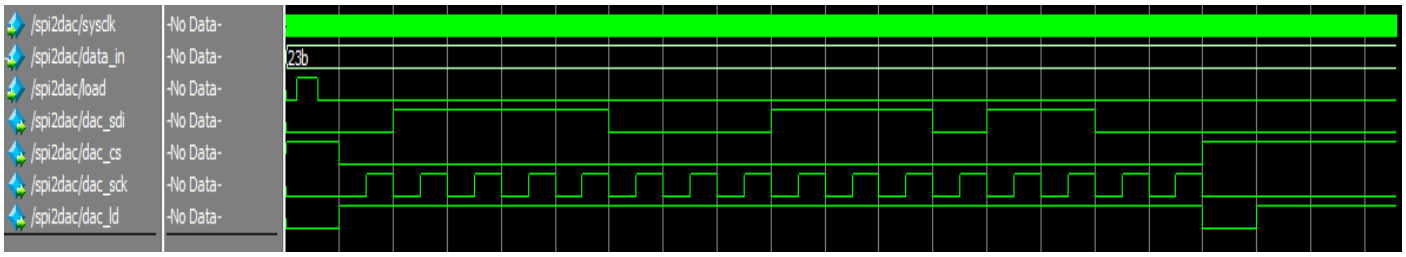
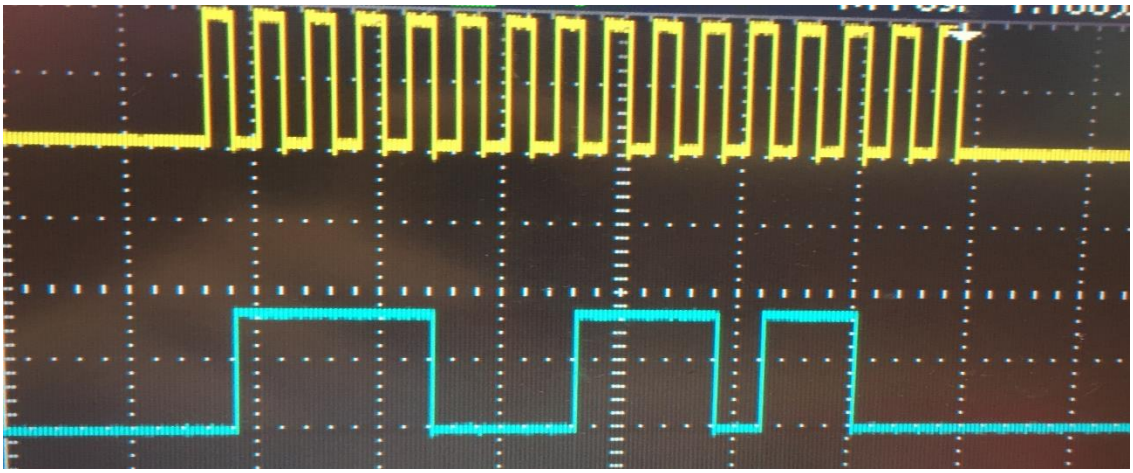
Verilog Experiment - Part 3

Experiment 10

In this experiment we used switches to load data to the spi2dac module, when all switches were off the output was 0V at SP9 and when all switches were on the output was 3.3V on SP9 which was expected and required.





Above are the modelsim and oscilloscope outputs for an input of 10'h23b which both agree with each other. DAC_SCK is measured at TP3 and DAC_SDI is measured at TP1, where the scope had to be triggered using external signal DAC_CS at TP2.

Experiment 11

```
1   module ex11_top (
2   SW,
3   CLOCK_50,
4   DAC_CS,
5   DAC_SDI,
6   DAC_LD,
7   DAC_SCK,
8   PWM_OUT
9   );
10          input CLOCK_50;
11          input [9:0] SW;
12          output DAC_CS, DAC_SDI,DAC_LD, DAC_SCK;
13          output PWM_OUT;
14          wire    tick;
15
16
17          clk_tick                clockdivider(CLOCK_50, count, tick);
18          spi2dac                 dac1(CLOCK_50, SW, tick, DAC_SDI,DAC_CS,DAC_SCK,DAC_LD);
19          pwm                     pwm1(CLOCK_50, SW, tick, PWM_OUT);
20
21          endmodule
```

In this experiment the output voltage range for the PWM was the same as for the DAC, but PWM was as clearer which I presume is due to the low pass filter removing high frequency noise components, however they did show the same output to one another as expected. Again both have the same clock and load clock as well as same data coming in from the switches. The DAC and PWM outputs had to be found from the qsf file as they differ from the names on the board in contrast with that on the FPGA, so wiring the top module requires more care in this case. Above is the top module.

