Time-Average-Frequency Flying-Adder Frequency Synthesis Architecture and Digital-to-Frequency Converter

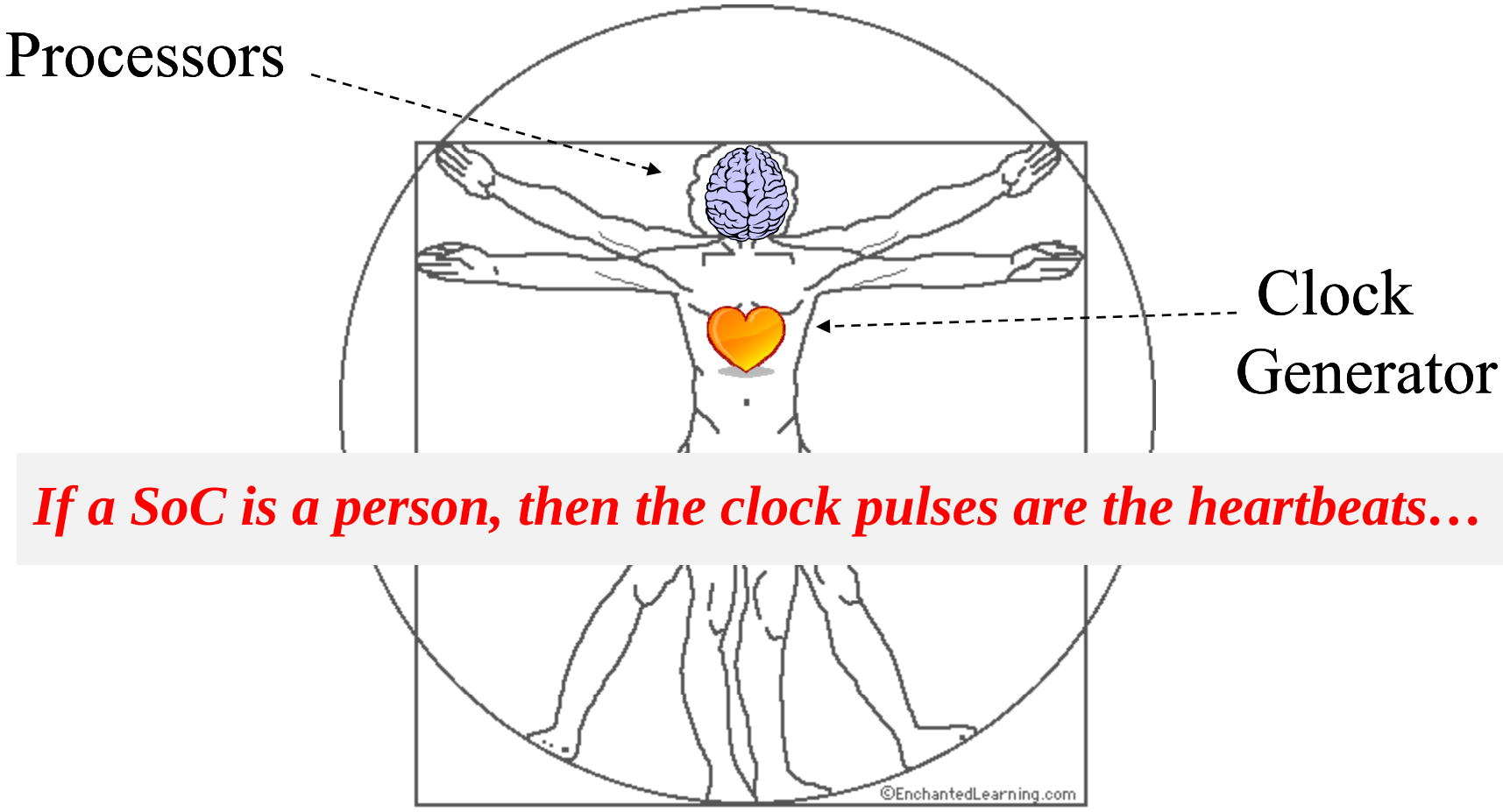
Liming Xiu

08/01/2010

Tutorial Outline

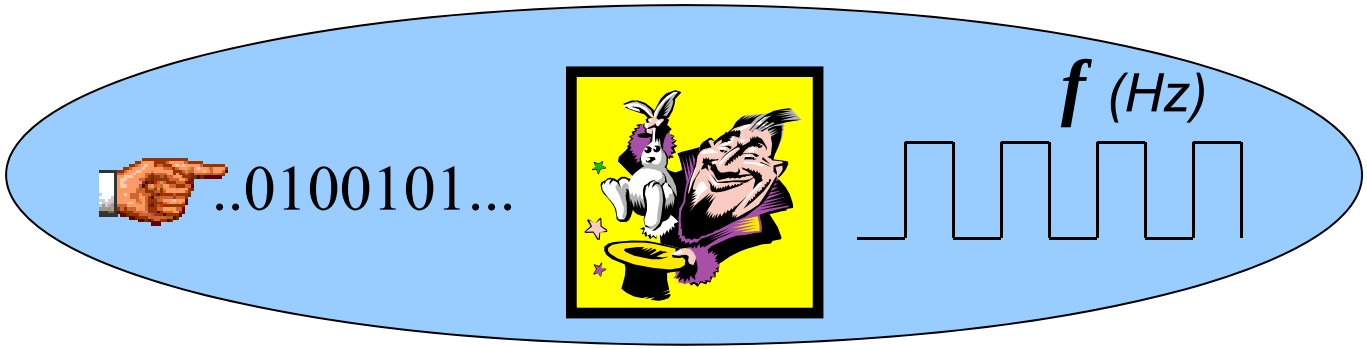
- Part I: The Concept of Time-Average-Frequency (TAF) 7
- Part II: The implementation: Flying-Adder architecture 29
- Part III: The Digital-to-Frequency Converter (DFC) 104
- Part IV: The DFC based information processing approach 110
- Part V: Commercial examples: a circuit-level enabler for system-level innovations 117
- Part VI: The unsolved problems and the opportunities for future 132

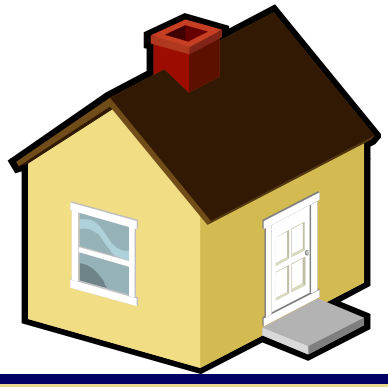
The significance of Clock Signal



If a SoC is a person, then the clock pulses are the heartbeats…

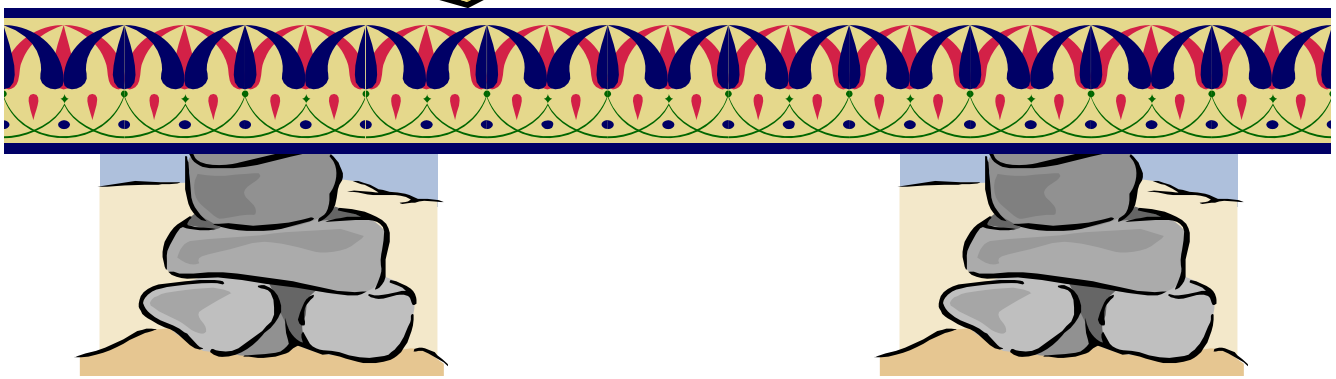
Other than the brilliant brain, you also need a strong heart to compete.





DFC

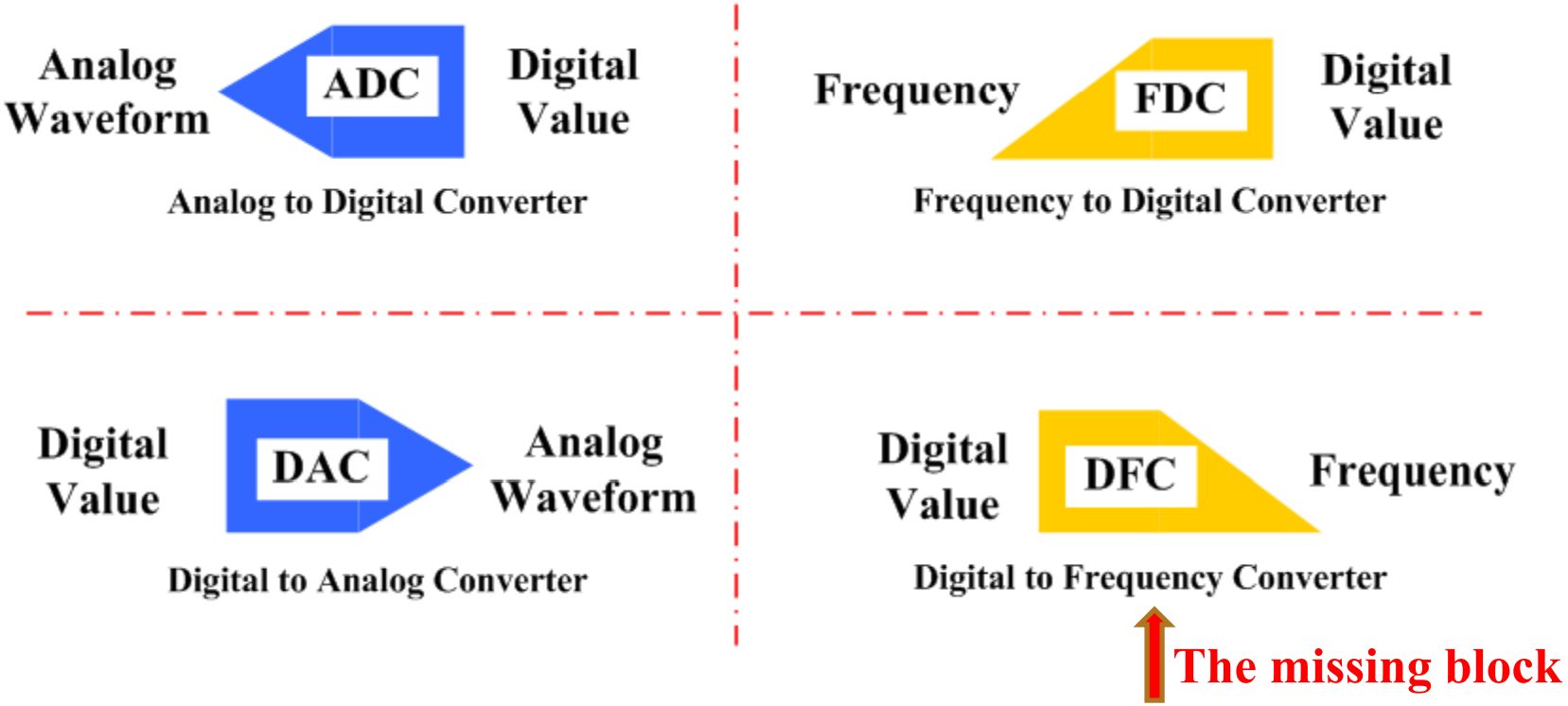
Digital-to-Frequency Converter (DFC)



Flying-Adder Synthesizer

time-average-frequency

It is a new thing



In electronic system design, there is one missing block

- Level ↔ digital: ADC, DAC
- Timing ↔ digital: FDC, DFC

Note: timing is signal crossing a predetermined level threshold. In electronic system, level is a direct variable, timing is an indirect variable.

PLL and DFC: the high level comparison

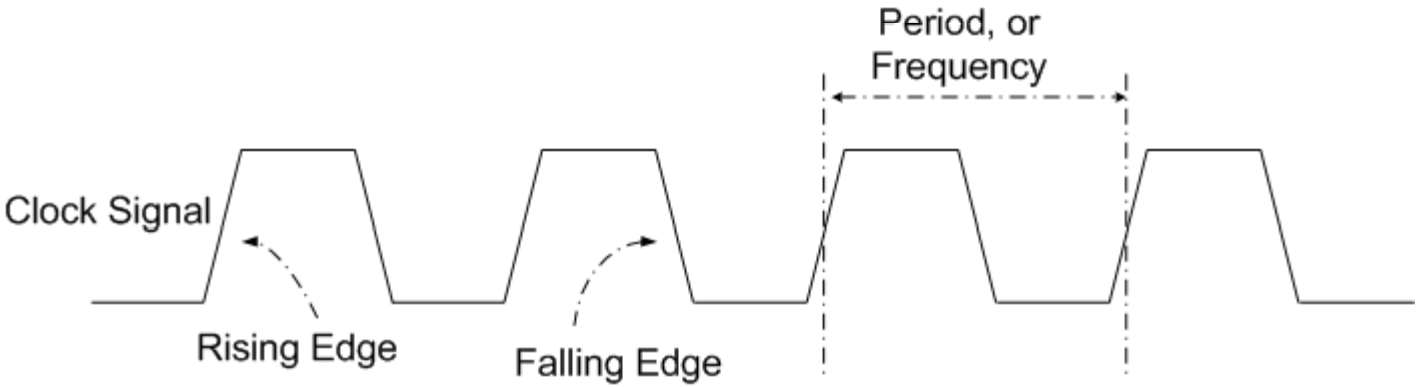
- Both PLL and DFC are timing circuits (to synthesize frequency)
- PLL
  - More than eighty years' history (started ~30s of last century).
  - Circuit: Direct approach (Direct Analog Synthesis and Direct Digital Synthesis) and Indirect approach (PLL based: Integer-N PLL, ($\sum\Delta$) Fractional-N PLL, etc).
  - Primary focus: timing quality (jitter, phase noise)
  - **Two issues which have not been solved to our satisfaction: arbitrary frequency generation and fast response.**
- DFC
  - New: circuit invented in late 90's, theoretical foundation established 2008.
  - Circuit: Direct Period Synthesis (Flying-Adder) → directly construct the pulse waveform.
  - **Fundamental advancement: the concept of Time-Average-Frequency.**
  - Primary focus: **the capability of producing frequency** (with decent timing quality)
  - Impact: a powerful on-chip frequency generator → the heart of complex SoC, a circuit enabler for system level innovation → a new era ahead.

Part I:
The Concept of Time-Average-Frequency (TAF)

- Part I.A: What does frequency mean?
- Part I.B: The definition of Time-Average-Frequency
- Part I.C: TAF time domain behaviour
- Part I.D: TAF clock driving digital circuit
- Part I.E: TAF frequency domain behaviour
- Part I.F: TAF clock driving sampling circuit (ADC, DAC, etc)

Part I.A: What does frequency mean?

The clock signal in electronic systems



Key points:

- Square pulse (not sinusoidal wave).
- Frequency: number of cycles within one second.
- All cycles have same length in time.

Part I.A: What does frequency mean?

What is 1 MHz frequency ?

- Within one second, there are 1000000 cycles (clock pulses, operations, etc).
- Each cycle uses time of 1 us (called period).

What is 0.999999 MHz ?

- Within one second, there are 999999 cycles
- Each cycle uses time of 1.000001 us.

Therefore, 0.999999 MHz can be achieved by designing the clock waveform in such a way that each cycle uses time of 1.000001 us.

This is the conventional approach that everyone uses.

Part I.A: What does frequency mean?

time-average-frequency example: 0.999999 MHz

0.999999 MHz can be achieved by another way

- Using the concept of time-average-frequency
- Having two types of cycles: 1us and 1.01us (as contrast to the 1.000001 us in previous approach).

For every 10000 cycles

- 9999 cycles have period of 1 us.
- one cycle has period of 1.01 us (1% longer).
- Total time used for this 10000 cycles: 9999*1 us + 1* 1.01 us = 10000.01 us.

Within 10000.01 us, there are 10000 cycles → within 1 s, there are 999999 cycles → This is the definition of 0.999999 MHz.

Part I.A: What does frequency mean?

time-average-frequency example: 49.9 MHz

What is 49.9 MHz frequency ?

- Within one second, there are 49.9 millions cycles (clock pulses, operations, etc).
- Each cycle uses time of 20.04 ns.

Therefore, 49.9 MHz can be achieved by designing the clock waveform in such way that each cycle uses time of 20.04 ns.

Another way: two types of cycles:

Type A: T_A = 20 ns

Type B: T_B = 20.1 ns

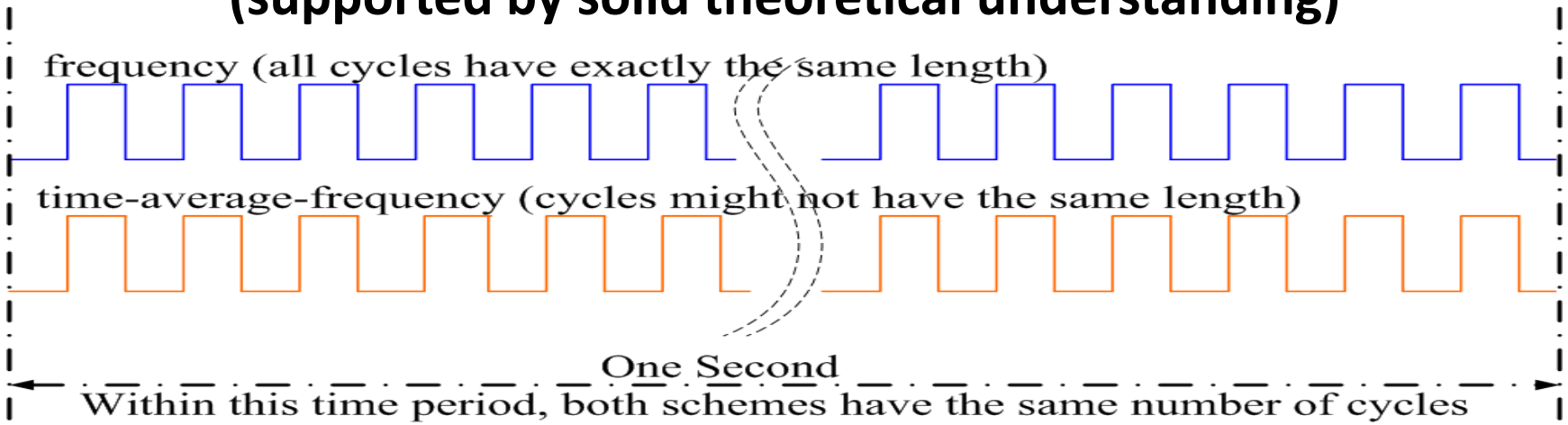
Pattern: ABAAB

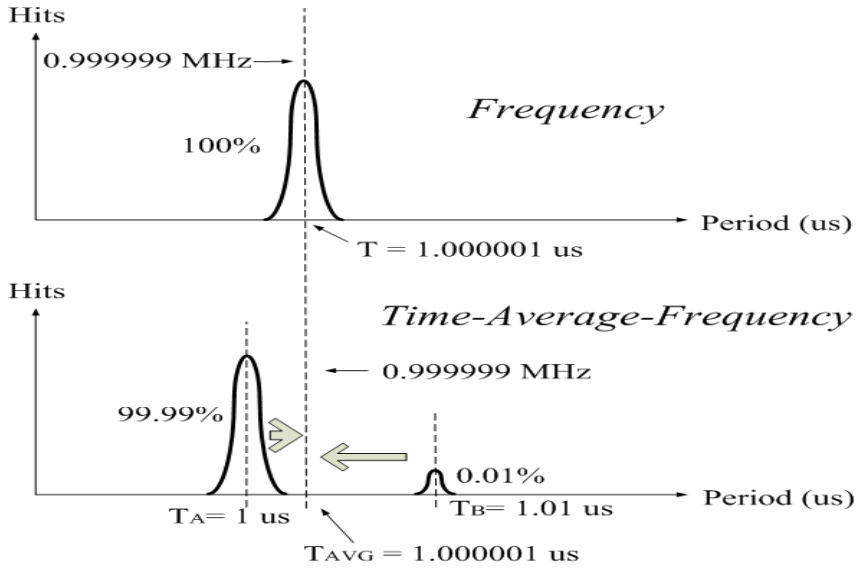
Average frequency: (20*3+ 20.1*2)/5 = 20.04 ns (49.9 MHz)

Part I.B: The definition of Time-Average-Frequency

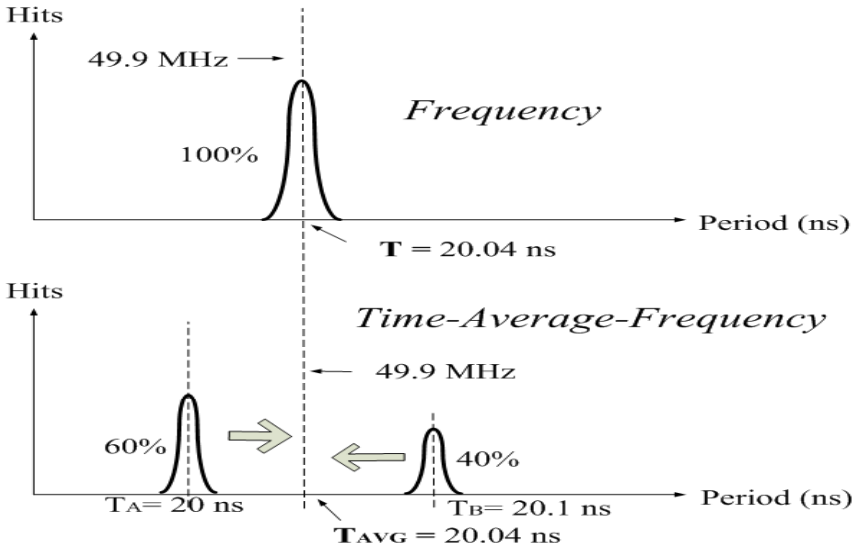
A breakthrough in concept

(supported by solid theoretical understanding)





a) 0.999999 MHz



b) 49.9 MHz

Frequency vs. Time-Average-Frequency

Conventional clock signal

1. The waveform of clock signal is constructed by a defined pattern which repeats itself indefinitely.
2. The defined pattern has two, and only two, distinguishable voltage levels: one is regarded as high and the other is low.
3. The defined pattern has one rising edge which is the intermediate state when the signal transits from its low voltage level to high voltage level, and one falling edge that is the intermediate state when the signal transits from high to low level.
4. Within a given timeframe, such as one second, the number of times that the defined pattern repeats is defined as the frequency of this clock signal, f.
5. The time required to complete one such defined pattern is termed as period of this clock signal, T. By definition, $T \equiv 1/f$.

