A Stepper Motor Controller in an Actel FPGA

Introduction

Stepper motors are electromechanical devices that provide accurate incremental rotation. Printer paper feeders, floppy drives, and robotic manipulators are popular stepper motor applications. The most common stepper motor uses four windings for a four-phase operation. Rotation is effected by actuating the phases in a specific sequence. Field programmable gate arrays (FPGAs) are ideal for integrating the control logic for these motors with other system control logic to minimize device count and board size.

Four-Phase Motor Circuit .

A typical four-phase motor driving circuit is shown in Figure 1 using an FPGA to generate the sequence logic. The four windings have a common connection to the motor supply voltage (V_S) which typically ranges from 5 to 30 Volts. Each of the four phases is driven by a high power NPN transistor, since the FPGA cannot drive the motor directly. Each motor phase current may range from 100 mA to as much as 10 A. The transistor selection depends on drive current, power dissipation, and gain. The series resistors should be selected to limit the FPGA current to 8 mA per output. Power MOSFET devices can also be used to drive the stepper motor

Within the Actel FPGA, four inputs are required to fully control the stepper motor. The clock (CLK) input synchronizes the logic and determines the speed of rotation. The motor advances one step per clock period; the angle of rotation of the shaft will depend on the particular motor (the angle ranges from 1.8 to 90 degrees per step). To determine the clock period, consider that the stepper motor torque increases as frequency decreases. The direction (DIR) control input changes the sequence at the outputs (PH1 to PH4) to reverse the motor direction. The enable input (EN) determines whether the motor is rotating or holding. The active low reset input (RST) initializes the circuit to ensure that the correct starting sequence is provided to the outputs.

The basic control sequence of a four-phase motor is achieved by activating one phase at a time as shown in Figure 2. Figure 3 shows an enhanced sequence that uses an overlap technique with two phases active at any one time. The enhanced sequence provides increased torque but requires twice the current.

