EECS 3216

Winter 2021

LAB 144

INVERTER

INVERTER OR DC-TO-AC CONVERTER

Lab Objective

The objective of the lab is to use FSM to model and design the control unit of the inverter and implement it on FPGA.

Introduction

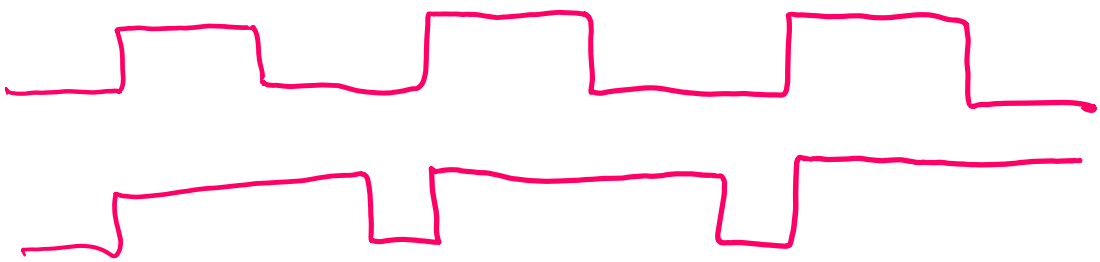
In this lab you will design a very simple inverter. The inverter is a circuit that takes DC voltage and produce AC voltage. An inverter is used to control the speed of many AC motors.

In this lab, we will not be concerned with the circuit that produces the Ac voltage (although we will present it without explanation for completeness). The main concern is control of the PWM signal that turns the transistors ON and OFF to produce the sine wave, we will just concentrate of the control signal.

PWM

A PWM signal is a square wave modulated such that the duty cycle (the percentage of the time the signal is high to the entire period) have a specific value.

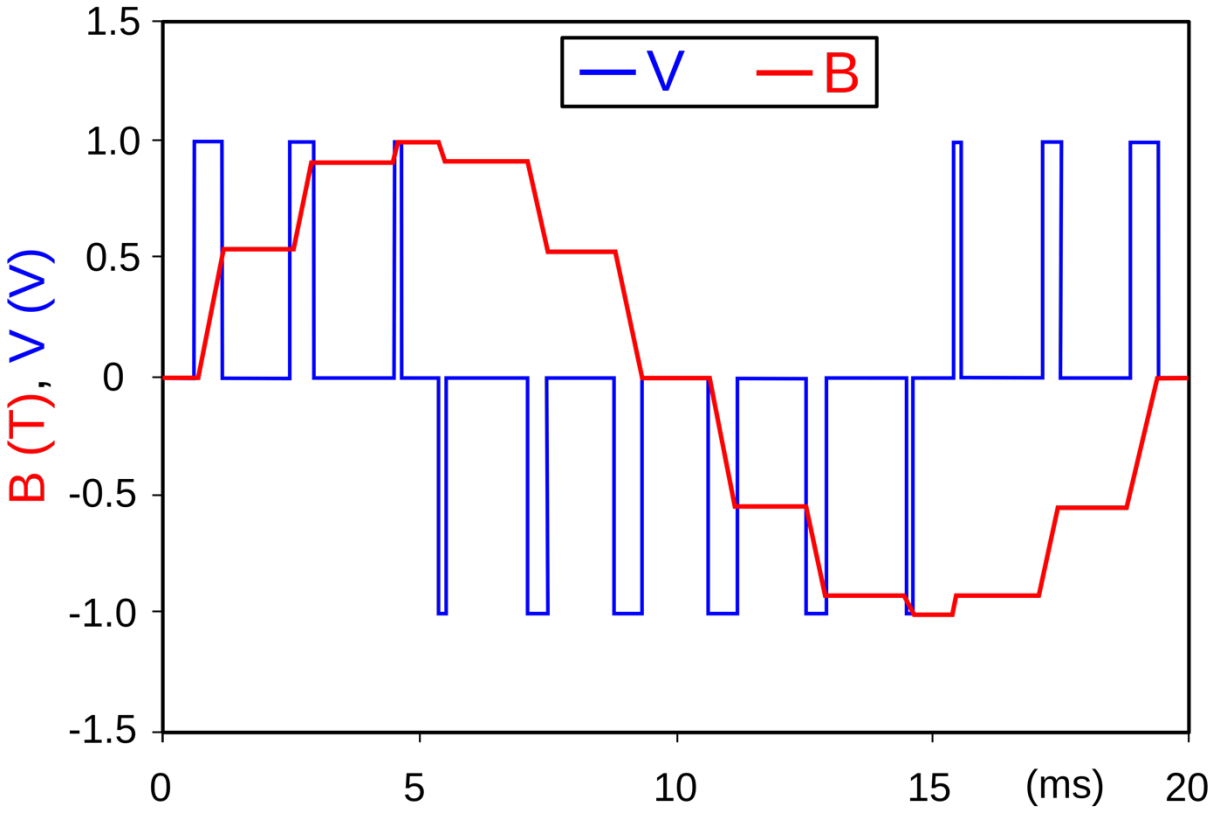
For example, we can use PWM to control the speed of a dc motor. Consider the following



