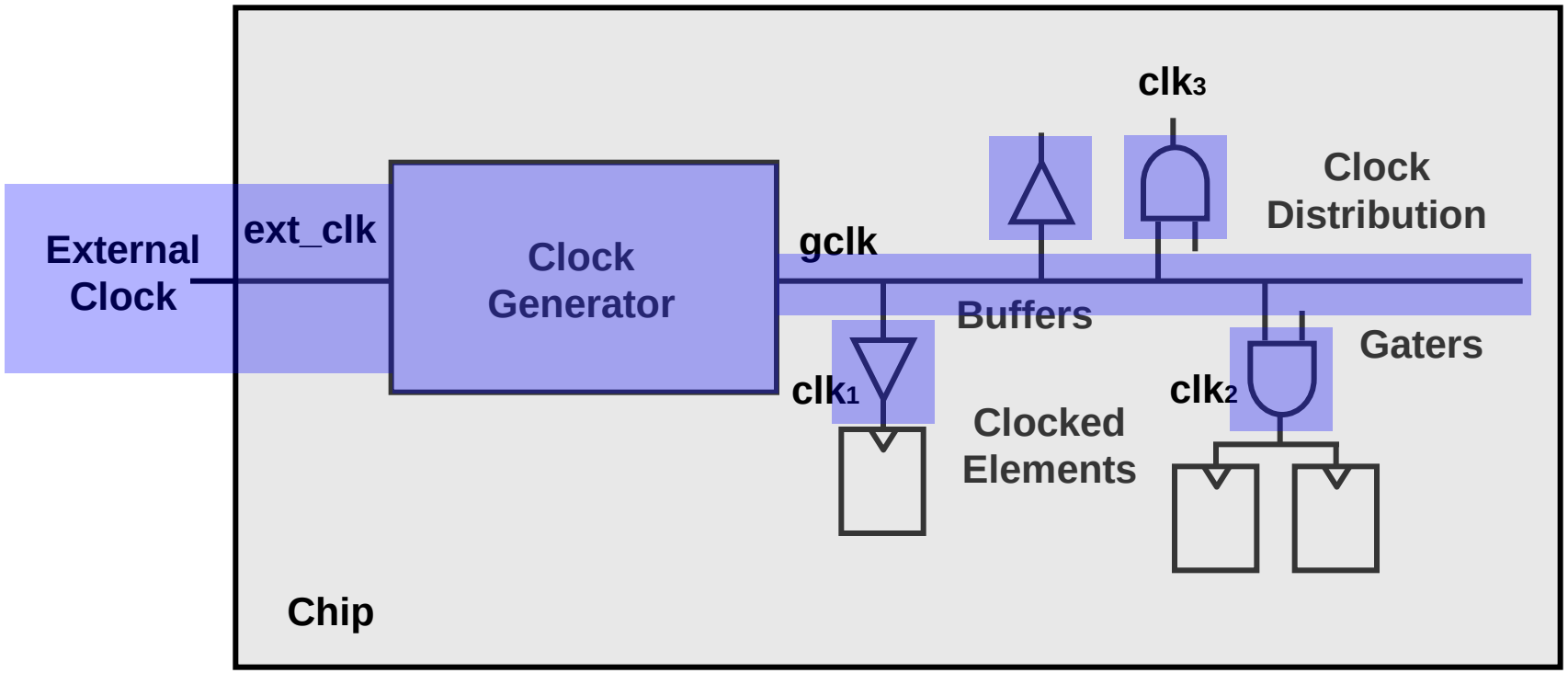
Clock Distribution

Shmuel Wimer
Bar Ilan Univ. Eng. Faculty
Technion, EE Faculty

Clock System Architecture



Chip receives external clock through I/O pad.

Clock generator adjusts the global clock to the external clock.

Global clock is distributed across the chip.

Local drivers and gaters drive the physical clocks to clocked elements.

Global Clock Generation

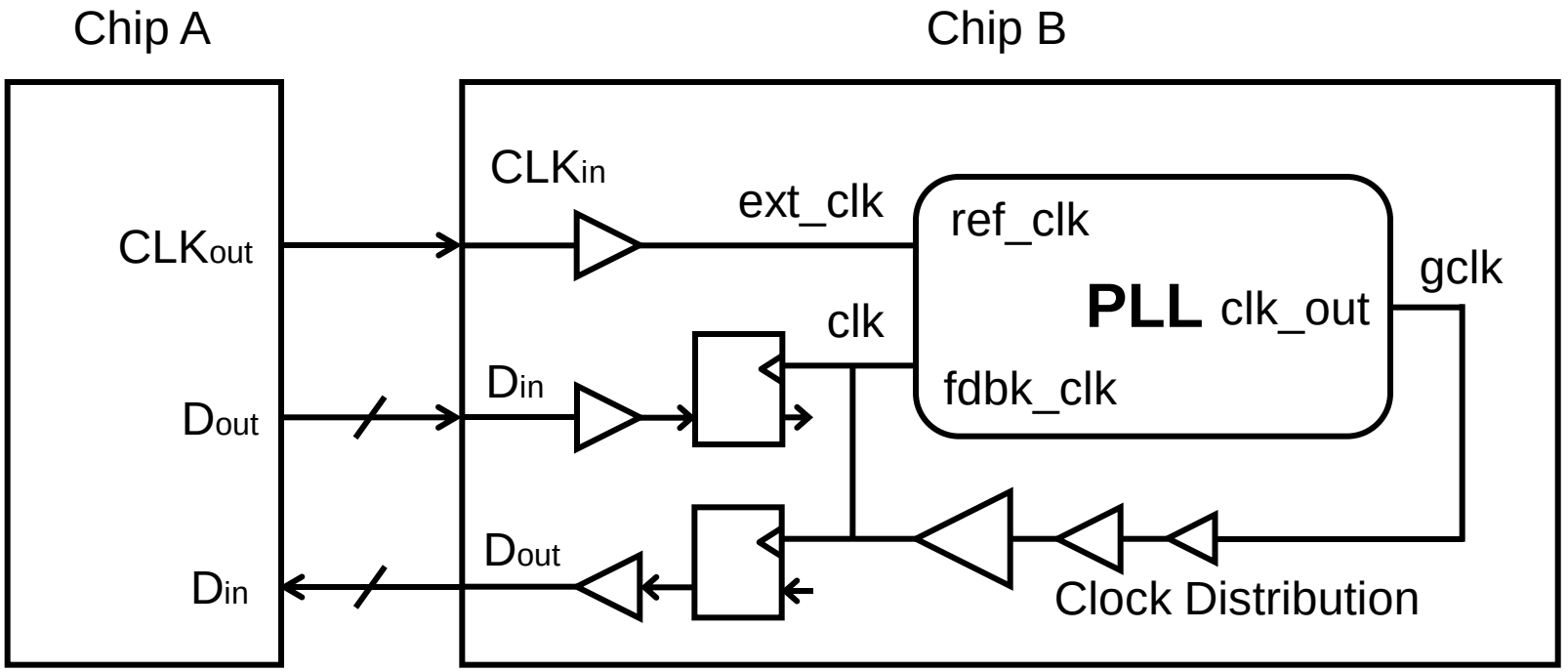
- Receives external clock signal and produce the global clock distributed across the die.
- A large skew occurs between external clock and the physical clocks at clocked elements due to delay of distribution network (wires, buffers, gaters).
- Therefore, data at clocked elements is no more in sync with data at I/O pins.
- ***Phased Locked Loop (PLL)*** compensates this delay.
- PLL can perform frequency multiplication to obtain the required on-chip frequencies.

Synchronous Chip Interface with PLL

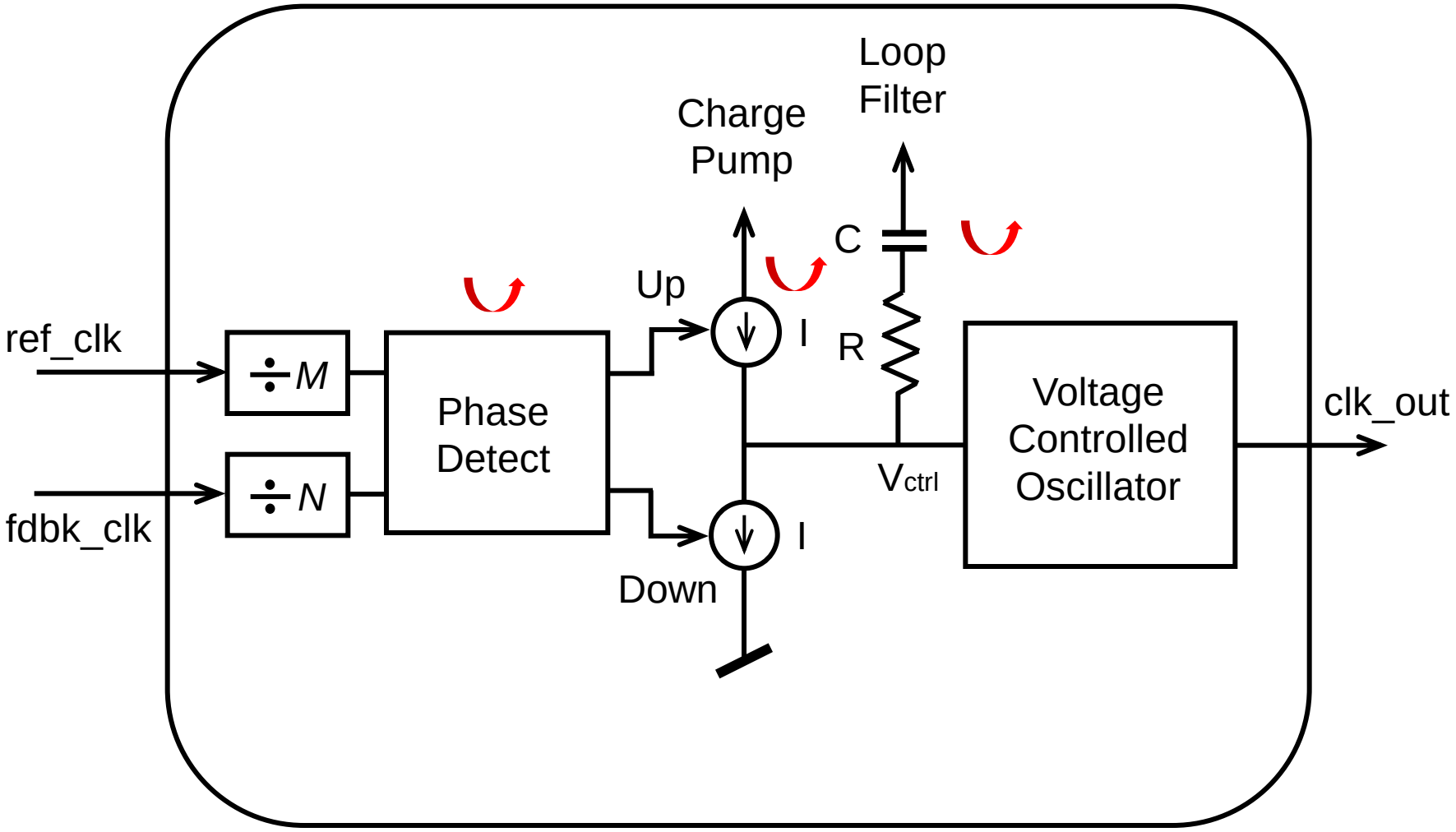
Chip A communicates synchronously with chip B

Chip B uses the clock sent by chip A. Data in and out must be synchronized to the common clock.

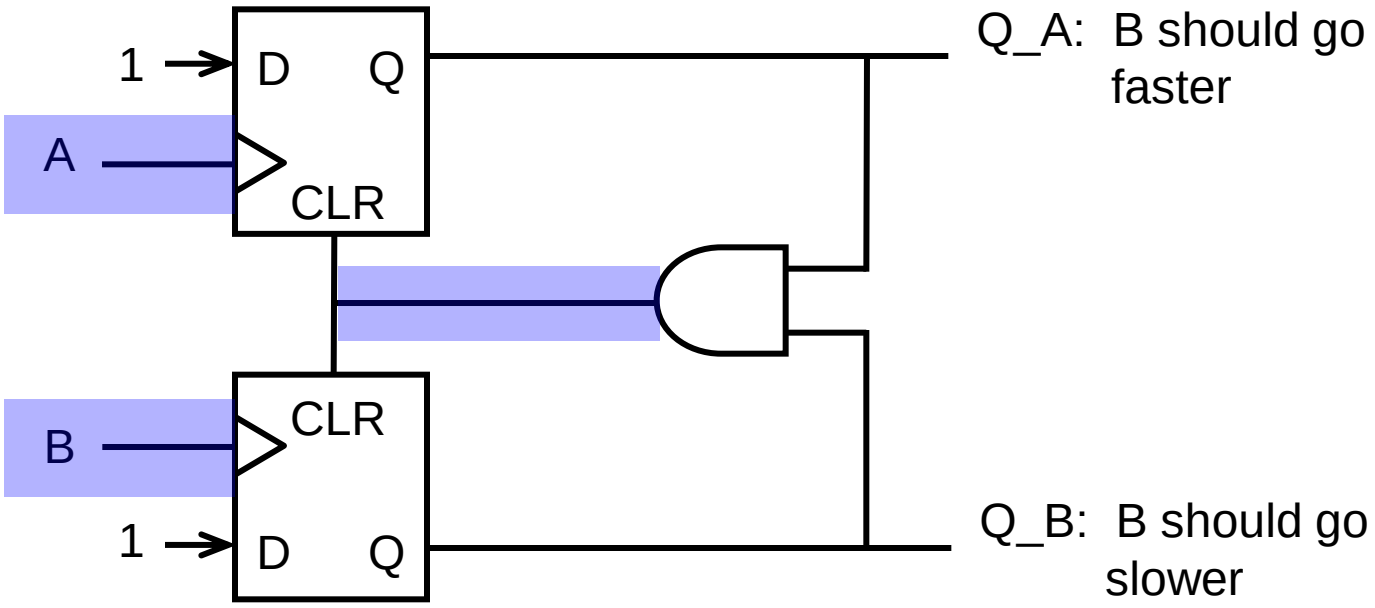
A PLL produces the global clock of chip B such that it is in sync with the external clock.



How PLL Works?



Phase – Frequency Detector (PFD)

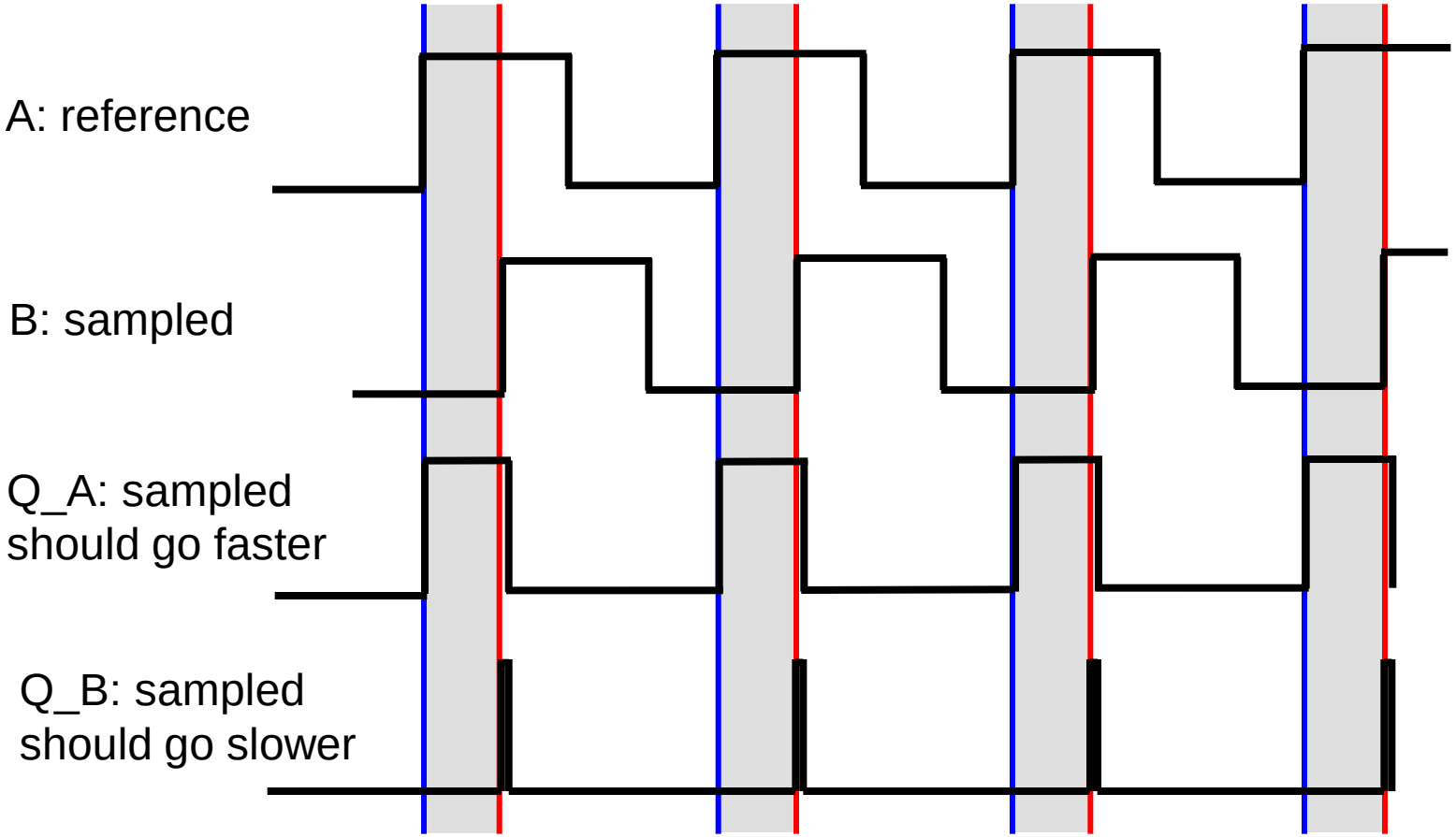


The two flip-flops receive the signals at their clock input (one is usually a reference and the other is the sampled).

The output of the leading flip-flop is 1 for the lead duration.

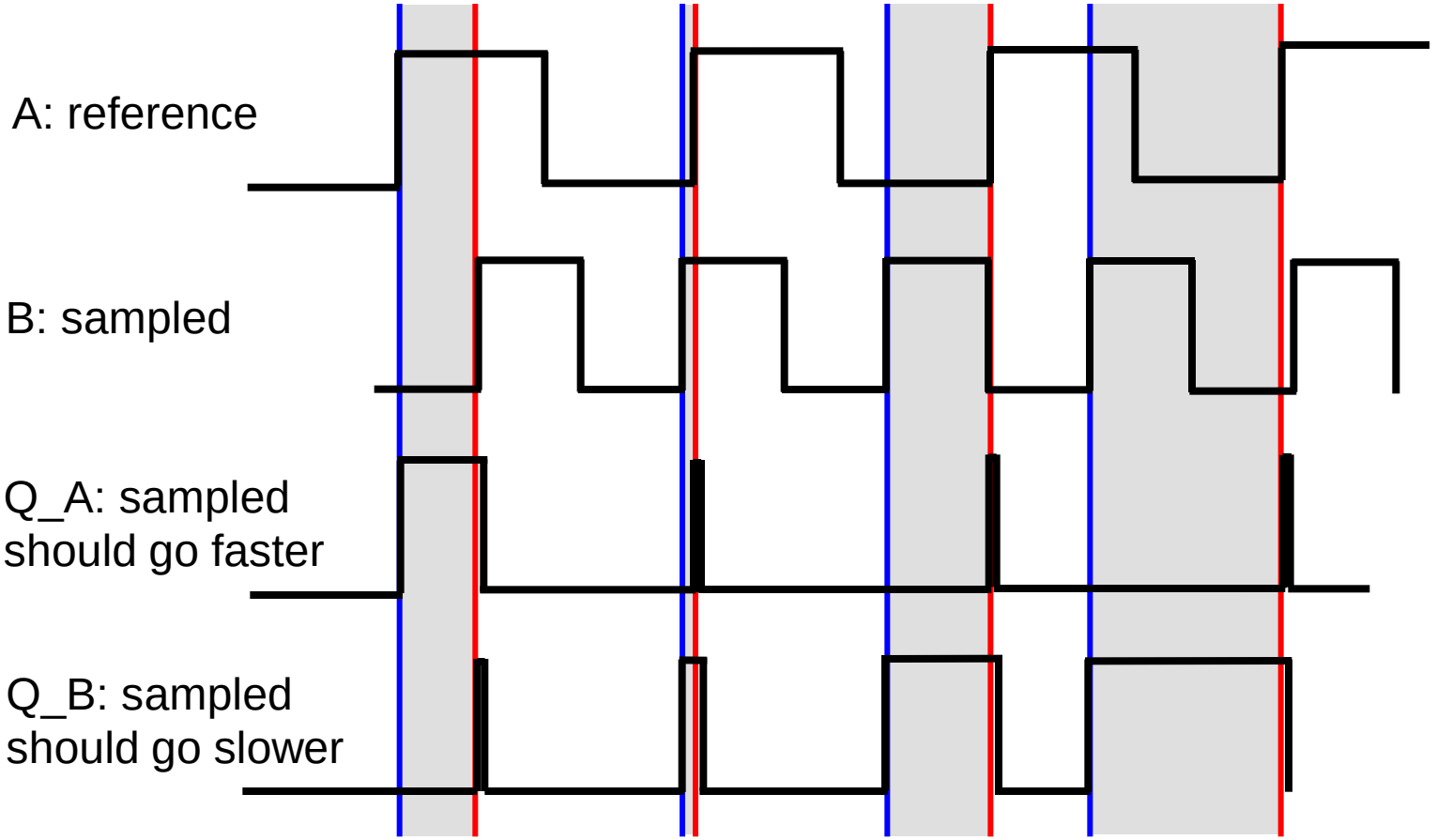
Once the lagging signal arrives, a reset turns both Q_A and Q_B to zero.

What happens when the reference and the sampled signals are a shift of each other?



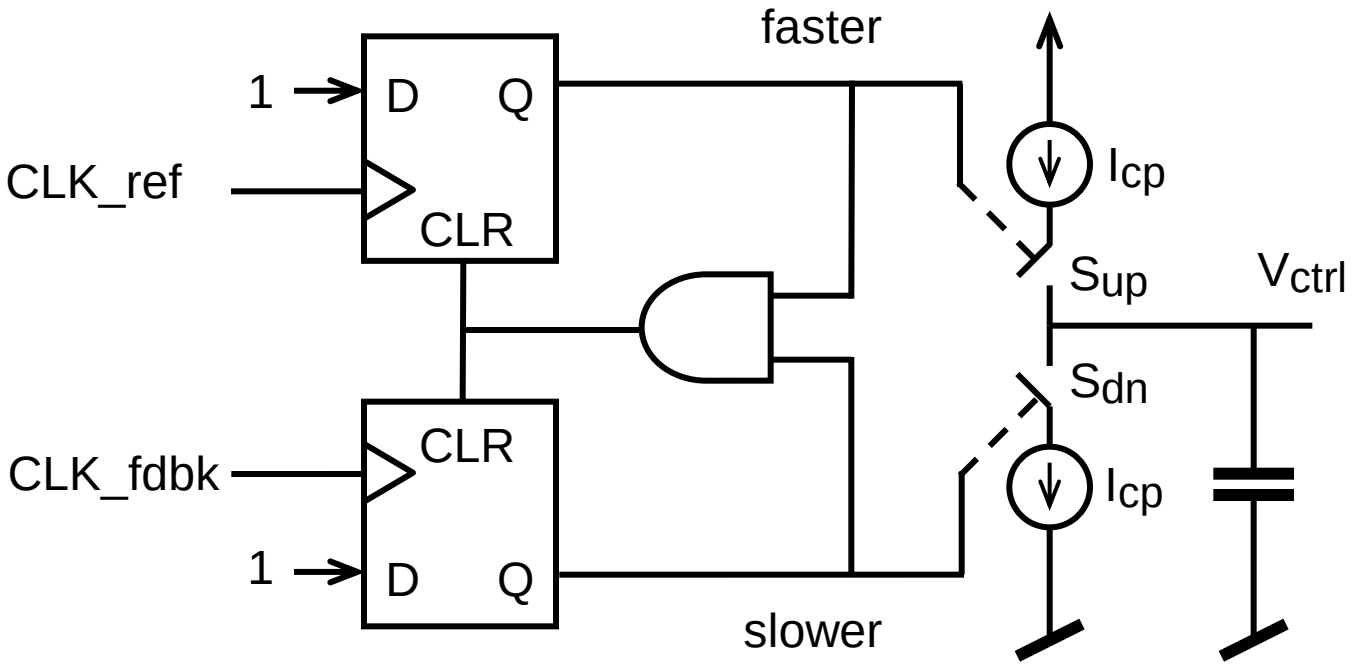
The spikes at Q_B are a result of the delay of the AND gate driving the CLR input of flip-flip and the internal delay from CLR to Q.

