FPGA-Based Implementation of Kalman Filter for Real-Time Estimation of Tire Velocity and Acceleration

Fargham Sandhu, *Member, IEEE*, Hazlina Selamat, S. E. Alavi, and Vahid Behtaji Siahkal Mahalleh, *Member, IEEE*

***Abstract*— Antilock braking systems require accurate vehicle speed and acceleration estimations from wheel speed sensors. Adaptive Kalman filtering is used to estimate wheel speed and acceleration for each wheel. Single core or multicore processor-based embedded systems perform the required estimation using a two-stage extended Kalman filter in many clock cycles, which limits Kalman filter speed and accuracy. In the proposed system, purely arithmetic operations used in the process are simplified and implemented as embedded logic to reduce the overhead delays and increase the update rate by 48%. The proposed system is run sequentially using finite-state machines, while a large number of multiplication and division operations are removed through rewiring and logic reuse. The simplifications not only reduce the amount of logic required to perform the operations by 39.7% but also use simple logic, which can run faster than the replaced logic on field-programmable gate array. The results presented show comparable root-mean-square wheel speed error with a high update rate of 20 MHz when running on a 50-MHz board in all scenarios. The system also provides accurate wheel acceleration estimation, which is crucial for many systems, including the reference system.**

***Index Terms*— Kalman filter, finite state machine, parallel processing, computational overheads, non-restoring division algorithm, algorithm reuse.**

I. Introduction

At HIGH speeds, emergency braking in vehicles causes the tires to lock, forcing the tire-road friction(μ) to reduce the speed to a halt. Wheel locking causes the tire forces to become saturated, creating uncontrolled tire slips and instability. The list of various states used to define this condition are given in Fig. 1.

The antilock braking system (ABS) guesses the value of μ using wheel slip-friction relationships from calculated wheel slips using V_x and V_w and optimizes it by adjusting the brakes. Meanwhile $\dot{V}_w[k]$ is used to limit the rate at which the tires are forced to reach saturation [1]–[4].

Manuscript received April 22, 2017; revised May 21, 2017 and June 28, 2017; accepted July 8, 2017. Date of publication July 13, 2017; date of current version August 9, 2017. The associate editor coordinating the review of this paper and approving it for publication was Dr. Ioannis Raptis.

F. Sandhu, H. Selamat, and V. B. S. Mahalleh are with the Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia (e-mail: fsandhu2@gmail.com; hazlina@utm.my; vahidbehtaji2013@gmail.com).

S. E. Alavi is with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 83111-84156, Iran, and also with the Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia (e-mail: sayedehsan@utm.my).

Digital Object Identifier 10.1109/JSEN.2017.2726529

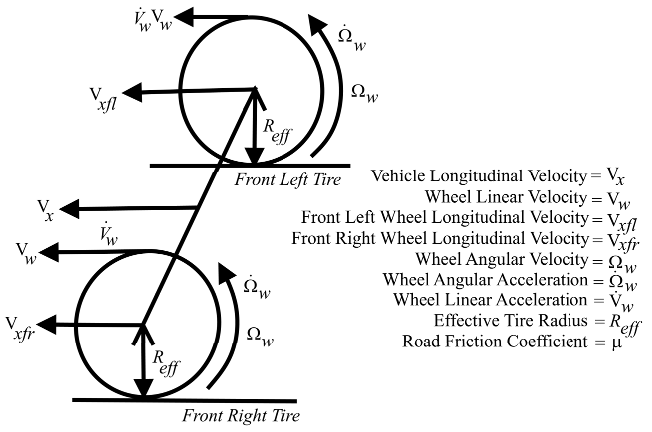

$\dot{V}_w V_w$
$\dot{\Omega}_w$
Ω_w
V_{xfl}
R_{eff}
Front Left Tire
V_x
$\dot{V}_w$
V_w
$\dot{\Omega}_w$
V_{xfr}
Ω_w
R_{eff}
Front Right Tire
Vehicle Longitudinal Velocity = V_x
Wheel Linear Velocity = V_w
Front Left Wheel Longitudinal Velocity = V_{xfl}
Front Right Wheel Longitudinal Velocity = V_{xfr}
Wheel Angular Velocity = Ω_w
Wheel Angular Acceleration = $\dot{\Omega}_w$
Wheel Linear Acceleration = $\dot{V}_w$
Effective Tire Radius = R_{eff}
Road Friction Coefficient = μ

Fig. 1. List of parameters and states used to control ABS system.

V_x is measured using whether bulky optimal correlation method based sensors (OC) with high installation costs and less reliability or cheaper and more reliable wheel speed sensors (WSS); hence, WSS sensors are more suitable for critical applications [5]. WSS sensors measure Ω_w with less accuracy due to sensor nonlinearities causing miscounts, magnetic hysteresis, gap-noise and signal distortions [5]. The sensor nonlinearities are overcome by estimating Ω_w using EKF [4], provided the sensor and process noises are limited by their respective variances [5].

For better V_x estimation, the front left V_{xfl} and front right V_{xfr} velocities of each tire are separately estimated from their respective tire angular velocities at each sampling instant k and used to obtain V_x using (1)

$$V_x[k] = \left[\frac{V_{xfl}[k] + V_{xfr}[k]}{2}\right] \tag{1}$$

In (1), the estimated instantaneous velocities of both tires must be synchronized with the sampling instant k. Existing digital circuits using very large scale integrated circuits (VLSI) for this purpose have the following problems.

1) Existing systems use multicore processors with real-time schedulers to synchronize and sequentially run 2 EKFs in parallel. The schedulers reduce the clock skew problem to at least 1 schedule time by increasing system complexity [21] with higher power consumption [19] and slower speeds [10]. The clock skew caused by the schedulers causes randomly delayed estimations,

which requires uncorrelated data fusion methods to achieve higher accuracy [20], [21]. The data fusion methods consume logic and provide reduced system throughput.

2) Among various types of VLSI circuits, FPGAs provide maximum flexibility to the designer by allowing them to use pipelined, custom-built parallel architecture and high-speed multipliers [6]–[12], floating point (FP) support for higher accuracy [22]–[24] in resource constraint designing [13] and multiprocessor designing [14]. Digital application specific integrated circuit (ASIC) design on the other hand has fixed architecture with the least amount of logic. More logic is used in FPGAs because of the logic mapping mechanism, in which some logic is used where only wires are required [14], [15], increasing the latency [15]. The latency created by the mapping mechanism can be reduced by custom-building the architecture of the system with embedded logic [25]. Existing custom-built parallel implementation of 2 EKF estimators in a multicore FPGA-based system requires shared memory and local memory to run both filters on multiple cores with embedded logic [14]. The algorithm consumes logic for maximum available arithmetic resource utilization [15], increasing the latency. The latency causes random sampling of the input wheel speeds [15].
3) The tremendous flexibility provided by FPGA is through extensive programming, thorough testing and verification [18]. For this purpose, rapid prototyping tools are designed using high level languages [19], [26], [27]. These tools reduce the time to market, but cannot guarantee optimal coding efficiency and resource usage [19]. Therefore, several options are offered by high-level programmers to optimize the code using standard IP cores [19], [26], [27]. These options have shown optimal resource usage for typical case studies, but still require extensive testing to ensure optimization in all systems [19].

Following is a list of solutions provided in this system.

1) FPGA is used to implement a state machine (SM) to simultaneously run custom logic with on-chip block memory for each filter. This method reduces the chances for data collision as compared to shared memory, reduces the access time delays as compared to external memory and eliminates scheduling logic required for multicore processors. These modifications eliminate the clock skew problem so that (1) can be applied without wasting extra logic for implementing the uncorrelated data fusion methods.
   Optimization techniques used during hand coding can significantly reduce the cost for EKF by customizing the logic to use optimal FPGA resources [28]. Therefore, the system is hand-coded with logic optimizations. During optimization, the operations are reduced while multiplication operations are performed using embedded, hardwired 18-bit multipliers.
2) In the proposed implementation, the accuracy of all operations is limited to a signed fixed point modulo-32 structure. This method is different from UD factorization [23], where Kalman gain is fixed and FP calculations for Kalman gain are eliminated.