If you consider the above two PWM signals, the first one has 50% duty cycle, if we consider the high to be +5v, then the average voltage delivered to the motor is 2.5V. While in the second case, duty cycle is 90%, 4.5 average voltage is delivered to the motor. Assuming the motor speed is proportional to the voltage, we can control the motor speed this way.

Creating a sine wave using PWM is a little similar to the above situation. If we sample the sine wave at different points of time, and then for every sample, we generate a PWM signal with duty cycle equal to the value of the sample. Using a low pass filter a sine wave could be generated.

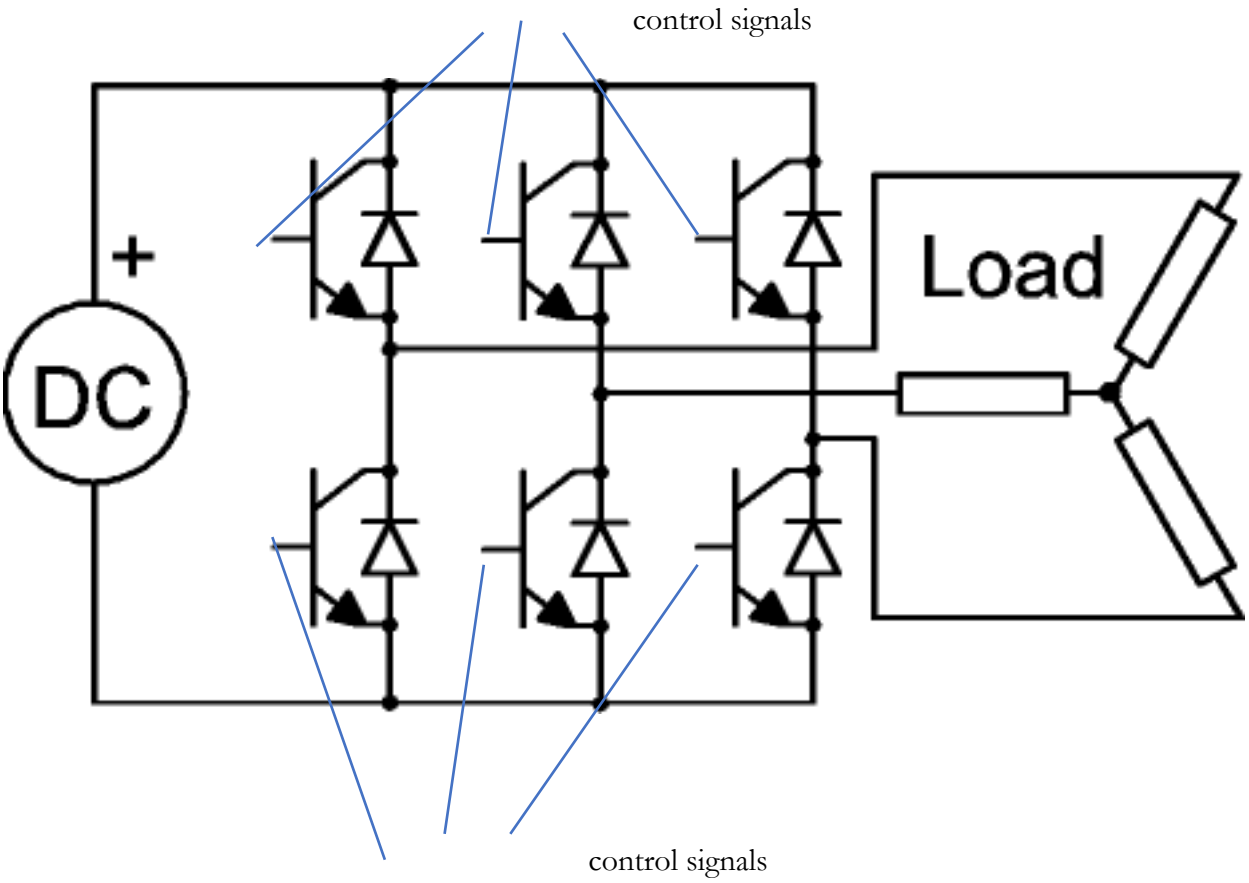
Consider the following pulses (accessed from Wikiopedia, image by Zureks under license https://creativecommons.org/licenses/by-sa/3.0/legalcode). The wave in blue is a sequence of pulses with varying duty cycle, the wave in B is the filtering of the blue signal. The red is not a perfect sine wave, however, by increasing the switching frequency we can approximate a sine wave.



