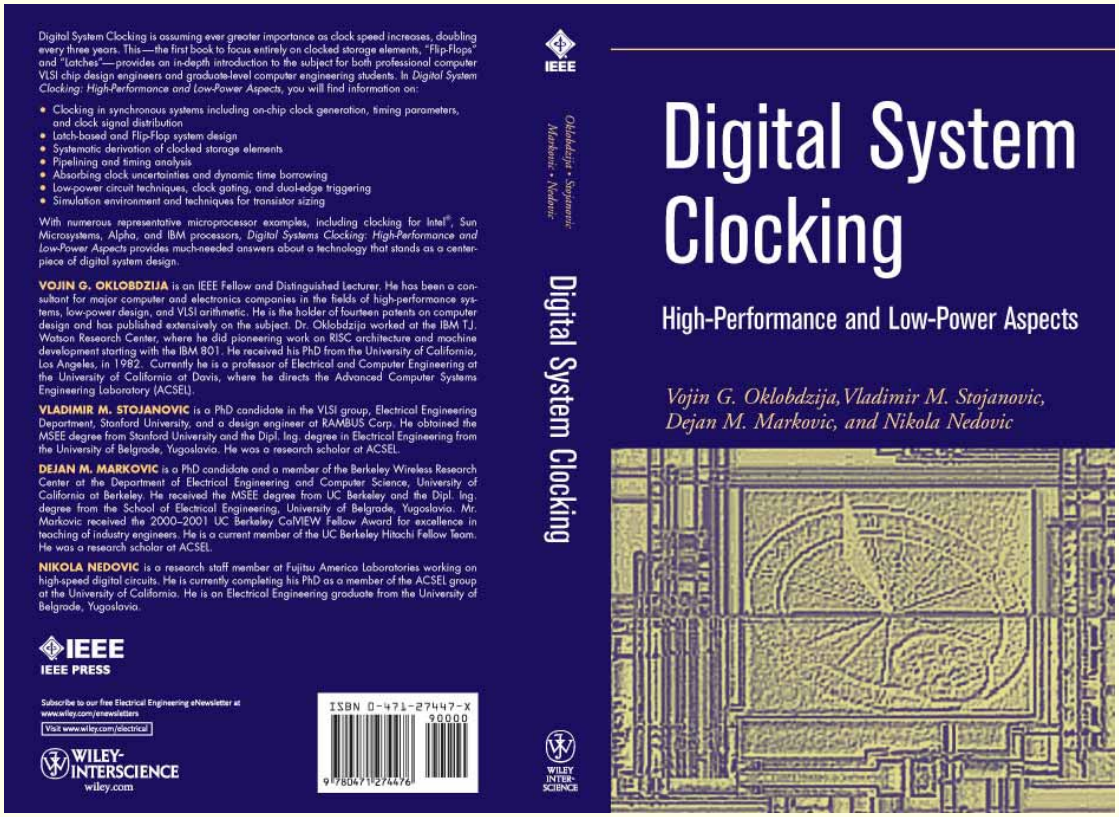
Digital System Clocking:

High-Performance and Low-Power Aspects

Vojin G. Oklobdzija, Vladimir M. Stojanovic, Dejan M. Markovic, Nikola M. Nedovic

Chapter 7: Simulation Techniques



Wiley-Interscience and IEEE Press, January 2003

Simulation Techniques

- Results and conclusions about the performance of different CSEs depend significantly on the simulation setup and evaluation environment.
- CSE is just one of the elements in the pipeline, and has to be sized in such a way as to achieve optimum performance for given output load.
- The CSE output loads vary a lot across the processor core, depending on the level of parallelism in each unit and also whether the CSE is on the critical path or not.
- In modern data-paths CSEs experience heavy load due to the parallel execution units and increase in interconnect capacitance.
- It is the performance of these CSEs, on the critical path, that has the highest impact on the choice of the processor cycle time.

Simulation Techniques

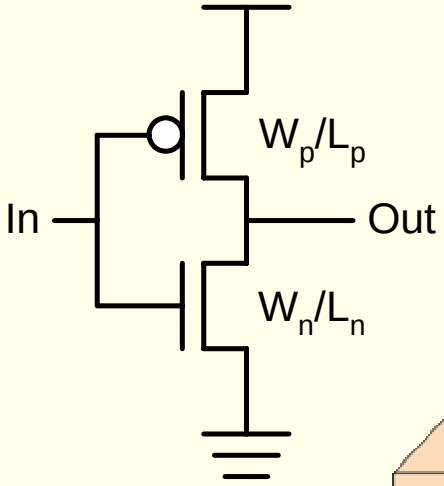
- In high-speed designs, the design and evaluation of CSEs is focused on the elements on the critical path and often implicitly assumes such conditions during performance comparisons.
- There are a lot of CSEs that are placed on non-critical paths with relatively light load. While these CSEs do not directly impact the performance of the processor, careful design of these elements can significantly reduce the energy consumption and alleviate the clock distribution problems.
- The purpose of this presentation is to recommend simulation techniques that designers can use to evaluate the performance of CSEs, depending on the desired application.
- We try to build the understanding of the issues involved in creating a simulation environment of the CSE, so that such information can be used to build own setups tailored to the specific application.
- There is no universal setup that is good for all CSE applications.

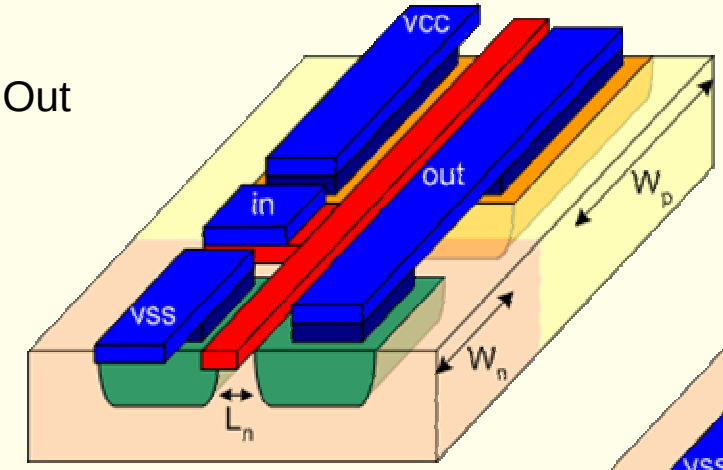
The Method of Logical Effort

Logical Effort Basics

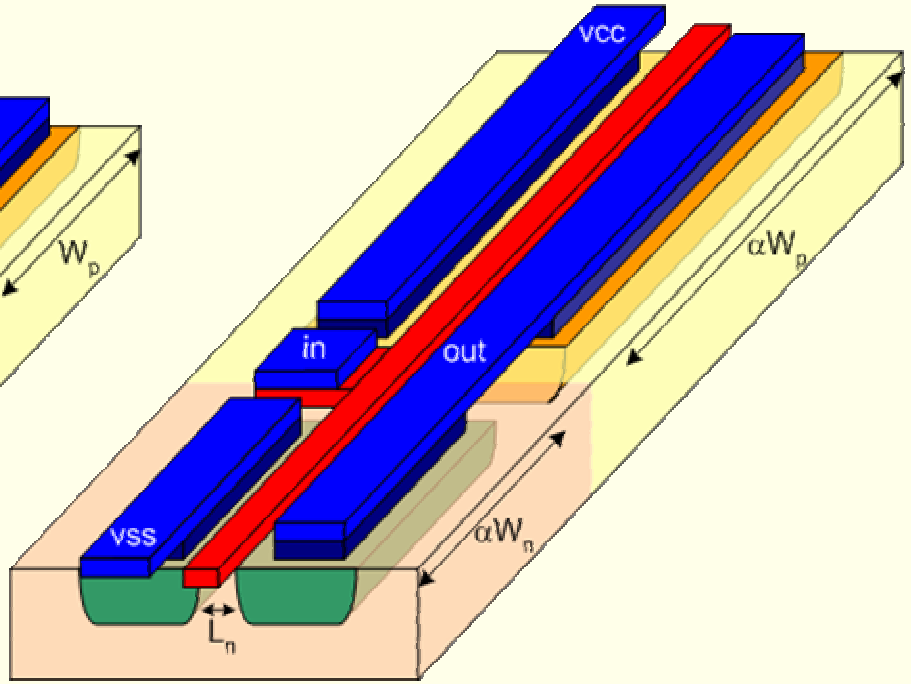
Contribution of Bart Zeydel, PhD candidate
ACSEL group
University of California
Davis

Logical Effort Components





Template



Arbitrary Size

- Input Capacitance increases by $\alpha \cdot C_{template}$
- Resistance decreases by $R_{template} / \alpha$

Logical Effort Input Capacitance

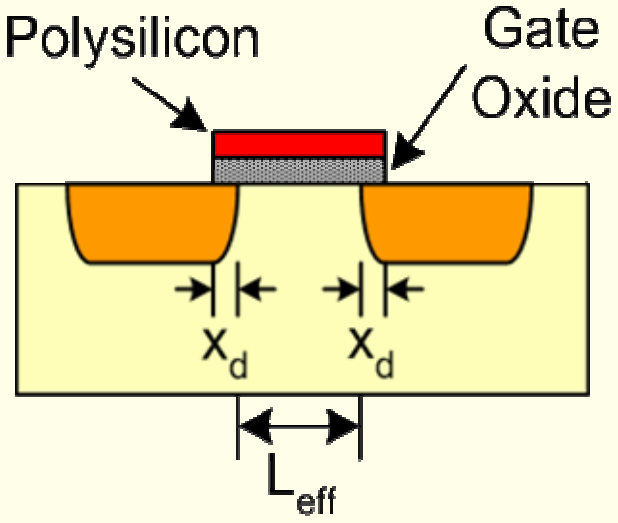
$$C_{ox} = \varepsilon_{ox} / t_{ox} \qquad L_{eff} = L - 2x_d$$

$$C_{gate} = C_{ox} W L_{eff} = C_{ox} W L (1 - 2x_d / L)$$

$$\kappa_1 = C_{ox}(1 - 2x_d / L)$$

$$C_{template(inv)} = \kappa_1 W_n L_n + \kappa_1 W_p L_p$$

$$C_{in(inv)} = \alpha \cdot C_{template(inv)}$$



- Input Capacitance is the sum of the gate capacitances
- Using LE, α denotes size in multiples of the template

Logical Effort Resistance

$R = (\rho / t)\,(L / W)$ ohms

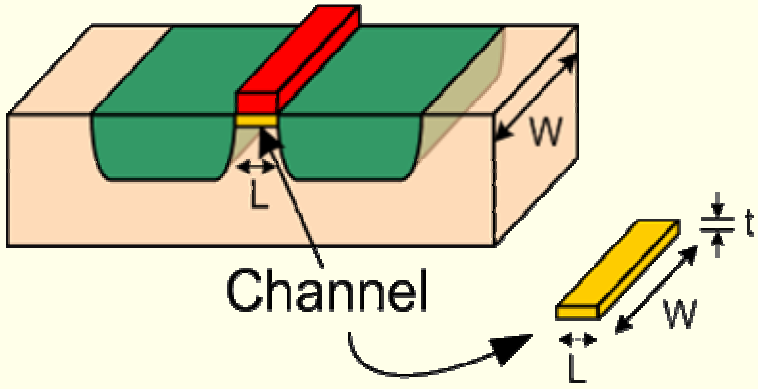
$R = R_{sheet}\,(L / W)$ ohms

$R_{channel} = R_{sheet}\,(L / W)$ ohms

$R_{sheet} = 1 / (\mu C_{ox}\,(V_{gs} - V_t))$ ohms

$\kappa_2 = C_{ox}\,(V_{gs} - V_t)$

$R_{channel} = L / (\kappa_2\,\mu\,W)$ ohms



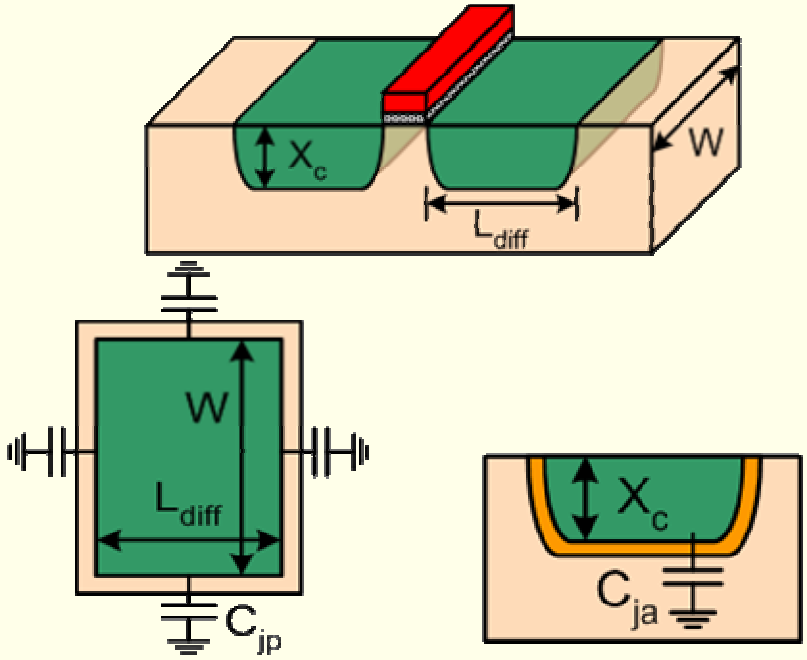
μ = surface mobility

$R_{up(inv)} = R_{up\text{-}template(inv)} / \alpha$ $R_{down(inv)} = R_{down\text{-}template(inv)} / \alpha$

- Resistance is dependent on transistor size and process
- Larger values of α result in lower resistance

