GENERATING STEPPER MOTOR SPEED PROFILES USING FPGA

Daniel Piši
Doctoral Degree Programme (1), FEEC BUT
E-mail: xpisid00@stud.feec.vutbr.cz

Supervised by: Ilona Kalová
E-mail: kalova@feec.vutbr.cz

Abstract: This paper deals with implementation of linear speed controller for stepper motors within an FPGA. It shows how to deal with computational complexity and balance performance against resources requirements using Xilinx MicroBlaze soft microcontroller. The designed peripheral is able to generate linear speed profiles including microstepping with no microcontroller load involved, while performing computationally complex initialization the software way.

Keywords: Stepper motor, linear speed controller, FPGA, Xilinx, MicroBlaze

1 INTRODUCTION

Stepper motors have found its use in a range of electric devices, in consumer electronics as well as industry. They became popular mainly thanks to the possibility of being controlled accurately without feedback, eliminating need for expensive position sensing.

Besides the mentioned advantage, they profit from the ubiquity of digital circuits. They can be controlled by dedicated integrated circuits as well as general purpose microcontrollers. However, when complex speed profiles needs to be generated precisely with microstepping involved, the situation gets complicated. Complex calculations with deterministic timing becomes excessively demanding, not speaking of driving several motors simultaneously.

This led the author to investigate a possibility to solve the problem using field programmable logic array. More precisely, a combination of soft microcontroller core with programable logic. This synergic combination makes an ideal platform for designing high performance deterministic logic along with complex mathematics within single chip.

The presented solution becomes an ideal choice when having an FPGA already on board. Integrating the stepper motor driver on existing chip lowers the total system cost and maintaining high flexibility.

2 GENERATING LINEAR SPEED PROFILES

The whole problem of generating linear speed profiles comes from complexity of inter-step delay calculation. The relation between shaft position and its angular speed ω is defined as

$$\varphi(t) = \int_{\tau=0}^{t} \dot{\omega}(\tau)\mathrm{d}t = \frac{1}{2}\dot{\omega}t^2 \tag{1}$$

Considering an ideal stepper motor, the position φ is discrete, a multiply of pulse count n and motor's angular step α. The time at which the shaft reaches position $\varphi = n\alpha$ can be derived directly from the equation (1).

$$t_n = \sqrt{\frac{2n\alpha}{\dot{\omega}}} \tag{2}$$

Expressing a difference between two subsequent steps leads us to the formula for inter-step delay.

$$\Delta t = t_{n+1} - t_n = \sqrt{\frac{2\alpha}{\dot{\omega}}} \left(\sqrt{n+1} - \sqrt{n} \right) \tag{3}$$

Unfortunately, computational complexity of square root is too high for general purpose microcontrollers to be performed in real time, each time a motor's step is performed. With logic oriented FPGAs, the situation is no better. Logic able to handle the complex computation would be unreasonably resource demanding. Therefore, an approximation is considered.

The article [1] shows that omitting higher Taylor series members with conjunction of simple compensation reduces mathematical complexity significantly, maintaining the error well acceptable for a wide range of applications.

The approximation is based on Taylor series

$$\sqrt{1 \pm \frac{1}{n}} = 1 \pm \frac{1}{2n} - \frac{1}{8n^2} + O\left(\frac{1}{n^3}\right) \tag{4}$$

Expressing a fraction of two subsequent inter-step delays using the Taylor series, omitting the higher degree members gives us

$$\frac{c_n}{c_{n-1}} = \frac{c_0 \left(\sqrt{n+1} - \sqrt{n} \right)}{c_0 \left(\sqrt{n} - \sqrt{n-1} \right)} = \frac{1 + \frac{1}{2n} - \frac{1}{8n^2} + O\left(\frac{1}{n^3}\right) - 1}{1 - \left(1 - \frac{1}{2n} - \frac{1}{8n^2} + O\left(\frac{1}{n^3}\right)\right)} = \frac{4n-1}{4n+1} \tag{5}$$

Leading to the approximation we have been looking for

$$c_n = c_{n-1} - \frac{2c_{n-1}}{4n+1} \tag{6}$$

3 IMPLEMENTATION

The stepper motor controller has been designed as PLB bus peripheral. This makes it usable within Xilinx MicroBlaze soft microcontroller based systems as well as systems based on PowerPC cores integrated on selected Xilinx Virtex FPGAs.

