FPGA Coprocessor-based IMM-SDCMKF Application to a Radar Maneuvering Target Tracking System

Bao-bao Wang, Quan-sheng Wang, Lian-zhong Zhang
Jiangsu Automation Research Institute
No.18, Sheng Hu Road, Lianyungang 222006
PEOPLE'S REPUBLIC OF CHINA
wangbaobao_zdh@126.com, wangquansheng_zdh@126.com, zhanglianzhong_zdh@126.com

Abstract: - This paper presents a maneuvering target tracking system using Interacted Multiple Model-Second Debiased Converted Measurement Kalman Filter (IMM-SDCMKF) based on FPGA coprocessor. IMM-SDCMKF is considered one of the most effective algorithms for maneuvering target tracking. However, the computation of this algorithm is time-cost and complex. The traditional design approach is unable to meet the high-speed real-time signal processing needs. In this tracking system, the FPGA is used as a coprocessor of the DSP, and the large amount of calculation of IMM-SDCMKF algorithm is realized in FPGA. DSP is in charge of the scheduling of the total tracking algorithm and the control of the data stream, which resolves the problem of the concurrency and real time in the realization of the single DSP scheme. The designed tracking system ensures the accuracy of the data processing as well. The experiment results show that the designed scheme meets the high precision and real time of the maneuvering target tracking system.

Key-Words: Maneuvering target tracking, IMM-SDCMKF, FPGA, Coprocessor, Real-time

1 Introduction

Maneuvering target tracking is widely used in the military field. Providing high-precision target information is the main task of maneuvering target tracking. With the growing maneuverability of new weapons, the effectiveness of maneuvering target tracking algorithm and the real-time performance become more demanding. The main problems of maneuvering target tracking are to establish target tracking model and to select the appropriate filter tracking algorithm.

Interacted Multiple Model (IMM) algorithm is considered the most effective algorithm for manoeuvring target tracking. IMM belong to variable maneuvering structure multiple-model calculation of determinate structure and allocate a certain transition probability to each model by increasing interaction among different models. Not by detecting the target maneuver but switching and updating model probability between models，the final states of output targets are realized through weighting fusion approach. IMM show a well-acknowledged tracking effect of maneuvering target, receiving wide-spread attention from domestic and foreign scholars.

In the system of radar maneuvering target tracking, the dynamic target is usually modeled and tracked in the Cartesian coordinates, whereas the measurements are provided in terms of range and angle with respect to the radar sensor location in the spherical coordinate. Therefore, radar target tracking becomes a kind of non-liner estimation [1]. General method is the use of extended Kalman filter (EKF). Another method is to use the debiased converted measurement Kalman filter (DCMKF) [2]. The standard DCMKF is commonly derived from a first order Taylor expansion of the state dynamics and measurement model. The truncation of Taylor series covers the bias of converted measurement error, which may lead to linearization error and divergence because of dealing with maneuvering targets as a type of nonlinear actual systems. However this problem can be avoided to some extent by using the second-order term of Taylor series [3]. In this paper, the second-order debiased converted measurement Kalman filter (SDCMKF) is applied to radar target tracking.

This paper presents a maneuvering target tracking system using IMM-SDCMKF algorithm. In the traditional software system design, IMM-SDCMKF is usually realized by DSP, which would be restricted by the serial instruction stream due do the complex computations of IMM-SDCMKF and unable to meet the high-speed real-time signal processing needs. However, using the hardware parallel architecture feature of FPGA to realize IMM-SDCMKF can resolve the problem of high precision and real time.

Nowadays, how to make use of FPGA to realize target tracking algorithm is always a hotspot and difficulty in engineering field. Many scholars have done a great deal of research in this field. In [4], the fundamental arithmetics of floating point numbers based on FPGA are presented. In [5], the algorithms of matrix operation based on FPGA are optimized. In [6,7],the simulation of KF based on FPGA is realized. In [8], a fixed point KF based on FPGA is designed. In [9,10], the optimized FPGA-based EKF is designed and applied to control synchronous motors. In [11], a floating point FPGA-based self-tuning regulator is proposed. In [12], the radar target tracking system based on DSP/FPGA is designed. This paper utilizes FPGA as a coprocessor to realize IMM-SDCMKF algorithm, which can satisfy the requirement of real time and high precision of the system, as well as simplify the difficulty of system design.

2 General system design

In this paper, we adopt DSP TMS320VC5509A chip as core processor of the radar maneuvering target tracking system. This fixed point DSP is responsible for the scheduling of the whole tracking algorithm and the control of data stream. The FPGA EP3C120F484C8N chip is adopted as floating point coprocessor of DSP. DSP receives radar measurements values then transmits to FPGA. FPGA informs DSP to retrieve the filtered system status values when one frame data is processed.

2.1 Hardware design of the system

The structure diagram of System hardware design is shown as in Fig. 1, This system is composed of DSP, FPFA, FLASH, synchronous dynamic random access memory (SDRAM), secure digital (SD) card, electrically erasable programmable read-only memory (EEPROM) and some peripheral interface circuit, wherein DSP is in charge of the scheduling of the total tracking algorithm and the control of the data stream; FPGA is used to realize the complicated IMM-SDCMKF algorithm; FLASH is used to store DSP program; SDRAM is used to buffer pre-filtered and post-filtered data; SD card is used to store both filtered and unfiltered data; EEPROM is used to store FPGA program.

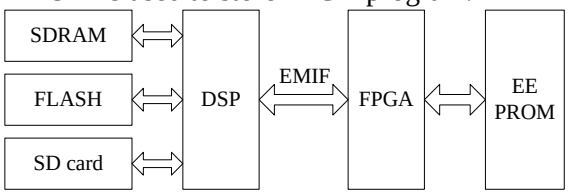

Fig.1. The structure diagram of system hardware design

Fig.2 shows the hardware connection diagram between DSP and FPGA. EMIF connects DSP and FPGA. Wherein, $\overline{CE}$ is chip electing signal, $\overline{AOE}$ is asynchronous output enable signal, $\overline{AWE}$ is asynchronous writing electing signal, $\overline{ARE}$ is asynchronous reading enable signal, $\overline{INT}$ is interrupt enable signal, D[7:0] is data bus signal and A[7:0] is address bus signal.

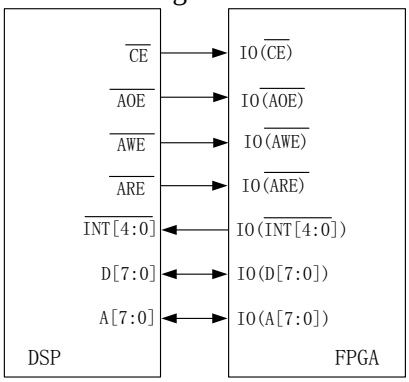

Fig.2. The hardware connection diagram between DSP and FPGA

Fig. 3 shows the sequence diagram of writing operation between DSP and FPGA. $\overline{AOE}$ and $\overline{ARE}$ are set high when writing.

