ELSEVIER

Microprocessors and Microsystems 21 (1997) 257–265

MICROPROCESSORS AND MICROSYSTEMS

High-performance FPGA-based implementation of Kalman filter

C.R. Lee*, Z. Salcic[1]

Department of Electrical and Electronic Engineering, Auckland University, 20 Symonds St., Auckland, New Zealand

Received 19 December 1996; received in revised form 8 July 1997; accepted 9 July 1997

Abstract

An FPGA-based fully hardware Kalman filter has been designed and presented and a reconfigurable Kalman filter-based coprocessor in FPGAs has been proposed. High-speed arithmetic function implementations and pipelining have been used and a substantial improvement in performance has been gained. The cycle time (one iteration) for computing Kalman filter is reduced from 1.8274 μs in our previous design to 0.4013 μs. The performance gained in our approach includes two to four orders of magnitude higher speed than other implementations. The high-speed, recongifuration and easy-to-develop characteristics of the FPGA-based Kalman filter will largely broaden the real-time application area of Kalman filter. © 1997 Elsevier Science B.V.

Keywords: Field programmable gate array (FPGA); Kalman filter; Implementation

1. Introduction

The Kalman filter is an optimal linear estimator which provides the estimation of the signals in noise. Since the Kalman filter was introduced by R.E. Kalman in 1960 [1], it has been widely used in the areas of modern control, signal processing, communication, target tracking, navigation and guidance. It has brought much interest to academic, military and industrial researchers. Being an optimal first-order recursive estimator, Kalman filter provides a real-time algorithm to estimate the unknown state vector recursively for each measurement based on minimization of the mean-square error, which is a measure of the quality of noisy data processing. However, the Kalman filter is also computational intensive because there are many matrix–matrix multiplications and divisions performed in each iteration. The excessive computational requirements have limited the use of the Kalman filter in many real-time applications.

Many different approaches have been proposed to implement original complex Kalman filter-based algorithms (i.e. nonsimplified ones) for reducing the computational time as well as keeping original high precision. Some authors have used multiple parallel transputers to implement an extended Kalman filter [2], a square-root Kalman filter [3], an interacting multiple model (IMM) algorithm [4] and the standard Kalman filter [5]. A speed-up factor of around two to three has been obtained compared with sequential processing in a single transputer. The average cycle times are about 6 ms [3] and 5 ms [4] for calculating a single four-state Kalman filter, and about 7 ms [5] for calculating a 12-state Kalman filter. The other authors combined the power of VLSI and signal processing to develop parallel structures, such as systolic arrays, wavefront arrays and linear arrays [6–8], to implement Kalman filter-based algorithms. Further speed-up has been achieved, with the price of more complicated designs. The best result we have found is 8000 Hz (i.e. 125 μs) for calculating a nine-state standard Kalman filter implemented in multiple Warp cells [8].

Hardware implementation with parallel and pipelining processing provides a solution to complex algorithms calculation. Usually, the performance (speed) of hardware implementation is two to three orders of magnitude better than its software counterpart. By combining the flexibility and high level of logic integration of gate arrays and the benefits of the convenience, ease-of-use and shorter design time, FPGAs have become a mainstream technology for digital system design, especially through the use of FPGAs as reconfigurable computing machines.

Recently, we proposed to use FPGAs to implement the target tracking filter for the multi-target-tracking (MTT) radar system. An FPGA-based Kalman filter for a track-while-scan (TWS) radar system has been designed and

* Corresponding author. Tel.: 0064-9-373-7599 ext. 5386; fax: 0064-9-479-1041; e-mail: cr.lee@auckland.ac.nz

[1] E-mail: z.salcic@auckland.ac.nz

presented in Ref. [9]. In this paper, we present a further improvement of performance by using some inherent features of FPGAs [10] to implement arithmetic functions, and extending it to pipeline architecture. The cycle time (each iteration) has been reduced from 1.8247 μs in our previous design [9] to 0.4031 μs. The performance gained shows two to four orders of magnitude higher speed than other known implementations.

2. Description of implemented Kalman filter

The Kalman filter is the most powerful linear estimator for continuous random variables. Kalman used the state-transition models for dynamic systems. Kalman filter equations can be solved numerically by using a recursive type structure whose outputs only depend on the current inputs and current states (previous outputs). Such a representation of the Kalman filter is suitable for hardware (FPGA) implementation. The example of implementation in our paper is a Kalman tracking filter in a two-dimensional TWS radar which is the same algorithm as in Refs. [9,11]. The system and measurement model equations are

$$X(k+1) = \mathbf{A}X(k) + W(k) \tag{1}$$

$$Y(k) = \mathbf{B}X(k) + V(k) \tag{2}$$

where

$$X(k) = [X_1(k), X_2(k), X_3(k), X_4(k)]^{\mathrm{T}}$$

$$Y(k) = [Y_1(k), Y_2(k)]^{\mathrm{T}}$$

$$W(k) = [0, U_1(k), 0, U_2(k)]^{\mathrm{T}}$$

$$V(k) = [V_1(k), V_2(k)]^{\mathrm{T}}$$

$$\mathbf{A} = \begin{bmatrix} 1 & C & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In our system equation, Eq. (1), there are four state variables including $X_1(k) = \rho(k)$ which represents the range, $X_2(k) = \dot{\rho}(k)$ which represents the range rate, $X_3(k) = \theta(k)$ which represents the angle, and $X_4(k) = \dot{\theta}(k)$ which represents the angle rate. The process noise term $W(k)$ in Eq. (1) includes $U_1(k)$ which represents the change in radial velocity over interval C, and $U_2(k)$ which represents the change in angular velocity over interval C. In our measurement equation, Eq. (2), two sensors measure $Y_1(k)$ which represents the range and $Y_2(k)$ which represents the angle. The measurement noise term $V(k)$ in Eq. (2) includes $V_1(k)$ which represents the range measurement noise and $V_2(k)$ which represents the angle measurement noise. $W(k)$ and $V(k)$ are assumed to be white gaussian noise here. Besides that, T is the matrix transpose operator and C is the sampling cycle time (i.e. the inverse of radar antenna scan rate). The four-state Kalman filter equations can then be written as follows:

