HIGH PRECISION REFERENCE POSITION GENERATOR FOR MOTOR CONTROL IN FPGA

Vojtech Dvorak

Doctoral Degree Programme (3), FEEC BUT

E-mail: xdvora99@stud.feec.vutbr.cz

Supervised by: Lukas Fujcik

E-mail: fujcik@feec.vutbr.cz

Abstract: The paper deals with an architecture of a high precision reference position generator for a motor controller implemented in FPGA. The architecture was developed to achieve high accuracy and tunability with concern to resource efficient implementation in FPGA. Simulations of the proposed generator as well as synthesis results are presented in the paper.

Keywords: FPGA, Motor Controller, Synchronization

1. INTRODUCTION

A motor driving is wide group of task for electronic design. The most common approach is to utilize microcontroller due to low cost and well understand implementation of motor controller there. However, in some cases utilizing FPGA is desired, especially when other monitoring and control functions are required to implement beside motor controller. The structure of the motor controller in FPGA (in Figure 1) was developed to fulfill requirements to rotate motor with constant speed and reach commanded position with high precision in a time.

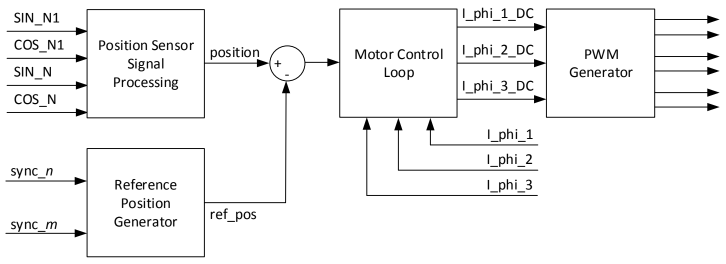

Figure 1: Block diagram of motor controller

The current position of the motor is computed in the *Position Sensor Signal Processing* from a position sensor with two tracks with different number of sinewaves per one revolution (Vernier principle). The difference between the current position and the reference position is input for the *Motor Control Loop* that is implemented as a cascaded structure of PI regulators regulating position, speed and current through motor windings. The output of the *Motor Control Loop* is voltage to be applied on the motor and is transferred to the *PWM Generator* that generates PWM modulated voltage for a motor driver. In this case, the reference position is a ramp computed from periodic pulses generated by a superior system. The computation of the reference position is performed by the *Reference Position Generator* block. The accuracy of motor driving directly depends on the accuracy of the computed current position of the motor and the accuracy of the reference position.

The FPGA vendors provide dedicated IP core resources for motor driving. However, the IP cores deal mostly with the motor control loop PID regulators [1][2].The algorithm to process data from position sensor is implemented according algorithms described in [3][4]. In this paper, an architecture of the *Reference Position Generator* is described.

2. REFERENCE POSITION GENERATOR

The reference position generator generates the ramp representing required position of the motor in time. There are two synchronization pulses to indicate required position in time. When rising edge of the pulse arrives, it is expected to be in the position defined in Figure 2. However, motor has to rotate with constant speed between rising edges of the pulses. This is achieved by interpolation of the required position with high resolution ramp.

The synchronization pulse *sync_m* indicates one revolution of the motor. The second synchronization pulse *sync_n* is used for accurate synchronization during rotation. The timing diagram of pulses together with the required position of the motor is in Figure 2.

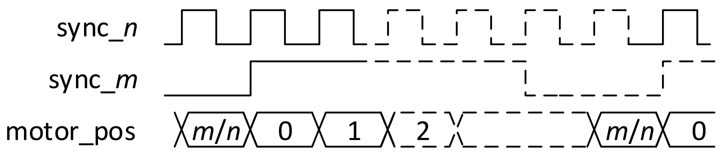

Figure 2: Timing diagram of synchronization pulses

The period of synchronization pulses can vary in a time and thus can be used to change speed of rotation. There is only one requirement for speed variability of the motor, the ratio *m*/*n* has to be a constant (i.e. the number of *sync_n* pulses in one *sync_m* period has to be still same when periods of synchronization pulses are changed).

