Lab 6

All Digital Phase-Locked Loop

指導教授 葉丙成
助教 朱峰森
Group 6
B92901035 柯浩賢
B92901038 粘 紘
B92901049 黃柏堯

Index

A. Introduction to ADPLL

This lab is about ADPLL. Our goal is to build up an All-Digital Phase-Locked Loop which can lock the central frequency of 100 kHz with the use of Verilog on an FPGA board.

I. Overall Structure

All-Digital Phase-Locked Loop (ADPLL) is commonly used in communication systems, especially for the applications of synchronization and frequency synthesis. It consists of three parts: (i) Phase Detector (PD) (ii) Loop Filter (LF) and (iii) Digital-Controlled Oscillator (DCO). Fig. 1 shows the general block diagram of an ADPLL.

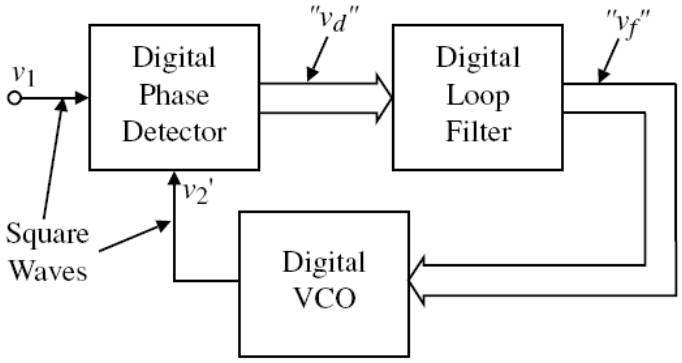


Fig. 1 General block diagram of an ADPLL

The task of a PLL is to lock the phase and the frequency of $U_1(t)$ to those of $U_2(t)$. The phase detector is used to detect the difference between $U_1(t)$ and $U_2(t)$. The loop filter is used to filter out out-of-band noise. Finally, the voltage-controlled oscillator (VCO) receives the output of the loop filter and adjusts the phase/frequency of the output signal $U_2(t)$ accordingly.

To realize an ADPLL, all function blocks of the system must be implemented by purely digital circuits. The signal are digital (binary) and may be a single digital signal or a combination of parallel digital signals. There are some advantages: No off-chip components and Insensitive to technology.

II. Phase Detector (PD)

The phase detector (PD) part of the ADPLL consists of an SR (JK) flip-flop. The block diagram of a JK flip-flop phase detector is shown in Fig. 2. The operation of the phase detector is illustrated near by. The difference between $U_1(t)$ and $U_2(t)$ will be detected by the phase detector. This phase detector counts the number of high frequency clock periods between the phase difference of U1(t) and U2(t).

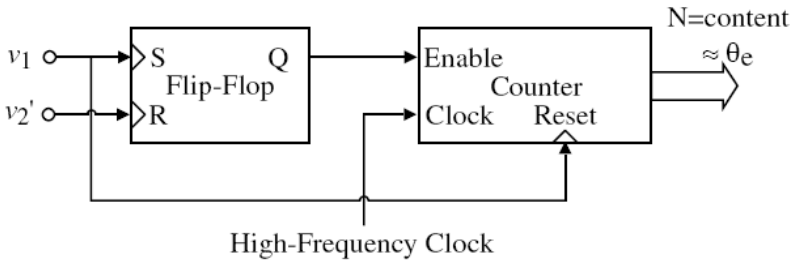


Fig. 2 Block diagram of a JK flip-flop phase detector

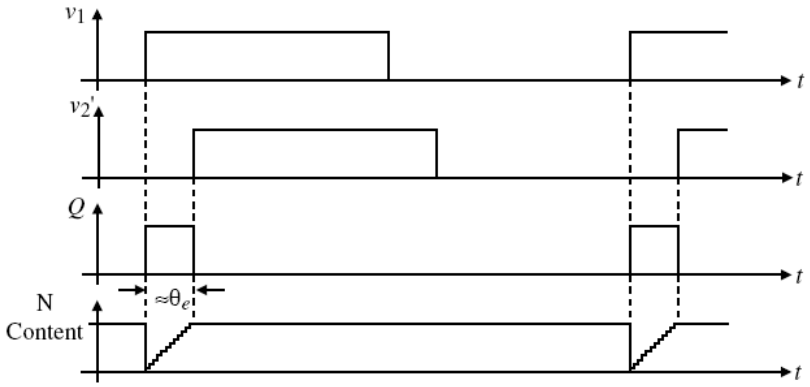


Fig. 3 Timing diagram of the phase detector

Fig. 3 is cut from P.E Allen's reference. For simplicity, v1 and v2' here are U1 and U2 respectively.

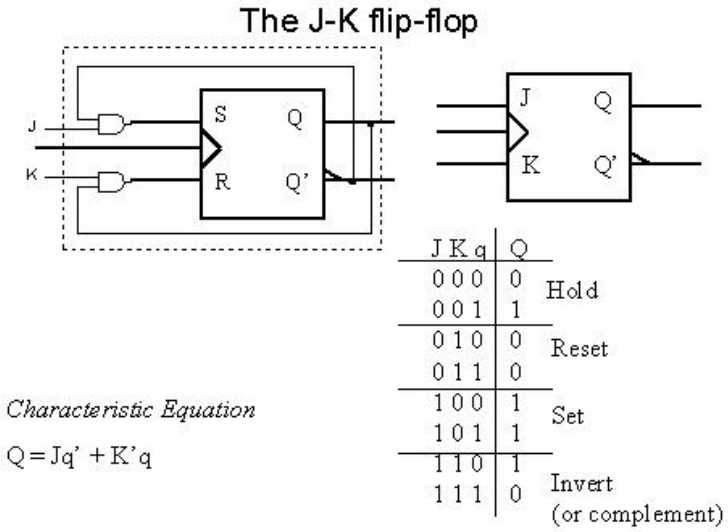


Fig. 4 Truth table of JK/SR transformation

The JK/SR is list as in Fig. 4 In the point of view in realization of digital circuit design, the waveform above can be seen as a circuit detecting the raising edge of $U_1(t)$ and $U_2(t)$. When there is a raising edge of $U_1(t)$ set the output Q (or say U_d in this lab handout) to 1. When there is a raising edge of $U_2(t)$, reset the output Q to 0.

In real time realization, the most used design of S-R or J-K is simply use "XOR" to fasten the computation and reduce the error or distortion in phase caused by clock

sampling.

III. Loop Filter (FL)

The loop filter in this lab is the K-counter. The block diagram of the loop filter is shown as in Fig. 5. The DN/UP signal in the block diagram is actually the Q signal of the phase detector last part. The associated waveforms of the loop filter at the steady state are shown in Fig. 6.

