Clocks and Clock Generation

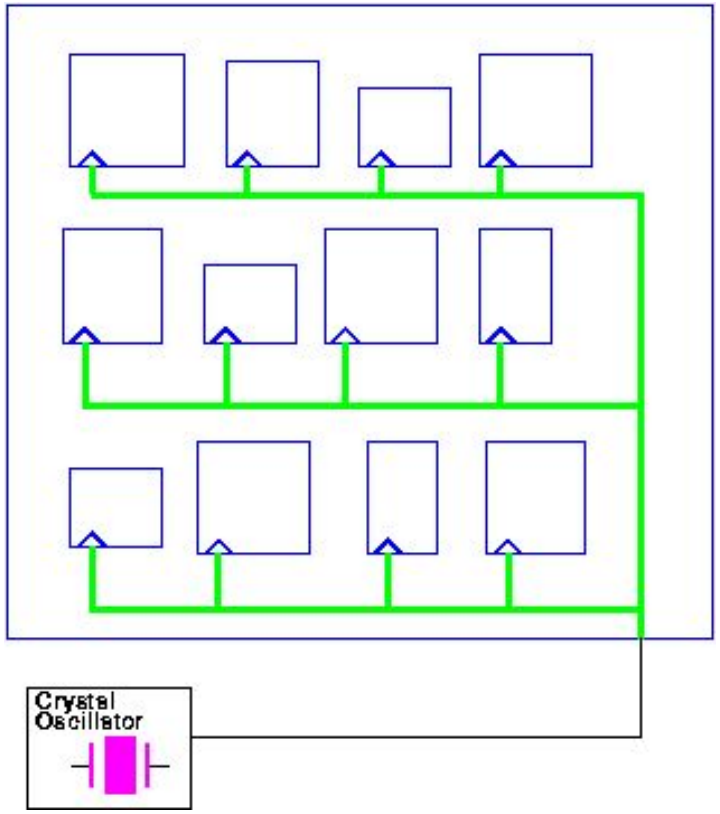
John Morris
Chung-Ang University
The University of Auckland

Iolanthe' at 13 knots on Cockburn Sound, Western Australia'

Clocks

- Clocks are a fundamental part of synchronous circuits
- With a large, complex circuit, management of the clock becomes a significant problem
  - Clock signals must
    - have fixed, known frequencies
      - Crystal oscillators are very stable
        - f is temperature dependent
        - This only affects high precision (>1 in 10^5) timing circuits
        - Crystal `ovens' control T ⇨ very stable clocks
    - be 'clean'

      ie have sharp, well-defined edges
    - have low skew
      - Clock edge must arrive at all parts of the circuit simultaneously

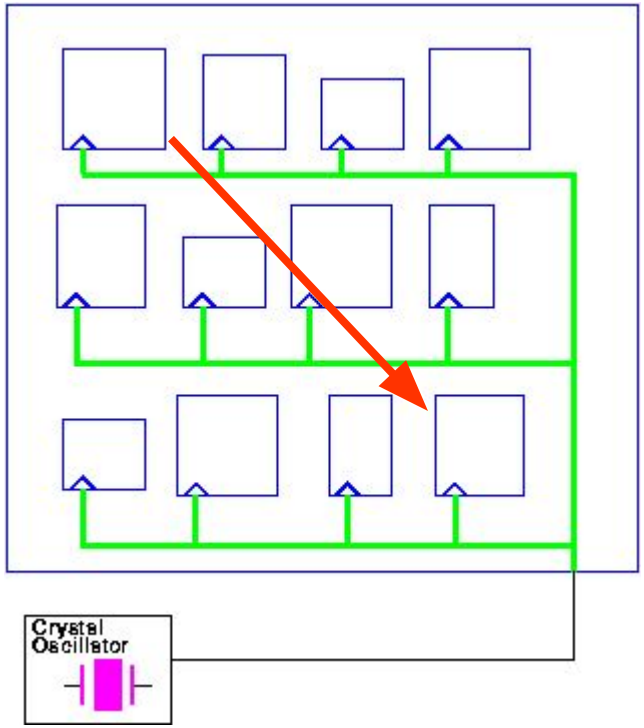


Clock Skew

- Correct operation of synchronous circuits assumes all circuits receive a clock edge at the same time

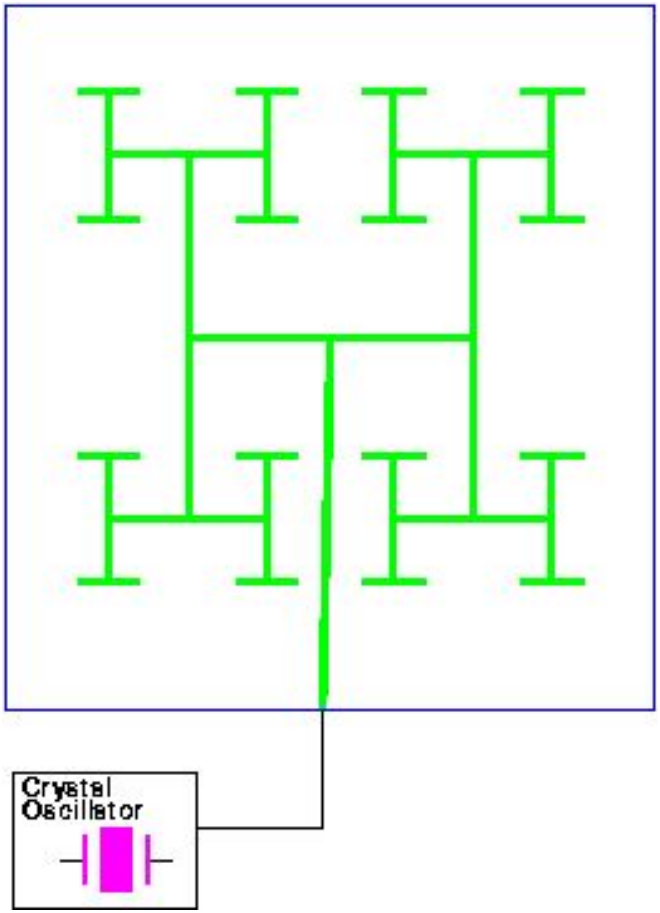
  ie clock skew = 0

  - Physical characteristics of a real circuit make this impossible!
  - Tolerance for clock skew
  - Adds to the propagation delays
    - Signal generated by circuit reached last by a clock edge
      may be 'read' by a circuit which the clock reaches first
  - Lengthens minimum cycle times
  - Reduces potential operating frequency