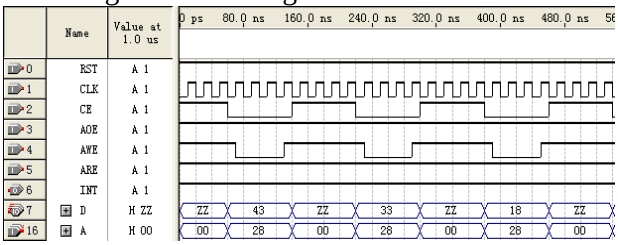

Fig.3. The sequence diagram of writing operation between DSP and FPGA

2.2 Software design of the system

In software design of the system, intensity data and highly repetitive algorithm are processed by FPGA, while low repetitive algorithm is processed by DSP. In this radar target tracking system, initial value of IMM-SDCMKF needs to be calculated only once. Therefore, the initial value of IMM-SDCMKF is calculated by DSP then transmits to FPGA. However, the subsequent each frame data needs to be filtered and consumes a large number of processing time. Therefore, we choose FPGA to complete IMM-SDCMKF. Fig.4 shows the software block diagram of the radar target tracking system. The first three frames data correctly received from radar are calculated for initial values of IMM-SDCMKF. After the system correctly receives the fourth frame data, DSP transmits the calculated

initial values and the measurement value to FPGA for IMM-SDCMKF filtering. An interrupt signal is transmitted to DSP to retrieve the filtered target state values after FPGA processing each frame data.

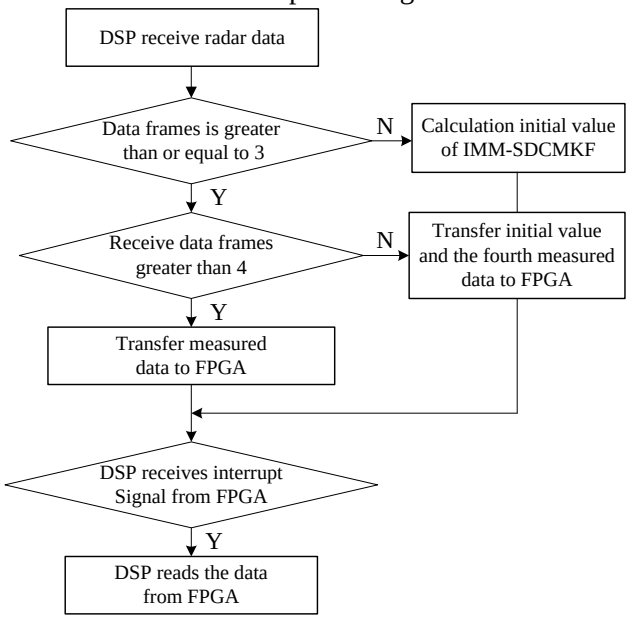

Fig.4. Software design of the system

3 IMM-SDCMKF principle

3.1 Target tracking model

IMM covers several filters, one model probability estimator, one interactive effector and one estimator commingler. Algorithm recursion each time concludes the following four steps.

1) Interaction of state estimation

Suppose there are r models, then transition probability from model i to model j is P_{ij}. Let $\hat{X}_i(k|k)$ as state estimation of filter i at time k, $P_i(k|k)$ as the corresponding covariance matrix and $\mu_i(k)$ as probability of model i at time k, while $i, j = 1,2,...,r$, then inputs of r filters at $k+1$ by interactive computing are as follows:

$$\hat{\hat{X}}_{oi}(k|k) = \sum_{i=1}^{r} X_i(k|k)\mu_{i|j}(k|k) \tag{1}$$

$$P_{oi}(k|k) = \sum_{i=1}^{r}[P_i(k|k) + (\hat{X}_i(k|k) - X_{oi}(k|k)) (\hat{X}_i(k|k) - X_{oi}(k|k))^T]\mu_{i|j}(k|k) \tag{2}$$

where

$$\mu_{i|j}(k|k) = \frac{1}{\bar{C}_i} P_{ij}\mu_i(k) \tag{3}$$

$$\bar{C}_i = \sum_{i=1}^{r} P_{ij}\mu_i(k) \tag{4}$$

2) Model conditional filtering

Filter output is carried out as $\hat{X}_i(k+1|k+1)$ and $P_i(k+1|k+1)$ when taking $\hat{X}_{oi}(k|k)$ and $P_{oi}(k|k)$ as input in i model of $(k+1)$.

3) Updating model probability

$$\mu_i(k+1) = \frac{1}{C}\Lambda_i(k+1)\bar{C}_i \tag{5}$$

where

$$C = \sum_{i=1}^{r}\Lambda_i(k+1)\bar{C}_i \tag{6}$$

$$\Lambda_i(k+1) = \frac{\exp[-\frac{1}{2}(v_i^T(k+1))S_i^{-1}(k+1)v_i(k+1)]}{\sqrt{|2\pi S_i(k+1)|}} \tag{7}$$

where

$$v_i(k+1) = Z(k+1) - H_i(k+1)\hat{X}_i(k+1|k) \tag{8}$$

$$S_i(k+1) = H_i(k+1)P_i(k+1|k)H_i^T(k+1) + R_i(k+1) \tag{9}$$

4) Filter interacted output

$$\hat{\hat{X}}(k+1|k+1) = \sum_{i=1}^{r} X_i(k+1|k+1)\mu_i(k+1) \tag{10}$$

$$P(k+1|k+1) = \sum_{i=1}^{r}[(\hat{\hat{X}}_i(k+1|k+1) - X(k+1|k+1)) (\hat{\hat{X}}_i(k+1|k+1) - X(k+1|k+1))^T]\mu_i(k+1) + \sum_{i=1}^{r} P_i(k+1|k+1)\mu_i(k+1) \tag{11}$$

This paper selects CV model and Singer acceleration model interact[13]. State equation of the system is

$$X(k+1) = \Phi_i(k)X(k) + \Gamma_i(k)W_i(k) \tag{12}$$

Measurement equation is

$$Z(k) = HX(k) + V(k) \tag{13}$$

Where,

$X(k) = [x(k), y(k), z(k), \dot{x}(k), \dot{y}(k), \dot{z}(k), \ddot{x}(k), \ddot{y}(k), \ddot{z}(k)]^T$ serves as state vector of the system, including target's position, velocity and acceleration in X, Y and Z direction, respectively. $\Phi_1(k)$ is system state transition matrix of CV, $\Gamma_1(k)$ is noise gain matrix of CV, $W_1(k)$ is system process noise matrix of CV, $\Phi_2(k)$ is system state transition matrix of Singer, $\Gamma_2(k)$ is noise gain matrix of Singer, $W_2(k)$ is system process noise matrix Singer of Singer, $Z(k)$ is the system measurement vector; H is measure matrix, $V(k)$ is measurement noise vector.

