EE671: VLSI Design
Spring 2024/25

Laxmeesha Somappa

Department of Electrical Engineering

IIT Bombay

laxmeesha@ee.iitb.ac.in

LECTURE – 25
LOGICAL EFFORT

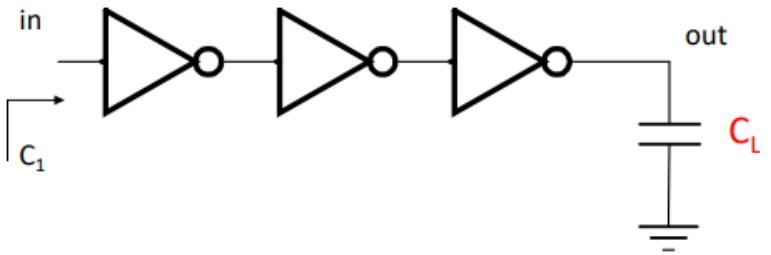
laxmeesha@ee.iitb.ac.in

TAKEAWAYS FROM LAST CLASS

- An inverter has an intrinsic delay (due to parasitic) and external delay due to the load it is driving
- If C_1 is input cap of INVX1, C_L is the load, inverter delay modelled as:

$$t_p = t_{p0}\left(1 + \frac{C_L}{\gamma C_1}\right) \text{ and } t_{p0} = 0.69\, R\, (\gamma C_1)$$

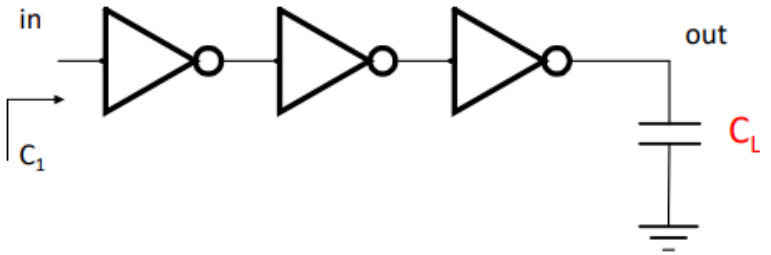
- Observations:
  - Total drive delay can be reduced by adding inverters



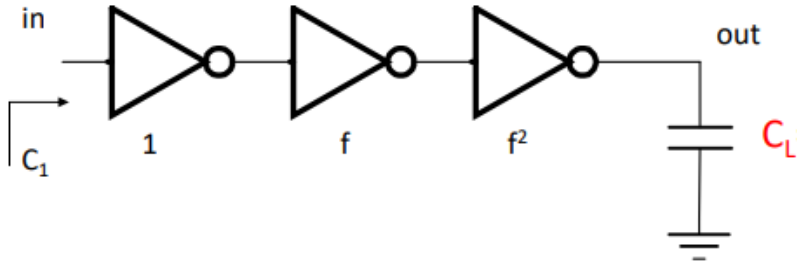
  - Delay will reduce ONLY if successive inverter size increases gradually

TAKEAWAYS FROM LAST CLASS

- Observations:
  - Total drive delay can be reduced by adding inverters



  - Delay will reduce ONLY if successive inverter size increases gradually
  - Condition for minimum delay is: $\frac{C_i}{C_{i-1}} = \frac{C_{i+1}}{C_i} = f$
    - Ratio of capacitance between successive stages increase by a factor f. i.e., inverter size must increase at every stage by a factor f



TAKEAWAYS FROM LAST CLASS

- Observations:
  - Under the minimum delay condition, each stage offers the same delay of (f_{opt} is optimal f)

$$t_p = t_{p0}\left(1 + \frac{f_{opt}}{\gamma}\right)$$

  - The total delay of the chain of inverters will be $\left[N \cdot t_{p0}\left(1 + \frac{f_{opt}}{\gamma}\right)\right]$
  - Since with each stage the strength increases by factor f_{opt}, $C_1 \cdot f_{opt}^N = C_L$
  - If $H = \frac{C_L}{C_1}$, $f_{opt} = \sqrt[N]{H}$, and $N = \frac{\ln H}{\ln f_{opt}}$ relates N, f_{opt}, and C_L
  - The total delay can be re-written as $\left[N \cdot t_{p0}\left(1 + \frac{\sqrt[N]{H}}{\gamma}\right)\right]$
  - Under the no-parasitic (no self loading) condition, $f_{opt} = e$ (~2.7) and $N_{opt} = \ln H$
  - Under practical conditions, f_{opt} and N_{opt} depends on γ (technology dependent)
  - For $\gamma = 1$, $f_{opt} \approx 3.6$ (rounded off to 4) → INVX1 → INV4x → INV16X … and find N_{opt}

GENERALIZING FOR ALL LOGIC GATES

- An inverter has an intrinsic delay (due to parasitic) and external delay due to the load it is driving.
- If C_1 is input cap of INVX1, C_L is the load, inverter delay modelled as:

$$t_p = t_{p0}\left(1 + \frac{C_L}{\gamma C_1}\right) \text{ and } t_{p0} = 0.69\, R\, (\gamma C_1)$$

- The INVX1 delay t_{p0} is technology dependant (lower nodes → lower t_{p0})
- We represent normalized delay d

$$d = \frac{t_p}{t_{p0}} = 1 + \frac{C_L}{\gamma C_1}$$

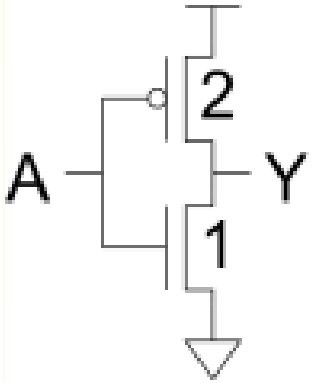
- First term represents the intrinsic (parasitic) delay and second term represents the effort to drive the load C_L. We used H for series of inverters. We will use h for a single inverter

$$d = p + h$$

GENERALIZING FOR ALL LOGIC GATES

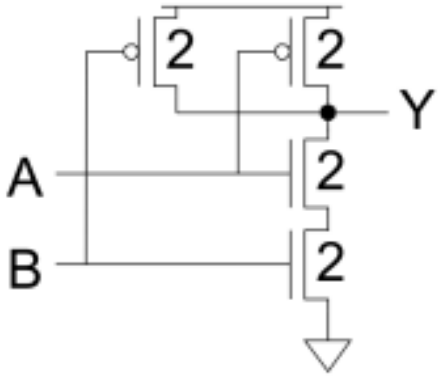
- What about the parasitics for other gates? $d = p + h$