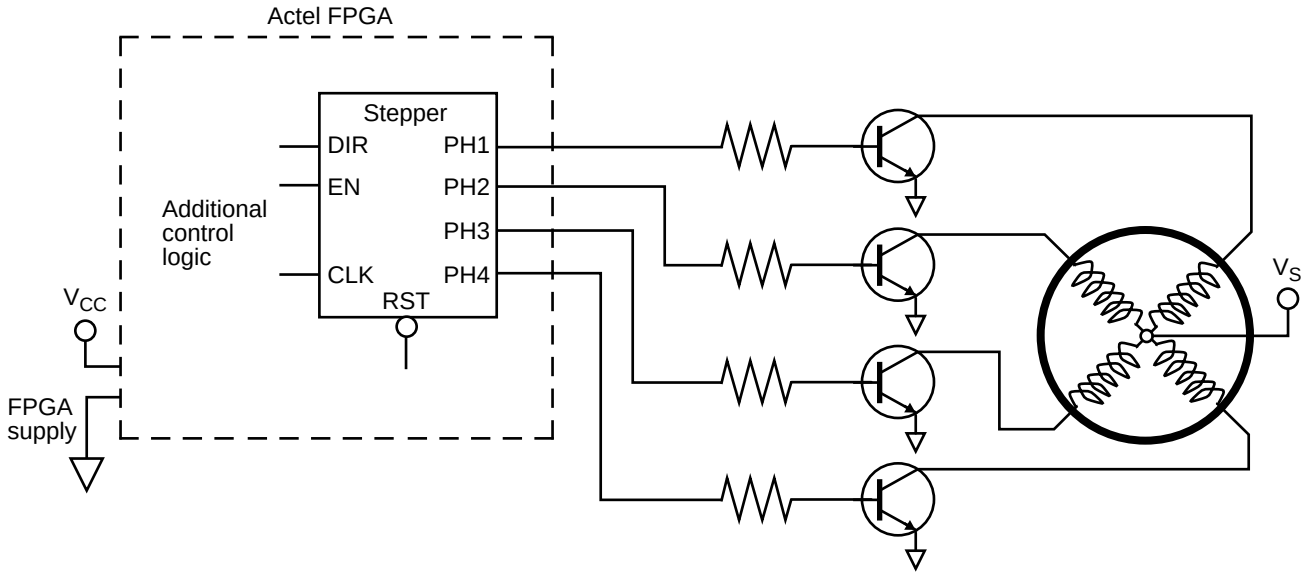

Figure 1 • *Four-Phase Stepper Motor Driving Circuit*

Step Sequence ->					
PH1	1	0	0	0	1
PH2	0	1	0	0	0
PH3	0	0	1	0	0
PH4	0	0	0	1	0

Figure 2 • *Basic Stepper Motor Sequence*

Step sequence ->					
PH1	1	1	0	0	1
PH2	0	0	1	1	0
PH3	1	0	0	1	1
PH4	0	1	1	0	0

Figure 3 • *Enhanced Stepper Motor Sequence*

Boolean equations using PALASM®2 syntax for the basic control sequence are shown in Figure 4. The phase equations (PH1 to PH4) are written with a colon and equal sign (:=) to indicate a registered implementation of the combinatorial equation. Each phase equation is either enabled (EN), indicating that the motor is rotating, or disabled (/EN), indicating that the current active phase remains on and the motor is locked. The value of the direction input (DIR) determines which product term is used to sequence clock wise or counterclockwise. The asynchronous equations (for example, ph1.setf = /rst) initialize the circuit. The Boolean equations for the enhanced motor stepper sequence shown in Figure 5 are simpler than the basic sequence. By inspection of the sequence from Figure 3, phases one and three are mere inversions of phases two and four. Therefore, two phase equations can be drastically reduced (ph1 = /ph2, ph3 = /ph4). Also, the next sequence for each phase is only dependent on one other phase and not all four, thereby reducing the number of terms required for the remaining phase two and four equations. Note that inverting phases one and three provides the correct initial sequence values.

```
CHIP step2 ACT
clk en dir rst ph1 ph2 ph3 ph4
EQUATIONS
ph2 := en * /dir * ph4 + en * dir * ph3 + /en * ph2
ph4 := en * /dir * ph1 + en * dir * ph2 + /en * ph4
ph2.rstf = /rst
ph4.rstf = /rst
ph1 = /ph2
ph3 = /ph4
```

Figure 5 • *Boolean Equations for Enhanced Stepper Motor Sequence*

```
CHIP step1 ACT
clk en dir rst ph1 ph2 ph3 ph4

EQUATIONS

ph1 := /dir * en * (/ph1 * /ph2 * /ph3 * ph4)
     + dir * en * (/ph1 * ph2 * /ph3 * /ph4)
     + /en * ph1

ph2 := /dir * en * ( ph1 * /ph2 * /ph3 * /ph4)
     + dir * en * (/ph1 * /ph2 * ph3 * /ph4)
     + /en * ph2

ph3 := /dir * en * (/ph1 * ph2 * /ph3 * /ph4)
     + dir * en * (/ph1 * /ph2 * /ph3 * ph4)
     + /en * ph3

ph4 := /dir * en * (/ph1 * /ph2 * ph3 * /ph4)
     + dir * en * ( ph1 * /ph2 * /ph3 * /ph4)
     + /en * ph4

ph1.setf = /rst
ph2.rstf = /rst
ph3.rstf = /rst
ph4.rstf = /rst
```

Figure 4 • *Boolean Equations for Basic Stepper Motor Sequence*

The Boolean files were synthesized and optimized for the Actel ACT™ 2 family. Modules utilized for the basic and enhanced sequences were 14 and 6 modules respectively. Figures 6 and 7 are the schematic representations of the basic and enhanced sequencing circuits respectively. Note that these schematics do not show the output I/O macros (OUTBUF) that are needed to connect to the external circuit (that is, outside the FPGA).