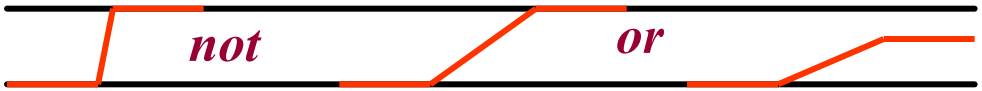
Clock Distribution

- Clock skew can be reduced by layout
  - Clock distribution trees
  - 'H' configurations
- Circuits derive clocks from the 'leaves' (endpoints) of the tree
  - Ideally, clock edges travel the same distance to each leaf
  - *and*
  - Arrive at the same time

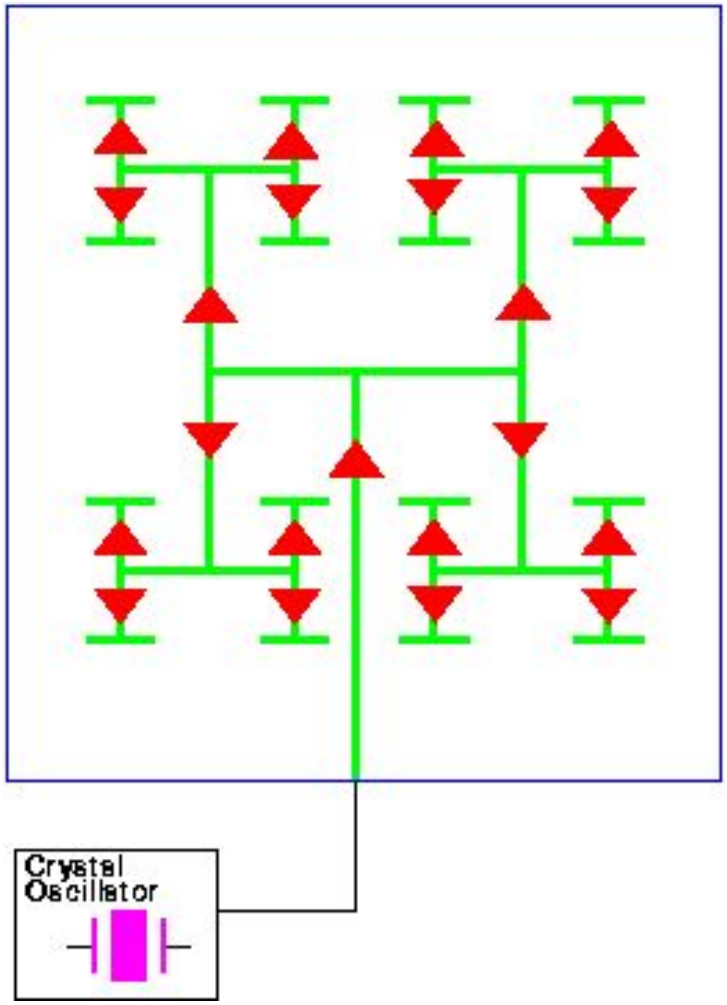


Clock Skew

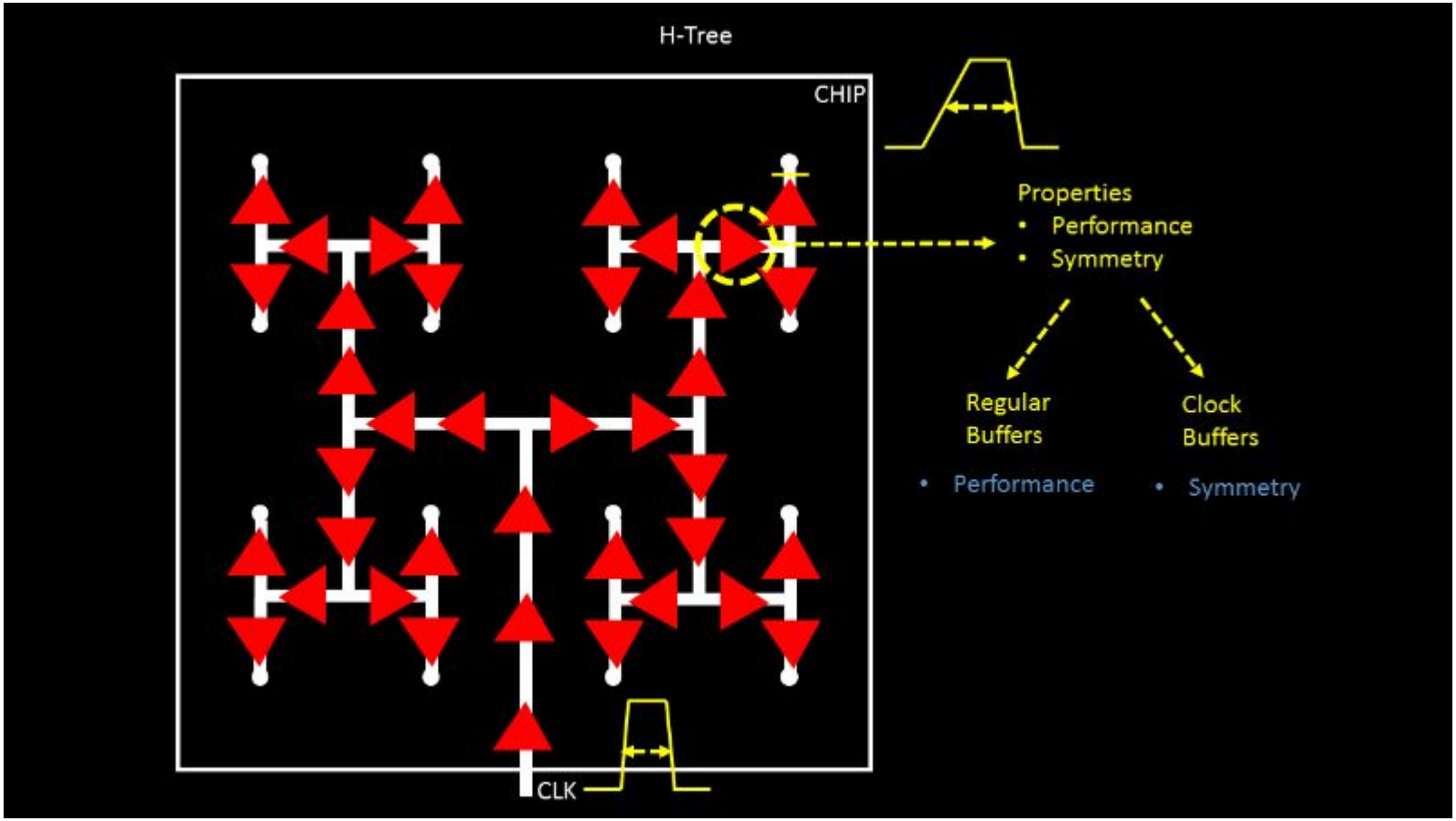
- Amplifiers are commonly added
  - ⇨ Signal re-shaping
  - ⇨ 'Clean' clocks at all receivers
    - ⇨ 'Clean' = fast rise time!