What happens when the reference and the sampled signals have different frequencies?



Sampled is more often 1-value than the reference is, since rising edge of B occurs more often than rising edge of A.

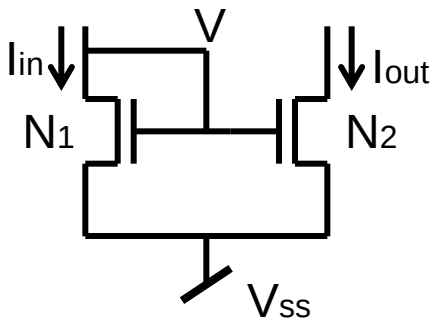
Charge Pump

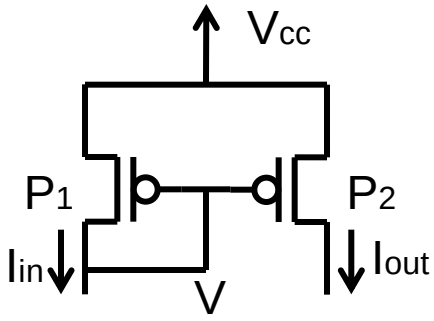


Converts PFD error (digital) to charge (analog), which then controls PLL VCO.

Charge is proportional to PFD widths: $Q_{\text{cp}} = I_{\text{up}} \times t_{\text{faster}} - I_{\text{dn}} \times t_{\text{slower}}$.

Current Mirror





Charge pump consists of current mirrors which are sources of constant current. Device N_1 is in saturation since its gate is connected to high voltage. I_{ds} ($=I_{in}$) depends only on V_{gs}. V_{gs} is similar in N_2, hence $I_{out}=I_{in}$.

This is an ideal current source with infinite output impedance since I_{out} is independent of N_2 load; a change in output voltage doesn't affect I_{out}.

Current mirror works similarly for P transistors.

How Charge Pump Works?

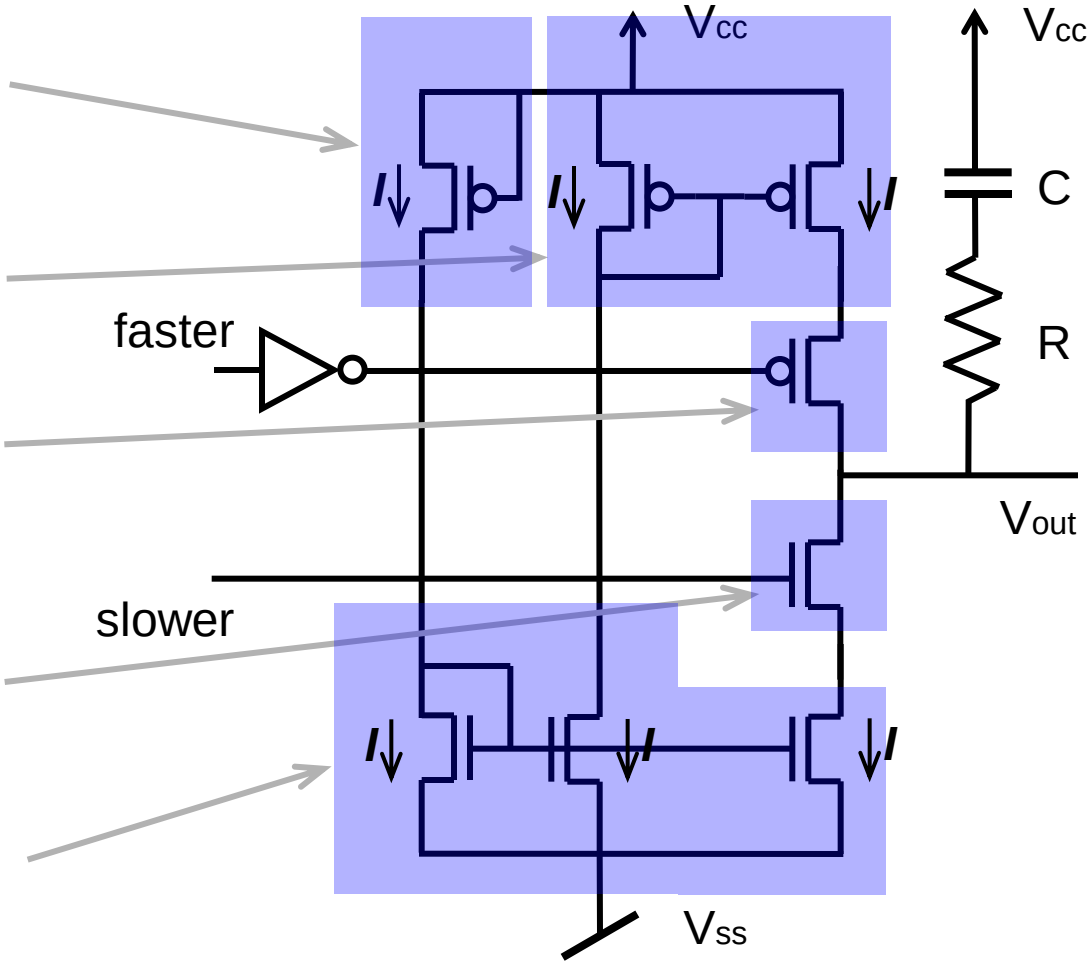
R load – determines the current through current mirror

P-type current mirror

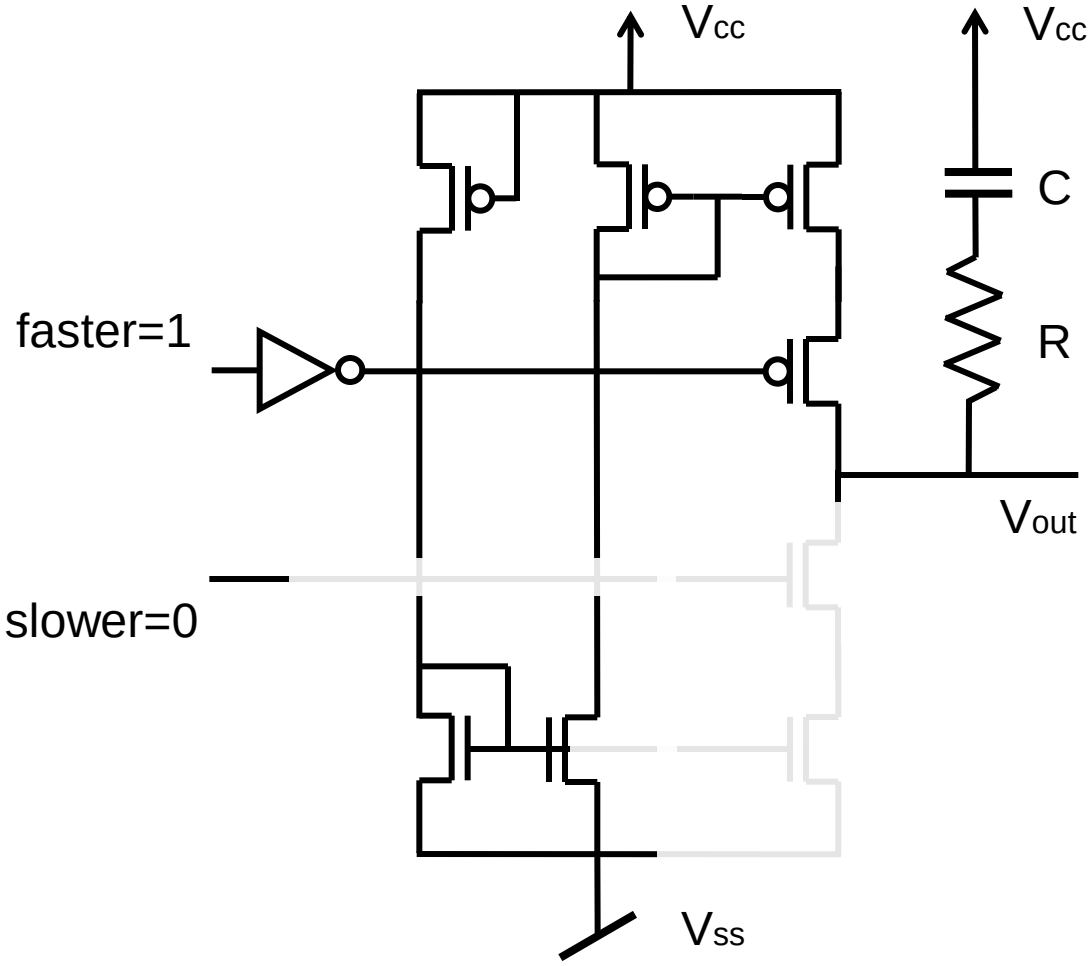
Switch – open when faster = 1

Switch – open when slower = 1

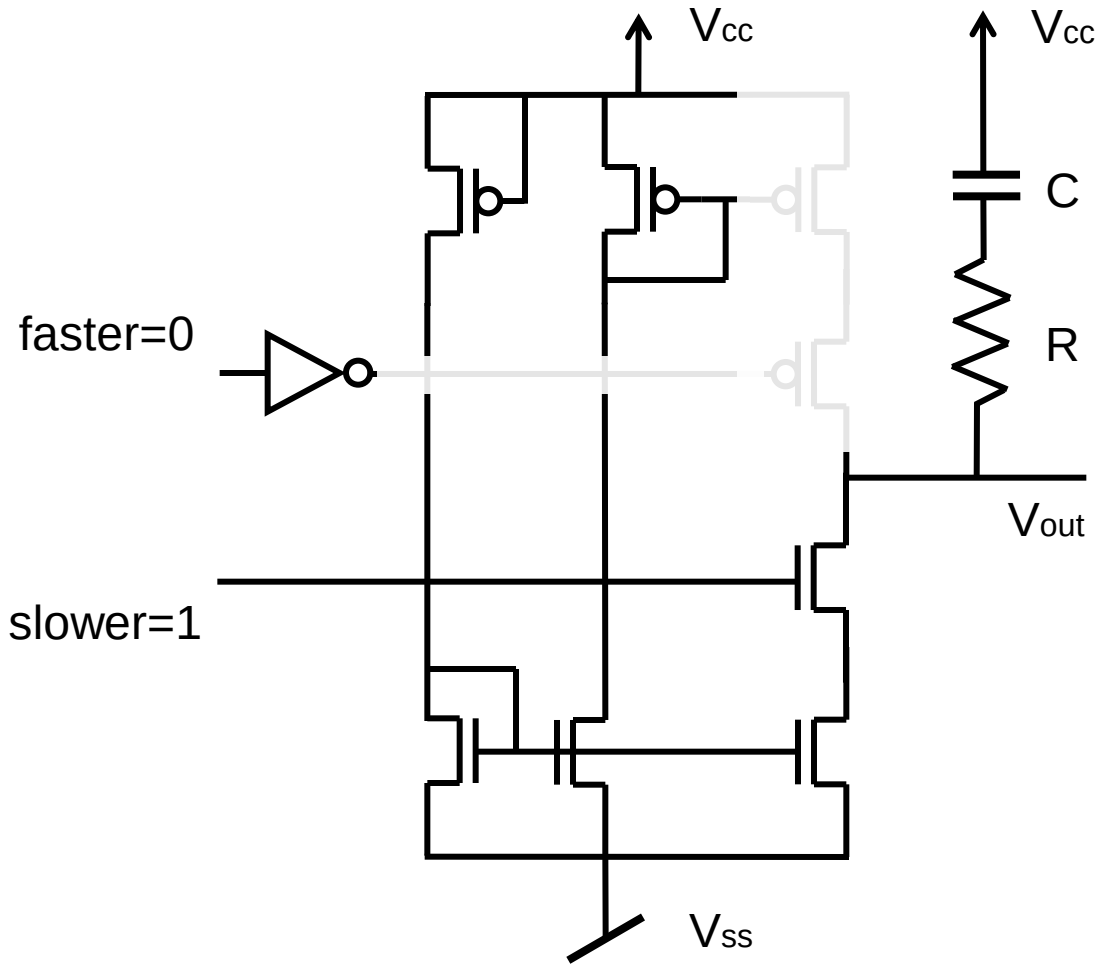
N-type current mirror



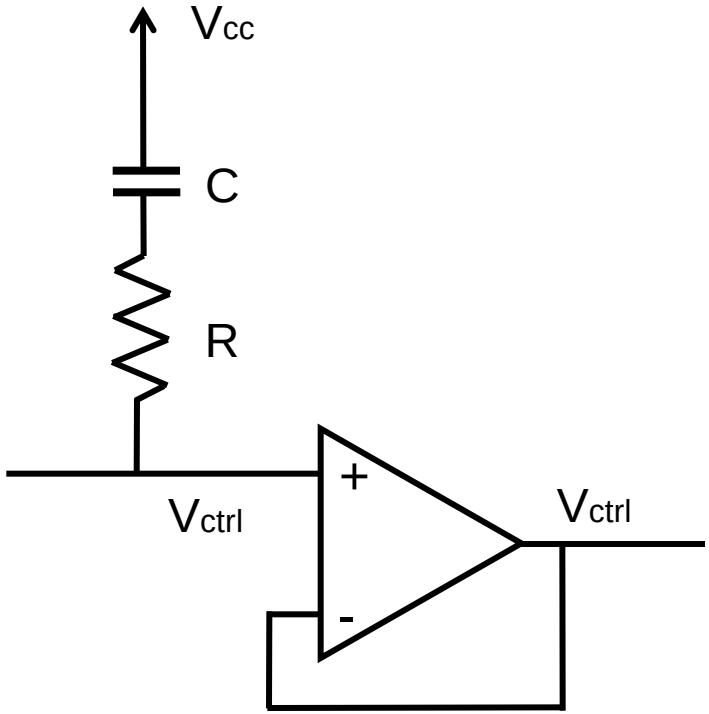
Faster Mode $V_{out} \rightarrow V_{cc}$



Slower Mode $V_{out} \rightarrow V_{ss}$



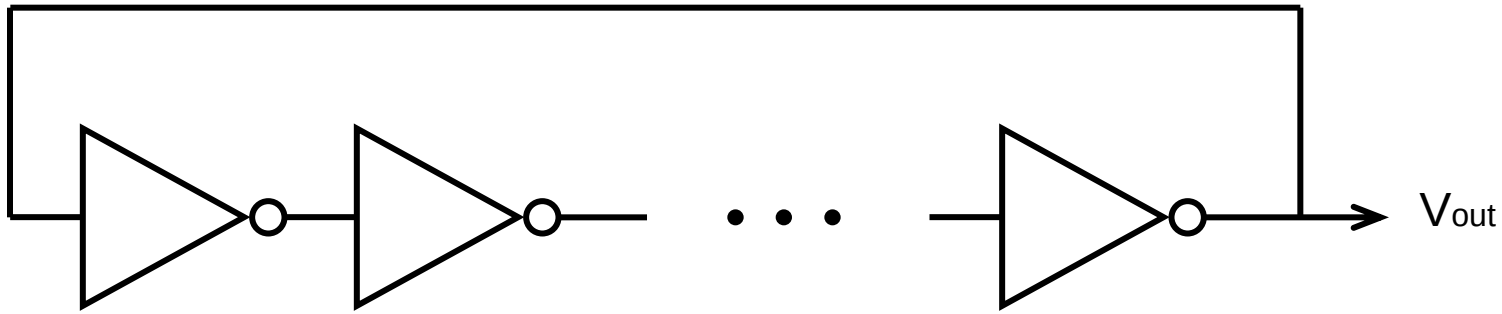
Loop Filter



Differential amplifier connected as a unity-gain follower is used.

Voltage Controlled Oscillator (VCO)

A **Ring Oscillator** cascades an **odd** number of inverters and feeds back the last output to first inverter (even number of inverters will be stable). It starts to oscillate spontaneously.

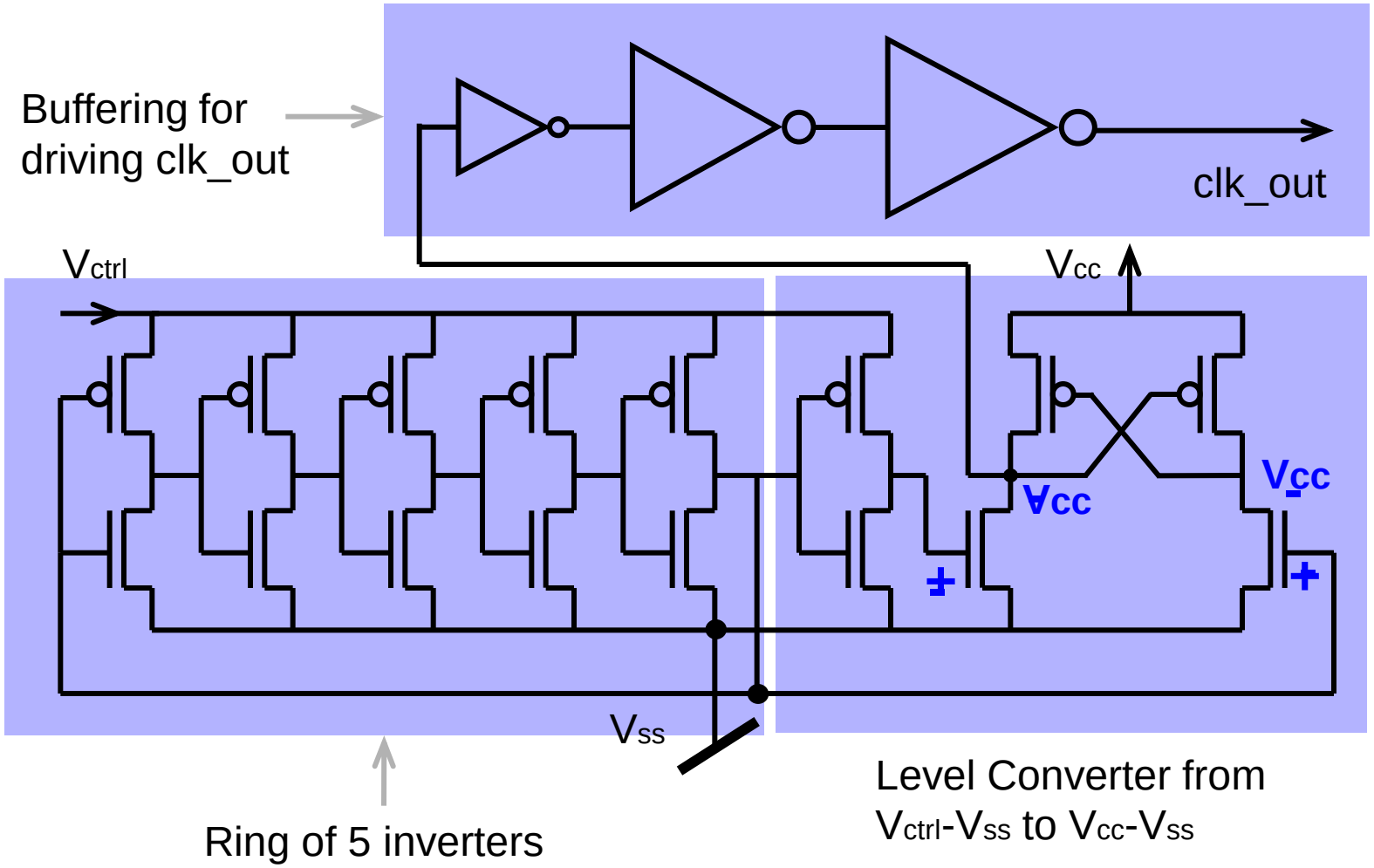


If t_{inv} is the delay of an inverter and n is the number of inverters,

oscillation frequency is $f = 1/(2nt_{inv})$

Frequency can be controlled by number of inverters and supply voltage of inverter (higher voltage obtains faster inverter).

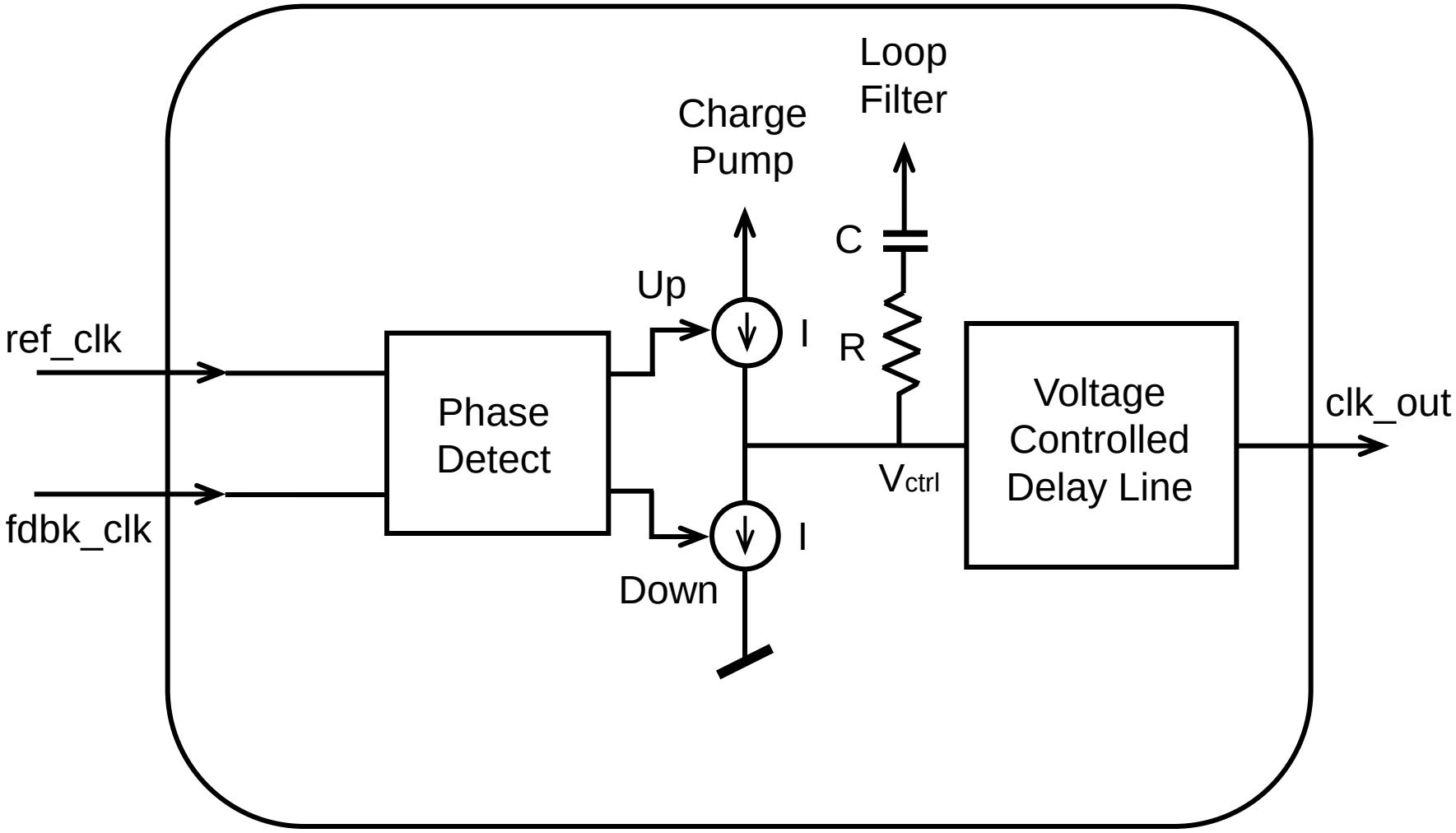
Components of VCO



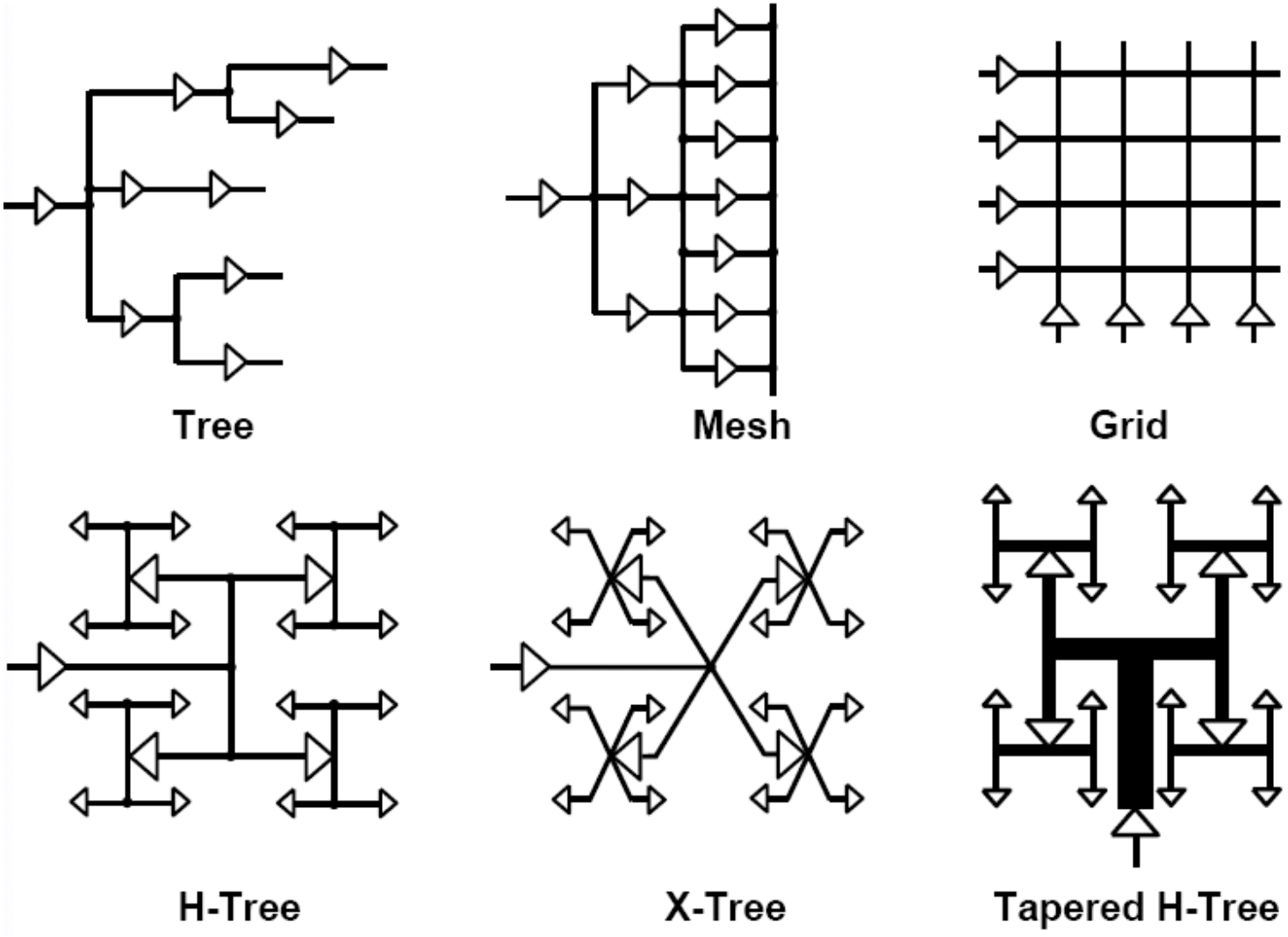
Delay Locked Loop (DLL)

- It is a variant of PLL that uses voltage-controlled delay line rather than oscillator.
- It adjusts phase only. Frequency multiplication is impossible.
- It is simpler than PLL, less sensitive to Vctrl noise and requires simpler loop filter.
- It is very difficult to correctly design PLL and DLL. It requires expertise in control systems and analog circuit design.

