FPGA BASED REAL TIME SIMULATION FOR ELECTRICAL MACHINES

Y. J. Zhou and T. X. Mei*

*School of Electronic and Electrical Engineering, The University of Leeds, Leeds, LS2 9JT, UK, *Email:- t.x.mei@leeds.ac.uk*

Abstract: This paper presents a new approach for the real-time simulation of electrical motors based on a single FPGA chip. The study is aimed at not only developing a high performance and flexible platform for testing and rapid prototyping of motor control systems, but also providing a low-cost replacement supplement of large and expensive electrical machines for education purposes. A non-linear model is introduced, which represent a typical motor with a rated power of 70kW. The real time simulation is based on the well-known Runge-Kutta solver, but it is revised and optimised for PFGA implementations. Other implementation issues are also studied, including the conversion from floating point to fixed point data processing and development of appropriate processing architecture.

Keyword: Control, Real-time, Simulation, Electrical motors, FPGA

1. INTRODUCTION

Electrical machines are found in many different industrial and domestic applications, and the level of control for those machines varies significantly and is mainly dependent upon specific requirements and conditions the machines are designed to work with. High performance drive systems often involves complex algorithms and control structures, especially where the load characteristics are complex and uncertain. In the last two-three decades, design techniques for the development of motor control systems have made huge advances and computer simulations are widely used particularly at early stages of a development/design cycle to establish an essential understanding of the problems and to find appropriate solutions. However, there is a still a desire and real need for effective testing tools to commission real time control systems in full operation conditions off-line in order to minimise the time and cost. A number of hardware-in-loop systems are available and can be used for the purpose, but those tend to be expensive and require great computing power to ensure the execution of simulation algorithms in real time. In one such a development, three Texas TMS320C40 DSPs were used in order to achieve desired real time performances (Jack et al, 1998). The complexity of the models has been identified as the main cause of the high demand, but the nature of the largely sequential processing of software algorithms also contributes significantly to the problem.

Recently, there is a growing trend to use the system-on-chip technology of the implementation of large and complex processing tasks on a single chip (hardware) enabled by the much greater and increasing capacities of the latest FPGA chips. There are obvious advantages such as simplicity and low costs, because there is no longer the need to rely on traditional CPUs and associated peripherals. In the area of motor control systems, the new approach has been used to realise fairly complex control algorithms (Kim et al, 2001; and Ricci and Le-Huy,

2002), and to process signals from encoder measurements (Tsai and Chen, 2002).

This paper studies a novel solution for the real time simulation of electrical motors, based on the latest FPGA devices. The main aim of the study is to provide a flexible platform for rapid development, testing and commissioning of real time control algorithms and functionalities, which will be suitable and adaptable for use at any stage of the development of control systems. There is also an added benefit that such a system can be readily used in higher education to replace/supplement large and expensive electrical machines. Although real time simulations based on DSPs and/or microprocessors have been developed for many applications before, this study will be concerned with the FPGA-only approach by exploring a number of advantages offered by the new technology. Some findings of the research such as the selection of word-length and step size have been reported (Mei and Zhou, 2004), and this paper is to focus on detailed implementations. A revised Runge-Kutta (or R-K) solver will be introduced, which is particularly suited for parallel data processing and hence FPGA implementations. Simulation algorithms and the development of appropriate processing architecture are also presented.

2. SIMULATION MODEL

A model of a medium-size dc motor as defined in equations 1-4 is used in the implementation (Mei and Zhou, 2004). Equation 1-3 represent models of the armature winding, the rotor dynamics and the field winding respectively. Equation 4 defines the electrical and mechanical constants of the motor, which are a function of the motor flux. There are three inputs to the motor – V_a, V_f and T_L The armature voltage (V_a) is normally regulated by an external controller to achieve pre-defined performance requirements. The field voltage (V_f) is controlled independently in a separately-excited machine, but changes are normally slow and only needed at high speeds where field weakening becomes necessary. The load T_L may vary significantly from application to application. It can be largely static in some cases and therefore straight forward to model. However in many high performance drives, the load characteristics are very complex and sometimes nonlinear and therefore it would be necessary to include the models for the system which the motor is used to drive. The number of output required from the model will be dependent upon those demanded by the controllers, but the armature current, field current, rotor speed and angular displacement are likely outputs required for a feedback control.

