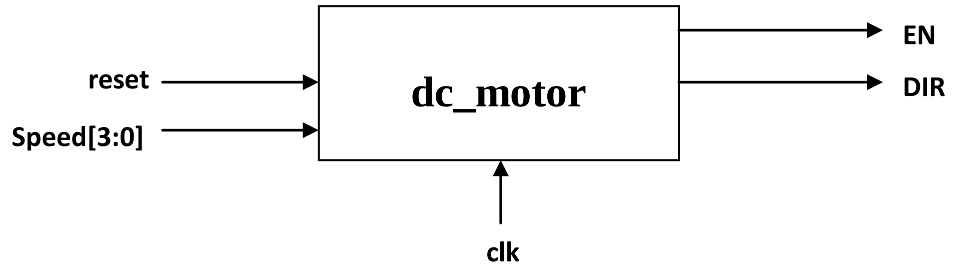
HDL code to control speed, direction of DC motor

External View:

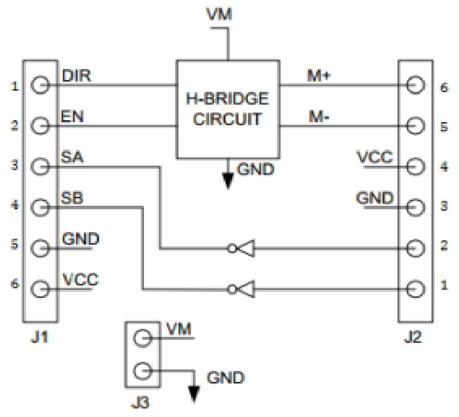


Truth Table:

clk	reset	EN	DIR	Motor Direction
↑	0	1	0	Clockwise
↑	1	1	1	Anticlockwise

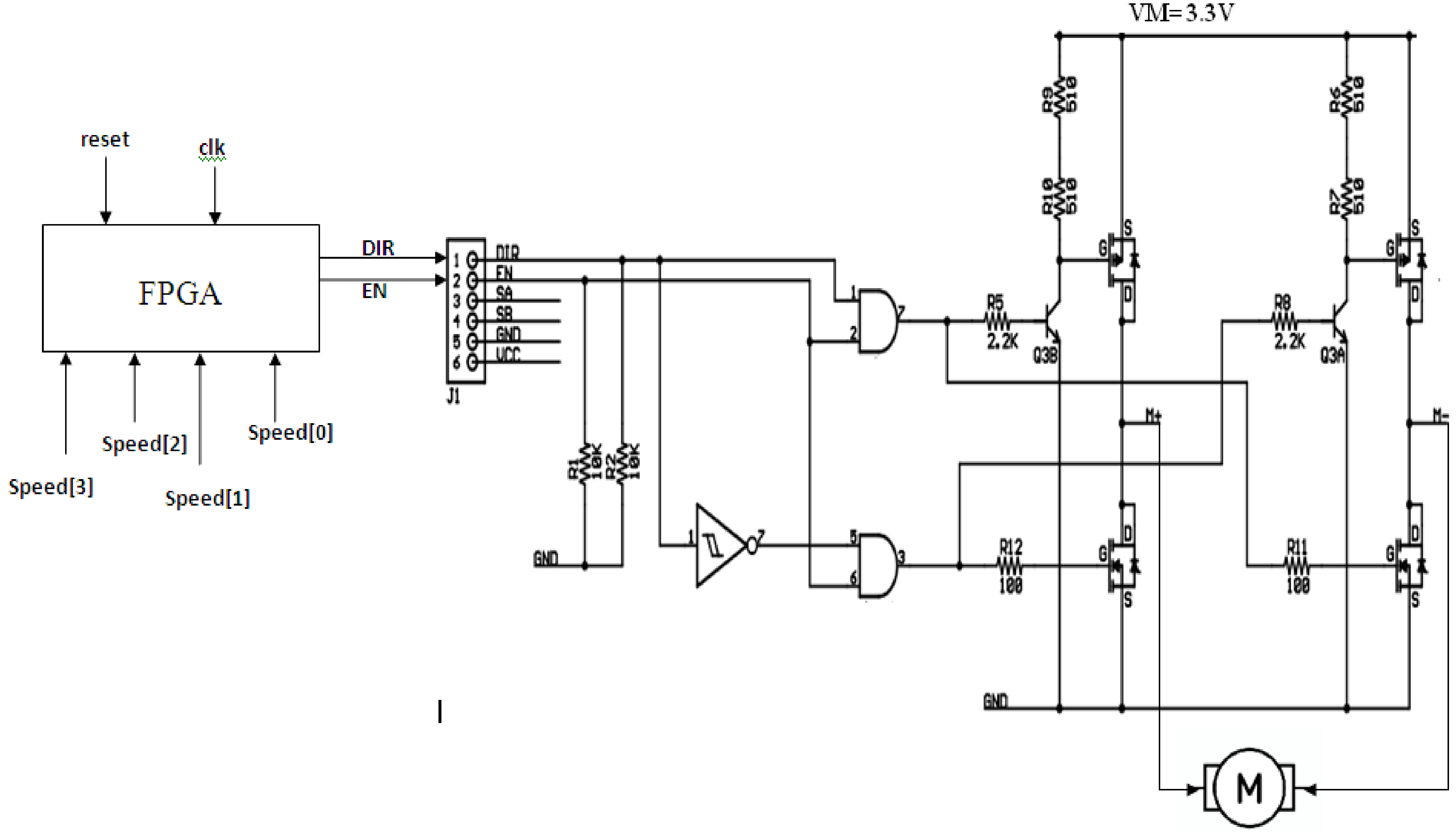
Speed of Motor		
Speed[3:0]	Duty Cycle	Remark
1110	10%	Lowest Speed
1101	25%	
1011	50%	
0111	75%	Highest Speed

