FPGA Based EKF Estimator for DTC Induction Motor Drives

Presented by Yadollah Sabri

Laboratoire Hubert Curien, UMR CNRS 5516
Royal Military College of Canada, RMCC

Bourse Accueil Pro, UJM
February - December , 2010

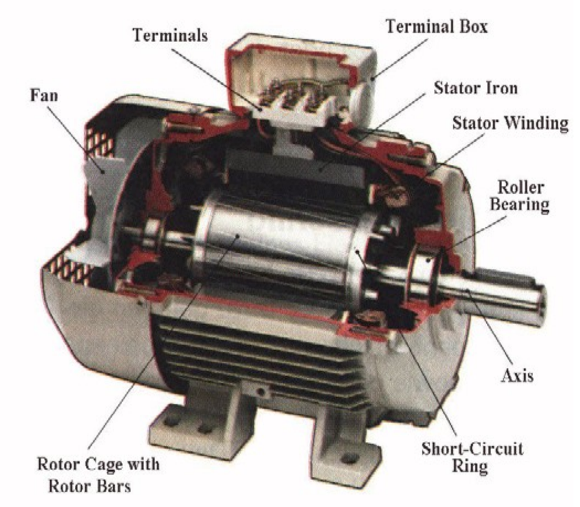
•Contents

- Induction Motors (IM) characteristics
- Mathematical model of IM
- Direct Torque Control (DTC) strategies of IM
- State space model of Induction Motor
- DTC simulation in MATLAB SIMULINK environment
- Implementation of EKF algorithm in FPGA
- Conclusions

- **Induction Motors (IM) characteristics:**

- **Advantages:**

1) **Highly reliable, as its simple design has no brushes that could wear out,**
2) **No electrical sparking.**
3) **Available in single-phase and three-phase; three phase is an ideal choice for variable-speed applications.**
4) **Manufactured at a very low cost.**



Induction Motor Cross Sectional view

- **Applications:**

1) **Traction and Propulsion systems**
2) **Manufacturing machines (CNC) used in variety of production lines such as Drilling, Grinding, Hardening and Etc..**

- **Drawback:**

1) **Less efficient and more sluggish than other motor types like DC Machines**

- **Solutions**
- **Use of effective speed controllers**

1) **Vector controller: Direct or Indirect Field Oriented (DFOC or IFOC)**
2) **Direct Torque Controller (DTC)**

•Mathematical model of IM

•Stator and Rotor voltage equations

$$\begin{cases} U_s^s = R_s.I_s^s + \dfrac{d\psi_s^s}{dt} \\ 0 = R_r I_r^s - j\omega_r.\psi_r^s + \dfrac{d\psi_r^s}{dt} \end{cases} \qquad (1)$$

•Stator and Rotor flux equations

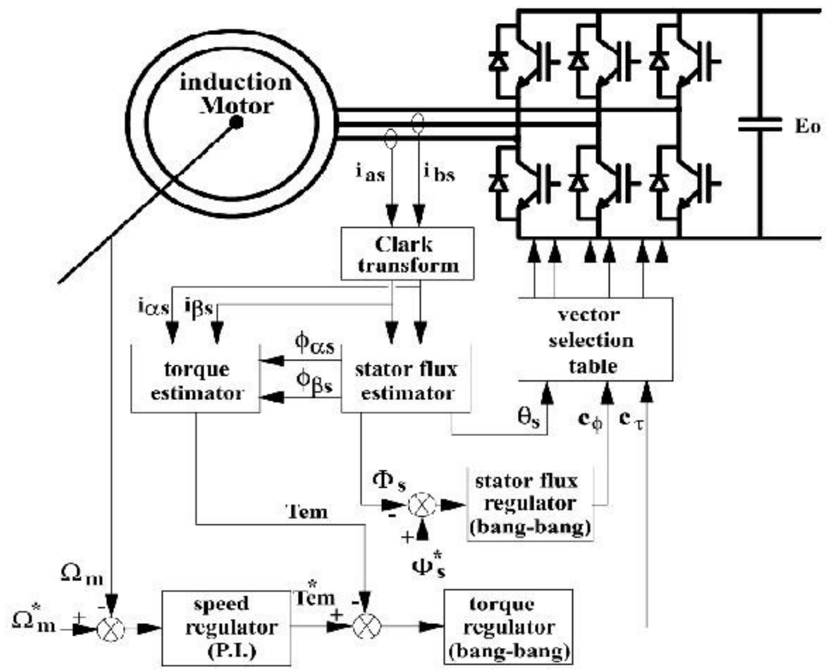
$$\begin{cases} \psi_s^s = L_s.I_s^s + L_m.I_r^s \\ \psi_r^s = L_r.I_r^s + L_m I_s^s \end{cases} \qquad (2)$$

