Design and Realization of Short Range Defence Radar Target Tracking System Based on DSP/FPGA

Pan-long Wu, Bao-bao Wang and Cun-hui Ji
Department of Automation
Nanjing University of Science and Technology
No. 200, Xiao Ling Wei Street, Nanjing 210094
PEOPLE'S REPUBLIC OF CHINA
plwu@163.com, wangbaobao_zdh@126.com, jicunhui198771@163.com

Abstract: - This paper presents the hardware design of a real time tracking system based on DSP/FPGA regarding the research and development of a certain type of short range defence radar. In this tracking system, the FPGA is used as a floating point co-processor of the fixed point DSP, and the large amount of calculation of the debiased converted measurement Kalman Filter (DCMKF) 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 requirement of the high precision and real time of the tracking system.

Key-Words: Target tracking, DCMKF, Co-processor, FPGA, DSP, Floating point arithmetic, Time division multiplexing

1 Introduction

In the system of radar target tacking, 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 polar coordinates. Therefore, the radar target tracking becomes a kind of non-liner estimation problem [1]. One solution to this problem is the extended Kalman filter (EKF) but would results in filter distortion. the other solution is debiased converted measurement Kalman filter (DCMKF) [2].

In the traditional software system design, DCMKF is usually realized by digital signal processor (DSP), which would be restricted by the serial instruction stream due to the complex computations of DCMKF and unable to meet the high-speed real-time signal processing needs. However, using the hardware parallel architecture feature of field programmable gate array (FPGA) to realize floating point of DCMKF can resolve the problem of high precision and real time.

Nowadays, how to make use of FPGA to realize Kalman filter (KF) and the related extended algorithm is always a hotspot and difficulty in engineering field. Many scholars have done a great deal of research in this field. In [3], the fundamental arithmetics of floating point numbers based on FPGA are presented. In [4], the algorithms of matrix operation based on FPGA are optimized. In [5,6], the simulation of KF based on FPGA is realized. In [7], a fixed point KF based on FPGA is designed. In [8,9], the optimized FPGA-based EKF is designed and applied to control synchronous motors. In [10], a floating point FPGA-based self-tuning regulator is proposed. This paper utilizes FPGA as a floating point co-processor of the fixed point DSP to realize DCMKF algorithm, which can satisfy the requirement of real time and high precision of the short range defence radar target tracking, 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 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 co-processor of fixed point DSP. DSP receives radar measurements values then transmits to FPGA. FPGA informs DSP to retrieve the filtered system state values when one frame data is processed.

2.1 Hardware design of the system

The structure diagram of system hardware design is shown in Fig.1. This system is composed of DSP

subsystem and FPGA subsystem. DSP subsystem consists of DSP, complex programmable logic device (CPLD), FLASH, synchronous dynamic random access memory (SDRAM), secure digital (SD) card and some peripheral interface circuits, wherein CPLD is used to as the logical interface between the universal asynchronous receiver transmitter (UART), FPGA and DSP. FLASH is used to store program. SDRAM is used to buffer pre-filtered and post-filtered data. SD card is used to store both filtered and unfiltered data. FPGA subsystem consists of FPGA and electrically erasable programmable read-only memory (EEPROM). FPGA is used to realize the complicated DCMKF algorithm, EEPROM is used to store FPGA program.

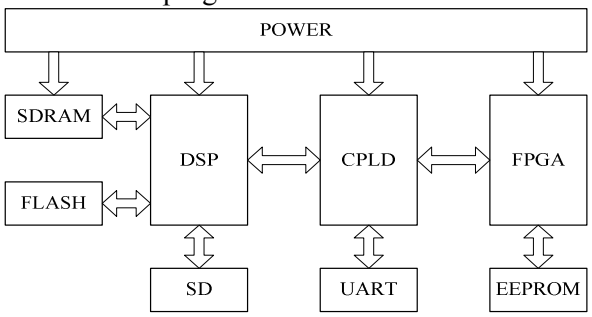

Fig.1. The structure diagram of system hardware design.

Fig.2 shows the hardware connection diagram between DSP, CPLD and FPGA. External memory interface (EMIF) connects DSP and CPLD. 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, $\overline{INT[4:0]}$ is interrupt enable signal, D[7:0] is data bus signal and A[7:0] is address bus signal. The connection between CPLD and FPGA is simple. Wherein, IO(WCLK) is writing clock signal, IO(RCLK) is reading clock signal, IO(D[7:0]) is data bus signal ,IO($\overline{INT}$) is interrupt enable signal.

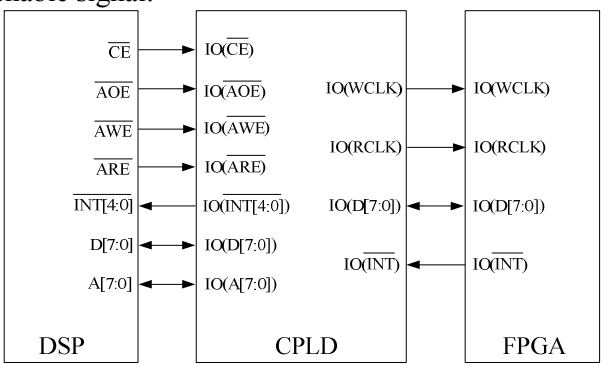

Fig.2. The hardware connection diagram between DSP, CPLD and FPGA.

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

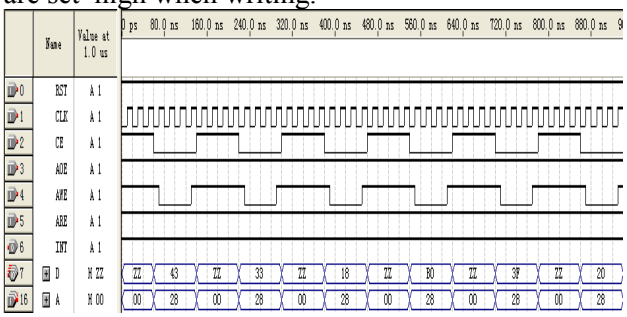

Fig.3. The sequence diagram of writing operation between DSP and CPLD.

