On the FPGA Implementation of Tracking Algorithms

A. Belaabed, Ch. Benbouchama

***Abstract*—This paper presents the FPGA (Field Programmable Gate Array) implementation of tracking algorithms based on the estimation of a moving target status in space. Two Bayesian estimators, Kalman Filter (KF) and Extended Kalman Filter (EKF), are used to track the targets with several interacting models. Hardware architectures have been proposed for different cases. The targeted circuit was an XC2v1000 FPGA embedded on the Celoxica RC200 board. A tool was designed with visual C++ to generate automatically FPGA configurations, through the use of Handel-C language. This work was completed with an experimental validation which consists in the real-time tracking of sinusoidal signal.**

I. Introduction

The target tracking is a necessary part of systems that perform functions such as surveillance, guidance, obstacle avoidance. Tracking algorithms take their input measurements from sensors which provide the signals such as radar, sonar or video [1]. The measurements are taken at regular intervals and the task is to estimate the state of a target at each point in time, such as its position, velocity or other parameters. Several methods have been developed to estimate those parameters. Among these methods, the Kalman Filter (KF) appears to be an efficient technique for the online estimation. The last few years have seen an unprecedented effort by researchers on the tracking algorithms hardware implementation. In order to accelerate the processing, it was agreed to be directed towards solutions based on parallel computation. Many works were interested on the approach using the parallelism aspect provided by the FPGA circuits [2],[3],[4],[5] and [6]. These implementations were constrained by two main constraints: The real-time processing and the resources of the targeted FPGA. In this paper we were interested on the hardware implementation of a Kalman Filter (KF) and Extended Kalman Filter (EKF) algorithms, which takes advantage of data parallelism for a further implementation on FPGA. Those implementations require complex arithmetic operations, such us trigonometric and square root elementary functions. The targeted circuit was an XC2v1000 FPGA embedded on the Celoxica RC200 board. A tool was designed to generate automatically FPGA configurations implementations for Kalman Filter implementation.

Manuscript received January, 2012.

A. Belaabed was with the the Department of Electronic, Ecole Militaire Polytechnique, Bordj-El-Bahri 16111, Alger, Algeria.

Ch. Benbouchama is with the Department of Electronic, Ecole Militaire Polytechnique, Bordj-El-Bahri 16111, Alger, Algeria (corresponding author e-mail: ben_cherrad@ yahoo.fr).

II. Theoretical Notions

A. Kalman Filter (KF)

The block diagram of the Kalman filter can be drawn as in Fig. 1. The current output only depends on the current input and current state (previous output) [7].

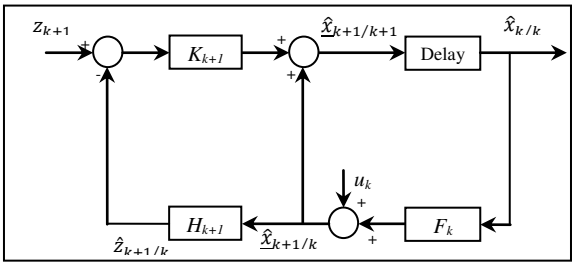


Fig. 1: Block diagram of Kalman filter.

Now we can set up the state equation of KF as the following:

$$\begin{cases} x_{k+1} = Fx_k + Bu_k \\ z_k = Hx_k + v_k \end{cases} \quad (1)$$

where

x_k : State estimation vector for time instant *k.*

z_k *:* Observation vector at time instant *k.*

u_k : Process noise.

v_k : Measurement noise.

H : Measurement (observation) matrix.

F : State transition matrix.

B : Control Matrix.

Filter starts with an initial estimation of the state and the covariance matrix $\hat{x}_0$ and P_0 and estimate the state and covariance to time step 1. Then the following steps follow:

Prediction :

$$\hat{x}_{k/k-1} = F\hat{x}_{k-1} \quad (2)$$

$$P_{k/k-1} = FP_{k-1}F^T + B.Q.B \quad (3)$$

Estimation :

$$\hat{x}_k = \hat{x}_{k/k-1} + K_k\left(y_k - H\hat{x}_{k/k-1}\right) \quad (4)$$

$$P_k = (I - K_kH)P_{k/k-1}$$

$$K_k = P_{k/k-1}H^T\left(HP_{k/k-1}H^T + R\right)^{-1} \quad (5)$$

where

$\hat{x}_{k/k-1}$: Sate Prediction vector for time instant *k.*

$P_{k/k-1}$: Prediction error covariance matrix.