Interfacing Diagram:



PmodHB5 block diagram (top-down view)

Any external power applied to the PmodHB5 must be within 2.7V and 5.25V; however, it is recommended

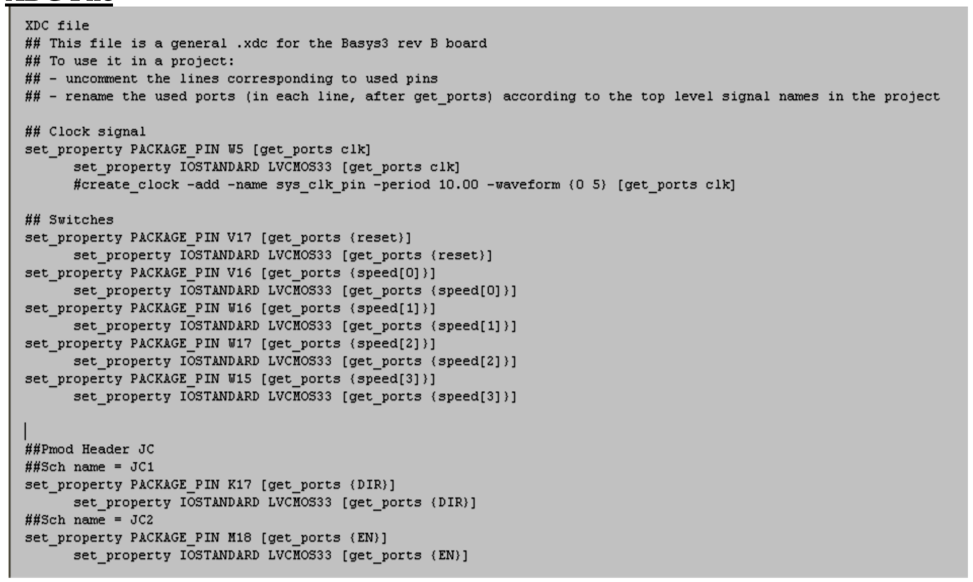


VM= 3.3V
reset
clk
FPGA
DIR
EN
Speed[3]
Speed[2]
Speed[1]
Speed[0]
J1
DIR
EN
SA
SB
GND
VCC
R1 10K
R2 10K
GND
R9 510
R10 510
R6 510
R7 510
R5 2.2K
R8 2.2K
Q3B
Q3A
R12 100
R11 100
M+
M-
GND
M

VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity  dc_motor is
Port(EN,DIR : out std_logic; reset:in std_logic;
    clk: in std_logic;
    speed: in std_logic_vector(3 downto 0));
end dc_motor;

architecture dc_motor1 of dc_motor is
signal clk_div : std_logic_vector(25 downto 0);
signal clk_int: std_logic;
signal onperiod: integer range 0 to 100 :=0;
signal duty_cycle: integer range 0 to 100;
begin

process(clk)
begin
if rising_edge (clk) then
clk_div <= clk_div + '1';
end if;
end process;
clk_int<=clk_div(1);

process(speed)
begin
case speed is
when "1110" => duty_cycle <= 90;
when "1101" => duty_cycle <= 75;
when "1011" => duty_cycle <= 50;
when "0111" => duty_cycle <= 25;
when others => duty_cycle <= 10;
end case;
end process;
process(clk_int)
begin
if rising_edge(clk_int) then
onperiod <= onperiod +1;
end if;
end process;

Process(onperiod)
begin
if onperiod>=duty_cycle then
if reset='0' then
EN <='1';DIR<='0';
else
EN<='1';DIR<='1';
end if;
end if;
end process;
end dc_motor1;
```

XDC File



```
XDC file
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
      set_property IOSTANDARD LVCMOS33 [get_ports clk]
      #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {reset}]
      set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN V16 [get_ports {speed[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {speed[0]}]
set_property PACKAGE_PIN W16 [get_ports {speed[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {speed[1]}]
set_property PACKAGE_PIN W17 [get_ports {speed[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {speed[2]}]
set_property PACKAGE_PIN W15 [get_ports {speed[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {speed[3]}]