INVX1



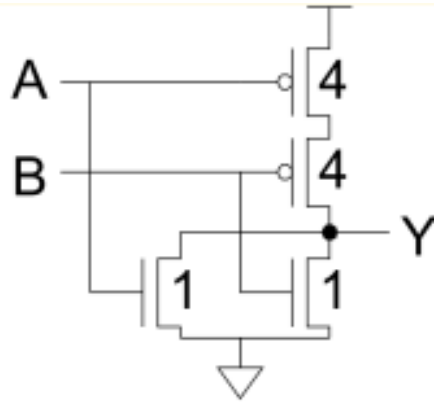
At node Y, total 3 widths and we took $p = 1$

2NANDX1



At node Y, total 6 widths therefore, $p = 2$

2NORX1



At node Y, total 6 widths therefore, $p = 2$

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n

laxmeesha@ee.iitb.ac.in

GENERALIZING FOR ALL LOGIC GATES

- Recall, for INVX1, we defined delay $\boldsymbol{d}$

$$d = \frac{t_p}{t_{p0}} = 1 + \frac{C_L}{\gamma C_1}$$

$$d = p + h$$

- For gates, other than inverter, the input capacitance will be larger than C_1
- To account for the increased input cap (which will increase the gate delay – since the input itself will be slowed due to increased input cap), we will add an additional term "$\boldsymbol{g}$"

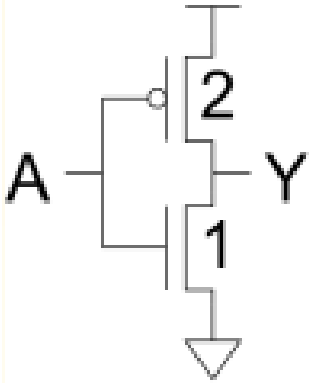
$$d = p + gh$$

- $\boldsymbol{g}$ = 1 for INVX1

GENERALIZING FOR ALL LOGIC GATES

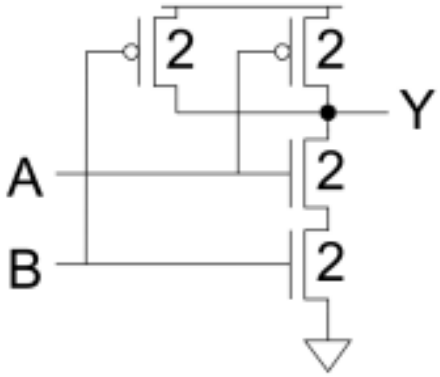
- What about the parasitics for other gates? $d = p + gh$

INVX1



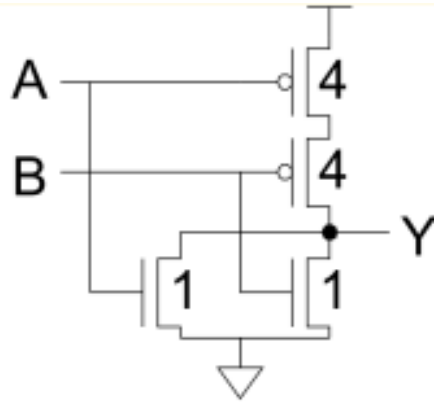
At node A, total 3 widths and we took $g = 3/3 = 1$

2NANDX1



At node A, total 4 widths therefore, $g = 4/3$

2NORX1



At node Y, total 5 widths therefore, $g = 5/3$

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		4/3	5/3	6/3	(n+2)/3
NOR		5/3	7/3	9/3	(2n+1)/3

laxmeesha@ee.iitb.ac.in

GENERALIZING FOR ALL LOGIC GATES

- Summary: $d = p + gh$

Parasitic delay for gates (p)

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n

Logical effort for gates (g)

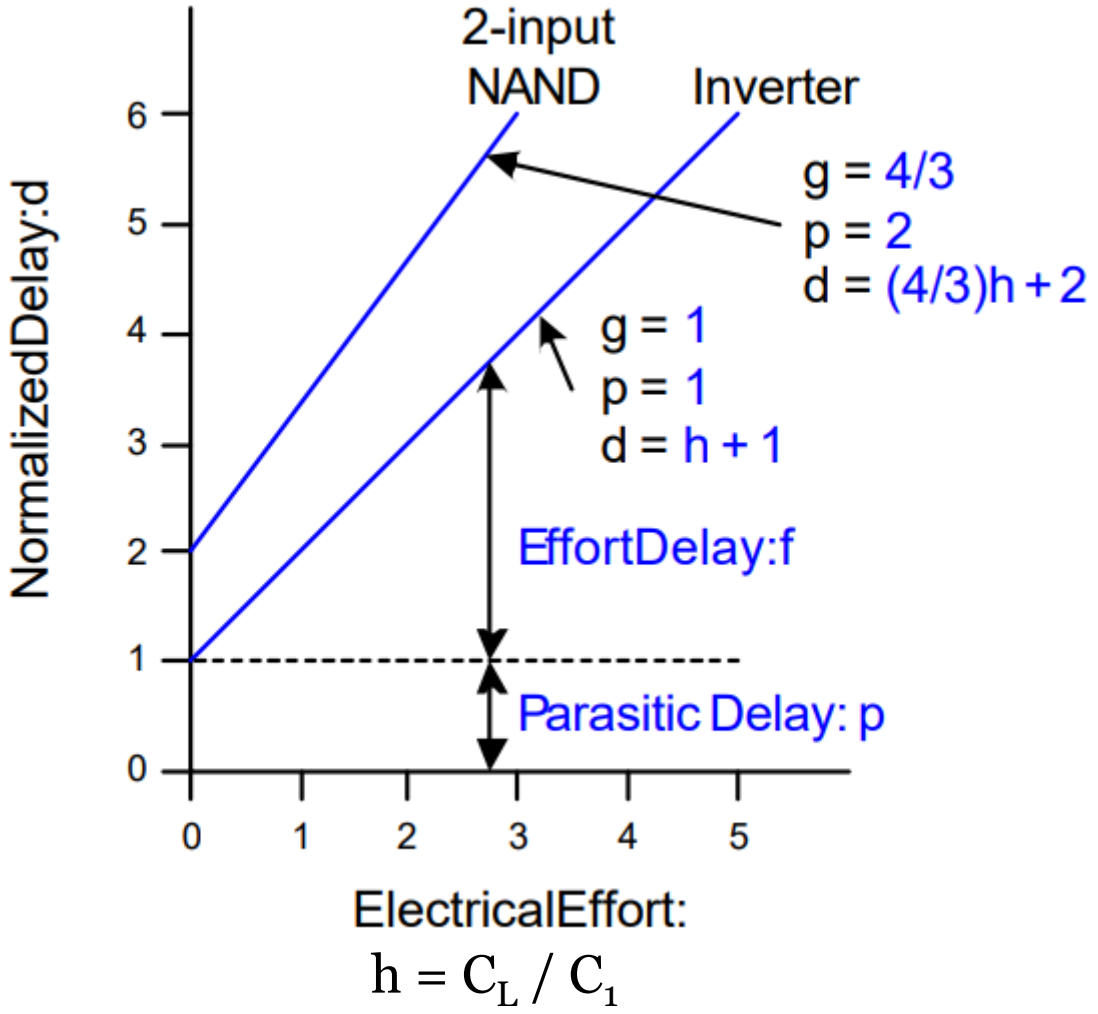
Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		4/3	5/3	6/3	(n+2)/3
NOR		5/3	7/3	9/3	(2n+1)/3