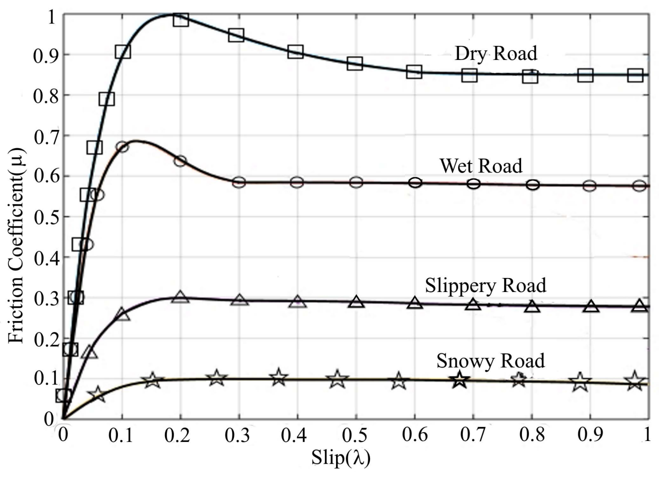


Fig. 2. Relationship between (μ) and slip (λ) on different road surfaces.

The modifications proposed in the paper reduce the cost of implementing the EKF filters in terms of the logic required and increase the throughput of the EKF filters.

II. Problem Formulation

Equation (2) shows the relationship of $V_w[k]$ with $\Omega[k]$ and sensor noise ($\eta[k]$) at instant k.

$$V_w[k] = (R_{eff} + R_{error})(\Omega_w[k] + \eta[k]) \tag{2}$$

In (2) the WSS sensor noise (η) is considered additive white Gaussian noise (AWGN) with zero mean and variance ($R = E(\eta[k]^T \eta[k+1]) \approx 0.4$). R_{eff} is the effective tire radius and R_{error} is the error in tire radius due to tire deformations [1].

During ABS control R_{error} is essentially zero ($R_{error} \approx 0$), so its effect is neglected in the equations [1].

In ABS systems $V_{xfl}$and V_{xfr} are obtained using the respective V_{wi}, slip (λ_i) and steering angle input (δ) at sampling instant k, as given in (3) [2].

$$V_{xi} = \begin{cases} \dfrac{V_{wi}[k]}{\lambda_i[k]+1} & \text{During braking for wheel} \\ & \text{i} \in \{\text{fl,fr}\} \\ (1-\lambda_i[k])V_{wi}[k] & \text{During Acceleration} \\ V_{wi}[k]\cos\delta & \text{During cruising} \end{cases} \tag{3}$$

The slip (λ) is important because μ is approximated from it using the Burkhardt Equation (4) [29], which is a nonlinear function as given in Fig. 2 [2].

$$\mu(\lambda) = (C_1(1 - e^{-C_2\lambda}) - C_3\lambda)\psi \tag{4}$$

In (4), C_1 is the steady state value of the friction coefficient, C_2 is the constant that defines the slops of the friction coefficient, while C_3 defines the rate of change of the friction coefficient after reaching steady state values.

These coefficients for dry road surfaces are obtained by plugging in the values at $\mu|_{\lambda=0.7} = 0.85$ and $\mu|_{\lambda=0.18} = 1$, and the relationship of C_3 given in (5). The scaling constant ψ

is obtained by plugging the maximum value of μ at $\lambda = 0.18$ in the final result.

$$\frac{\mu|_{0.7}}{\mu|_{0.18}} = \frac{1 - e^{-0.7C_2} + 0.7C_2}{1 - e^{-0.18C_2} + 0.18C_2} = 0.85,$$
$$\lim_{S\to\infty} \dot{\mu} = 0 \Rightarrow C_3 = -C_1C_2, \quad C_1 = 0.85, \quad \psi = 0.45 \quad (5)$$

The variable slip (λ) in (4) is obtained at each sample interval k using (6).

$$\lambda = \begin{cases} (R\Omega[k] - V[k])/V[k] & \text{During braking} \\ (R\Omega[k] - V[k])/R\Omega[k] & \text{During Acceleration} \\ 0 & \text{During cruise} \end{cases} \quad (6)$$

During braking, the ABS system maximizes the friction coefficient between the tires and road surface by adjusting the slips at an optimal value of 0.18 [2], obtained by calculating the maxima of (4), as given by (7).

$$\dot{\mu} = \psi C_1 C_2 e^{-C_2\lambda} - C_3 = 0 \Rightarrow \lambda = -\frac{1}{C_2}\ln\{\frac{C_3}{\psi C_1 C_2}\} \approx 0.18 \quad (7)$$

Once the slip reaches the optimal value, it must not change any further ($\dot{\lambda} = 0$) to maintain optimal conditions. The requirement for maintaining optimal slips during braking is given in (8)

$$\dot{\lambda} = \frac{1}{R_{eff}\Omega[k]}\dot{\Omega}_w[k] - \frac{1}{R_{eff}\Omega_w[k]}\dot{V}_x \approx 0 \Rightarrow \dot{\Omega}_w[k] = \dot{V}_x[k] \quad (8)$$

According to (8), the slip stays optimal if $\dot{V}_x$ is equal to $\dot{\Omega}_w$ during braking. Under this condition, $\dot{V}_w$ and $V_x[k]$ during braking are calculated in (9).

$$\dot{V}_w[k] = R_{eff}\dot{\Omega}_w[k] \quad \text{and} \quad V_x[k] = \Sigma_{l=0}^{k}\dot{V}_x[l] \quad (9)$$

Since Ω_w is corrupted by η, $\dot{V}_w$ is not directly measured from the wheel speed with high accuracy; therefore, the expected value of $\dot{V}_w$ is obtained in terms of the expected values of $\dot{\Omega}_w$ as given by (10)

$$E\{\dot{V}_w^T\dot{V}_w\} \approx R_{eff}^2(E\{\dot{\Omega}_w^T\dot{\Omega}_w\} + E\{\dot{\eta}^T\dot{\eta}\}) \quad (10)$$

In (10), the variance in $\dot{\Omega}_w[k]$ is defined as $E\{\dot{\Omega}_w^T[k]\dot{\Omega}_w[k+1]\} = Q$ which is the process noise variance, while the variance of $\dot{\eta}[k]$is defined as $E\{\dot{\eta}[k]^T\dot{\eta}[k+1]\} = 0$.

Equation (10) suggests the wheel linear acceleration variance is dependent on the sensor and system noise variance. Since both noises are AWGN, stochastic filters like EKF are suitable for the estimation of wheel acceleration [4], [5].

The EKF for estimating the linear velocity and acceleration of both wheels is modelled using the WSS sensor model given in the next section.

III. System Model and Implementation

A. Wheel Speed and Acceleration Estimator

The low-cost form of EKF estimator for Ω_w and $\dot{\Omega}_w$ is designed using the Ω_w model given in the form of the Taylor formula in terms of the sampling time ($T = \Delta t$) in (11),

$$\Omega_w(t + \Delta t) = \Omega_w(t) + T\dot{\Omega}_w(t) + \frac{T^2}{2}\ddot{\Omega}_w(t)\dots w(t) \quad (11)$$

In this system, $X(t) = [\Omega_w, \dot{\Omega}_w, \ddot{\Omega}_w]$ is considered, such that $\Omega_w = \Omega_w + T\dot{\Omega}_w + w_1(t)$, $\dot{\Omega}_w(t) = \dot{\Omega}_w(t) + T\ddot{\Omega}_w(t) + w_2(t)$ and $\ddot{\Omega}_w = \ddot{\Omega}_w + w_3(t)$ where $w(t) = [w_1(t), w_2(t), w_3(t)]$ is a vector in which $w_1(t)$ is the wheel speed noise, $w_2(t)$ is the wheel acceleration noise and $w_3(t)$ is the rate of change of wheel acceleration noise.

