Digital Hardware Implementation of a Current Controller for IM Variable-Speed Drives

Soren J. Henriksen, Robert E. Betz, *Member, IEEE*, and Brian J. Cook

Abstract— This paper presents the design of an induction machine current controller that is entirely implemented in digital hardware. A hardware current controller allows high switching frequencies with only modest processing power, as well as simplified controller hardware and software. The paper briefly presents the concepts of the algorithm implemented, and then outlines the changes that are made to make the digital implementation even more efficient. It then discusses the architecture used for the hardware design. Experimental results are presented to demonstrate the algorithm's performance.

Index Terms— Current control, induction machine control, pulsewidth modulation.

I. INTRODUCTION

THE current control loop is the most fundamental and important control loop for any variable-speed drive system. If the current loop has a very fast and accurate transient response, then, with appropriate reference currents, the machine can be regarded as an ideal torque source as far as the outer loops are concerned. Consequently, there has been considerable research carried out to improve the performance of this loop. The traditional approach for current control was to use a proportional integral (PI) regulator to manipulate the inverter voltage via a voltage space-vector algorithm to achieve some desired current. This approach has a number of disadvantages: the transient performance of the loop is poor, the controller tuning is parameter dependent, and it has to operate in a rotating reference frame to prevent steady-state errors.

The limitations of the PI controller have motivated research into alternative current control techniques. Of these, the two most promising candidates are hysteresis controllers and predictive controllers [1], [2]. Hysteresis controllers offer very high bandwidths, since they belong to the bang-bang control family. However, in their traditional form, they do not control the switching frequency and, under some circumstances, it can reach unacceptably high values. Improvements made to the basic hysteresis idea have overcome these problems at the expense of greater complexity in the controller. Furthermore,

the solutions proposed are inherently analog in nature [1], which is undesirable in a power electronics environment.

Predictive controllers have also been extensively investigated in the literature [2]–[4]. These algorithms are suitable for digital implementation and offer high bandwidth due to their deadbeat nature. However, unlike hysteresis controllers, the algorithm requires some knowledge of model parameters and estimation of some model states. To date, most of the work on predictive controllers has been directed at software implementation. Existing work has produced successful results in predictive current control in both simulation [4] and software implementation [3]. In these implementations, good current control has been achieved, while maintaining a constant switching frequency.

While there have been several published digital hardware implementations of current controllers for switched reluctance machines [5], [6], completely digital hardware current controllers for three-phase inverter-driven induction machines (IM's) have not appeared in the open literature, although a proprietary digital controller chip has been produced by the European VCON consortium. A single-chip hardware implementation, as presented in this paper, has the following potential benefits over a software digital implementation:

- 1) reduced system component count;
- 2) reduced software investment—the current controller and associated functionality is now a single chip, with no initial programming and no software maintenance required;
- 3) current controller can be self-commissioning for all types of drives (dependent on the algorithm);
- 4) high switching frequencies can be obtained—the switching devices are the main switching frequency limitation and not the microprocessor speed; the prototype controller was designed to accommodate a switching frequency up to 20 kHz.

The remainder of this paper is organized as follows. The current control algorithm to be implemented is outlined. Alterations to the basic algorithm to make it easier to implement in hardware are then presented. The hardware architecture is then described and, finally, the experimental system and results are discussed.

II. THE ALGORITHM

A very detailed discussion of the algorithm to be implemented appears in [8] and [9]. However, in order to understand the hardware implementation, it is necessary to appreciate the salient points of this algorithm and, therefore, the main concepts will be briefly presented.

Paper IPCSD 99–15, presented at the 1998 Industry Applications Society Annual Meeting, St. Louis, MO, October 12–16, and approved for publication in the IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS by the Industrial Drives Committee of the IEEE Industry Applications Society. Manuscript released for publication February 15, 1999.

The authors are with the Department of Electrical and Computer Engineering, University of Newcastle, Callaghan, NSW 2308, Australia (e-mail: eesjh@ee.newcastle.edu.au; reb@ee.newcastle.edu.au; brian@ee.newcastle.edu.au).

Publisher Item Identifier S 0093-9994(99)05359-1.

Fig. 1. The double-edge switching pattern used and the transformed two-phase voltages for sector 1.

