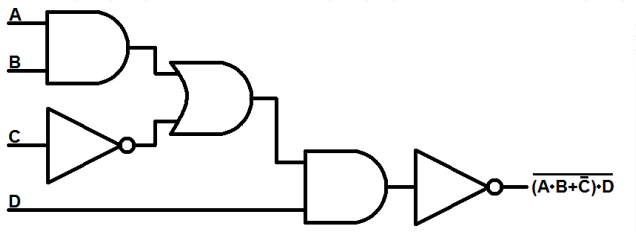
Direct Digital Synthesis Function Generator



Asst. Prof. Dr. Sumek Wisayataksin

Department of Electronics Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

Revised by Sirus Thongarun
28 Jun 2023

KMITL
พระจอมเกล้าลาดกระบัง

Overview

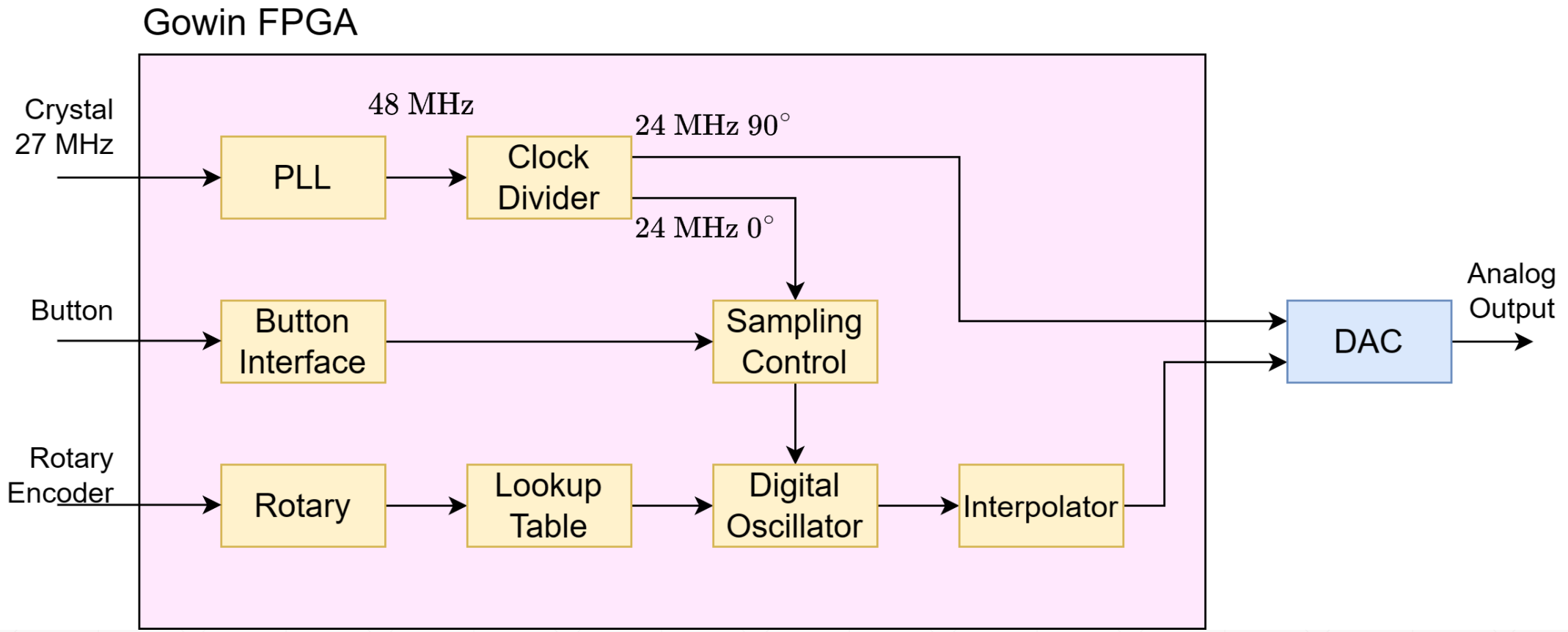
Digital Direct Synthesis Function Generator

Specification and Requirement

KMITL
พระจอมเกล้าลาดกระบัง

1. Generate Sinusoidal Signal with frequency range from 10 Hz to 200 kHz.
2. Capable of generate Sinusoidal Signal with Peak-to-Peak voltage of 30 Volt.
3. Capable of generate Sinusoidal Signal with Offset from -5 Volt to 5 Volt.
4. Total harmonic distortion below 5% at all frequency.

Block Diagram

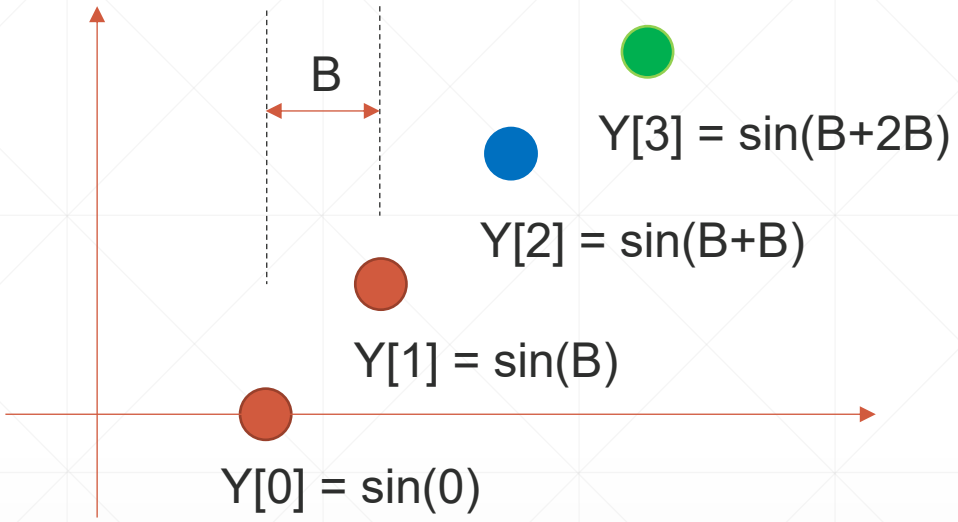


KMITL
พระจอมเกล้าลาดกระบัง

Digital Oscillator

An equation that can maintain signal without input

Digital Oscillator



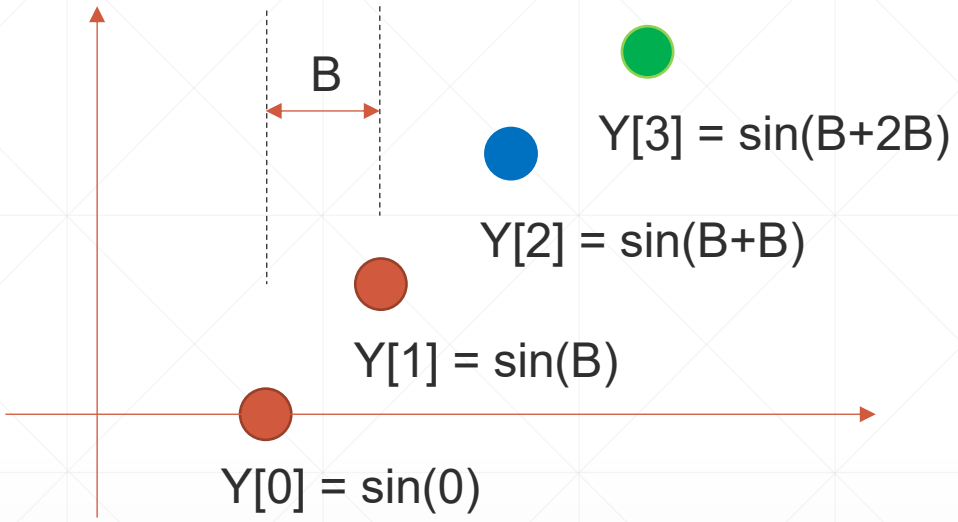
$$y[2] = \sin(B + B)$$

$$y[2] = \sin(B)\cos(B) + \sin(B)\cos(B)$$

$$y[2] = 2\cos(B)\,\boldsymbol{y[1]}$$

$$* \cos(A)\sin(B) = \frac{1}{2}[\sin(A + B) - \sin(A - B)]$$

Digital Oscillator



$$y[3] = \sin(2B + B)$$

$$y[3] = \sin(2B)\cos(B) + \sin(B)\cos(2B)$$

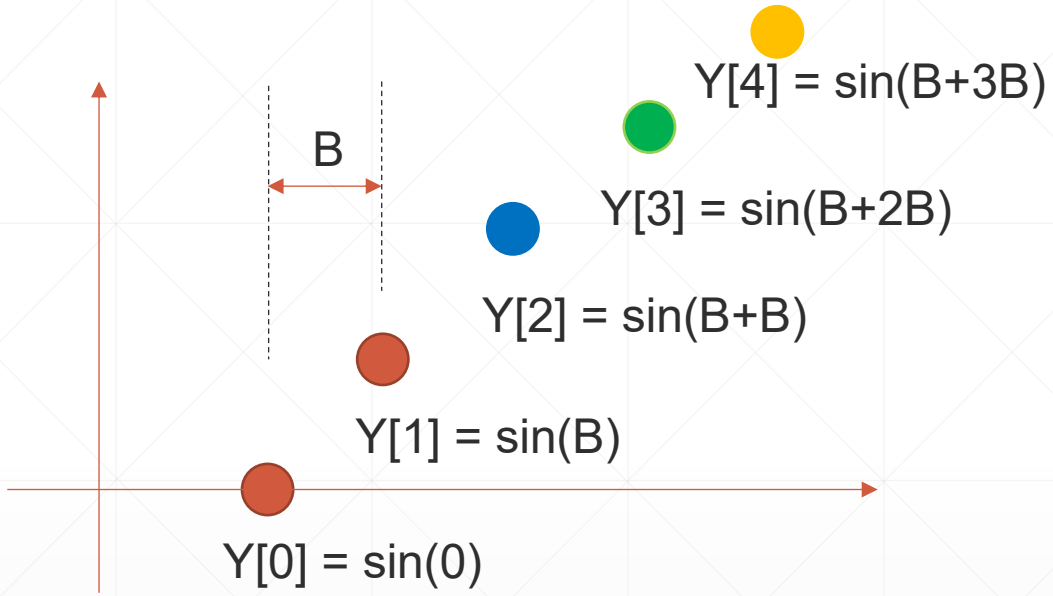
$$y[3] = \cos(B)\,y[2] + \left(\frac{\sin(3B) - \sin(B)}{2}\right)$$

$$y[3] = \cos(B)\,y[2] + \left(\frac{y[3] - y[1]}{2}\right)$$

$$y[3] = 2\cos(B)\,y[2] - y[1]$$

$* \cos(A)\sin(B) = \frac{1}{2}[\sin(A + B) - \sin(A - B)]$

Digital Oscillator



$$y[4] = \sin(3B + B)$$

$$y[4] = \sin(3B)\cos(B) + \sin(B)\cos(3B)$$

$$y[4] = \cos(B)\, y[3] + \left(\frac{\sin(4B) - \sin(2B)}{2}\right)$$

$$y[4] = \cos(B)\, y[3] + \left(\frac{y[4] - y[2]}{2}\right)$$

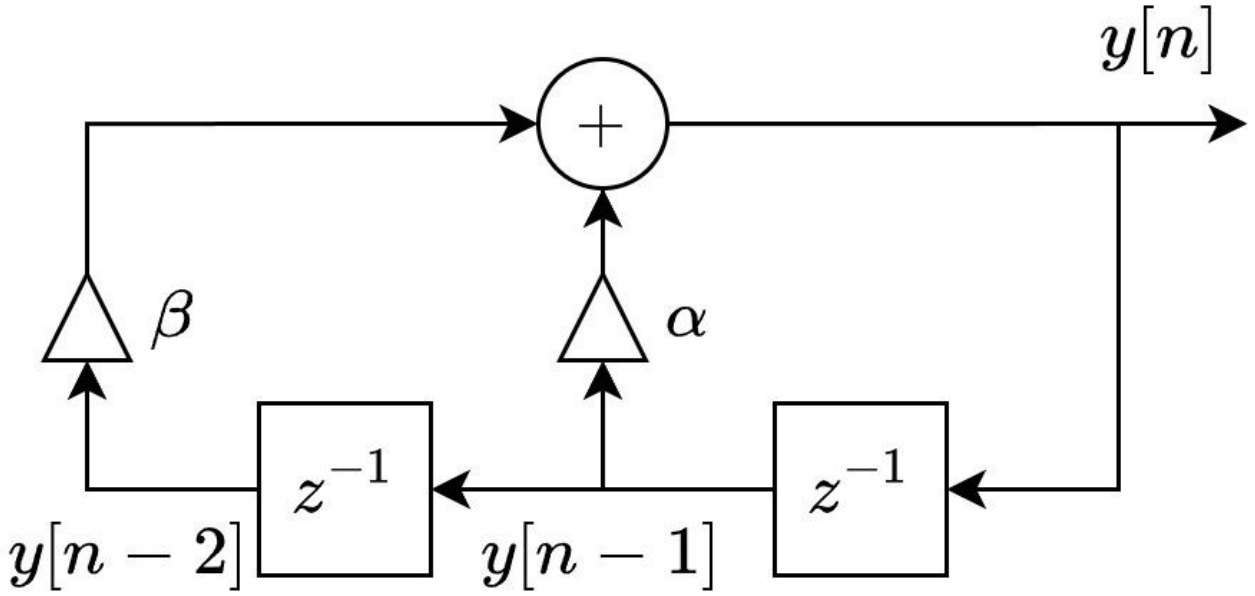
$$y[4] = 2\cos(B)\, y[3] - y[2]$$

$$y[n] = 2\cos(B)\, y[n-1] - y[n-2]$$

$$* \cos(A)\sin(B) = \frac{1}{2}[\sin(A+B) - \sin(A-B)]$$

Block Diagram for D.E.

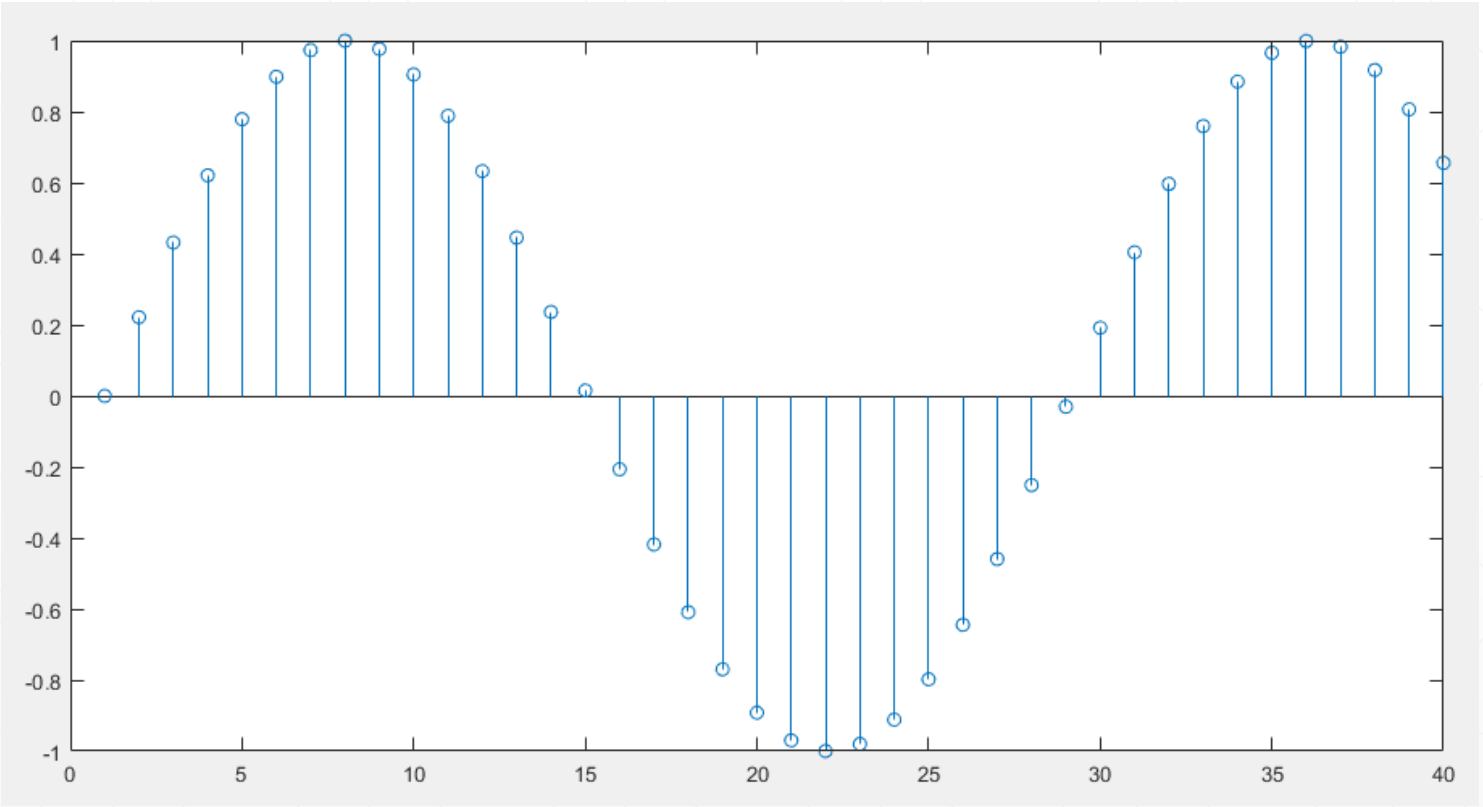
$$y[n] = 2\cos(B)y[n-1] - y[n-2]$$
$$y[0] = 0$$
$$y[1] = \sin(B)$$



Example of D.E. using MATLAB (1)

- Generate sinusoidal signal with frequency of 853 kHz

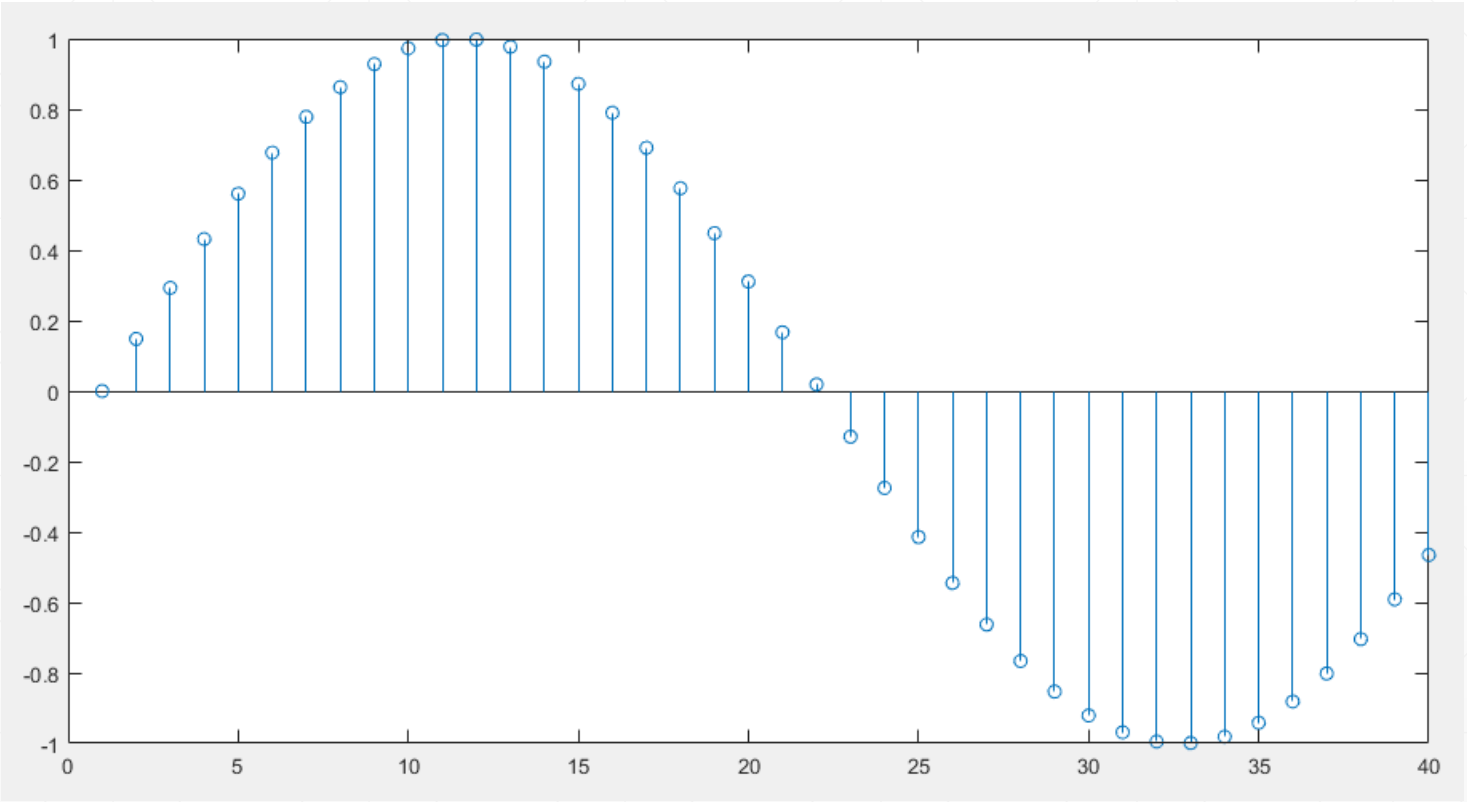
```
f = 853e3;
fs = 24e6;
Ts = 1/fs;
B = 2*pi*f*Ts

alpha = 2*cos(B);
beta = -1
y(1) = 0;
y(2) = sin(B);

for i=3:500
   y(i) = alpha*y(i-1) + beta*y(i-2);
end
```

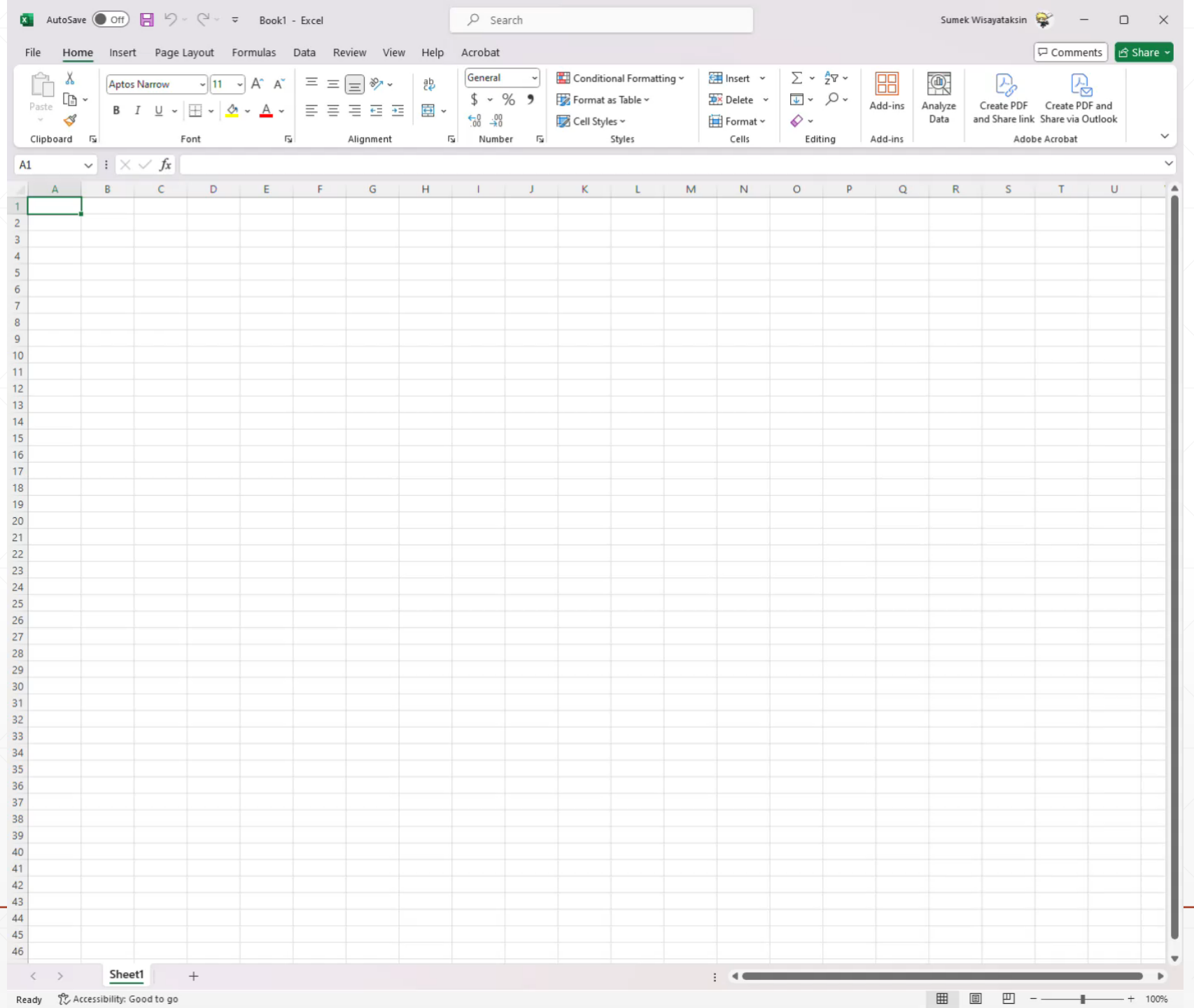


Example of D.E. using MATLAB (2)

- Generate sinusoidal signal with frequency of 568 kHz

```
f = 568e3;
fs = 24e6;
Ts = 1/fs;
B = 2*pi*f*Ts

alpha = 2*cos(B);
beta = -1
y(1) = 0;
y(2) = sin(B);

for i=3:500
   y(i) = alpha*y(i-1) + beta*y(i-2);
end
```



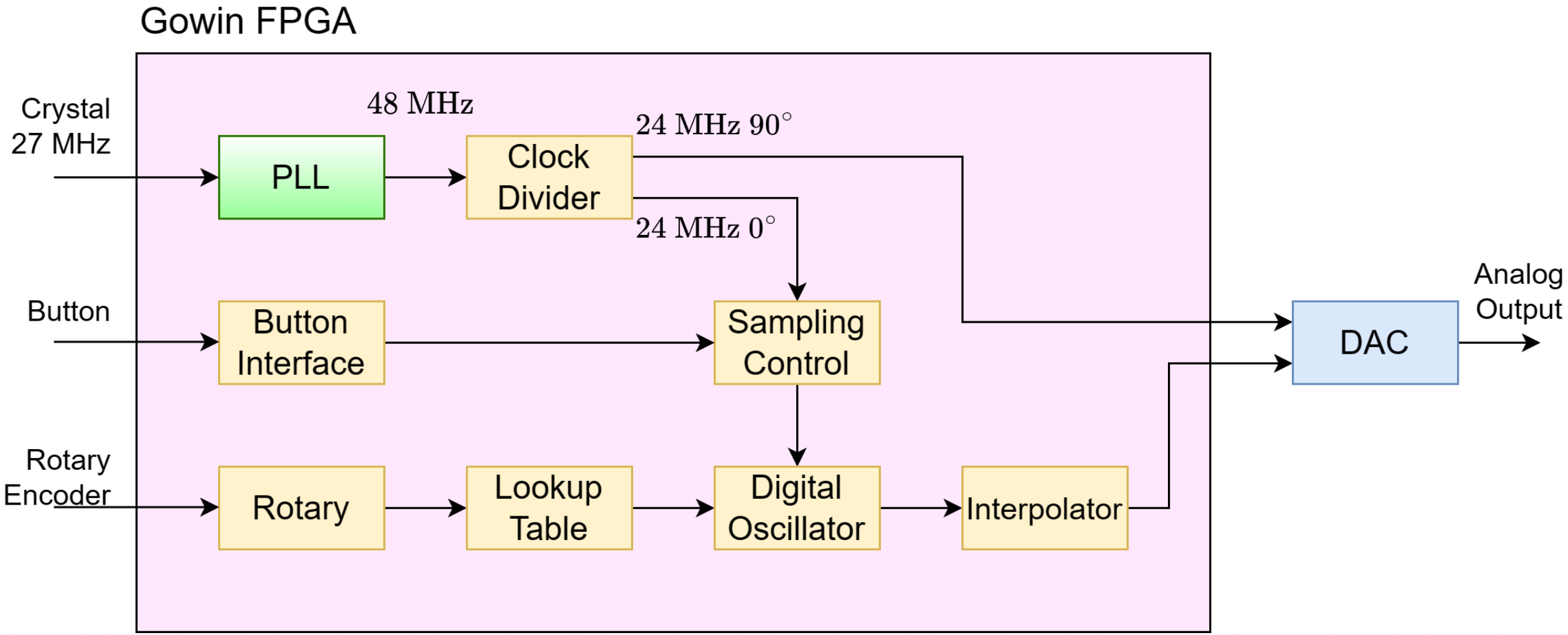


KMITL
พระจอมเกล้าลาดกระบัง

Phase-Locked Loop

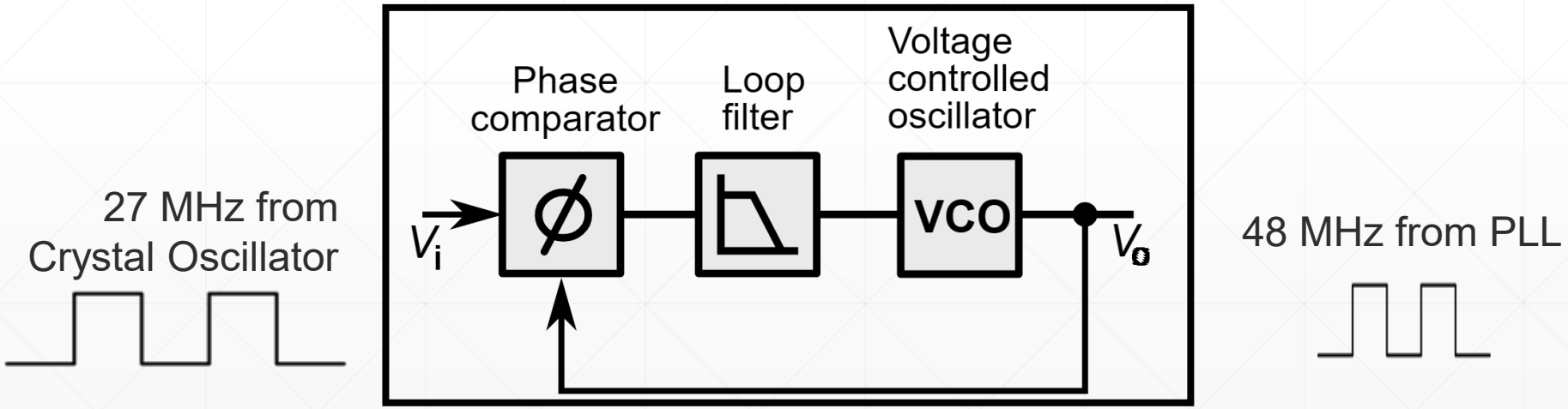
A thing for generate various clock frequency.

DDS Progress Update



Why Phase-Locked Loop?

- Phase-Locked Loop can be used to generate a clock at any frequency (How?).
- Because onboard Crystal Oscillator generate 27 MHz but we need 48 MHz clock for generate two 24 MHz clock (Why 48MHz?). So, we must use PLL to generate the desired clock.

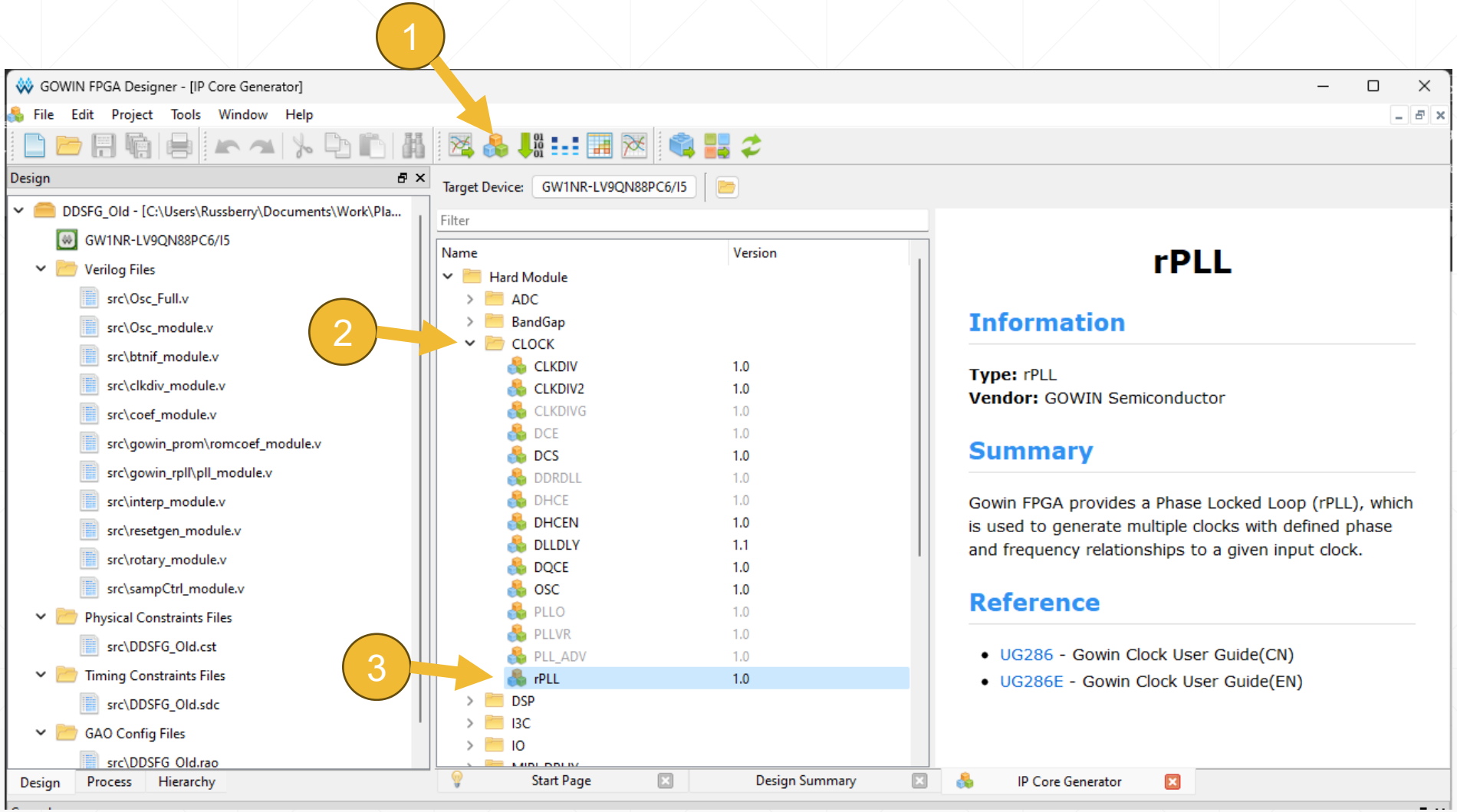


PLL Module

PLL Block

Note that “reset” is “Active High! Please make sure that you have already transform it before drive the reset signal to PLL module.