Logical Effort Parasitics

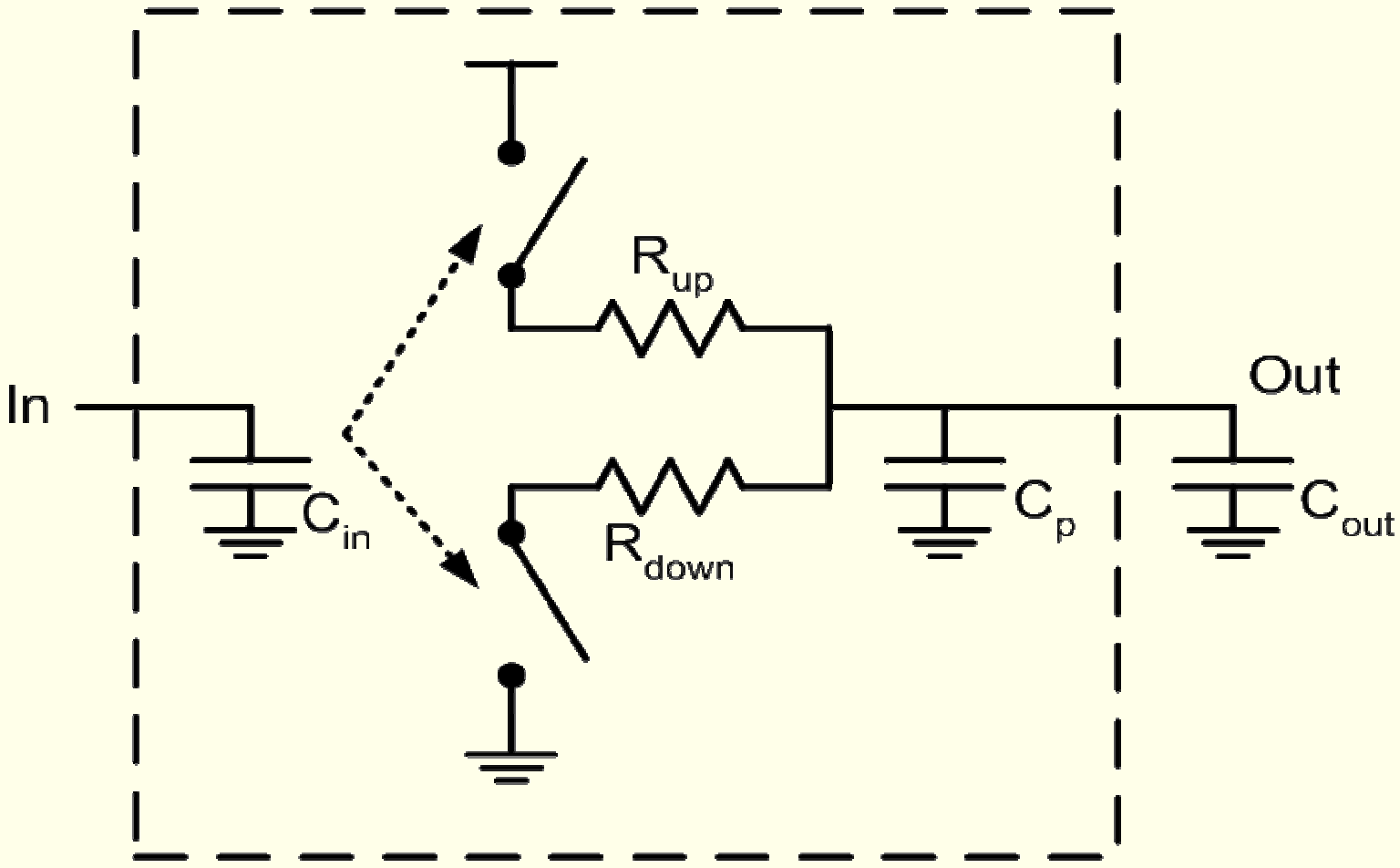
C_{ja}	junction capacitance per μ^2
C_{jp}	periphery capacitance per μ
W	width of diffusion region μ
L_{diff}	length of diffusion region μ



$$C_j = C_{ja} \cdot W \cdot L_{diff} + C_{jp} \cdot (2 \cdot W + 2 \cdot L_{diff})$$

- Larger values of α result in increased parasitic capacitance, however R decreases at same rate, thus constant RC delay

RC Model for CMOS logic gate



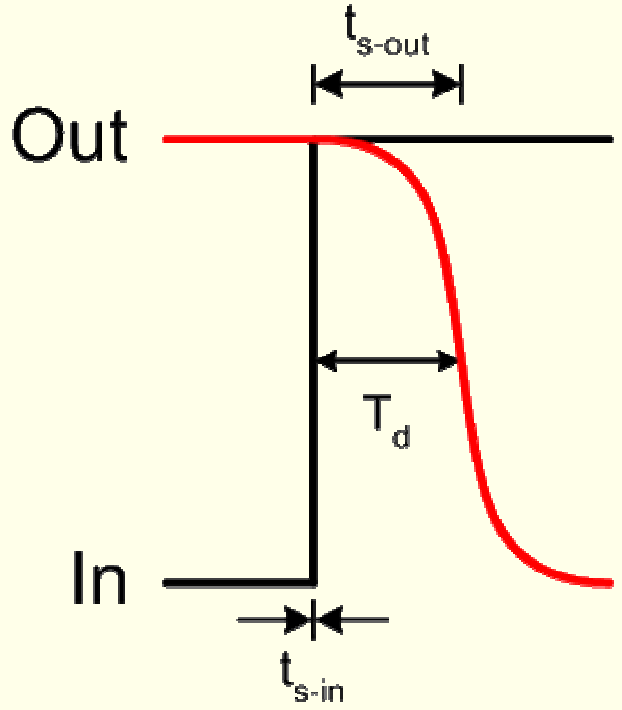
LE Delay derivation for step input

$$I_{sat} = \mu \cdot C_{ox} \cdot \frac{W}{L} \frac{(V_{GS} - V_t)^2}{2}$$

$$I_{linear} = \mu \cdot C_{ox} \cdot \frac{W}{L} \left((V_{GS} - V_t) \cdot V_{out} - \frac{V_{out}^{\ 2}}{2} \right)$$

$$t_{s-in} = 0$$

$$t_D = t_{s-out(sat)} + t_{s-out(linear)}$$



$$t_{f-out} = \int_{Vdd-Vt}^{Vdd} \frac{C_{out}}{I_{sat}} \cdot dV + \int_{Vdd/2}^{Vdd-Vt} \frac{C_{out}}{I_{linear}} \cdot dV$$

LE Model derivation (cont.)

$$t_{f-out(sat)} = \frac{C_{out}}{\mu \cdot C_{ox} \cdot \frac{W}{L} \frac{(V_{DD}-V_t)^2}{2}} \int_{Vdd-Vt}^{Vdd} dV_{out} = \frac{2C_{out}V_t}{\mu \cdot C_{ox} \cdot \frac{W}{L}(V_{DD}-V_t)^2}$$

$$t_{f-out(linear)} = \frac{C_{out}}{\mu \cdot C_{ox} \cdot \frac{W}{L}} \int_{Vdd-Vt}^{Vdd} \frac{1}{\left((V_{DD}-V_t)\cdot V_{OUT} - \frac{V_{OUT}^2}{2}\right)} dV_{out}$$

$$t_{f-out} = \frac{C_{out}}{\mu \cdot C_{ox} \cdot \frac{W}{L}(V_{DD}-V_t)} \left(\frac{2 \cdot V_t}{V_{DD}-V_t} + \ln \cdot \frac{4 \cdot (V_{DD}-V_t) - V_{DD}}{V_{DD}} \right)$$

Substituting in RC to obtain

$$t_D = R \cdot C_{out} \cdot \underbrace{\left(\frac{2 \cdot V_t}{V_{DD}-V_t} + \ln\left(3 - 4\frac{V_t}{V_{DD}}\right) \right)}_{\kappa}$$

Logical Effort Gate Delay Model

$$d = \kappa \cdot R_{in}(C_{out} + C_{parasitic})$$

$$d = \kappa \cdot \left(\frac{R_{template}}{\alpha}\right) \cdot C_{in} \cdot \left(\frac{C_{out}}{C_{in}}\right) + \kappa \cdot \left(\frac{R_{template}}{\alpha}\right) \cdot (\alpha \cdot C_{parasitic})$$

$$d = (\kappa \cdot R_{template} \cdot C_{template})\left(\frac{C_{out}}{C_{in}}\right) + \kappa \cdot R_{template} \cdot C_{parasitic}$$

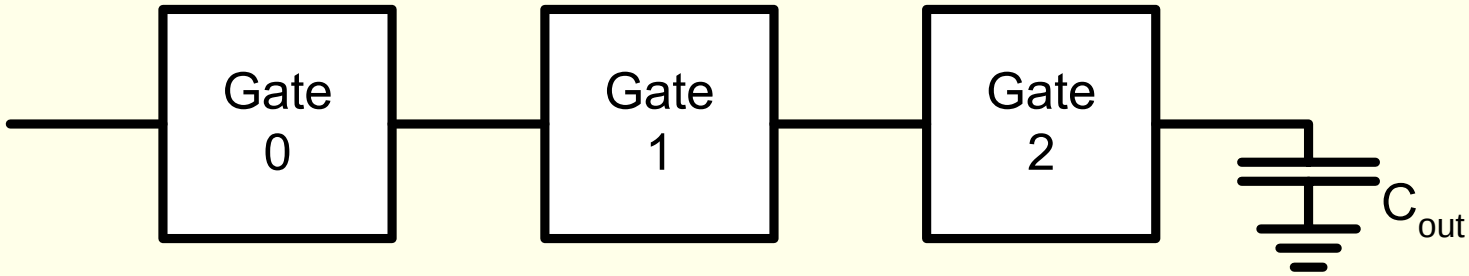
$$\tau = \kappa \cdot R_{inv} \cdot C_{inv}$$

$$g = \frac{R_{template} C_{template}}{R_{inv} C_{inv}} \qquad h = \frac{C_{out}}{C_{in}} \qquad p = \frac{R_{template} C_{parasitic}}{R_{inv} C_{inv}}$$

$$\boxed{d = \tau \cdot (g \cdot h + p)}$$

Harris, Sutherland, Sproull

LE Path Delay



Input Capacitance:	C_{in}	C_1	C_2
Logical Effort:	g_0	g_1	g_2
Parasitic Delay:	p_0	p_1	p_2
Stage Effort:	f_0	f_1	f_2

$$D = [(g_0 h_0 + p_0) + (g_1 h_1 + p_1) + (g_2 h_2 + p_2)] \cdot \tau$$

- Any n-stage path can be described using Logical Effort

LE Path Delay Optimization

$$D = [(g_0 h_0 + p_0) + (g_1 h_1 + p_1) + (g_2 h_2 + p_2)] \cdot \tau$$

$$h_0 = \frac{C_1}{C_{in}} \qquad h_1 = \frac{C_2}{C_1} \qquad h_2 = \frac{C_{out}}{C_2}$$

$$h_0 h_1 h_2 = \frac{C_1}{C_{in}} \cdot \frac{C_2}{C_1} \cdot \frac{C_{out}}{C_2} = \frac{C_{out}}{C_{in}} = H$$

$$D = [(g_0 \frac{C_1}{C_{in}} + p_0) + (g_1 \frac{C_2}{C_1} + p_1) + (g_2 \frac{C_{out}}{C_2} + p_2)] \cdot \tau$$

LE Path Delay Optimization (cont.)

By Definition, C_{in} and C_{out} are fixed.

$$D = [(g_0 \frac{C_1}{C_{in}} + p_0) + (g_1 \frac{C_2}{C_1} + p_1) + (g_2 \frac{C_{out}}{C_2} + p_2)] \cdot \tau$$

Solve for C_1 and C_2:

$$\frac{\partial D}{\partial C_1} = \frac{g_0}{C_{in}} - g_1 \frac{C_2}{C_1^2} = 0 \qquad \frac{\partial D}{\partial C_2} = \frac{g_1}{C_1} - g_2 \frac{C_{out}}{C_2^2} = 0$$

$$g_0 \frac{C_1}{C_{in}} = g_1 \frac{C_2}{C_1} \qquad g_1 \frac{C_2}{C_1} = g_2 \frac{C_{out}}{C_2}$$

$$f_0 = f_1 \qquad f_1 = f_2$$

Minimum occurs when stage efforts are equal

$$\boxed{f_0 = f_1 = f_2}$$

Simplified Path Optimization

We want the effort of each stages to be equal.