$$v_a = L_a \cdot \dot{i}_a + R_a \cdot i_a + K_e \cdot \omega \tag{1}$$

$$J \cdot \dot{\omega} = K_t \cdot i_a - F_m \cdot \omega - T_L \tag{2}$$

$$v_f = L_f \cdot \dot{i}_f + R_f \cdot i_f \tag{3}$$

$$K_e = K_t = K_{const} \cdot \lambda \tag{4}$$

The non-linearity due to the saturation of motor flux is a common property for many electrical machines, and it must be considered in the model. Figure 1 shows how the flux and the inductance of the field winding vary nonlinearly with the field current, which is included in the simulation in the form of look-up tables.

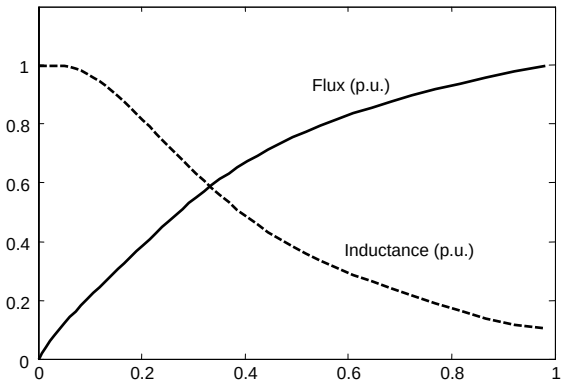

Figure 1. Non-linear characteristics of the field flux and (small signal) inductance

The model represents a motor with a power rating of $70kW$. The rated armature voltage and current are $700V$ and $100A$, and rated field current is $60A$. All relevant motor parameters are defined in the appendix.

3. REVISED ALGORITHMS

The selection of a solver for the differential equations is a trade-off between the accuracy requirement and algorithm complexity. Amongst a number of different options (Davis, 1992), a fourth-order R-K solver is chosen and shown to achieve sufficient accuracy, and will also be feasible for a hardware implementation. The standard R-K solver is shown in Equation 5.

$$\left.\begin{aligned}
\mathbf{K}_1 &= h \cdot \mathbf{F}(t_n, \mathbf{x}_n) \\
\mathbf{K}_2 &= h \cdot \mathbf{F}(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{1}{2}\mathbf{K}_1) \\
\mathbf{K}_3 &= h \cdot \mathbf{F}(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{1}{2}\mathbf{K}_2) \\
\mathbf{K}_4 &= h \cdot \mathbf{F}(t_n + h, \mathbf{x}_n + \mathbf{K}_3) \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4)
\end{aligned}\right\} \tag{5}$$

For the motor model:

$$\mathbf{x}_n = \begin{bmatrix} i_n & i_{fn} & \omega_n \end{bmatrix}^T$$

$$\mathbf{F}(t,\mathbf{x}_n)=\begin{bmatrix} \dot{i} & \dot{i}_f & \dot{\omega} \end{bmatrix}^T$$

$$\mathbf{K}_j=\begin{bmatrix} k_j & m_j & p_j \end{bmatrix}^T, \qquad j=1,2,3,4$$

When implemented directly, the standard R-K solver takes five consecutive steps in each iteration to compute and update the state variables and there are also some computing repetitions. On the other hand, one of the key features of FPGA devices is the capability of parallel processing. The standard R-K algorithms are obviously not suited to take advantage of that feature, and a revised version is derived as shown below. Without the loss of generality, only two of the three differential equations of the model (equation 1 and 2) are used to demonstrated the problems and explain the proposed revision. Equation 3 will be dealt with separately later in section 5, as it is decoupled from equations 1 and 2.

