



*The Abdus Salam
International Centre for Theoretical Physics*



1977-9

**First ICTP Regional Microelectronics Workshop and Training on
VHDL for Hardware Synthesis and FPGA Design in Asia-Pacific**

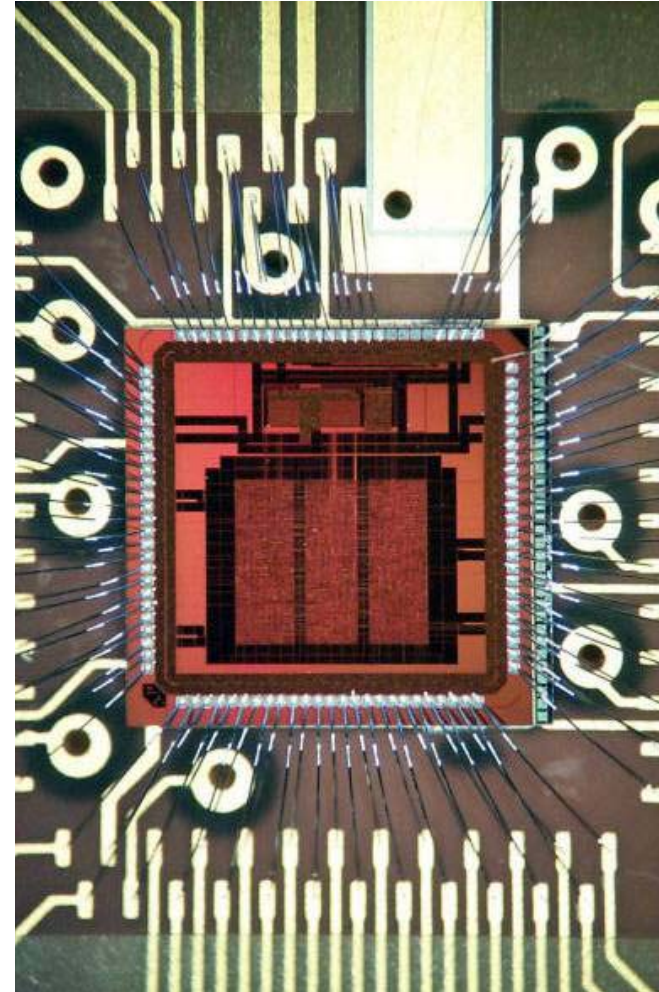
16 June - 11 July, 2008

Phase-Locked Loops.

Paulo Moreira
*PH ESE ME Division
CERN
CH-1211 Geneva 23
SWITZERLAND*

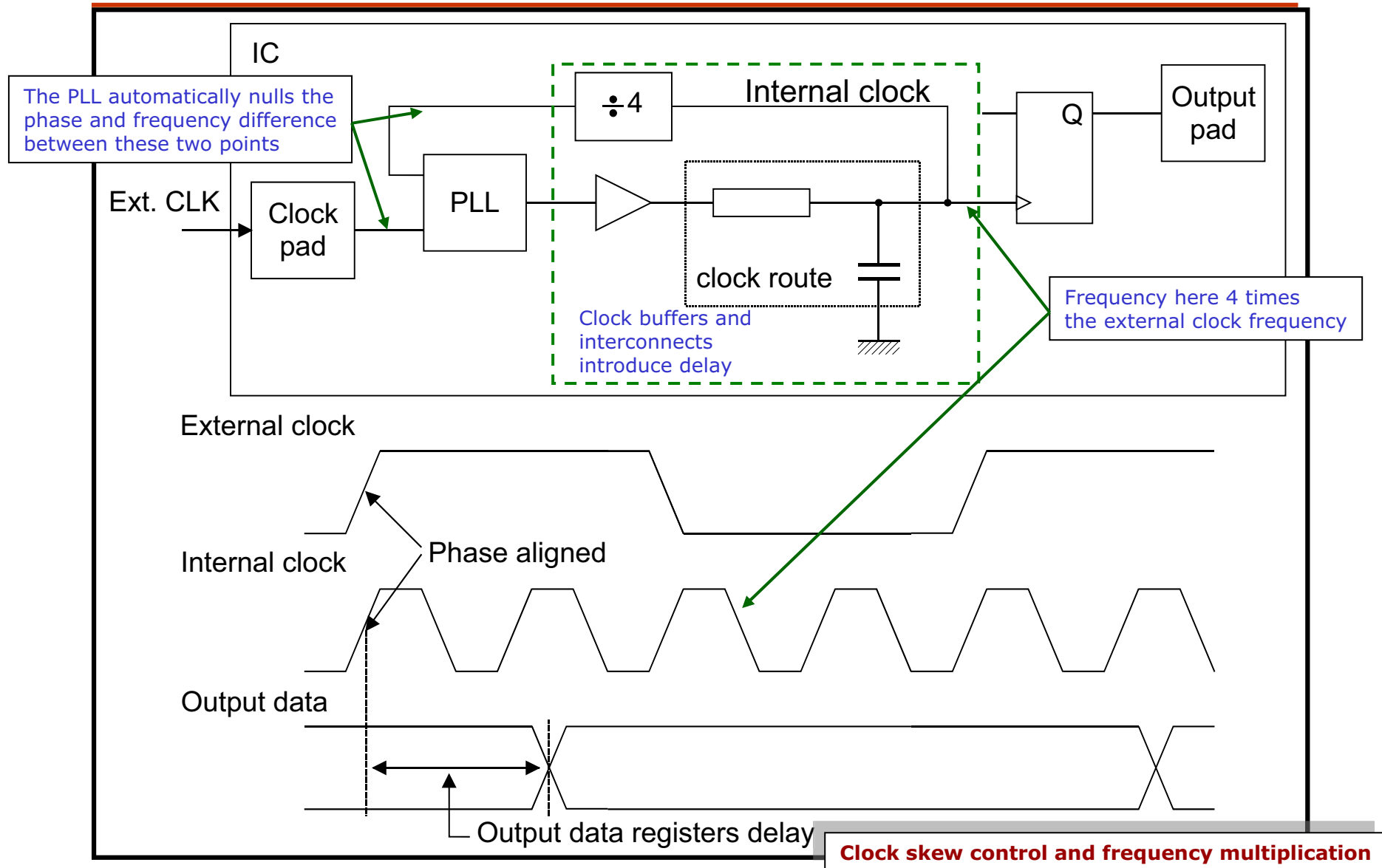
Outline

- Introduction
- Transistors
- The CMOS inverter
- Technology
- Scaling
- Gates
- Sequential circuits
- Storage elements
- Phase-Locked Loops
 - PLL overview
 - Building blocks:
 - PLL analysis:
 - PLL simulation with Verilog
- Example

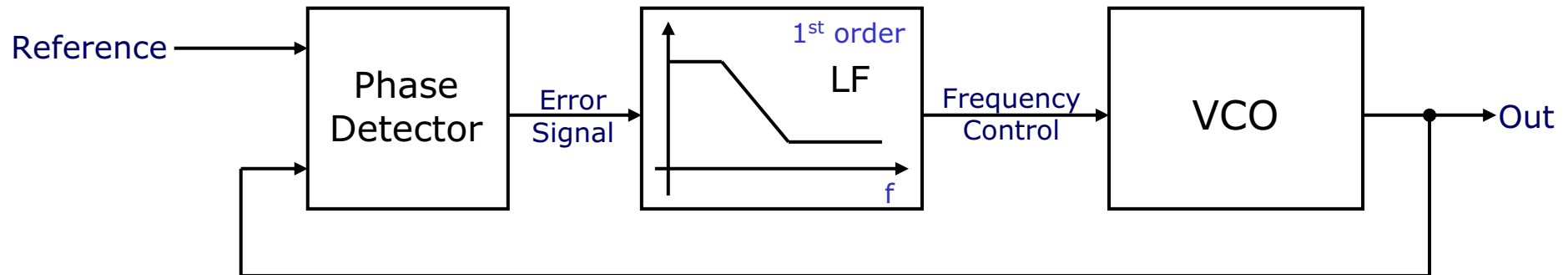


Complete set of slides and notes on DLLs and PLLs can be found in:
<http://paulo.moreira.free.fr/microelectronics/padova/padova.htm>

Why Phase-Locked Loops?



PLL Block Diagram



- **Phase-Locked Loop functional blocks**

- **Phase Detector (PD):**

- Compares the phase of the reference signal to the VCO phase
- Depending on the type, produces an error signal:
 - Proportional to the phase difference between the input and output phases;
 - Gives just an indication on the sign of the phase error (bang-bang detector).
- Phase detectors can be also frequency sensitive; in this case they are called Phase-Frequency Detectors (PFD).