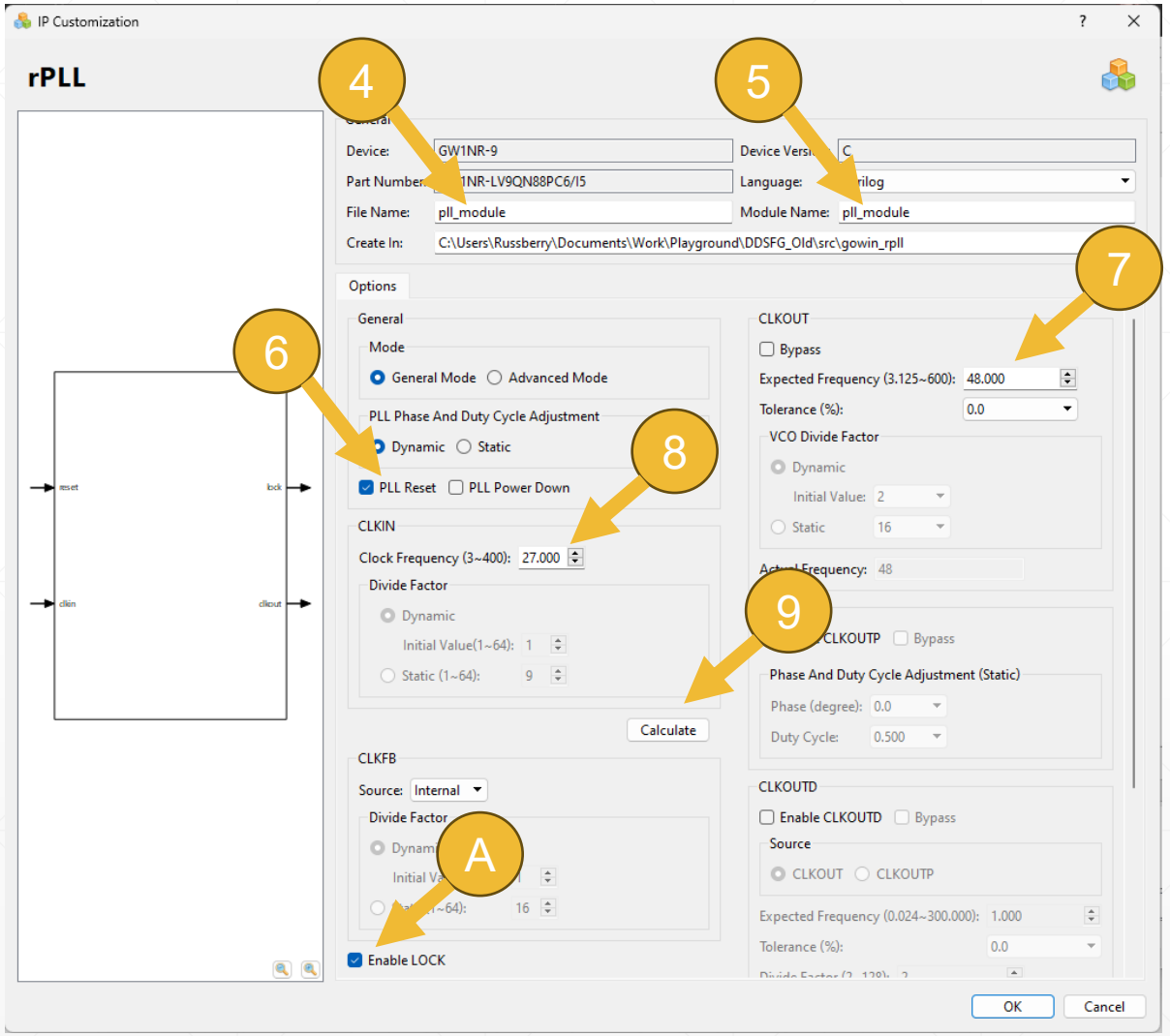
PLL IP-Core Generation (2)



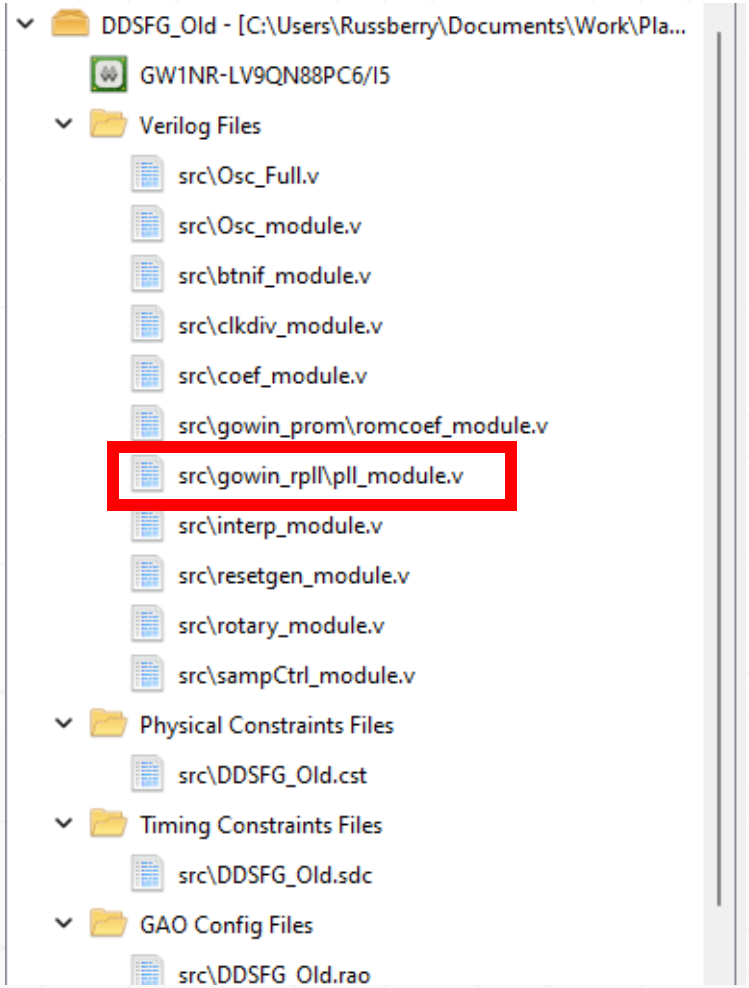
PLL IP-Core Generation (2)

KMITL
พระจอมเกล้าลาดกระบัง

4. File Name
5. Verilog Module Name
6. Enable PLL External Reset
7. Set Output Frequency (48 MHz)
8. Set Input Frequency (27 MHz)
9. Press “Calculate” button
10. Enable “LOCK” signal



Integrate PLL to RTL




```
pll_module pll(
        .clkin          (External Clock),
        .reset          (~PLL Reset),
        .clkout         (48 MHz Clock),
        .lock           (PLL Lock)
);
```

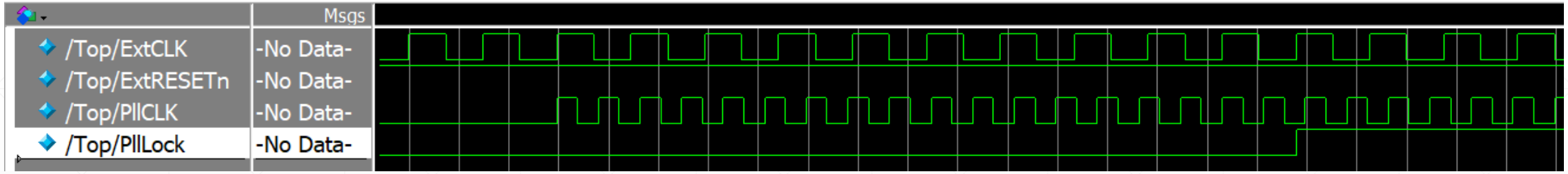
KMITL
พระจอมเกล้าลาดกระบัง

Clock Divider

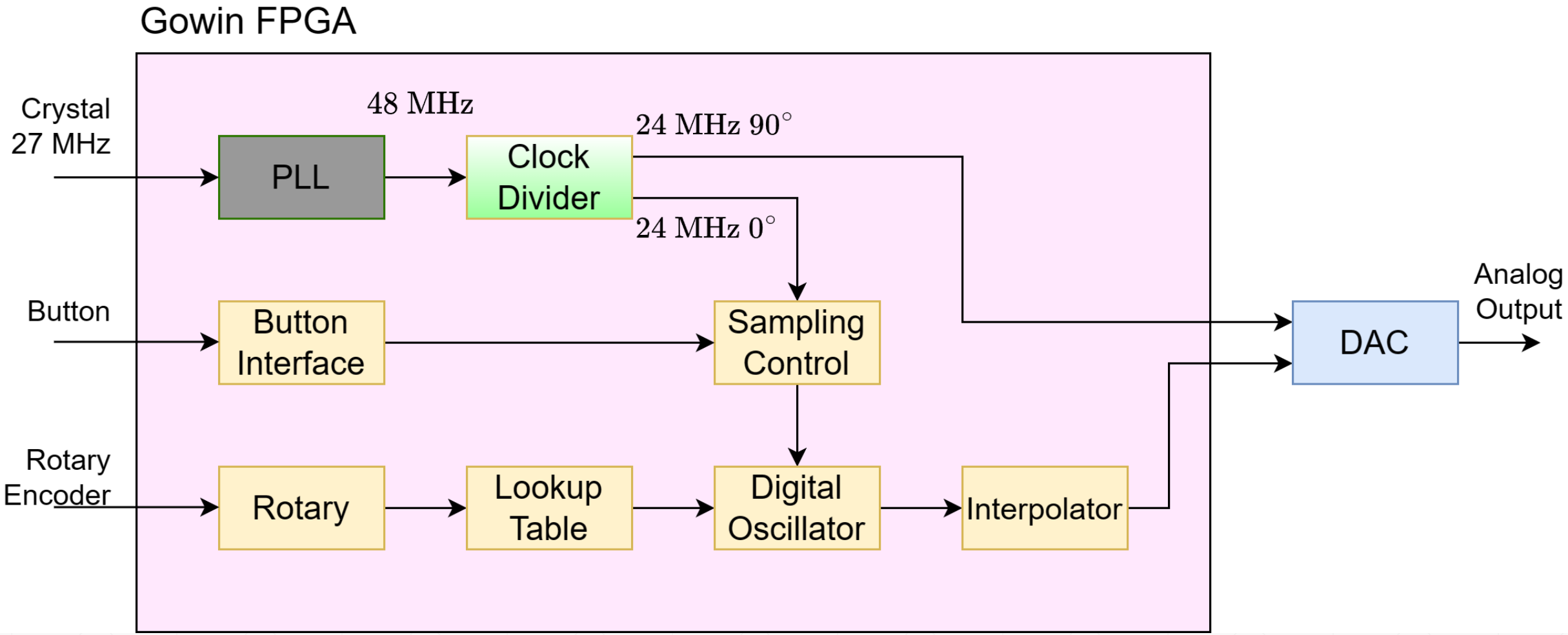
Just a Flip-flop block!

PLL Summary

- Generate 48 MHz clock from 27 MHz clock from External Crystal Oscillation

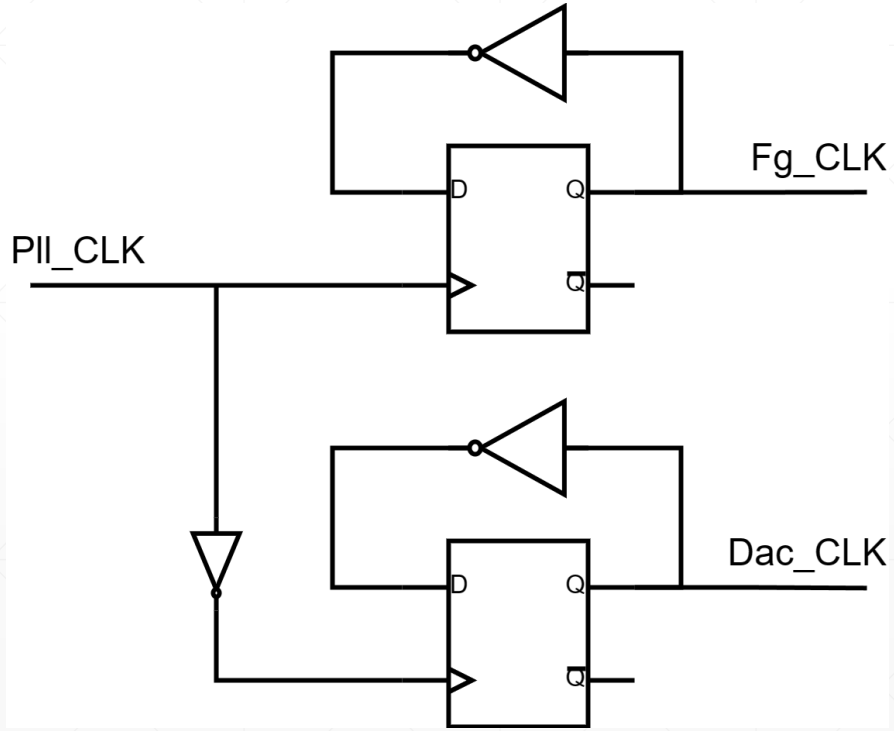


DDS Progress Update

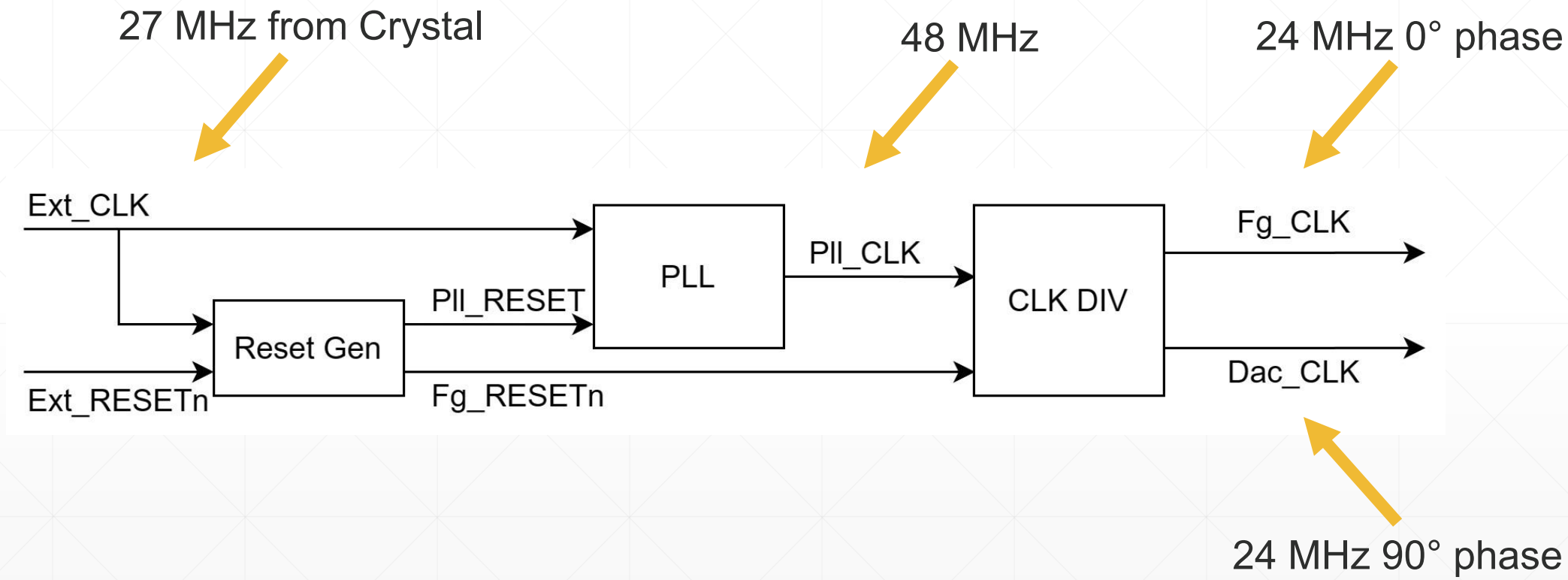


Clock Divider Block Diagram

- The basic structure of Clock Divider is a Flip-Flop which can be accomplished the 90 degrees phase shift by using different clock polarity.

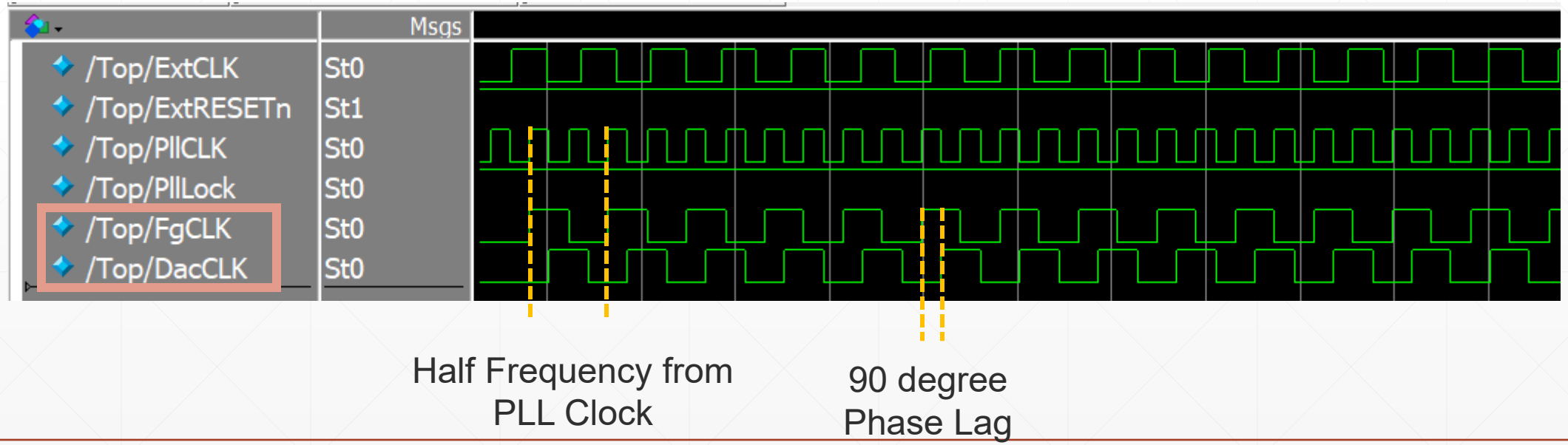


DDS Clock System



Clock Divider Summary

- Generate two clock which have a half frequency of PLL Clock.
- Two clock signal have the phase different of 90 degrees. (The lag one is for DAC Clock)



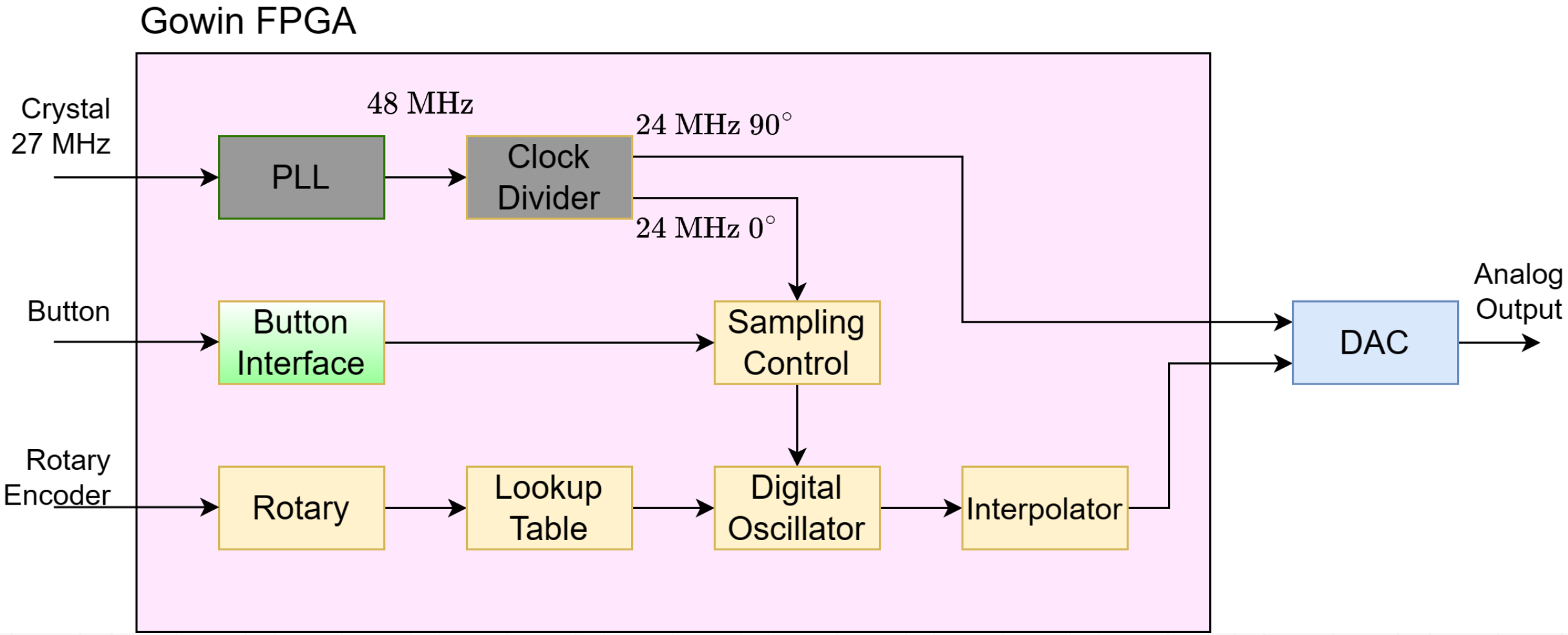
KMITL
พระจอมเกล้าลาดกระบัง

Button Interface

Who cares about the button bounce?

DDS Progress Update

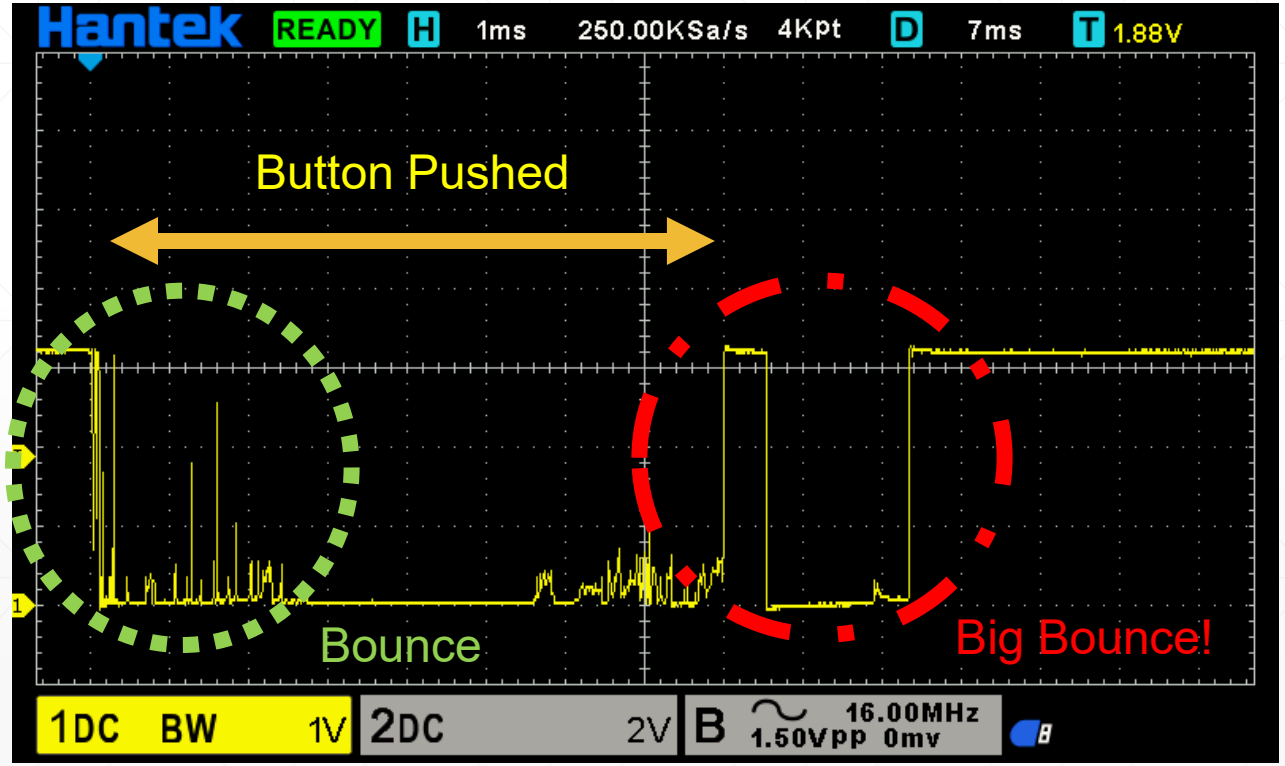
KMITL
พระจอมเกล้าลาดกระบัง



Why we need button interface? (1)

KMITL
พระจอมเกล้าลาดกระบัง

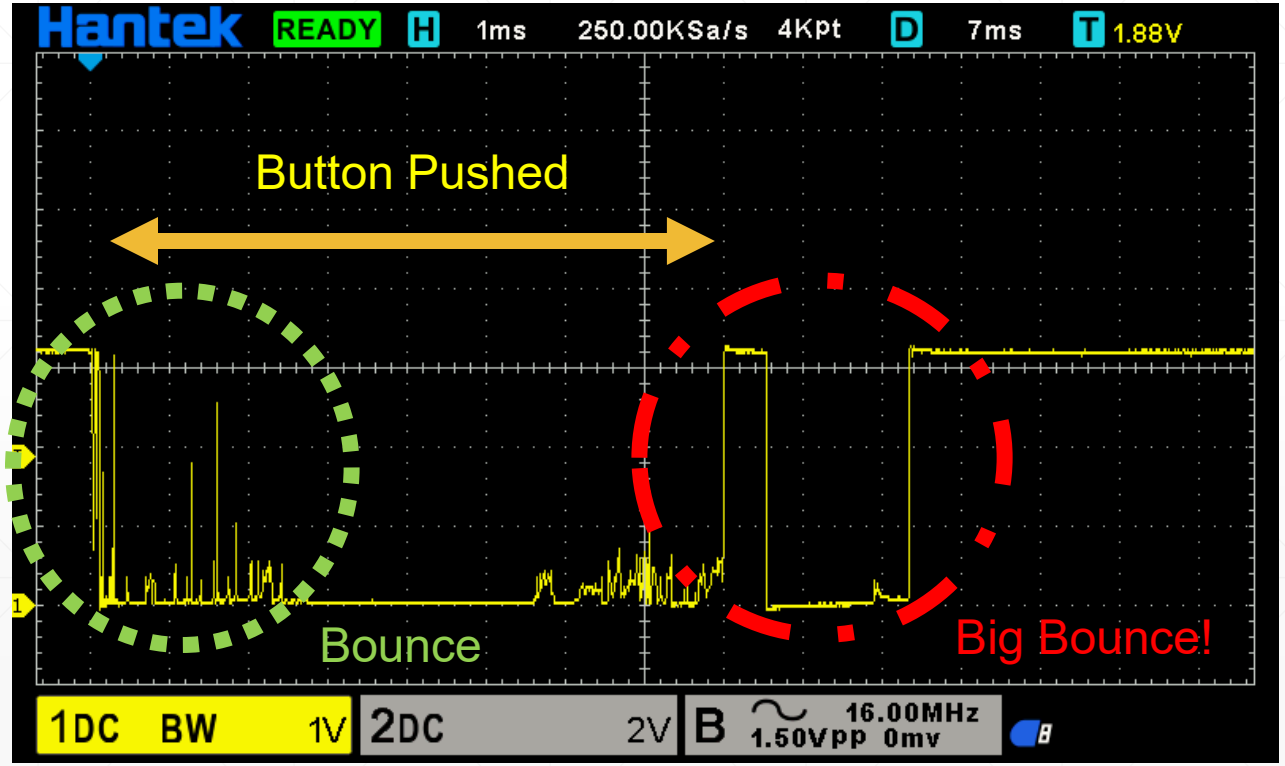
- When we push the button, one thing that always annoy us is the bounce! (What cause?)



Why we need button interface? (2)

KMITL
พระจอมเกล้าลาดกระบัง

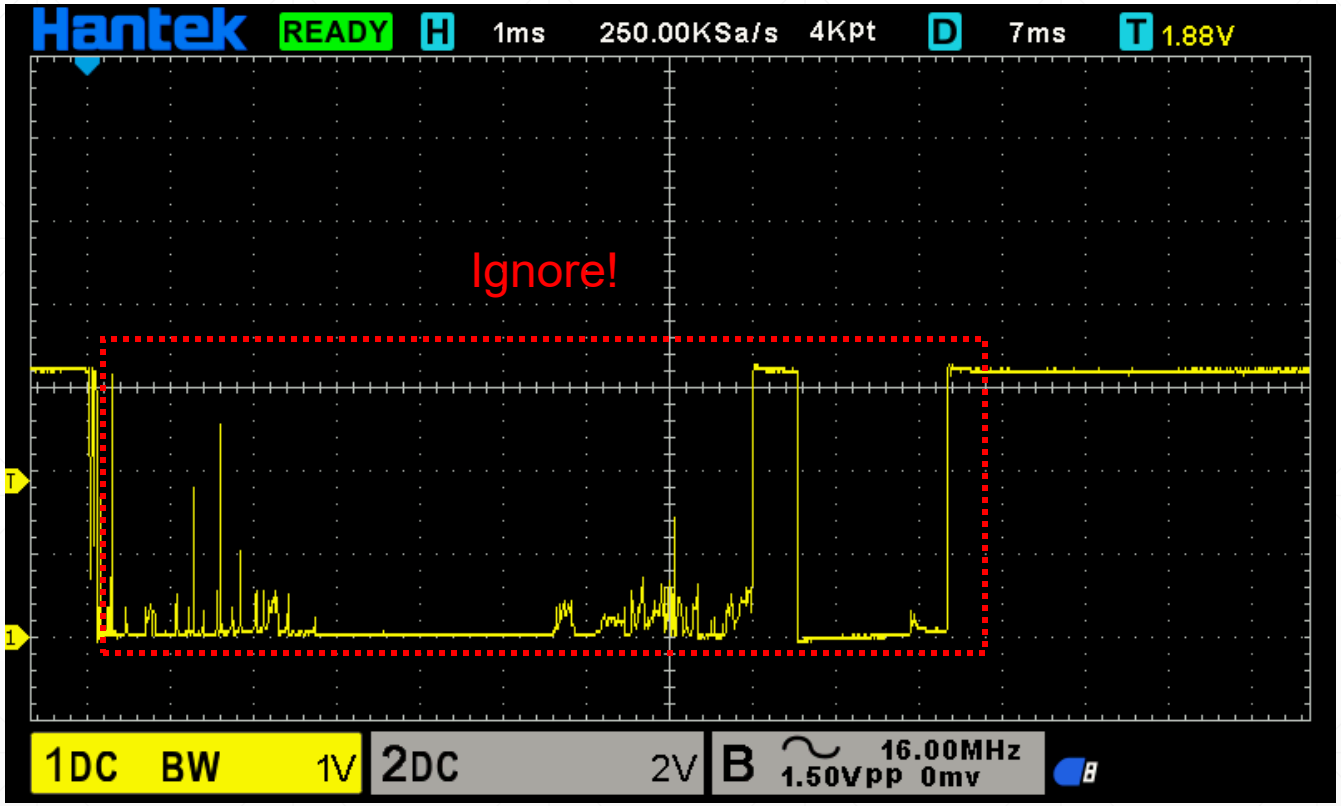
- One way to solve this problem is just ignore it and “Generate our own button push signal that last for a clock cycle”!



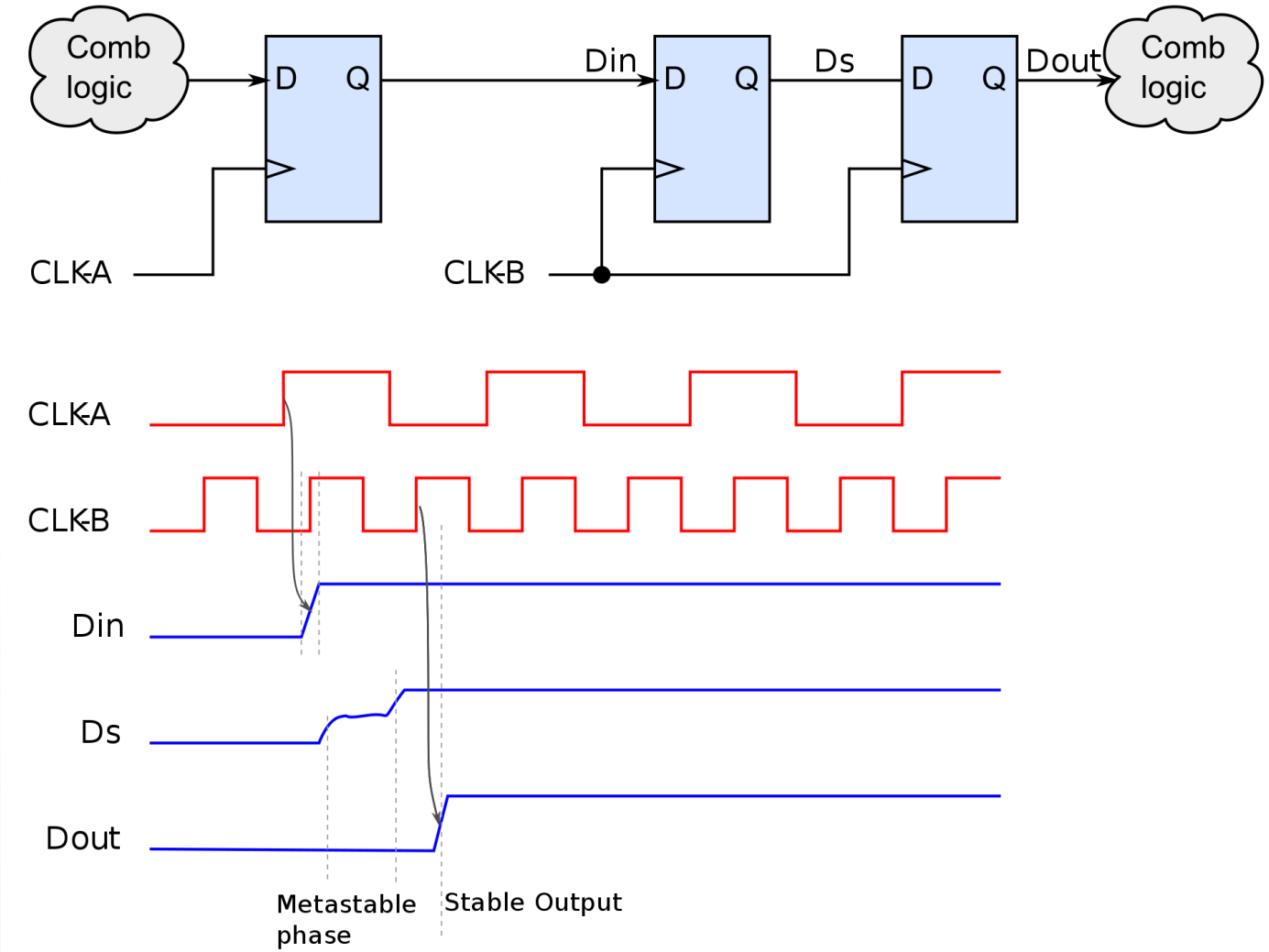
Bounce Ignorance

In order to ignore the bounce, here are simple solution.

1) Detect only falling edge of the button.
2) Ignore any falling edge for a period of time.

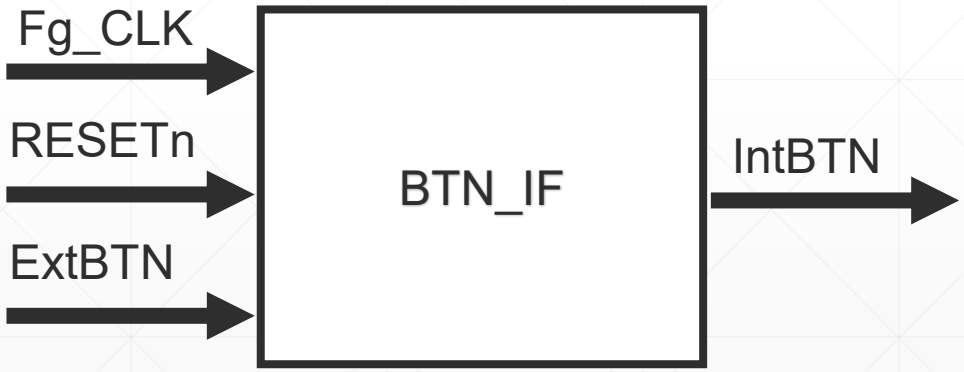


Metastability

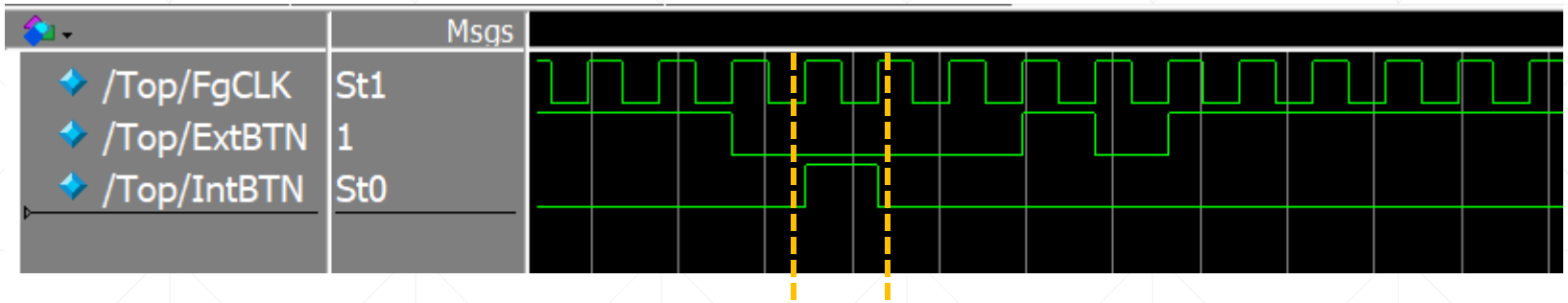


Button Interface Summary

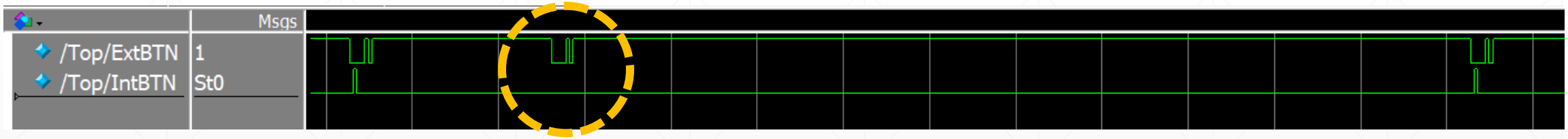
- Generate a pulse whenever the button is pushed (falling edge only) and ignore any falling edge signal for a short period of time.