$$\Phi_1(\mathrm{k})=\begin{bmatrix}1&0&0&T&0&0&0&0&0\\0&1&0&0&T&0&0&0&0\\0&0&1&0&0&T&0&0&0\\0&0&0&1&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{bmatrix} \quad (14)$$

$$\Phi_2(\mathrm{k})=\begin{bmatrix}1&0&0&T&0&0&\phi_{17}&0&0\\0&1&0&0&T&0&0&\phi_{28}&0\\0&0&1&0&0&T&0&0&\phi_{39}\\0&0&0&1&0&0&\phi_{47}&0&0\\0&0&0&0&1&0&0&\phi_{58}&0\\0&0&0&0&0&1&0&0&\phi_{69}\\0&0&0&0&0&0&e^{-\alpha_x T}&0&0\\0&0&0&0&0&0&0&e^{-\alpha_y T}&0\\0&0&0&0&0&0&0&0&e^{-\alpha_z T}\end{bmatrix} \quad (15)$$

where

$$\phi_{17}=(\alpha_x T-1+e^{-\alpha_x T})/\alpha_x{}^2,\quad \phi_{28}=(\alpha_y T-1+e^{-\alpha_y T})/\alpha_y{}^2$$

$$\phi_{39}=(\alpha_z T-1+e^{-\alpha_z T})/\alpha_z{}^2,\quad \phi_{47}=(1-e^{-\alpha_x T})/\alpha_x,$$

$$\phi_{58}=(1-e^{-\alpha_y T})/\alpha_y,\qquad \phi_{69}=(1-e^{-\alpha_z T})/\alpha_z$$

$$\Gamma_1(k)=\begin{bmatrix}\Gamma_{11}\\\Gamma_{12}\\\Gamma_{13}\end{bmatrix} \quad (16)$$

$$\Gamma_2(k)=\begin{bmatrix}\Gamma_{21}\\\Gamma_{22}\\\Gamma_{23}\end{bmatrix} \quad (17)$$

where

$$\Gamma_{11}=diag\left(\begin{bmatrix}T^2/2\\T^2/2\\T^2/2\end{bmatrix}\right)\qquad \Gamma_{12}=diag\left(\begin{bmatrix}T\\T\\T\end{bmatrix}\right)$$

$$\Gamma_{13}=diag\left(\begin{bmatrix}0\\0\\0\end{bmatrix}\right)$$

$$\Gamma_{21}=diag\left(\begin{bmatrix}\gamma_x\left[1-\alpha_x T-e^{-\alpha_x T}+(T^2\alpha_x{}^2/2)\right]/\alpha_x{}^3\\\gamma_y\left[1-\alpha_y T-e^{-\alpha_y T}+(T^2\alpha_y{}^2/2)\right]/\alpha_y{}^3\\\gamma_z\left[1-\alpha_z T-e^{-\alpha_z T}+(T^2\alpha_z{}^2/2)\right]/\alpha_z{}^3\end{bmatrix}\right)$$

$$\Gamma_{22}=diag\left(\begin{bmatrix}\gamma_x(\alpha_x T+e^{-\alpha_x T}-1)/\alpha_x{}^2\\\gamma_y(\alpha T+e^{-\alpha_y T}-1)/\alpha_y{}^2\\\gamma_z(\alpha T+e^{-\alpha_z T}-1)/\alpha_z{}^2\end{bmatrix}\right)$$

$$\Gamma_{23}=diag\left(\begin{bmatrix}\gamma_x(1-e^{-\alpha_x T})/\alpha_x\\\gamma_y(1-e^{-\alpha_y T})/\alpha_y\\\gamma_z(1-e^{-\alpha_z T})/\alpha_z\end{bmatrix}\right)$$

where, $\gamma_x=\gamma_y=\gamma_z=\gamma$, $\alpha_x=\alpha_y=\alpha_z=\alpha$ describes the first-order forming filter parameter of the attacking target's acceleration in the Cartesian coordinate. T is system measurement period.

$$\mathrm{H}=\begin{bmatrix}1&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\end{bmatrix} \quad (18)$$

3.2 Radar data second-order debiased converted measurement

In the spherical coordinate, the true measurements of radar are azimuth angle β_m, pitch angle θ_m and radial distance r_m, with noise variance as $\sigma_\beta^2, \sigma_\theta^2, \sigma_r^2$, respectively. The average true deviation μ_k and average true covariance R_k of converted measurement are respectively as:

$$\mu_k=[\mu_k^x,\mu_k^y,\mu_k^z]^T \quad (19)$$

$$R_k=\begin{bmatrix}R_k^{xx}&R_k^{xy}&R_k^{xz}\\R_k^{yx}&R_k^{yy}&R_k^{yz}\\R_k^{zx}&R_k^{zy}&R_k^{zz}\end{bmatrix} \quad (20)$$

where

$$\mu_k^x=-\frac{1}{2}(\sigma_\theta^2+\sigma_\beta^2)(1-\frac{\sigma_\theta^2}{2}-\frac{\sigma_\beta^2}{2})r_m\cos\theta_m\cos\beta_m$$

$$\mu_k^y=-\frac{1}{2}(\sigma_\theta^2+\sigma_\beta^2)(1-\frac{\sigma_\theta^2}{2}-\frac{\sigma_\beta^2}{2})r_m\cos\theta_m\sin\beta_m$$

$$\mu_k^z=-\frac{1}{2}\sigma_\theta^2(1-\frac{\sigma_\theta^2}{2})r_m\sin\theta_m$$

$$R_k^{xx}=\left(\sigma_\theta^2\sigma_r^2+\sigma_\theta^2 r_m^2\right)\frac{\tilde{\alpha}_2}{4}+\left(\frac{\sigma_\theta^4}{2}r_m^2+\frac{\sigma_\beta^4}{2}r_m^2+\sigma_r^2\right)\frac{\tilde{\alpha}_4}{4}$$
$$+\left(\sigma_\beta^2\sigma_r^2+\sigma_\beta^2 r_m^2\right)\frac{\tilde{\alpha}_3}{4}+\sigma_\theta^2\sigma_\beta^2\sigma_r^2\tilde{\delta}_1+\sigma_\theta^2\sigma_r^2\tilde{\delta}_2$$
$$+\sigma_\beta^2\sigma_r^2\tilde{\delta}_3+\left(\frac{\sigma_\theta^4}{2}+\frac{\sigma_\beta^4}{2}\right)\sigma_r^2\tilde{\delta}_4+\sigma_\theta^2\sigma_\beta^2 r_m^2\frac{\tilde{\alpha}_1}{4}$$