The algorithm to be implemented is a fairly standard predictive control algorithm, similar to others published in the literature [3], [4]. The novel feature of [8] is the fact that the whole algorithm is implemented in a stationary dq reference frame, from required switching duty cycle determination through to firing signals for the inverter. This leads to an elegant and simple implementation, and makes the algorithm amenable to digital hardware implementation. The IM model assumed consists of a leakage inductance in series with the back EMF [7] in each of the two axes.

The control expressions for the algorithm are in terms of duty cycles of hypothetical switching waveforms for the d and q axes. Using the same notation as [8], the control expression for each axis is

$$\alpha[k, k+1] = \frac{1}{V} \left\{ \frac{2L}{T} (i_{\text{des}}[k, k+1] - i[k]) + e[k] \right\} \quad (1)$$

where

$$\begin{aligned} i[k] &= 2i[k-0.5] - i[k-1] \\ e[k] &= 2e[k-1] - e[k-2] \end{aligned}$$

and $\alpha[k, k+1]$ is the required duty cycle of the switching waveform for the respective axis to achieve a desired average current $i_{\text{des}}[k, k+1]$ over the control interval from k to $k+1$. Note that α has values $-1 \leq \alpha \leq 1$. $e[k]$ and $i[k]$ are the back-EMF voltage and current at the beginning of the control interval, respectively. The e 's can be estimated by rearranging (1) to make e the subject of the expression.

Remark 1: Note that the above expression is specific to an IM with a leakage inductance of L .

Remark 2: Voltage constraints are also taken into account by the algorithm. These will be examined in a following section.

Using the technique shown in [8], one can convert the α 's into the switching times of the three vectors for a switching sector of a three-phase inverter. An example switching pattern is shown in Fig. 1. Using standard notation for pulsewidth modulation (PWM) switching vectors, one can obtain the switching times shown in Table I in terms of the α 's [8].

The other feature of the algorithm is the parameter estimation aspects. As noted previously, the back EMF's are estimated using past measurements of the currents. The main

machine parameter to estimate is the leakage inductance. In order to estimate this, an independent expression is developed, involving measurements of the currents at the quarter and midpoints of a control interval. If the three phase values are mapped back to the d axis [see d in Fig. 1] for the case of $t_1 > t_2$, we get an expression of the form

$$L_{\text{est}} = \frac{t_2 V}{2(\Delta i_1 - \Delta i_2)} \quad (2)$$

where V is the magnitude of the applied voltage space vector, and

$$\begin{aligned} \Delta i_1 &= \text{the change of current over } [i-1, i-0.75] \\ \Delta i_2 &= \text{the change of current over } [i-0.75, i-0.5]. \end{aligned}$$

A similar expression can also be obtained for the $t_1 < t_2$ case. If $t_1 = t_2$, this technique cannot be used, as the denominator of the L_{est} expression is zero.

Since the current is no longer being sampled at the zero vector times, and expression (2) involves derivatives, it is clear that L_{est} is going to be very noisy. However, because L is fairly constant, we can heavily filter the estimates from (2), thereby eliminating the effects of the noise.

Much more detail on the algorithm, together with reasonably comprehensive simulation results, can be found in [8] and [9].

III. HARDWARE IMPLEMENTATION ISSUES

The basic algorithm as written in (1) and (2) can be manipulated to minimize the number of multiplications and divisions and improve scaling of the values, allowing a simpler and more efficient hardware implementation with an add/subtract/shift arithmetic logic unit (ALU). This section outlines these alterations.

A. Duty Cycle

The back EMF may be found by rearranging (1)

$$\begin{aligned} \frac{e[k]}{V} &\approx \frac{e_{\text{av}}[k, k+1]}{V} \\ &= \alpha[k, k+1] + \frac{2L}{VT} (i[k] - i[k+0.5]). \end{aligned} \quad (3)$$

Assuming that V is constant over two successive control intervals, and using linear extrapolation on e and α , α in (1) may be expressed as

$$\begin{aligned} \frac{T}{2} \alpha[k, k+1] &= 2 \left(\frac{T}{2} \alpha[k-1, k] \right) - \frac{T}{2} \alpha[k-2, k-1] \\ &\quad + \frac{L}{V} (i_{\text{des}}[k, k+1] - 4i[k-0.5] \\ &\quad + 3i[k-1] + i[k-1.5] - i[k-2]). \end{aligned} \quad (4)$$

The $\frac{T}{2} \alpha$ variable is a convenient representation of duty cycle because it appears in various forms in the conversion to the switching times. Equation (4) involves five multiply/divide operations, whereas (1) involves seven multiply/divide operations.