Button Interface Expected Behavior



Generate a pulse when detect the falling edge of ExtBTN



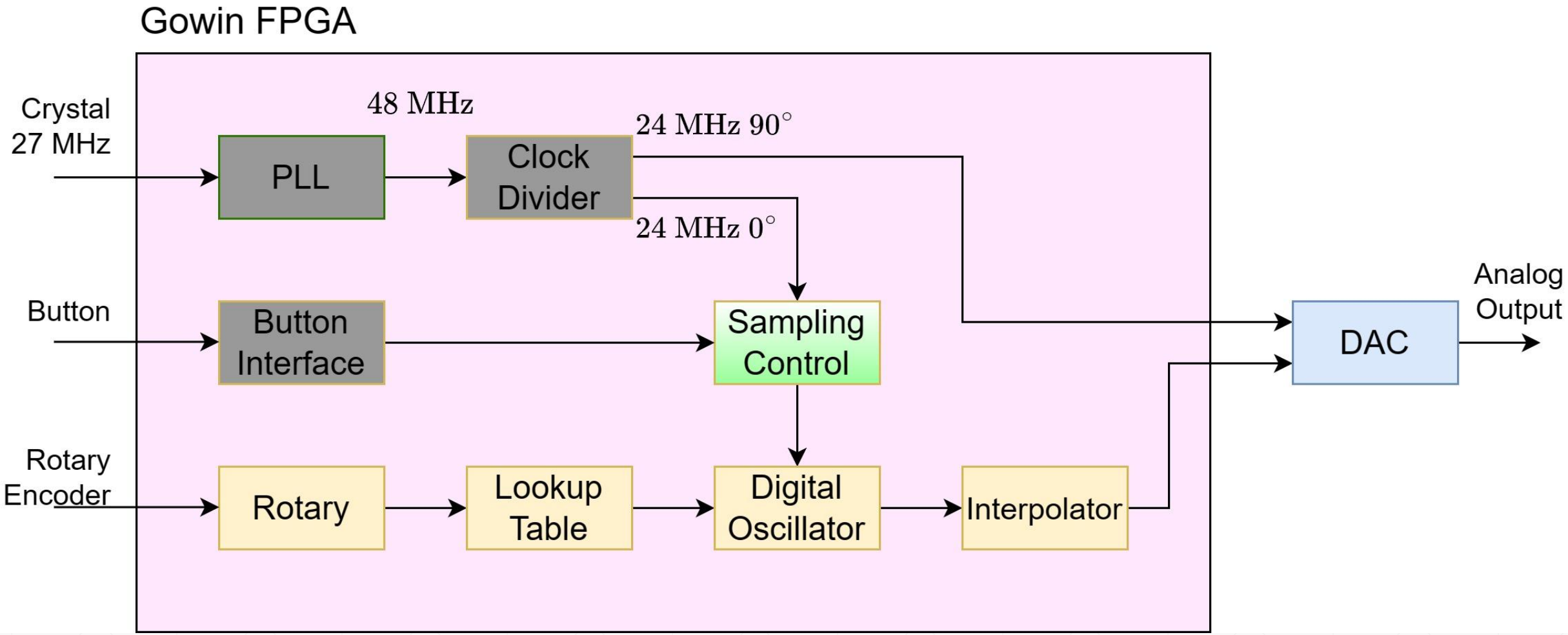
Reject button bounce

KMITL
พระจอมเกล้าลาดกระบัง

Sampling Control

A block for control the time in difference equation!

DDS Progress Update



Why we need Sampling Control (2)

KMITL
พระจอมเกล้าลาดกระบัง

- As we said before, we use sampling frequency (f_s) of 24 MHz for own D.E. to generate any frequency of sinusoidal signal.
- For example, if we need to generate sinusoidal signal with frequency of 500kHz. We must use a parameter

$$\alpha = 2\cos\left(\frac{2\pi f}{f_s}\right) = 2\cos\left(\frac{2\pi(500k)}{24M}\right) = 0.9914$$

$$y[1] = \sin\left(\frac{2\pi f}{f_s}\right) = 2\sin\left(\frac{2\pi(500k)}{24M}\right) = 0.2611$$

Why we need Sampling Control (2)

KMITL
พระจอมเกล้าลาดกระบัง

- What if we need to generate 50kHz signal without change the value of parameter as show earlier?

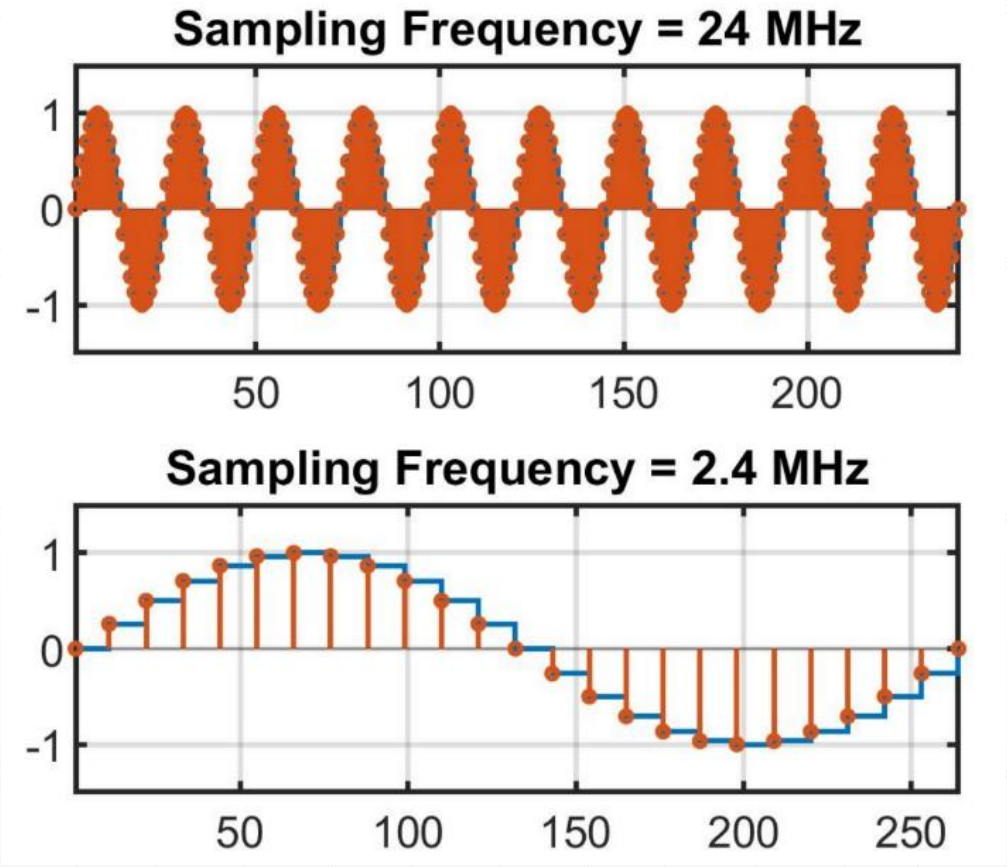
$$\alpha = 2\cos\left(\frac{2\pi f}{f_s}\right)$$

$$y[1] = \sin\left(\frac{2\pi f}{f_s}\right)$$

- From the equation above, if we decrease the value of desired frequency by factor of 10. We should decrease the value of sampling frequency by the same factor to preserve the value of parameter.

Why we need Sampling Control (3)

- So that we need a something for control how fast our D.E. will update the value to create a wide frequency range of sinusoidal signal without change the parameter α and $y[1]$. (Why we not change the parameter instead?)



Sampling Control Mode (1)

KMITL
พระจอมเกล้าลาดกระบัง

- We can assume that we have only the parameter α and $y[1]$ in range of 100 kHz to 100 MHz only. So, we will control the sampling frequency instead for generate low frequency sinusoidal signal.

- Assume that we have 5 different mode of operation which is
  - Mode 0 : Frequency Range of 100 kHz to 1 MHz (Generate Enable signal all the time)
  - Mode 1 : Frequency Range of 10 kHz to 100 kHz (Generate Enable signal every 10 clock cycles)
  - Mode 2 : Frequency Range of 1 kHz to 10 kHz (Generate Enable signal every 100 clock cycles)
  - Mode 3 : Frequency Range of 100 Hz to 1 kHz (Generate Enable signal every 1000 clock cycles)
  - Mode 4 : Frequency Range of 10 Hz to 100 Hz (Generate Enable signal every 10000 clock cycles)

Sampling Control Mode (2)

KMITL
พระจอมเกล้าลาดกระบัง

- In order to change the mode, we use the button signal from “Button Interface” module to change.

- Requirement for Sampling Control Mode Change
  - Initial mode for our DDS after reset is mode 0 (100 kHz to 1MHz)
  - Whenever there is a pulse from button interface, the mode will be increase by 1.
  - When current mode is mode 4 (10 Hz to 100 Hz) and there is a pulse from button interface. The next mode will be back to mode 0.

Sampling Control Ready Indication

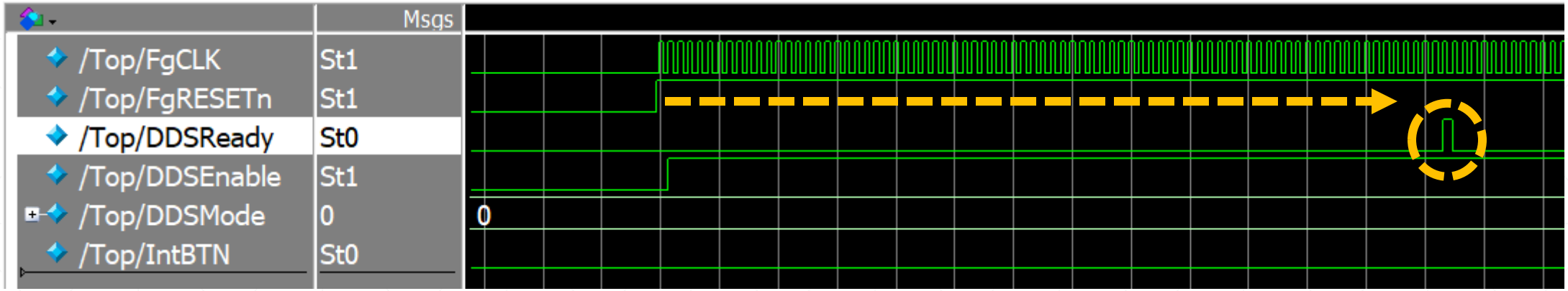
KMITL
พระจอมเกล้าลาดกระบัง

- Before start the Oscillator Module (D.E.), for generate the sinusoidal signal, we need to take sometimes to wait our DDS system stable.

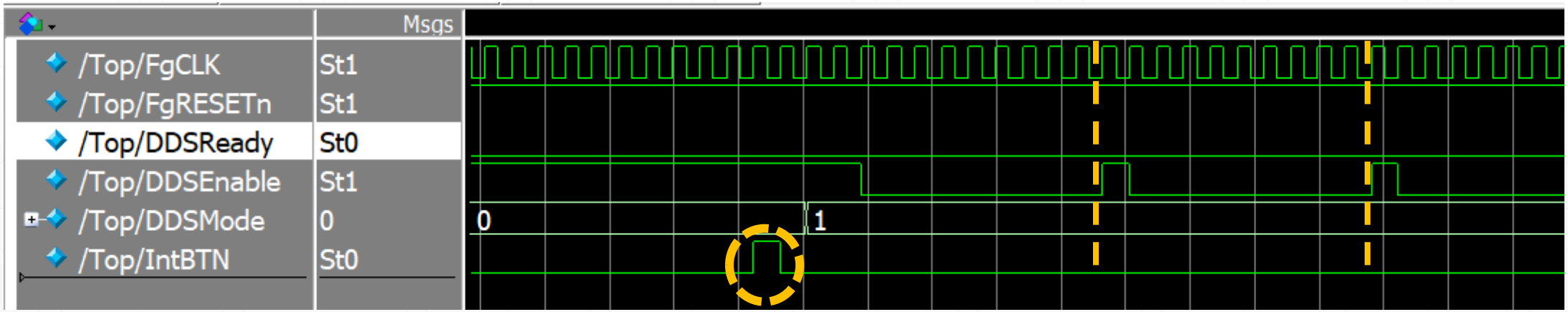
- So that we will create a “Ready” signal for tell the Oscillator Module to start the calculation.

- Requirement for Sampling Control Ready Indication
  - The reset value of Ready signal is 0 (Not ready)
  - Whenever there are 80 clock cycles passed after reset, Ready signal is 1 (Ready)

Sampling Control Expected Behavior



A pulse of Ready signal after 80 clocks from RESET state



After detect pulse of button, Mode change from 0 to 1 and Enable signal change from constant to a pulse width period around 10 clock

Sampling Control Summary

- Can change the mode whenever the button is pushed
- Can generate “Enable” signal by the requirement as show earlier.
- Can generate “Ready” signal whenever there are 80 clock cycles passed after reset.

KMITL
พระจอมเกล้าลาดกระบัง

Fixed-point Number

Method for store decimal in binary number system.

Fixed-point Number (1)

KMITL
พระจอมเกล้าลาดกระบัง

- Let's recall Binary Number System

Bits	3	2	1	0
Bit Value	8	4	2	1

- If we want to convert 5_{10} into 4-bit binary number system. The result will be like this

Bits	Bit Value	5_{10} Value
3	8	0
2	4	1
1	2	0
0	1	1

Fixed-point Number (2)

KMITL
พระจอมเกล้าลาดกระบัง

- However, what if we want to store the value of 0.375?
  - No, we cannot do that!

Bits	Bit Value	0.375_{10} Value
3	8	0
2	4	0
1	2	0
0	1	0

- There is no way to store decimal in standard binary number system.
- The solution for that problem is “Scale Up by 2^n” the value so that our binary number system can store.

Fixed-point Number (3)

- If we choose to scale up by 2^3, so that 0.375 will be 3. and its binary value will be 0011_2 instead of 0000_2 like the previous slide.
- Notice that not only decimal part is scaled up but also the integer part!! So the binary table value will be changed.

Bits	Bit Value	0.375_{10} Value
3	8	0
2	4	0
1	2	0
0	1	0

Before Value Scaled

Bits	Bit Value	$0.375_{10}*2^3$ Value
3	8	0
2	4	0
1	2	1
0	1	1

After Value Scaled

Fixed-point Number (4)

KMITL
พระจอมเกล้าลาดกระบัง

- The other way to think is the bit value is scale down!

- We will use same scaling factor from previous slide, that is, 2^3

Bits	Bit Value	0.375_{10} Value
3	8	0
2	4	0
1	2	0
0	1	0

Before Bit Value Scaled

Bits	Bit Value / 2^3		0.375_{10} Value
3	$8/2^3$	= 1	0
2	$4/2^3$	= 0.5	0
1	$2/2^3$	= 0.25	1
0	$1/2^3$	= 0.125	1

After Bit Value Scaled

Fixed-point Number Summary

KMITL
พระจอมเกล้าลาดกระบัง

- An operation that we “Scale Up the value” or “Scale Down the bit value” is the method for allow decimal number to store in binary number system which called Fixed-point Number System

- From previous slide, we scaled up the value by 2^3 so that, our “virtual” decimal point in binary system is at the left side of bit 3.

Number Multiplication and its digits

KMITL
พระจอมเกล้าลาดกระบัง

- All of number multiplication, the maximum digits after multiplication will be double!

- Let say we have 2 digits decimal number system. For example, 99, 18, 27, 55. If we multiply 2 digits decimal number together. The result will have maximum of 4 digits decimal number!

- For example, 99 * 99 = 9,801
  2 dig 2 dig 4 dig

Number Multiplication and its digits

KMITL
พระจอมเกล้าลาดกระบัง

- This also true with decimal point number.
- Let's say 99.99 * 99.99 = 9,998.0001
- To summarize, any number system (Binary, Octal, Decimal, Hexadecimal) will have the double number of digits or bits of its original value after multiply.

KMITL
พระจอมเกล้าลาดกระบัง

Oscillator Module

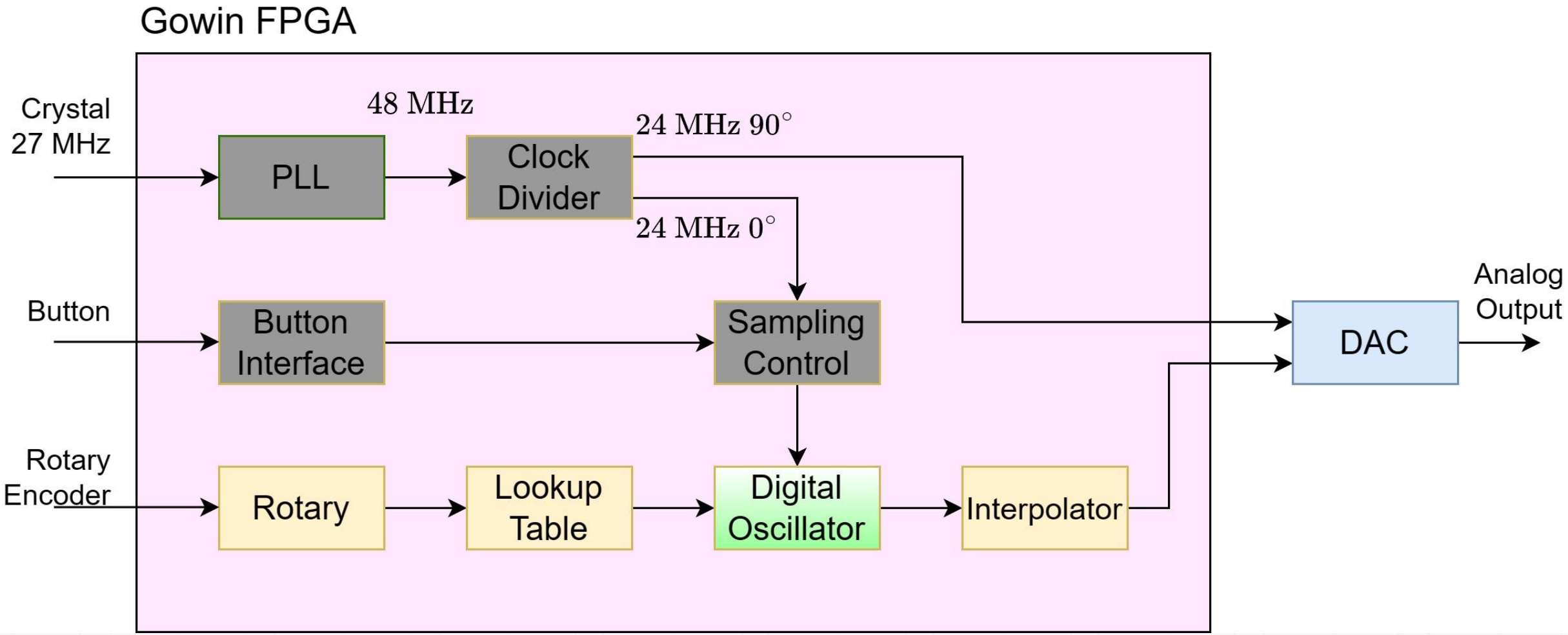
A module for calculate the Difference Equation.

Selected Fixed-Point Value (1)

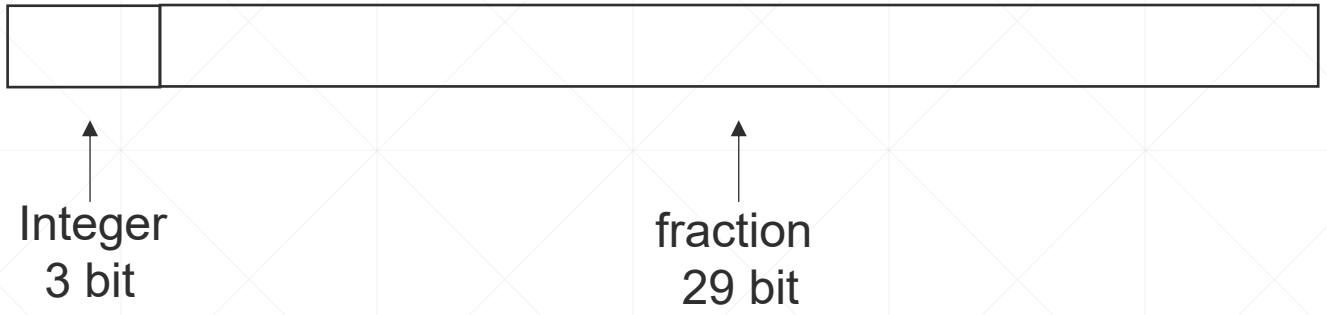
KMITL
พระจอมเกล้าลาดกระบัง

- In this project, we will use 32-bit binary number and use 2^{29} as scaling factor.
- So that if we want to represent any base 10 number, we should multiply by 2^{29}
- The result Multiplication will be 64-bit but we will truncate to 32-bit. (Why)

DDS Progress Update



Oscillator : Fix point decimal



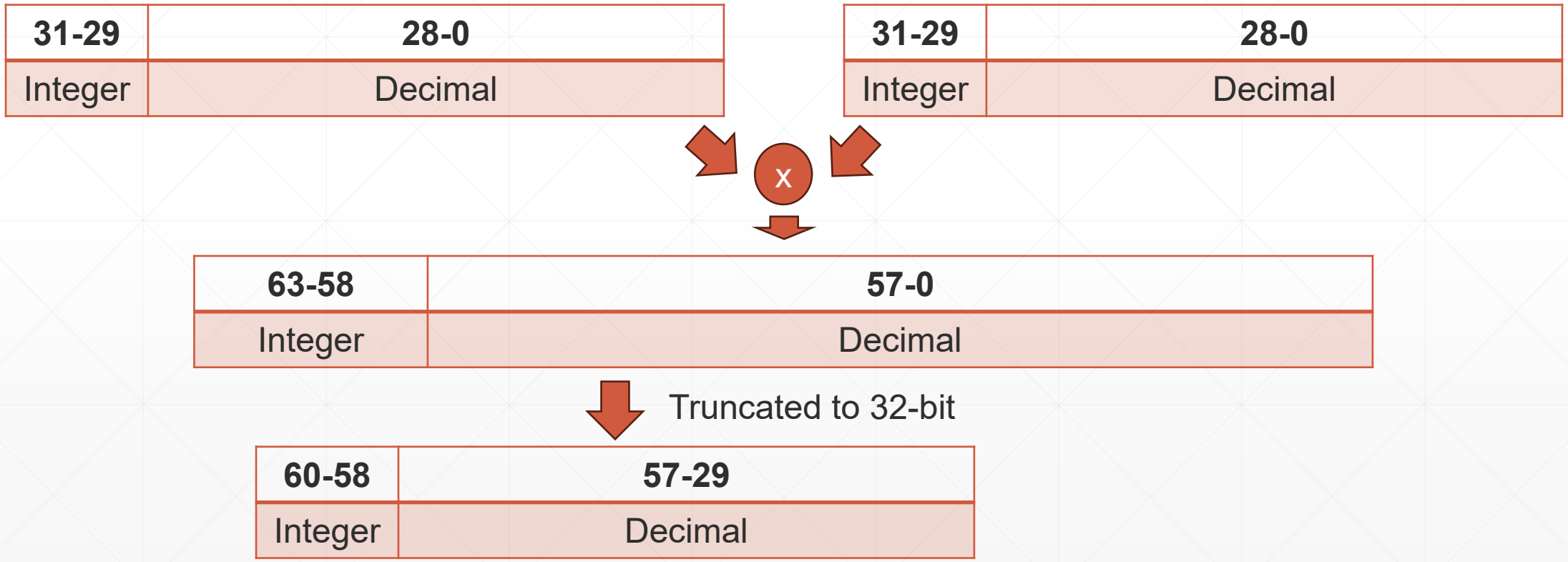
How to convert fraction to integer 32 bit

Ex. $0.42458 \times 2^{29} = 227,944,651 = 0x0D9628CB$

$-0.74214 \times 20^{29} = -398,433,378 = 0xE840639E$

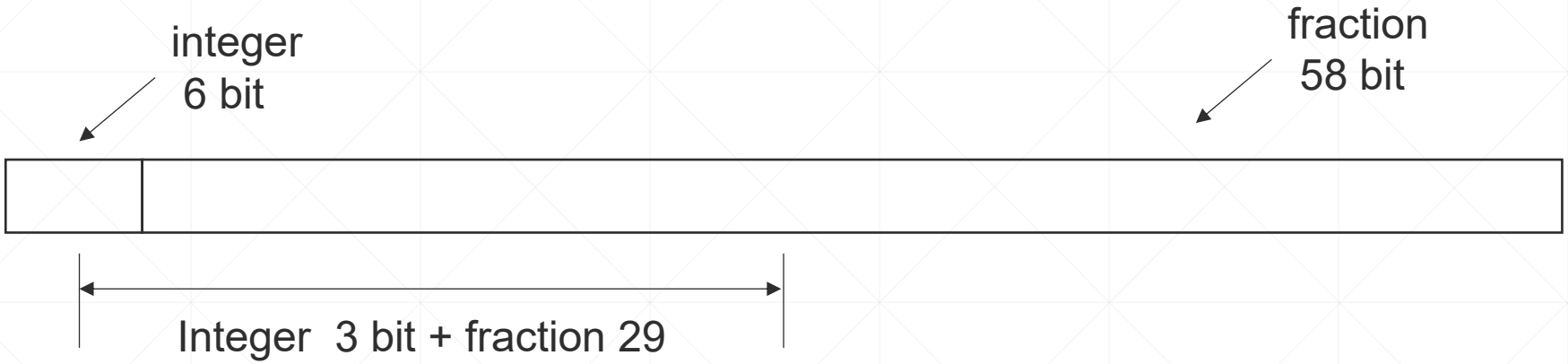
Selected Fixed-Point Value (2)

- When we use multiplication operator in this project. The result must be truncated to 32-bit with this form.



Oscillator : Fraction Multiplier

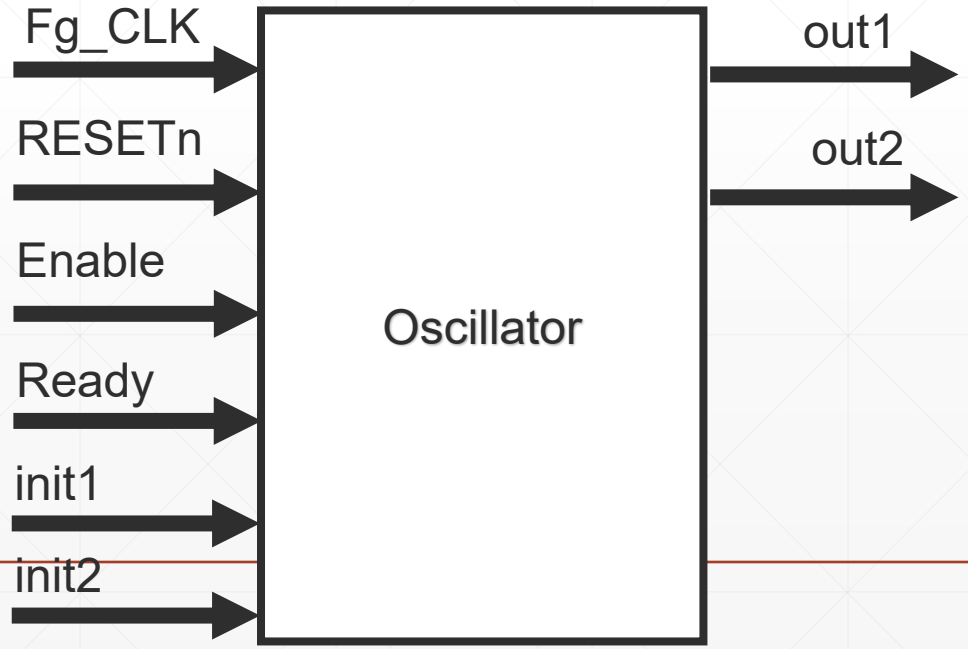
```
reg [31:0] a;
reg [31:0] b;
reg [63:0] c;
reg [31:0] out1_a;
```



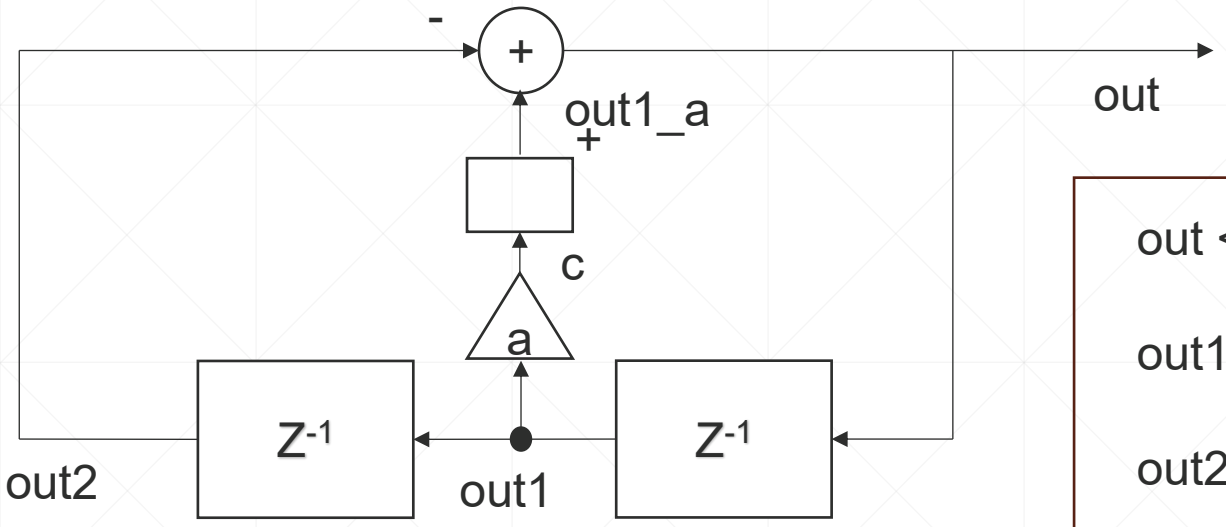

```
always@(*) begin    // combi
        c        <= $signed(a)*$signed(b);
        out1_a <= c[60:29];
end
```

Oscillator Module Summary

- Can change generate $y[n-1]$ and $y[n-2]$
- Can set the value of $y[n-1]$ to Initial Condition from Input port whenever the “Ready” signal from Oscillator Module is 0
- Can calculate the value of $y[n], y[n-1], y[n-2]$ whenever the “Enable” signal from Oscillator Module is 1



Oscillator : How to Generate Delay




```
out <= out1_a-out2;                          // combi

out1 <= out;                                 // sequential

out2 <= out1;                                // sequential

c      <= $signed(a)*$signed(out1);          // combi

out1_a <= c[60:29];                          // combi
```

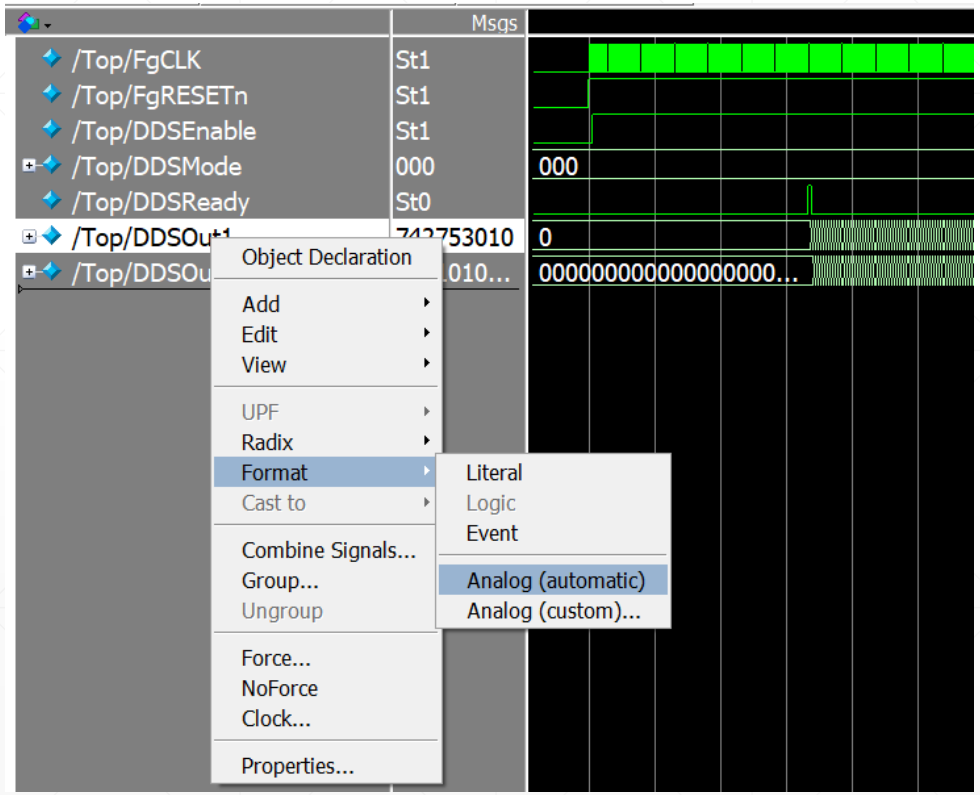
$$\text{init1} = 2^{29}\sin\left(\frac{2\pi f_s}{f_c}\right) = 2^{29}\sin\left(\frac{2\pi 0.73M}{24M}\right) * 0.95 = 96{,}878{,}045$$

$$\text{init2} = 2^{29}2\cos\left(\frac{2\pi f_s}{f_c}\right) = 2^{30}\cos\left(\frac{2\pi 0.73M}{24M}\right) = 1{,}054{,}193{,}702$$

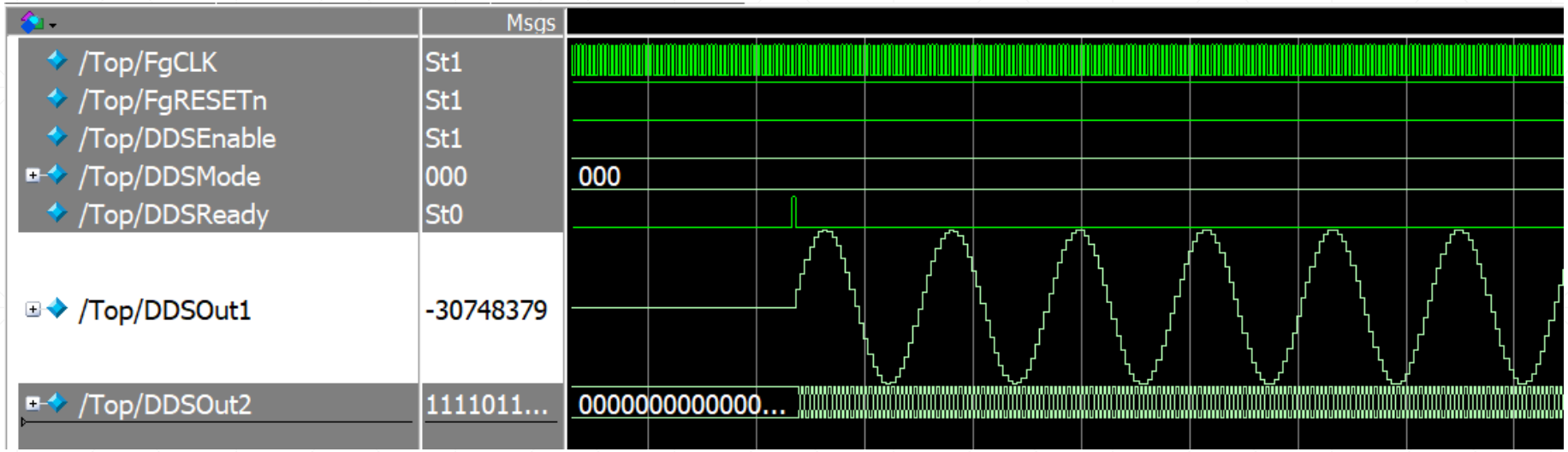
Oscillator Module Expected Behavior

- Please note that the signal into this port $\alpha, y[1]$ will be set manually when we declare this module.

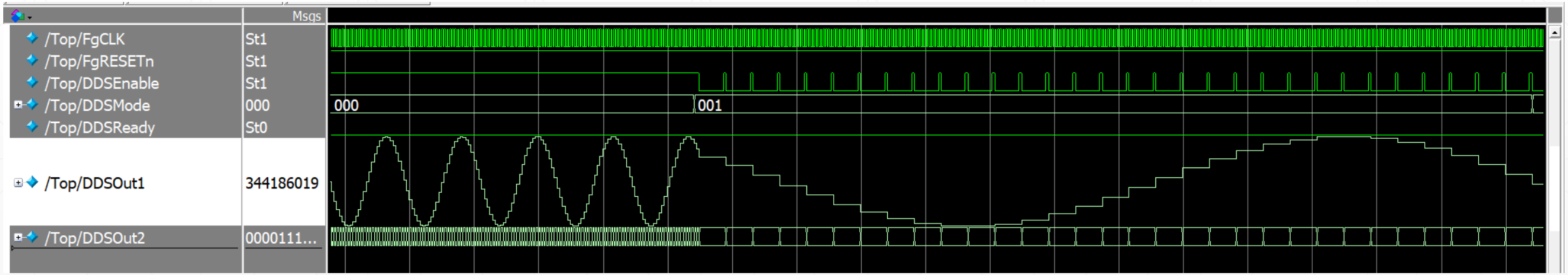
```
Osc_module osc_mod (
    .CLK        (FgCLK),
    .RESETn     (FgRESETn),

    .Enable     (DDSEnable),
    .Ready      (DDSReady),
    .init1      (32'd96878045),
    .init2      (32'd1054193702),

    .out1       (out1),
    .out2       (out2)
);
```



Oscillator Module Expected Behavior



If there is no error, the output of DDSOut1 will be sinusoidal.



