifting operations like as explained in first sub-module's discussion and equation (1). Since state space matrix B is constant, the term BB^T (line no: 21 in algorithm-1) is computed once and the same results are reused for computing $\sum(P)_{Estimated}$ in every iteration of DKF. Finally $X_{Estimate}$ and $\sum(P)_{Estimated}$ were computed using addition and shifting operation and these results are stored in registers for providing input to DKF in next iteration. In the Similar fashion 64-DKF modules were designed for each state space matrix set $A(i)$, $B(i)$, $H(i)$ where $1 \leq i \leq 64$. The 64 absolute values of innovations are stored in a buffer and send to module-2 for selecting appropriate model.

2) Residuals and Model Selection: The architecture shown in Fig. 2(b) was used for computing the residuals and model selection (line no: 23-29 in algorithm-1). This module takes input ($Innovation$) from DKF module and computes the final model which is close to the physical plant. The operation of residual sub-module is explained as follows:

Initially innovations from N-DKF modules are forwarded to the N-residual sub-modules (Fig. 2(b)). Each residual sub-module stores these innovations in their respective shift registers. At each clock cycle these innovations are shifted their location by one place. Once the shift registers filled by m innovations it starts computation of residual. In this interval the model selection block sends the default identified model as 1 and after filling all the shift registers, each value in the shift register multiplied with weights (w_1, w_2, w_3) as shown in Fig. 2(b) and then added together for getting final residual. To reduce hardware complexity the weights w_1, w_2, w_3 are chosen as multiples of two such that the residuals can be computed just by left shifting the innovations. This module comprises of shift register, adders and shifting operations. Subsequently N-residuals are forwarded to $argmin$ (Fig. 2(b)) module for computing the minimum location among all. This $argmin$ sub-module is a pipelined architecture with $\log_2 N_{max}$ stages of comparators organized in tree structures.

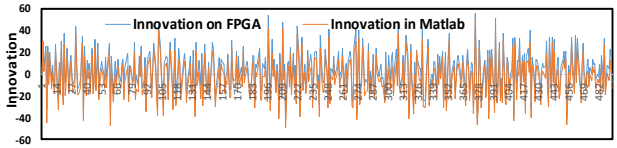

Fig. 4: Innovation comparison between the FPGA and Matlab computations.

III. Results and Discussion

The proposed architecture has been coded in Verilog, synthesized using Xilinx′s Vivado 2017.4 and prototyped on Virtex ultra scale plus FPGA (VCU118).

For the validation purpose of this prototype, we considered position and velocity estimation models for autonomous automobile application- one of the major applications in CPS [10]. The state space equations of estimated position and velocity defined as

$$P_{k+1} = P_k + T.V_k + 1/2.T^2.u_k + 1/2.T^2.D_e + Noise \quad (3)$$

$$V_{k+1} = V_k + T.u_k + T.D_e + Noise \quad (4)$$

respectively where $P = position$, $V = velocity$, $k = time$, $T = time\ constant$ u_k= $Input\ acceleration$, D_e= $External\ disturbances$.

In this design we have considered 64 uncertainty levels such that $D_{external}$, $Noise \in \left[-31.3, 68.5\right] \cup \left[-5.64, 6.05\right]$ and corresponding state space models with matrix set $A(i)$, $B(i)$, $H(i)$ were generated where $1 \leq i \leq 64$. Based on these models, 64 bank of DKFs are designed on the hardware. Physically measured data (y_2 in $Algorithm-1$), emulated using MMAE Algorithm-1 comprising of 64 models as discussed in the last section, were generated in Matlab and combined with random noise and abrupt changes. These emulated data were stored in the BRAM of FPGA and fed to the design at every clock cycle. The proposed architecture was designed using 32 bit word length and it was found to be able to identify the physical plant model by processing the system's input and output in the presence of external disturbances. Vivado integrated logic analyser (ILA) was used for verifying the obtained results from the proposed architecture.