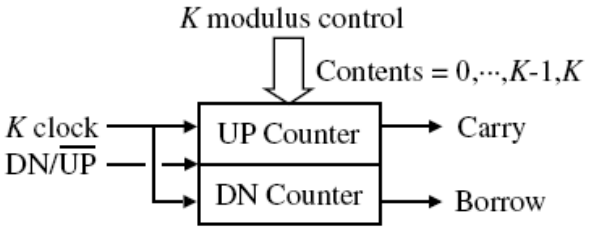


Fig. 5 Block diagram of the loop filter

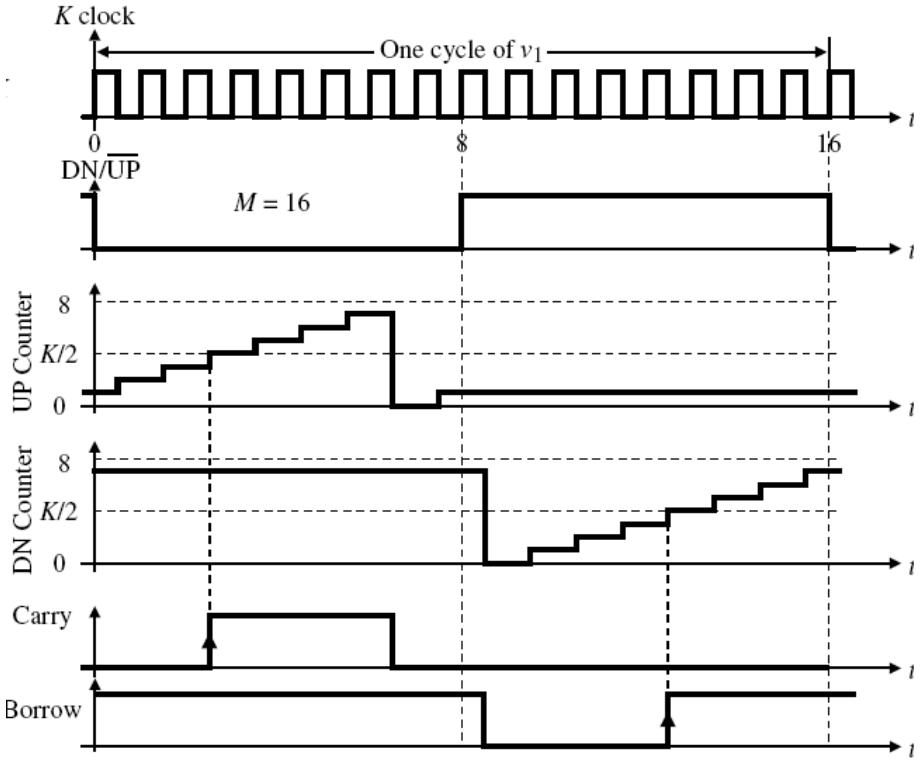


Fig. 6 Waveforms of the loop filter at steady state

The Loop Filter (LF) can be seen as a low pass filter. Since it is a integrator with transfer function H(s) = 1/sTi, where Ti is the time constant of integrator. There are two counters in LF, Both counters count upwards. Carry equals 1 when contents of the UP counter>=K/2 and Borrow equals 1 when contents of the Down counter>=K/2. Finally, positive going edges of the Carry and Borrow control the DCO, the carry and borrow waveforms are then fed to the digital controlled oscillator to adjust the frequency and the phase of $U_2(t)$.

IV. Digital Control Oscillator (DCO)

The digital version of voltage-controlled oscillator, i.e. digitally-control oscillator (DCO) is increment-decrement counter (ID-counter). The overview of DCO block diagram of the counter is shown in Fig. 7.

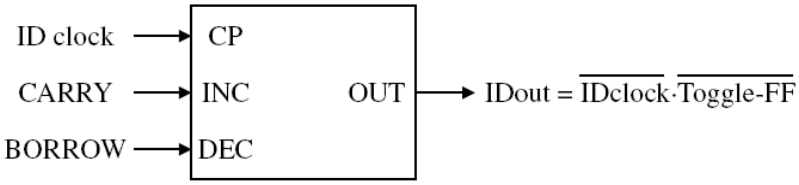
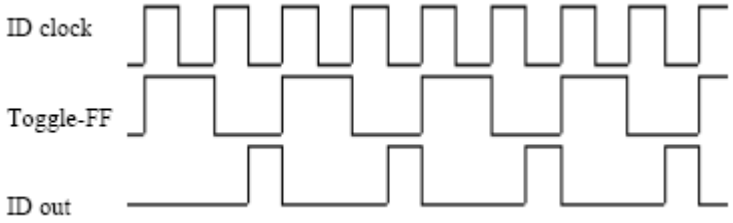


Fig. 7 Block diagram of DCO

The carry and borrow are from the K-counter loop filter and the ID-counter is sensitive to positive edges. A Toggle-FF signal is generated following to the following rules:

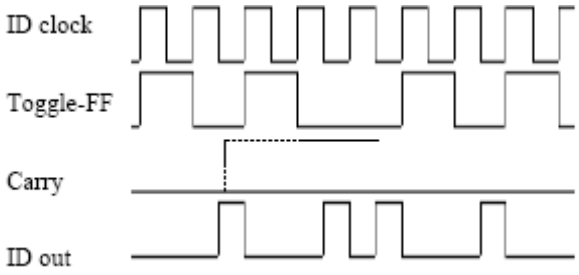
1) No BORROW or CARRY pulses

The toggle-FF switches on every positive edge of the ID clock if no CARRY or BORROW pulses are present.



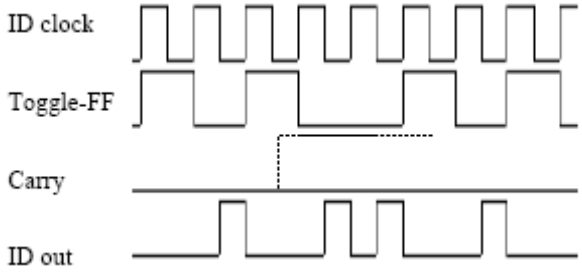
2) CARRY input applied when the toggle-FF is in the low state

When the toggle-FF goes high on the next positive edge of the ID clock but stays low for the next two clock intervals, the IDout is advanced by one ID clock period.



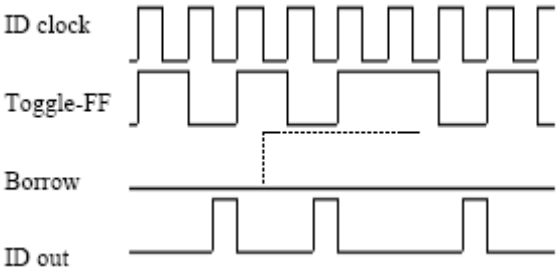
3) CARRY input applied when the toggle-FF is in the high state

The toggle-FF is set to low for the next two clock intervals. Because the CARRY can only be processed when the toggle-FF is in the high state, the maximum frequency of the IDout signal is reached when the toggle-FF follows the pattern of “high-low-low-high-low-low”. Therefore, the maximum IDout frequency = 2/3 ID clock frequency. This will limit the hold range of the ADPLL.



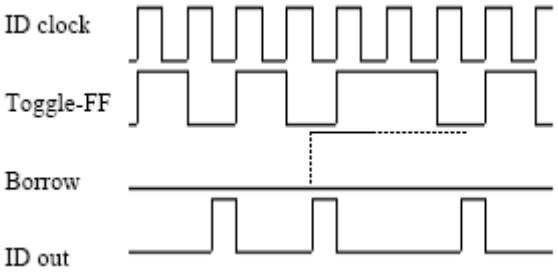
4) BORROW input applied when the toggle-FF is in the high

A BORROW pulse causes the toggle-FF to be set high on the succeeding two positive edges of the ID clock. This causes the next IDout pulse to be delayed by one ID clock period. The toggle-FF has the pattern of “low-high-high-low-high-high” which gives the min. IDout frequency = 1/3 ID clock frequency. Basically, 1 CARRY pulse adds 1/2 cycle and 1 BORROW pulse removes 1/2 cycle.



5) state BORROW input applied when the toggle-FF is in the low state

A BORROW pulse causes the toggle-FF to be set high on the succeeding two positive edges of the ID clock. This causes the next IDout pulse to be delayed by one ID clock period. The toggle-FF has the pattern of "high-high-low-high-low-high" which gives the min. IDout frequency = 1/3 ID clock frequency. Basically, 1 CARRY pulse adds 1/2 cycle and 1 BORROW pulse removes 1/2 cycle.



The output signal of the ID-counter is then $ID_{out} = ID_{clk} \cdot$ Toggle-FF. A general case waveform is shown in Fig. 8 for reference.

