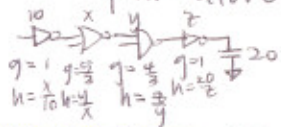


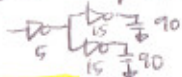
Lecture 7 (continued)

Multi-stage Logic Networks

logical effort = $G = \prod g_i$
 electrical effort = $H = \frac{C_{out, path}}{C_{in, path}}$
 path effort = $F = \prod f_i = \prod g_i h_i$



Note: $F \neq GH$



Lecture 8 Mon. Feb 11, 2008

Gate Size

$f = gh = g \frac{C_{out}}{C_{in}}$

best stage effort

$\hat{f} = \frac{\# \text{ stage}}{F}$

= logical effort @ every stage (try to get)

to find input capacitance,

value = output (gate's size) / \hat{f}
 ↑
 logical effort

* time is exponential compared to the effort taken



of stages, N
 = minimal effort at every stage does not mean minimal delay overall

Number of stages

$N = \log_4 F$, 4 = min. value at every stage (f value) for logical effort

Multiple output minimization (Q-M)

(a, b, c, d) : 3, 7, 2, 9, 3, 3, 3 ignore!!

1	0001	1101	00-1	1101	0-1	0001
4	0100	0110	0-01	0101	0-1	1100
3	0011	1101	001	1100	11-	0001
5	0101	0111	010	0110		
6	0110	0001	0-0	0000		
9	1001	1100	100	0110		
12	1100	0110				
7	0111	0001	0-11	0001		
11	1011	1100	0-11	1100		
14	1110	0001	0-1-	0001		
15	1111	0001	0-1-	0001		

(a, b, c, d) (3) 111- 0001 0001

when combining, use bitwise **AND**

Note: when fusing, you can't ignore tags (∴ needed)

Designing Fast Circuits

$D = \sum d_i = D_F + P$
 minimum delay in N is

$D = NF^{\frac{1}{N+1}}$, N stage path
 F = effort through each stage

Multiple output minimization (continued)

making the table:

P4	0-1	0001																	
P0	-11-	0001																	
P2	-0-1	1100																	
P0	010-	0110																	
P0	-100	0110																	
P0	0-01	0101																	
P4	00-1	1101																	
P4	0101	0111																	

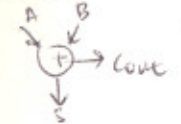
prime implicants

3₁ : 1
 3₂
 3₃
 3₄

Note: only look @ column dominance! (for # of literals)

Adders

Half Adder



AB	S	Cout
00	0	0
01	1	0
10	1	0
11	0	1

$S = A \oplus B$
 $C = AB$

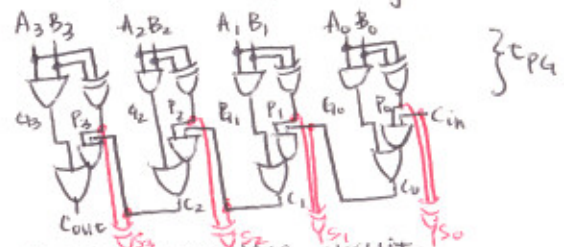
Generation & Propagation (continued)

$G_i = A_i B_i$
 $P_i = A_i \oplus B_i$
If $G_0 = A_0 B_0$ & $P_0 = A_0 \oplus B_0$
then $C_0 = G_0 + P_0 C_{in}$
 $C_1 = G_1 + P_1 C_0$
 $= G_1 + P_1 (G_0 + P_0 C_{in})$
 $= G_1 + P_1 G_0 + P_1 P_0 C_{in}$
 $C_2 = G_2 + P_2 C_1$
 $= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_{in})$
 $= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$

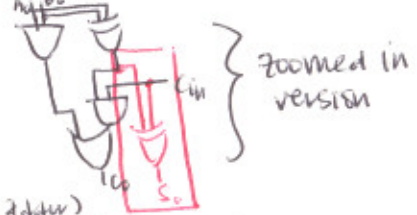
∴ Generally,

$C_n = G_n + P_n C_{n-1}$

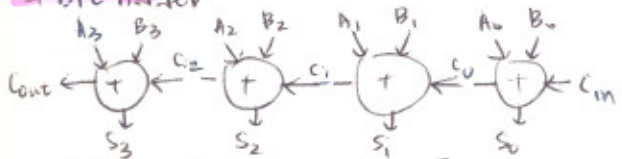
ex 4 bit adder (ripple carry adder)



Note: to make the circuit complete, (you need S values), add the circuit in red



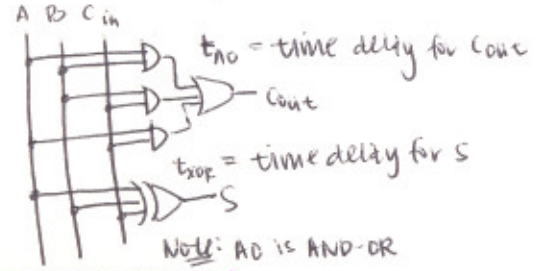
4 bit Adder



[cascading full adders]

the functionality consists of $A_{3:0} + B_{3:0}$

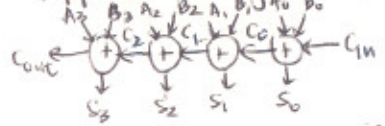
the circuit of a full adder



Note: AO is AND-OR

Ripple carry Adder

"ripples" through when adding two



time delay: $(N-1)t_{AO} + t_{xor}$

Generate - keeps track if there is a carry ($G_i = A_i B_i$)

Propagate - see if propagating a signal (when you cascade more than 1 unit)

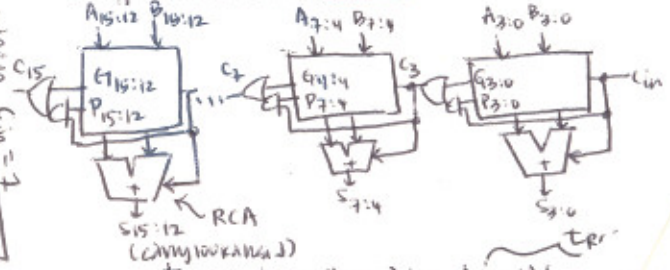
- example: when A_i, B_i & C_{in} is 1, C_i is 1 ∴ propagate signal is 1
 $P_i = A_i \oplus B_i$

ONLY WHEN $C_{in} = 1$

(ripple carry adder)

$t_{RCA} = t_{PG} + (N-1)t_{AO} + t_{xor}$
(for $P \& G$) (for C_i) (for S)

Carry look ahead adder



$t_{CLA} = t_{PG} + (K-1)t_{AO} + (n+1)t_{xor}$

$N = \#$ of bits
 $K = \#$ of stages
 $N = nK = \#$