In terms of the state vector, $X[k]$ the discrete time equation of the WSS system, is obtained using zero order hold (ZOH) as given in (12) and (13).

$$X[k+1] = AX[k] + G\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}^T, \quad y[k] = HX[k] \quad (12)$$

where

$$A = \frac{\delta\Omega}{\delta x} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \; G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$
$$H = [1, 0, 0] \quad (13)$$

B. EKF Structure

A generalized discrete EKF filter for a discrete time system is given in (12) [30]. In this equation, $x[l|m]$ means the value of x at instant l given the previous sample at m.

Initialize with $\hat{x}_0 = E[x_0]$, $P_{x_0} = E[(x - \hat{x}_0)(x - \hat{x}_0)^T]$

For $k \geq 1$ the time update equations for EKF filter are

$$\begin{aligned} x[k|k-1] &= Ax[k-1|k-1] \\ P[k|k-1] &= A[k-1]P[k-1]A^T[k-1] + Q \end{aligned} \quad (14)$$

The measurement update equations are

$$\begin{aligned} K &= P[k|k-1]H^T[k](H[k]P[k|k-1]H^T[k] + R)^{-1} \\ \hat{x}[k] &= \hat{x}[k|k-1] + K(y[k] - H\hat{x}[k|k-1]) \\ P[k] &= (I - KH[k])P[k|k-1] = P[k|k-1] \\ &\quad - KH[k]P[k|k-1] \end{aligned}$$

In (14), $\hat{x}[k]$ is the estimate of x from sampled input y[k]. A is the linear state matrix of the system, H is the output vector, P is the covariance matrix while Q is the process noise covariance matrix and R is the sensor noise covariance matrix, as given in (15)

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}, \quad R = r \quad (15)$$

In (15) r is the noise density of WSS sensors and q is the specified process noise density. The proposed EKF filter initialization and measurement update instructions are given in (16). The remaining instructions of EKF are the same as given in (14).

Initialize with

$$x_0 = \begin{bmatrix} \Omega_0 \\ 0 \\ 0 \end{bmatrix}, \quad P(0) = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$K = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix}^T = \begin{bmatrix} \frac{p_1}{p_1+r} & \frac{p_3}{p_1+r} & \frac{p_7}{p_1+r} \end{bmatrix}^T \quad (16)$$

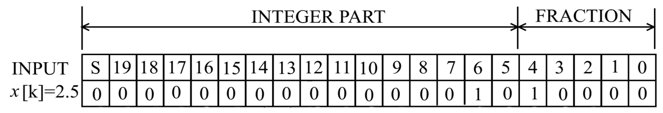


Fig. 3. Q (16,5) format used to represent x = +2.5.

TABLE I

LIST OF OPERATIONS FOR EKF

Operations	Resources Used
x[k|k-1]	5 Multiplications, 3 Additions
d_i	6 Multiplications, 6 Additions, 3 Squares, 3 Divisions
e_i	6 Multiplications, 6 Additions
p_1[k|k-1], p_2[k|k-1], p_3[k|k-1]	6 Multiplications, 6 Additions, 3 Squares, 3 Divisions
p_4[k|k-1], p_5[k|k-1], p_6[k|k-1]	6 Multiplications, 6 Additions
p_9[k|k-1]	1 Addition
K[k]	3 Divisions, 3 Additions
$\hat{x}[k\,\|\,k]$	3 Multiplications, 4 Additions
p_i[k|k]	9 Multiplications, 9 Additions

A direct implementation of the EKF filter defined in (14) to (16) requires many arithmetic operations as given in Table. I

According to Table I, there are 41 multiplications, 46 signed additions, 6 squares and 9 division operations required to implement 1 EKF filter. All these operations require FP support, which is implemented using fixed-point modulo-32 format.

C. Modulo-32 Format

In modulo-32 format, n = 5 least significant bits are used for representing the fractional part of the FP and m = 16 bits are used for representing the integer part of the FP with the most significant bit of the integer part representing the sign bit (S). The total length of the register is 21 bits. This type of data structure is represented as Q (m, n) = Q (16, 5). In this format, a floating-point number like $x = +2.5$ is represented, as shown in Fig. 3.

According to Table I, a large number of operations is required in each EKF implementation. In order to efficiently run both EKF filters on 1 chip with perfect synchronization and least latency, the number of instructions and their complexities are optimized in the subsequent subsections.

D. Algorithm Optimization

In the proposed system, the equations used in the process are simplified by fixing the system sampling time T (T = 250ms).

For addition and subtraction operations, the addition operator is modified to perform addition and subtraction according to the sign bit. The addition operator is re-used by the SM to perform all the addition and subtraction operations in same instructions. Following is the list of optimizations used in the design

1) Since EKF sampling time $T = 0.25s = 2^{-2}s$ (a binary multiple of 2), any multiplication with sampling time will cause a shift in the value by 2 places. For example, as shown in Fig. 4, since $T = 2^{-2}$ at any instant $x_1[k] = 2.5$, $Tx_1[k] = 2.5 * 2^{-2} = 0.625$, which is similar to repositioning the bits of $x_1[k]$ two places to the right, moving the 1's at bits position 4 and 6 to positions 2 and 4, which represents 0.625. Other sampling times would cause shifts to the data by $\log_2(T)$ places. In the system, repositioning of the bits is achieved by rewiring the registers that hold the result [15]. This optimization is applicable to sampling times which are binary multiples of 2 only.

   After replacing any multiplications with sampling time by shift operations, which are removed through register rewiring as shown in Fig. 4, the total number of operations are reduced as given in Table II.

   The modified EKF instructions after rewiring are given in the Appendix. The number of operations are thus reduced to 3 divisions, 12 multiplications, and 46 signed additions. In all these operations, decimal point rounding and adjustment is performed using comparators.
2) For reducing the logic and maintaining the accuracy during FP operations, some instructions of the algorithm are scaled and modified. The scaling of the algorithm is explained in (17) and (18).

   Let $2^5 * P = \chi$ such that EKF is initialized using (17)

$$\chi[0] = 2^5 * P[0] \tag{17}$$

Then some of the time update equations for EKF are modified as follows.

$$\begin{aligned} 2^5P[k|k-1] &= 2^5APA^T + 2^5Q \Rightarrow \chi[k|k-1] \\ &= A\chi A^T + 2^5Q \\ 2^5p_i[k|k] &= 2^5p_i[k|k-1] + k_i2^5p_j[k|k-1] \\ &\Rightarrow \chi_i[k|k] = \chi_i[k|k-1] - k_i\chi_j[k|k-1] \end{aligned} \tag{18}$$

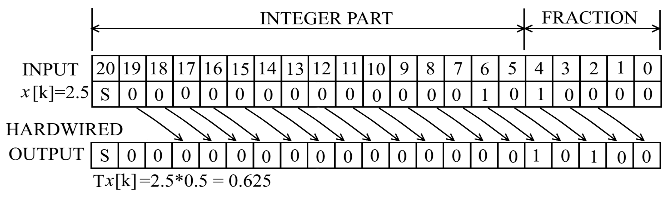


Fig. 4. Multiplication of x[k] with sample time (T) is performed by repositioning the bits (shown by arrows).