$$F = f_0 \cdot f_1 \cdot f_2 \ldots f_{n-1}$$

$$\hat{f} = F^{1/n}$$

To quickly solve for F:

$$F = g_0 h_0 \cdot g_1 h_1 \cdot g_2 h_2 \ldots g_{n-1} h_{n-1}$$

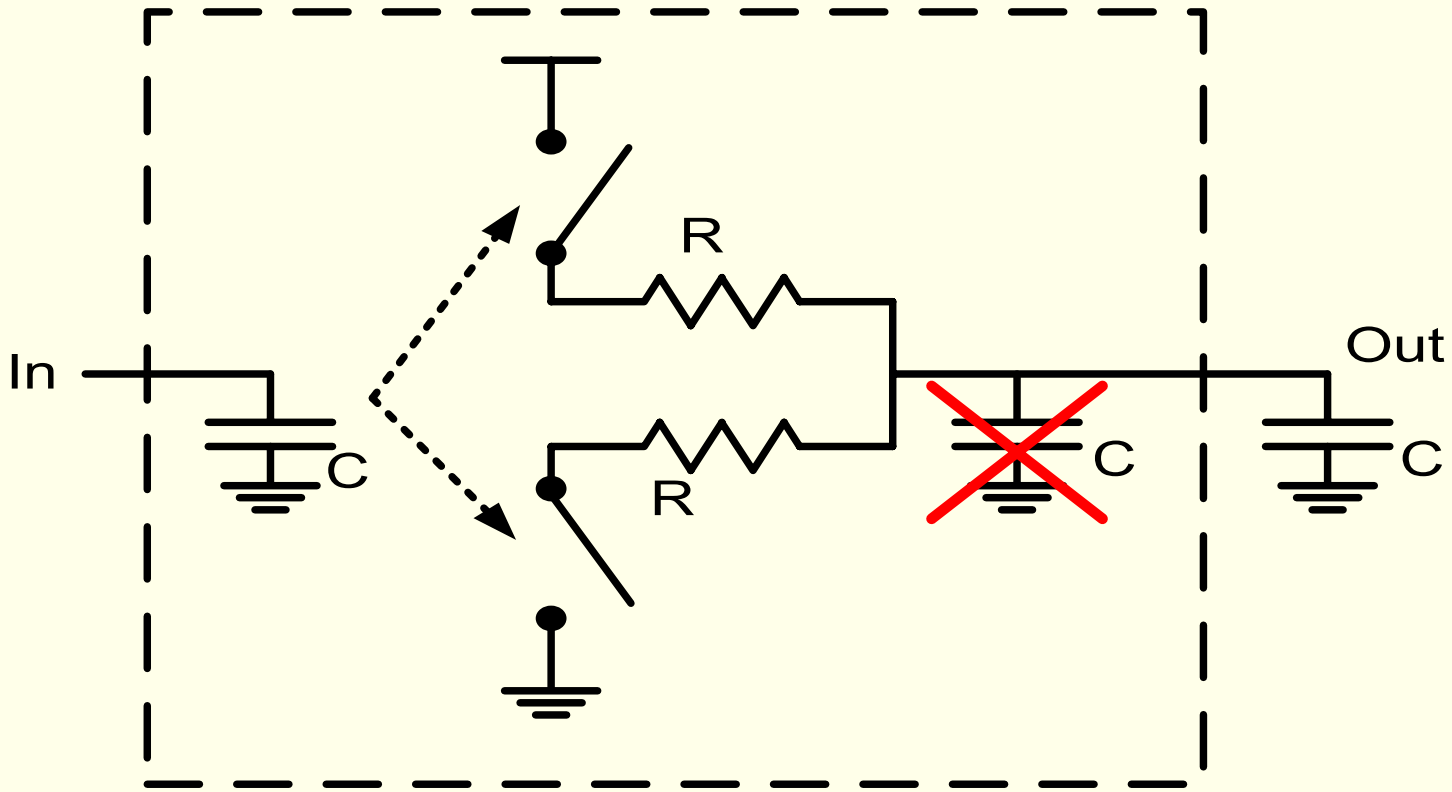
$$H = h_0 \cdot h_1 \cdot h_2 \ldots h_{n-1} = \frac{C_{out}}{C_{in}}$$

$$G = g_0 \cdot g_1 \cdot g_2 \ldots g_{n-1}$$

$$\boxed{F = G \cdot H}$$

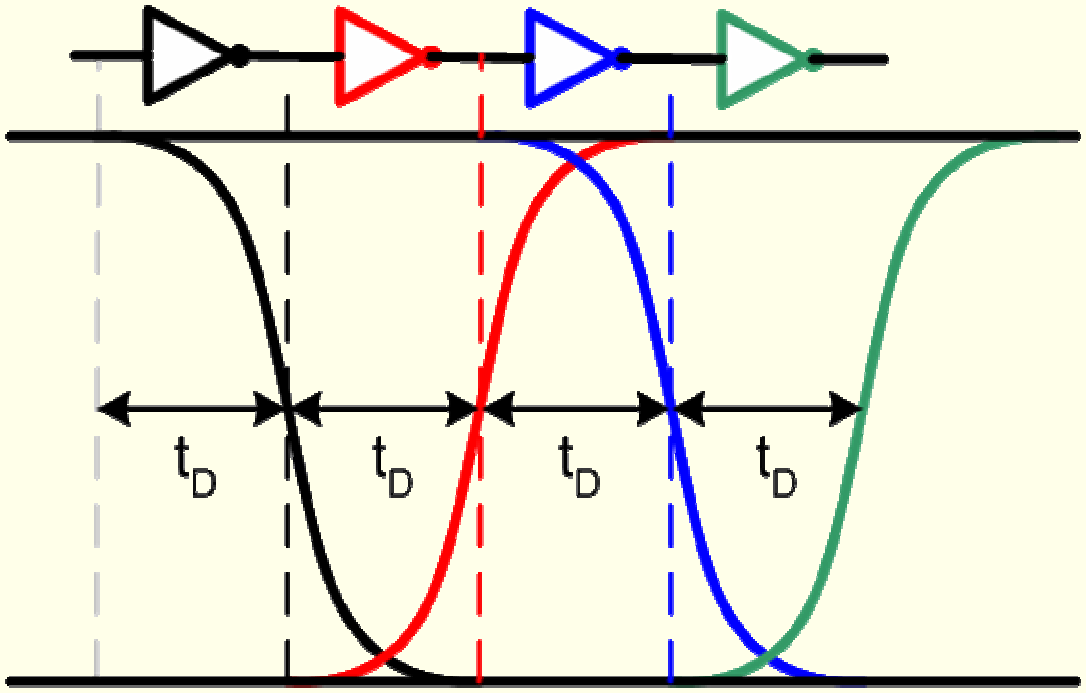
Harris, Sutherland, Sproull

LE Delay Model for Path Optimization



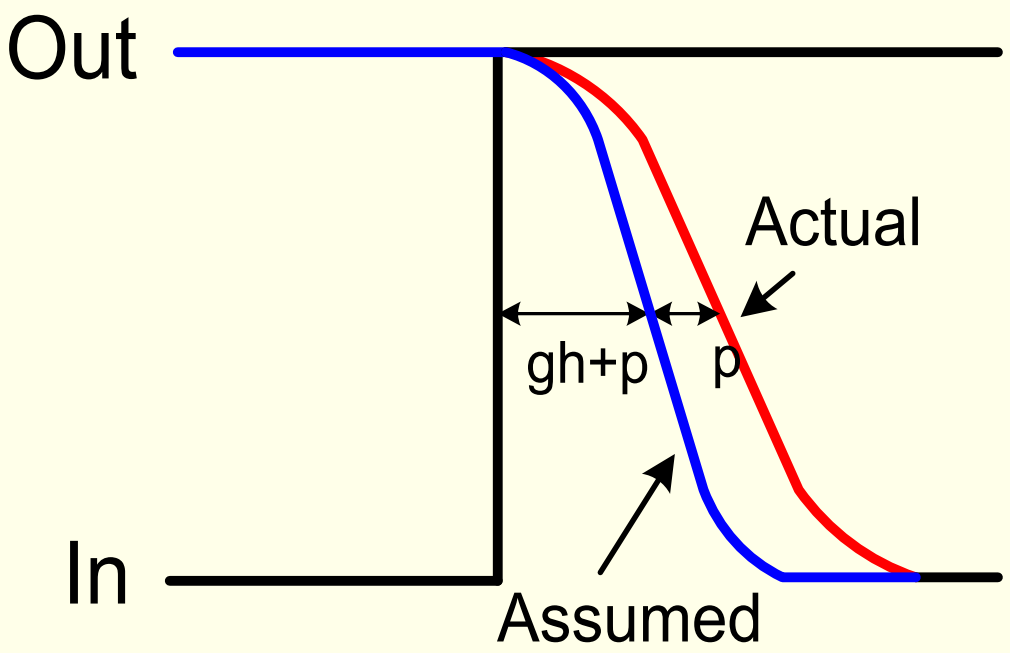
- Input slope effects on g and p are ignored
- Parasitics assumed *independent* of input slope
  - Therefore have no effect on optimization result

LE Slopes Assumed for Path Optimization



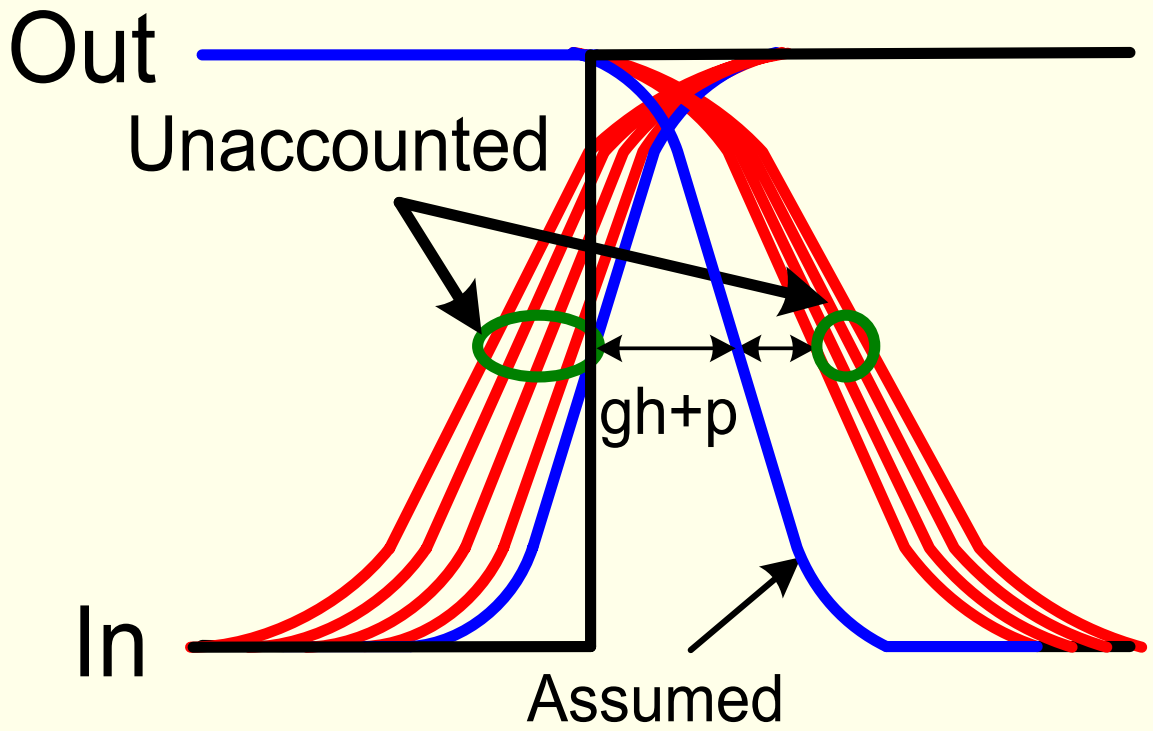
- Slopes are drawn for each gate assuming step input
- Slopes are equal for each gate at completion of optimization
  - f_i is constant, thus constant RC (ignoring parasitics).

LE Optimization Assumptions



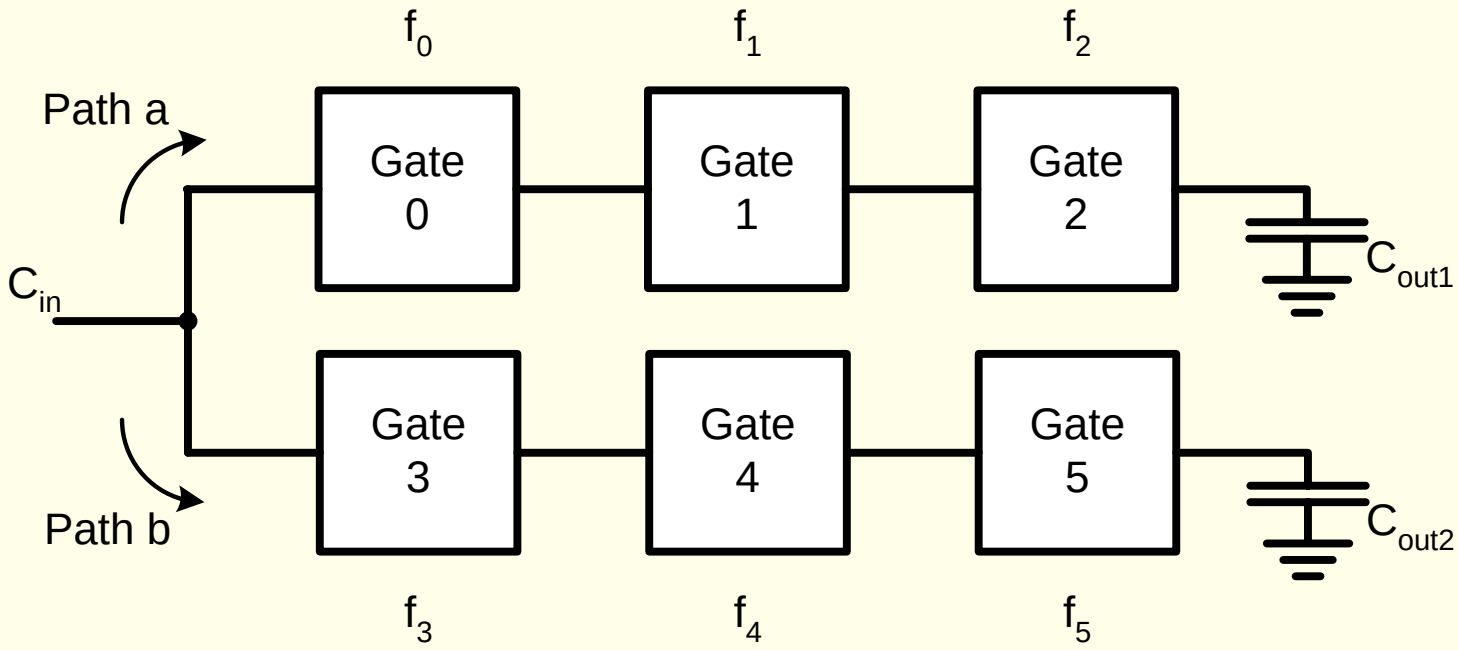
- gh is constant when using LE optimization
- Slope variations are due to differences in p and are **not** accounted for in optimization

LE Slope Variation Effects



- Input slope degradation causes output slope degradation
- Different gate types also cause output slope variation

