Topic 11

Logical Effort:

Designing for Speed on the Back of an Envelope

David Harris

David_Harris@hmc.edu

Harvey Mudd College

Claremont, CA

Outline

□ Introduction

□ Delay in a Logic Gate

□ Multi-stage Logic Networks

□ Choosing the Best Number of Stages

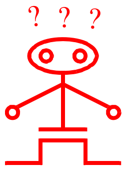
□ Example

□ Asymmetric & Skewed Logic Gates

□ Circuit Families

□ Summary

Introduction



Chip designers face a bewildering array of choices.

□ What is the best circuit topology for a function?

□ How large should the transistors be?

□ How many stages of logic give least delay?

Logical Effort is a method of answering these questions:

□ Uses a very simple model of delay

□ Back of the envelope calculations and tractable optimization

□ Gives new names to old ideas to emphasize remarkable symmetries

Who cares about logical effort?

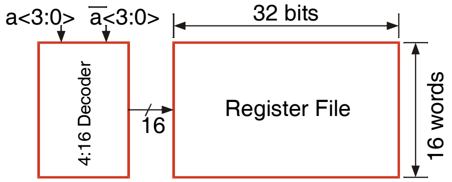
□ Circuit designers waste too much time simulating and tweaking circuits

□ High speed logic designers need to know where time is going in their logic

□ CAD engineers need to understand circuits to build better tools

Example

Ben Bitdiddle is the memory designer for the Motoroil 68W86, an embedded processor for automotive applications. Help Ben design the decoder for a register file:



Decoder specification:

□ 16 word register file

□ Each word is 32 bits wide

□ Each bit presents a load of 3 unit-sized transistors

□ True and complementary inputs of address bits *a*<3:0> are available

□ Each input may drive 10 unit-sized transistors

Ben needs to decide:

□ How many stages to use?

□ How large should each gate be?

□ How fast can the decoder operate?

Outline

□ Introduction

□ **Delay in a Logic Gate**

□ Multi-stage Logic Networks

□ Choosing the Best Number of Stages

□ Example

□ Asymmetric & Skewed Logic Gates

□ Circuit Families

□ Summary

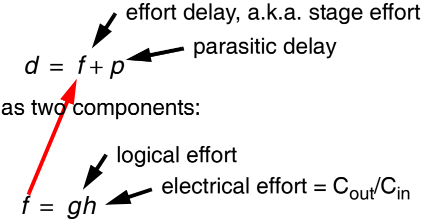
Delay in a Logic Gate

Let us express delays in a process-independent unit:

$$d = \frac{d_{abs}}{\tau}$$

$\tau \approx 12$ ps
in 0.18 μm
technology

Delay of logic gate has two components:



$$d = f + p$$

effort delay, a.k.a. stage effort

parasitic delay

Effort delay again has two components:

$$f = gh$$

logical effort

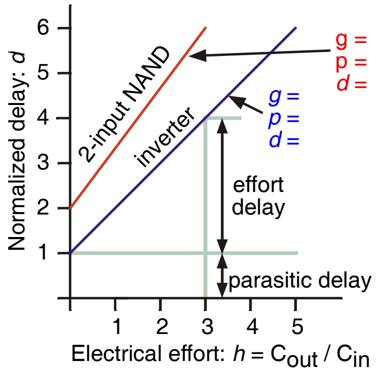
electrical effort = C_{out}/C_{in}

electrical effort
is sometimes
called *"fanout"*

□ Logical effort describes relative ability of gate topology to deliver current (defined to be 1 for an inverter)

□ Electrical effort is the ratio of output to input capacitance

Delay Plots



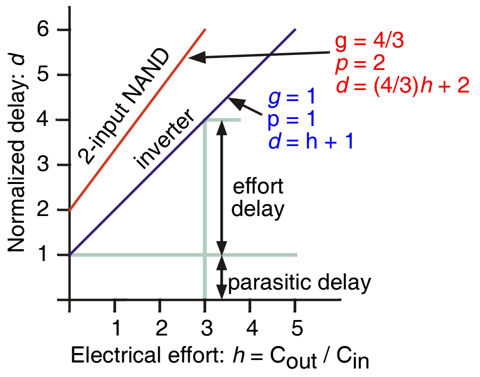
How about a
2-input NOR?

□ $d = f + p = gh + p$

□ Delay increases with electrical effort

□ More complex gates have greater logical effort and parasitic delay

Delay Plots



□ $d = f + p = gh + p$

□ Delay increases with electrical effort

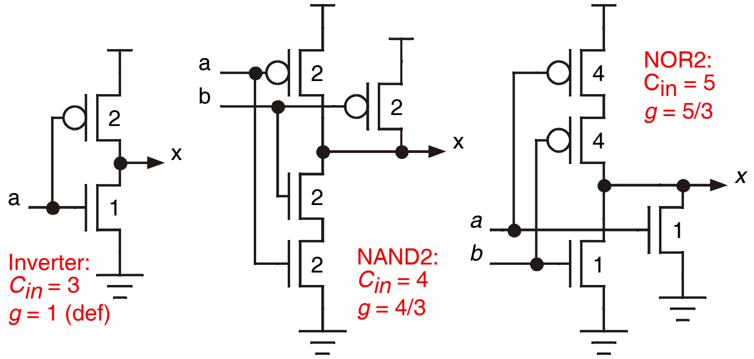
□ More complex gates have greater logical effort and parasitic delay

Computing Logical Effort

DEF: Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current.