Time-Average-Frequency based clock signal

1. The waveform of clock signal is composed of infinite number of cycles.
2. Each cycle has two, and only two, distinguishable voltage levels: one is regarded as high and the other is low.
3. Each cycle has one rising edge which is the intermediate state when the signal transits from low voltage level to high voltage level, and one falling edge that is the intermediate state when the signal transits from high to low level.
4. Within a given timeframe, such as one second, the number of cycles that exist in this clock waveform is defined as the time-average-frequency of this clock signal, f.
5. Between any two adjacent rising (falling) edges, the time occupied is defined as the instant period, T, of that cycle which lies between these two edges.
6. The magnitude of the instant period, T, of each cycle must be controllable, or must be known by its creator (the clock synthesizer) when in construction.

time-average-frequency vs. frequency

Why do this ?

- Frequency generation is difficult! **very difficult!!**
- Time-average-frequency makes the task easier.

- The reason that we can do this:
  - In electronic systems, frequency is defined as the number of cycles per second
  - It means: the number of operations within one second.

- **As long as we can guarantee that the specified number of operations (events) happened within one second, the system cannot tell the difference between "frequency" and "time-average-frequency".**

This idea is original, and bold.

time-average-frequency is powerful

- The constraint of "every cycle must have the same length" is removed. This opens up many options:
  - The types of cycles
  - The many different combinations of these cycles
  - The result: a frequency can be achieved in many different ways
  - Many results from a major mathematic field of "number theory" can potentially be used here.

- Now, for a clock frequency of *F* Hz, the constraints become:
  1. Within one second, there are *F* cycles.
  2. When this clock is used to drive systems (digital or analog), reliable *F* operations per second needs to be guaranteed.

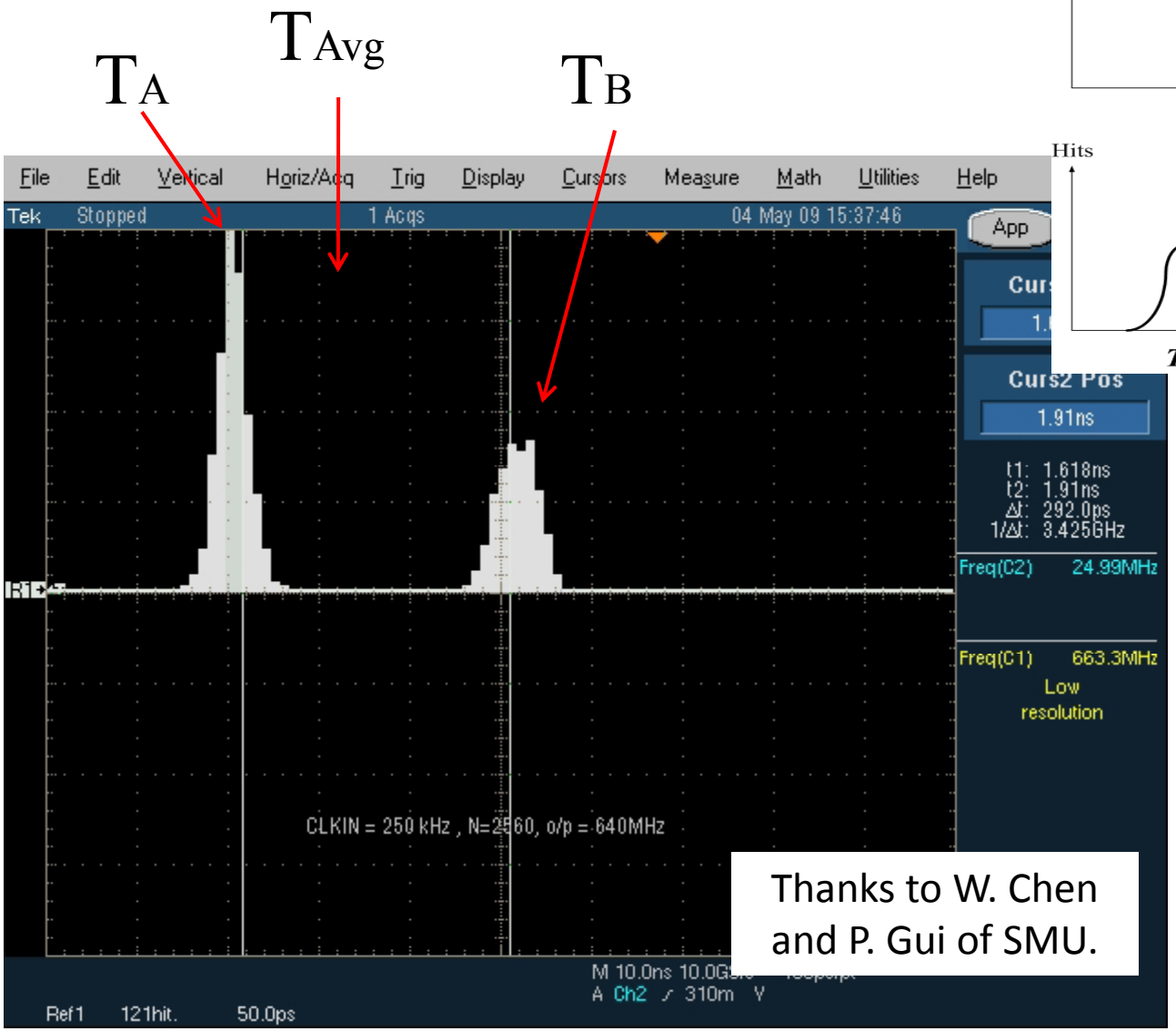
History tells us:

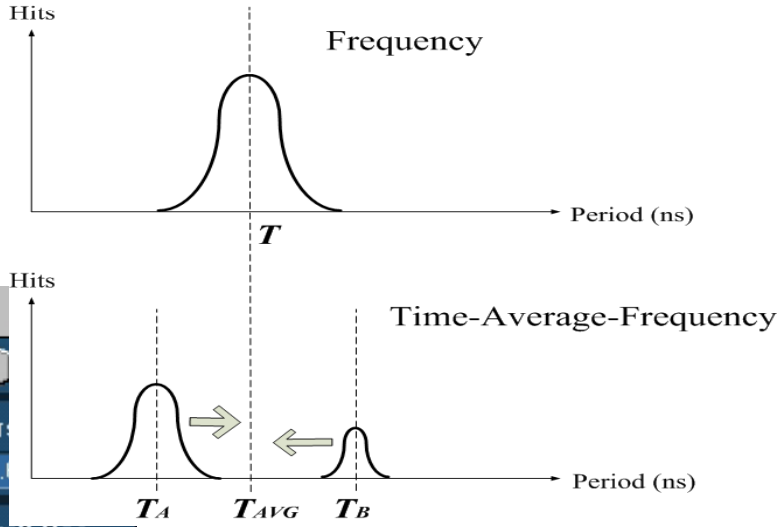
- **Great things start from adventurous thinking.**
- **Breakthrough innovation is usually born from concept advancement.**

• L. Xiu, "The Concept of Time-Average-Frequency and Mathematical Analysis of Flying-Adder Frequency Synthesis Architecture," *IEEE Circuit And System Magazine*, 3rd quarter, pp.27-51, Sep.2008.

A different way of thinking, A new angle of investigation

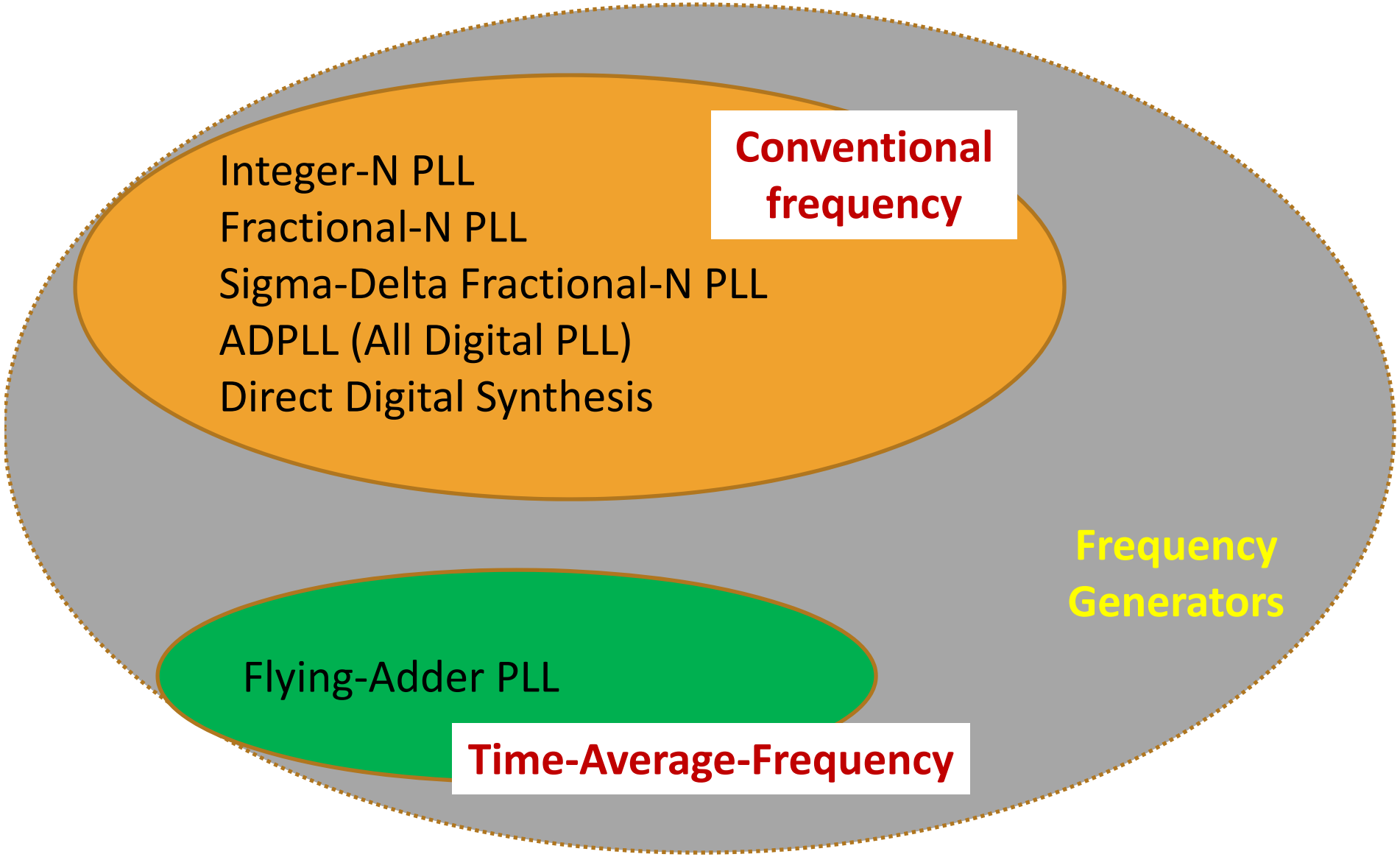
Part I.B: Time-Average-Frequency, an example



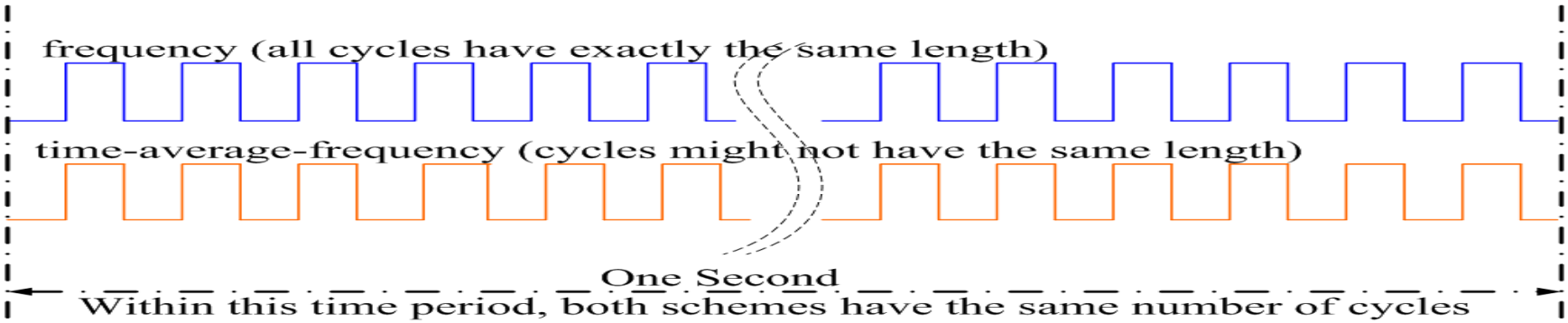


Thanks to W. Chen and P. Gui of SMU.

Part I.B: Frequency vs. Time-Average-Frequency



Part I.C: TAF time domain behaviour



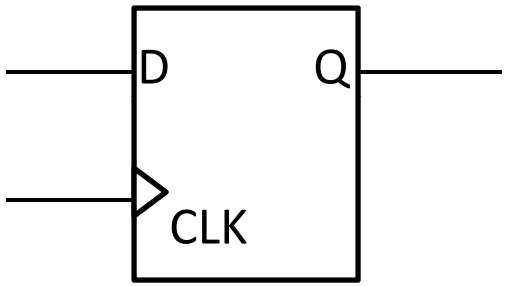
- Two, or more, types of cycles. Each with known and fixed length-in-time.

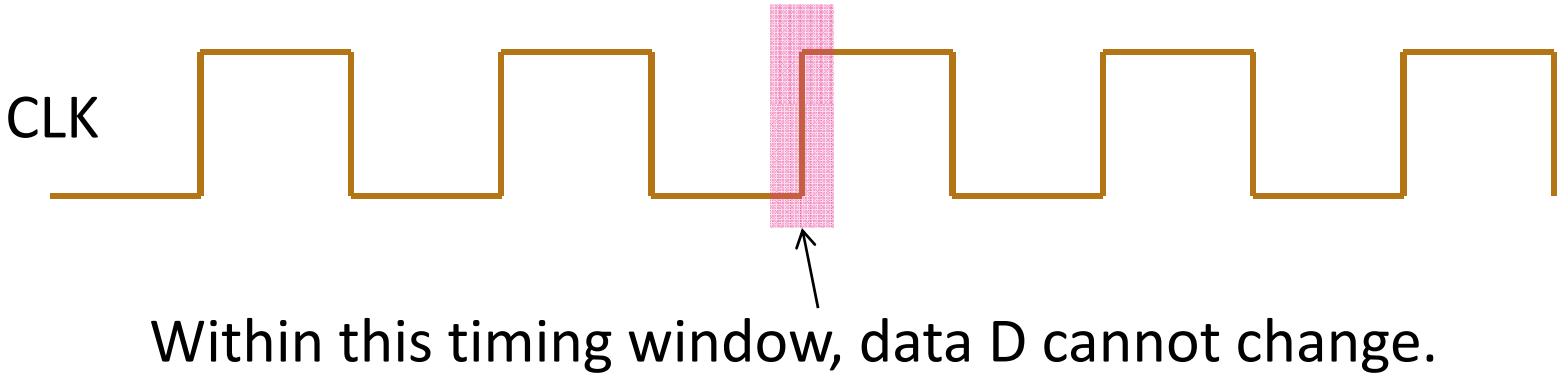
- Flying-Adder implementation: two types of cycles T_A and T_B
  - $T_A = I*\Delta$, $T_B = (I+1)*\Delta$

- The weight of each type of cycle's occurrence is precisely predetermined to achieve the desired frequency.

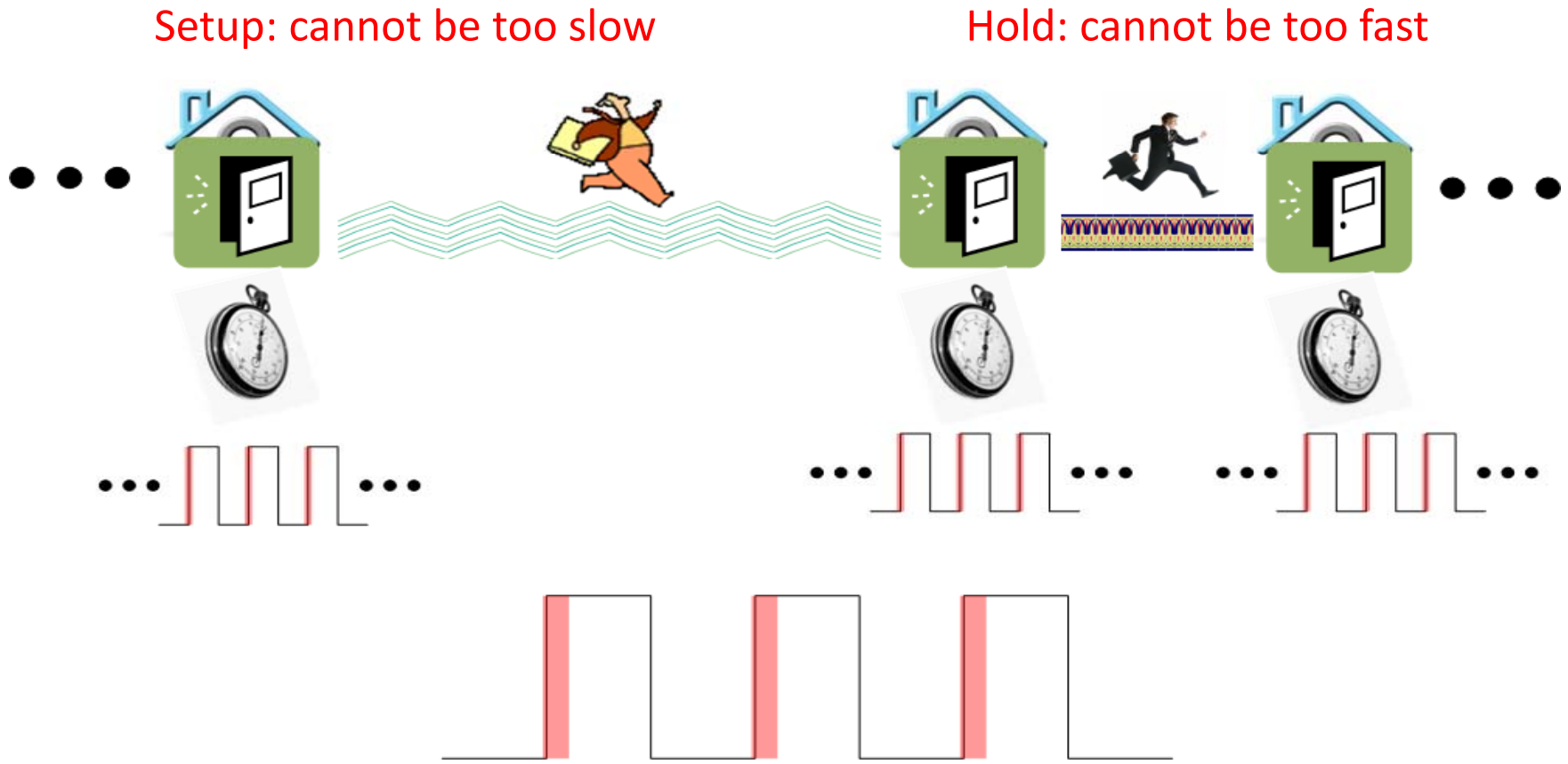
Part I.D: TAF clock driving digital circuit

- The safeguard for digital circuit's operation: setup check and hold check.