Fig.4 shows the sequence diagram of writing operation between CPLD and FPGA. Data is sampled at the rising edge of IO(WCLK). IO($\overline{INT}$) outputs low level pulse enable signal after FPGA processing one frame data. The enable signal is transmitted to DSP through CPLD, and triggers DSP interrupt to read filtered data from FPGA.

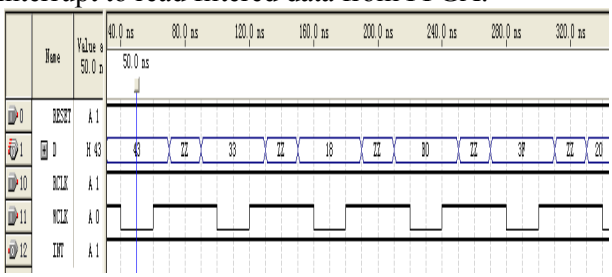

Fig.4. The sequence diagram of writing operation between CPLD and FPGA.

2.2 Software design of the system

In software design of the system, intensive 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 DCMKF needs to be calculated only once. Therefore, the initial value of DCMKF 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 DCMKF algorithm. Fig.5 shows the software block diagram of the radar target tracking system. The first three frames data of correctly received from radar are calculated for initial values of DCMKF. After the system receives correctly the fourth frame data, DSP transmits the calculated initial values and the measurement value to FPGA for DCMKF filtering. An interrupt signal is transmitted to DSP to retrieve the filtered target state values after FPGA processing each frame data.

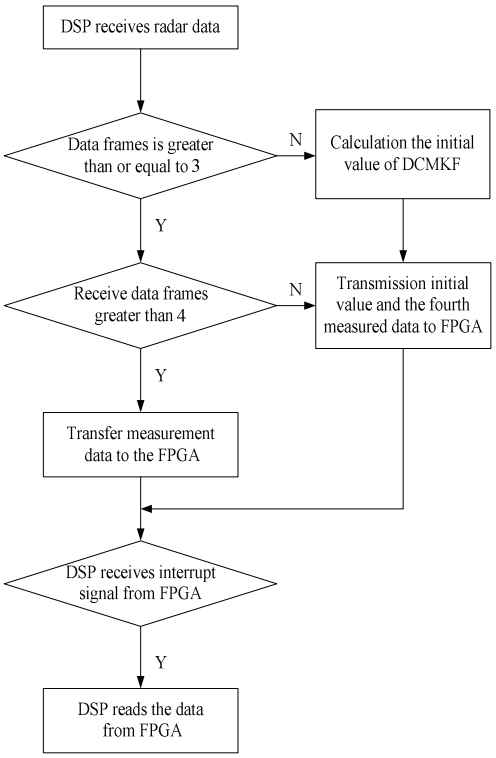

Fig.5 Software design of the system

3 DCMKF principle

3.1 Target tracking model

This paper selects Singer acceleration model as target dynamic model [11]. State equation of the system is

$$X_{k+1}=\Phi X_k+\Gamma_k W_k \tag{1}$$

Measurement equation is

$$Z_k=H_k X_k+V_k \tag{2}$$

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. Φ is system state transition matrix, Γ_k is noise gain matrix, W_k is system process noise matrix, Z_k is the system measurement vector, H_k is measurement matrix, V_k is measurement noise vector.

$$\Phi=\begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 & \phi_{17} & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 & 0 & \phi_{28} & 0 \\ 0 & 0 & 1 & 0 & 0 & T & 0 & 0 & \phi_{39} \\ 0 & 0 & 0 & 1 & 0 & 0 & \phi_{47} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \phi_{58} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \phi_{69} \\ 0 & 0 & 0 & 0 & 0 & 0 & e^{-\alpha_x T} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\alpha_y T} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-\alpha_z T} \end{bmatrix} \tag{3}$$

where

$$\phi_{17}=(\alpha_x T-1+e^{-\alpha_x T})/\alpha_x^{\,2}$$
$$\phi_{28}=(\alpha_y T-1+e^{-\alpha_y T})/\alpha_y^{\,2}$$
$$\phi_{39}=(\alpha_z T-1+e^{-\alpha_z T})/\alpha_z^{\,2}$$
$$\phi_{47}=(1-e^{-\alpha_x T})/\alpha_x$$
$$\phi_{58}=(1-e^{-\alpha_y T})/\alpha_y$$
$$\phi_{69}=(1-e^{-\alpha_z T})/\alpha_z$$

$$\Gamma_k=\begin{bmatrix} \Gamma_1 & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times1} & \Gamma_2 & 0_{3\times1} \\ 0_{3\times1} & 0_{3\times1} & \Gamma_3 \end{bmatrix} \tag{4}$$

where

$$\Gamma_1=\begin{bmatrix} \gamma_x\left[1-\alpha_x T-e^{-\alpha_x T}+(T^2\alpha_x^{\,2}/2)\right]/\alpha_x^{\,3} \\ \gamma_y\left[1-\alpha_y T-e^{-\alpha_y T}+(T^2\alpha_y^{\,2}/2)\right]/\alpha_y^{\,3} \\ \gamma_z\left[1-\alpha_z T-e^{-\alpha_z T}+(T^2\alpha_z^{\,2}/2)\right]/\alpha_z^{\,3} \end{bmatrix}$$

$$\Gamma_2=\begin{bmatrix} \gamma_x(\alpha_x T+e^{-\alpha_x T}-1)/\alpha_x^{\,2} \\ \gamma_y(\alpha T+e^{-\alpha_y T}-1)/\alpha_y^{\,2} \\ \gamma_z(\alpha T+e^{-\alpha_z T}-1)/\alpha_z^{\,2} \end{bmatrix}$$

$$\Gamma_3=\begin{bmatrix} \gamma_x(1-e^{-\alpha_x T})/\alpha_x \\ \gamma_y(1-e^{-\alpha_y T})/\alpha_y \\ \gamma_z(1-e^{-\alpha_z T})/\alpha_z \end{bmatrix}$$