- Although the clock may arrive at all 'receivers' at the same time
  - ⇨ At best, this will only occur on this device
- Clock skew (relative to other devices in a system) will now be high!

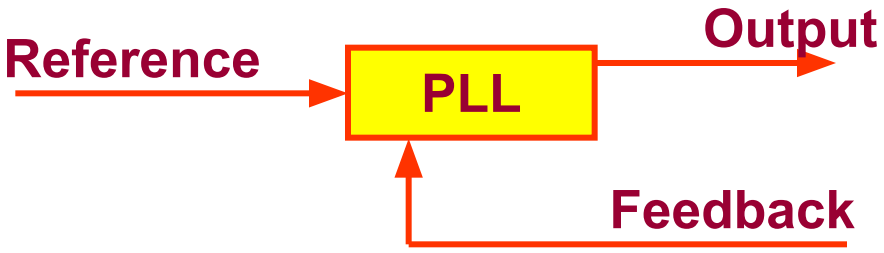


Clock Skew

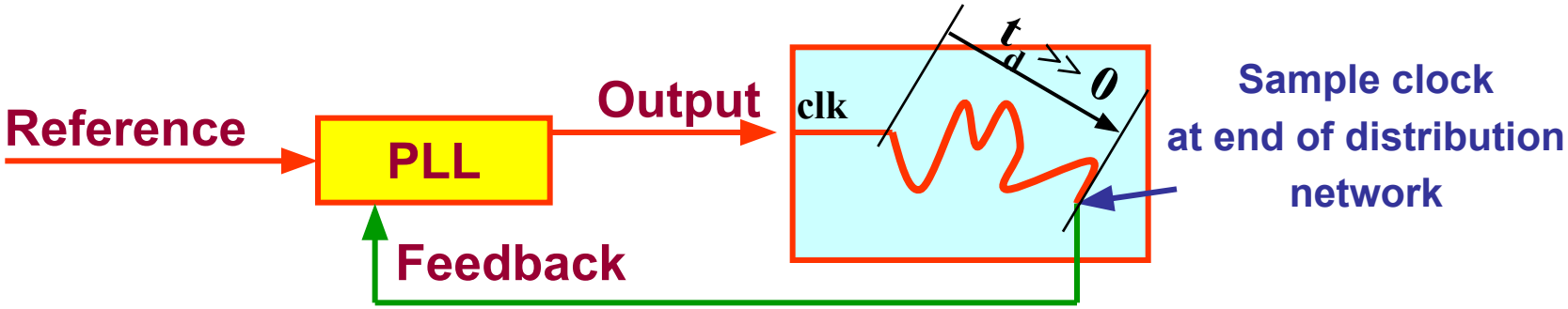

H-Tree
CHIP
CLK
Properties
• Performance
• Symmetry
Regular Buffers
• Performance
Clock Buffers
• Symmetry

Synchronizing clocks

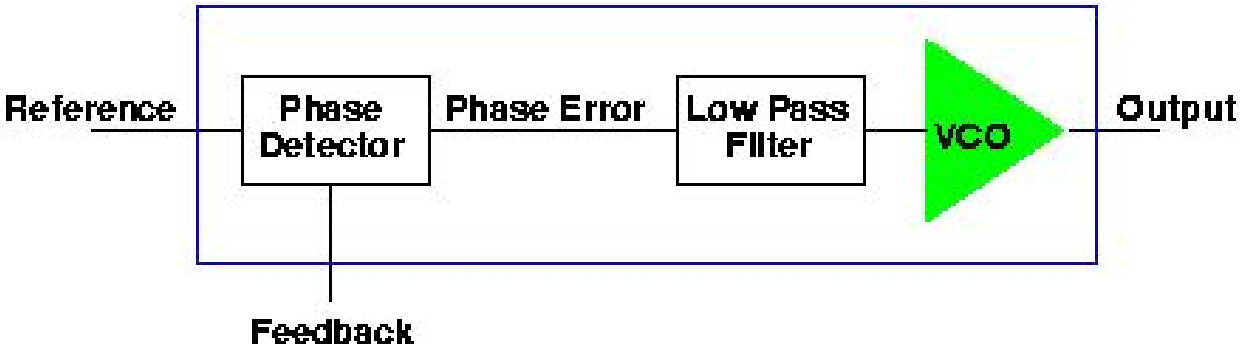
- Phase locked loops (PLLs)
  - Circuit blocks that 'lock' the phases of signals to a reference



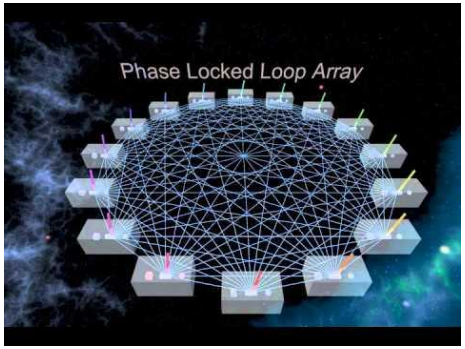
  - PLL adjusts phase of output so that the feedback signal and the reference are in phase
    - Feedback could be derived from a 'leaf' of the clock tree
      - Output will be 'advanced' so that clocks at all leaves are in phase with the reference (and the rest of a system!)