TABLE I
PWM FIRING TIMES FOR VARIOUS SECTORS

	Condition for sector	Firing order	t_0	t_1	t_2
Sector 1	$\alpha_d > 0; \alpha_q > 0; \alpha_q < \sqrt{3}\alpha_d$	$V_8 V_1 V_2 V_7 V_7 V_2 V_1 V_8$	$\frac{T}{4}(1 - \alpha_d - \frac{\alpha_q}{\sqrt{3}})$	$\frac{T}{2}(\alpha_d - \frac{\alpha_q}{\sqrt{3}})$	$\frac{\alpha_q T}{\sqrt{3}}$
Sector 2	$\alpha_q > 0; \alpha_q > \sqrt{3} \alpha_d $	$V_8 V_3 V_2 V_7 V_7 V_2 V_1 V_8$	$\frac{T}{4}(1 + 2\alpha_d)$	$\frac{T}{2}(\frac{\alpha_q}{\sqrt{3}} - \alpha_d)$	$\frac{T}{2}(\alpha_d + \frac{\alpha_q}{\sqrt{3}})$
Sector 3	$\alpha_d < 0; \alpha_q > 0; \alpha_q < \sqrt{3} \alpha_d $	$V_8 V_3 V_4 V_7 V_7 V_4 V_3 V_8$	$\frac{T}{4}(1 + \alpha_d - \frac{\alpha_q}{\sqrt{3}})$	$\frac{\alpha_q T}{\sqrt{3}}$	$-\frac{T}{2}(\alpha_d + \frac{\alpha_q}{\sqrt{3}})$
Sector 4	$\alpha_d < 0; \alpha_q < 0; \alpha_q < \sqrt{3} \alpha_d $	$V_8 V_5 V_4 V_7 V_7 V_4 V_3 V_8$	$\frac{T}{4}(1 + \alpha_d + \frac{\alpha_q}{\sqrt{3}})$	$-\frac{\alpha_q T}{\sqrt{3}}$	$\frac{T}{2}(-\alpha_d + \frac{\alpha_q}{\sqrt{3}})$
Sector 5	$\alpha_q < 0; \alpha_q > \sqrt{3} \alpha_d $	$V_8 V_5 V_6 V_7 V_7 V_6 V_5 V_8$	$\frac{T}{4}(1 + \frac{2\alpha_d}{\sqrt{3}})$	$-\frac{T}{2}(\alpha_d + \frac{\alpha_q}{\sqrt{3}})$	$\frac{T}{2}(\alpha_d - \frac{\alpha_q}{\sqrt{3}})$
Sector 6	$\alpha_d > 0; \alpha_q < 0; \alpha_q < \sqrt{3}\alpha_d$	$V_8 V_1 V_6 V_7 V_7 V_6 V_1 V_8$	$\frac{T}{4}(1 - \alpha_d + \frac{\alpha_q}{\sqrt{3}})$	$\frac{T}{2}(\alpha_d + \frac{\alpha_q}{\sqrt{3}})$	$-\frac{\alpha_q T}{\sqrt{3}}$

The standard expressions for converting three-phase to two-phase currents are

$$3i_d = 2i_a - i_b - i_c \quad (5)$$

$$\sqrt{3}i_q = i_b - i_c. \quad (6)$$

To further reduce the total operations, the currents used in the internal variables are $3i_d$ and $\sqrt{3}i_q$. Therefore, the d -axis duty cycle equation becomes

$$\begin{aligned} & \frac{T}{2}\alpha_d[k, k+1] \\ &= 2\left(\frac{T}{2}\alpha_d[k-1, k]\right) - \frac{T}{2}\alpha_d[k-2, k-1] \\ &+ \frac{L}{3V}(3i_{d\text{des}}[k, k+1] - 4 \times 3i_d[k-0.5] \\ &+ 3 \times 3i_d[k-1] + 3i_d[k-1.5] - 3i_d[k-2]). \end{aligned} \quad (7)$$

A similar expression $T\alpha_q/(2\sqrt{3})$ for the q axis can be calculated so that only one multiplication by $\sqrt{3}$ is required.