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 [12]. T is system measurement period.

$$H_k=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

3.2 Radar data debiased converted measurement

In the spherical coordinate, the true measurements of radar are azimuth angle θ_m, pitch angle η_m and radial distance r_m,with noise variance as $\sigma_\theta^2,\sigma_\eta^2,\sigma_r^2$, respectively. The average true deviation μ_k and average true covariance R_k of converted measurement are described as [13,14]

$$\mu_k=[\mu_k^x,\mu_k^y,\mu_k^z]^T \tag{6}$$

$$R_k=\begin{bmatrix} R_k^{xx} & R_k^{xy} & R_k^{xz} \\ R_k^{yx} & R_k^{yy} & R_k^{yz} \\ R_k^{zx} & R_k^{zy} & R_k^{zz} \end{bmatrix} \tag{7}$$

where

$$\begin{cases}\mu_k^x = r_m\cos\eta_m\cos\theta_m(e^{-\sigma_\eta^2}e^{-\sigma_\theta^2} - e^{-\sigma_\eta^2/2}e^{-\sigma_\theta^2/2})\\ \mu_k^y = r_m\cos\eta_m\sin\theta_m(e^{-\sigma_\eta^2}e^{-\sigma_\theta^2} - e^{-\sigma_\eta^2/2}e^{-\sigma_\theta^2/2})\\ \mu_k^z = r_m\sin\eta_m(e^{-\sigma_\eta^2} - e^{-\sigma_\eta^2/2})\end{cases}$$

$$\begin{cases}R_k^{xx} = [r_m{}^2(\tilde\beta_x\tilde\beta_{xy} - \tilde\alpha_x\tilde\alpha_{xy}) + \sigma_r^2(2\tilde\beta_x\tilde\beta_{xy} - \tilde\alpha_x\tilde\alpha_{xy})]e^{-2\sigma_\theta^2}e^{-2\sigma_\eta^2}\\ R_k^{yy} = [r_m{}^2(\tilde\beta_y\tilde\beta_{xy} - \tilde\alpha_y\tilde\alpha_{xy}) + \sigma_r^2(2\tilde\beta_y\tilde\beta_{xy} - \tilde\alpha_y\tilde\alpha_{xy})]e^{-2\sigma_\theta^2}e^{-2\sigma_\eta^2}\\ R_k^{zz} = [r_m{}^2(\tilde\beta_z - \tilde\alpha_z) + \sigma_r^2(2\tilde\beta_z - \tilde\alpha_z)]e^{-2\sigma_\eta^2}\\ R_k^{xy} = [r_m{}^2(\tilde\beta_{xy} - \tilde\alpha_{xy}e^{\sigma_\theta^2}) + \sigma_r^2(2\tilde\beta_{xy} - \tilde\alpha_{xy}e^{\sigma_\theta^2})]\sin\theta_m\cos\theta_m e^{-4\sigma_\theta^2}e^{-2\sigma_\eta^2}\\ R_k^{yz} = [r_m{}^2(1-e^{\sigma_\eta^2}) + \sigma_r^2(1-e^{\sigma_\eta^2})]\sin\theta_m\sin\eta_m\cos\eta_m e^{-\sigma_\theta^2}e^{-4\sigma_\eta^2}\\ R_k^{xz} = [r_m{}^2(1-e^{\sigma_\eta^2}) + \sigma_r^2(1-e^{\sigma_\eta^2})]\cos\theta_m\sin\eta_m\cos\eta_m e^{-\sigma_\theta^2}e^{-4\sigma_\eta^2}\end{cases}$$

$$\begin{cases}\tilde\alpha_x = \sin^2\theta_m * \sinh\sigma_\theta^2 + \cos^2\theta_m * \cosh\sigma_\theta^2\\ \tilde\alpha_y = \sin^2\theta_m * \cosh\sigma_\theta^2 + \cos^2\theta_m * \sinh\sigma_\theta^2\\ \tilde\alpha_z = \sin^2\eta_m * \cosh\sigma_\eta^2 + \cos^2\eta_m * \sinh\sigma_\eta^2\\ \tilde\alpha_{xy} = \sin^2\eta_m * \sinh\sigma_\eta^2 + \cos^2\eta_m * \cosh\sigma_\eta^2\end{cases}$$

$$\begin{cases}\tilde\beta_x = \sin^2\theta_m * \sinh 2\sigma_\theta^2 + \cos^2\theta_m * \cosh 2\sigma_\theta^2\\ \tilde\beta_y = \sin^2\theta_m * \cosh 2\sigma_\theta^2 + \cos^2\theta_m * \sinh 2\sigma_\theta^2\\ \tilde\beta_z = \sin^2\eta_m * \cosh 2\sigma_\eta^2 + \cos^2\eta_m * \sinh 2\sigma_\eta^2\\ \tilde\beta_{xy} = \sin^2\eta_m * \sinh 2\sigma_\eta^2 + \cos^2\eta_m * \cosh 2\sigma_\eta^2\end{cases}$$

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

$$Z_c = Z_k - \mu_k = \begin{bmatrix} r_m\cos\eta_m\cos\theta_m \\ r_m\cos\eta_m\sin\theta_m \\ r_m\sin\eta_m\end{bmatrix} - \mu_k \tag{8}$$

3.3 DCMKF algorithm

The calculation steps of DCMKF is given as

1) Calculating the initial value X_0 and initial covariance matrix P_0, and assuming $X_f = X_0$, $P_f = P_0$.

