

Chapter 2

Implementation of Kalman Filter to Monitor the Level Fluctuations in a Dam Using FPGA

K. Shashank, Nitin Ravi, M. Rakshith and J. V. Alamelu

Abstract In this paper we study the design, implementation and evaluate the performance of a Kalman filter using FPGA. It is essential to be familiar with minimum mean square error filtering and state space methods. It is important that the set of equations, their relevance to one another and indeed the overall functionality of the algorithm that defines the Kalman filter require complete understanding. The filter will be implemented with field programmable gate arrays (FPGA), to monitor the level fluctuations for a dam/reservoir.

Keywords Kalman · FPGA · State space method · Mean square error filtering

2.1 Introduction

The Kalman Filter is a means to predict the future behavior of a system based on past behavior. A system's past behavior is, in a way, remembered and used along with measurements to make the predictions of how the system might behave in the future.

K. Shashank (✉) · N. Ravi · M. Rakshith · J. V. Alamelu
Department of Instrumentation Technology, M.S. Ramaiah Institute of Technology,
Bangalore, India
e-mail: shashbeckmmm@gmail.com

N. Ravi
e-mail: nitinravi1@gmail.com

M. Rakshith
e-mail: rakshith_swk@yahoo.co.in

J. V. Alamelu
e-mail: jvalamelu@gmail.com

According to the paper published by Kleeman (1995) the reason that mathematical models such as the Kalman Filter are useful to a designer is because virtually all systems are non-deterministic. In other words, few if any systems are devoid of randomness or stochastic behavior. Whether a system contains stochastic processes or the environment that may act upon a system is itself stochastically governed is non-deterministic [1].

A DSP (Digital Signal Processor) processor on the other hand is a normal processor optimized for faster floating point calculations to aid in signal processing without much modification. Preferably FPGA is chosen when aimed to test/simulate. Current DSP's have one two MAC (Multiply Accumulator) units. In our summary of results for Kalman Filtering we draw heavily upon the work of Dan Simon [2].

These units are used sequentially. If one needs more than two MAC's (for example, over 100 tap FIR filter with sample rate of 200 MHZ) then parallel MAC's with single cycle computation is possible to realize only using FPGA's with current trends.

Why FPGA and not ASIC?

- Integrated circuit costs are rising aggressively
- ASIC complexity has lengthened development time
- R&D resources and headcount are decreasing.
- Revenue losses for slow time-to-market are increasing.
- Financial constraints in a poor economy are driving low-cost technologies.

These trends make FPGA's a better alternative than ASIC's for a larger number of higher-volume applications than they have been historically used for, to which the company attributes the growing number of FPGA design starts.

The paper is explained under the following topics:

- Kalman filter.
- Design optimization.
- Application and Outcome of the project.
- Block diagram for implementation.
- Results and discussions.
- Advantages of designing the model.

2.2 Kalman Filter

The Kalman filter equations are a set of mathematical equations that provide an efficient computational means to estimate the state of a process, in a way that minimizes the mean of the squared error [3]. The filter is a very powerful device as it supports the estimation of past, present and future states. It even extends its functionality so it can carry out this procedure when the precise nature of the modelled system is unknown. The system may or may not be subjected to a series of random disturbances, when this occurs it is required to estimate the state variables from noisy observations. The Kalman filter takes inaccurate, incomplete and noisy data

combined with environmental disturbances beyond a designer's control and over time develops an optimal estimate of desirable quantities [3]. The Kalman Filter is a means to predict the future behaviour of a system based on past behaviour. A system's past behaviour is, in a way, remembered and used along with measurements to make the predictions of how the system might behave in the future.

The filter estimates its process by using a form of feedback control, as implied in the previous section. The filter will estimate the process state at some time and then obtains its feedback in the form of noisy measurements. These equations fall into the category of either Time update equations or measurement update equations.

$$X_k = AX_{k-1} + Bu_{k-1} + w_{k-1} \quad (2.1)$$

$$z_k = HX_k + v_k \quad (2.2)$$

2.2.1 Time Update Equations

The time update equations are used to predict the current state and covariance matrix, used in time $t + 1$ to predict the previous state. These equations can be generally seen as predictor equations as they are responsible for projecting forward in time. K is representative of the time step, so the time update equations are basically indicative of $K + 1$.