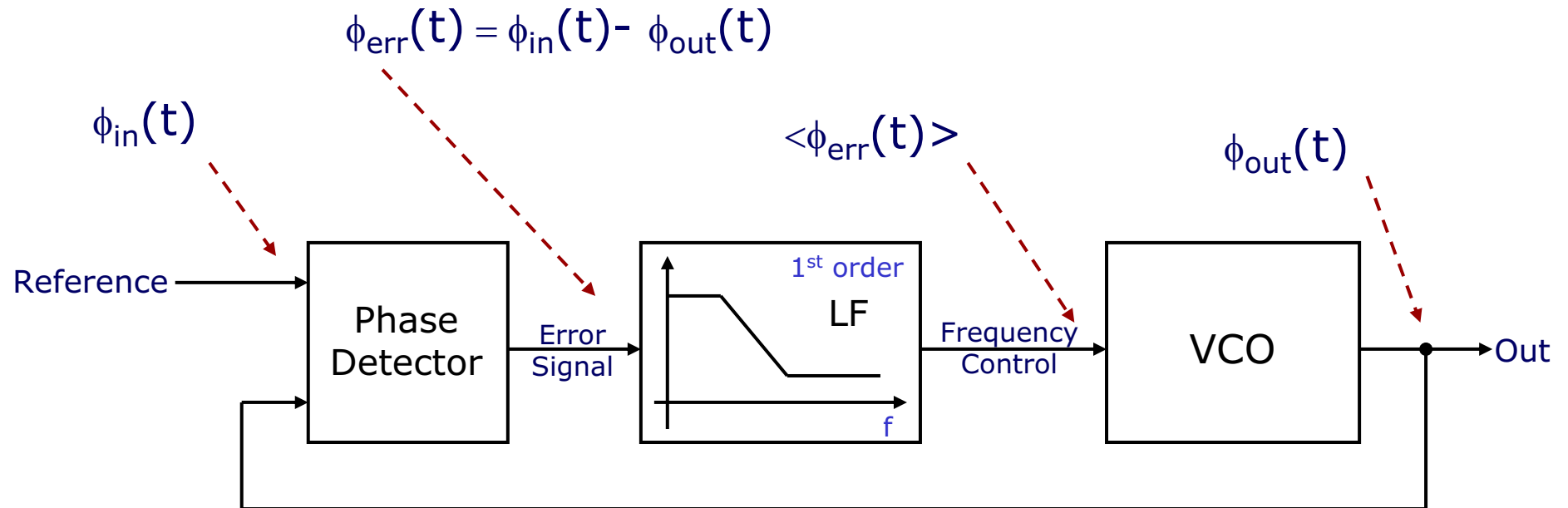
- **Loop filter (LF):**

- Eliminates the high frequency components of the error signal
- Introduces a loop-stabilizing zero
- It can be implemented as:
 - An RC low-pass filter
 - An active low-pass filter
 - A charge-pump a resistor and a capacitor

- **Voltage Controlled Oscillator (VCO):**

- As the name indicates is an oscillator whose frequency is controlled by a voltage: $f_{out} = F(V_{control})$
- Sometimes the control quantity can be a current. In this case we have a Current Controlled Oscillator (CCO)
- We will assume that the higher the voltage (or the current) the higher the frequency

PLL Basic Operation



Starved Inverter VCO

The VCO is an oscillator

- The oscillation frequency depends on the control voltage
- It is usually modeled as:

$$f(t) = K_{vco} \cdot V_{cnt}(t) + f_0$$

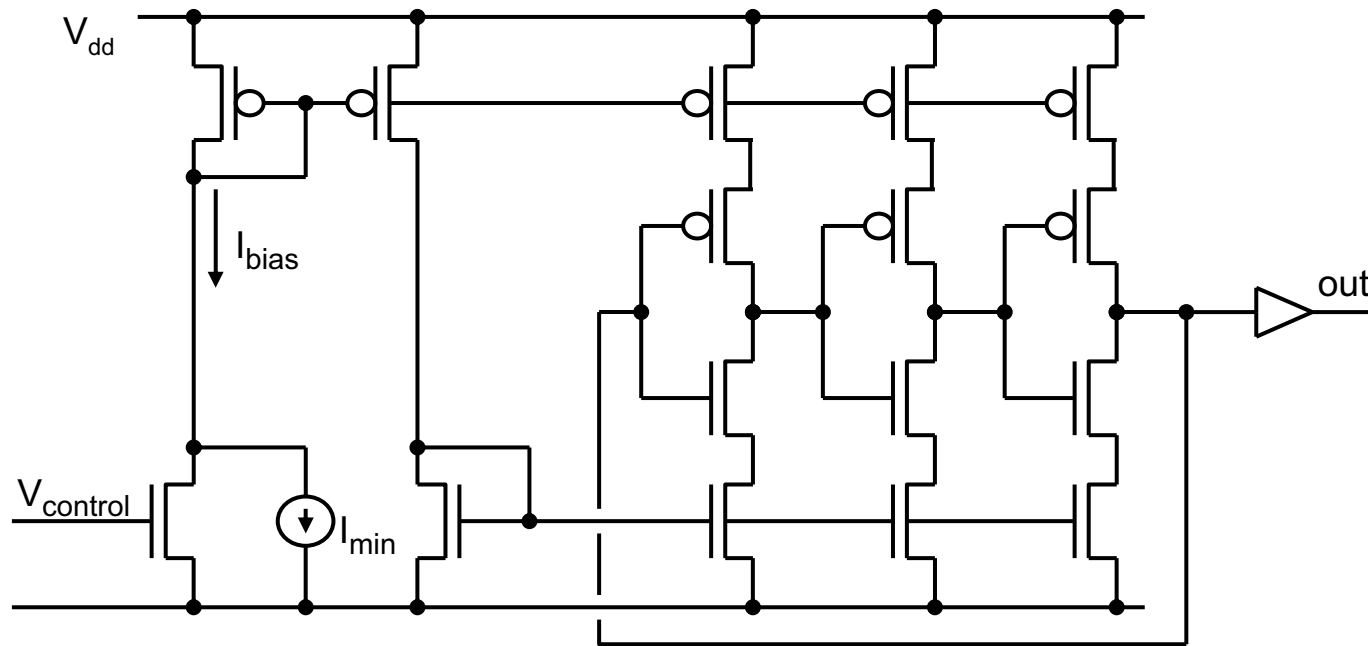
$$\Phi(t) = \int_0^t f(t) dt + \Phi_0$$

$$\Phi(s) = K_{vco} \cdot \frac{1}{s} \cdot V_{cnt}(s)$$

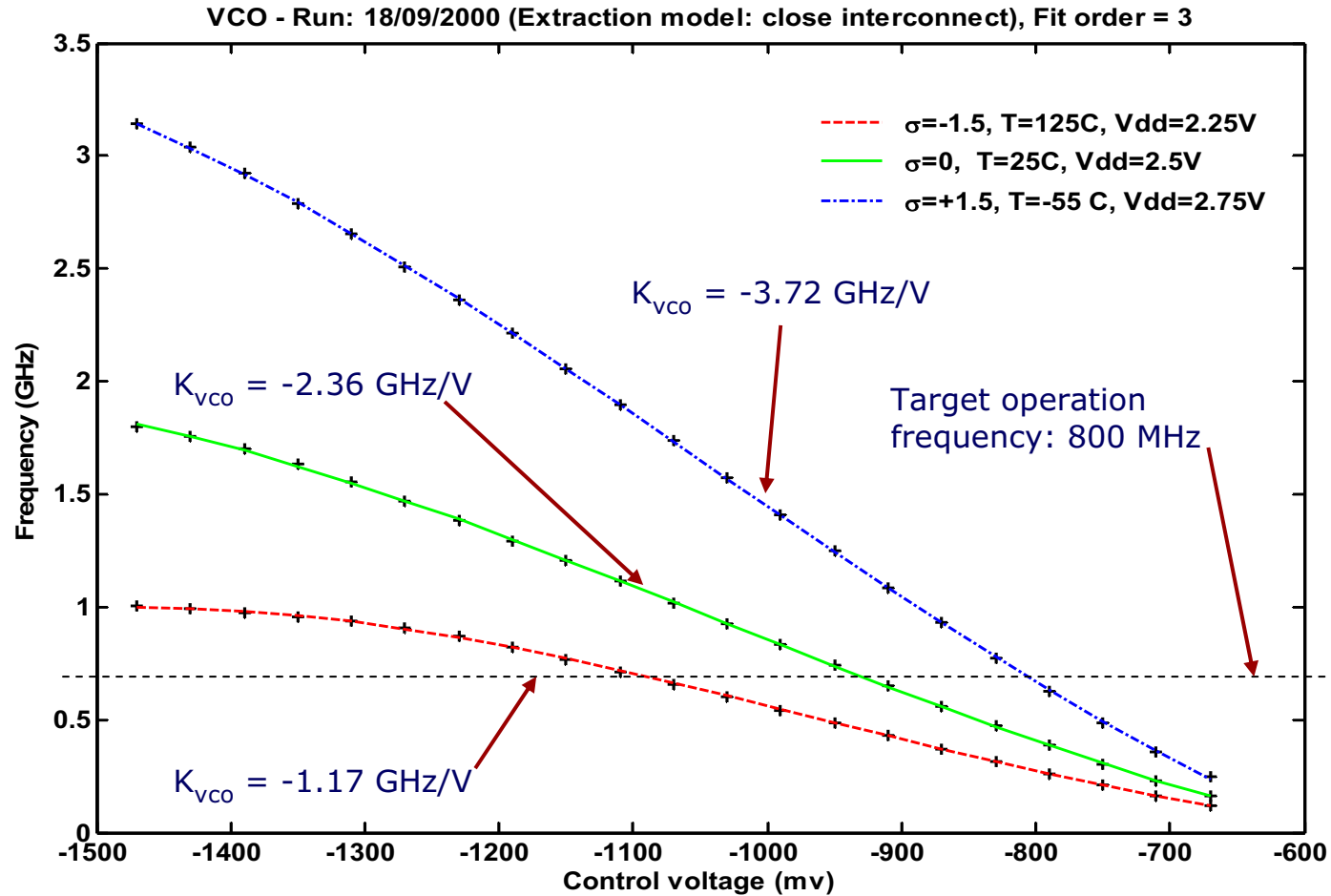
Practical advice:

- An odd-number of inverters is mandatory;
- Always buffer the output signal;
- Ensure a minimum oscillation frequency;
- Preferably use a minimum of 3 inverters.

