Interfacing Analog and Digital Worlds

LAB2: Step Motor Controller

2024

Rawan shalata 211864608
Marsel nasr 205728702

instructors:
Gabriel Zini

David Madar

intro

In this lap we learned about step motor and we build a simple controller that will control the motor according to switches and buttons that we will mention later.

Stepper motors are small synchronous motors, built from a stator with a certain number of poles, and a rotor whose shape depends on the type of motor.

Mainly the two most useful types of stepper motors are Unipolar and Bipolar According to the data sheet of the HB_Stepper_Motor_E (1).pdf we found that our motor is a Bipolar type because it has 4 wires. And so we will explain about Bipolar's engine.

A stepper motor consists of several coils arranged in a circuit. When a current pass through one of the coils, it becomes magnetic (electromagnetism) and thus pulls the motor head in its direction.

Driving a stepper motor using a normal stepper

Step number	Coil 1	Coil 2	Coil 3	Coil 4
1	On	Off	Off	Off
2	Off	On	Off	Off
3	Off	Off	On	Off
4	Off	Off	Off	On

Stepper motor drive using half-steps

This method makes it possible to achieve a precision that is twice as high as the engine's basic capacity. For example, for a motor that moves 1.8 degrees for each step, you can reach a movement of 0.9 degrees.The disadvantage is that in half of the steps the current consumption is double the current consumption in the "full step" method.

Driving a stepper motor using a half stepper

Step number	Coil 1	Coil 2	Coil 3	Coil 4
1	On	Off	Off	Off
2	On	On	Off	Off
3	Off	On	Off	Off
4	Off	On	On	Off
5	Off	Off	On	
6	Off	Off	On	On
7	Off	Off	Off	On
8	On	Off	Off	On

bipolar motor:

A bipolar motor has a single winding per phase and requires current to be reversed to change the magnetic field direction, necessitating a more complex control circuit like an H-bridge. This configuration allows the entire winding to be energized at any time, resulting in higher torque and efficiency. Bipolar motors usually have four wires for two phases, though some versions have six wires, with only four used in bipolar operation.

unipolar motor:

In contrast, a unipolar motor has center-tapped windings, effectively splitting each winding into two halves. This design simplifies control since current only needs to flow in one direction, making unipolar motors easier to drive with a simpler switching circuit. However, only half of the winding is energized at any time, leading to lower torque and efficiency. Unipolar motors typically have five or six wires, with the center tap providing a common connection. While bipolar motors are more efficient and powerful, unipolar motors are favored for their simpler control requirements.

The meaning of each switch and button in the system:

1- Let's start with the first switch, SW1, which is responsible for the direction of the motor's movement. When SW1 is held at one, the motor will rotate clockwise, and when it is held at zero, it should move counterclockwise. This is handled by giving it the position of the previous state when SW1 is held at one. When SW1 is held at zero, we give it the next state.

2- The system will have SW3 to determine the size of the step, in case SW3 is held at one the motor will make a normal step, and when it is held at zero it will make half a step, in this case we activated the two coils so that each will pull the magnet from the other side and then the magnet will be caught in the middle.

3- The system will include SW2 which, if it is held at one, the motor will rotate continuously, and when it is held at zero, the motor will make a quarter turn in the event that KEY1 is pressed.

4- The speed of the motor was controlled by KEY3, the initial speed of the system is 10 spins per minute, and each press the speed will increase by 10 spins per minute until it reaches speed 60 and then each press after that will cause a decrease of 10 spins per minute until it reaches speed 10 and start to increase in speed again.