2) Predicting state vector

$$X_p = \Phi X_f \tag{9}$$

3) Calculating covariance matrix of the predicted states

$$P_p = \Phi P_f \Phi^{\mathrm{T}} + \Gamma Q \Gamma^{\mathrm{T}} \tag{10}$$

4) Calculating the mean deviation μ_k and covariance R_k of the converted measurement according to equation (6) and (7)

5) Calculating gain matrix

$$K_k = P_p H^{\mathrm{T}}(HP_pH^{\mathrm{T}} + R_k)^{-1} \tag{11}$$

6) Updating state vector

$$X_f = X_p + K_k(Z_k - \mu_k - HX_p) \tag{12}$$

7) Updating covariance matrix

$$P_f = (I - K_k H)P_p \tag{13}$$

8) Repeating step 2) to 6) for recursive computation

The block diagram of DCMKF algorithm is shown in Fig.6.

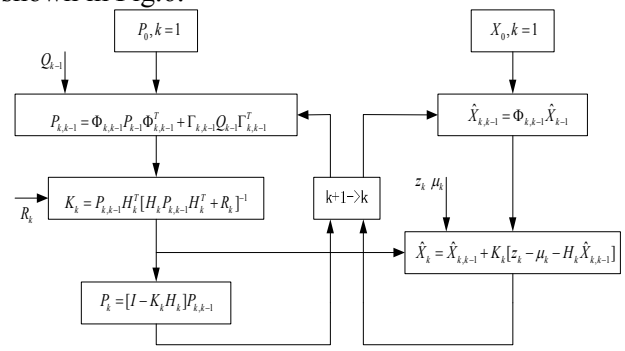

Fig.6. Block diagram of DCMKF algorithm

4 Design of DCMKF based on FPGA

4.1 Structural hierarchy design

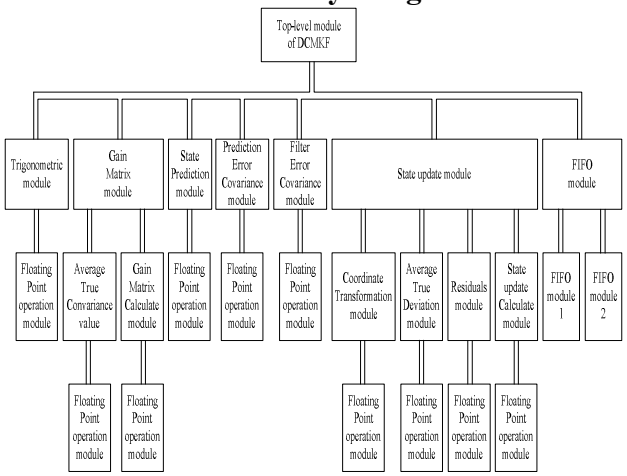

Fig.7. Structural hierarchy design of DCMKF based on FPGA

This paper adopts the structural hierarchy design idea to design DCMKF algorithm, the hierarchy diagram of DCMKF based on FPGA is shown in Fig.7. The bottom module selects very high speed integrated circuits hardware description language (VHDL) as input, while the top module selects schematic diagram as input. This design scheme improves code readability, facilitates simulation in the design, and easy to module partition. To guarantee the calculation precision, the 32-bit single precision floating point number based on IEEE754 standard is utilized in the module design. We choose Altera' Cyclone□ series EP3C120F484C8N chip to achieve DCMKF algorithm on Quartus□platform. The arithmetic operation modules of Quartus Ⅱ on floating point number consist of addition, subtraction, multiplication and division. The basic floating point number operation modules are instantiated and the corresponding parameters are set accordingly in the VHDL code design, which would improve the design performance and shorten

the design time, simply the realization the data path of floating point.

4.2 Design of computing modules of DCMKF

To realize DCMKF algorithm by FPGA, this algorithm needs to be pre-processed from matrix form to vector form [15]. 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[16]. Scientific Workspace software can show clearly the variable relationship between input matrix array and output matrix array. 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 [17,18].

4.2.1 State prediction module

Take the state prediction module as an example. State predicted value can be modified as

$$Xp = [Xp_0, Xp_1, Xp_2, Xp_3, Xp_4, Xp_5, Xp_6, Xp_7 Xp_8]^{\mathrm{T}} \quad (14)$$

where

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

Fig. 8 is the structure diagram of state prediction module. State prediction module occupies 2 floating point addition operation units and 2 floating point multiplication 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 2^n, 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.

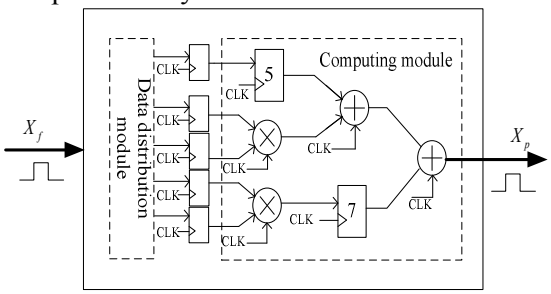

Fig.8. Structure diagram of state prediction module

4.2.2 Prediction error covariance module

Fig.9 is the structure diagram of prediction error covariance module. According to equation (10), this module uses the previous state error covariance value $P_{k-1|k-1}$ to calculate prediction error covariance $P_{k|k-1}$. Before calculating prediction error covariance, the equation (10) needs to be broken down into the scalar form. Predicted error covariance module occupies 8 floating point multiplication computing units and 9 floating point addition computing units.

In order to guarantee data synchronization, the data which don't participate in computing should be set for 5 clock delays in the first stage floating point multiplication. Similarly, the previous stage result should be set for 7 clock delays in the third and fourth stage floating point addition. When input port receives the full previous filter error covariance value one clock cycle by another, the data distribution module sends its corresponding multiplicand and multiplier to the right register for processing at each clock, and enable computing module. The prediction error covariance module output processed data at per clock cycle after 33(5+7+7+7+7) clock cycles.