$$\mathbf{P}1(k/k-1) = \mathbf{AP}(k-1/k-1)\mathbf{A}^{\mathrm{T}} + \mathbf{Q}(k-1) \tag{3}$$

$$\hat{\mathbf{X}}1(k/k-1) = \mathbf{A}\hat{\mathbf{X}}(k-1/k-1) \tag{4}$$

$$\hat{\mathbf{Y}}(k) = \mathbf{B}\hat{\mathbf{X}}1(k/k-1) \tag{5}$$

$$\mathbf{G}(k) = \mathbf{P}1(k/k-1)\mathbf{B}^{\mathrm{T}}[\mathbf{BP}1(k/k-1)\mathbf{B}^{\mathrm{T}} + \mathbf{R}(k)]^{-1} \tag{6}$$

$$\hat{\mathbf{X}}(k/k) = \hat{\mathbf{X}}1(k/k-1) + \mathbf{G}(k)[Y(k) - \hat{\mathbf{Y}}(k)] \tag{7}$$

$$\mathbf{P}(k/k) = \mathbf{P}1(k/k-1) - \mathbf{G}(k)\mathbf{BP}1(k/k-1) \tag{8}$$

where $\mathbf{P}1(k/k-1)$ is the priori error covariance estimate, $\hat{\mathbf{X}}1(k/k-1)$ is the priori state estimate, $\hat{\mathbf{Y}}(k)$ is the output estimate, $\mathbf{G}(k)$ is the Kalman gain, $\hat{\mathbf{X}}(k/k)$ is the posteriori state estimate, and $\mathbf{P}(k/k)$ is the posteriori error covariance estimate. The system noise covariance matrix $\mathbf{Q}(k)$ and measurement noise covariance matrix $\mathbf{R}(k)$ are

$$\mathbf{Q}(k) = E[\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(k)] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma_1^2(k) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_2^2(k) \end{bmatrix} \tag{9}$$

$$\mathbf{R}(k) = E[V(k)V^{\mathrm{T}}(k)] = \begin{bmatrix} \sigma_\rho^2(k) & 0 \\ 0 & \sigma_\theta^2(k) \end{bmatrix} \tag{10}$$

where $\sigma_1^2(k) = E[U_1^2(k)]$ and $\sigma_2^2(k) = E[U_2^2(k)]$ are the variances of C multiplied by the radial and angular acceleration, respectively, and $\sigma_\rho^2(k) = E[V_1^2(k)]$ and $\sigma_\theta^2(k) = E[V_2^2(k)]$ are the variances of C multiplied by the radial and angular measurement noise, respectively. The Kalman filter can then be represented as shown in Fig. 1.

As a numerical example in Ref. [11], we have range = 160 km (= 100 miles), scan cycle time C = 15 s, a maximum acceleration M = 2.1 m s^{-2}, the r.m.s. noise in the range sensor = 10^3 m (i.e. $\sigma_\rho = 10^3$ m), and the r.m.s. noise in the bearing sensor = 0.017 rad (i.e. σ_θ = 0.017 rad). Then, the noise variances in the $\mathbf{Q}$ and $\mathbf{R}$ matrices can be calculated as $\sigma_1^2 = 330$, $\sigma_2^2 = 1.3 \times 10^{-8}$, $\sigma_\rho^2 = 10^6$, $\sigma_\theta^2 = 2.9 \times 10^{-4}$. The initialization data of Kalman filter can be derived by using first two measurements, range and angle, at times k = 1 and k = 2 as follows:

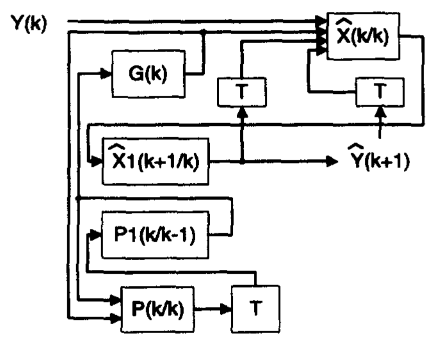


Fig. 1. The block diagram of Kalman filter.

$$\hat{\mathbf{X}}(2)=\hat{\mathbf{X}}(2/2)=\begin{bmatrix} Y_1(2) \\ (Y_1(2)-Y_1(1))\ /\ T \\ Y_2(2) \\ (Y_2(2)-Y_2(1))\ /\ T \end{bmatrix} \tag{11}$$

Then, the error covariance matrix **P**(2) (or **P**(2/2)) can be calculated by

$$\mathbf{P}(2)=\mathbf{P}(2/2)=E\{[X(2/2)-\hat{\mathbf{X}}(2/2)][X(2/2)-\hat{\mathbf{X}}(2/2)]^{\mathrm{T}}\}$$

where

$$X(2/2)-\hat{\mathbf{X}}(2/2)=\begin{bmatrix} -V_1(2) \\ U_1-(V_1(2)-V_1(1))\ /\ T \\ -V_2(2) \\ U_2(1)-(V_2(2)-V_2(1))\ /\ T \end{bmatrix} \tag{12}$$

Due to the independence of noise sources U and V, and also the independence between individual noise samples, **P**(2/2) can be simplified as

$$\mathbf{P}(2/2)=\begin{bmatrix} P_{11} & P_{12} & 0 & 0 \\ P_{21} & P_{22} & 0 & 0 \\ 0 & 0 & P_{33} & P_{34} \\ 0 & 0 & P_{43} & P_{44} \end{bmatrix}$$

$$=\begin{bmatrix} 10^6 & 6.7\times10^4 & 0 & 0 \\ 6.7\times10^4 & 0.9\times10^4 & 0 & 0 \\ 0 & 0 & 2.9\times10^{-4} & 1.9\times10^{-5} \\ 0 & 0 & 1.9\times10^{-5} & 2.6\times10^{-6} \end{bmatrix} \tag{13}$$