•Mechanical equation

$$J\frac{d\omega_r}{dt} + f_v \omega_r = T_e - T_l \qquad (3)$$

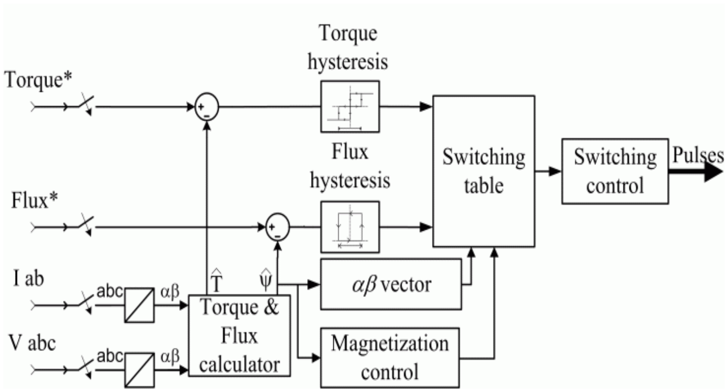
•Electromagnetic Torque equation

$$T_e = \frac{3p}{2}\frac{L_m}{L_s.L_r - L_m^2}\psi_s \otimes \psi_r = \psi_s \psi_r \sin\theta_{sr} \qquad (4)$$