Design of the engine:

Entrances:

- KEY0: System reset
- CLK: The system clock
- SW1: button of the direction, when it is at 1 the direction will be clockwise and when at zero it will be counter-clockwise.
- SW2: Determines the mode of operation of the motor, that is, at 1 it operates continuously and at zero it performs a quarter turn after each press of KEY1.
- SW3: determines the size of the step that the motor takes, when it is held at 1 it makes a normal step and when it is at zero it makes a half step.
- KEY1: with each press, the motor makes a quarter turn, provided that SW2 is held at 0.
- KEY3: speed button, the initial speed of the system is 10 spins per minute, that is, during the reset, the speed is 10, and each time you press it (pulse decrease from 1 to 0) the speed will increase by 10 spins per minute until you reach speed 60, then each press after that will cause a decrease of 10 spins for a minute until you reach speed 10 and start speeding up again.

Outs:

Our engine will have 3 outputs, two outputs that represent the speed on the screen and one output which is the command given to the engine (pulses_out), this command is of size 4 so that it will contain a certain binary sequence.

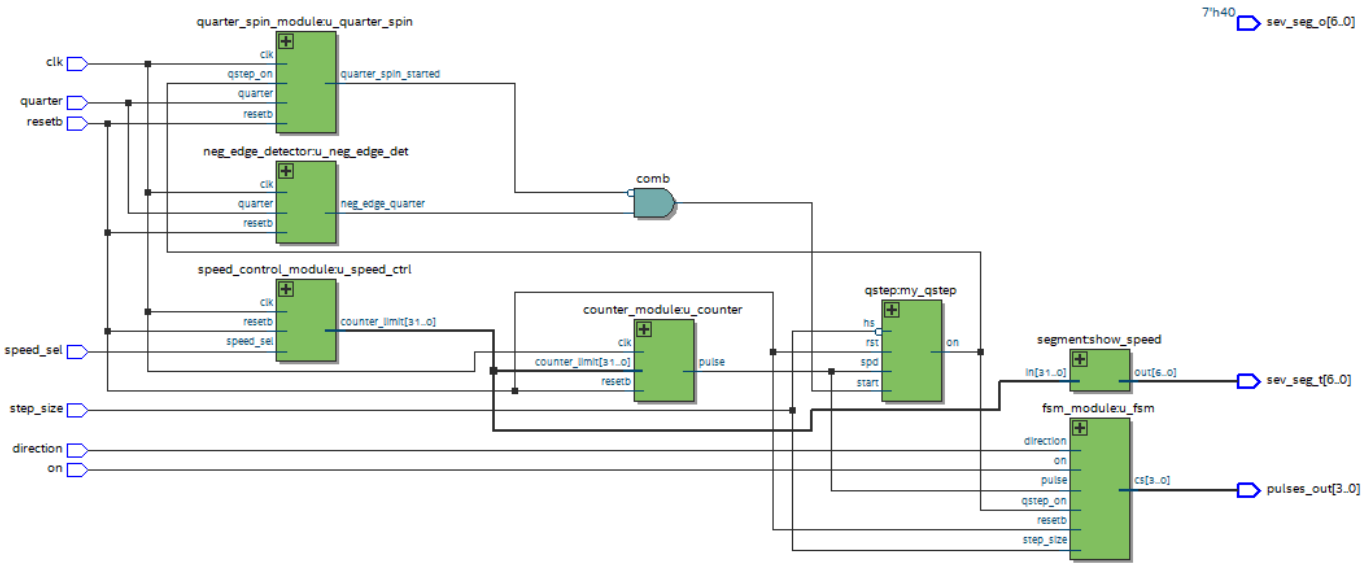
HEX0: which represents the unity digit of the speed which is basically always zero and is of size 7

HEX1: which represents the tens digit of the speed and it is size 7.

Top level :

We have 7 blocks in the top level and each one does a different job.