□ Measured from delay *vs.* fanout plots of simulated or measured gates

□ Or estimated, counting capacitance in units of transistor width:



Inverter:
$C_{in} = 3$
$g = 1$ (def)

NAND2:
$C_{in} = 4$
$g = 4/3$

NOR2:
$C_{in} = 5$
$g = 5/3$

A Catalog of Gates

Table 1: Logical effort of static CMOS gates

Gate type	Number of inputs					
	1	2	3	4	5	*n*
inverter	1					
NAND		4/3	5/3	6/3	7/3	(*n*+2)/3
NOR		5/3	7/3	9/3	11/3	(2*n*+1)/3
multiplexer		2	2	2	2	2
XOR, XNOR		4	12	32		

Table 2: Parasitic delay of static CMOS gates

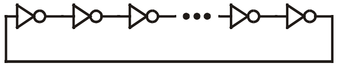
Gate type	Parasitic delay
inverter	p_{inv}
n-input NAND	np_{inv}
n-input NOR	np_{inv}
n-way multiplexer	$2np_{inv}$
2-input XOR, XNOR	$4np_{inv}$

$p_{inv} \approx 1$

parasitic delays depend on diffusion capacitance

Example

Estimate the frequency of an *N*-stage ring oscillator:



Logical Effort: g =

Electrical Effort: h =

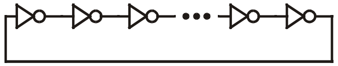
Parasitic Delay: p =

Stage Delay: d =

Oscillator Frequency: F =

Example

Estimate the frequency of an *N*-stage ring oscillator:



Logical Effort: $g \equiv 1$

Electrical Effort: $h = \frac{C_{out}}{C_{in}} = 1$

Parasitic Delay: $p = p_{inv} \approx 1$

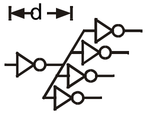
Stage Delay: $d = gh + p = 2$

Oscillator Frequency: $F = \frac{1}{2Nd_{abs}} = \boxed{\frac{1}{4N\tau}}$

A 31 stage ring oscillator in a 0.18 µm process oscillates at about 670 MHz.

Example

Estimate the delay of a fanout-of-4 (FO4) inverter:



Logical Effort: $g =$

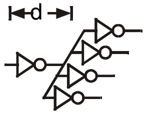
Electrical Effort: $h =$

Parasitic Delay: $p =$

Stage Delay: $d =$

Example

Estimate the delay of a fanout-of-4 (FO4) inverter:



Logical Effort: $g \equiv 1$

Electrical Effort: $h = \frac{C_{out}}{C_{in}} = 4$

Parasitic Delay: $p = p_{inv} \approx 1$

Stage Delay: $d = gh + p = \boxed{5}$

The FO4 inverter delay is a useful metric to characterize process performance.

1 FO4 delay = 5τ

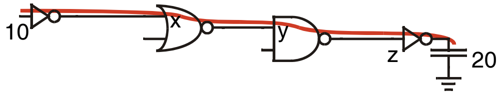
This is about 60 ps in a 0.18 µm process.

Outline

- □ Introduction
- □ Delay in a Logic Gate
- □ **Multi-stage Logic Networks**
- □ Choosing the Best Number of Stages
- □ Example
- □ Asymmetric & Skewed Logic Gates
- □ Circuit Families
- □ Summary

Multi-stage Logic Networks

Logical effort extends to multi-stage networks:



$g_1 = 1$ $h_1 = x/10$

$g_2 = 5/3$ $h_2 = y/x$

$g_3 = 4/3$ $h_3 = z/y$

$g_4 = 1$ $h_4 = 20/z$

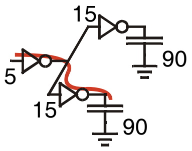
- □ Path Logical Effort: $G = \prod g_i$
- □ Path Electrical Effort: $H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
- □ Path Effort: $F = \prod f_i = \prod g_i h_i$

Don't define $H = \prod h_i$ because we don't know h_i until the design is done

Can we write $F = GH$?

Branching Effort

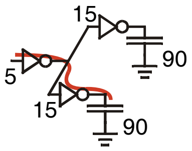
No! Consider circuits that branch:



G =
H =
GH =
h_1 =
h_2 =
F = = GH?

Branching Effort

No! Consider circuits that branch:



G = 1
H = 90 / 5 = 18
GH = 18
h_1 = (15+15) / 5 = 6
h_2 = 90 / 15 = 6
F = 36, not 18!

Introduce new kind of effort to account for branching within a network:

- □ Branching Effort: $b = \frac{C_{on\ path} + C_{off\ path}}{C_{on\ path}}$
- □ Path Branching Effort: $B = \prod b_i$

Note:

$$\prod h_i = BH \neq H$$

in circuits that branch

Now we can compute the path effort:

- □ Path Effort: $F = GBH$

Delay in Multi-stage Networks

We can now compute the delay of a multi-stage network:

- □ Path Effort Delay: $D_F = \sum f_i$
- □ Path Parasitic Delay: $P = \sum p_i$
- □ Path Delay: $D = \sum d_i = D_F + P$

We can prove that delay is minimized when each stage bears the same effort:

$$\hat{f} = g_i h_i = F^{1/N}$$

Therefore, the minimum delay of an *N*-stage path is:

$$\boxed{NF^{1/N} + P}$$

- □ This is a key result of logical effort. Lowest possible path delay can be found without even calculating the sizes of each gate in the path.

Determining Gate Sizes

Gate sizes can be found by starting at the end of the path and working backward.

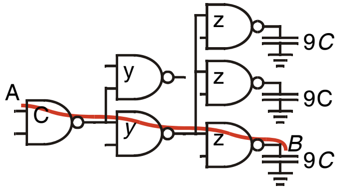
- □ At each gate, apply the capacitance transformation:

$$C_{in_i} = \frac{C_{out_i} \bullet g_i}{\hat{f}}$$

- □ Check your work by verifying that the input capacitance specification is satisfied at the beginning of the path.