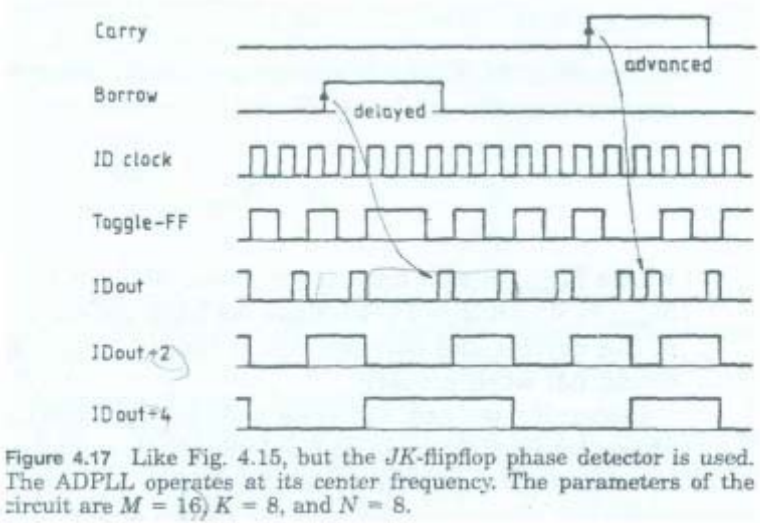


Figure 4.17 Like Fig. 4.15, but the *JK*-flipflop phase detector is used. The ADPLL operates at its center frequency. The parameters of the circuit are $M = 16$, $K = 8$, and $N = 8$.

Fig. 8 Waveform of the DCO

B. Original ADPLL

I. Block Diagram

As Section A discussed, an ADPLL consists of three parts: (i) Phase Detector (PD) (ii) Loop Filter (LF) and (iii) Digital Control Oscillator (DCO). In this lab, our goal is to achieve ADPLL on FPGA using Verilog code compiled by Quartus II. In order to fit the lab requirement and hardware design requirements (e.g. timing and synchronization), we made a few changes in our Verilog design which will be discussed further. Fig. 10 is the overall structure of our ADPLL in Verilog design.

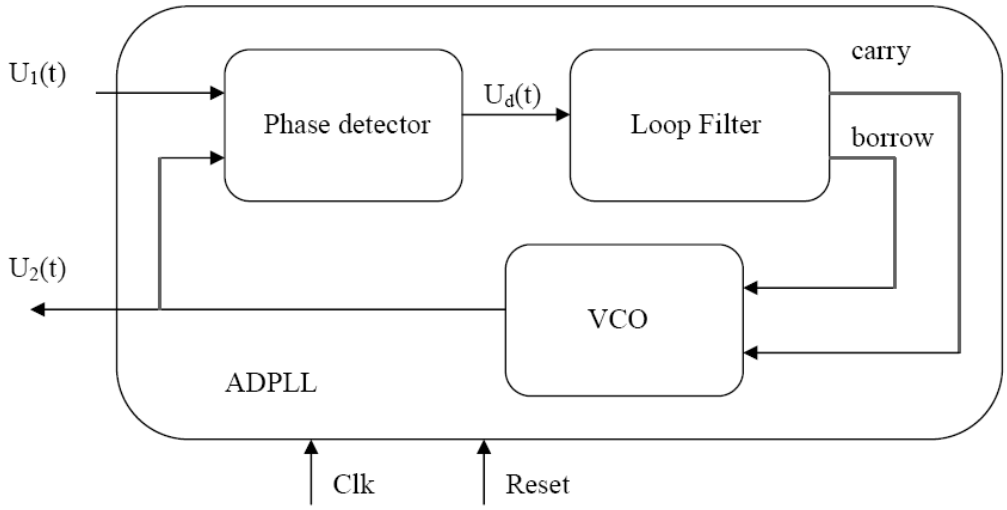


Fig. 10 Overall structure of our ADPLL

Two signals (origin and locked one) enter the phase detector which detect the difference in phase of the two signals. Then the signal is enter the loop filer as a low pass. Then the result enters the VCO to change phase/frequency. Finally, it feed back to the phase detector to complete the signal circle. If all components work fine, two signals will be more closely in phase and frequency after one signal loop.

$U_1(t)$ is the input signal (square wave)with frequency 100kHz, and $U_2(t)$ is our phase-locked result. The internal connection is shown above. $U_d(t)$ is the output of phase detector (its structure is similar to J-K flip-flop). Carry and borrow is the counted result of the loop filter (K-counter). The final phase-locked result is at the output of the voltage control Oscillator. The overall I/O is listed as below.

Notation	Description	Pin assignment	I/O
$U_1(t)$	Original signal	PIN_D13	Input
$U_2(t)$	Phase-locked signal	PIN_F23	Output
clk	Module clock	PIN_D13	Input
reset	Reset signal	PIN_N25	Input

II. Hardware Implementation Guideline

This part is to set all the experiment devices and standardize the process.

Fig. 11 All devices we will need in real implementation

There are 5 components in our hardware implementation of ADPLL.

1) FPGA

   Connecting FPGA GPIO2 pin4 for signal input $U_1(t)$, pin6 for signal output $U_2(t)$ and pin12 for ground. Note no lines or nodes are short.

2) PC

   Using Quartus II 7.0 to compile *.sof and using USB-blaster to load the program into FPGA.

3) Board

   Connecting 330ohm resisters in prevent for short and too many current, and wisely connecting all the lines.

4) Display

   Auto adjusted display that makes sure all probe is well functioned and properly grounded.

5) Signal generator

   Setting frequency to 100kHz and adjusting amplitude to Vpp=3.3V with a offset -075 to make V_{max} is about 3~3.3V and V_{min} is about 0V for FPGA to work properly.

III. Experimental Results of the Original ADPLL

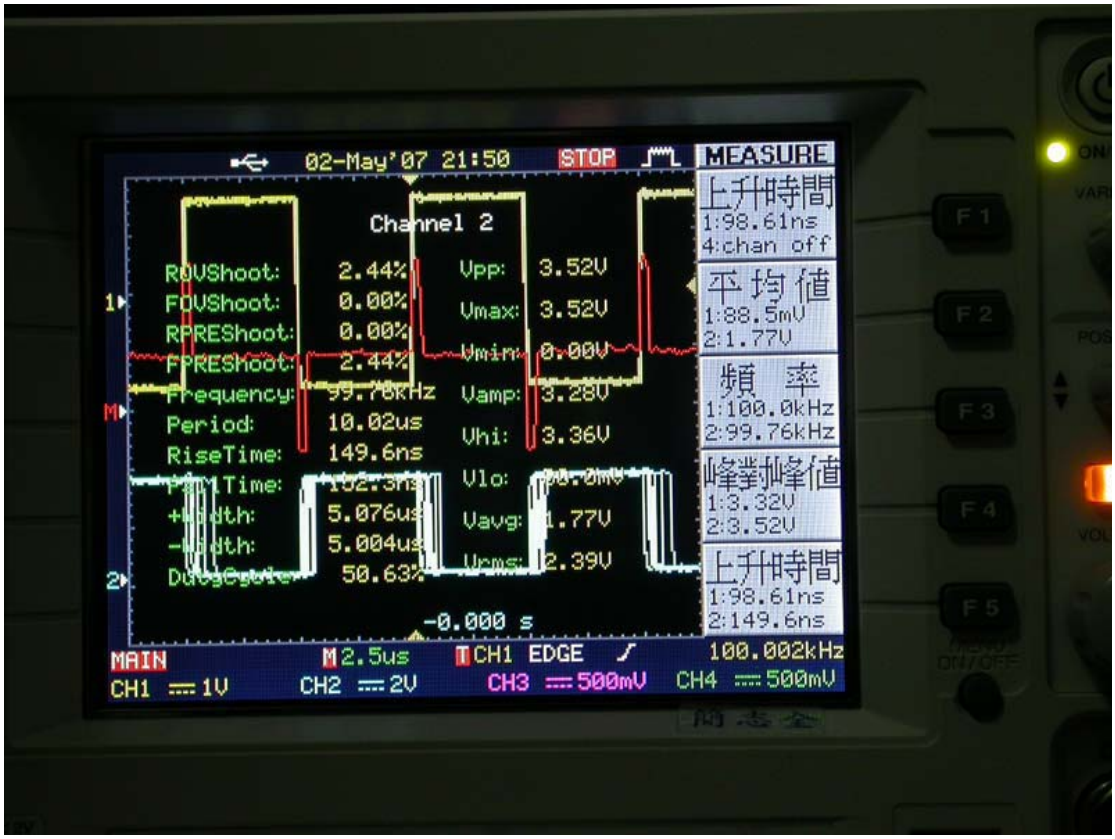


Fig. 12 A typical oscilloscope output for ADPLL

Fig. 12 is a typical oscilloscope output for the original ADPLL, here the yellow waveform is input and the blue one is the ADPLL output. We can see the PLL indeed locks the phase. There are some blurs in the cutting edge, and this is due to non-perfect locking ability of our ADPLL. This problem can be eliminated and will be discussed after.

