Features

Waves

0) SQUARE

- Duty cycle
- amplitude control
- frequency control

1) TRIANGLE

-amplitude control

-frequency control

2) TRIANGLE2

-amplitude control

-frequency control

3) SAWTOOTH

4) SINE WAVE

-amplitude control

-frequency control

5) RECTIFIED SINE

-amplitude control

-frequency

6) DIAMOND

7 segment

-voltage display

-12 bit display

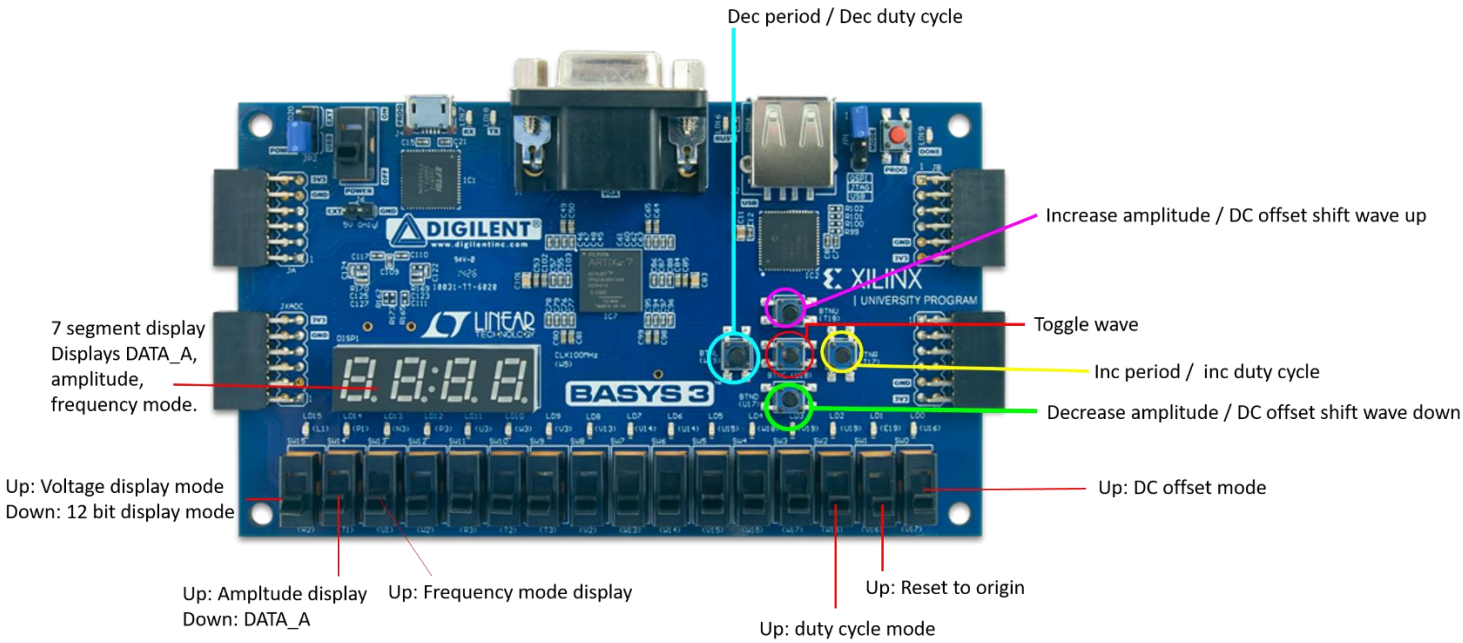
-Amplitude

-Frequency mode

DC offset

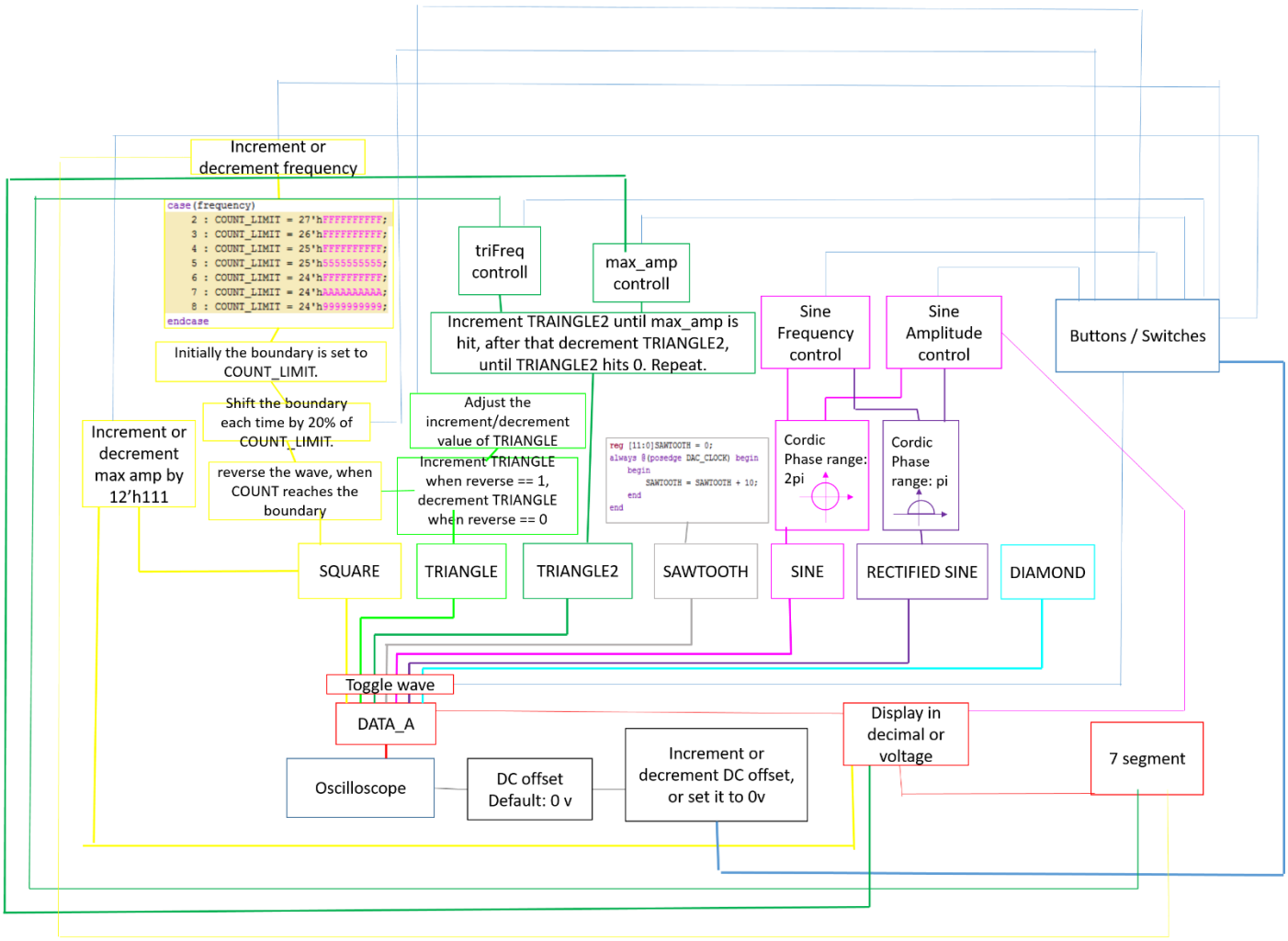
- DC offset control
- reset to origin

USER INTERFACE



	Switch						Button				
Function	**SW15**	**SW14**	**SW13**	**SW2**	**SW1**	**SW0**	**BTNU**	**BTND**	**BTNL**	**BTNR**	**BTNC**
7 segment display DATA_A in 12 bit	Down	Down	Down								
7 segment display DATA_A in volts	Up	Down	Down								
7 segment display amplitude in 12 bit	Down	Up	Down								
7 segment display amplitude in volts	Up	Up	Down								
7 segment display frequency mode	Down	Down	Up								
Toggle wave											Push
Increase period decrease frequency				Down						Push	
Decrease period Increase frequency				Down					Push		
Increase Amplitude						Down	Push				
Decrease Amplitude						Down		Push			
Shift wave down (DC offset)						Up		Push			
Shift wave up (DC offset)						Up	Push				
Reset wave to origin					Up						
Increase duty cycle				Up						Push	
Decrease duty cycle				Up					Push		