Example

Select gate sizes *y* and *z* to minimize delay from *A* to *B*



Logical Effort: $G =$

Electrical Effort: $H =$

Branching Effort: $B =$

Path Effort: $F =$

Best Stage Effort: $\hat{f} =$

Delay: $D =$

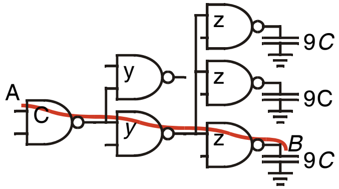
Work backward for sizes:

$z =$

$y =$

Example

Select gate sizes *y* and *z* to minimize delay from *A* to *B*



Logical Effort: $G = (4/3)^3$

Electrical Effort: $H = \frac{C_{out}}{C_{in}} = 9$

Branching Effort: $B = 2 \bullet 3 = 6$

Path Effort: $F = GHB = 128$

Best Stage Effort: $\hat{f} = F^{1/3} \approx 5$

Delay: $D = 3 \bullet 5 + 3 \bullet 2 = 21$

Work backward for sizes:

$$z = \frac{9C \bullet (4/3)}{5} = 2.4C$$

$$y = \frac{3z \bullet (4/3)}{5} = 1.92C$$

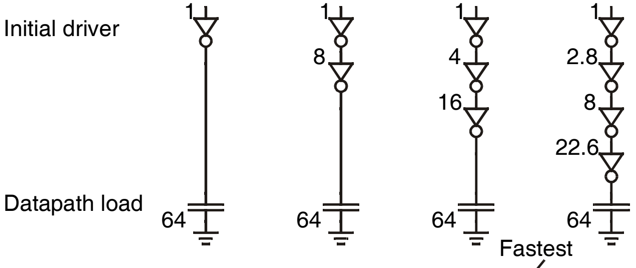
Outline

- □ Introduction
- □ Delay in a Logic Gate
- □ Multi-stage Logic Networks
- □ Choosing the Best Number of Stages
- □ Example
- □ Asymmetric & Skewed Logic Gates
- □ Circuit Families
- □ Summary

Choosing the Best Number of Stages

How many stages should a path use?

- □ Delay is not always minimized by using as few stages as possible
- □ Example: How to drive 64 bit datapath with unit-sized inverter



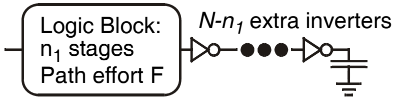
N:	1	2	3	4
f:	64	8	4	2.8
D:	65	18	$\boxed{15}$	15.3

$D = NF^{1/N} + P = N(64)^{1/N} + N$ assuming polarity doesn't matter

Derivation of the Best Number of Stages

Suppose we can add inverters to the end of a path without changing its function.

□ How many stages should we use? Let $\hat{N}$ be the value of N for least delay.



$$D = NF^{1/N} + \sum_{1}^{n_1} p_i + (N - n_1)p_{inv}$$

$$\frac{\partial D}{\partial N} = -F^{1/N}\ln(F^{1/N}) + F^{1/N} + p_{inv} = 0$$

□ Define $\rho \equiv F^{1/\hat{N}}$ to be the best stage effort. Substitute and simplify:

$$p_{inv} + \rho(1 - \ln\rho) = 0$$

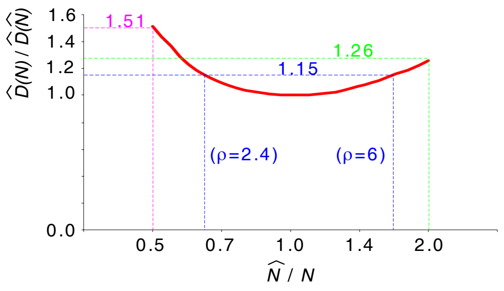
Best Number of Stages (continued)

$p_{inv} + \rho(1 - \ln\rho) = 0$ has no closed form solution.

□ Neglecting parasitics (*i.e.* $p_{inv} = 0$), we get the familiar result that $\rho = 2.718$ (*e*)

□ For $p_{inv} = 1$, we can solve numerically to obtain $\rho = 3.59$

How sensitive is the delay to using exactly the best number of stages?



I like to use
ρ = 4

□ 2.4 < ρ < 6 gives delays within 15% of optimal -> we can be sloppy

Outline

□ Introduction

□ Delay in a Logic Gate

□ Multi-stage Logic Networks

□ Choosing the Best Number of Stages

□ **Example**

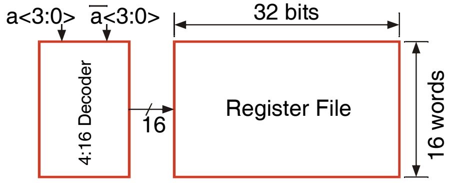
□ Asymmetric & Skewed Logic Gates

□ Circuit Families

□ Summary

Example

Let's revisit Ben Bitdiddle's decoder problem using logical effort:



Decoder specification:

□ 16 word register file

□ Each word is 32 bits wide

□ Each bit presents a load of 3 unit-sized transistors

□ True and complementary inputs of address bits *a*<3:0> are available

□ Each input may drive 10 unit-sized transistors

Ben needs to decide:

□ How many stages to use?

□ How large should each gate be?

□ How fast can the decoder operate?

Example: Number of Stages

How many stages should Ben use?

□ Effort of decoders is dominated by electrical and branching portions

□ Electrical Effort: $H =$

□ Branching Effort: $B =$

If we neglect logical effort (assume G = 1),

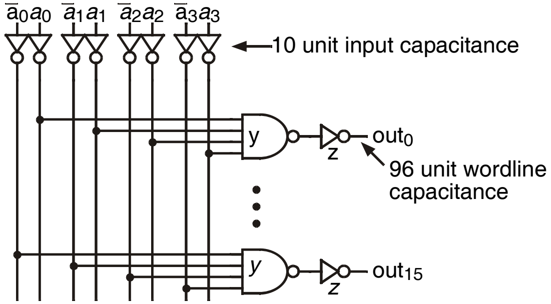
□ Path Effort: $F =$

Remember that the best stage effort is about $\rho = 4$

□ Hence, the best number of stages is: $N =$

Example: Gate Sizes & Delay

Lets try a 3-stage design using 16 4-input NAND gates with $G =$



$\bar{a}_0 a_0$ $\bar{a}_1 a_1$ $\bar{a}_2 a_2$ $\bar{a}_3 a_3$
10 unit input capacitance
y
z
out$_0$
96 unit wordline capacitance
y
z
out$_{15}$

□ Actual path effort is: $F =$

□ Therefore, stage effort should be: $f =$

□ Gate sizes: $z =$ $y =$

□ Path delay: $D =$

Example: Number of Stages

How many stages should Ben use?