- d: normalized delay
- p: parasitic delay
- h: electrical effort
- g: logical effort

- $h = C_L/C_1$

GENERALIZING FOR ALL LOGIC GATES

- Visualizing the gate delays



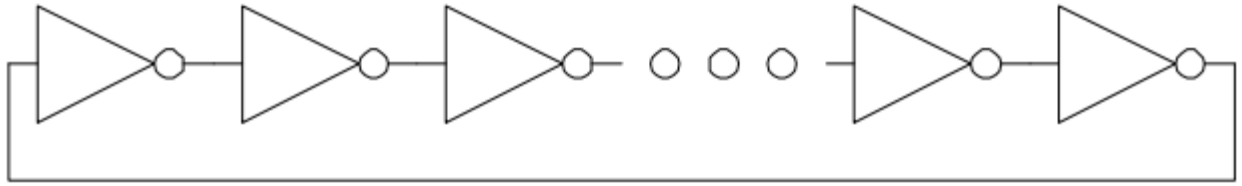
$$d = p + gh$$

- d: normalized delay
- p: parasitic delay
- h: electrical effort
- g: logical effort

- $h = C_L/C_1$

GENERALIZING FOR ALL LOGIC GATES

- Example: A ring oscillator
- A typical use of ring oscillator in a digital chip:
  - Identify the process corner!



Logical Effort: g =

Electrical Effort: h =

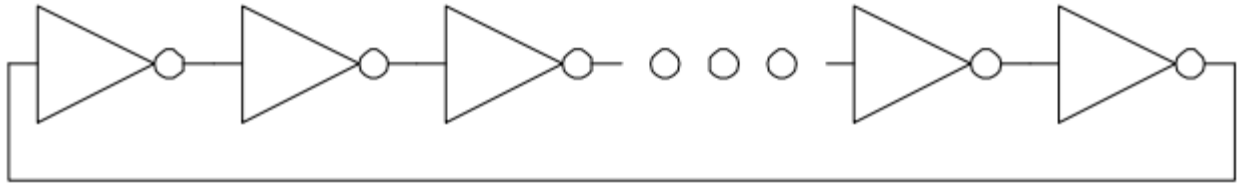
Parasitic Delay: p =

Stage Delay: d =

Frequency: f_{osc} =

GENERALIZING FOR ALL LOGIC GATES

- Example: A N-stage ring oscillator
- A typical use of ring oscillator in a digital chip:
  - Identify the process corner!



Logical Effort: g = 1

Electrical Effort: h = 1

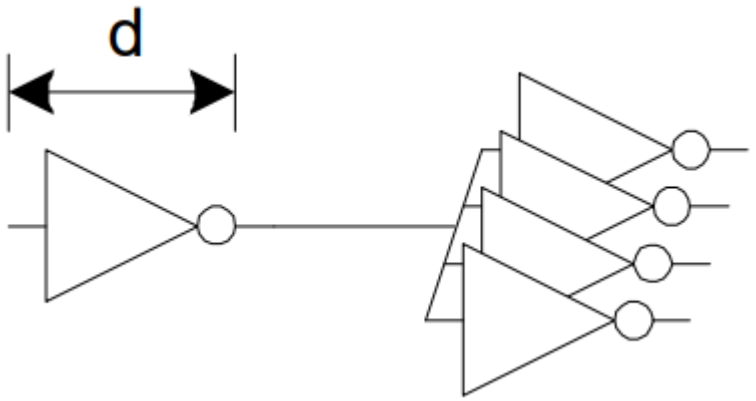
Parasitic Delay: p = 1

Stage Delay: d = 2

Frequency: $f_{osc} = 1/(2*N*d) = 1/4N$

GENERALIZING FOR ALL LOGIC GATES

- Example: delay of a Fanout-4 (FO-4) inverter



Logical Effort: g =

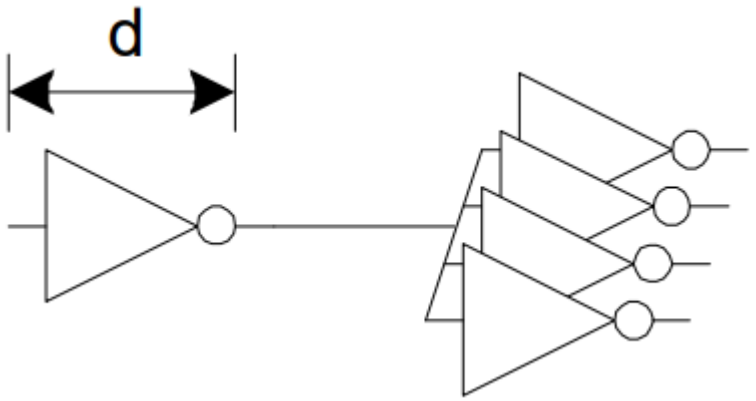
Electrical Effort: h =

Parasitic Delay: p =

Stage Delay: d =

GENERALIZING FOR ALL LOGIC GATES

- Example: delay of a Fanout-4 (FO-4) inverter



Logical Effort: g = 1
Electrical Effort: h = 4
Parasitic Delay: p = 1
Stage Delay: d = 5