Part I.D: The setup and hold check revisit



Important:

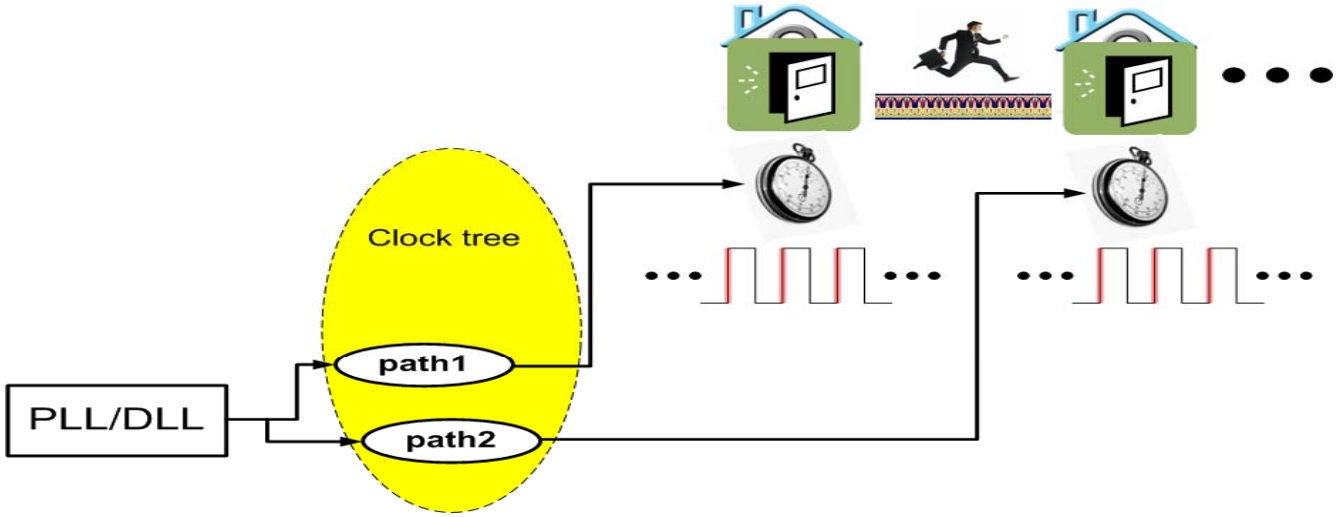
the setup check uses two edges (one cycle)

The hold check uses only one edge (the current edge)

Part I.D: Hold check -> more discussion

- All the clock edge uncertainties (jitters) originated at the clock source (PLL/DLL) have no impact on hold constraint since every FF senses the same effect.

- Only the clock edge uncertainties caused by the clock distribution network (clock tree) need to be considered by hold constraint. Reason: each FF see different thing because of the unique distribution path.



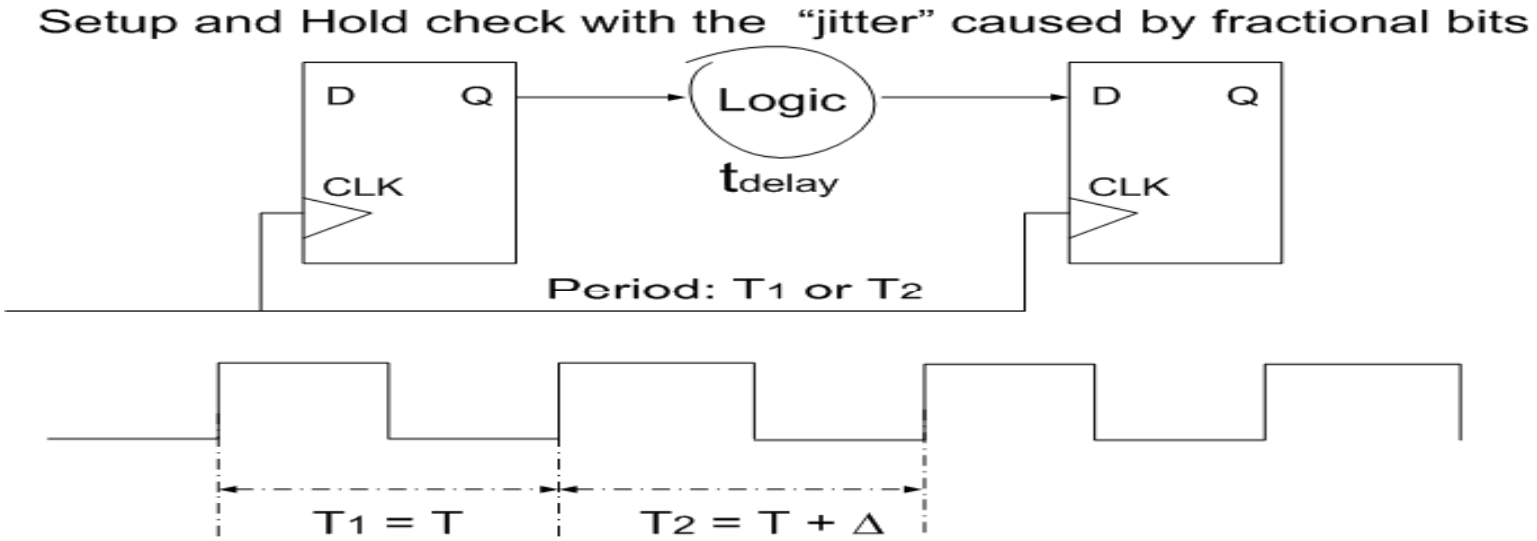
The clock-edge-movement introduced by the Time-Average-Frequency is originated at the source → no impact on hold check!

Part I.D: TAF clock driving digital circuit

- The cycle-prolong relaxes the setup constraint, or make it easier to meet, since t_{delay} compete with T_2 now, which is one Δ longer.

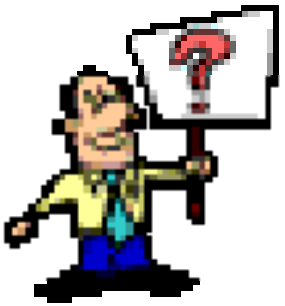
- This has no impact on hold constraint since hold check uses one clock edge. It has nothing to do with clock frequency (period) .

- Therefore, this cycle-prolong has no impact on digital circuit's operation!



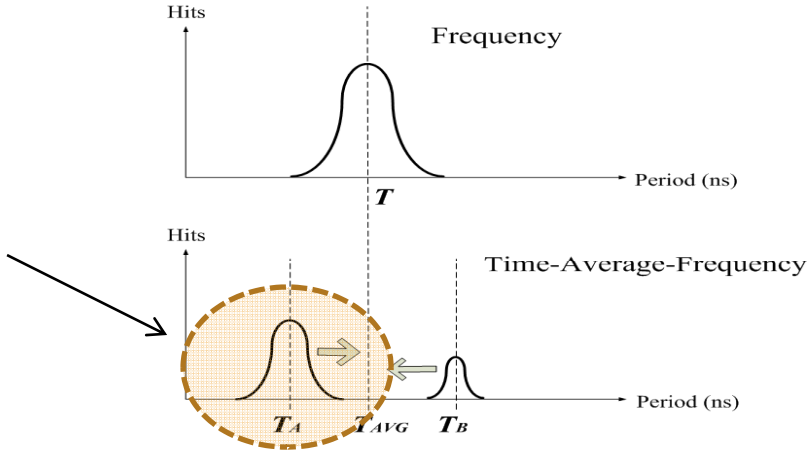
Part I.D: What about jitter ?

- What about jitter?



- The cycle-length-manipulation of time-average-frequency is deterministic.

As long as you use this peak (left branch) as setup constraint, your digital circuit is 100% safe.



- **For anything in electronic systems, as long as it is deterministic, we have ways to deal with it!**

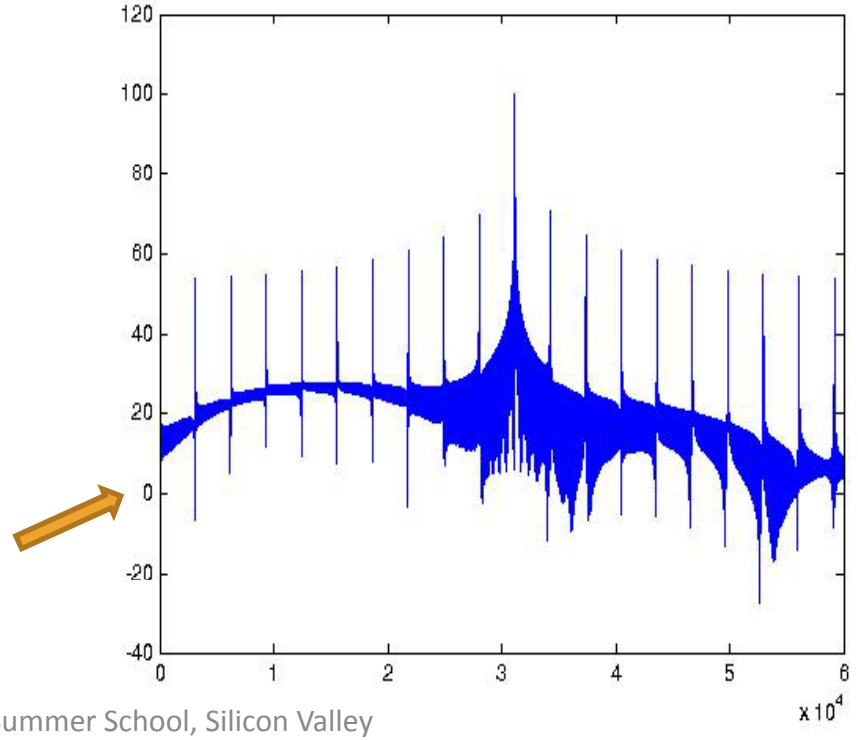
It is not jitter.

Part I.E: TAF frequency domain behaviour

- The TAF uses two, or several, types of cycles. These cycles occurs alternatively.
- If their alternative appearances happen regularly, they are periodical events.
- These periodical events will show up as spurs in spectrum.

Example:

- for every ten cycles, there are nine T_A cycles and one T_B cycle
- $T_{avg} = 9*T_A + 1*T_B$
- This periodical event generates spurs



Part I.E: TAF frequency domain behaviour

- The spurs' location and magnitude can be precisely calculated.
- The spurs can be converted to noise.
- Theoretical foundation is already established , but still lots of work ahead, and lots of opportunities …

• L. Xiu, "Some Open Issues Associated with the New Type of Component: Digital-to-Frequency Converter," *IEEE Circuit And System Magazine*, 3rd quarter, pp.90-94, Sep.2008.

• **Paul Sotiriadis, "Theory of Flying-Adder Frequency Synthesizers Part I: Modeling, Signals Periods and Output Average Frequency," *IEEE Transaction on Circuit And System I*, available now in IEEE Xplore.**

• Paul Sotiriadis, "Theory of Flying-Adder Frequency Synthesizers Part II: Time and Frequency Domain Properties of the Output Signal," *IEEE Transaction on Circuit And System I*, available now in IEEE Xplore.

• L. Xiu, C.W. Huang and P. Gui, "The Analysis of Harmonic Energy Distribution Portfolio for Digital-to-Frequency Converters," *IEEE Trans. Instrum. Meas,* to appear.

Part I.F: TAF clock driving sampling circuit (ADC, DAC, etc)

- When TAF clock is used to drive ADC and DAC, the impact of these spurs has to be studied case by case.

- An example:
  - Gui, P., Gao, Z., Huang, C.W. and Xiu, L. "The Effect of Flying-Adder clock on Digital-to-Analog Converter", *IEEE Transaction on Circuit And System II*, *Jan. 2010.*

- Other approaches:
  - Convert the spurs to noise
  - Move the spurs to other locations (without alerting the carrier frequency)

- Still a long way to go …
  - Need talented people to investigate this issue in more depth

The Implementation: Flying-Adder architecture

TAF is the concept breakthrough, FAPLL is the work horse

Part II:
The implementation: Flying-Adder architecture

Part II.A: The principle idea

Part II.B: The circuit implementation

Part II.C: The Integer-Flying-Adder architecture

Part II.D: The Post Divider Fractional Bit Recovery (PDFR)

Part II.E: The summary on its distinguished features

Part II.F: The phase synthesis

Part II.G: Deal with the spurs

Part II.H: The energy distribution of TAF clock signal

Part II.I: The comparison with other frequency synthesis techniques (Integer-N PLL, Delta-Sigma Fractional-N PLL, DDS, ADPLL, etc)

Part II.J: A phase divider

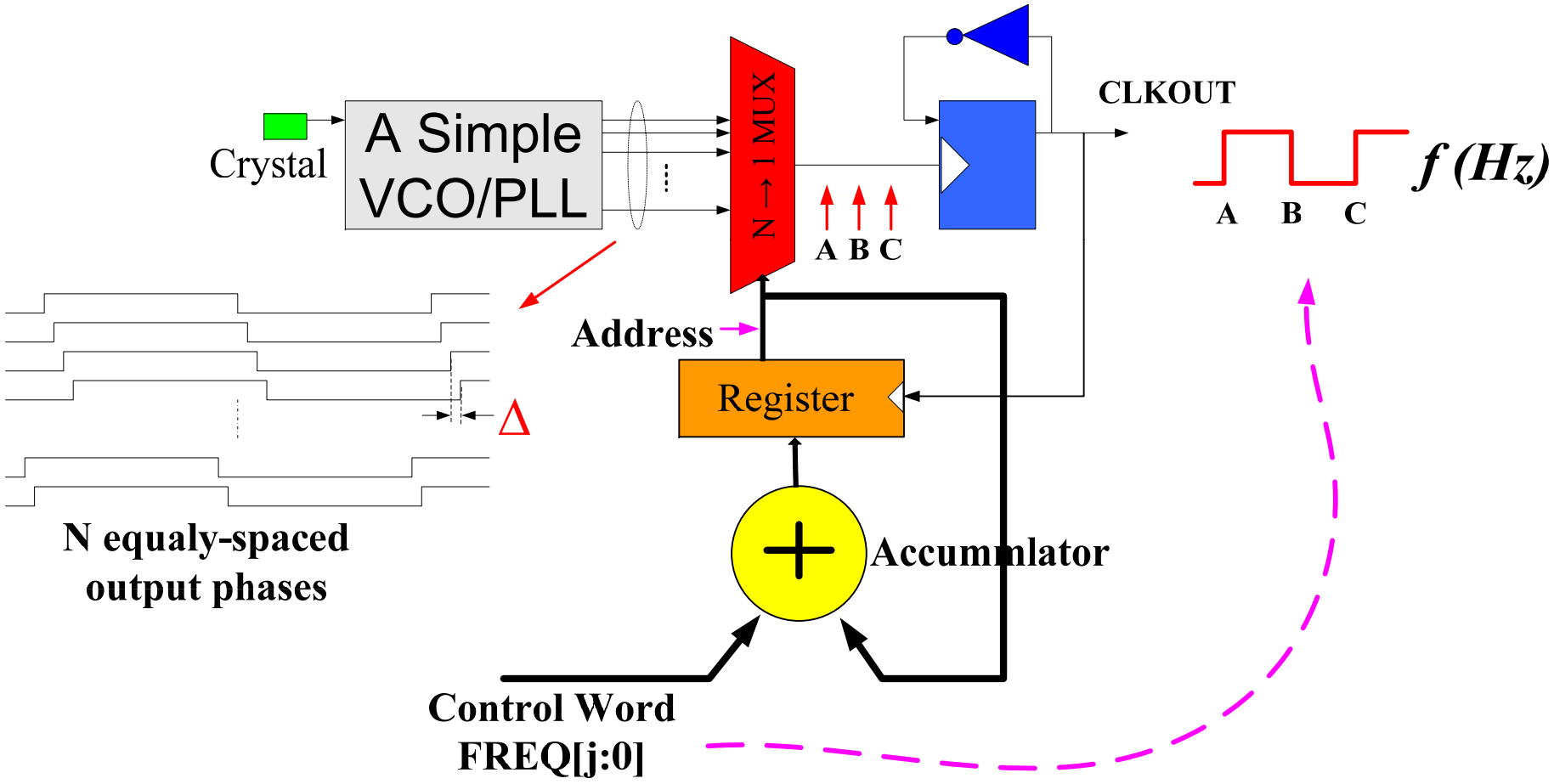
Part II.K: A digital control oscillator

Part II.L: A new type of accumulator: Xiu-accumulator

Part II.M: The layout mismatch and deterministic jitter

Part II.A: The principle idea

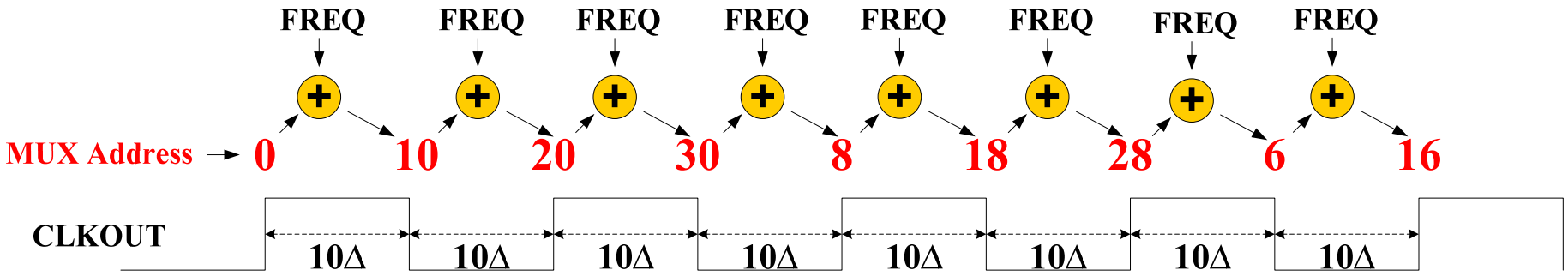
The very first idea: a circuit in principle



Part II.A: The principle idea

- Assume: VCO has 32 outputs (N = 32), the time difference between any two adjacent VCO outputs is Δ.