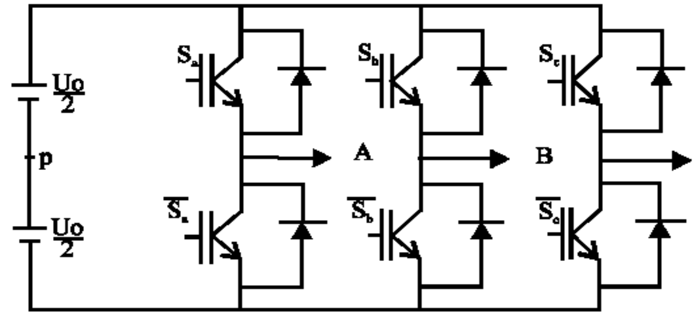


DTC black diagram using speed sensor

•Direct Torque Control (DTC) strategies of IM



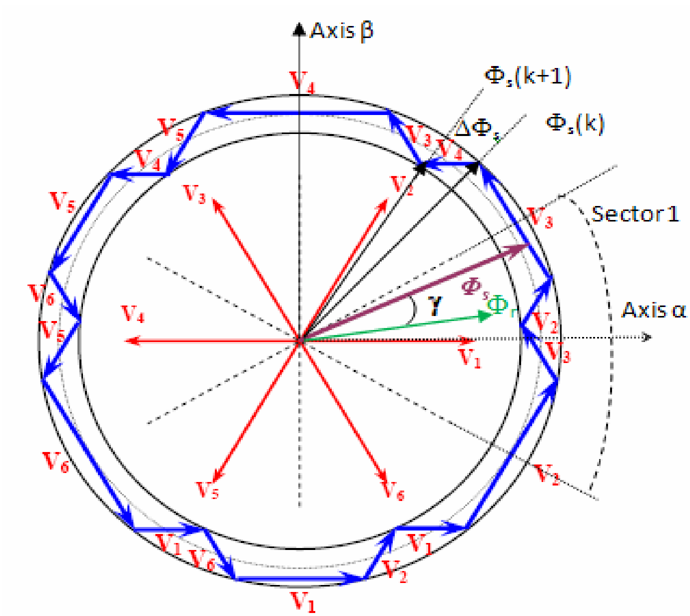
Flux and Torque hysteresis bands



Three phase PWM voltage source inverter

	Sectors (S_i: i = 1-6)						
τ_f	τ_T	S_1	S_2	S_3	S_4	S_5	S_6
1	1	V_2	V_3	V_4	V_5	V_6	V_1
	0	V_7	V_0	V_7	V_0	V_7	V_0
	-1	V_6	V_1	V_2	V_3	V_4	V_5
0	1	V_3	V_4	V_5	V_6	V_1	V_2
	0	V_0	V_7	V_0	V_7	V_0	V_7
	-1	V_5	V_6	V_1	V_2	V_3	V_4

Switching lookup table



Sector and voltage vectors

$$V_1 = 001 \quad V_2 = 010 \quad V_3 = 011$$
$$V_4 = 100 \quad V_5 = 101 \quad V_6 = 101$$
$$V_0 = 000 = V_7 = 111$$

1: Connected to positive source
0: Connected to negative source

$$\Delta\varphi_s \approx U_s \Delta t \qquad (5)$$

•State space model of IM

$$\begin{cases} \dot{x} = f(x,u) + w \\ y = Cx + v \end{cases} \quad (6)$$

$$x = [i_{s\alpha}, i_{s\beta}, \varphi_{r\alpha}, \varphi_{r\beta}]$$

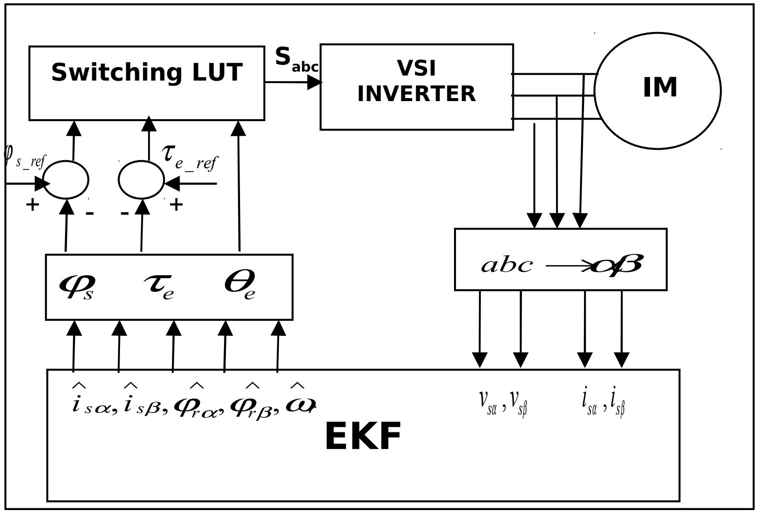
$$U = [V_{s\alpha}, V_{s\beta}]^T, \; y = [i_{s\alpha}, i_{s\beta}]^T$$

