Embedded Des Enabling Robotics
Lab 9 report

Shiyu Wang
Hao Jiang

Fall 2016
2 December

Introduction

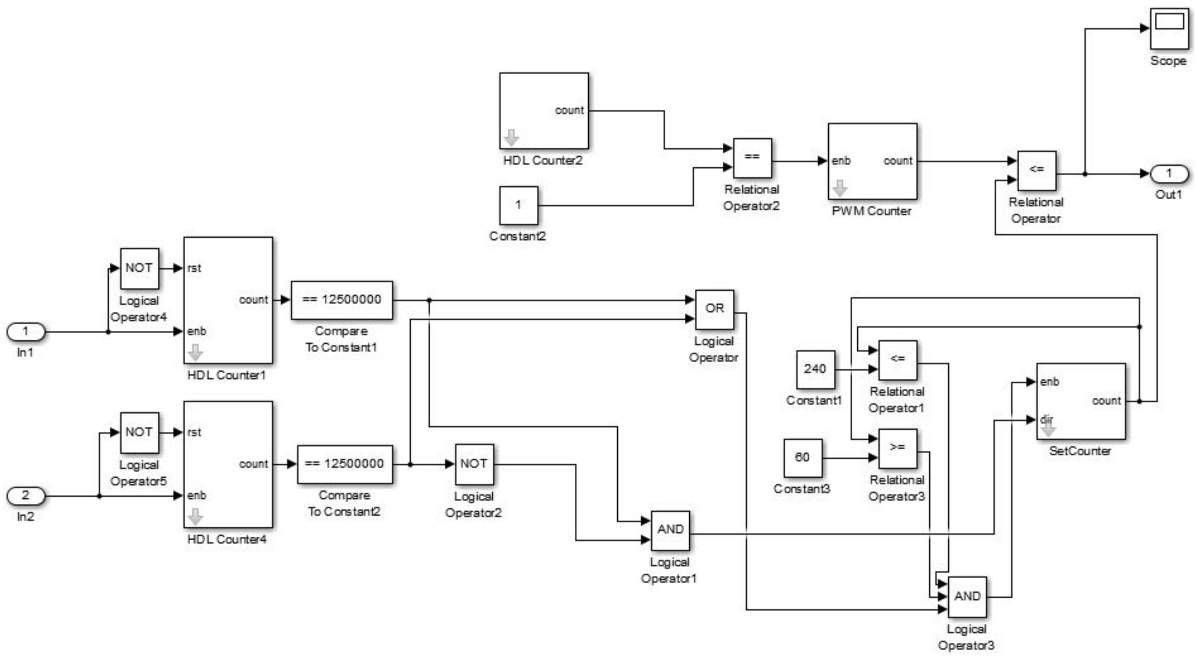
The goal of this lab is to control the RC servos of the Robotic arm. The digital logic was implemented in the FPGA on the ZedBoard. The FPGA generated PWM signals. The PWM signal controlled RC servos in the robotic arm.

Lab Preparation

We first logged onto the computer using our credentials. We then connected the Zedboard to the PC host via Ethernet to the RJ45 connector. The ZedBoard was boosted after connected to the computer. We made an SSN connection to the ZedBoard where the IP is 192.168.1.10, and port 22. We logged in as root.

Lab 9.1

In this lab, we instantiated what we created in prelab. Since a single push of up/down button will increment/ decrease the SetCounter by 1, and it is assumed in the prelab that a counter rolls over every 20 ms when reaching the value of 2000), steps needed to pass completely through the whole range from a duty cycle of 0.6ms to 2.4ms are 180 ((2.4-0.6) / (20ms/2000)) . The step size can be set to be 180/15=12. The "Model Configuration Parameters" were updated to: Fixed step, discrete, fixed-step size =1. We updated our design from pre-lab, and the new design looks like this:

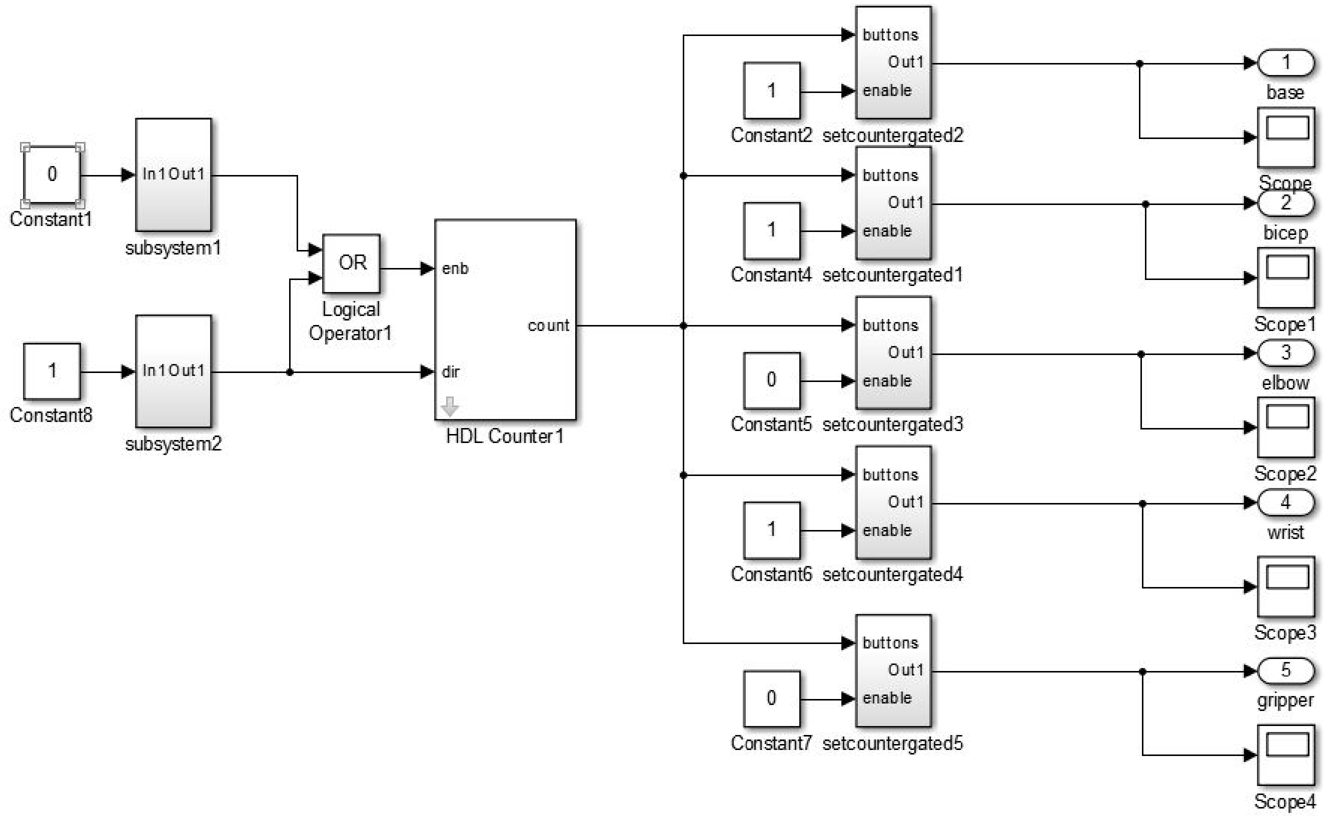


We tested the design by replacing the inputs with constants 0 and 1, and run for many cycles. What we observed was that when we used up =1, down = 0, the duty cycle was increased by 0.01 ms for each cycle and when it was up = 0, down = 1, the duty cycle was decreased by 0.01 ms each time, but never lower than 0.06 ms. This meant our design worked. As a result, we kept on doing the rest of the lab projects.

We connected the robotic arm with ZedBoard. After connecting successfully, we found out that when we push up and down push buttons in order to reach different duty cycles, the robotic arm moved to corresponding positions.