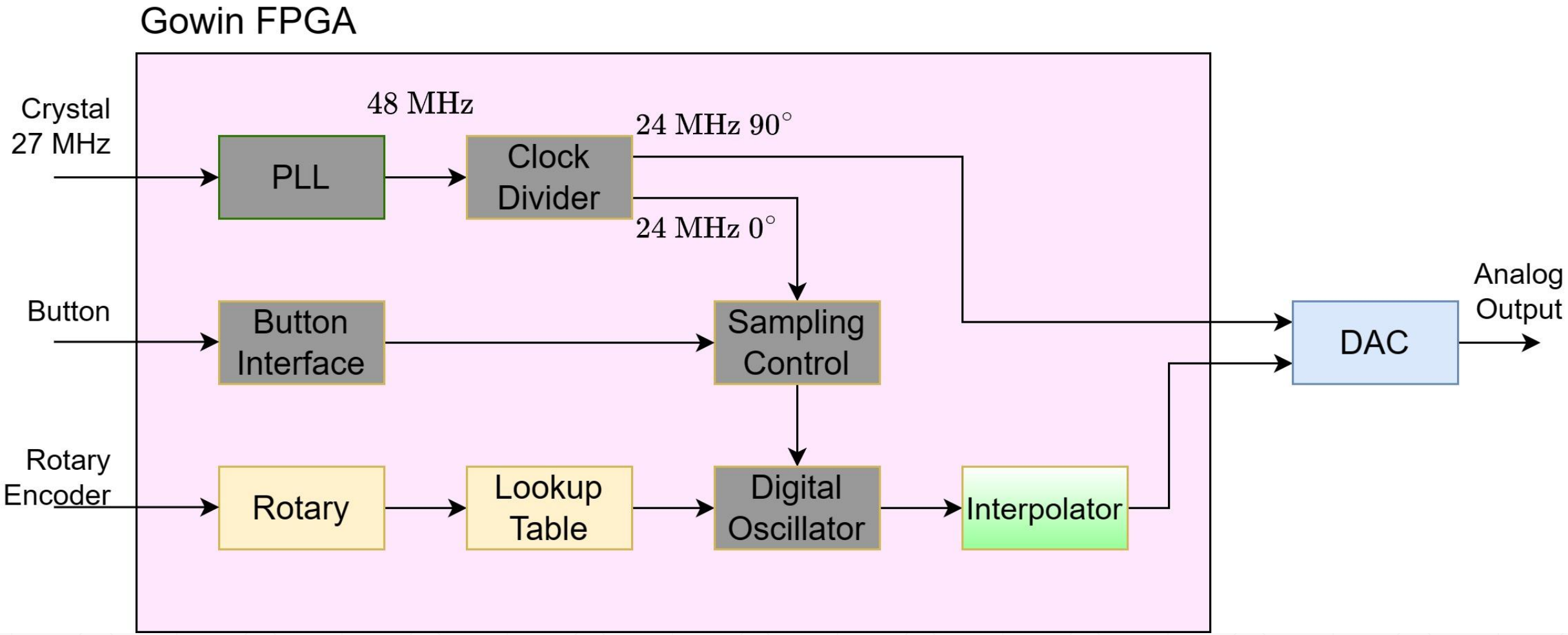
If the mode change from 0 to 1, the sinusoidal will be dilate by the factor of 10

KMITL
พระจอมเกล้าลาดกระบัง

Interpolator Module

Smooth out the discontinuity!

DDS Progress Update



Why we need interpolation? (1)

KMITL
พระจอมเกล้าลาดกระบัง

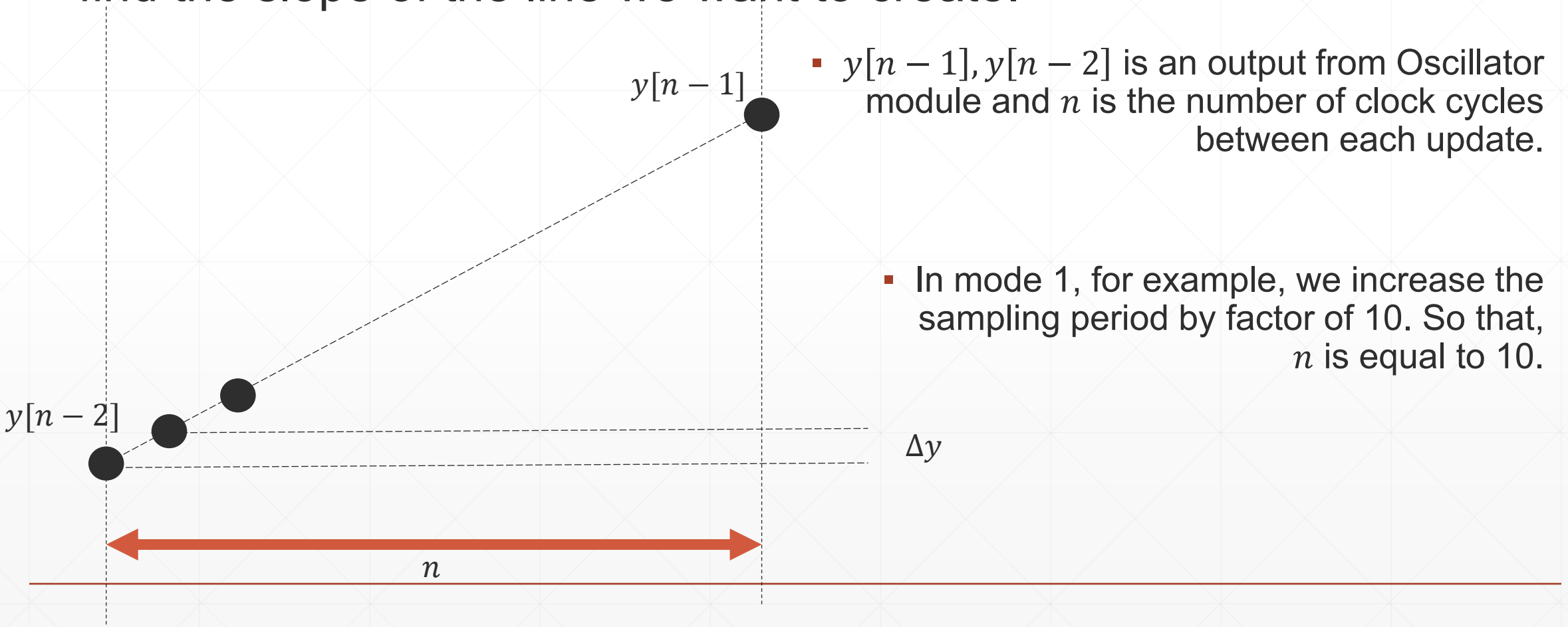
- In this project, we decrease the sampling period in order to generate lower sinusoidal frequency. So, that the output of low frequency signal will look like step function.

- The term interpolation is used to describe something that fill the gap between two data.

- So, we need to fill the gap between each sampling period by the straight line so that our output sinusoidal wave not look like step function too much.

Interpolation Calculation (1)

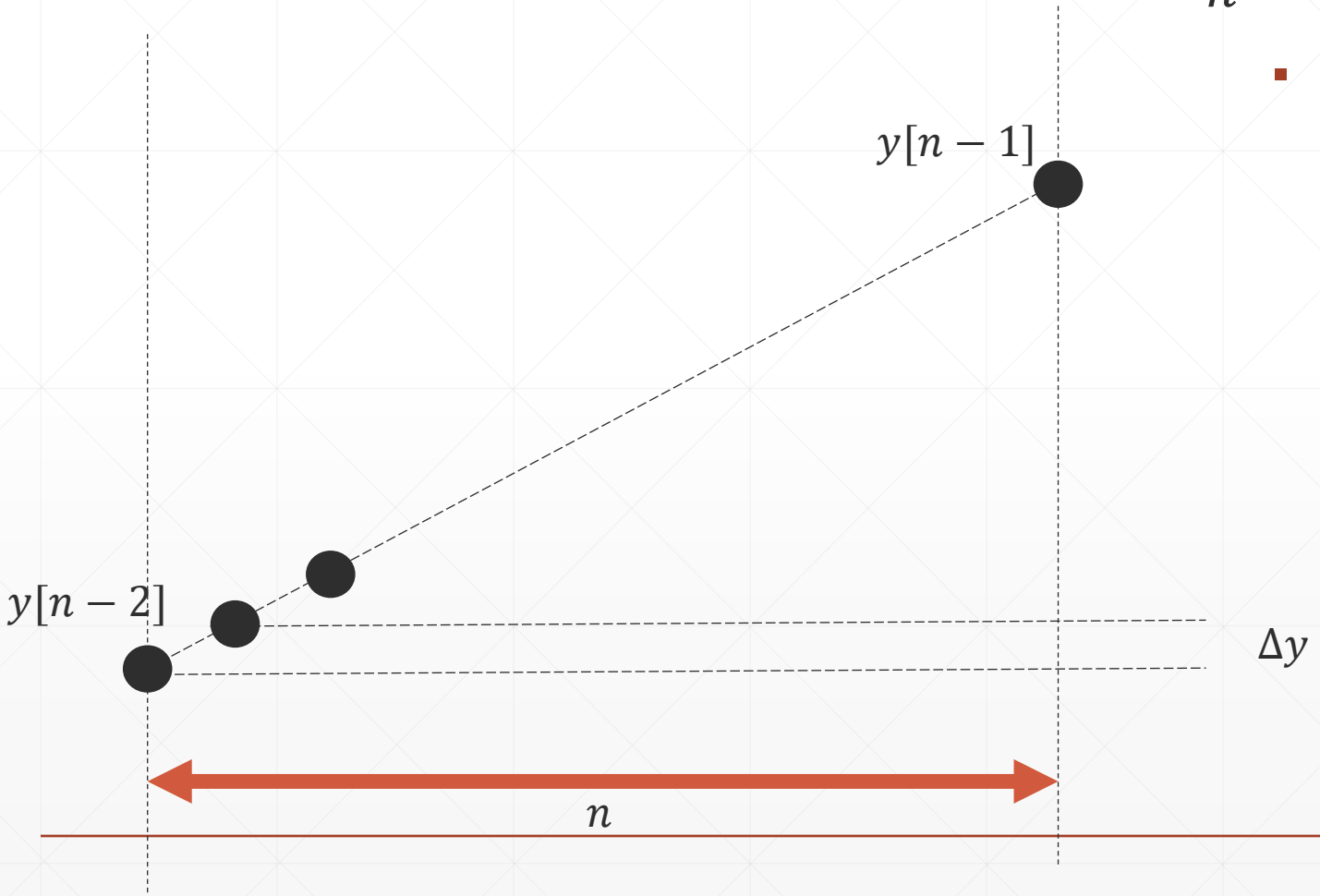
- In order to calculate the line for fill the gap. First, we should create find the slope of the line we want to create.



- $y[n-1], y[n-2]$ is an output from Oscillator module and n is the number of clock cycles between each update.
- In mode 1, for example, we increase the sampling period by factor of 10. So that, n is equal to 10.

Interpolation Calculation (2)

- So that we can find that $\Delta y = \frac{y[n-1]-y[n-2]}{n}$

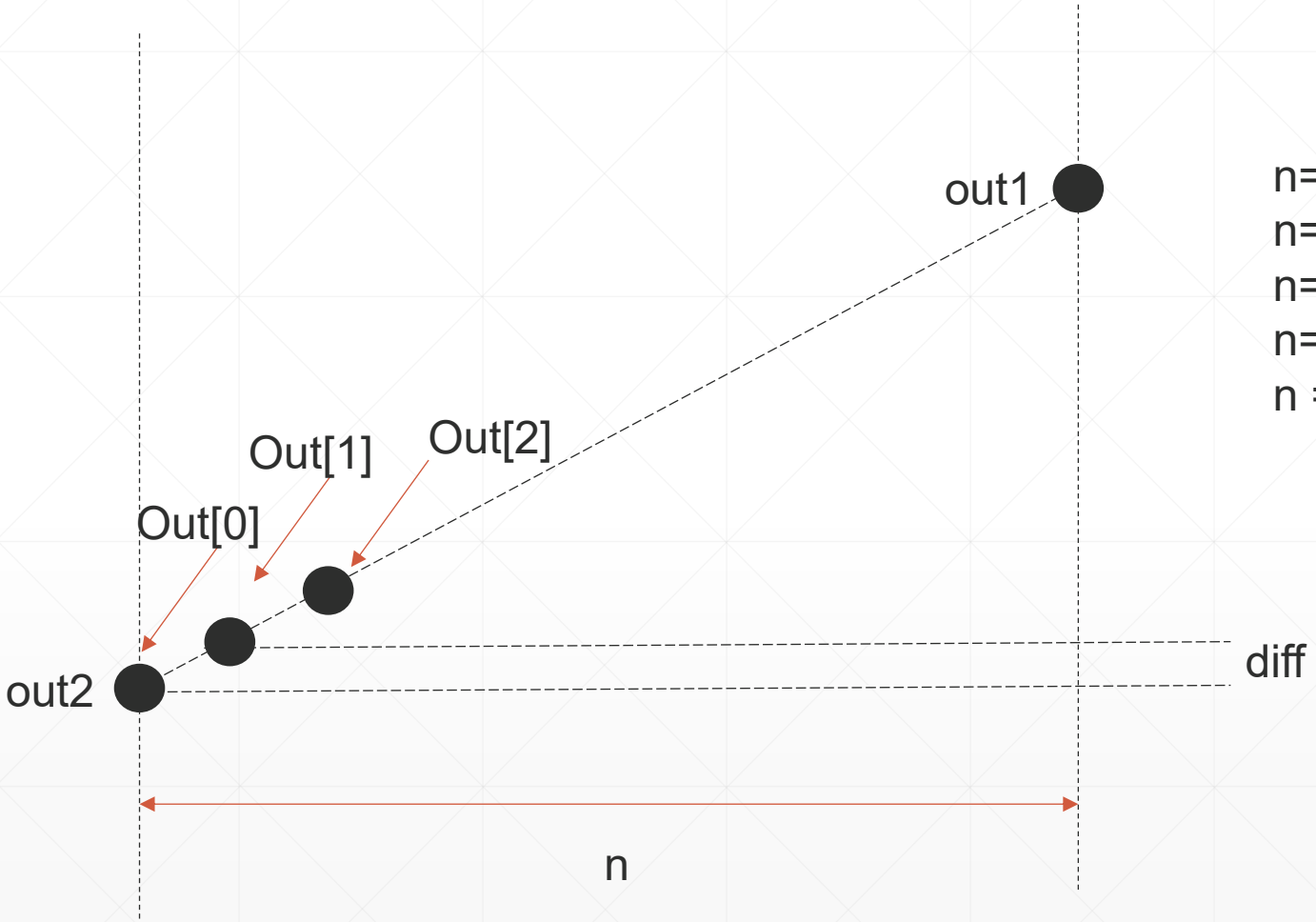


- For each clock cycles, the output from oscillator will be increase by Δy.

Interpolation

$$diff = \frac{out1 - out2}{n}$$

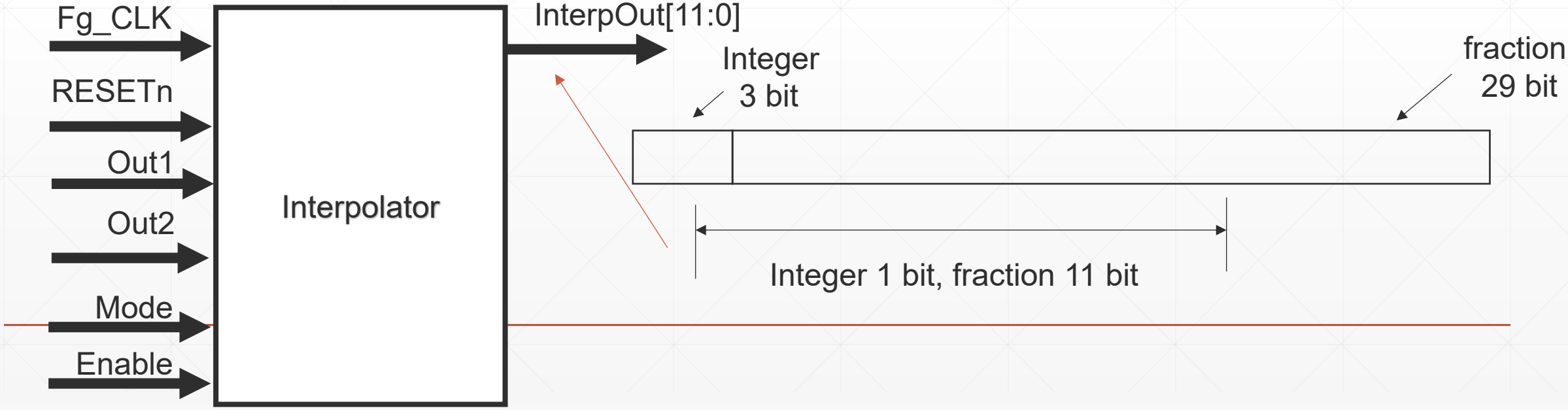
$$out[i] = out[i-1] + diff$$



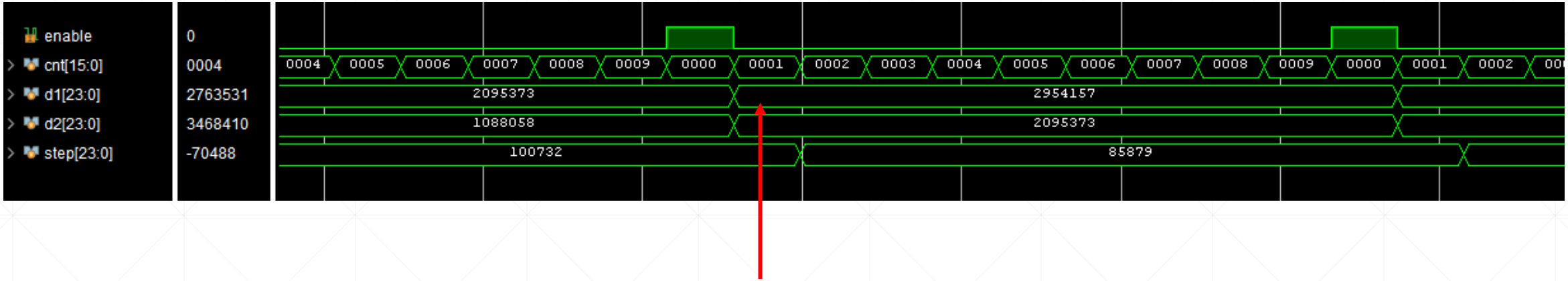
n= 1 ; 1/n = 1
n=10 ; 1/n = 0.1*2^29 = 53687091
n=100 ; 1/n = 0.01*2^29 = 5368709
n=1000 ; = 536871
n =10000 ; = 53687

Interpolator Module Summary

- Can generate output with interpolated value.
- Can passthrough the input value whenever the interpolation is not required (Mode 0)
- Can update the value whenever the output from “Digital Oscillator” is change (Enable = 1)

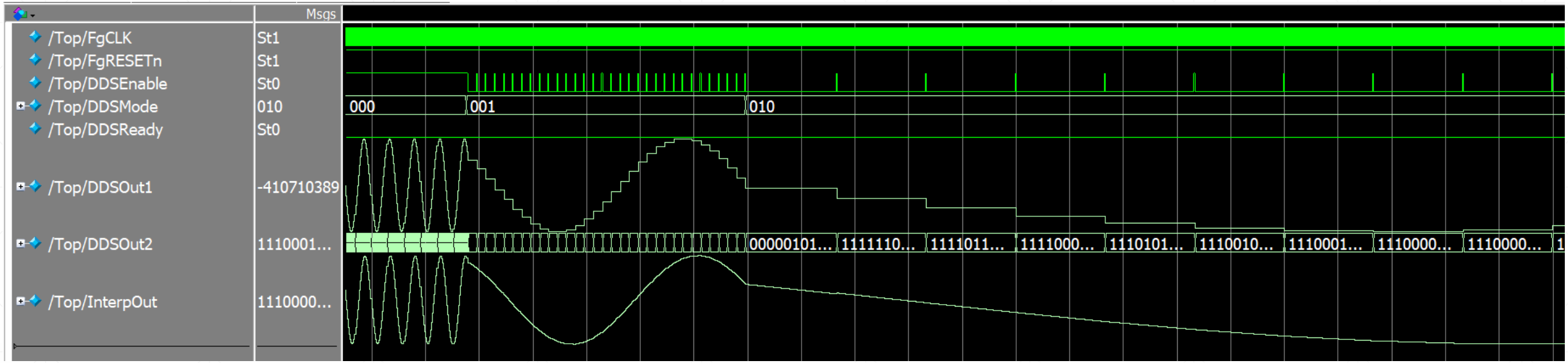


Interpolation



Cal diff when count == 1

Interpolator Module Expected Behavior



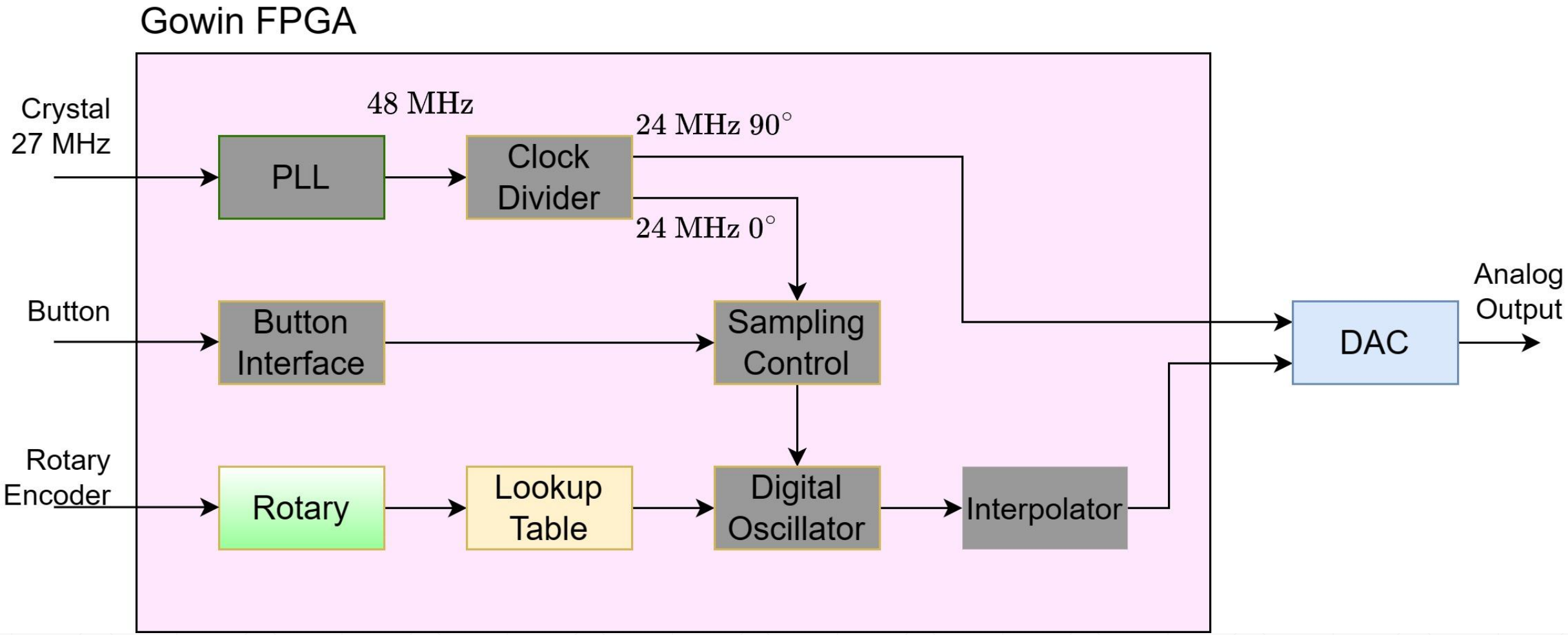
If there are no any error, the result from Interpolator will look like the same as Output from Digital Oscillator except for the smoothness

KMITL
พระจอมเกล้าลาดกระบัง

Rotary Encoder

Knob for change frequency.

Progress Update



Why we need rotary encoder?

KMITL
พระจอมเกล้าลาดกระบัง

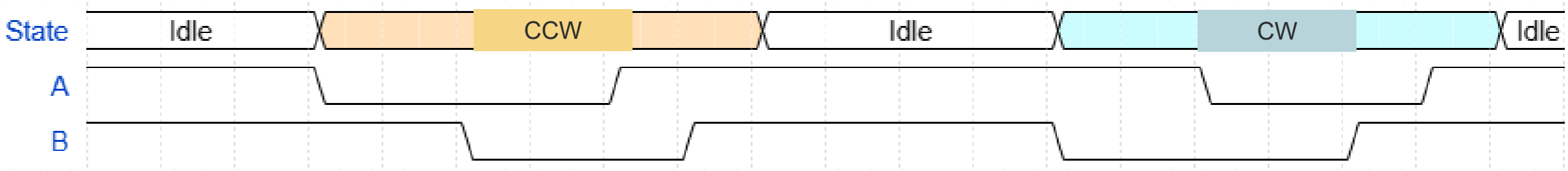
- In order to change the frequency, we choose rotary encoder as a frequency selector.

- The operation of frequency selection can be described by the “counter” which is used for select the α, β, $y[0]$ for that frequency and use “Rotary Encoder” to change the “counter” value up and down.

How Rotary Encoder work? (1)

KMITL
พระจอมเกล้าลาดกระบัง

- The rotary consist of two inputs (Rot_A, Rot_B) and the waveform will look like this.



- When the knob was going to rotate in Clockwise Direction, output A will be “LOW” first. After the knob was rotated successfully, the B will be “LOW” and vice versa.

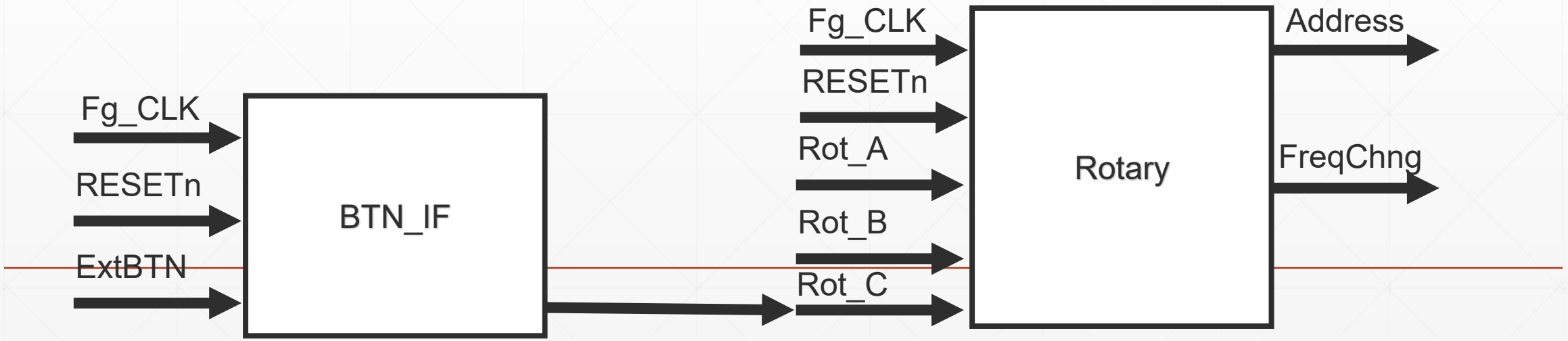
How Rotary Encoder work? (2)

KMITL
พระจอมเกล้าลาดกระบัง

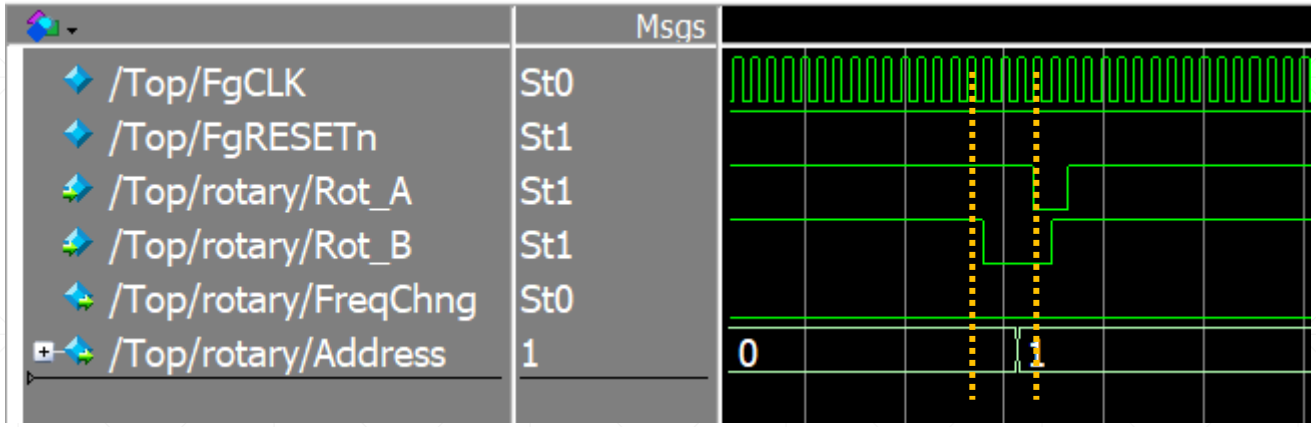
- After detect the rotation direction of rotary encoder, update the “Address” variable which have the range from 0 to 1800 by 1.
- When detect the CW direction, increase the “Address” variable by the factor below and decrease in another direction.
- Furthermore, this module will consist of 3 mode which are
  - Mode 0 : Update the “Address” value by factor of 1
  - Mode 1 : Update the “Address” value by factor of 10
  - Mode 2 : Update the “Address” value by factor of 100
- The mode will change by press the Rotary Button (Rot_C)

Rotary Encoder Module Summary

- Can change the value of “Address” by factor respect to the current rotary mode.
- Increase or decrease “Address” value whenever detection the CW and CCW direction respectively.
- Can change the mode using the button.
- Can generate “FreqChng” signal whenever there is no rotation for 100 ms.



Rotary Encoder Expected Behavior



Rot_B down first means Rotary rotated in CW Direction. So, Address will increase by 1 (Mode 0)

```
`define SIM

`ifdef SIM
    parameter RD_FREQ = 2400;
`else
    parameter RD_FREQ = 2400000;  // 100 ms
```

KMITL
พระจอมเกล้าลาดกระบัง

Lookup Table

A table for Alpha and Beta constant

LUT Generator

KMITL
พระจอมเกล้าลาดกระบัง

$$\sin\left(\frac{2\pi f_{osc}}{f_s}\right)2^{29}$$

f_{osc} = 100 ; 14,053,228; 0x00D66F6C

000.0 | 0000 1101 0110 0110 1111 011 0 | 1100

f_{osc} = 1000 ; 138,948,413; 0x08482F3D

000.0 | 1000 0100 1000 0010 1111 001 1 | 1101

$$\cos\left(\frac{2\pi f_{osc}}{f_s}\right)2^{29}$$

f_{osc} = 100 ; 536,686,951; 0x1FFD3167

000.1 111 | 1 1111 1101 0011 0001 0110 011 | 1

f_{osc} = 1000 ; 518,578,552; 0x1EE8E178

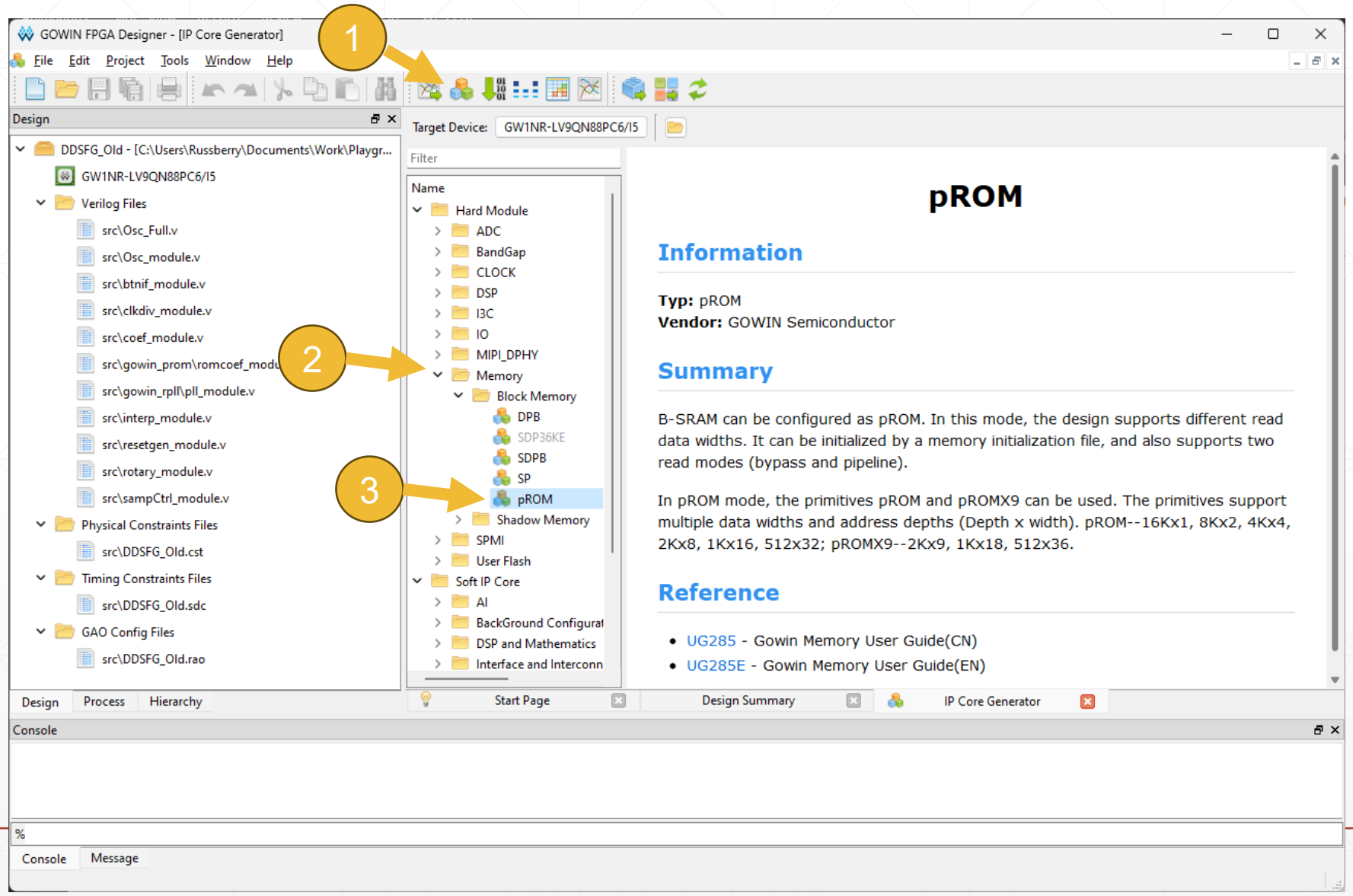
000.1 111 | 0 1110 1000 1110 0001 0111 100 | 0

LUT Generator

- Run the code on the left to generate the Coefficient file.
- The file we have generated will be use as “Table” for select the constant according to “Address” from Rotary Encoder Module.

```
import math

f = open("coefficient.mi", "w")

f.write("#File_format=Hex\n")
f.write("#Address_depth=1024\n")
f.write("#Data_width=48\n") 2048

n = 100

                    2048
for i in range(0, 1024):
    alpha = math.sin(2*math.pi*n/24000) * 0.95
    alpha = round(alpha * 1073741824, 0)
    alpha = int(alpha) >> 5  4   536870912

    y1 = math.cos(2*math.pi*n/24000)
    y1 = round(y1 * 1073741824, 0) 536870912
    y1 = (int(y0) & 0x03FFFFFC) >> 2  1
                    0x01FFFFFE
    if (n <= 1000):
        f.write("%06x%06x \n" % (alpha, y1))
        n = n + 1  0.5
    else :
        f.write("%06x%06x \n" % (0, 0))
        n = n + 1  0.5
```

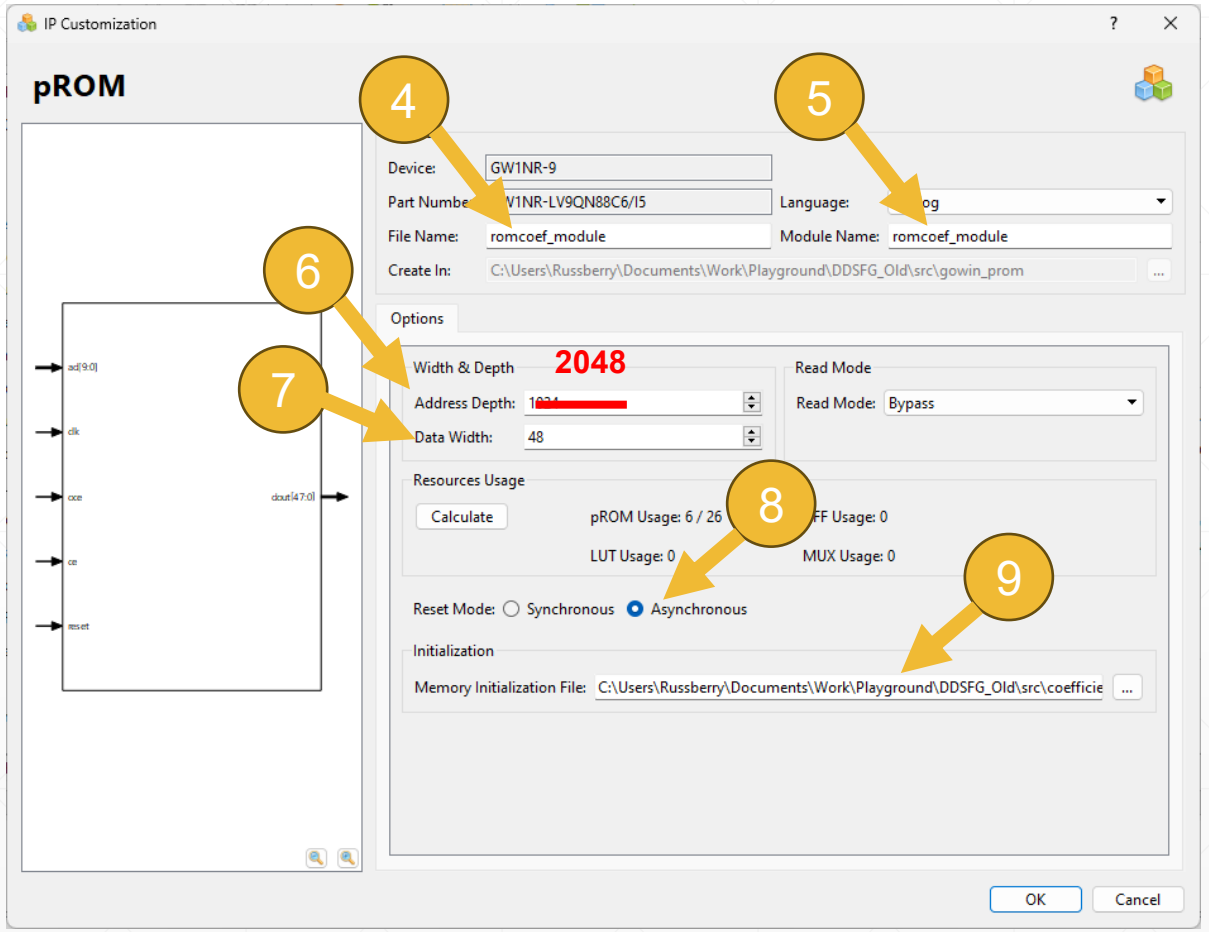
ROM IP-Core Generation (1)



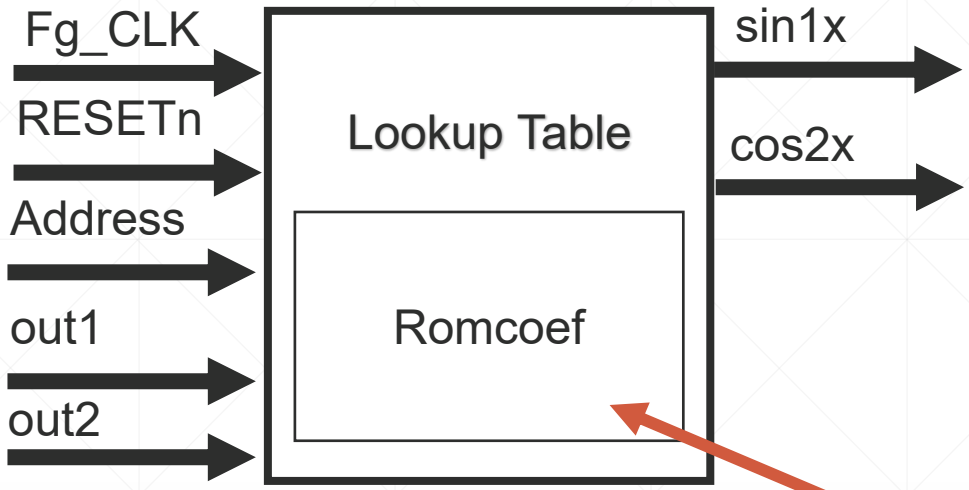
ROM IP-Core Generation (2)

KMITL
พระจอมเกล้าลาดกระบัง

4. File Name
5. Verilog Module Name
6. Set Address Depth to **2048**
7. Set Data Width to 48
8. Set Reset Mode to Asynchronous
9. Select the “Coefficient” file that we generated before.