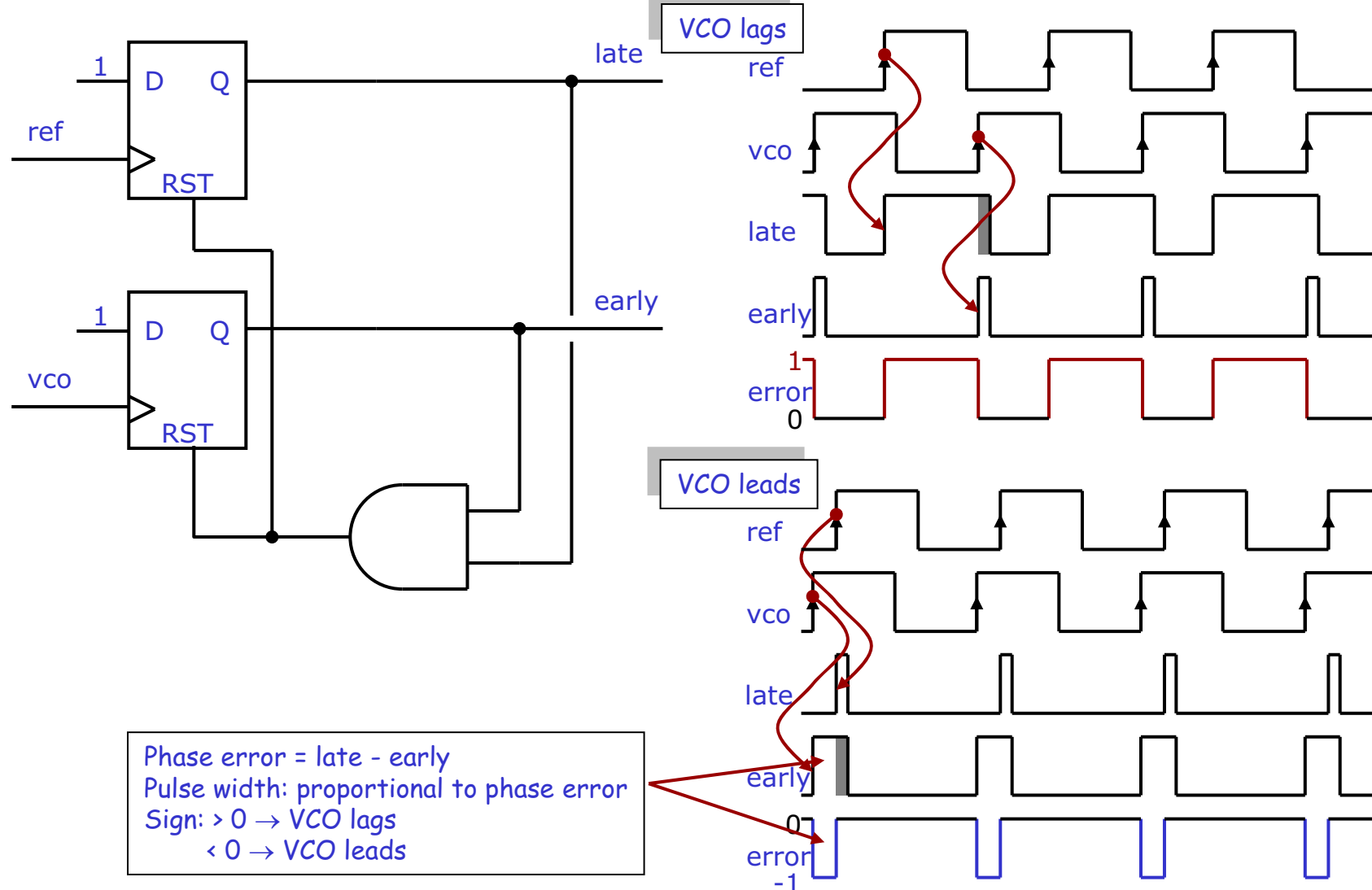
The VCO behaves as a phase integrator



VCO Transfer function

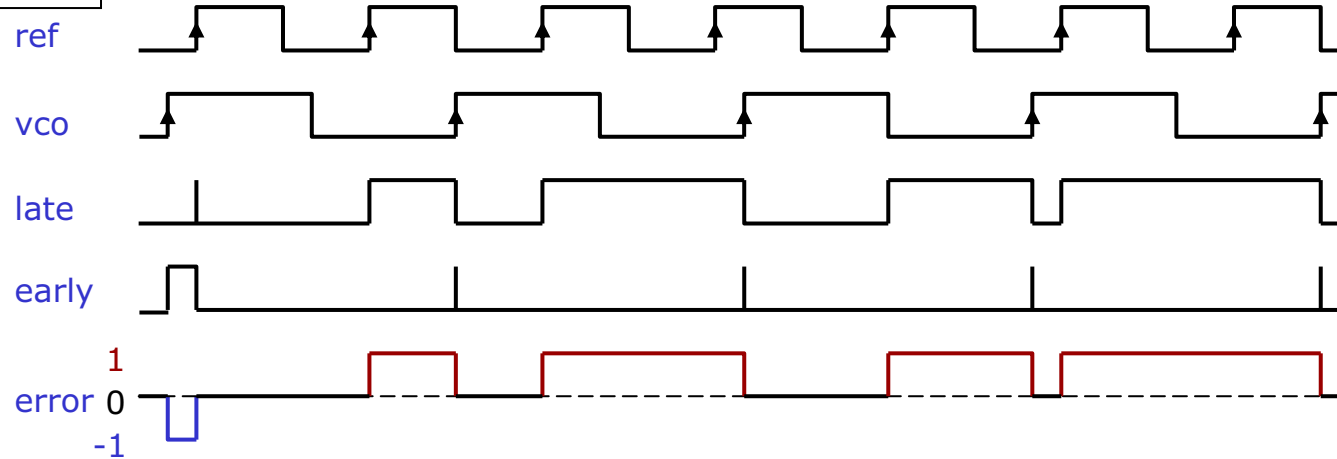


Phase Frequency Detector

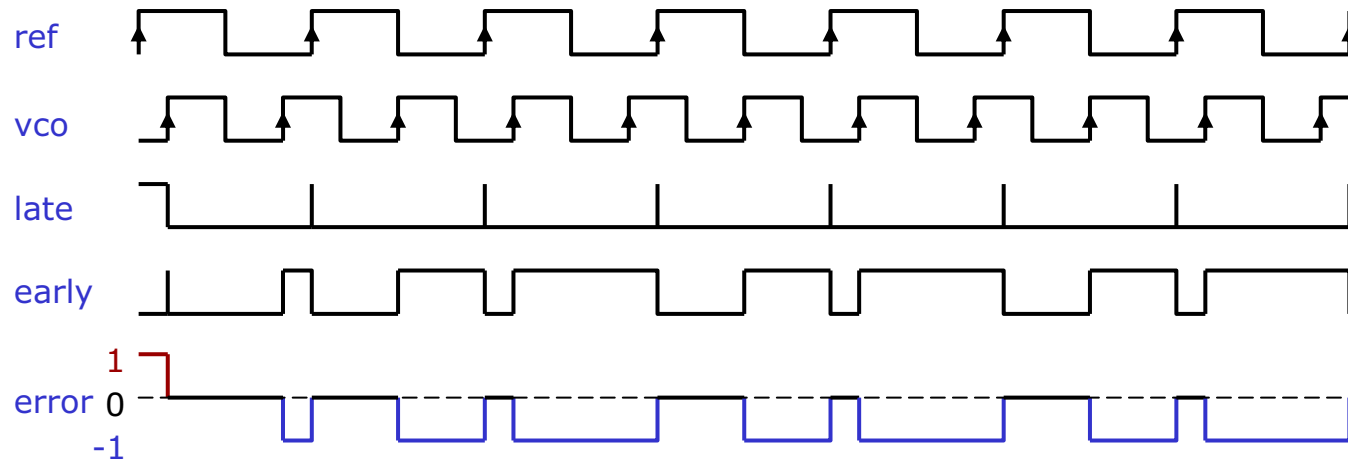


PFD: Frequency sensitivity

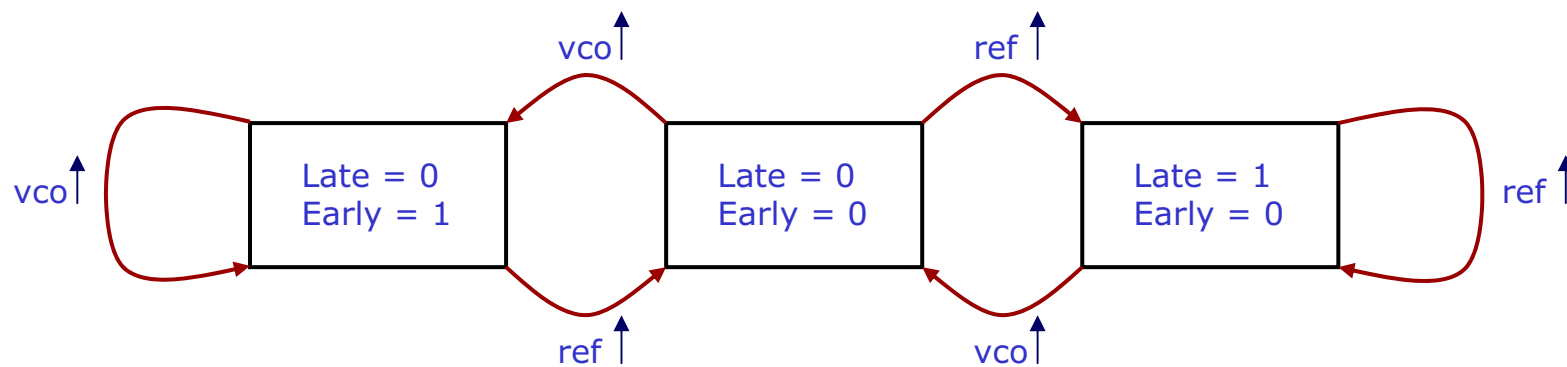
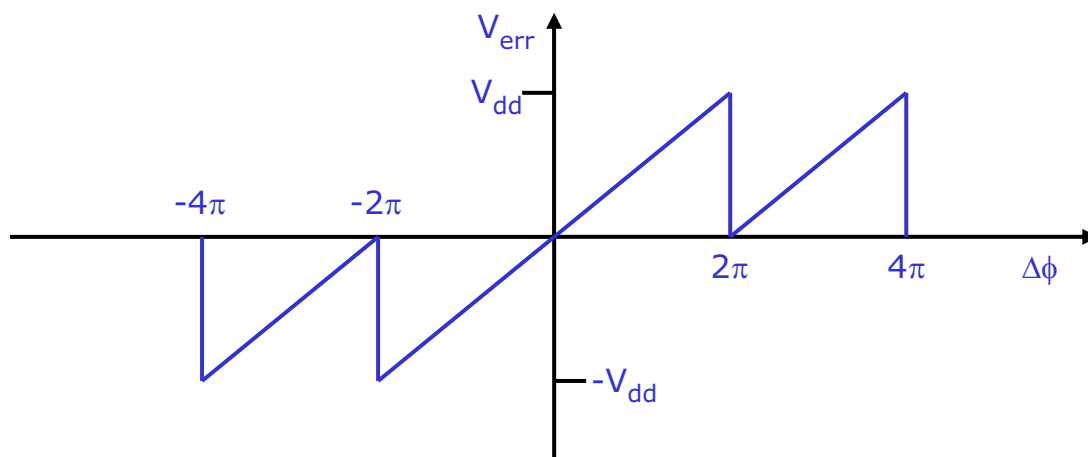
VCO slow



VCO fast



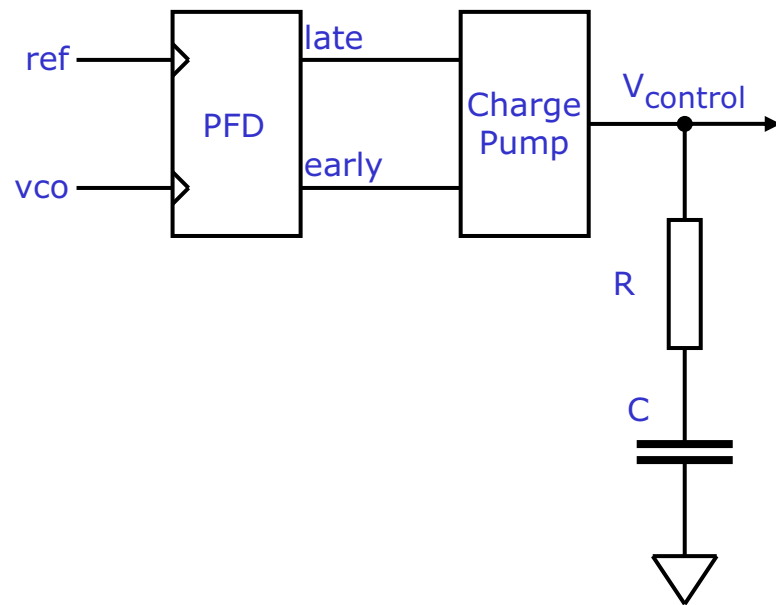
PFD Characteristics



Active Filter: Charge-Pump + RC network

$$\phi_{err,n}(t) = \frac{\phi_{err}}{2\pi} \propto late(t) - early(t)$$

$$\phi_{err,n}(t) \in [-1, 1]$$



$$V_{control}(t) = V_{res}(t) + V_{cap}(t)$$

$$V_{control}(t) = I_{cp} \cdot \left[R \cdot \phi_{err,n}(t) + \frac{1}{C} \int_0^t \phi_{err,n}(t) dt \right] + V_0$$

Integral term controlled by 'C'

Proportional term controlled by 'R'