□ Effort of decoders is dominated by electrical and branching portions

□ Electrical Effort: $H = \frac{32 \bullet 3}{10} = 9.6$

□ Branching Effort: $B = 8$ because each address input controls half the outputs

If we neglect logical effort,

□ Path Effort: $F = GBH = 8 \bullet 9.6 = 76.8$

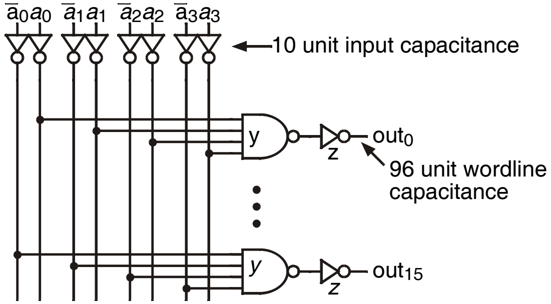
Remember that the best stage effort is about $\rho = 4$

□ Hence, the best number of stages is: $N = \log_4 76.8 = 3.1$

□ Let's try a 3-stage design

Example: Gate Sizes & Delay

Lets try a 3-stage design using 16 4-input NAND gates with $G = 1 \bullet 2 \bullet 1 = 2$



$\bar{a}_0 a_0$ $\bar{a}_1 a_1$ $\bar{a}_2 a_2$ $\bar{a}_3 a_3$
10 unit input capacitance
y
z
out$_0$
96 unit wordline capacitance
y
z
out$_{15}$

□ Actual path effort is: $F = 2 \bullet 8 \bullet 9.6 = 154$

□ Therefore, stage effort should be: $f = (154)^{1/3} = 5.36$

Close to 4, so f is reasonable

□ $z = 96 \bullet 1/5.36 = 18$ $y = 18 \bullet 2/5.36 = 6.7$

□ $D = 3f + P = 3 \bullet 5.36 + 1 + 4 + 1 = 22.1$

Example: Alternative Decoders

Table 3: Comparison of Decoder Designs

Design	Stages	G	P	D
NAND4; INV	2	2	5	29.8
INV; NAND4; INV	**3**	**2**	**6**	**22.1**
INV; NAND4; INV; INV	4	2	7	21.1
NAND2; INV; NAND2; INV	4	16/9	6	19.7
INV; NAND2; INV; NAND2; INV	5	16/9	7	20.4
NAND2; INV; NAND2; INV; INV; INV	6	16/9	8	21.6
INV; NAND2; INV; NAND2; INV; INV; INV	7	16/9	9	23.1
NAND2; INV; NAND2; INV; INV; INV; INV; INV	8	16/9	10	24.8

We underestimated the best number of stages by neglecting the logical effort.

□ Logical effort facilitates comparing different designs before selecting sizes

□ Using more stages also reduces G and P by using multiple 2-input gates

□ Our design was about 10% slower than the best

Outline

□ Introduction

□ Delay in a Logic Gate

□ Multi-stage Logic Networks

□ Choosing the Best Number of Stages

□ Example

□ **Asymmetric & Skewed Logic Gates**

□ Circuit Families

□ Summary

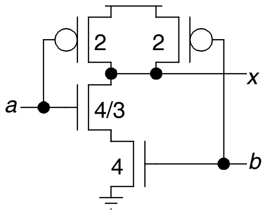
Asymmetric Gates

Asymmetric logic gates favor one input over another.

Example: suppose input A of a NAND gate is most critical.

□ Select sizes so pullup and pulldown still match unit inverter

□ Place critical input closest to output



□ Logical Effort on input A: $g_A =$

□ Logical Effort on input B: $g_B =$

□ Total Logical Effort: $g_{tot} = g_A + g_B$

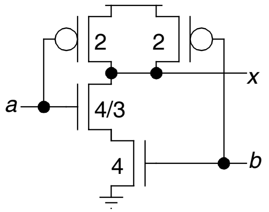
Asymmetric Gates

Asymmetric logic gates favor one input over another.

Example: Suppose input A of a NAND gate is most critical:

□ Select sizes so pullup and pulldown still match unit inverter

□ Place critical input closest to output



□ Logical Effort on input A: $g_A = 10/9$

□ Logical Effort on input B: $g_B = 2$

□ Total Logical Effort: $g_{tot} = g_A + g_B = 28/9$

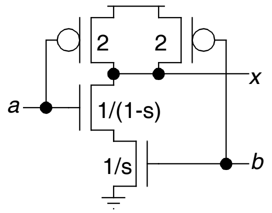
Effort on A goes down at expense of effort on B and total gate effort

Symmetry Factor

In general, consider gates with arbitrary symmetry factor s:

□ s = 1/2 in symmetric gate with equal sizes

□ s = 1/4 in previous example



Logical effort of inputs:

$$g_A = \frac{\frac{1}{1-s}+2}{3} \qquad g_B = \frac{\frac{1}{s}+2}{3} \qquad g_{tot} = \frac{\frac{1}{s(1-s)}+4}{3}$$

□ Critical input approaches logical effort of inverter = 1 for small s

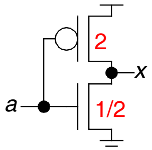
□ But total logical effort is higher for asymmetric gates

Skewed Gates

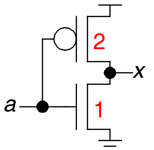
Skewed gates favor one edge over the other.