$$R_k^{yy}=\left(\sigma_\theta^2\sigma_r^2+\sigma_\theta^2 r_m^2\right)\frac{\tilde{\alpha}_1}{4}+\sigma_\theta^2\sigma_\beta^2 r_m^2\frac{\tilde{\alpha}_2}{4}+\left(\sigma_\beta^2\sigma_r^2+\sigma_\beta^2 r_m^2\right)\frac{\tilde{\alpha}_4}{4}$$
$$+\left(\frac{\sigma_\theta^4}{2}r_m^2+\frac{\sigma_\beta^4}{2}r_m^2+\sigma_r^2\right)\frac{\tilde{\alpha}_3}{4}+\sigma_\theta^2\sigma_\beta^2\sigma_r^2\tilde{\delta}_2+\sigma_\theta^2\sigma_r^2\tilde{\delta}_1$$
$$+\sigma_\beta^2\sigma_r^2\tilde{\delta}_4+\left(\frac{\sigma_\theta^4}{2}+\frac{\sigma_\beta^4}{2}\right)\sigma_r^2\tilde{\delta}_3$$

$$R_k^{zz}=\frac{1}{2}\left(4\sigma_\theta^2\sigma_r^2-\sigma_r^2-\frac{5}{2}\sigma_\theta^4\sigma_r^2\right)\cos 2\theta_m+\frac{1}{2}\left((\sigma_\theta^6+\sigma_\theta^2)-\frac{5}{2}\sigma_\theta^4\right)r_m^2\cos 2\theta_m+\frac{1}{2}\left(\frac{1}{2}\sigma_\theta^4+\sigma_\theta^2\right)r_m^2+\frac{1}{2}\left(\sigma_r^2+2\sigma_\theta^2\sigma_r^2+\frac{1}{2}\sigma_\theta^4\sigma_r^2\right)$$

$$R_k^{xy}=\left((\frac{\sigma_\theta^4}{2}+\frac{\sigma_\beta^4}{2}-\sigma_\beta^2)r_m^2+\sigma_r^2-\sigma_\beta^2\sigma_r^2\right)\frac{\tilde{\gamma}_1}{4}+\left((\sigma_\theta^2-\sigma_\theta^2\sigma_\beta^2)r_m^2+\sigma_\theta^2\sigma_r^2\right)\frac{\tilde{\gamma}_3}{4}+\left(\frac{\sigma_\theta^4}{2}+\frac{\sigma_\beta^4}{2}-\sigma_\beta^2\right)\sigma_r^2\frac{\tilde{\gamma}_2}{4}+\left(\sigma_\theta^2-\sigma_\theta^2\sigma_\beta^2\right)\sigma_r^2\frac{\tilde{\gamma}_4}{4}-\frac{1}{4}\left(\left(\frac{\sigma_\theta^2}{\sqrt{2}}+\frac{\sigma_\beta^2}{\sqrt{2}}\right)^2-2\sigma_\beta^2+1\right)\sigma_r^2-\frac{1}{4}\left(\left(\frac{\sigma_\theta^2}{\sqrt{2}}-\frac{\sigma_\beta^2}{\sqrt{2}}\right)^2-\sigma_\beta^2+\sigma_\theta^2\right)r_m^2$$

$$R_k^{xz}=\left[\left((\frac{\sigma_\theta^4}{2}-\sigma_\theta^2)r_m^2+\sigma_r^2-\sigma_\theta^2\sigma_r^2\right)\left(1-2\sigma_\theta^2-\sigma_\beta^2\right)+\left(\frac{\sigma_\theta^4}{2}-\sigma_\theta^2\right)\sigma_r^2\right]*\sin\theta_m\cos\theta_m\cos\beta_m$$

$$R_k^{yz}=\left[\left((\frac{\sigma_\theta^4}{2}-\sigma_\theta^2)r_m^2+\sigma_r^2-\sigma_\theta^2\sigma_r^2\right)\left(1-2\sigma_\theta^2-\sigma_\beta^2\right)+\left(\frac{\sigma_\theta^4}{2}-\sigma_\theta^2\right)\sigma_r^2\right]*\sin\theta_m\cos\theta_m\sin\beta_m$$

$$\tilde{\alpha}_1=1-(1-2\sigma_\beta^2)\cos 2\beta_m-(1-2\sigma_\theta^2)\cos 2\theta_m+(1-2\sigma_\theta^2-2\sigma_\beta^2)\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\alpha}_2=1+(1-2\sigma_\beta^2)\cos 2\beta_m-(1-2\sigma_\theta^2)\cos 2\theta_m+(1-2\sigma_\theta^2-2\sigma_\beta^2)\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\alpha}_3=1+(1-2\sigma_\beta^2)\cos 2\beta_m+(1-2\sigma_\theta^2)\cos 2\theta_m-(1-2\sigma_\theta^2-2\sigma_\beta^2)\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\alpha}_4=1-(1-2\sigma_\beta^2)\cos 2\beta_m+(1-2\sigma_\theta^2)\cos 2\theta_m+(1-2\sigma_\theta^2-2\sigma_\beta^2)\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\delta}_1=1-\cos 2\beta_m-\cos 2\theta_m+\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\delta}_2=1+\cos 2\beta_m-\cos 2\theta_m-\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\delta}_3=1-\cos 2\beta_m+\cos 2\theta_m-\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\delta}_4=1+\cos 2\beta_m+\cos 2\theta_m+\cos 2\theta_m\cos 2\beta_m$$

$$\tilde{\gamma}_1=1+(1-2\sigma_\beta^2)\sin 2\beta_m+(1-2\sigma_\theta^2-2\sigma_\beta^2)\cos 2\theta_m\sin 2\beta_m$$

$$\tilde{\gamma}_2=1+\sin 2\beta_m+\cos 2\theta_m\sin 2\beta_m$$

$$\tilde{\gamma}_3=1+(1-2\sigma_\beta^2)\sin 2\beta_m-(1-2\sigma_\theta^2-2\sigma_\beta^2)\cos 2\theta_m\sin 2\beta_m$$

$$\tilde{\gamma}_4=1+\sin 2\beta_m-\cos 2\theta_m\sin 2\beta_m$$

When measurement in the spherical coordinate is converted to be in Cartesian coordinate, the measurement is modified as

$$Z_c=Z-\mu_k=\begin{bmatrix}r_m\cos\theta_m\cos\beta_m\\ r_m\cos\theta_m\sin\beta_m\\ r_m\sin\theta_m\end{bmatrix}-\mu_k \tag{21}$$

3.3 IMM-SDCMKF algorithm

The calculation step of IMM-SDCMKF is given as:

1) Calculating the initial value $\hat{X}_i(0|0)$ and initial covariance matrix $P_i(0|0)$, and initial model probability $\mu_i(0)$