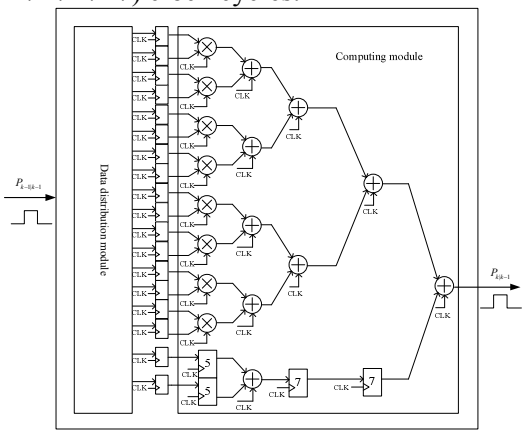

Fig.9. Structure diagram of prediction error covariance module

4.2.3 Average real deviation module

Fig.10 is the structure diagram of average real deviation module. Average real deviation module uses equation (6) to calculate average real deviation of coordinate transform. Before calculating average real deviation, the equation (6) needs to be broken down into the scalar form. Average true deviation module occupies 3 floating point multiplication computing units. When input port receives the radial distance, the pitch angle and azimuth angle 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 10(5+5) clock cycles, average real deviation module output processed data at per clock cycle.

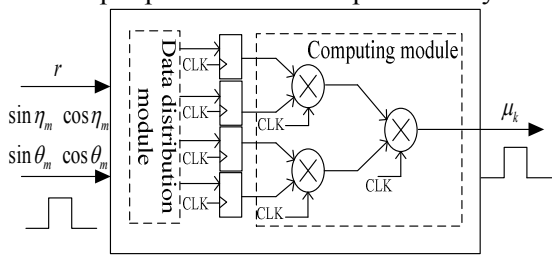

Fig.10. Structure diagram of average real deviation module

4.2.4 Gain matrix module

Fig.11 is the structure diagram of gain matrix module. According to equation (11), gain matrix module uses prediction error covariance module and innovation module to calculate gain matrix. Before calculating gain matrix, the equation (11) needs to be broken down into the scalar form. Gain matrix module occupies 1 floating point addition computing units, 3 multiplication computing units and 2 subtraction computing units. Period parameter of floating point subtraction module is set as 5 clock cycles in LPM. The data which don't participate in operation should be set 7 clock delays in the second stage floating point multiplication to guarantee data synchronization. When input port receives prediction error covariance and innovation 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, the gain matrix module output processed data at per clock cycle.

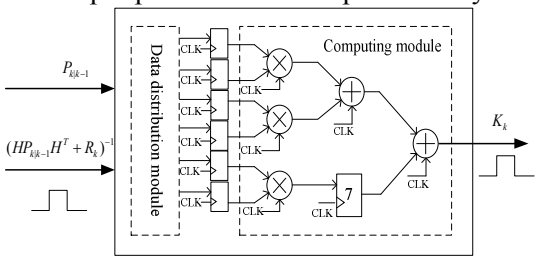

Fig.11. Structure diagram of gain matrix module

4.2.5 State update module

Fig.12 is the structure diagram of state update module. State update module uses gain matrix module, innovation module and state prediction module, according to equation (12) to calculate state update value. Before calculating state update value, equation (12) needs to be broken down into the scalar form. State update module occupies 3 addition floating point computing units and 3 floating point multiplication computing units. The data which don't participate in computing should be set 5 clock delays in the first stage floating point multiplication computing, 7 clock delays in the second and third stage floating point addition computing to guarantee data synchronization. When input port receives full state prediction value, gain matrix and innovation 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 update module output processed data at per clock cycle.

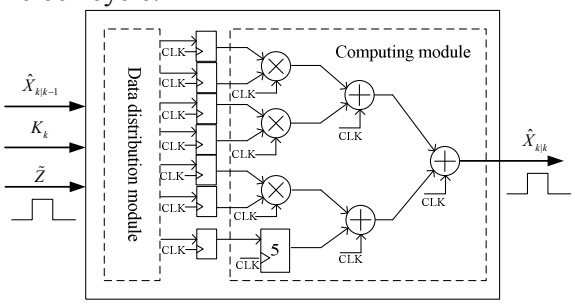

Fig.12. Structure diagram of state update module

4.2.6 Filter error covariance module

Fig.13 is the structure diagram of filter error covariance module. Filter error covariance module uses gain matrix module and prediction error covariance module, according to equation (13) to calculate filter error covariance. Before calculating filter error covariance, the equation (13) needs to be broken down into the scalar form. Filter error covariance module occupies 1 floating point addition computing units, 3 multiplication computing units and 2 subtraction computing units. The data which don't participate in computing should be set 5 clock delays in the floating point multiplication computing and 7 clock delays in the floating point addition computing to guarantee data synchronization. When input port receives full prediction error covariance and gain matrix 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, the filter error covariance module output processed data at per clock cycle.

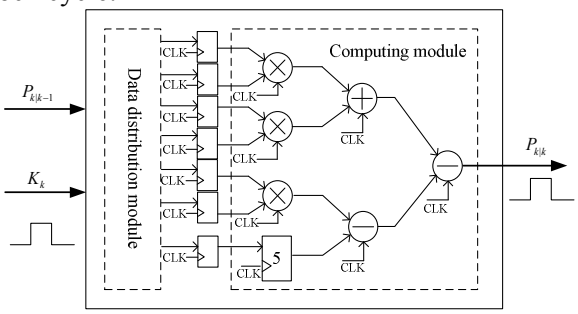

Fig.13. Structure diagram of filter error covariance module

4.3 Top level schematic diagram of DCMKF

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 DCMKF are connected, while others are independent absolutely. For example, there are no direct interaction between state prediction module and prediction error covariance module, the two modules can be parallel calculated. Moreover, some modules are connected. The output of prediction error covariance module is one input of gain matrix module. In order to make the modules run synchronously, we design handshaking signal among modules to enable the next module to receive data 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.

