Logical Effort*:

Designing for Speed on the Back of an Envelope

David Harris

harrisd@vlsi.stanford.edu

August, 1998

Stanford University

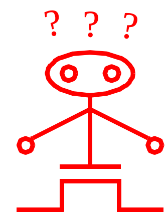
Stanford, CA

* Based on a book by Ivan Sutherland, Bob Sproull, and David Harris

Outline

- **Introduction**
- Delay in a Logic Gate
- Multi-stage Logic Networks
- Choosing the Best Number of Stages
- Example
- 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:

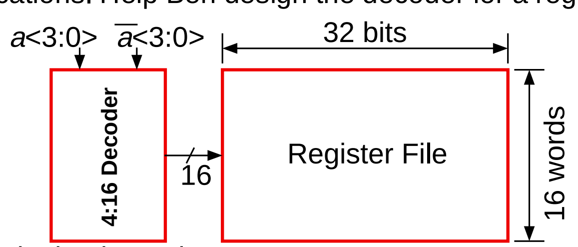
- Uses a very simple model of delay
- Back of the envelope calculations and tractable optimization
- Gives new names to old ideas to emphasize remarkable symmetries

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
- Summary

Delay in a Logic Gate

Let us express delays in a process-independent unit:

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

$\tau \approx 20$ ps
in 0.25 μm
technology

Delay of logic gate has two components:

$$d = f + p$$

f: effort delay, a.k.a. stage effort

p: parasitic delay

Effort delay again has two components:

$$f = gh$$

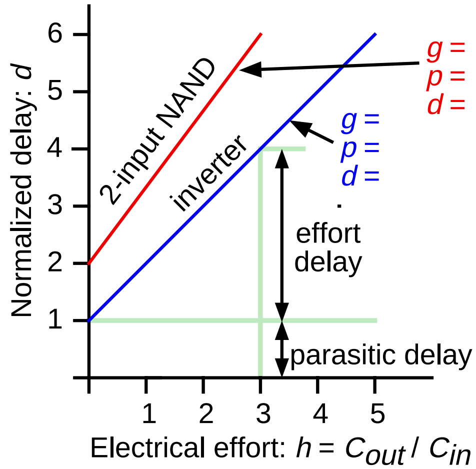
g: logical effort

h: 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



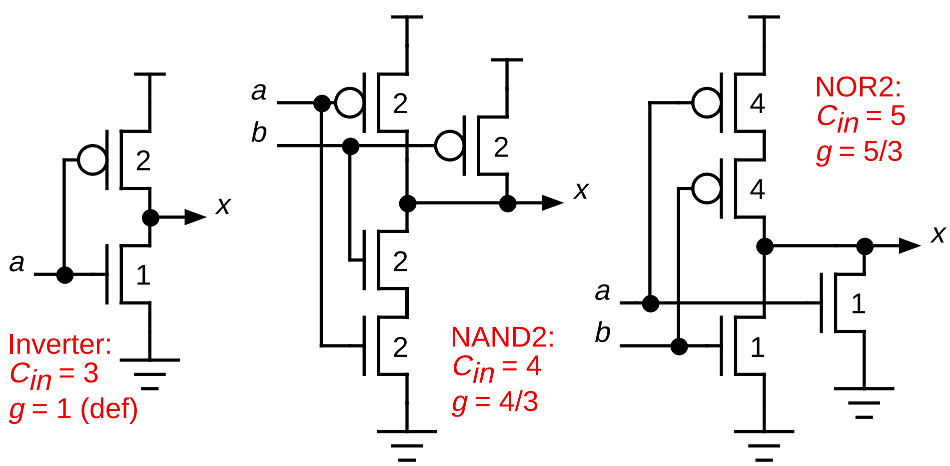
How about a 2-input NOR?