W: System disturbance, zero-mean Gaussian noises

V : Measurement disturbance, zero-mean Gaussian noises

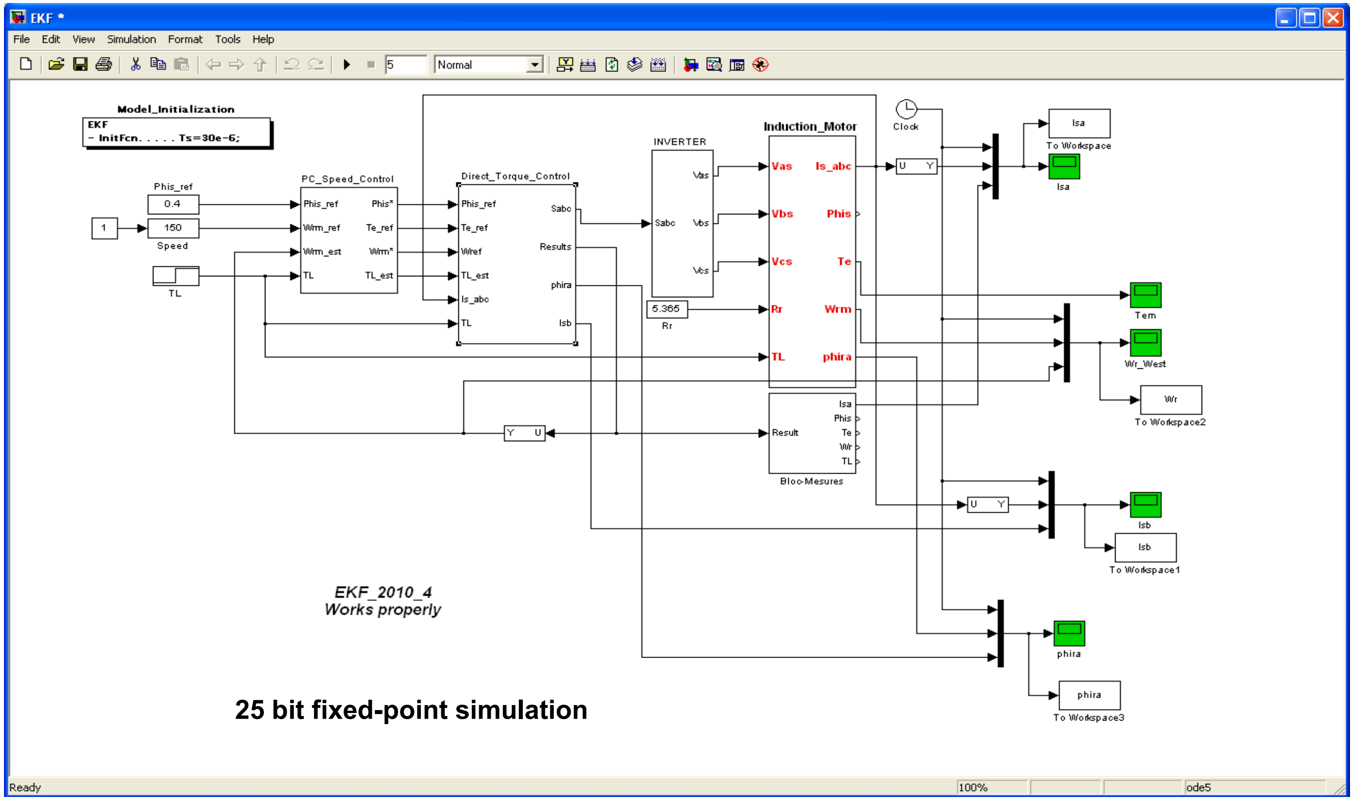
$$f(x,u) = \begin{bmatrix} (1 - T_s\gamma) i_{s\alpha} + T_s \frac{MR_r}{L_r^2 k}\varphi_{r\alpha} + T_s \frac{M\omega_r}{L_r k}\varphi_{r\beta} + T_s \frac{1}{k} V_{s\alpha} \\ (1 - T_s\gamma) i_{s\beta} - T_s \frac{M\omega_r}{L_r k}\varphi_{r\alpha} + T_s \frac{MR_r}{L_r^2 k}\varphi_{r\beta} + T_s \frac{1}{k} V_{s\beta} \\ T_s \frac{M}{T_r} i_{s\alpha} + (1 - \frac{T_s}{T_r})\varphi_{r\alpha} - T_s \omega_r \varphi_{r\beta} \\ T_s \frac{M}{T_r} i_{s\beta} + T_s \omega_r \varphi_{r\alpha} + (1 - \frac{T_s}{T_r})\varphi_{r\beta} \end{bmatrix}$$