From $\hat{\mathbf{X}}(2/2)$, $\hat{\mathbf{X}}1(3/2)$ and $\hat{\mathbf{Y}}(3)$ can be calculated from Eqs. (4) and (5), respectively. Now, we have first data set [**P**(2/2), $\hat{\mathbf{X}}1(3/2)$, $\hat{\mathbf{Y}}(3)$] and measurement inputs $Y(3)$, and we can use Eqs. (3)–(8) to calculate next data set [**P**(3/3), $\hat{\mathbf{X}}1(4/3)$, $\hat{\mathbf{Y}}(4)$] and so on. This is the operation of the Kalman filter to track the target in TWS radar system.

3. FPGAS as implementation technology

Traditional hardware implementations used SSI/MSI chips to design digital logic circuit. With the advent of VLSI circuits, the cost of hardware implementations has been reduced and the performance (speed) of the systems can also be improved. Some authors have combined the power of VLSI and parallel signal processing algorithms, e.g. systolic architecture, to implement complex algorithms, e.g. the Kalman filter, and achieved noticeable performance [6–8].

Since the mid-1980s a new VLSI technology, the field programmable gate arrays (FPGAs) [12] was introduced to implementing digital logic. The FPGA architectures include a matrix of logic cells, a perimeter of input/output cells and programmable interconnections. The advanced architectures of newer FPGAs also include internal three-state buffers, I/O registers, dedicated arithmetic carry logic, on-chip memory resources, multiple low-skew clock networks, programmable output slew rate controls, and other system-integration features. The FPGA designs are easy to modify because schematics and high-level hardware description languages (HDLs) can be used as the design entry tools on PCs or workstations. The development tools automatically map the designs into FPGAs logic resources and carry out necessary routing. In addition, FPGA designs can be easily verified by the simulation in the development environment or/and the testing in the target system once the design is completed and compiled.

Compared to PLDs, many more logic and logic levels can be implemented in FPGAs. Compared to MPGAs, the cost for small volume parts is greatly reduced, long fabrication delay has been avoided and security can be kept for the military use because FPGA does not have to be custom-fabricated. The FPGA technology has advanced dramatically over the past few years. SRAM-based FPGAs have brought a new and unique capability to electronic systems, enabling them to change a system logic function simply by reconfiguring the FPGA. On the other hand, hardware implemented in other logic technologies cannot be changed once implemented in a system. Therefore, by combining the flexibility, convenience, ease-of-use, and time-to-market advantages of a user-programmable device, FPGAs have rapidly become a mainstream logic technology and brought a big impact on digital system design.

The capability of FPGA to reconfigure, especially for SRAM-based FPGAs, is used in our approach to design custom-configurable, application-specific processors that implement the Kalman filter and enable easy integration into wider systems which use Kalman filters. This approach is described in the following section.

4. Reconfigurable Kalman filter-based coprocessors system

In our design, the Kalman filter is a fully hardware type which is implemented in FPGAs and it can be considered as

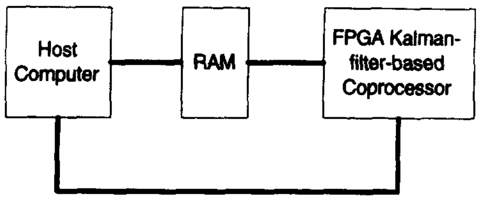


Fig. 2. Basic structure of FPGA-based Kalman filter coprocessor system.

a coprocessor to a host computer in a wider control system. The basic system architecture is shown in Fig. 2. From this figure we see that the Kalman filter is controlled by the host computer, but also has access to a shared memory as the source of input data prepared by the host computer.

FPGA-based reconfigurable digital systems provide a radical change in the forms of computation and digital logic. FPGAs provide 'virtual prototype' without the time delay and inflexibility of prototype fabrications. Because it is a multi-mode system, they reduce the complexity and inventory risk for logic implementation and furnish significant hardware savings by supplying truly generic hardware due to the characteristics of reconfiguration. An FPGA-based system, plus several different configurations stored in an external ROM, can be used for multi-mode hardware, with the functionality on the chip changed to suit current demands.

Various Kalman filter-based algorithms, e.g. extended Kalman filter and interacting multiple model (IMM), are widely used in many different areas. In some cases, different algorithms are required for different system states, and all these algorithms can be implemented and stored in ROM and reconfigurated in FPGAs to function with the current demands without hardware modification. The reconfigurable Kalman filter-based coprocessor system logical structure is shown in Fig. 3.

Dynamically reconfigurable systems, in which FPGA can be partially reconfigurated at run time, represent a new form of digital systems. Selection of different configurations in FPGAs is similar to a selection of different subroutines in software. A specific architecture configuration corresponding to an algorithm can be chosen from memory and configured in hardware which can be accomplished without wiring modification, and with greatly enhanced processing performance, on the time-multiplexed basis.

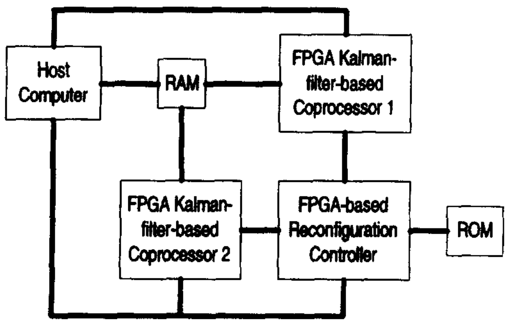

Host Computer
RAM
FPGA Kalman-filter-based Coprocessor 1
FPGA Kalman-filter-based Coprocessor 2
FPGA-based Reconfiguration Controller
ROM

Fig. 3. Dynamically reconfigurable Kalman-filter-based coprocessor system.