Logical Effort for Multi-path



From LE,

$$f_0 = f_1 = f_2 \quad \text{and} \quad f_3 = f_4 = f_5$$

Minimum delay occurs when $D_a = D_b$ or $F_a = F_b$ (ignoring parasitics)

$$f_0 = f_1 = f_2 = f_3 = f_4 = f_5$$

Logical Effort for Multi-path (cont.)

$$D_a = [(g_0 h_0 + p_0) + (g_1 h_1 + p_1) + (g_2 h_2 + p_2)] \cdot \tau$$

$$D_b = [(g_3 h_3 + p_3) + (g_4 h_4 + p_4) + (g_5 h_5 + p_5)] \cdot \tau$$

$$f_0 f_1 f_2 = g_0 g_1 g_2 \cdot \frac{C_{out1}}{C_0} \qquad f_3 f_4 f_5 = g_3 g_4 g_5 \cdot \frac{C_{out2}}{C_3}$$

$$C_0 = \frac{g_0 g_1 g_2 \cdot C_{out1}}{f_0 f_1 f_2} \qquad C_3 = \frac{g_3 g_4 g_5 \cdot C_{out2}}{f_3 f_4 f_5}$$

Branching: Ratio of total capacitance to on-path capacitance

$$b_a = \frac{C_o + C_3}{C_0} \qquad b_b = \frac{C_3 + C_0}{C_3}$$

Logical Effort for Multi-path (cont)

Substituting C_0 and C_3

$$b_a = \frac{\frac{g_0 g_1 g_2 \cdot C_{out1}}{f_0 f_1 f_2} + \frac{g_3 g_4 g_5 \cdot C_{out2}}{f_3 f_4 f_5}}{\frac{g_0 g_1 g_2 \cdot C_{out1}}{f_0 f_1 f_2}}$$

Assume Minimum Delay occurs when $f_0 f_1 f_2 = f_3 f_4 f_5$

$$b_a = \frac{g_0 g_1 g_2 \cdot C_{out1} + g_3 g_4 g_5 \cdot C_{out2}}{g_0 g_1 g_2 \cdot C_{out1}}$$

Similarly

$$b_b = \frac{g_3 g_4 g_5 \cdot C_{out2} + g_0 g_1 g_2 \cdot C_{out1}}{g_3 g_4 g_5 \cdot C_{out2}}$$

- Branching = 2 when $G_a = G_b$ and $C_{out1} = C_{out2}$

Complex Multi-path Optimization

If each path has internal branching, b_a and b_b are as follows

$$b_a = \frac{g_0 b_0 g_1 b_1 g_2 b_2 \cdot C_{out1} + g_3 b_3 g_4 b_4 g_5 b_5 \cdot C_{out2}}{g_0 b_0 g_1 b_1 g_2 b_2 \cdot C_{out1}}$$

$$b_b = \frac{g_3 b_3 g_4 b_4 g_5 b_5 \cdot C_{out2} + g_0 b_0 g_1 b_1 g_2 b_2 \cdot C_{out1}}{g_3 b_3 g_4 b_4 g_5 b_5 \cdot C_{out2}}$$

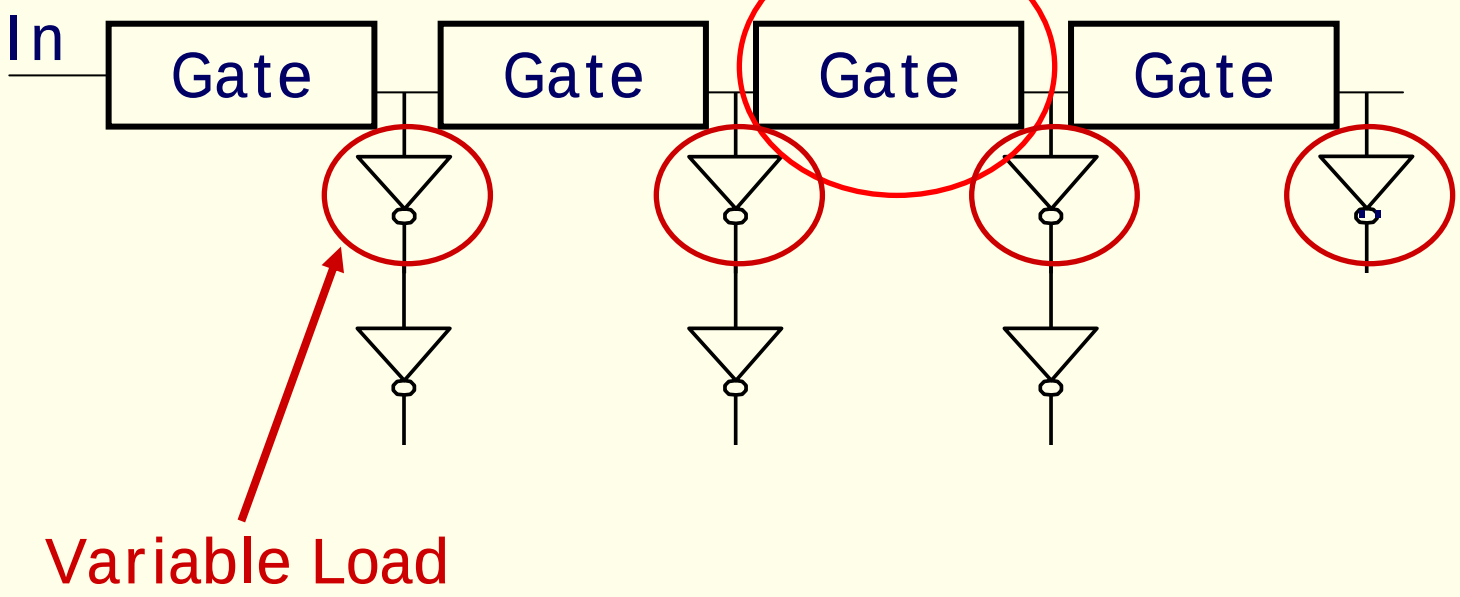
- Note: This solution and previous solution differ from that are described in LE book (all simplify by ignoring parasitics)

Slopes in LE

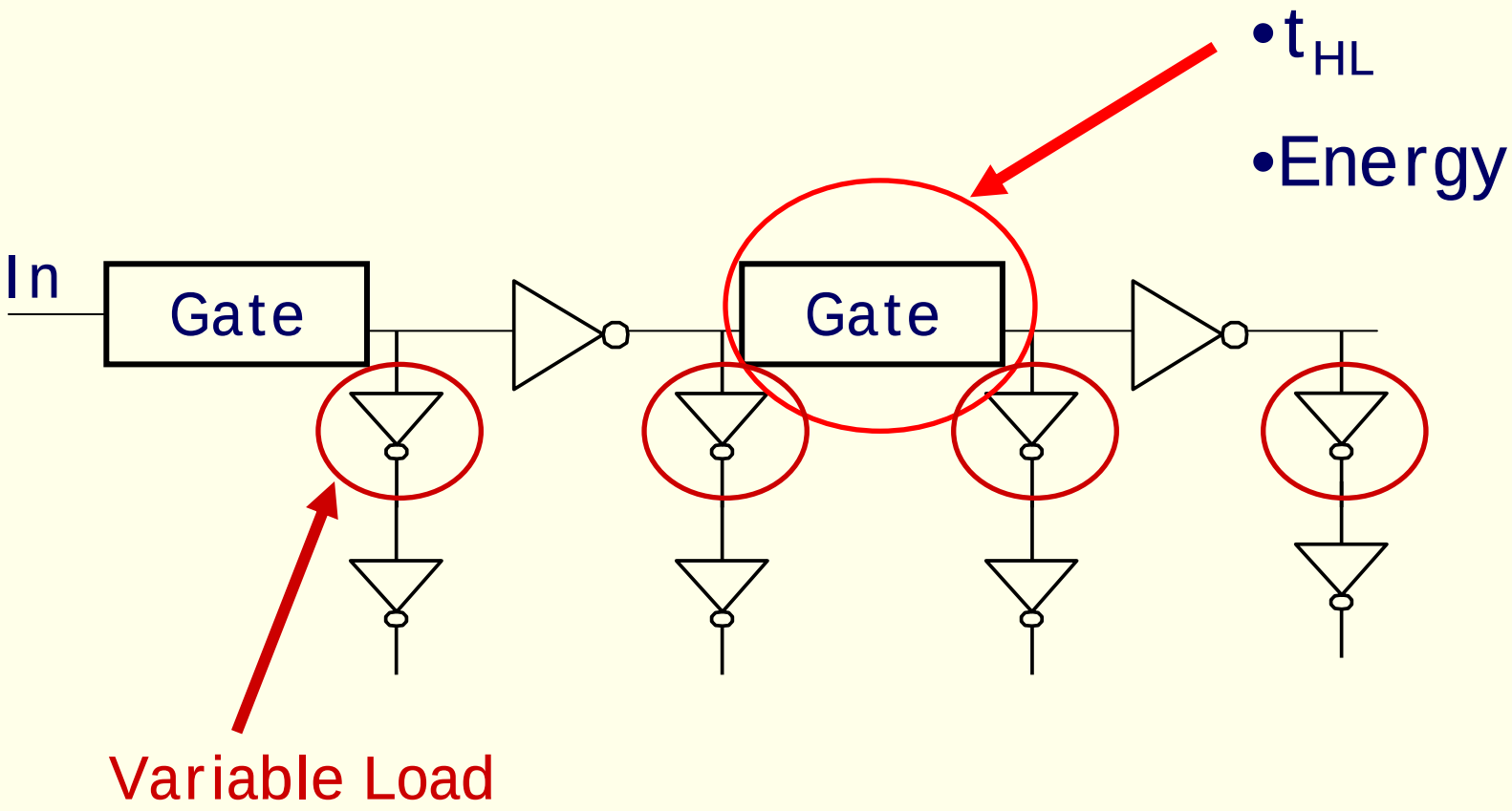
- LE optimization assumes step input for gate
  - Hence, slope variations are not accounted for.
- Spice characterization is invalid for step input derivation of Logical Effort, as g and p relate to non-practical values.
- Fortunately Logical Effort can be derived assuming equal slopes if we assume no g and p dependence on input slope variation. These “improved” g and p values can be obtained from spice characterization

LE Characterization Setup for Static Gates

- t_{LH}
- t_{HL}
- Energy

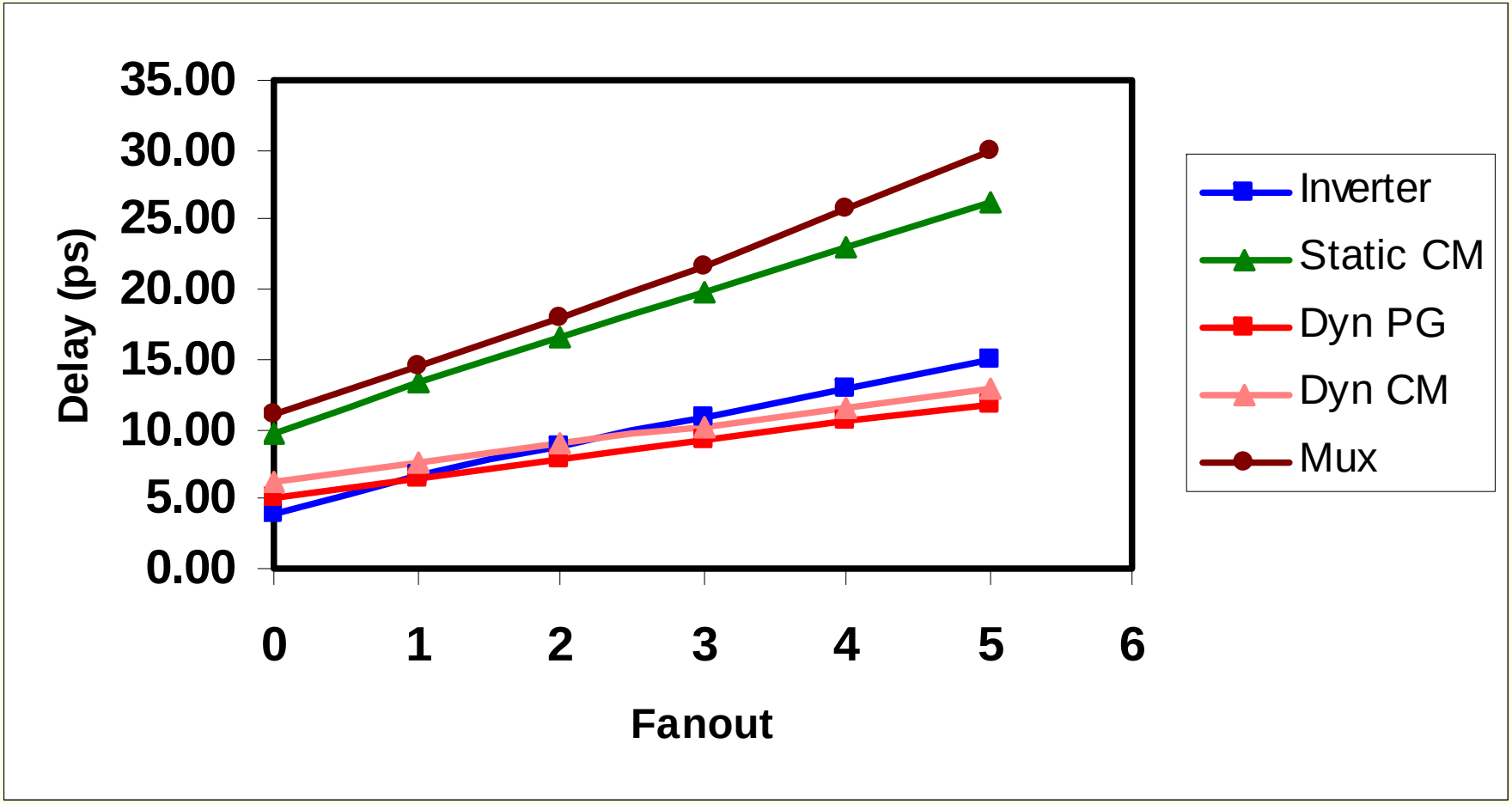


LE Characterization Setup for Dynamic Gates



L.E for Adder Gates

**from Mathew Sanu / D. Harris*



- Logical effort parameters obtained from simulation for std cells
- Define logical effort 'g' of inverter = 1
- Delay of complex gates can be defined with respect to **d=1**