2) Calculating filter input:

$$\hat{\hat{X}}_{oi}(k|k)=\sum_{i=1}^{r}X_i(k|k)\mu_{i|j}(k|k) \tag{22}$$

$$P_{oi}(k|k)=\sum_{i=1}^{r}[P_i(k|k)+(\hat{X}_i(k|k)-X_{oi}(k|k))(\hat{X}_i(k|k)-X_{oi}(k|k))^T]\mu_{i|j}(k|k) \tag{23}$$

where

$$\mu_{i|j}(k|k)=\frac{1}{\bar{C}_i}P_{ij}\mu_i(k) \tag{24}$$

$$\bar{C}_i=\sum_{i=1}^{r}P_{ij}\mu_i(k) \tag{25}$$

3) Predicting state vector

$$\hat{\hat{X}}_i(k+1|k)=F_i(k)X_{oi}(k|k) \tag{26}$$

4) Calculating covariance matrix of the predicted states

$$P_i(k+1|k)=F_i(k)P_{oi}(k|k)F_i(k)^T+G_i(k)Q_i(k)G_i(k)^T \tag{27}$$

5) Calculating the mean deviation $u(k+1)$ and covariance $R(k+1)$ of the converted measurement according to equation (19) and (20)

6) Calculating gain matrix

$$K_i(\mathrm{k+1})=P_i(k+1|k)H^{\mathrm{T}}(HP_i(k+1|k)H^{\mathrm{T}}+R(k+1))^{-1} \tag{28}$$

7) Updating state vector

$$\hat{X}_i(k+1|k+1)=X_i(k+1|k)+K_i(k+1)(Z(k+1)-\mu(k+1)-HX_i(k+1|k)) \tag{29}$$

8) Updating covariance matrix

$$P_i(k+1|k+1)=(I-K_i(k+1)H)P_i(k+1|k) \tag{30}$$

9) Updating model probability

$$\mu_i(k+1)=\frac{1}{C}\Lambda_i(k+1)\bar{C}_i \tag{31}$$

where

$$C=\sum_{i=1}^{r}\Lambda_i(k+1)\bar{C}_i \tag{32}$$

where

$$\Lambda_i(k+1)=\frac{\exp[-\frac{1}{2}(\nu_i^{\ T}(k+1))S_i^{-1}(k+1)\nu_i(k+1)]}{\sqrt{|2\pi S_i(k+1)|}} \tag{33}$$

where

$$\nu_i(k+1)=Z(k+1)-H_i(k+1)\hat{X}_i(k+1|k) \tag{34}$$

$$S_i(k+1)=H_i(k+1)P_i(k+1|k)H_i^{\ T}(k+1)+R(k+1) \tag{35}$$

10) Filter interacted output

$$\hat{\hat{X}}(k+1|k+1)=\sum_{i=1}^{r}X_i(k+1|k+1)\mu_i(k+1) \tag{36}$$

$$P(k+1|k+1)=\sum_{i=1}^{r}[(\hat{\hat{X}}_i(k+1|k+1)-X(k+1|k+1))(\hat{\hat{X}}_i(k+1|k+1)-X(k+1|k+1))^T]\mu_i(k+1)+\sum_{i=1}^{r}P_i(k+1|k+1)\mu_i(k+1) \quad (37)$$

11) Repeating step 2) to 10) for recursive computation

4 Design of IMM-SDCMKF based on FPGA

4.1 Module structure design

This adopts the module structure design idea to design IMM-SDCMKF algorithm. The bottom module selects very high speed integrated circuits hardware description language (VHDL) as input, while the top module selects schematic diagram as input. The module structure diagram of IMM-SDCMKF based on FPGA is shown in Fig5. Three FIFO modules are used to temporarily store state update value, filter error covariance value and model probability update value, ready for circular filtering of the next frame data. Multiplexer (MUX) is designed as options either as the initial value or circular filtering value, and the three multiplexers together are designed respectively for state value, covariance value and model probability value.