2.2.2 Measurement Update Equations

The measurement equations are responsible for feedback and for correcting the errors that have been made in the time update equations [3]. In a sense they are back propagating to get new values for the prior state to improve the guess for the next state. These equations can be seen as corrector equations and the final estimation algorithm resemble that of a predictor corrector algorithm. So by definition measurement equations adjust the projected estimate by an actual measurement at that time.

TIME UPDATE (PREDICT):

$$\hat{x}_{-k} = A\hat{x}_{k-1} + Bu_k \quad (2.3)$$

$$P_{-k} = AP_{k-1}A^T + Q \quad (2.4)$$

MEASUREMENT UPDATE (CORRECT):

$$K_k = P_{-k}H^T(HP_{-k}H^T + R)^{-1} \quad (2.5)$$

$$\hat{x}_k = \hat{x}_{-k} + K_k(z_k - H\hat{x}_{-k}) \quad (2.6)$$

$$P_k = (I - K_kH)P_{-k} \quad (2.7)$$

P_k	Prior Error Convergence
K	Kalman Gain
z_k	State Measurement
\hat{x}	Posterior State Estimate
R_k	Measurement Error Covariance
Q_k	Random White Noise
A_k	Variable
B_k	Variable
μ_k	Control Variable
H_k	Matrix—valued Function

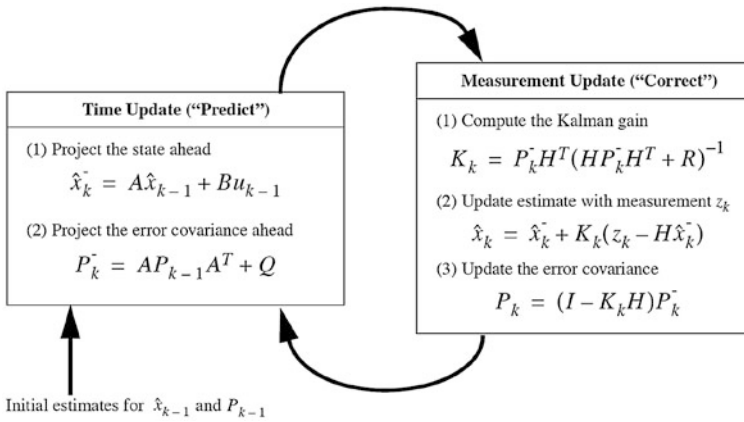


Fig. 2.1 Operation of discrete Kalman filter

Above set of equations provide a definition of variables in the Kalman filter equations. In the equations, a measurement of the process, Z_k and X_k are previously defined by linear stochastic difference equations. For practical examples, process noise covariance Q and measurement noise covariance R matrices, might change with each time step or measurement. However for the purposes of our project, we have assumed them to be constant values. A is an n by n matrix in the difference equation and relates the state at the previous time step $k - 1$ to the state at the current time step k , without the presence of process noise. Once again A is assumed to be fixed despite the fact that this would more realistically be susceptible to change with each time step. Matrix B relates the control variable to the state x . Matrix H relates the state to the measurement Z_k (Fig. 2.1) [4].

The primary goal of this design is to maximize speed or throughput or drive through as much as possible. As a secondary goal, minimization of area and power will also be considered (Fig. 2.2).

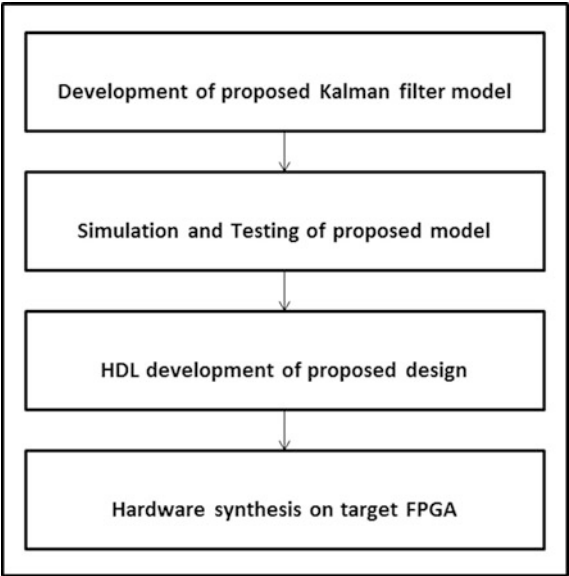


Fig. 2.2 Flow chart of methodology

2.3 Design Optimization

Design optimization can be accomplished in several ways depending on what type of optimization is required [5]. For the Kalman filter described by us, optimization for speed is most critical. Parallelization and pipelining are two methods used to help create a hardware design that fulfils this requirement. These optimizations are studied in order of priority as they relate to our system design. The designed model proposes a system that allows a designer to greatly reduce the time needed for