Substituting $\mathbf{x}_n$, $F(t,\mathbf{x}_n)$ and K_j into equations 5 produces the expanded polynomials of k_j, m_j (j=1, 2, 3, 4) and also i_{n+1}, ω_{n+1}. The five-step computations in each iteration required to obtain the state variables in every iteration are illustrated in Figure 2.

$$k_1 = C_1 v_1 + C_2 i_n + C_3 \omega_n \tag{6}$$

$$k_2 = C_1 v_2 + C_2 i_n + C_3 \omega_n + C_4 k_1 + C_5 m_1 \tag{7}$$

$$k_3 = C_1 v_2 + C_2 i_n + C_3 \omega_n + C_4 k_2 + C_5 m_2 \tag{8}$$

$$k_4 = C_1 v_3 + C_2 i_n + C_3 \omega_n + 2C_4 k_3 + 2C_5 m_3 \tag{9}$$

$$m_1 = C_6 T_{l1} + C_7 i_n + C_8 \omega_n \tag{10}$$

$$m_2 = C_6 T_{l2} + C_7 i_n + C_8 \omega_n + C_9 k_1 + C_{10} m_1 \tag{11}$$

$$m_3 = C_6 T_{l2} + C_7 i_n + C_8 \omega_n + C_9 k_2 + C_{10} m_2 \tag{12}$$

$$m_4 = C_6 T_{l3} + C_7 i_n + C_8 \omega_n + 2C_9 k_3 + 2C_{10} m_3 \tag{13}$$

$$i_{n+1} = i_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{14}$$

$$\omega_{n+1} = \omega_n + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4) \tag{15}$$

where

$$v_1 = v_a(t_n) \qquad T_{l1} = T_L(t_n)$$

$$v_2 = v_a(t_n + h/2) \qquad T_{l2} = T_L(t_n + h/2)$$

$$v_3 = v_a(t_{n+1}) \qquad T_{l3} = T_L(t_{n+1})$$

$$C_1 = \frac{h}{L_a},\ C_2 = -\frac{h \cdot R_a}{L_a},\ C_3 = -\frac{h \cdot K_e}{L_a},$$

$$C_4 = -\frac{h \cdot R_a}{2 \cdot L_a},\ C_5 = -\frac{h \cdot K_e}{2 \cdot L_a}$$

$$C_6 = -\frac{h}{J},\ C_7 = \frac{h \cdot K_t}{J},\ C_8 = -\frac{h \cdot F_m}{J},$$

$$C_9 = \frac{h \cdot K_t}{2 \cdot J},\ C_{10} = -\frac{h \cdot F_m}{2 \cdot J}$$

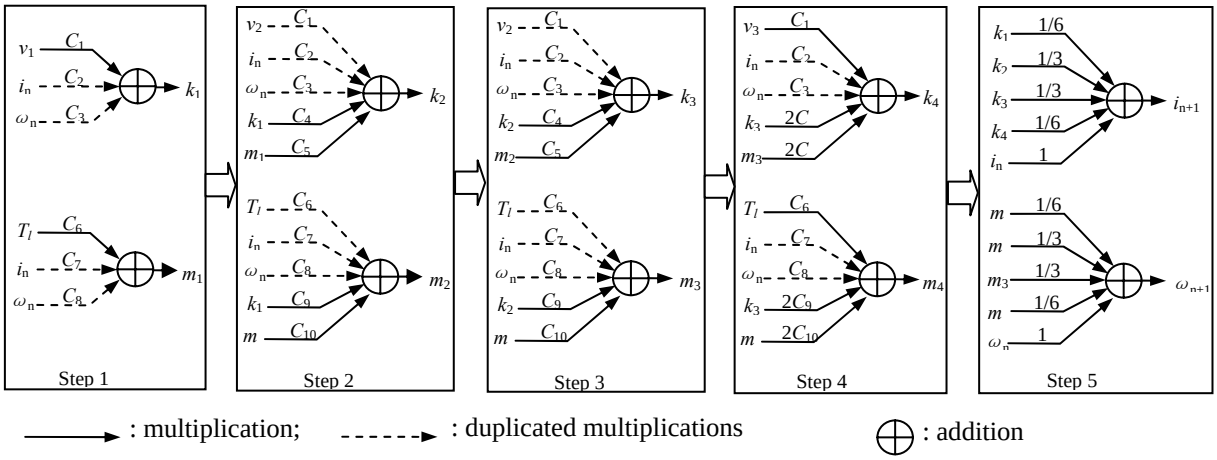

Figure 2. Computation steps for standard R-K solver

This would be a highly inefficient use of FPGA resources and can be improved as follows. Equation 7 (for computing k_2) can be updated by eliminating k_1 and m_1 using equations 6 and 10, which gives:

$$k_2 = D_1 v_1 + D_2 v_2 + D_3 T_{l1} + D_4 i_n + D_5 \omega_n \tag{16}$$

where D_j (j=1, 2, 3, 4, 5) are constants derived from C_j (j=1, 2, 3, …, 8). Expressions for k_3, k_4, m_2, m_3, and m_4 can be similarly obtained. Consequently all k_j and m_j (j=1, 2, 3, 4) in equations (14) and (15) can be eliminated as shown in equations 17 and 18, where all coefficients E_j (j=1,2, ..,14) are determined by the values of C_j (j=1,2,..,10).