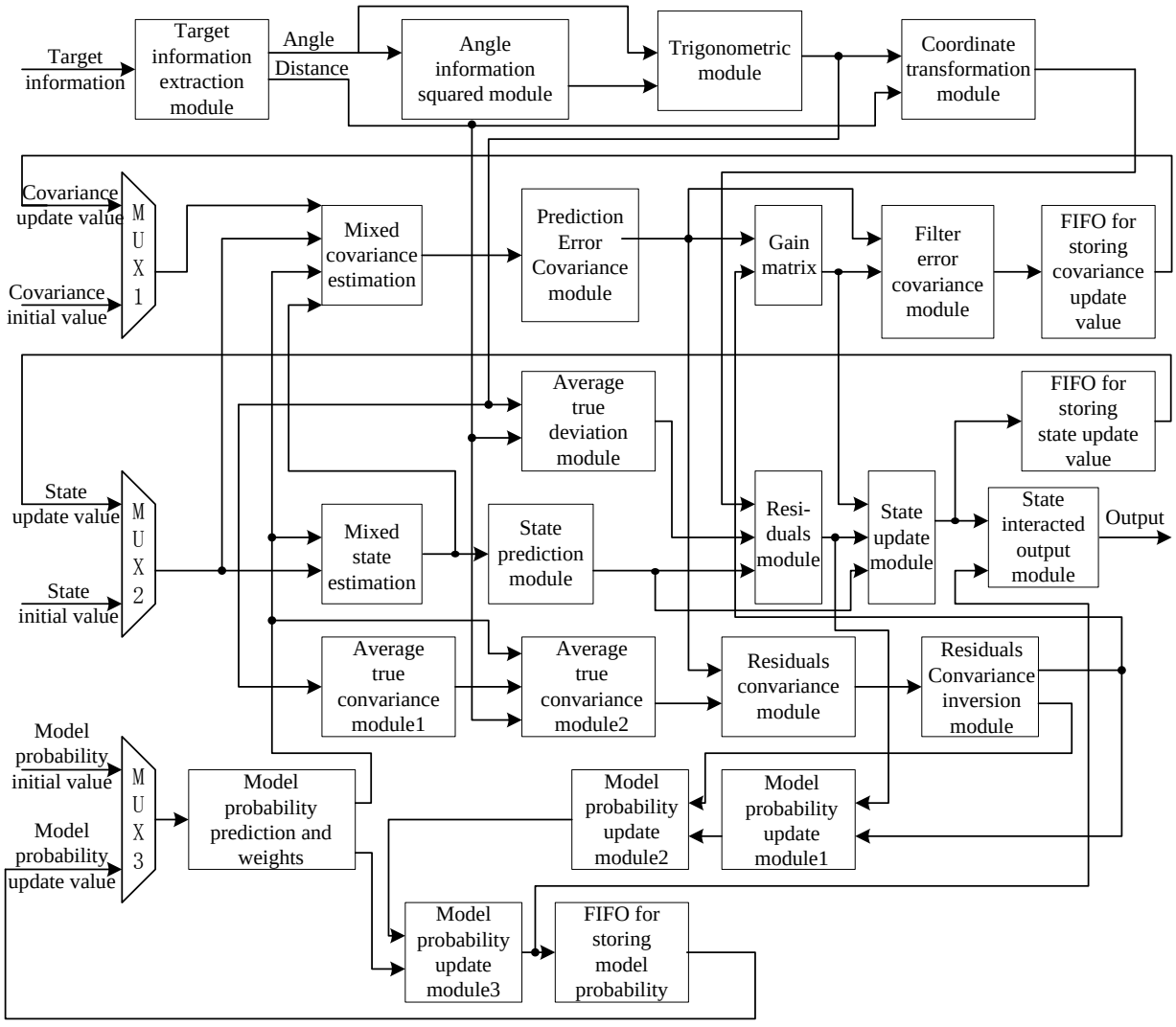

Fig.5. Module structure design of IMM-SDCMKF based on FPGA

4.2 Pre-processing algorithm

To realize IMM-SDCMKF algorithm by FPGA, this algorithm needs to be pre-processed from matrix form to vector form [14]. This design scheme can realize codes easily, simplify scalar calculation, avoid the complicated multiplication, addition calculation of sparse matrix with a large number of zero cells, and save FPGA resource efficiently [15]. Scientific Workspace software can show clearly the variable relationship between input matrix array and

output matrix array. As shown in Fig 6, Scientific Workspace has realized state prediction value $X_p = \Phi X_f$ decomposed from matrix form to scalar form.

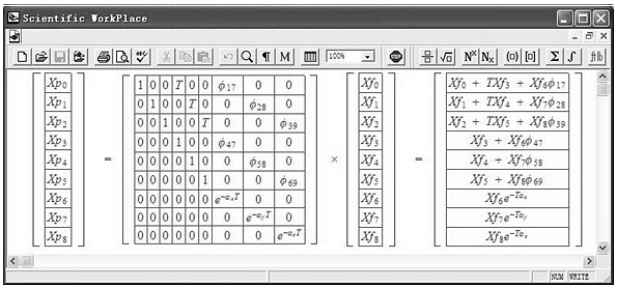

Fig 6. Scientific Workspace

4.3 Design of computing modules of IMM-SDCMKF

When FPGA processes floating point calculation, it occupies more computing resource. Therefore, time division multiplexing technology is applied for the fundamental calculation module in this paper [16,17]. Take the state prediction module as an example. State predicted value can be modified as

$$Xp = [Xp_0, Xp_1, Xp_2, Xp_3, Xp_4, Xp_5, Xp_6, Xp_7 Xp_8]^{\mathrm{T}}$$

where

$$Xp_0 = Xf_0 + T * Xf_3 + \phi_{17} * Xf_6$$
$$Xp_1 = Xf_1 + T * Xf_4 + \phi_{28} * Xf_7$$
$$Xp_2 = Xf_2 + T * Xf_5 + \phi_{39} * Xf_8$$
$$Xp_3 = Xf_3 + \phi_{47} * Xf_8$$
$$Xp_4 = Xf_4 + \phi_{58} * Xf_7$$
$$Xp_5 = Xf_5 + \phi_{69} * Xf_8$$
$$Xp_6 = e^{-\alpha T} * Xf_6$$
$$Xp_7 = e^{-\alpha T} * Xf_7$$
$$Xp_8 = e^{-\alpha T} * Xf_8$$

Fig.7 is the structure diagram of State prediction module. State prediction module occupies 2 floating point addition operation units and 2 floating point multiplication operation units. In this paper, the period parameters of floating point addition and multiplication units are set respectively as 7 and 5 clock cycles in library parameter module (LPM). Because the data number of input port cannot always make up 2n, the data which don't participate in computing should be set for 5 clock delays at the processing data in the first stage floating point multiplication processing to guarantee data synchronization. Similarly, the previous stage results which don't participate in computing should be set for 7 clock delays in the second stage floating point addition. When input port receives 9 state values one clock cycle by another, the data distribution module sends its corresponding multiplicand and multiplier to the right register for processing at each clock. After 19 (5+7+7) clock cycles, state prediction module output processed data at per clock cycle. The design idea of input and output port of other operation module is similar, here not to introduce again.

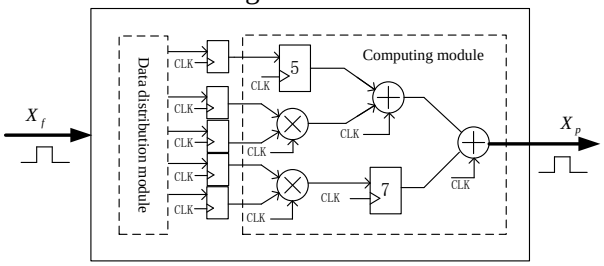

Fig.7. Structure diagram of state prediction module

4.4 Top level schematic diagram of IMM-SDCMKF

The top level of this design uses schematic diagram as input. In the top level schematic diagram, we utilize directly the packaged calculating modules in the above section. All related data bus, enable signal, clock signal and reset signal are connected.

As mentioned above, some modules of IMM-SDCMKF are connected, while others are independent absolutely. In order to make the modules run synchronously, we design handshaking signal among modules to enable the next module to receive and calculate. All modules can be orderly operated in corresponding time sequence .During the process of the design, not only the data input and output ports among modules are needed, but the handshaking signal, clock signal and reset signal are needed.

4.4.1 State prediction module

Initial state prediction value is transmitted from DSP to FPGA, while values hereafter will be read in FIFO module within FPGA. Fig.8 is state prediction module. State prediction module's input port is the previous time state update value, clock signal, reset signal and input enable signal. State prediction module's output port is the state prediction value and output enable signal for next module.

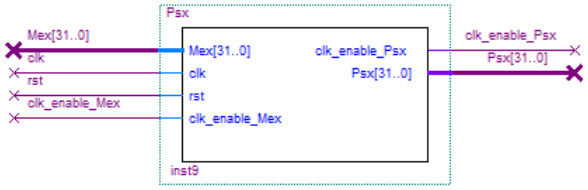

Fig.8. State prediction module

4.4.2 FIFO memory module

In this design, FIFO is used to temporarily store state update value, filter error covariance value and model probability update value, ready for circular filtering for the next frame data.