The developed *Reference Position Generator* contains two counters (the *Reference Counter* and the *Ramp Counter* in Figure 3). *The Reference Counter* generates a time-accurate low-resolution ramp similar to the signal *motor_pos* in Figure 2. The low-resolution ramp is used as a reference for the high-resolution Ramp Counter that generates reference position. Due to required tunability, the additional correction logic is implemented to control the slope of the generated ramp to be the same as the slope of the output of the Reference Counter.

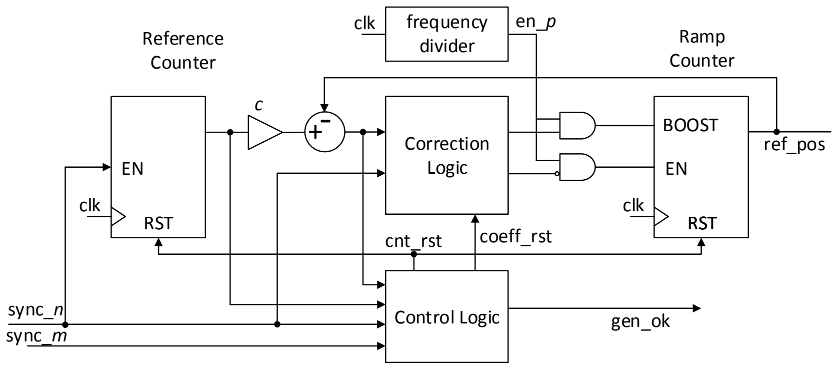

Figure 3: Block diagram of Reference Position Generator

The block schema of the developed Reference Position Generator is in Figure 3. Beside the counters, there is a correction logic, a control logic and a frequency divider. The frequency divider is optional and was implemented to reduce size of the Ramp Counter and to decrease sensitivity to signal jitter on the synchronization link.

The Control Logic ensures correct sequence of synchronization after power on. It also detects the loss of the synchronization link and provides sequence to resynchronize after synchronization link is established again.

The correction of the slope of the *Ramp Counter* is driven by the *Correction Logic* with control signals. The control signals can disable the *Ramp Counter* (to slow down counting) or boost counting of the *Ramp Counter*. When the boost is enabled, the *Ramp Counter* increments its output by 2.

A difference between both counters is used as an input for both the *Correction Logic* and the *Control Logic*. Because the *Reference Counter* has smaller resolution than the *Ramp Counter*, the value of the *Reference Counter* needs to be multiplied by a constant *c* computed according the (1).

$$c = (m / n) / p \tag{1}$$

Where:

m is a period of synchronization pulse *sync_m* (i.e. period of one motor revolution)

n is a period of synchronization pulse *sync_n*

p is a period of divided frequency of a system clock (if the frequency divider is omitted, *p* is period of system clock).

The proposed reference position generator was developed to achieve tunability ±10% with respect to the nominal speed of rotation. However, the tunability can be improved without any change in the architecture of the generator.

2.1. CORRECTION LOGIC

The *Correction Logic* was implemented to allow tunability of the slope of the generated reference position. The block diagram of the *Correction Logic* is in Figure 4. There are two parts of the logic, the Correction Coefficient Computation block computes correction coefficient to correct slope of the *Ramp Counter*. The second part applies the computed correction coefficient to the *Ramp Counter* with two signals – boost and disable.

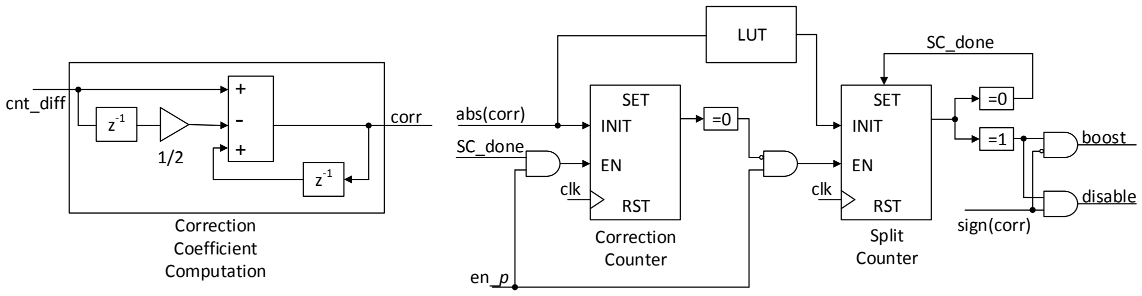

Figure 4: Block Schema of the Correction Logic

The correction coefficient is computed according (2).