Developer's guide



SQUARE

Frequency control

There are 6 modes of frequency to choose from. The variable frequency is incremented/decremented by buttons, and a COUNT_LIMIT is selected. The boundary is set by the variables dc0 and dc1. Initially dc0 and dc1 is set to COUNT_LIMIT. When COUNT reaches the boundary, reverse toggles. When reverse toggles, the wave is reversed.

```
case(frequency)
    2 : COUNT_LIMIT = 27'hFFFFFFFFF;
    3 : COUNT_LIMIT = 26'hFFFFFFFFF;
    4 : COUNT_LIMIT = 25'hFFFFFFFFF;
    5 : COUNT_LIMIT = 25'h5555555555;
    6 : COUNT_LIMIT = 24'hFFFFFFFFF;
    7 : COUNT_LIMIT = 24'hAAAAAAAAA;
    8 : COUNT_LIMIT = 24'h9999999999;
endcase
```

```
always @(posedge CLOCK) begin
    COUNT<=COUNT+1;
        if(COUNT == dc1 && reverse == 1)
        begin
            COUNT <= 0;
            reverse <= reverse + 1;
        end

        if(COUNT == dc0 && reverse == 0)
        begin
            COUNT <=0;
            reverse <= reverse + 1;
        end

        if(incFreq==1 || decFreq==1 || toggleShape==1)
        begin
            COUNT <=0;
            reverse <= 0;
        end
end
```

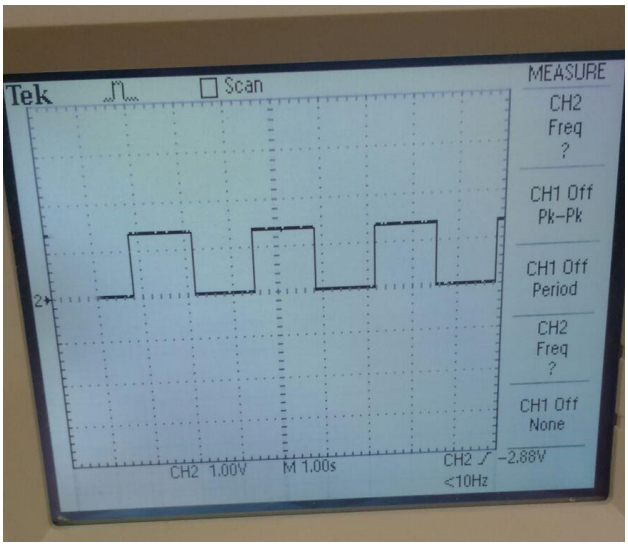
Duty cycle

Shift the boundary by 20% of COUNT_LIMIT each time signal to increase / decrease duty cycle is received.

```
always @(posedge SLOWCLOCK)
begin
    if(dc_sw == 0)
    begin
        dc0 = COUNT_LIMIT;
        dc1 = COUNT_LIMIT;
    end
    if(dc_sw == 1 && decFreq == 1 && (dc1>(COUNT_LIMIT / 5)))
    begin
        dc0 = dc0 + (COUNT_LIMIT / 5);
        dc1 = dc1 - (COUNT_LIMIT / 5);
    end

    if(dc_sw == 1 && incFreq == 1 && (dc0>(COUNT_LIMIT / 5)))
    begin
        dc0 = dc0 - (COUNT_LIMIT / 5);
        dc1 = dc1 + (COUNT_LIMIT / 5);
    end
end
```

Amplitude Control

Increment / decrement the variable square_add by 12'h111. If reverse == 0, SQUARE = square_add. Else SQUARE = 0.