B. Switching Times

The switching time expressions appear in Table I. Since the duty cycles are represented internally as $T\alpha_d/2$ and $T\alpha_q/(2\sqrt{3})$, only one multiply operation is required to calculate t_1 and t_2 . t_0 is calculated using $t_0 = T/2 - (t_1 + t_2)$.

Voltage limits must be applied to the output so that it may be physically realizable with a switching waveform. Using the above method to calculate switching times, limiting must be applied when the switching times satisfy $2(t_1 + t_2) > T$. This may be detected by examining the sign of the calculated t_0 . The limit is applied by maintaining the angle of the voltage vector, but scaling the magnitude by a value γ . The required γ is

$$\gamma = \frac{T}{2(t_1 + t_2)}. \quad (8)$$

The duty cycles are modified through the calculation $\alpha_{d\text{lim}} = \gamma\alpha_d$ and $\alpha_{q\text{lim}} = \gamma\alpha_q$. As these are the actual voltages applied, the limited values must be used for the back-EMF estimate on the next control cycle.

A more complex constant angle active voltage vector approach can also be applied. This technique has the advantage that it does not introduce dq axis crosscoupling (as the previous technique does) when used in a vector-controlled drive. However, the price paid for the better performance

Fig. 2. Overall block diagram of the chip.

is extra computation. This can be seen from the following expressions for the limited α 's using this constraint [9]:

$$\alpha_{d\text{lim}}[k+1] = \frac{\sqrt{3}V + K^*e_d[k] - e_q[k]}{\sqrt{3}V(1 + \frac{K^*}{\sqrt{3}})} \quad (9)$$

$$\alpha_{q\text{lim}}[k+1] = \frac{K^*(V - e_d[k]) + e_q[k]}{V(1 + \frac{K^*}{\sqrt{3}})}. \quad (10)$$

Remark 3: This form of limiting has not been implemented in the hardware at this stage.

C. Inductance Estimation

The machine leakage inductance is estimated using (2). The values produced from this expression are low-pass filtered. An efficient first-order filter implementation from a hardware perspective is

$$L_{k+1} = L_k + \frac{L_{\text{est}} - L_k}{2^n}$$

where n determines the time constant.

IV. HARDWARE ARCHITECTURE

The current control algorithm described above has been implemented in an Altera 10K series field-programmable gate array (FPGA). Note that an FPGA was used only for a prototype. An application-specific integrated circuit (ASIC) would be the optimal final implementation. The overall design of the chip hardware is shown in Fig. 2. There are three main sections used in the architecture.

- 1) *Data Acquisition*—The analog currents are sampled using Hall-effect transducers and the values are converted

Fig. 3. Input and output connections to the ALU.

to digital values using a serial A/D, and then transmitted to the FPGA via an isolated serial channel that incorporates error detection. Overcurrent protection on each phase, together with dc-link protection, is implemented digitally in the FPGA. A 5-Mbps link is used, allowing measurement at 250 000 samples per second. Given the machine inductance, this is fast enough to offer an additional level of protection over that offered by the drivers themselves.

- 2) *Computational Unit*—This consists of an ALU and its associated sequencer. The ALU is capable of 16-b addition, subtraction, and shifting. A 32-word register file is used to store intermediate values. The sequencer controls the type of operations performed in the ALU and their sequence, and is essentially microprogrammed.
- 3) *PWM Generator*—This constructs the three-phase switching pattern from the switching times calculated in the ALU. As an option, these times can be compensated to account for inverter deadtime.

A. Computational Structure

A microcoded machine design philosophy has been used for the chip. Most of the calculations for the control algorithm are performed in a single ALU. The system inputs and outputs are connected to the ALU via a common bus, as shown in Fig. 3. A register file is used to store the intermediate results. This arrangement allows the calculations to be performed in sequence, as they would be in a microprocessor. An alternative design is to build separate calculation blocks for each of the mathematical operations. This avoids the need for bus logic, multiplexers, registers, and a microcode controller, but requires more resources for adders and multipliers. The microcode design requires fewer on-chip resources, but sacrifices speed, especially for complex calculations. The hardware-logic-based solution will give increased speed of execution compared to the microcode design, but will consume more on-chip resources.

The microcode design approach was chosen, as it was the appropriate solution for the speed and size requirements of the current control algorithm. Furthermore, the Altera 10K FPGA chosen for the prototype is particularly amenable to this approach, as it can efficiently implement register files and memory required for this type of design.