P_k: Estimation error covariance matrix.

R : Measurement noise covariance matrix.

Q : Process noise covariance matrix.

K : Kalman gain matrix.

In the steady state Kalman filter the P_k and K_k matrices are constant, so they can be hard-coded as constants, and the only Kalman filter equation that needs to be implemented in real time, which consists of simple multiplies and addition steps.

B. Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) must be used if the measurements are a non-linear combination of the state variable vector [8]. The EKF model is:

$$\begin{cases} x_k = f(x_{k-1}, u_{k-1}) \\ y_k = h(x_k, v_k) \end{cases} \quad (6)$$

The linearization of the system equations is done by the calculation of the Jacobian matrix:

$$F_{k-1} = \frac{\partial f(x_{k-1})}{\partial x_{k-1}}\Big|_{x_{k-1}=\hat{x}_{k-1}} \qquad B_{k-1} = \frac{\partial f(\hat{x}_{k-1})}{\partial u_{k-1}}\Big|_{u_{k-1}=0}$$

$$H_k = \frac{\partial h(x)}{\partial x}\Big|_{x=\hat{x}_{k/k-1}} \qquad C_k = \frac{\partial h(\hat{x}_{k/k-1})}{\partial v_k}\Big|_{v_k=0}$$

The different steps of the EKF algorithm are described as follows [9]:

Initialization $\hat{x}_0$ et P_0

Prediction :

$$\hat{x}_{k/k-1} = f(\hat{x}_{k-1}) \quad (7)$$

$$P_{k/k-1} = F_{k-1} P_{k-1} {F_{k-1}}^T + B_{k-1}.Q.{B_{k-1}}^T \quad (8)$$

Estimation :

$$\hat{x}_k = \hat{x}_{k/k-1} + K_k\left(y_k - h(\hat{x}_{k/k-1})\right) \quad (9)$$

$$K_k = P_{k/k-1} {H_k}^T (H_k P_{k/k-1} {H_k}^T + C_k R {C_k}^T)^{-1} \quad (10)$$

$$P_k = (I - K_k H_k) P_{k/k-1} \quad (11)$$

C. Movement Models

The models of the target movement, used in this work can be done as following [10]:

Constant velocity (M1 model):

$F_1 = diag\{F, F\}$ and $B_1 = diag\{B, B\}$

$$\text{with } F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} T/2 \\ T \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Constant acceleration (M2 model):

$F_2 = diag\{F, F\}$ and $B_2 = diag\{B, B\}$

$$\text{with } F = \begin{bmatrix} 1 & T & T/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} B = \begin{bmatrix} T/2 \\ T \\ 1 \end{bmatrix} H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

III. Implementation Techniques

The FPGA implementation of the algorithms cited above requires the use of different hardware implementation methods of some mathematical functions, for example: the square root, cosine, the sine and the arc-tangent. Moreover, and for an optimal implementation, it is recommended to implement the various blocks of the filter with 18-bit fixed-point representation, in order to benefit from available 18-bit x 18-bit hardware multiplier blocks [11].

A. The Taylor Series

The general form of the Taylor expansion is shown in the following equation [11]:

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \cdots + \frac{f^{(n)}(x_0)(x - x_0)^n}{n!} \quad (12)$$

where $f^{(n)}$ is the n-th derivative of f function. Commonly in practice $x_0 = 0$, and the above expansion simplifies to the Maclaurin series as shown in the following equation:

$$P_n(x) = f(0) + f'(0)x + \cdots + \frac{f^{(n)}(0)x^n}{n!} \quad (13)$$

B. CORDIC Algorithm

The basic idea of CORDIC (Coordinate Rotation Digital Computer) is to rotate a vector by multiplying its magnitude by a series of constant values [12]. The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shifts and adds to calculate the trigonometric and hyperbolic functions. The mathematical relations for the CORDIC rotations are as follows:

$$v_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$v_{i+1} = R_i v_i$$

$$R_i = \begin{pmatrix} \cos\gamma_i & -\sin\gamma_i \\ \sin\gamma_i & \cos\gamma_i \end{pmatrix} \qquad v_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

$$v_{i+1} = R_i v_i = K_i \begin{pmatrix} x_i & -y_i \sigma_i 2^{-i} \\ x_i \sigma_i 2^{-i} & y_i \end{pmatrix} \quad (14)$$

with : $\sigma_i = \pm 1$

$$K_i = \cos(\tan^{-1}(2^{-i})) \quad (15)$$

$$K_{(n)} = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1 + 2^{-2i}}} \approx 0{,}6073 \quad (16)$$

From equations (14), (15) and (16) it is obvious that all the multiplications are in powers of 2, which can be implemented with shifters. Simple elementary arithmetic functions like shifters and adders are only required for implementing the CORDIC equations.

C. Newton-Raphson Method

Algebraically, the method derives from the familiar Taylor series expansion of a function in the neighborhood of a point [13]:

$$f(x + \delta) \approx f(x) + f'(x)\delta + \frac{f^{(2)}(x)}{2}\delta^2 + \cdots \quad (17)$$

For small enough values of δ, and for well-behaved functions, the terms beyond linear are unimportant, hence $f(x + \delta) = 0$ implies:

$$\delta = -\frac{f(x_n)}{f'(x_n)} \quad (18)$$

IV. Implementation Results

The objective of this section is to exhibit the various results of Kalman filter implementation with the two movement models M1 and M2, intended for target tracking application, with fixed (FXP) and floating-point (FP) data representation. In addition to this, EKF implementation is

used to estimate the parameters of a sinusoidal signal (amplitude, pulsation and phase). The targeted FPGA was Virtex II XC2V1000-4FG456C embedded on the RC200 board (Figure 2).

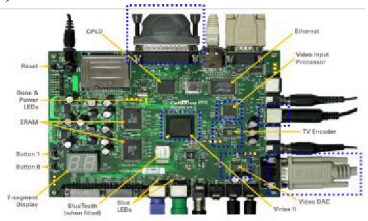


Fig. 2: RC200 board.

The main results were evaluated and compared with those obtained by MATLAB with the use of a Handel-C/MATLAB co-simulation.

Using a high-level tool represented by the GUI (graphical unit interface) of figure 3, we obtained the following implementation results.

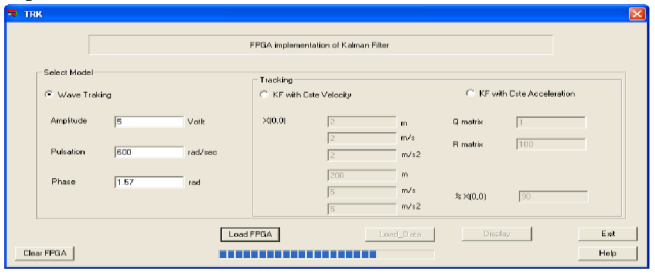


Fig. 3: The graphical unit interface.