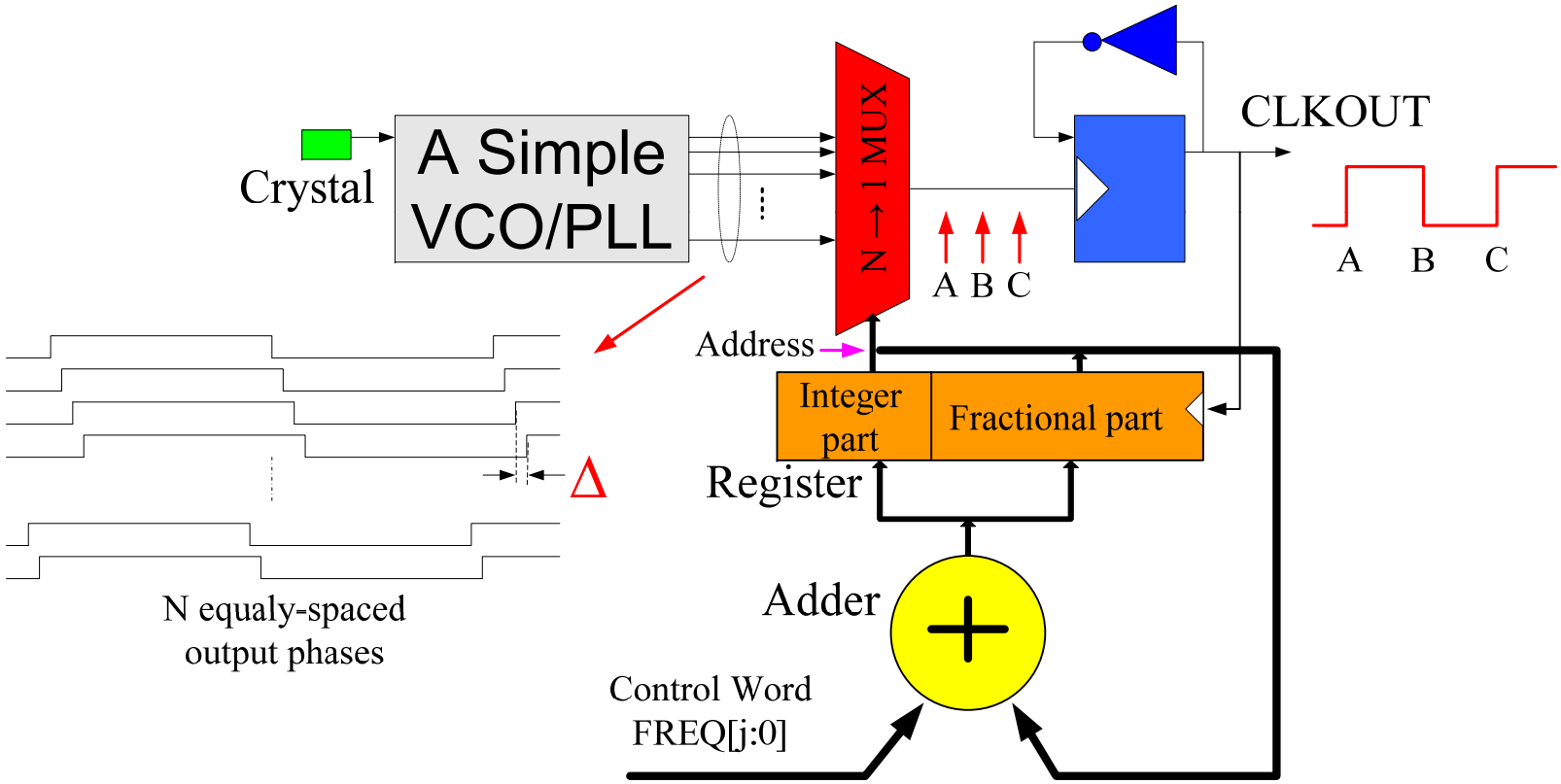
- If we set the frequency control word FREQ[4:0] = 10 = 01010b, then the output clock frequency (period): 20Δ.



- If VCO is running at 625 MHz (1.6 ns), then Δ = 1.6/32 = 50 ps → the current clock output, when FREQ = 10, is at 20Δ = 1 ns (1 GHz).

Part II.A: The principle idea

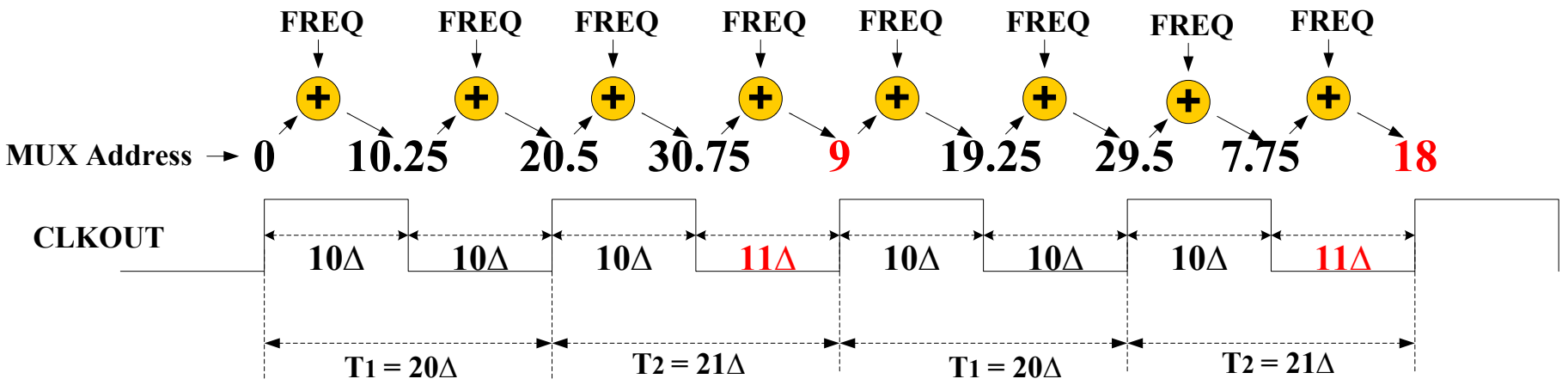
One step forward: add fractional number in FREQ



Part II.A: The principle idea

One step forward: add fractional number in FREQ

- Now we set the frequency control word FREQ[9:0] = 10.25 = 01010.01000b,
  → the output clock frequency (period): 20.5Δ.



- By using fractional number in FREQ, we have two types of cycles T_1 and T_2
  → we produce a new frequency!

- The average frequency: 20.5Δ = 1.025 ns (975.6 MHz).

Part II.A: The principle idea

The cycle-prolong and Jitter

- The Flying-Adder clock is built upon two type of cycles: T_S and T_L: T_L is one Δ longer than T_S, or $T_L - T_S = \Delta$.
- The cycle of T_L appears whenever there is fractional carry-in happening from the operation of accumulator. This is called cycle-prolong.
- The cycle-prolong is deterministic, it is also always in one direction (making cycle longer).
- The cycle-prolong is different than the "jitter".
  - Jitter is edge uncertainty
  - Cycle-prolong is known event. Its size is fixed. Its occurrence is predictable.
- The cycle-prolong has no impact on digital operation.

Part II.B: The Circuit Implementation

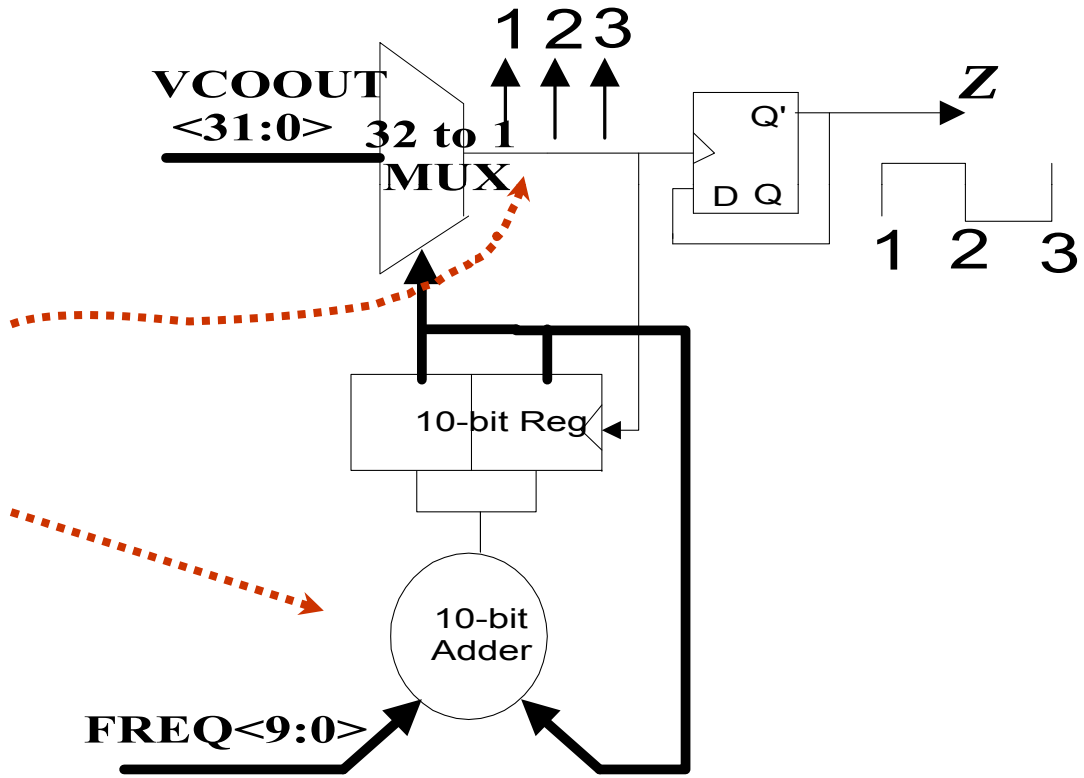
The circuit implementation:

the issues need to be addressed for real working circuit

Two problems:

The glitch of the MUX

The speed of the adder

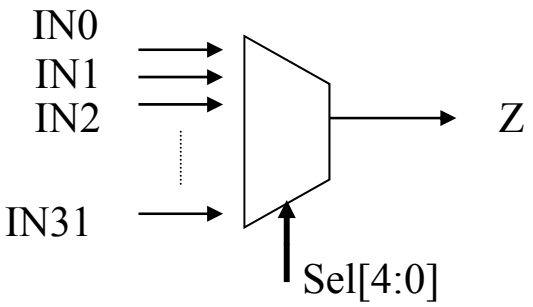


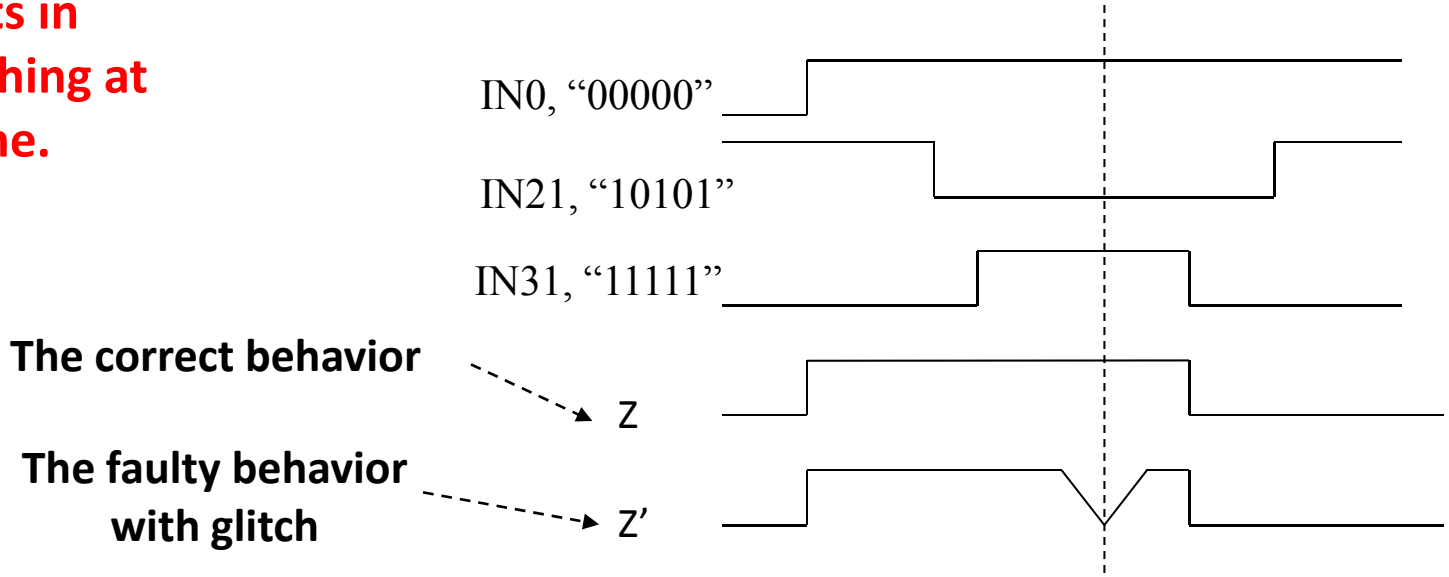
Part II.B: The Circuit Implementation

the glitch of MUX

The source of glitch: there is no guaranty that the address-change can be done instantly.

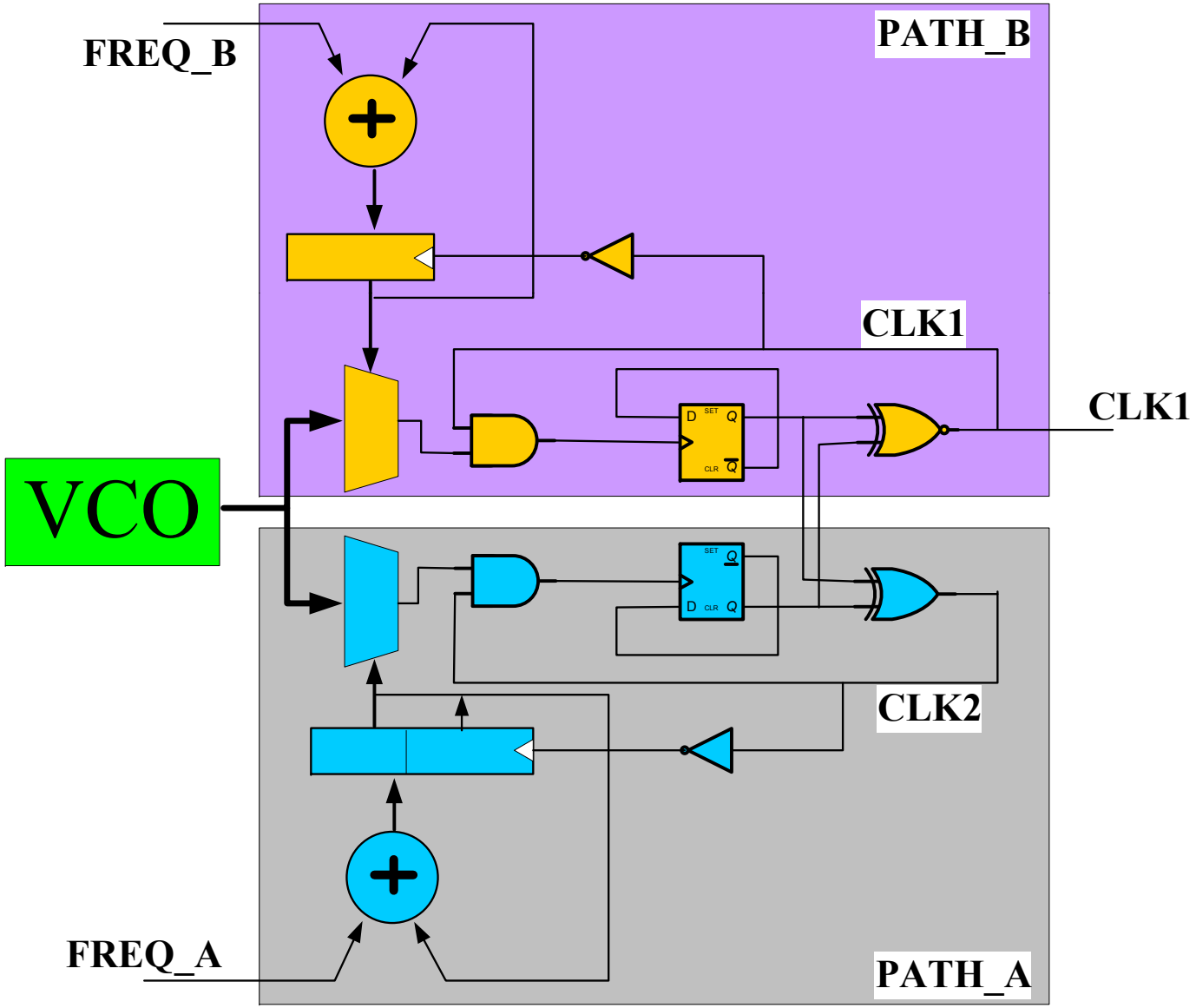
Or, there is no guaranty that all the bits in sel[4:0] are switching at the same time.