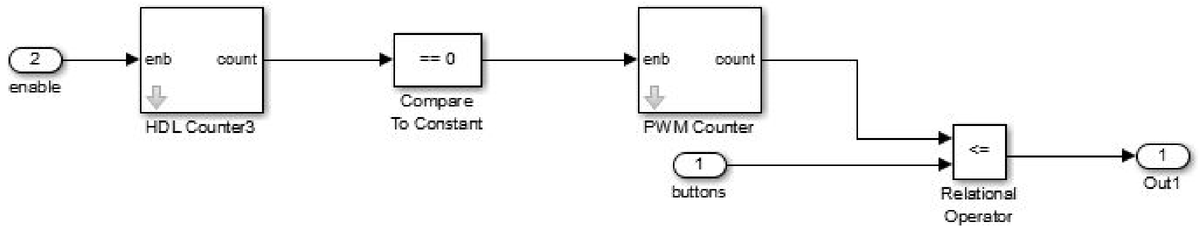
9.2

In this part, we expanded the SetCounter design to only change its value if enabled by a switch. What we had designed as follows:



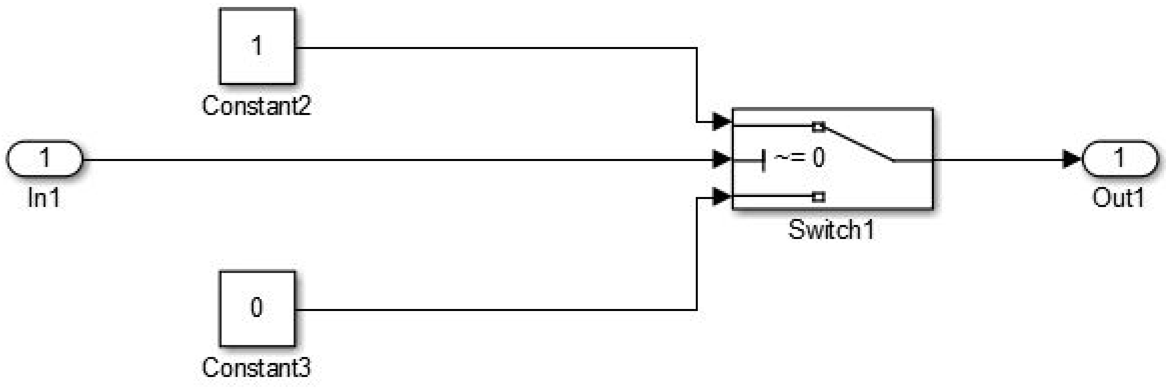
Designs of setcountergated and subsystem are as follows:

Setcountergated



This setcountergated design compares the enable input with 0, and output the result to another counter. If it is 0, then counter is enabled and send an 1 to relational operator. Only if the output value is smaller than the button value, can a 1 output will be send to the next level.

Subsystem:

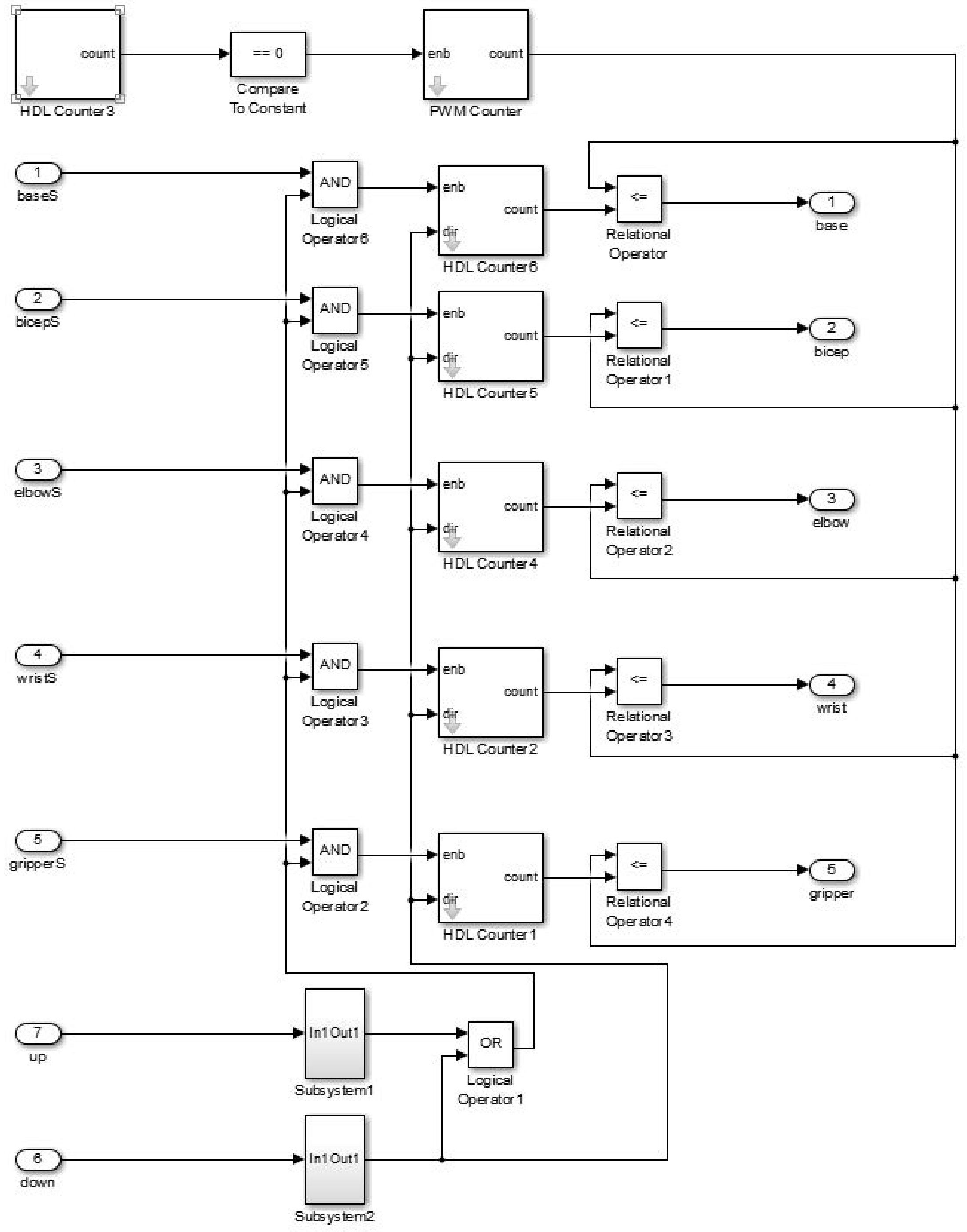


This design essentially selects the switch the input will connect to. Only if the switch is at its off position can the switch connected to the input and pass on the value.

After the design was created, it was simulated. We found it works. Because only if changeEnable is one, can the outputs be changed. Else, the output signal will stay the same.

9.3

Now that, we have had five thresholds generated by SetCounterGated. We used them to derive five different PWM signals.



count
HDL Counter3
== 0
Compare To Constant
enb
count
PWM Counter
1
baseS
AND
Logical Operator6
enb
count
dir
HDL Counter6
<=
Relational Operator
1
base
2
bicepS
AND
Logical Operator5
HDL Counter5
Relational Operator1
2
bicep
3
elbowS
AND
Logical Operator4
HDL Counter4
Relational Operator2
3
elbow
4
wristS
AND
Logical Operator3
HDL Counter2
Relational Operator3
4
wrist
5
gripperS
AND
Logical Operator2
HDL Counter1
Relational Operator4
5
gripper
7
up
In1 Out1
Subsystem1
OR
Logical Operator1
6
down
In1 Out1
Subsystem2