The host computer shown in Figs. 2 and 3 can, in practice, be any general-purpose computer or a very small general-purpose processor core [13] with custom-configurable hardware to bridge the gap between a general-purpose and a Kalman filter application-specific computer. In the later case, the Kalman filter appears as a functional unit to the processor core which is invoked by a single user-specific instruction. The advantage of this approach is that it allows the users to use general programming facilities to design the broader application in which the Kalman filter is invoked by a single instruction. Depending on performance requirements, several Kalman filter-based algorithms can be implemented in separate functional units, or configurated in the same hardware from external ROM, and invoked from the main program sequence (thread). The emphasis in the rest of this paper is to consider efficient implementation of the Kalman filter as a separate functional unit, rather than as a general system.

5. Algorithm implementation

5.1. Parallelism

By using FPGAs technology, we can apply the maximum of parallelism to implement all linearized algorithms. The fully hardware implementation can reach the highest performance (speed) by using parallel processing to compute every scalar equation at the same time. Then the Kalman filter algorithm formulas can be decomposed into many scalar linear equations which are composed of simple arithmetic operations, i.e. addition, subtraction, multiplication and division. The Kalman filter equations [Eqs. (3)–(8)] have to be rewritten to the scalar form. From Eqs. (1), (3) and (9), the priori error covariance estimate $\mathbf{P}1(k/k-1)$ in Eq. (3) can be rewritten as follows:

$$\mathbf{P}1(k/k-1)=\mathbf{AP}(k-1/k-1)\mathbf{A}^{\mathrm{T}}+\mathbf{Q}(k)$$

$$=\begin{bmatrix} P1_{11} & P1_{12} & 0 & 0 \\ P1_{21} & P1_{22} & 0 & 0 \\ 0 & 0 & P1_{33} & P1_{34} \\ 0 & 0 & P1_{43} & P1_{44} \end{bmatrix} \tag{14}$$

where

$$P1_{11}=P_{11}+CP_{12}+CP1_{12}$$

$$P1_{12}=P_{12}+CP_{22}$$

$$P1_{21}=P_{21}+CP_{22}$$

$$P1_{22}=P_{22}+\sigma_1^2$$

$$P1_{33}=P_{33}+CP_{34}+CP1_{34}$$

$$P1_{34}=P_{34}+CP_{44}$$

$$P1_{43} = P_{43} + CP_{44}$$

$$P1_{44} = P_{44} + \sigma_2^2$$

To minimize the hardware circuit in fixed data accuracy or to maximize data accuracy in a fixed number of bits of hardware circuit, we can normalize $P1_{11}$, $P1_{12}$, $P1_{21}$ and $P1_{22}$ by dividing by σ_1^2 in each term and normalize $P1_{33}$, $P1_{34}$, $P1_{43}$, $P1_{44}$, and P_{33}, P_{34}, P_{43}, P_{44} by dividing by σ_2^2. Therefore, the initialization data $\mathbf{P}(2/2)$ can be rewritten as follows:

$$\mathbf{P}(2/2) = \begin{bmatrix} \mathrm{P}_{11} & \mathrm{P}_{12} & 0 & 0 \\ \mathrm{P}_{21} & \mathrm{P}_{22} & 0 & 0 \\ 0 & 0 & \mathrm{P}_{33} & \mathrm{P}_{34} \\ 0 & 0 & \mathrm{P}_{43} & \mathrm{P}_{44} \end{bmatrix} =$$

$$\begin{bmatrix} 3.03 \times 10^3 & 2.03 \times 10^2 & 0 & 0 \\ 2.03 \times 10^2 & 3.0 \times 10^1 & 0 & 0 \\ 0 & 0 & 2.2308 \times 10^4 & 1.462 \times 10^3 \\ 0 & 0 & 1.462 \times 10^3 & 2 \times 10^2 \end{bmatrix}$$

From Eqs. (1) and (4), the priori state estimate $\hat{\mathbf{X}}1(k+1/k)$ can be rewritten as follows:

$$\hat{\mathbf{X}}1(\mathrm{k}+1/\mathrm{k}) = \mathbf{A}\hat{\mathbf{X}}(k/k) = [X1_1 X1_2 X1_3 X1_4]^{\mathrm{T}} \qquad (15)$$

where

$$X1_1 = X_1 + CX_2$$

$$X1_2 = X_2$$

$$X1_3 = X_3 + CX_4$$

$$X1_4 = X_4$$

From Eqs. (2), (5) and (15), the output estimate $\hat{\mathbf{Y}}(\mathrm{k}+1)$ can be rewritten as follows:

$$\hat{\mathbf{Y}}(k+1) = \mathbf{B}\hat{\mathbf{X}}1(k+1/k) = [\hat{\mathbf{Y}}_1 \hat{\mathbf{Y}}_2]^{\mathrm{T}} \qquad (16)$$

where

$$\hat{\mathbf{Y}}_1 = X/1$$

$$\hat{\mathbf{Y}}_2 = X/3$$

From Eqs. (2), (6), (10) and (14), the Kalman gain can be rewritten as follows:

$$\mathbf{G}(k) = \mathbf{P}1(k/k-1)\mathbf{B}^{\mathrm{T}}[\mathbf{B}\mathbf{P}1(k/k-1)B^{\mathrm{T}} + \mathbf{R}(k)]^{-1}$$

$$= \begin{bmatrix} G_{11} & 0 \\ G_{21} & 0 \\ 0 & G_{32} \\ 0 & G_{42} \end{bmatrix} \qquad (17)$$

where

$$G_{11} = P1_{11}/(P1_{11} + \sigma_\rho^2)$$

$$G_{21} = P1_{21}/(P1_{11} + \sigma_\rho^2)$$

$$G_{32} = P1_{33}/(P1_{33} + \sigma_\theta^2)$$

$$G_{42} = P1_{43}/(P1_{33} + \sigma_\theta^2)$$

From Eqs. (7), (16) and (17), the posteriori state estimate can be rewritten as follows:

$$\hat{\mathbf{X}}(k/k) = \hat{\mathbf{X}}1(k/k-1) + \mathbf{G}(k)[Y(k) - \hat{\mathbf{Y}}(k)] = [X_1 X_2 X_3 X_4]^{\mathrm{T}} \qquad (18)$$

where

$$X_1 = X1_1 + G_{11}(Y_1 - \hat{\mathbf{Y}}_1)$$

$$X_2 = X1_2 + G_{21}(Y_1 - \hat{\mathbf{Y}}_1)$$

$$X_3 = X1_3 + G_{32}(Y_2 - \hat{\mathbf{Y}}_2)$$

$$X_4 = X1_4 + G_{42}(Y_2 - \hat{\mathbf{Y}}_2)$$

From Eqs. (2), (8), (14) and (17), the posteriori error covariance estimate can be rewritten as follows:

$$\mathbf{P}(k/k) = \mathbf{P}1(k/k-1) - \mathbf{G}(k)C\mathbf{P}1(k/k-1)$$

$$= \begin{bmatrix} P_{11} & P_{12} & 0 & 0 \\ P_{21} & P_{22} & 0 & 0 \\ 0 & 0 & P_{33} & P_{34} \\ 0 & 0 & P_{43} & P_{44} \end{bmatrix} \qquad (19)$$

where

$$P_{11} = P1_{11} - P1_{11}G_{11}$$

$$P_{12} = P1_{12} - P1_{12}G_{11}$$

$$P_{21} = P1_{21} - P1_{11}G_{21}$$

$$P_{22} = P1_{22} - P1_{12}G_{21}$$

$$P_{33} = P1_{33} - P1_{33}G_{32}$$

$$P_{34} = P1_{34} - P1_{34}G_{32}$$

$$P_{43} = P1_{43} - P1_{33}G_{42}$$

$$P_{44} = P1_{44} - P1_{34}G_{42}$$

From the above equations, the hardware circuit of the Kalman filter can be decomposed to three parts [9]. The first part of the Kalman filter circuit is the calculation of P_{11}, P_{12}, P_{21}, P_{22} and G_{11}, G_{21}. The second part of the Kalman filter circuit is the calculation of P_{33}, P_{34}, P_{43}, P_{44}, and G_{32}, G_{42}. The third part of the Kalman filter circuit is the calculation of $\hat{\mathbf{X}}_{11}$, $\hat{\mathbf{X}}_{12}$, $\hat{\mathbf{X}}_{13}$, $\hat{\mathbf{X}}_{14}$, and $\hat{\mathbf{Y}}_1$, $\hat{\mathbf{Y}}_2$. Because the elements of $\hat{\mathbf{X}}(2/2)$ and $\mathbf{P}(2/2)$ are all constant, we can select the number of bits needed in each part of the Kalman filter to achieve desired accuracy. In part 1, we require 13 bits for P_{11}, P_{12}, P_{21} and P_{22}. In parts 2 and 3, we require 16 bits for P_{33}, P_{34}, P_{43}, P_{44}, and $X1_1$, $X1_2$, $X1_3$, $X1_4$ and Y_1, Y_2. Additionally, we select eight bits for G_{11}, G_{21}, G_{32} and G_{42} for sufficient accuracy. In the Kalman filter circuit [9], each part of the Kalman filter includes arithmetic elements such as adder, subtracter, multiplier, and/or divider, and form a tree-like structure.

5.2. Arithmetic functions design

If the model is nonlinear, a linearization is required, or look-up tables (LUTs) in FPGAs can be used to furnish nonlinear functions. Here, the Kalman filter is based on a linear model. Only arithmetic operations, e.g. addition, subtraction, multiplication, and division, should be efficiently implemented in FPGAs. In our new design, we use an lpm_add_sub function from Altera's library [10] for the implementation of adders/subtracters. It enables efficient implementation of adders/subtracters of desired length, which appears as its parameter.

For the design of generic multiplier we use a Wallace tree and the Baugh and Wooley method [14] to speed up multiplication. We also use a number of instances of adder/subtracter elements to design divider whose regular geometric pattern lends itself well to modifications in the future. In our design environment, all designed modules become new library macrofunctions which can be used in other designs and in the future modifications of existing one.

5.3. Ripple-carry adders

In this design, we used Altera FLEX 8000 devices and respective logic elements in arithmetic mode which offers two three-input look-up tables (LUTs). These LUTs are suitable to implement adders. One LUT provides a three-bits summation function and the other LUT gives a carry-out signal. The carry chain feature used in the ripple-carry adders provides the easiest way to implement adder in the FLEX 8000 architecture. Only less than 1 ns is required to forward carry signal to adjacent logic elements (LEs) within a logic array block (LAB) and to adjacent LABs. The ripple-carry adders designed in FLEX 8000 arithmetic mode offer the best performance per LE because both the sum and carry generation logic for each adder are implemented within a single LE.

We use the lpm_add_sub macrofunction to design ripple-carry adders/subtracters which were used in the design of multipliers and dividers of different specifications. Compared to our previous implementation [9], the computation times were reduced for adder/subtracter (18 bits) from 155 to 51 ns, for multiplier (17 × 8 bits) from 248 to 84.9 ns, and for most time consuming element—the divider (17 bits of divisor and dividend, and eight bits of quotient) from 1100 to 380 ns.

