Custom Processor for AC-motor Control Implemented in Spartan 3E FPGA

Jakobsen, Uffe; Ahn, Jin-Woo

Published in:
Proceedings of the KIEE Annual Conference, 2009

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

2009년도 대한전기학회 전기기기 및 에너지변환시스템부문회 추계학술대회 논문집 (2009. 10. 23)

Custom Processor for AC Motor Control implemented in Spartan 3E FPGA

Uffe Jakobsen, 안진우*
IET, Aalborg University, Denmark 경성대학교 전기전자메카트로닉스 공학부*

Custom Processor for AC Motor Control implemented in Spartan 3E FPGA

Uffe Jakobsen, Jin-Woo Ahn*
IET, Aalborg University, Denmark, EME., Kyungsung University, Korea*

Abstract - Motor control for small series sometimes requires specialized control logic, requiring rewiring if new logic needs to be added. This paper describes a different approach to hardware and software co-design, namely designing a 32 bit softcore processor with an instruction set to fit the purpose of control of drives.. The designer can then choose between resource usage on the FPGA and speed in new ways. The approach is tested for two different motor types, synchronous and hybrid switched reluctance motors, using a Spartan 3E FPGA. The instruction set allows for higher code density than Texas MSP430 and Texas TMS320F2812 for field oriented control.

1. Introduction

To control an AC machine a DSP or a micro controller is typically used. While they are excellent, their architecture is not directly targeted for motor control, since they do not feature an instruction set dedicated to motor control. For some applications even a fast DSP may be too slow, an example can be to replace an analog hysteresis controller as the inner control loop with a digital implementation as is the case in [1]. The FPGA has for some time been used for signal processing and the use of the FPGA for motor control has been explored before. There seems to be at least two different approaches are used: An approach based on signal flow is used in [2], [3] and [4]. They implemented the control loop by making custom elements for each part. These elements are then connected to form the closed loop. A general purpose softcore connected to custom external elements needed for the control of a PMSM is shown in [5] and [6]. The signal flow approach, also known as dataflow processing, has a hardwired setup which requires design and synthesis of a new processor if the control loop is changed. This issue can be addressed by using the more flexible approach of connecting external blocks to a general purpose softcore. However a general purpose softcore can be difficult to modify, since most of them are either proprietary, with closed source tied to a specific FPGA vendor, or open source with a design not suited for AC motor control. Having a simple method to modify the softcore enables a more flexible hardware-software co-design. This paper presents a proposed new processor, which makes the hardware-software co-design simpler by a simple plug-in instruction set approach. Three different implementations are presented: One processor customized for a permanent magnet synchronous machine (PMSM) without hardware support for the external ADC, one customized for a PMSM with hardware support for the external ADC, and yet another customized for a single phase hybrid switched reluctance machine (HSRM).

2. Softcore Architecture

2.1 Softcore Architecture

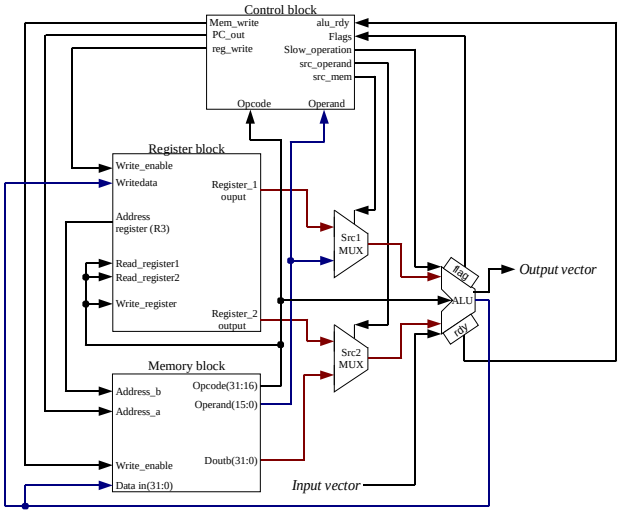

Fig. 1 The layout of the softcore processor

The softcore consists of four distinct modules and two multiplexers, as it is shown on Fig 1. A synchronous design is used for the softcore, with two state machines governing the behavior. One state machine is used for the control block (see Fig. 2) and the other is for the algorithmic logic

unit (see Fig. 3). The major task for the control block is to control the program counter and to prohibit writes during compare Instructions.