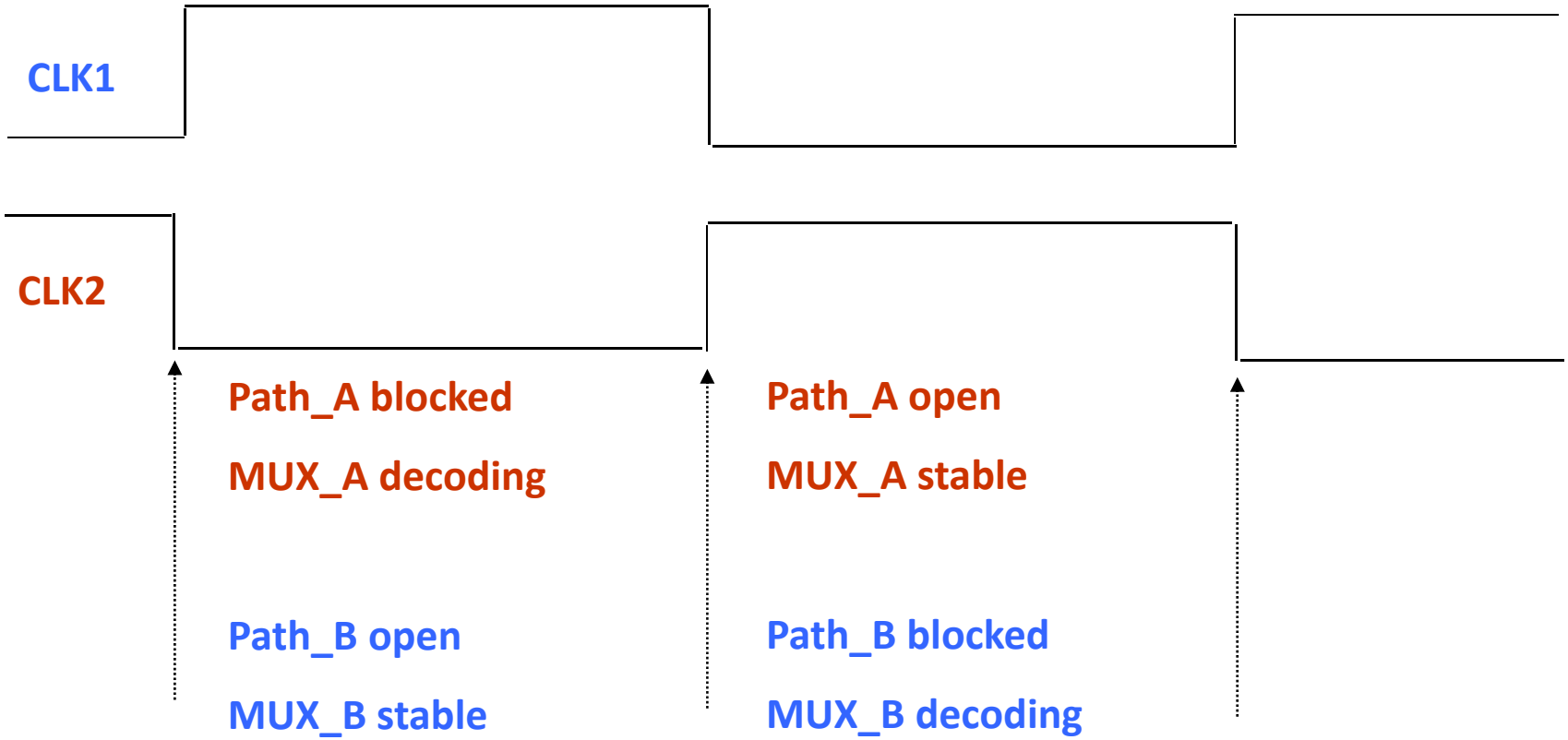
Part II.B: The Circuit Implementation

Using two paths to solve the glitch problem



Part II.B: The Circuit Implementation

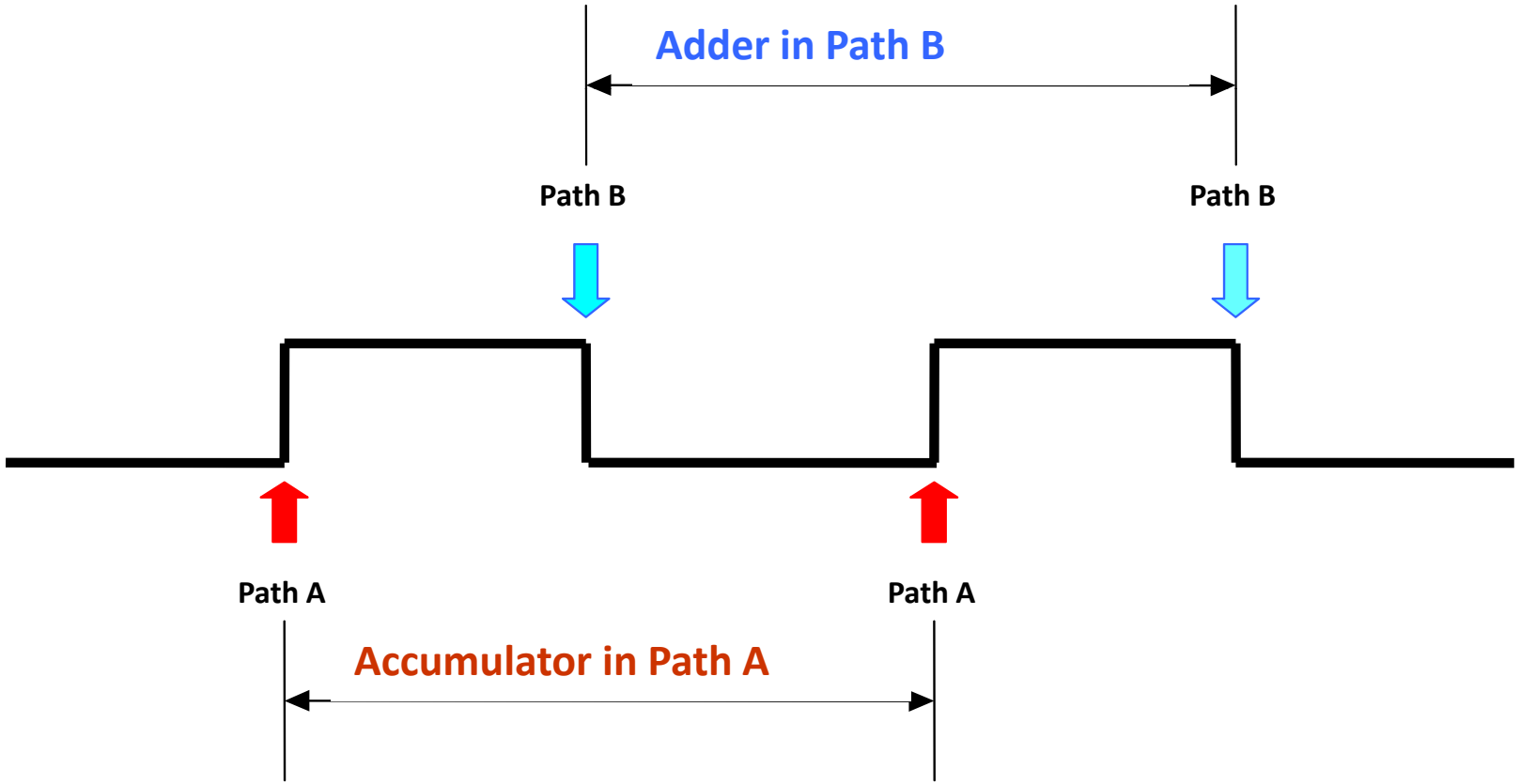
The two paths are interlocked → no glitch on CLK1



Part II.B: The Circuit Implementation

The two paths also relax the speed constraint on adder

- Relaxed the constraint on adders => double the circuit speed
- One path generates the rising edge, whereas the other for falling edge

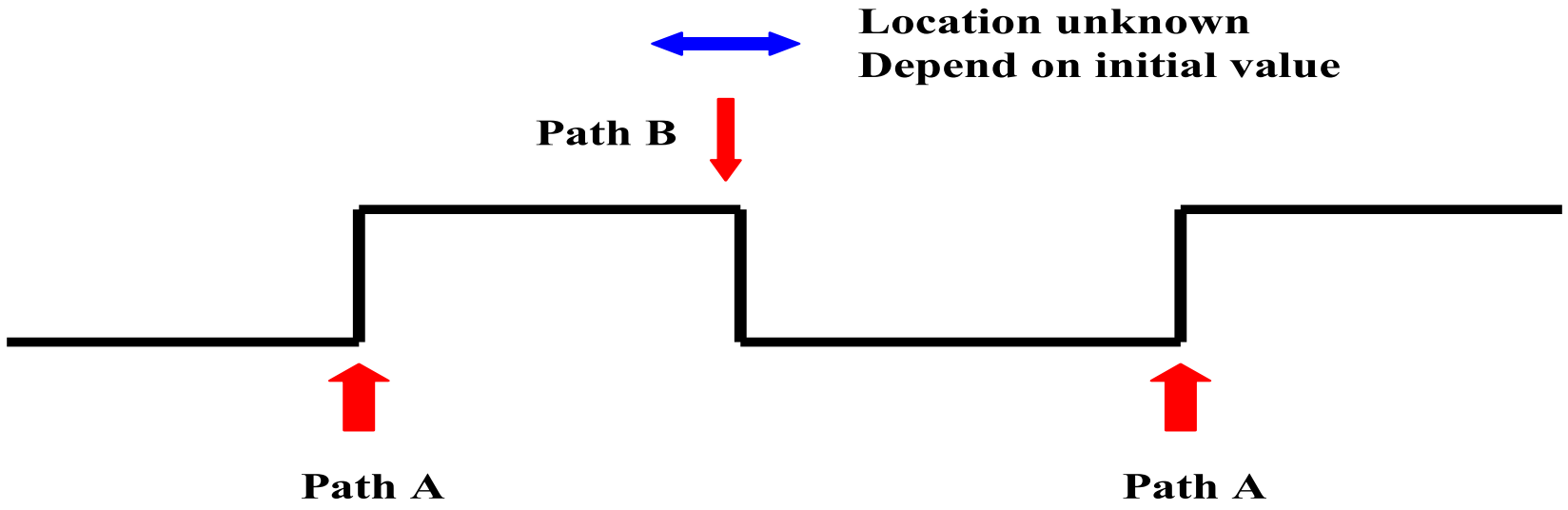


Part II.B: The Circuit Implementation

However, the two paths have synchronization problem

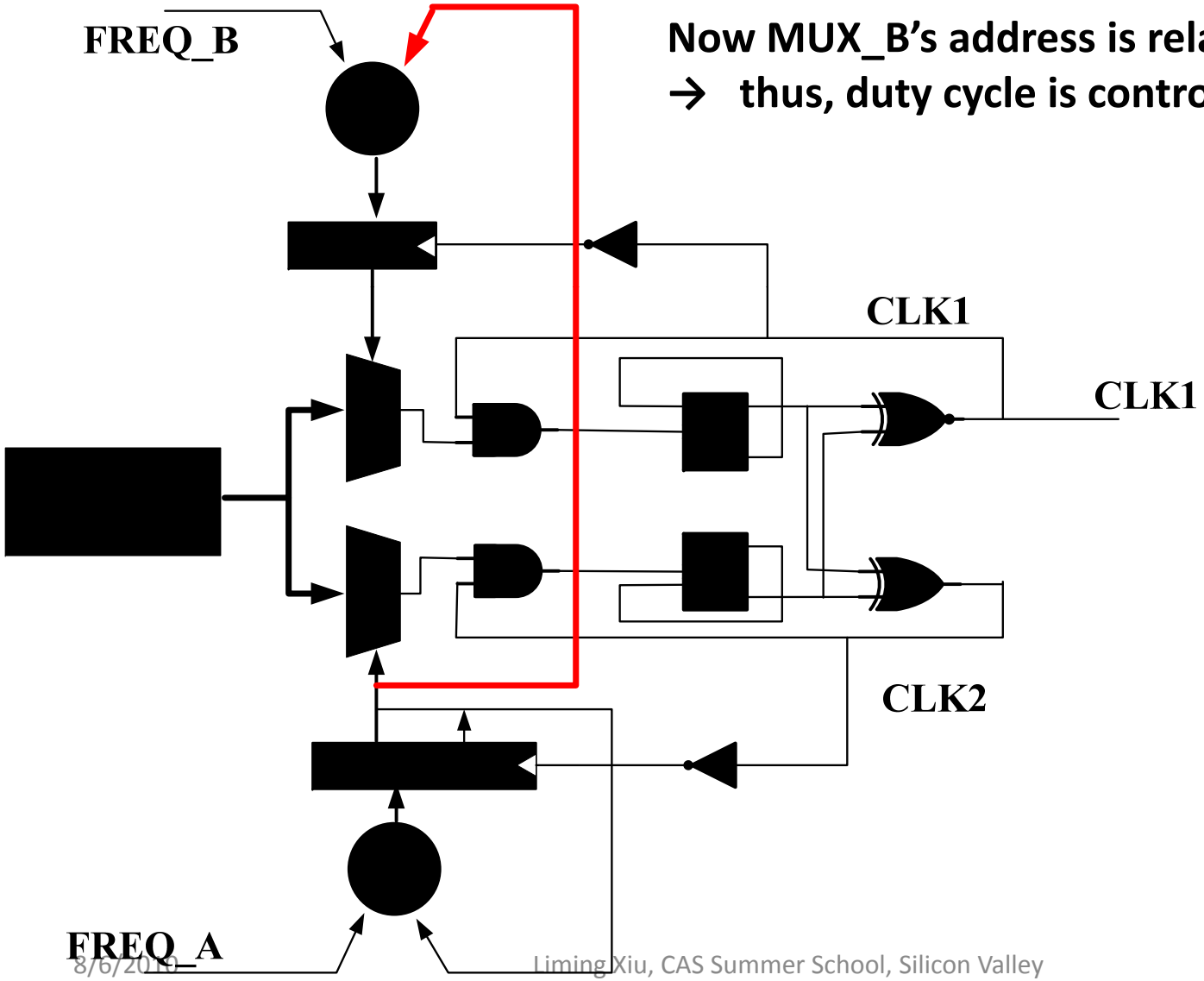
- **The two-paths architecture solves the previous two problems, but creates a new one: the synchronization of the two paths.**

- **In other words, MUX_A and MUX_B's address values are unrelated → duty cycle is uncontrollable.**



Part II.B: The Circuit Implementation

synchronize the two paths

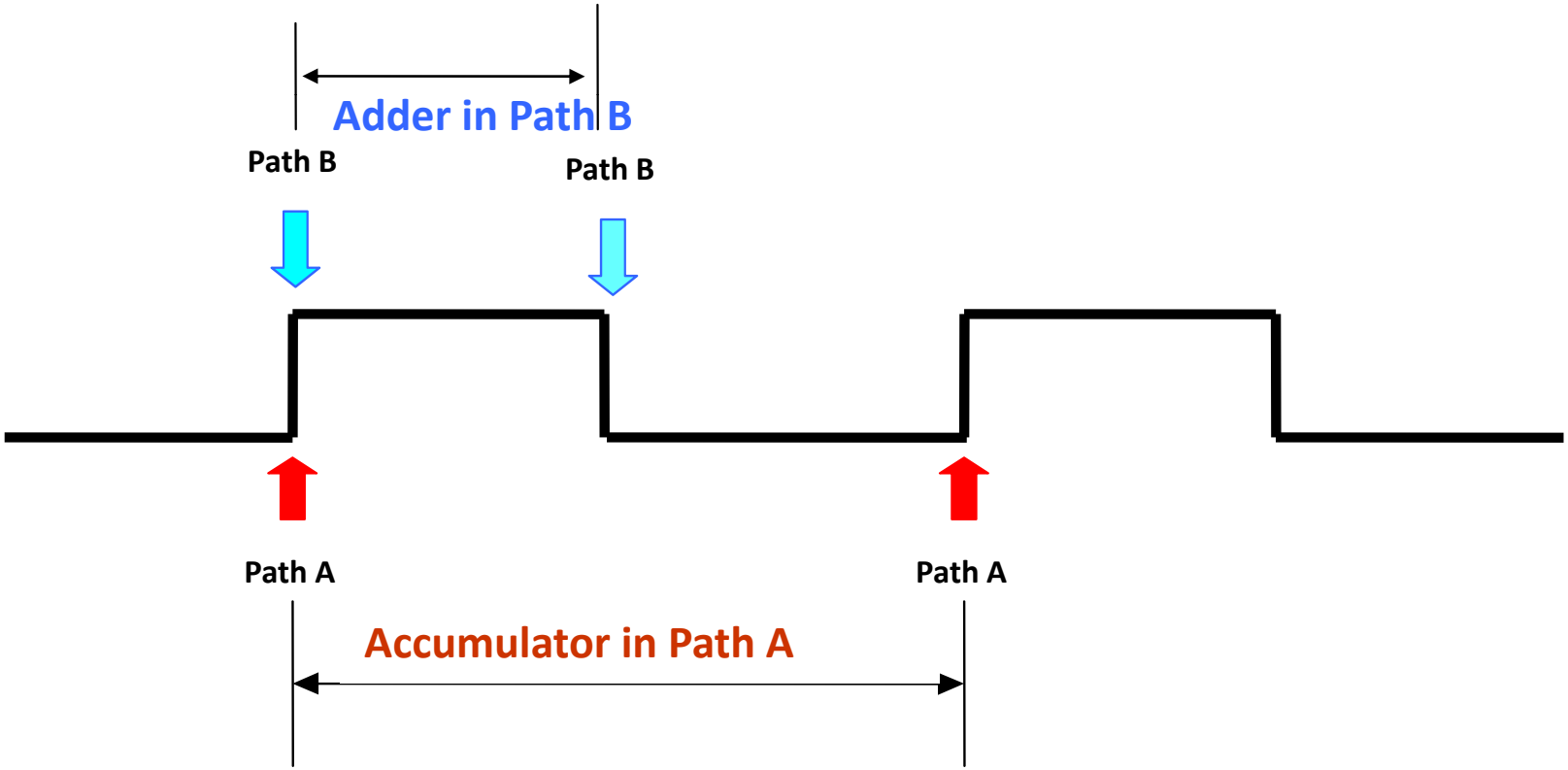


Now MUX_B's address is related to MUX_A's
→ thus, duty cycle is controllable.

Part II.B: The Circuit Implementation

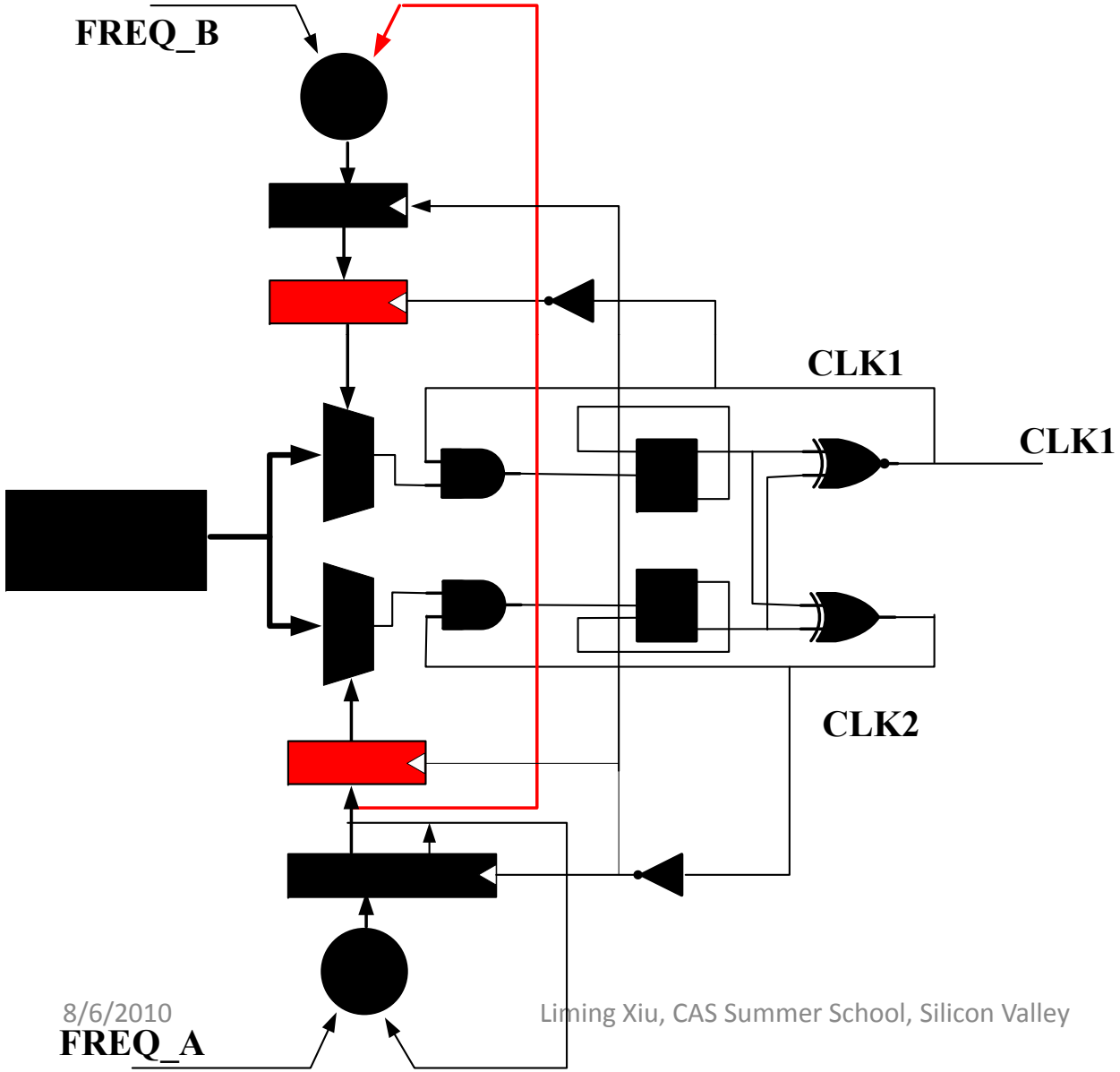
the synchronized two paths: still have a problem

New problem: Adder in PATH_B doesn't have full cycle to work



Part II.B: The Circuit Implementation

the synchronized two paths: add a pipeline stage



Part II.B: The Circuit Implementation

Summary of the first generation circuit

- **First generation Flying-Adder circuit development history:**
  - **One Path**
  - **Two Paths**
  - **Synchronized**
  - **Pipelined**

- **Key features of this architecture:**
  - **interlocking between paths**
  - **self-clocking (embedded feedback)**
  - **pipeline**