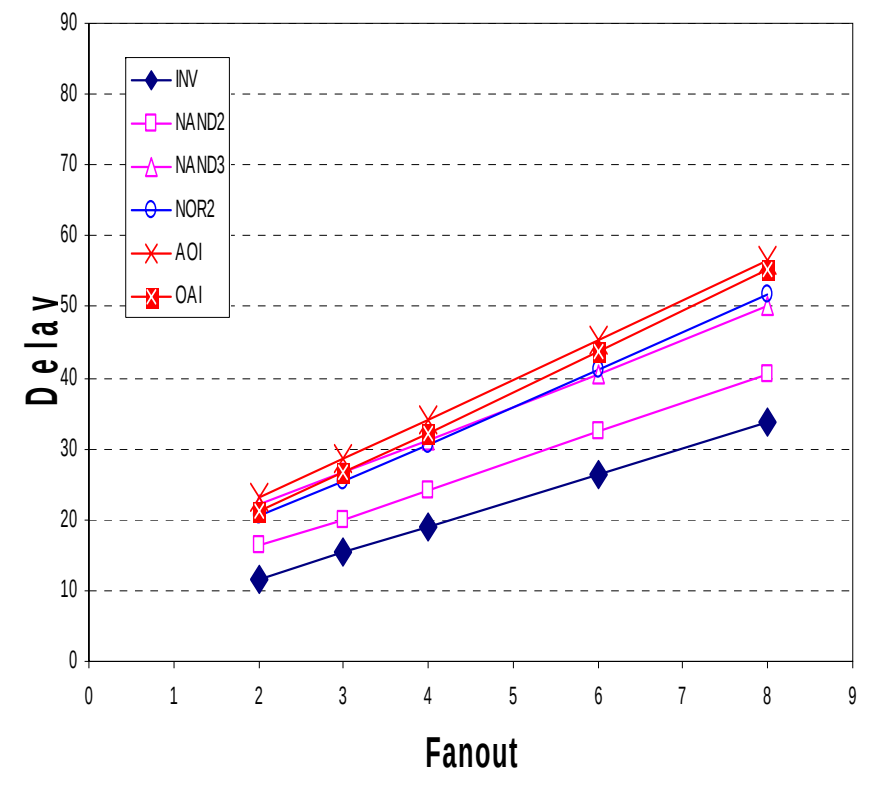
Normalized L.E

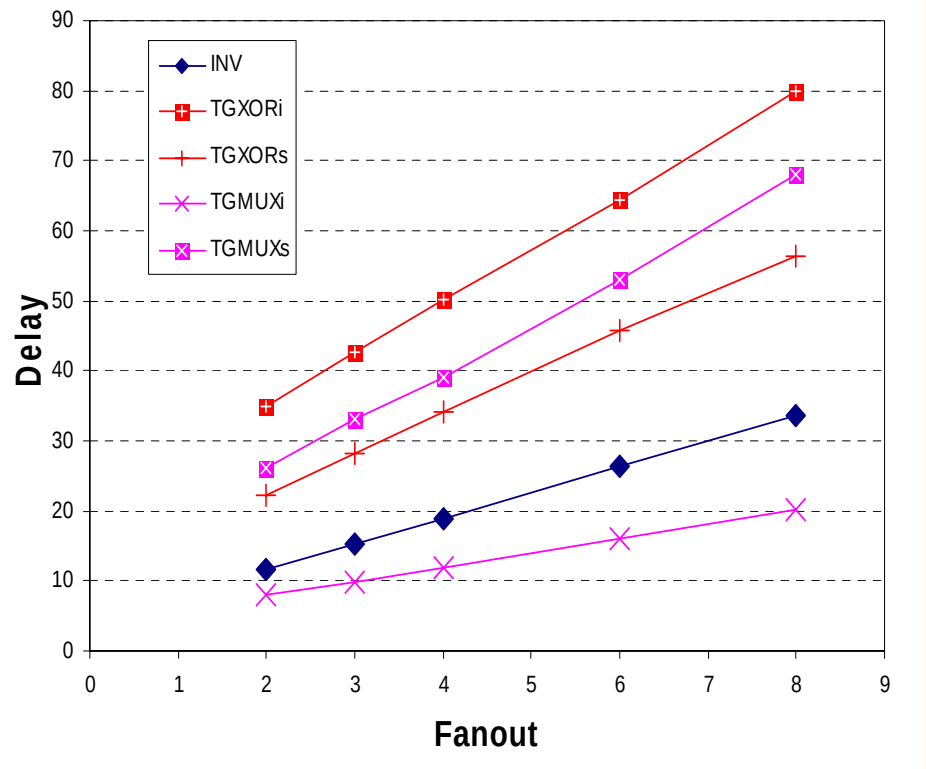
**from Mathew Sanu*

Gate type	Logical Eff. (g)	Parasitics (Pinv)
Inverter	1	1
Dyn. Nand	0.6	1.34
Dyn. CM	0.6	1.62
Dyn. CM-4N	1	3.71
Static CM	1.48	2.53
Mux	1.68	2.93
XOR	1.69	2.97

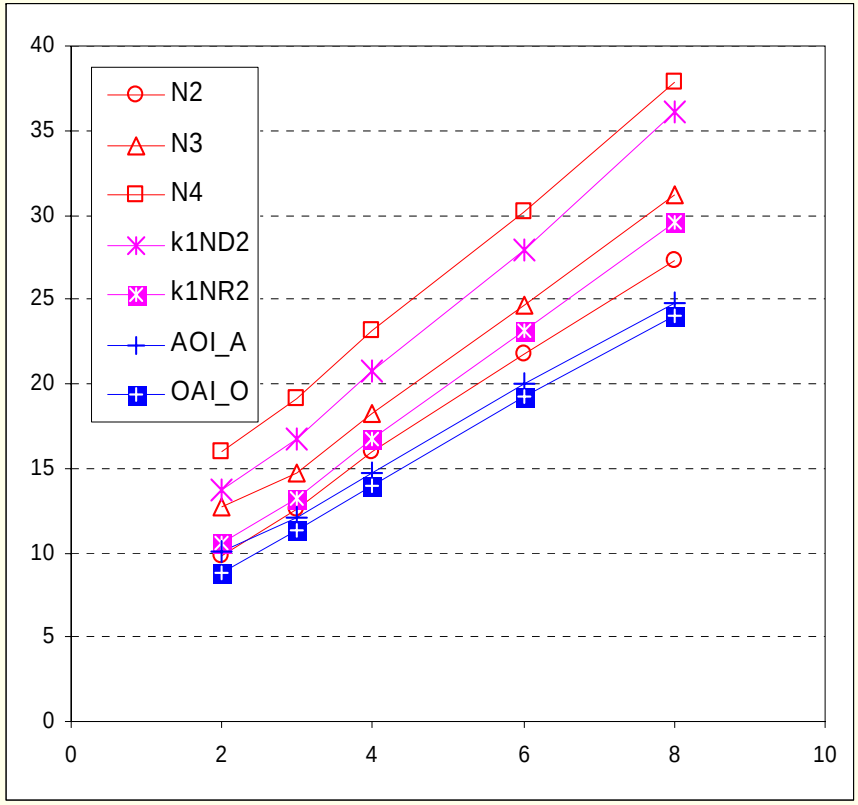
- Logical effort & parasitic delay normalized to that of inverter

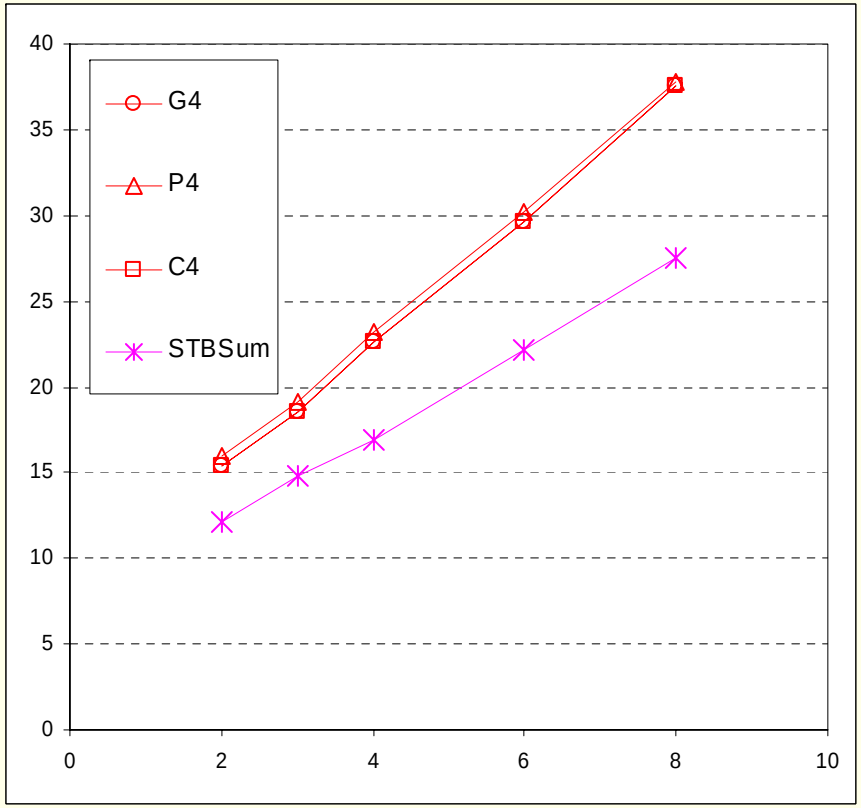
Static CMOS Gates: Delay Graphs





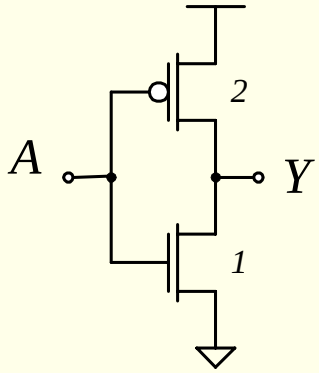
Dynamic CMOS: Delay Graphs





Simple logic gates:

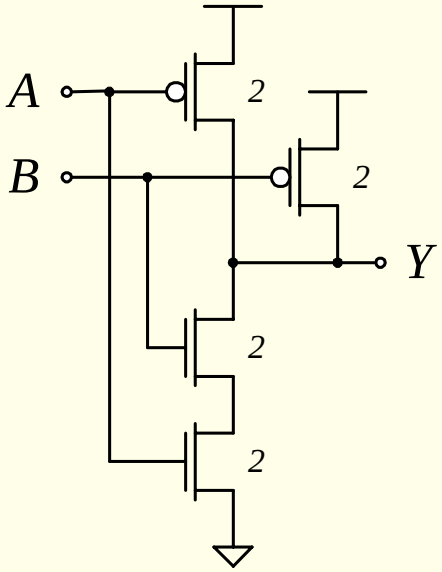
(a) reference static inverter, (b) two-input static NAND gate, (c) two-input static NOR gate, (d) domino-style inverter



$$RC_{A,inv} = 1 \cdot (2+1) = 3$$

$$g_{A,inv} = \frac{RC_{A,inv}}{RC_{A,inv}} = 1$$

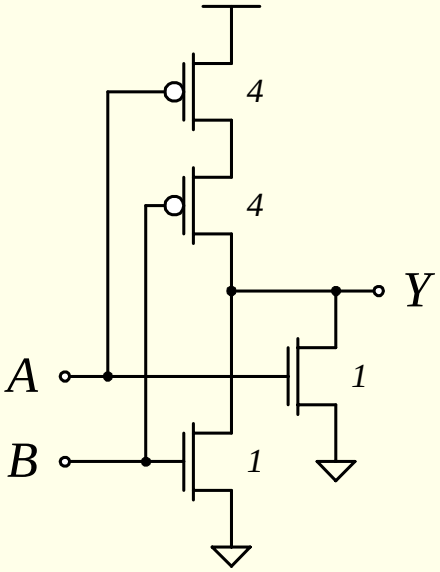
(a)



$$RC_{A,nand\,2} = \left(\frac{1}{2}+\frac{1}{2}\right) \cdot (2+2) = 4$$

$$g_{A,nand\,2} = \frac{RC_{A,nand\,2}}{RC_{A,inv}} = \frac{4}{3}$$

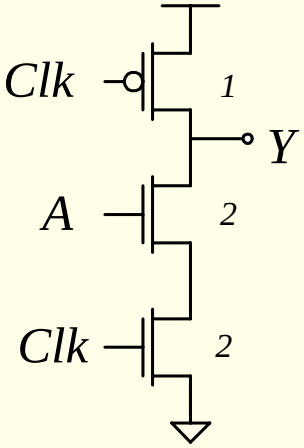
(b)



$$RC_{A,nor\,2} = 1 \cdot (4+1) = 5$$

$$g_{A,nor\,2} = \frac{RC_{A,nor\,2}}{RC_{A,inv}} = \frac{5}{3}$$

(c)