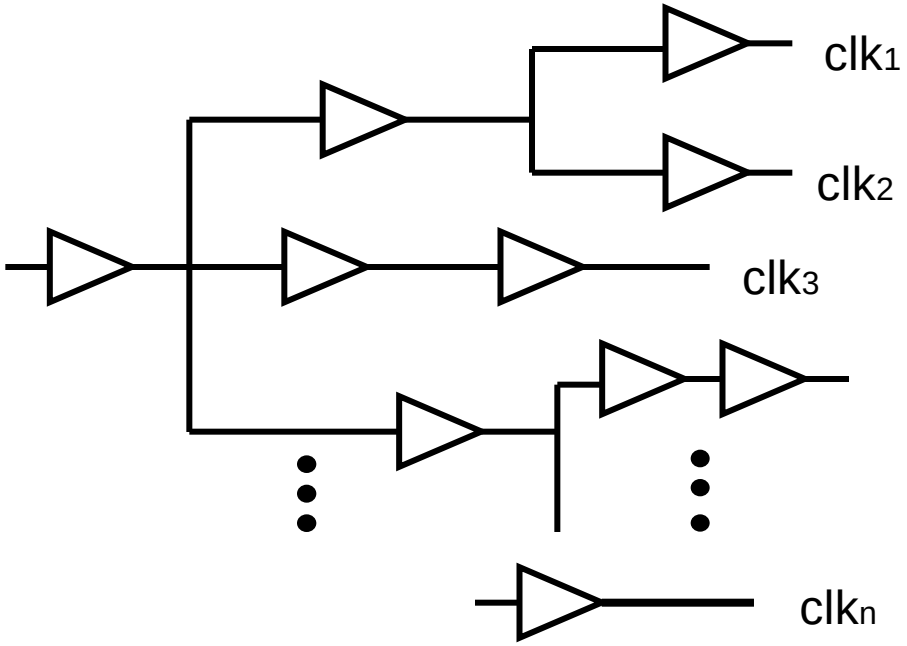
How DLL Works?



Clock Distribution Networks



Tree Clock Network (Unconstrained)



No constraints imposed on buffers and wires.

Used mostly by automatic tools in automatic synthesis flows.

Can be used for small blocks within large design.

Tools aim at minimizing the variance of clock delays.

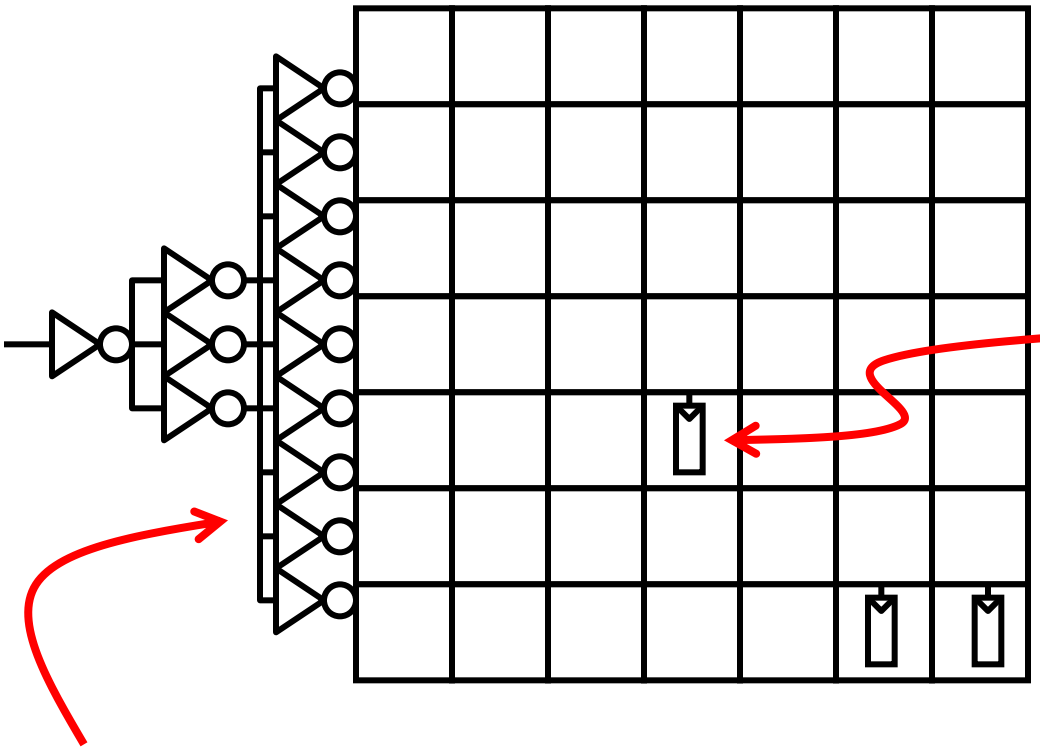
If T_{CLK_i} is the clock delay at a leaf, the variance exapression

$$\sum_{i=1}^{n}\left[T_{CLK_i}-\frac{1}{n}\sum_{i=1}^{n}T_{CLK_i}\right]^2$$ should be minimized.

Serpentine routing or extra buffers may be introduced to obtain small variance.

Constraints on power can be imposed by limiting number and size of clock buffers and width of wires.

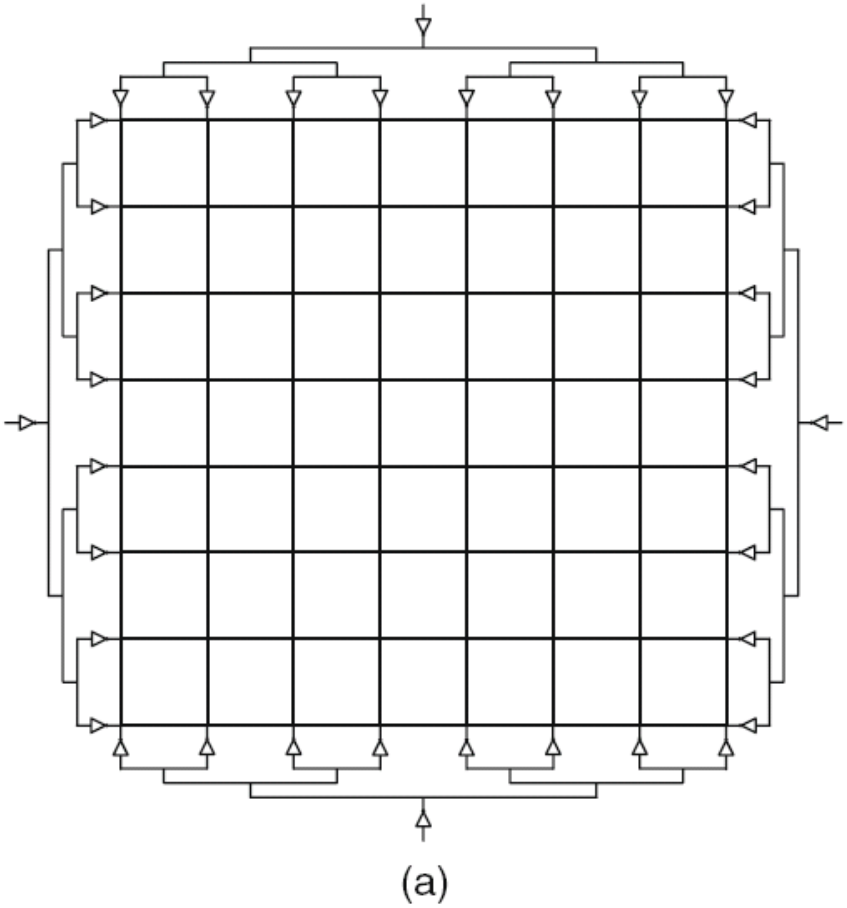
Clock Distribution with Grids



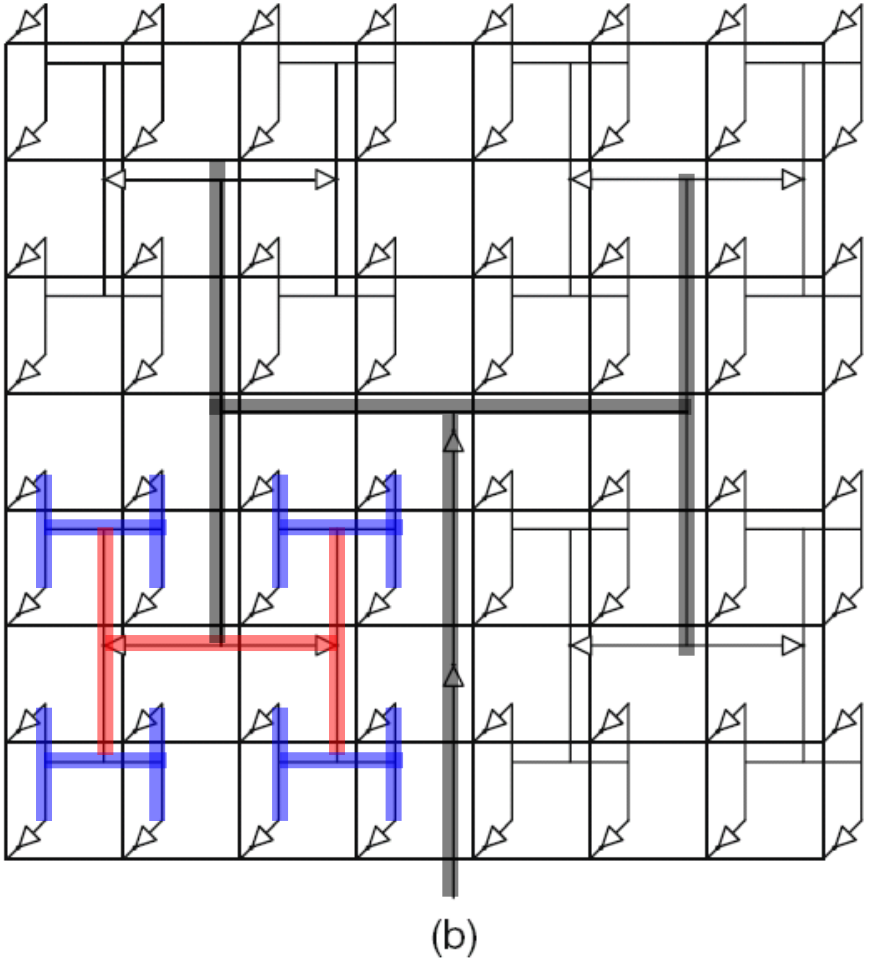
Grid feeds flops directly, no local buffers