$$corr = cnt\_diff - \frac{1}{2} cnt\_diff * z^{-1} + corr * z^{-1} \tag{2}$$

An application of the computed correction coefficient is split over the whole period of the *sync_n* pulse. This approach was selected to generate ramp with constant slope for whole period. Two counters were utilized to generate boost or disable signal for the *Ramp Counter*. The *Correction Counter* is first counter and counts number of required corrections. It is a down counter which is set to the value of the correction coefficient with rising edge of the *sync_n* and decrease its value by 1 when the second counter is finished. The second counter is the *Split Counter* that is used to split period of *sync_n* pulse to smaller periods according to the size of the current correction coefficient.

2.2. CONTROL LOGIC

The *Control Logic* starts synchronization of the *Reference Counter* and the *Ramp Counter* after power on and checks whether the generated reference position is valid. The *Control Logic* is implemented as a finite state machine according the state diagram in Figure 5.

The *Control Logic* generates sequence of resets for the counters and for the *Correction Logic* after power on. Both counters and the *Correction Logic* are hold in reset until the first synchronization pulse *sync_n* arrives. Consecutively, counters and the *Correction Logic* are enabled to determine the period of synchronization pulse *sync_n* and to compute corresponding correction coefficient. Until the first sync_*m* pulse arrives, the generated reference position is not valid (indicated by signal *gen_ok*). A rising edge on the *sync_m* is the trigger to reset both counters and start counting from zero with the correction coefficient computed in the previous states.

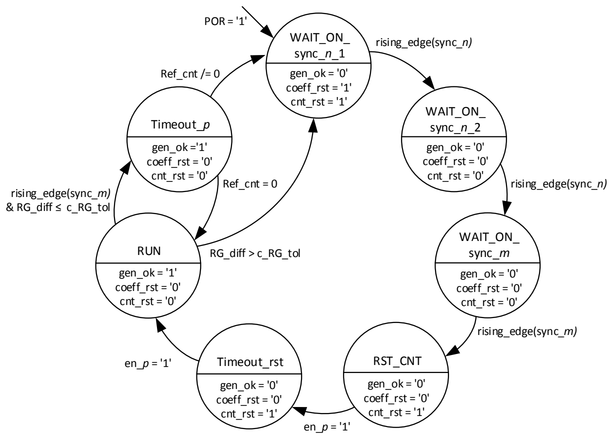

Figure 5: State diagram of Control Logic

The detection of the errors on the synchronization link is performed only in the RUN state of the *Control Logic*. There are two possible errors on the synchronization link. First error is the loss of the link due to disconnection. To detect this error, difference between the *Reference Counter* and the *Ramp Counter* is observed and once the difference exceeds limit defined by the variable *c_RG_tol*, the *Control Logic* goes to the WAIT_ON_sync_n_1 state and waits for the new synchronization pulse. The second error is caused by error in the superior system generating pulses. The number of *sync_n* pulses in one *sync_m* pulse is incorrect. The output of the Reference Counter is observed and checked whether is in zero after *sync_m* arrives. A timeout is utilized to avoid false error detection due to jitter on the synchronization link.

3. SIMULATION AND SYNTHESIS RESULTS

The parameters of the *Reference Position Generator* was set according Table 1 to perform simulation and synthesis to FPGA of the proposed architecture.

Parameter	Value
n	25 ms
m	5.5 s
p	5 μs

Table 1: Parameters of the Reference Position Generator

The functional simulation of the proposed architecture was performed with QuestaSim 10.2b digital simulator. The synchronization pulses were generated in verification environment coded in SystemVerilog. Simulation waveform can be found in Figure 6.