A. *Kalman Filter with M1 model*

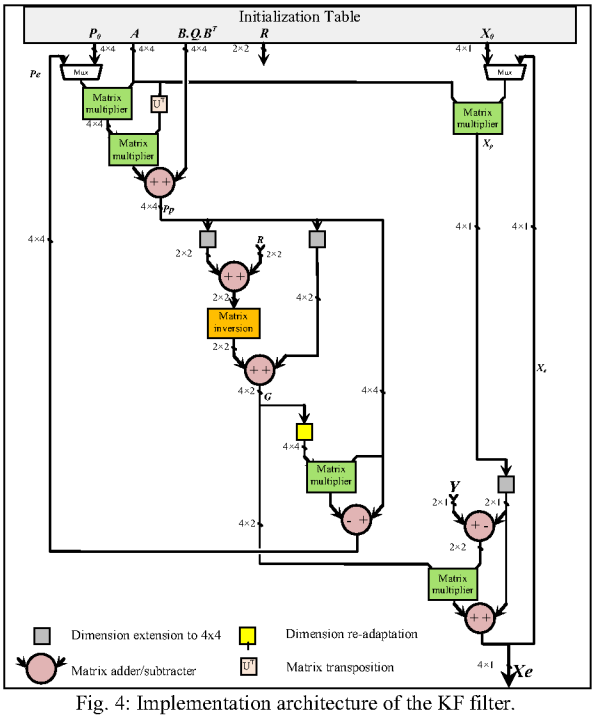
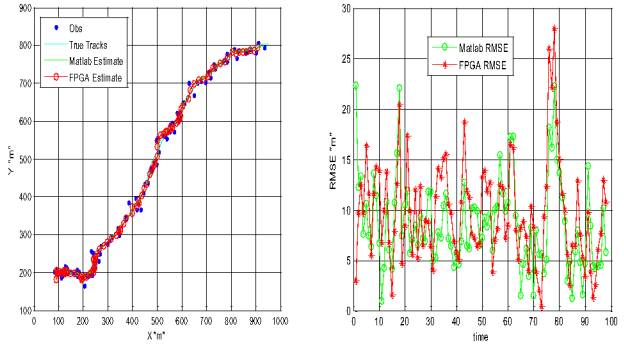


Fig. 4: Implementation architecture of the KF filter.

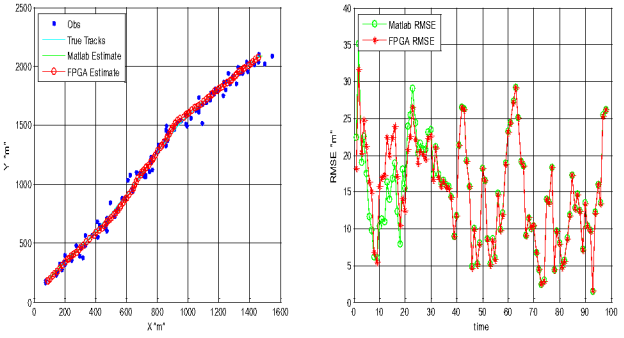
Due to the intensive and complex KF processing, the development of the FPGA architecture needs a special care regarding the optimization of the consumed hardware resources without downgrading timing performances. The implementation architecture is presented in figure 4. The matrix multiplications have been factorized and made by only one 4x4 matrix multiplier. The last one is also factorized, and it uses 4 multipliers to perform successively 64 multiplications. The matrix inversion module implements a divider module for the determinant inversion and a scalar-matrix.

Figures (5-6-7) show the obtained fixed-point and floating-point co-simulation results. The real and estimated state vectors have been compared and a proper convergence of the KF has been obtained. The figure (a) show target real and estimate trajectories and the figure (b) show the root mean squared error (RMSE) for the position.



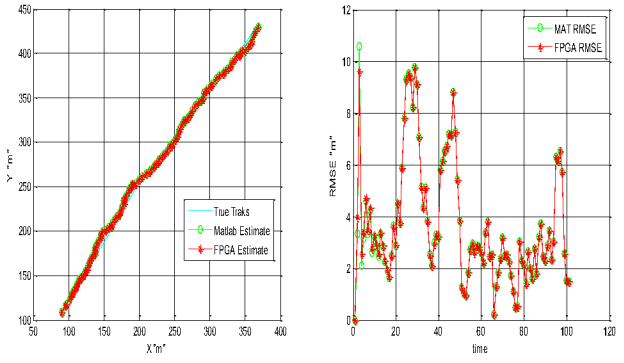
(a) (b)

Fig. 5: Real and estimate trajectories Q=diag([2^2, 2^2]), R=diag([10^2,10^2]) with fixed-point 8Q8: (a) Estimate trajectories, (b) RMSE.



(a) (b)

Figure 6: Real and estimate trajectories Q=diag([0.2^2, 0.2^2]), R=diag([32^2,32^2]) with fixed-point 16Q16: (a) Estimate trajectories, (b) RMSE.