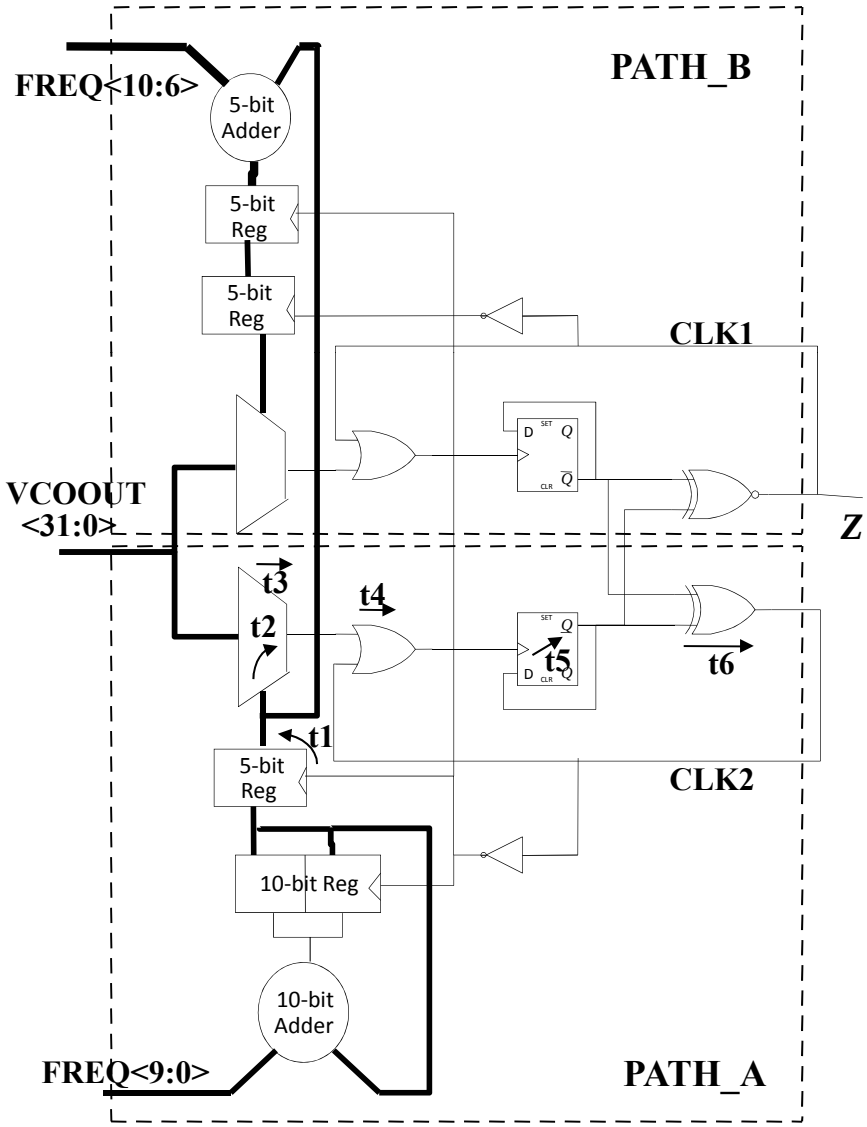
Part II.B: A Milestone

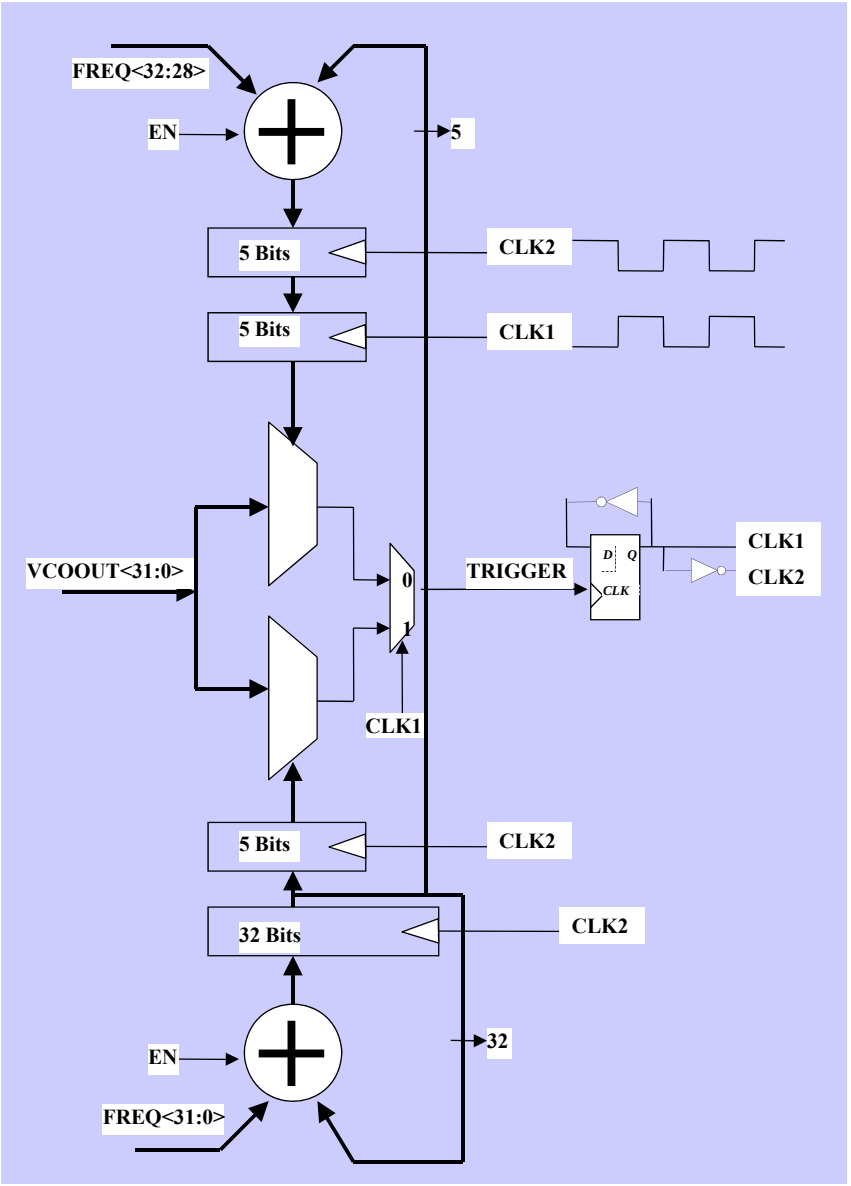
- This circuit is a milestone, it opens up a new subfield in frequency synthesis: Direct Period Synthesis.

- The first two Flying-Adder patents was granted in 2001 (US 6329850, TI) and 2005 (US6940937,TI). In the following 8 years, there are 24 other patents cited these groundbreaking patents. More are coming …
  - Intel (US7124154, 2006), NEC (US6856658, 2005), Samsung (US7336702, 2008), ADI (US7,437,590, 2008), Infineon (US6639435, 2003), Broadcom (US6791379, 2004), Hewlett-Packard (US7457904, 2008) , Altera (US7358783, 2008 and several others), Fujitsu (US7319349, 2008), Panasonic (US7532250, 2009), Marvell (US7525351, 2009) and etc.

- However, the average frequency concept is only used subconsciously. The Time-Average-Frequency concept is not rigorously formed until 2008.
  - To go further, theoretical understanding is definitely needed.

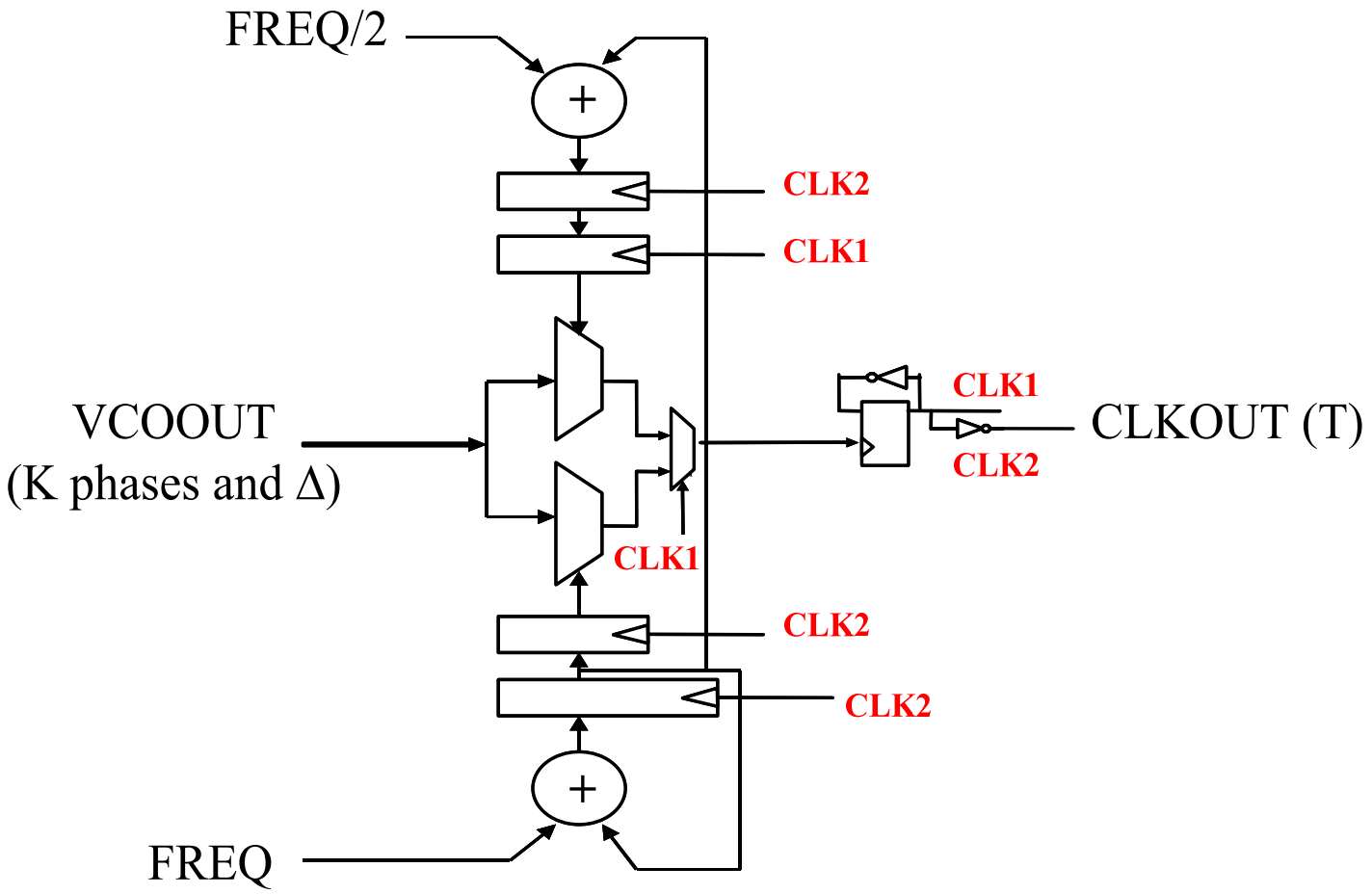
Part II.B: The Circuit Implementation - second generation circuit





Part II.B: The Circuit Implementation

A perfect circuit. A circuit of no fault.
Battle-proven for more than ten years now.
An example of "simple and elegant"



Part II.B: The Circuit Implementation

Summary: the frequency range and resolution

- **The operating equation:**
  - ***T = 1/f = FREQ*Δ***
    - **K: number of VCO outputs**
    - **T = 1/f : desired frequency,**
    - **Δ: time difference between any two adjacent VCO outputs,**
    - **FREQ: frequency control word, could use fractional number, *2 ≤ FREQ < 2K*.**

- **Output frequency range:**
  - $(1/2)f_{vco} \le f_{out} \le (K/2)f_{vco}$

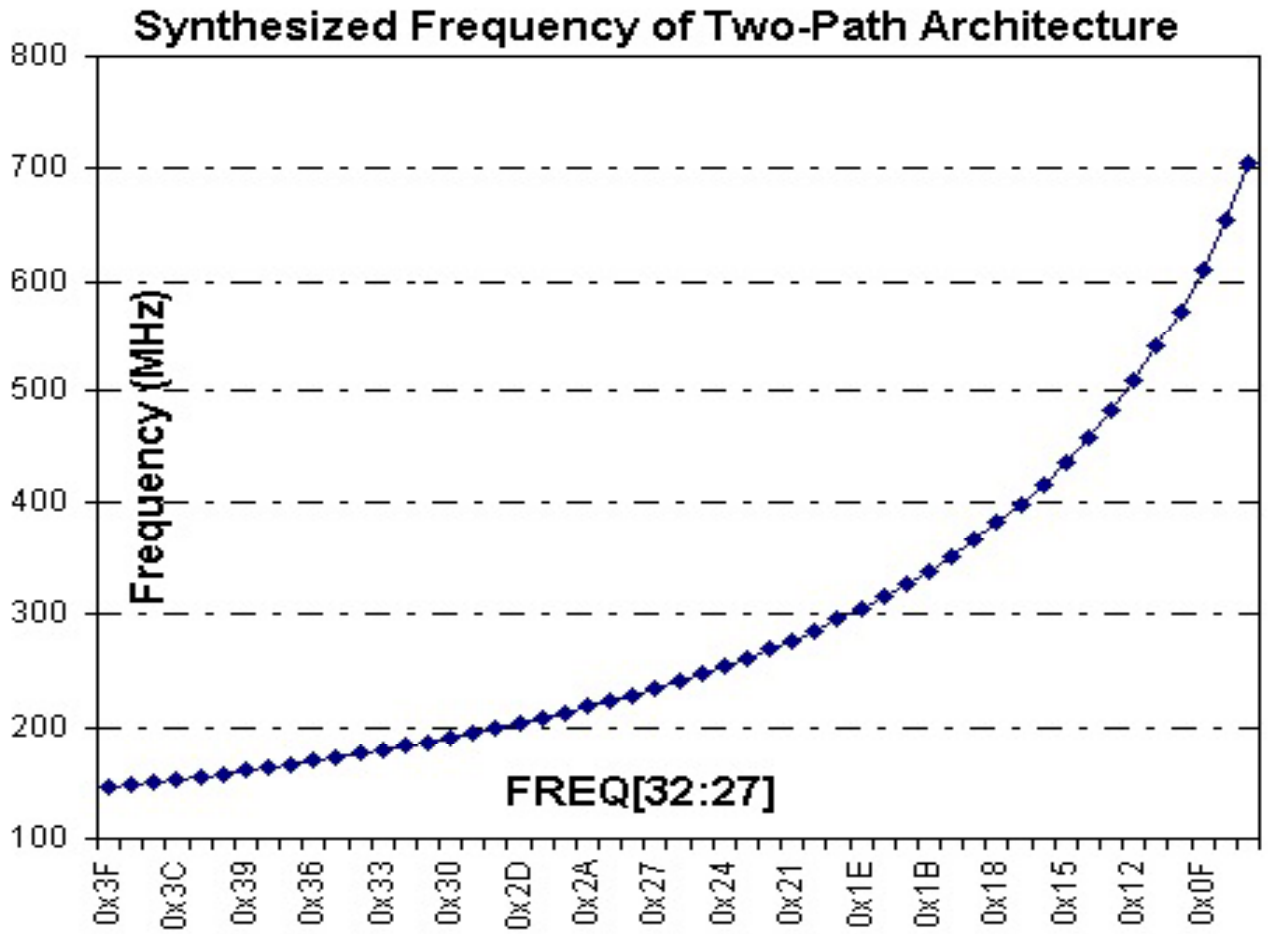
- **In practice, the highest achievable frequency is limited by the speed of the process in which this architecture is implemented.**

- **Frequency resolution (step):** $\delta f = -2^{-p} * \Delta * f^2$

  p: number of fractional bits in FREQ

Part II.B: The Circuit Implementation

Frequency transfer function: it looks like 1/x



Part II.B: The Circuit Implementation

Some reference on circuit details

L. Xiu, Z. You, "A Flying-Adder architecture of frequency and phase synthesis with scalability," *IEEE Trans. on VLSI*, vol.10, pp. 637-649, Oct., 2002.

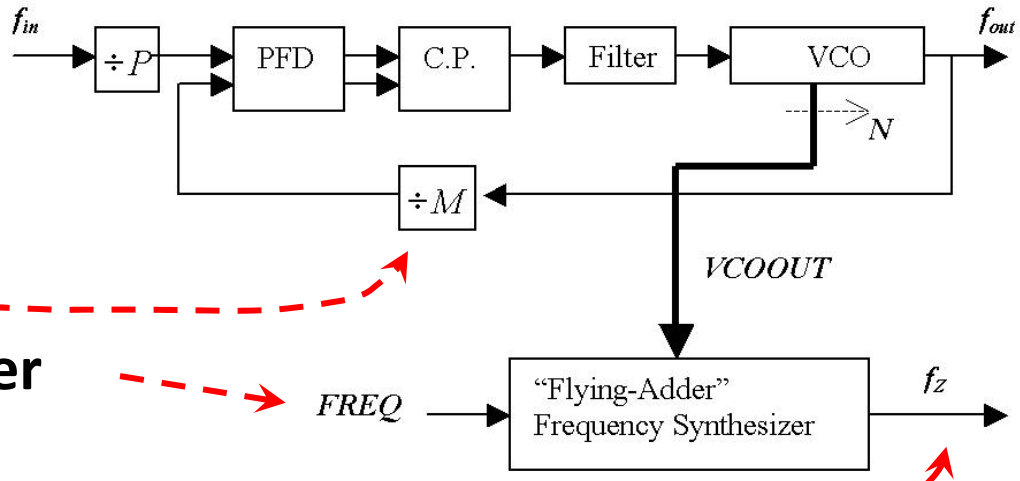
H. Mair, L. Xiu, "An architecture of high-performance frequency and phase synthesis," *IEEE J. Solid-State Circuits*, vol. 35, pp. 835-846, June 2000.

L. Xiu, Z. You, "A Flying-Adder Frequency Synthesis Architecture of Reducing VCO Stages," *IEEE Trans. on VLSI*, vol.13, pp. 201-210, Feb., 2005.

Part II.C: The Integer-Flying-Adder Architecture

Idea:

Make PLL programmable (Δ can be adjusted now)



Get ride of fractional number

→ **Eliminate the spurs**

This could be very useful for certain applications (where spurious tones are concerns).

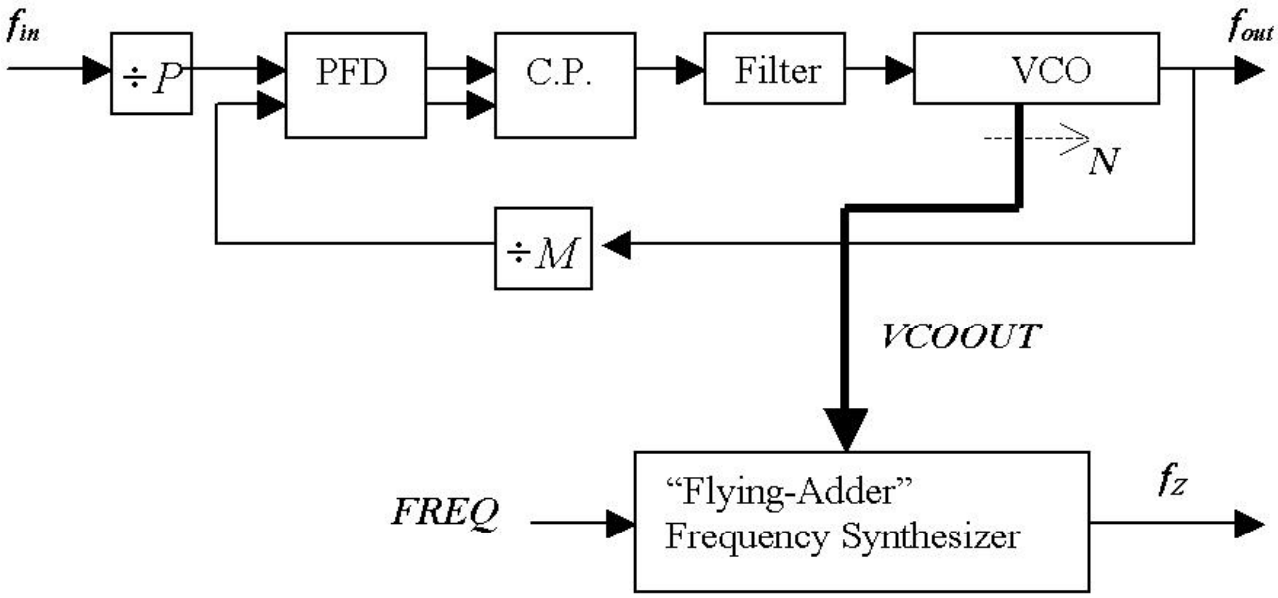
Part II.C: The Integer-Flying-Adder Architecture

Using two integers to approximate a real number

$$FREQ = T/\Delta = 1/(f*\Delta) = ((f_{in}*N)/(f*P)) *M$$

Using two integers, *FREQ* and *M*, to approximate a real number *f* .

Under conditions: $2 \le FREQ < 2N, \quad M_1 \le M \le M_2$



Part II.C: The Integer-Flying-Adder Architecture

The algorithm to search the best control parameters

For an asked f, we need to find the M and FREQ for use, the algo. follows

```
error_min = very_big_number
    for ( M1≤M≤M2 ) {
        freq = ((fin*N)/(f*P))*M
        error = min( freq-floor(freq),  ceiling(freq)-freq )
        if (error < error_min ) {
                error_min = error
                Mbest = M
                if (freq – floor(freq)) < 0.5 {
                        FREQ = floor(freq)
                }
                else {
                        FREQ = ceiling(freq)
                }
        }
    }
```

Part II.C: The Integer-Flying-Adder Architecture

The frequency error

- For the problem of using two integers to approximate a real number, a = q/p, the accuracy can reach any degree desired if all the integers can be used. The algorithm: continuous fraction approximation.