There were 8 tests performed to examine the behavior of the *Reference Position Generator* under different conditions. Operation with the nominal period of synchronization pulses was tested in Test 1, Test 2, Test 6 and Test 8. All the tests were performed to prove that the *Reference Position Generator* is able to work correctly after power on or after previous different periods of the synchronization pulses. Tunability of ±10% to the nominal period was tested in Test 3 and Test 4. Test 5 and Test 7 cover situation when the period of the synchronization pulses is out of allowed range (greater than ±10%).

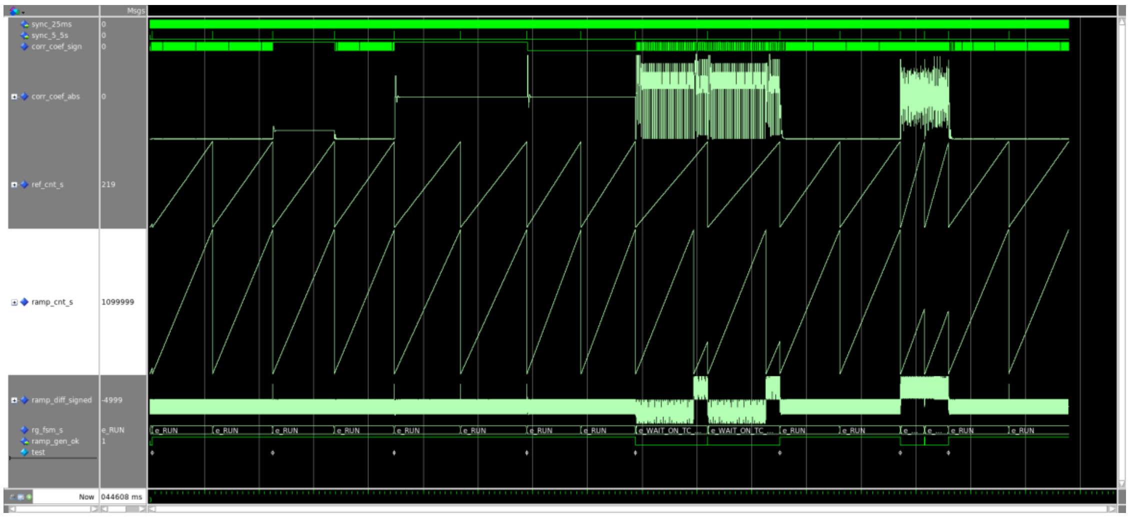

Figure 6: Simulation Waveform of the Reference Position Generator

The logic synthesis was performed to estimate resource utilization in FPGA. Microsemi RTAX 1000S FPGA was selected as a target FPGA. Results of the synthesis for the parameters defined in Table 1 are summarized in Table 2.

	Used	Total	Utilization
LUT [-]	564	12096	5%
DFF [-]	169	6048	3%
f_{max} [MHz]	75.5	-	-

Table 2: RTAX1000S FPGA Resource Utilization

4. CONCLUSION

The architecture of a high precision reference position generator was presented in the paper. The proposed generator is able to generate high precision reference ramp for motor controller to rotate motor with constant speed and reach commanded position indicated by events on synchronization pulses. Simulation of the proposed architecture was performed to prove functionality of the concept. Synthesis for target FPGA was performed to provide resource utilization of the generator in hardware.

ACKNOWLEDGEMENT

Research was supported from the project no. FEKT-S-14-2300 "A new types of electronic circuits and sensors for specific applications".

REFERENCES

[1] Microsemi: Dual-Axis Motor Control on a Single SoC FPGA, Microsemi, 2015

[2] Altera: Motor Control IP Suite Components for Drive-on-Chip Reference Designs, Altera, 2014

[3] iC Haus: High-Precision Sine/Cosine Interpolation, iC-House GmbH, 2014

[4] GEMAC: 2-Channel Interpolation Circuit with Nonius Calculation, GEMAC, 2013