The algorithmic and logic unit (ALU) performs all calculations and access to custom instructions and external blocks. The two multiplexers determine the source of the operands for the ALU. The register block contains control logic and storage for the registers. The memory block is a standard two port memory block, enabling a modified Harvard structure for the processor.

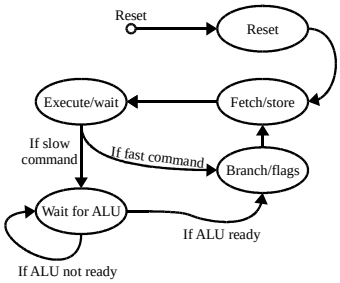

Fig. 2 The control block state machine

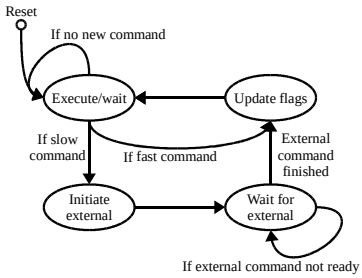

Fig. 3 The algorithmic and logic unit block state machine

2.2. Instruction Set Design

A custom instruction set is implemented to enable a simple mapping from the equations needed in AC-motor control to the actual machine code operations. The closer the instruction set is to the task, the fewer instructions are needed. The instruction set consists of core instructions, performed by the ALU, and external plug-in instructions.

2.2.1 Instruction Set

The used set of instructions is shown on table I. There are eight different addressing modes for all three operand instructions.

Table. 1 Instruction set of the proposed AC control processor

Instruction	Meaning
ANG	Calculates angle of a 2D-vector
ABS	Calculates magnitude of a 2D-vector
COS	Cosine
SIN	Sine
PWM	Set PWM-timers, wait and return rotor-pos.
SCL	Fixed point multiplication
ADD	Addition
SUB	Subtraction
MUL	Integer multiplication
SWP	Swap upper and lower part of number
ORB	Logical OR
AND	Logical AND
XOR	Logical XOR
CMP	Compare two operands
Bxx	Conditional branch
JMP	Unconditional Jump
MOV	Copy first operand
MVY	Copy Second Operand
LTC	Latch and roll from input vector
OUT	Output masked value
INP	Input masked value

2.2.2 Instruction Set Format

There is only one format for instructions, which is shown on Fig. 4, where the upper 16 bit determine what instruction it is, addressing mode and if it is a plug-in or slow instruction (S).

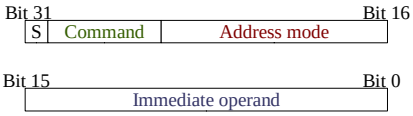

Fig. 4 The instructions all follow a single format with some bit patterns directly connected to e.g. multiplexers. Since no exceptions exist to this format, the control logic can be kept simple

2.2.3 Trigonometric Functions

To approximate the sine, cosine and the rectangular to polar functions a coordinate rotation digital computer (CORDIC) was implemented.

2.2.4 Peripheral Functions

The peripheral blocks are arranged in a typical system on a chip layout as shown in Fig. 5. To control the three phase inverter three identical PWM timers are instantiated. The PWM command sets the duty cycles for the three timers. The rotor position was acquired by implementing a quadrature interface. The PWM command returns the current encoder value. Since the FPGA used has no ADC internally, an external SPI interface was implemented. Finally to make a simpler interface to the assembler an UART for RS232 was implemented. The assembled machine code program can be uploaded to the softcore using the assembler RS232 interface on the PC-side and a simple download program in the softcore.

2.3 Implementation Results

2.3.1 PMSM Instance with Software logic for ADC

For the PMSM the softcore is kept as presented in the article, and the current loops are implemented using field oriented control. The loop times for the current loop is measured in a VHDL simulation is presented in

table 2. A loop time for a complete FOC of 71.1 microseconds, is more than 10 times slower than the 5.4 microseconds presented in [3]. The entire system for a PMSM consumes 35% of the resources in a Spartan 3E FPGA (xc3s500e-5fg320).