K	K clk (X)	ID clk (X)	Lower Bound (kHz)	Upper Bound (kHz)	Bandwidth (kHz)
8	1	1	91.2	108.1	16.9
8	1	2	95.3	104.1	8.8
8	1	4	97.3	102.5	5.2
8	2	1	86.5	116.2	29.7
8	4	1	86.5	116.4	29.9
4	1	1	86.8	114.1	27.3
16	1	1	97	104.2	7.2

Table 1 Experiment result of ADPLL with different parameters

Table 1 showed our experimental results of the original ADPLL. The nominal frequency of the input signal is fu1 = 100kHz. We tried to deviate fu1 from 100kHz and find the lock range of the original ADPLL. If we defined ADPLL bandwidth as

"upper bound - lower bound." Row 1 with K=8, K clock and ID clock remaining unchanged is our original parameters. In our original parameter settings we have a bandwidth approximately 17% of the nominal frequency.

As we make ID clock faster, we trade bandwidth for a better locking ability so that the bandwidth deteriorate. If we make K clock larger and others remain unchanged, the bandwidth would have a large improvement but there is a larger oscillation about the locked phase. For the effect of K counter, the larger the K, the poorer the bandwidth of ADPLL, but a tradeoff between bandwidth and locking ability still exists.

C. Full Lock-Range ADPLL

In this section, we are going to discuss the factors that influence lock range of the original ADPLL. According to these discussions, we proposed a simple and effective control algorithm that extends lock range of the proposed full lock-range ADPLL to the theoretical bound.

I. Phase Jitter

Short-term frequency instabilities, seen in the time domain as jitter, can cause problems in both analog and digital signals. As system operating frequencies have increased, these instabilities have gained increasing importance, because their relative size to the total period length is larger. The instabilities can eventually cause slips or missed signals that result in loss of data. Fig. 13 shows a square wave with jitter compared to an ideal signal at the same long-term frequency.

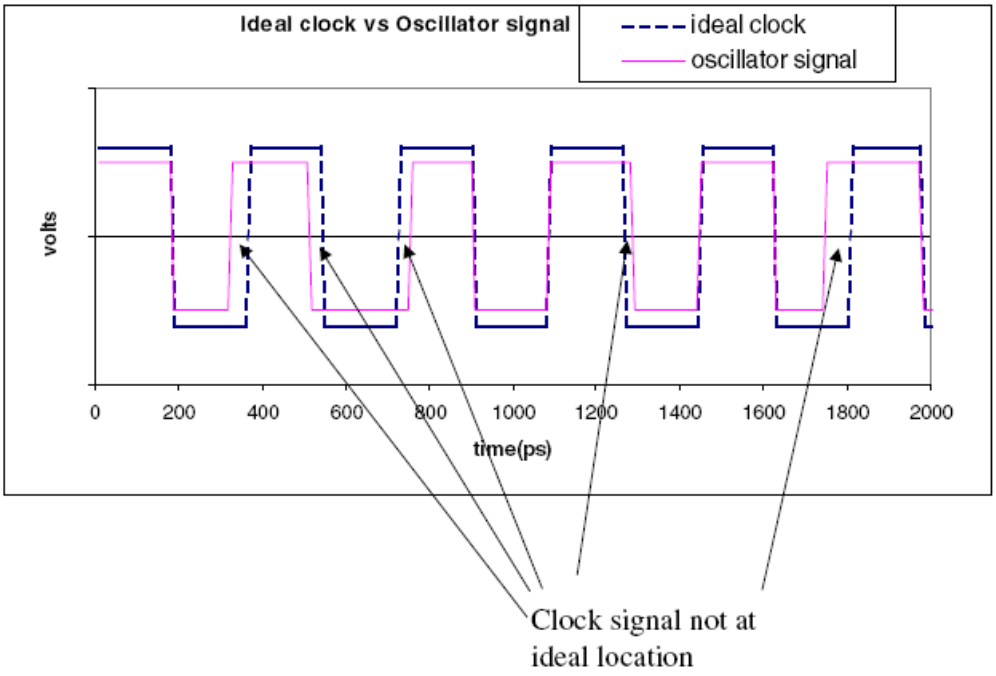


Fig. 13 The clock signal with phase jitter

As what we can see in the above figure, the short-term frequency varies while the long-term frequency remains. Besides, the duty cycle varies from period to period. From this point of view, if we want to analyze the cause of phase jitter in ADPLL, we could take a look at the duty cycle range first.

II. Duty Cycle Variation

To analyze the duty cycle varying range when the loop is locked, we have considered two kinds of phase detector – XOR phase detector and JKFF phase detector.

1) XOR phase detector

Assume that the loop has already been in lock, both counters count on negative edges of the K counter, the toggle flip-flop within the ID counter toggles on the positive edge of ID clock, and all flip-flop of the $\div N$ counter count on the negative edge of the corresponding clock signal. Because the loop is in lock, the number of CARRY rising edges will be identical to the number of Borrow rising edges and the phase difference between input signal and output signal will be $\pi/2$ due to the evenly distributed DN/UP signal (the clock-like signal with duty cycle about 50%).

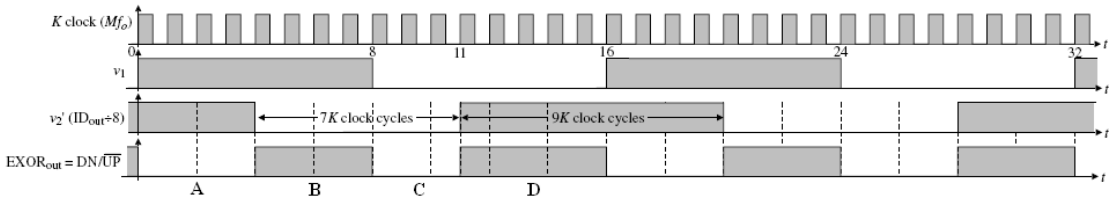


Fig. 14 The DN/UP signal in locked loop with XOR PD

Fig. 14 shows the snapshot of the waveform, where v_1 is the input signal and v_2 is the output locked signal. Because of the in-lock assumption and the according $\pi/2$ phase difference, there is at least one pair of {low, high} waveform in {{A, B}, {C, D}} with 50% duty cycle. Without loss of generality, we can assume that A and B have the same duration. Since CARRY/BORROW could only changed when DN/UP was low/high and the frequency of CARRY/BORROW rising edge in the same interval would be Mf_0/K, the number of CARRY rising edges and BORROW rising edges would be at most different by 1.

Fig. 15 Number of points along the segment

Hence, the number of CARRY and BORROW rising edges in A and B interval would be at most different by 1. In other words, the resultant half cycle interval of v_2' will be different by

$$1/(2Nf_0).$$

The resultant duty cycle would be

$$\frac{1/(2f_0)\pm 1/(2Nf_0)}{1/f_0}=\frac{1}{2}(1\pm\frac{1}{N}).$$

Therefore,

$$0.5\cdot(1-1/N)\le\delta\le 0.5\cdot(1+1/N).$$

This is independent of K and M. To minimize the phase jitter, we can choose a larger N.