Lookup Table and ROM Module




```
    romcoef_module romcoefmod(
        .clk    (clk),
        .reset  (~resetn),
        .ce     (1'd1),
        .ad     (Rotary Address),
        .dout   (Coefficient),
);
```

Generate sin1x, cos2x

$$\sin\left(\frac{2\pi f_{osc}}{f_s}\right)2^{29}$$

f_{osc} = 100 ; 14,053,228; 0x00D66F6C
000.0 0000 1101 0110 0110 1111 011 0 1100

f_{osc} = 1000 ; 138,948,413; 0x08482F3D
000.0 1000 0100 1000 0010 1111 001 1 1101

$$\cos\left(\frac{2\pi f_{osc}}{f_s}\right)2^{29}$$

f_{osc} = 100 ; 536,686,951; 0x1FFD3167
000.1 11 11 1111 1101 0011 0001 0110 01 11

f_{osc} = 1000 ; 518,578,552; 0x1EE8E178
000.1 11 10 1110 1000 1110 0001 0111 10 00

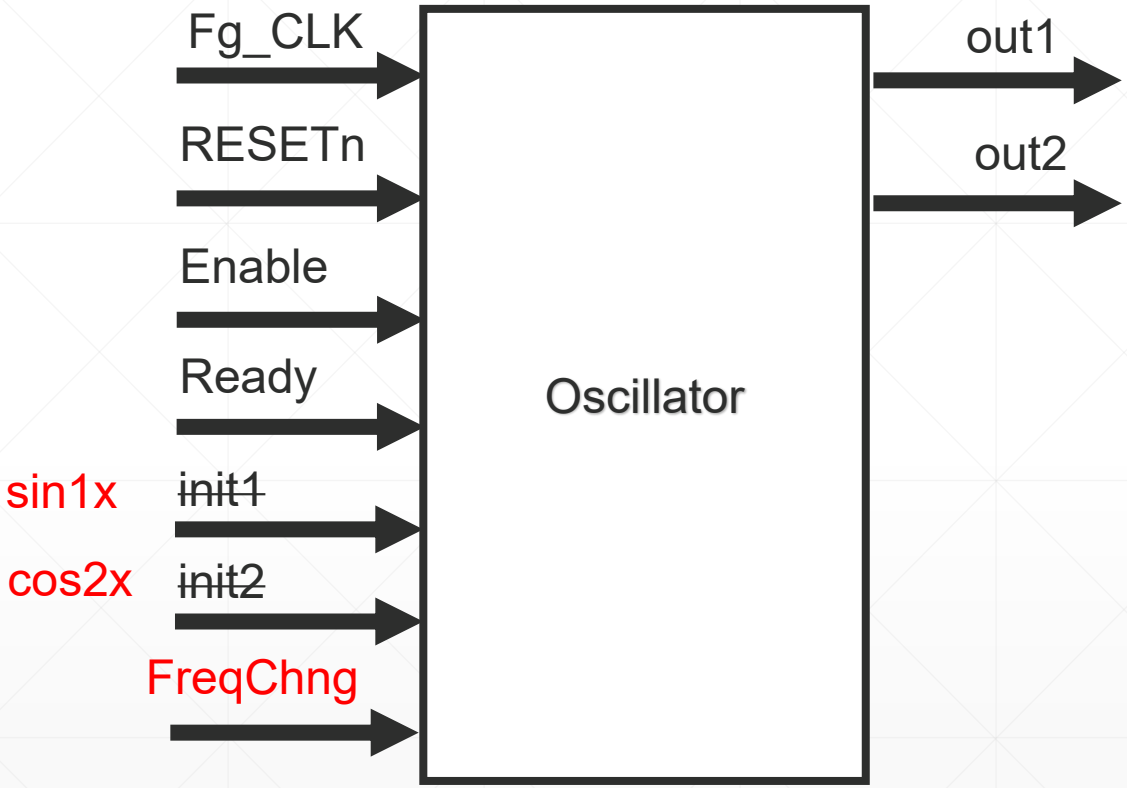
```
wire [31:0]  sin1x;
wire [31:0]  cos2x;

assign sin1x =  {   4'b0000 , coef[47:24] ,  4'b0000 };
assign cos2x = { 6'b001111,  coef[23:0]   ,  2'b00    };
```

KMITL
พระจอมเกล้าลาดกระบัง

Oscillator Module Revised

Updated Version of Oscillator Module

Updated Version for Oscillator Module



- Add “FreqChng” input port for Oscillator Module
- Whenever the “FreqChng” is 1, the Oscillator Module will set the parameter for D.E. by this value

$$y[0] = 0$$
$$y[1] = \text{init1 from LUT}$$
$$\alpha = \text{init2 from LUT}$$

Updated Version for Oscillator Module

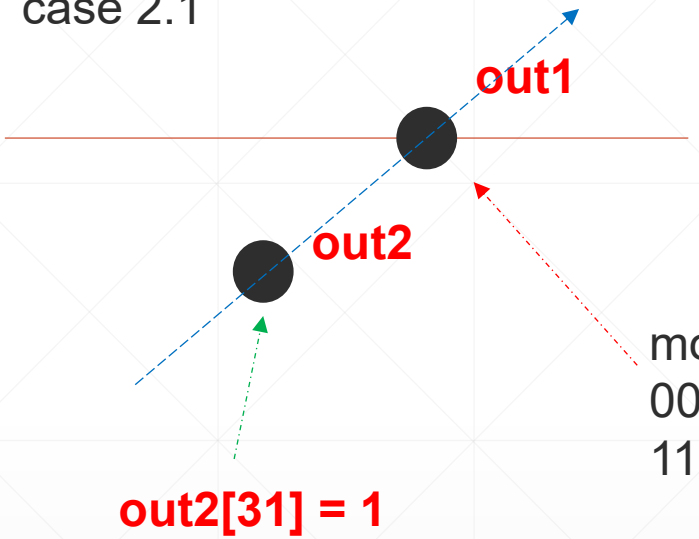
KMITL
พระจอมเกล้าลาดกระบัง

```
Osc_module osc_mod (
    .CLK        (FgCLK),
    .RESETn     (FgRESETn),

    .Enable     (DDSEnable),
    .Ready      (DDSReady),
    .init1      (32'd96878045),
    .init2      (32'd1054193702),
    .freqchgn   (freqchgn),
    .out1       (out1),
    .out2       (out2)
);
```

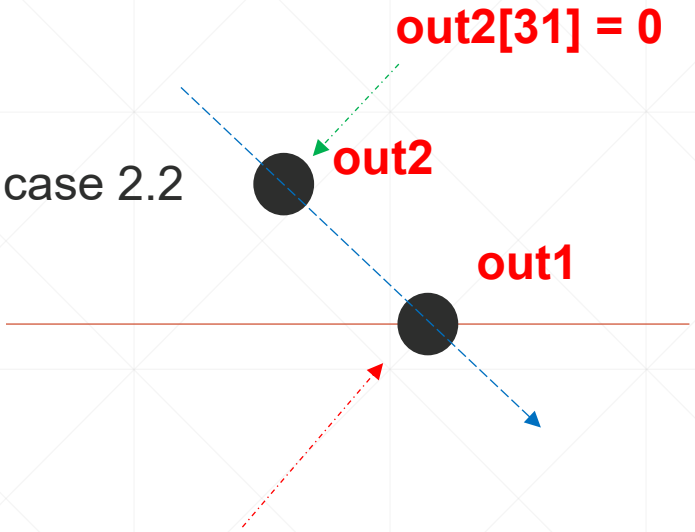
sin1x
cos2x

zero crossing check

case 2.1



case 2.2



mode 0-3
00000000 00xxxxxx xxxxxxxx xxxxxxxx
111111111 11xxxxxx xxxxxxxx xxxxxxxx

mode 4
00000000 0xxxxxxx xxxxxxxx xxxxxxxx
111111111 1xxxxxxx xxxxxxxx xxxxxxxx

zero crossing check

KMITL
พระจอมเกล้าลาดกระบัง

Condition for update wait (Sequential)

```
if          (freqchng==1)          update_wait =1
else if     (do_update==1)         update_wait = 0
```

Condition for zero cross (combination)

```
if (mode!=4) and (out1[31:22]==10'b0000000000) | (out1[31:22]==10'b1111111111)   OR
                    (out1[31:23]==    9'b000000000) | (out1[31:23]==  9'b111111111)
          zcross = 1 else zcross = 0
```

Condition for direction (combination)

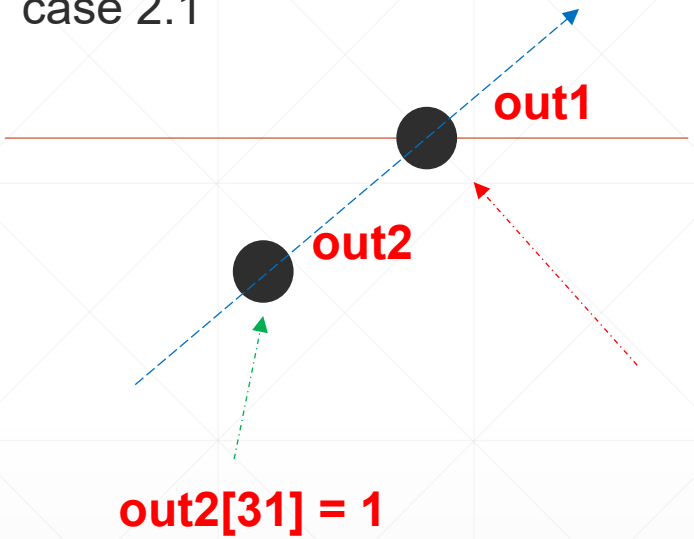
```
if (out2[31]==1) dir = 1 else dir = 0
```

Condition for doing update (Combination)

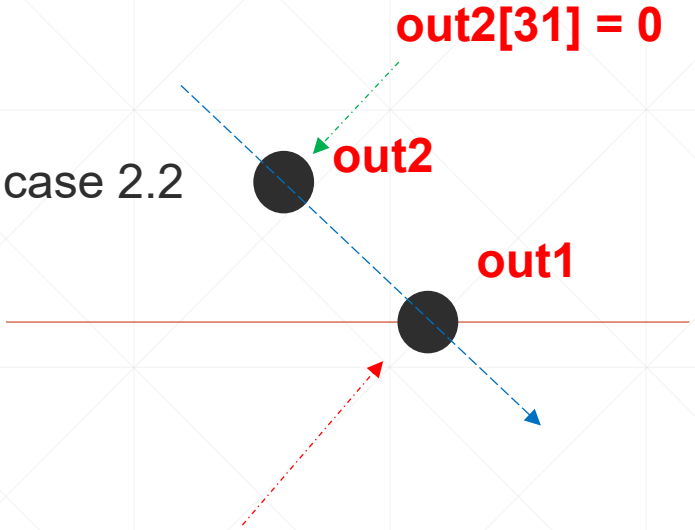
```
If (zcross & update_wait) do_update = 1 else do_update = 0
```

Update Case 2.1 & 2.2

case 2.1



case 2.2




```
input [31:0] out1;
input [31:0] out2;
sine   = (out2[31]==1'b1) ? sin1x : ~sin1x+1;
```

cos2x is the same as before.

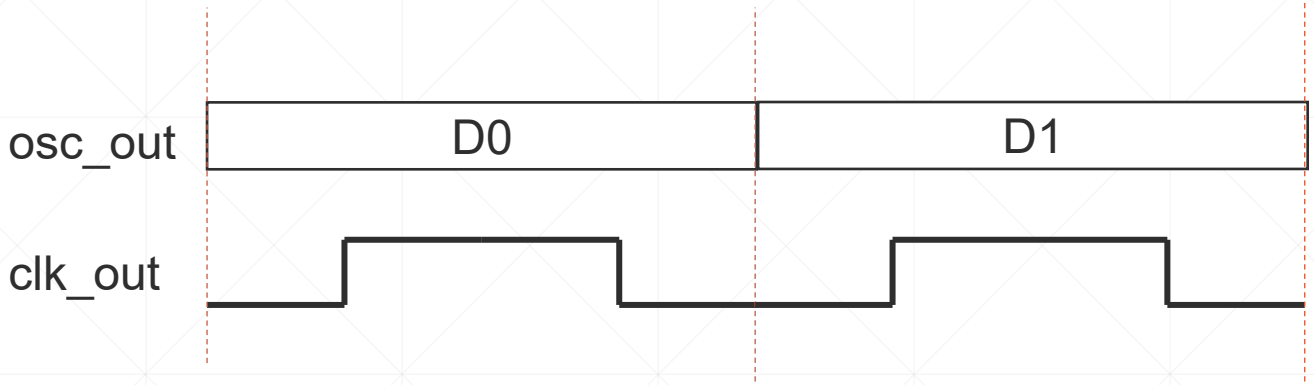
Frequency Update

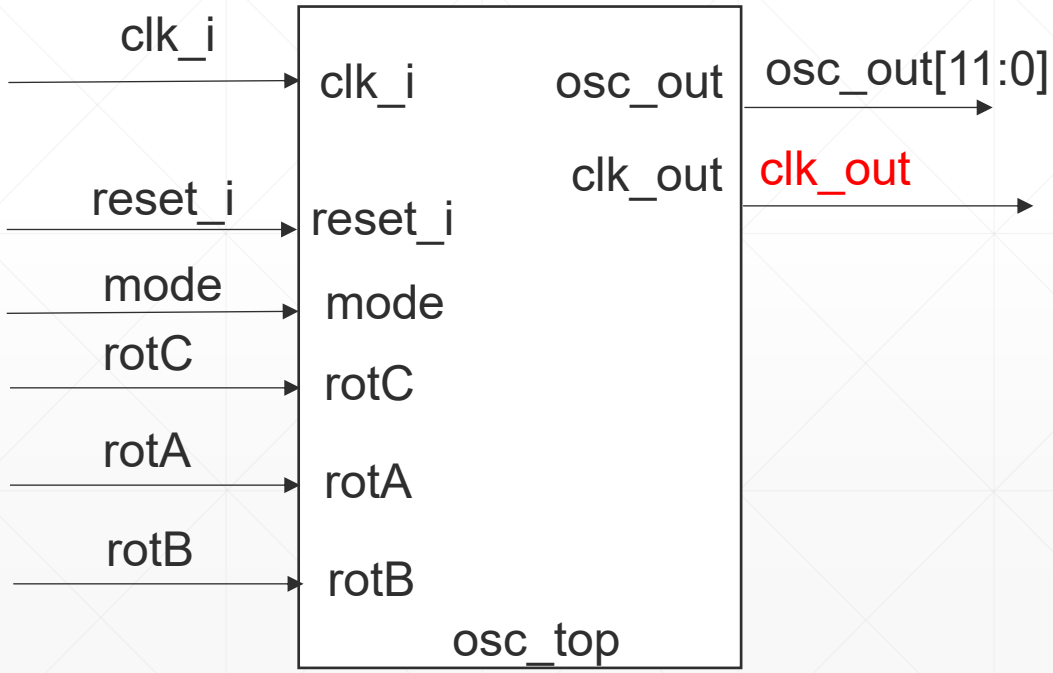
Offset Binary <-> 2's complement

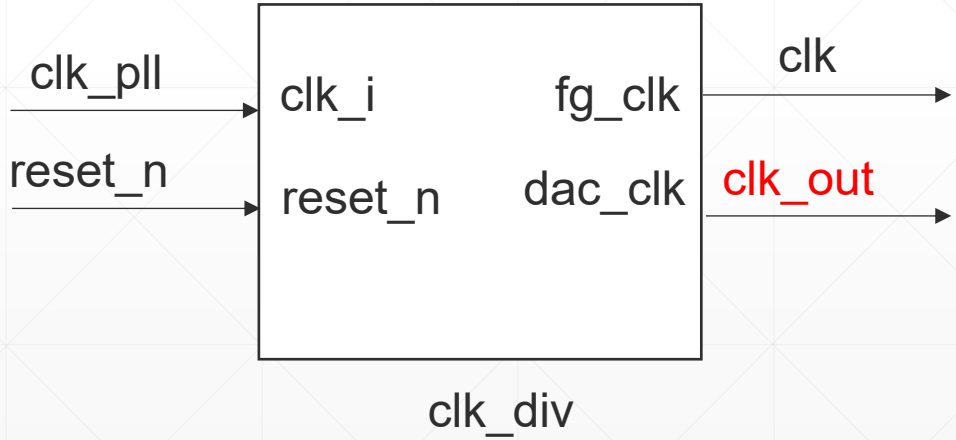
KMITL
พระจอมเกล้าลาดกระบัง

$A_{IN}^{+} - A_{IN}^{-}$ (2V Range)	D15-D0 (OFFSET BINARY)	D15-D0 (2's COMPLEMENT)
>1.000000V	1111 1111 1111 11XX	0111 1111 1111 11XX
+0.999970V	1111 1111 1111 1111	0111 1111 1111 1111
+0.999939V	1111 1111 1111 1110	0111 1111 1111 1110
+0.999909V	1111 1111 1111 1101	0111 1111 1111 1101
+0.999978V	1111 1111 1111 1100	0111 1111 1111 1100
+0.000030V	1000 0000 0000 0001	0000 0000 0000 0001
+0.000000V	1000 0000 0000 0000	0000 0000 0000 0000
+0.000030V	0111 1111 1111 1111	1111 1111 1111 1111
+0.000061V	0111 1111 1111 1110	1111 1111 1111 1110
–0.999878V	0000 0000 0000 0011	1000 0000 0000 0011
–0.999909V	0000 0000 0000 0010	1000 0000 0000 0010
–0.999939V	0000 0000 0000 0001	1000 0000 0000 0001
–1.000000V	0000 0000 0000 0000	1000 0000 0000 0000
< –1.000000V	0000 0000 0000 00XX	1000 0000 0000 00XX

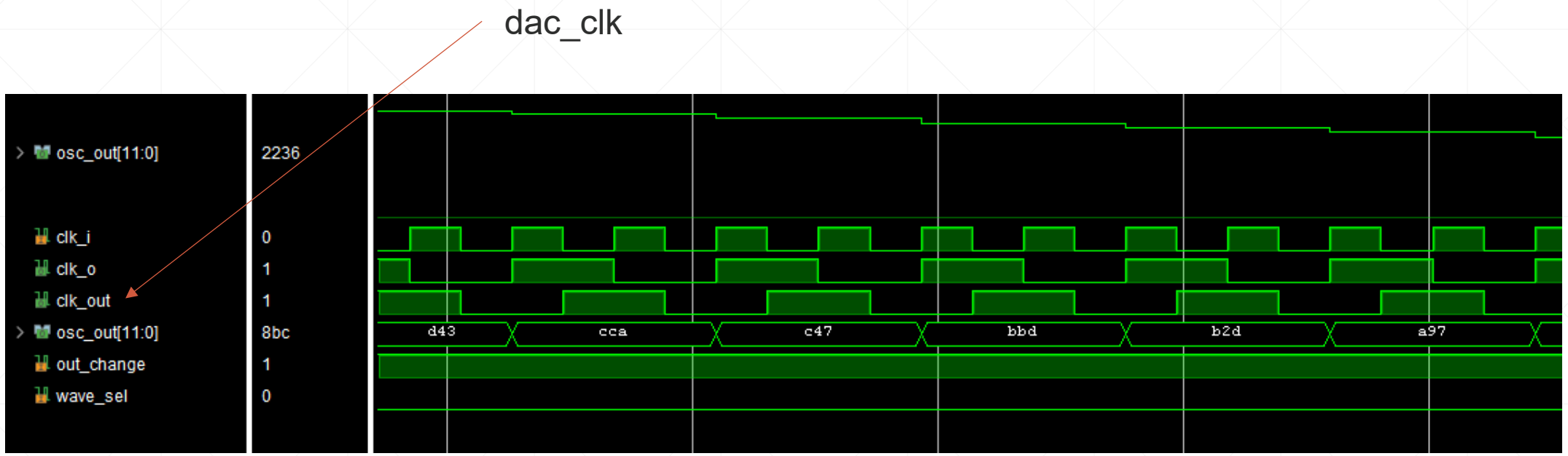
DAC clock output



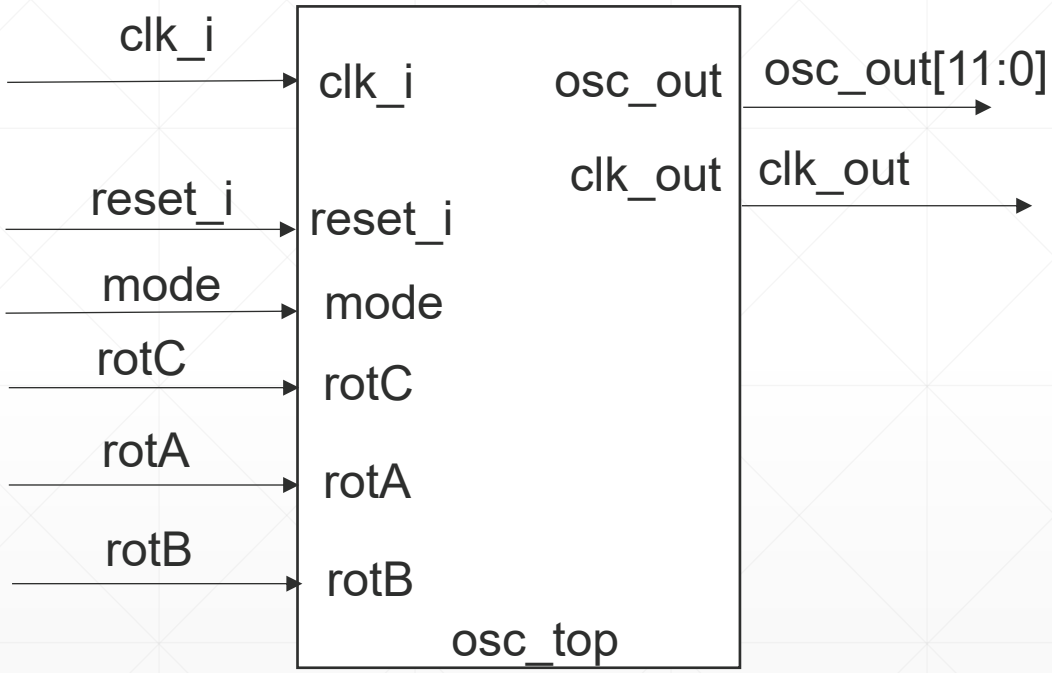




DAC clock output (Sine Wave)



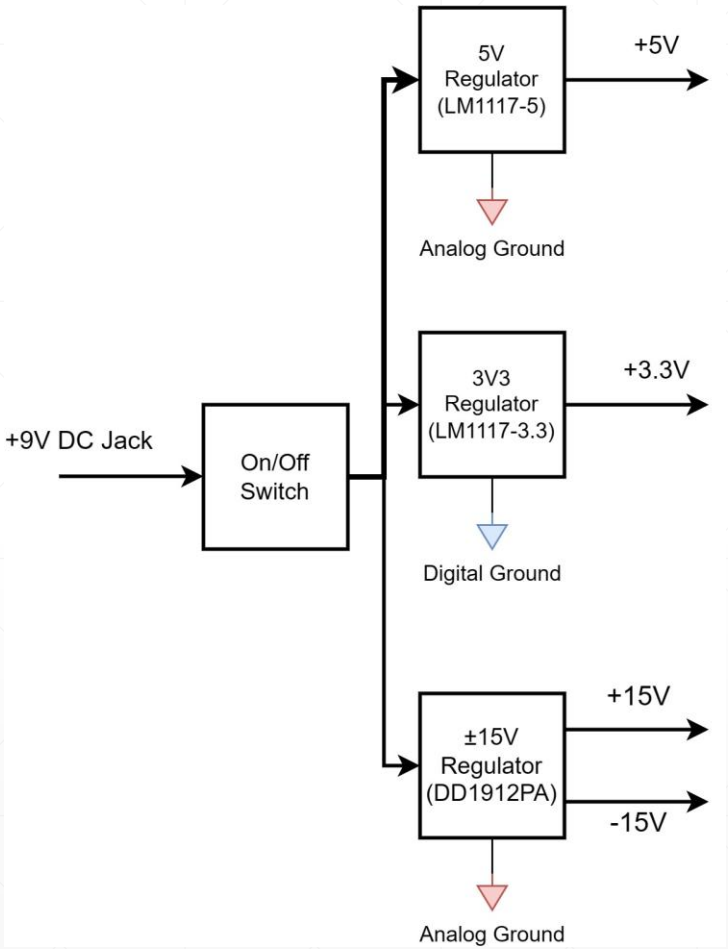
Summary

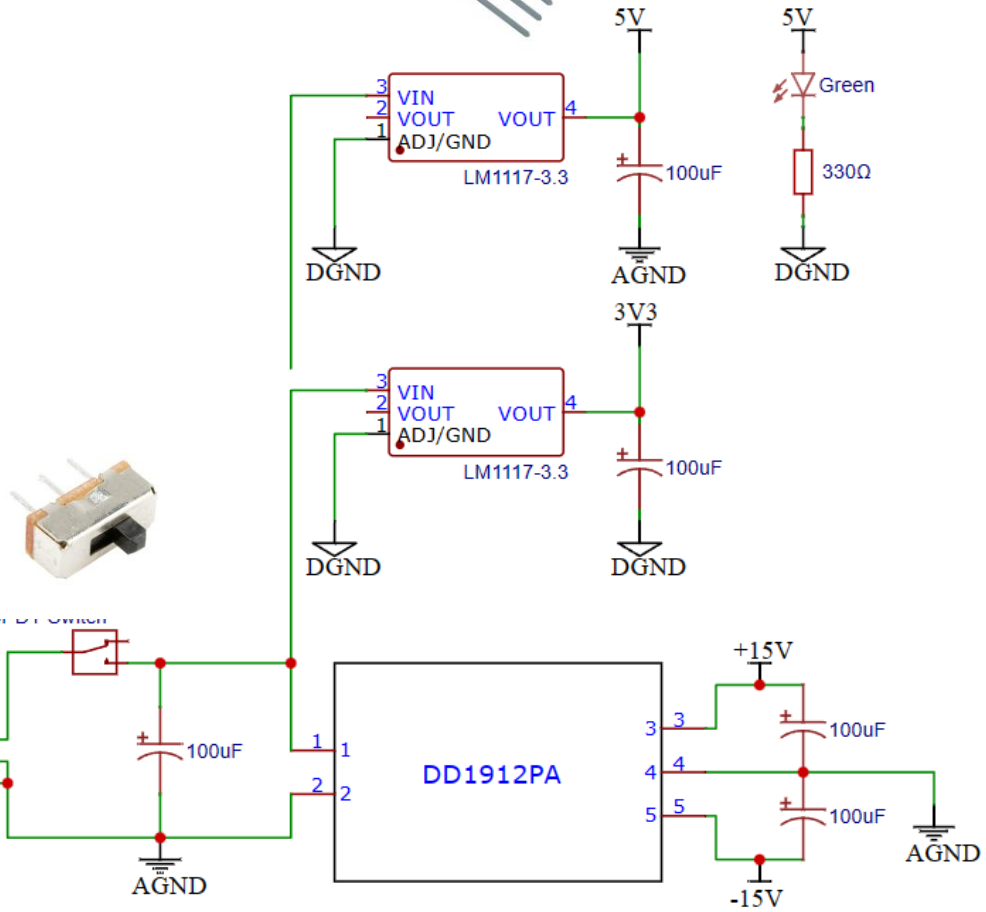

clk_i
clk_i
osc_out
osc_out[11:0]
clk_out
clk_out
reset_i
reset_i
mode
mode
rotC
rotC
rotA
rotA
rotB
rotB
osc_top