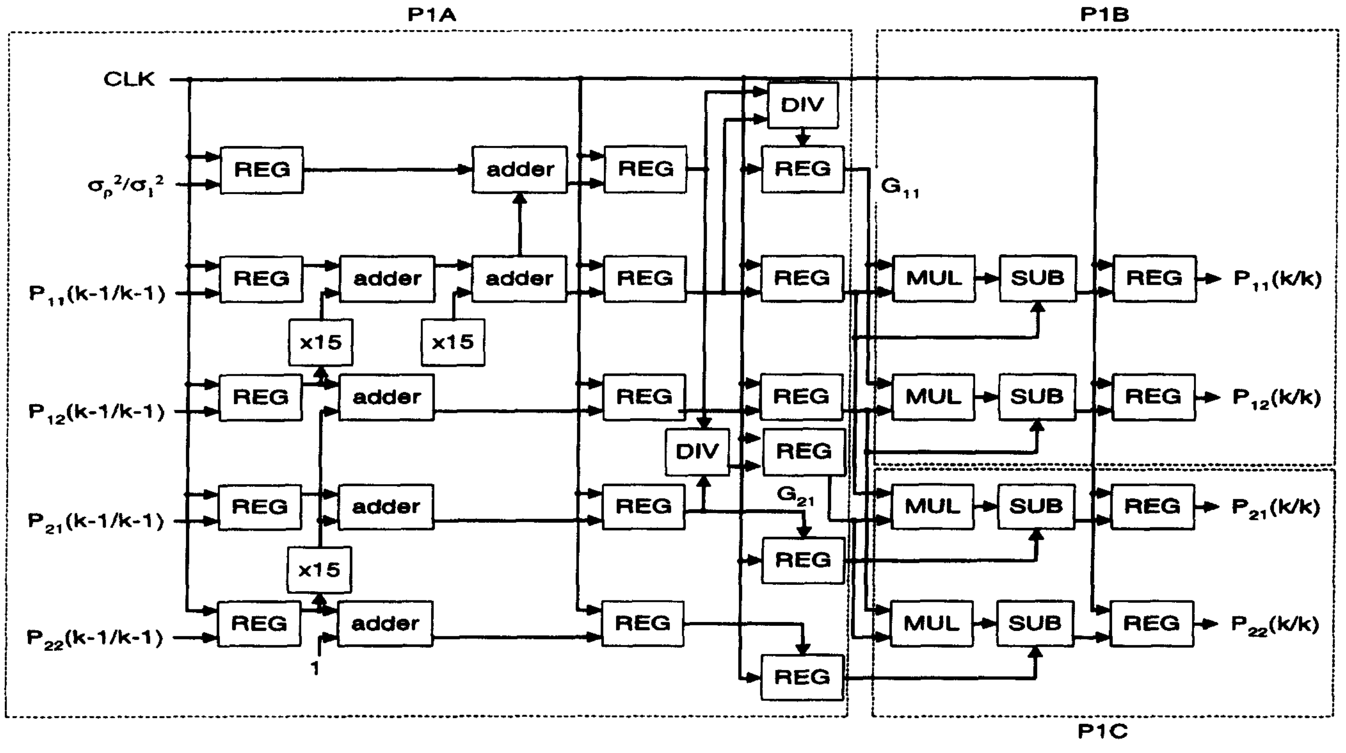


Fig. 4. Pipelined first part of FPGA-based Kalman filter.

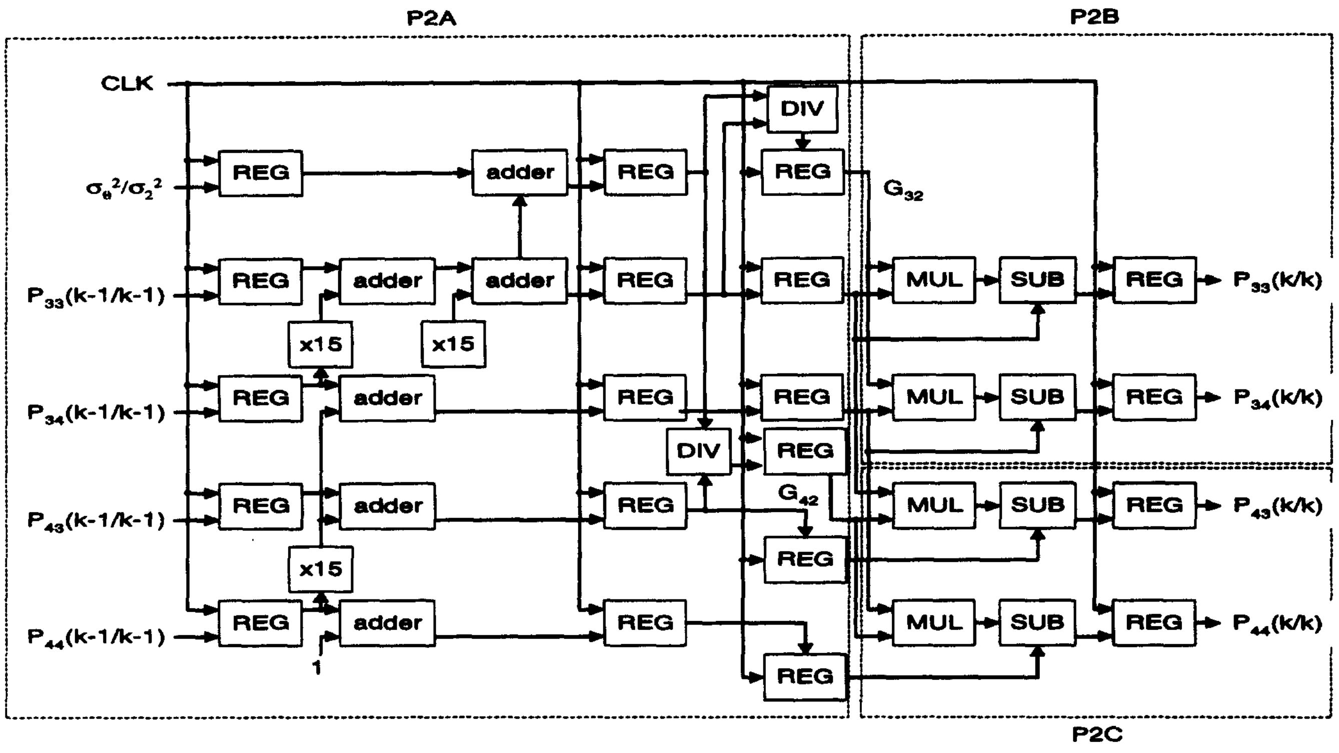


Fig. 5. Pipelined second part of FPGA-based Kalman filter.