Experiment 12

```
module ex12_top(
        SW,
        CLOCK_50,
        HEX0,
        HEX1,
        HEX2
);

        input [9:0] SW;
        input CLOCK_50;
        output  [6:0]   HEX0, HEX1, HEX2;
        wire            [9:0] q;

        ROM     ROM1(SW, CLOCK_50, q);
        hex_to_7seg SEG0(HEX0, q[3:0]);
        hex_to_7seg SEG1(HEX1, q[7:4]);
        hex_to_7seg SEG2(HEX2, q[9:8]);

endmodule
```

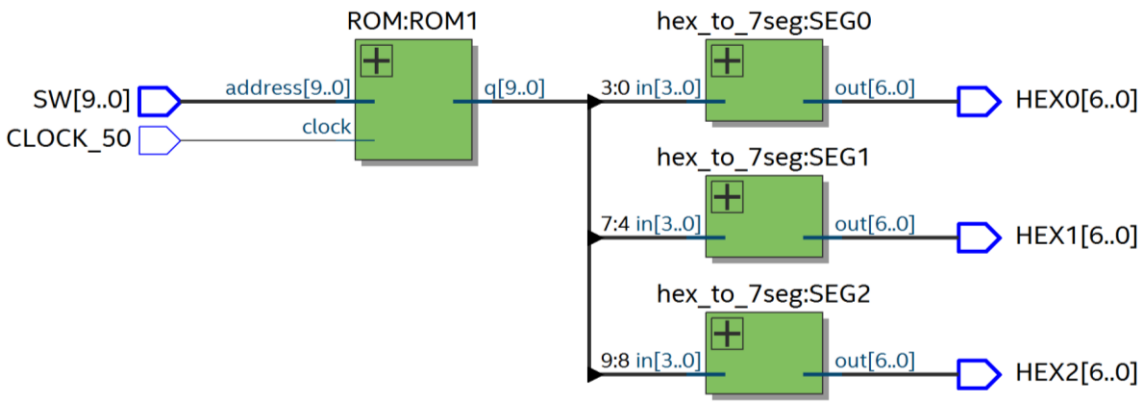
This experiment uses a 1K X 10bit ROM that stores a table of sine values which is addressed using switches and the data is outputted to the three SEG displays. When testing the output on the screens had the same values as in the script .mif file, so the code was verified. The rom checks the address every positive clock cycle.

The output equation is:

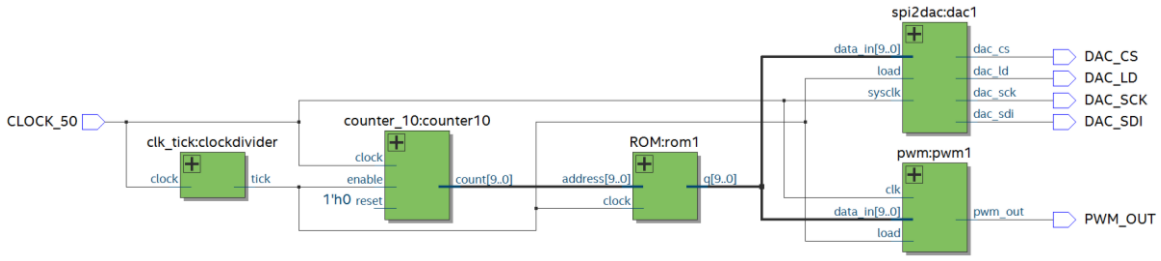
$$D[9:0] = \text{int}(511*\sin(A[9:0]*2*\pi/1024)+512) \qquad \text{for } 1023 \geq A[9:0] \geq 0$$

Since the DAC accepts an input range of 0 to 1023, we must add an offset of 512 in this equation. (This number representation is know as **off-set binary** code.)