The presented solution does not rely on microcontroller's core while the stepper motor is running, while taking advantage of soft microcontroller for complex computations being performed only once, before the motor starts. This way, precise timing with no CPU usage for linear acceleration/deceleration is achieved, while not wasting expensive logic resources for complex mathematics where unreasonable.

The source code and mathematics are based on Atmel's application note AVR446 [2]. Beside reimplementing the logic in VHDL, resulting in accurate timing, the microstepping has been introduced and accuracy of calculations within initialization improved by using floating point arithmetic instead of fixed point.

3.1 SOFTWARE DRIVER

The main purpose of the C library implementing software driver is to make the peripheral available to a programmer. Beside this, it performs computations necessary to initialize the peripheral before the rotation starts. The first inter-step delay c_0 has to be calculated and step at which deceleration starts determined.

The acceleration rate is proportional to the first inter-step delay c_0, thus the accuracy of acceleration rate relies on accuracy of c_0.

$$c_0 = \frac{1}{t_t}\sqrt{\frac{2\alpha}{\dot{\omega}}} \tag{7}$$

where t_t is timer period, α a single step angle and $\dot{\omega}$ desired acceleration rate in $\mathrm{rad} \cdot \mathrm{s}^{-1}$.

Having the powerful MicroBlaze core and standard C libraries available, there is no problem to calculate c_0 accurately using floating point arithmetic.

Further, the driver has to determine speed profile's shape and find out step in which to start deceleration in order to perform exactly the desired number of steps.

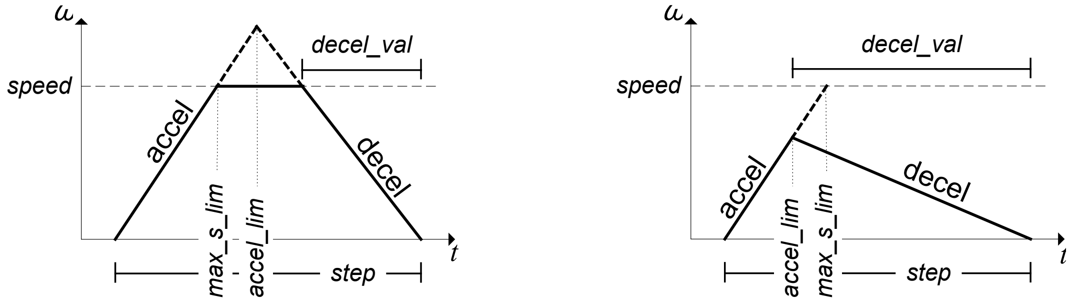

Figure 1: Speed profiles, source: [2]

The shape of the profile is defined by maximal allowable acceleration/deceleration rate (*accel*/*decel*), maximal speed (*speed*) and number of steps being performed (*step*).

Beside the two mentioned profiles, there are two more, trivial possibilities. When performing only a single step, there is no sense to calculate anything. The same situation occurs when having the permissible acceleration rate so high, the desired speed can be achieved immediately, leaving the *accel* part out. Nevertheless, these are situations where no linear speed controller is needed.

3.2 Peripheral Design

The peripheral consist of five sequential VHDL processes and auxiliary combinatorial logic. The block scheme can be seen in figure 2.

