Analysis of PLL code

Manikanta Krishnamurthy

July 1, 2024

Abstract

In this report, I will be giving a brief introduction to PLL circuit and the contribution of each of its components. Additionally, I will provide an analysis of the PLL code that I have developed to reduce jitter time for the specified phase and frequency noise magnitudes.

1 Introduction

A typical PLL consists of three main components: a Phase Detector (PD), a Low-Pass Filter (LPF), and a Voltage-Controlled Oscillator (VCO). The Phase Detector compares the phase of the input reference signal with that of the VCO output, generating an error signal proportional to the phase difference. This error signal is then passed through the Low-Pass Filter to remove high-frequency components, resulting in a smoother control signal.
Finally, the filtered signal adjusts the VCO's frequency, reducing the phase error over time and locking the VCO output to the reference signal's phase and frequency.

By continually adjusting the VCO based on the reference signal, the PLL maintains synchronization, making it an essential tool for maintaining stable frequencies and reducing phase noise and jitter in electronic systems.

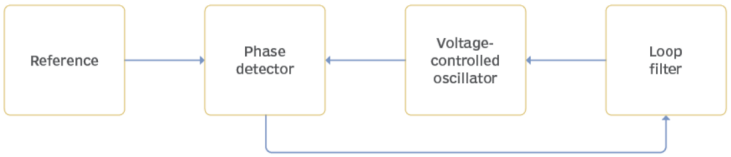


Figure 1: Basic model of a PLL.

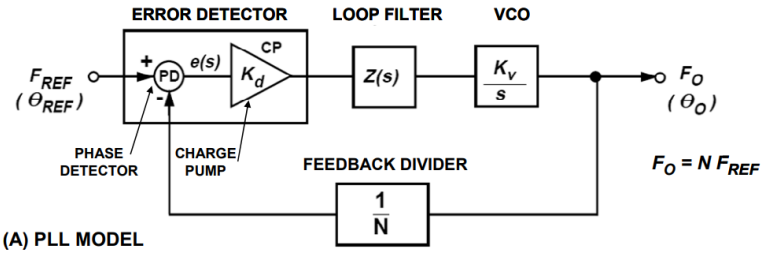


Figure 2: A more detailed model of a general PLL.

1.1 Phase Frequency Detector(PFD)

The PFD works by detecting the phase difference between two input signals, typically a reference signal and a feedback signal. The output signal is generated based on the phase difference, which is used to control the frequency of a voltage-controlled oscillator (VCO) in a PLL circuit.

1.2 Loop filter(LF)

The design of the PLL loop filter is crucial, as it affects the overall performance of the PLL. The filter should have a low-pass characteristic, with a cutoff frequency that is higher than the frequency of the input signal and lower than the frequency of the VCO. This ensures that the filter removes high-frequency noise and allows the PLL to track the input signal accurately.

1.3 Voltage controlled oscillator(VCO)

A Voltage-Controlled Oscillator (VCO) in a Phase-Locked Loop (PLL) generates a signal whose frequency varies with the input voltage. In a PLL, the VCO adjusts its frequency to match the input signal, ensuring synchronization and stable output. This makes VCOs essential for tasks like frequency synthesis and signal modulation in communication systems.

2 Code Analysis

The code given shows the implementation of an All Digital PLL , where all the components PFD,LF, and Oscillator are all based on digital circuits.The code exhi bits the following model,

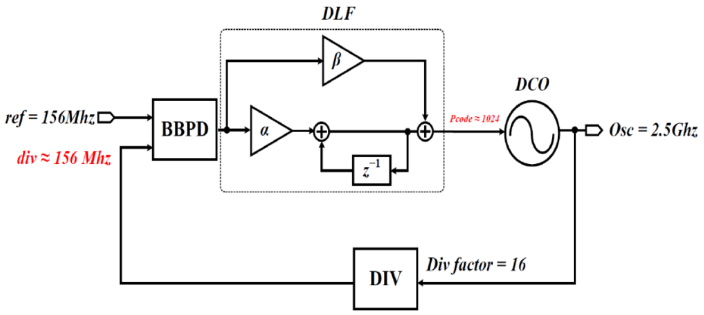


Figure 3: All Digital PLL.

