

Implementing a Stepper Motor Controller on an IGLOO FPGA

1.0 Introduction

The main objective of the design is to commutate a stepper motor. The control inputs are available through hardware (I/Os) or a two-wire serial interface.

2.0 General Implementation Overview

A stepper motor requires four push-pull drivers to commutate. The stepper motor requires that the motor windings be supplied with a fixed sequence of phase voltages for proper commutation. A four-wire stepper motor has two windings; one winding is powered while the current in the other winding is gradually dropped to zero, reversed, and then ramped up again for commutation. The sequence (full-step sequence or half-step sequence) and period will define the speed of commutation.

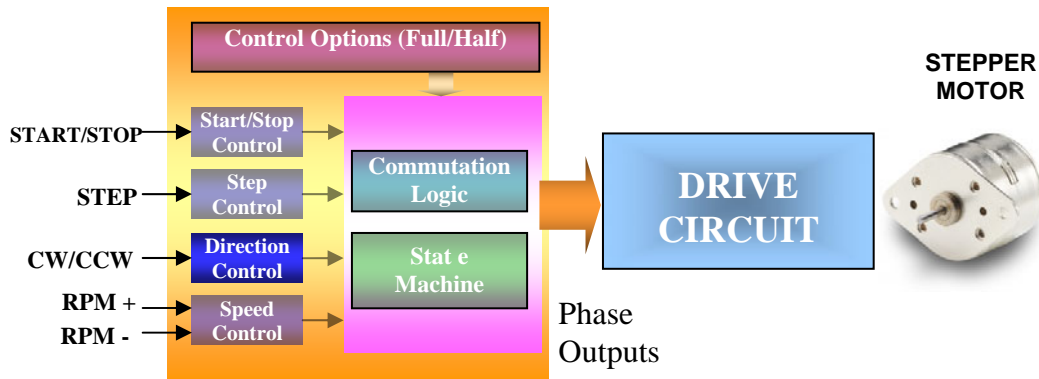


Figure 1. Stepper Motor Control on IGLOO® Device

3.0 Reference Design Implementation

Four vectored inputs are used to directly control which switches are open or closed in the push-pull stage. In some motors the inputs may be encoded while others may control subsystems such as the analog-to-digital converter in a micro-stepping interface.