5.4. Pipelining

To increase the speed further, pipelining have been proved to be a good choice to use in our new implementation. The overall circuitry is split into three stages using registers. The maximum computation time in each stage determines the new cycle (system clock) time. The three parts of our new FPGA-based Kalman filter can be seen in Figs. 4–6 where 'REG' represents the register, 'DIV' the divider, 'MUL' the multiplier, 'SUB' the subtracter, and 'CLK' the system clock.

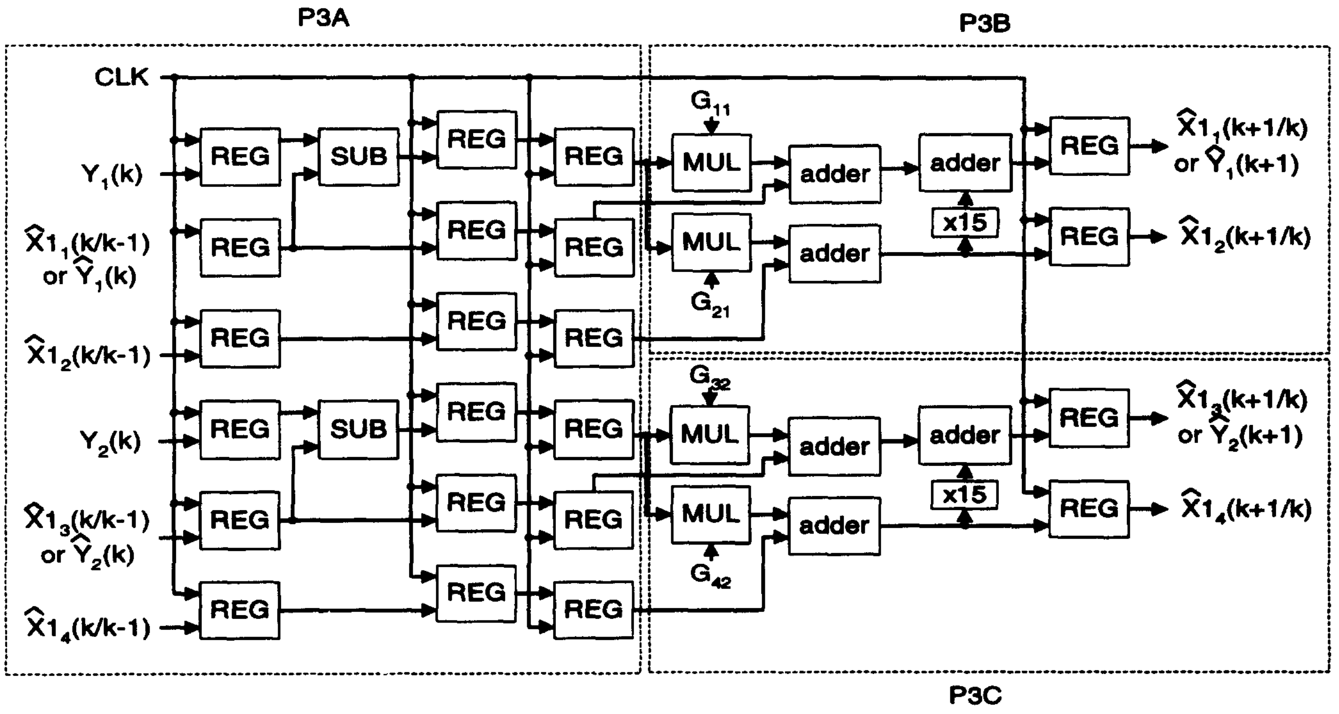


Fig. 6. Pipelined third part of FPGA-based Kalman filter.

Table 1
The compilation results of Kalman filter

Kalman filter	Devices	Logic cells
P1A	EPF81500ARC240	858
P1B	EPF8820AQC160	602
P1C	EPF8820AQC160	600
P2A	EPF81500GC280	911
P2B	FPF81188AQC208	764
P2C	EPF81188AQC208	829
P3A	EPF81500GC280	320
P3B	EPF81188AQC208	649
P3C	EPF81188AQC208	651
Total		6184

5.5. Implementation

The Kalman filter was designed using Altera Max + PlusII programmable logic development software which provides an architecture independent design environment and supports the design for Altera's FPLDs. Using Max + PlusII, we manually partitioned each part of Kalman filter into several devices to get the optimal results. Each part of the Kalman filter shown in Figs. 4–6 can be divided into three areas, and each area can be fitted into a single FPGA chip. Therefore, we have nine chips in total.

After compiling this Kalman filter by choosing FLEX 8000 series of Altera standard chips in manually chip selecting mode, the results have shown that nine chips (devices) are required for this Kalman filter. Table 1 shows nine devices selected and the number of logic cells required.

Although a number of registers have been added to support pipelining, the number of logic elements necessary for this design is almost the same as in our previous design. This is because the FLEX 8000 logic cell contains a register (flip–flop) which was not used in our previous design. The simulator was used to test the logic operation and internal timing of the circuit. The worst-case computational cycle time is 0.4031 μs.

5.6. Comparison with other implementations

The most important improvement to our previous design is the computational time, which has been substantially reduced. The maximum computational time (worst case) is 0.4013 μs which can be kept in the same level even when the number of system states and measurements increases. The performance gained in our approach includes about two to four orders of magnitude higher speed than other implementations [3–8]. Compared to our previous design result, 1.8247 μs [9], we still obtain a 4.5 times better performance due to the use of the ripple-carry feature of macrofunction 'lpm_add_sub' and three-stage pipeline function in the circuit. This design results in the increased tracking accuracy, decreased track loss and increased tracking target number. They are all advantages in MTT radar.

