Application Notes: PWM EasyDriver

Objectives

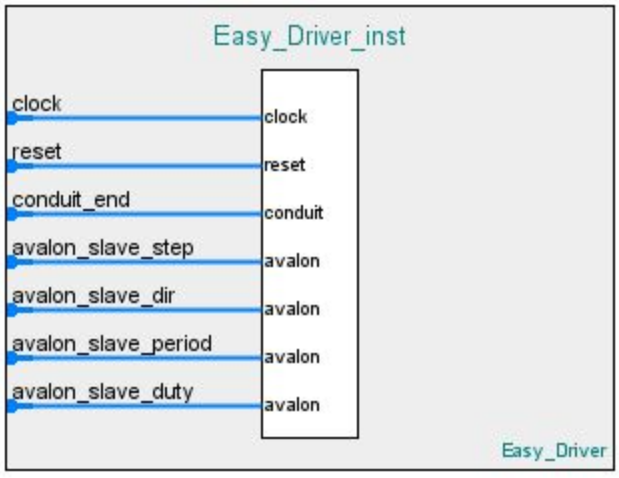
- To configure an EasyDriver custom component
- To generate a PWM signal to feed the driver
- To drive stepper motors using the EasyDriver component and C software

Introduction

In this application note, creation of an EasyDriver Qsys component and the code to drive simple stepper motors are explained. The platform used for development and testing is Altera's De1-SoC, Cyclone V FPGA board [1] with Quartus Prime 17.0. The stepper motors used for this application are from Adafruit Industries LLC [2] but any other simple stepper motor may be substituted with minimal modifications from the original notes presented.

Design

Hardware Design



There are four avalon memory mapped slave interfaces (step, dir, period and duty). They each have an address so that their values can be changed from software.

From this equation for a frequency divider: [3]

$$f_{out} = \frac{f_{in}}{n}$$

Solving for n gives the number of clock cycles needed such that f_{out} can be generated from f_{in}.

$$n = \frac{f_{in}}{f_{out}}$$

f_{in} is assumed to always be higher than f_{out} and is defaulted to 50MHz (as set by the DE1-SoC board.) This way, n is an integer always greater than 1. This value (n) is called ‘period’.

Aside: Period has the units of seconds, while here it is an integer. The reason we have called it period is because it refers to the wavelength and the corresponding measurement in seconds can be calculated by :

$$period\ of\ f_{out} = period\ of\ f_{in} \times n = \frac{1}{f_{in}} \times n$$

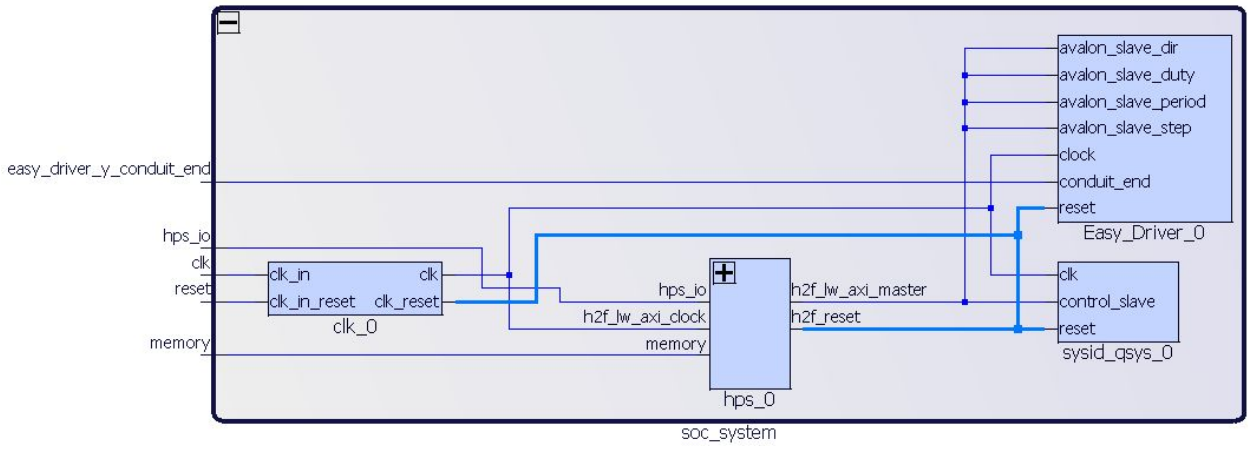
Duty cycle is the amount of time the wave spends pulled high. Duty is calculated from a hard coded constant DUTY_CYCLE found in motorConstants.h which defines the percentage of high time. Period was previously calculated, so Duty is derived from that by multiplying it by the constant.