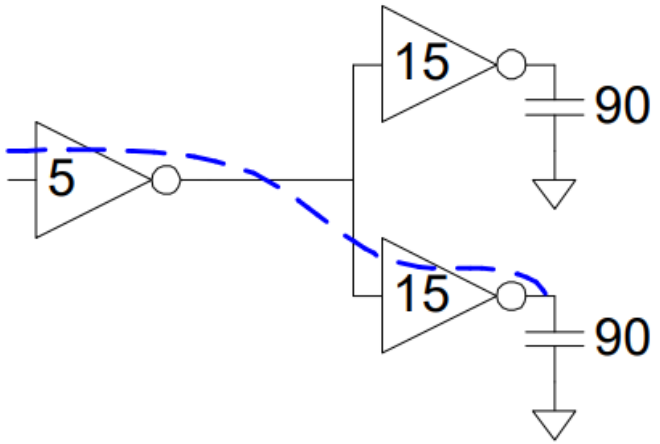
Generalizing for Multi gate paths

- Now that we have looked at individual gates, what about a network of these gates? $d = p + gh$ $d = p + f$
- Can we generalize a delay model for network of gates?
- We define **path** logical effort, $G = \prod g_i$
- We define **path** effort, $F = \prod f_i = \prod g_i \cdot h_i$, and $h_i = \frac{C_{out,i}}{C_{in,i}}$
- We define path electrical effort, $H = \frac{C_{L,path}}{C_{in,path}}$
- Are we missing anything else?
  - To answer this, let's pose a question: can we write $F = G \cdot H$?

GENERALIZING FOR MULTI GATE PATHS

- Consider the example:

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



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

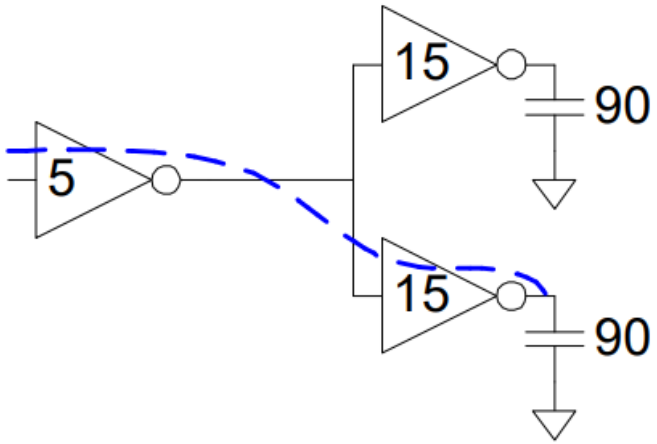
- $\boldsymbol{G} = \prod \boldsymbol{g_i}$
- $\boldsymbol{H} = \dfrac{\boldsymbol{C_{L,path}}}{\boldsymbol{C_{in,path}}}$
- $\boldsymbol{F} = \prod \boldsymbol{g_i} \cdot \boldsymbol{h_i}$

- $\boldsymbol{h} = C_L/C_1$

GENERALIZING FOR MULTI GATE PATHS

- Consider the example:

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



G = 1
H = 90 / 5 = 18
GH = 18
h_1 = (15 +15) / 5 = 6
h_2 = 90 / 15 = 6
F = $g_1g_2h_1h_2$ = 36 = 2GH

- $G = \prod g_i$
- $H = \frac{C_{L,path}}{C_{in,path}}$
- $F = \prod g_i \cdot h_i$

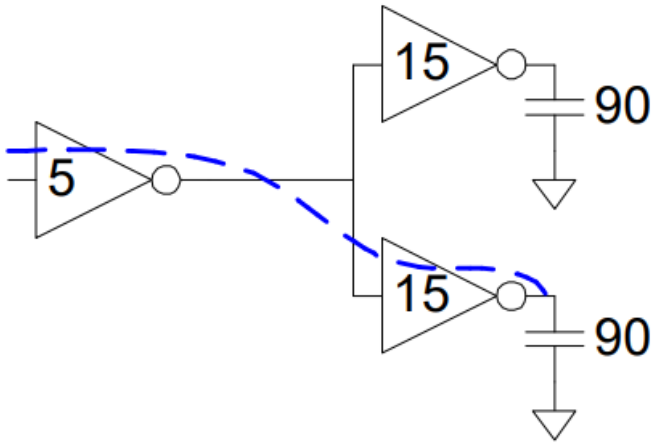
- $h = C_L/C_1$

- F is not the same as GH!! (because of the branching in the circuit)

GENERALIZING FOR MULTI GATE PATHS

- Consider the example:

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



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

- $\boldsymbol{G} = \prod \boldsymbol{g_i}$
- $\boldsymbol{H} = \dfrac{\boldsymbol{C_{L,path}}}{\boldsymbol{C_{in,path}}}$
- $\boldsymbol{F} = \prod \boldsymbol{g_i} \cdot \boldsymbol{h_i}$
- $\boldsymbol{h} = C_L/C_1$

- F is not the same as GH!! (because of the branching in the circuit)