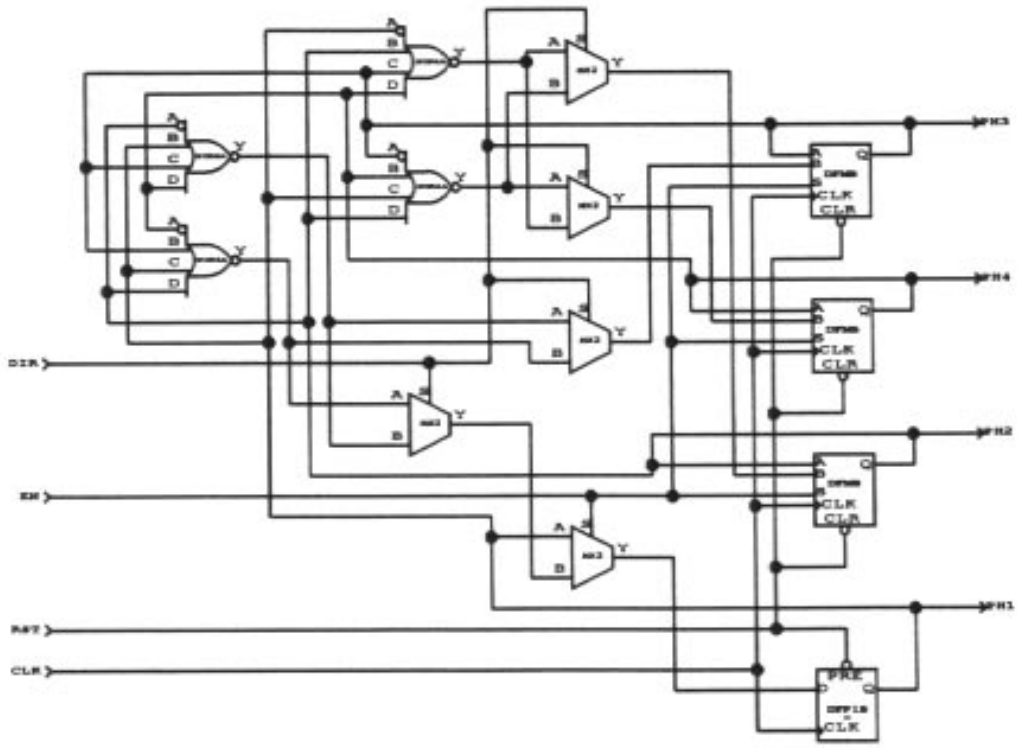

Figure 6 • *Basic Stepper Motor Sequence Schematic*

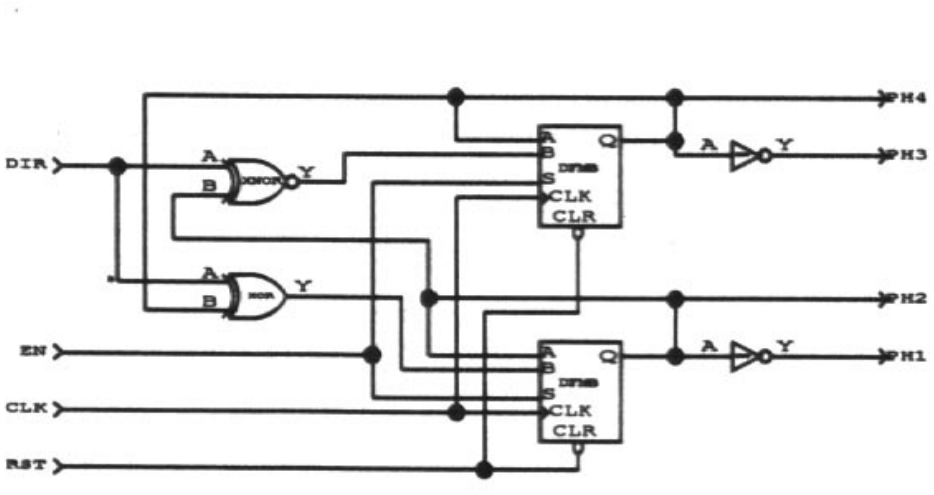

Figure 7 • *Enhanced Stepper Motor Sequence Schematic*

The timing diagrams in Figures 8 and 9 show the operation of the two stepper motor sequences. Reset, enable, and direction inputs are exercised for a clear understanding of full circuit operation. Additional logic can be added to control the enable and direction inputs. For example, a pre-loadable down counter with a zero detect output connected to the stepper motor enable input can be used to rotate the motor a predetermined number of steps.