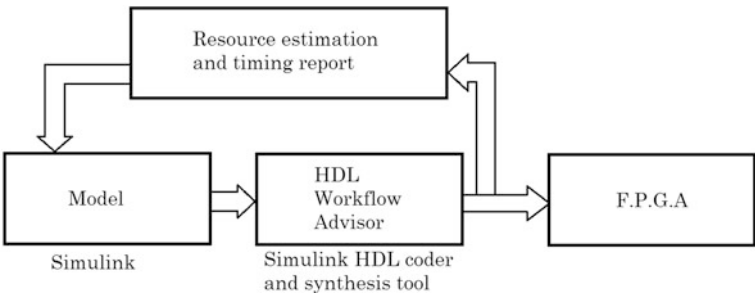


Fig. 2.3 Block diagram to convert MATLAB to Verilog

design, verification, and testing; as such, the risk factor or the rate of failure is also reduced tremendously. Specifications for a design requiring or benefiting from the use of a Kalman Filter can be entered into the system and an efficient and optimized hardware description suitable for implementation on an FPGA (Field Programmable Gate Array).

The code flexibility means that parameters are adjustable allowing for experimentation of various combinations as to optimize or tune the algorithm for different applications (Fig. 2.3).

Model-Based Design is a process that enables faster, more cost-effective development of dynamic systems. In Model-Based Design, a system model is at the centre of the development process, from requirements development, through design, implementation, and testing [6]. The model is an executable specification that is continually refined throughout the development process. After model development, simulation shows whether the model works correctly.

2.4 Application and Outcome of the Project

The Kalman filter removes noise by assuming a pre-defined model of a system. Therefore, the Kalman filter model is meaningful.

The Kalman filter model can be set up in all dams across the world. The outcome of this project is complete automation of dam operations such as the opening and closing of the gates based on the set level values. Leakage or any irregularity in functioning of the dam can be detected by comparing measured value with the estimated level value.

This technique aims at estimating the level of water in the tank, which is unknown. The measurements obtained are from the level of the float. This could be an electronic device, or a simple mechanical device

The water could be:

1. Filling, emptying or static (i.e., the average level of the tank is increasing, decreasing or not changing).
2. Sloshing or stagnant (i.e., the relative level of the float to the average level of the dam or reservoirs changing over time, or is static) [7].

2.5 Block Diagram for the Implementation

2.5.1 Case Study with Reference to KRS Dam (Mysore)

We will measure the level changes with respect to the change in flow level. We will also measure if any loss of water occurs due to leakage. We will present a look up table (LUT) with the parameters of the dam such as inlet capacity, outlet capacity, level (storage capacity) (Fig. 2.4).