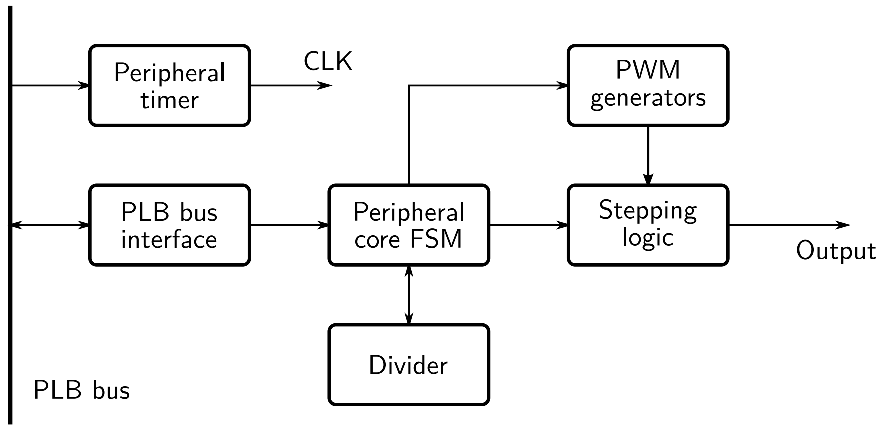

Figure 2: Block scheme of stepper controller peripheral

The peripheral is clocked synchronously with microcontroller's core. Time base is provided by resettable modulo timer *Peripheral timer*.

Connecting the peripheral to PLB bus allows microcontroller to access its registers as any other memory address. The process *PLB bus interface* provides a range of registers allowing to define parameters of the controlled motor such as maximal acceleration/deceleration rate, maximal speed or direction. Beside the data registers, there are registers initiating actions such as starting or stopping the rotation as soon as a value is written to them.

The essential part of the peripheral is made up of process *Peripheral core FSM*. It realizes controller's finite state machine, generates stepping pulses and initiates required calculations. The diagram illustrating the FSM as well as possible transitions can be found in Figure 3.

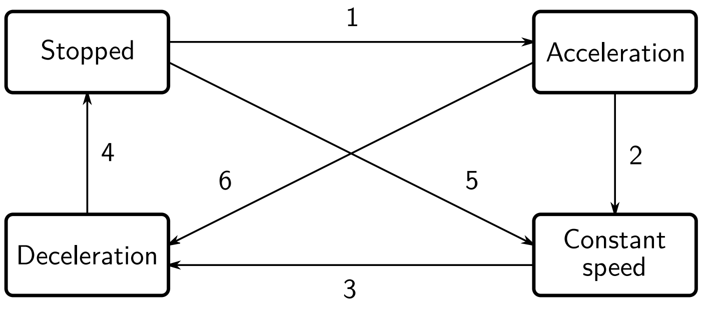

Figure 3: Stepper controller depicted as finite state machine

1. Starting a rotation, constant acceleration rate is required.
2. The maximal motor speed has been achieved.
3. Step in which deceleration must begin has been reached.
4. The motor has performed required number of steps.
5. Starting a rotation, while smooth acceleration is not required. This occurs when single step is required or the motor's dynamics allow to achieve maximal speed instantly.
6. Deceleration must be started, despite the maximal speed has not been reached.

The state transitions can be fired only when the peripheral's base timer overflows. Thus, the inter-step delay becomes a multiply of base timer's period. Once the inter-step delay elapses, variable *micro_step* is incremented and computation of oncoming delay started, so its value is ready to be used when needed. Overflow of the *micro_step* counter means the whole *step* has been made.

Since the expression for inter-step delay contains division, its evaluation is not a single step process. The division is realized by process *Divider* in conjunction with priority encoders *num_digit* and *div_digit*. The chosen algorithm belongs to the CORDIC family. The principle of the algorithm can be found explained at [4]. The division takes up to 33 clock cycles for 32-bit numbers.

The division process prevents lowering the modulo of base timer under 34 clock cycles. 33 steps has to be reserved for division itself and one for transferring the result to *Peripheral core FSM* process. However, considering the fact a mechanical device is being driven, this does not introduce a limitation. Having the microcontroller's core clocked at $f = 50$ MHz, the minimum base timer period becomes

$$T = \frac{34}{50 \cdot 10^6} = 0.68\,\mu\text{s} \tag{8}$$

For a stepper motor with 400 steps per round, this means the maximum speed $\omega_{max} = 23100\ \text{rad} \cdot \text{s}^{-1}$, what is an unachievable value for contemporary stepper motors.