2.1 BBPFD

The given code for this module exhibits behavior as the circuit given below:

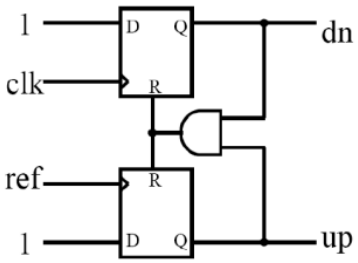


Figure 4: PFD circuit.

Here **ref** is the reference signal and **clk** is the feedback signal which comes after the frequency division.

bb_up signal: This signal indicates that the posedge of reference signal occurs before the feedback div signal.
bb_dn signal: This signal indicates that the posedge of reference signal occurs later than the feedback div signal.

Figure 5: $\mathrm{bb}_up, bb_dnsignals$.

The fact that both the signals are not periodic indicates that the frequencies don't match.

2.2 DLF

This module is majorly focused on calculating the dlf_out value , which is used to adjust the target time period.
bb_up Signal: When this signal is high, it means that the reference signal has achieved posedge before IDCO signal . This means we have to increase the frequency of IDCO signal in order for its posedge to occur earlier .The DLF increases the dlf_out value(above 1024) when signal this is high.
bb_dn Signal: When this signal is high, it means that the reference signal has achieved posedge later than IDCO signal . This means we have to decrease the frequency of IDCO signal in order for its posedge to occur later .The DLF decreases the dlf_out value(below 1024) when signal this is high.

The DLF has these two parameters Alpha , Beta .

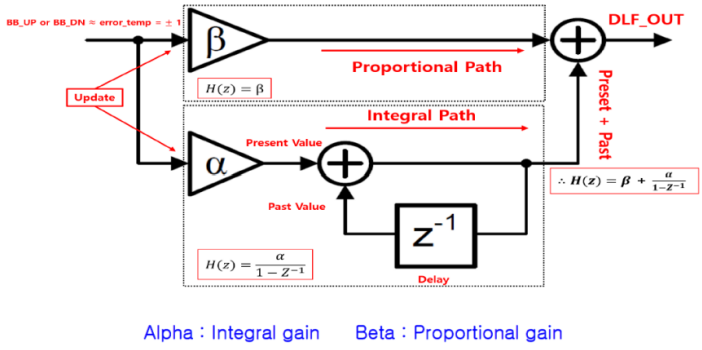


Figure 6: DLF model.

2.3 IDCO

This module primarily takes the dlf_out value and adjusts the time period of the output signals. The

```
real p_dco;
always @(code)  p_dco = p_tar - code*p_step + 1024*p_step;
```

Figure 7: Time Period adjusting.

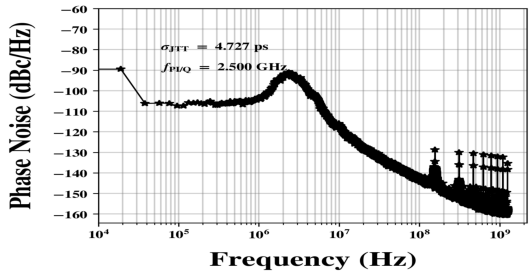
clock is updated based on these signals.

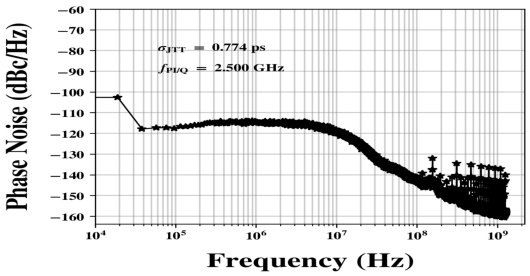
```
always @(clk_i[0] or pwdn) begin
        if(pwdn) begin
                clk_i[1] <= 1'b0;
                clk_i[3] <= 1'b1;
        end else begin
                        clk_i[1] <= #(p_dco/4+j_fm/2**0.5) clk_i[0];
                        clk_i[3] <= #(p_dco/4+j_fm/2**0.5) clk_i[2];
                end
end

always @(clk_i[1] or pwdn) begin
        if(pwdn) begin
                clk_i[0] <= 1'b0;
                clk_i[2] <= 1'b1;
        end else begin
                        clk_i[0] <= #(p_dco/4+j_fm/2**0.5) clk_i[3];
                        clk_i[2] <= #(p_dco/4+j_fm/2**0.5) clk_i[1];
                end
end
```