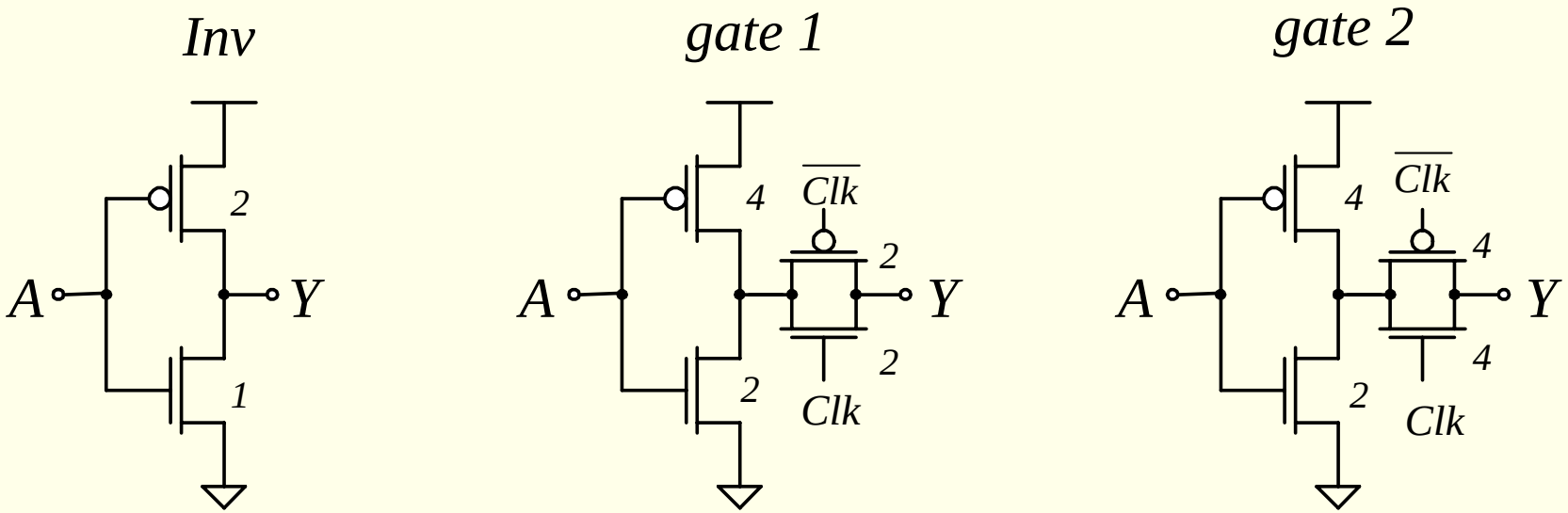


$$RC_{A,Dom-inv} = \left(\frac{1}{2}+\frac{1}{2}\right) \cdot 2 = 2$$

$$g_{A,Dom-inv} = \frac{RC_{A,Dom-inv}}{RC_{A,inv}} = \frac{2}{3}$$

(d)

Logical effort of gate driving a transmission gate:
(a) reference inverter, (b) slow-data input, (c) fast-data input sizing



$$RC_{A,inv} = 1\cdot(2+1) = 3 \qquad RC_{A,gate1} = \left(\frac{1}{2}+\frac{1}{2}\right)\cdot(4+2) = 6 \qquad RC_{A,gate2} = \left(\frac{1}{2}+\frac{1}{4}\right)\cdot(4+2) = \frac{9}{2}$$

$$g_{A,inv} = \frac{RC_{A,inv}}{RC_{A,inv}} = 1 \qquad g_{A,gate1} = \frac{RC_{A,gate1}}{RC_{A,inv}} = 2 \qquad g_{A,gate2} = \frac{RC_{A,gate2}}{RC_{A,inv}} = \frac{3}{2}$$

(a) (b) (c)

Environment Setup

Environment Setup

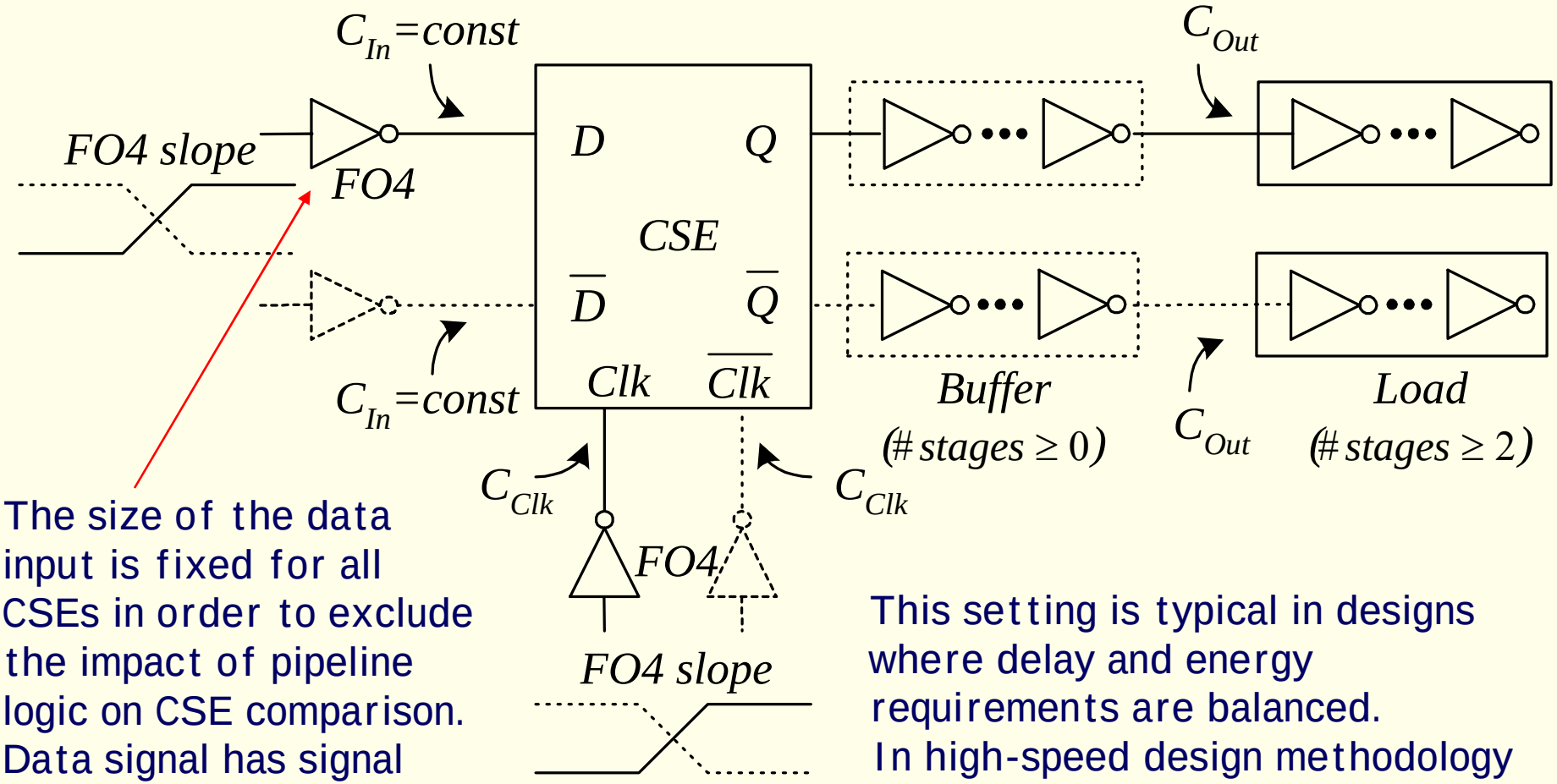
Several studies used different simulation setups. We present some of the important concepts and global simulation setup framework that can be further tuned for the particular intended application:

- The setup for the comparison of the CSEs in (Stojanovic and Oklobdzija 1999) used a single size load.
  - Load was chosen that resembles the typical situation in a moderately-to-heavy loaded critical path in a processor with a lot of parallelism. All the CSEs were sized such as to achieve optimum data-to-output delay for given output load, while driven from the fixed size inverters.
- In most practical situations the CSEs are designed in a discrete set of sizes, each optimized for a particular load. It is useful to examine the performance of the CSE for a range of the loads around the load for which the CSE was optimized. This technique is illustrated in (Nikolic and Oklobdzija 1999)
  - Different CSEs are initially sized to drive a fixed load, and the load is then varied.
  - The delay of a CSE exhibit linear dependence on the load, with the slope of the delay curve illustrating the logical effort of the driving stage of the CSE and the zero-load crossing illustrating the parasitic delay of the driving stage together with the delay of the inner stages of the CSE.
  - This is not the optimal behavior of the CSE delay curve, but is the best that can be achieved when there are only a few CSE sizes available in the library.

Environment Setup

- In case where CSE can be re-optimized for each particular load, further speedup can be achieved since the effort can be shared between stages rather than resting solely on the output stage. This approach was illustrated in (Heo and Asanovic 2001).
  - Contrary to their conclusions, in case when a general performance of a CSE needs to be assessed, the proper approach is to optimize the CSE for the most important application that determines the performance of the whole system, not the most frequent application.
- In high-speed systems the most important are the elements on the critical path which is typically moderately-to-heavy loaded due to branching to parallel execution units and wire capacitance. The small number of critical paths in a processor does not decrease their importance since it is their delay that determines the clock rate of the whole system.
- The performance of the large number of the lightly loaded CSEs that are placed off of the critical path is of concern only if it can be traded for energy savings.
- The simulation approach should attempt to resemble the actual data-path environment. The number of logic stages in a CSE and their complexity highly depend on particular circuit implementation, leading to differences in logical effort, parasitic delay and energy consumption.
- Every CSE structure needs to be optimized to drive the load with best possible effort delay.

General simulation setup



The size of the data input is fixed for all CSEs in order to exclude the impact of pipeline logic on CSE comparison. Data signal has signal slope identical to that of a FO4 inverter, which is the case in a well-designed pipeline.

This setting is typical in designs where delay and energy requirements are balanced. In high-speed design methodology of Intel, Sun Microsystems and former Digital Corp, FO3 inverter metric is more common than FO4 due to more aggressive design style.

General simulation setup

- The size of the clocked transistors is set to the size needed in order not to compromise the speed of the whole structure.
- A direct tradeoff exists between the CSE delay and clock energy (size of clocked transistors), as some of the clocked transistors are always on the critical path of the CSE.
- Increase in sizes of clocked transistors on a critical path results in diminishing returns since data input is fixed. Depending on the CSE topology, some structures can trade delay for clocked transistor size more efficiently than others, so we allow this to happen up to a certain extent.
- Our goal is to examine CSEs that are used on a critical path, hence the assumption that designer might be willing to spend a bit more clock power to achieve better performance.
- Differences in clock loads (*CClk*) among devices illustrate potential drawbacks in terms of clock power requirements, and serve as one of the performance metrics. Clock inputs have identical signal slope to that of a FO4 inverter. This can be changed depending on the clock distribution design methodology.

General simulation setup

The question on how to compare differential and single-ended structures has always been one of the key issues among the people characterizing and designing the CSEs.

- Differential and single-ended structures should not be compared with each other, due to the overhead that single-ended structures incur to generate the complementary output.
- We do not require that single-ended structure generate both true and complementary value at the output.

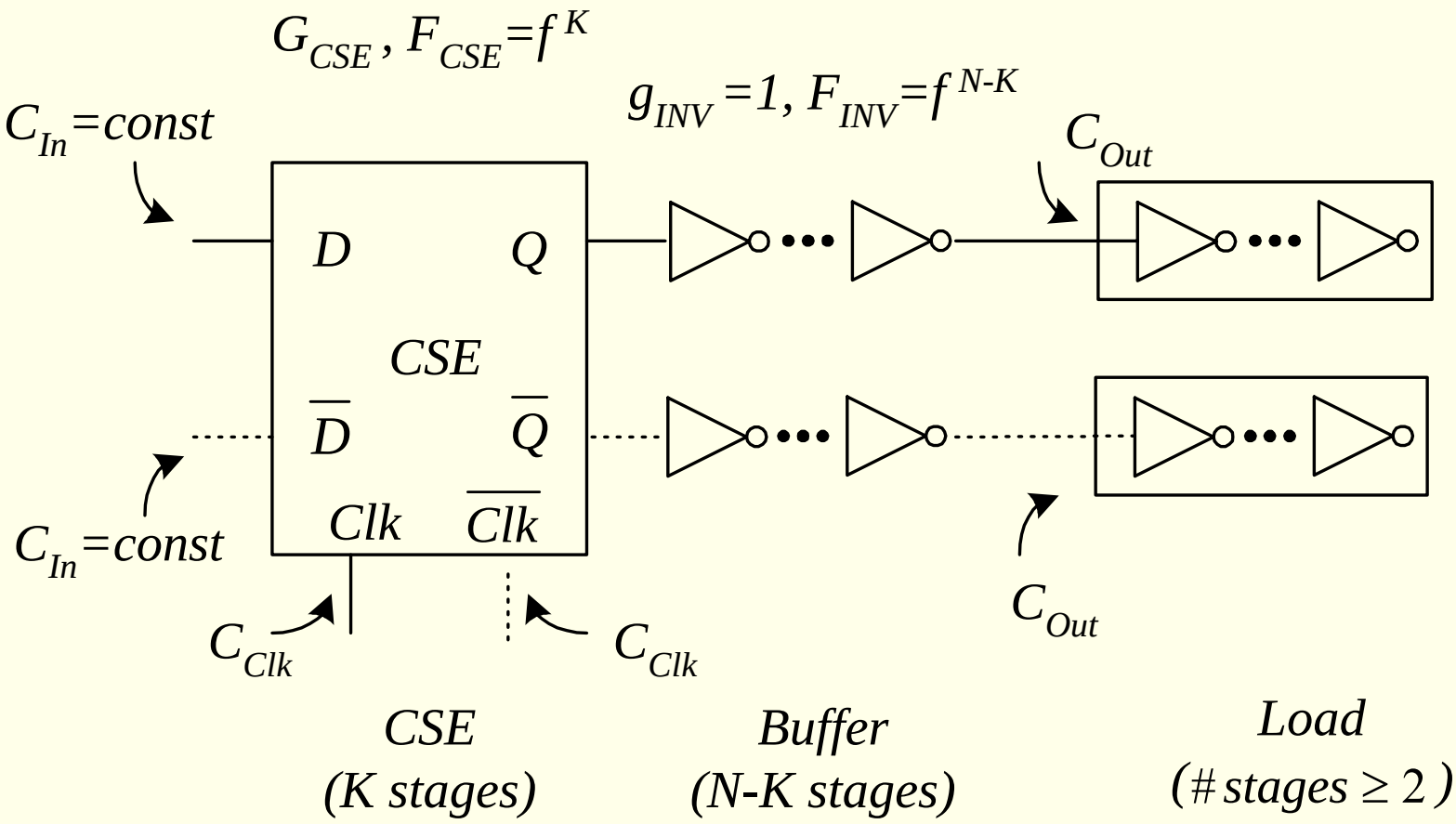
- The worst-case analysis requires that the CSE generates the output that has worse data-to-output delay. However, it is also beneficial to measure both *D*-*Q* and *D*-$\overline{Q}$ delay.