TABLE II

LIST OF OPERATIONS AFTER REPLACING MULTIPLICATIONS WITH TIME

Operations	Resources Used
x[k|k-1]	3 Additions
d_i	6 Additions
e_i	6 Additions
p_1[k|k-1], p_2[k|k-1], p_3[k|k-1]	6 Additions
p_4[k|k-1], p_5[k|k-1], p_6[k|k-1]	6 Additions
p_9[k|k-1]	1 Addition
K(k)	3 Divisions, 3 Additions
$\hat{x}[k\,\|\,k]$	3 Multiplications, 4 Additions
p_i[k|k]	9 Multiplications, 9 Additions

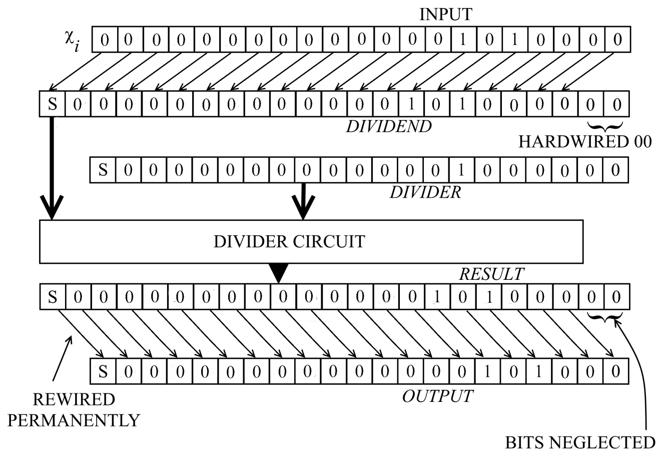


Fig. 5. Division operation with pre-scaling and post-scaling. The shift operations are performed by rewiring the registers (shown with arrows).

In (18), the values of p_i are scaled by 2^5 and represented by χ_i. This method converts the FP numbers of p_i into 16-bit integers, doubling the FP accuracy while limiting its maximum value to $p_i \leq 2^{16}$. For example, due to scaling, $\chi_1 = 2.5$ at any point actually represents a value of $p_1 = 0.07812$. The scaling of P requires the process noise and sensor noise be scaled and saved as $q = 0.25 * 2^5 = 8$, $r = 0.1875 * 2^5 = 6$. The scaling of P does not affect the values of k_i as shown in (19). Any further scaling is not required.

3) Equation (16) shows that for calculating the Kalman gains k_i, 3 division operations are needed with a single divider. The divider Z is pre-calculated before the division operation is performed and placed in a Block RAM location.

During division operations, the updated χ_i are scaled by $\sigma = 2^2$ to ensure $2^2\chi_i \geq Z$, by increasing the divider register size and division algorithm by $\log_2(\sigma) = 2$ bits. This value is chosen because any value higher than σ would increase the register size and division algorithm further, reducing the margin of saving logic while smaller values may not satisfy the requirement $2^2\chi_i \geq Z$. The scaled division algorithm is given in (19).

$$K = \left[\frac{2^2\chi_1}{Z}, \frac{2^2\chi_3}{Z}, \frac{2^2\chi_7}{Z}\right]^T * \left[\frac{1}{2^2}\right], \quad Z = \chi_1 + r \tag{19}$$

In (19), the product $\chi_i$is first scaled by 2^2, divided by Z and rescaled to obtain the correct value of k_i. For example, as shown in Fig. 5, $\chi_i[20:1] = 2.5$ is divided by $Z\ [20:0] = 2$ by first scaling χ_i to obtain $2^2\chi_i = 10$, which is divided by Z = 2 ($2^5 * \chi_i/Z = 10/2 = 5$) and rescaled by 2^{-2} to obtain the result $k_i = 5*2^{-2} = 1.25$. The process of scaling and rescaling is performed by rewiring the dividend and result registers.

For division operations, an efficient, non-restoring division algorithm with the minimum number of lookup tables and 2-stage pipelining is used to double the throughput at the expense of only a 2-clock-cycle increase in latency [31].

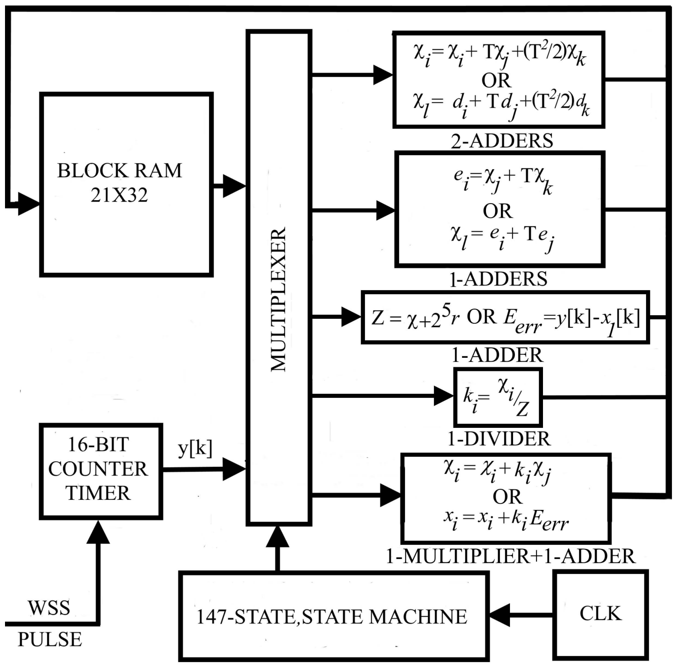

BLOCK RAM 21X32
MULTIPLEXER
$\chi_i = \chi_i + T\chi_j + (T^2/2)\chi_k$ OR $\chi_l = d_i + Td_j + (T^2/2)d_k$
2-ADDERS
$e_i = \chi_j + T\chi_k$ OR $\chi_l = e_i + Te_j$
1-ADDERS
$Z = \chi + 2^5 r$ OR $E_{err} = y[k] - x_l[k]$
1-ADDER
$k_i = \chi_i / Z$
1-DIVIDER
$\chi_i = \chi_i + k_i\chi_j$ OR $x_i = x_i + k_i E_{err}$
1-MULTIPLIER+1-ADDER
16-BIT COUNTER TIMER
y[k]
WSS PULSE
147-STATE,STATE MACHINE
CLK

Fig. 6. Block diagram of the proposed Kalman filter.

When sampling the measurements, the error between the estimated state and sample ($E_{err} = y[k] - \hat{x}_1[k|k-1]$) is calculated and saved in a memory location. The sample update equations are calculated using a hardwired multiplier and signed addition operator, as given in (20).

$$\hat{x}_i[k|k] = x_i[k|k-1] + k_i E_{err} \tag{20}$$

The final form of the fully parallel Kalman architecture reuses common operations by running the algorithm with a fixed SM.

E. Algorithm Implementation

The instructions in (14) must be executed sequentially. In order to run the instructions sequentially, the algorithm is executed using a large SM. The block diagram of the system is given in Fig. 6.

In Fig. 6, the EKF instructions are executed at higher speeds so that all the time update instructions are executed before the next input $y[k]$ is sampled. The instructions are run in sequence by the SM. The wheel speed is measured by a 16-bit counter/timer at each sample time and the error E_i is calculated and saved in a memory location. The measurement update equations are executed only after the sample input is available, since they depend upon the current error E_i. The dual-port Block RAM is used to save all partial and final products with minimum access time.

The multiplexer is used to reuse dedicated logic blocks to compute common operations and save the results to memory for further use. By reusing common blocks, less logic is used. In the blocks OR items mean the same logic can be used to compute both instructions.