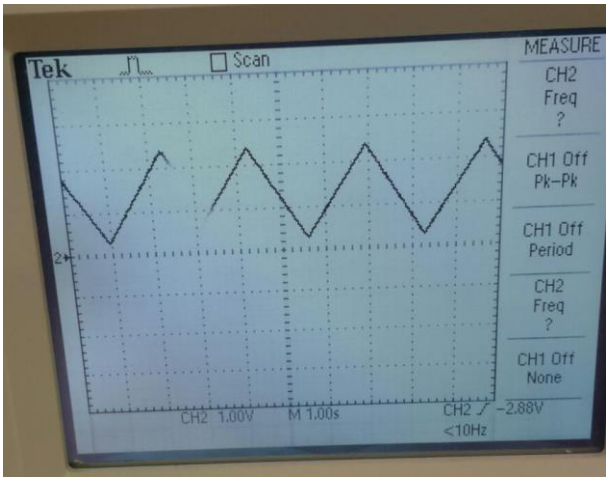


TRIANGLE

Controlled by the same frequency as SQUARE. When reverse == 0, TRIANGLE is incremented by triangle_add. When reverse == 1, TRIANGLE is decremented by triangle_add. Increment/decrement of TRIANGLE is done at the rate of 381.5 Hz

Amplitude control

Works by controlling triangle_add. Signal from buttons increment/decrement triangle_add by 5.



TRIANGLE2

Increment TRAINGLE2 until max_amp is hit, after that decrement TRIANGLE2, until TRIANGLE2 hits 0. Repeat.

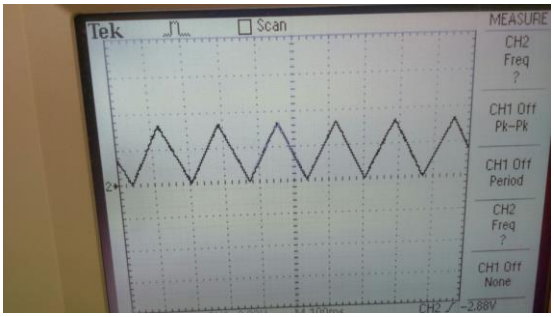
Amplitude control

max_amp is incremented/ decremented each time signal is received from buttons by 12'h111.

Frequency control

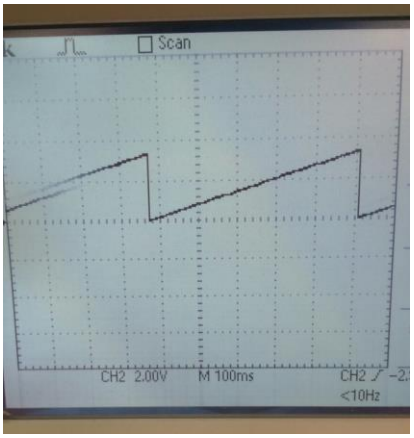
```
    case(triFreq)
        0 : bit_count = 10'hFFFFFFF; //longest period
        1 : bit_count = 11'hFFFFFFFFF;
        2 : bit_count = 12'hFFFFFFFFF;
        3 : bit_count = 13'h5555555555;
        4 : bit_count = 14'hFFFFFFFFF;
        5 : bit_count = 15'hAAAAAAAAAA;
        6 : bit_count = 16'h9999999999;
    endcase
end

variable_clock four09hz(CLOCK, bit_count, triangle2_clock);

//triangle frequency toggle
always @(posedge triangle2_clock)
```

The rate at which TRIANGLE2 increments is controlled by triangle2_clock, which is controlled by the selected bit_count.



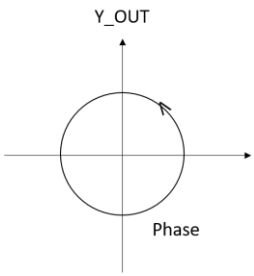
SAWTOOTH

SAWTOOTH is incremented at the rate of DAC_CLOCK (1.5kHz). Overflow will happen, which gives the sawtooth shape.



SINE

Uses CORDIC. CORDIC is a preinstalled IP library in XILINX. Takes in a PHASE as input. The output, M_AXIS_OUT, has the Y_AXIS component stored in the most significant 8 bits of M_AXIS_OUT, and transferred to the variable Y_OUT. The phase range is 2 pi. Phase increments continuously by 1'b1 from 0. When phase hits pi, it is set to -pi, and from there increases back to 0. Fixed floating point is used to express the input and outputs. Phase input is 2's complement expressed in 2QN format, where the 3 MSBs are integers. The MSB is the signed bit. The other 5 LSB are the fractional values (5 dp). The resolution is 2^-5 = 0.03125.