(a) (b)

Fig. 7: Real and estimate trajectories Q=diag([0.2^2, 0.2^2]), R=diag([32^2,32^2]) with flaoting-point: (a) Estimate trajectories, (b) RMSE.

Table I shows the resources consumption and the maximal frequency of each implemented algorithm with a various size of fixed-point and floating-point data representations.

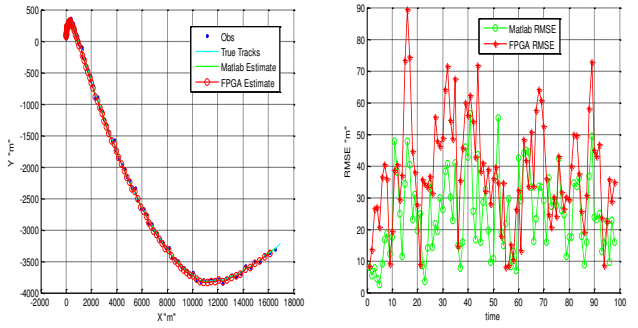
TABLE I IMPLEMENTATION RESULTS FOR M1 MODEL.

Resources	FXP 8Q8	FXP 16Q16	FXP 16Q16 gain offline	FP	FP gain offline
I/O	20 (6%)	20 (6%)	20 (6%)	20(6%)	20(6%)
LUTs	165 (1%)	176 (1%)	113 (1%)	18910 (184%)	118 (1%)
Slices Flip Flops	108 (1%)	111 (1%)	80 (1%)	3835 (37%)	84(1%)
CLB Slice	122 (2%)	123 (2%)	85 (1%)	11363 (221%)	87 (1%)
Frequency	73.42MHz (13.62ns)	73.42 MHz (13.62ns)	73.42MHz (13.62ns)	/	73.42 MHz (13.62ns)

It is noticed that the two major constraints: the technological limits and the processing time are acceptable. We note here that the KF with offline Kalman gain computing is more interested because it gives better result in term of resources consumption.

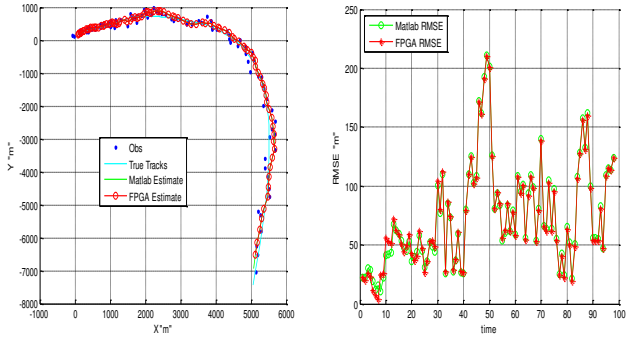
B. Kalman Filter with M2 model

Same thing as the first case, the only difference is the size of used matrices. For this movement model (M1) we use 6x6 matrix multipliers. Figures (8-9-10) show the obtained fixed-point and floating-point co-simulation results. The actual and estimated state vectors have been compared and a proper convergence of the KF has been obtained. The figure (a) shows target real and estimate trajectories and the figure (b) illustrates the root mean squared error (RMSE) for the position.



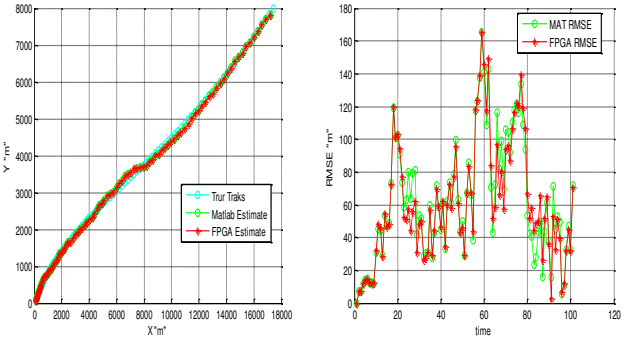
(a) (b)

Fig. 8: Real and estimate trajectories Q=diag([1^2, 1^2]), R=diag([32^2,32^2]) with fixed-point 8Q8: (a) Estimate trajectories, (b) RMSE.