Example: suppose rising output of inverter is most critical.

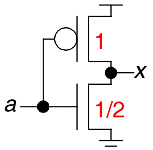
□ Downsize noncritical NMOS transistor to reduce total input capacitance



HI-Skewed inverter



Unskewed w/ equal rise



Unskewed w/ equal fall

Compare with unskewed inverter of the same rise/fall time to compute effort.

□ Logical Effort for rising (up) output: $g_u =$

□ Logical Effort for falling (down) output: $g_d =$

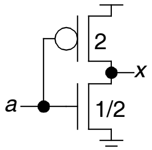
□ Average Logical Effort: $g_{avg} = (g_u + g_d)/2$

Skewed Gates

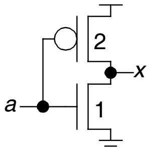
Skewed gates favor one edge over the other.

Example: suppose rising output of inverter is most important.

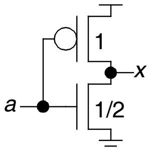
□ Downsize noncritical NMOS transistor to reduce total input capacitance



Skewed inverter



Unskewed w/ equal rise



Unskewed w/ equal fall

Compare with unskewed inverter of the same rise/fall time

□ Logical Effort for rising (up) output: $g_u = 5/6$

□ Logical Effort for falling (down) output: $g_d = 5/3$

□ Average Logical Effort: $g_{avg} = (g_u + g_d)/2 = 5/4$

Critical rising effort goes down at expense of noncritical and average effort

HI- and LO-Skewed Gates

DEF: Logical effort of a skewed gate for a particular transition is the ratio of the input capacitance of that gate to the input capacitance of an unskewed inverter delivering the same output current for the same transition.

Skew gates by reducing size of noncritical transistors.

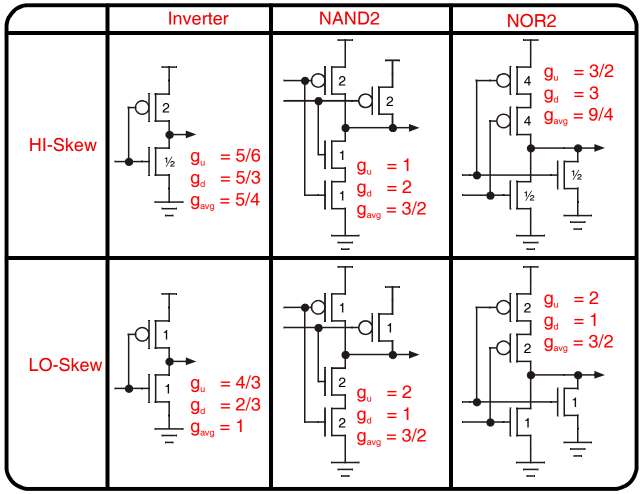
□ HI-Skewed gates favor rising outputs by downsizing NMOS transistors

□ LO-Skewed gates favor falling outputs by downsizing PMOS transistors

□ Logical effort is smaller for the favored input due to lower input capacitance

□ Logical effort is larger for the other input

Catalog of Skewed Gates



Outline

- □ Introduction
- □ Delay in a Logic Gate
- □ Multi-stage Logic Networks
- □ Choosing the Best Number of Stages
- □ Example
- □ Asymmetric & Skewed Logic Gates
- □ **Circuit Families**
- □ Summary

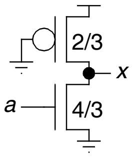
Pseudo-NMOS

Pseudo-NMOS gates replace fat PMOS pullups on inputs with a resistive pullup.

- □ Resistive pullup must be much weaker than pulldown stack (e.g. 4x)
- □ Reduces logical effort because inputs must only drive the NMOS transistors
- □ However, NMOS current reduced by contention with pullup
- □ Unequal rising and falling efforts
- □ Quiescent power dissipation when output is low

Example: Pseudo-NMOS inverter

- □ Logical Effort for falling (down) output: $g_d =$
- □ Logical Effort for rising (up) output: $g_u =$
- □ Average Logical Effort: $g_{avg} = (g_u + g_d)/2$



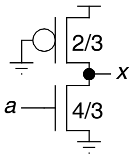
Pseudo-NMOS

Pseudo-NMOS gates replace fat PMOS pullups on inputs with a resistive pullup.

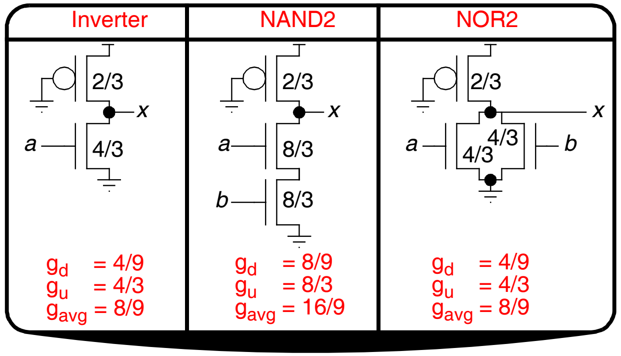
- □ Resistive pullup must be much weaker than pulldown stack (e.g. 4x)
- □ Reduces logical effort because inputs must only drive the NMOS transistors
- □ However, NMOS current reduced by contention with pullup
- □ Unequal rising and falling efforts
- □ Logical effort can be applied to domino, pseudo-NMOS, and other logic families

Example: Pseudo-NMOS inverter

- □ Logical Effort for falling (down) output: $g_d = 4/9$
- □ Logical Effort for rising (up) output: $g_u = 4/3$
- □ Average Logical Effort: $g_{avg} = (g_u + g_d)/2 = 8/9$



Pseudo-NMOS Gates



Inverter	NAND2	NOR2
g_d = 4/9 g_u = 4/3 g_{avg} = 8/9	g_d = 8/9 g_u = 8/3 g_{avg} = 16/9	g_d = 4/9 g_u = 4/3 g_{avg} = 8/9

Tradeoffs exist between power and effort by varying P/N ratio.