PLL structure



- Phase of Reference and feedback are compared
- Phase difference (error) drives a Voltage Controlled Oscillator
  - A VCO outputs a frequency proportional to its input voltage
  - Low Pass Filter restricts rate of change of VCO input
    - Stops oscillatory behaviour
- VCO outputs desired frequency
- Delays in the feedback cause
  - Non zero phase error output, so the VCO frequency increases to 'catch up'
- Eventually VCO will settle so that its frequency matches the reference and its phase ensures that the feedback and reference are 'phase locked'



Frequency Multipliers

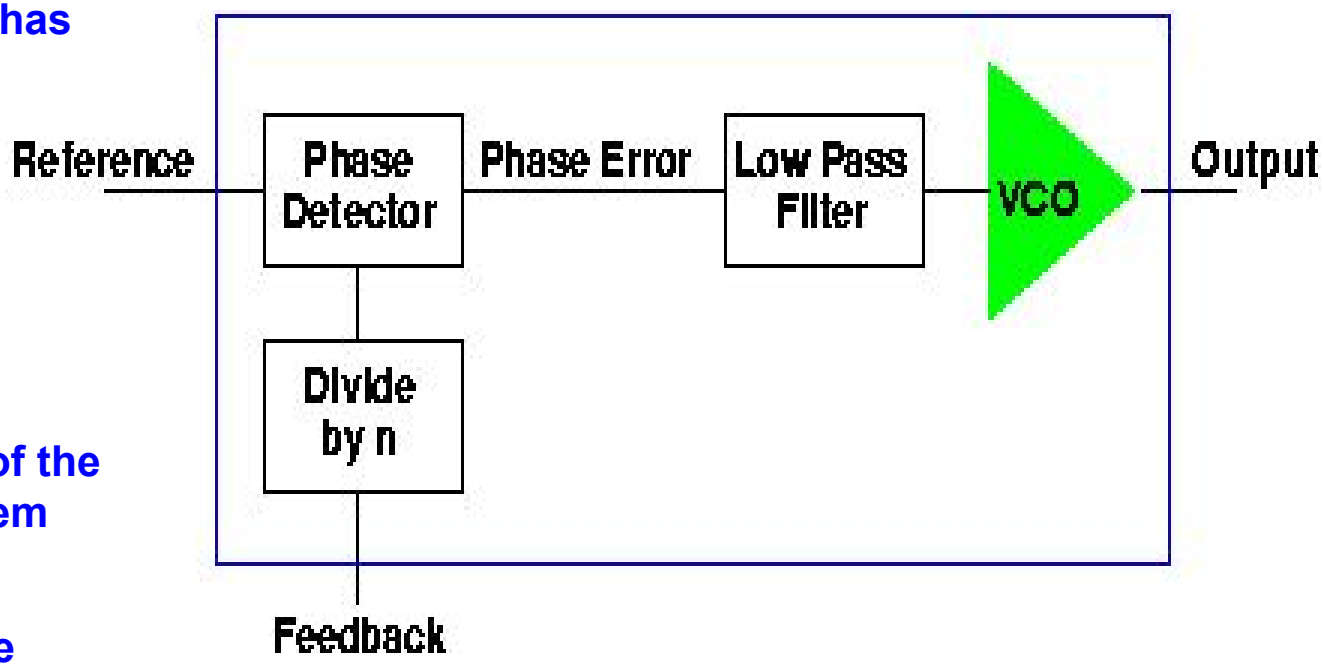
- Let's add a divider to the feedback
- Now, when the PLL has settled
  - $f_{FB} / n = f_{ref}$
    and
  - $\varphi(FB) = \varphi\ (ref)$
  - $f_{output} = n \times f_{ref}$

or

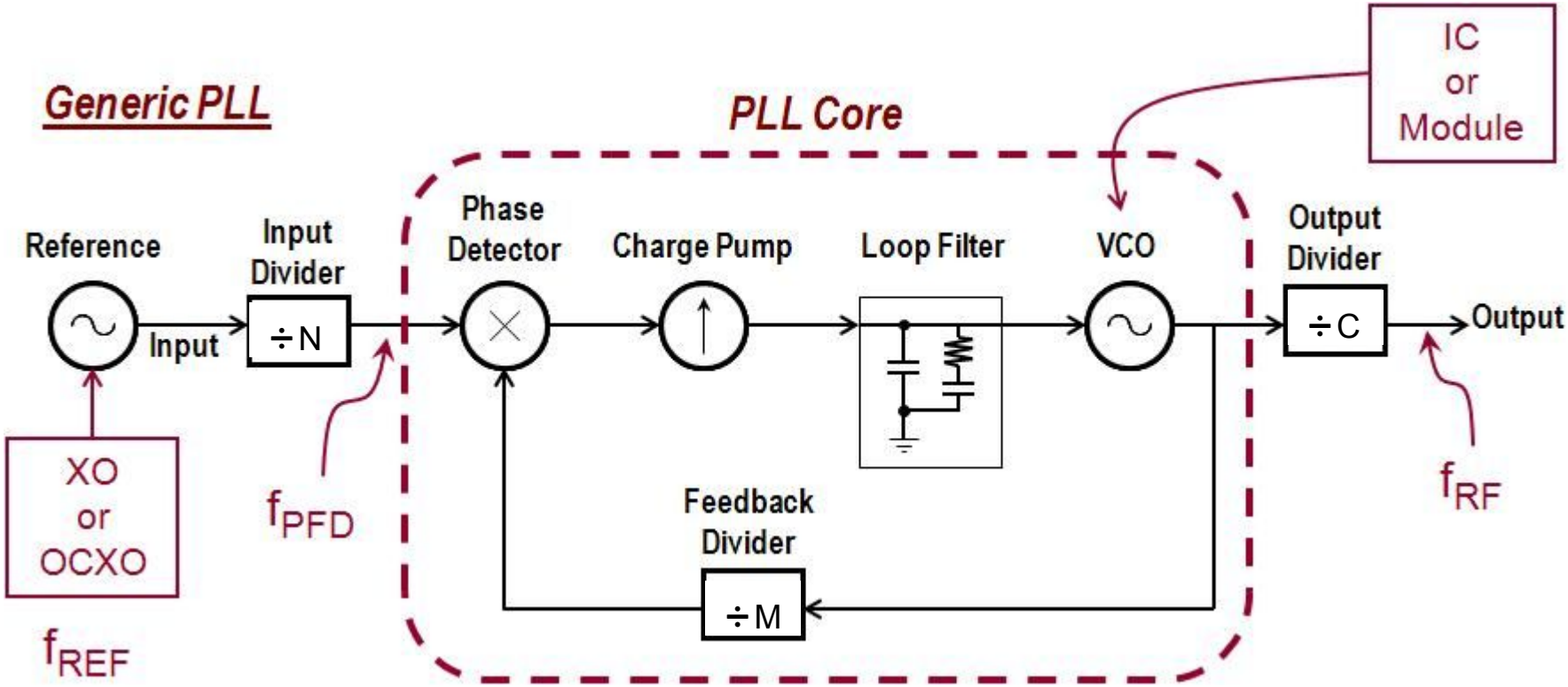
- we have a multiple of the f_{ref} to drive our system