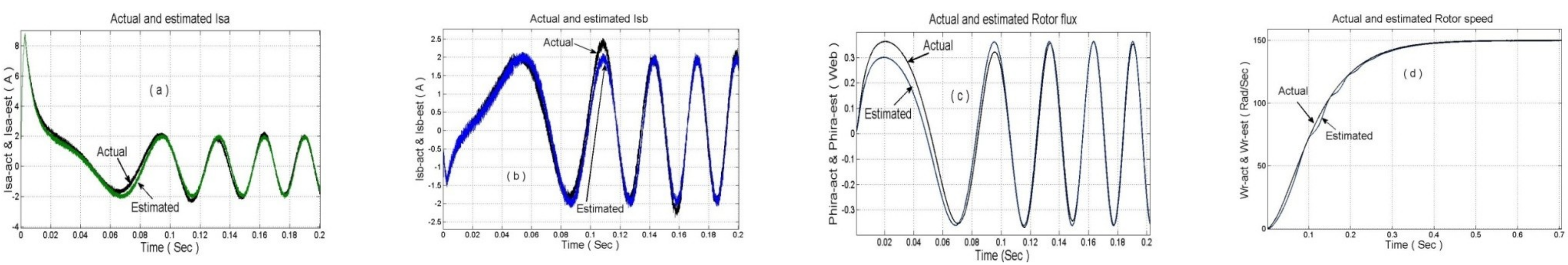
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, k = \frac{M}{\sigma L_s L_r}, \gamma = \frac{R_s}{\sigma L_s} + \frac{R_r M^2}{L_r^2 \sigma L_s}$$



Block diagram of EKF- based DTC control system

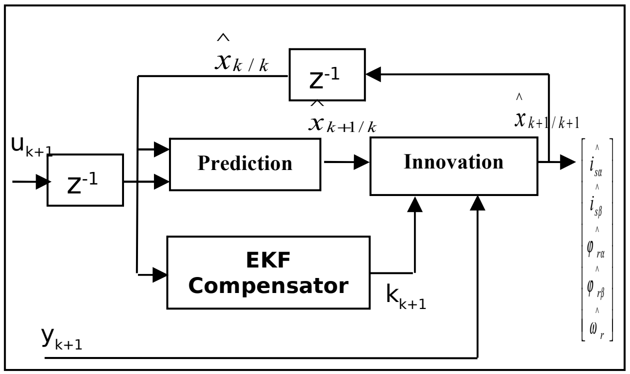
•DTC Simulation in MATLAB SIMULINK



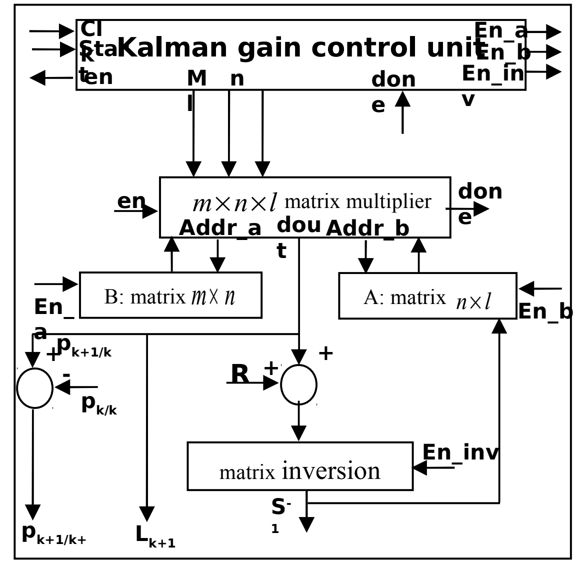


Actual and estimated state space variables: a) Isa, b) Isb, c) Phira, d) Wr

•Implementation of EKF algorithm in FPGA



Systolic of the EKF algorithm



FPGA- based architecture of the EKF compensator

1) Prediction state vector:

$$\hat{x}(k+1/k)=F(\hat{x}(k/k),U(k)) \qquad (7),\ Where: F(k)=\left.\frac{\partial f}{\partial x}\right|_{x(k)=\hat{x}(k/k)}$$

2) Prediction covariance computation:

$$p(k+1/k)=F(k)p(k/k)F(k)^T+Q \qquad (8)$$

Q: System covariance matrix characterizing W
p: State vector covariance matrix

3) Kalman gain computation:

$$L(k+1)=p(k+1/k)C(k)^T(C(k)P(k+1)C(k)^T+R)^{-1} \qquad (9)$$

R: Measurement covariance matrix characterizing V

4) State vector estimation or innovation step

$$\hat{x}(k+1/k+1)=\hat{x}(k+1/k)+L(k+1)(y(k+1)-C\hat{x}(k+1/k)) \qquad (10)$$

•Implementation of EKF algorithm in FPGA

Fixed point math packages：

Features:

1) It is a step between integer and floating point math
2) It is almost as fast as numeric_std arithmetic capable to represent fractional numbers
3) A fixed-point number has an assigned width and an assigned location for the decimal point so as the integer and fractional parts are recognizable
4) Because it is based on integer math, it is extremely efficient, as long as the data does not vary too much in magnitude

•Fixed point number representation and arithmatic operations:

y = 6.5 = “00110.10000”, 10 bits unsigned Fixed point representation of number 6.5
y = 6.5 = “000110.11000”, 11 bits signed Fixed point representation of number 6.75

Format in VHDL:

```
signal a, : ufixed (4 downto -5);
signal b, : sfixed (5 downto -5);
```

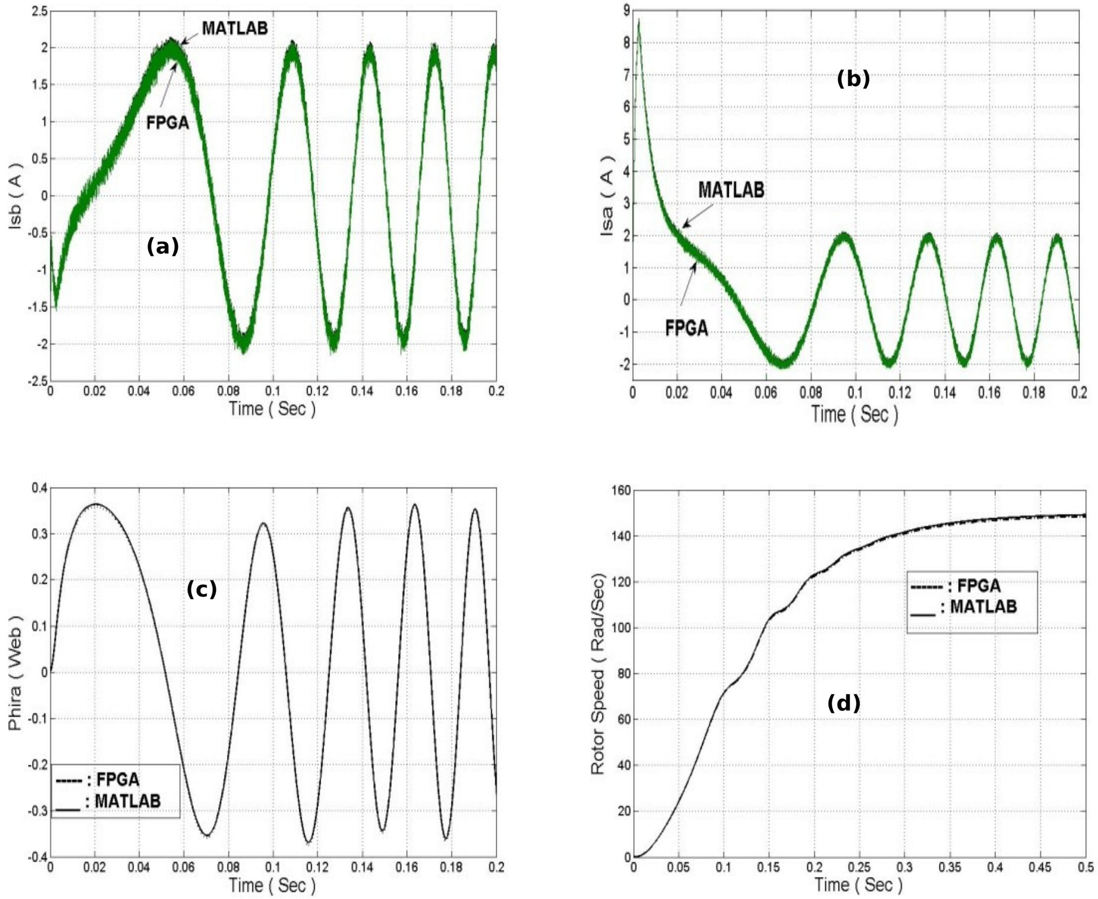
The most often used conversions and arithmetic operations :

```
a<= to_sfixed(6.5, a’high, a’low);
b<= to_sfixed(3.15, b’high,b’low);
m<= to_slv(a);
c<=(a+b, Max(a’high, b’high)+1 downto Min(a’low, b‘’low));
d<=(a-b , Max(a’high, b’high)+1 downto Min(a’low, b’low));
e<=(a*b , a’high+b’high+1 downto  a’low+b’low);
acc<=resize(e, acc’high,acc’low);
```