Current magnitude I_{cp} affects both

Zero at: $f_z = \frac{1}{2\pi \cdot R \cdot C}$

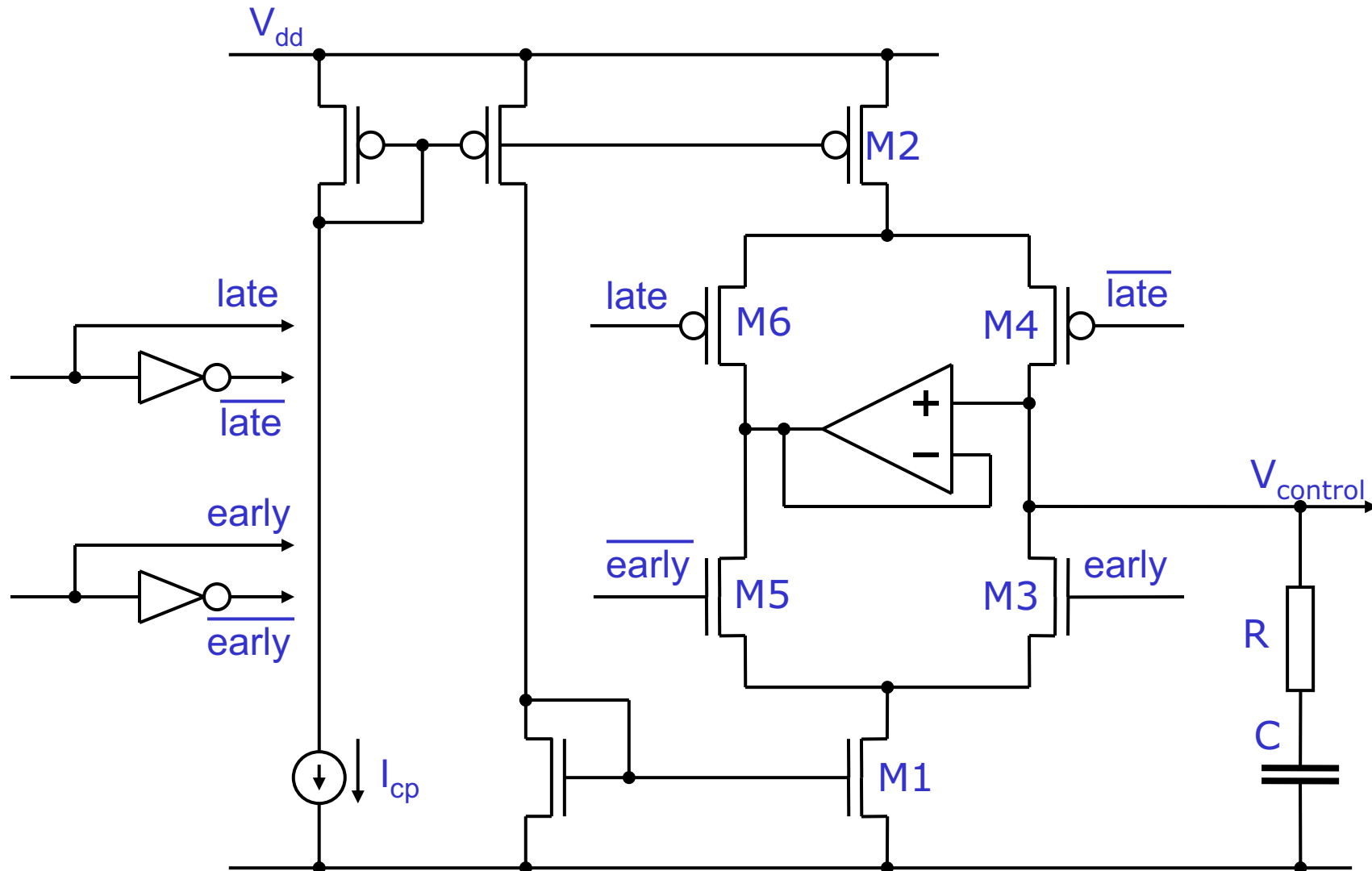
Infinite gain at DC

$$H_{LF}(s) = I_{cp} \cdot \frac{1 + s \cdot R \cdot C}{s \cdot C}$$

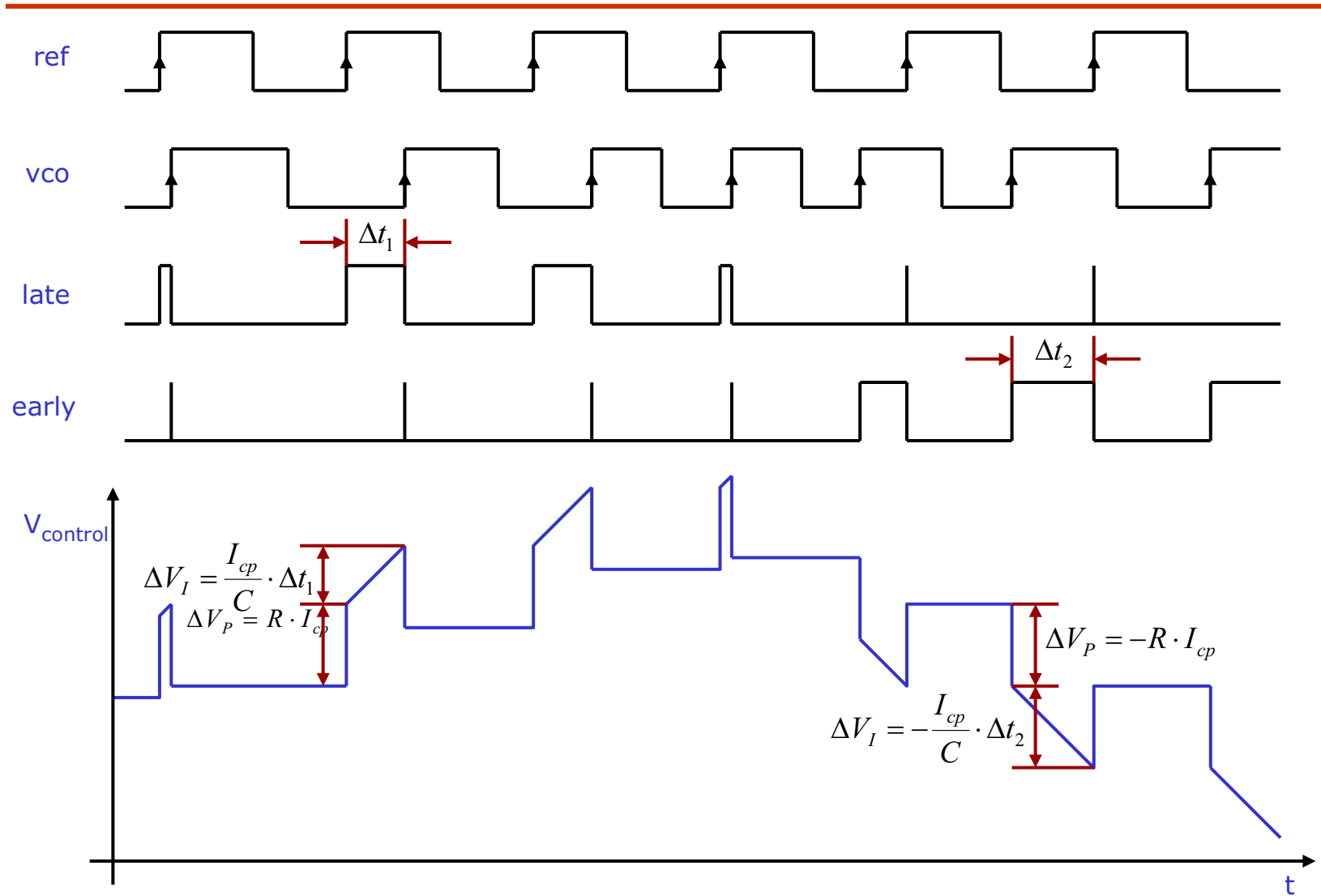
Independent

Pole at the origin

Charge Pump Implementation



Charge Pump Operation



Charge-Pump PLL with PFD

Assume the PLL is locked $\omega_{in} = \omega_{out}$

Phase error

$$\phi_{err}(s) = \phi_{in}(s) - \phi_{out}(s) \quad [1]$$

'ON' time for either 'Late' or 'Early'

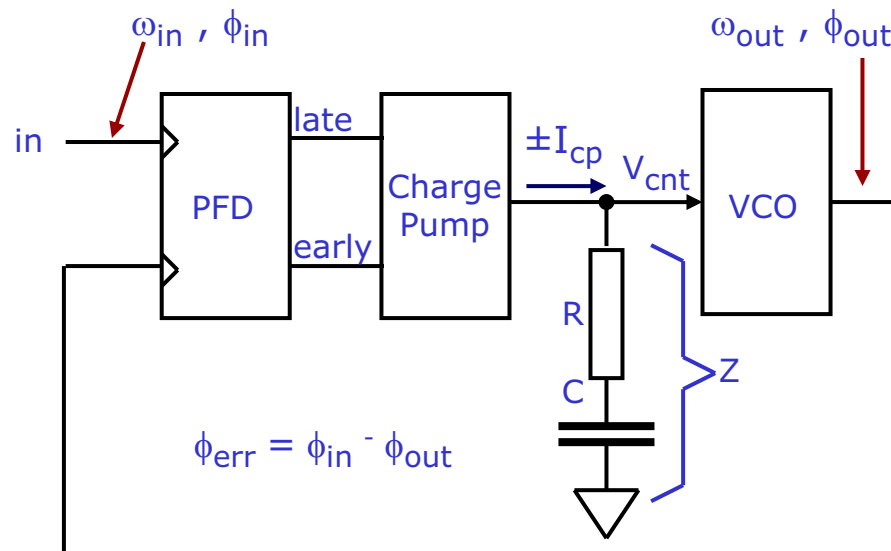
Average current in one cycle

Lumped contributions of:

- Phase-Frequency Detector
- Charge-Pump
- Filter impedance

$$t_{on} = \frac{|\phi_{err}|}{\omega_{in}} \rightarrow Q = I_{cp} \frac{\phi_{err}}{\omega_{in}} \rightarrow i_d = I_{cp} \frac{\phi_{err}}{2\pi} \rightarrow V_{cnt}(s) = I_{cp} \cdot \frac{\phi_{err}(s)}{2\pi} \cdot Z(s) \quad [2]$$

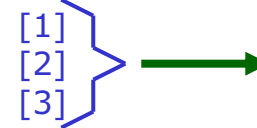
The charge delivered in one cycle is proportional to the phase error ϕ_{err}



VCO phase

$$\phi_{out}(s) = \frac{K_{vco}}{s} \cdot V_{cnt}(s) \quad [3]$$