(a) (b)

Fig. 9: Real and estimate Trajectories Q=diag([0.2^2, 0.2^2]), R=diag([100^2,100^2]) with fixed-point 16Q16: (a) Estimate trajectories, (b) RMSE.



(a) (b)

Fig. 10: Real and estimate trajectories (floating-point): (a) Estimate trajectories, (b) RMSE.

Table II shows the resources consumption and the maximal frequency of each implemented algorithm with a various size of fixed-point and floating-point data representations.

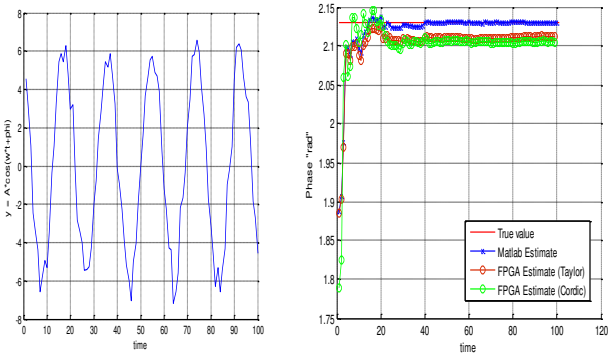
TABLE II IMPLEMENTATION RESULTS FOR KF FILTER WITH M2 MODEL.

Resources	FXP8Q8	FXP 16Q16	FXP 16Q16 gain offline	FP	FP gain offline
I/O	20(6%)	20(6%)	20(6%)	20(6%)	20(6%)
LUTs	198(1 %)	200(1 %)	123(1 %)	24683 (241%)	8972 (87 %)
Slices Flip Flops	131(1 %)	134(1 %)	86(1 %)	5440 (53 %)	1664 (16 %)
CLB Slice	158(3%)	159(3%)	91(1%)	14268 (278%)	5118 (99%)
Frequency	73.42 MHz (13.62ns)	73.42 MHz (13.62ns)	73.42 MHz (13.62ns)	/	73.42 MHz (13.62ns)

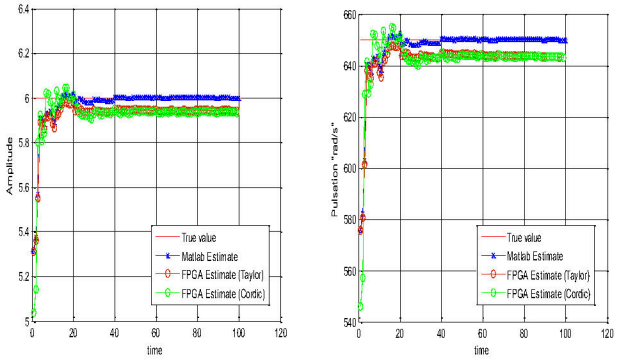
In this part, we remark that the results obtained are the same as precedent results. The consumption does not exceed the total resources available on the Virtex II FPGA.

C. Sinusoidal Signal Parameters Estimation with EKF

The aim is the use of EKF implementation to estimate the parameters of a sinusoidal signal (amplitude, pulsation and phase). Figure 11 shows the obtained co-simulation results. The implementation architecture is presented on figure 12.



(a) (b)



(c) (d)

Fig. 11: Parameters estimation of sinusoidal signal with EKF filter: (a) disturbed signal, (b) phase estimation, (c)amplitude estimation, (d) pulsation estimation.

Table III shows the implementation results obtained by computing the trigonometric functions with CORDIC and Taylor algorithms.