- Any imbalance between the two can lead to big delay savings in cases where proper logic polarity manipulation in the stages preceding or following the CSE can change the polarity requirement of the CSE, and hence its data-to-output delay.
- Load model always consists of several inverters in a chain to avoid the error in delay caused by Miller capacitance effects from the fast switching load back to the driver.

General simulation setup

- The logical effort framework offers analogy between the CSE and a simple logic gate.
  - At light load, logic gate is dominated by its parasitic delay, i.e. self-loading.
  - At high load, effort delay becomes the dominant factor.
- Similarly, at light load, delay of a CSE with large number of stages is entirely dominated by parasitic delay.
- However, at high load, more stages are beneficial in reducing the effort delay, which then dominates over parasitic delay.
- Therefore, the performance of the CSE is best assessed if it is evaluated in a range of output loads of interest for the particular application.
- CSE evaluation can either be performed using some representative critical path load or a set of loads can be used in which case CSE has to be re-optimized for each load setting.

Additional buffering in simulation test bed



Depending on the choice of the output load size, some CSE structures with inherently small number of stages and high logical effort may require additional buffering in order to achieve the best effort delay.

Additional buffering in simulation test bed

For each CSE we need to find optimal effort per stage and number of stages to drive the required load. Starting from total electrical fan-out *H*, optimal number of stages *N* is obtained through rounding of the logarithm of the total effort of the path (assuming g_{INV} is unity):

$$H = \frac{C_{OUT}}{C_{IN}} \qquad N = round\left(\log_4\left(G_{CSE}H\right)\right)$$

The logarithm is of base 4 since stage effort of 4 is a target for optimal speed. Once the integer number of stages is obtained, the updated value of stage effort is found from:

$$f = \sqrt[N]{G_{CSE}H}$$

After the stage effort is obtained, CSE internal stages have to be resized for new stage effort and also the external inverters, if there are any.
This sizing approach is optimal even in case no additional inverters are required, since it will serve to distribute the effort between the internal stages of the CSE.

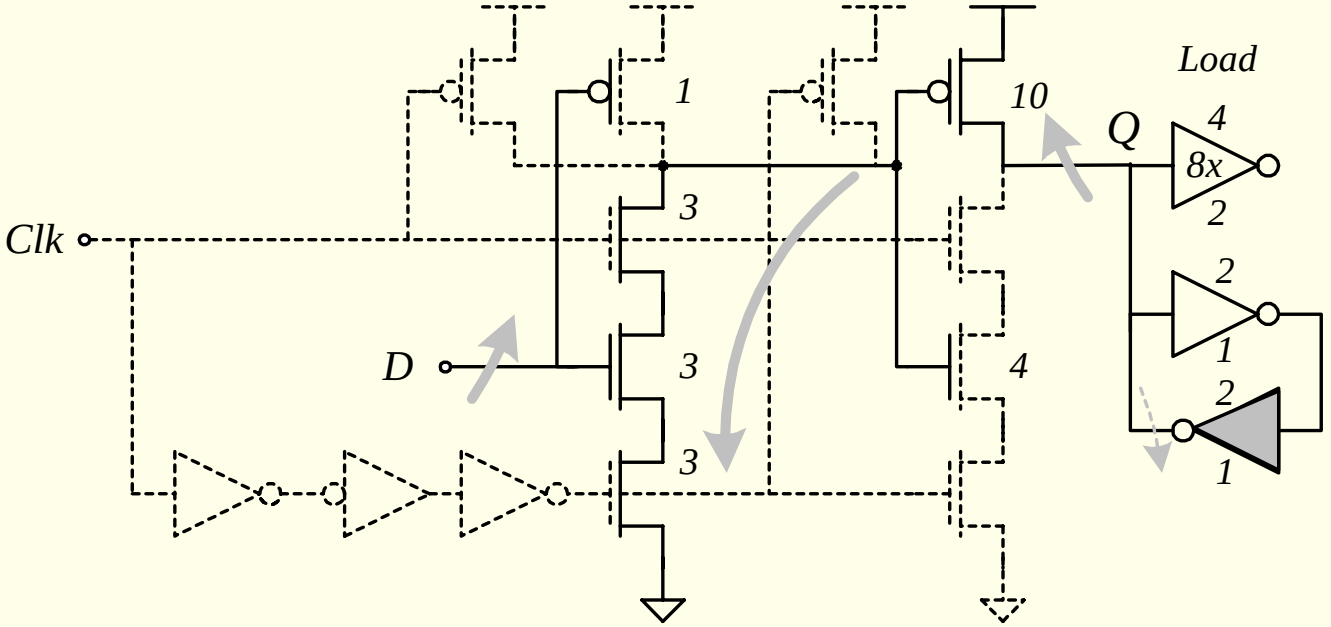
HLFF Sizing Example

HLFF Sizing Example

In this problem we observe the change in minimum data-to-output (*D*-*Q* or *D*- $\overline{Q}$) delay as the output load of the CSE increases:

- Before we start investigating the effect of different loads on the sizing of the HLFF, we show how the logical effort can be calculated for the given sizing.
- It is relatively easy to recognize that the HLFF is built up of three-input static NAND gate as the first stage and domino-like three-input NAND in the second stage.
- Minor variations from standard static NAND sizing for equal logical effort on all inputs are needed to speed up the data input and enable the first stage to evaluate before the transparency window closes. Similar situation occurs in the second stage.
- This HLFF sizing example also illustrates the application of logical effort to skewed gates (gates in which one output transition is faster than the other), and gates with keepers.

HLFF sizing example



$$g_{1,0\to1} = \frac{\left(3\cdot\frac{1}{3}\right)\cdot(1+3)}{1\cdot(1+2)} = \frac{4}{3} \qquad g_{2,1\to0} = \frac{\left(\frac{1}{\frac{10}{2}-1}\right)\cdot(4+10)}{1\cdot(1+2)} = \frac{7}{6}$$

$$f_{1,0\to1} = \frac{4}{3}\cdot\frac{4+10}{3+1} = 4.7 \qquad f_{2,1\to0} = \frac{7}{6}\cdot\frac{8\cdot(2+4)+(1+2)}{4+10} = 4.25$$

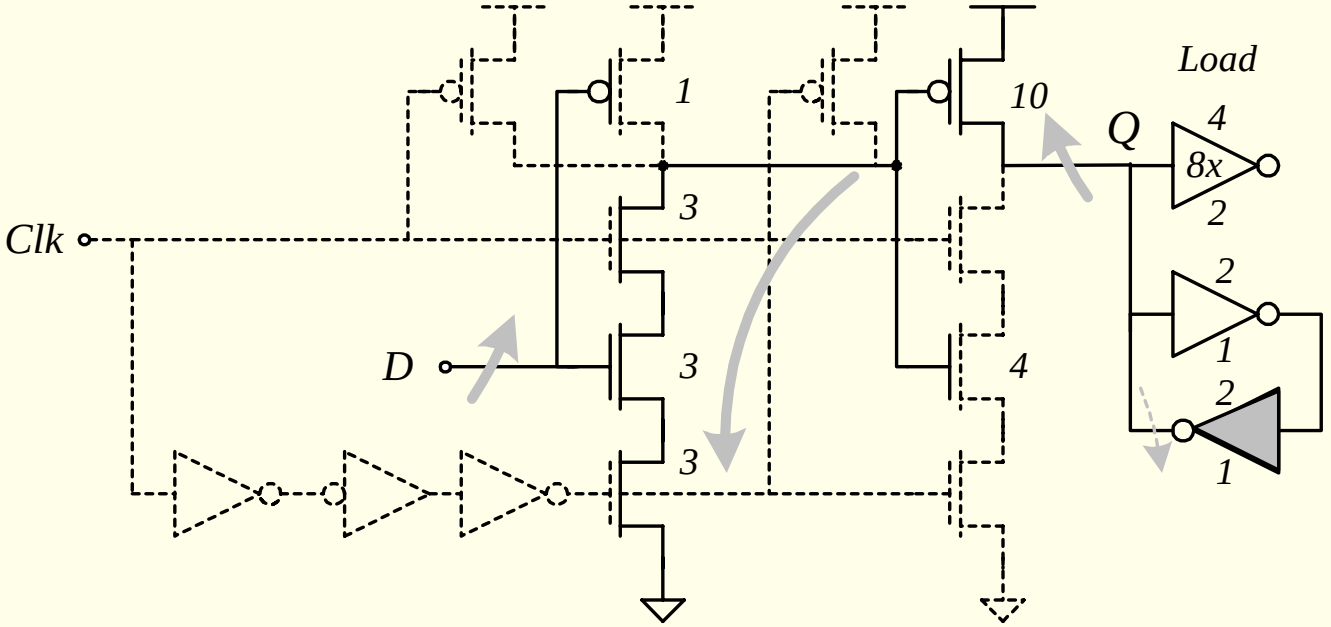
The critical path of the HLFF is exercised with a "0-to-1" transition at data input. The first stage of the HLFF is a skewed NAND gate.

HLFF sizing example

- The logical effort calculation of the second stage is slightly more complicated because of the keeper inverter pair. A keeper sinks a portion of the current that is sourced by the PMOS transistor to node *Q* which can be taken into account as negative conductance.
- This negative conductance accounted for by subtracting the conductance of the NMOS transistor (1) of the shaded keeper inverter, from the conductance of the driving PMOS transistor (10/2).
- For the particular load, efforts per stage were calculated to be 4.7 and 4.25, which is near the optimum value of 4, indicating that this example sizing is nearly optimal.

- The sizing in example above is somewhat simplified because the short channel stack effect has not been taken into account, the logical effort values for the NMOS transistor stack are somewhat pessimistic.
- Once the logical effort of each stage is known it can be used to adjust the sizing of each stage as the load is increased or decreased.
- The alternative method is to use one of the automated circuit optimizers, however, it is not recommend it as initial method, simply because it is essential that designer gets to know the circuit through manual sizing and logical effort estimation.
- This builds intuition about the circuit and ability to verify optimizer results.

HLFF sizing example



$$g_{1,0\to1} = \frac{\left(3\cdot\frac{1}{3}\right)\cdot(1+3)}{1\cdot(1+2)} = \frac{4}{3} \qquad g_{2,1\to0} = \frac{\left(\frac{1}{\frac{10}{2}-1}\right)\cdot(4+10)}{1\cdot(1+2)} = \frac{7}{6}$$

$$f_{1,0\to1} = \frac{4}{3}\cdot\frac{4+10}{3+1} = 4.7 \qquad f_{2,1\to0} = \frac{7}{6}\cdot\frac{8\cdot(2+4)+(1+2)}{4+10} = 4.25$$

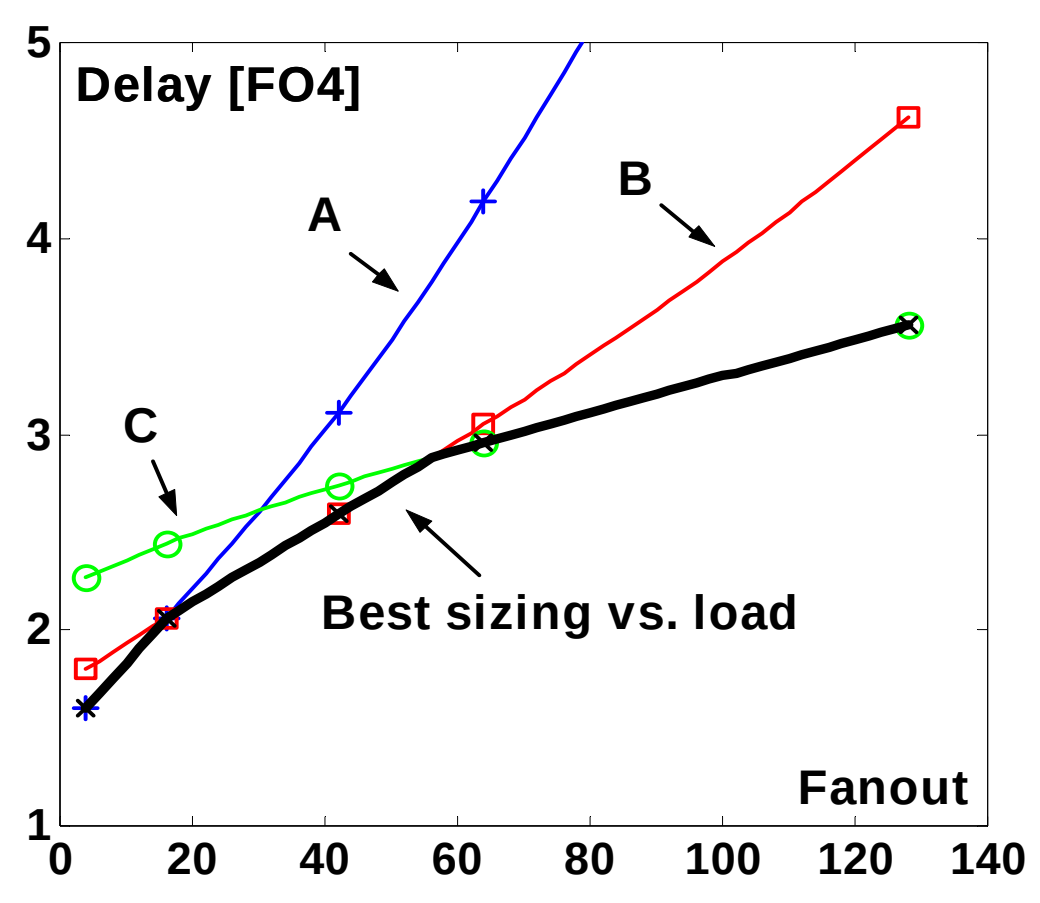
The critical path of the HLFF is exercised with a "0-to-1" transition at data input. The first stage of the HLFF is a skewed NAND gate.