Picture of the top level:

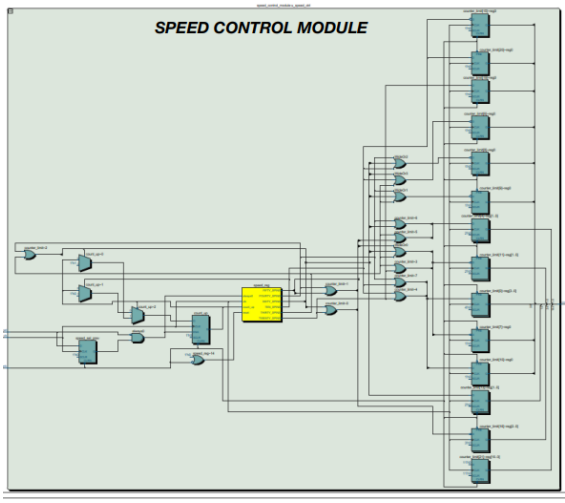


Units:

- Speed Controller:

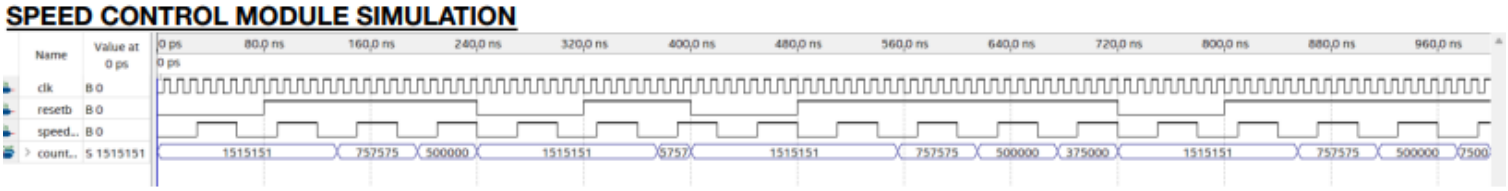
This unit receives at the entrance:

KEY0 Reset the system, KEY3 Change speed which is the button responsible for increasing and decreasing the speed, meaning that every time you press the button, it changes the speed as we explained above in the inputs.

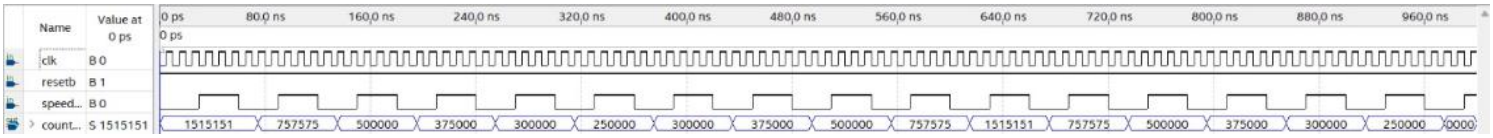


This unit has one output which is actually the port that contains the current speed of the motor for example 10 20 30 40 50 60, and we used the same output for the segment in order to display the speed on the screen.

In order to check the function, we did the following simulation:



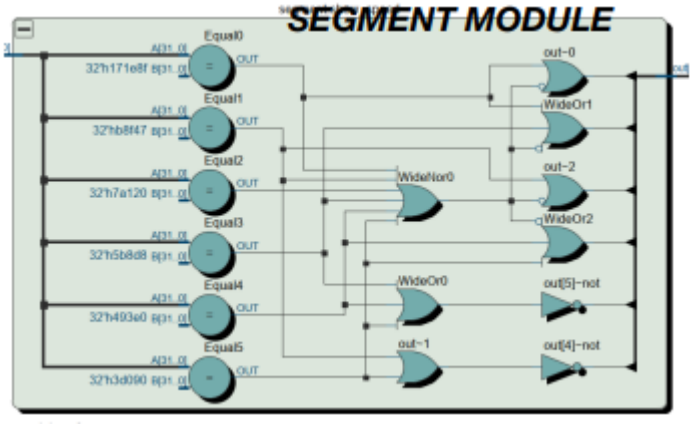
We can see from the simulation that the function works as it should. You can see that the initial speed was 10 spins and the reset is held at 0 therefore the in would not affect the output because our reset returns the speed to 10, and indeed as the in increases the speed increases by 10 spins per minute, it continues Increase with each increase to in.
We have another simulation when the reset is of for a long time to see that the speed after getting to 60 spins per minute it goes to 50 spins per a minute:



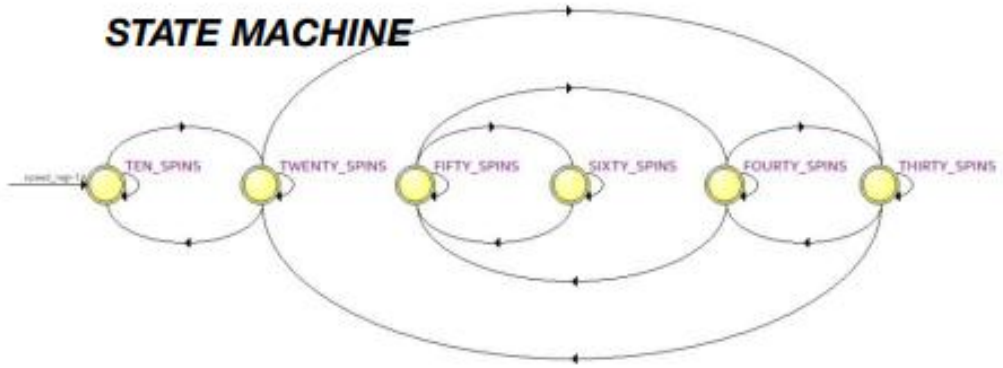
So the function works perfectly!

- Seven segments

The segment has two outputs HEX0 and HEX1 which shows to us the current speed we have on the board and HEX0 is always 0. when giving a reset HEX1 is 1 and for every input KEY3 the HEX1 goes from 1 to 6 and then continue from 6 to 1 like we explained before (giving that the reset is zero).



- Finite state machine



The unit actually checks several things and outputs pulses at a frequency corresponding to the speed so that for each pulse the motor has to take one step, it always give an output off zero and as soon as the motor needs to take a step it outputs a peak, that is, output=1. We created this unit using a state machine, meaning we took all the cases we have so that each case presents one of the possible speeds for us, and we calculated for each speed how many CYCLES we need to count in order to issue a pulse.

We calculated the number of CYCLES that need to be performed for each speed according to the structure of the engine, let's make an example:

Speed of 10 rounds per minute:

We take a complete step and according to the table we have in the model we know that a complete step is actually 1.8 degrees

Full step 360/1.8=200

The speed is 10 spins per minute, so he needs to take 200*10=2000 steps

The frequency of the system clock is 50 MHZ, which means it makes 50 million increments per second, and we multiplied this number by 60 seconds , and then at the end we divided by the number of steps needed according to our speed, in our case 2000 steps, and then we got at the end how many clock increments we need to count in order to output a pulse in the example our:

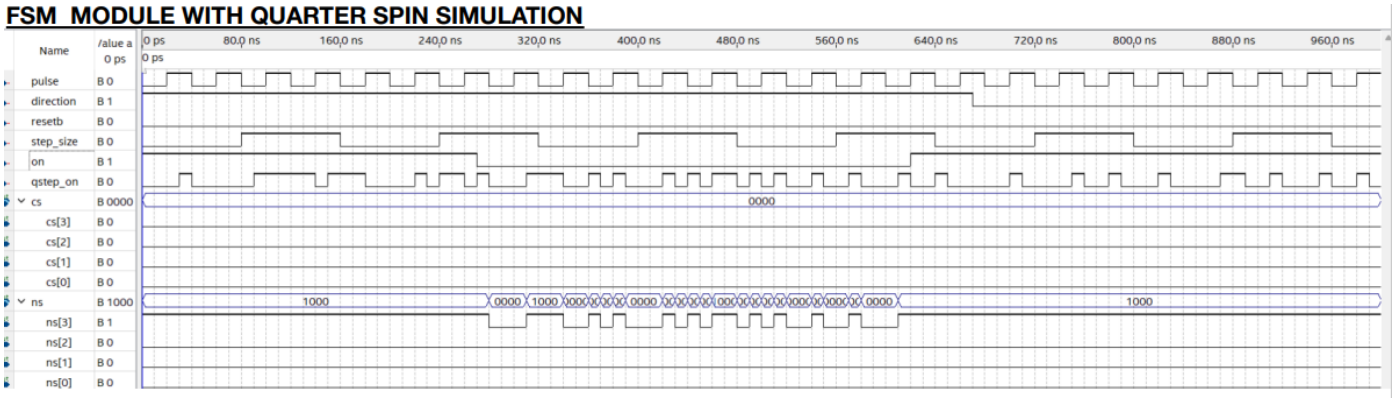
60*50000000=3000000000

3000000000/2000= 1500000

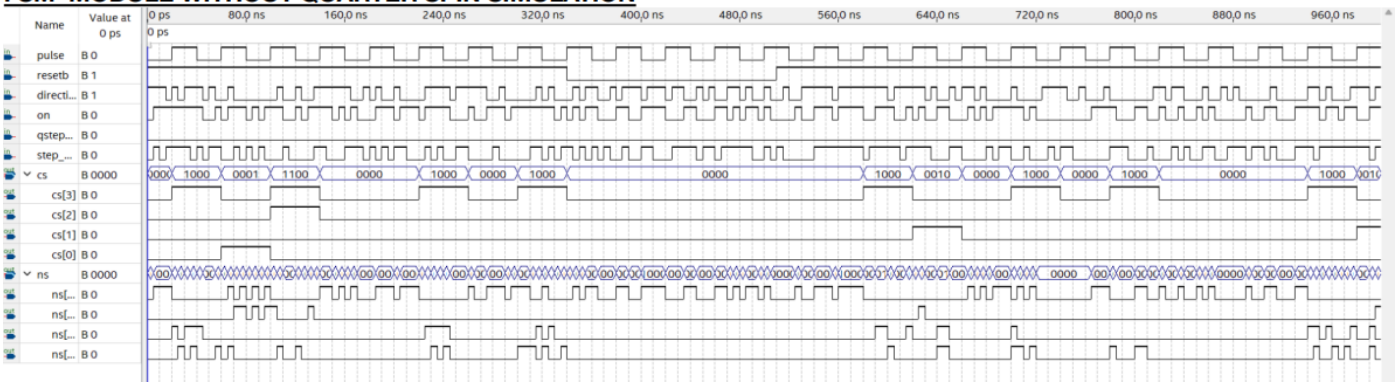
is for speed 10 and for a whole step, and the same thing is 1500000 calculated for the whole speed, and if it is half a step, simply multiply the number of steps needed according to the speed by 2, in our case 4000 .

in order to check the function, we did the following simulation:

FSM MODULE WITH QUARTER SPIN SIMULATION



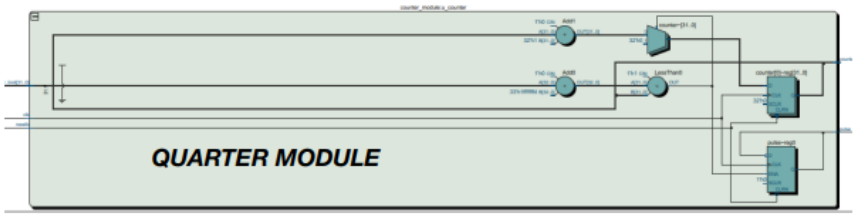
FSM MODULE WITHOUT QUARTER SPIN SIMULATION

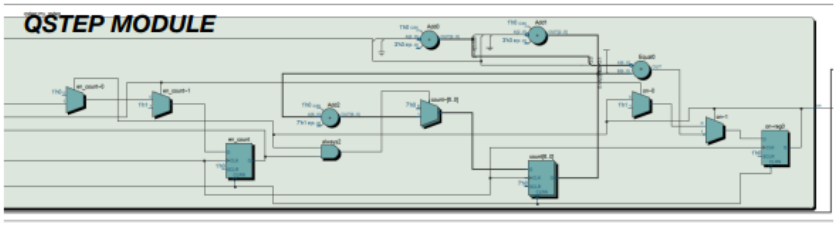


when the on is high and the quarter is also high the motor does not move a quarter spin but when the on is low we get a quarter spin if the quarter is high and when the on is high we go throw the states correctly.