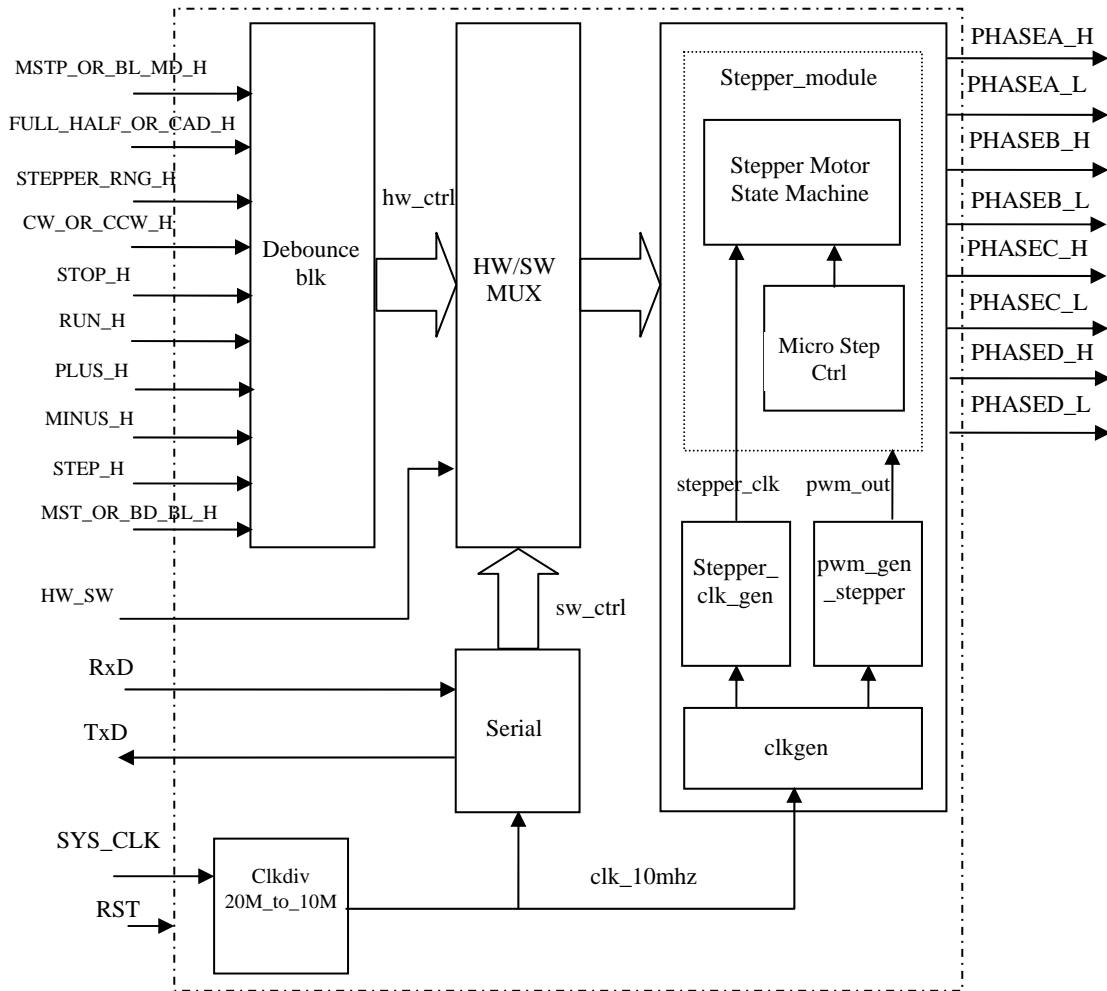


Figure 2. Stepper Motor Control – Logic Block Diagram

A control vector is defined as the state of each logic input, and control trajectory is defined by the sequence of states used to commutate the rotor. The control trajectory remains the same for both types of motors. *Note: There could be a different control trajectory for different motor designs. Please make the design changes accordingly, to conform to motor specifications.*

The control vectors required for micro-stepped motors are more complex, but the basic idea remains the same. A higher level control system is designed that will generate appropriate control trajectories, moving the motor one step, half-step, or micro-step. *This version does not support 6-Wire Stepper Motor Configuration.*

3.1 Full-Step Mode

The control trajectory for stepping though one full electrical cycle using full stepping is as follows:

| Sequence | A | C | B | D |
|----------|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 |



Clockwise

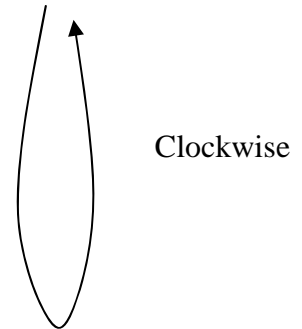
Table 1. Full-Step Stepper Motor Sequence

3.2 Half-Step Mode

The control trajectory for stepping through one full electrical cycle using half-stepping is as follows:

| Sequence | A | C | B | D |
|----------|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 0 |
| 8 | 0 | 0 | 1 | 0 |

Table 2 Half-Step Stepper Motor Sequence



3.3 Micro-Step Mode

Micro-stepping can be done in trapezoidal form with 8 or 16 steps. Micro-steps means a fraction of a full step (1/8 or 1/16). The step rate has to be increased by a corresponding factor (8 or 16) for the same RPM. Pulse width modulation (PWM) technique is used to implement micro-step mode, by varying the duty cycle of the applied voltage.