GENERALIZING FOR MULTI GATE PATHS

- Need to introduce a branching effort $\boldsymbol{B}$
  - This will account for branching between stages in a path
- We define $\boldsymbol{b} = \frac{C_{on-path}+C_{off-path}}{C_{on-path}}$
- We define **path** branching effort $\boldsymbol{B} = \prod \boldsymbol{b_i}$
- **path** logical effort, $\boldsymbol{G} = \prod \boldsymbol{g_i}$
- $\boldsymbol{B} \cdot \boldsymbol{H} = \prod \boldsymbol{b_i h_i}$
- **path** effort, $\boldsymbol{F} = \boldsymbol{G} \cdot \boldsymbol{B} \cdot \boldsymbol{H}$

- The problem we are trying to tackle:
  - Given a network of logic gates, driving a load, what is the optimal sizing and number of stages to implement the logic → for least delay

GENERALIZING FOR MULTI GATE PATHS

- Recall that the total normalized delay has two components:
  - Parasitic delay and effort delay

$$d = p + gbh = p + f$$

- **Path** effort delay, $D_F = \sum f_i$
- **Path** parasitic delay, $P = \sum p_i$
- Total **path** delay, $D = P + D_F$
- Delay is the smallest when each stage (each gate in this case) offers the same delay → that is each stage effort increases by factor f_{opt} → this was our takeaway from inverter chain
- $f_{opt} = F^{\frac{1}{N}}$ *and* $F = G \cdot B \cdot H$
- For an inverter chain, $G = 1$ *and* $B = 1, f_{opt} = H^{\frac{1}{N}}$ → we obtained this in the inverter chain case!

Generalizing for Multi gate paths

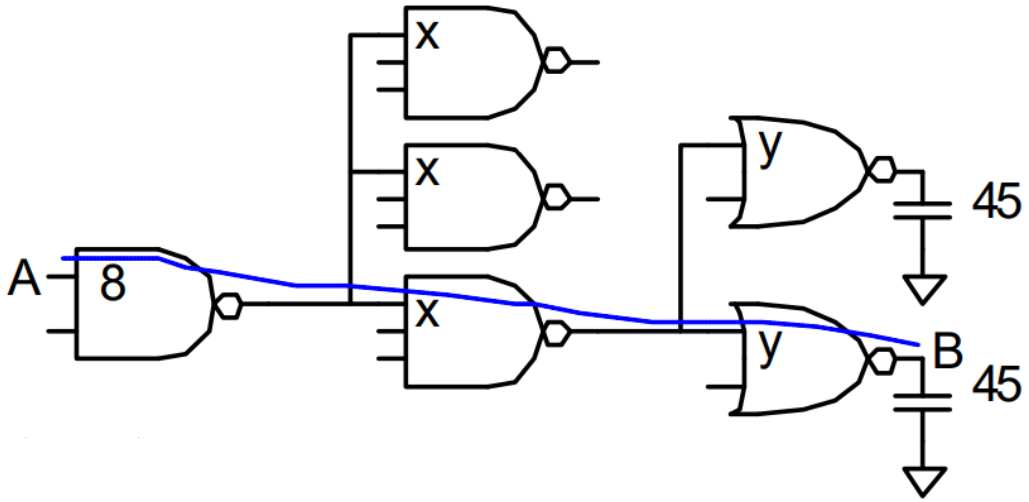
- Recall that the total normalized delay has two components:
  - Parasitic delay and effort delay

$$d = p + gbh = p + f$$

- **Path** effort delay, $D_F = \sum f_i$
- **Path** parasitic delay, $P = \sum p_i$
- Total **path** delay, $D = P + D_F$
- Delay is the smallest when each stage (each gate in this case) offers the same delay → that is each stage effort increases by factor f_{opt} → this was our takeaway from inverter chain
- $f_{opt} = F^{\frac{1}{N}}$ *and* $F = G \cdot B \cdot H$
- Total **path** delay, $D = P + N\,F^{\frac{1}{N}}$

GENERALIZING FOR MULTI GATE PATHS

- Example: find gate sizes for minimum delay

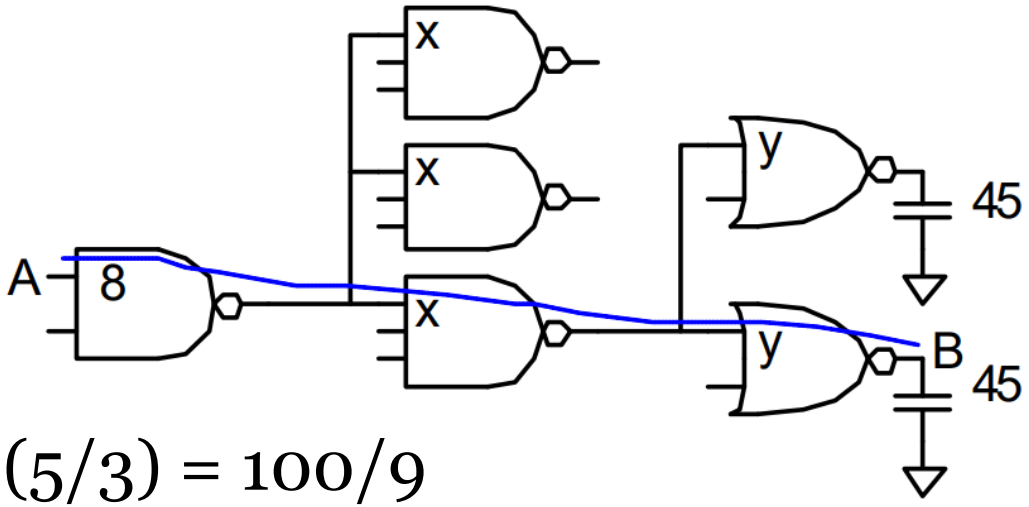


- Logical effort, G =
- Electrical effort, H =
- Branching effort, B =
- Path effort, F = GBH =
- Optimal stage effort, f_{opt} =
- Parasitic delay =
- Total Delay =

- $\boldsymbol{f_{opt} = F^{\frac{1}{N}}}$ *and* $\boldsymbol{F = G \cdot B \cdot H}$
- Total **path** delay, $\boldsymbol{D = P + N\,F^{\frac{1}{N}}}$