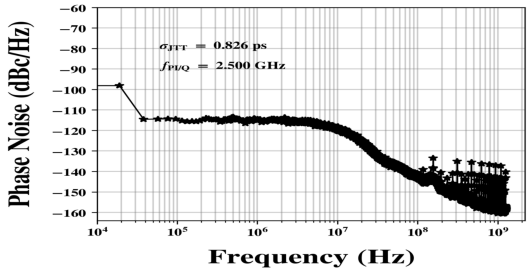
3 Jitter Time Optimization

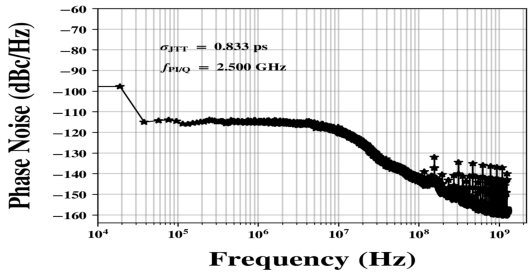
By changing the Beta and Tau values I tried to reduce the rms jitter value.

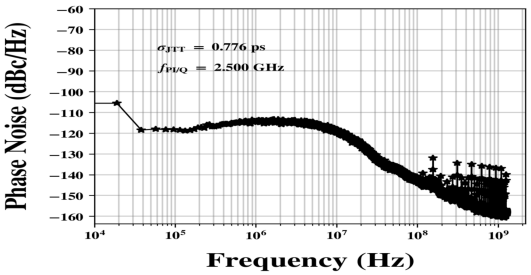
3.1 Some Simulations

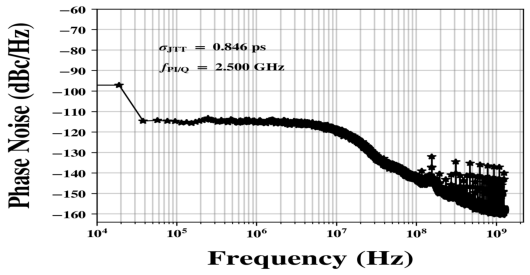












3.2 Data

First I changed the Beta values , Here we can notice that at Beta = 20 we get the least jitter.

Beta	**Jitter (ps)**
8	1.186
9	1.107
10	1.162
11	1.001
12	0.931
14	0.839
18	0.827
20	0.776
24	0.817
28	0.815
32	0.993
36	1.035

Table 1: Tau = 2×10^{10}

Then I changed Tau keeping Beta = 7. This tells me that the jitter is the least when Tau = 5×10^9.

Tau (Hz)	**Jitter (ps)**
2.5×10^8	1.284
5×10^8	1.244
7.5×10^8	1.259
1×10^9	19837.355
1.5×10^9	5.462
2×10^9	1.263
2.5×10^9	1.263
4×10^9	1.216
4.5×10^9	1.221
5×10^9	1.2
7.5×10^9	1.276
1×10^{10}	19837.355
5×10^{10}	1.231

Table 2: Beta = 7

Tau (Hz)	**Jitter (ps)**
1.5×10^9	4.727
2.5×10^9	0.774
3.5×10^9	0.846
5×10^9	0.833
2×10^{10}	0.776
5×10^{10}	0.826

Table 3: Beta = 20

If Beta and Tau had a linear relationship with jitter , then the values Beta = 20 and Tau = 5×10^9 should give me even lesser jitter.

Beta	**Jitter (ps)**
7	1.2
12	0.986
16	0.8
20	0.833
22	0.907
24	0.871
28	0.81
32	0.824
36	1.019

Table 4: Tau = 5×10^9

But the above data says other wise. Beta and Tau don't have a linear relationship with jitter. I tried some other simulations .

Tau (Hz)	**Jitter (ps)**
1×10^9	5.515
1.5×10^9	5.515
2×10^9	0.788
2.5×10^9	0.788
3×10^9	0.802
3.5×10^9	0.802
5×10^9	0.907
7.5×10^9	0.846
2.5×10^{10}	0.908
4×10^{10}	0.817
7.5×10^{10}	0.945

Table 5: Beta = 22

The least jitter time I was able to produce was 0.774 picoseconds. for Beta = 20 and Tau = 2.5×10^9