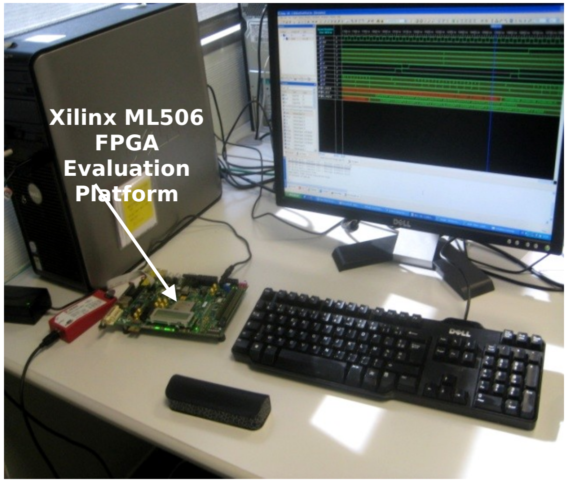
Usage model:

```
library ieee, ieee_proposed;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;

entity matrix is
   Port ( clk : in  STD_LOGIC;
          start : in  STD_LOGIC;
          Go : out  STD_LOGIC);
end matrix;

architecture arc_mat of matrix is
     component mult is
       port (  clk : in  STD_LOGIC;
          en : in  STD_LOGIC;
          dir : std_logic;
          Done : out  STD_LOGIC;
          Dout : out std_logic_vector (43 downto 0);
          kk0 : out std_logic_vector (2 downto 0);
          jj0 : out std_logic_vector (2 downto 0);
          ii0 : out std_logic_vector (2 downto 0);
          ii4 : out std_logic_vector (2 downto 0);
          jj4 : out std_logic_vector (2 downto 0);
          Din1 : in std_logic_vector (24 downto 0);
          Din2 : in std_logic_vector (24 downto 0);
          mm : in std_logic_vector(2 downto 0);
          nn : in std_logic_vector (2 downto 0);
          ll : in std_logic_vector (2 downto 0);
     end component;
 ---------------------------------------------------------------
subtype sfixed25 is sfixed( 10 downto -14);
subtype sfixed26 is sfixed( 10 downto -15);
subtype sfixed18 is sfixed( 10 downto -7);
.
.
.
```

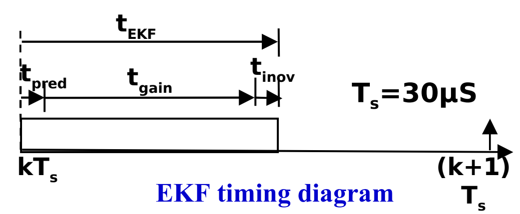
•Implementation of EKF algorithm in FPGA