- For any real world problem, such as in this case, this is not true. We can only use integers of: $2 \le FREQ < 2N, \quad M_1 \le M \le M_2.$

- Thus, there is an issue of frequency error. The error upper bound:

$$|T-T'| / T = r*\Delta/T$$
$$\le (1/2) * ((fin*N)/(f*P)) / (((fin*N)/(f*P))*M)$$
$$= 1/(2*M)$$
$$\le 1/(2*M_1)$$

Part II.C: The Integer-Flying-Adder Architecture

The frequency error: a math problem

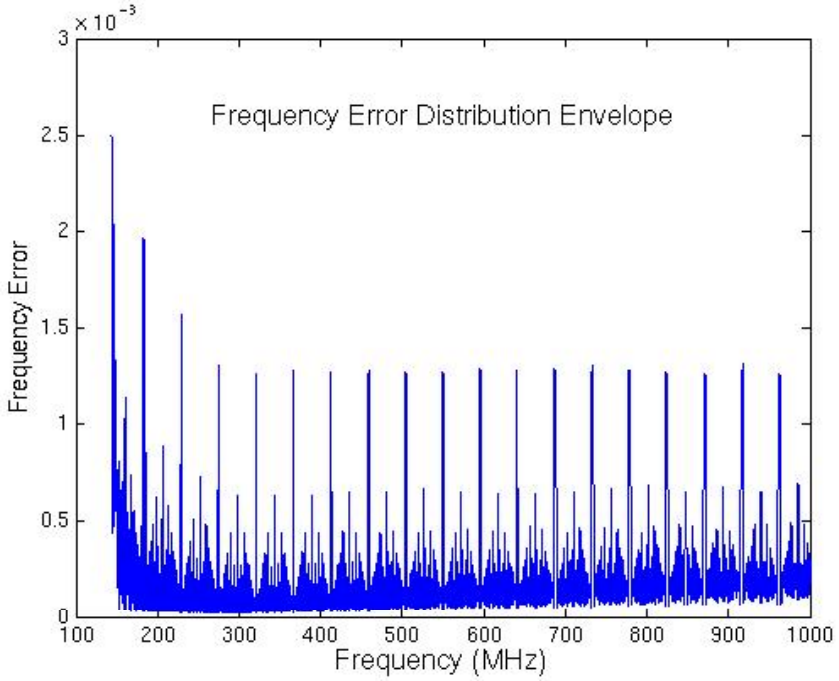
- **Given a real number: a**
- **And the integers can be used: $q1 \le q \le q2$ and $p1 \le p \le p2$**
- **To approximate: $q/p = a$,**
- **Find the q and p? What is the approximation error? Is there an analytic form?**

Part II.C: The Integer-Flying-Adder Architecture

Error Distribution Envelope

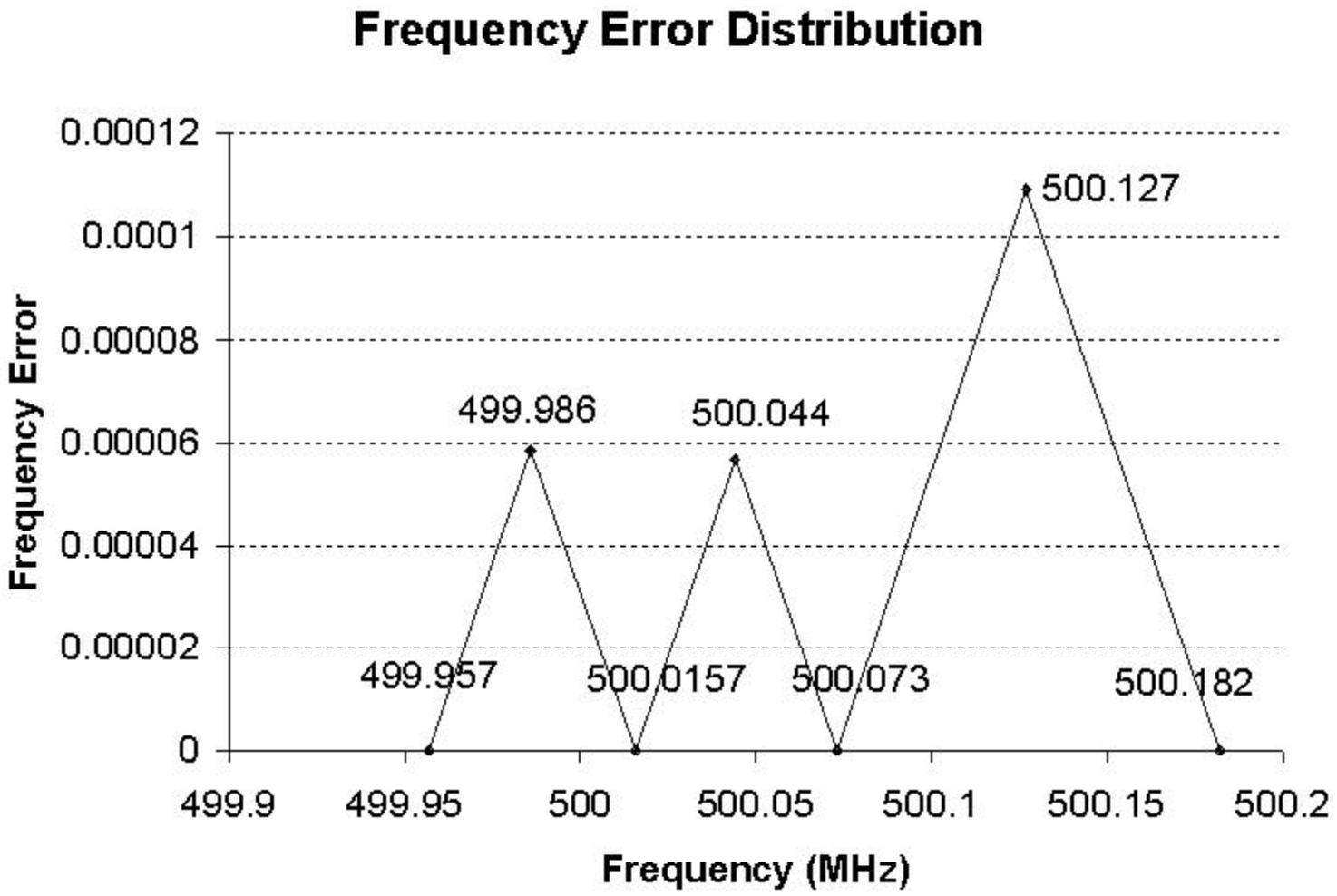
```
for (2<=F<=64) {
    for (M_1<=M<=M_2) {
        F-M-seq(index) = M/F
    }
}

foreach M/F in F-M-sorted-seq(index) {
    F-M_curr = M/F
    p_max = 2/(F-M_curr + F-M_prev)
    e_max = (F-M_curr - F-M_prev)/( F-M_curr +F
    F-M_prev = F-M_curr
}
```



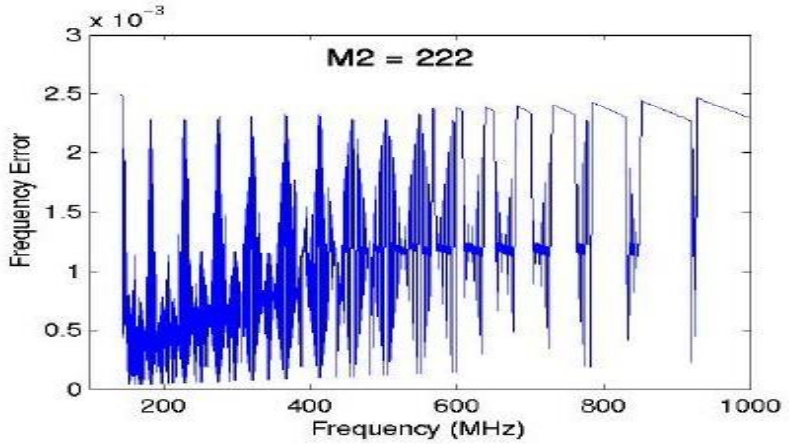
Part II.C: The Integer-Flying-Adder Architecture

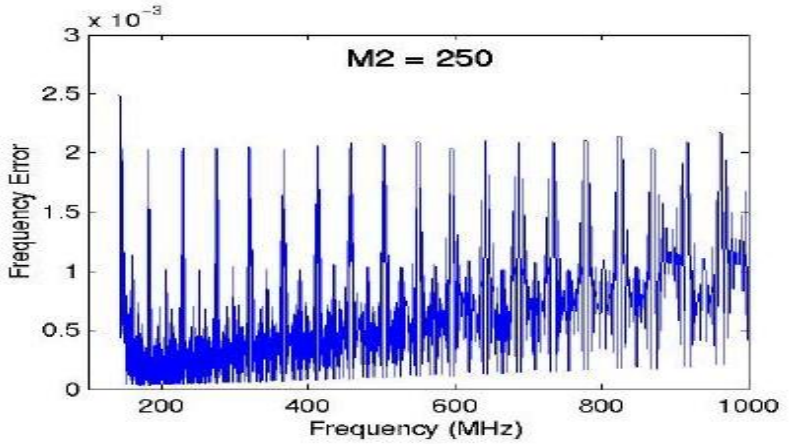
Error Distribution Envelope: zoom in

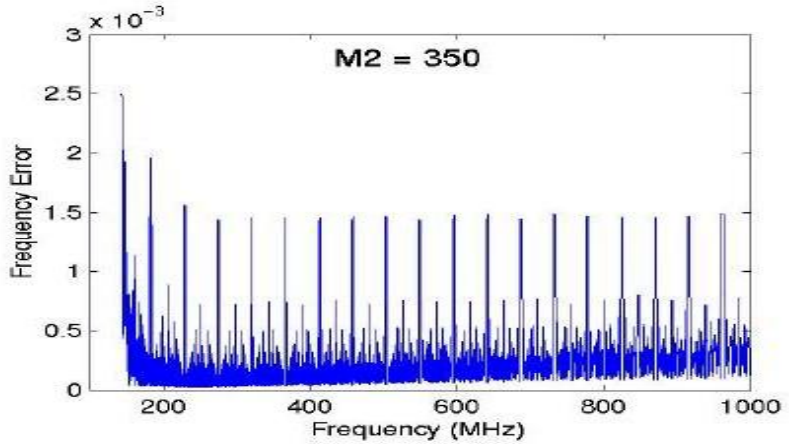


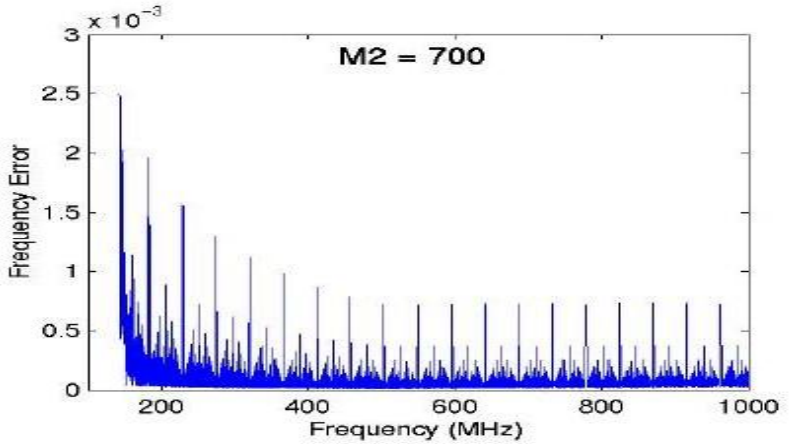
Part II.C: The Integer-Flying-Adder Architecture

The effect of M_2 on the error distribution envelope

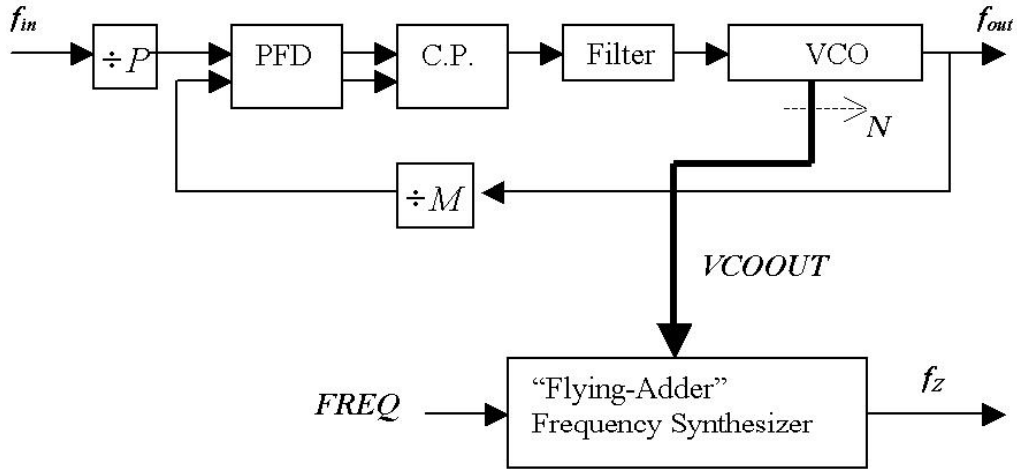








Part II.C: The Integer-Flying-Adder Architecture



- $f_z/f_{in} = (M*N)/(P*FREQ)$ => lots of possibility

Integer-Flying-Adder:

L. Xiu, Z. You, "A new frequency synthesis method based on Flying-Adder architecture," ***IEEE Trans. on Circuit & System II*, March, 2003**

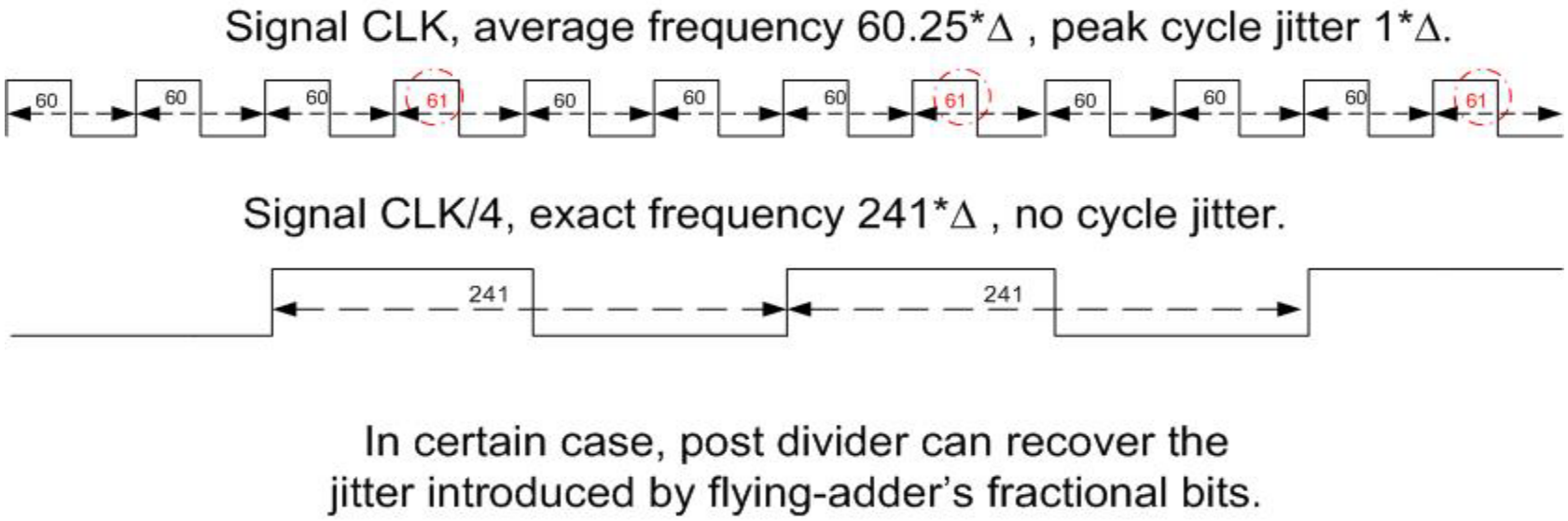
Part II.D: The Post Divider Fractional Bit Recovery (PDFR)

- When fractional number is used in FREQ, there is periodical carry-in from the fractional part to integer part → cycle prolong → spurs.

- However, when a post divider is used, the carry-in can be recovered in certain case → no spurs in output clock.

- This technique is referred as Post Divider Fractional Bit Recovery (PDFR).

Part II.D: The Post Divider Fractional Bit Recovery (PDFR)



- **FREQ = 60.25, => the frequency of output signal: FREQ* Δ = 60.25 * Δ**
- **Periodic fractional bit carry-in => cycle prolong 1* Δ, once every four cycles (since fractional part is 0.25).**
- **However, if we choose post divider M = 4, then CLK/4 = 241 * Δ**
- **Signal CLK/4 doesn't have cycle prolong of 1* Δ => recovered!**

Part II.D: The Post Divider Fractional Bit Recovery (PDFR)

- In real applications, for a given input reference *fr*, need to generate output frequency of *fo*, then:

  $$fr / fo = FREQ * P * M / (N * K)$$

- Without PDFR, FREQ can only take integer value between $2 \leq FREQ < 2K$ to avoid cycle prolong.

- With PDFR, many fractional numbers can be used as long as the fraction's inverse is one of the factors of M.

- For example, if M = 16, then its factors are: 2, 4, 8, 16. Thus, fractional numbers 0.5, 0.25, 0.75, 0.125, 0.875, 0.0625 and 0.9375 can be used in FREQ.

- Thus, resulting in many more frequencies.

Part II.D: The Post Divider Fractional Bit Recovery (PDFR)

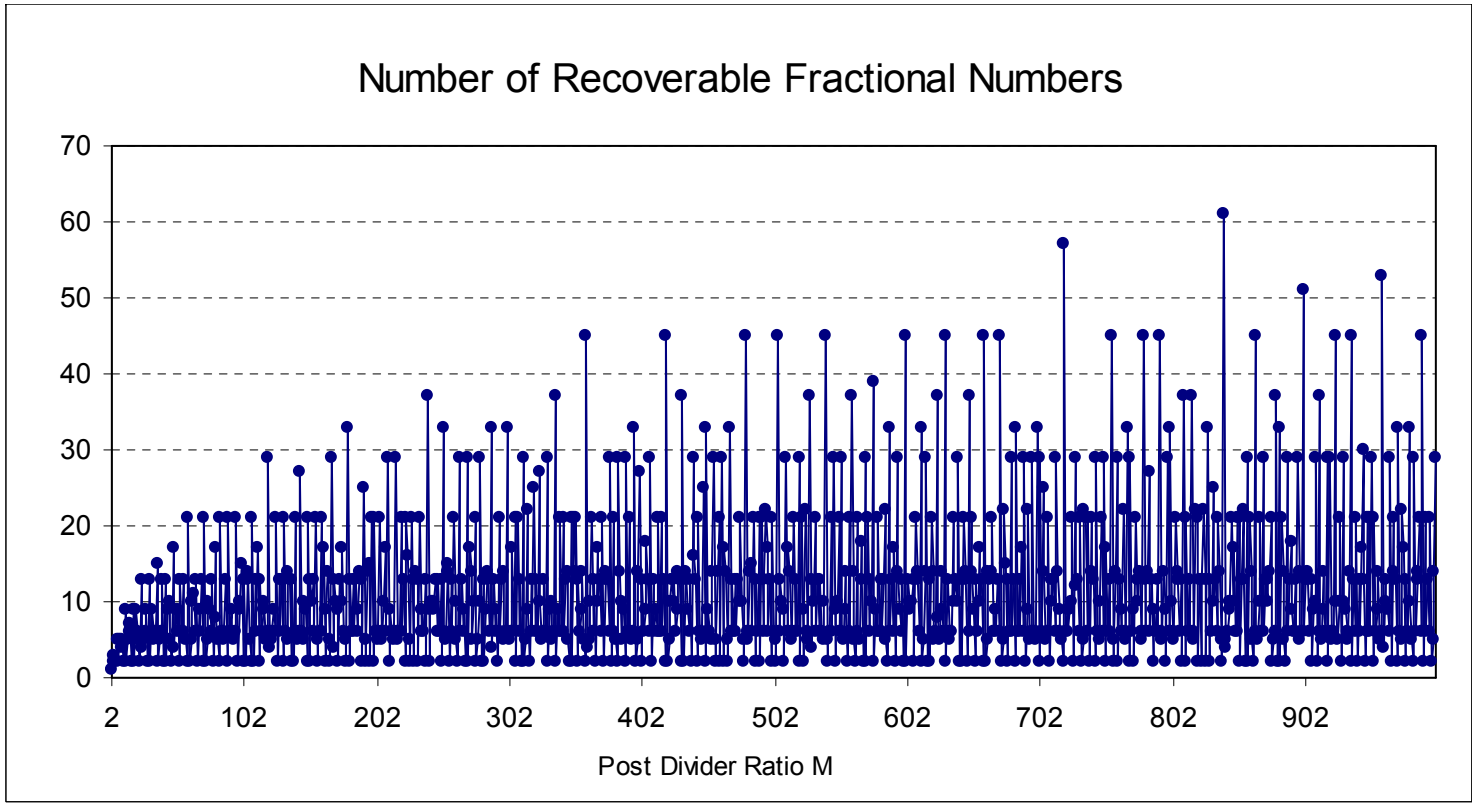
- The new frequencies added by the recoverable fractions lie between: $f_{o1} = C/FREQ$ and $f_{o2} = C/(FREQ+1)$.