K_{vco} in rad/(s.V)



$$\frac{\phi_{out}(s)}{\phi_{in}(s)} = H(s) = \frac{K_{vco} \cdot I_{cp} \cdot Z(s)}{2\pi \cdot s + K_{vco} \cdot I_{cp} \cdot Z(s)} \quad [4]$$

A Second Order System

$$Z(s) = R + \frac{1}{s \cdot C} \quad [5]$$

$$\left. \begin{array}{l} [4] \\ [5] \end{array} \right\} \longrightarrow H(s) = \frac{I_{cp} \cdot (1 + R \cdot C \cdot s) \cdot K_{vco}}{2\pi \cdot C \cdot s^2 + I_{cp} \cdot (1 + R \cdot C \cdot s) \cdot K_{vco}} \quad [6]$$

A zero appears in the transfer function:
It is used to compensate the PLL response

[6] can be put in the form:

$$H(s) = \frac{(1 + \tau_z \cdot s)}{\frac{1}{\omega_n^2} \cdot s^2 + \frac{2 \cdot \xi}{\omega_n} \cdot s + 1} \quad [7]$$

The loop is second order

$$\tau_z = R \cdot C$$

Compensating zero
time constant

$$\omega_n = \sqrt{\frac{I_{cp} \cdot K_{vco}}{2\pi \cdot C}}$$

Natural frequency

$$\xi = \frac{R \cdot C}{2} \cdot \omega_n$$

Damping factor

$$K = 2 \cdot \xi \cdot \omega_n = \frac{R \cdot I_{cp} \cdot K_{vco}}{2\pi}$$

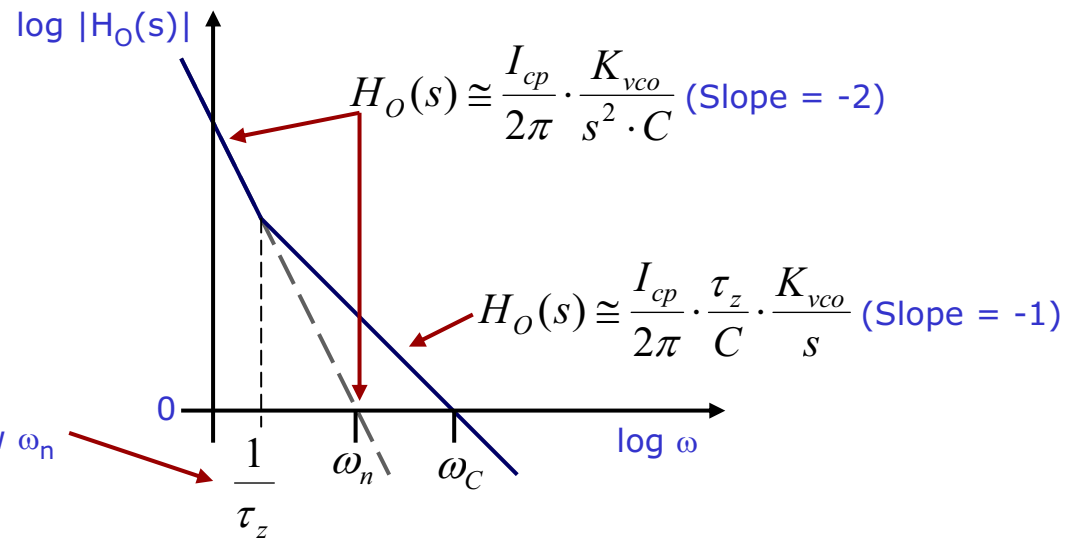
Loop gain

Any two of these
parameters define
the linearized,
time-averaged
behavior of the
PLL