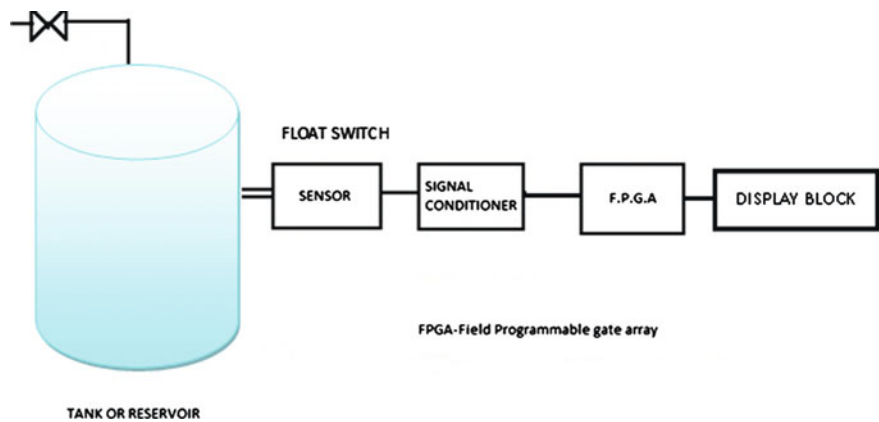


Fig. 2.4 Block diagram of the prototype model

2.6 Project Diagram

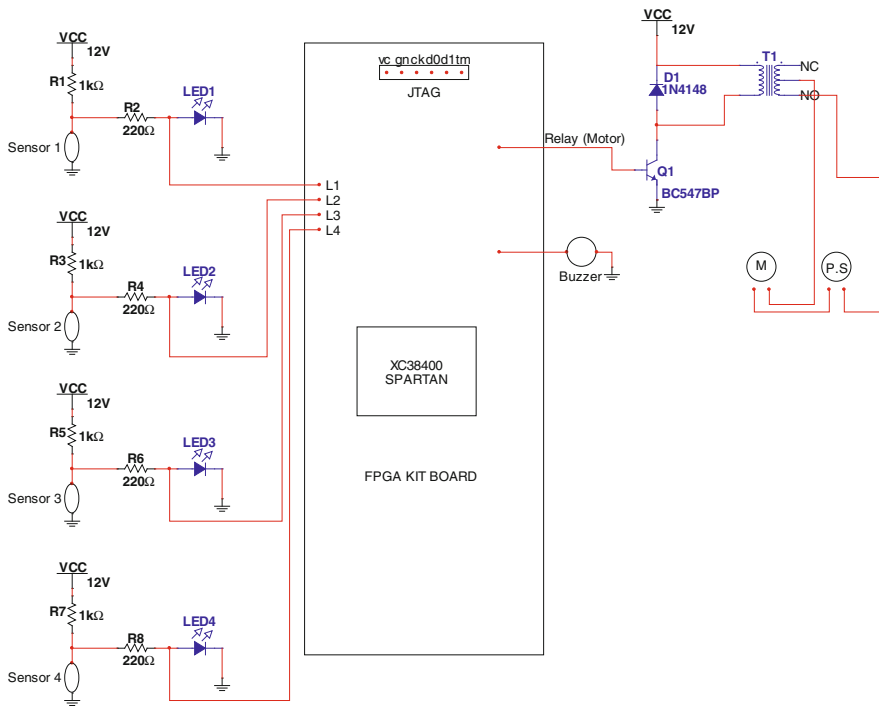


Fig. 2.5 Level sensors connected to motor, buzzer through FPGA board

2.7 Results and Discussions

This project is done keeping in mind its implementation in dams such as KRS dam Mysore. The height of the dam is 124.8 ft. The operations are completely manual. This project is completely automated i.e. an official can monitor the level readings from an enclosed room (control room) avoiding the risk of manually noting the readings (Fig. 2.5).

Level sensors were used dividing the total height into 4 levels, each representing a particular level (namely 30, 60, 90 and 120 ft) for easier monitoring [8].

This project also constitutes predict and updating the level of water through Kalman filter in FPGA and driving a buzzer and a motor through the algorithm. A buzzer is mainly used as an alarming mechanism to alert officials in the control room regarding water level rise and taking necessary safety steps. A motor is used as gate mechanism. When overflow level is reached, gates (motor) will automatically be switched on reducing the possibility of human delay and error in operations. The shortcomings of the present operational mechanism is that the level readings are noted down manually only twice a day (0630 and 1,830 hrs). This type of manual level monitoring is very risky during rainy seasons. This project being automated reduces the element of risk involved.

For the first test, the true level of the dam or reservoir is $L = 1$ is assumed. Initialization of the state with an arbitrary number, with an extremely high variance as it is completely unknown: $x_0 = 0$ and $p_0 = 1,000$. If initialized with a more meaningful variable, a faster convergence will be obtained. The chosen system noise will be $q = 0.0001$, assuming that an accurate model is acquired.

Predict:

$$x_{1/0} = 0 \quad (2.8)$$

$$p_{1/0} = 1000 + 0.0001 \quad (2.9)$$

The hypothetical measurement, $y_1 = 0.9$ (due to noise) is obtained.

A measurement noise of $r = 0.1$ is assumed.

Update:

$$K_1 = 1000.0001(1000.0001 + 0.1)^{-1} = 0.9999 \quad (2.10)$$

$$x_{1/1} = 0 + 0.9999(0.9 - 0) = 0.8999 \quad (2.11)$$

$$p_{1/1} = (1 - 0.9999)1000.0001 = 0.1000 \quad (2.12)$$

So Step 1, the initialization of 0, has been brought close to the true value of the system. Also, the variance (error) has been brought down to a reasonable value (Figs. 2.6, 2.7, 2.8, 2.9, 2.10).