The same SM is used to run both EKFs with dedicated Block RAM for each filter to avoid data collisions while

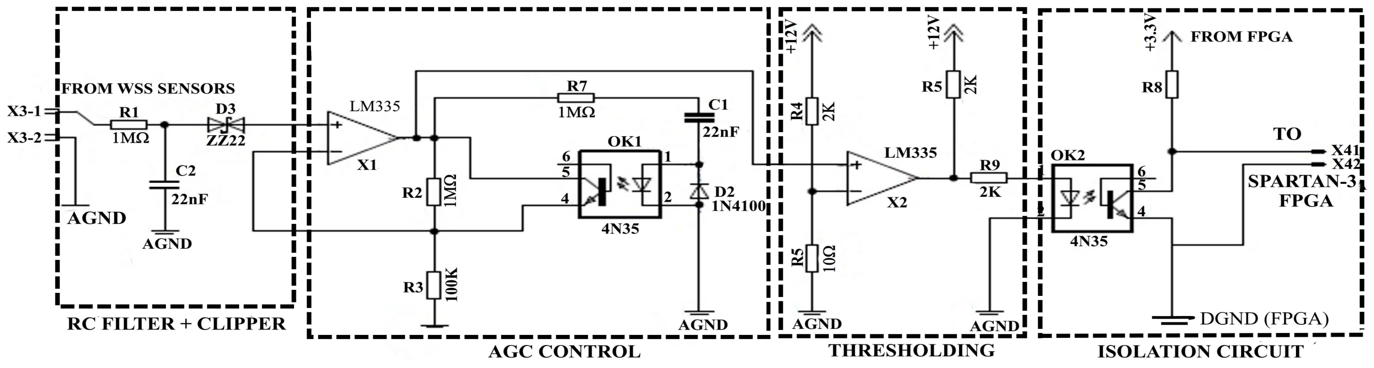


Fig. 7. Wheel speed sensor interface circuit.

on-chip Block RAM is used to reduce access time. The SM runs both EKFs with the same clock speed to ensure perfect synchronization.

Before EKF filtering, analog circuits are used to limit the sensor signal with variance R and system noise with variance Q as given in the following section.

F. Analog Circuit Interface

The analog interface circuit consists of 4 essential blocks as shown in Fig. 7.

1) RC Filter and Clipper: The input signal from the wheel speed sensors consists of distorted pulses with high amplitude, ranging from a few millivolts at near zero speed to tens of volts. The high amplitude signal is passed through a passive, low-pass RC filter (R_1C_2) and clipper circuit (D_3) with a cutoff frequency of 7.23Hz [32], [33] to block the spurious high frequency noise signal from entering the circuit and limit the input signal between $\pm 20V$, to prevent permanent damage to the circuit [31].

2) Automatic Gain Control (AGC) Circuit: The AGC circuit adaptively adjusts the amplitude of the signal for better thresholding, because the wheel speed sensor signal changes with an amplitude from a few volts at low speed to 20V at maximum speed. The AGC circuit adjusts its gain according to the output using opamp (R2R3X1) with feedback through vectral (R7C1OK1D2). The AGC circuit ensures that the comparator (R4R5X2) functions properly at all speeds and does not allow the opto-isolator (R5R9OK2R8) to become saturated. The AGC circuit also improves the circuit electromagnetic interference (EMI), which is a major problem in wheel speed circuits [32].

3) Thresholding Circuit: The thresholding circuit is designed using an X2 opamp that has a set threshold value using resisters R4-R5 with output signals adjusted using resistors R5-R9.

4) Isolation Circuit: The output of the thresholding circuit consists of pulses, which are isolated from the FPGA by the opto-isolator OK2. The output end of the opto-isolator is run by the FPGAs 3.3V supply and FPGA ground for galvanic isolation. Separate analog and digital grounds are used to restrict the analog ground noises from destroying the signal.

TABLE III

LIST OF PARAMETERS USED FOR THE CONVERSION OF ESTIMATED ANGULAR VELOCITY AND ACCELERATION

Parameter	Value
Sensor Disk Teeth (N)	40
Tire radius (R_{eff})	0.292m
q	0.25
r	0.1875

IV. RESULTS AND DISCUSSIONS

A. Assumptions

The estimated angular velocity and acceleration from the EKF filters are calibrated according to the WSS sensor disk teeth N and R_{eff} of the test vehicle, which uses 175/65R14-sized tires as given in Table III.

The WSS used in the test vehicle is a general-purpose magnetic permeability measuring device used in all commercial cars with sensor noise density r [32]. The process noise density q of the system is selected for best performance. The conversion factor (υ) used to convert the wheel angular speed into linear wheel speed for both filters is given in (21) [31]–[34].

$$\upsilon = \frac{2\pi R_{eff}}{N} = \frac{0.292 R_{eff}}{20} \approx 2^{-5} \tag{21}$$

For implementing υ, the actual wheel counter bits are moved to the right by 5 places to amplify the value by 2^{-5}, according to (21). For example, as shown in Fig. 8, if $\Omega[k] = 200 rev/s$ then $V_w = \upsilon * \Omega = 6.25 m/s$.

The actual conversion constant in this method is larger than 2^{-5} by an error of $+0.01475$, which is used to compensate for tire deformations related to tire loading, temperature variations and tire conditions. A positive error means that the speed calculated will be slightly less than the actual speed by a small error, which is a multiple of 2.

B. Reference System as Benchmark

The results of the system are compared with *Estimated V_x-scenarios-1-4* [34] as reference systems. In *Estimated Vx-scenario-1* the four-wheel speed angular (4WS) velocities are measured and used in the 4-state Kalman filter, while in

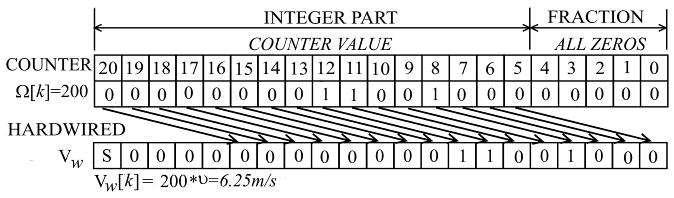


Fig. 8. Hardwired register used to perform the conversion.

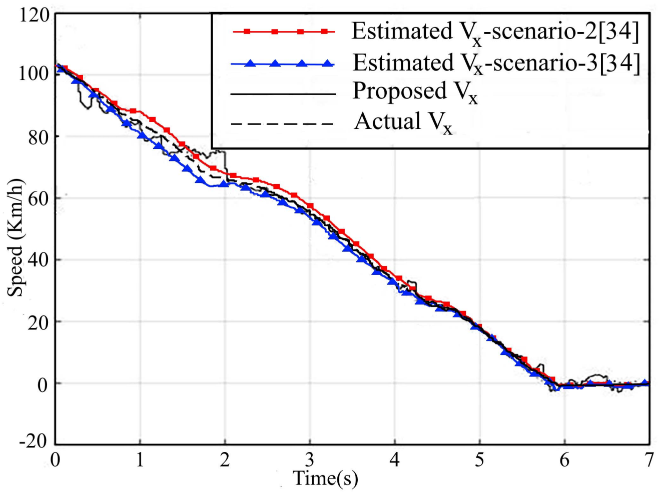


Fig. 9. Comparison of actual, estimated and referenced (V_x) speeds.