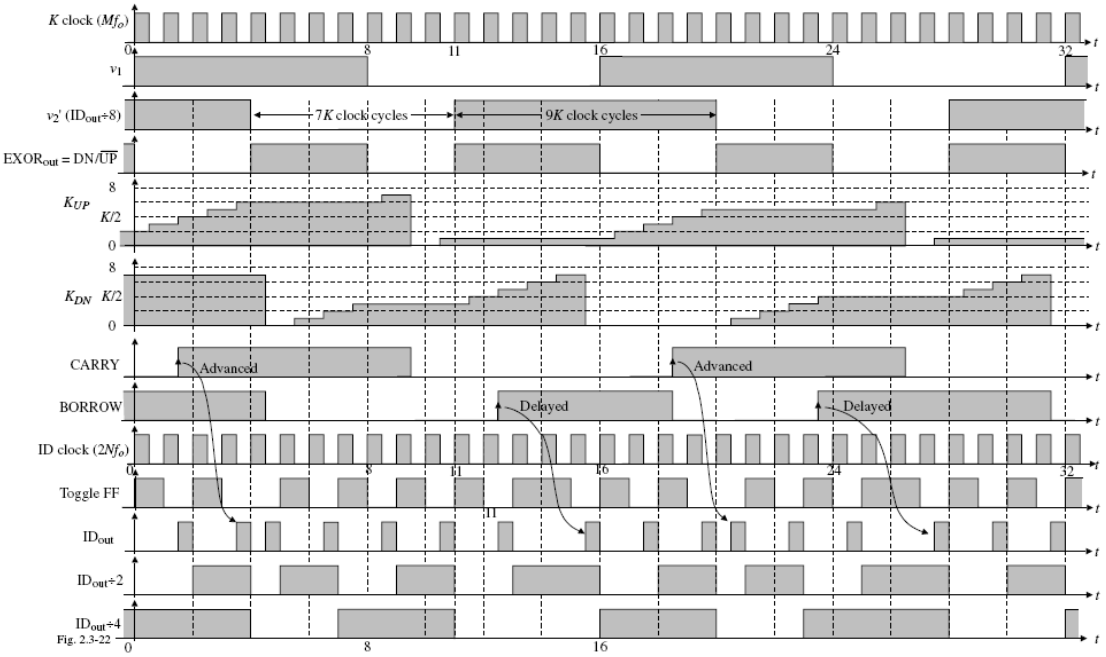


Fig. 16 Waveforms of ADPLL using XOR PD, M=16, K=8, N=8

Besides, when $K=M/4$, there is at least one of CARRY/BORROW signals with more than $(Mf_0/K)\cdot(1/(2f_0))-1=1$ rising edge; therefore, two CARRY signals and two BORROW signals. It can be further showed that no phase jitter would show in this case. When $K>M/4$, the average number of CARRY/BORROW is less than 1, the phase jitter will be more serious because of the short-term frequency error.

2) JKFF phase detector

Followed by the same assumption described in (1), the phase difference of input and output will be π. The DN/UP signal will be act as the following figure.

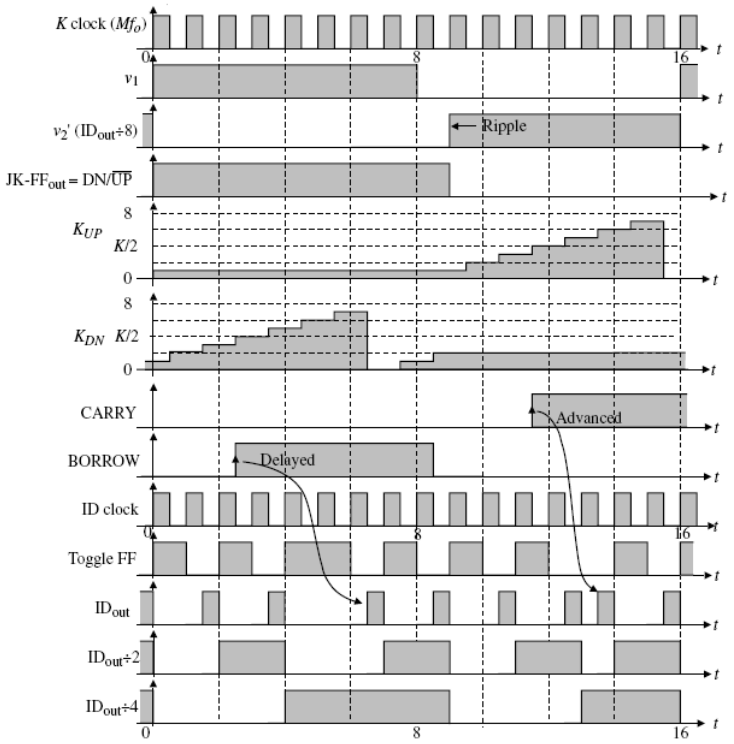


Fig. 17 Waveforms of ADPLL using JKFF PD, M=16, K=8, N=8

There is one of interval last more than half of input period. Therefore, there are $(Mf_0/K)\cdot(1/(2f_0)) = M/2K$ CARRY and $M/2K$ BORROW. Because of JKFF PD, phase jitter will exist regardless of the value of parameters. The corresponding duty cycle is

$$\frac{1/(2f_0) \pm (M/2K)\cdot(1/2Nf_0)}{1/f_0} = \frac{1}{2}\cdot(1 \pm \frac{M}{2KN})$$

Therefore,

$$0.5\cdot(1-\frac{M}{2KN}) \le \delta \le 0.5\cdot(1+\frac{M}{2KN})$$

To minimize phase jitter, we can choose $K = M/2$.

III. Hold range

The maximum output frequency occurs when the K counter is counting up and is $f_{\max} = Mf_0/K$. Because each carry applied to the ID counter causes 1/2 cycle to be added to the IDout signal, the output frequency of the IDout increases by

$$\Delta f_{IDout} = \frac{Mf_0}{2K}$$

Therefore, the frequency of output signal v_2' will be

$$f' = f_0 \pm \Delta f_{IDout}/N = f_0(1 \pm \frac{M}{2KN})$$

Besides, the frequency can be never higher than $2f_0$ according to the procedure to form IDout signal.

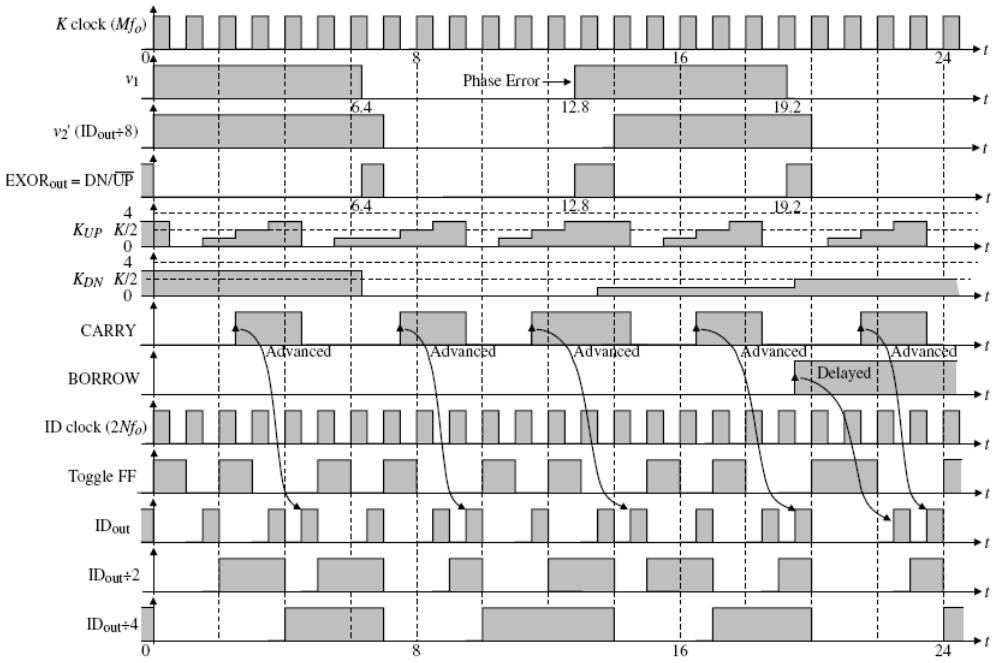


Fig. 18 $f_1 = 1.25 f_0$, XOR PD, M=16, K=4, N=8

IV. Proposed Full Lock-Range ADPLL

According to the discussion above, we can draw some conclusions:

1) Larger M value leads to larger hold range
2) Smaller K value leads to larger hold range
3) Smaller N value leads to larger hold range
4) In XOR FD, larger N value leads to smaller jitter regardless the value of M and K
5) In JKFF FD, smaller M value, larger K value, and larger N value leads to smaller jitter
6) Hold range and jitter is a trade-off among all the parameters
7) XOR FD is much cost effective and releases the conflict of M/K value between larger hold range and small jitter
8) It is better to use the same clock for K counter and ID counter because of the synchronization problem

We first analyze the original design of ADPLL on the lecture note. The parameter is: M = 16, K = 8, N = 8, JKFF phase detector. According to the theoretical analysis above, the hold range is $(1-1/8) f_0 \sim (1+1/8) f_0$, i.e. $0.875 f_0 \sim 1.125 f_0$ and the duty cycle is $0.5(1-1/8) \sim 0.5(1+1/8)$, i.e. 0.4375~0.5625. However, the experiment result shows a smaller hold range. This may be caused by the state transition DCO, which can not deal with the adjacent CARRY → BORROW signal because of the insensibility of state machine in different states. This deficiency may reduce the balance ability of DCO because it may ignore the following signal which may cancel the previous advance or delay effect to make the output frequency stable. Therefore,

when the input frequency approaches the hold boundaries, DCO may run too fast or too slow and then be out of lock.

To deal with this problem, we have rewritten the DCO code. Instead of finite state machine, we use two flags to mark the delay low/high signals and apply the cancel policy on the "CARRY → BORROW" or "BORROW → CARRY" signal. In addition, we change the JKFF PD by XOR PD to release the conflict of M/K value and raise the N value to reduce the phase jitter. At the same time, to cancel the large N value effect on hold range, we raise the M value and lower the K value. The parameter of this new ADPLL is: M =256, K = 4, N = 128, XOR phase detector. The consequent hold range and duty cycle are $(1-1/4)f_0 \sim (1+1/4)f_0$, i.e. $0.75f_0 \sim 1.25f_0$ and the duty cycle is $0.5(1-1/128) \sim 0.5(1+1/128)$, i.e. 0.4961~0.5039. The experiment result shows the high coherence between the theoretical analysis and practical implementation. (Hold range: $0.76f_0 \sim 1.24f_0$)

In order to further improve the hold range of ADPLL, we have tried to change the parameter of ADPLL. However, when we increase the M value or lower the K value, although the hold range extend, it does not increase accordingly. To interpret this phenomenon, we can see that if the ID clock frequency is too low or the density of CARRY or BORROW is too high (K value is too small or K counter frequency is too high), the ID counter is unable to process all the CARRY and BORROW signals. The redundant CARRY or BORROW will be ignored as the original state transition DCO. Consequently, the hold range cannot extend to the boundaries, and here comes a question, "What is the minimum N value that can process all the CARRY and BORROW signals?" Because we have apply the cancellation policy of adjacent C→B and B→C signals, the sequence contain both CARRY and BORROW may not approach the limit of ID counter clock. Hence, we can focus on the sequence with only CARRY or BORROW. Take the CARRY sequence as an example, if a number of CARRY signals have to be processed in succession by the ID counter, the delay between any two CARRY signals, which is K/Mf_0, should be larger than 3 times of ID counter periods, i.e. $3 \cdot 1/2Nf_0$ since the maximum frequency of IDout is 2/3 of ID counter under this DCO. Therefore,

$$\frac{K}{Mf_0} \geq \frac{3}{2Nf_0} \Rightarrow N \geq \frac{3M}{2K}$$

Besides, since M, K, and N are mostly integer powers of 2, the practical minimum is

$$N_{practical} = \frac{2M}{K}$$

According to the discussion, we can find that the N value of the second ADPLL implementation has already been the minimum of N (128 = 2*256/4). Once the value of M has been raised or the value of K has been lowered, the ID counter clock will be

never catch up with the CARRY signals. Fig. 19 shows the relation of ID clock and CARRY density.

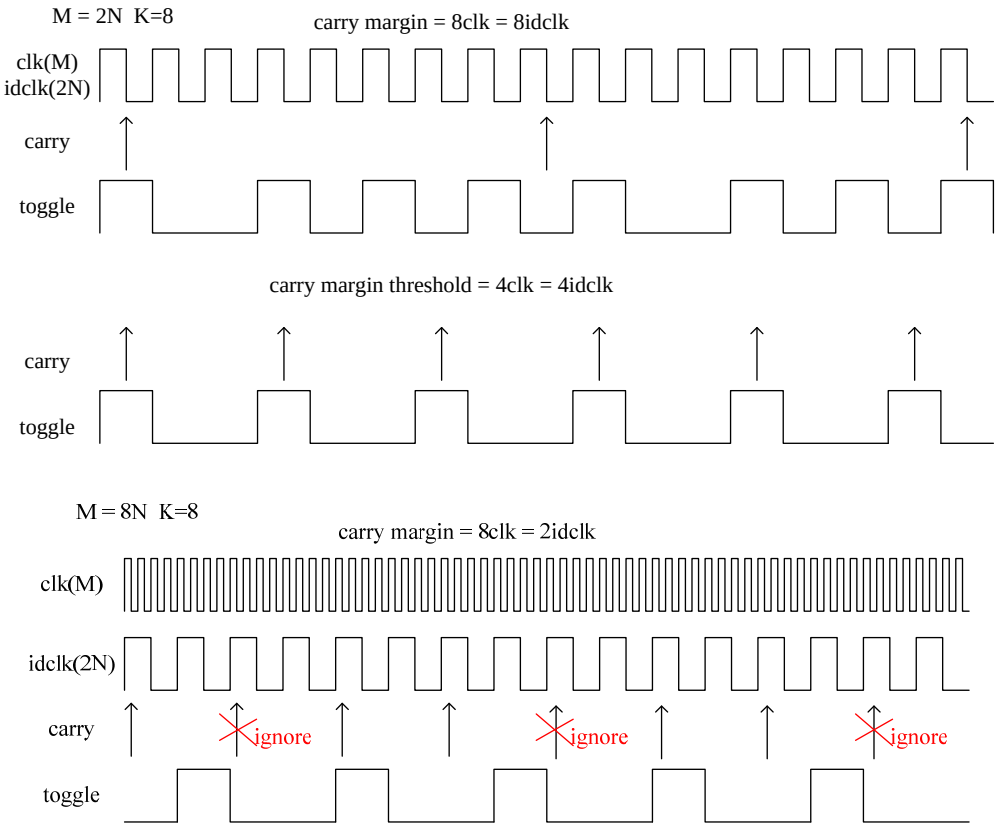


Fig. 19 Relation between ID clock and CARRY density

Let's retrace the previous analysis. The reason why N value must be larger than 2M/K is that under this DCO, the maximum frequency of IDout is at most 2/3 of ID clock. If we could raise the maximum frequency of IDout, we could then further improve the process efficiency. The limitation of IDout frequency is caused by the "high-low-low-high-low-low" pattern of toggle, so why not changes another pattern that could lead to a larger range. For example, the "high-low-low-low-high-low-low-low" pattern can raise the maximum frequency up to $(3+3)/(4+4)=3/4$ of ID counter frequency, so the ID counter clock can handle the CARRY sequence when M = 8N, K = 8. ($K/Mf_0 \geq 4 \cdot 1/Nf_0 \Rightarrow N \geq 4M/K$)

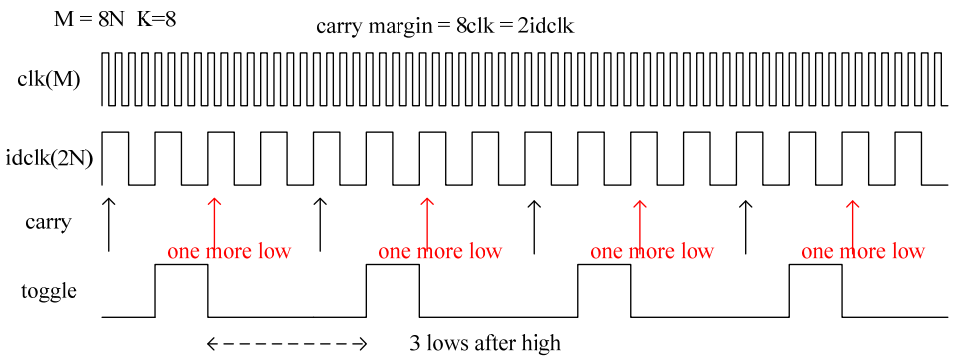


Fig. 20 Extended toggle pattern

Using this special pattern, the hold range can then extend to the theoretical boundaries, i.e. $0.5 f_0 \sim 1.5 f_0$.