The following figure shows 3-phase inverter with Y load (C J Cowie at the English-language Wikipedia, CC BY-SA 3.0 http://creativecommons.org/licenses/by-sa/3.0/ via Wikimedia Commons).

The control unit is the unit that produces the input pulses (control signals in the figure) in order to turn the BJTs (actually IGBT) on and off.

Again, for simplicity here we are dealing with a single-phase inverter (think about the modifications you have to do to your design to make it 3-phase inverter.



Simple inverter

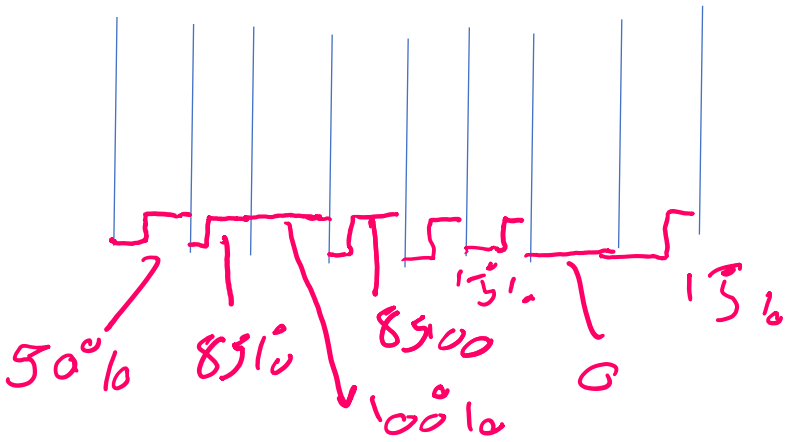
Consider the following sine wave, which is sampled at points A, B, C, D, E, F, G, H. Assuming the samples are done every 45 degree, the values of the samples are

0, 0.7, 1, 0.7, 0, -0.7, -1, -0.7

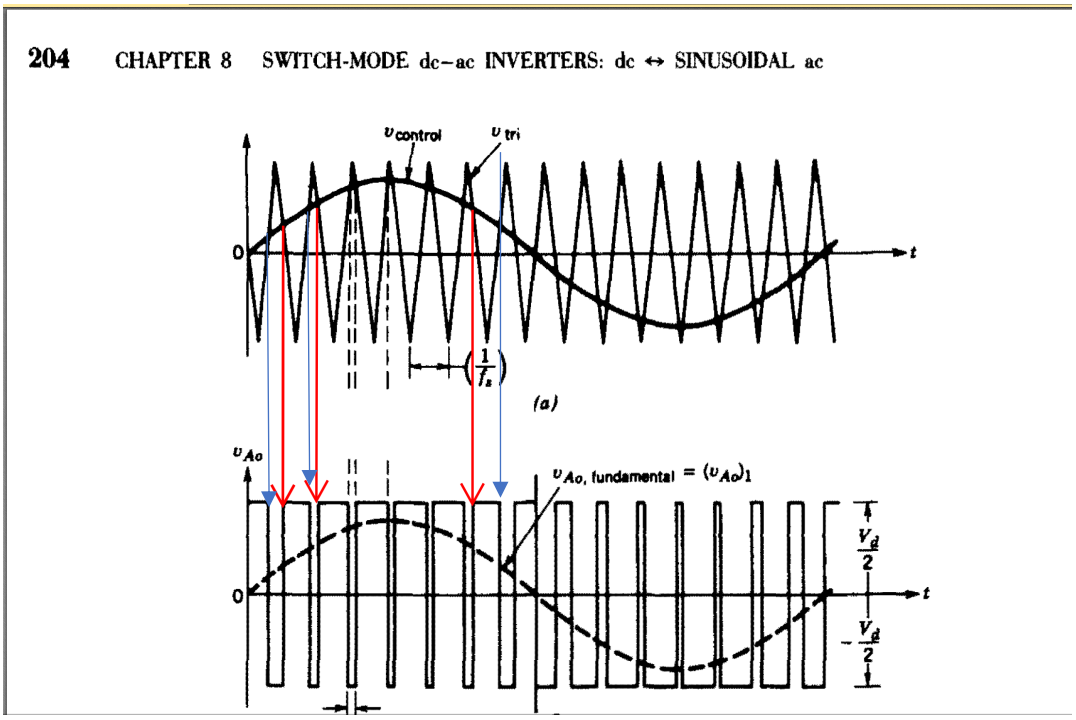
We assume the sample is displaced by +1 to make all samples positive, the samples values are

1, 1.7, 2, 1.7, 1, 0.3, 0, 0.3, duty cycles 50%, 85%, 100%, 85%, 15% 0%, 15%

Then we generate for every sample a PWM with a duty cycle that is equal to the sample value (divided by 2 of course, since the maximum value after displacement is 2).