KMITL
พระจอมเกล้าลาดกระบัง

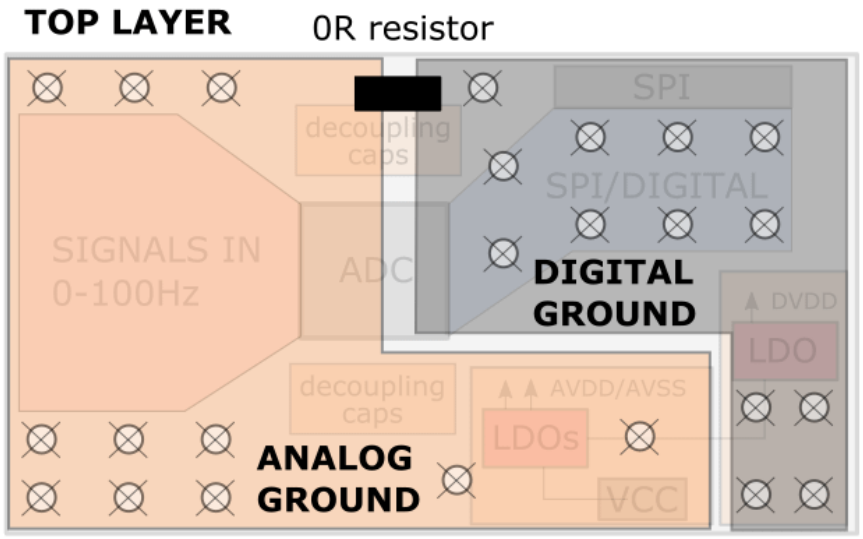
Hardware Schematic

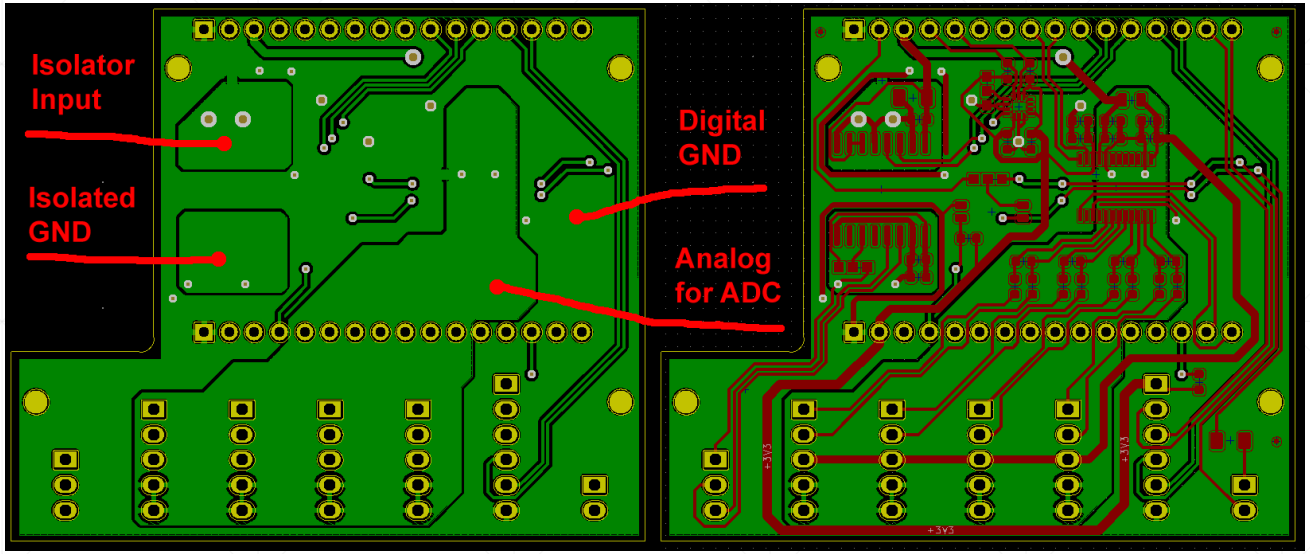
Power Supply



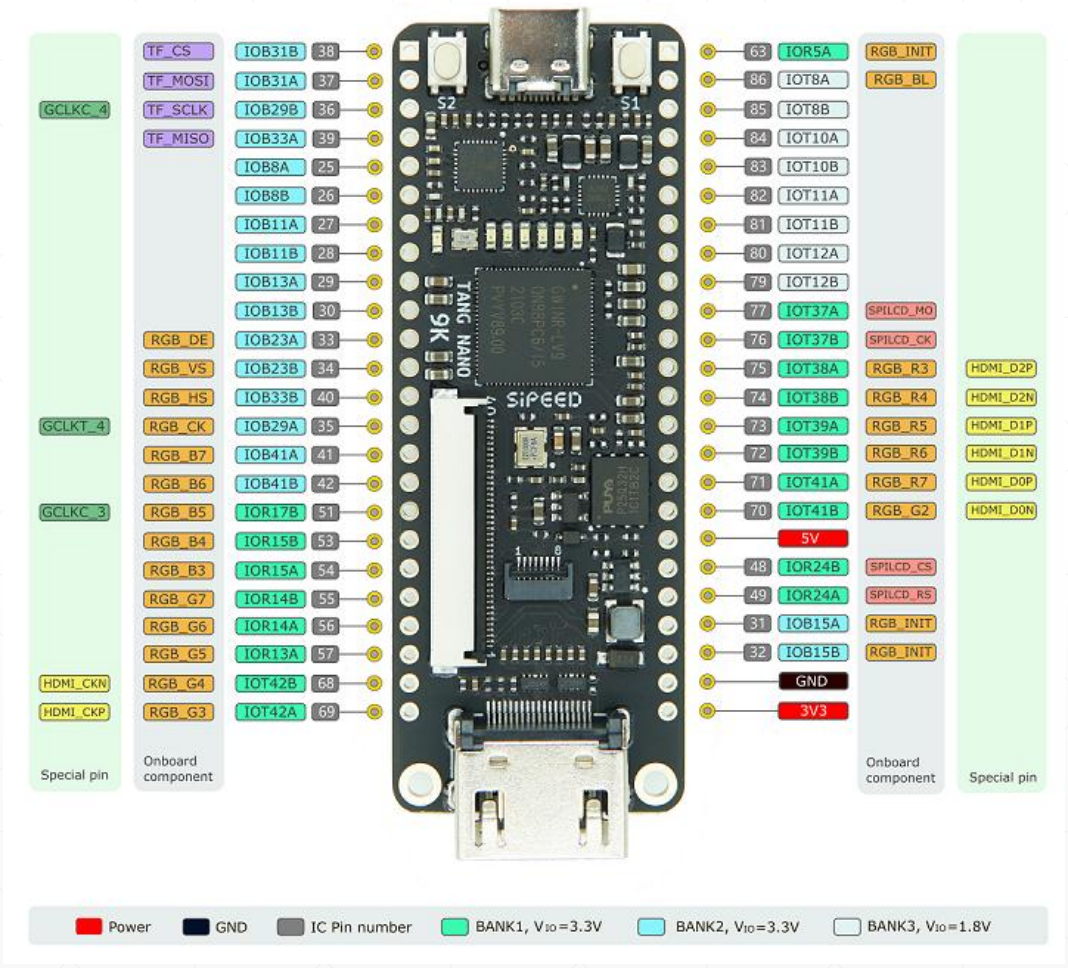


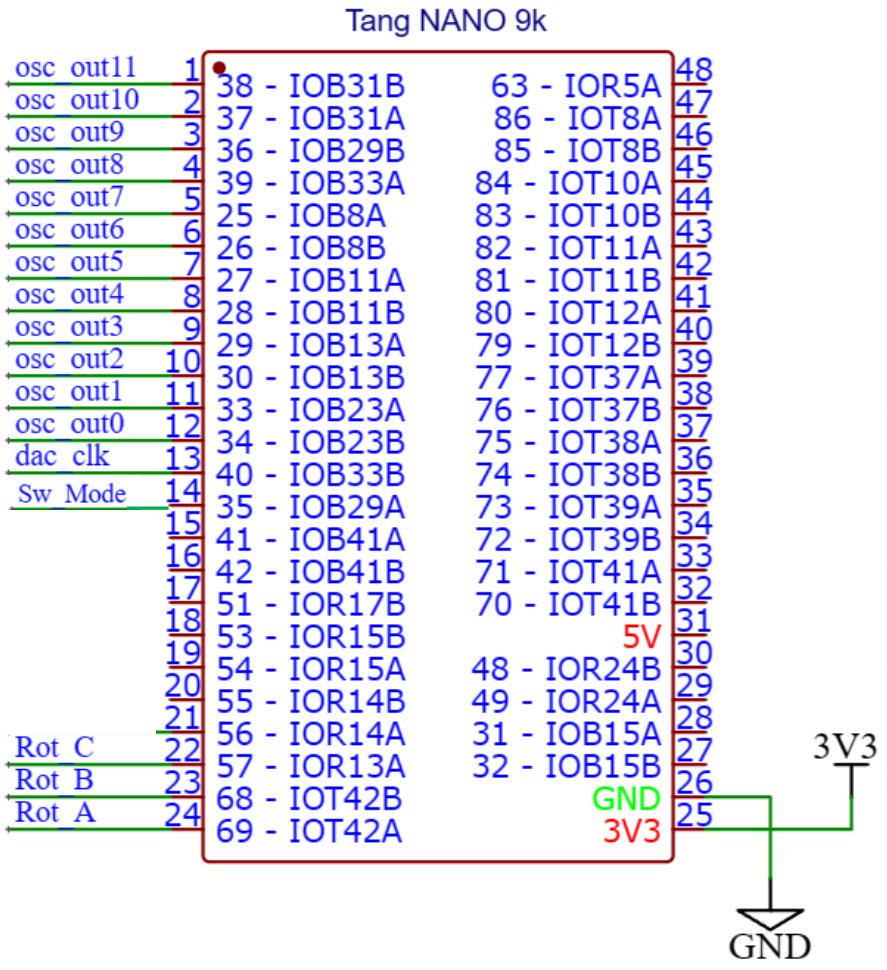
Analog and Digital Ground Plane



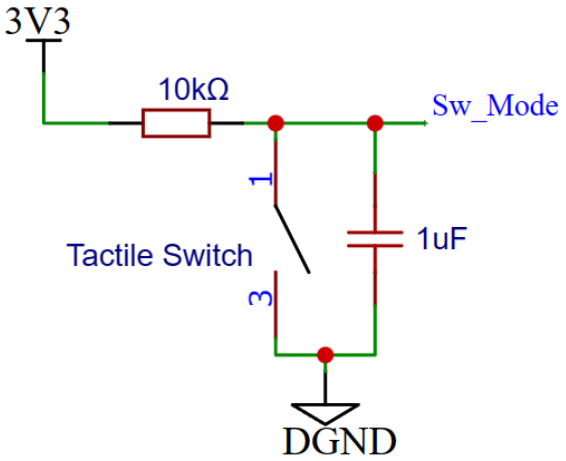


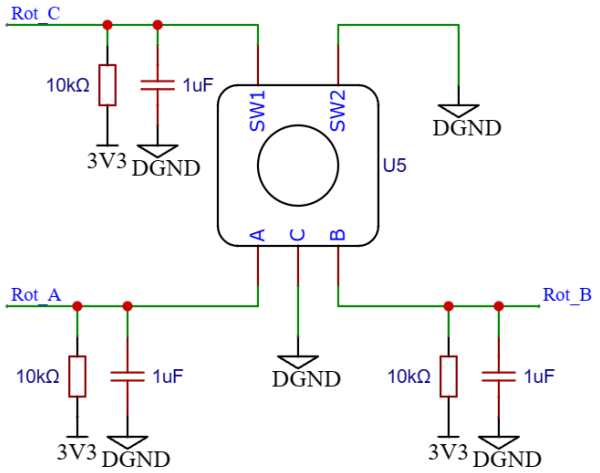
FPGA


TF_CS
TF_MOSI
TF_SCLK
TF_MISO
RGB_DE
RGB_VS
RGB_HS
RGB_CK
RGB_B7
RGB_B6
RGB_B5
RGB_B4
RGB_B3
RGB_G7
RGB_G6
RGB_G5
RGB_G4
RGB_G3
IOB31B 38
IOB31A 37
IOB29B 36
IOB33A 39
IOB8A 25
IOB8B 26
IOB11A 27
IOB11B 28
IOB13A 29
IOB13B 30
IOB23A 33
IOB23B 34
IOB33B 40
IOB29A 35
IOB41A 41
IOB41B 42
IOR17B 51
IOR15B 53
IOR15A 54
IOR14B 55
IOR14A 56
IOR13A 57
IOT42B 68
IOT42A 69
GCLKC_4
GCLKT_4
GCLKC_3
HDMI_CKN
HDMI_CKP
Special pin
Onboard component
S2
S1
TANG NANO 9K
SiPEED
63 IOR5A
86 IOT8A
85 IOT8B
84 IOT10A
83 IOT10B
82 IOT11A
81 IOT11B
80 IOT12A
79 IOT12B
77 IOT37A
76 IOT37B
75 IOT38A
74 IOT38B
73 IOT39A
72 IOT39B
71 IOT41A
70 IOT41B
5V
48 IOR24B
49 IOR24A
31 IOB15A
32 IOB15B
GND
3V3
RGB_INIT
RGB_BL
SPILCD_MO
SPILCD_CK
RGB_R3
RGB_R4
RGB_R5
RGB_R6
RGB_R7
RGB_G2
SPILCD_CS
SPILCD_RS
RGB_INIT
RGB_INIT
HDMI_D2P
HDMI_D2N
HDMI_D1P
HDMI_D1N
HDMI_D0P
HDMI_D0N
Onboard component
Special pin
Power
GND
IC Pin number
BANK1, VIO=3.3V
BANK2, VIO=3.3V
BANK3, VIO=1.8V

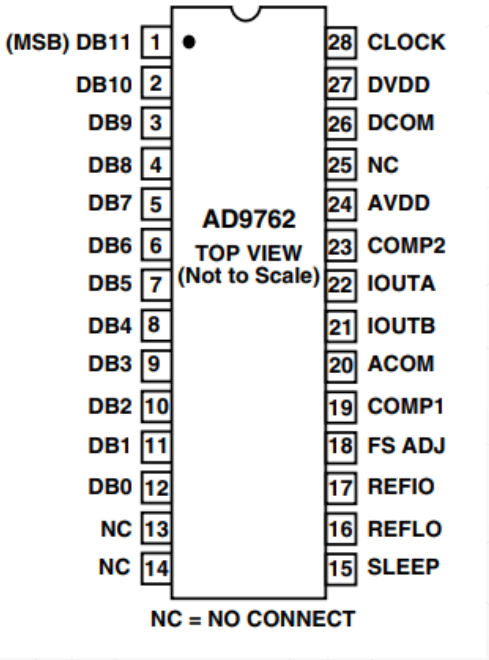

Tang NANO 9k
osc_out11 1
osc_out10 2
osc_out9 3
osc_out8 4
osc_out7 5
osc_out6 6
osc_out5 7
osc_out4 8
osc_out3 9
osc_out2 10
osc_out1 11
osc_out0 12
dac_clk 13
Sw_Mode 14
15
16
17
18
19
20
21
Rot_C 22
Rot_B 23
Rot_A 24
38 - IOB31B
37 - IOB31A
36 - IOB29B
39 - IOB33A
25 - IOB8A
26 - IOB8B
27 - IOB11A
28 - IOB11B
29 - IOB13A
30 - IOB13B
33 - IOB23A
34 - IOB23B
40 - IOB33B
35 - IOB29A
41 - IOB41A
42 - IOB41B
51 - IOR17B
53 - IOR15B
54 - IOR15A
55 - IOR14B
56 - IOR14A
57 - IOR13A
68 - IOT42B
69 - IOT42A
63 - IOR5A 48
86 - IOT8A 47
85 - IOT8B 46
84 - IOT10A 45
83 - IOT10B 44
82 - IOT11A 43
81 - IOT11B 42
80 - IOT12A 41
79 - IOT12B 40
77 - IOT37A 39
76 - IOT37B 38
75 - IOT38A 37
74 - IOT38B 36
73 - IOT39A 35
72 - IOT39B 34
71 - IOT41A 33
70 - IOT41B 32
5V 31
48 - IOR24B 30
49 - IOR24A 29
31 - IOB15A 28
32 - IOB15B 27
GND 26
3V3 25
3V3
GND

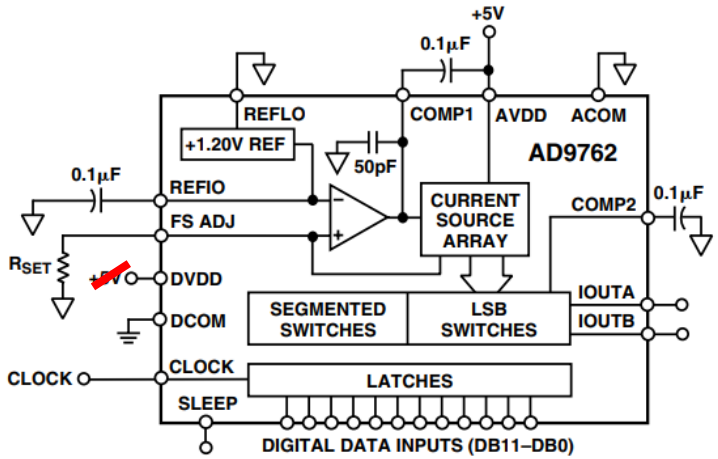
Mode Switch and Rotary

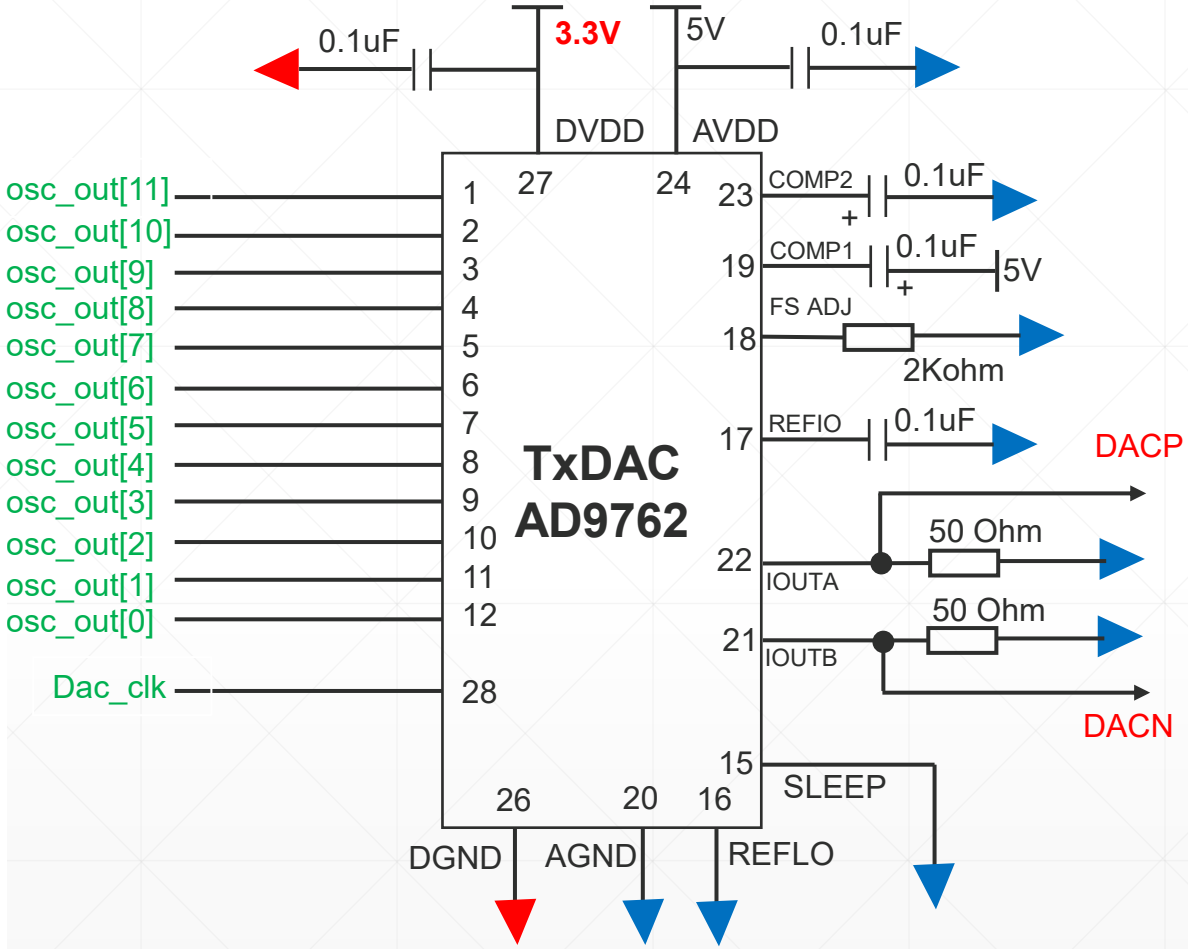




Digital to Analog Converter







Digital to Analog Convertor

KMITL
พระจอมเกล้าลาดกระบัง

FUNCTIONAL DESCRIPTION

Figure 39 shows a simplified block diagram of the AD9762. The AD9762 consists of a large PMOS current source array that is capable of providing up to 20 mA of total current. The array is divided into 31 equal currents that make up the 5 most significant bits (MSBs). The next 4 bits or middle bits consist of 15 equal current sources whose value is 1/16th of an MSB current source. The remaining LSBs are binary weighted fractions of the middle-bits current sources. Implementing the middle and lower bits with current sources, instead of an R-2R ladder, enhances its dynamic performance for multitone or low amplitude signals and helps maintain the DAC's high output impedance (i.e., >100 kΩ).

All of these current sources are switched to one or the other of the two output nodes (i.e., I_{OUTA} or I_{OUTB}) via PMOS differential current switches. The switches are based on a new architecture that drastically improves distortion performance. This new switch architecture reduces various timing errors and provides matching complementary drive signals to the inputs of the differential current switches.

The analog and digital sections of the AD9762 have separate power supply inputs (i.e., AVDD and DVDD) that can operate independently over a 2.7 volt to 5.5 volt range. The digital section, which is capable of operating up to a 125 MSPS clock rate, consists of edge-triggered latches and segment decoding logic circuitry. The analog section includes the PMOS current sources, the associated differential switches, a 1.20 V bandgap voltage reference and a reference control amplifier.

The full-scale output current is regulated by the reference control amplifier and can be set from 2 mA to 20 mA via an external resistor, R_{SET}. The external resistor, in combination with both the reference control amplifier and voltage reference V_{REFIO}, sets the reference current I_{REF}, which is mirrored over to the segmented current sources with the proper scaling factor. The full-scale current, I_{OUTFS}, is thirty-two times the value of I_{REF}.

DAC TRANSFER FUNCTION

The AD9762 provides complementary current outputs, I_{OUTA} and I_{OUTB}. I_{OUTA} will provide a near full-scale current output, I_{OUTFS}, when all bits are high (i.e., DAC CODE = 4095) while I_{OUTB}, the complementary output, provides no current. The current output appearing at I_{OUTA} and I_{OUTB} is a function of both the input code and I_{OUTFS} and can be expressed as:

$$I_{OUTA} = (DAC\ CODE/4096) \times I_{OUTFS} \quad (1)$$

$$I_{OUTB} = (4095 - DAC\ CODE)/4096 \times I_{OUTFS} \quad (2)$$

where *DAC CODE* = 0 to 4095 (i.e., Decimal Representation).

As mentioned previously, I_{OUTFS} is a function of the reference current I_{REF}, which is nominally set by a reference voltage V_{REFIO} and external resistor R_{SET}. It can be expressed as:

$$I_{OUTFS} = 32 \times I_{REF} \quad (3)$$

where $I_{REF} = V_{REFIO}/R_{SET}$ (4)

The two current outputs will typically drive a resistive load directly or via a transformer. If dc coupling is required, I_{OUTA} and I_{OUTB} should be directly connected to matching resistive loads, R_{LOAD}, which are tied to analog common, ACOM. Note, R_{LOAD} may represent the equivalent load resistance seen by I_{OUTA} or I_{OUTB} as would be the case in a doubly terminated 50 Ω or 75 Ω cable. The single-ended voltage output appearing at the I_{OUTA} and I_{OUTB} nodes is simply :

$$V_{OUTA} = I_{OUTA} \times R_{LOAD} \quad (5)$$

$$V_{OUTB} = I_{OUTB} \times R_{LOAD} \quad (6)$$

Note the full-scale value of V_{OUTA} and V_{OUTB} should not exceed the specified output compliance range to maintain specified distortion and linearity performance.

The differential voltage, V_{DIFF}, appearing across I_{OUTA} and I_{OUTB} is:

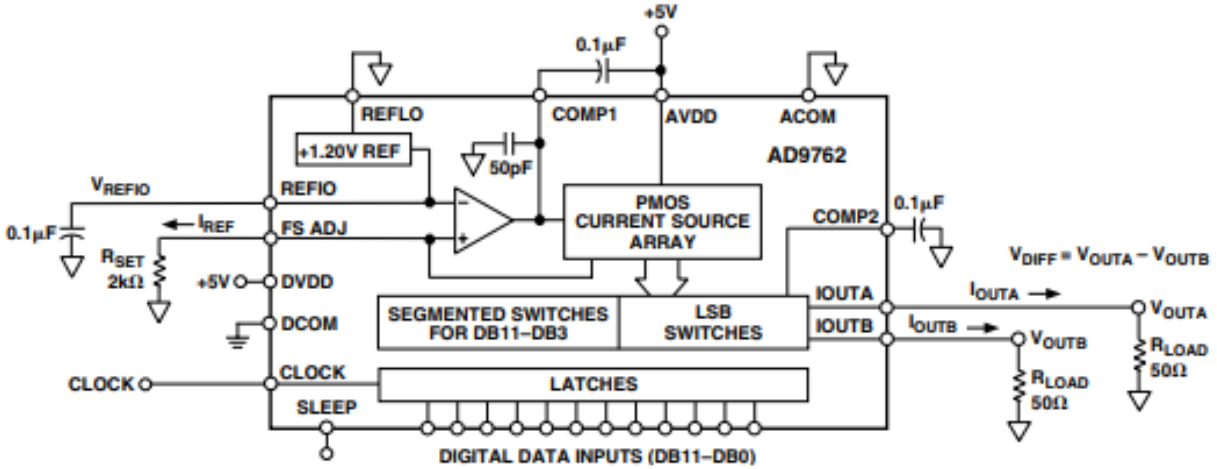
$$V_{DIFF} = (I_{OUTA} - I_{OUTB}) \times R_{LOAD} \quad (7)$$

Substituting the values of I_{OUTA}, I_{OUTB}, and I_{REF}; V_{DIFF} can be expressed as:

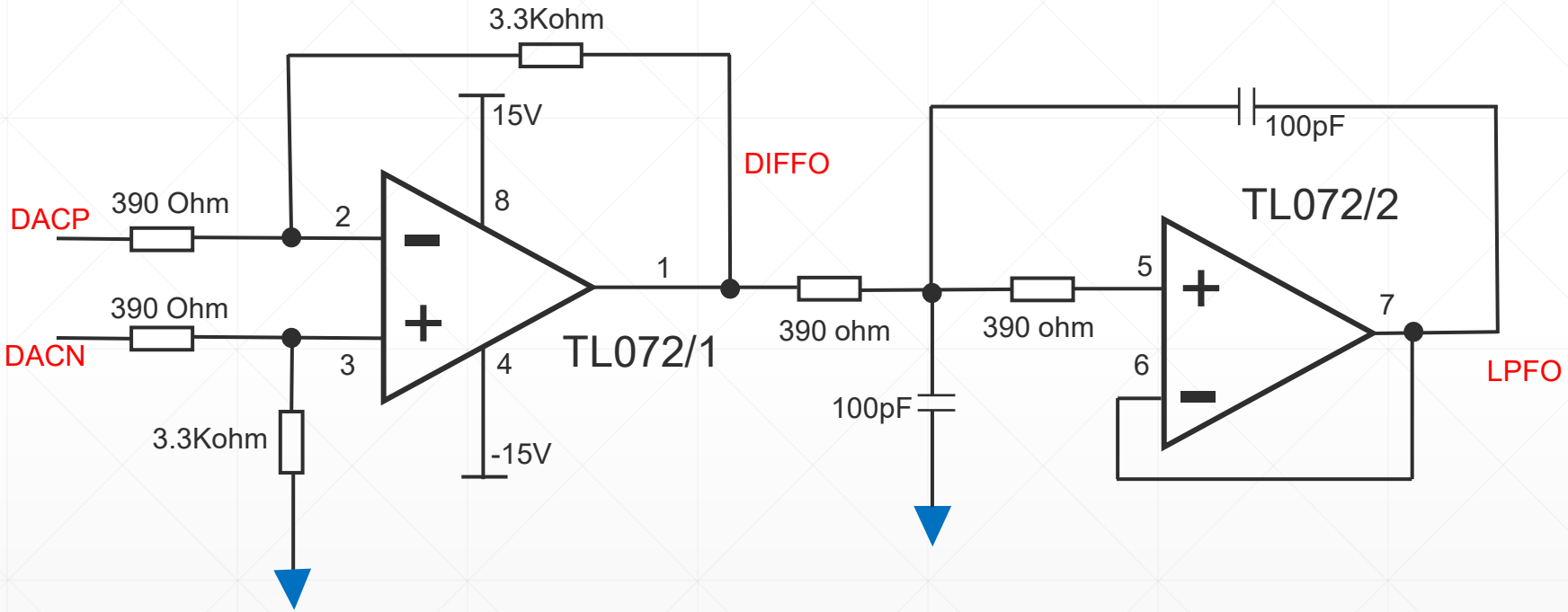
$$V_{DIFF} = \{(2\ DAC\ CODE - 4095)/4096\} \times (32\ R_{LOAD}/R_{SET}) \times V_{REFIO} \quad (8)$$

These last two equations highlight some of the advantages of operating the AD9762 differentially. First, the differential operation will help cancel common-mode error sources associated with I_{OUTA} and I_{OUTB} such as noise, distortion and dc offsets. Second, the differential code dependent current and subsequent voltage, V_{DIFF}, is twice the value of the single-ended voltage output (i.e., V_{OUTA} or V_{OUTB}), thus providing twice the signal power to the load.

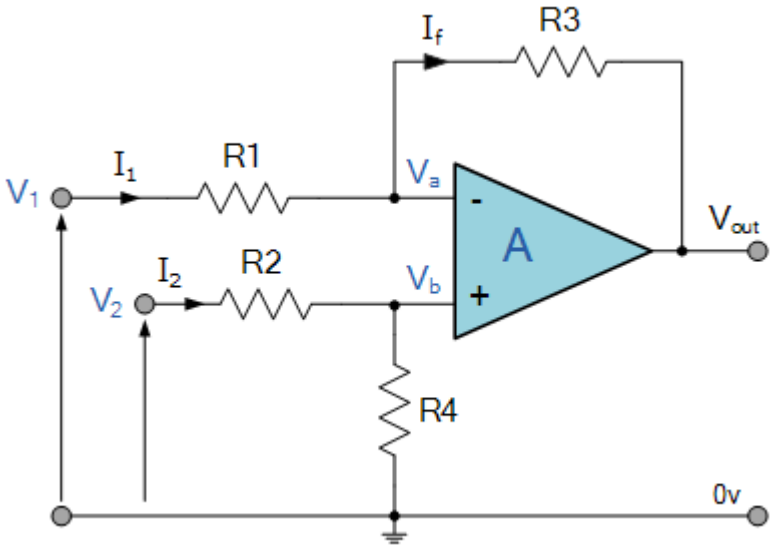
Note, the gain drift temperature performance for a single-ended (V_{OUTA} and V_{OUTB}) or differential output (V_{DIFF}) of the AD9762 can be enhanced by selecting temperature tracking resistors for R_{LOAD} and R_{SET} due to their ratiometric relationship as shown in Equation 8.



Differential Amplifier & LPF



Differential Amplifier



When resistors, R1 = R2 and R3 = R4 the above transfer function for the differential amplifier can be simplified to the following expression:

Differential Amplifier Equation

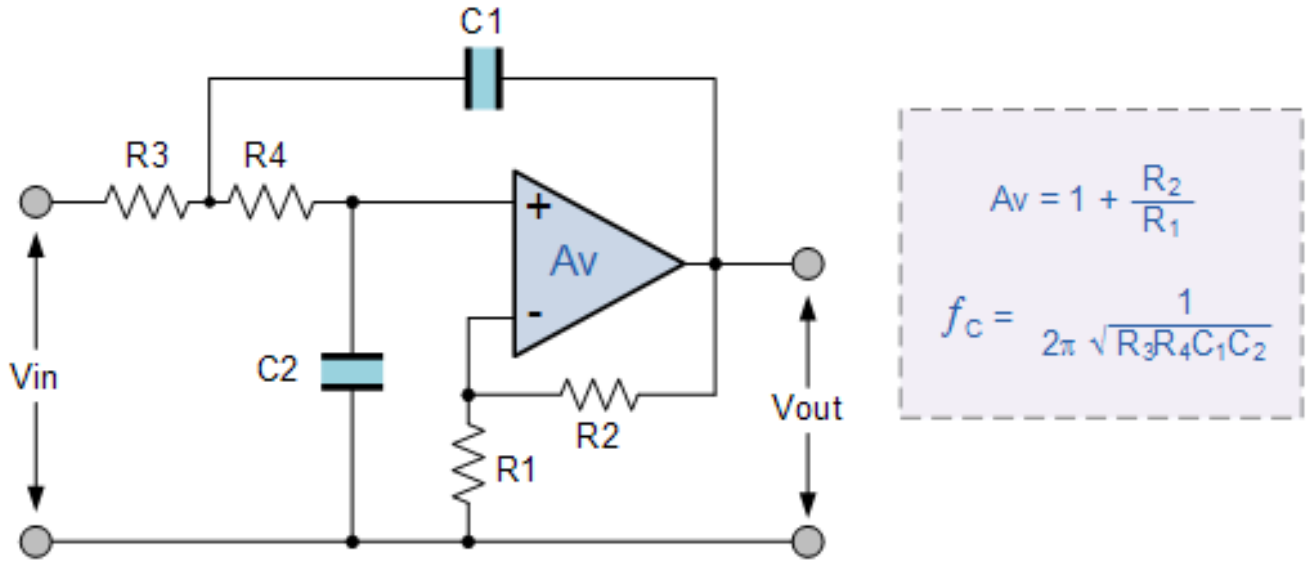
$$V_{OUT} = \frac{R_3}{R_1}(V_2 - V_1)$$

$$f = \frac{3300}{390}(v_2 - v_1) = 8.46(v2 - v1)$$

Low pass filter

KMITL
พระจอมเกล้าลาดกระบัง

Second-order Active Low Pass Filter Circuit



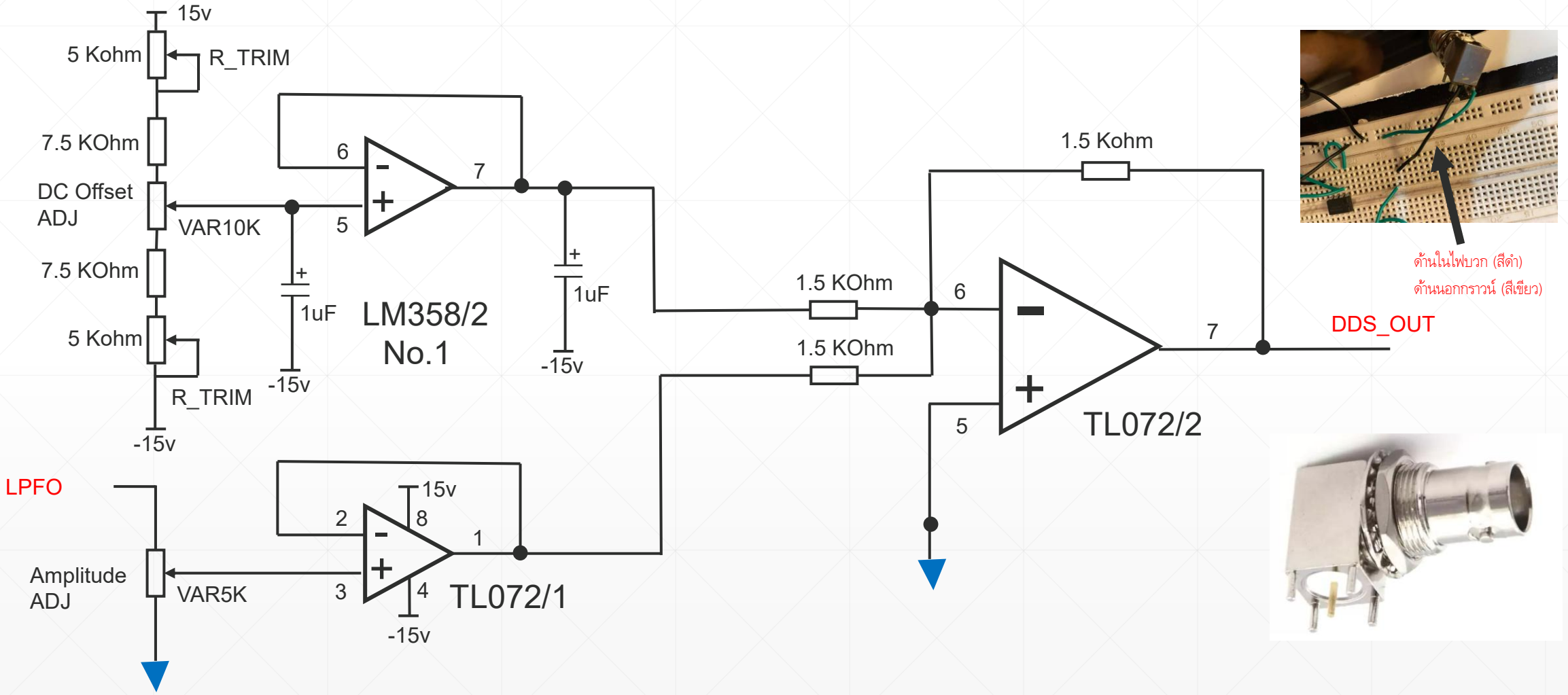
$$A_V = 1 + \frac{R_2}{R_1}$$

$$f_C = \frac{1}{2\pi\sqrt{R_3R_4C_1C_2}}$$

$$f = \frac{1}{2\pi RC} = \frac{1}{2\pi(390)(100p)} = 4\ MHz$$

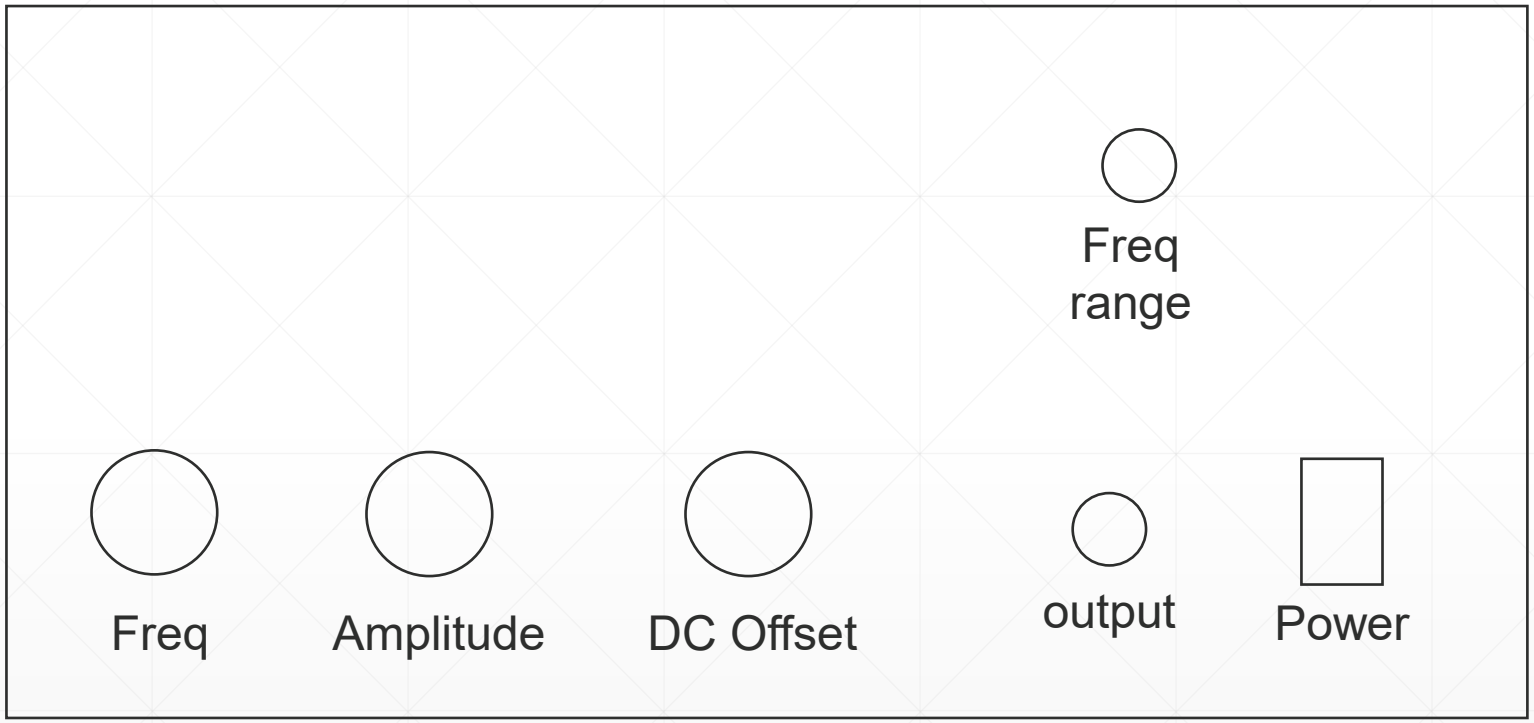
Summing Amp (Attenuator and DC Offset)