HIL results, a) Isa, b) Isb, c) Phira, d) Wr



Hardware in Loop



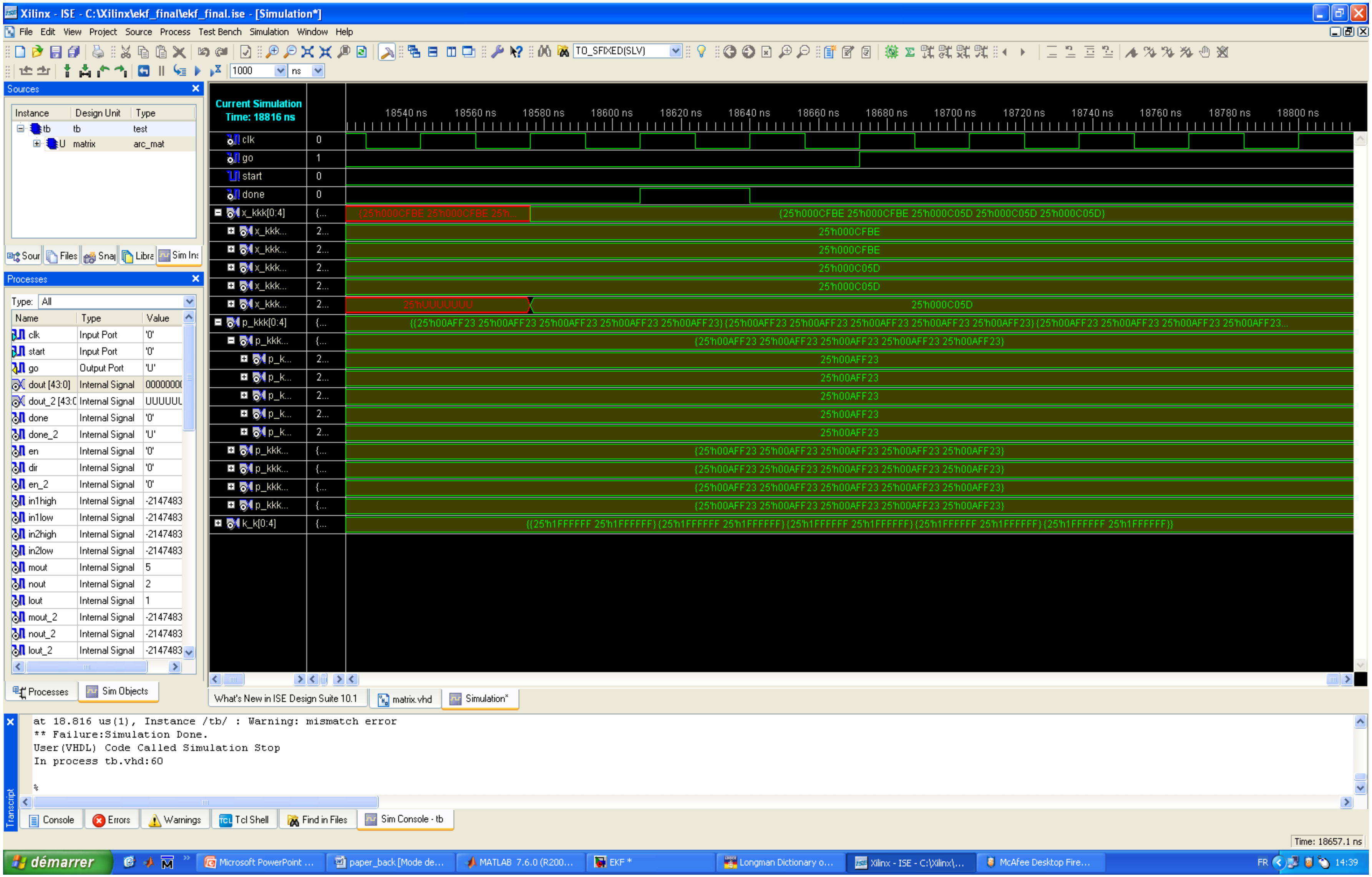
EKF timing diagram

t_{EKF}=18.68µ S t_{pred}=1.1µ S t_{inov}=1.61µ S

Table 1, FPGA resources usage

	Available	Used	Utilization
slice registers	32640	41	1%
Slice LUTs	32640	195	1%
Logics	32640	195	1%
25×18 Hw multipliers	136	9	7%

•Implementation of EKF algorithm in FPGA



Post-Rout simulation of EKF: $X_{k+1/k+1}$, P_{k+1} , K_{k+1}

•Conclusions

•The design and implementation of the FPGA-based Extended Kalman Filter (EKF) estimator was presented.

•The implemented EKF algorithm as a part of DTC control system was described focusing on the FPGA architecture.

•Simulation and hardware in loop (HIL) results were provided in order to validate the performance and effectiveness of the developed design and the time/area analysis was then presented.

•An examination of time/ area performance of FPGA architecture reveals that the whole DTC controller is implementable using one single stand alone Xilinx ML506 FPGA Platform.

•The total recourses used for implementing EKF algorithm is only 10% of available FPGA hardware.
The rest of the resource is quite enough to implement all low speed adders or multipliers required
for designing the remaining parts of complete DTC controller.

Thank you