Fig.9 is synchronous FIFO memory module, where lpm_fifo1 module is used to temporarily store state update value, lpm_fifo2 module is used to temporarily store filter error covariance value, lpm_fifo3 module is used to temporarily store model probability update value and can easily called for using later. The three clock signals of FIFO module are effective, thus the access of synchronous FIFO can be realized by controlling the writing enable signal. When the state update module begins to output the results, the writing enable signal of lpm_fifo1 is valid. When the state update module completes exporting the results, the writing enable signal of lpm_fifo1 is invalid. Therefore lpm_fifo1 module saves the state update value of the current clock cycle. When receiving the radar data at next clock cycle, the writing enable signal of lpm_fifo1 is valid, the state estimate value of lpm_fifo1 is read and the state prediction value is calculated. The storage for lpm_fifo2 and lpm_fifo3 are similarly with lpm_fifo1.

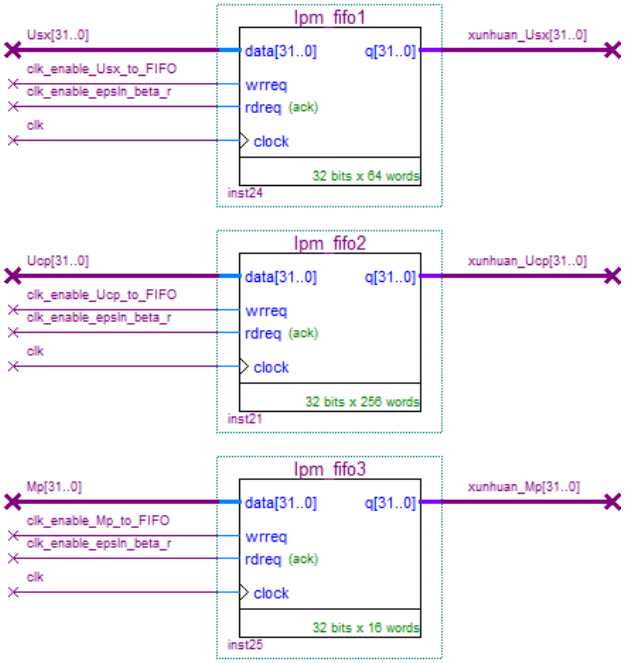

Fig.9. Synchronous FIFO memory module

5 Experimental results and analysis

Fig.10 shows hardware circuit board of the system, this system is composed of DSP, FPGA, SDRAM, FLASH, EEPROM and other components.

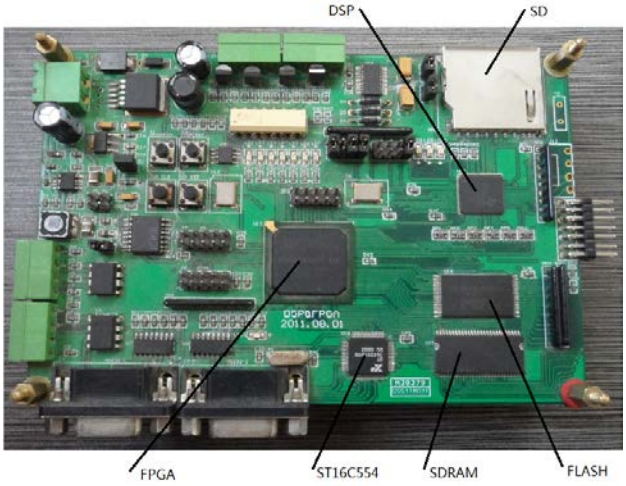

Fig.10. Hardware circuit board

5.1 Correctness of the analysis

The target trajectory is shown in Fig.11. The parameters of target are given as follows. The initial conditions of the target is (9000m, 5000m, 4000m) for position and (-250m/s, -250m/s, -100m/s) for velocity. The segments are defined as follows. 1st segment, t=(0-5)s, constant velocity flight with acceleration 0. 2nd segment, t=(5-30)s, S type acceleration maneuver. 3rd segment, t=(30-35)s constant velocity flight with acceleration 0. The sampling rate is t=10ms. Measurement noise covariance of radial distance, azimuth angle and pitch angle are $\sigma_r^2 = 1$, $\sigma_\beta^2 = \pi^2/32400$, $\sigma_\theta^2 = \pi^2/72900$, especively. The IMM-SDCMKF algorithm is realized respectively in FPGA and Matlab platform using the same measurements. Fig.12 shows position comparison in X direction, Fig.13 and Fig.14 show position comparison in Y direction and Z direction. It is easily seen from the three figures that the IMM-SDCMKF is capable of denoising and smoothing for target position. Fig.12 Fig.13 and Fig.14 show the results of FPGA are consistent with the simulated results by Matlab. The high precision proves the correctness of this design scheme.