How to implement this



LAB:

As shown in the above figure, a sine wave is compared with a triangle signal. if the triangle signal crosses the sine wave up, the output is reset to 0, if the triangle wave crosses the signal down, the output is set to 1.

Implementation

There are two ways to implement this.

The first, you need to implement the triangular waveform (up and down counter) and the sine wave. The counter is free running and the comparison between the counter and the sine wave determines if the output is 0 or 1.

The second, you have the sinewave, divide it into *n* segments (64 or 128). The value of the sine wave for each segment is represented by *m* bits (8 or 16 bits). In each segment the output is HIGH for percentage of the segment that is equal to the value/2^m . Please see the explanation aftr the sine wave table below

The easiest way to implement the sine wave is to calculate the values and store it in a ROM. Keep in mind that in Quartus I couldn't implement the memory as IP (the only

way to do this is to implement it in VHDL and include it in your design). You can always write a Verilog module for ROM.

What should be the maximum value for the counter, and the number of points (entries in the ROM)? Think about that yourself. I will explain it in the lecture and give some pointers, but I will give you time to think about it yourself first.

Preparing the sine wave table

The following table shows 18 samples of a full sine wave, each sample is shown in degrees, radians, the sample value, the sample value shifted by 1 (to eliminate negative values), and finally the sample value+1 represented in 8 bits. You can do this in excel, C, Java, or any other programming language.

deg	rad	sin	sin+1	Value(8-bit)	% of the cycle active
0	0	0	1	127	0.498039
20	0.349066	0.34202	1.34202	170	0.666667
40	0.698132	0.642788	1.642788	208	0.815686
60	1.047198	0.866025	1.866025	236	0.92549
80	1.396263	0.984808	1.984808	252	0.988235
100	1.745329	0.984808	1.984808	252	0.988235
120	2.094395	0.866025	1.866025	236	0.92549
140	2.443461	0.642788	1.642788	208	0.815686
160	2.792527	0.34202	1.34202	170	0.666667
180	3.141593	1.23E-16	1	127	0.498039
200	3.490659	-0.34202	0.65798	83	0.32549
220	3.839724	-0.64279	0.357212	45	0.176471
240	4.18879	-0.86603	0.133975	17	0.066667

260	4.537856	-0.98481	0.015192	1	0.003922
280	4.886922	-0.98481	0.015192	1	0.003922
300	5.235988	-0.86603	0.133975	17	0.066667
320	5.585054	-0.64279	0.357212	45	0.176471
340	5.934119	-0.34202	0.65798	83	0.32549
360	6.283185	-2.50E-16	1	127	0.498039

The value above is limited between 0 and 2, and is represented as 8-bit number (with 2 equal to 11111111).

In the table above, the full wave is divided into 18 segments. Assume the wave is 100 HZ, a full wave is *10 msec*. which means every segment is 10/18 = 0.55 *msec*.

Consider the first segment (an angle of 0), the value is 1 (sine 0 + 1) to make it all positive, that is corresponding to a value of 127. 127/256 = 0.498. That means for the first segment (0.55 *msec*) 49.8% of that (0.274 *msec*) the output is HIGH, the remaining 0.276 *msec*, the output is LOW.

The second segment value is 1.34202, in 8 bit that is 170. 170/256 = 0.6667, that means for the next 0.55 *msec*, the output is HIGH for 0.367 *msec*, and LOW for the rest of the 0.55 *msec*. And so on.

Specification

Design the control circuit to produce the signals required to generate a 200Hz wave. The output should be connected to the right most LED on the board

Deliverables

1. A report containing the following
    a. Your name, student number, and Lab number
    b. Design: How did you reach your implementation. For this lab use FSM. You must explain/show any fixed numbers in your design (A 5-bit counter, why did you choose 5?)
    c. Verilog code
    d. The simulation result showing one whole cycle.
2. A video that shows your circuit working

Marking

Video presentation (50%), the report (50%). For each the marks are 3, 2, and 1. For doing everything you were asked to, most of what you are asked to do, less than half, or none at all

The lab is due 11:59 pm March 7.