- $d = f + p = gh + p$
- Delay increases with electrical effort
- More complex gates have greater logical effort and parasitic delay

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:



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	$(2n+1)/3$
multiplexer		2	2	2	2	2
XOR, XNOR		4	12	32		

Table 2: Parasitic delay of static CMOS gates

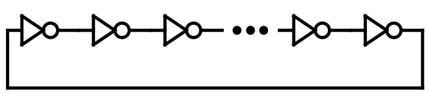
Gate type	Parasitic delay
inverter	p_{inv}
n-input NAND	np_{inv}
n-input NOR	np_{inv}
n-way multiplexer	$2np_{inv}$
2-input XOR, XNOR	$4np_{inv}$

$p_{inv} \approx 1$

parasitic delays depend on diffusion capacitance

Example

Estimate the frequency of an N-stage ring oscillator:



Logical Effort: g =

Electrical Effort: h =

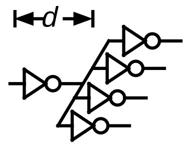
Parasitic Delay: p =

Stage Delay: d =

Oscillator Frequency: F =

Example

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



Logical Effort: g =

Electrical Effort: h =

Parasitic Delay: p =

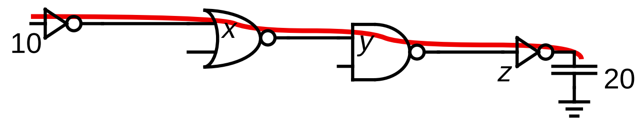
Stage Delay: d =

Outline

- Introduction
- Delay in a Logic Gate
- **Multi-stage Logic Networks**
- Choosing the Best Number of Stages
- Example
- Summary

Multi-stage Logic Networks

Logical effort extends to multi-stage networks:



$g_1 = 1$ $\quad$ $g_2 = 5/3$ $\quad$ $g_3 = 4/3$ $\quad$ $g_4 = 1$

$h_1 = x/10$ $\quad$ $h_2 = y/x$ $\quad$ $h_3 = z/y$ $\quad$ $h_4 = 20/z$

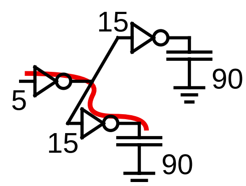
- Path Logical Effort: $G = \prod g_i$
- Path Electrical Effort: $H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
- Path Effort: $F = \prod f_i = \prod g_i h_i$

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?

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.

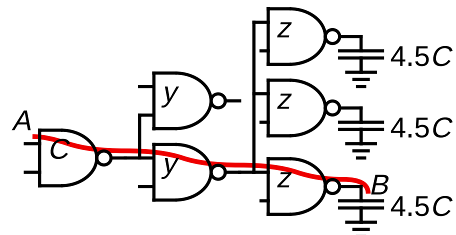
- At each gate, apply the capacitance transformation:

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

- 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 =

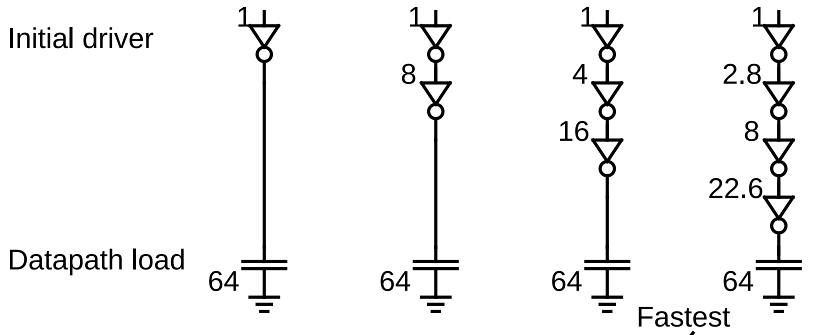
Outline

- Introduction
- Delay in a Logic Gate
- Multi-stage Logic Networks
- **Choosing the Best Number of Stages**
- Example
- 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