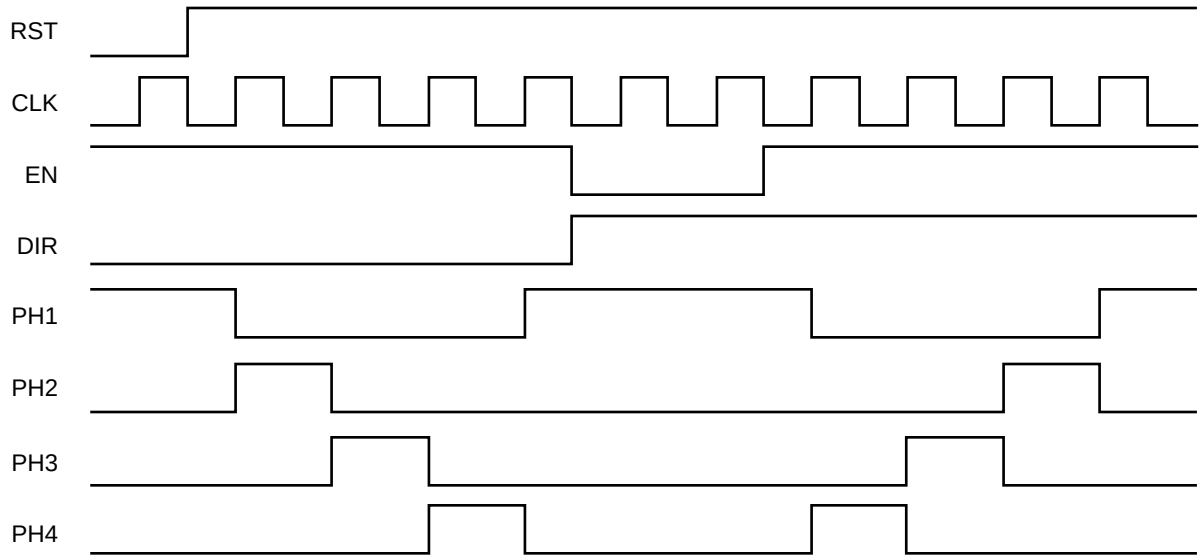

Figure 8 • *Basic Stepper Motor Sequence Timing Diagram*

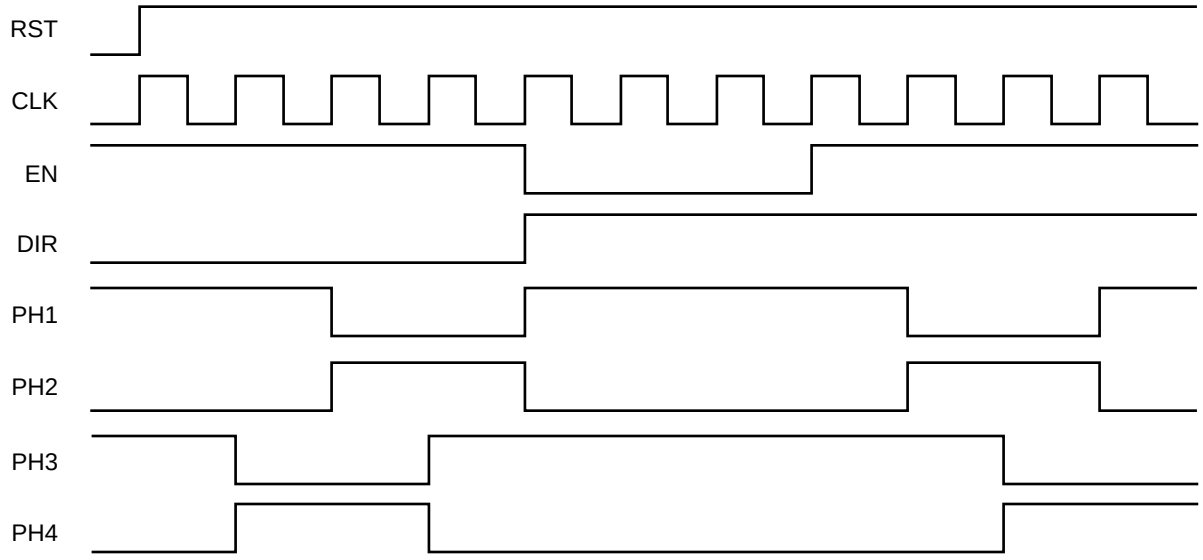

Figure 9 • *Enhanced Stepper Motor Sequence Timing Diagram*