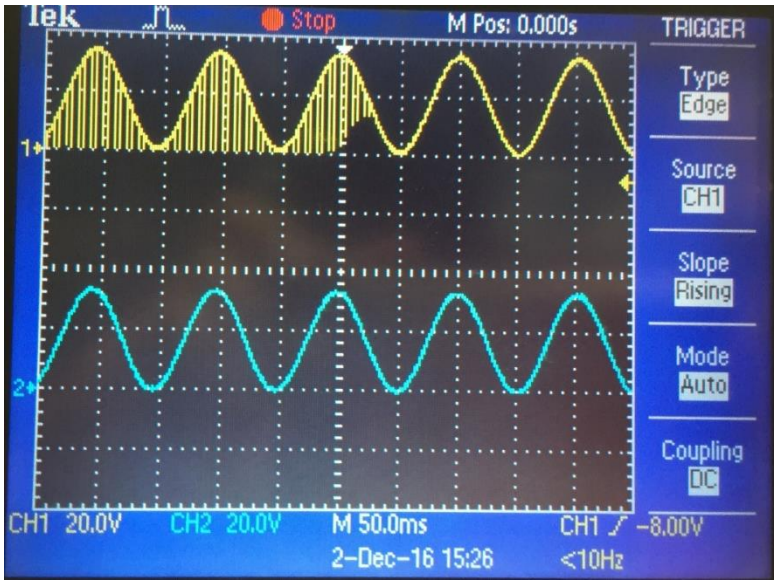
Below is the RTL top – level design which gives a clean overview of how the code is implemented.



Experiment 13



In this experiment I replaced the sliding switches with a 10-bit binary counter which is enabled every 10KHz. The ROM contains 1024 cycles of a sinewave and at every cycle of the 10Khz clock it is incremented by 1. Therefore 10000/1024 = 9.77Hz which is just below 10 and has been verified by the output of both the PWM and the DAC which both have frequency around 10Hz as shown below.