##Pmod Header JC
##Sch name = JC1
set_property PACKAGE_PIN K17 [get_ports {DIR}]
      set_property IOSTANDARD LVCMOS33 [get_ports {DIR}]
##Sch name = JC2
set_property PACKAGE_PIN M18 [get_ports {EN}]
      set_property IOSTANDARD LVCMOS33 [get_ports {EN}]
```

Theory

The Digilent **PmodHB5** offers a 2A H-bridge circuit to drive small to medium sized DC motors. This module was specifically designed to work with the Digilent gearbox motor, which incorporates quadrature encoder feedback.

Features include:

- 2A H-bridge circuit
- Drive a DC motor with operating voltage up to 12V
- 6-pin JST connector for direct connection to Digilent motor/gearboxes
- Two screw terminals for external motor power supply
- Small PCB size for flexible designs 1.2 in × 0.8 in (3.0 cm × 2.0 cm)
- 6-pin Pmod connector with GPIO interface

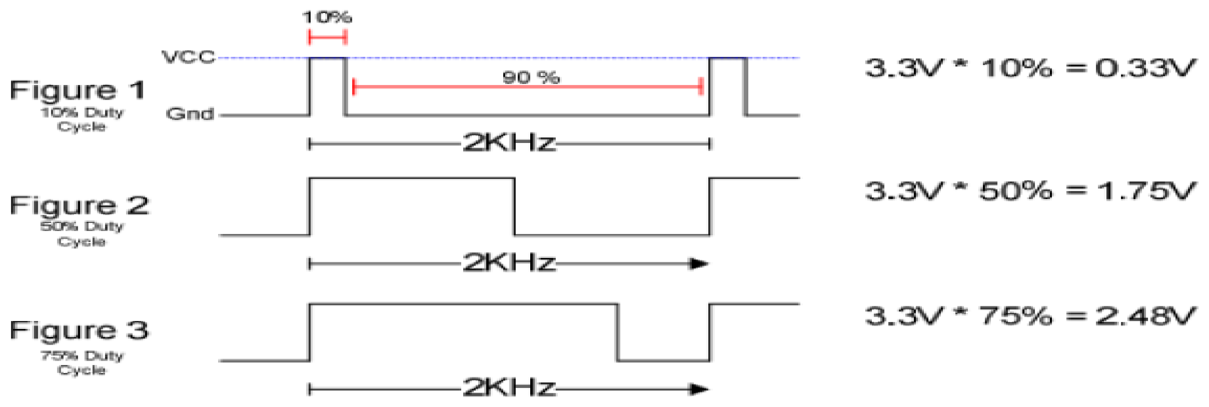
1 Functional Description

The PmodHB5 utilizes a full H-Bridge circuit to allow users to drive DC motors from the system board. Two sensor feedback pins are incorporated into the motor connection header and are specifically designed to work with the Digilent motor/gearbox.

2 Interfacing with the Pmod

The PmodHB5 communicates with the host board via the GPIO protocol. Like all H-Bridges, care must be taken to avoid causing a potential short within the circuitry. In terms of this Pmod, this means that the Direction pin must not change state while the Enable pin is at a high voltage state. If this does occur, one set of switches that are driving the motor will be closing while the other set is opening, allowing for the possibility for both sets of switches to be open at the same time, creating a short.