Fig. 4. Block diagram of the ALU.

From Fig. 3, it can be seen that external input values enter the ALU via a single 16-b-wide data path. The source for this path is selected by the input multiplexers, giving a choice between external inputs or internal register values. The multiplexers are controlled by the sequencer. The register file consists of storage for 32 values, each 16 b in width. These locations are used for storing parameters and intermediate calculation results. The output of each calculation may be latched into any of the output latches or written back into the register memory. The overall operation of this is controlled by the sequencer, which is briefly described later.

B. ALU Structure

Fig. 4 shows the basic structure of the ALU. Each iteration of the algorithm requires a number of arithmetic operations using this unit. The approximate requirements (excluding the inductance estimation) are three multiplies, three divides, 60 additions/subtractions, and eight left shifts. The small number of multiply/divide operations allows implementation using add/subtract/shift operations.

The ALU was designed to handle 16 b, as this was consistent with the resolution of the inputs to, and outputs from, the system. There are 32-b registers to allow the manipulation of the results of the multiply and divide operations. The unit is designed for pipelined execution of two concurrent instruction streams. This is possible through the use of two internal register sets ($R2$ and $R1/R3$). On each cycle, the contents of each register alternates between the two instruction paths.

Fig. 4 shows that the ALU has one input and one output. For operations requiring two parameters, such as addition, one parameter is read in first and stored in the internal registers of the ALU. On each clock cycle, a new value is read into each of the registers $R1$, $R2$, and $R3$. The source of the values is controlled by the command inputs $C1$, $C2$, and $C3$, respectively. These commands control the operation of the ALU. The possible commands are shown in the Table II.

TABLE II
COMMAND INPUT DEFINITIONS

C1	Description
00	R1=External read from Addr
01	R1=Read memory location Addr
10	R1=R2 (used for $a=-a$ and $a=abs(a)$)
11	R1 undefined, Write memory location Addr with R2

C2	Description
000x	R2=0
010x	R2=R3
0011	R2=R1
0010	R2=-R1
0111	R2=R3+R1
0110	R2=R3-R1
1000	Multiply Instruction 1
1001	Divide Instruction 1
1010	Divide Instruction 3
1011	R2=abs(R1)
11xx	Undefined

C3	Description
x00	R3=R2
001	R3=R2 shifted left
101	Shift+conditional add for division
x10	R3=R2 shifted right
x11	R3=R2 shifted right 16 bits

C4	Description
0	Select low word of R2 for mem/ext write
1	Select high word

The operation of the ALU may be demonstrated with the simple example of $C = A + B$, where A , B , and C are locations in the register file. The actions on each rising edge of the system clock and the commands required to achieve them are as follows.

- 1) Load A into R1 ($C1 = 01$).
- 2) Load A into R2 ($C2 = 0011$).
- 3) Load B into R1 and A into R3 ($C1 = 01$ and $C3 = 000$).
- 4) Load $A + B$ into R2 ($C2 = 0111$).
- 5) Store $A + B$ into location C ($C1 = 11$ and $C4 = 1$).

The structure of the ALU was governed by the operations required in executing the current controller algorithm. The main operations are addition, subtraction, and shifts. This type of calculation is performed in a similar way to the example. Multiplication and division are also required, but due to the small number of these operations and the complexity in performing them directly in hardware, they are implemented by shift/add techniques. The ALU was designed to perform a 16×16 -b signed multiplication in 32 clock cycles, and a 16-b division in 64 cycles.

C. The Sequencer

The sequencer generates the control signals for the ALU. On each clock cycle, a number of control signals needs to be supplied. For a simple calculation sequence, this would be achieved using a state machine. Unfortunately, conventional state machines suffer from increasing output decoding delays

Fig. 5. The structure of the sequencer.

TABLE III
INSTRUCTION WORD FORMAT

15	14	10	9	8	6	5	2	1	0
1/2 Words	Addr	C4	C3	C2	C1				
Word 1									
15	8	7	5	4	0				
Br Address	Br Condition	Command							
Word 2									

as the number of states increases. State machine approaches were abandoned when it was found that the required number of states for this application was unwieldy.

An alternative to using a large state machine is a sequencer, which is based on using microcode stored in an internal ROM. A small state machine is used to fetch the microcode words from the ROM and generate the commands for the ALU. A diagram of this is shown in Fig. 5.