$$Duty = Period \times DUTY\,CYCLE$$

Dir indicates which way the motor will spin, clockwise/counter-clockwise. For our system, 0 means CW motor spin and 1 is CCW direction.

Step refers to the steps of the stepper motor. The component will count down from this step value until it hits zero, at this point the motor will no longer rotate (but will stay stalled.) Our motor does 200 steps in full step mode which corresponds to 4 cm per revolution with our timing belt.

The conduit variables are used for connecting to the GPIO header. More information on how these variables are controlled in software is found below.



Software Design

These four files should be included in the ‘HWLIBS’ folder within eclipse: stepper.c, stepper.h, motorConstants.c, motorConstants.h.

Stepper.c/Stepper.h
This file contains functions that perform the calculations described above for the four memory mapped variables that the vhdl requires to do the pwm signal.

MotorConstants.c/MotorConstants.h
This file contains the addresses of the memory mapped variables which you will have to change. Other constants for the stepper motor itself are included here and may have to be changed according to your stepper motor.

Please Note that our project required two motors (x and y) called through enum Motor, you can add as many motors as you would like.

Procedure

Hardware Procedure

1. Open Quartus 17.0
2. Import the .qar file (an archived file)
    a. Imported files should have the .qip file, motorPwm.vhd, and EasyDriverPwm.vhd
3. Open Qsys
4. Import soc_system.qsys
    a. A new component called ‘EasyDriver’ should be visible in the IP catalog
5. Generate HDL
6. Go back to Quartus
7. Analyze & Synthesize
8. Run both tcl scripts
9. Compile Design

Software Procedure

1. Open DS-5 v5.25.0
2. Import the archived software file. This is a complete project and can be run as is.

Below is a more detailed explanation of the provided functions.

```
INT32U currFreq = 900;
InitMotor(motorY, currFreq);
StepMotor(motorY, 200);

OSTimeDlyHMSM(0, 0, 0, 500);
```

To operate the motors, place the above code snippet within a task (inside app.c) to initialize the pwm component via the InitMotor(...) function. StepMotor(...) can be given a hard coded amount of steps to spin the motor that many steps. To convert from distance (cm) to steps, the functions found in Stepper.c can be used. XYCoor2Steps(int y) calculates the number of steps given a distance in cm (- is left/down and + is right/up). StepMotor(...) can now be called with this converted number of steps.

Source Documentation and References

[1] Terasic Technologies Inc., “DE1-SoC User Manual Rev F” [Online]. Available: https://eclass.srv.ualberta.ca/pluginfile.php/3468647/mod_resource/content/1/DE1-SoC_User_manual_REV_F.pdf [Accessed Mar. 15, 2018]

[2] Digi-Key Electronics., “324 Adafruit Industries LLC | Motors, Solenoids, Driver Boards/ Modules | DigiKey” [Online]. Available: https://www.digikey.ca/product-detail/en/adafruit-industries-llc/324/1528-1062-ND/5022791 [Accessed Mar. 15, 2018]

[3] Wikipedia, “Frequency Divider” [Online]. Available: https://en.wikipedia.org/wiki/Frequency_divider [Accessed Mar.15, 2018]

[4] Allegro MicroSystems, LLC, “A3967 Microstepping Driver with Translator” [Online] Available: https://cdn.sparkfun.com/datasheets/Robotics/A3967-Datasheet.pdf [Accessed Mar. 15, 2018]

[5] Toni_K. SparkFun Electronics ®, “Easy Driver Hook-up Guide” [Online] Available: https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide [Accessed Mar. 15, 2018]

Attached Files

MotorDriver.zip
MotorPwmEasyDriver.qar
Preloader.ds
U-boot-spl

Application Notes by: Kelly Chin & Jessica Huynh