and

- it's in phase with the reference
  - and the remainder of the system!

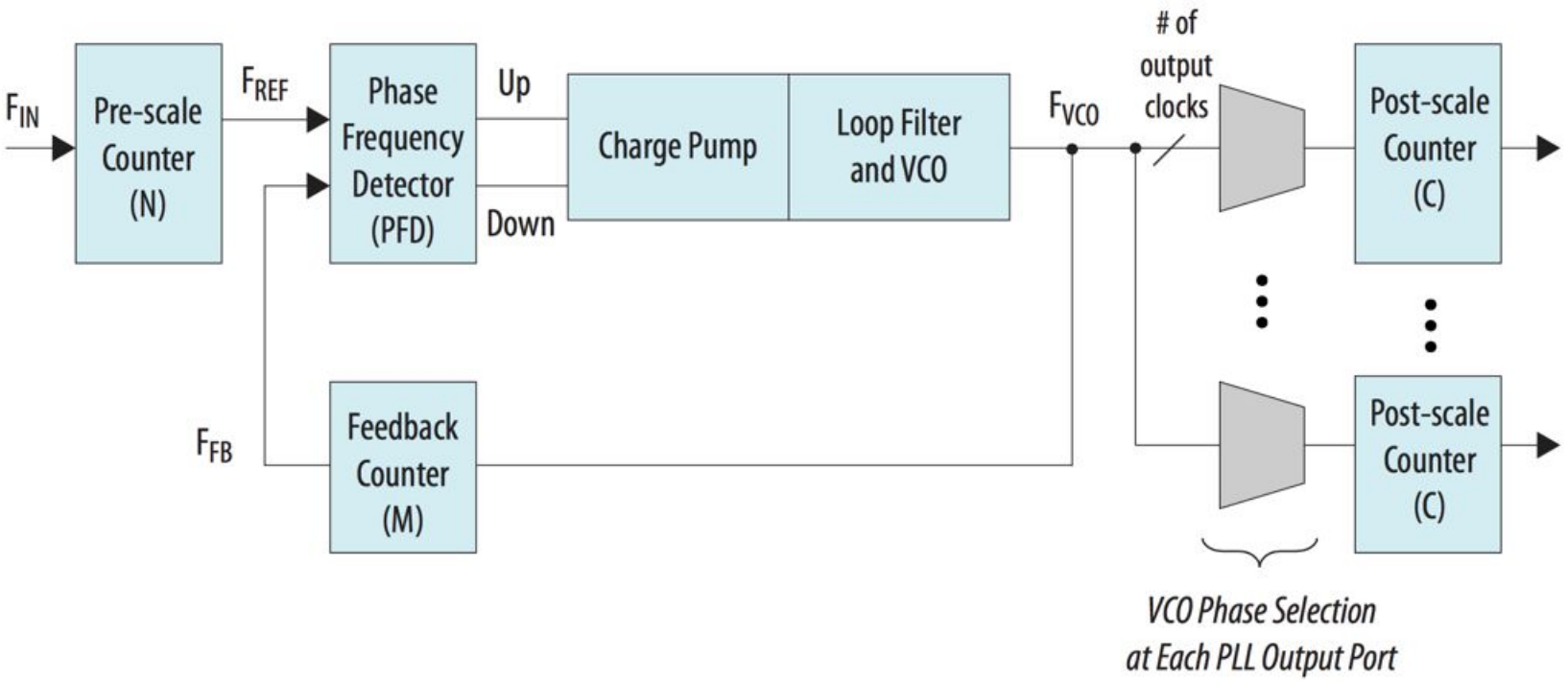


Fractional PLLs



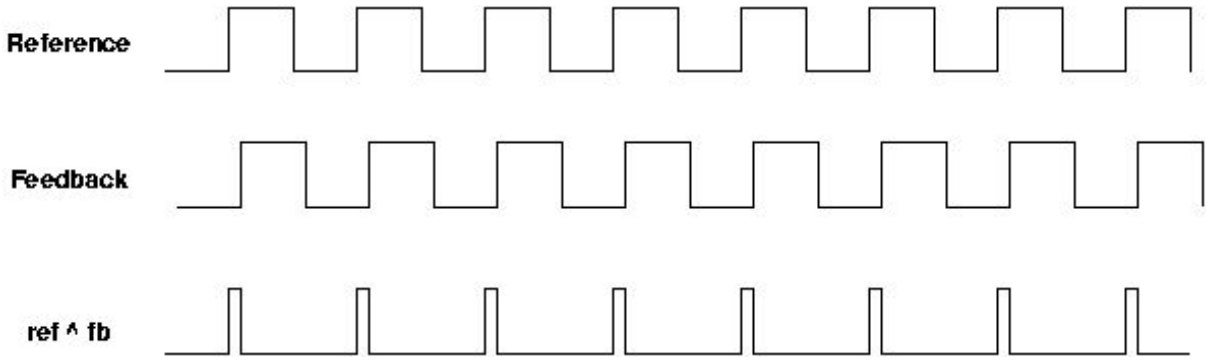