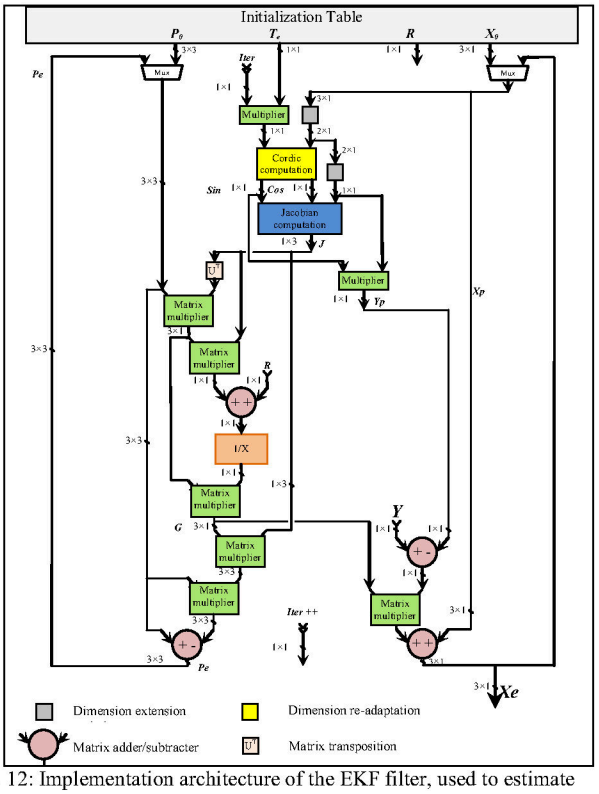


Fig. 12: Implementation architecture of the EKF filter, used to estimate sinusoidal signal parameters.

TABLE III IMPLEMENTATION RESULTS FOR EKF FILTER WITH M1 MODEL.

Resources	CORDIC	Taylor
I/O	20(6%)	20(6%)
LUTs	21951(214%)	24057(234%)
Slices Flip Flops	1686(16%)	1552(15%)
CLB Slice	12849(250%)	13875(270%)

We remark that the consumption does not exceed the total resources available on the Virtex II FPGA. Finally, it is noticed that the best results of EKF implementation are obtained by computing the trigonometric functions with CORDIC algorithm, in terms of resources and precision.

V. EXPERIMENTAL VALIDATION

In order to validate the design approach, we test it by the experimentation of an application utilizing an implemented EKF filter used to track sinusoidal signal. The experimental system consists of a computer, an oscilloscope and the RC200 FPGA board (Figure 13).

Through the sound board of the computer, we generate a disturbed signal defined by a reference frequency. So, the implemented EKF algorithm estimates the reference frequency and the signal is tracked.

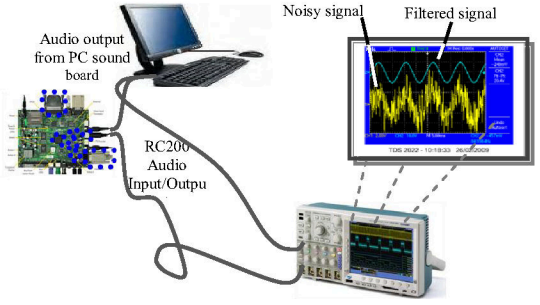


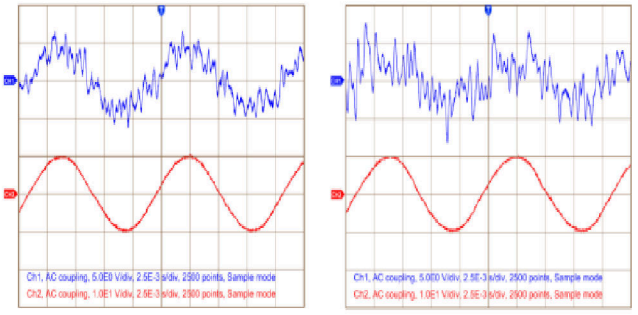
Fig. 13: The experimental system.

Finally, the resulting noise and filtered signals are viewed on the oscilloscope. Table 4 shows the implementation results of the EKF filter used to tracking sinusoidal signal.

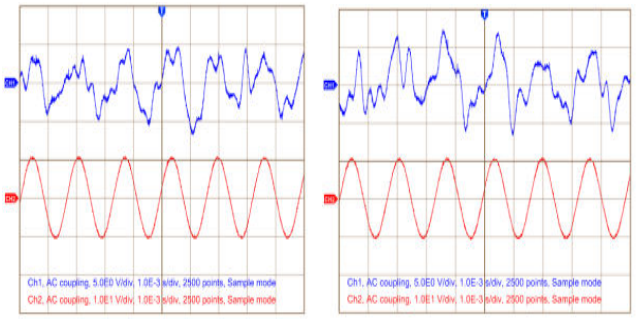
TABLE IV EKF FILTER USED TO TRACKING SINUSOIDAL SIGNAL.