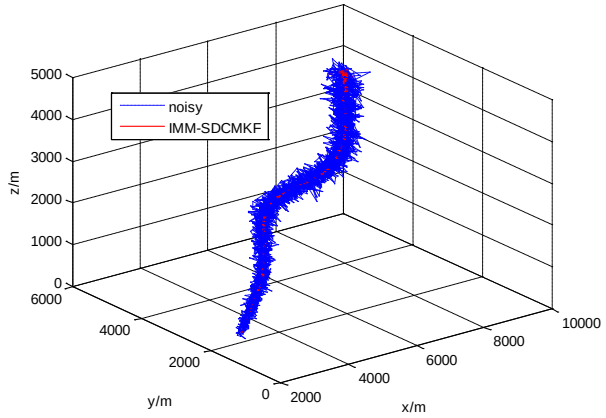

Fig.11. The target trajector

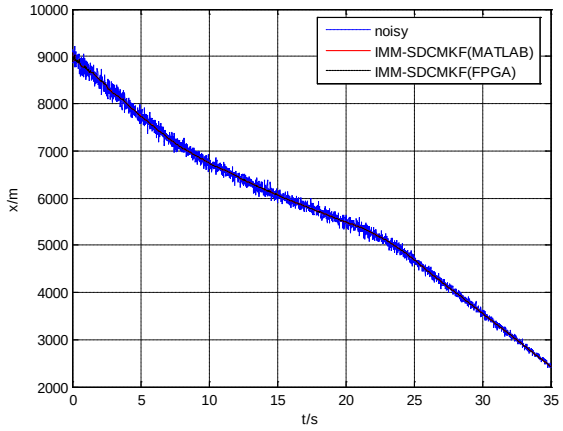

Fig.12. Position comparisons in X direction

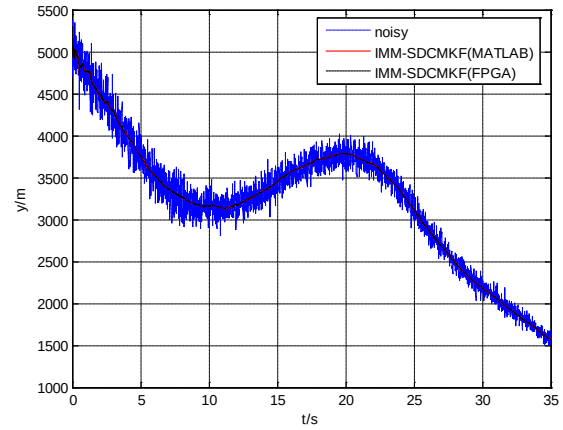

Fig.13. Position comparisons in Y direction

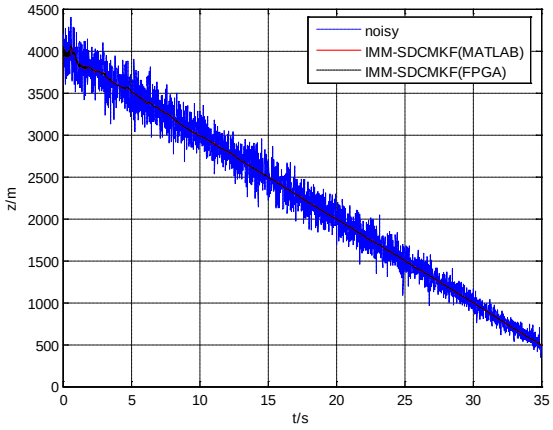

Fig.14. Position comparisons in Z direction

5.2 Real-time of the analysis

Table 1 shows time comparison between DSP and FPGA by finishing one-time IMM-SDCMKF separately. Experiment results prove the designed IMM-SDCMKF algorithm based on single DSP spends 2.981ms to complete one filter process. While the designed IMM-SDCMKF algorithm based on FPGA coprocessor spends only 0.037ms to complete one filter process. This paper utilizes FPGA as a coprocessor to realize IMM-SDCMKF algorithm, which can satisfy the requirement of real time of the system

Table 1 Computing time comparisons

Type of processor	Model	Clock frequency	Computing time
DSP	TMS320 VC5509A	200MHz	2.981ms
FPGA	EP3C120 F484C8N	25MHz	0.037ms

6 Conclusion

In the radar maneuvering target tracking system, the tracking precision and real time are highly required. IMM-SDCMKF algorithm includes a great deal of matrix arithmetic, such as matrix addition, matrix subtraction, matrix multiplication and inverse. The computational time for calculating IMM-SDCMKF algorithm in software is too long to meet the real time of realizes hard-to-reach real time performance of maneuvering target tracking. In this paper, the FPGA is used as a floating point coprocessor of fixed point DSP. This software and hardware reasonable design scheme can solve the concurrency and speed problems and guarantee the tracking

precision. Therefore, it is an effective approach to complete target tracking algorithm. The design based on FPGA has degree flexibility for programming, updates codes at any time, and largely reduces the research cost. This research results have been successfully applied to a certain type of radar maneuvering target tracking system.

References:

[1] ZHOU Hong-bo, GENG Bo-ying. Converted Measurements Kalman Filter Algorithm for Target Tracking[J] . *Journal of System Simulation*, Vol.20, No.3, 2008,pp. 682-688.

[2] HE Ming-ke, WANG Zheng-ming, ZHU Ju-bo. Debiased Converted Measurement KF for Radar Target Tracking[J]. *Journal of National University of defense technology*, Vol.24, No.5, 2002, pp. 57-60.

[3] CHEN Hao, TAN Jiu-bin. Target Tracking Based on Second-order Converted Measurement Kalman Filter[J]. *Opto-Electronic Engineering*, Vol.35, No.4, 2008, pp. 6-11.

[4] ZHOU Ning-ning, CHEN Yan-li, LI Ai-qun. Design and implementation of floating point calculator based on FPGA technology[J]. *Computer Engineer and Design*, Vol.26, No.6, 2005, pp.1578-1581.

[5] ZHONG Sheng, HOU Chao-huan, YANG Chang-an. Optimized design of matrix multiplier based on FPGA[J]. *Electronic Measurement Technology*, Vol.31, No.2, 2008, pp 95-102.

[6] CHEN Gang, GUO Li. The FPGA Implementation of Kalman Filter[C]. *Proceedings of the 5th WSEAS Int. Conf. on Signal Processing*, 2005, pp.61-65.

[7] C.R. Lee, Z. Salcic. High-performance FPGA-based Implementation of Kalman Filter[J]. *Mircroprocessors and Microsystems*, Vol.21, No.4, 1997, pp. 257-265.

[8] Ruchi Pasricha, Sanjay Sharma. An FPGA-Based Design of Fixed-Point Kalman Filter[J]. *DSP Journal*, Vol.9, No.1, pp.1-9.

[9] L.Idkhajine, E.Monmasson, A.Maalouf. FPGA-based Sensorless Controller for Synchronous Machine using an Extended Kalman Filter[C]. *13th European Conference on Power Electronics and Application*, 2009, pp.1-10.

[10] L.Idkhajine, E.Monmasson. Optimized FPGA-based Extended Kalman Filter Application to an AC Drive Sensorless Speed Controller[C]. *2010 International Symposium on Power Electronics Electrical Drives Automation and motion*, 2010, pp. 1012-1017.

[11] Zoran Salcic, Jiaying Cao, Sing Kiong Nguang. A Floating-Point FPGA-Based Self-Tuning Regulator[J]. *IEEE Transactions on Industrial Electronics*, Vol. 53, No. 2, 2006, pp.693-704.

[12] Pan-long Wu, Bao-bao Wang, Cun-hui Ji. Design and realization of short range defence radar target tracking system based on DSP/FPGA[J]. *WSEAS Transactions on System*, Vol. 10, No. 11, 2011, pp.376-386.

[13] ROBERT A. SINGER. Estimating optimal tracking filter performance for manned maneuvering targets[J]. *IEEE Transactions on Aerospace and Electronic Systems*, Vol.6, No. 4, 1970, pp. 473-483.

[14] Dr S M Shalinie，Associate Menber. Design and Analysis of Customized Embedded Kalman Filter[J]. IE(I) Journal-CP, 2007, 88(5):39-42.

[15] Neri F. Cooperative Evolutive Concept Learning: An Empirical Study[J]. *WSEAS Transactions on Information Science and Applications*, Vol. 2, No. 5, 2005, pp.559-563.

[16] Abbas Bigdeli, Morteza Biglari-Abhari, Zoran Salcic, and Yat Tin Lai. A New Pipelined Systolic Array-Based Architecture for Matrix Inversion in FPGAs with Kalman Filter Case Study[J].*EURASIP Journal on Applied Signal Processing*, Vol. 2006, Article ID 89186, 2006, pp. 1-12.

[17] Ventzas, D. A 4-bit Quantized Skip Algorithm Software Correlator for Microcomputer Systems[C]. *IASTED Symposium on Measurement, Signal Processing and Control*, MECO'86, Taormina, Italy, IASTED.