GENERALIZING FOR MULTI GATE PATHS

- Example: find gate sizes for minimum delay

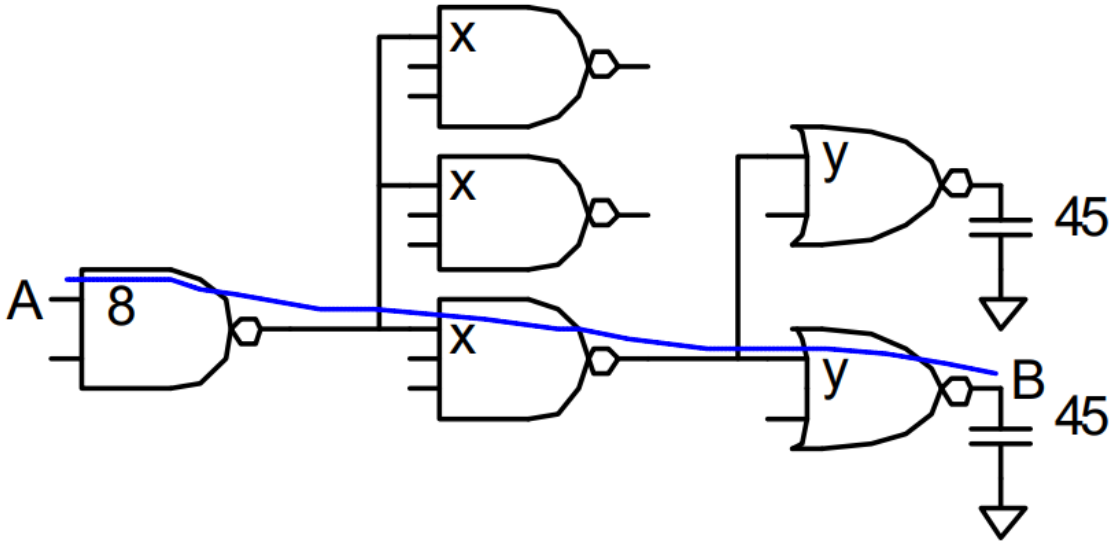


- Logical effort, G = (4/3) * (5/3) * (5/3) = 100/9
- Electrical effort, H = 45/8
- Branching effort, B = 3 * 2 = 6
- Path effort, F = GBH = 125
- Optimal stage effort, $f_{opt} = \sqrt[3]{F} = 5$
- Parasitic delay = 2 + 3 + 2 = 7
- Total Delay = 7 + (3*5) = 22

- $\boldsymbol{f_{opt} = F^{\frac{1}{N}}}$ *and* $\boldsymbol{F = G \cdot B \cdot H}$
- Total **path** delay, $\boldsymbol{D = P + N\,F^{\frac{1}{N}}}$

GENERALIZING FOR MULTI GATE PATHS

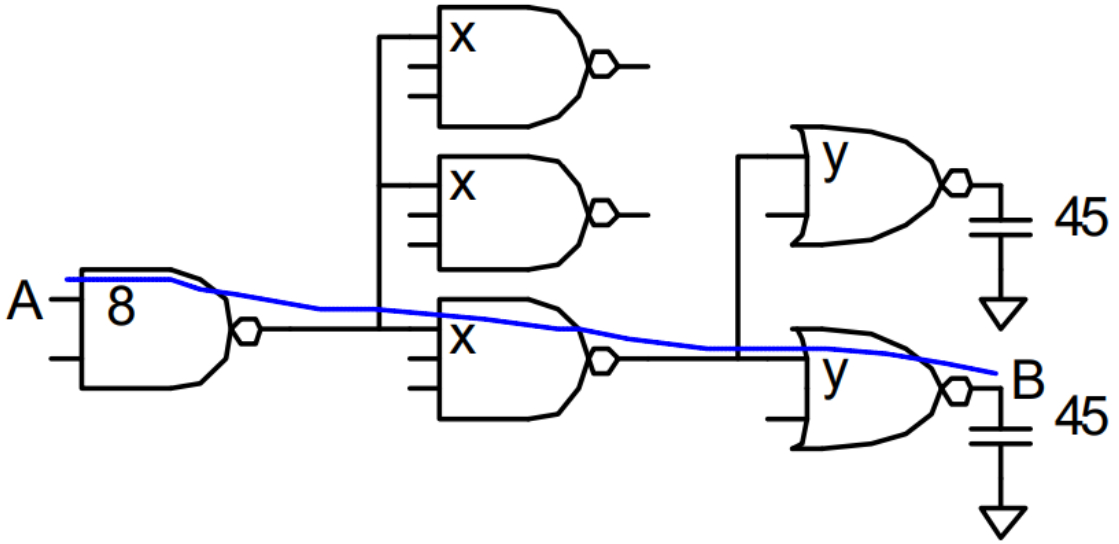
- Example: find gate sizes for minimum delay



- Optimal stage effort, $f_{opt} = \sqrt[3]{F} = 5$
- Recall, for each gate, $f_{opt} = gh = g\,(C_{out}/C_{in}) \rightarrow C_{in} = g\,C_{out}/f_{opt}$
- y = (5/3) * 45/5 = 15
- x = (5/3) * (15 * 2)/5 = 10
- To verify for the first gate, $C_{in} = (4/3) * (10*3)/5 = 8$

GENERALIZING FOR MULTI GATE PATHS

- Example: find gate sizes for minimum delay



- y = (5/3) * 45/5 = 15
- x = (5/3) * (15 * 2)/5 = 10
- To verify for the first gate, = (4/3) * (10*3)/5 = 8
- Stage-1: 2-input NAND with input cap of 8 → NANDx2
- Stage-2: 3-input NAND with input cap of 10 → NANDx2
- Stage-3: 2-input NOR with input cap of 15 → NORx3

Generalizing for Multi gate paths

- Summary:

Term	Stage	Path
number of stages	1	N
logical effort	g	$G = \prod g_i$
electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\text{-}path}}{C_{in\text{-}path}}$
branching effort	$b = \frac{C_{on\text{-}path} + C_{off\text{-}path}}{C_{on\text{-}path}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
delay	$d = f + p$	$D = \sum d_i = D_F + P$
effort delay	f	$D_F = \sum f_i$
parasitic delay	p	$P = \sum p_i$