Resources	Total	Consumption
I/O	324	7(2%)
LUTs	10240	6296(61%)
Slice Flip Flops	10240	1898(18%)
CLB Slice	5120	4924(96%)
Frequency	/	19.47 MHz (51.36ns)

It is noticed that the two major constraints: the technological limits and the real-time aspect are respected. The figure 14 shows the obtained results for different values of signal frequency and signal/noise ratio (SNR).



90Hz, 10dB 90Hz, 1dB



611Hz, 5dB 611Hz, 1dB

Fig. 14: Sinusoid tracking with diverse SNR and frequencies.

From the obtained results, it appears that the tracking of the sinusoidal signal is perfect. The maximal used pulsation in our application is 635rad/s because of the hardware limitation (resources) of used FPGA.

VI. Conclusion

We have demonstrated the feasibility of the FPGA implementation of the Kalman algorithms for tracking applications. A tool was designed to generate automatically FPGA configurations for the implementation of Kalman filters. The advantage which it offers resides in the facility of the approach: there is no language and it requires only the use of the graphical menus. To be able to implement Kalman filters for complicated applications with this tool, we must use a board with an FPGA containing significant resources. Finally, experimental validation has been developed to demonstrate the validity of the design approach.

References

[1] S. Blackman, *Multipl-Target Tracking with Radar Applications*. Artech House, 1986.

[2] A. M. Reza and al., "FPGA Implementation of Adaptative Temporal Kalman Filter for Real Time Video Filtering", *IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 2231-2234, 1999.

[3] E. Monmasson and al., "Fully FPGA-based Sensorless Control for AC Drive using an Extended Kalman Filter", *IECON* 2009, pp. 2925-2930, 2009.

[4] C. Charoensak and S. S. Abqsekera, "FPGA Implementation of Efficient Kalman Band-Pass Sigma-Delta Filter for Application in FM Demodulation", *IEEE International Conference SOC*, pp. 137-138, 2004.

[5] S. Reddy.P and R. Reddy. G, "Design and Implementation of Autocorrelation and CORDIC Algorithm for FDM Based WLAN", European Journal of Scientific Research, 2009.

[6] A. Saparon, M.K. Hamzah and M.A.Rongi, "Sinusoidal Pulse Width Modulation using CORDIC Algorithm for Single Phase Matrix Converter", *5th IEEE Conference on Industrial Electronics and Applications*, 2010, pp. 1088-1093.

[7] G. Welch and G. Bishop, "An introduction to the Kalman", *in SIGGraph-2001 course 8: Computer Graphics, Annual Conference on Computer Graphics and Interactive Techniques*, Aug 2001.

[8] J.P. Krief and al., *Le filtrage et ses applications*, 3rd ed., Cépaduès Editions, Toulouse, France, Nov 1993.

[9] P. Lombardo and M. Marsella A. Farina, "Joint tracking and identification algorithms for mulitsensor data", IEEE Proce. Radar Sonar Navig., vol. 149, no. 6, pp. 271-280, 2002.

[10] X. Li and V. Jilkov, *A survey of maneuvring target traking: Dynamic Models*, SPIE: Signal Data Process. Small Targets 2000, vol. 4048-22, Apr 2000.

[11] Mazen A. R. Saghir and al., *A Reconfigurable Platform Architecture for an Automotive Multiple-Target Tracking System*, ACM SIGBED Review, vol. 6, no. 3, october 2009.

[12] J. E. Volder, *The birth of CORDIC*, J. VLSI Signal Processing, vol. 25, no. 2, pp. 101-105, June 2002.

[13] S.R. Otto and J.P. Denier, *An Introduction to Programming and Numerical Methods in MATLAB*, Springer, USA, 2005.