$$F_{RF} = F_{REF} * M /(N*C)$$

Altera PLL Block Diagram



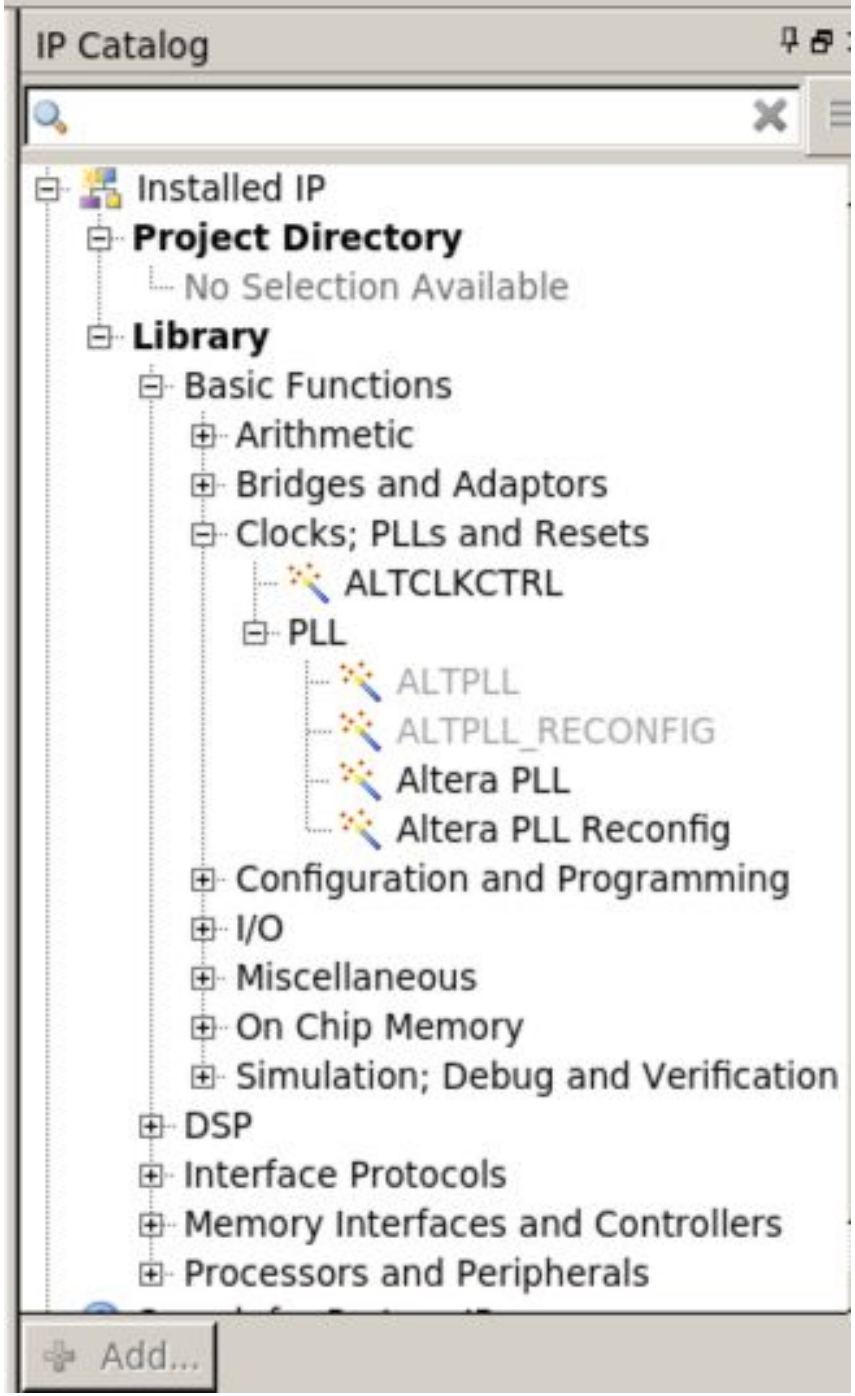
Phase detectors

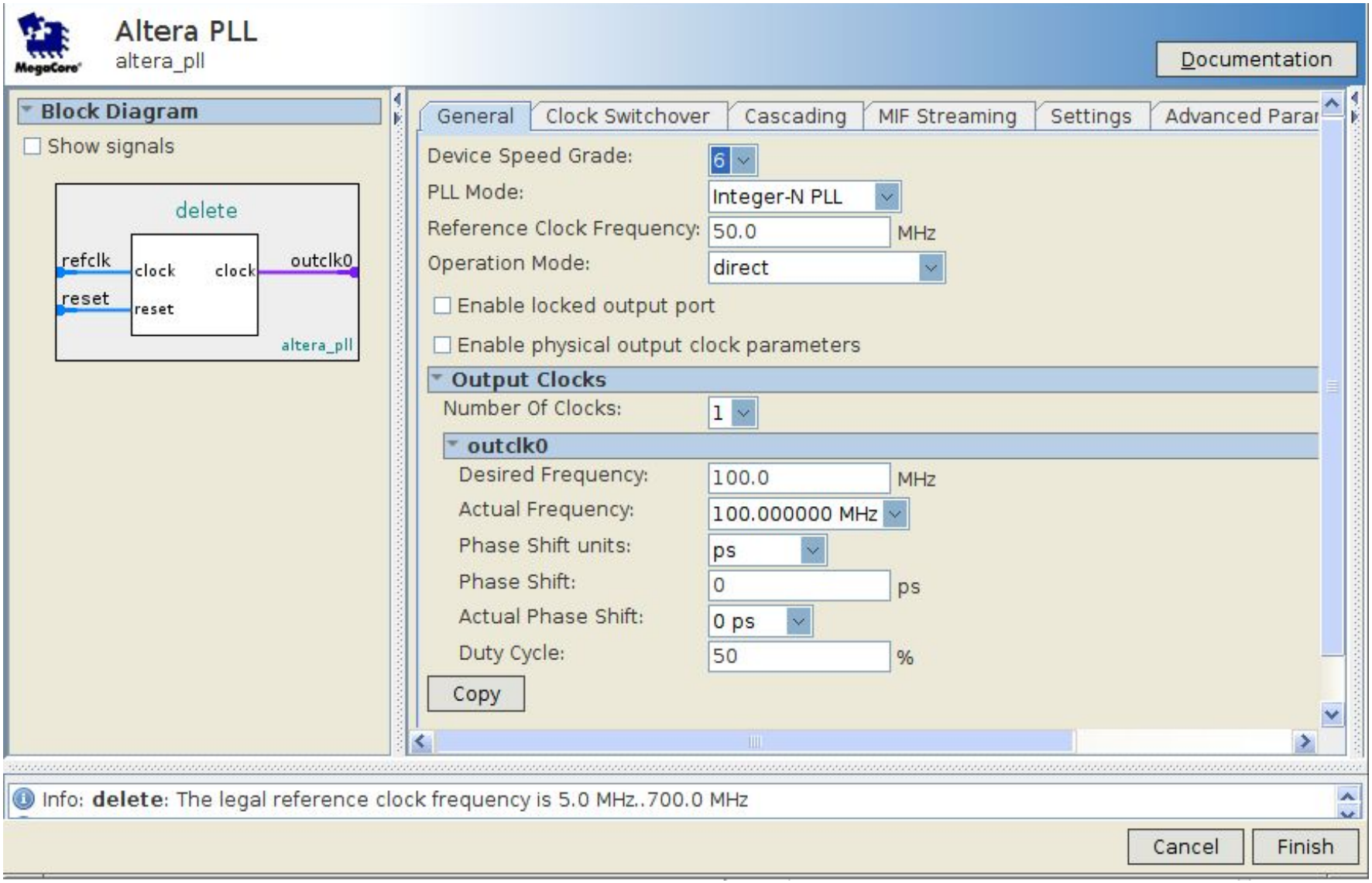
- An XOR gates produces a simple phase detector