PLL Stability

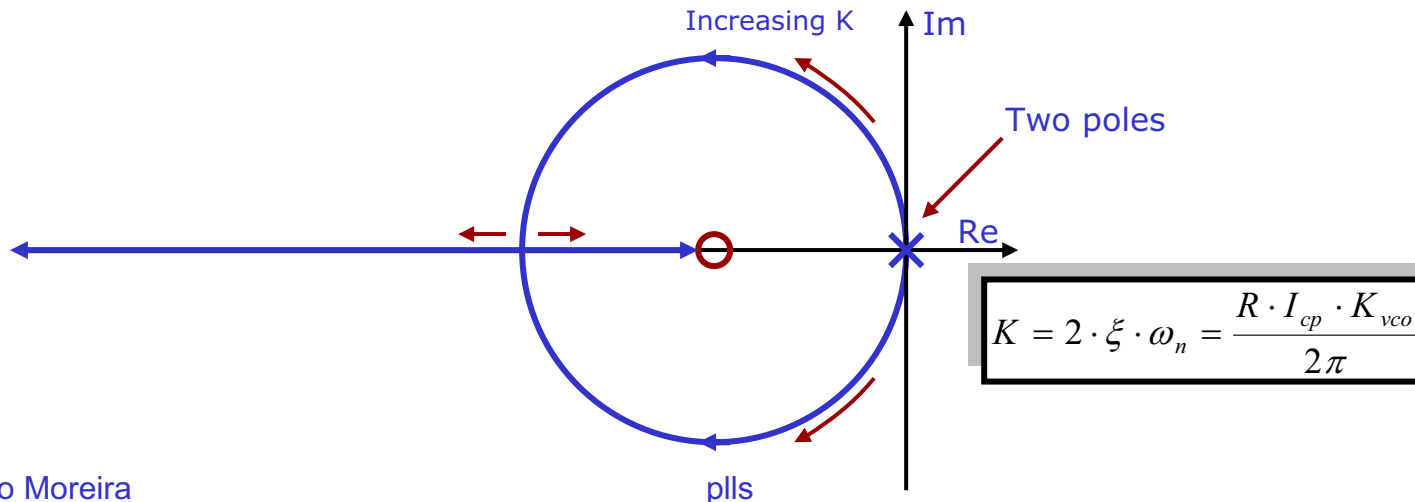
Open-loop transfer function:

$$H_O(s) = \frac{I_{cp}}{2\pi} \cdot \frac{1 + s \cdot \tau_z}{s \cdot C} \cdot \frac{K_{vco}}{s}$$

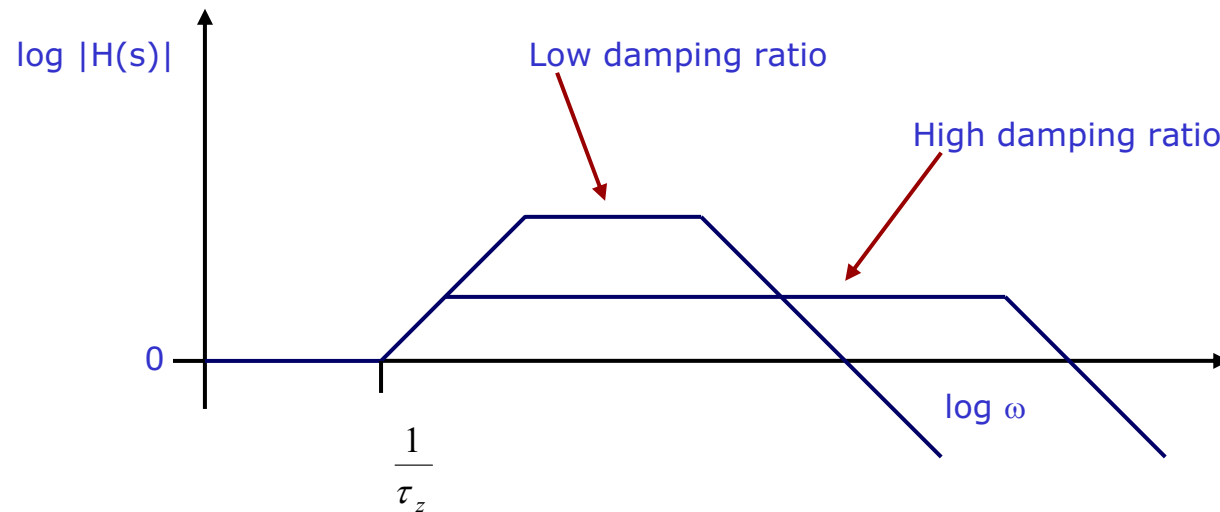


For acceptable phase margin
Place the zero $1/\tau_z$ well below ω_n

(ω_n cross over frequency of
 $H_O(s)$ if no zero was present)



Jitter Peaking



$$|H(s)| = \left| \frac{(1 + \tau_z \cdot s)}{\frac{1}{\omega_n^2} \cdot s^2 + \frac{2 \cdot \xi}{\omega_n} \cdot s + 1} \right|$$

For large damping ratios the zero frequency is below the closed-loop poles

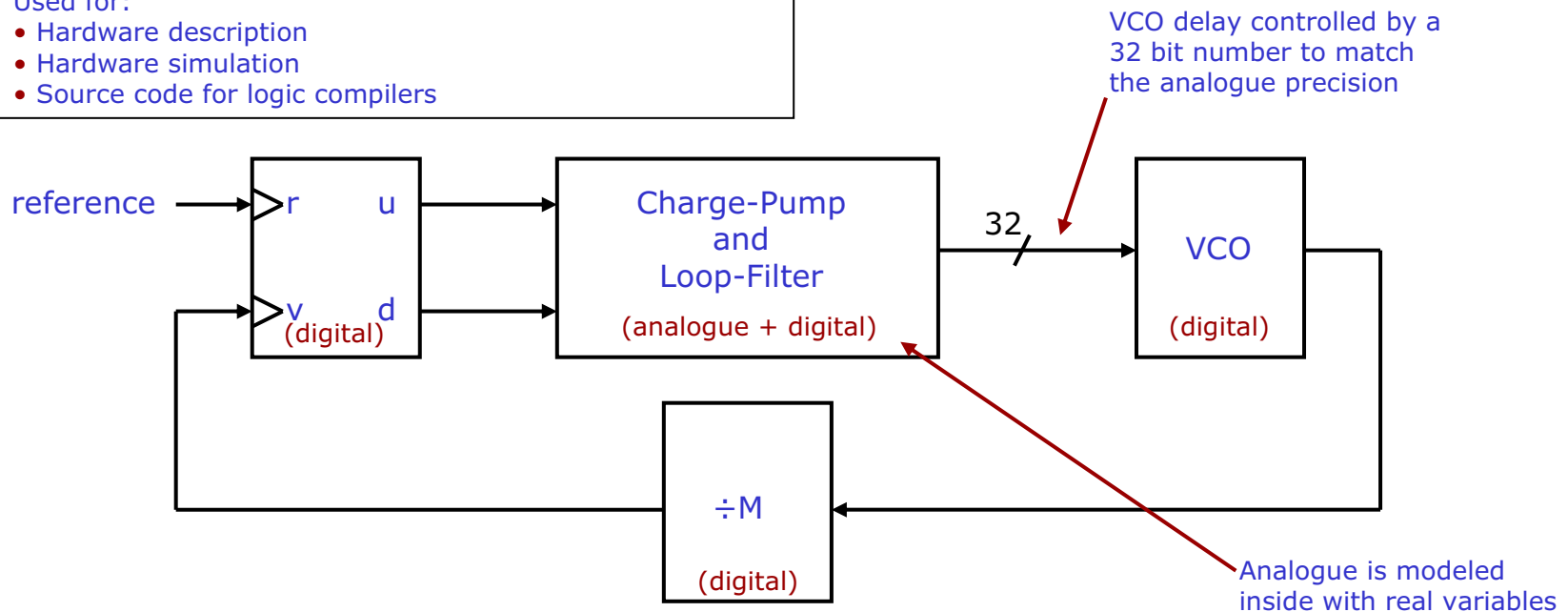
Over a given band of frequencies, $H(s)$ will exceed unity. Jitter frequencies within this band will be amplified