- quarter spin

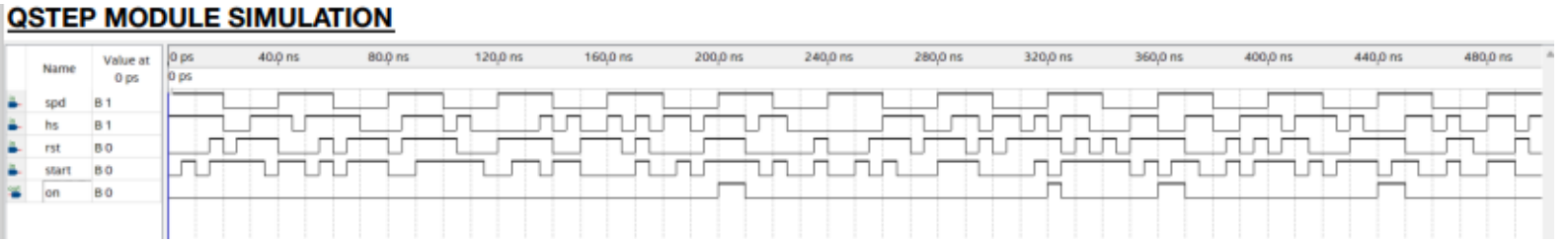
There is two units for this the quarter spin and the q step. The q step unit is responsible for the quarter turn case. KEY1, which is the button responsible for making a quarter circle, meaning that every time you press it (a pulse from 1 to 0), we start to move a quarter circle (only if SW2=0). We said



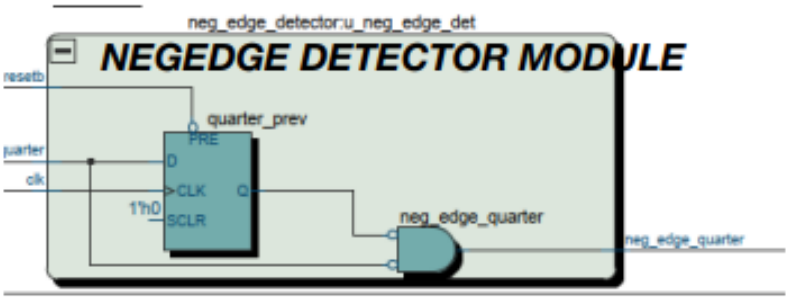


before that a complete spin is 200 clock raise, therefore to make a quarter circle we will take 50 clock raise if SW3 = 1 or 100 clock raise if SW3 =0. Every time we press KEY1 we start counting to 50 or 100 clock raise which is a quarter spin. The quarter spin unit makes sure that if we press KEY1 more than one time before finishing a quarter spin it will still do one exact quarter spin not more.

we did the following simulation:



- negative edge detector



This unit is designed to detect the falling edge of the input signal quarter. It operates by using a clock signal and an active-low reset signal. Internally, it maintains a register to store the previous state of the `quarter` signal. On each rising edge of the clock, the module updates `quarter_prev` with the current state of `quarter`, or resets it if `resetb` is low. The module outputs a signal which is high only when a negative edge (transition from high to low) is detected on the `quarter` signal and then the motor begin doing a quarter spin with consideration to the step size and direction.

- counter

The counter module plays a crucial role in timing and frequency control within a digital system. By counting clock cycles and toggling the pulse output at a specified limit, it generates a pulse signal with a controllable frequency. This functionality help us manage the stepper motor pulses in the step motor top module.

Video of the motor and the board:

https://drive.google.com/file/d/1mPK3cCsCBrTO6Z6PHUh9RFMxDq8YPn4j/view?usp=drivesdk