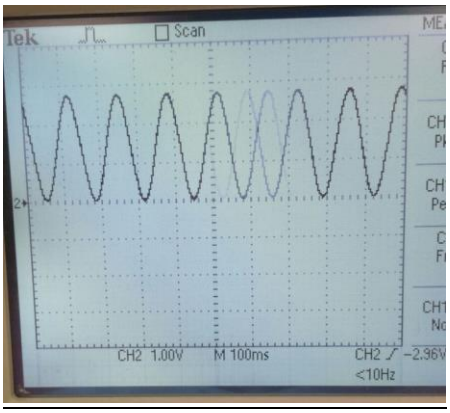
Y_OUT has max and min value of 1 and -1 respectively. Y_OUT is 2's complement in 1QN format, where the 2 MSB is the integer, and the MSB is the signed bit. Hence scaling has to be done, by using fixed floating point for the fractional value, translating the graph by 1 such that min value is 0, and then multiplying the Y_OUT by MAX_AMP.

Frequency Control

Clocks with controllable frequencies are fed into the driver clock for CORDIC. Phase is also incremented using this same clock.

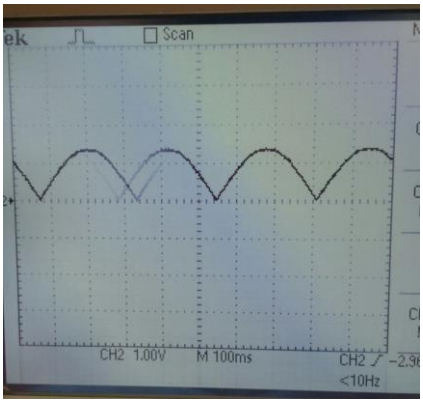
Amplitude control

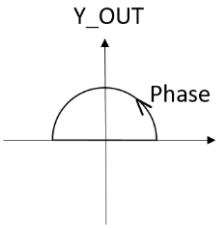
MAX_AMP is incremented/decremented by 12'h111 each time signal from respective button is received.



Rectified Sine

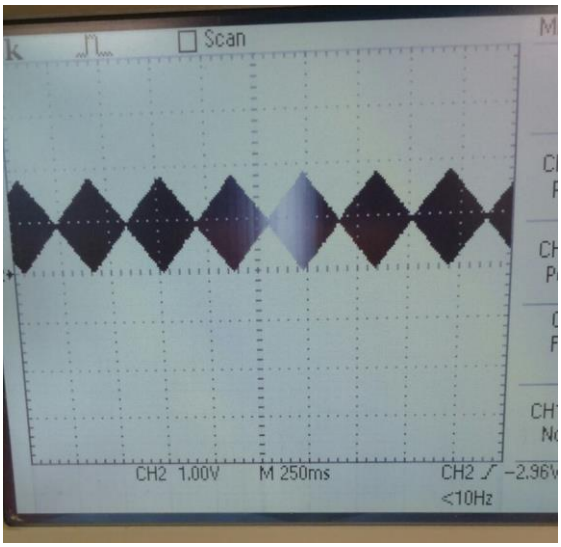
Same is done for Rectified sine, but the phase range is 0 to pi.





DIAMOND

Works the same as TRIANGLE2, but with a 50Mhz clock. The amplitude is fixed at 12'h8888.



DC offset

Increment or decrement DC_offset variable, or set it to 0v when reset to origin switch == 1. DC_offset is connected to DATA_2 of DAC. The bigger the value of DC_offset, the greater the negative offset of wave from origin will be. i.e wave is shifted down. Voltage output = DATA_A – DC_offset.

7 segment 12 bit to voltage display

4095 / 33 = 124. 124 in bits represents 0.1v. Divide DATA_A or amplitude, which is in 12 bits, by 124. This will give the actual voltage *10. Separate the ones from the tenths place by using modulus 10, and separate the tenths from the once place by dividing by 10.

Feedback

Great module, but would appreciate more info for report