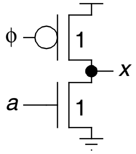
Dynamic Logic

Dynamic logic replace fat PMOS pullups on inputs with a clocked precharge.

- □ Reduces logical effort because inputs must only drive the NMOS transistors
- □ Eliminates pseudo-NMOS contention current and power dissipation
- □ Only the falling (“evaluation”) delay is critical
- □ Downsize noncritical precharge transistors to reduce clock load and power

Example: Footless dynamic inverter

- □ Logical Effort for falling (down) output: $g_d =$



Robust gates may require keepers and clocked pulldown transistors (“feet”).

- □ Feet prevent contention during precharge but increase logical effort
- □ Weak keepers prevent floating output at cost of slight contention during eval

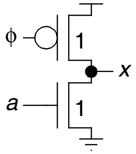
Dynamic Logic

Dynamic logic replace fat PMOS pullups on inputs with a clocked precharge.

- □ Reduces logical effort because inputs must only drive the NMOS transistors
- □ Eliminates pseudo-NMOS contention current and power dissipation
- □ Critical pulldown (“evaluation”) delay independent of precharge size

Example: Footless dynamic inverter

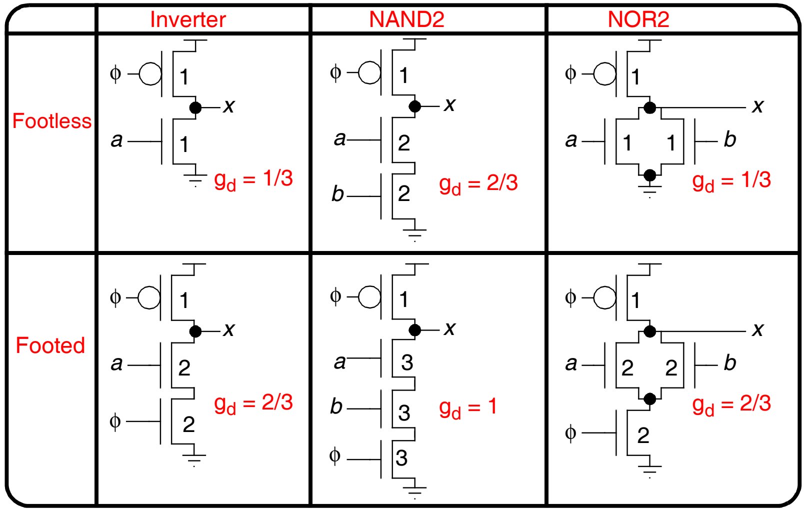
- □ Logical Effort for falling (down) output: $g_d = 1/3$



Robust gates may require keepers and clocked pulldown transistors (“feet”).

- □ Feet prevent contention during precharge but increase logical effort
- □ Weak keepers prevent floating output at cost of slight contention during eval

Dynamic Gates



	Inverter	NAND2	NOR2
Footless	g_d = 1/3	g_d = 2/3	g_d = 1/3
Footed	g_d = 2/3	g_d = 1	g_d = 2/3

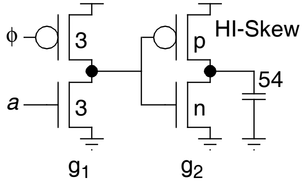
Domino Gates

Dynamic gates require monotonically rising inputs.

□ However, they generate monotonically falling outputs
□ Alternate dynamic gates with HI-skew inverting static gates
□ Dynamic / static pair is called a domino gate

Example: Domino Buffer

□ Constraints: maximum input capacitance = 3, load = 54
□ Logical Effort: G =
□ Branching Effort: B =
□ Electrical Effort: H =
□ Path Effort: F =
□ Stage Effort: f =
□ HI-Skew Inverter: size =
□ Transistor Sizes: n = p =



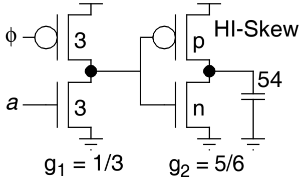
Domino Gates

Dynamic gates require monotonically rising inputs.

□ However, they generate monotonically falling outputs
□ Alternate dynamic gates with HI-skew inverting static gates
□ Dynamic / static pair is called a domino gate

Example: Domino Buffer

□ Constraints: maximum input capacitance = 3, load = 54
□ Logical Effort: G = (1/3) * (5/6) = 5/18
□ Branching Effort: B = 1
□ Electrical Effort: H = 54/3 = 18
□ Path Effort: F = (5/18) * 1 * 18 = 5
□ Stage Effort: f = $\sqrt{5}$ = *2.2*
□ HI-Skew Inverter: size =54 * (5/6) / 2.2 = 20
□ Transistor Sizes: n = 4 p = 16



Comparison of Circuit Families

Assumptions:

□ PMOS transistors have half the drive of NMOS transistors
□ Skewed gates downsize noncritical transistors by factor of two
□ Pseudo-NMOS gates have 1/4 strength pullups

Table 4: Summary of Logical Efforts

Circuit Style	Inverter g		n-input NAND g		n-input NOR g	
	g_u	g_d	g_u	g_d	g_u	g_d
Static CMOS	1		(n+2)/3		(2n+1)/3	
HI-Skew	5/6	5/3	(n/2+2)/3	(n+4)/3	(2n+.5)/3	(4n+1)/3
LO-Skew	4/3	2/3	2(n+1)/3	(n+1)/3	2(n+1)/3	(n+1)/3
Pseudo-NMOS	4/3	4/9	4n/3	4n/9	4/3	4/9
Footed Dynamic	2/3		(n+1)/3		2/3	
Footless Dynamic	1/3		n/3		1/3	

Adjust these numbers as you change your assumptions.