Clock driver tree spans height of chip
Internal levels shorted together

Low skew but high power

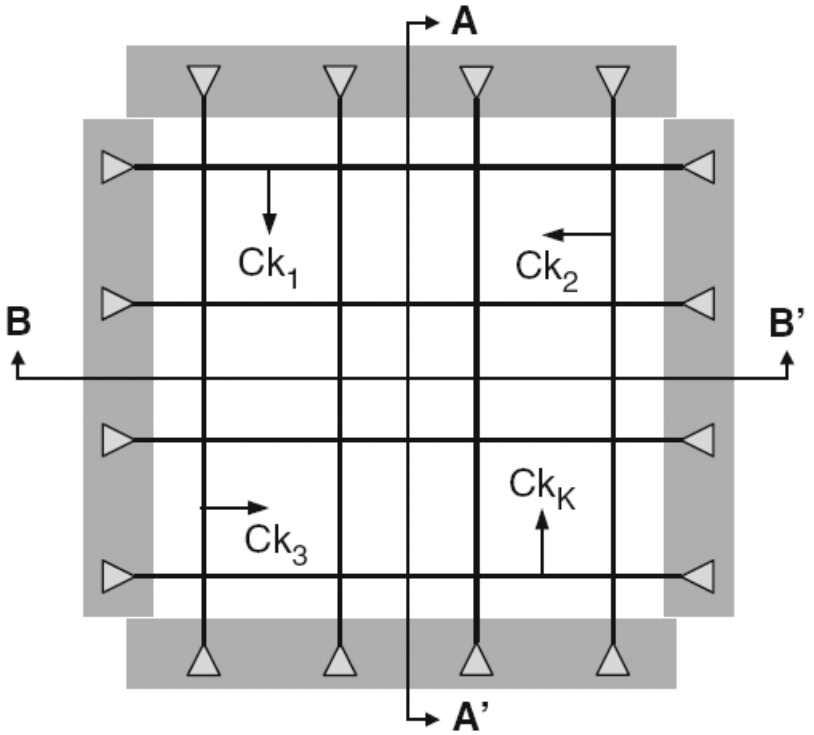


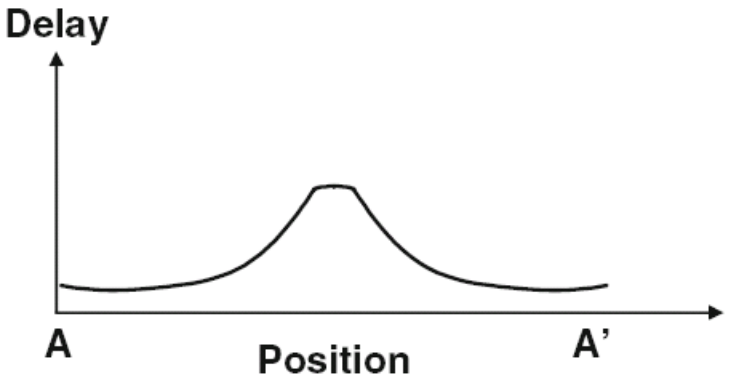
Clock drivers are on perimeter

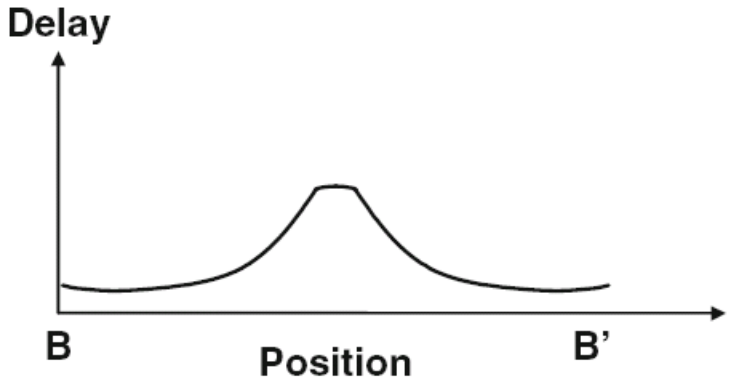


Clock drivers are on grid points

Delay and Skew in Grid Distribution







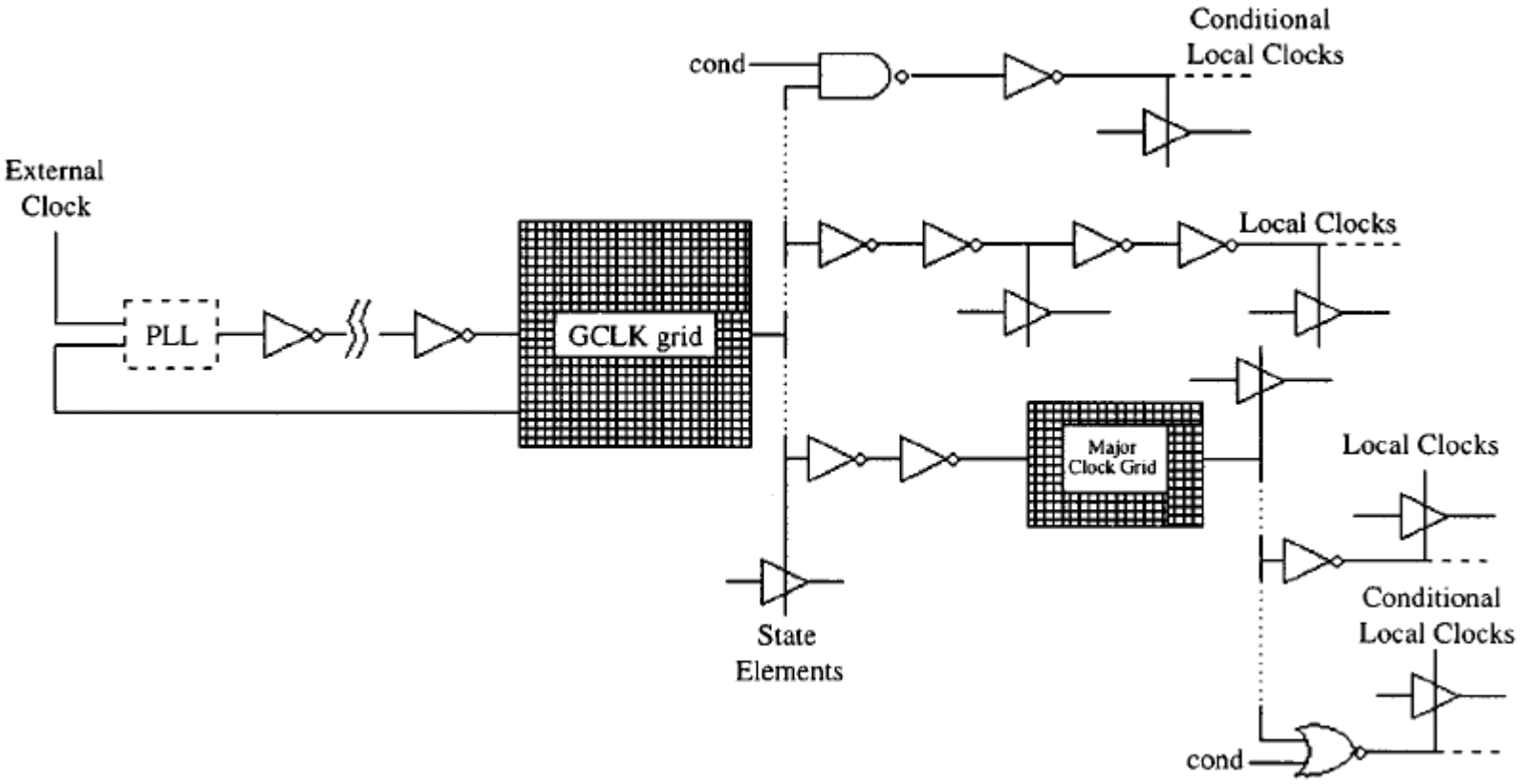
DEC's Alpha Microprocessor Clocking

Product
Frequency
Transistors
Process
Power
Clock load

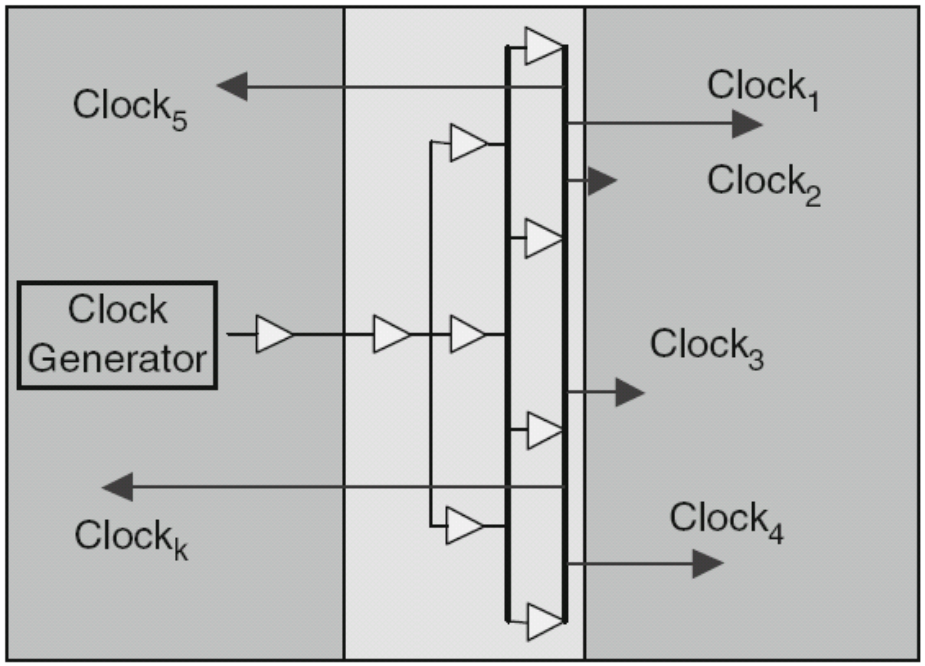
Clock Floorplan

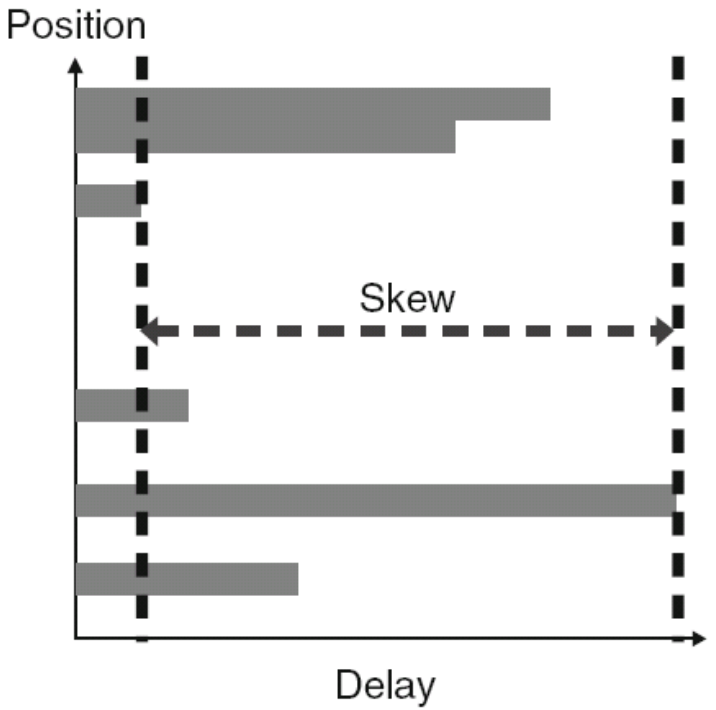
Clock skew plot

DEC Alpha 21264 Microprocessor Clock distribution

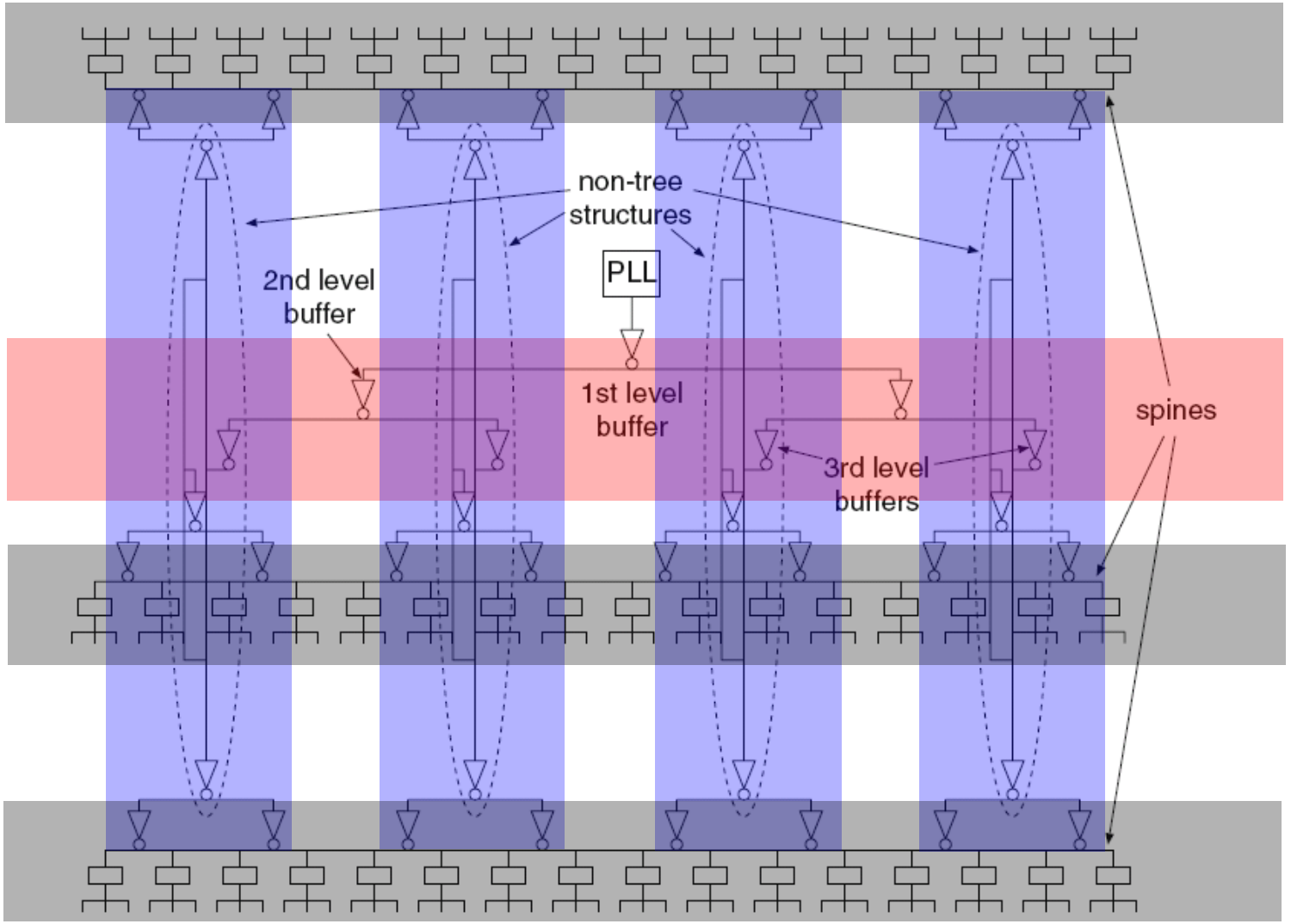


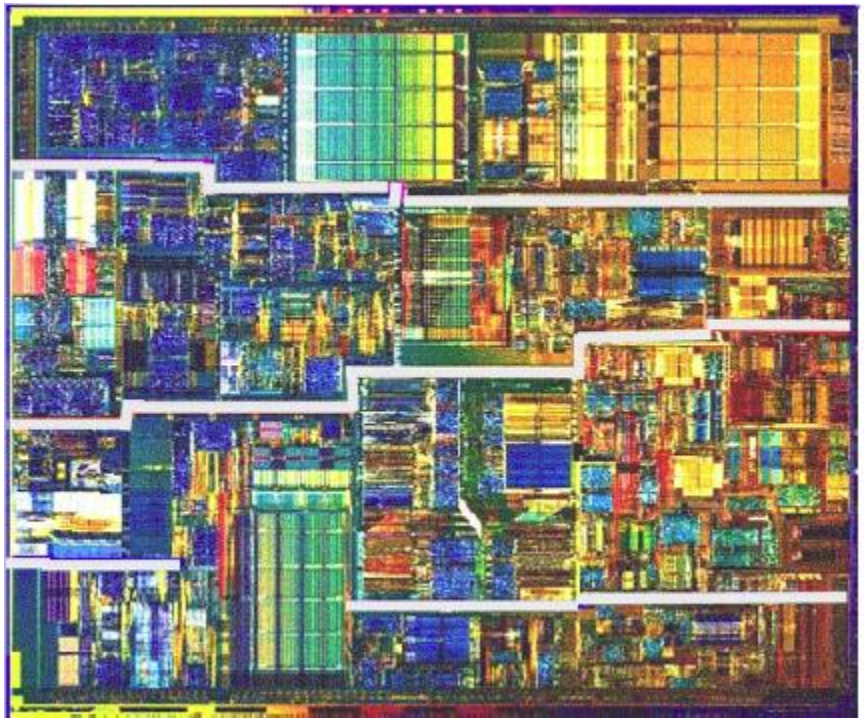
Clock Distribution with Spines

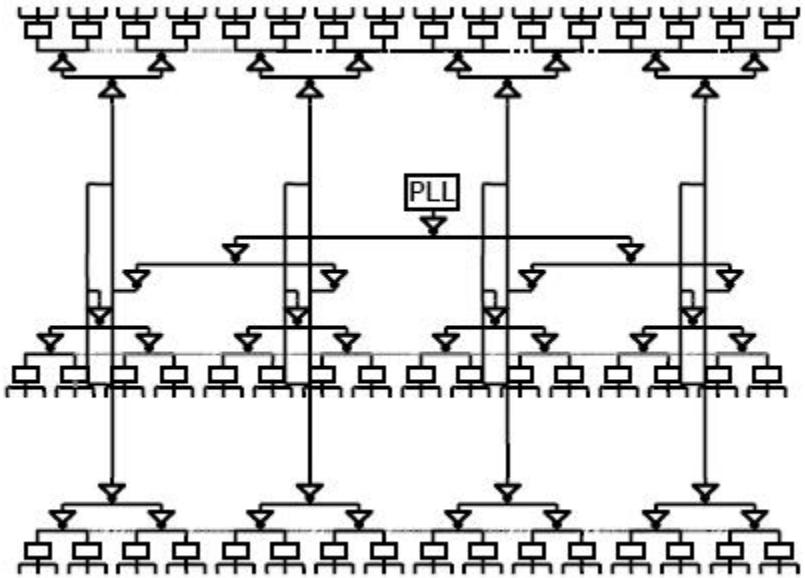




Intel's Pentium4 Clock Distribution

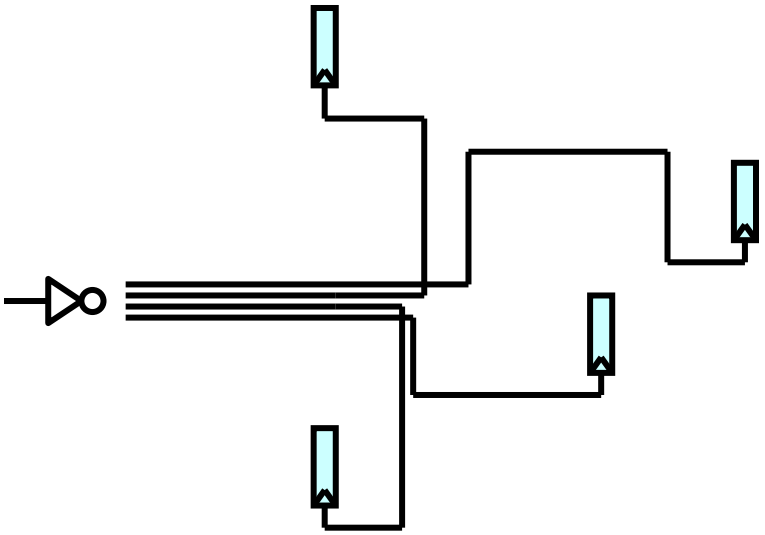





PLL

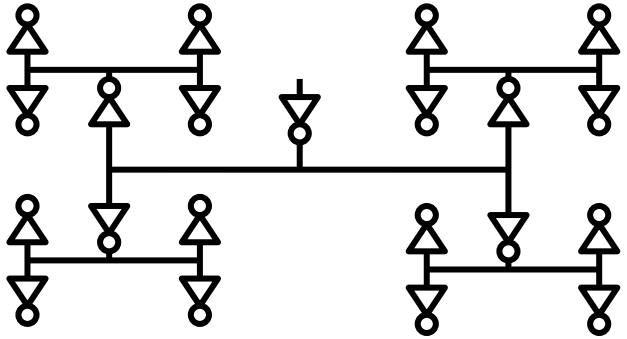
Clock Distribution with Trees

RC-Tree



Each branch is individually routed to balance RC delay

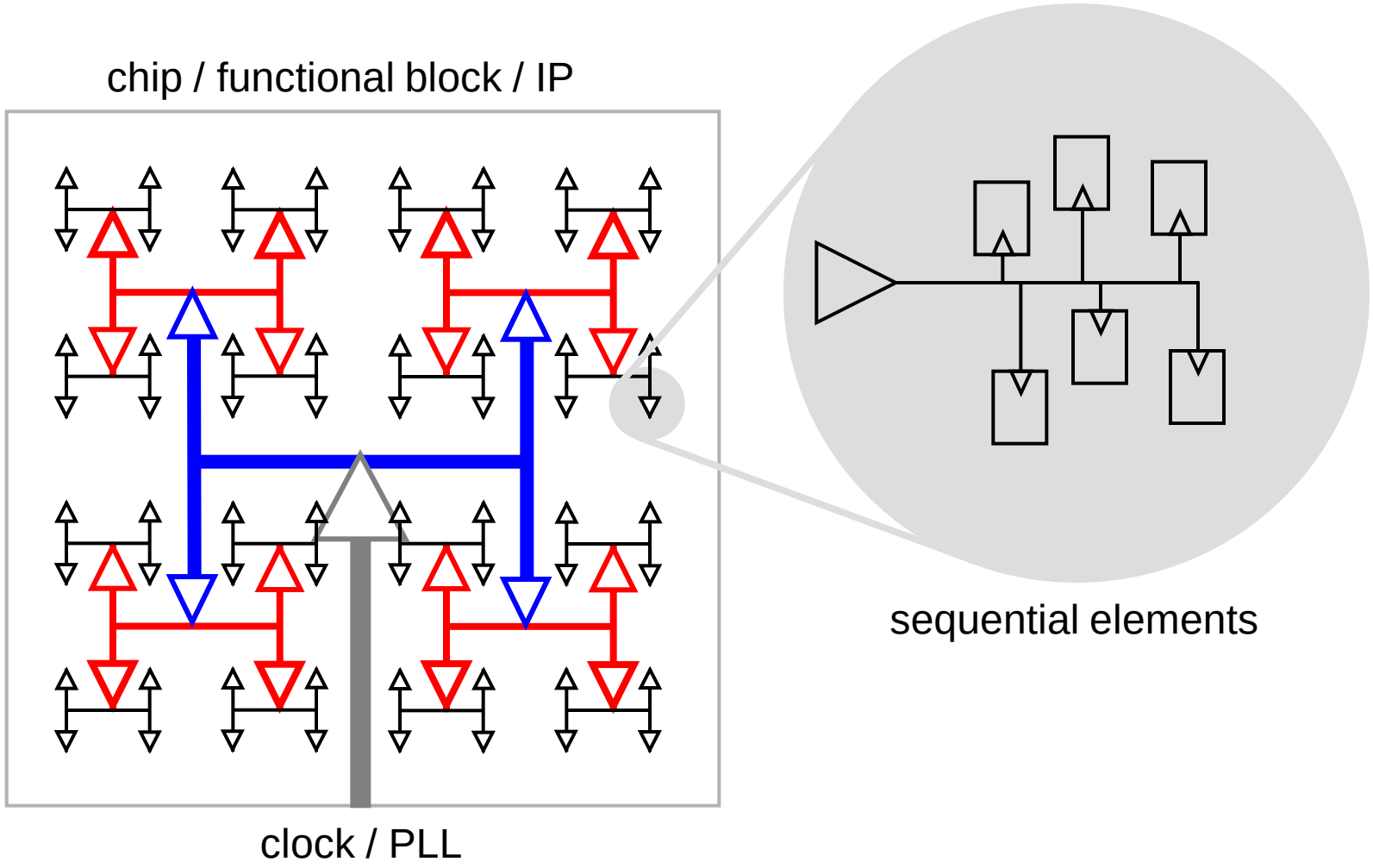
H-Tree



Recursive pattern to distribute signals uniformly with equal delay over area

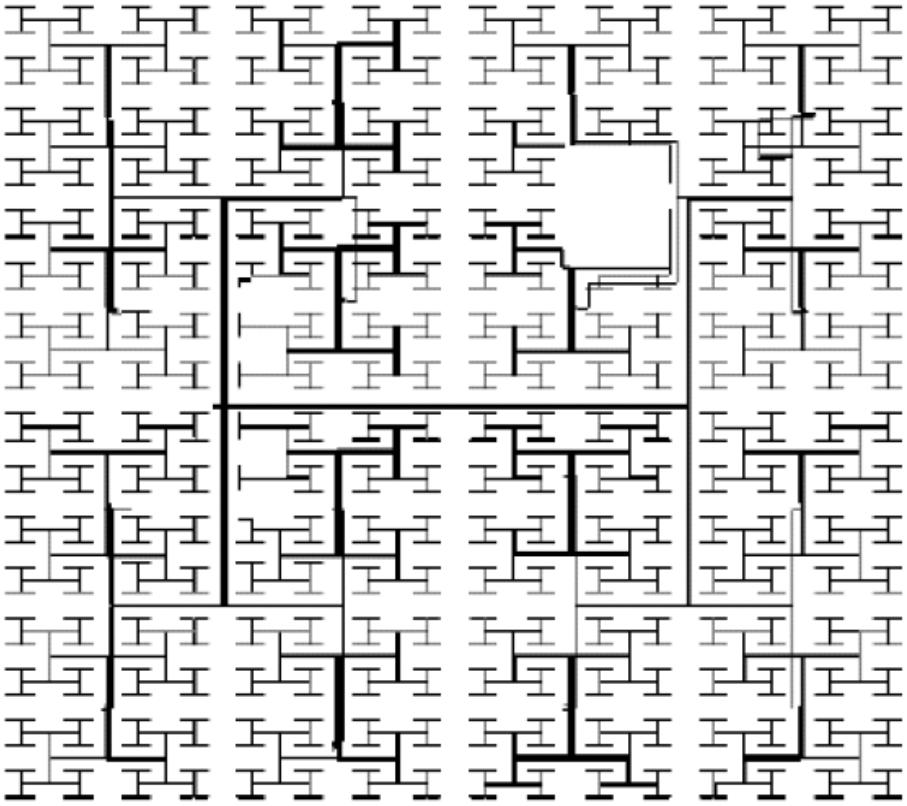
More skew but less power

Clock H-Tree

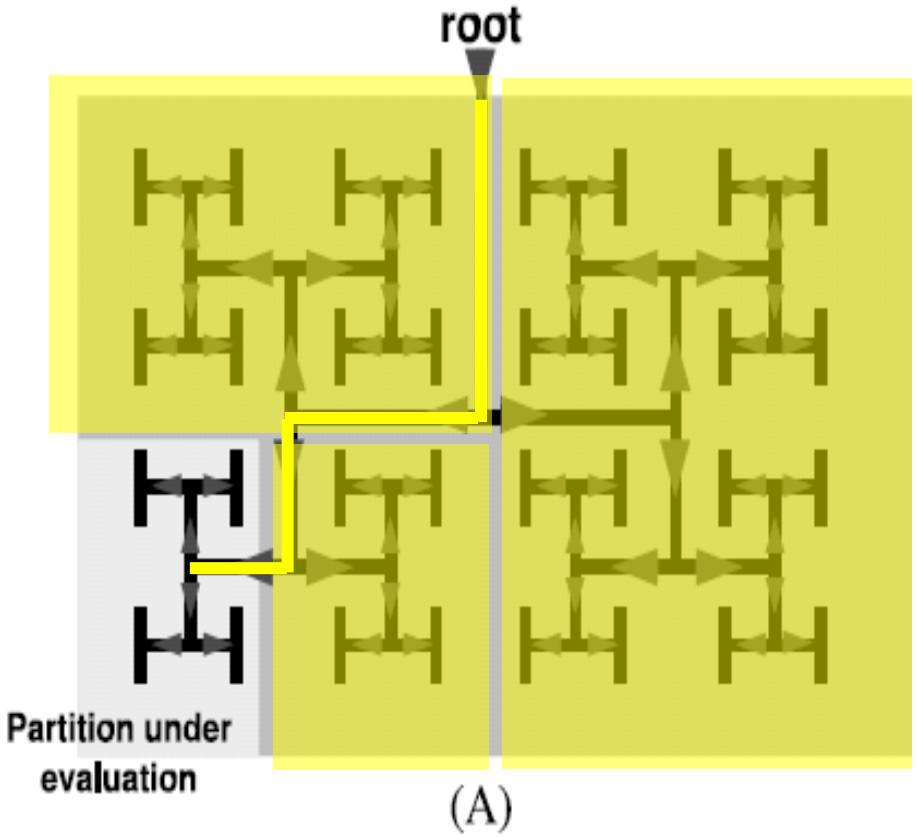


IBM / Motorola PowerPC Clock Distribution

- 0.22µm technology
- 17mm x 17mm die size
- 19M transistors
- 6 level metal with copper interconnect technology
- Clock tree on top 2 metal levels
- 1 GHz clock frequency
- Almost symmetric H-tree
- Simulated clock skew under 15ps



Delay Calculation



We use Elmore delay model. Sub trees are modeled as capacitive loads

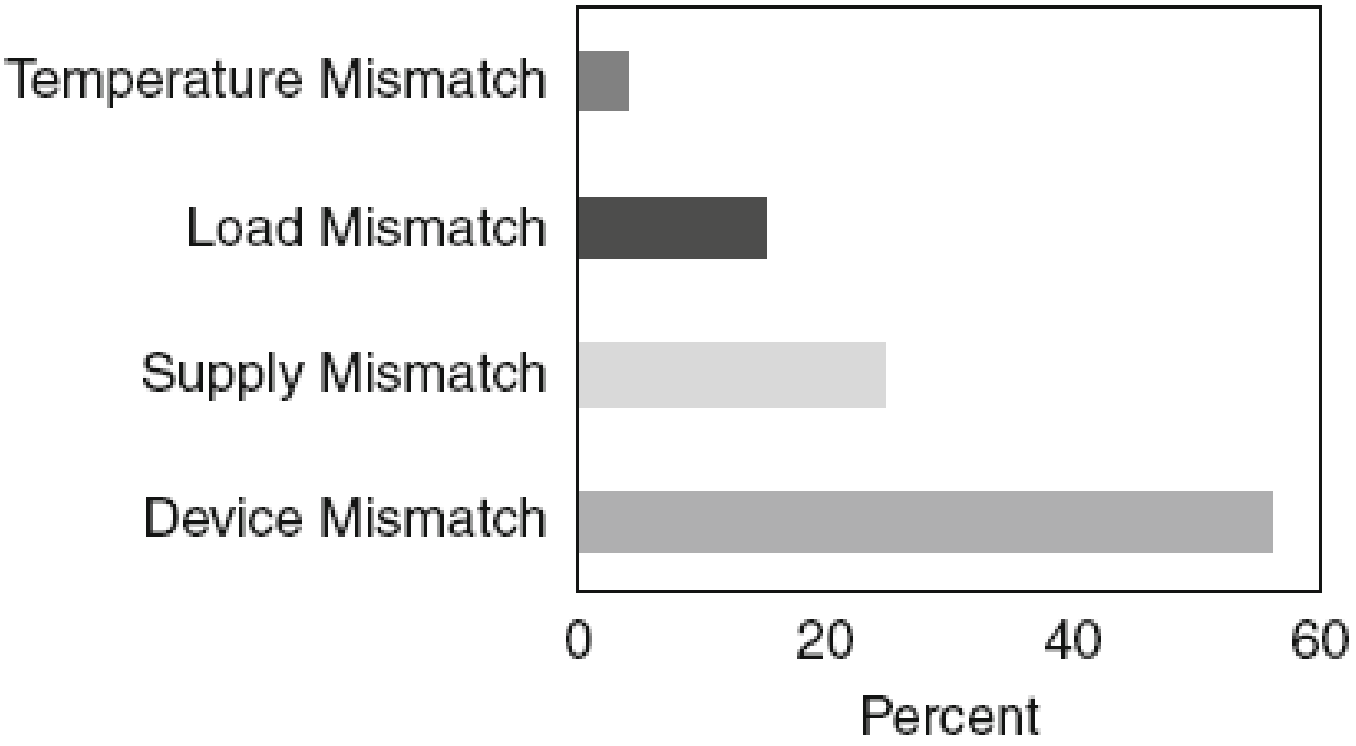
Clock Skew and Jitter

- Clock should theoretically arrive simultaneously to all sequential circuits.
- Practically it arrives in different times. The differences are called ***clock skews***.
- Skews result from paths mismatches, process variations and ambient conditions, resulting ***physical clocks***.
- Most systems distribute a ***global clock*** and then use local ***clock gaters*** located near clocked elements.