The ROM consists of 256 words of width 16 b. In the prototype version, it is implemented within the FPGA. The address is provided by a presettable up-counter, which is controlled by the instruction decoder. Normal program execution is achieved by incrementing the counter and branches are implemented by presetting the counter to the new address.

The instruction decoder reads the contents of the ROM to generate the required commands. Each microcode instruction consists of either one or two 16-b memory words. The format of these words is shown in Table III.

All instructions include the first word. The “1/2 Words” bit indicates whether the second word is to follow. If this bit is set to zero, the second word is assumed to be zero, and the next word in the ROM is interpreted as the first word of the next instruction. This approach reduces the amount of chip resources required to represent the algorithm, as the second word is only needed infrequently. The four fields $C1 \dots C4$ contain the command signals to be sent to the ALU, as shown in Table II. These bit patterns are sent directly to the ALU at the appropriate time. The *Addr* field supplies the address for the register file and the select lines for the external input multiplexer.

The second word is included when the instruction requires branching to another microcode location, or if an “external command” is required. The external commands are used primarily to signal the presence of output values from the ALU. For example, there are three external commands connected to the PWM module to indicate when the switching times are available on the ALU output. The five bits allocated to the “Command” field are decoded to allow up to 32

```

//allocate an entry in the register file
int Counter=2;
//pointer to external input
constant IB=6;

a=Counter;
a=a*two+Counter; // a=3*counter
a=a-IB;          // subtract from IB
a=-a;            // load a back via R1

```

Fig. 6. Example source code for the assembler.

external commands to be generated. Although only one of these commands may be asserted in a single instruction, this is not a limitation, as the algorithm requires only infrequent use of commands.

The branching logic makes use of two fields in the second instruction word. The “BrAddress” field specifies the destination location of a branch if a branch is to occur. The “BrCondition” field specifies which condition is to be used for the branch decisions. The two basic conditions are “branch never” and “branch always,” which result in sequential execution and unconditional branching, respectively. The other conditions are similar in nature to “Branch if the ALU result is negative”.

D. Microcode Assembler

A simple microcode assembly language and assembler was developed to ease the writing of the microcode. Both development and debugging time are reduced considerably through using the more intuitive representation allowed with an assembler. In addition, it greatly simplifies the testing of modifications of some aspects of the algorithm, such as the inductance estimator, for example. An example segment of the source code is shown in Fig. 6.

This example multiplies the contents of a register in the file by three and then subtracts it from the latest measurement of the B-phase current.

E. Inductance Estimator

Due to the precision and special operations required, the inductance estimator has additional hardware support outside the ALU. In order to accommodate variations in possible inductance values and switching frequencies, a floating-point scheme was adopted. A 27-b register is used from the mantissa and a 3-b counter for the exponent. The basic structure is shown in Fig. 7.

The first step in updating the inductance average is to find the difference between the newest estimate and the existing average. This is calculated within the ALU, although the inductance counter is used to count the appropriate number of shifts. The hardware sign extends the ALU result and adds it to the existing average. In order to use the inductance average in the calculation, the 16 most significant bits are used. The counter is again utilized to apply the correct amount of shifting in the ALU.

(a)

(b)

Fig. 7. Inductance estimator structures. (a) Mantissa register. (b) Exponent counters.

Fig. 8. Connection of the current controller chip.

F. Ancillaries

The connection of the current controller chip to the rest of the system is shown in Fig. 8. The only external components are the inverter, A/D converters, and the outer loop controller. Future plans include the integration of a field-oriented torque controller into the current control chip, leaving only the application-specific controller.

In order to achieve this level of integration, there are several other units on the chip, in addition to the computational unit. As alluded to previously, these include a number of high-speed synchronous serial channel receivers that are used to acquire data from the serial A/D converters that are located remotely on the Hall-effect transducers. The bit rate for each channel is 5 Mb/s, and each data packet is 20 b. This allows 250 000 samples per second, giving an effective bandwidth of approximately 100 kHz. This matches the bandwidth of the Hall-effect transducers now in common use. The use of

Fig. 9. Block diagram of the experimental setup.

serial transmission has the advantage that far fewer I/O pins are required on the FPGA. Furthermore, it allows simple low-cost isolation of the cables from the Hall-effect devices (if required), and eliminates potentially long cables carrying analog signals.