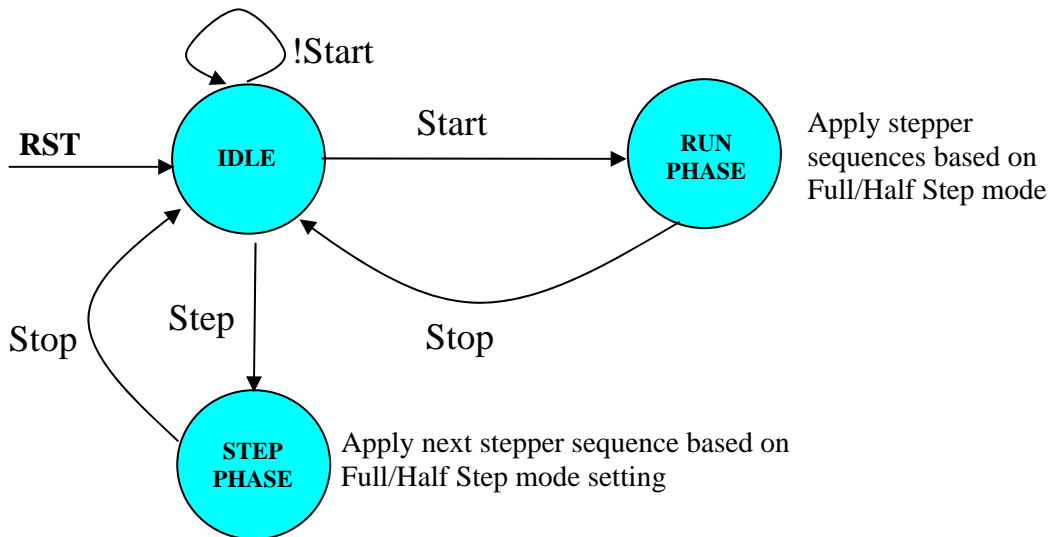


Figure 3. Stepper Motor Control State Machine

4.0 Waveforms

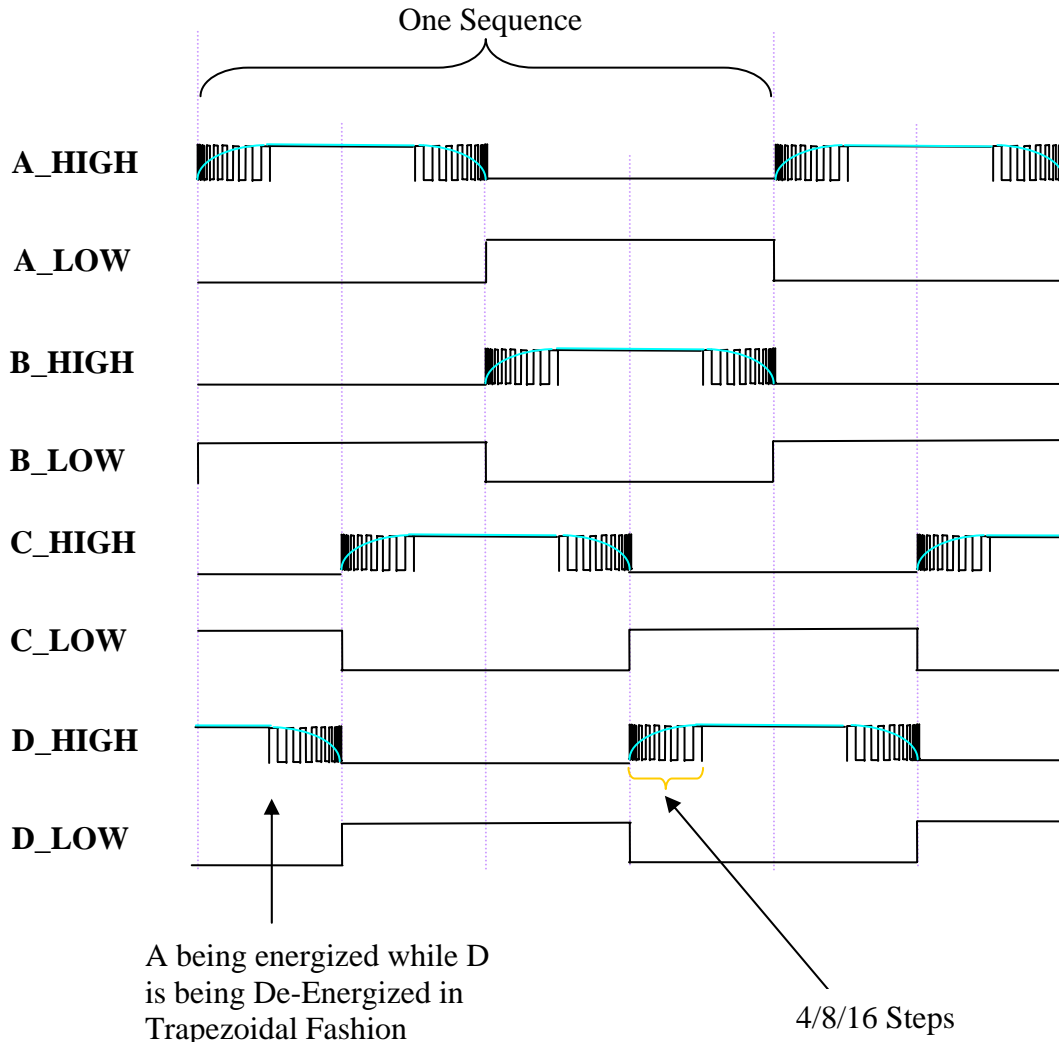


Figure 4. Trapezoidal Micro-Stepping Method

5.0 I/Os

The following table describes the main I/O's in the design.