HLFF Delay (normalized to FO4 inverter delay) vs. Fanout for Different HLFF Cell Sizes

Fanout	4	16	42	64	128
Load-Size (#stages)					
Small-A (2)	1.60	2.06	3.11	4.19	7.80
Medium-B (2)	1.80	2.06	2.59	3.05	4.62
Large-C (2+1)	2.27	2.44	2.74	2.96	3.56

There is only one optimal solution for each load size.

Sizing versus load, HLFF example: linear scale



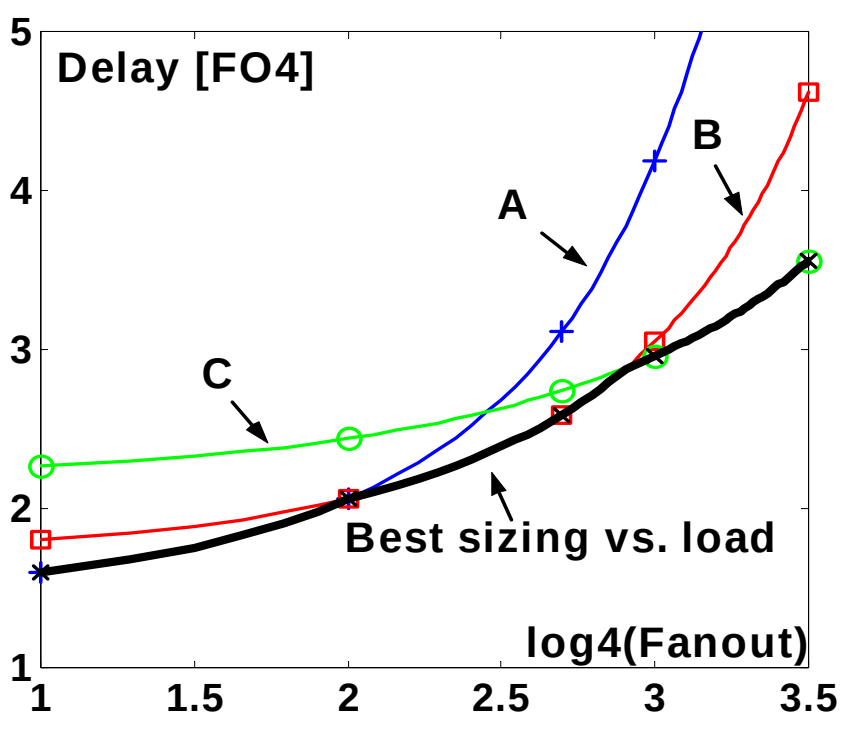
According to the logical effort theory, the optimal delay versus fan-out curve should have logarithmic shape, which indeed holds for the "best sizing vs. load curve".

Sizing versus load, HLFF example: log4 scale

Similarly, the optimal delay is a linear function of the logarithm of the electrical effort (fanout), as shown. The logarithmic fanout scale makes it easy to see if the stage effort is properly determined.

Delay is linear function of stage effort and number of stages.

The logarithm of the electrical effort approximately illustrates the needed number of stages, and if the delay checks out to be multiple of number of stages and FO4 delay, then the optimal effort per stage is chosen.



Case of Modified SAFF

M-SAFF Delay vs. Fanout for Different M-SAFF Cell Sizes

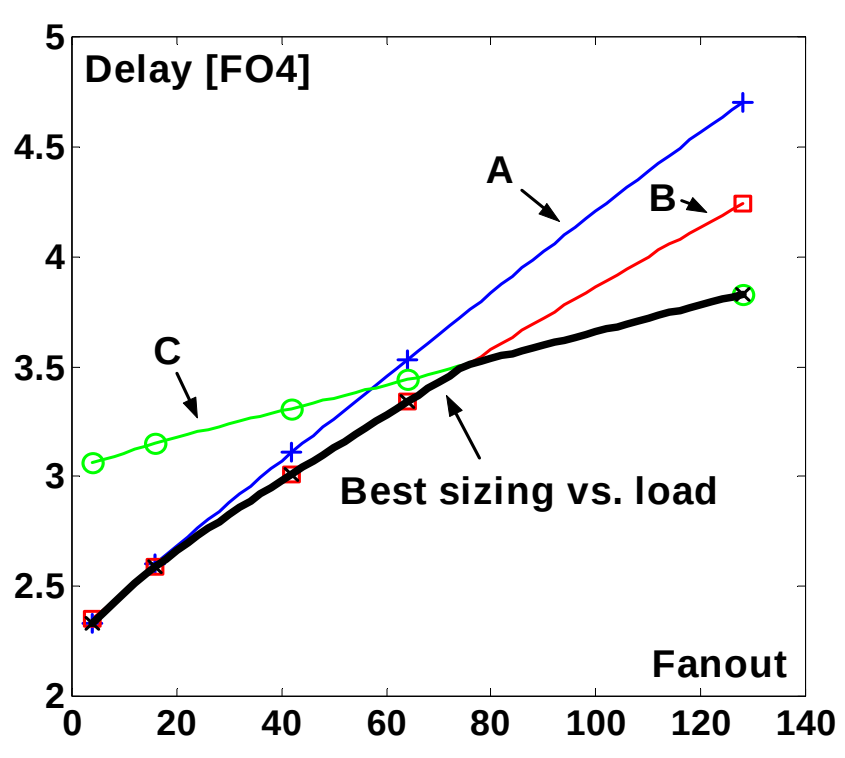
Fanout	4	16	42	64	128
Optimal Load-Size (#stages)					
Small-A (2)	2.33	2.60	3.11	3.53	4.70
Medium-B (2)	2.35	2.59	3.01	3.34	4.24
Large-C (2+1)	3.06	3.15	3.31	3.44	3.83

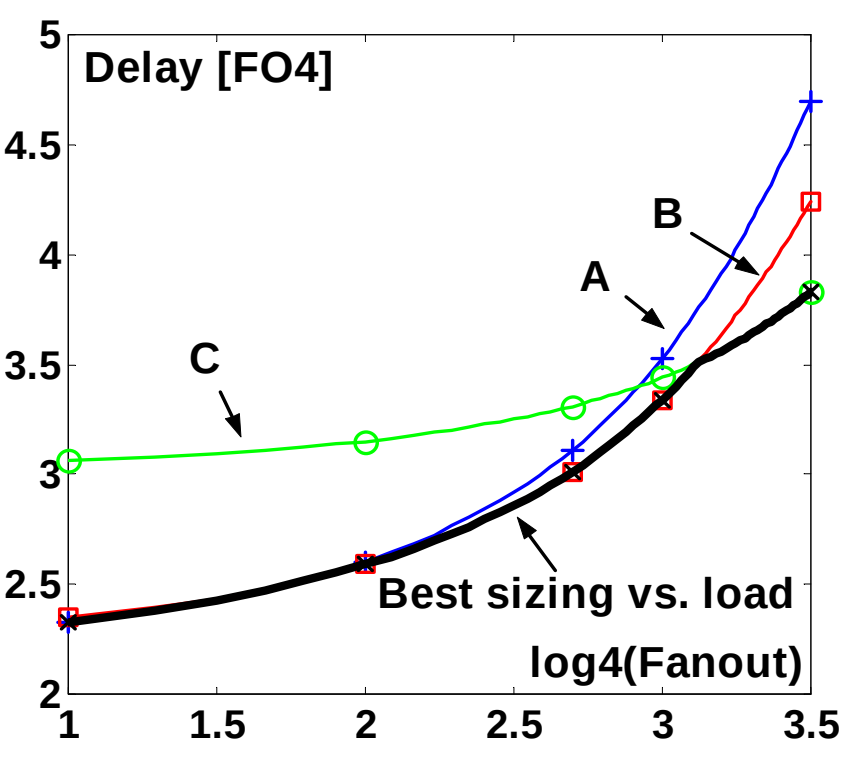
In this case we also observe the minimum data-to-output (*D-Q*) delay as the output load of the CSE increases. The performance of three different sizing solutions is illustrated versus the electrical fanout, normalized to the delay of the FO4 inverter.

Modified SAFF

- The sizing is done in a way similar to that described in the HLFF example.
- We recognize that the logical effort of the input stage is very small, better than that of an inverter, because of the small input capacitance.
- This implies that the sizing changes will mostly be located in the output stage since the input stage can accommodate larger load variations without the need for resizing.
- While it was relatively easy to find different sizes that perform better at certain loads, in the case of HLFF, it was not so in the case of M-SAFF.
- The small logical effort of the whole structure enables it to cover a huge range of loads with a single size achieving relatively good performance.
- This is the case with structure of size B in Table. Size A is only slightly better than size B, and only for very light load of FO4, and then size B device takes the lead all the way up to the FO64 after which additional inverter is needed to prevent excessive delay.

Sizing versus load, M-SAFF example: (a) linear, (b) log4 scale





-Size A is only slightly better than size B, and only for very light load of FO4

- Size B device takes the lead all the way up to the FO64 after which additional inverter is needed to prevent excessive delay.

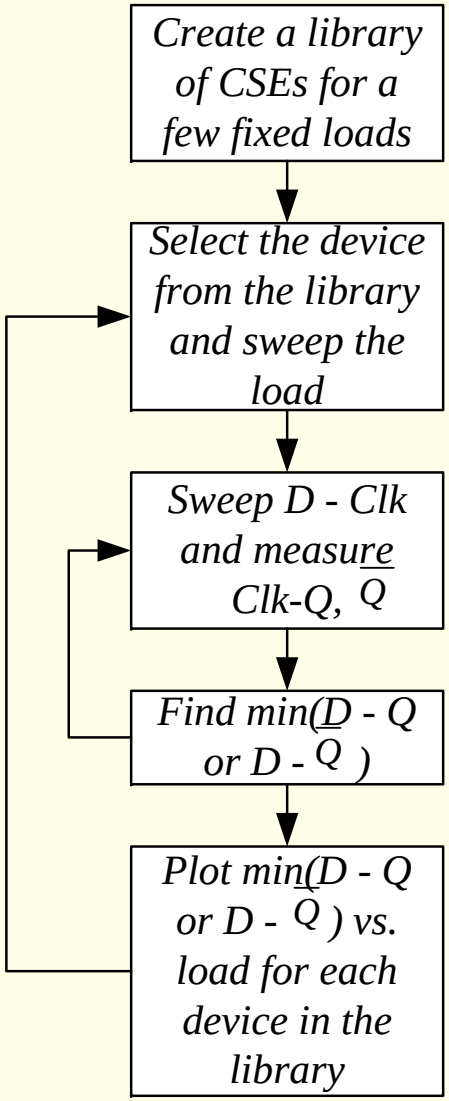
Energy Measurements

- While we were concerned about the performance aspects of the simulation setup, it is very important to prepare the simulation environment correctly such that the energy parameters of the CSE are measured accurately.
- We only need set the measurements to capture the energy for each of four possible binary transitions.
- With these values accurate average energy estimates can be made based on the statistics of the incoming data.
- More formal methods, using state transition diagrams (Zyban and Kogge 1999), can be used to exactly evaluate the effect of regular transitions and glitches on total switching energy of the CSE.
- It is essential to provide separate supply voltages for different stages of the CSE in order to measure different energies.

Automating the Simulations

- The delay vs. load CSE evaluation described in the examples can be implemented automatically.
- *Perl* is suggested as one of the most convenient scripting languages today.
- For each CSE, we need to determine:
  - the logical effort of every stage based on its topology (e.g. 2 NAND-like stages, 1 inverter stage, would be 4/3, 4/3, 1), or better yet,
  - exact logical-effort values obtained from the simulation.
  - FO4 delay and other data for a given technology process.
  - The product of logical efforts of all stages should equal the total logical effort of the CSE. After total logical effort is found, optimal number of stages and updated stage effort can be calculated.
- With stage effort and logical efforts obtained from the topology of the CSE, taking the data input of fixed size, and assuming that the clock is "on" (i.e. treating the structure as cascade of logic gates), transistor sizes for every stage can be calculated, progressing from the data input to the final load in the simulation setup.
- When a library of CSEs is created, a pre-simulation should be run for each environment parameter setup. This includes various process corners, supply voltages, etc., to determine the FO4 inverter slope and set that value as the rise/fall time of signals that drive data and clock into the CSE.

CSE flow simulation



1. For each device in the library, *D*-*Q*($\overline{Q}$) delay and *Clk*-*Q*($\overline{Q}$) delay are stored in each run, decreasing the delay between the edge of the input data and clock edge (setup time).
2. Script should check for the setup/hold time failure (i.e. when the CSE fails to pass the input value to the output).
   - This is typically detected by the large *Clk*-*Q* delay (i.e. measurement target occurred in the next cycle) or failure to measure delay if only one cycle is simulated.
3. The script automatically finds the minimum delay point at all the specified outputs.
4. The whole procedure is repeated for a range of loads and the best sizing curve.

The Appendix A of this book contains an example script written in *Perl* that can serve as a basis of a more sophisticated tool for CSE characterization. In addition it also contains example spice decks for HLFF and M-SAFF used in this example. These files are a good start for a designer who wants to evaluate various CSE topologies.