4.3.1 Trigonometric function module

Both trigonometric function values of pitch angle and azimuth angle are applied in coordinate transform module, the real average deviation module and the real average covariance module of DCMKF. To save hardware recourse, these trigonometric function values are calculated only once. The function values will be utilized when necessary to realize the rare mutiplication modules with the least register.

Fig.14 is trigonometric function module. Trigonometric function module's input port is the pitch angle of the target, the azimuth angle of the target, clock signal, reset signal and input enable signal. Trigonometric function module's output port is the sine and cosine value of the pitch angle, the azimuth angle and output enable signal for next module.

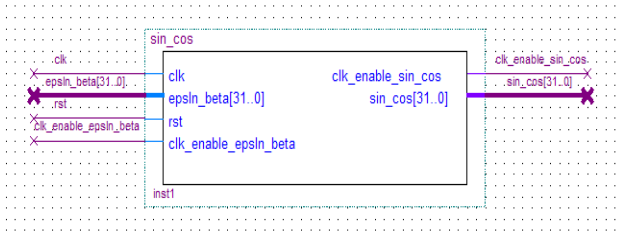

Fig.14. Trigonometric function module

4.3.2 State prediction module and prediction error covariance module

Both initial state prediction value and prediction error covariance value are transmitted from DSP to FPGA, while values hereafter will be read in FIFO module within FPGA. Therefore, a data selector is designed before state prediction module and prediction error covariance module to select different data path in different time.

Fig.15 is state prediction module and prediction error covariance 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. Prediction error covariance module's input port is the previous time filter error covariance value, clock signal, reset signal and input enable signal. State prediction module's output port is the prediction error covariance value and output enable signal for next module.

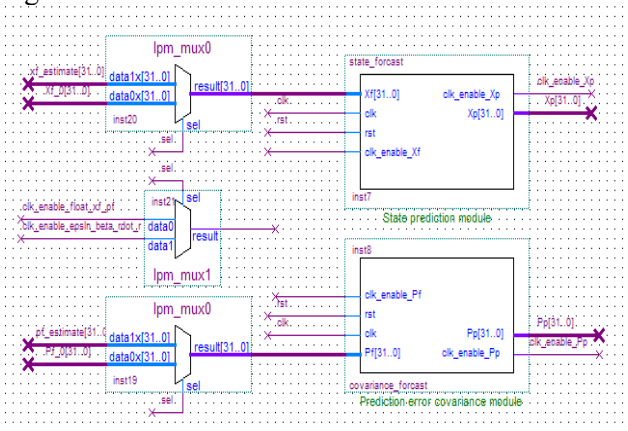

Fig.15. State prediction module and prediction error covariance module

4.3.3 Gain matrix module

In gain matrix module, the average true covariance module is used to calculate the value of gain matrix, which is shown by the two sub modules in Fig.16.

For gain matrix module, its input ports are the radial distance of target, the trigonometric values of pitch angle and azimuth angle, prediction error covariance value, clock signal, reset signal and these inputs enable signals. Its output ports are gain matrix value and enable signal.

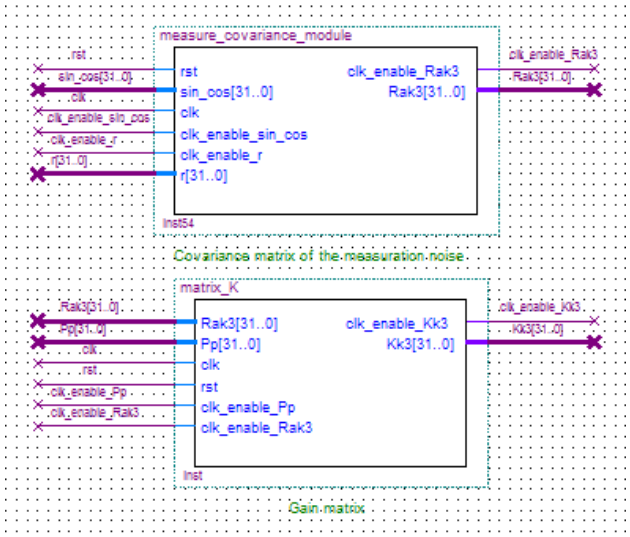

Fig.16 Gain matrix module

4.3.4 State update module

In state update module, the coordinate transform module, average real debias modification module, innovation calculating module and state update module are used to calculate the updated value of state. These four processes are shown by the 4 sub modules in Fig.17.

For state update module, its input ports are the radial distance of target, the trigonometric values of pitch angle and azimuth angle, state prediction value, gain matrix value, clock signal, reset signal and these inputs enable signals. Its output ports are state update value and two enable signals, where one is for lpm_fifo1 writing enable module, the other for asynchronous FIFO writing enable module.

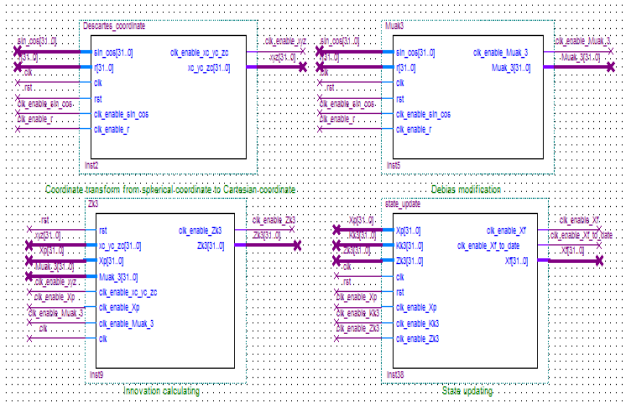

Fig.17 State update module

4.3.5 Filter error covariance module

Fig.18 is the filter error covariance module. The input ports are prediction error covariance value, gain matrix value, clock signal, reset signal and two input enable signals. Output ports are filter error covariance value and enable signal. This enable signal is writing enable signal of lpm_fifo2.