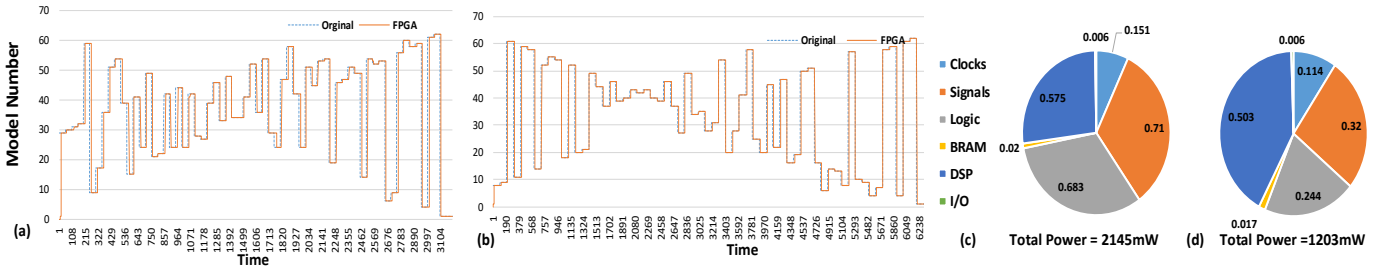

Fig. 5: Identified model by the proposed architecture when models switched for every (a) 3000ns (b) 6000ns, FPGA Power consumption results for 64 number of models with 32-bit word length (c) Conventional architecture (d) Proposed architecture

TABLE III: Module wise breakdown of the FPGA resource utilization of the proposed architecture

Module	LUT	FF	DSP
DKF	847	887	16
Residuals	152	0	0
Comparator	36	40	0
Division	40	0	0
Model Selection	10	10	0

A. Performance

For the Cyber-space performance metrics are accurate model identification in the presence of uncertainty and time taken to identify the model after physical plant changed its model. As discussed in section-II, model identification considerably depend on $Innovation$ values (line number-5 in Algorithm-1). Therefore $Innovation$ (which is output of DKF module) results are shown in Fig. 4. It is observed that the Matlab and FPGA results match closely with each other. In order to assess the performance of proposed architecture in-terms of model identification we consider, 2 case studies here, when emulated physical plant changes at (a) 3000ns, (b) 6000ns and corresponding results were shown in Fig. 5(a) and (b) respectively. To illustrate, considering at 254^{th} time instant in Fig. 5(a), physical plant changes its model from 59 to 9 (shown in dotted blue) which is accurately identified by the FPGA prototype (shown in brown). Similarly the physical plant, as shown in Fig. 5(b), changes the model from 58 to 14 at 604^{th} time instant and it was also accurately identified by the prototype. The proposed architecture is able to identify the models in just 510 ns between $1 \leq N \leq 64$ corresponding to the changing physical plant.

B. Hardware Resource Utilization

1) FPGA: The breakdown of the FPGA resource utilization of the proposed architecture is presented in Table-III. Among the all modules, 78% of LUT's, 94% of FF's and 100% of DSP's are utilized by DKF. This is evident to say that DKF is the computationally intensive. In order to assess the variation of resource consumption with respect to model size and resource utilization shown in Table-IV. It can be observed that the resource utilization increases with respect to model size for a fixed word length. However, for the data fetching to the FPGA and corresponding validation required Vivado ILA to be incorporated that in turn would consume 62328 (5.27%) of lookup table (LUT), 63070 (2.67%) of Flip-Flops, 1024 (14.97%) of DSP's and 28 (1.3%) Block RAM Tile on Virtex ultra scale plus FPGA (VCU118) at 100MHz operating frequency. To show the performance improvement of proposed low-complexity architecture, since there is no reported architecture present, the MMAE algorithm also implemented using conventional architecture (direct mapping of MMAE algorithm to the hardware) it is also 1^{st} of its kind, which results in 100813 LUT's, 69980 FF's and 1408 DSP's resource utilization for 64 number of models with 32-bit word length. It can be noticed that the proposed low-complexity architecture improved by 39%, 13% and 27% of LUT's, FF's, DSP's resource utilization respectively when compared with the conventional architecture, detailed utilization with respect to various bank of filter models is given in Table-IV. The utilization percentage with respect to the total available resources on VCU118 also give in same table.