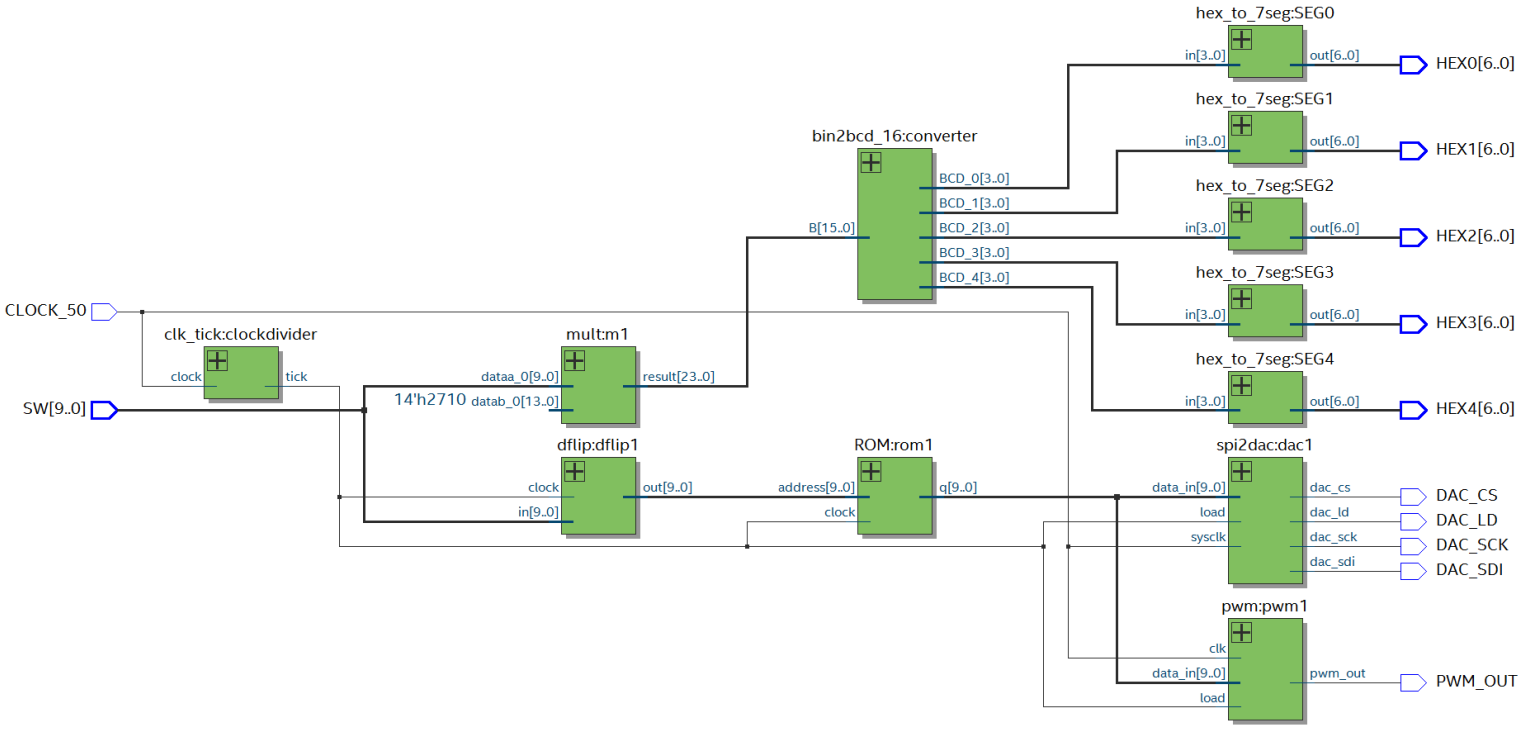


On the left, each interval is 50ms. Each cycle occurs every 2 intervals, so 1/100ms = 10Hz.

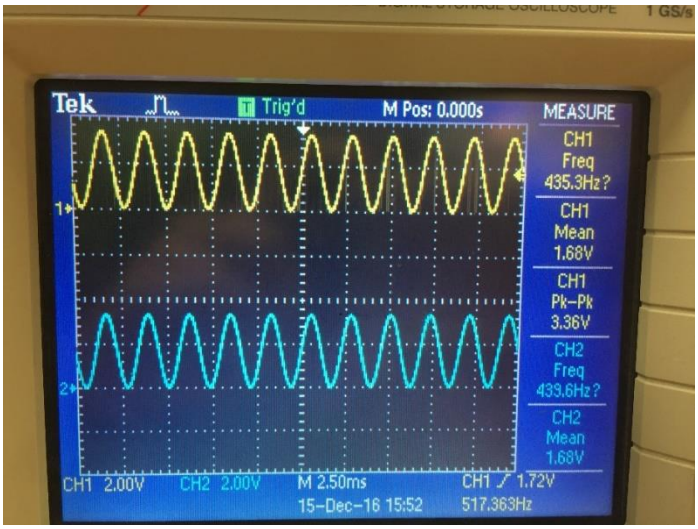
```
1   module ex13_top (
2   CLOCK_50,
3   DAC_CS,
4   DAC_SDI,
5   DAC_LD,
6   DAC_SCK,
7   PWM_OUT
8   );
9           input CLOCK_50;
10          output DAC_CS, DAC_SDI,DAC_LD, DAC_SCK;
11          output PWM_OUT;
12          wire    tick;
13          wire [9:0] A;
14          wire [9:0] D;
15
16          clk_tick                clockdivider(CLOCK_50, count, tick);
17          counter_10  counter10(CLOCK_50,tick,A,0);
18          ROM         rom1(A, tick, D);
19          spi2dac                 dac1(CLOCK_50, D, tick, DAC_SDI,DAC_CS,DAC_SCK,DAC_LD);
20          pwm                     pwm1(CLOCK_50, D, tick, PWM_OUT);
21
22          endmodule
```

On the left is the top level module with the 10-bit counter operating on the same basis as the 16-bit counter already seen in previous exercises.