KMITL
พระจอมเกล้าลาดกระบัง

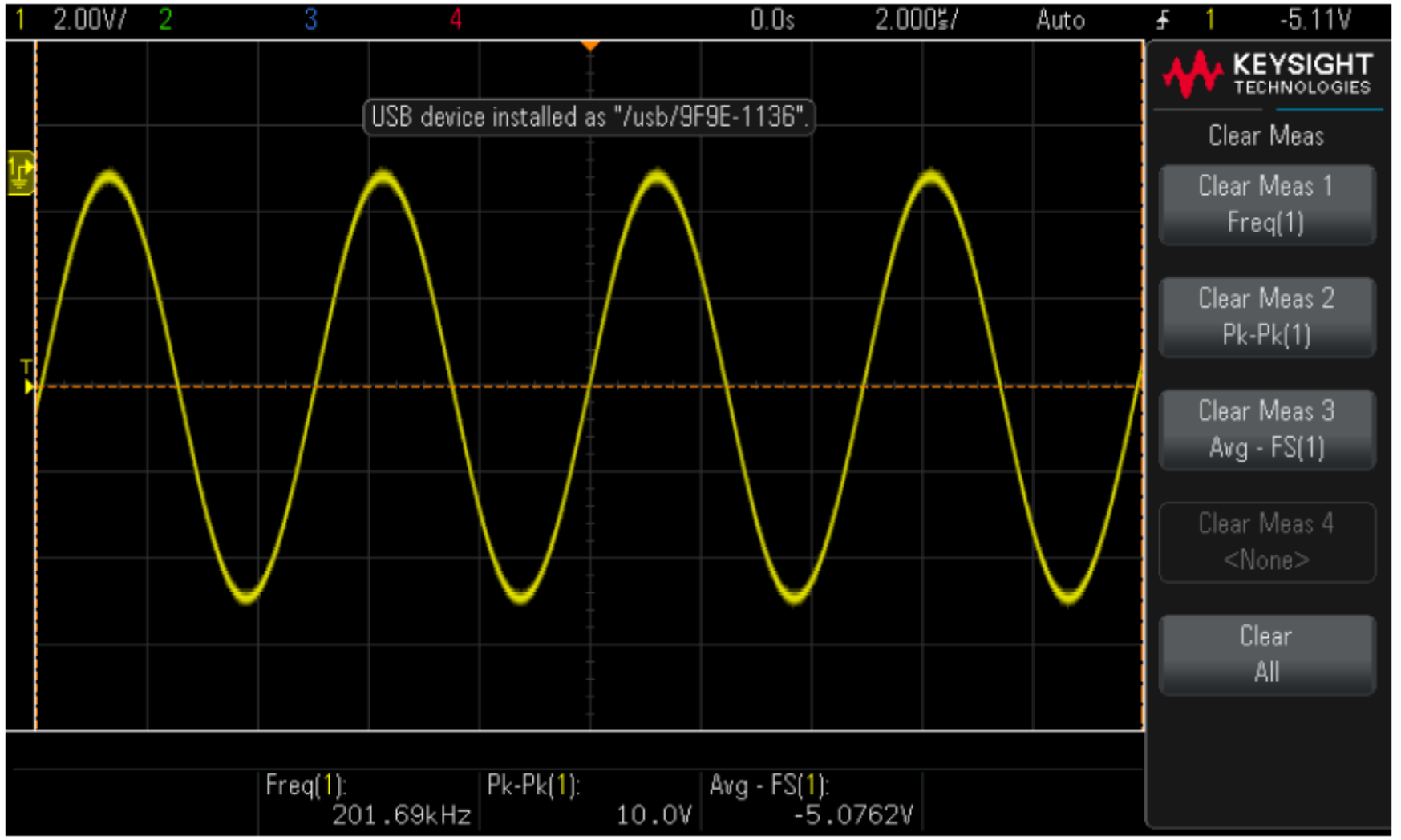


ด้านในไฟบวก (สีดำ)
ด้านนอกกราวน์ (สีเขียว)

DDS Case

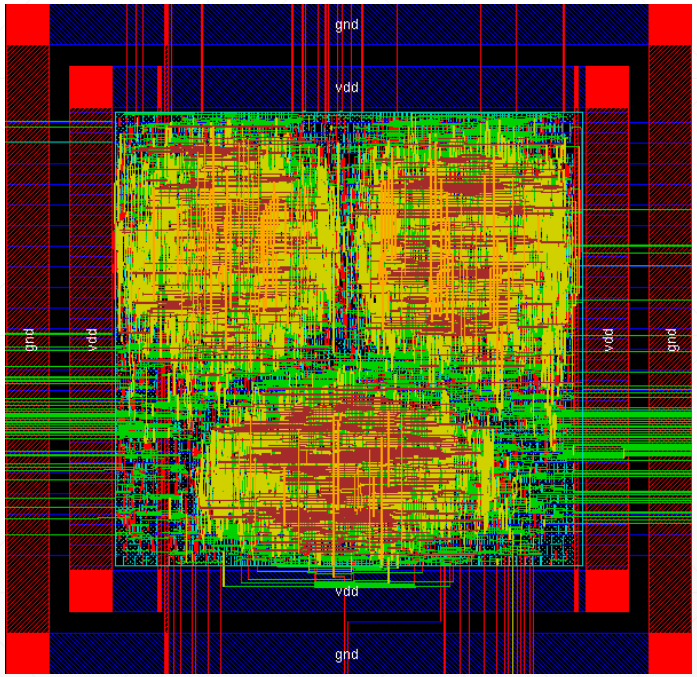



Mode


1 2.00V/ 2 3 4 0.0s 2.000s/ Auto 1 -5.11V
USB device installed as "/usb/9F9E-1136".
KEYSIGHT TECHNOLOGIES
Clear Meas
Clear Meas 1 Freq(1)
Clear Meas 2 Pk-Pk(1)
Clear Meas 3 Avg - FS(1)
Clear Meas 4 <None>
Clear All
Freq(1): 201.69kHz
Pk-Pk(1): 10.0V
Avg - FS(1): -5.0762V

KMITL
พระจอมเกล้าลาดกระบัง

Chip Implementation



Implementation Checklist

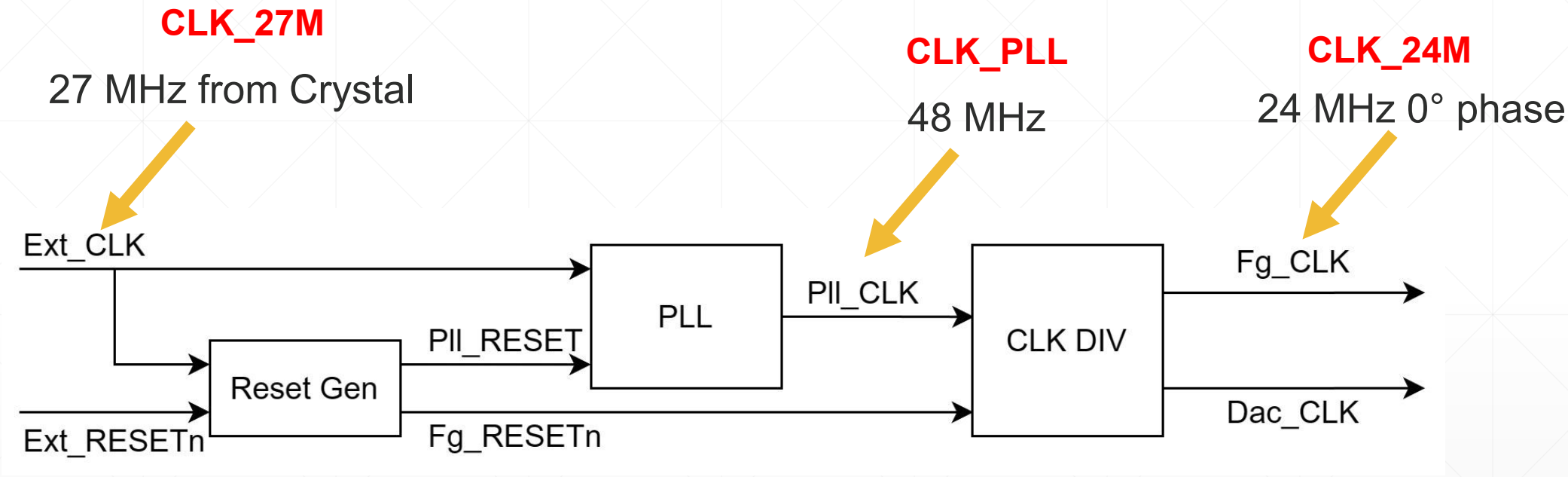
KMITL
พระจอมเกล้าลาดกระบัง

1. Correct Function -> Logic Simulation 

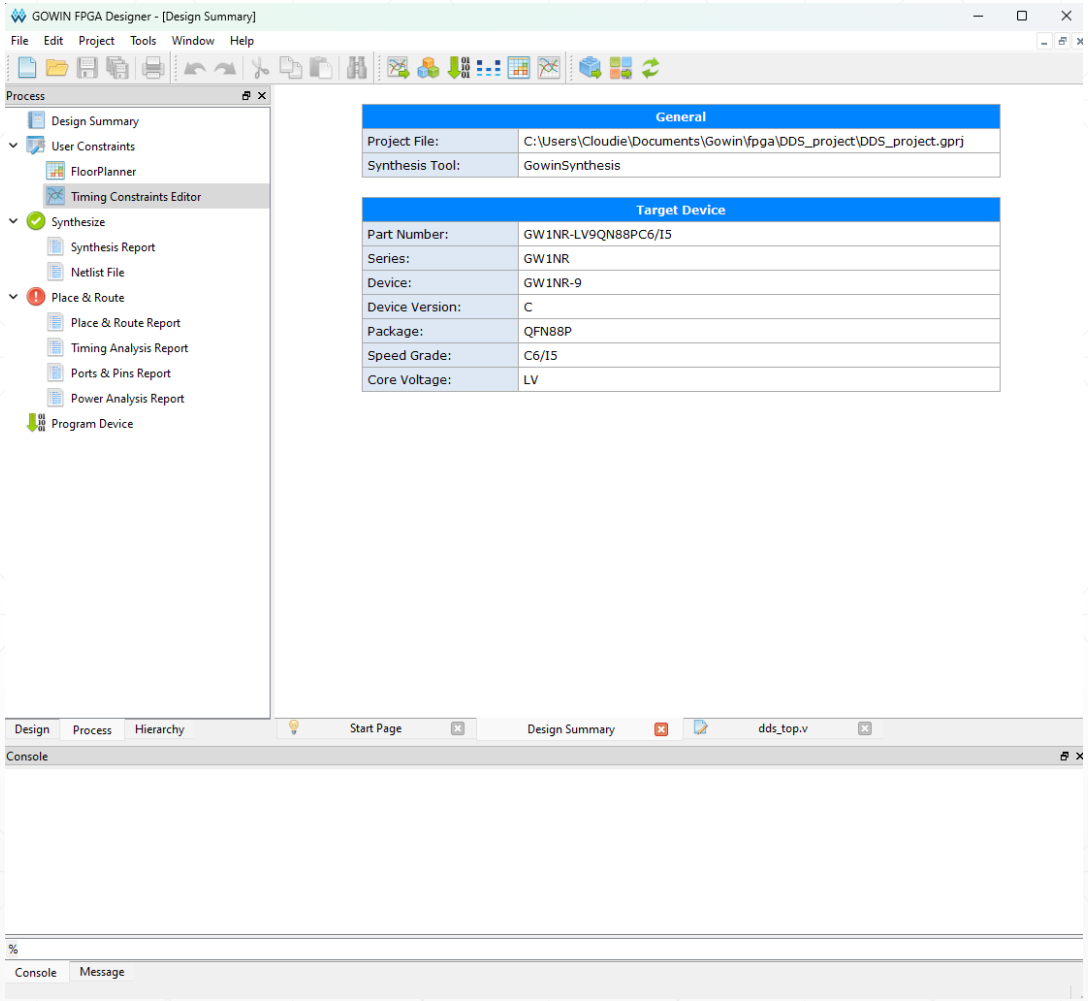
2. Synthesizable Code -> Synthesis Tool 

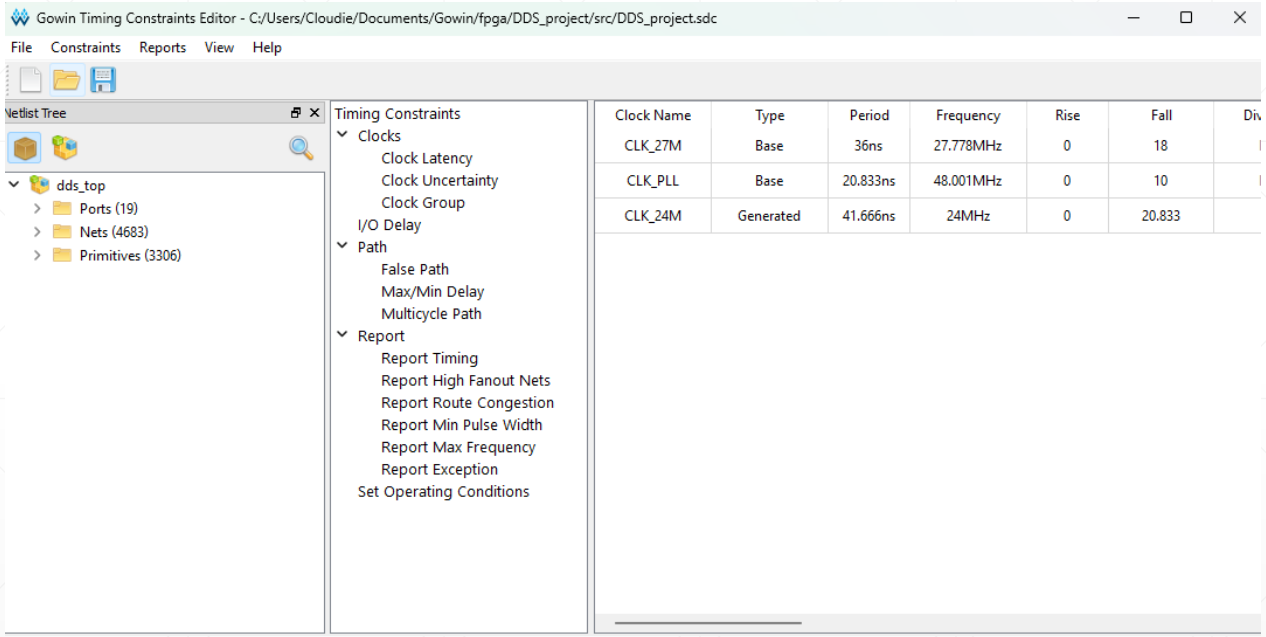
3. Place and Route -> Floorplan and Synthesis Tool 

Timing Constraint

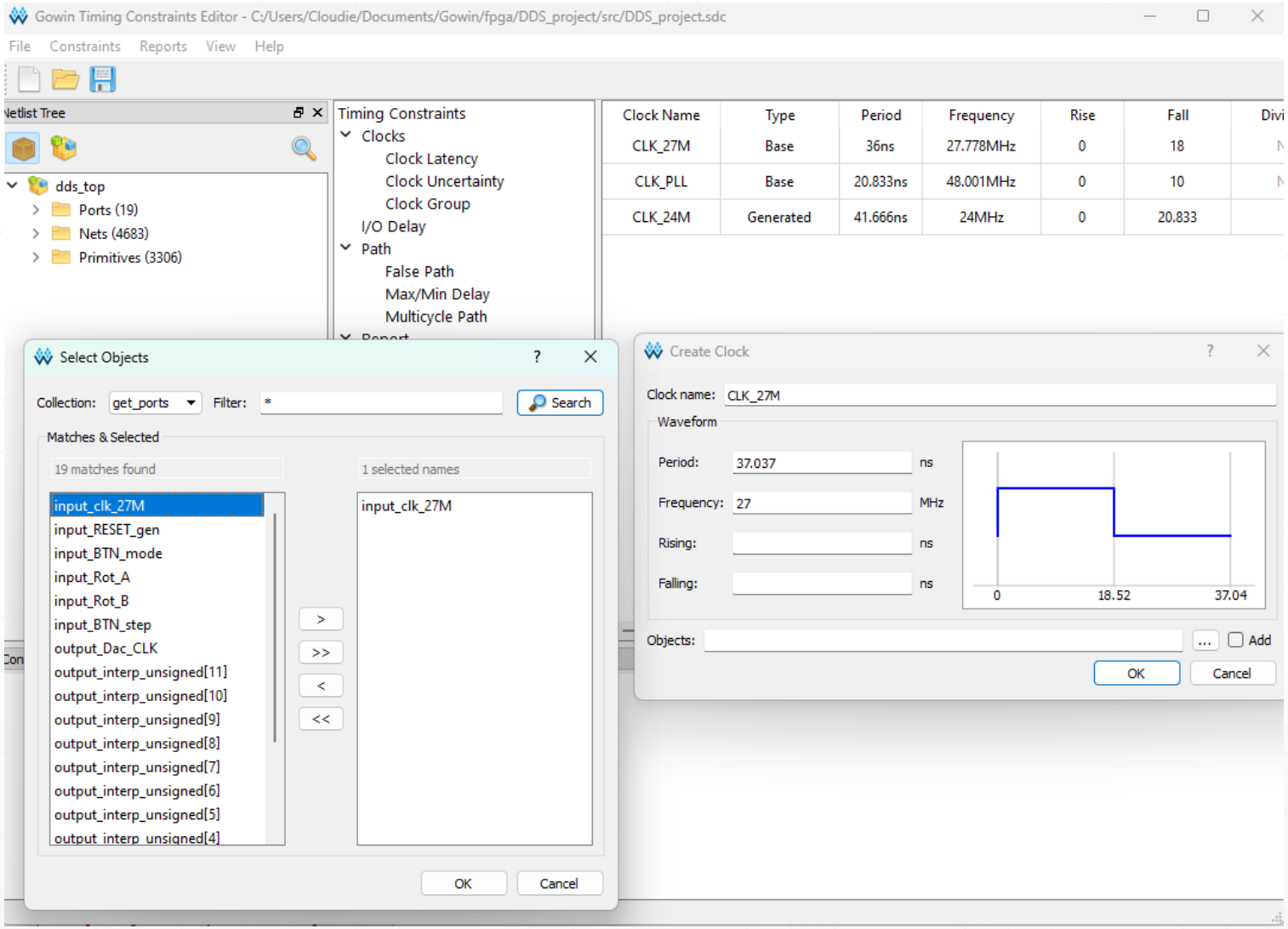


Timing Constraint Setup

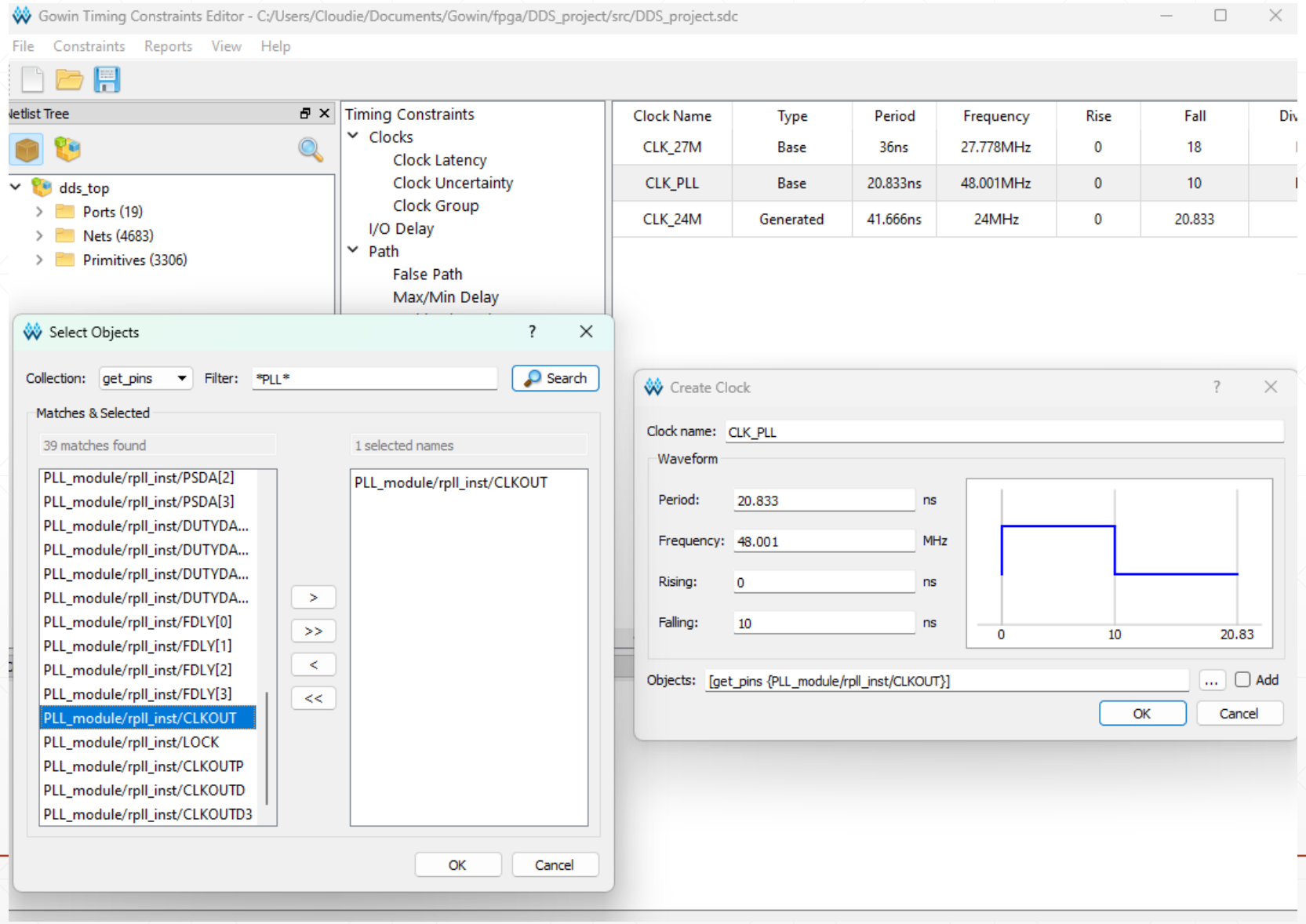




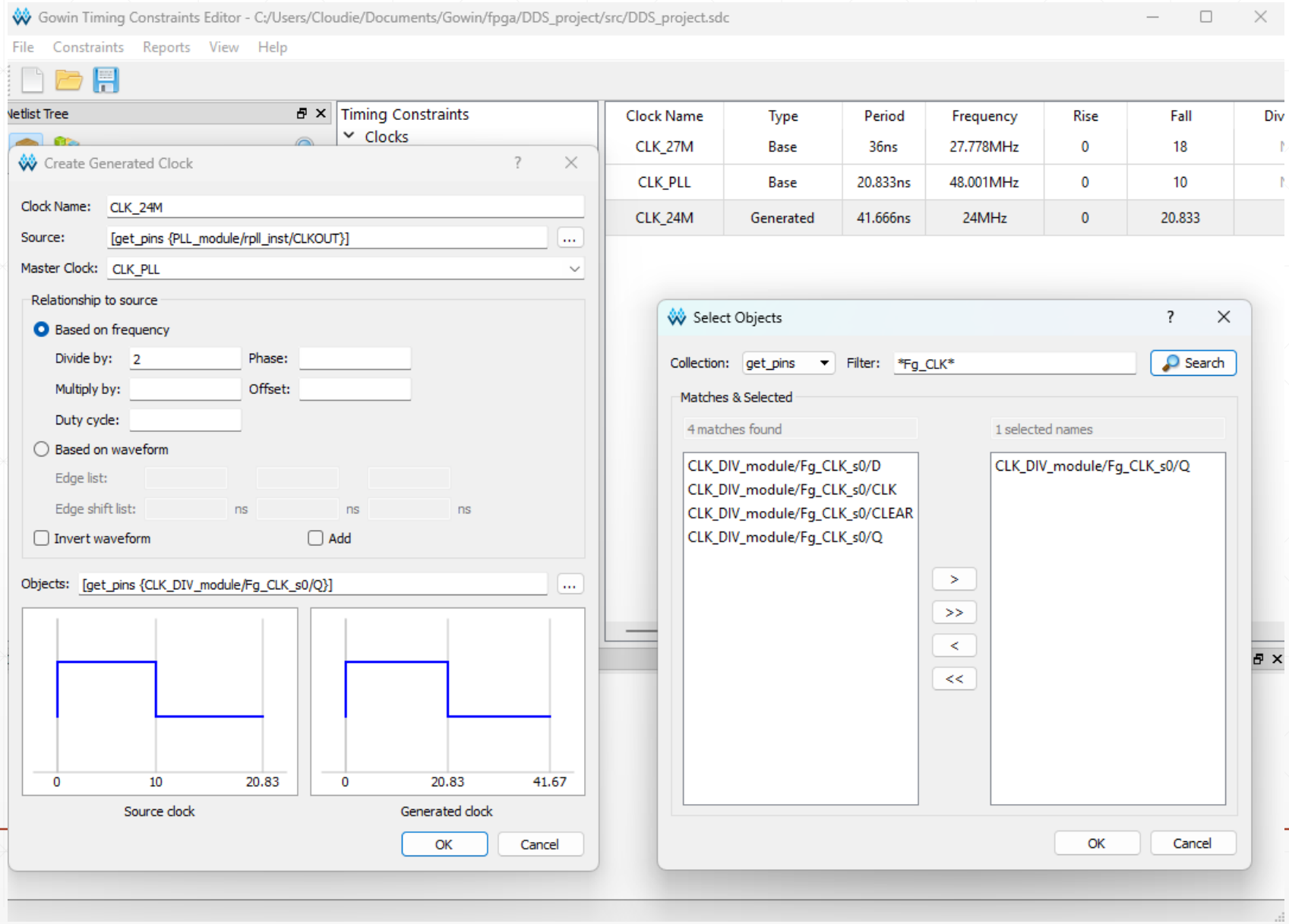
Timing Constraint Setup



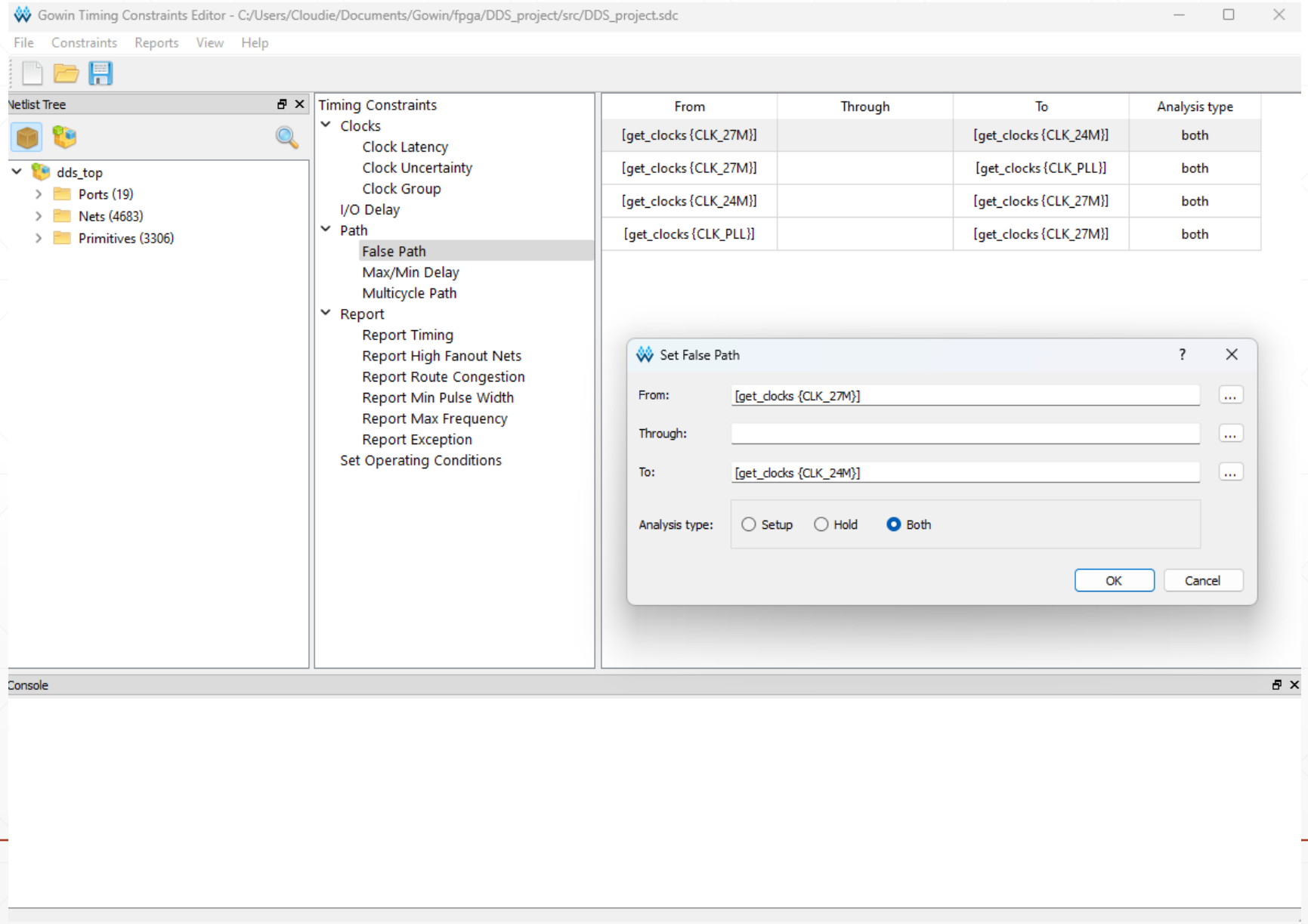
Timing Constraint Setup



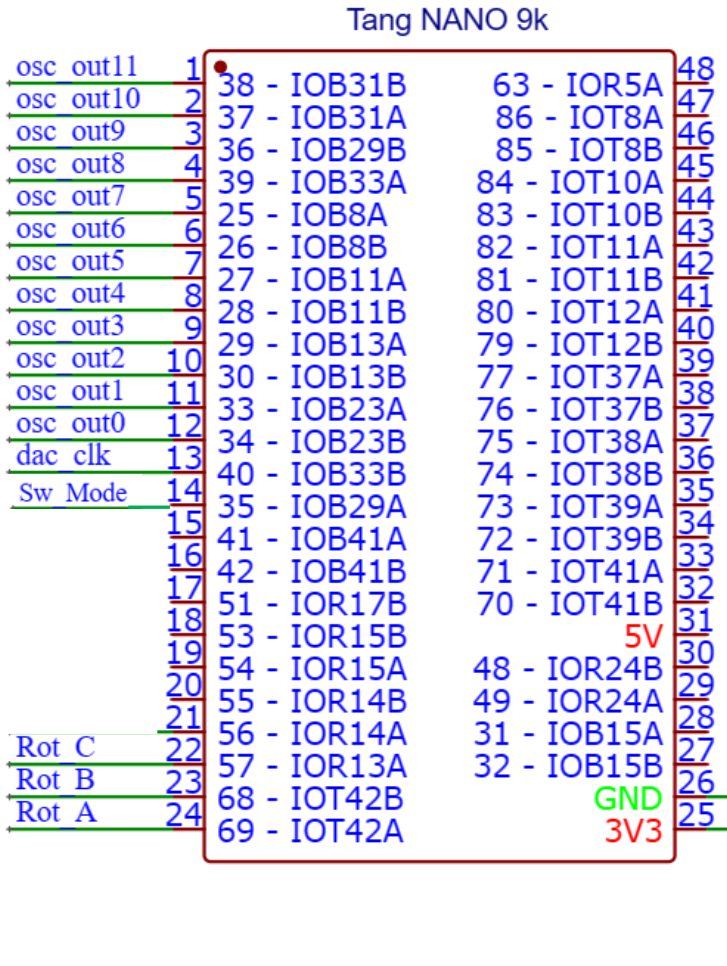
Timing Constraint Setup

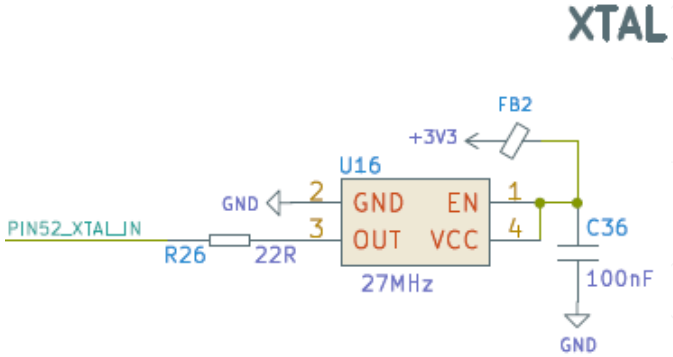


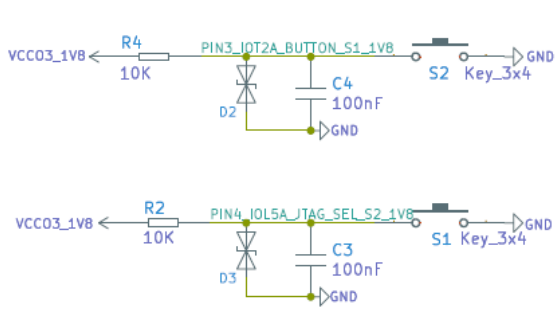
Set False Path

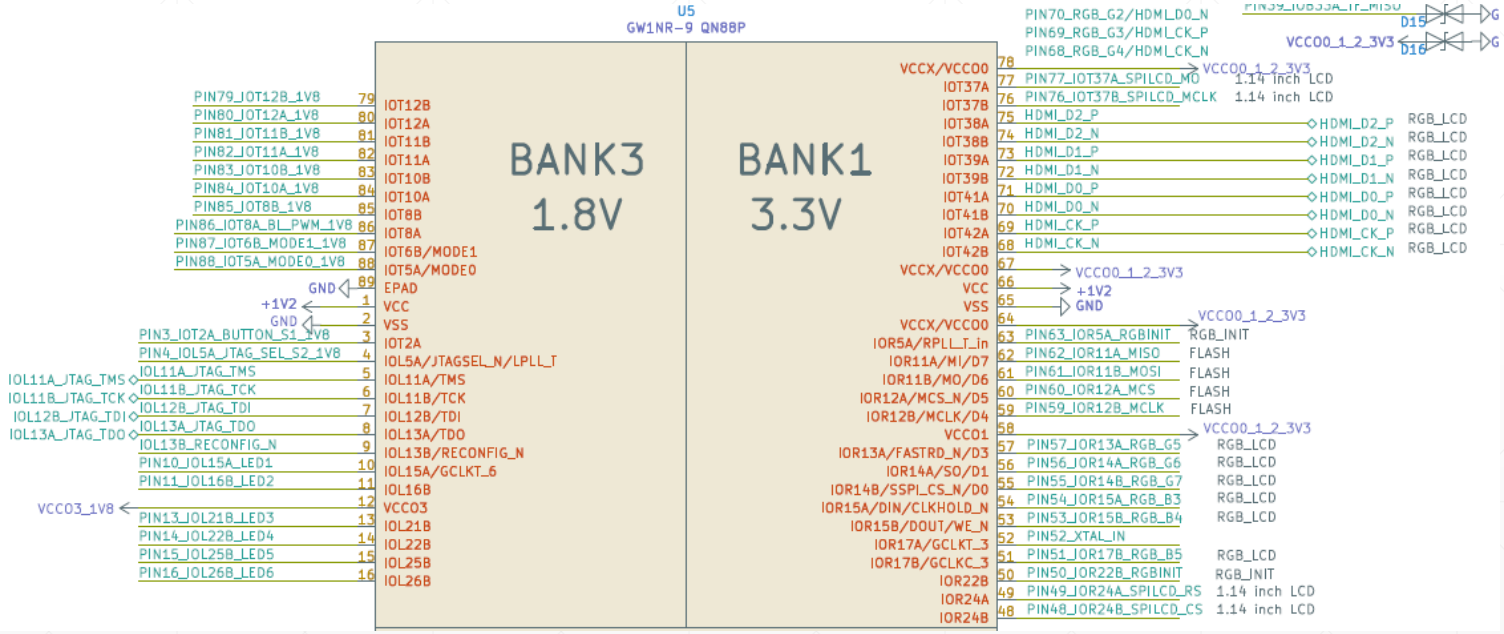


I/O Setup

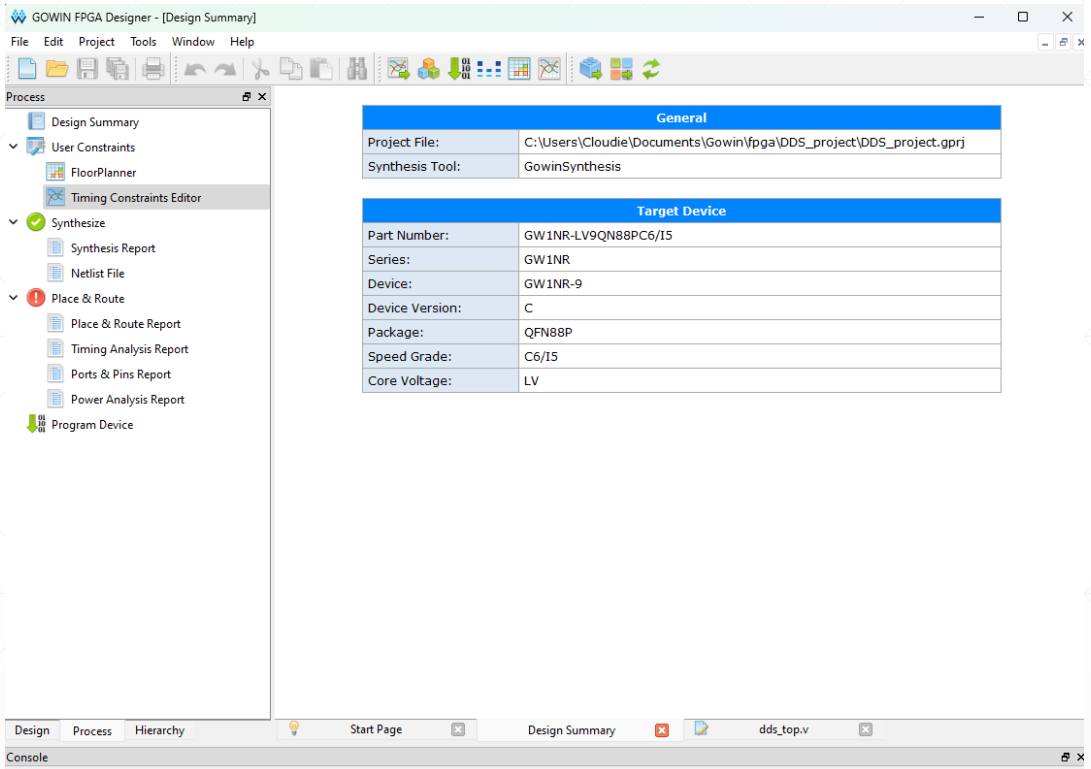

Tang NANO 9k
osc_out11 1
osc_out10 2
osc_out9 3
osc_out8 4
osc_out7 5
osc_out6 6
osc_out5 7
osc_out4 8
osc_out3 9
osc_out2 10
osc_out1 11
osc_out0 12
dac_clk 13
Sw_Mode 14
15
16
17
18
19
20
21
Rot_C 22
Rot_B 23
Rot_A 24
38 - IOB31B
37 - IOB31A
36 - IOB29B
39 - IOB33A
25 - IOB8A
26 - IOB8B
27 - IOB11A
28 - IOB11B
29 - IOB13A
30 - IOB13B
33 - IOB23A
34 - IOB23B
40 - IOB33B
35 - IOB29A
41 - IOB41A
42 - IOB41B
51 - IOR17B
53 - IOR15B
54 - IOR15A
55 - IOR14B
56 - IOR14A
57 - IOR13A
68 - IOT42B
69 - IOT42A
63 - IOR5A 48
86 - IOT8A 47
85 - IOT8B 46
84 - IOT10A 45
83 - IOT10B 44
82 - IOT11A 43
81 - IOT11B 42
80 - IOT12A 41
79 - IOT12B 40
77 - IOT37A 39
76 - IOT37B 38
75 - IOT38A 37
74 - IOT38B 36
73 - IOT39A 35
72 - IOT39B 34
71 - IOT41A 33
70 - IOT41B 32
5V 31
48 - IOR24B 30
49 - IOR24A 29
31 - IOB15A 28
32 - IOB15B 27
GND 26
3V3 25
3V3
GND