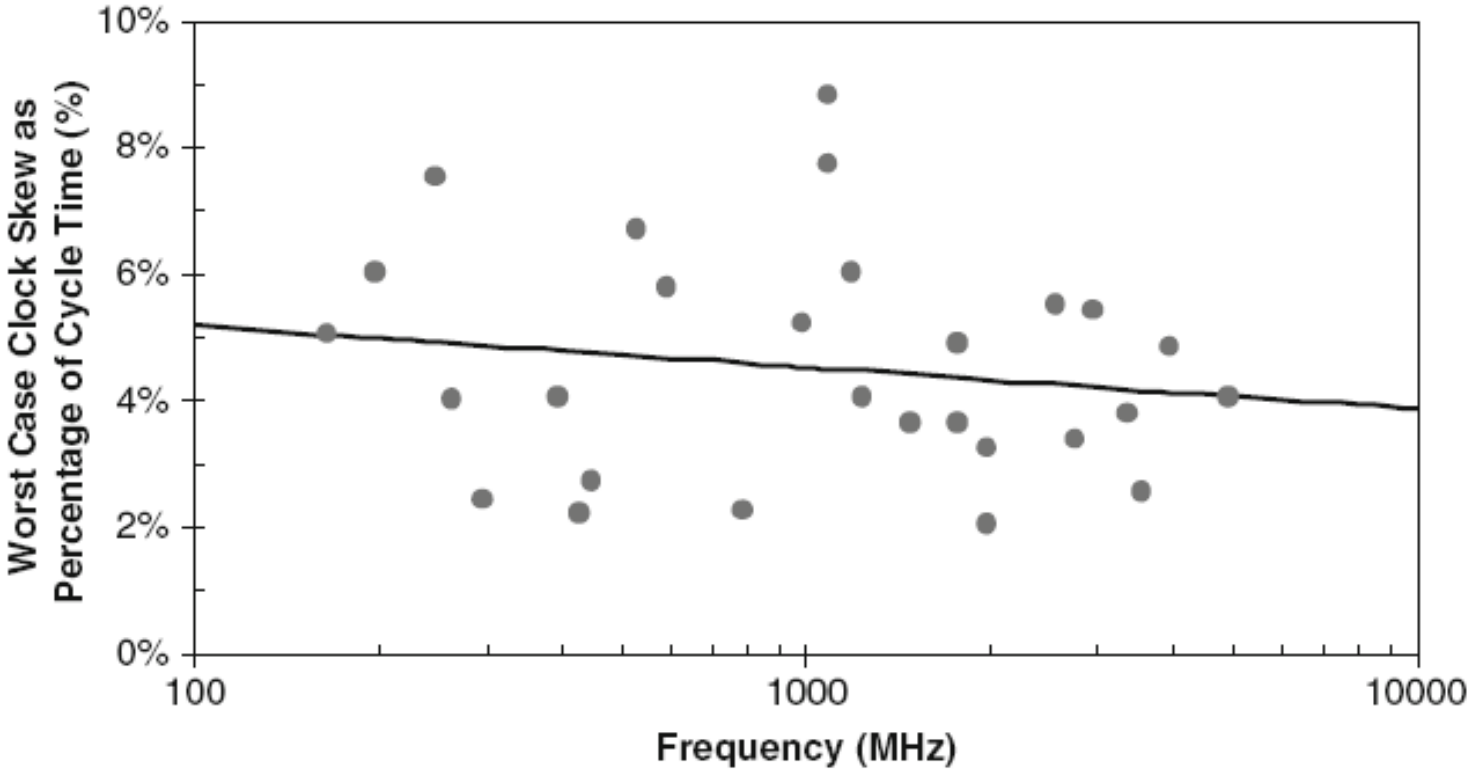
Clock skew consists of the following components:

- ***Systematic*** is the portion existing under nominal conditions. It can be minimized by appropriate design.
- ***Random*** is caused by process variations like devices' channel length, oxide thickness, threshold voltage, wire thickness, width and space. It can be measured on silicon and adjusted by delay components.
- ***Drift*** is caused by time-dependent environmental variations, occurring relatively slowly. Compensation of those must takes place periodically.
- ***Jitter*** is rapid clock changes, occurring by power noise and clock generator jitter. It cannot be compensated.

Factors affecting clock skew, Intel 1998, 0.25u.

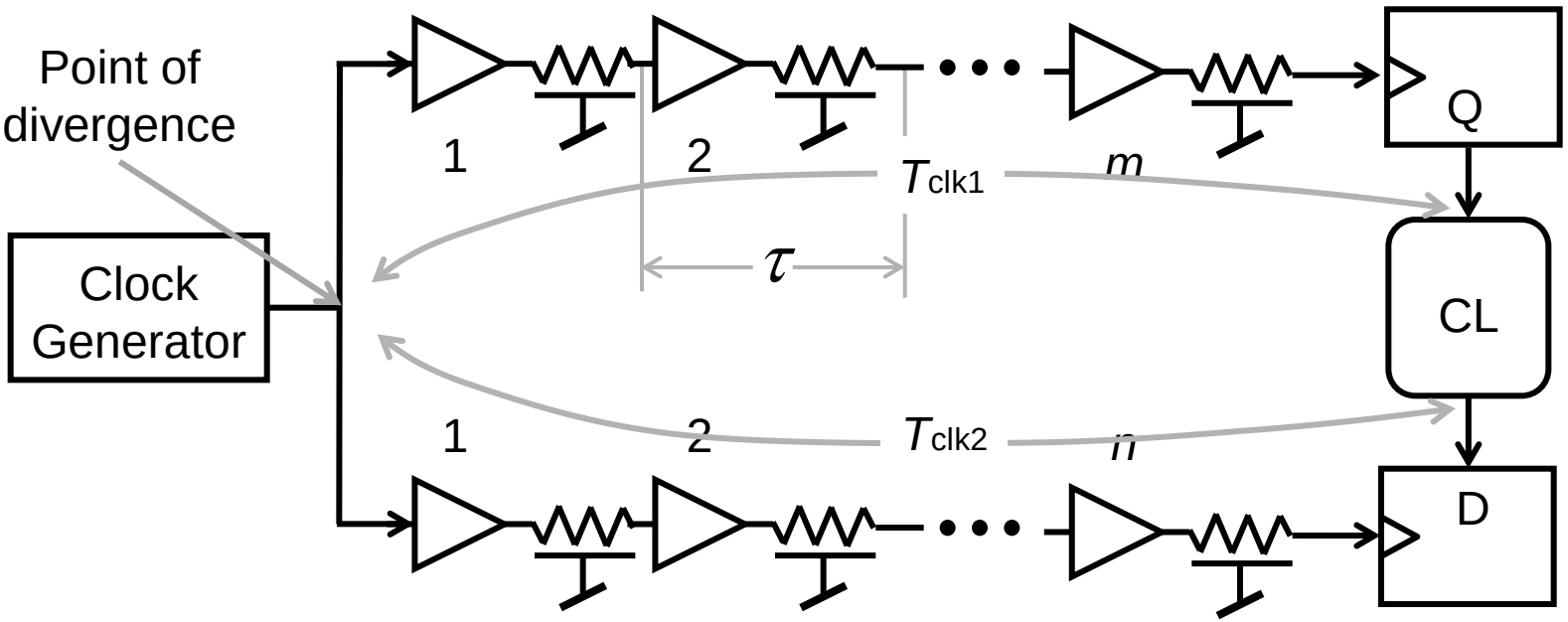


Skew, Clock Cycle and Design Margins



Clock Jitter is the same order as skew, but far more difficult to compensate.

Skew Modeling



$$T_{\mathrm{CLK1}} = \sum_{i=1}^{m} T_i \cong m\tau \qquad T_{\mathrm{CLK2}} = \sum_{i=1}^{n} T_i \cong n\tau$$

C_{l} - capacitive load, I_{d} - drive current, V_{CC} - voltage swing

$$\tau = C_{\mathrm{l}} V_{\mathrm{CC}} / I_{\mathrm{d}}$$

Consider a small change in the delay, taking the linear term.

$$\Delta\tau = \frac{\partial\tau}{\partial V_{CC}}\Delta V_{CC} + \frac{\partial\tau}{\partial C_l}\Delta C_l + \frac{\partial\tau}{\partial I_d}\Delta I_d = \frac{C_l}{I_d}\Delta V_{CC} + \frac{V_{CC}}{I_d}\Delta C_l - \frac{C_l V_{CC}}{I_d{}^2}\Delta I_d$$

$$= \tau\left(\frac{\Delta V_{CC}}{V_{CC}} + \frac{\Delta C_l}{C_l} - \frac{\Delta I_d}{I_d}\right) = \alpha\tau$$

Standard deviation of stage delay is $\sigma(\tau)=\alpha\tau$. α is around 5%.

If clock buffers delays are normally distributed and independent of each other, then $\sigma(T_{CLK1}) = \sqrt{m}\alpha\tau$, and $\sigma(T_{CLK2}) = \sqrt{n}\alpha\tau$.

Skew is: $\sigma(T_{skew}) = \sigma(|T_{CLK2} - T_{CLK1}|) = \left(\sqrt{m+n}\right)\alpha_{skew}\tau$.

Clock Distribution Switching Power

Consider m-level clock tree.

Let C_{l_m} be the total load of its far-end driven sequential elements.

Assuming a fixed fanout k of each clock tree buffer, the dynamic power is: $P_{CLK\_m} = C_{l\_m} V_{CC}^2 f$,

$$P_{CLK\_(m-j)} = \frac{C_{l\_m}}{k^j} V_{CC}^2 f,\ 0 \le j \le m-1.$$

Summing over whole clock tree:

$$P_{\mathrm{CLK}} = \sum_{j=0}^{m-1} C_{\mathrm{l\_}(m-j)} V_{\mathrm{CC}}^2 f =$$

$$V_{\mathrm{CC}}^2 f \sum_{j=0}^{m-1} \frac{C_{\mathrm{l\_}m}}{k^j} = V_{\mathrm{CC}}^2 f C_{\mathrm{l\_}m} \left[\frac{1-(1/k)^m}{1-(1/k)} \right]$$

How much of the power is consumed by the far end drivers of clock tree?

$$\frac{P_{\mathrm{CLK\_}m}}{P_{\mathrm{CLK}}} = \frac{1-(1/k)}{1-(1/k)^m}.$$

Given the number of sequential elements in a block, at least 50% of the switching power is consumed by the far end drivers (clock tree is binary, k=2). This number approaches 1 rapidly with k growth.

Example: Assume a block with 2^{14} sequential elements and H-tree clock distribution. Then k=4 and m=7. The far end drivers consume nearly 75% of the clock tree switching power, while adding the next upper level drivers brings it to more than 90%.

Active Clock De-Skewing

- Compensates process variability, temperature gradients, imperfect design.
- Can be implemented for global fixes (small HW overhead) or local fixes (high HW overhead).
- Can be used at testing for one time fix (variability occurring during manufacturing), or dynamically concurrently with chip operation.
- Its implementation is a difficult design challenge.

Intel's Pentium2 De-Skewing System

1998, 450MHz Clock

0.25u process

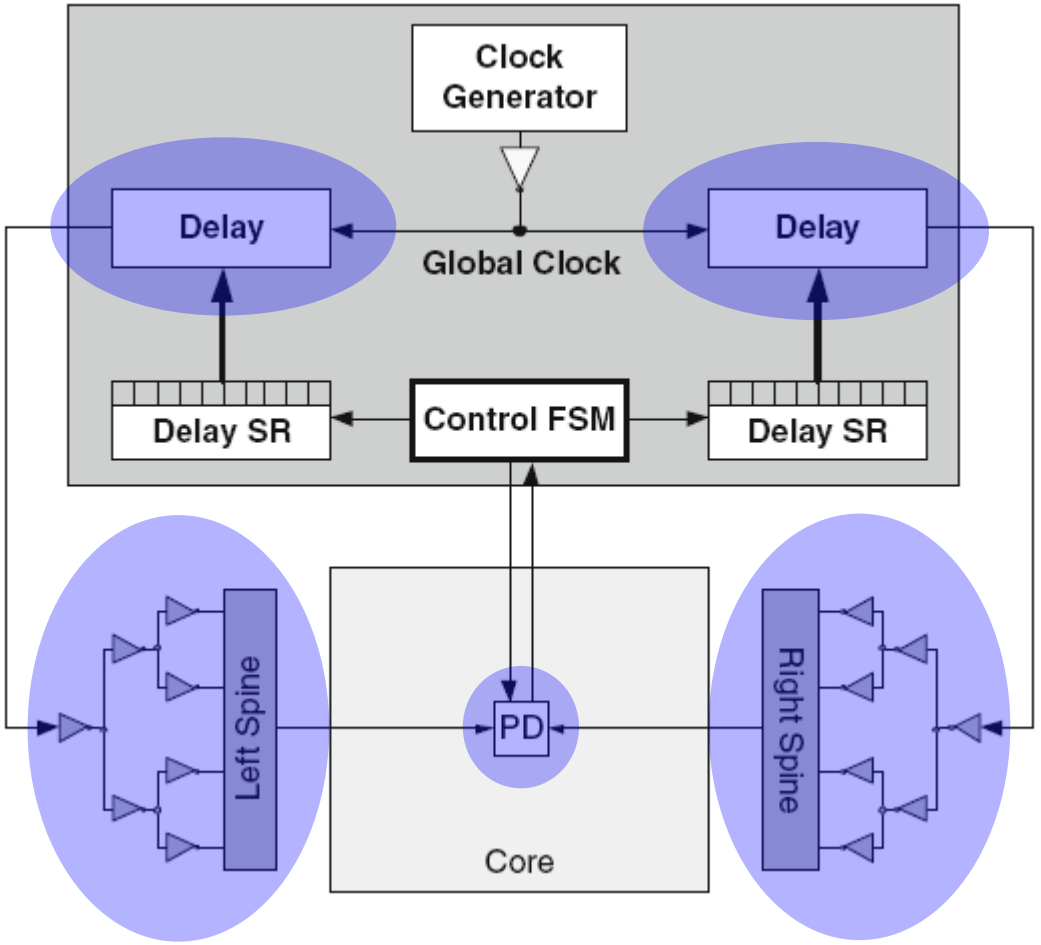
60pSec skew w/o fix

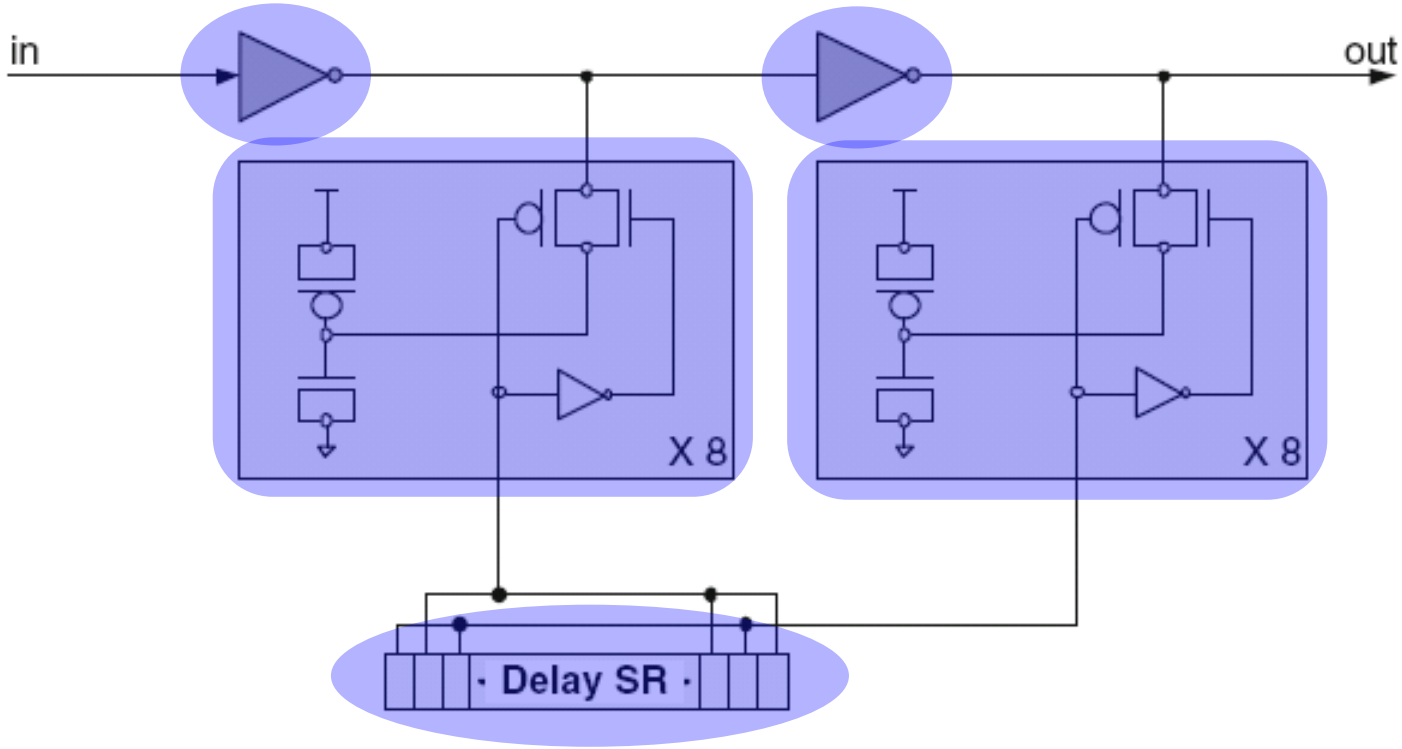
15pSec skew with fix

Two clock spines for two clock regions.

A phase detector detects relative shifts.

Clock of a region is shifted by a delay line.





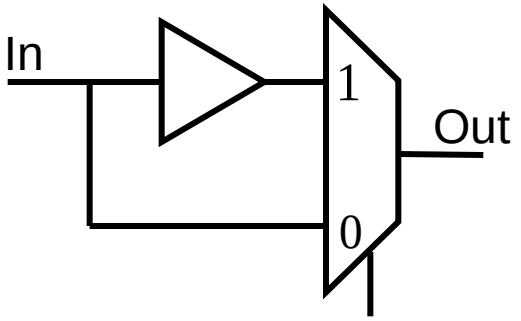
Delay line consists of two cascaded inverters.

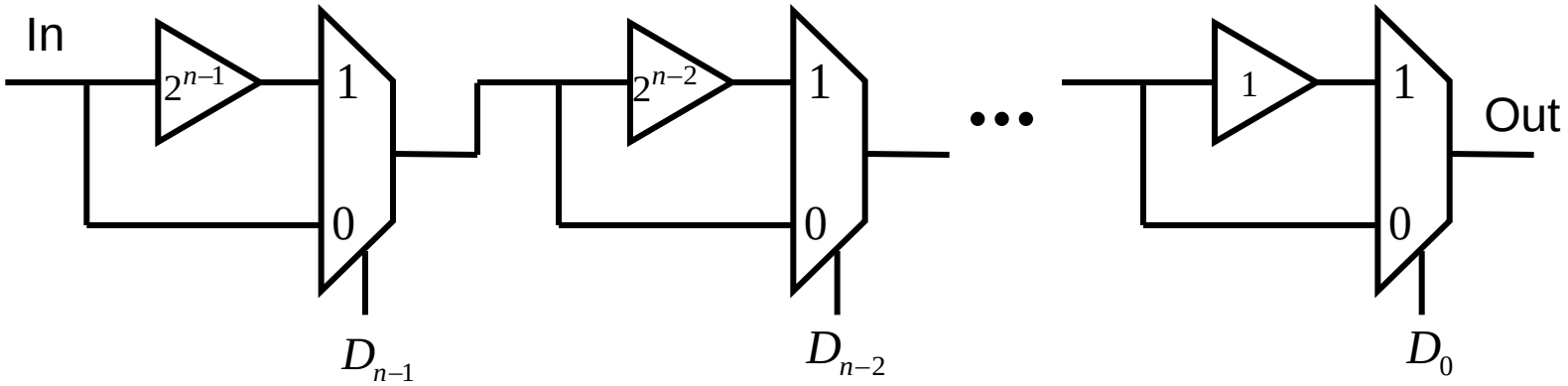
Each has a programmable load consists of eight parallel P-N gate capacitors.

The shift register stores a thermometer code for load programming in steps of 12pSec.

Another Programmable Delay Line

The signal from input to output is delayed or not according to the control bit.





Connecting a series of n delay elements, each of delay 2^i, $n-1 \geq i \geq 0$, it is possible to control the dely of the line to any value from 0 to $2^n - 1$ dealy units by an n-bit control word $(D_{n-1}, D_{n-2}, \ldots, D_0)$.

Intel's IA64 Itanium1 De-Skewing System

2000, 800MHz clock

0.18u process

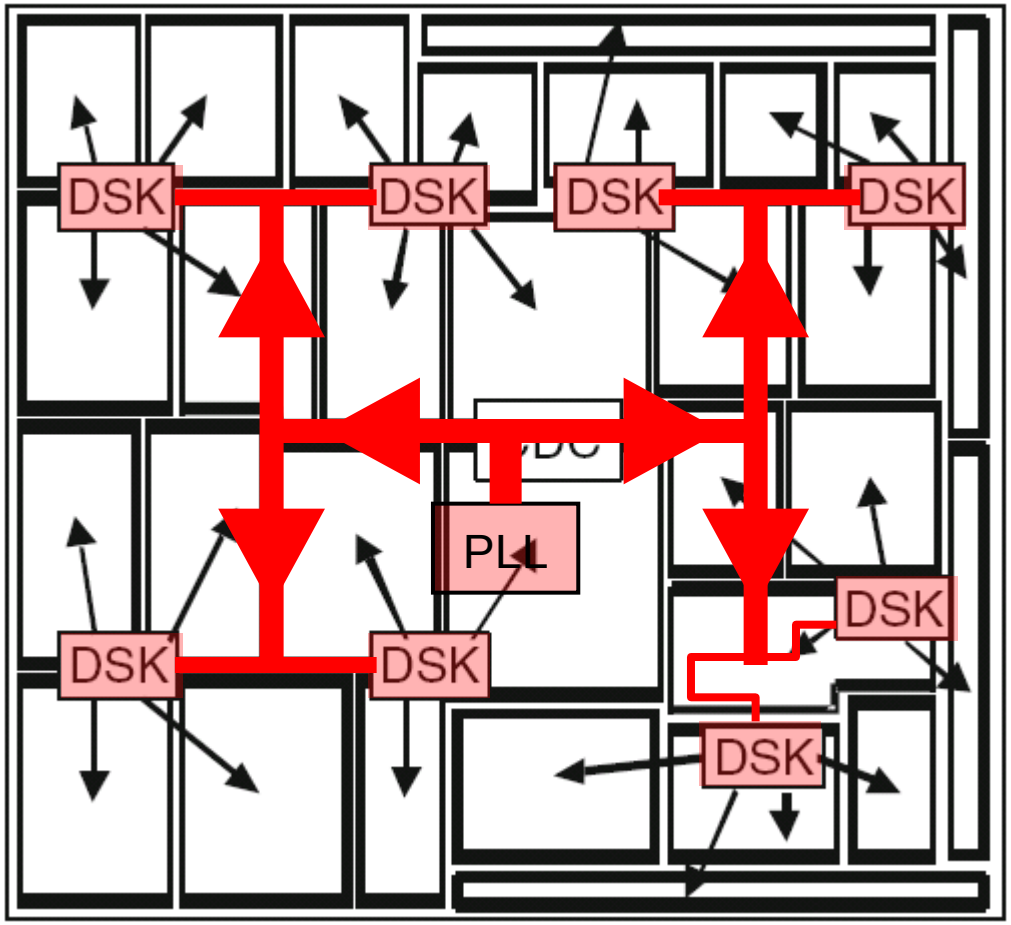
28pSec skew with fix

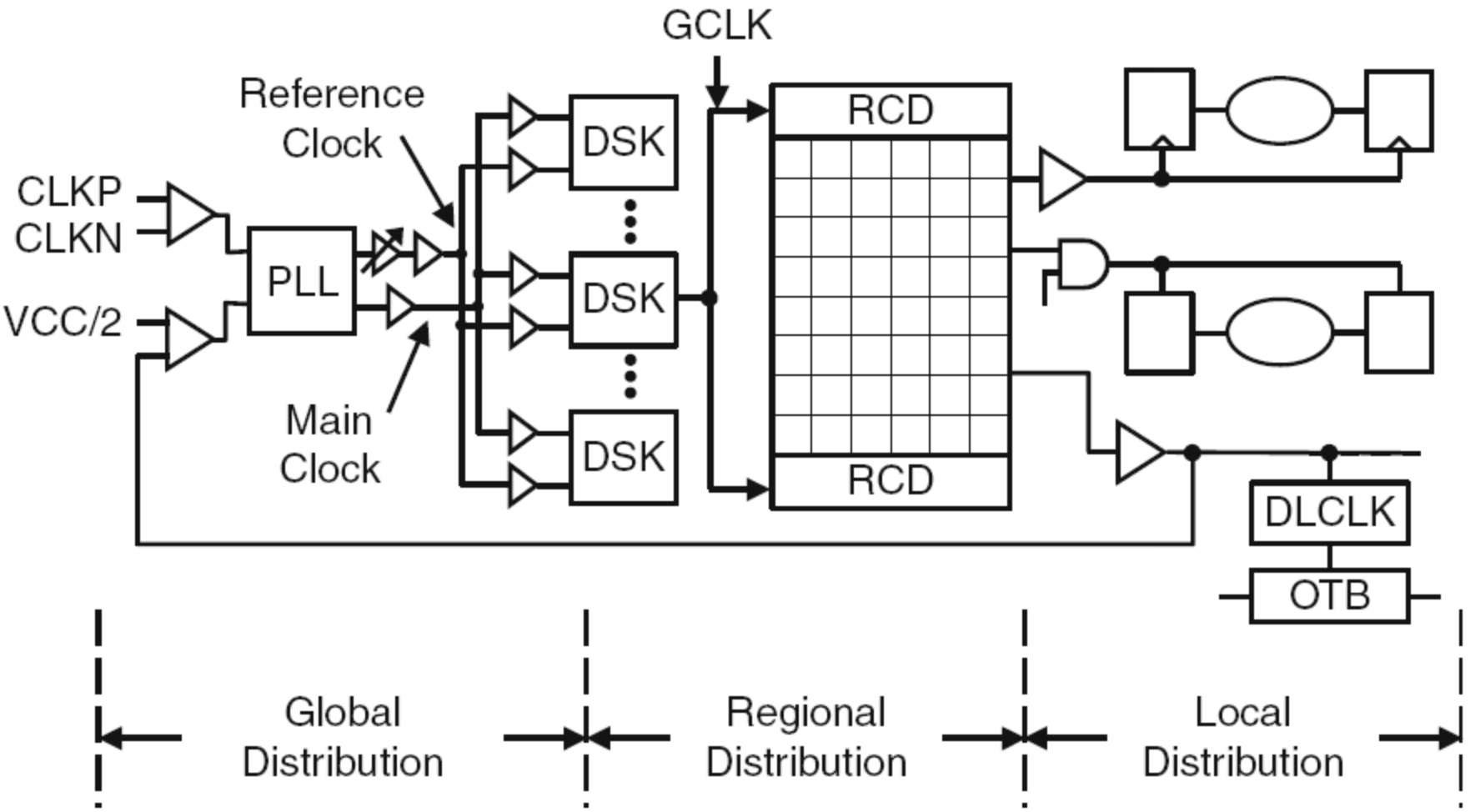
X4 increase w/o fix

30 independent de-skew regions.