Experiment 14 (Optional Challenge)



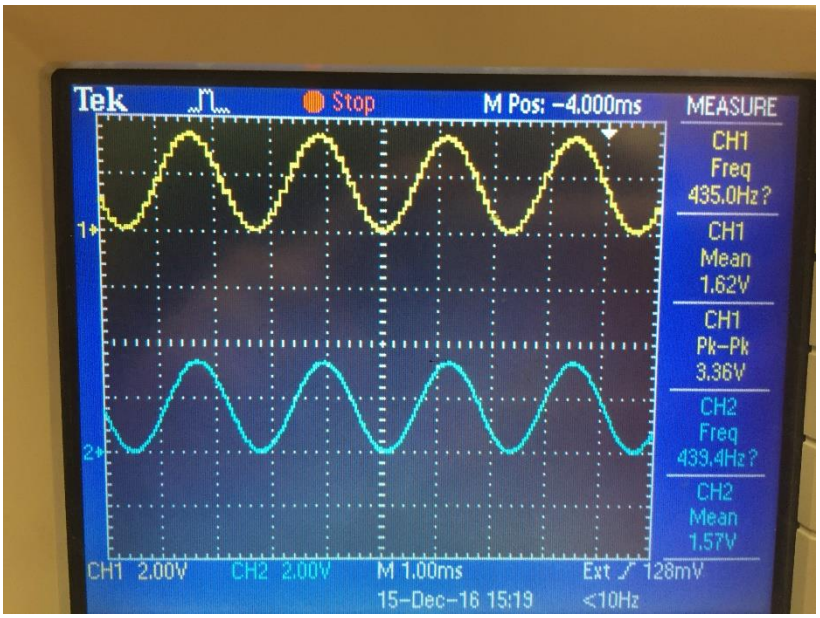
The above circuit combines all previous experiments in this part. The switches are used to advance the address i.e the phase every clock cycle to the ROM. The switch address value is multiplied by 10000 and the top 14 bits are outputted to the SEG displays to show the frequency of the sine wave.



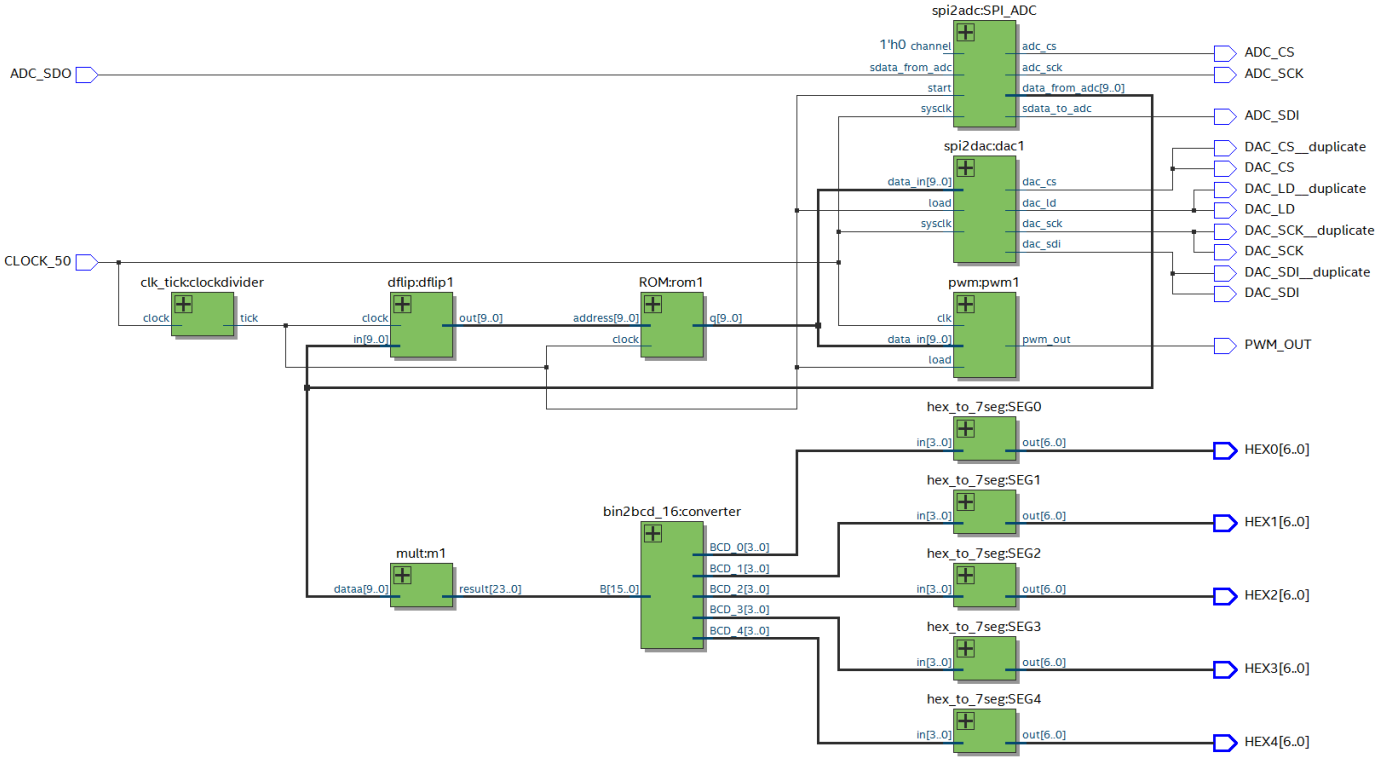
This is indeed correct as when the switches are placed as below on the left the output to the SEGS and therefore the frequency should be 439Hz which is shown very close to the values measured on the left. Switch configuration for 439Hz is 10'b00000101101.