$$i_{n+1} = E_1 \cdot v_1 + E_2 \cdot v_2 + E_3 \cdot v_3 + E_4 \cdot T_{l1} + E_5 \cdot T_{l2} + E_6 \cdot i_n + E_7 \cdot \omega_n \tag{17}$$

$$\omega_{n+1} = E_8 \cdot v_1 + E_9 \cdot v_2 + E_{10} \cdot T_{l1} + E_{11} \cdot T_{l2} + E_{12} \cdot T_{l3} + E_{13} \cdot i_n + E_{14} \cdot \omega_n \tag{18}$$

The computation of all the terms in equations 17 and 18 can be carried in parallel as shown in Figure 3. There is no extra space capacity required and there is in fact a small reduction in the number of logic elements used (from 1385 to 1271). However the computation time can be significantly reduced (from 498.5ns to 134.80 ns). The comparisons are made using the report generated by a FPGA compiler, where 1) programming software–Quartus II 4.0 Web Edition; 2) language–VHDL; 3) chipset–Altera FLEX10K EPF10K70RC240-4; and 4) precision of input signals of 14 bits are used.

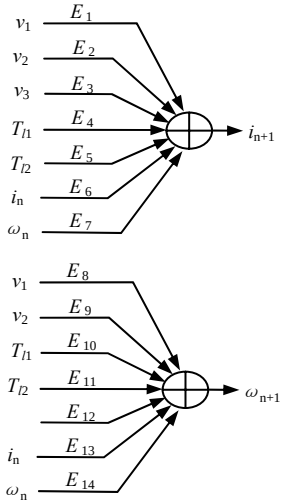

Figure 3. Computations for the revised algorithms

4. WORD-LENGTH AND SCALING FACTORS

Fixed-point processing is used to implement the revised algorithms. Simulation accuracy is ensured by selecting appropriate word-lengths (up to 33 bits) for the coefficients and variables. Unlike the standard processors, there is a much greater flexibility in the use of different word-lengths for different parameters and hence use of logic elements is minimised (Mei and Zhou, 2004).

For given word lengths, the revised simulation algorithms are converted into integer format before the implementation. Equations 17 and 18 can be expressed in the forms as shown in equations 19 and 20, where all terms in brackets are rounded to the nearest integer numbers.

$$\begin{aligned} i_{n+1}\cdot S_i &= (E_1\cdot S_1)\cdot(v_1\cdot S_v)+(E_2\cdot S_2)\cdot(v_2\cdot S_v)\\ &+(E_3\cdot S_3)\cdot(v_3\cdot S_v)+(E_4\cdot S_4)\cdot(T_{l1}\cdot S_T)\\ &+(E_5\cdot S_5)\cdot(T_{l2}\cdot S_T)+(E_6\cdot S_6)\cdot(i_n\cdot S_i\cdot S_{i\_b\_1})\\ &+(E_7\cdot S_7)\cdot(\omega_n\cdot S_\omega\cdot S_{\omega\_b\_1}) \end{aligned} \quad (19)$$

$$\begin{aligned} \omega_{n+1}\cdot S_\omega &= (E_8\cdot S_8)\cdot(v_1\cdot S_v)+(E_9\cdot S_9)\cdot(v_2\cdot S_v)\\ &+(E_{10}\cdot S_{10})\cdot(T_{l1}\cdot S_T)+(E_{11}\cdot S_{11})\cdot(T_{l2}\cdot S_T)\\ &+(E_{12}\cdot S_{12})\cdot(T_{l3}\cdot S_T)+(E_{13}\cdot S_{13})\cdot(i_n\cdot S_i\cdot S_{i\_b\_2})\\ &+(E_{14}\cdot S_{14})\cdot(\omega_n\cdot S_\omega\cdot S_{\omega\_b\_2}) \end{aligned} \quad (20)$$