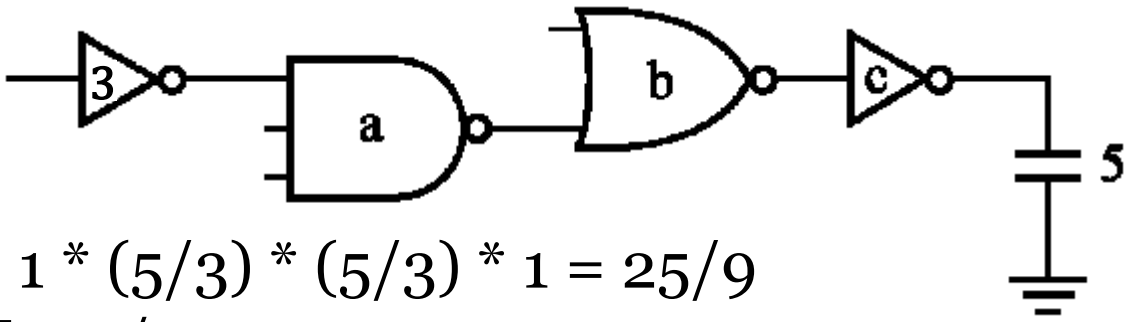
GENERALIZING FOR MULTI GATE PATHS

- Method/algorithm to size the path:

Step-1	**Compute the path effort**	**F = GBH**
Step-2	**If N is already chosen, proceed to step-3. If N is not given, the first estimate of N is calculated:** **(log base-4 since we saw that optimality is usually f_{opt} = 3.6)**	**$N = \log_4 F$**
Step-3	**Determine the optimal stage effort**	**$f_{opt} = \sqrt[N]{F}$**
Step-4	**Find individual gate caps (start from the output side)**	**$C_{in} = \frac{g\ Cout}{f_{opt}}$**
Step-5	**Based on input cap, obtain the gate strength**	
Step-6	**Estimate optimal delay – go back to Step-3 and iterate "N"**	**$D = P + NF^{\frac{1}{N}}$**

GENERALIZING FOR MULTI GATE PATHS

- Example-2:

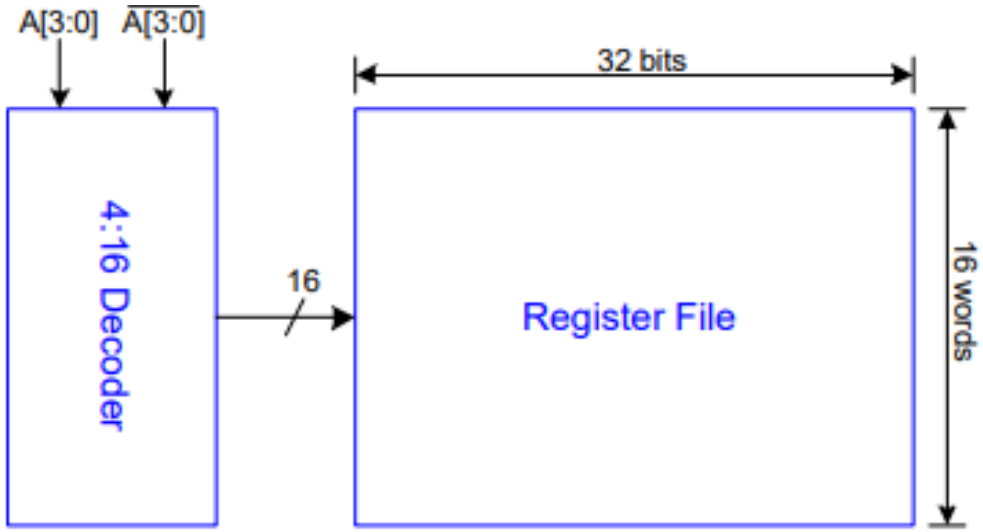


- Logical effort, G = 1 * (5/3) * (5/3) * 1 = 25/9
- Electrical effort, H = 5/3
- Branching effort, B = 1
- Path effort, F = GBH = 125/27
- Optimal stage effort, $f_{opt} = \sqrt[4]{F} = 1.46$
- c = 1 *5 /1.46 = 3.42 ~ 3 (INVx1)
- b = (5/3) * 3.42/1.46 = 3.90 → (5/3) * 3/1.46 = 3.42 ~ 5 (NORx1)
- a = (5/3) * 3.90/1.46 = 4.45 → (5/3) * 5/1.46 = 5.7 ~ 5 (NANDx1)
- INVx1 → 1*4.45/1.46 = 3.0 → 1 * 5/1.46 = 3.42 ~ 3 (INVx1)
- Parasitic delay: 1 + 3 + 2 +1 = 7
- Total delay=7 + 1* (5/3) + (5/3)*(5/5) + (5/3)*(3/5) + 1*(5/3) = 13
- Ideal delay = 7 + 4*1.46 =12.84

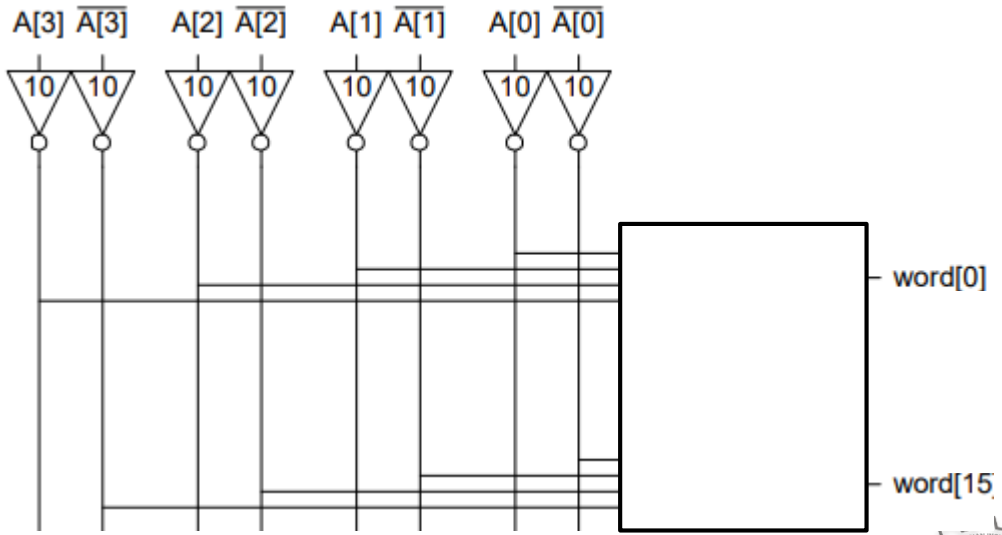
- $f_{opt} = F^{\frac{1}{N}}$
- $F = G \cdot B \cdot H$
- $D = P + N\,F^{\frac{1}{N}}$
- $C_{in} = g\,C_{out}/f_{opt}$