To drive the motor at a specific speed, users will need to choose a static direction (forwards or backwards corresponding to high or low voltage) on the Direction pin, and then perform pulse width modulation on the Enable pin. The more often that an enable pin is driven high within a set time frame, the faster the DC motor will spin.



The way that this works is that when voltage is being applied, the motor is driven by the changing magnetic forces. When voltage is stopped, momentum causes the motor to continue spinning a while. At a high enough frequency, this process of powering and coasting enables the motor to achieve a smooth rotation that can easily be controlled through digital logic.

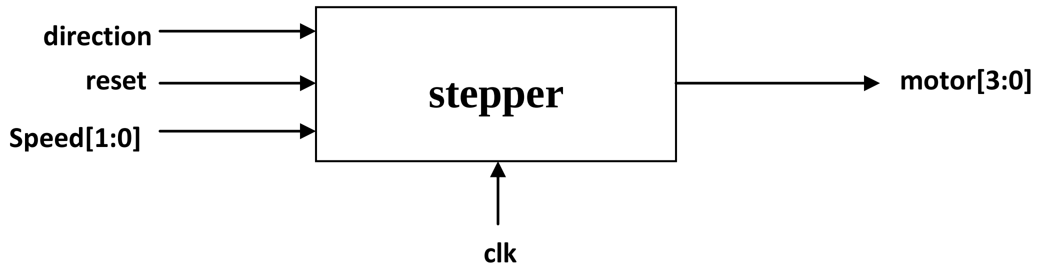
Header J1 (pin 1 on the top)		
Pin	Signal	Description
1	DIR	Direction pin
2	EN	Enable pin
3	SA	Sensor A feedback pin
4	SB	Sensor B feedback pin
5	GND	Power Supply Ground
6	VCC	Positive Power Supply (3.3/5V)

Header J2 (pin 1 on the bottom)		
Pin	Signal	Description
1	SB	Sensor B feedback pin
2	SA	Sensor A feedback pin
3	GND	Power Supply Ground
4	VCC	Positive Power Supply (3.3/5V)
5	M+	Motor positive pin
6	M-	Motor negative pin

Table 1. Pinout description table.

HDL code to control speed, direction of Stepper Motor

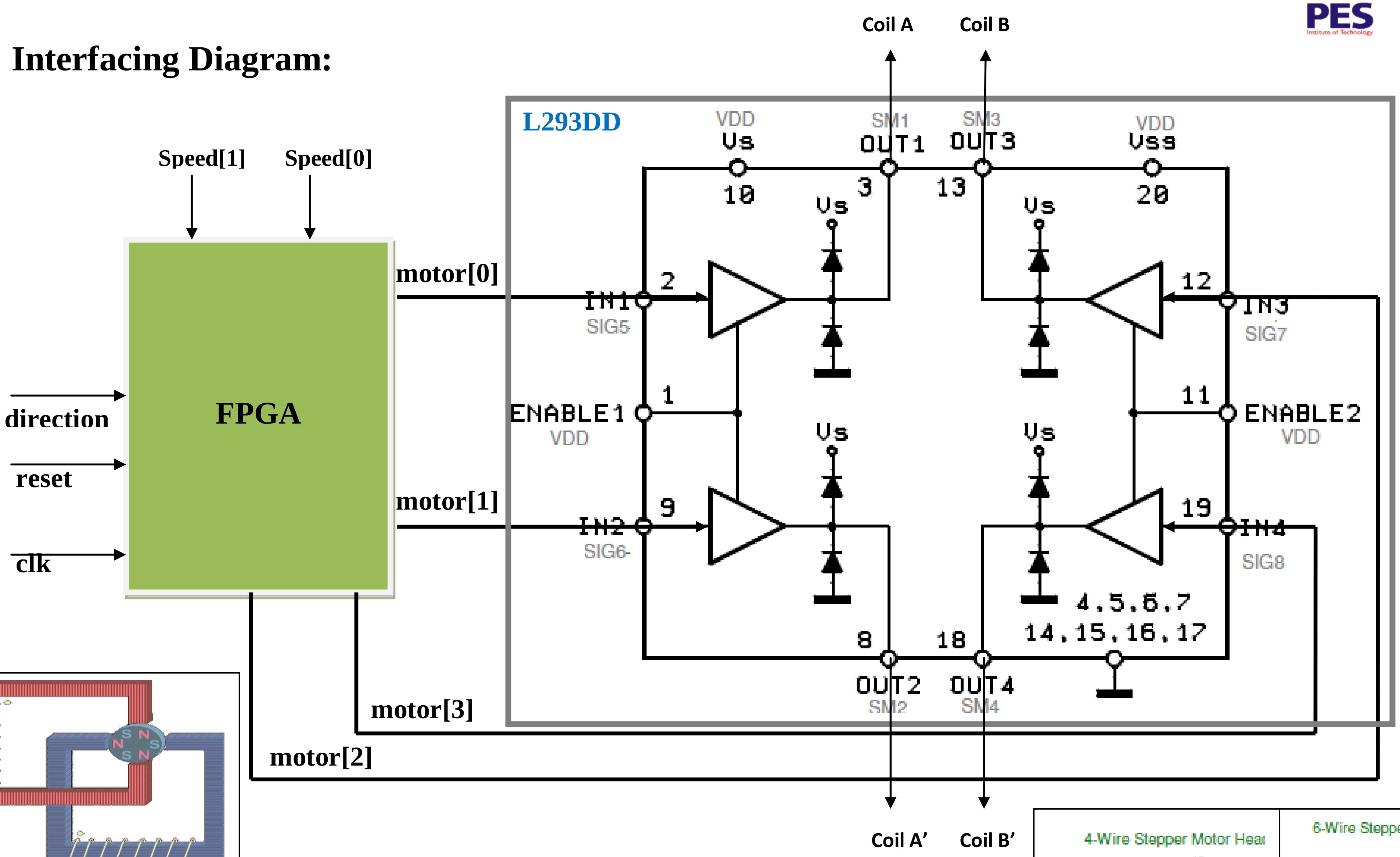