Table. 2 Times for a vector control loop (FOC). The times are given using a 50 MHz clock

Functional element	Time in nanoseconds
Get measurements from ADC	38178
Transform to rotating dq reference frame	11277
Control updates	8117
Transform to stationary alfa-beta reference frame	5246
Space vector modulation calc.	8309
Complete FOC	71127

2.3.2 PMSM Instance with Hardware logic for ADC

Except for the extra logic for the ADC interface the implementation is the same. The initialization of the ADC is still handled in software, but the current measurements are reduced to two instructions.. The effect on the loop time is shown in table III. The entire system for a PMSM consumes 37% of the FPGA resources.

Table. 3 Times for a vector control loop (FOC). The times are given using a 50 MHz clock

Functional element	Time in nanoseconds
Get measurements from ADC	120
Transform to rotating dq reference frame	11277
Control updates	8117
Transform to stationary alfa-beta reference frame	5246
Space vector modulation calc.	8309
Complete FOC	33249

2.3.3 HSRM Instance with Software logic for ADC

For the single phase HSRM only a very simple voltage control is implemented. The loop time of 981 ns is therefore also much lower. The commands COS, SIN, ABS and ANG are removed from the softcore and the resource usage drops to 14% in the same spartan 3E FPGA (xc3s500e-5fg320).

2.4 Comparison between different versions and DSP and microcontroller

To compare the FPGA PMSM implementations two implementations of FOC PMSM control are considered. A Texas TMS320F2812 DSP implementation map file indicates that 4431 bytes of RAM are used, and a MSP430 implementation cnsumed 52342 bytes of which 48kbytes are used for look-up tables to speed up the control loop, but still the full FOC is limited to 8kHz. The resource use for the FPGA implementations is shown on Table 4.

Table. 4 Comparative table regarding resource usage, for a Spartan-3E XC3S500E-5 FPGA at 50 MHz, with 20 blocks of on-chip memory, 4656 slices of logic in total, and 20 Hardware -multipliers (Hw-mul.). All implementations were compiled using Xilinx ISE 9.1I VHDL-compiler with a mixture of behavioural and register transfer logic.

Type	FPGA Slices	Memory used	Hw-mul.
FPGA/PMSM,SW-ADC	1629	1368 bytes	4
FPGA/PMSM,HW-ADC	1718	1128 bytes	4
FPGA/HSRM,SW-ADC	*691*	*112 bytes*	*1*

3. Conclusion

This paper presents a processor for motor control and discusses the design and implementation of such a processor in three different forms. To verify that the instruction set is close the application it is compared against FOC implemented in a DSP and in a microcontroller. The code density for the FPGA softcore compares quite well against both a DSP and a microcontroller, and still obtains a 20kHz FOC. The plug-in instruction set approach allows for easy updates of a FPGA design for different tasks.

[References]

[1] Frede Blaabjerg, Philip C. Kjaer, Peter Omand Rasmussen, and Callum Cossar Improved Digital Current Control Methods in Switched Reluctance Motor Drives, IEEE Transactions on Power Electronics, Volume 14, No. 3 , Page(s): 563 - 572, 1999

[2] Vincenzo Delli Colli, Robert Di Stefano, Fabrizio Marignetti, Maurizio Scarano Design of System-On-Chip PMSM Drive Sensorless Control, 2007 IEEE Symposium on Indstrial Electronics, Vigo

[3] T. Takahashi and J. Goetz Implementation of complete AC servo control in a low cost FPGA and subsequent ASSP conversion, Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition, Volume 1, Page(s): 565 - 570, 2004

[4] Zhaoyong Zhou, Tiecai Li, T. Takahashi, and E. Ho, FPGA realization of a high-performance servo controller for PMSM, Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition, Volume: 3, pages 1604-1609, 2004

[5] Ying-Shieh Rung and Pin-Ging Huang and Chien-Wu Chen , Development of a SOPC for PMSM drives, 47th Midwest Symposium on Circuits and Systems, 2004.

[6] Ying-Shieh Kung, Chia-Sheng Chen, Kiing-Ing Wong, and Ming-Hung Tsai, Development of a FPGA-based control IC for PMSM drive with adaptive fuzzy control, 31st Annual Conference of IEEE Industrial Electronics Society, 2005. 6 pp.-