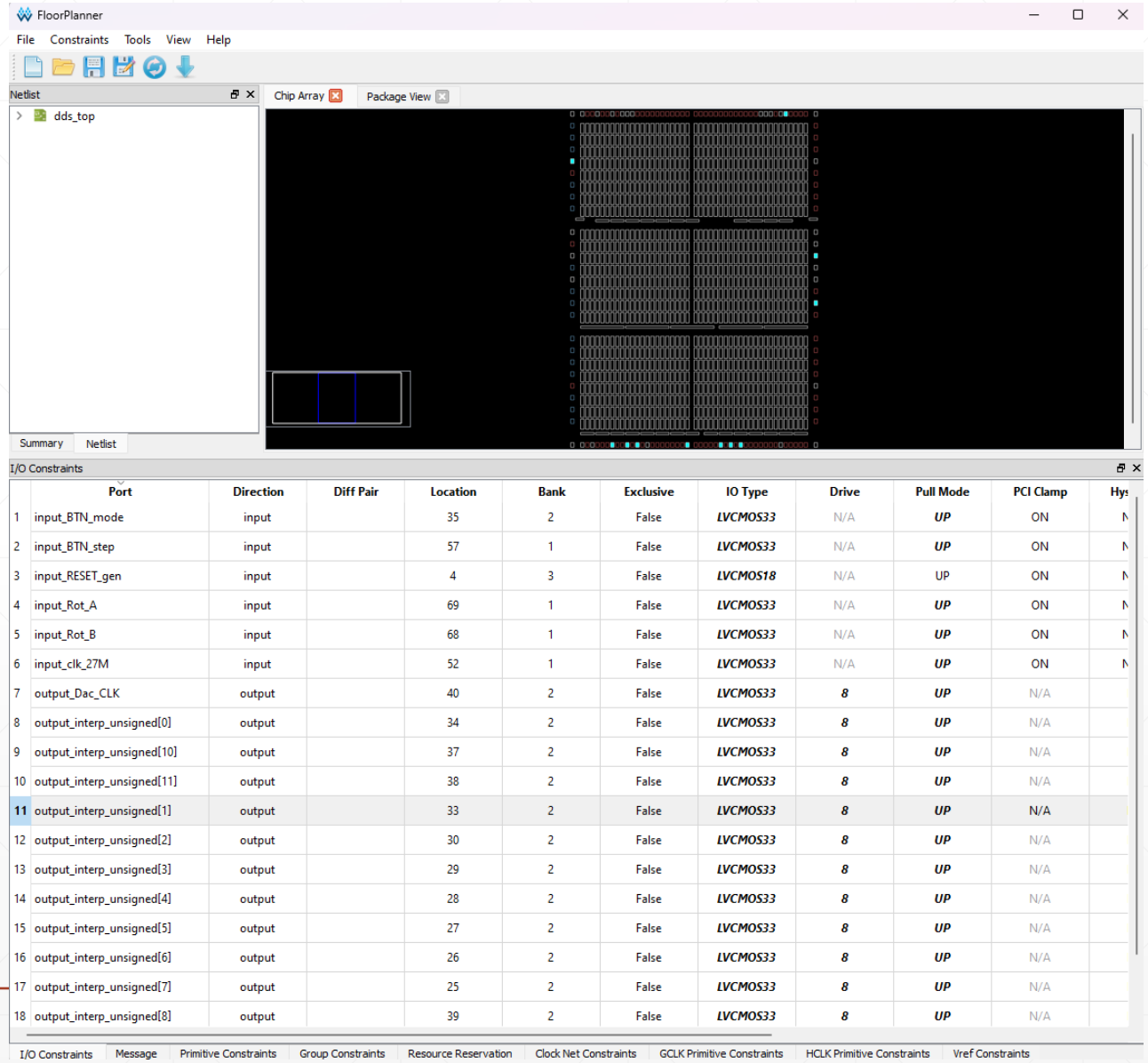

XTAL
FB2
+3V3
U16
GND
2
3
GND EN
OUT VCC
1
4
C36
100nF
GND
PIN52_XTAL_IN
R26
22R
27MHz


Button x 2
VCCO3_1V8
R4
10K
PIN3_IOT2A_BUTTON_S1_1V8
S2 Key_3x4
GND
C4
100nF
D2
R2
PIN4_IOL5A_JTAG_SEL_S2_1V8
S1 Key_3x4
C3
D3


U5
GW1NR-9 QN88P
BANK3
1.8V
BANK1
3.3V
IOT12B
IOT12A
IOT11B
IOT11A
IOT10B
IOT10A
IOT8B
IOT8A
IOT6B/MODE1
IOT5A/MODE0
EPAD
VCC
VSS
IOT2A
IOL5A/JTAGSEL_N/LPLL_T
IOL11A/TMS
IOL11B/TCK
IOL12B/TDI
IOL13A/TDO
IOL13B/RECONFIG_N
IOL15A/GCLKT_6
IOL16B
VCCO3
IOL21B
IOL22B
IOL25B
IOL26B
VCCX/VCCO0
IOT37A
IOT37B
IOT38A
IOT38B
IOT39A
IOT39B
IOT41A
IOT41B
IOT42A
IOT42B
VCC
VSS
IOR5A/RPLL_T_in
IOR11A/MI/D7
IOR11B/MO/D6
IOR12A/MCS_N/D5
IOR12B/MCLK/D4
VCCO1
IOR13A/FASTRD_N/D3
IOR14A/SO/D1
IOR14B/SSPI_CS_N/D0
IOR15A/DIN/CLKHOLD_N
IOR15B/DOUT/WE_N
IOR17A/GCLKT_3
IOR17B/GCLKC_3
IOR22B
IOR24A
IOR24B
PIN79_IOT12B_1V8
PIN80_IOT12A_1V8
PIN81_IOT11B_1V8
PIN82_IOT11A_1V8
PIN83_IOT10B_1V8
PIN84_IOT10A_1V8
PIN85_IOT8B_1V8
PIN86_IOT8A_BL_PWM_1V8
PIN87_IOT6B_MODE1_1V8
PIN88_IOT5A_MODE0_1V8
PIN3_IOT2A_BUTTON_S1_1V8
PIN4_IOL5A_JTAG_SEL_S2_1V8
IOL11A_JTAG_TMS
IOL11B_JTAG_TCK
IOL12B_JTAG_TDI
IOL13A_JTAG_TDO
IOL13B_RECONFIG_N
PIN10_IOL15A_LED1
PIN11_IOL16B_LED2
PIN13_IOL21B_LED3
PIN14_IOL22B_LED4
PIN15_IOL25B_LED5
PIN16_IOL26B_LED6
PIN70_RGB_G2/HDMI_D0_N
PIN69_RGB_G3/HDMI_CK_P
PIN68_RGB_G4/HDMI_CK_N
PIN77_IOT37A_SPILCD_MO
PIN76_IOT37B_SPILCD_MCLK
HDMI_D2_P
HDMI_D2_N
HDMI_D1_P
HDMI_D1_N
HDMI_D0_P
HDMI_D0_N
HDMI_CK_P
HDMI_CK_N
PIN63_IOR5A_RGBINIT
PIN62_IOR11A_MISO
PIN61_IOR11B_MOSI
PIN60_IOR12A_MCS
PIN59_IOR12B_MCLK
PIN57_IOR13A_RGB_G5
PIN56_IOR14A_RGB_G6
PIN55_IOR14B_RGB_G7
PIN54_IOR15A_RGB_B3
PIN53_IOR15B_RGB_B4
PIN52_XTAL_IN
PIN51_IOR17B_RGB_B5
PIN50_IOR22B_RGBINIT
PIN49_IOR24A_SPILCD_RS
PIN48_IOR24B_SPILCD_CS
VCCO0_1_2_3V3
+1V2
GND
RGB_LCD
RGB_INIT
FLASH
1.14 inch LCD

I/O Setup

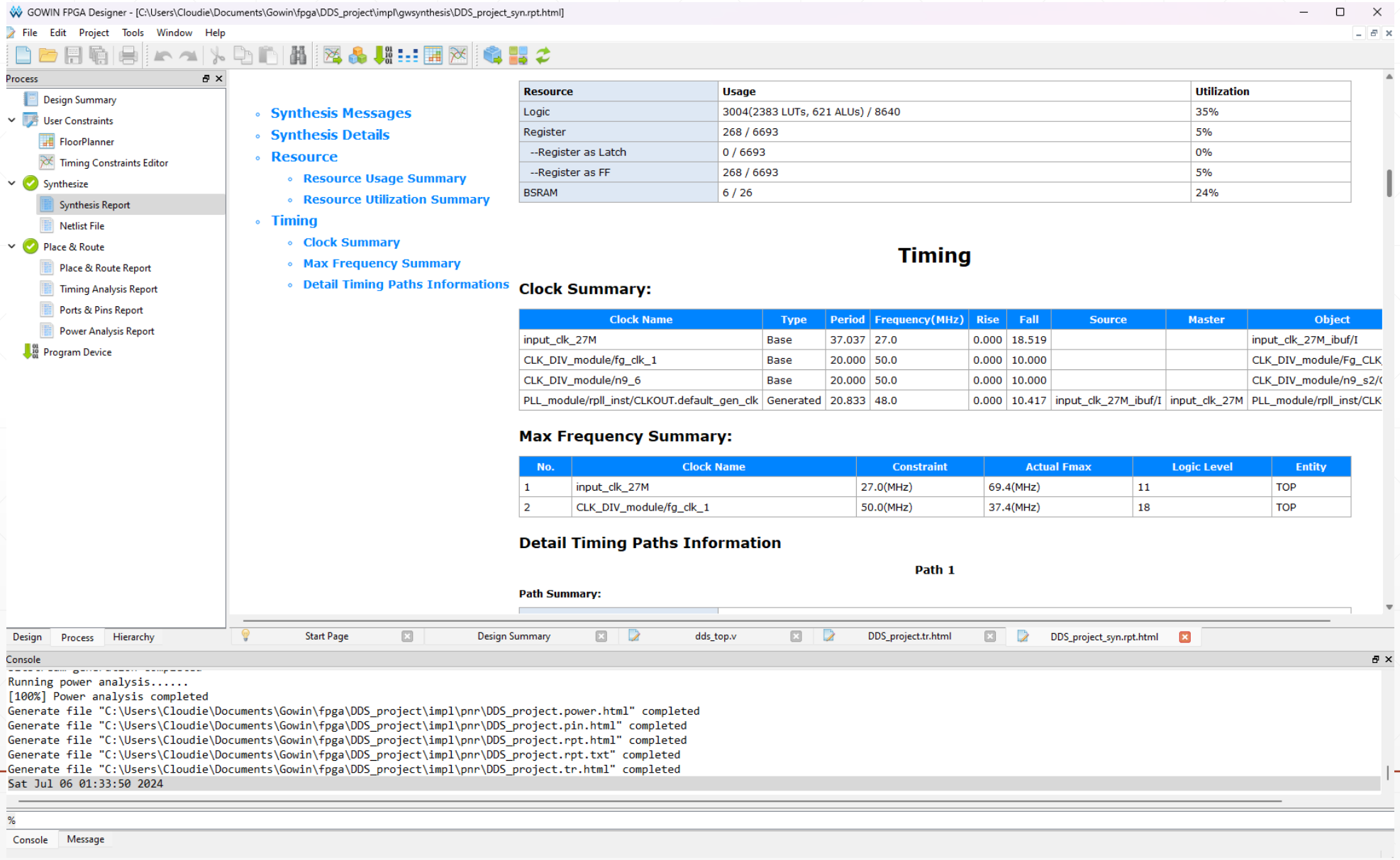




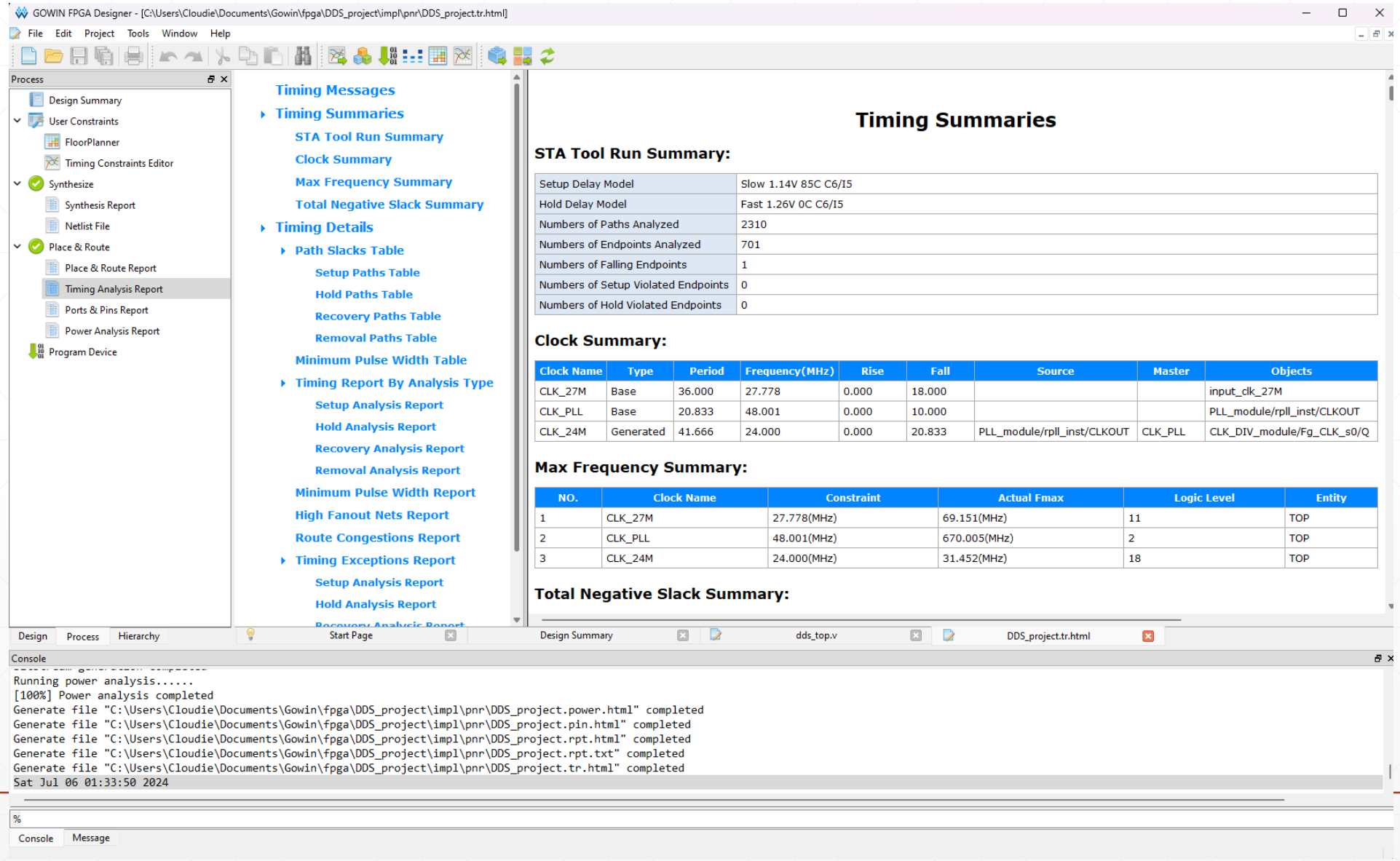
	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Drive	Pull Mode	PCI Clamp	Hys
1	input_BTN_mode	input		35	2	False	LVCMOS33	N/A	UP	ON	N
2	input_BTN_step	input		57	1	False	LVCMOS33	N/A	UP	ON	N
3	input_RESET_gen	input		4	3	False	LVCMOS18	N/A	UP	ON	N
4	input_Rot_A	input		69	1	False	LVCMOS33	N/A	UP	ON	N
5	input_Rot_B	input		68	1	False	LVCMOS33	N/A	UP	ON	N
6	input_clk_27M	input		52	1	False	LVCMOS33	N/A	UP	ON	N
7	output_Dac_CLK	output		40	2	False	LVCMOS33	8	UP	N/A	
8	output_interp_unsigned[0]	output		34	2	False	LVCMOS33	8	UP	N/A	
9	output_interp_unsigned[10]	output		37	2	False	LVCMOS33	8	UP	N/A	
10	output_interp_unsigned[11]	output		38	2	False	LVCMOS33	8	UP	N/A	
11	output_interp_unsigned[1]	output		33	2	False	LVCMOS33	8	UP	N/A	
12	output_interp_unsigned[2]	output		30	2	False	LVCMOS33	8	UP	N/A	
13	output_interp_unsigned[3]	output		29	2	False	LVCMOS33	8	UP	N/A	
14	output_interp_unsigned[4]	output		28	2	False	LVCMOS33	8	UP	N/A	
15	output_interp_unsigned[5]	output		27	2	False	LVCMOS33	8	UP	N/A	
16	output_interp_unsigned[6]	output		26	2	False	LVCMOS33	8	UP	N/A	
17	output_interp_unsigned[7]	output		25	2	False	LVCMOS33	8	UP	N/A	
18	output_interp_unsigned[8]	output		39	2	False	LVCMOS33	8	UP	N/A	

I/O Constraints | Message | Primitive Constraints | Group Constraints | Resource Reservation | Clock Net Constraints | GCLK Primitive Constraints | HCLK Primitive Constraints | Vref Constraints

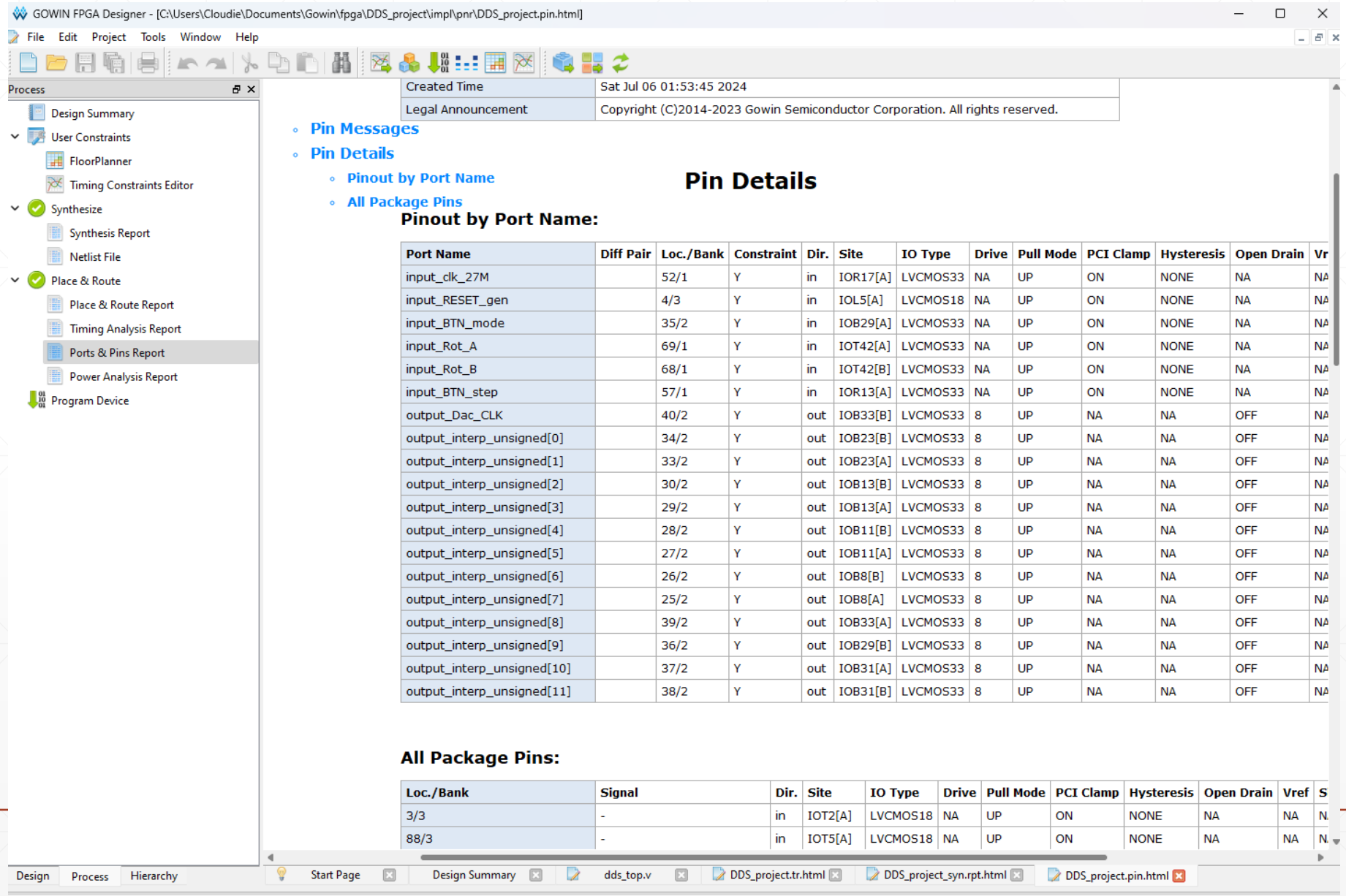
Synthesis Report



Timing Analysis Report



Ports & Pins Report



Created Time	Sat Jul 06 01:53:45 2024
Legal Announcement	Copyright (C)2014-2023 Gowin Semiconductor Corporation. All rights reserved.

- Pin Messages
- Pin Details
  - Pinout by Port Name
  - All Package Pins

Pin Details

Pinout by Port Name:

Port Name	Diff Pair	Loc./Bank	Constraint	Dir.	Site	IO Type	Drive	Pull Mode	PCI Clamp	Hysteresis	Open Drain	Vr
input_clk_27M		52/1	Y	in	IOR17[A]	LVCMOS33	NA	UP	ON	NONE	NA	NA
input_RESET_gen		4/3	Y	in	IOL5[A]	LVCMOS18	NA	UP	ON	NONE	NA	NA
input_BTN_mode		35/2	Y	in	IOB29[A]	LVCMOS33	NA	UP	ON	NONE	NA	NA
input_Rot_A		69/1	Y	in	IOT42[A]	LVCMOS33	NA	UP	ON	NONE	NA	NA
input_Rot_B		68/1	Y	in	IOT42[B]	LVCMOS33	NA	UP	ON	NONE	NA	NA
input_BTN_step		57/1	Y	in	IOR13[A]	LVCMOS33	NA	UP	ON	NONE	NA	NA
output_Dac_CLK		40/2	Y	out	IOB33[B]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[0]		34/2	Y	out	IOB23[B]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[1]		33/2	Y	out	IOB23[A]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[2]		30/2	Y	out	IOB13[B]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[3]		29/2	Y	out	IOB13[A]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[4]		28/2	Y	out	IOB11[B]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[5]		27/2	Y	out	IOB11[A]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[6]		26/2	Y	out	IOB8[B]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[7]		25/2	Y	out	IOB8[A]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[8]		39/2	Y	out	IOB33[A]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[9]		36/2	Y	out	IOB29[B]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[10]		37/2	Y	out	IOB31[A]	LVCMOS33	8	UP	NA	NA	OFF	NA
output_interp_unsigned[11]		38/2	Y	out	IOB31[B]	LVCMOS33	8	UP	NA	NA	OFF	NA

All Package Pins:

Loc./Bank	Signal	Dir.	Site	IO Type	Drive	Pull Mode	PCI Clamp	Hysteresis	Open Drain	Vref	S
3/3	-	in	IOT2[A]	LVCMOS18	NA	UP	ON	NONE	NA	NA	N
88/3	-	in	IOT5[A]	LVCMOS18	NA	UP	ON	NONE	NA	NA	N

DDS Specification

KMITL
พระจอมเกล้าลาดกระบัง

1. มีเอาท์พุทเป็นสัญญาณไซน์ที่ความถี่ตั้งแต่ 50 Hz ถึง ~~1 MHz~~ 200KHz .แรงดันเอาท์พุทตั้งแต่ 1 Vpp ถึง 10 Vpp

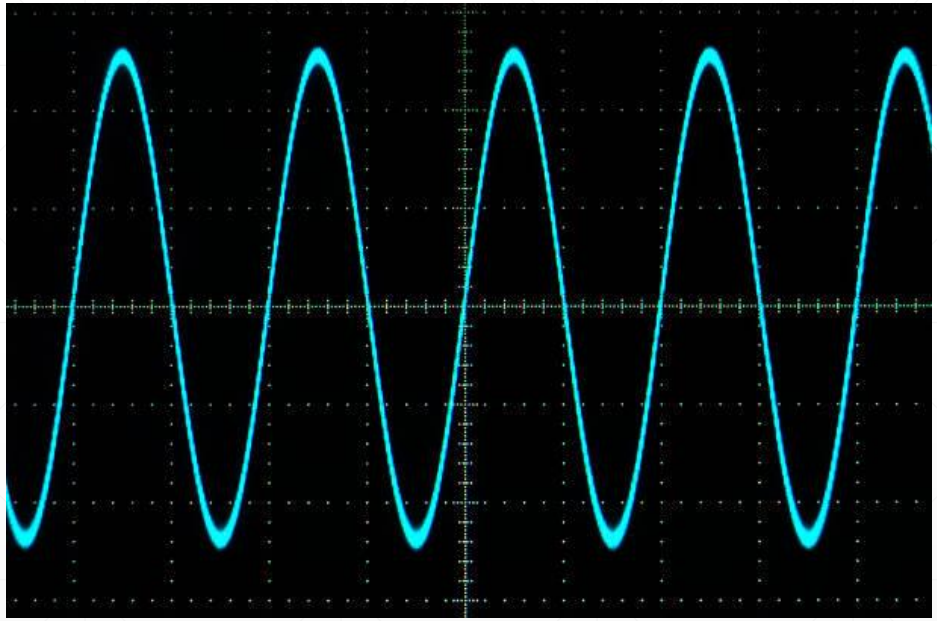
3. ต้องมีแรงดันออฟเซทเป็นไม่เกิน 30 mV ที่เอาท์พุท (จะสามารถปรับแรงดันออฟเซทเป็นค่าอื่น ๆ ได้หรือไม่ก็ได้)

4. คุณสมบัติของสัญญาณไซน์ที่สร้างได้จะต้องมี Total Harmonic Distortion (THD-F) ที่ Harmonic ที่ 1-4 น้อยกว่า 5% ตลอดช่วงความถี่ที่แรงดัน 10 Vpp

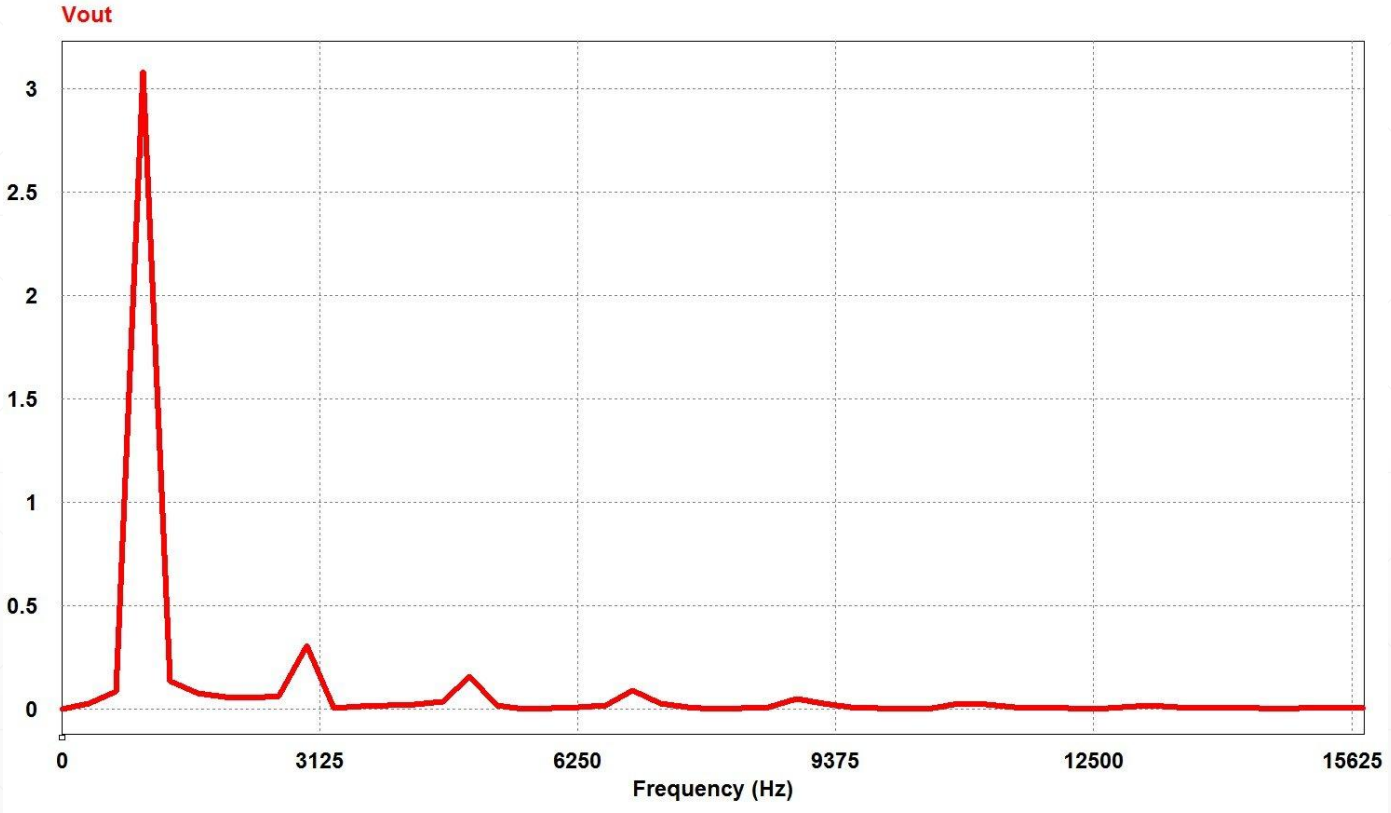
6. ไม่จำกัดวิธีการปรับความถี่, แอมพลิจูดและออฟเซท สามารถควบคุมจากวงจรดิจิทัลหรืออนาล็อกก็ได้

KMITL
พระจอมเกล้าลาดกระบัง

DDS Evaluation



Total Harmonic Distortion (THD)

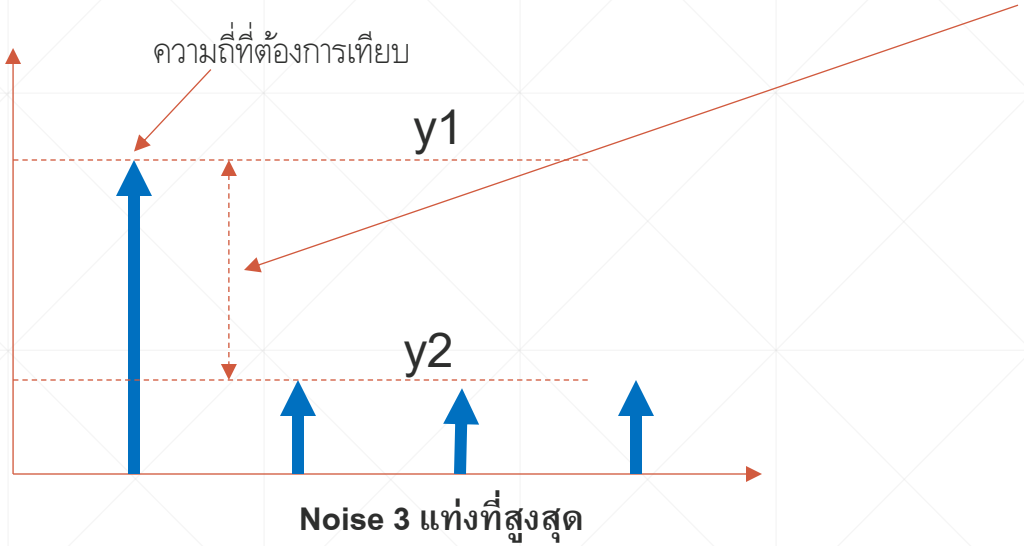


V Peak

$$THD = \frac{\sqrt{v_2^2 + v_3^2 + v_4^2}}{v_1}$$

กรณีนับแค่ 3 Harmonics

วิธีวัดค่า THD จากสโคป



Noise 3 แท่งที่สูงสุด

$$y2 - y1 = 10\log(v2)^2 - 10\log(v1)^2$$ หน่วย dB

$$\frac{y2 - y1}{10} = \log(v2)^2 - \log(v1)^2$$

$$\frac{y2 - y1}{10} = \log(\frac{v2}{v1})^2$$

$$10^{(\frac{y2-y1}{10})} = \frac{(v2)^2}{(v1)^2}$$ สมการที่ 2

$$THD = \frac{\sqrt{v_2^2 + v_3^2 + v_4^2}}{v_1} = \sqrt{\frac{(v2)^2}{(v1)^2} + \frac{(v3)^2}{(v1)^2} + \frac{(v4)^2}{(v1)^2}}$$

ในโปรเจคนี้วัดแค่ noise สามแท่งแรก

เนื่องจาก v2 มีค่าสูงสุดเทียบกับ v3 และ v4 สมมุติให้ v3 และ v4 เท่ากับ v2

$$THD \approx \sqrt{3\frac{(v2)^2}{(v1)^2}} = \sqrt{3(10^{(\frac{y2-y1}{10})})}$$ → คิดเป็นเปอร์เซ็นต์ $$THD \approx 100 * \sqrt{3 * (10^{(\frac{y2-y1}{10})})}$$

Evaluation Table

KMITL
พระจอมเกล้าลาดกระบัง

สัญญาณไซน์

	50 Hz	100 Hz	1 KHz	10 KHz	100 KHz	200 KHz
แรงดันเอาท์พุท						
(1-10 Vpp)	ผ่าน			ไม่ผ่าน		
แรงดันออฟเซท						
(<30 mv)	ผ่าน			ไม่ผ่าน		
Distortion (THD)						
(<5%)	ผ่าน			ไม่ผ่าน		

Experiment results in the report

KMITL
พระจอมเกล้าลาดกระบัง

- Sine Wave 10v_{PP} @200KHz, 100KHz, 10KHz, 1KHz, 100Hz, 50Hz
- Sine Wave 1v_{PP} @200KHz, 100KHz, 10KHz, 1KHz, 100Hz, 50Hz
- Sine Wave 10V_{PP} $\pm$5V offset @200KHz
- FFT result of sine Wave 10v_{PP} @ 200KHz, 100KHz, 10KHz, 1KHz, 100Hz, 50Hz