External View:

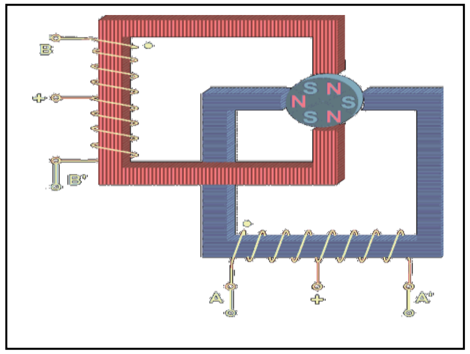


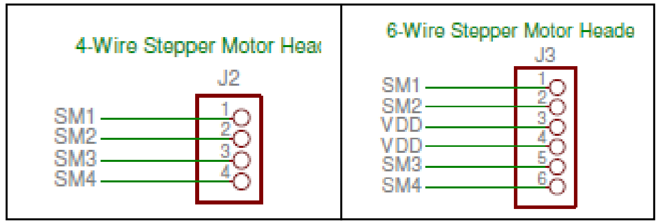
Truth Table:

Speed of motor			reset	clk	Direction	motor[3:0]	Remark
speed[1:0]	clk	Remark					
00	clk_div(25)	Lowest	0	X	X	1001	
01	clk_div(23)		1	↑	0	1100	Clock wise
10	clk_div(21)			↑		0110	
11	clk_div(19)	Highest		↑		0011	
				↑		1001	
				↑	1	0011	Anticlockwise
				↑		0110	
				↑		1100	
				↑		1001	

Interfacing Diagram:



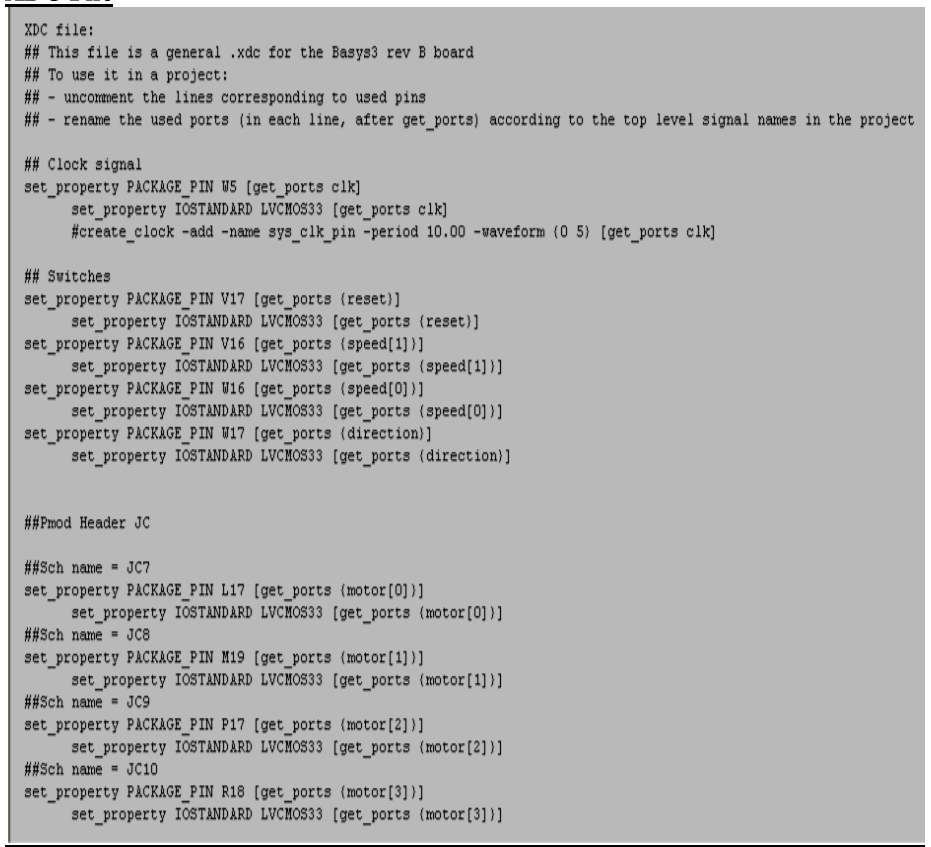




VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity stepper is
    Port ( motor: out std_logic_vector(3 downto 0);
             clk,reset: in std_logic;
             speed:in std_logic_vector(1 downto 0);
            direction:in std_logic);
end stepper;

architecture stepper of stepper is
signal clk_div : std_logic_vector(25 downto 0);
signal clk_int: std_logic;
signal angle : std_logic_vector(3 downto 0):="1001";
begin

process(clk)
begin
if rising_edge (clk) then
clk_div <= clk_div + '1';
 end if;
end process;

clk_int<=clk_div(25) when speed ="00"else
                clk_div(23) when speed ="01"else
                clk_div(21) when speed ="10"else
                 clk_div(19) ;


process(reset,clk_int,direction)
begin
 if reset='0' then
             angle <= "1001";
elsif rising_edge(clk_int) then
     if direction='0' then
     angle <= angle(0) & angle(3 downto 1);
    else
    angle<=angle(2 downto 0) & angle(3);
    end if;
end if;
end process;
motor<=angle;
end stepper;
```