All $S_{subcript}$ are scaling factors used to maximise data precisions, and following relationships must stand.

$$\begin{aligned} S_i &= S_1\cdot S_v = S_2\cdot S_v = S_3\cdot S_v = S_4\cdot S_T = S_5\cdot S_T\\ &= S_6\cdot S_i\cdot S_{i\_b\_1} = S_7\cdot S_\omega\cdot S_{\omega\_b\_1} \end{aligned} \quad (21)$$

$$\begin{aligned} S_\omega &= S_8\cdot S_v = S_9\cdot S_v = S_{10}\cdot S_v = S_{11}\cdot S_T = S_{12}\cdot S_T\\ &= S_{13}\cdot S_i\cdot S_{i\_b\_2} = S_{14}\cdot S_\omega\cdot S_{\omega\_b\_2} \end{aligned} \quad (22)$$

The coefficients are scaled (using S_1, S_2, … S_{14}) such that all dividers are chosen to be the power of 2. This offers the obvious advantage of not needing the use of complex divisions, as they are replaced by the right shifts executable in a single clock cycle.

The scaling factors for the input variables are determined according to the resolution of the A/D converters and range of the input signals, whereas those for the state variables can be chosen in terms of the allocated word-lengths (WL in bits) and the ranges of the parameters, e.g.

$$S_v = 2^{Res\_v}/v_{\text{range}} \quad (23)$$

$$S_i = 2^{WL\_i}/i_{\text{range}} \quad (24)$$

Furthermore, different scaling factors for the state variables at current and next time steps are needed, as an use of same scaling factors affects the scaling of some coefficients (e.g. E_6, E_7, E_{13} and E_{14}) and leads to a significant loss of data precision. Therefore additional scaling factors $S_{i\_b\_1}$, $S_{i\_b\_2}$, $S_{\omega\_b\_1}$, and $S_{\omega\_b\_2}$ are used for re-scaling at each iteration.

Computations of equations 19 and 20 on a FPGA including considerations for scaling and re-scaling are illustrated in Figure 4, where the superscript (′) represents scaled variables and coefficients.

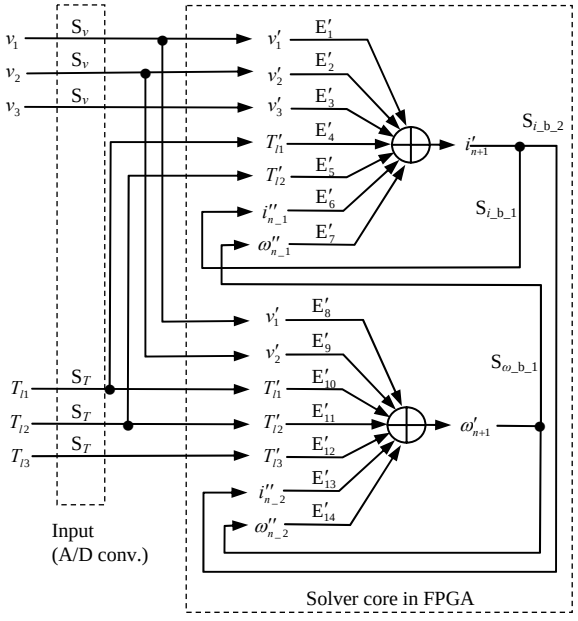

Figure 4. FPGA computations

5. FIELD CUURENT AND NON-LINEAR PROPERTIES

As shown in Figure 3, the generation of field current is only dependent on parameters of the field winding and the voltage applied to the winding, and therefore can be solved independently. A solution for equation 3 is derived using the procedure presented in section 3 which is then applied with appropriate scaling to convert it into integer format as given in equation 25.

$$i_{f,n+1} \cdot S_f = (E_{f1}S_{f1}) \cdot (v_{f1}S_{fv}) + (E_{f2}S_{f2}) \cdot (v_{f2}S_{fv}) + (E_{f3}S_{f3}) \cdot (v_{f3}S_{fv}) + (E_{f4}S_{f4})(i_n S_f \cdot S_{f\_b}) \quad (25)$$

where

$$v_{f1} = v_f(t_n), \quad v_{f2} = v_f(t_n + h/2), \quad v_{f3} = v_f(t_{n+1})$$

The field current produces the necessary flux in the motors, which directly affects the generation of the armature current (via the back emf, or the electrical time constant K_e) and that of the motor torque (via the mechanical time constant K_t, and indirectly via the armature current). This is further complicated by the non-linear relationships between the field current and the flux due to flux saturations which is common in many electrical machines.