TABLE IV
RMS AND UPDATE TIME OF EKF IMPLEMENTED IN REFERENCE AND PROPOSED SYSTEM

Method	RMS error of Longitudinal Velocity(Km/h)	Update Time (μs)
Estimated Vx-scenario-1(4WS)	0.1501	21.65
Estimated V_x-scenario-2(FWS)	4.2799	5.61
Estimated V_x-scenario-3(RWS)	2.2028	5.61
Estimated V_x-scenario-4(FRWS)	0.1714	7.27
Proposed Velocity(FWS)	1.6384	0.05

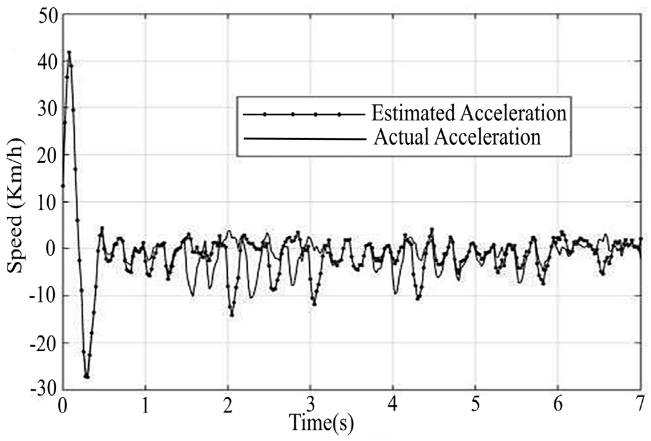


Fig. 10. Wheel actual and estimated deceleration during ABS control.

Estimated Vx-scenario-2, only the rear wheel speeds (RWS) are measured and used in the filter. In the *Estimated Vx-scenario-3* only the front wheel speeds (FWS) are measured and used in the filter to estimate the vehicle and wheel velocities. In *Estimated Vx-scenario-4*, the results of scenario-2 and scenario-3 are combined using the Millman data fusion method, which is suitable for fusing uncorrelated signals (FRWS) [21] to improve the results.

The state matrix in the 4-state Kalman filter is updated using the accelerometer-measured deceleration as wheel deceleration to switch relevant coefficients of the state matrix. This assumption, according to (6) is valid if the slips in tires are optimal and further changes in the slips do not occur.

C. V_x Estimation

The estimated $\hat{V}_x$ and actual V_x are compared with the referenced-V_x speeds during scenario-2 and scenario-3 [34], as given in Fig. 9.

The *Estimated V_x-scenario-1* and *Estimated V_x-scenario-4* [34] are not plotted in this figure since they overlap the actual V_x. The *estimated Vx-scenario-2* and *estimated Vx-scenario-3* form the upper and lower limits of the benchmark reference systems. The system results are between these 2 benchmarks and satisfy the 6-sigma criteria for accepting estimated results. According to the 6-sigma criteria the estimated velocity should be less than or equal to 6 times $r(|\pm 6 * r| = 1.7Km/h)$ [30].

In Fig. 9, $\hat{V}_x$, *Estimated V_x-scenario-2* and *Estimated V_x-scenario-3* [34] overlap at the start of the journey from 0 to 0.8 seconds. After 0.8s, the *Estimated Vx-scenario-2* has slightly more velocity than V_x. At the same time, *Estimated Vx-scenario-3* shows slightly less velocity than V_x. $\hat{V}_x$ during this time starts to show higher velocity than V_x, and it achieves the highest velocity for a short time during the interval 1.8-2 seconds. After 2 seconds, $\hat{V}_x$ closely follows V_x for the rest of the journey. The *Estimated Vx-scenario-2* stays a little higher than V_x while *Estimated Vx-scenario-3* stays slightly lower than V_x during the next 2 seconds. In the last 2 seconds of the journey, all the velocities follow V_x closely. The RMS error in $\hat{V}_x$ and reference systems is given in Table. IV.

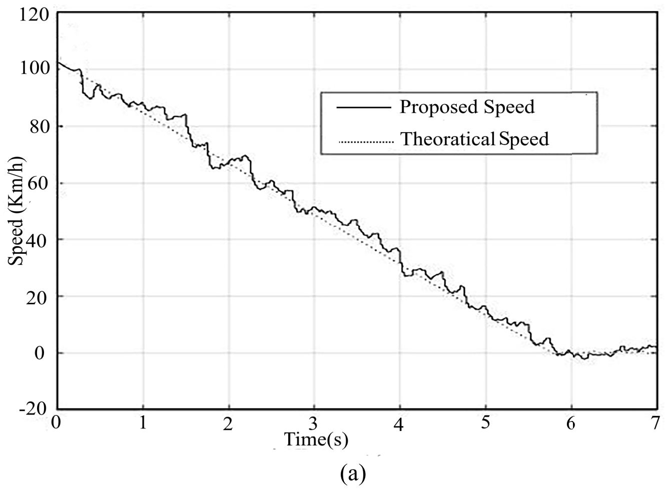
The results presented in Fig. 9 and Table IV show that the proposed velocity RMS error is less than the velocity RMS error in scenarios 2 and 3 using front or rear wheel sensor data, but worse than scenarios 1 and 4, in which 4-wheel sensor data is used.

D. $\dot{V}_w$ Estimation

During ABS braking, $\dot{V}_w$ is rapidly changing, as given in Fig. 10. Therefore, the reference system $\dot{V}_w$ must be estimated in real time to update the state matrix and to obtain the longitudinal velocity in scenarios 1 through 4.

The estimated acceleration in Fig. 10 is plotted with the actual wheel acceleration, measured by using a high-speed rate gyroscope with KF and bias compensation to measure the actual angular accelerations ($\dot{\Omega}_w$) and calculate $\dot{V}_w$ using (8).

The estimated acceleration has an RMS error of $8.022km/h^2(2.2283m/s^2)$ in each wheel. Earlier systems



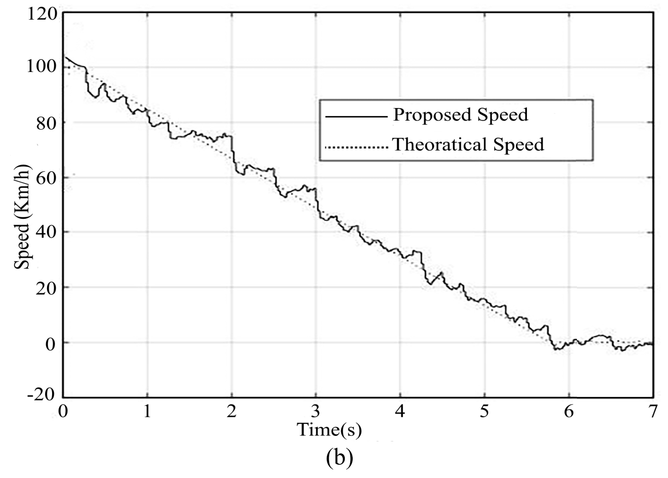


Fig. 11. Estimated and actual speed for (a) front right V_{xfr} and (b) front left V_{xfl} wheels.

showed an acceleration RMS error of $4m/s^2$ [5] in each wheel. The estimated acceleration satisfies the 6-sigma criteria $\left|\pm 36 * (q+r)^2\right| = 9.7488km/h^2$ [30], which specifies a margin of acceptable estimated values depending upon the sensor and process noise densities q and r defined in the estimation process.

E. V_{xfl} and V_{xfr} Estimation

In Fig. 11, the V_{xfr} and V_{xfl} are perfectly synchronized as as they are estimated using the same clock source. Hence, the instantaneous velocities of the front right and front left wheels can be used in (1).

The theoretical speed mentioned in Fig. 11 is calculated based on (8), according to which $\dot{V}_x$ of the vehicle must be equal to $\dot{\Omega}_w$. A fixed linear deceleration rate is assumed so that $\dot{\Omega}_w$ is assumed to stay a non-zero constant during the experimental time. Since this result is based on the analysis, the reference V_x is mentioned as the theoretical speed of the wheels.

F. Logic Resources and Speed

The proposed system is also compared with similar 3-state EKF systems to compare the number of resources used and the speed of the proposed EKF. Table V lists the number of operations required to implement a simple EKF with the same state matrix and number of operations as the proposed system [7].