To minimize jitter peaking keep the first closed-loop pole next to the zero by using high loop gains

PLL Modeling with Verilog

Verilog → Hardware description language for digital circuits
Used for:

- Hardware description
- Hardware simulation
- Source code for logic compilers



Algorithm:

- Slice the time in very thin intervals (much smaller than T_{VCO})
- Make the time advance in these time increments
- At every time increment do:
 - Update the phase detector outputs
 - Calculate the new filter voltage according to the phase detector state
 - Update the VCO frequency as function of the filter voltage

Phase Detector and VCO

Phase Frequency Detector

```
`timescale 1 fs / 1 fs
module ThreeStatePD (down, up, r, v);
output
    down,      // Early signal
    up;        // Late signal

input
    r,          // Reference input
    v;          // VCO input

wire    r, v, reset;
reg     up, down;

initial
    begin
        up = 0;
        down = 0;
    end

always @ (posedge r)
    up <= #1 1'b1;

always @ (posedge v)
    down <= #1 1'b1;

always @ (posedge reset)
    begin
        up <= #1 1'b0;
        down <= #1 1'b0;
    end

assign #1
    reset = up & down;

endmodule
```

VCO

```
`timescale 1 fs / 1 fs
module VCO( delay_control, vco_output);
input[31:0]    delay_control;
output vco_output;
reg reset;

initial begin
    reset=1;
    #100000;
    reset=0;
end

delay_element10
    delay1( .in(~vco_output & ~reset), .delay_control(delay_control),
            .out(tap1)),
    delay2( .in(tap1), .delay_control(delay_control), .out(tap2)),
    delay3( .in(tap2), .delay_control(delay_control), .out(tap3)),
    delay4( .in(tap3), .delay_control(delay_control), .out(vco_output));
endmodule

// ----- //
`timescale 1 fs / 1 fs
module delay_element10( in, delay_control, out);
input    in;
input[31:0]    delay_control;
output    out;
reg        out;

initial
    out = 1'b0;

always @(in)
    begin
        out <= #( delay_control/8 ) in; // delay =
            delay_control/(2*number_of_delay_cells);
    end

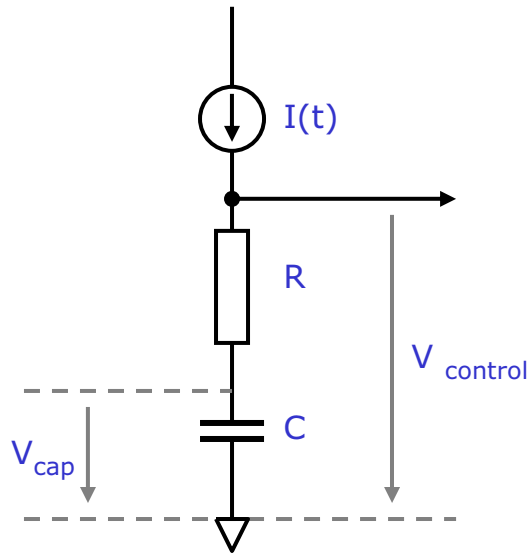
endmodule
```

Delay control: 32 bit number

Ring oscillator with delay control

Cell delay is controlled here

Loop Filter Simulation: R-C filter



Notice that the time increments don't need to be constant (they have to be small). In that case, replace in the equation Δt by: $\Delta t_n = t_n - t_{n-1}$

$$\frac{T_{VCO}}{100} \leq \Delta t \leq \frac{T_{VCO}}{10}$$

Typically

Paulo Moreira

- Simulation step: $t_n = t_{n-1} + \Delta t$
- The current $I(t)$
 - It is "imposed" on the circuit (it is the independent variable)
 - It is controlled by the phase-detector output
- For accuracy the time advances in small increments Δt :
 - Capacitor voltages change very little during a simulation step
 - The time integral of a function in the interval t_{n-1} to t_n can be approximated by:

$$\int_{t_{n-1}}^{t_n} f(t) dt = f(t_{n-1}) \cdot \Delta t \quad [1]$$

- For the simple R-C filter:

$$V_{\text{cap}}(t_n) = V_{\text{cap}}(t_{n-1}) + \frac{1}{C} \int_{t_{n-1}}^{t_n} I(t) dt \quad [2]$$

$$V_{\text{control}}(t_n) = R \cdot I(t_n) + V_{\text{cap}}(t_n) \quad [3]$$

- Simulation equations:

$$V_{\text{cap}}(t_n) \cong V_{\text{cap}}(t_{n-1}) + \frac{1}{C} \cdot I(t_{n-1}) \cdot \Delta t \quad [4]$$

$$V_{\text{control}}(t_n) = R \cdot I(t_n) + V_{\text{cap}}(t_n) \quad [5]$$

pll

Charge Pump & Loop Filter

Charge-Pump (includes the loop filter)

```
`timescale 1 fs / 1 fs
module ChargePump(up, down, delay_control);
.
.
.
`define DeltaVProportional (`Icp * `Rfilt)/1.0E-3    // PLL proportional term (in mV)
`define DeltaVIntegral (`Tref * `Icp / `Cfilt)/1.0E-3 // PLL integral term (in mV)
.
.
.
Variables used in 'analogue' computations declared as real
real dv_capacitor,           // Differential capacitor voltage (in mV)
     control_voltage,       // Integral plus proportional control voltage (in mV)
     frequency,             // VCO frequency (in GHz)
     period,                // VCO period (in ns)
     integral_term,         // loop control integral term (in mV)
     direct_term;           // loop control proportional term (in mV)

initial
begin
.
.
.
     integral_term = `DeltaVIntegral/`IntegrationPoints;
     direct_term = `DeltaVProportional;
     integral_evaluation_time = 25000000/`IntegrationPoints;
end
.
.
.
```

Charge Pump

Main loop, runs at regular time intervals

```

always
begin
  if ( ~reset )
  begin
    // The integral and proportional terms are expressed as voltages
    if (up) // Up refers to frequency not voltage
    begin
      if ( `InitialFilterVoltage + dv_capacitor > `MinimumFilterVoltage)
        dv_capacitor = dv_capacitor - integral_term;
      end
    end
    if (down) // Up refers to frequency not voltage
    begin
      if ( `InitialFilterVoltage + dv_capacitor < `MaximumFilterVoltage)
        dv_capacitor = dv_capacitor + integral_term;
      end
    end
  end
  if ((~up & ~down) | (up & down))
    control_voltage = `InitialFilterVoltage + dv_capacitor;
  else if (up)
    control_voltage = `InitialFilterVoltage + (dv_capacitor - direct_term);
  else if (down)
    control_voltage = `InitialFilterVoltage + (dv_capacitor + direct_term);
  frequency = `C0 + control_voltage*(`C1 + control_voltage*(`C2 + control_voltage*`C3));
  period = 1.0/frequency;
  DelayControl = period*1e6; // first convert the real into an integer
  #(integral_evaluation_time);
end
assign
/* The variable "delay_control" represents the ring oscillator period in fs */
#1 delay_control = DelayControl + vco_phase_noise;
endmodule