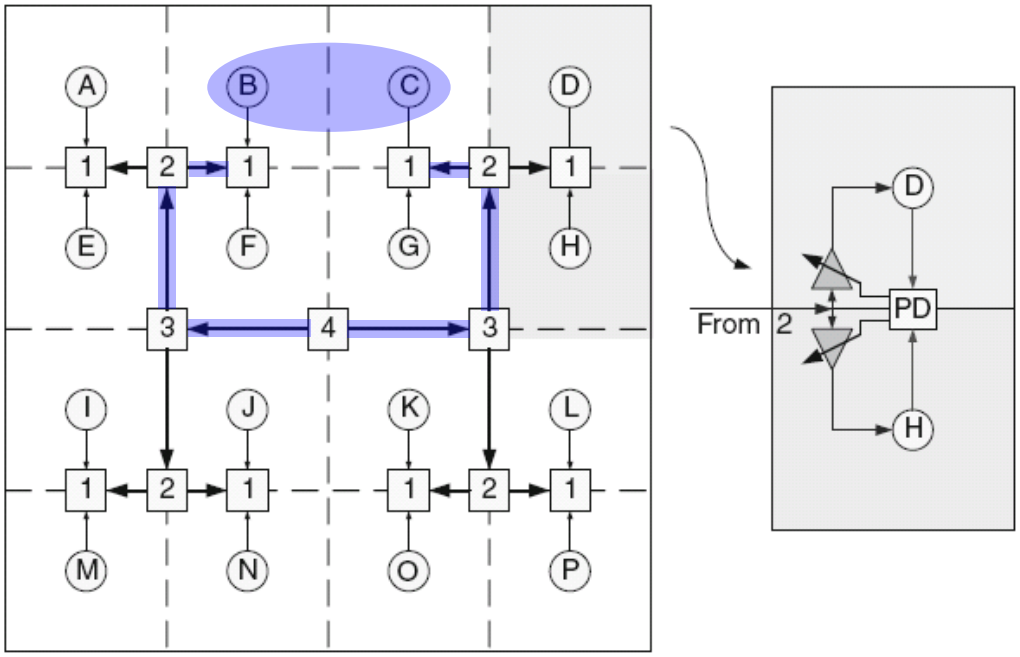
Each cluster is driven from a global H-tree.

Delay circuit in de-skew region are similar to Pentium3 with 20-bit registers.




GCLK
Reference
Clock
CLKP
CLKN
VCC/2
PLL
Main
Clock
DSK
DSK
DSK
RCD
RCD
DLCLK
OTB
Global
Distribution
Regional
Distribution
Local
Distribution

Proposal for H-Tree Clock De-Skew – Hierarchical Approach



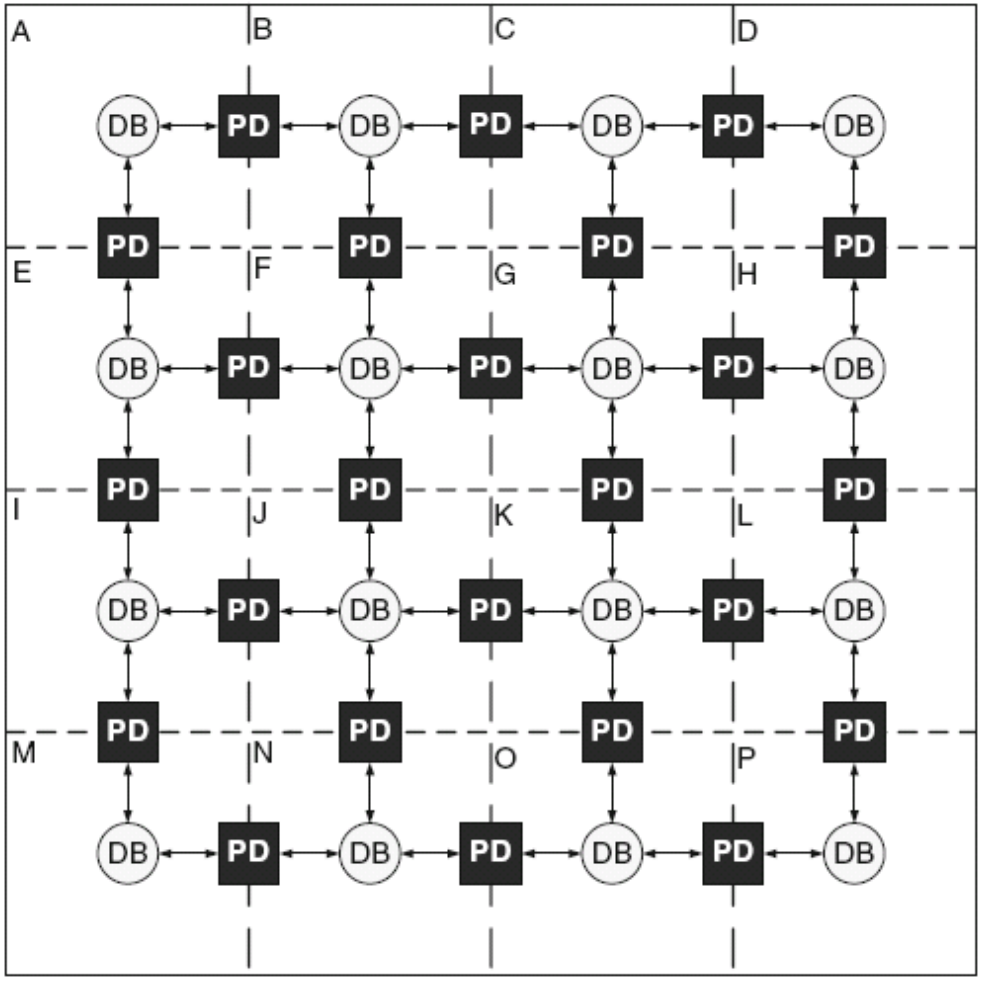
If a phase detector (PD) has a skew guard band g, then guard bands may accumulate along tree paths.

For example, if a logic stage is shared between region B and C, it may add 7g time units to path delay.

Proposal for H-Tree Clock De-Skew – Mesh Approach

Clock is distributed by H-tree, but de-skew takes place by neighbor leaves phase detection.

A delay buffer accepts phase inputs from its 4 neighbors and then decides of whether to increase, decrease or not change its delay.



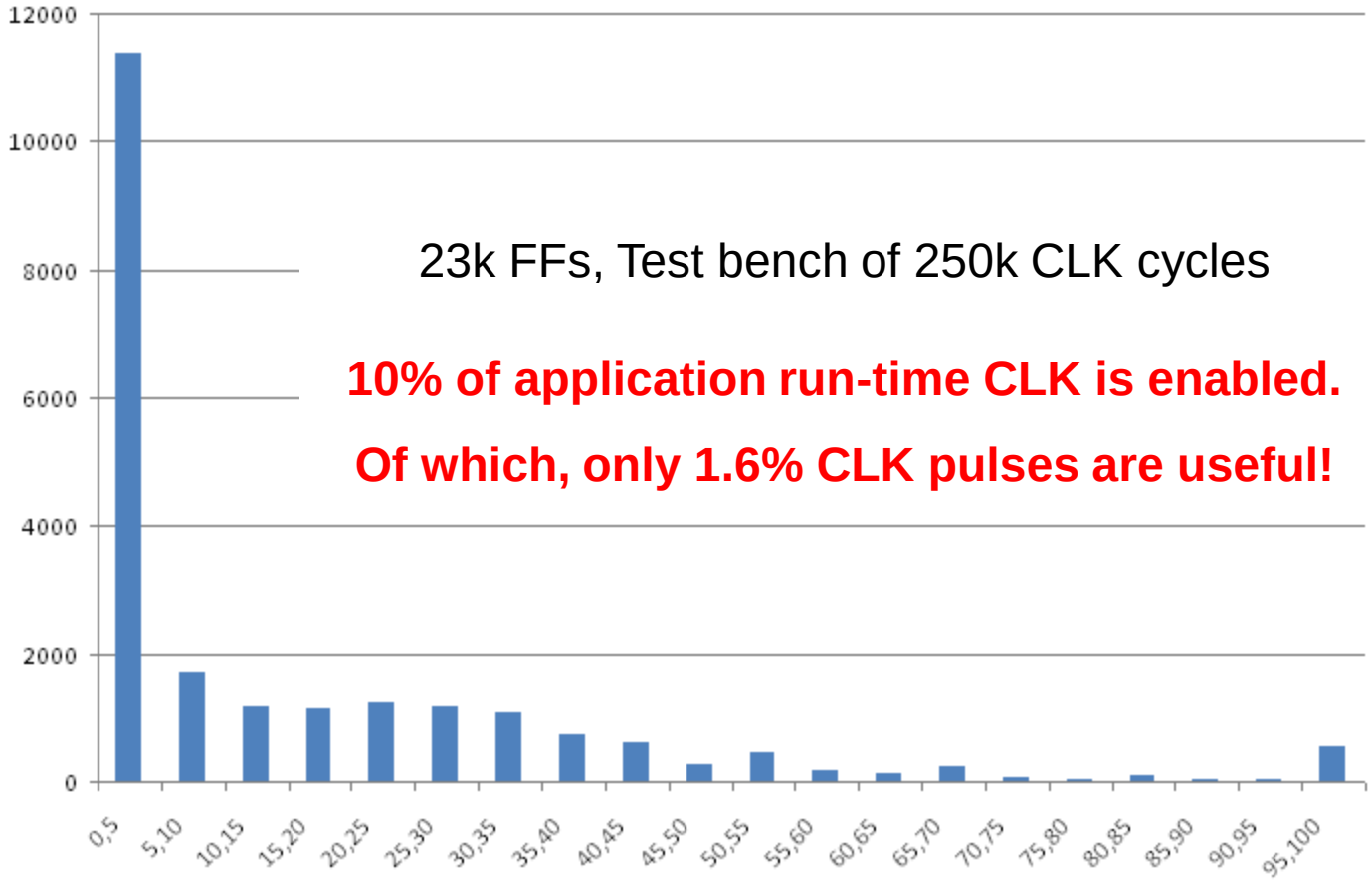
Clock Characteristics of Commercial Processors

Name	Frequency (MHz)	skew (ps)	Technology (nm)	Clock Dist. style	Deskew
Merom	3,000	18	65	Tree/Grid	Yes
Power6®	5,000	8	65	Sym. H-Tree/Grid	Yes
Quad-Core Opteron™	2,800	12	65	Tree/Grid	
Xeon® processor	3,400	11	65	Tree/Grid	Yes
Itanium® 2 processor	>2,000	10	90	Asymmetric tree	Yes
Power5®	>1,500	27	130	Sym. H-Tree/Grid	No
Pentium® 4 processor	3,600	7	90	Recombinant tile	Yes
Itanium® 2 processor	1,500	24	130	Asymmetric tree	Yes
Power4®	>1,000	25	180	Tree/Grid	No
Itanium® 2 processor	1,000	52	180	Asymmetric tree	No
Pentium® 4 processor	>2,000	16	180	Spine/Grid	Yes
Itanium® processor	800	28	180	H-Tree/Grid	Yes

Power Consumption in Chips

- Clock power may reach 50% of total (dynamic + static).
- Clock gating is very useful and standard design practice
- Four gating methods:
  - Synthesis based, automated by EDA tools, RTL compilers, inserted into clock-tree
  - Clock enable signals manually defined by designer, inserted into clock-tree, FFs' clock input
  - Data-Driven clock gating, inserted at FF-level
  - Auto-gated FF, inserted at latch-level

FF Data Toggling (DSP core)



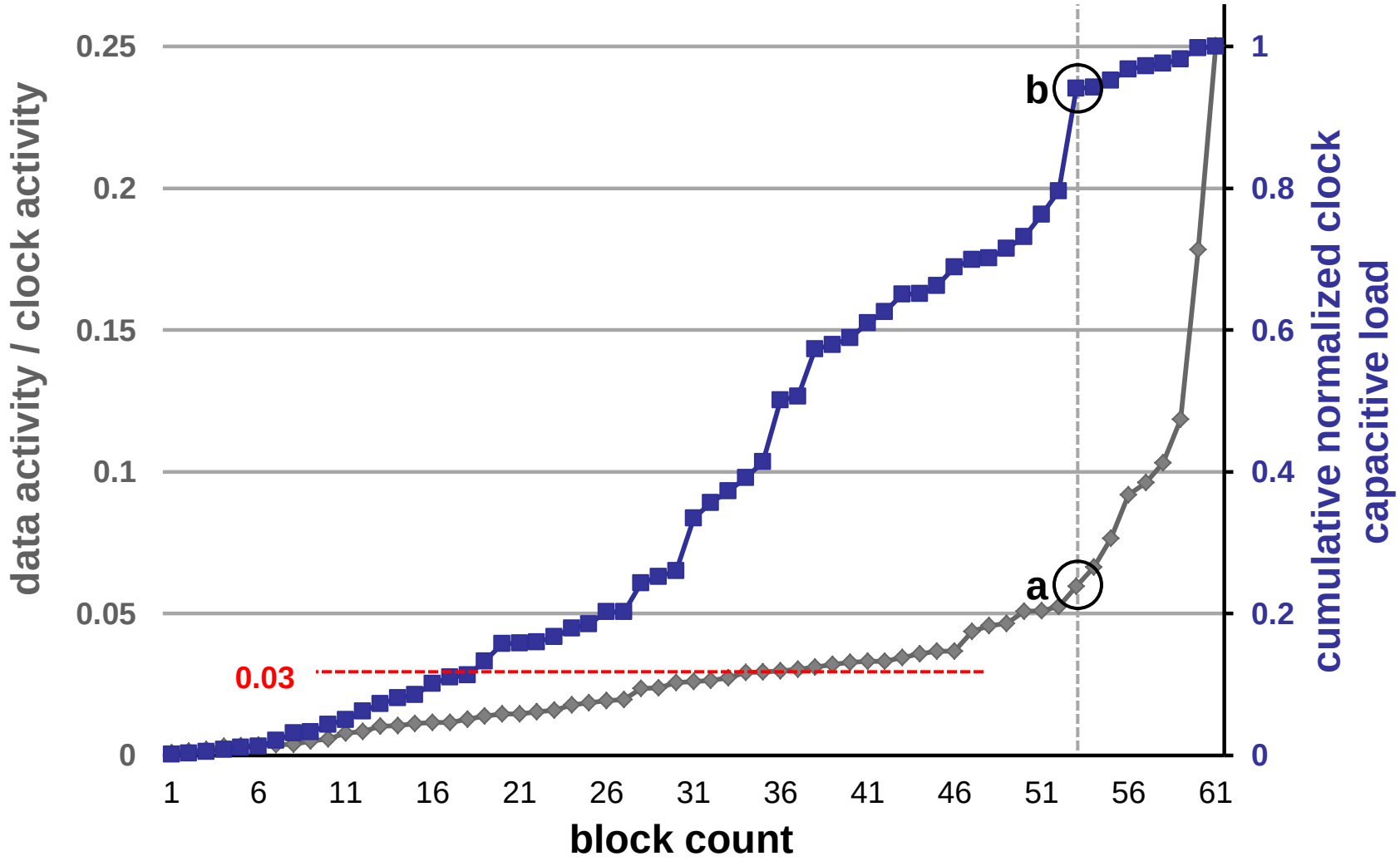
23k FFs, Test bench of 250k CLK cycles

10% of application run-time CLK is enabled.

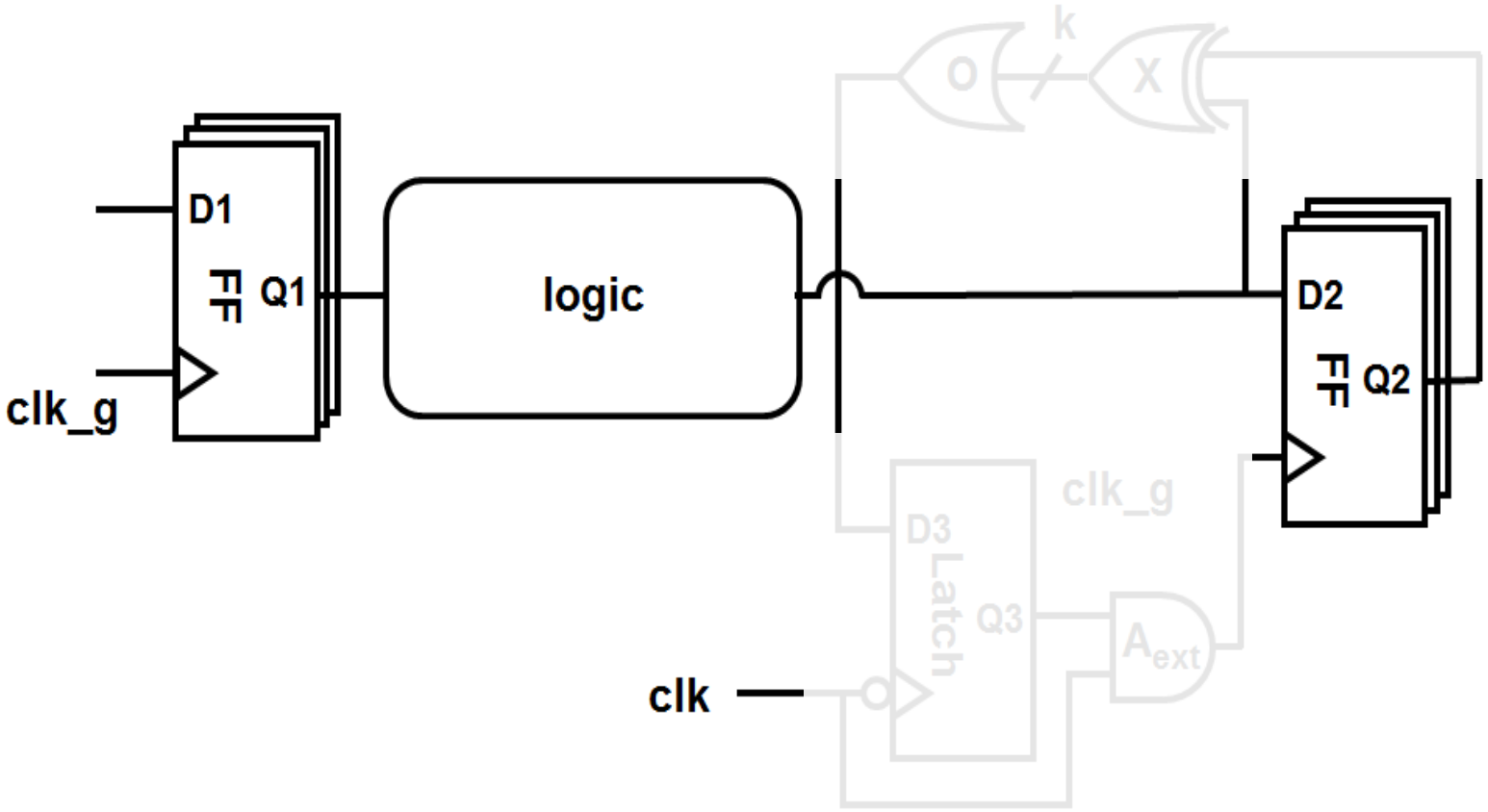
Of which, only 1.6% CLK pulses are useful!

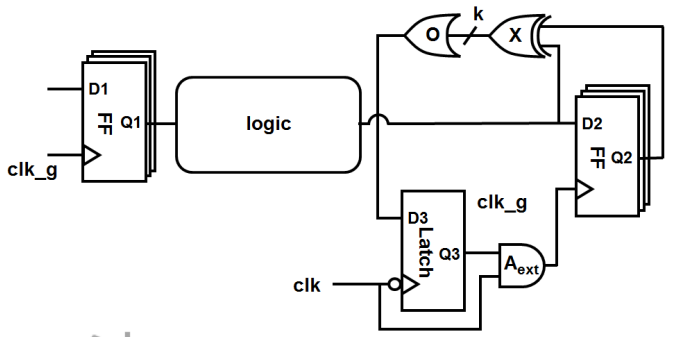
FF Data Toggling

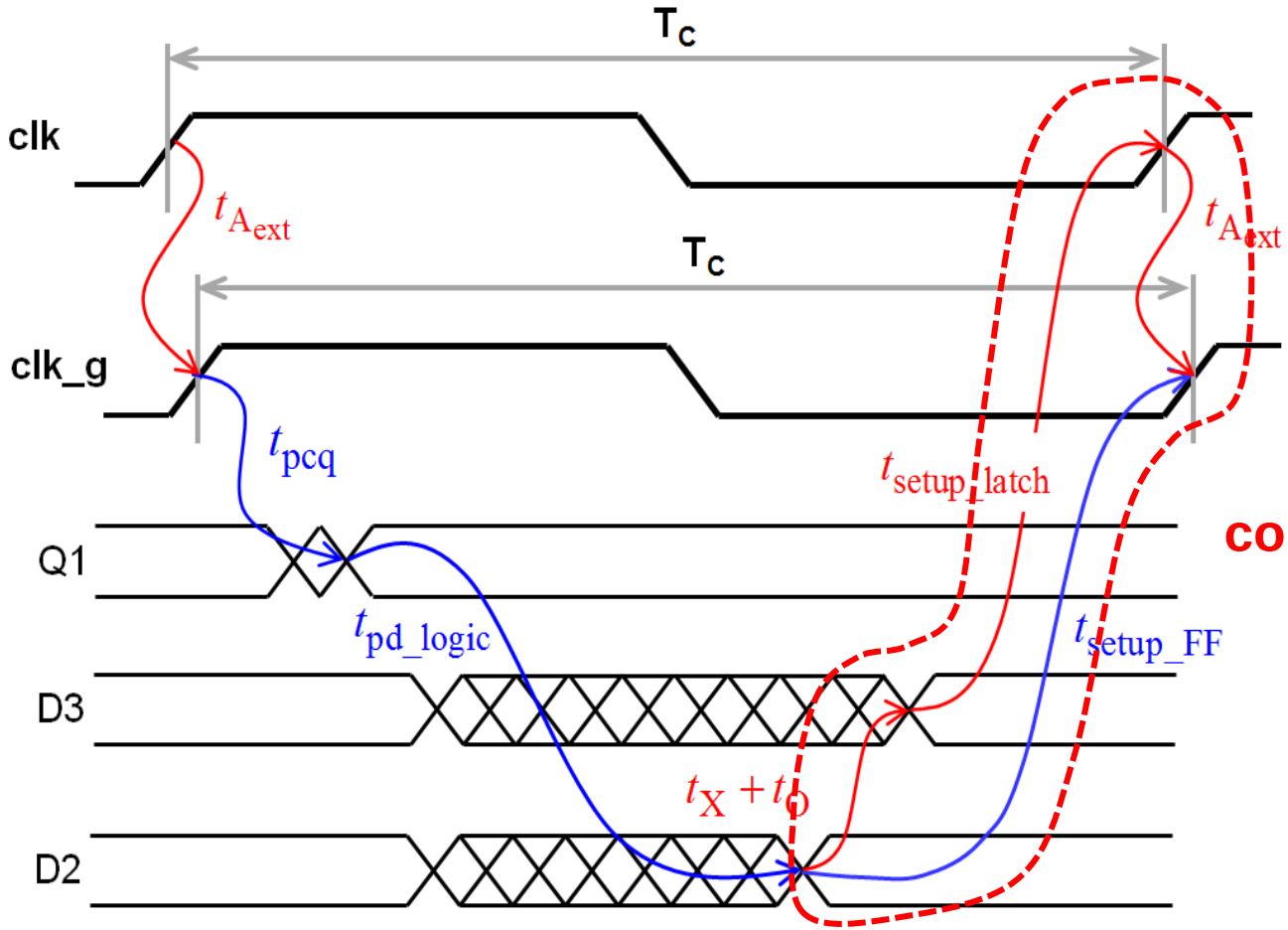
63 control blocks of μP, 200k FFs



Data-Driven CLK Gating





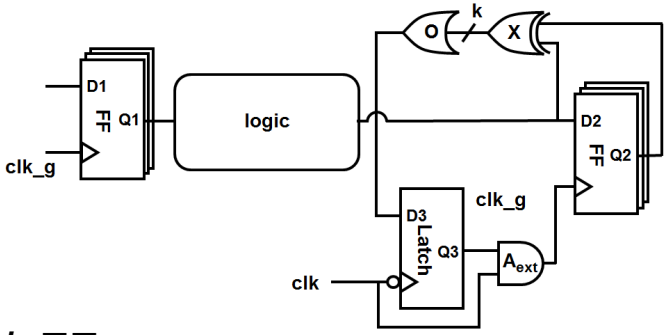


tight constraint

How Many FFs To Group ?

k: # FFs, q: FF probability of D=Q , $q = 1 - p$

Worst case: All FF are toggling independently of each other.