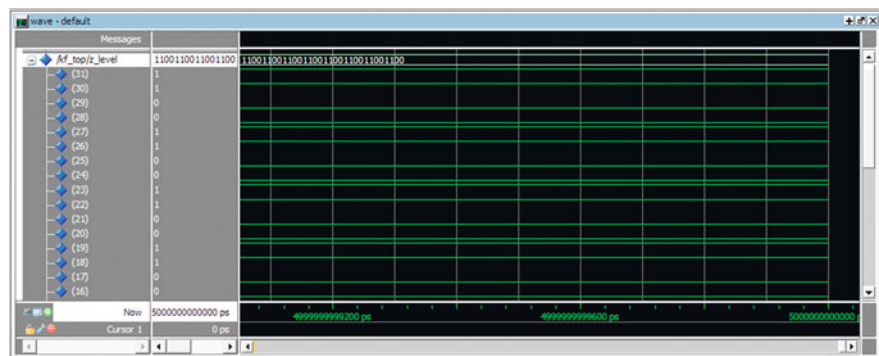


Fig. 2.6 Intial level input [bit 31–16]

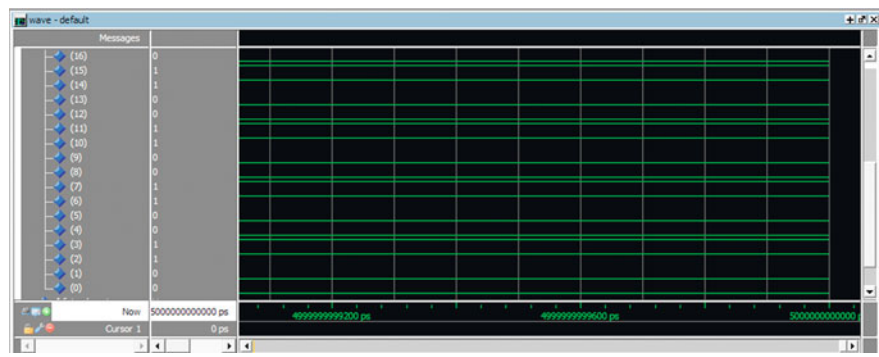


Fig. 2.7 Intial level input [bit 15–0]

Here the initial level input is
 $Z_level = 11001100110011001100110011001100$

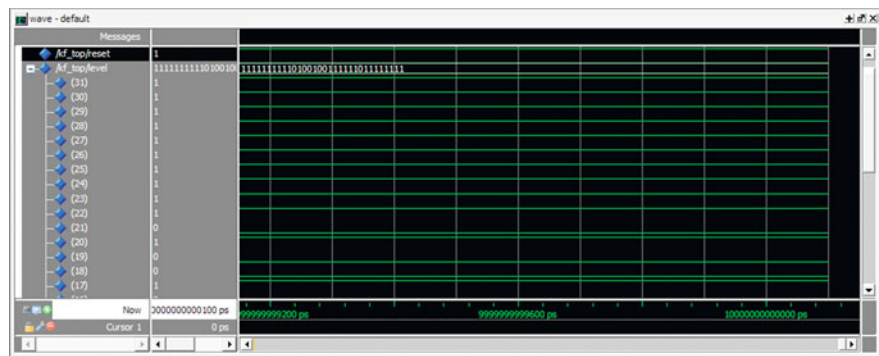


Fig. 2.8 Predicted level value [bit 31–16]

The predicted level value is
level = 11111111110100100111111011111111

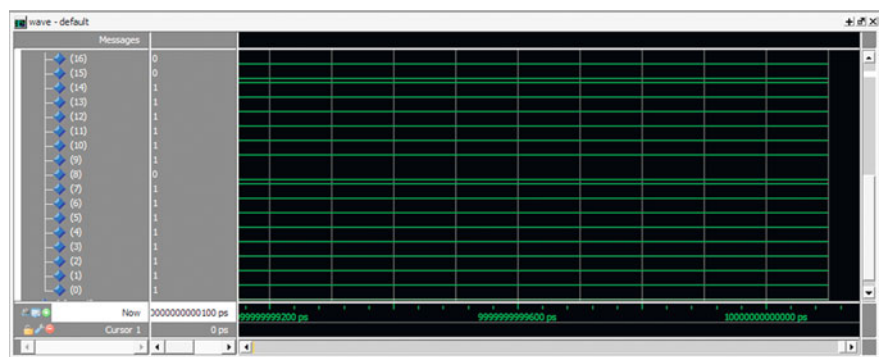


Fig. 2.9 Predicted level value [bit 15–0]