  - However, this doesn't indicate a 'lead' or a 'lag'
- A slightly more complex circuit produces 'up' and 'down' correction signals which can drive the VCO

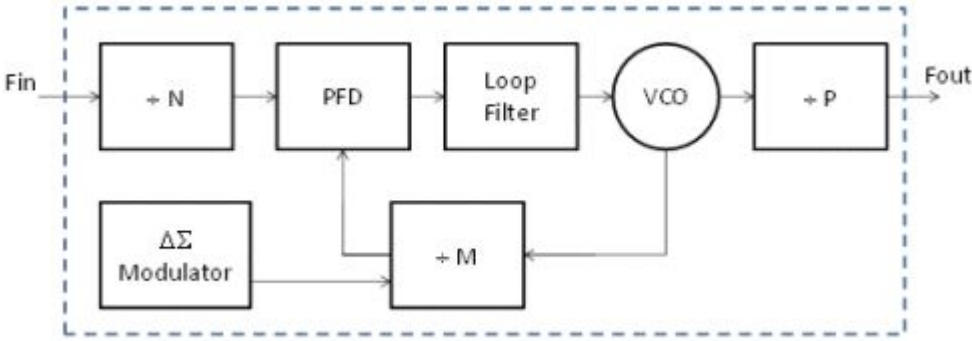
Altera PLL Module




Altera PLL
altera_pll
Documentation
Block Diagram
Show signals
delete
refclk
reset
clock
reset
clock
outclk0
altera_pll
General
Clock Switchover
Cascading
MIF Streaming
Settings
Advanced Parar
Device Speed Grade:
6
PLL Mode:
Integer-N PLL
Reference Clock Frequency:
50.0
MHz
Operation Mode:
direct
Enable locked output port
Enable physical output clock parameters
Output Clocks
Number Of Clocks:
1
outclk0
Desired Frequency:
100.0
MHz
Actual Frequency:
100.000000 MHz
Phase Shift units:
ps
Phase Shift:
0
ps
Actual Phase Shift:
0 ps
Duty Cycle:
50
%
Copy
Info: delete: The legal reference clock frequency is 5.0 MHz..700.0 MHz
Cancel
Finish

Altera PLL Modes

Parameter Names	Paramete...
M-Counter Hi Divide	3
M-Counter Low Divide	3
N-Counter Hi Divide	256
N-Counter Low Divide	256
M-Counter Bypass Enable	false
N-Counter Bypass Enable	true
M-Counter Odd Divide Enable	false
N-Counter Odd Divide Enable	false
C-Counter-0 Hi Divide	6
C-Counter-0 Low Divide	6
C-Counter-0 Coarse Phase Shift	1
C-Counter-0 VCO Phase Tap	0
C-Counter-0 Input Source	ph_mux_clk
C-Counter-0 Bypass Enable	false
C-Counter-0 Odd Divide Enable	false
VCO Post Divide Counter Enable	2
Charge Pump current (uA)	30
Loop Filter Bandwidth Resistor (O...	2000
PLL Output VCO Frequency	300.0 MHz
K-Fractional Division Value (DSM)	1
Feedback Clock Type	none
Feedback Clock MUX 1	glb
Feedback Clock MUX 2	m_cnt