Although the development time of the first version design of FPGA-based Kalman filter is longer than that of the software-based one, the flexibility of high-level development tools and the reusability of macrofunctions and libraries make the development time of the former one and that of the latter one comparable.

6. Conclusions and future work

The Kalman filter, a linear optimal estimator, is a popular and complex algorithm. The same algorithm in our previous work has been implemented in nine FPGA chips and whole cycle time for one iteration has been greatly reduced from 1.8247 μs in our previous implementation [9] to 0.4031 μs. The characteristics of parallelism, pipelining and ripple-carry function of FLEX 8000 devices have been used to speed up the computation cycle of the Kalman filter. The implementation of the Kalman filter is already near optimal, because of the limitations of performance of the slowest arithmetic element, divider (380 ns).

Due to the advantages of user programmability, high capacity, simple bug fixes and design modifications, fully tested devices, instant manufacturing changes, and reconfiguration in the run time, FPGAs have surpassed other custom and semicustom ASIC technologies, principally in prototype design, initial system production and small quantity production. We therefore believe the FPGA-based Kalman filter can largely broaden the real-time application area of the Kalman filter in the future. The customized Kalman-filter-based application-specific system, the reconfiguration controller and the different applications of FPGA-based Kalman filter are the next targets in our research.

References

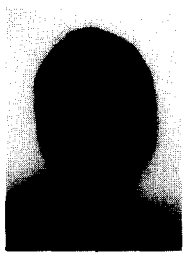
[1] R.E. Kalman, A new approach to linear filtering and prediction problems, Journal of Basic Engineering 82D (1960) 34–45.
[2] K.D. Rao, G. Sridhar, Improving performance in pulse radar detection using neural networks, IEEE Transactions of Aerospace Electronic Systems 31 (1995) 1193–1198.
[3] L.P. Maguire, G.W. Irwin, Transputer implementation of Kalman filters, IEE Proceedings D 138 (1991) 355–362.
[4] D.P. Atherton, H.J. Lin, Parallel implementation of IMM tracking algorithm using transputer, IEE Proceedings—Radar, Sonar Navigation 141 (1994) 325–332.
[5] W.J. Wilson, C.C.W. Hulls, G.S. Bell, Relative end-effector control using Cartesian position based visual servoing, IEEE Transactions on Robotics and Automation 12 (1996) 684–696.
[6] J.M. Jover, T. Kailath, A parallel architecture for Kalman filter measurement update and parameter estimation, Automatica 22 (1986) 32–57.
[7] S.Y. Kung, J.N. Hwang, Systolic array designs for Kalman filtering, IEEE Transactions on Signal Processing 39 (1991) 171–182.
[8] R.S. Baheti, D.R. O'Hallaron, H.R. Itzkowitz, Mapping extended Kalman filters onto linear arrays, IEEE Transactions on Automatic Control 35 (1990) 1310–1319.

[9] C.R. Lee, Z. Salcic, A fully-hardware-type maximum-parallel architecture for Kalman tracking filter in FPGAs, accepted for ICICs'97 First International Conference on Information, Communications and Signal Processing, Singapore, September, 1997.
[10] Altera Corporation Data Book, 1996.
[11] S.M. Bozic, Digital and Kalman Filtering, Edward Arnold, London, 1979, pp. 136–140.
[12] W.S. Carter, The future of programmable logic and its impact on digital system design, IEEE International Conference on Computers, 1994, pp. 10–16.
[13] Z. Salcic, B. Maunder, CCSimP—An Instruction-level Custom-Configurable Processor for FPLDs, Springer Computer Science Series, Berlin, no. 1142, 1996, pp. 280–289.
[14] N.R. Scott, Computer Number Systems and Arithmetic, Prentice-Hill, Englewood Cliffs, NJ, 1985, pp. 33–139.

C.R. Lee was born in Taiwan, Republic of China, on 28 December, 1951. He received the B.S. degree and first M.S. degree (in automatic control area) in Electrical Engineering from Chung Cheng Institute of Technology, Taiwan, in 1974 and 1981, respectively. He finished the Ph.D. course work and received the second M.S. degree (in the field of semiconductors) in Electrical and Computer Engineering from the University of Texas at Austin, USA, in 1990.

He worked as a Platoon Leader for military service in Chinese Army from 1974 to 1976, and as a teacher in the Chinese Army Communication and Electronics School from 1977 to 1979. From 1981 to 1994, he worked in the Chung Shan Institute of Science and Technology which is the biggest military research center in Taiwan. He was an Assistant Scientist from 1981 to 1983 and the Radar Computer and Control Group Leader from 1983 to 1985 in the Radar Section. He was the Vice-Section Head in the Antenna Section from 1985 to 1987, in charge of the antenna testing systems. From 1990 to 1994 he was responsible for missile integrated testing and quality assurance.

He is currently working toward the Ph.D. degree at the University of Auckland. His research interests include the application of field programmable gate arrays (FPGAs) and Kalman filter, algorithm implementation, complex digital system design and target tracking.



Z. Salcic was born in Sarajevo, Bosnia and Herzegovina, in 1950. He received the B.E., M.E. and Ph.D. degrees in Electrical Engineering from Sarajevo University in 1972, 1974, and 1976, respectively. Part of his graduate work was undertaken at the City College of the City University New York in 1974 and 1975. He worked as an assistant professor and associate professor at the Sarajevo University and Czech Technical University, Prague. Between 1985 and 1990 he was Deputy and then CEO of the Institute for Computer and Information Systems of Energoinvest Corporation, Sarajevo. He has been with the Auckland University since 1994.

He has published over 70 technical papers, numerous technical reports and four books. His current research interests are custom-computing machines, field-programmable logic and its applications in embedded and reconfigurable systems, complex digital systems design, automatic vehicle tracking, and applications of mobile computing.