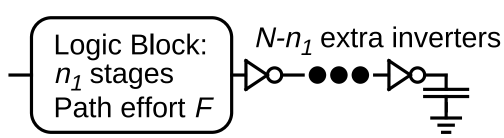
N:	1	2	3	4
f:	64	8	4	2.8
D:	65	18	15	15.3

$$D = NF^{1/N} + P = N(64)^{1/N} + N$$ 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:

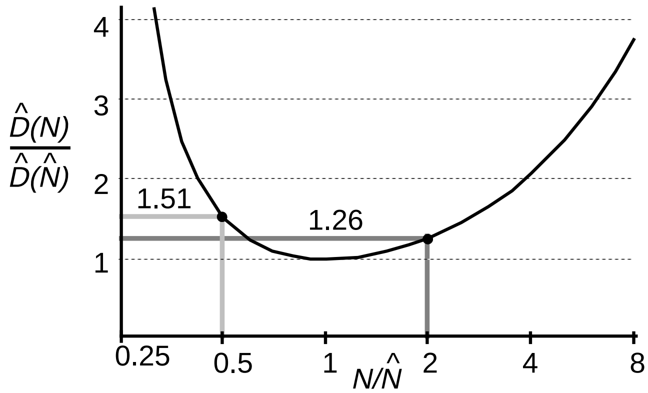
$$p_{inv} + \rho(1 - \ln\rho) = 0$$

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 $\rho = 4$

Better use too many stages than too few.

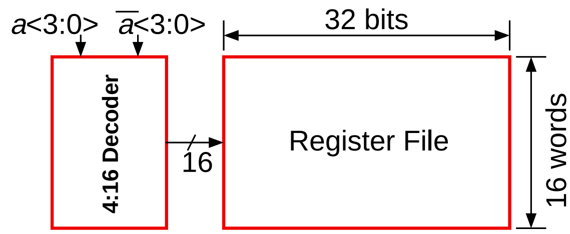
- $2.4 < \rho < 6$ gives delays within 15% of optimal -> we can be sloppy

Outline

- Introduction
- Delay in a Logic Gate
- Multi-stage Logic Networks
- Choosing the Best Number of Stages
- **Example**
- 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),

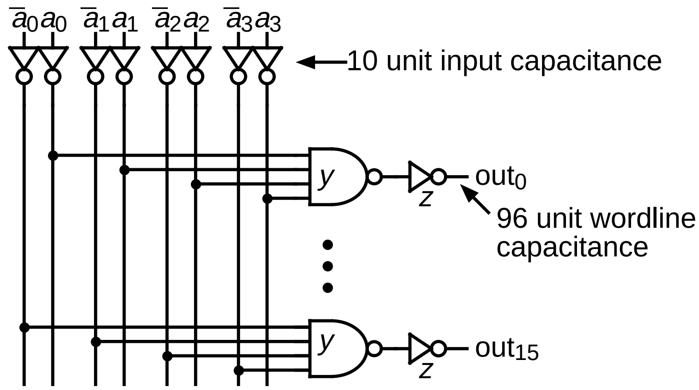
- ❑ Path Effort: F =

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 =



- ❑ Actual path effort is: F =
- ❑ Therefore, stage effort should be: f =
- ❑ Gate sizes: z = y =
- ❑ Path delay: D =

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
- **Summary**

Summary

Table 4: 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

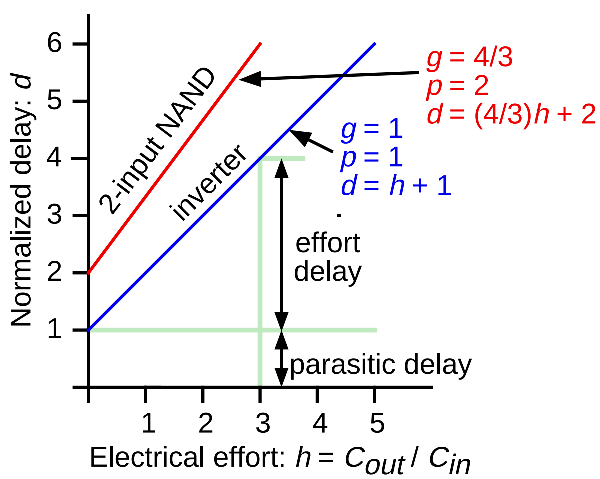
Some further results from logical effort include:

- ❑ Logical effort can be applied to domino, pass gate, and other logic families
- ❑ Logic gates can be skewed to favor one input or edge at the cost of another
- ❑ While the logical effort of a multiplexer is independent of the number of inputs, parasitic delay increases with size, so 4-way multiplexers are best
- ❑ Circuits that fork should equalize delays between legs of the fork

A book on Logical Effort will be available in Feb. 1999 from Morgan Kaufmann

`http://www.mkp.com/Logical_Effort`

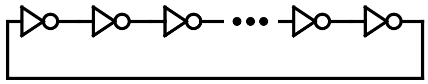
Delay Plots



- ❑ $d = f + p = gh + p$
- ❑ 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: $p = p_{inv} \approx 1$

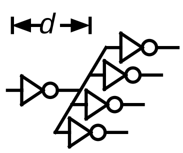
Stage Delay: $d = gh + p = 2$

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

A 31 stage ring oscillator in a 0.25 µm process oscillates at about 400 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: $d = gh + p = \boxed{5}$

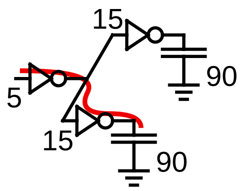
The FO4 inverter delay is a useful metric to characterize process performance.

1 FO4 delay = 5τ

This is about 100 ps in a 0.25 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: $B = \prod b_i$

Now we can compute the path effort:

- Path Effort: $F = GBH$

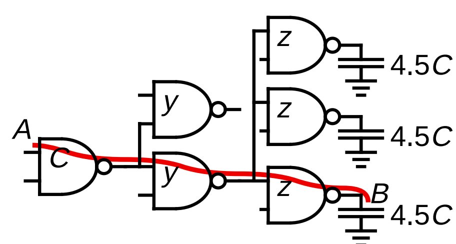
Note:

$\prod h_i = BH \neq H$

in circuits that branch

Example

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



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

Electrical Effort: $H = \frac{C_{out}}{C_{in}} = 4.5$

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

Path Effort: $F = GHB = 64$

Best Stage Effort: $\hat{f} = F^{1/3} = 4$

Delay: $D = 3 \bullet 4 + 3 \bullet 2 = 18$

Work backward for sizes:

$z = \frac{4.5C \bullet (4/3)}{4} = 1.5C$

$y = \frac{3z \bullet (4/3)}{4} = 1.5C$

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,

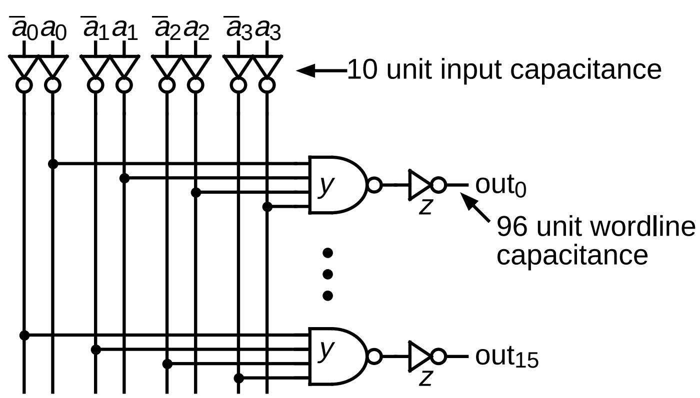
- Path Effort: $F = GBH = 8 \bullet 9.6 = 76.8$

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 $G = 1 \bullet 2 \bullet 1 = 2$



- 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
- $z = 96 \bullet 1/5.36 = 18$ $y = 18 \bullet 2/5.36 = 6.7$
- $D = 3f + P = 3 \bullet 5.36 + 1 + 4 + 1 = 22.1$