PLL Uses

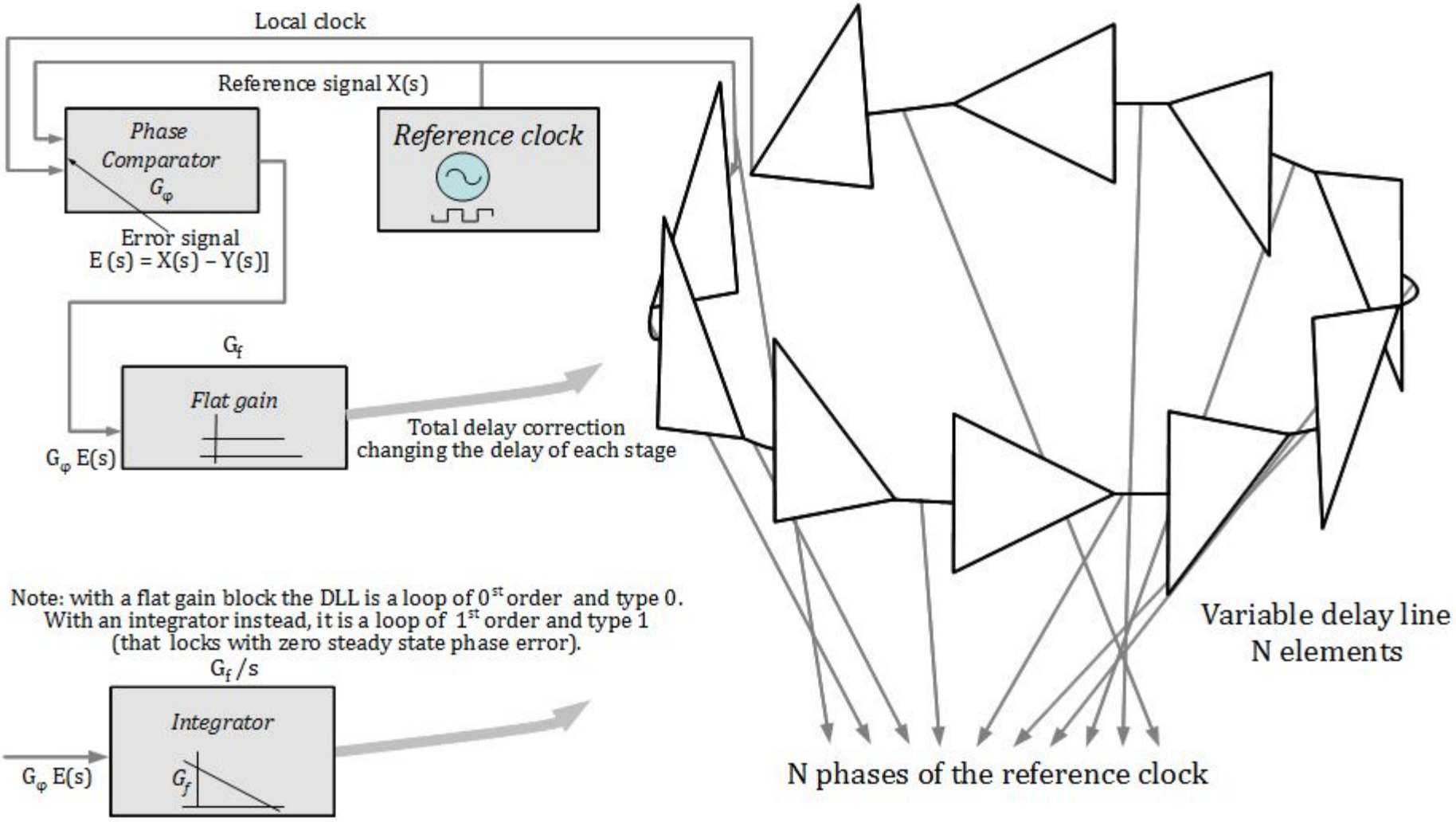
- Thus PLLs
  - Keep clock edges aligned
  - Provide multiples of reference clocks
- You can drive your system with a low frequency clock
  - One you can drive over PCB tracks!
- FPGA internal clock frequencies can be much higher
  - Shorter distances, lower C allow higher f!
- Thus a PLL is a vital component of a high performance FPGA
  - Once it can support ‘core’ frequencies > ~300MHz

PLL Uses

- Note that delays through the I/O buffers driving each pin are also significantly longer than, say, through a logic cell
  - For example, for an Altera FLEX10K10-3
    - Through a logic cell,
      delay = $t_{LUT} + t_{SU}$ = 1.4 + 1.3 ns = 2.7 ns
    - Through an I/O buffer,
      delay = $t_{IOD} + t_{OD1,2,3}$ = 1.3 + (2.6 *or* 4.9 *or* 6.3) ns = 3.9 – 7.6 ns
  - This makes it hard to use an external clock >~ 80MHz whereas internal circuits can be clocked up to nearly 200MHz
  - PLLs to provide internal fast clocks are essential

Delay Locked Loop (DLL)

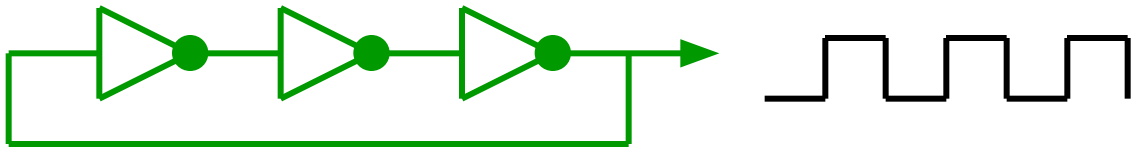
A delay line whose total delay is locked to the clock period



Note: with a flat gain block the DLL is a loop of 0^{st} order and type 0.
With an integrator instead, it is a loop of 1^{st} order and type 1
(that locks with zero steady state phase error).

'Rough' clocks

- We all know how to make an oscillator with an odd number of inverters



- Some FPGAs provide internal oscillators
  - Clock without any external crystal
  - However, the frequency of such clocks may vary widely

    $f \propto t_d(inv)$
  - Looking at the Altera data sheets
  - For an EPF10K10-3, $t_{LUT} \leq 1.4ns$
  - For an EPF10K10-4, $t_{LUT} \leq 1.7ns$

'Denotes the 'speed grade

'Rough' clocks

- Speed grade?
  - Devices made on the same fabrication line
  - Tested
  - Binned (classified) as -3 or -4 depending on their performance, eg
    - If $t_{LUT} \leq 1.4$ ns, it's a -3
    - If $t_{LUT} > 1.4$ns, it's a -4
    - If $t_{LUT} > 1.7$ns, it's a reject - or maybe a -5
  - This means that a "-4" device could have
    $1.4 \text{ ns} < t_{LUT} < 1.7\text{ns}$
  - Thus an oscillator constructed in this way will have a frequency uncertainty of ~18%
    (based on as assumption of 1.7ns)
- So these internal oscillators are fine ..

'Rough' clocks

- So these internal oscillators are fine
  - If your design is
    - (A)
      - working with a slow asynchronous device
        - It must be slow enough so that you can clock a state machine much faster than the fastest response from the device and still guarantee any response times required by the device
      - and
      - has no critical output timing (*ie* you don't need to use the clock to accurately time an output)
    - *or*
    - (B)
      - it's fast, but
      - timing isn't critical (*ie* you can 'miss' an event in one clock cycle and detect it in the next without violating protocol rules)