XDC File

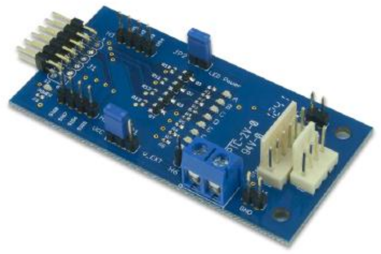


```
XDC file:
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
      set_property IOSTANDARD LVCMOS33 [get_ports clk]
      #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {reset}]
      set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN V16 [get_ports {speed[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {speed[1]}]
set_property PACKAGE_PIN W16 [get_ports {speed[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {speed[0]}]
set_property PACKAGE_PIN W17 [get_ports {direction}]
      set_property IOSTANDARD LVCMOS33 [get_ports {direction}]


##Pmod Header JC

##Sch name = JC7
set_property PACKAGE_PIN L17 [get_ports {motor[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {motor[0]}]
##Sch name = JC8
set_property PACKAGE_PIN M19 [get_ports {motor[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {motor[1]}]
##Sch name = JC9
set_property PACKAGE_PIN P17 [get_ports {motor[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {motor[2]}]
##Sch name = JC10
set_property PACKAGE_PIN R18 [get_ports {motor[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {motor[3]}]
```

Theory

Overview

The Digilent **PmodSTEP** can drive either a 4-pin or 6-pin stepper motor.



The PmodSTEP.

Features include:

- Utilizes a ST L293DD channel driver
- Capable of running both a 4 and 6 pin stepper motor simultaneously
- Multiple LEDs to indicate signal propagation

1 Functional Description

The PmodSTEP utilizes ST's four channel driver, a L293DD, to drive stepper motors at higher currents than a system board can typically provide from their logic outputs. External test point headers and LEDs are provided for easy testing and observation of the propagation of signals.

2 Interfacing with the Pmod

The PmodSTEP communicates with the host board via the GPIO protocol.

This Pmod offers headers for both 4-pin and 6-pin stepper motors. Stepper motors work by alternately energizing the coils to different polarities inducing the stepper motor to rotate.

4-pin stepper motors only work in the bipolar configuration, requiring that the two inputs on each electromagnetic coil are brought to the correct logic level voltages to induce current flow in the correct direction. The 6-pin stepper motor header on this Pmod can be oriented for either bipolar or unipolar configuration. The two extra pins on this header provide two positive power pins as a source of current for when an input on one end of a coil is driven to a logic low voltage level.

Pin	Signal	Description
1	SIG1	Signal 1
2	SIG2	Signal 2
3	SIG3	Signal 3
4	SIG4	Signal 4
5	GND	Power Supply Ground
6	VCC	Positive Power Supply
7	SIG5	Signal 5/Output 1 for the Stepper Motor
8	SIG6	Signal 6/Output 2 for the Stepper Motor
9	SIG7	Signal 7/Output 3 for the Stepper Motor
10	SIG8	Signal 8/Output 4 for the Stepper Motor
11	GND	Power Supply Ground
12	VCC	Positive Power Supply

Table 1. Pinout description table.

Any external power applied to the PmodSTEP must be within 4.5V and 36V; it is recommended that Pmod is operated at 5V.