```

Calculate capacitor voltage increment

Update the capacitor voltage

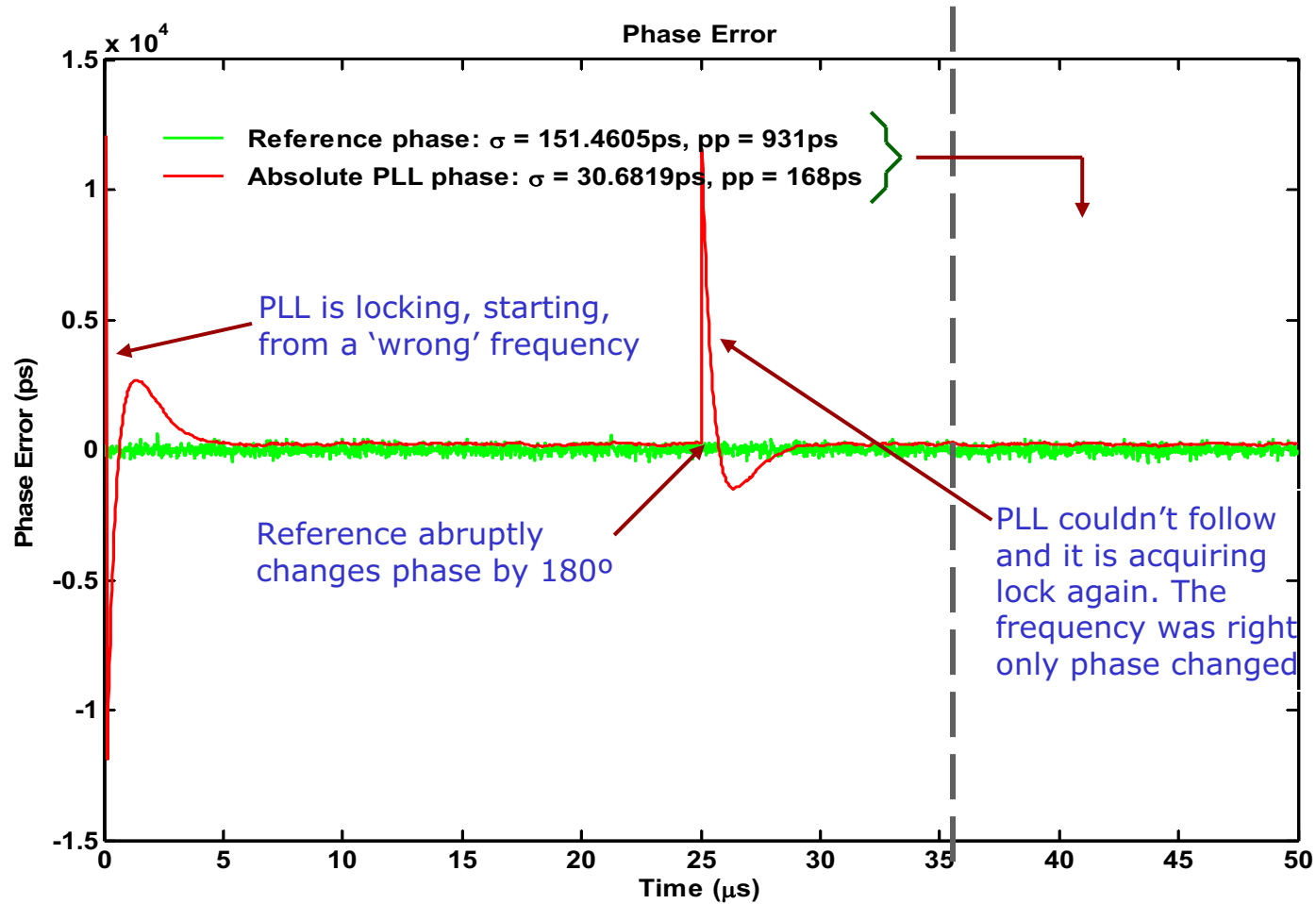
Update the VCO period

Add VCO phase noise

Time step control

pll

Simulation Example



Loop Filter Simulation: (R-C)||C filter

- The filter equations are:

$$V_1(t_n) = V_1(t_{n-1}) + \frac{1}{C_1} \int_{t_{n-1}}^{t_n} I_1(t) dt \quad [1]$$

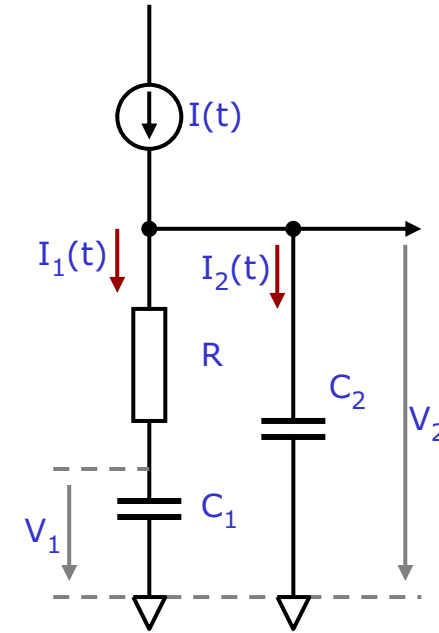
$$V_2(t_n) = V_2(t_{n-1}) + \frac{1}{C_2} \int_{t_{n-1}}^{t_n} I_2(t) dt \quad [2]$$

$$I_1(t) = \frac{V_2(t) - V_1(t)}{R} \quad [3]$$

$$I_2(t) = I(t) - I_1(t) \quad [4]$$

$$[1] \text{ \& } [3] \longrightarrow V_1(t_n) = V_1(t_{n-1}) + \frac{1}{C_1} \int_{t_{n-1}}^{t_n} \left[\frac{V_2(t) - V_1(t)}{R} \right] dt \quad [5]$$

$$[12], [3] \text{ \& } [4] \longrightarrow V_2(t_n) = V_2(t_{n-1}) + \frac{1}{C_2} \int_{t_{n-1}}^{t_n} \left[I(t) - \frac{V_2(t) - V_1(t)}{R} \right] dt \quad [6]$$



Loop Filter Simulation: (R-C)||C filter

$$I_1(t_{n-1}) = \frac{V_2(t_{n-1}) - V_1(t_{n-1})}{R}$$

$$V_1(t_n) = V_1(t_{n-1}) + \frac{I_1(t_{n-1}) \cdot \Delta t}{C_1}$$

$$V_2(t_n) = V_2(t_{n-1}) + \frac{[I(t_{n-1}) - I_1(t_{n-1})] \cdot \Delta t}{C_2}$$

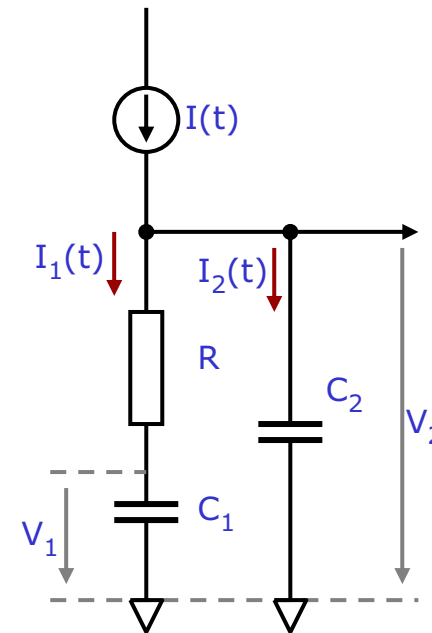
$$\frac{T_{VCO}}{100} \leq \Delta t \leq \frac{T_{VCO}}{10}$$

- Using the same approximations as before:

[7]

[8]

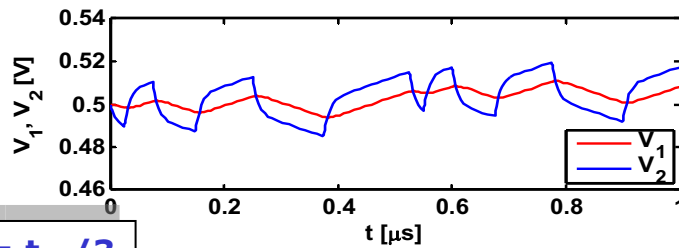
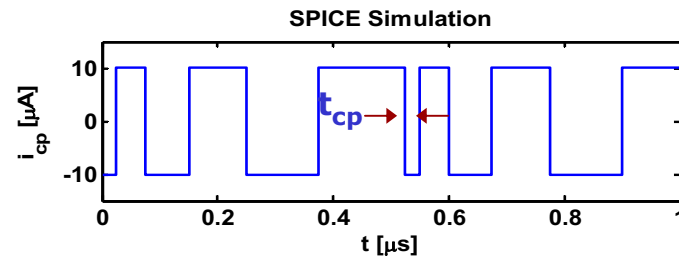
[9]



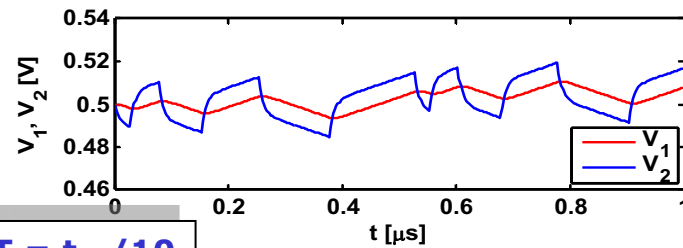
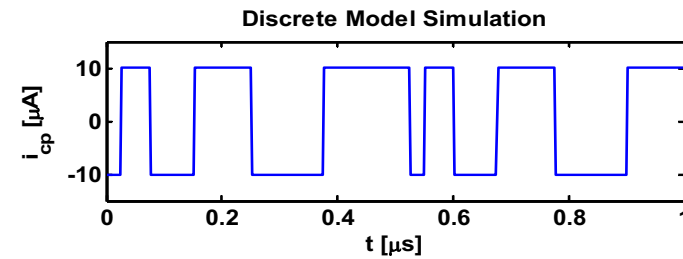
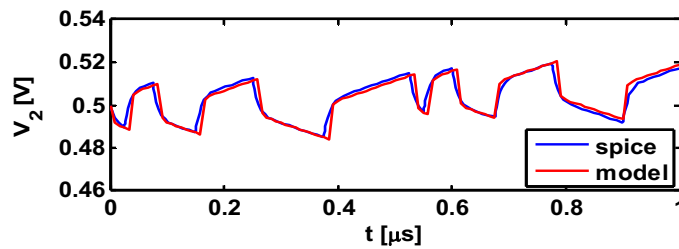
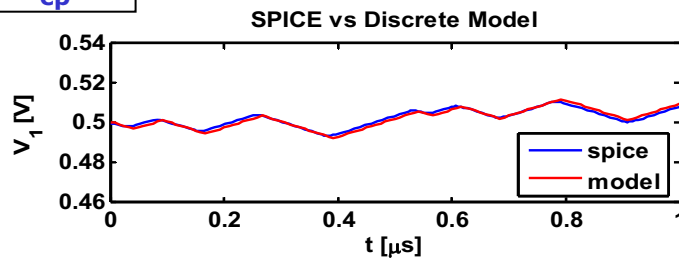
Time increment guideline:

- Simulation speed $\Delta T = T_{VCO}/10$
- Simulation accuracy $\Delta T = T_{VCO}/100$

SPICE versus Model



$$\Delta T = t_{cp}/3$$



$$\Delta T = t_{cp}/10$$