Outline

□ Introduction
□ Delay in a Logic Gate
□ Multi-stage Logic Networks
□ Choosing the Best Number of Stages
□ Example
□ Asymmetric & Skewed Logic Gates
□ Circuit Families
□ Summary

Summary

Table 5: Key Definitions of Logical Effort

Term	Stage expression	Path expression
Logical effort	g (seeTable 1)	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
Branching effort	n/a	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	f	$D_F = \sum f_i$
Number of stages	1	N
Parasitic delay	p (seeTable 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

Method of Logical Effort

Logical effort helps you find the best number of stages, the best size of each gate, and the minimum delay of a circuit with the following procedure:

- □ Compute the path effort: $F = GBH$
- □ Estimate the best number of stages: $\hat{N} \approx \log_4 F$
- □ Estimate the minimum delay: $D = \hat{N}F^{1/\hat{N}} + P$
- □ Sketch your path using the number of stages computed above
- □ Compute the stage effort: $\hat{f} = F^{1/N}$
- □ Starting at the end, work backward to find transistor sizes:

$$C_{in_i} = \frac{C_{out_i} \bullet g_i}{\hat{f}}$$

Limitations of Logical Effort

Logical effort is not a panacea. Some limitations include:

- □ **Chicken & egg problem**
  how to estimate G and best number of stages before the path is designed
- □ **Simplistic delay model**
  neglects effects of input slopes
- □ **Interconnect**
  iteration required in designs with branching and non-negligible wire C or RC
  same convergence difficulties as in synthesis / placement problem
- □ **Maximum speed only**
  optimizes circuits for speed, not area or power under a fixed speed constraint

Conclusion

Logical effort is a useful concept for thinking about delay in circuits:

- □ Facilitates comparison of different circuit topologies
- □ Easily select gate sizes for minimum delay
- □ Circuits are fastest when effort delays of each stage are equal and about 4
- □ Path delay is insensitive to modest deviations from optimal sizes
- □ Logic gates can be skewed to favor one input or edge at the cost of another
- □ Logical effort can be applied to domino, pseudo-NMOS, and other logic families

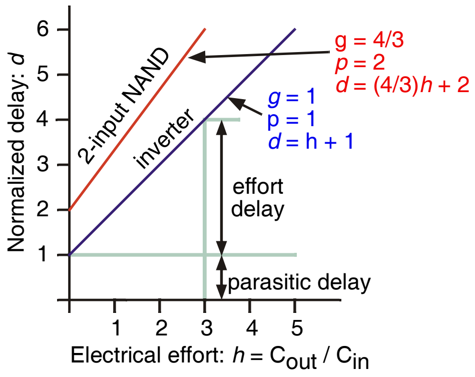
Logical effort provides a language for engineers to discuss why circuits are fast.

- □ Like any language, requires practice to master

A book on Logical Effort is available from Morgan Kaufmann Publishers

- □ `http://www.mkp.com/Logical_Effort`
- □ Discusses P/N ratios, gate characterization, pass gate logic, forks, wires, etc.

Delay Plots



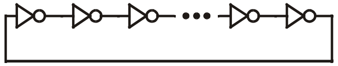
□ $d = f + p = gh + p$

□ Delay increases with electrical effort

□ More complex gates have greater logical effort and parasitic delay

Example

Estimate the frequency of an *N*-stage ring oscillator:



Logical Effort: $g \equiv 1$

Electrical Effort: $h = \frac{C_{out}}{C_{in}} = 1$

Parasitic Delay: $p = p_{inv} \approx 1$

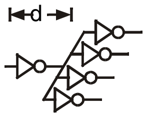
Stage Delay: $d = gh + p = 2$

Oscillator Frequency: $F = \frac{1}{2Nd_{abs}} = \boxed{\frac{1}{4N\tau}}$

A 31 stage ring oscillator in a 0.18 µm process oscillates at about 670 MHz.

Example

Estimate the delay of a fanout-of-4 (FO4) inverter:



Logical Effort: $g \equiv 1$

Electrical Effort: $h = \frac{C_{out}}{C_{in}} = 4$

Parasitic Delay: $p = p_{inv} \approx 1$

Stage Delay: $d = gh + p = \boxed{5}$

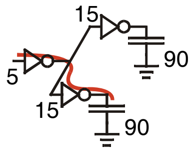
The FO4 inverter delay is a useful metric to characterize process performance.

1 FO4 delay = 5τ

This is about 60 ps in a 0.18 µm process.

Branching Effort

No! Consider circuits that branch:



G = 1
H = 90 / 5 = 18
GH = 18
h_1 = (15+15) / 5 = 6
h_2 = 90 / 15 = 6
F = 36, not 18!

Introduce new kind of effort to account for branching within a network:

□ Branching Effort: $b = \frac{C_{on\ path} + C_{off\ path}}{C_{on\ path}}$

□ Path Branching Effort: $B = \prod b_i$

Note:

$\prod h_i = BH \neq H$

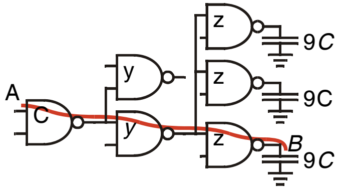
in circuits that branch

Now we can compute the path effort:

□ Path Effort: $F = GBH$

Example

Select gate sizes *y* and *z* to minimize delay from *A* to *B*



Logical Effort: $G = (4/3)^3$

Electrical Effort: $H = \frac{C_{out}}{C_{in}} = 9$

Branching Effort: $B = 2 \bullet 3 = 6$

Path Effort: $F = GHB = 128$

Best Stage Effort: $\hat{f} = F^{1/3} \approx 5$

Delay: $D = 3 \bullet 5 + 3 \bullet 2 = 21$

Work backward for sizes:

$$z = \frac{9C \bullet (4/3)}{5} = 2.4C$$

$$y = \frac{3z \bullet (4/3)}{5} = 1.92C$$

Example: Number of Stages

How many stages should Ben use?