The final stage of the peripheral design is combinatorial, titled *Stepping logic*. It generates PWM signals by comparing value of the peripheral's base timer with *micro_step* indexed data array. Then, the PWM signals are routed to proper stepper motor's winding, according to *step* counter value.

An illustration of peripheral's output is shown in Figure 4.

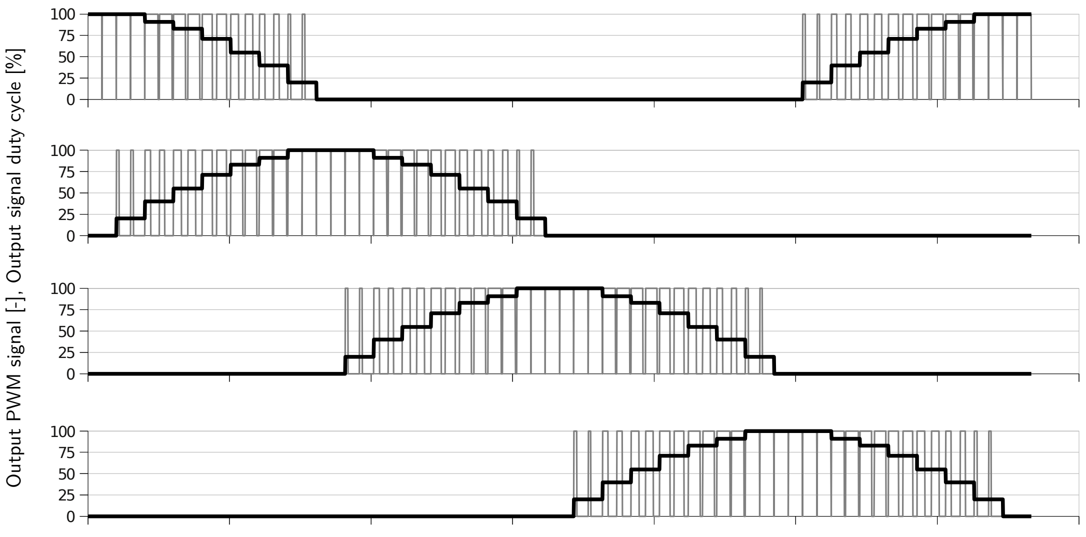

Figure 4: Output of stepper motor controller

4 CONCLUSION

This article has shown a way how to integrate a linear speed profile stepper controller into an FPGA. The design was verified using Avnet Xilinx Spartan-3A Evaluation Kit populated by Xilinx XC3-S400A. Using this FPGA, two stepper controllers can be implemented along with MicroBlaze soft microcontroller. The soft microcontroller occupies 47 % of LUT tables, while the stepper motor controllers 19 % each.

ACKNOWLEDGEMENT

This work was supported by grant "Modern Methods and Approaches in Automation" from the Internal Grant Agency of Brno University of Technology (grant No. FEKT-S-10-12).

REFERENCES

[1] Austin, D.: Generate stepper-motor speed profiles in real time, 5 p., EE Times-India, January 2005, available online at http://www.eetindia.co.in/ARTICLES/2005JAN/EEIOL_2005JAN03_EMS_TA.pdf?SOURCES=DOWNLOAD

[2] Atmel, AVR446: Linear speed control of stepper motor, 15 p. revision A, available online at http//www.atmel.com/dyn/resources/prod_documents/doc8017.pdf

[3] Piši, D., Mikrokontrolér v hradlovém poli XC3S400, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 30 s. Vedoucí semestrální práce Ing. Pavel Kučera, Ph.D.

[4] Fast division for PICs, web page, last update 10th August 2009, available online at http://www.convict.lu/Jeunes/Math/Fast_operations2.htm