GENERALIZING FOR MULTI GATE PATHS

- Example-3: An address decoder for memory
- Need to design a 4 to 16 decoder
- Assume address (A) and the complement are available



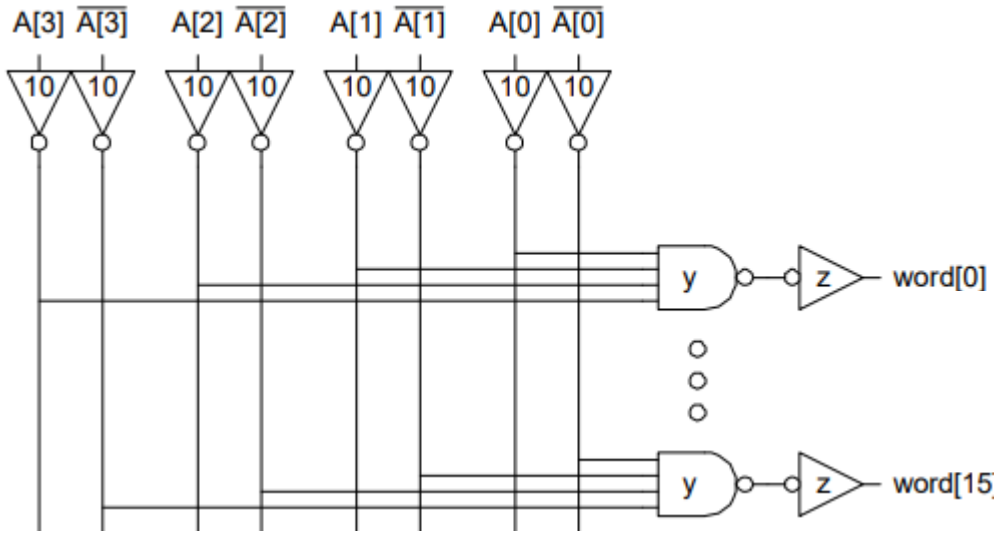
- Decoder specifications:
  - 16 word memory
  - Each word is 32 bits (4 bytes)
  - Each bit represents a load of 3-unit sized MOS
  - Each addr input can max drive 10-unit sized MOS



- Questions:
  - How many stages of gates required?
  - What will be the size of each gate?
  - What is the optimal decoder delay?

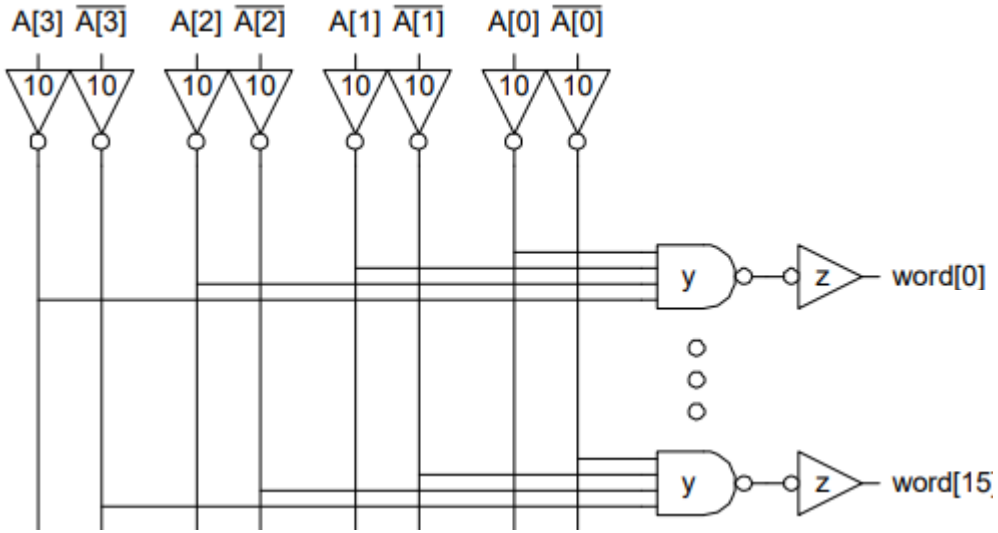
GENERALIZING FOR MULTI GATE PATHS

- We don't know what gates to use yet → start with assuming G = 1
- H = (32*3)/10 = 96/10
- B = 8 (since half of the inputs will be connected to A and other half to A-bar)
- F = GBH = 768/10 = 76.8
- N = $\log_4 F$ = 3.1
- Start iteration with a 3-stage design



laxmeesha@ee.iitb.ac.in

GENERALIZING FOR MULTI GATE PATHS



- G = 1 * 6/3 * 1 = 2
- B = 8, H = 9.6
- F = GBH = 153.6
- $f_{opt} = \sqrt[3]{153.6} = 5.35$
- Delay = (1 + 4 + 1) + (3*5.35) = 22.05
- z = 96*1/5.35 = 17.94 ~ 18
- y = 18*2/5.35 = 6.7

GENERALIZING FOR MULTI GATE PATHS

- To find the actual optima, you can write a script to iterate "N" and the associated logic (try the below table on pen and paper)

Design	N	G	P	D
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	3	2	6	22.1
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
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