Once the current samples are received, they are compared using digital comparators with the trip levels. If a positive or negative cycle of a phase current exceeds the trip level, then the FPGA disables the drives to the power devices and trips the drive system. In the prototype, the trip levels were set by the outer loop controller, but this could easily be achieved in a variety of other ways, such as switches on the controller printed circuit board (PCB).

The final unit in the chip is the PWM generator. It includes programmable deadbands and deadband compensation. The chip may be programmed to operate over a large range of switching frequencies, up to approximately 20 kHz with a pulse timing accuracy of 70 ns. Energy dumping regeneration is supported, with integrated control for a brakechopper.

V. EXPERIMENTAL RESULTS

Simulation results for noiseless and noisy current measurements with the algorithm have been previously presented in [8] and [9]. These indicated that the algorithm worked well in simulation (at least). In order to experimentally verify the algorithm's performance, the fairly conventional setup of Fig. 9 was constructed. The controller itself consisted of an Intel 80c186 microprocessor board connected to the Altera 10K50 FPGA containing the current controller. The microprocessor generated the current references for the current controller.

The digital hardware was developed on an in-house designed Altera 10K series development board, as shown in Fig. 10. The design as it exists at the moment uses 1204 of the 2880 logic cells available in the Altera 10K50. This represents approximately 41% of the available gates in the FPGA.

A direct field-oriented controller has been implemented on the attached processor to generate the current references. Preliminary experimental results from the hardware implementation have been obtained using this configuration. The following results were obtained from tests with a 7.5-kW machine connected to an inverter operating off a 120-Vdc supply. The switching frequency used was 2.9 kHz, and the

Fig. 10. Photograph of the development board.

TABLE IV
TEST MACHINE PARAMETERS

Parameter	Value
Rated Power	$P = 7.5\text{ kW}$
Core Losses	$R_c = 1.349\text{ k}\Omega$
Magnetising reactance	$X_m = 104.5\Omega/\text{Ph}$
Stator resistance	$R_1 = 1.89\Omega$
Rotor resistance	$r'_2 = 2.2\Omega$
Leakage Reactance	$X_1 + x'_2 = 9.36\Omega$
Frequency	$f = 50\text{ Hz}$

Fig. 11. Phase current and measured torque for a demanded torque transient with a free rotor.

machine parameters appear in Table IV. For these preliminary tests, the inductance estimator was disabled, with measured inductance parameters used instead. While the inductance estimator implemented has been successful in simulation [8], algorithm improvements are necessary for reliable operation in practice. In particular, the estimate is sensitive to delays introduced by inverter deadtime.

Fig. 11 shows the measured current and torque for a machine with an inertial load. The upper trace on the CRO display is the signal from a current probe attached to one phase of the machine. The lower trace is the output of a torque transducer attached to the output shaft. The peak current shown is approximately 7 A, and the torque output is about

Fig. 12. Phase current and measured torque for a demanded torque transient with a locked rotor.

Fig. 13. Response of the current controller to a step change in setpoint. Step is from -5 A to $+5$ A on the d axis.

5 N·m in each direction. The first part of the plot shows the machine accelerating under a negative torque setpoint. The sharp change in measured current indicates the reversal in the torque setpoint. Unfortunately, the torque measurement system has a bandwidth limitation of 10 Hz, so this measurement cannot reflect the true torque transient.

Fig. 12 was obtained from tests with a locked rotor. Fig. 12 shows square-wave torque output and the corresponding machine current. The step change in torque requires a step change in current. The required sharp change is visible just before the $t = 0$ axis on channel 1. The fast nature of the transient indicates the high bandwidth of the current controller in tracking the setpoint. The measured torque response is somewhat slower. As mentioned above, the transient on the torque measurement is governed by the filtering in the transducer electronics, not the actual torque produced by the machine.

Fig. 13 shows the response of the current controller to a step change in desired current. This shows a fast transient response and small steady-state error. With an appropriate outer loop torque controller, the electromagnetic torque bandwidth will follow the bandwidth of the current, resulting in good torque performance.

VI. CONCLUSION

In summary, this paper has presented a new completely digital hardware implementation of a novel computationally efficient predictive current control algorithm suitable for IM variable-speed drives. Experimental results show that this implementation is capable of high-bandwidth current control in a practical situation.