Therefore the entire simulation algorithms at each iteration are tackled as follows. Equation 25 is solved first to update the value of i_f which is then used to find corresponding flux level, coefficients E_{f1}, E_{f2}, E_{f3}, E_{f4} and values of K_e and K_t using look up tables defined by the non-linear relationships as shown in Figure 1. Finally, equations 19 and 20 are solved to update the values of the armature current and rotor speed.

6. HARDWARE IMPLEMENTATION

The real time simulation system is developed based on a FPGA chip (Altera FLEX 10K®EPF10K70), and a block diagram of the hardware design is shown in Figure 5. In addition to the FPGA chip, some peripherals are necessary to communicate with external sources. A/D converters are used to input the voltages applied to the motor model, whereas DC converters are used to output simulation results such as the stator current, the speed and angular position of the rotor.

On the FPGA chip, the core element is for the implementation of the revised R-K differential equation solver, but a number of other features are also included. A relatively small part of the FPGA is assigned for the control of A/D and D/A converters, which will be executed separately from the main part of the algorithms. The design also allows for the implementation of models to simulate the dynamics and other key features of sensors. This will be useful for testing motor controllers that require those measurements, but without the need for real sensors. The internal design of the FPGA is carried out using a combination of VHDL and schematic diagram methods. Loop up tables are stored in the data storage area on the FPGA chip.

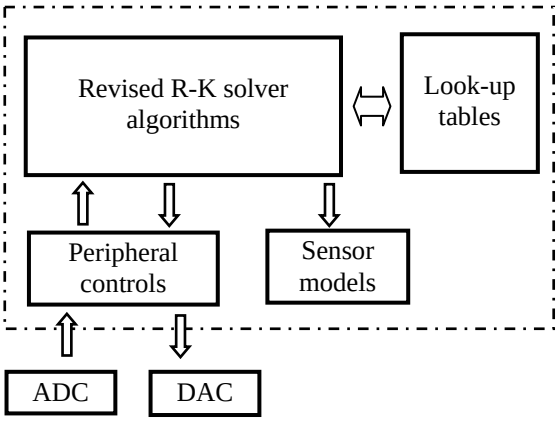

Figure 5. Block diagram

7. HARDWARE IMPLEMENTATION

Figures 6 and 7 give the experimental results of the armature current and the motor speed, where a superimposed triangular voltage is applied to the armature winding. It can be seen that, when the input voltage is increased or decreased at a constant rate, a constant current (after a transition) and hence a constant torque is produced to provide the constant acceleration/deceleration of the motor speed. The transition in the armature current is caused by the armature time constant. The time constant is small because of the relatively large inductance.

In Figure 8 where a sinusoidal voltage is superimposed, there is a clear phase difference which is determined largely the mechanical characteristics and the load of the motor.

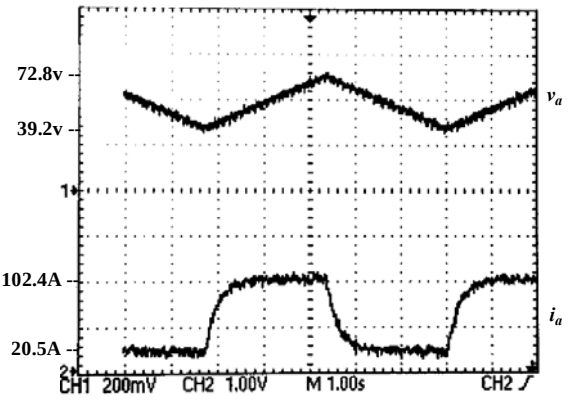

Figure 6. Experiment Results, superimposed triangular voltage (armature voltage and current)