2) ASIC: To give an insight into the low power consumption of the proposed architecture and to place the proposed complexity reduction methodology in the context of CPS, ASIC implementation has also been carried out using 65nm technology using Synopsis Design Compiler. Total synthesized cell area of the design is 3.04 mm^2 and power consumption is 737μW @ 1 MHz frequency

3) Power: The detailed breakdown of FPGA power consumption using the Xilinx power analyser between the conventional (Fig. 5(c)) and proposed low-complexity (Fig. 5(d)) architectures are shown in Fig. 7. Proposed low-complexity architecture showed that 64%, 54%, 24%, 12% power consumption reduction in logic, signals, clocks, and DSP respectively. The over all power consumption of proposed architecture reduced by 43% when compared to the conventional architecture, for 64-bank of filters with 32-bit word length.

IV. CONCLUSION

MMAE based model identification will play an important role in CPS. However, the intense computational demands imposed by MMAE algorithm precludes its use in resource constrained CPS applications. Therefore to make MMAE suitable for resource constrained CPS applications, a low-complexity architecture was introduced in this brief. It is to be noted that proposed architecture is generic and scalable to any number of models. The proposed architecture was

TABLE IV: Resource utilization comparison between the Proposed architectures with respect to the model size. Conventional = Direct mapping of MMAE algorithm without reduction in multiplications, Proposed = Low-complexity architecture, % percentage with respect to the total available resources on VCU118-FPGA

Model Size (N)	LUT		FF		DSP		Power in mW	
	Conventional	Proposed	Conventional	Proposed	Conventional	Proposed	Conventional	Proposed
16	26019 (2.2%)	15760 (1.3%)	22292 (0.94%)	16544 (0.69%)	352 (5.4%)	256 (3.9%)	582	329
32	50942 (4.3%)	28593 (2.4%)	39244 (1.6%)	30601 (1.2%)	704 (10.8%)	512 (7.9%)	1104	613
64	100813 (8.5%)	60989 (5.15%)	69980 (2.9%)	60860 (2.5%)	1408 (21.7%)	1024 (15.8%)	2145	1203

implemented on Virtex ultra scale plus FPGA and achieves an improvement of 39% LUT's, 13% FF's , 27% DSP's and 43% power consumption respectively when compared the conventional architecture.

REFERENCES

[1] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*. IEEE, 2008, pp. 495–500.

[2] I. Ruchkin, S. Samuel, B. Schmerl, A. Rico, and D. Garlan, "Challenges in physical modeling for adaptation of cyber-physical systems," in *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*. IEEE, 2016, pp. 210–215.

[3] V. Hassani, A. P. Aguiar, M. Athans, and A. M. Pascoal, "Multiple model adaptive estimation and model identification usign a minimum energy criterion," in *American Control Conference, 2009. ACC'09.* IEEE, 2009, pp. 518–523.

[4] B. Chen, L. Yu, W.-A. Zhang, and A. Liu, "Robust information fusion estimator for multiple delay-tolerant sensors with different failure rates," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 2, pp. 401–414, 2013.

[5] B. Pourbabaee, N. Meskin, and K. Khorasani, "Sensor fault detection, isolation, and identification using multiple-model-based hybrid kalman filter for gas turbine engines," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1184–1200, 2016.

[6] P. Lim, C. K. Goh, K. C. Tan, and P. Dutta, "Multimodal degradation prognostics based on switching kalman filter ensemble," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 1, pp. 136–148, 2017.

[7] D. Buchstaller and M. French, "Robust stability for multiple model adaptive control: Part i the framework," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 677–692, 2016.

[8] J. Fu, T. Chai, Y. Jin, and C.-Y. Su, "Fault-tolerant control of a class of switched nonlinear systems with structural uncertainties," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 2, pp. 201–205, 2016.

[9] R. Sakthivel, S. Mohanapriya, H. R. Karimi, and P. Selvaraj, "A robust repetitive-control design for a class of uncertain stochastic dynamical systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 427–431, 2017.

[10] D. Simon, "Kalman filtering," *Embedded systems programming*, vol. 14, no. 6, pp. 72–79, 2001.