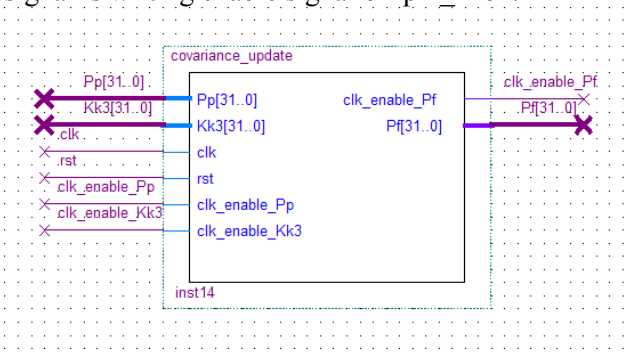

Fig.18 Filter error covariance module

4.3.6 FIFO memory module

Fig.19 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 and can easily called for using later. The two 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 is similarly with lpm_fifo1.

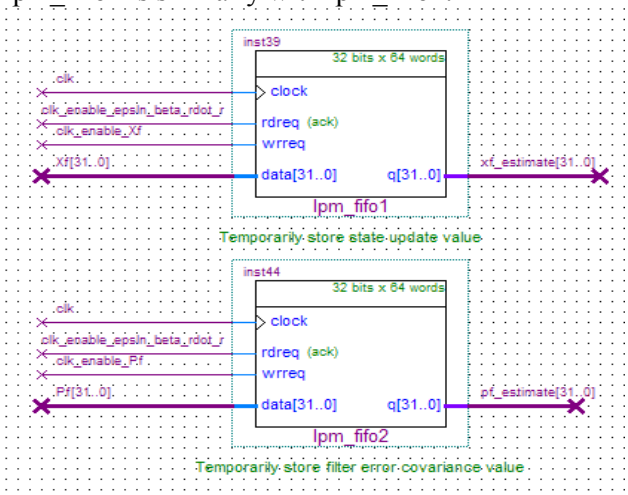

Fig.19 Synchronous FIFO memory module

5 Experimental results and analysis

Fig.20 shows the hardware circuit board of DSP subsystem, this subsystem is composed of DSP, CPLD, SDRAM, FLASH and other components. The power supply of this circuit board is DC 5V. This board is connected with the FPGA subsystem through two communication connector.

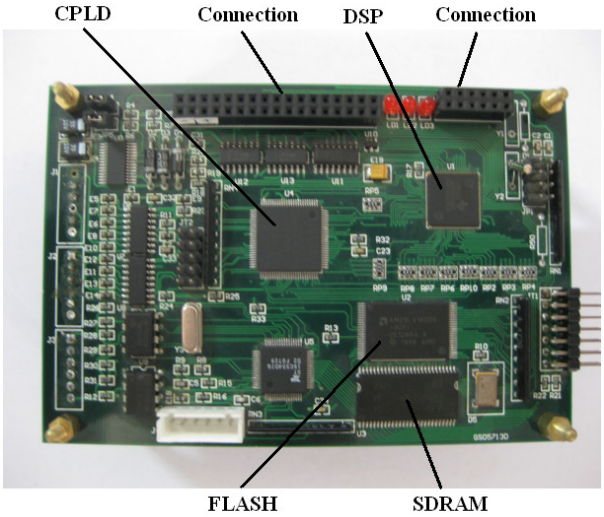

Fig.20. DSP subsystem

Fig.21 shows the hardware circuit board of FPGA subsystem. The core chip of this circuit board is the FPGA chip. This board's power supply is 24V DC. Firstly, the power convert chip converts the DC 24V to DC 5V, then supply to the DSP subsystem through the connection. In the FPGA subsystem, DC 5V is converted to 3.3V, 2.5V and 1.2V respectively.

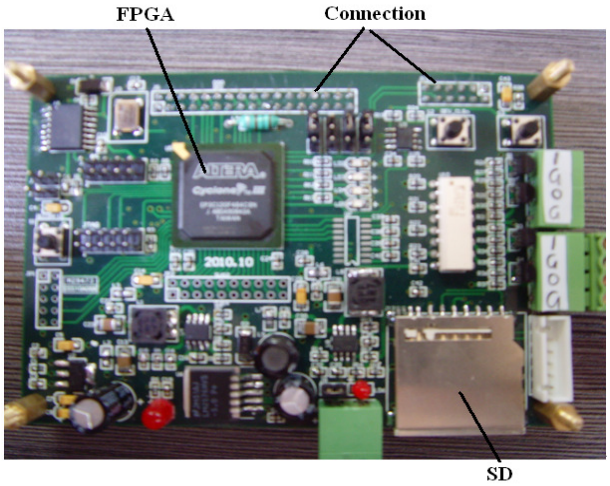

Fig.21. FPGA subsystem

The following example of tracking a short range attacking target is considered. The target trajectory is shown in Fig.22. The parameters of target are given as follows, the pitch angle remains $\pi/90$, the azimuth angle reduces from $\pi/90$ to $\pi/1000$, the initial radial distance is 700m, initial speed is 400m/s, radar tracking sampling period is 0.8192ms, system state noise covariance Q is 3×3 unit matrix, measurement noise covariance of radial distance, pitch angle and azimuth angle in equation (7) are $\sigma_r^2 = 1/36$, $\sigma_\eta^2 = \pi^2/72900$, $\sigma_\theta^2 = \pi^2/32400$, respectively. The target's first-order forming filter parameters of acceleration in Cartesian coordinate are $\gamma = 0.0129$, $\alpha = 0.001$. The DCMKF algorithm is realized respectively in FPGA and Matlab platform using the same measurements. Fig.23 shows position comparison in X direction, Fig.24 and Fig.25 show position comparison in Y direction and Z direction. It is easily seen from the 3 figures above that the DCMKF is capable of denoising and smoothing for target position. Fig.23, Fig.24 and Fig.25 show the results of FPGA are consistent with the simulated results by Matlab. The high precision proves the correctness of this design scheme.