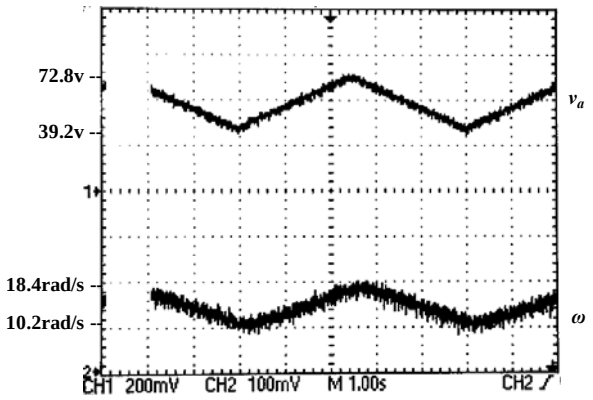

Figure 7. Experiment Results, superimposed triangular voltage (armature voltage and speed)

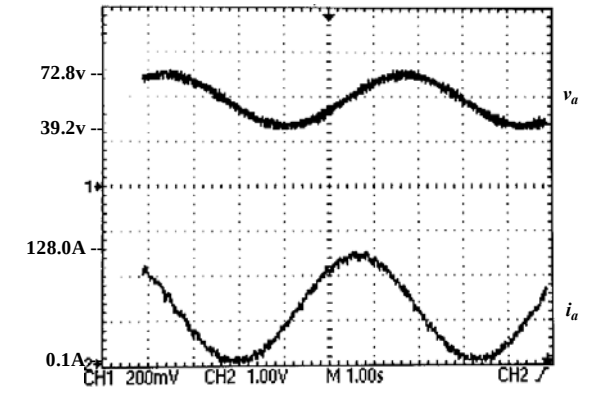

Figure 8. Experiment Results, superimposed sinusoidal voltage (armature voltage and current)

8. CONCLUSIONS

This paper has presented the development of a real time simulation system for electrical motors based on a single FPGA chip, without the use of conventional microprocessors or DSPs. A non-linear motor model has been introduced. A fourth order Runge-Kutta method is selected as the differential equation solver, but revised algorithms have been derived which converts a 5- step computation per iteration into a single step one in order to take advantages of parallel processing capability of a FPGA device. Scaling techniques for fixed-point calculations have also been shown to be effective.

The system development based on a FPGA chip (Altera FLEX10K EPF10K70RC240-4) has indicated that the simulation of the motor can be achieved in real time with a maximum error around 0.1% or less, which is acceptable in most practical applications. A total of 2462 logic elements are required and the run time per iteration is less than 195 ns.

REFERENCE

Jack, A G; Atkinson, D J and Slater H J. (1998) "Real-time emulation for power equipment development. Part 1: Real-time simulation", *IEE Proceedings – Electric Power Applications*, v145, No 2, pp. 92-97.

Kim, S J; Lee, H J; Kim, S K and Kwon Y A. (2001) "ASIC design for DTC based speed control of induction motor", *IEEE ISIE2001*, pp 956 – 961.

Mei, T.X. and Zhou, Y.J. (2004), "Real Time Simulation of Electrical Motors using System-on-chip Approach", UKACC Control 2004, Bath, UK.

Paul W. Davis, (1992) "Differential Equations for Mathematics, Science, and Engineering", Prentice Hall, Englewood Cliffs, New Jersey.

Ricci, F and Le-Huy, H (2002) "An FPGA based rapid prototyping platform for variable-speed drives", *IEEE IECON'02*, pp. 1156-1161.

Tsai, M F and Chen C P (2002) "Design of a quadrature decoder/counter interface IC for motor control using CPLD", *IEEE IECON'02*, pp. 1936-1941.

APPENDIX. SYMBOLS AND PARAMETERS

$\mathbf{x}$	State vector
$\mathbf{x}_n$, $\mathbf{x}_{n+1}$	State variables at discrete times
h	step-size
i, i_a	Armature current
i_n i_{n+1}	Armature current at discrete times
i_f	Field current
i_{fn}, $i_{f,n+1}$	Field current at discrete times
J	Moment of inertia of the rotor
K_{const}	Motor constant
K_e	Motor electrical constant
K_t	Motor mechanical constant
L_a	Armature inductance
L_f	Field inductance
R_a	Armature resistance
R_f	Field resistance
T_m	Motor torque
T_L	Motor load
v, v_a	Armature voltage
v_f	Field voltage
λ	Motor flux
ω	Motor speed
ω_n, ω_{n+1}	Motor speed at discrete times