If the method really work, how about trying to approach the design limit? To approach the design limit, the IDout frequency must approach the ID counter frequency and the corresponding pattern will be “low-low-…-low”, i.e. there can be arbitrary number of lows after high. From a different angle, one more CARRY stands for one more low in toggle. The magic pattern that can handle the n-CARRY sequence is the one which has n lows more than highs. Therefore, to design an optimal ADPLL, we must generate the “magic pattern” right when we receive a CARRY, i.e. we have to make sure that the toggle satisfies the “n-more” law all the time, and we have proposed the algorithm to generate toggle signal.

```
delay_one, delay_zero are two 2-bit flag
always @ (posedge reset or posedge clk)
    if (reset)
        reset toggle, delay_one, delay_zero to zero
    else
        if (only CARRY rising)
            if (toggle is high, i.e. toggle should fall if no “delay_one”)
                if (delay_one is equal to 2)
                    one delay_one has been cancelled by rising CARRY
                    therefore, there seems to be only one delay_one and no
                    CARRY rising
                    the other delay_one will be deflag because of the
                    passing of time
                    => delay_one is assigned to be 0 and toggle remains
                        unchanged because of the deflagged delay_one
                else if (delay_one is equal to 1)
                    one delay_one has been cancelled by rising CARRY
                    therefore, there seems to be only no delay_one and
```

CARRY rising
=> delay_one is assigned to be 0 and toggle is flipped as usual
else
there is no delay_one to cancel the rising CARRY
there is no delay_zero to delay the flip of toggle
=> flip toggle and add delay_zero by 1
else if (toggle is low, i.e. toggle should rise if no “delay_zero”)
if (delay_zero is equals to 2)
toggle remain unchanged by deflag one of “delay_zero”
the rising CARRY flag a new “delay_zero”
=> delay_zero and toggle remain unchanged
else if (delay_zero is equal to 1)
the rising CARRY will be treated as the second “delay_zero”
=> delay_zero is set to be 2 and toggle remain unchanged
else
the rising CARRY will be treated as the first “delay_zero”
the delay action will be start at the next low
=> delay_zero is set to be 1 and flip the toggle
else if (only BORROW rising)
the procedure is symmetric to the above case
else
if (toggle is high, i.e. toggle should fall if no “delay_one”)
if (delay_one is more than 0, i.e. high must be delay)
=> remain toggle unchanged and subtract 1 from delay_one
else
=> flip toggle
else begin
if (delay_zero is more than 0, i.e. low must be delay)
=> remain toggle unchanged and subtract 1 from delay_zero
else
=> flip toggle

V. Simulation Results

To verify the algorithm, we can see the waveforms in simulation. (test1: CARRY, test2: BORROW, test3: IDout, test4: toggle)