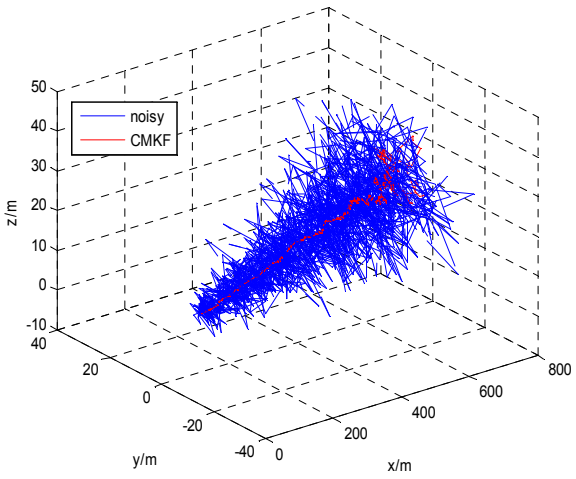

Fig.22 The target trajectory

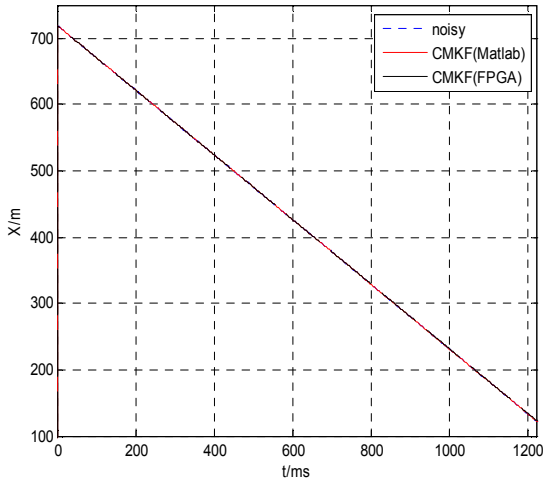

Fig.23 Position comparison in X direction

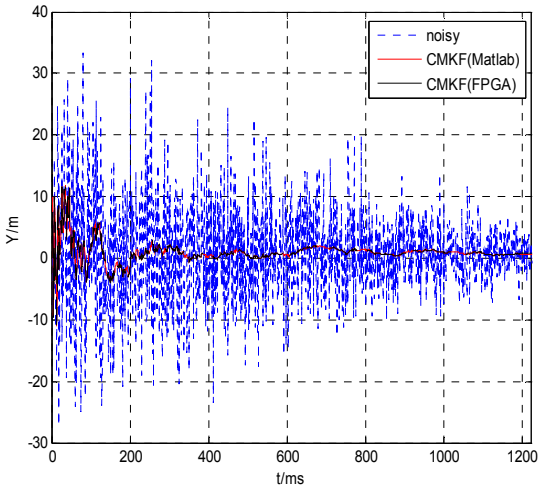

Fig.24 Position comparisons in Y direction

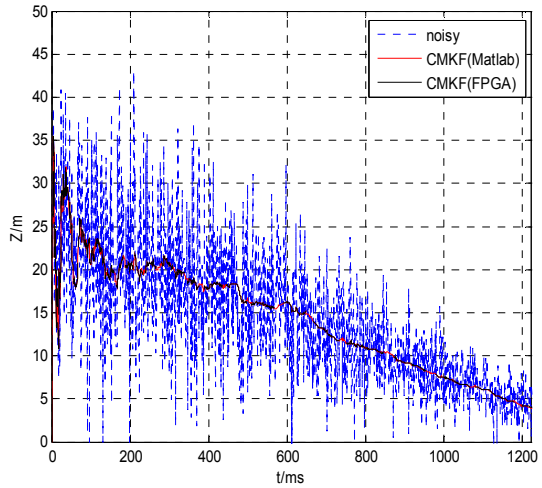

Fig.25 Position comparisons in Z direction

Experimental results prove the designed DCMKF algorithm based on FPGA spends 407 clock periods to complete one filter process. A filtering cycle is 4.07us when the clock period is 10ns. This time fully satisfies the real time requirement in target tracking system. The radar tracking performance gained in our design scheme includes two to three orders of magnitude higher speed than single DSP design scheme.

6 Conclusion

In the radar target tracking system, the tracking precision and real time are highly required. DCMKF algorithm includes a great deal of matrix arithmetic, such as matrix addition, matrix subtraction, matrix multiplication and inverse. The computational time for calculating DCMKF algorithm in software is too long to meet the real time of target tracking. In this paper, the FPGA is used as a floating point co-processor 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 large 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 short-range defence radar.

Acknowledgement

This work is partially supported by the National Nature Science Foundation of China (NO. 61104196), China Specialized Research Fund for the Doctoral Program of Higher Education(NO. 200802881017), NUST Research Funding(NO. 2010ZYTS051), “Zijin star” Research Funding(NO. AB41381).

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 defence technology*, Vol.24, No.5, 2002, pp. 57-60.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

[12] Bar-Shalom Y, Li X R, Kirubarajan T. *Estimation with Applications to Tracking and Navigation: Therory, Algorithm and Software*[M]. New York*:Wiley*, 2001.

[13] Liu Jian-feng, Zhuang Zhi-hong, Liu Zhong. Application Research of Converted Measuring Kalman Filter Used in Burst Control Technology of Missile[J]. *Journal of Electronics & Information Technology*, Vol. 27, No.9, 2005, pp. 1388-1392.

[14] Suchomski P. Explicit Expressions for Debiased Stastics of 3D Converted Measurements[J]. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, No.3, 1999, pp. 1015-1022.

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

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

[17] 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.

[18] 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.