TABLE V
LIST OF OPERATIONS USED IN THE IMPLEMENTATION OF EKF, PROPOSED SYSTEM AND REFERENCE SYSTEM

Operations	EKF	Reference System [7]	Proposed System
Multiplication	41	18	12
Addition	34	4	30
Square	6	0	0
Division	9	12	1
Subtraction	12	11	12

TABLE VI
LIST OF RESOURCES USED AND SPEED OF THE RESULTANT SYSTEM

Resources	Simulink generated EKF [7][a]	Reference system [7][a]	Proposed system[b]
Slices	2145	471	284
Multipliers	16	4	2
Speed	52.9MHz	67.1MHz	20MHz
Clock cycles/ EKF Cycle[c]	4.7259	3.6819	2.5

[a]The reference system is run on XC3S700A FPGA kit with a clock frequency of f_{clk} = 250MHz. [b]The proposed system is run on XC3S500E board with a clock frequency of f_{clk} = 50MHz. [c]Clock cycles /EKF cycle = f_{clk}/Speed

The aim of reducing the number of operations in the proposed system was to reduce the amount of logic required for implementing the system and increase the throughput of the system. The results show that 39.7% less logic is used in the proposed system. Table VI shows the amount of resources used in the reference system and the proposed system.

In Table VI, the logic used by the reference system consists of the logic for the EKF filter, the logic for implementing the control law, the logic for interfacing the ADC and DAC and the logic for the RS232 UART communication module. The proposed system consists of logic for 2 EKF filters, 2 counters, 1 RS232 UART communication module, 2 Block RAMs and logic for scaling the WSS sensor data. The control loop logic implemented in the reference system uses 3 adders [7] while the DAC and ADC interfacing logic consists of 2 4-state SMs. The RS232 UART module is common in the reference and proposed system. For comparison purposes, the 2 counters and data scaling logic are considered equal to the ADC and DAC interfacing logic.

In both systems, the block RAMs are used either for 4-input lookup tables, registers or for Block RAM implementation by the mapper. The proposed system uses 2 Block RAMs for data storage. The proposed system uses fewer slices due to logic reuse for both EKF filters, while the EKF logic for reference systems is specified for only 1 EKF filter.

In Table. VI, the reference system and proposed system are running at different clock speeds. The ratio of clock cycles per EKF cycle is calculated for comparison. The proposed system shows the minimum clock cycles per EKF. This means the proposed system is faster than the reference systems. The results show 48% improvement in the EKF update rate.

V. Conclusion

In this work, fewer resources using parallelism to optimize the execution time are used. Generally better parallelism means more resource usage, which in the proposed system is optimized using a single SM to run 2 parallel EKFs without any arbitration logic. In each EKF, the resources are reused to reduce the amount of logic used by 39.7%. The arithmetic logic blocks are customized to reduce logic by rewiring the registers. The algorithm is hand-coded so that customization in each arithmetic operation is possible. Both EKFs use separate on-chip dual port block RAMs to reduce excess time and data collision while storing results. Both EKFs run the logic in parallel so that their outputs are estimated at the same time. This results in a 48% improvement in EKF update rate.

During coding, some results in each arithmetic logic block were reused in the next arithmetic operation. This feature indicates that further optimization in the EKF algorithm is possible. The algorithm also shows the possibility of using pre-division of Z with pipelined, cascaded multiplications to improve the processing time of division operations. Further optimizations can make use of these options.

Appendix

Since rewiring the registers is a method of placing register bits to different locations, the equations are represented in the form $a[l:m]$, which means the data a has bits from l to m, with l being the most significant bit. For $k \geq 1$ the modified and rewired time update equations for EKF are given in (22)

$$\begin{aligned}
x_1[k|k-1] &= x_1[15:0][k-1|k-1] + x_2[15:2][k-1|k-1] \\
&\quad + x_3[15:5][k-1|k-1] \\
x_2[k|k-1] &= x_2[15:0][k-1|k-1] + x_3[15:2][k-1|k-1] \\
x_3[k|k-1] &= x_3[15:0][k-1|k-1] \\
P[k|k-1] &= \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix} \quad \text{Where} \\
p_1 &= d_1[20:0] + d_2[20:2] + d_3[20:5] \\
p_2 &= e_1[20:0] + e_2[20:2] + e_3[20:5] \\
p_3 &= p_3[20:0] + p_6[20:2] + p_9[20:5] \\
p_4 &= d_2[20:0] + d_3[20:2] \\
p_5 &= e_2[20:0] + e_3[20:2], \\
p_6 &= p_6[20:0] + p_9[20:2] \\
p_7 &= d_3[20:0], \\
p_8 &= e_3[20:0] \\
p_9 &= p_9[20:0] + q[20:0], \\
z &= p_1[20:0] + r[20:0]
\end{aligned} \tag{22}$$

Obtained using

$$\begin{aligned}
d_1 &= p_1[20:0] + p_2[20:2] + p_3[20:5] \\
d_2 &= p_4[20:0] + p_5[20:2] + p_6[20:5] \\
d_3 &= p_7[20:0] + p_8[20:2] + p_9[20:5] \\
e_1 &= p_2[20:0] + p_3[20:2], \quad e_2 = p_5[20:0] + p_6[20:2] \\
e_3 &= p_8[20:0] + p_9[20:0]
\end{aligned}$$

The measurement update equations are

$$\begin{aligned}
K[k] &= \left[\frac{p_1[17:2]}{z[15:0]}, \frac{p_3[17:2]}{z[15:0]}, \frac{p_7[17:2]}{z[15:0]}\right]^T, \quad z = p_1 + r \\
\hat{x}[k|k] &= \hat{x}[k|k-1] + K[k](y[k] - H\hat{x}[k|k-1]) \\
P[k|k] &= P[k|k-1] + K[k]\left[\, p_1 \; p_2 \; p_3 \,\right]
\end{aligned}$$

These equations only represent the integer part of the floating-point data. The fractional part is adjusted in each operation using comparators and logic operations for adjustment and rounding up numbers smaller than 2^{-5}.

Acknowledgment

The authors would like to thank the Malaysian Ministry of Education and Universiti Teknologi Malaysia for their help in implementing the system on the test vehicle under grant 13H78.

References

[1] F. Sandhu, H. Selamat, and V. B. S. Mahalleh, "Direct yaw control of vehicle using state dependent riccati equation with integral terms," *Adv. Elect. Comput. Eng.*, vol. 16, no. 2, pp. 101–110, 2016.

[2] A. A. Aly, E.-S. Zeidan, A. Hamed, and F. Salem, "An antilock-braking systems (ABS) control: A technical review," *Intell. Control Autom.*, vol. 2, no. 3, pp. 186–195, 2011.

[3] J. O. Pedro, O. T. C. Nyandoro, and S. John, "Neural network based feedback linearisation slip control of an anti-lock braking system," in *Proc. 7th. IEEE Asian Control Conf. (ASCC)*, Aug. 2009, pp. 1251–1257.

[4] G. Liu, Q. Zhang, Y. Wang, and T. Zhou, "An investigation of digital filter technology on ABS wheel speed signal," in *Proc. Int. Conf. Inf. Acquisition*, Jun. 2004, pp. 151–154.

[5] G. Liu, Q. Zhang, J. Xiong, X. Xie, and S. Peng, "An investigation of calculation method of wheel angular acceleration in anti-lock braking system," in *Proc. Int. Conf. Inf. Aut. (ICIA)*, Jun. 2008, pp. 840–843.

[6] B. Yatsalo, V. Didenko, S. Gritsyuk, and T. Sullivan, "*Decerns*: A framework for multi-criteria decision analysis," *Int. J. Comput. Intell. Syst.*, vol. 8, no. 3, pp. 467–489, 2015.