Net saving per FF

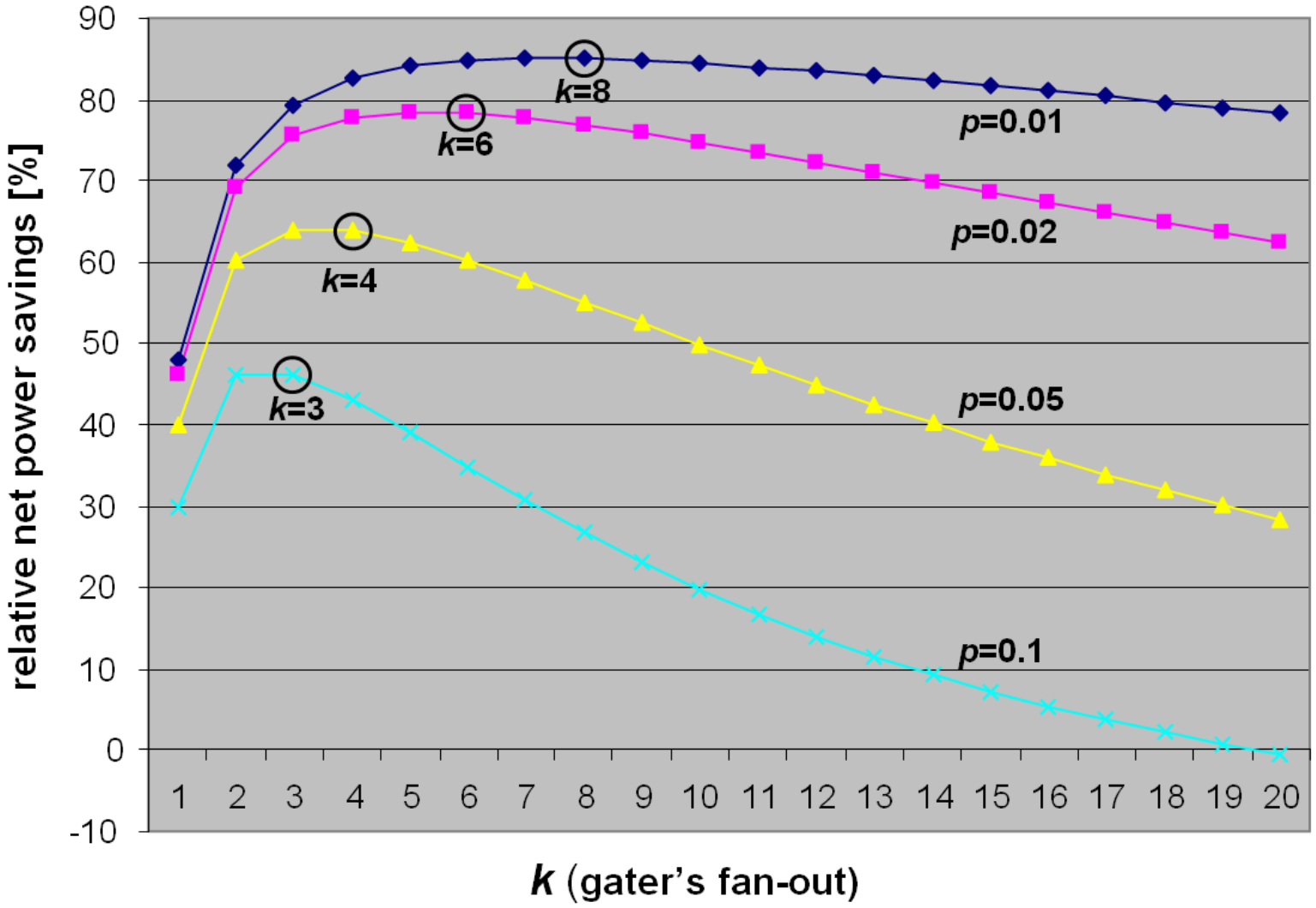
Latch overhead amortized over *k* FFs

$$\frac{c_{saving}}{FF} \geq q^k \left(c_{FF} + c_w\right) - \left[c_{latch}/k + (1-q)\left(c_w + c_{OR}\right)\right]$$

Gater's disabling probability

Probability of enabling FF

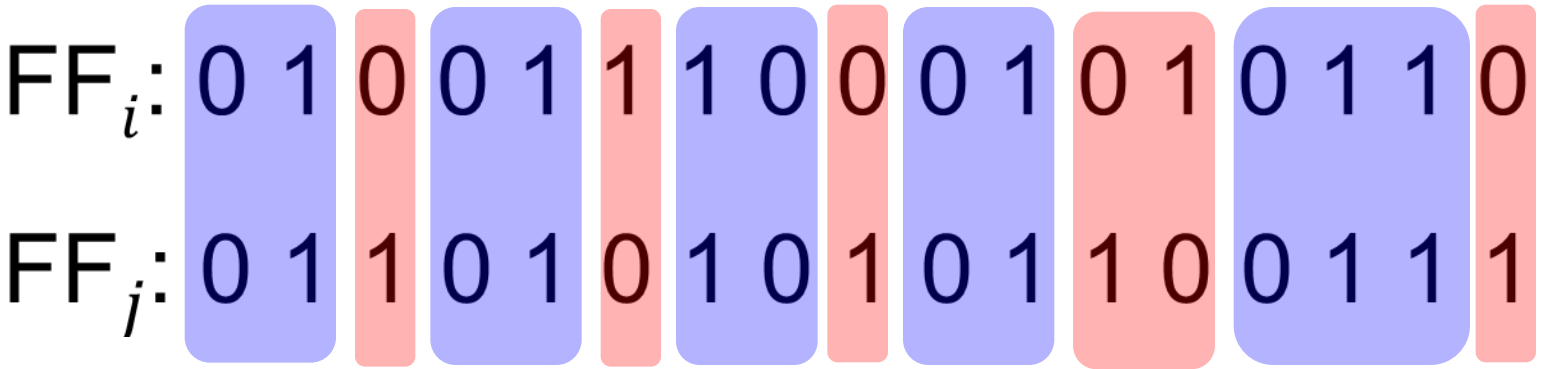
Derivate by *k*: $q^k \ln q \left(c_{FF} + c_w\right) + c_{latch}/k^2 = 0$


relative net power savings [%]
90
80
70
60
50
40
30
20
10
0
-10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
k (gater's fan-out)
k=8
k=6
k=4
k=3
p=0.01
p=0.02
p=0.05
p=0.1

Optimal Flip-flop k-size Grouping

Given n flip-flops and $m+1$ clock cycles

$\mathbf{a} = (a_1, \ldots, a_m)$ is the activity (toggling) of flip-flop



$\|\mathbf{a}_i \oplus \mathbf{a}_j\|$ is the number of redundant clock pulses ocurring by jointly clocking FF_i and FF_j

FF Pairwise Activity Model

$G(V, E, w)$: FF pairwise activity graph.

$v_i \in V$ corresponds to FF_i.

$e_{ij} = (v_i, v_j) \in E$ is FF pairing.

$\mathbf{a}_i \,|\, \mathbf{a}_j$ is joint toggling.

$w(e_{ij}) = \left\| \mathbf{a}_i \oplus \mathbf{a}_j \right\|$ is redundant clock pulses, hence a waste.

$E' \subset E$: vertex matching

Total power:

$$P = 2\sum_{e_{ij} \in E'} \left\| \mathbf{a}_i \mid \mathbf{a}_j \right\| =$$

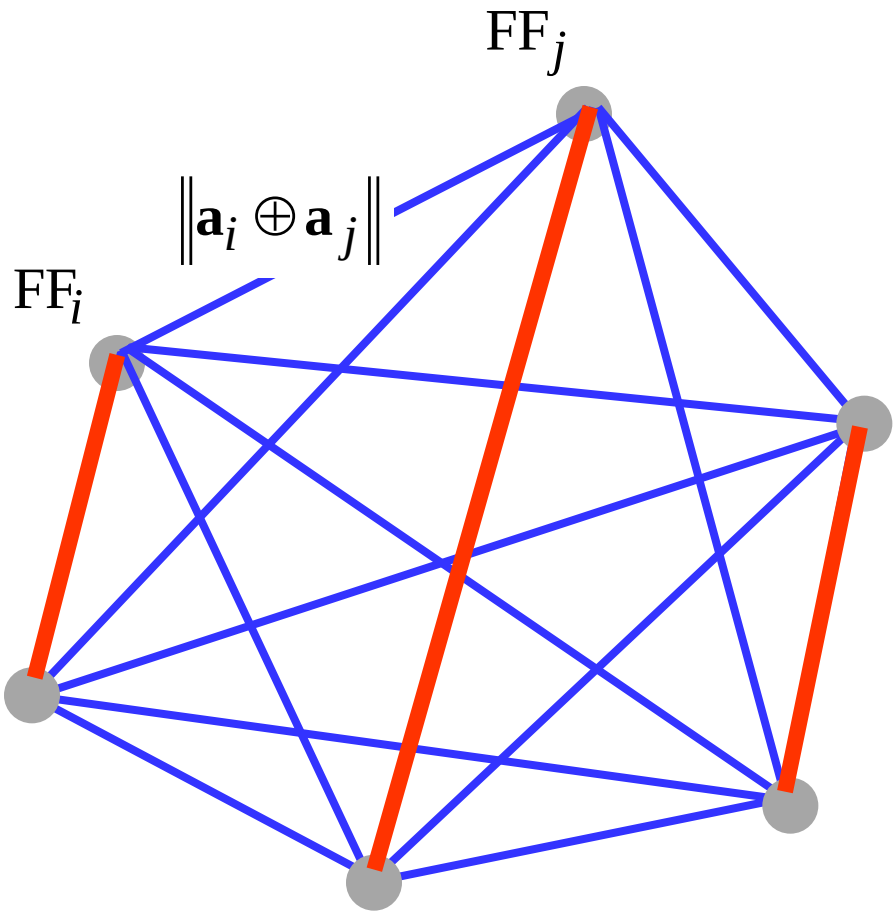
$$\sum_{v_i \in V} \left\| \mathbf{a}_i \right\| + \sum_{e_{ij} \in E'} \sum_{e_{ij} \in E'} \left[\left\| \mathbf{a}_i \oplus \left(\mathbf{a}_i \mid \mathbf{a}_j \right) \right\| + \left\| \mathbf{a}_j \oplus \left(\mathbf{a}_i \mid \mathbf{a}_j \right) \right\| \right] =$$

Essential + Waste

$$\sum_{v_i \in V} \left\| \mathbf{a}_i \right\| + \sum_{e_{ij} \in E'} \left\| \mathbf{a}_i \oplus \mathbf{a}_j \right\| = \sum_{v_i \in V} \left\| \mathbf{a}_i \right\| + \sum_{e_{ij} \in E'} w\left(e_{ij} \right)$$

Flip-Flop Grouping Algorithm

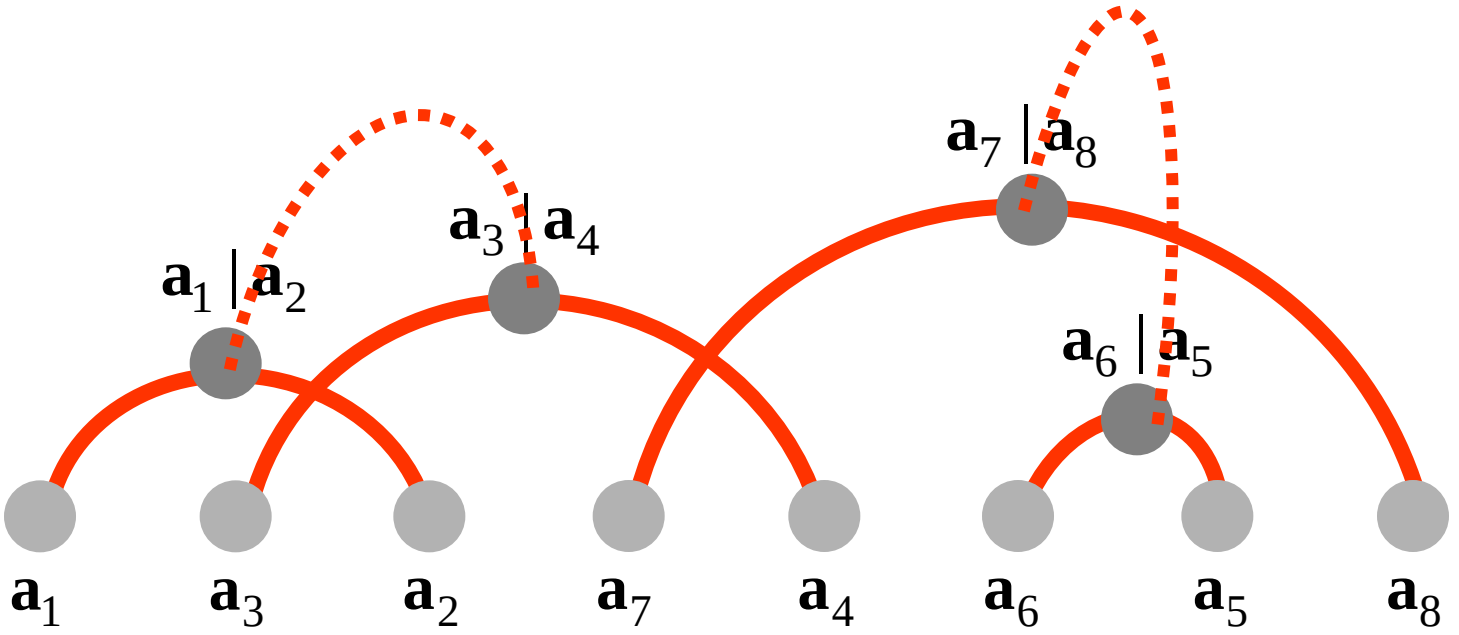
Minimize: $\sum \left\| \mathbf{a}_i \oplus \mathbf{a}_j \right\|$



Optimal FFs pairing can be solved by minimal cost perfect graph matching.

Can be repeated for groups of size 4, 8, 16 …

Is repeated perfect matching optimal ?

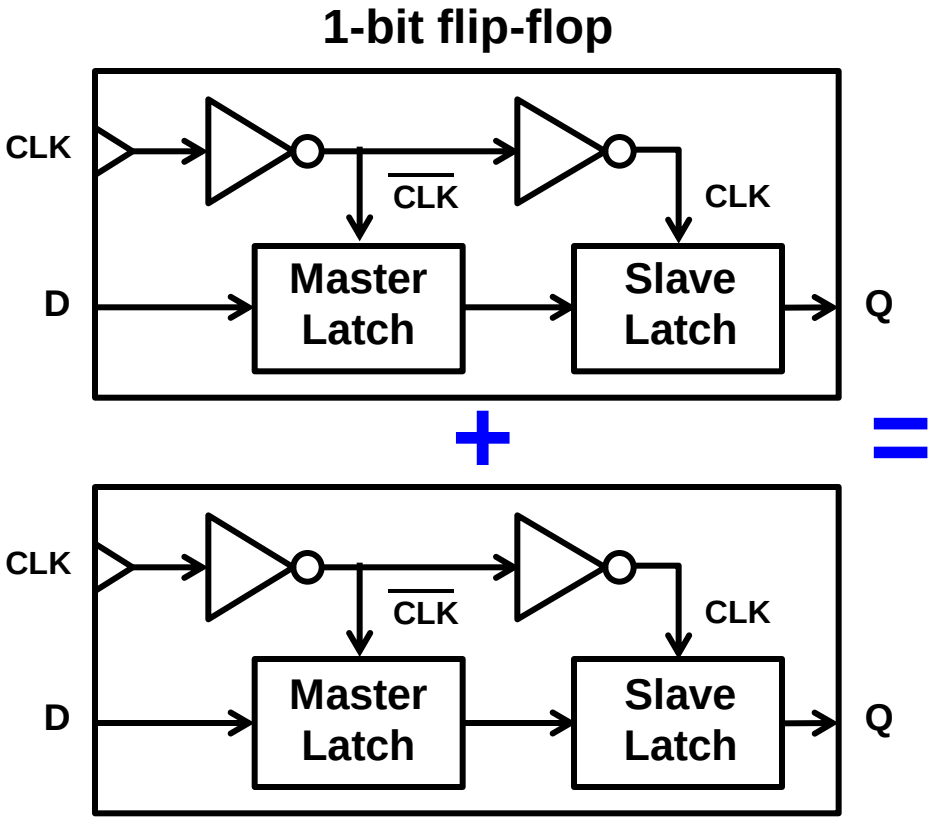


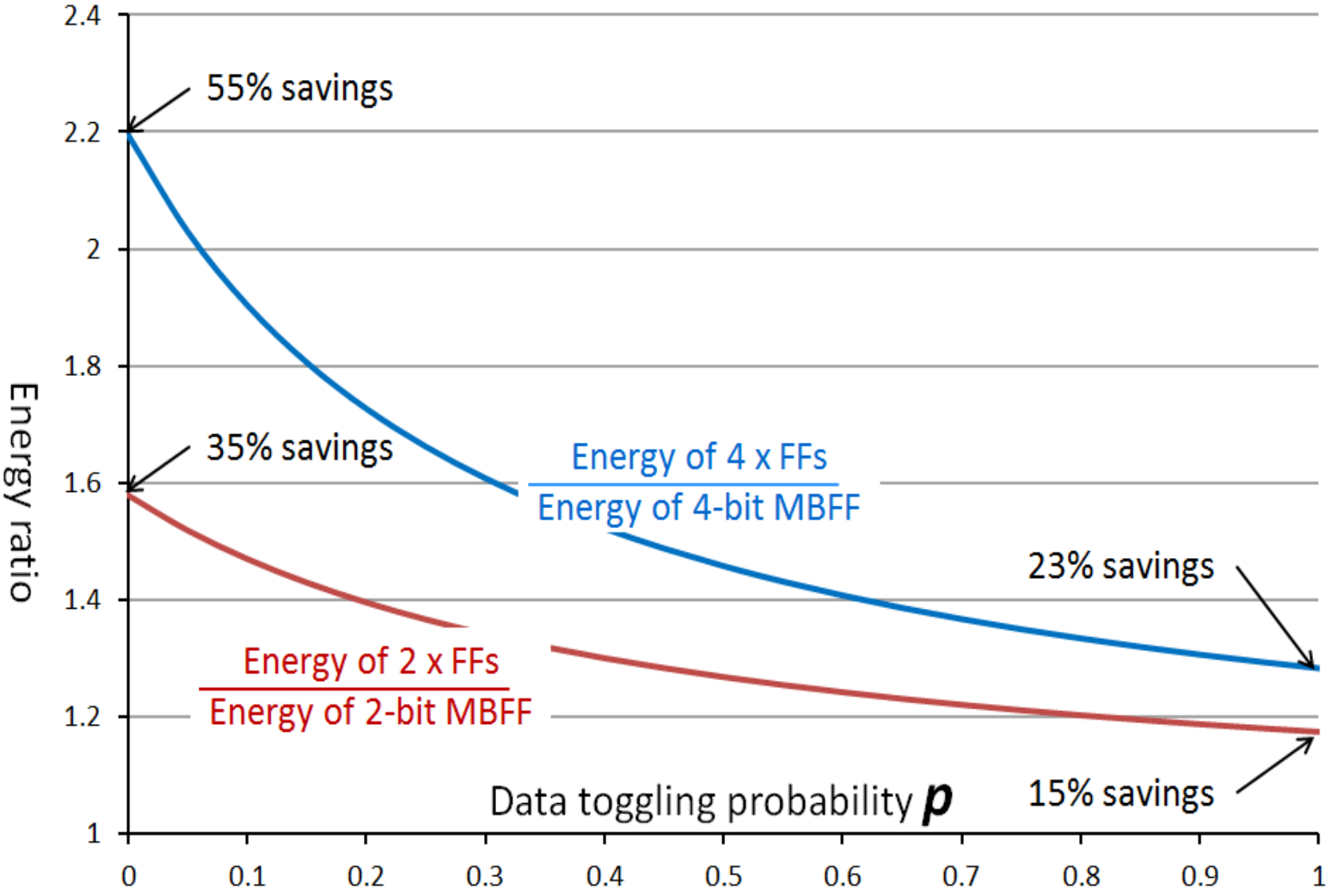
No! Here is the optimal 4-size grouping

FF1:	0	0	1	0	0	0	1	0	0	0	0	1
FF2:	0	1	0	0	0	1	1	0	1	1	0	1
FF6:	1	0	1	1	1	0	1	0	1	0	0	1
FF7:	0	1	1	1	0	1	0	0	1	0	0	1
FF3:	1	1	1	0	0	0	0	1	0	1	1	0
FF4:	1	0	1	0	0	0	0	1	1	1	1	0
FF5:	1	0	0	1	1	0	1	0	1	1	0	0
FF8:	0	1	1	1	1	0	0	1	0	0	0	0

Multi-Bit Flip-Flop

Saving the power of internal CLK drivers




2.4
2.2
2
1.8
1.6
1.4
1.2
1
Energy ratio
55% savings
35% savings
Energy of 4 x FFs
Energy of 4-bit MBFF
23% savings
Energy of 2 x FFs
Energy of 2-bit MBFF
15% savings
Data toggling probability p
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

MBFF should be combined with data-driven CG to maximize energy savings.

Toggling vectors (VCD) are unfortunately not always available.

Data-to-clock toggling ratio (probability) is more often available.

How to utilize it for MBFF optimal grouping?