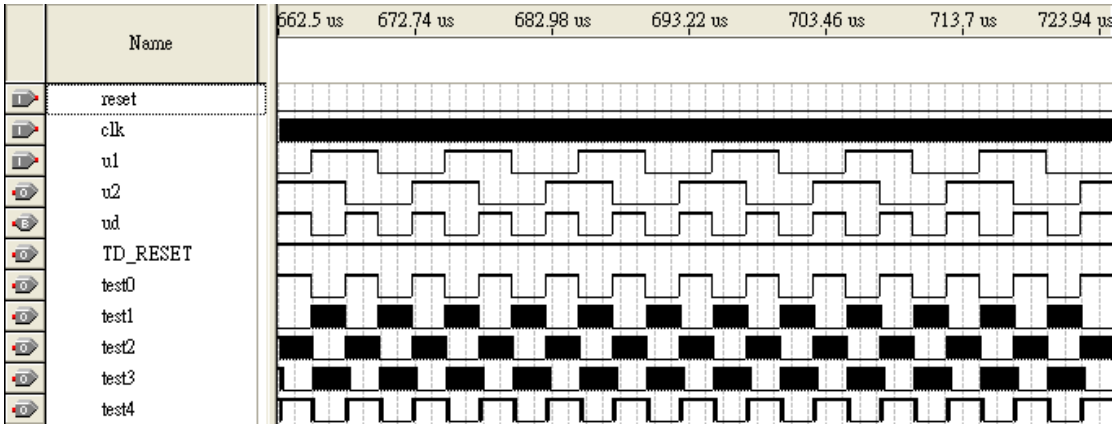


Fig. 21 Simulation result for $f_1 = f_0$, M=256, N=128, K=1

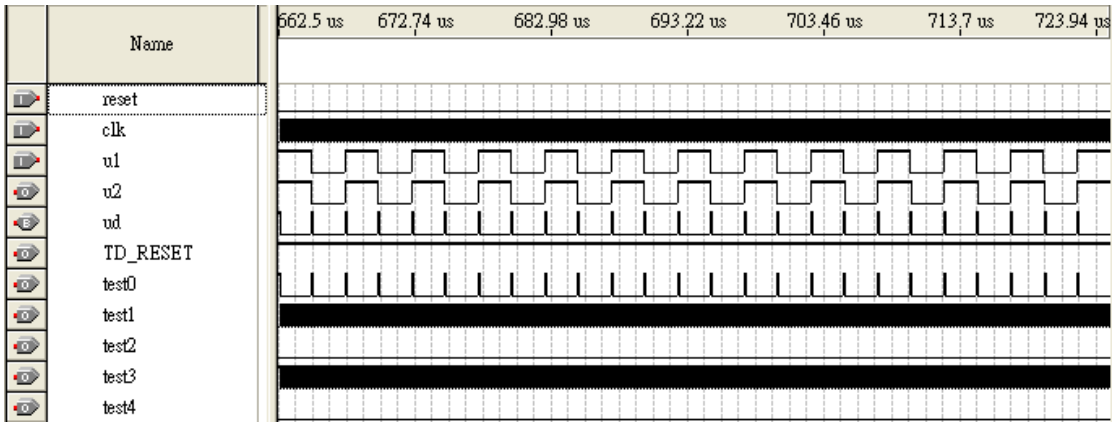


Fig. 22 Simulation result for $f_1 = 2 f_0$, M=256, N=128, K=1

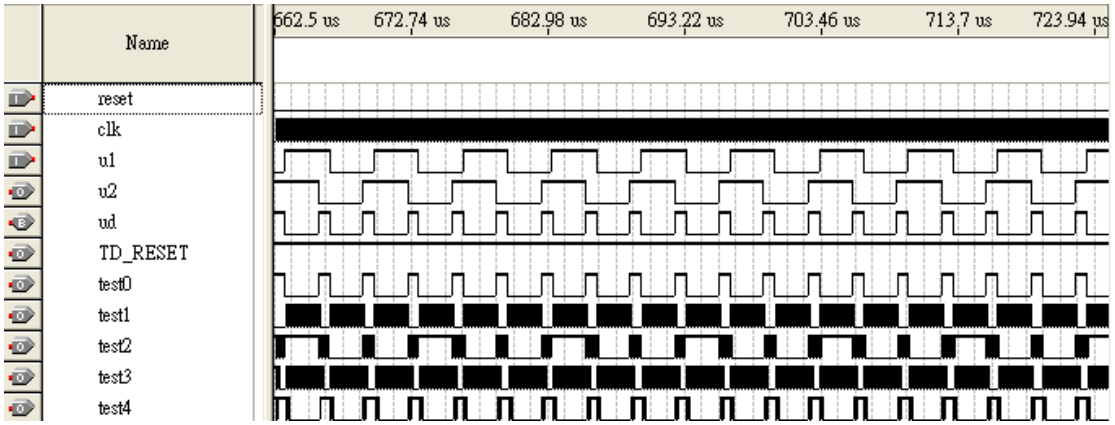


Fig. 23 Simulation result for $f_1 = 1.5 f_0$, M=256, N=128, K=1

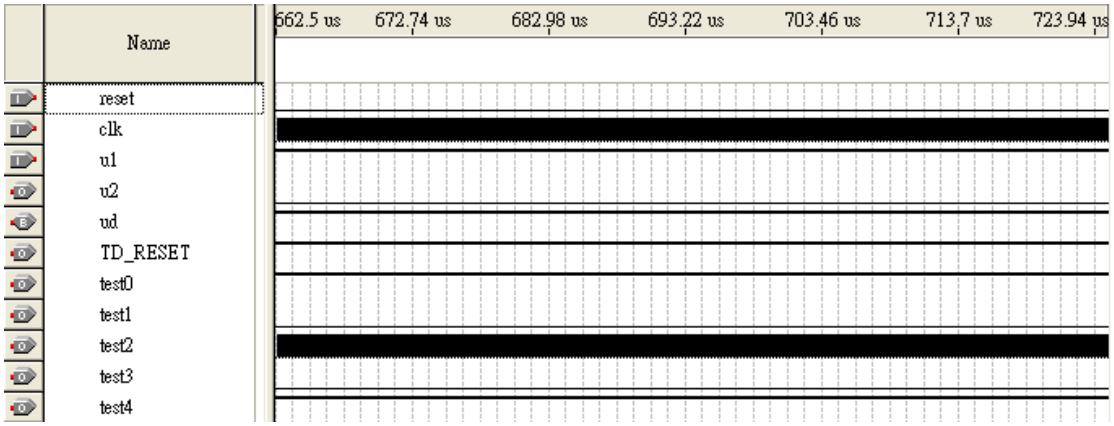


Fig. 24 Simulation result for $f_1 = 0$, M=256, N=128, K=1

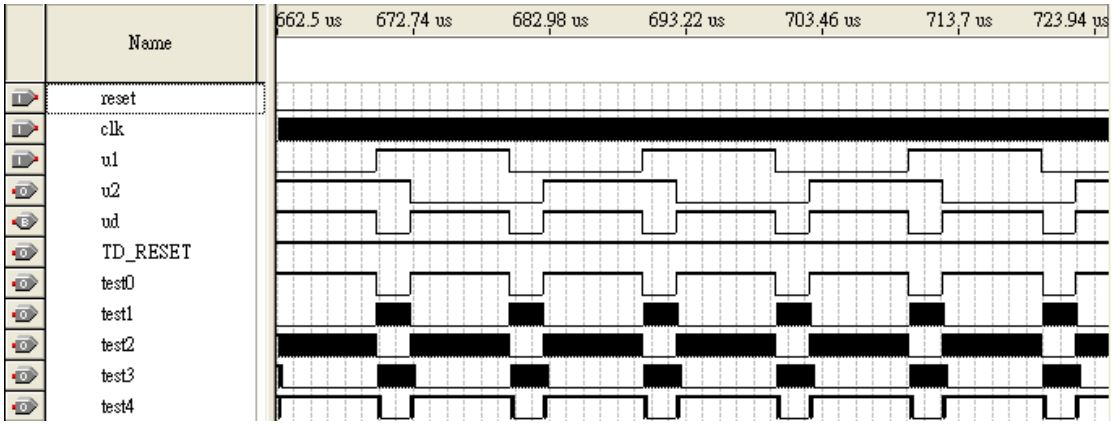


Fig. 25 Simulation result for $f_1 = 0.5 f_0$, M=256, N=128, K=1

D. Experimental Results

In this experiment, we use an XOR gate as a phase detector and an ID-counter as a digital-control oscillator. The center frequency (f_0) is 100 kHz. The important parameters using in this lab is listed below.

	Exp6_original	Exp6_new	Exp6_improved
FD	JK-FF	XOR	XOR
DCO	ID counter	ID counter	Modified ID counter
K	8	4	1
M	16 (Kclk = 1.6 MHz)	256 (Kclk = 25.6 MHz)	256 (Kclk = 25.6 MHz)
N	8 (IDclk = 1.6 MHz)	128 (IDclk = 25.6 MHz)	128 (IDclk = 25.6 MHz)

Table. 2 ADPLL parameter table

In addition to the according simulation result, the practical implementation supports the algorithm, too. The hold range of the modified ADPLL can lock the phase from **1Hz** to **196 kHz**. The slightly small upper bound may be caused by gate delay of the FPGA, and we can further approach the bound by increasing the values of M and N.

Finally, can we go beyond the limitation? The answer is “yes.” All we have to do is add a frequency detector which counts the number of zero crossing over a reasonable period of time. For example, if the ADPLL has found that the number of zero crossing exceeded a certain threshold, say 190 KHz, the final frequency divider in DCO module would be change from 128 to 64, i.e. $N' = N/2 = 64$, and the clock of both K counter and ID counter remained unchanged. Therefore,

$$25.6MHz = M' f_0' = 2N' f_0' = Nf_0' = 128 f_0' \Rightarrow f_0' = 200KHz$$

We have now changed the center frequency from 100 KHz to 200 KHz while maintain the ratio of M ($M' = 128$) and N ($N' = 64$). Hence, the hold range changes from $0Hz \sim 200KHz$ to $0Hz \sim 400KHz$ theoretically. However, because of the smaller N value, the phase jitter will be increase. In the other words, the quantization error will be increase. To make this jitter smaller, we need a still large N, or an even larger center clock, which is the limitation of practical implementation.

E. Report Problems

1. When the frequency of the input signal is fixed, the frequency of the output signal from the ADPLL still oscillates around the frequency of the input signal. Please explain why this happens.

Ans.

Please see section F-I, F-II-(1), F-II-(2)

2. Please list the pros and cons of ADPLL comparing to that of analog PLL.

Ans.

Pros:

(1) Easy to design because of the only three parameters

(2) Center frequency can be determined merely by M and N

(3) Unlike analog PLL, under-damping transient response is less probable in ADPLL (the frequency domain model shows that the ADPLL is a first order digital filter, which is free from second order damping)

(4) Hold range can be simply determined by M, K, and N, while analog PLL can only improve the performance by new technology or fabrication

(5) Unlike analog PLL, ADPLL has almost the same hold range and lock range

Cons:

(1) The locked frequency is discrete and jitter would be increase when quantization level is small

F. Appendix

This report includes the following as appendix:

- The report itself: *Group6_Lab6.pdf*
- Our first version of ADPLL implemented in verilog code: *original* folder

- Our second version of ADPLL which can lock frequency up to 25% deviation from the central frequency: $f_c \pm 25\%$ folder
- Our final version of ADPLL which can lock frequency up to 100% deviation from the central frequency, that is, $0 \sim 2f$: *final* folder
- Verilog code for our frequency detection module: *freq_detect.v*
- A sample movie of our first version ADPLL on oscilloscope: *original.MOV*

G. References

I. Best, Roland E, "Phase Lock Loop, 4th ed.," McGraw-Hall, 1999

II. Lecture note from the course "Frequency Synthesizers," summer 2003, Georgia Institute of Technology[1]

[1] http://users.ece.gatech.edu/~pallen/Academic/ECE_6440/Summer_2003/ece_6440_su2003.htm