[7] M. Stankovic, M. Naumovic, S. Manojlovic, and S. Simic, "Optimized pure hardware FPGA-based implementation of active disturbance rejection control," *Electrical Engineering*, 2016, pp. 1–11.

[8] A. Ben Atitallah, P. Kadionik, N. Masmoudi, and H. Levi, "FPGA implementation of a HW/SW platform for multimedia embedded systems," *Design Autom. Embedded Syst.*, vol. 12, no. 4, pp. 293–311, Dec. 2008.

[9] G. Lemieux and T. El-Ghazawi, "Designing with extreme parallelism," in *Proc. 16th Int. ACM/SIGDA Symp. Field Program. Gate Arrays*, 2008, pp. 1–2.

[10] A. W. Hill, A. Di Blas, and R. Hughey, "FPGA-based fine-grain parallel computing," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2011, p. 283.

[11] B. Apopei and T. J. Dodd, "FPGA automatic re-synchronisation for pipelined, floating point control systems applications," *Design Autom. Embedded Syst.*, vol. 15, nos. 3–4, pp. 247–288, Dec. 2011.

[12] G. Colón-Bonet and P. Winterrowd, "Multiplier evolution: A family of multiplier VLSI implementations," *Comput. J.*, vol. 51, no. 5, pp. 585–594, 2008.

[13] X. Wang and P. Gupta, "Resource-constrained multiprocessor synthesis for floating-point applications on FPGAs," *ACM Trans. Design Autom. Electron. Syst.*, vol. 16, no. 4, p. 41, 2011.

[14] C. Bobda, T. Haller, F. Muehlbauer, D. Rech, and S. Jung, "Design of adaptive multiprocessor on chip systems," in *Proc. 20th Annu. Conf. Integr. Circuits Syst. Design*, 2007, pp. 177–183.

[15] R. Nijssen, "Challenges and opportunities with place and route of modern FPGA designs," in *Proc. Int. Symp. Phys. Design*, 2016, p. 57.

[16] X. Wang, Y. Liang, Q. Pan, and C. Zhao, "Gaussian filter for nonlinear systems with one-step randomly delayed measurements," *Automatica*, vol. 49, no. 4, pp. 976–986, 2013.

[17] A. K. Yadav, V. K. Mishra, A. K. Singh, and S. Bhaumik, “Unscented Kalman filter for arbitrary step randomly delayed measurements,” in *Proc. Indian Control Conf. (ICC)*, Jan. 2017, pp. 82–86.

[18] S. Trimberger, “Trusted design in FPGAs,” in *Proc. 44th Annu. Design Autom. Conf.*, 2007, pp. 5–8.

[19] W. Meeus, K. Van Beeck, T. Goedemé, J. Meel, and D. Stroobandt, “An overview of today’s high-level synthesis tools,” *Design Autom. Embedded Syst.*, vol. 16, no. 3, pp. 31–51, Sep. 2012.

[20] H. B. Mitchell, *Multi-Sensor Data Fusion: An Introduction*, Berlin, Germany: Springer-Verlag, 2007.

[21] V. Shina, Y. Lee, and T.-S. Choia, “Generalized millman’s formula and its application for estimation problems,” *J Signal Process.*, vol. 86, no. 2, pp. 257–266, Feb. 2006.

[22] M. R. Stanković, S. M. Manojlović, S. M. Simić, S. T. Mitrović, and M. B. Naumović, “FPGA system-level based design of multi-axis ADRC controller,” *Mechatronics*, vol. 40, pp. 146–155, Dec. 2016.

[23] D. P. Singh, T. S. Czajkowski, and A. Ling, “Harnessing the power of fpgas using altera’s opencl compiler,” in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2013, pp. 5–6.

[24] L. Saldanha and R. Lysecky, “Float-to-fixed and fixed-to-float hardware converters for rapid hardware/software partitioning of floating point software applications to static and dynamic fixed point coprocessors,” *Design Autom. Embedded Syst.*, vol. 13, no. 3, pp. 139–157, Sep. 2009.

[25] C. Wang, E. D. Burnham-Fay, and J. D. Ellis, “Real-time FPGA-based Kalman filter for constant and non-constant velocity periodic error correction,” *Precision Eng.*, vol. 48, pp. 133–143, Apr. 2017. [Online]. Available: http://doi.org/10.1016/j.precisioneng.2016.11.013

[26] P. Martín, E. Bueno, F. J. Rodríguez, O. Machado, B. Vuksanovic, “An FPGA-based approach to the automatic generation of VHDL code for industrial control systems applications: A case study of MSOGIs implementation,” *Math. Comput. Simul.*, vol. 91, pp. 178–192, May 2013.

[27] V. Bonato, E. M. George, and A. Constantinides, “A floating-point extended Kalman filter implementation for autonomous mobile robots,” *J. Signal Process. Syst.*, vol. 56, no. 1, pp. 41–50, Jul. 2009.

[28] H. Guo, H. Chen, F. Xu, F. Wang, and G. Lu, “Implementation of EKF for vehicle velocities estimation on FPGA,” *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 3823–3835, Sep. 2013.

[29] M. Burckhardt, *Fahrwerktechnik: Radschlupf-Regelsysteme*. Würzburg, Germany: Vogel, 1993.

[30] S. S. Haykin, Ed. *Kalman Filtering and Neural Networks*. New York, NY, USA: Wiley, 2001.

[31] D. G. Bailey, “Space efficient division on FPGAs,” in *Proc. Electron. Newzealand Conf. (EnzCon)*, Christchurch, New Zealand, 2006, pp. 206–211.

[32] Q. M. Luo, “The Hall automotive wheel speed sensor and simulation for its signal processing,” *Appl. Mech. Mater.*, vols. 494–495, pp. 59–62, Feb. 2014.

[33] R. I. Lörincz and V. Tiponuţ, “A new rotational speed sensor interface circuit with improved EMC immunity,” in *Proc. 14th WSEAS Int. Conf. Circuits*, vol. 4227. 2010, pp. 22–24.

[34] B. Moaveni, M. K. R. Abad, and S. Nasiri, “Vehicle longitudinal velocity estimation during the braking process using unknown input Kalman filter,” *Vehicle Syst. Dyn.*, vol. 53, no. 10, pp. 1373–1392, 2015, doi: 10.1080/00423114.2015.1038279.

Fargham Sandhu (M’15) was born in Ndola, Zambia, in 1973. He received the Ph.D. degree from Universiti Teknologi Malaysia, Malaysia, in 2017, and the master’s degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2008. His current interest is in nonlinear optimal control. He has 11 years of experience, including 4 years teaching experience.

Hazlina Selamat received the degree in electrical and electronics engineering from the Imperial College of Science, Technology and Medicine, University of London, in 1998, and the M.Eng. and Ph.D. degrees in electrical engineering from Universiti Teknologi Malaysia, in 2000 and 2007, respectively. She has been an Associate Professor and a Professional Engineer with the Electrical Engineering Faculty, Universiti Teknologi Malaysia since 2000.

S. E. Alavi was born in Esfahan, Iran, in 1978. He received the Ph.D. degree in electrical engineering from the Universiti Teknolgi Malaysia (UTM) in 2012. From 2013 to 2014, he conducted his post-doctorate research with the Lightwave Communication Research Group, Faculty of Electrical Engineering, UTM, where he has been a Senior Lecturer since 2014. He is currently an outstanding researcher in the area of photonics-based millimeter-wave signal generation for radio over fiber applications.

Vahid Behtaji Siahkal Mahalleh (M’15) was born in Gilan, Iran, in 1984. He received the B.S. degree in electrical engineering from the Islamic Azad University of Lahijan in 2008 and the M.S. degree in control engineering from the Universiti Teknologi Malaysia in 2013, where he is currently pursuing the Ph.D. degree in control engineering.