What is the energy waste in 2-bit MBFF?

$$p_j(1-p_i)+p_i(1-p_j)=p_i+p_j-2p_ip_j$$

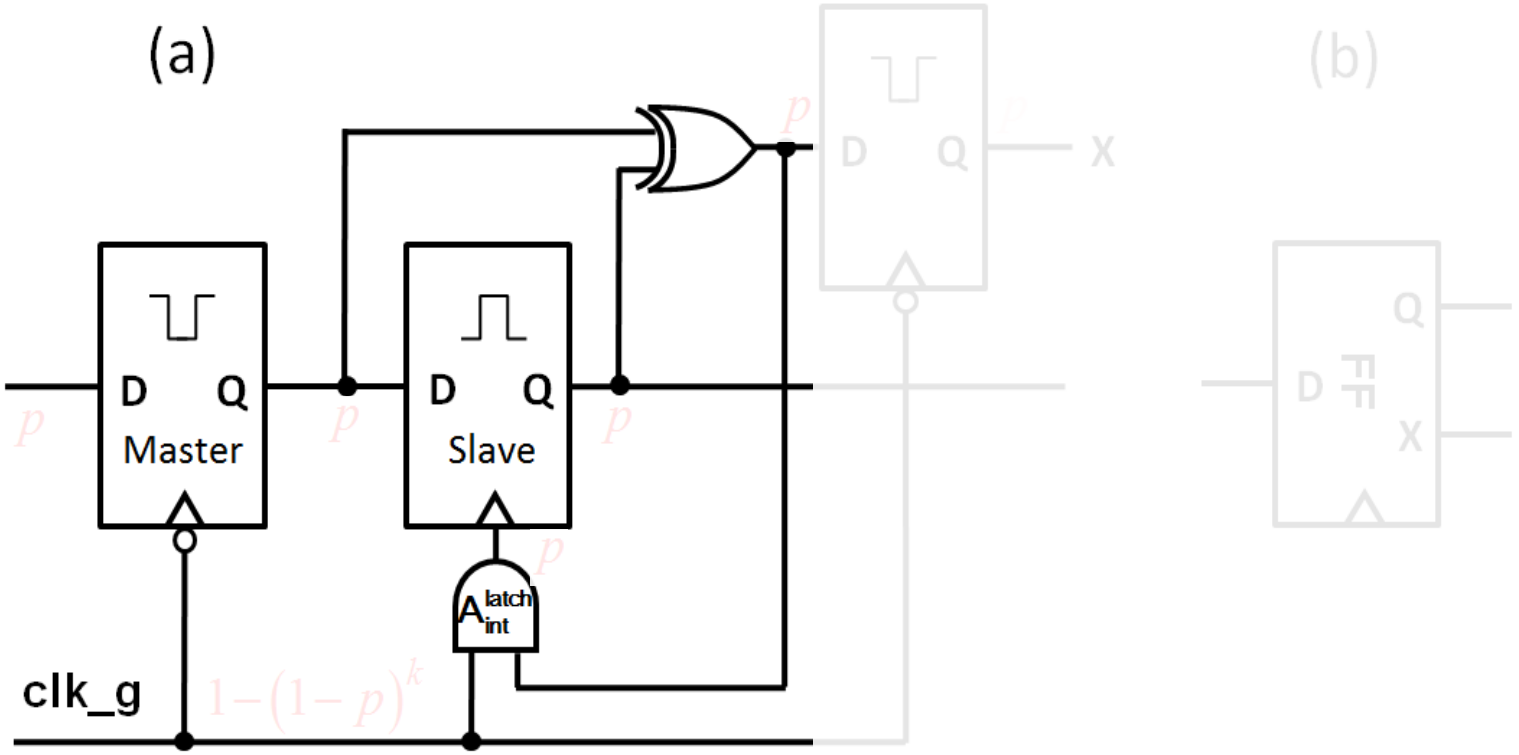
For n FFs grouped in n/2 MBFFs it is

$$\sum_{j=1}^{n} p_j - 2\sum_{i=1}^{n/2} p_{s_i}p_{t_i}$$

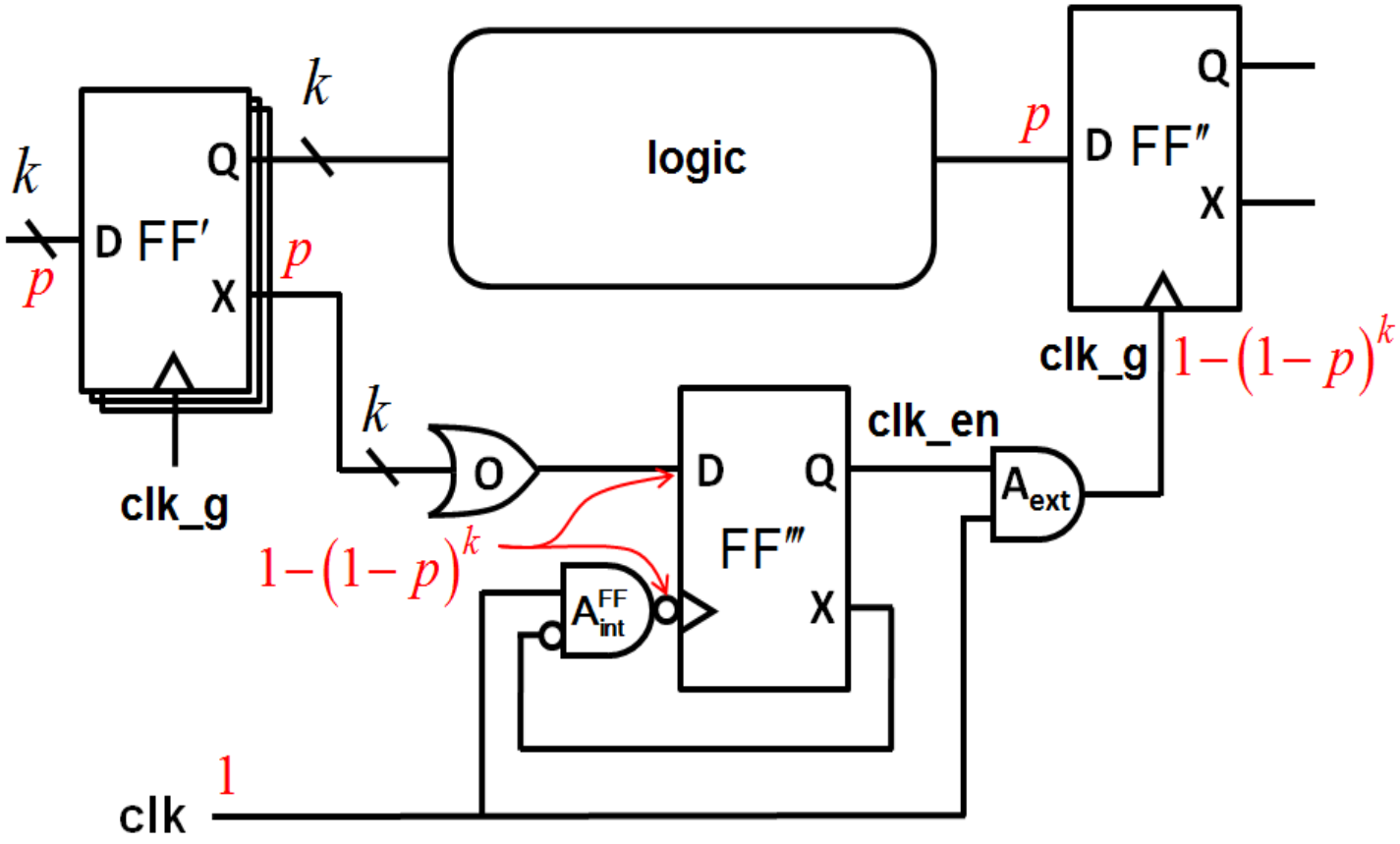
What is the optimal MBFF grouping?

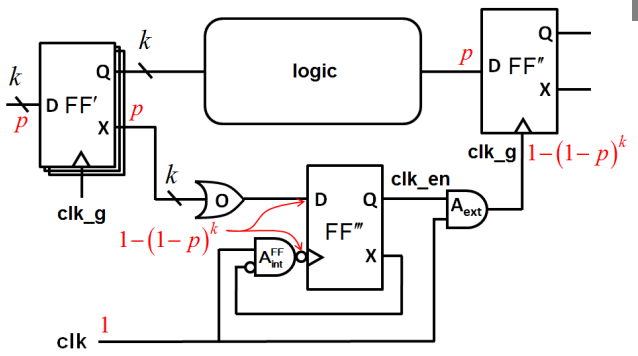
Group the FFs such that $p_1 \le p_2 \le \cdots \le p_n$

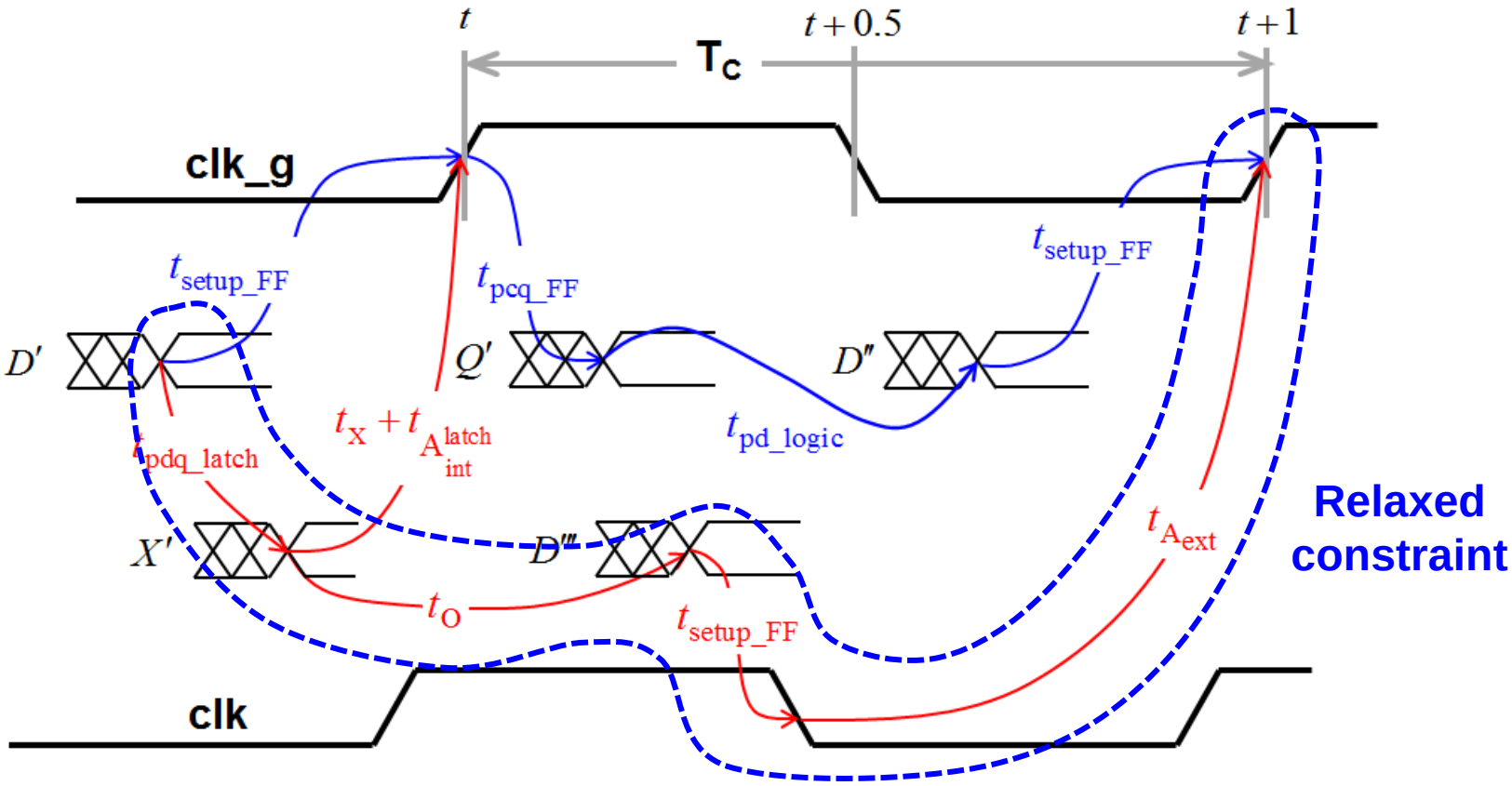
Auto-Gated FF



Look-Ahead CLK Gating

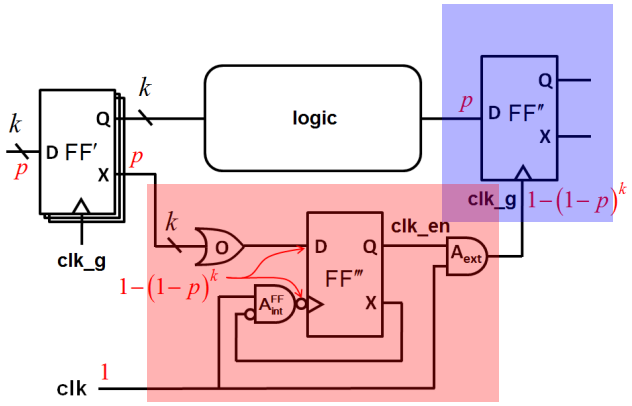






Relaxed constraint

Power Saving Per FF



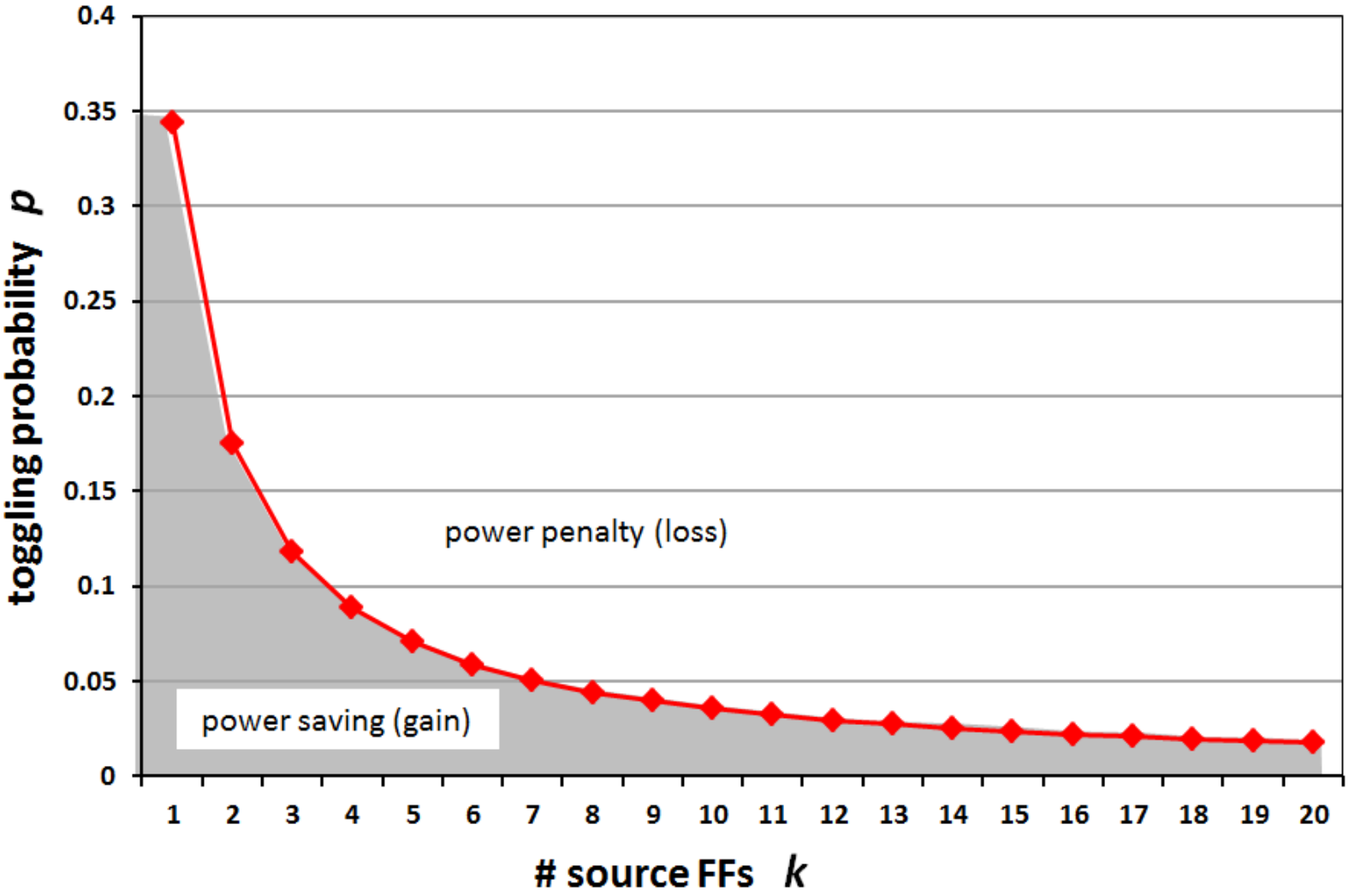
$$c_{\text{dyn}}^{\text{save}} =$$

$$(1-p)^k \left(c_{\text{FF+CLK}} + c_{\text{FF}} + c_{\text{O}}\right) -$$

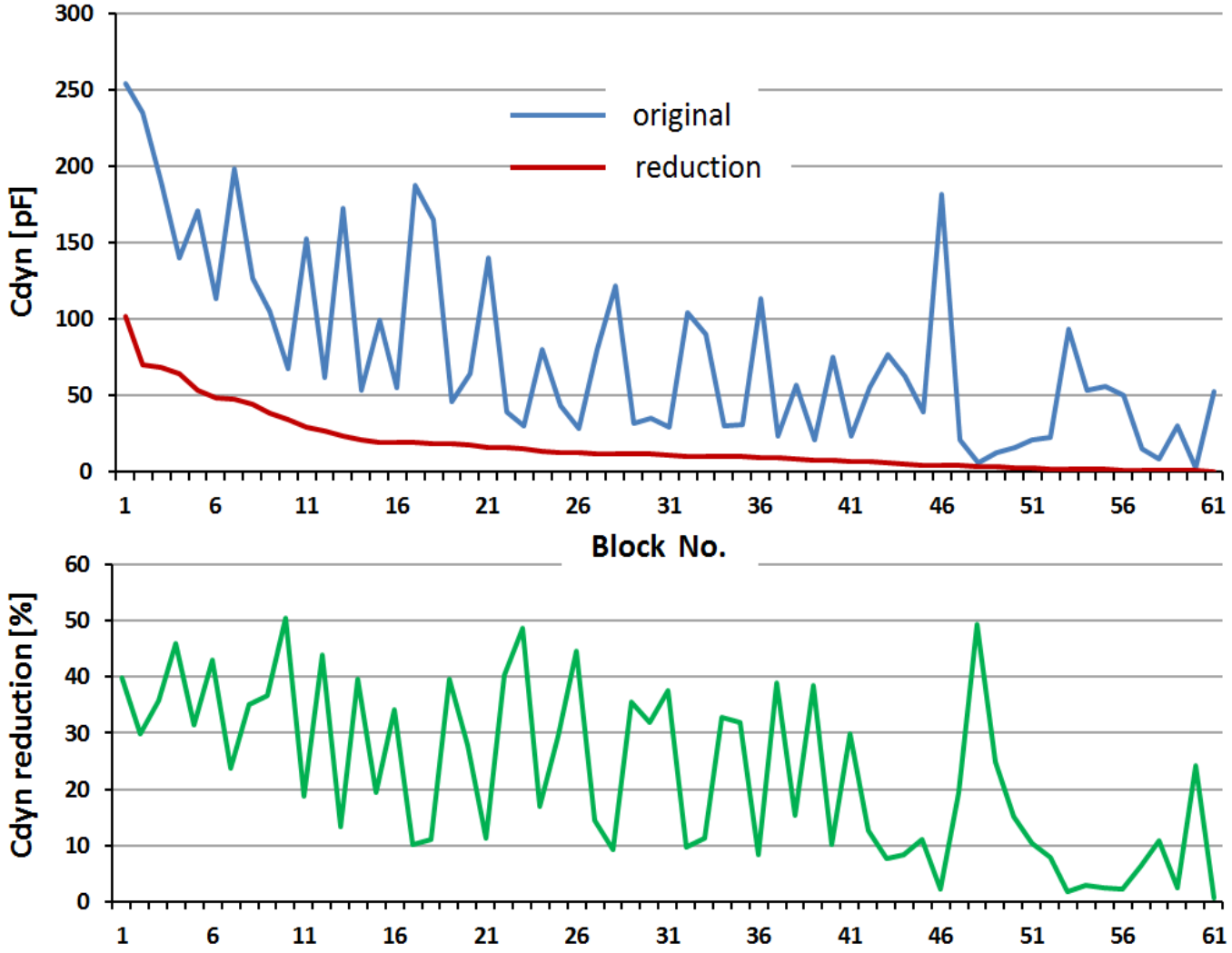
$$p\left(c_{\text{X}} + kc_{\text{O}}\right) - \left(\frac{c_{\text{FF+CLK}}}{3} + c_{\text{A}_{\text{int}}} + c_{\text{FF}} + c_{\text{O}}\right)$$

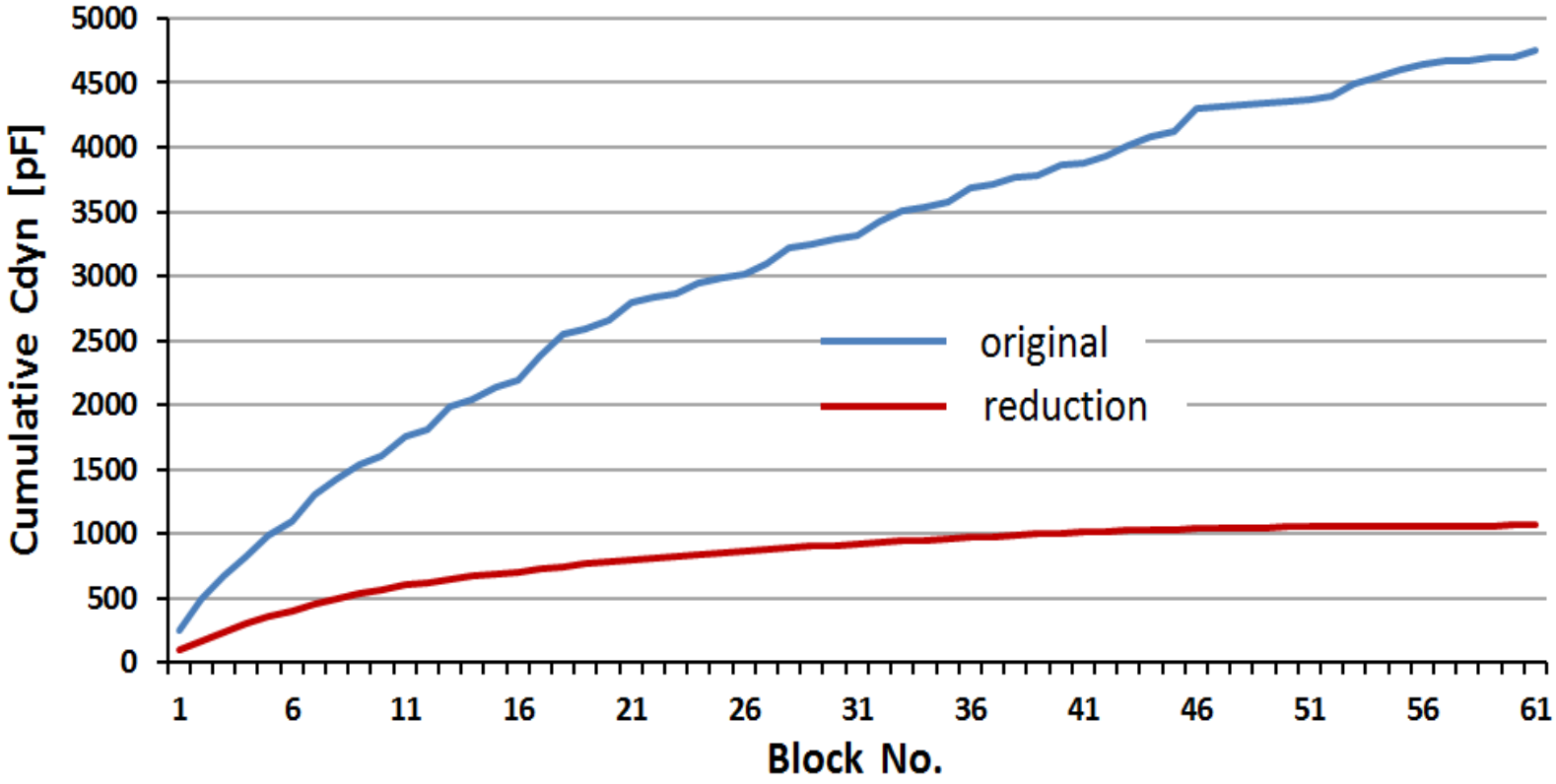
C_{FF}	C_{CLK}	C_{FF+CLK}	C_X	C_O	$C_{A_{int}}$
25.7	33.5	36.9	2.9	3.1	1.7

Which FF to Gate?



Results


300
250
200
150
100
50
0
Cdyn [pF]
original
reduction
1
6
11
16
21
26
31
36
41
46
51
56
61
Block No.
60
50
40
30
20
10
0
Cdyn reduction [%]


Cumulative Cdyn [pF]
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
6
11
16
21
26
31
36
41
46
51
56
61
Block No.
original
reduction