| Signal Name | Input/ Output | Description | Fusion Pin |
|--------------------|------------------|--|------------|
| RxD | Input | RS232 Receive | C31 |
| TxD | Output | RS232 Transmit | B34 |
| CW_OR_CCW_H | Input | Motor Direction Control 1 – Clockwise 0 – Counterclockwise | A31 |
| MSTP_OR_BL_MD_H | Input | For Stepper Motor, in micro step mode No. of Micro step 0 – 8steps 1 – 16steps | A21 |
| STOP_H | Input | Motor Stop | A7 |
| MST_OR_BD_BL_H | Input | For stepper motor Microstepping ON/OFF 1 – ON, 0 – OFF | B2 |
| RUN_H | Input | Motor Start/Run | C4 |
| RANGE_SELECT_H | Input | Stepper Motor Range Select ON – 1440 RPM, OFF – 720 RPM | B37 |
| SYS_RESET | Input | System Reset (Pulse through Switch SW6) | C1 |
| PHASEA_H | Output | PhaseA – High Side Signal | C26 |
| PHASEA_L | Output | Phase A – Low Side Signal | B30 |
| PHASEB_H | Output | PhaseB – High Side Signal | A36 |
| PHASEB_L | Output | PhaseB – Low Side Signal | C32 |
| PHASEC_H | Output | PhaseC – High Side Signal | C2 |
| PHASEC_L | Output | PhaseC – Low Side Signal | A16 |
| PHASED_H | Output | PhaseD – High Side Signal | A4 |
| PHASED_L | Output | PhaseD – Low Side Signal | A20 |
| FULL_HALF_OR_CAD_H | Input | For stepper motor Step Size when microstepping off 1 – Full Step (20 steps per revolution) – 18 degrees per step 0 – Half Step (40 steps per revolution) – 9 degrees per step (If supported by motor) | A37 |
| STEP_H | Input | Step Motor in Half or Full Step depending on FULL_HALF_STEP_H | A11 |
| HW_SW | Input | Hardware or Software Control ON – Hardware, OFF – Software | A35 |
| PLUS_H | Input | Increment Speed | A23 |
| MINUS_H | Input | Decrement Speed | A26 |
| SYS_CLK | Input | Sys Clock – 20mhz | A5 |

6.0 Conclusion:

This design example allows the user to run a stepper motor using the low-power IGLOO device. The design has been specifically developed with the drive circuit in mind. Please refer to the Icycle motor control documentation for detailed usage of the IP and the features available through hardware and software.

Appendix A – Stepper Motor Controller Design Example

Design Files Summary

| Files | Functionality |
|---------------------|--|
| baud_clk_gen.v | This block generates baud clk for Serial Comm. |
| clk_by_2.v | Divides input clock by 2 - Toggle F/F. |
| clk_gen.v | Clock Generator. |
| debounce.v | Debounce Logic Block. |
| debounce_blk.v | Interconnects all debounce blocks. |
| div_by_16.v | Divide by 16 block for serial communication – baud clock. |
| divideby5.v | Derived Clock for internal use. |
| clkdiv_20M_to_10M.v | Generate 10 MHz from 20 MHz Input. |
| global.v | Defines/Parameters for the design. |
| recv_control.v | This block receives data serially on RxD. |
| serial.v | This block generates software controls for stepper motor |
| mux_hw_sw | This block multiplexes between hardware and software controls. |
| xmit_control.v | This block transmits data serially on TxD. |
| stepper_clk_gen.v | Generates required stepper clock based on +/- switch inputs. |
| pwm_gen_stepper.v | Stepper clock – PWM for Microstep mode. |
| stepper_module.v | Generate signals to drive motor . |
| top_serial.v | This block connects xmit_control, recv_control, serial, baud_clk_gen and div_by_16 . |
| top_stepper.v | Top interconnect block – pwm_gen_stepper, stepper_module and stepper_clk_gen |
| stepper_ip.v | This block interconnects top_stepper and clk_gen |
| top_stepper_ip.v | This block interconnects clkdiv_20M_to_10M and stepper_ip |
| top_tb.v | Testbench for stepper_ip |

About Ishnatek

Ishnatek offers FPGA design and hardware prototyping services. Ishnatek offers Design and Verification Services for embedded solutions. We have IPs such as 8031, RTC, Timers, Enhanced PWM, UART/SIO/IrDA, I2C, LED Driver and Key scan, Parallel Port, ECP/EPP, etc., which can be building blocks for your embedded controller solutions.