The development of this digital current controller is the first part of a larger program to develop a single IC vector controller. This IC would not only contain the current controller, PWM generation, sampling system, and protection, but also the vector-based torque controller. If a simple 8-b processor is also integrated into the same IC, then one has the possibility of a single-IC drive system controller.

REFERENCES

- [1] L. Malesani, P. Mattavelli, and P. Tomasin, "Improved constant frequency hysteresis current control of VSI inverters with simple feed-forward bandwidth prediction," in *Conf. Rec. IEEE-IAS Annu. Meeting*, Lake Buena Vista, FL, Oct. 1995, pp. 2633–2639.
- [2] Q. Yao and D. G. Holmes, "A simple novel method for variable hysteresis band current control of a three phase inverter with constant switching frequency," in *Conf. Rec. IEEE-IAS Annu. Meeting*, Toronto, Ont., Canada, Oct. 1993, pp. 1122–1129.
- [3] D. G. Holmes and D. A. Martin, "Implementation of a direct digital predictive current controller for single and three phase voltage source inverters," in *Conf. Rec. IEEE-IAS Annu. Meeting*, San Diego, CA, Oct. 1996, pp. 906–913.
- [4] O. Kukrer, "Discrete-time current control of voltage-fed three-phase PWM inverters," *IEEE Trans. Power Electron.*, vol. 11, pp. 460–469, Mar. 1996.
- [5] P. C. Kjaer, C. Cossar, and T. J. E. Miller, "Very high bandwidth digital current controller for high-performance motor drives," in *Proc. Power Electronics and Variable-Speed Drives Conf.*, Nottingham, U.K., 1996, pp. 185–190.
- [6] F. Blaabjerg, P. C. Kjaer, P. O. Rasmussen, L. Christensen, S. Hansen, and J. R. Kristoffersen, "Fast digital current control in switched reluctance motor drive without current feedback filters," in *Proc. Eur. Conf. Power Electronics and Applications*, Trondheim, Norway, 1997, pp. 3625–3630.
- [7] W. Farrer and J. D. Miskin, "Quasisine-wave fully regenerative inverter," *Proc. Inst. Elect. Eng.*, vol. 120, no. 9, pp. 969–976, Sept. 1973.
- [8] R. E. Betz, B. J. Cook, and S. J. Henriksen, "A digital current controller for three phase voltage source inverters," in *Conf. Rec. IEEE-IAS Annu. Meeting*, New Orleans, LA, Oct. 1997, pp. 722–729.
- [9] R. E. Betz and B. J. Cook, *A digital current controller for three phase voltage source inverters* (Jan. 1997). [Online]. Available HTTP: <http://www.ee.newcastle.edu.au/users/staff/reb/Betz.html>

Soren J. Henriksen was born in Australia in 1974. He received the Bachelor of Engineering degree in computer engineering in 1996 from the University of Newcastle, Callaghan, Australia, where he is currently working towards the Masters of Engineering degree.

In 1997–1998, he was with the Centre for Integrated Dynamics and Control, University of Newcastle. His current interests are machine control and digital electronics design.

Robert E. Betz (M'92) received the B.E., M.E., and Ph.D. degrees from the University of Newcastle, Callaghan, Australia, in 1979, 1982, and 1984, respectively.

He is currently a Senior Lecturer in the Department of Electrical and Computer Engineering, University of Newcastle. His major interests are electrical machine drives, real-time operating systems, and industrial electronics. He was a Senior Research Fellow at the University of Glasgow, U.K., during 1990–1991 and the Danfoss Visiting

Professor at Aalborg University, Denmark, in 1998.

Dr. Betz is a member of the Industrial Drives Committee of the IEEE Industry Applications Society.

Brian J. Cook was born in the U.K. in 1945. He received a Higher National Diploma and a College Diploma from Plymouth Polytechnic, Plymouth, U.K., in 1966 and 1967, respectively, and the Ph.D. degree from Bristol University, Bristol, U.K., in 1973.

He was awarded a cadetship with the Central Electricity Generating Board in Britain's initial nuclear power industry. In 1969, he joined the Department of Electrical and Electronic Engineering, Bristol University, where he became a Research

Assistant. In 1974, he accepted a Lecturer position in the Department of Electrical and Computer Engineering, University of Newcastle, Callaghan, Australia, where he has been for 25 years. His current interests include machine control and renewable energy systems.