- □ Effort of decoders is dominated by electrical and branching portions
- □ Electrical Effort: $H = \frac{32 \bullet 3}{10} = 9.6$
- □ Branching Effort: $B = 8$ because each address input controls half the outputs

If we neglect logical effort,

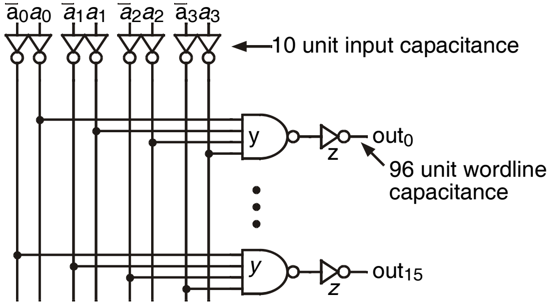
- □ Path Effort: $F = GBH = 8 \bullet 9.6 = 76.8$

Remember that the best stage effort is about $\rho = 4$

- □ Hence, the best number of stages is: $N = \log_4 76.8 = 3.1$
- □ Let's try a 3-stage design

Example: Gate Sizes & Delay

Lets try a 3-stage design using 16 4-input NAND gates with $G = 1 \bullet 2 \bullet 1 = 2$



10 unit input capacitance

96 unit wordline capacitance

- □ Actual path effort is: $F = 2 \bullet 8 \bullet 9.6 = 154$
- □ Therefore, stage effort should be: $f = (154)^{1/3} = 5.36$
- □ $z = 96 \bullet 1/5.36 = 18$ $y = 18 \bullet 2/5.36 = 6.7$
- □ $D = 3f + P = 3 \bullet 5.36 + 1 + 4 + 1 = 22.1$

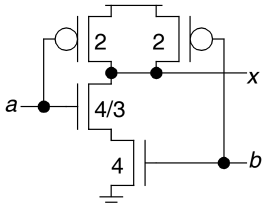
Close to 4, so *f* is reasonable

Asymmetric Gates

Asymmetric logic gates favor one input over another.

Example: Suppose input A of a NAND gate is most critical:

- □ Select sizes so pullup and pulldown still match unit inverter
- □ Place critical input closest to output



- □ Logical Effort on input A: $g_A = 10/9$
- □ Logical Effort on input B: $g_B = 2$
- □ Total Logical Effort: $g_{tot} = g_A + g_B = 28/9$

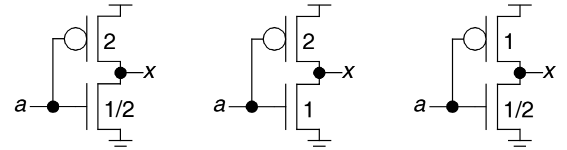
Effort on A goes down at expense of effort on B and total gate effort

Skewed Gates

Skewed gates favor one edge over the other.

Example: suppose rising output of inverter is most important.

- □ Downsize noncritical NMOS transistor to reduce total input capacitance



Skewed inverter

Unskewed w/ equal rise

Unskewed w/ equal fall

Compare with unskewed inverter of the same rise/fall time

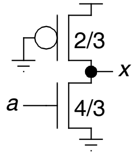
- □ Logical Effort for rising (up) output: $g_u = 5/6$
- □ Logical Effort for falling (down) output: $g_d = 5/3$
- □ Average Logical Effort: $g_{avg} = (g_u + g_d)/2 = 5/4$

Critical rising effort goes down at expense of noncritical and average effort

Pseudo-NMOS

Pseudo-NMOS gates replace fat PMOS pullups on inputs with a resistive pullup.

- □ Resistive pullup must be much weaker than pulldown stack (e.g. 4x)
- □ Reduces logical effort because inputs must only drive the NMOS transistors
- □ However, NMOS current reduced by contention with pullup
- □ Unequal rising and falling efforts
- □ Logical effort can be applied to domino, pseudo-NMOS, and other logic families



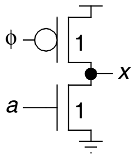
Example: Pseudo-NMOS inverter

- □ Logical Effort for falling (down) output: $g_d = 4/9$
- □ Logical Effort for rising (up) output: $g_u = 4/3$
- □ Average Logical Effort: $g_{avg} = (g_u + g_d)/2 = 8/9$

Dynamic Logic

Dynamic logic replace fat PMOS pullups on inputs with a clocked precharge.

- □ Reduces logical effort because inputs must only drive the NMOS transistors
- □ Eliminates pseudo-NMOS contention current and power dissipation
- □ Critical pulldown (“evaluation”) delay independent of precharge size



Example: Footless dynamic inverter

- □ Logical Effort for falling (down) output: $g_d = 1/3$

Robust gates may require keepers and clocked pulldown transistors (“feet”).

- □ Feet prevent contention during precharge but increase logical effort
- □ Weak keepers prevent floating output at cost of slight contention during eval

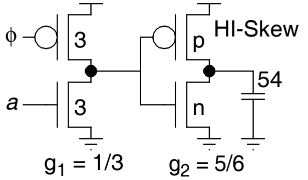
Domino Gates

Dynamic gates require monotonically rising inputs.

- □ However, they generate monotonically falling outputs
- □ Alternate dynamic gates with HI-skew inverting static gates
- □ Dynamic / static pair is called a domino gate

Example: Domino Buffer

- □ Constraints: maximum input capacitance = 3, load = 54
- □ Logical Effort: G = (1/3) * (5/6) = 5/18
- □ Branching Effort: B = 1
- □ Electrical Effort: H = 54/3 = 18
- □ Path Effort: F = (5/18) * 1 * 18 = 5
- □ Stage Effort: f = $\sqrt{5} = 2.2$
- □ HI-Skew Inverter: size =54 * (5/6) / 2.2 = 20
- □ Transistor Sizes: n = 4 p = 16



ϕ
3
a
3
p
HI-Skew
n
54
$g_1 = 1/3$
$g_2 = 5/6$