- Furthermore, these frequencies are distributed almost linearly between f_{o1} and f_{o2}, proportional to the magnitude of r.

$$f_o = \frac{C}{FREQ + r} = \frac{C}{FREQ(1 + r/FREQ)}$$

$$= \frac{C}{FREQ}(1 - r/FREQ + (r/FREQ)^2 - \ldots)$$

$$\approx f_{o1}(1 - r/FREQ)$$



Part II.D: The Post Divider Fractional Bit Recovery (PDFR)

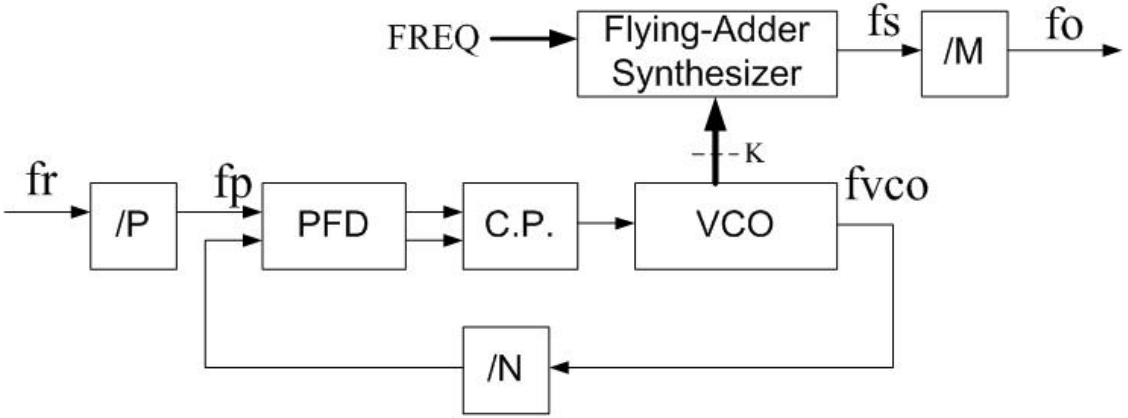
- The following equation can be used as the base for a PDFR algorithm, which drives the Integer-Flying-Adder PLL to its maximum potential.

$$fr / fo = FREQ * P * M / (N * K)$$

- A reference:
  L. Xiu, "A Flying-Adder On-Chip Frequency Generator for Complex SoC Environment," IEEE TCAS-II, VOL. 54, NO. 12, DECEMBER 2007

Part II.E: The summary of Flying-Adder distinguished features

- **fp = fr / P**
- **fvco = N * fr / P**
  Tvco = P/ (N*fr)
- **Δ = Tvco / K = P / (N * fr * K)**
- **Ts = FREQ * Δ**
- **fo = fs / M**

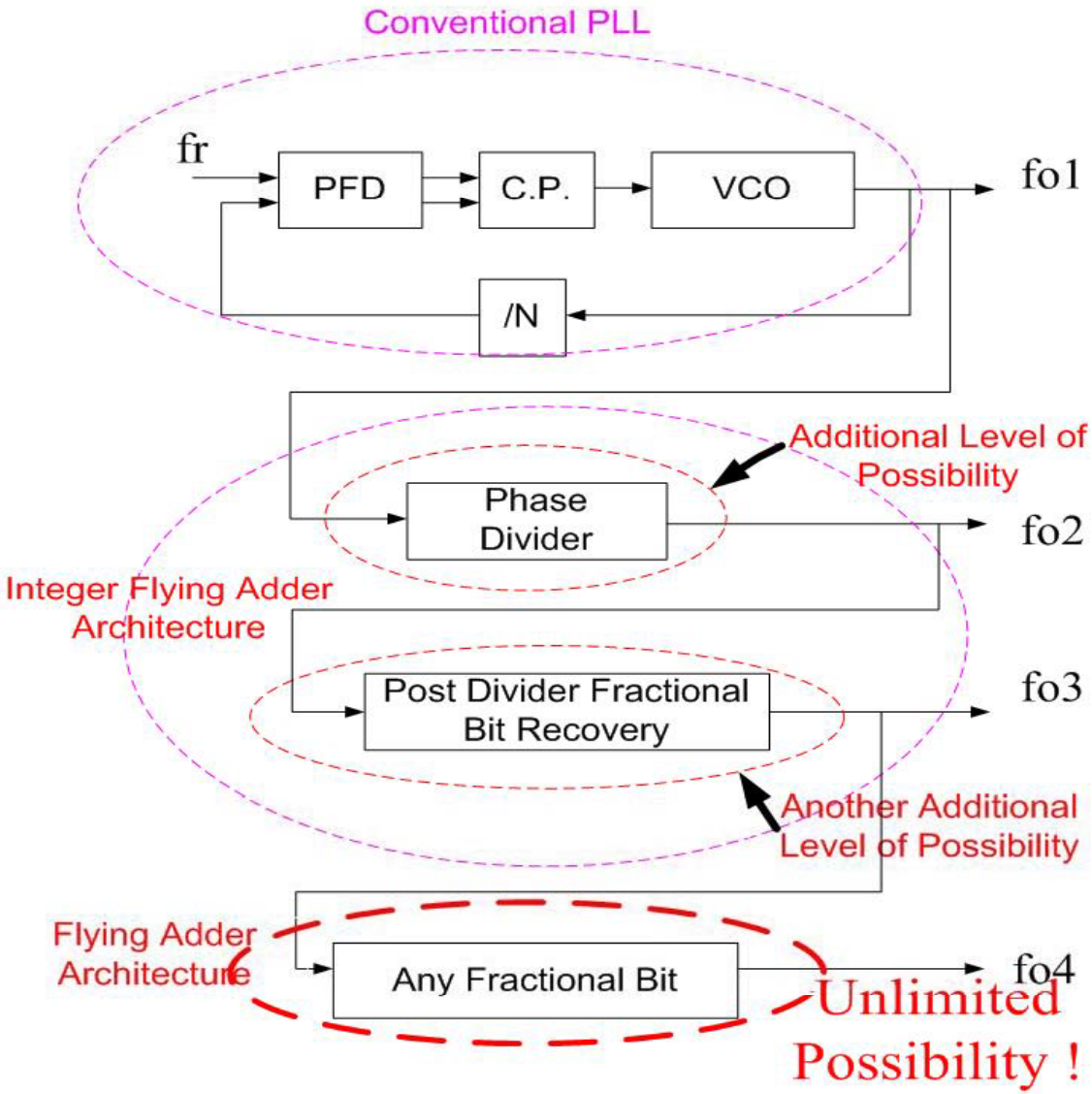


fr:	reference frequency
fp:	frequency after pre divider
fvco:	VCO frequency
fs:	frequency at synthesizer output
fo:	output frequency
FREQ:	frequency control word for the synthesizer
K:	number of VCO output phases

- **This is the equation, five variables can be adjusted!**

fo / fr = (N * K) / (FREQ * P * M)

Part II.E: The summary of Flying-Adder distinguished features



- What is good ?
- The flexibility of generate arbitrary frequency.
- And, the frequency-switching can be achieved instantly.

Part II.E: The summary of Flying-Adder distinguished features

- Two operating modes: integer-Flying-Adder and Fix-VCO-Flying-Adder.

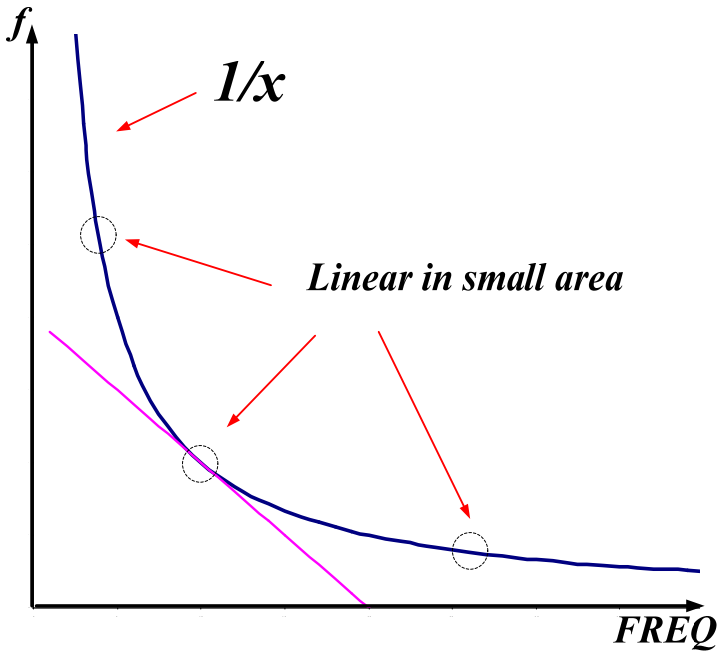
- Integer-Flying-Adder mode:
  - No fractional number used in FREQ
  - *fo / fr = (N * K) / (FREQ * P * M)*
  - VCO frequency can be adjusted

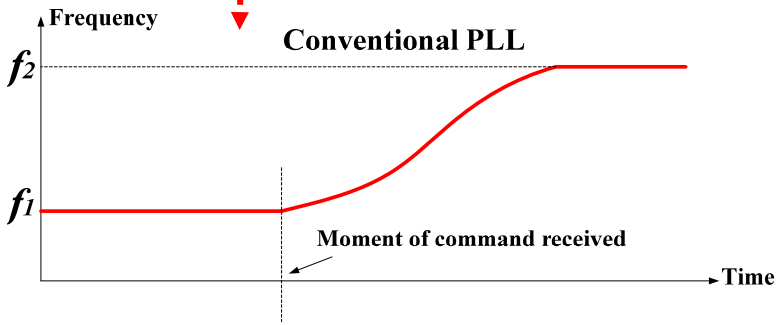
- Fix-VCO-Flying-Adder mode:
  - Fractional number is used
  - VCO is fixed at a given frequency
  - *fo = C/FREQ*, C is a constant
  - Frequency resolution: $\delta f = -2^{-p} * \Delta * f^2$

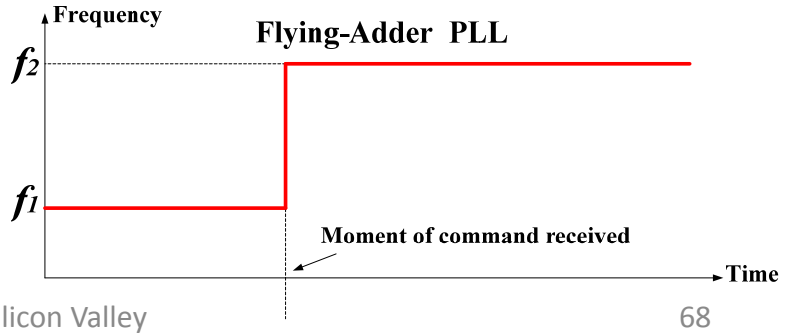
Part II.E: The summary of Flying-Adder distinguished features

The distinguished features of Fix-VCO-Flying-Adder mode:

- Precisely describable frequency transfer function: 1/x
- Linear transfer function when in small range
- Instantaneous response speed
- Very fine frequency resolution: sub-Hz
- The PLL/VCO design is much simplified since the frequency is fixed.



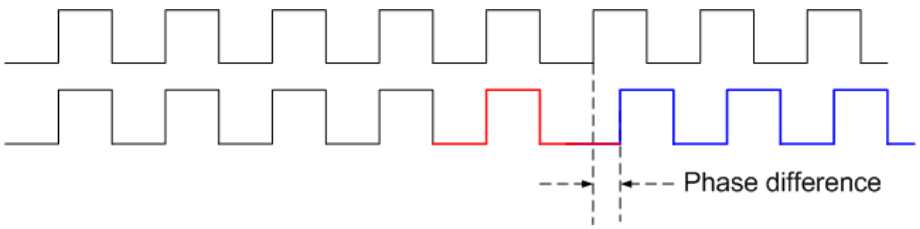




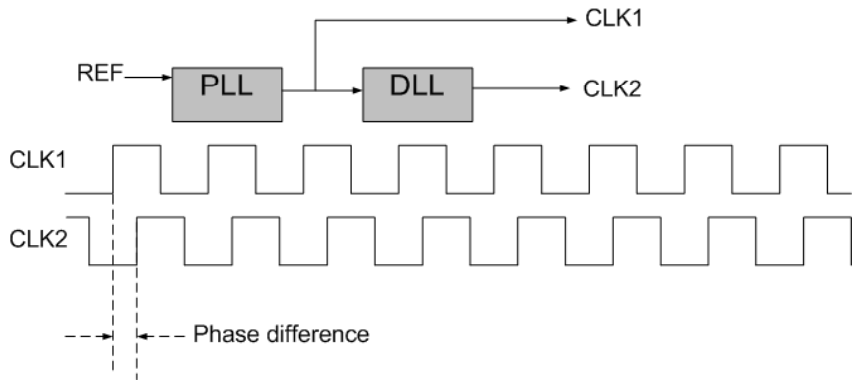
Part II.F: The phase synthesis

- In many applications, there is phase adjustment requirement on clock signal.
- The phase movement can be classified into two cases: one clock signal and two clock signals.
- Two clock signals: the phase difference is between the two signals.
- One clock signal: the phase movement is relative to itself.

One clock signal phase synthesis



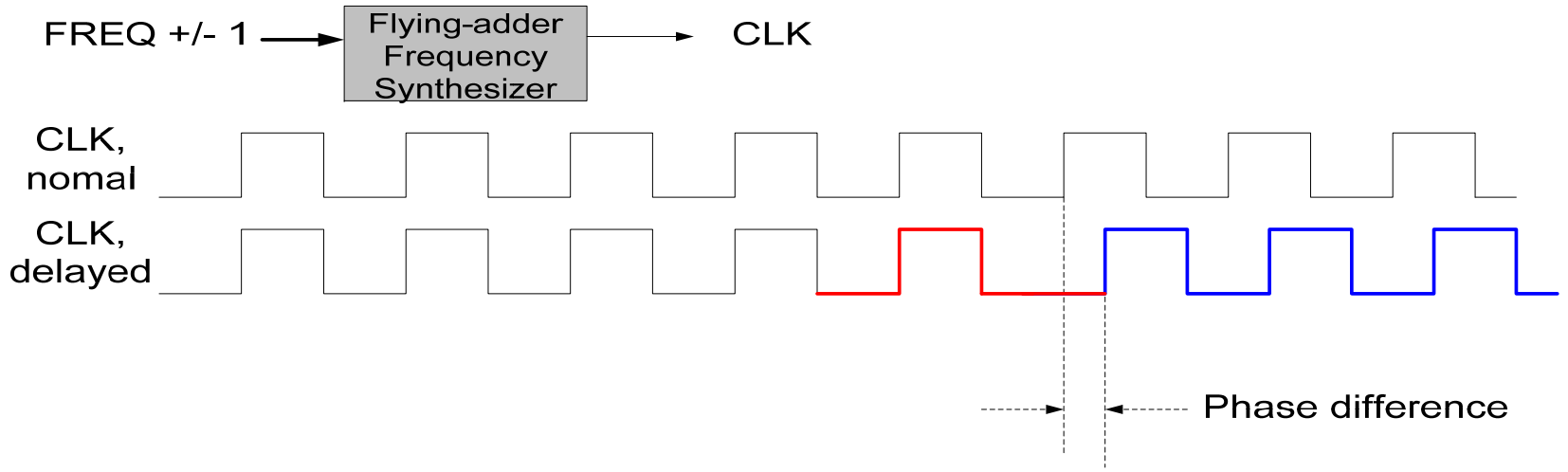
Two clock signals phase synthesis



Part II.F: The phase synthesis

Using Flying-Adder phase synthesis: one clock signal

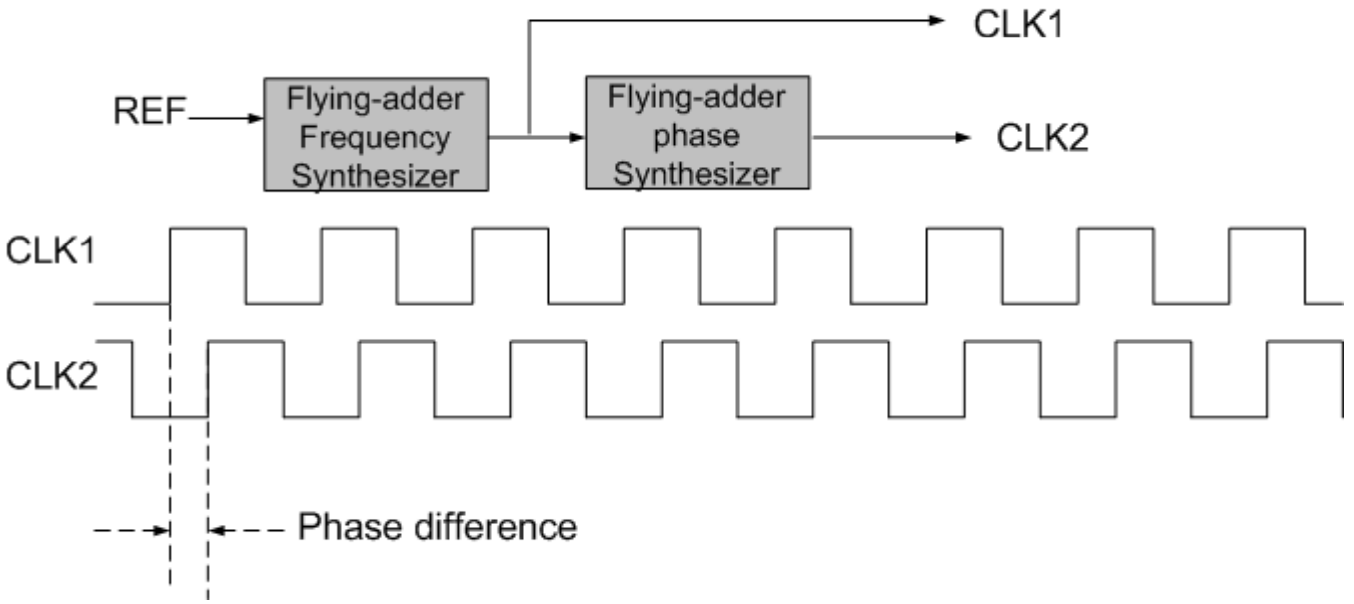
- The CLK signal's phase can be moved forward or backward.
- This is one-time-shot, or prolong/shorten one cycle, then back to normal.
- Implementation-wise, this can be easily achieved by flying-adder (by doing FREQ +/- 1 once).



Part II.F: The phase synthesis

Using Flying-Adder phase synthesis: two clock signals

- Flying-Adder architecture can be used as well.
- The resolution of the phase movement is Δ.
- The phase synthesizer can be built together with the "flying-adder" frequency synthesizer.



L. Xiu, Z. You, "A Flying-Adder architecture of frequency and phase synthesis with scalability," *IEEE Trans. on VLSI*, vol.10, pp. 637-649, Oct., 2002.