Experiment 15 (Optional)

This experiment was the same as the previous one but with the potentiometer controlling the frequency of the sine wave. The potentiometer was very sensitive and it was quite difficult to obtain the measurement below, however it was eventually obtained.



Below is the block diagram and the top-level code. I had to choose channel 0 to select the potentiometer and change some wiring to include the inputs and outputs of the ADC, however it was very similar to the previous experiment, but this time using the ADC.



```
module ex15_top (CLOCK_50,DAC_CS,DAC_SDI,DAC_LD,DAC_SCK,PWM_OUT,HEX0,HEX1,HEX2,
HEX3,HEX4,DAC_SDI, DAC_SCK, DAC_CS, DAC_LD,ADC_SDI, ADC_SCK, ADC_CS, ADC_SDO
);
        input    CLOCK_50;
        output  DAC_CS, DAC_SDI,DAC_LD, DAC_SCK;
        output  PWM_OUT;
        wire    tick;
        wire    [9:0] A;
        wire    [9:0] D;
        wire    [23:0] result;
        wire    [3:0]   BCD_0, BCD_1, BCD_2, BCD_3, BCD_4;
        output  [6:0] HEX0, HEX1, HEX2, HEX3, HEX4;
        wire    [9:0]   data_in;                // converted data from ADC
        wire                    data_valid;
        wire                    DAC_SCK, ADC_SCK;
        output          ADC_SDI;                        //Serial data out to SDI of the ADC
        output          ADC_SCK;                // ADC Clock signal
        output          ADC_CS;                 //Chip select to the ADC, low active
        input           ADC_SDO;                        //Converted serial data from ADC

        spi2adc SPI_ADC (
                .sysclk (CLOCK_50),
                .channel (1'b0),
                .start (tick),
                .data_from_adc (data_in),
                .data_valid (data_valid),
                .sdata_to_adc (ADC_SDI),
                .adc_cs (ADC_CS),
                .adc_sck (ADC_SCK),
                .sdata_from_adc (ADC_SDO));

        clk_tick        clockdivider(CLOCK_50, count, tick);
        mult            m1(data_in, result);
        dflip           dflip1(tick, data_in, A);
        ROM         rom1(A, tick, D);
        spi2dac dac1(CLOCK_50, D, tick, DAC_SDI,DAC_CS,DAC_SCK,DAC_LD);
        pwm     pwm1(CLOCK_50, D, tick, PWM_OUT);
        bin2bcd_16  converter(result[23:10], BCD_0, BCD_1, BCD_2, BCD_3, BCD_4);
        hex_to_7seg  SEG0(HEX0, BCD_0);
        hex_to_7seg  SEG1(HEX1, BCD_1);
        hex_to_7seg  SEG2(HEX2, BCD_2);
        hex_to_7seg  SEG3(HEX3, BCD_3);
        hex_to_7seg  SEG4(HEX4, BCD_4);

endmodule
```