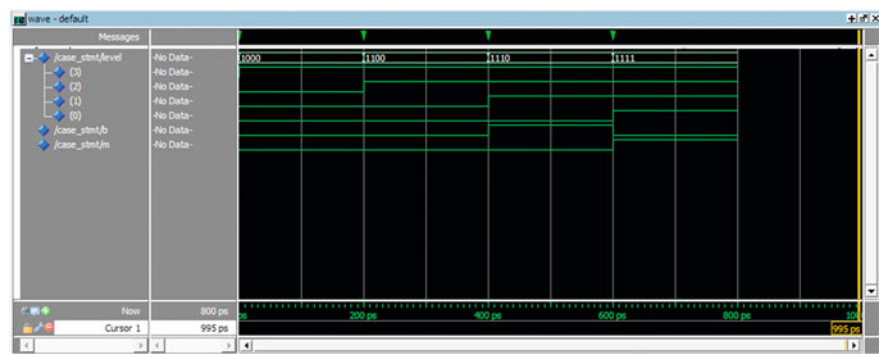


Fig. 2.10 Signals of sensors actuating the motor and buzzer

The above graph shows the automatic control of buzzer and gates through level measurements. Taking into consideration the KRS dam Mysore specifications, the total height of the dam is considered as 125 ft. This is divided into 4 levels namely 30, 60, 90, and 120 ft for operational purpose. However for safety purpose, the threshold level value is considered to be 120 ft.

- When water reaches 30 and 60 ft, only the level measurements are made and no controlling is required.
- When water reaches 90 ft, near to threshold level (120 ft) the buzzer switches on (alarm) alerting officials in control room that the level is approaching the threshold mark.

- When water reaches 120 ft i.e. threshold mark, the gates are automatically opened and required amount of water flows out of the dam. Once the water level falls below 120 ft the dam gates automatically close.
- The dam operations are hence completely automated. dam operations are hence completely automated.

2.8 Advantages of Designing Model in FPGA

- 1 Cost of the design is less compared to ASIC.
- 2 Unlike ASIC models, any changes can be implemented without having to make changes in the hardware i.e. it is flexible [9].
- 3 The model can be implemented into VLSI chip design which can be generated in large numbers making the model implementation easy and cost effective [10].

2.9 Scope for Future Work

This model strives to develop a VLSI chip as the end product and implement them in dams across the world. This work falls under the category of CPS (Cyber Physical Systems). CPS gives equal importance to the link between computational elements and physical elements unlike embedded systems which gives importance only to the computational elements. Development in this area will lead us to live in a more advanced and user friendly ‘Cyber Physical Society’.

References

1. Kleeman L (1995) Understanding and applying Kalman filtering. Department of Electrical and Computer Systems Engineering Monash University, Clayton
2. Kleinbauer R (2004) Kalman filtering implementation in MATLAB, study report in the field of study geodesy and geoinformatics. Universtat Stuttgart, Helsinki
3. Pasricha R, Sharma S (2009) An FPGA—based design of fixed-point Kalman filter. DSP J 9(1):1–9
4. Welch G, Bishop G (2001) An introduction to Kalman filtering. TR 95-041
5. Simon D (2001) Kalman filtering. Embed Syst Program
6. Cornell (2008) Kalman filter tank filling, subject MI63
7. Sorensen H (1985) Kalman filtering: theory and applications. IEEE Press, Los Alamitos
8. Chen G, Li G (2005) The FPGA implementation of Kalman filter. University of Science and Technology of China, China
9. Chi-Jui C, Mohanakrishnan S, Evans JB (1994) FPGA implementation of digital filters. University of Kansas, Lawrence, pp 66045–2228
10. Vij V, Mehra R (2011) FPGA based Kalman filter for wireless sensor networks. Lawrence

Recent Advancements in System Modelling Applications

Proceedings of National Systems Conference 2012

Malathi, R.; Krishnan, J. (Eds.)

2013, XIII, 508 p., Hardcover

ISBN: 978-81-322-1034-4