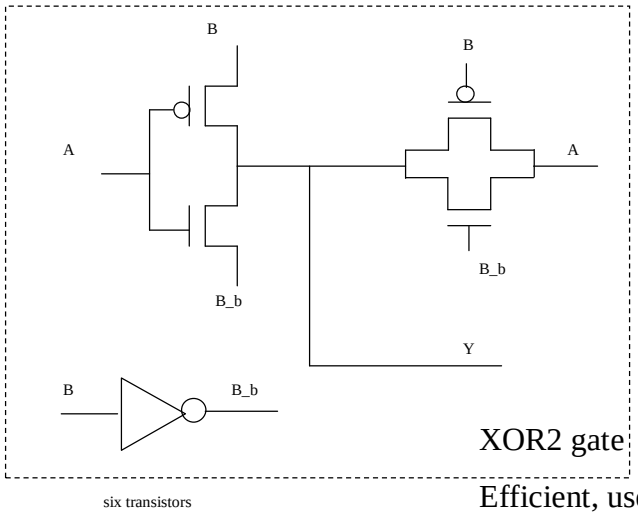
Pass Transistor Logic

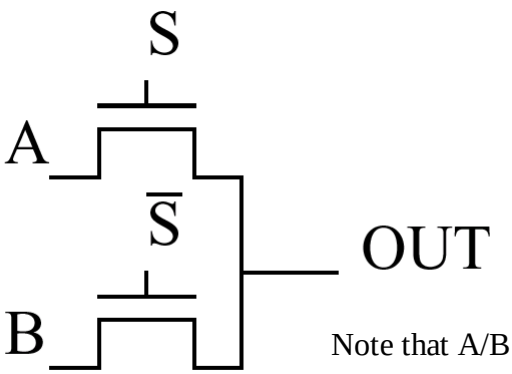


six transistors

XOR2 gate

Efficient, useful for arithmetic circuits

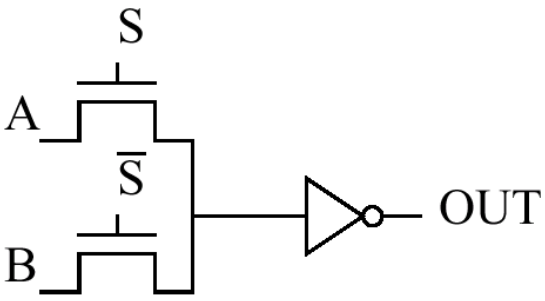
Simple Mux



Note that A/B provide drive.

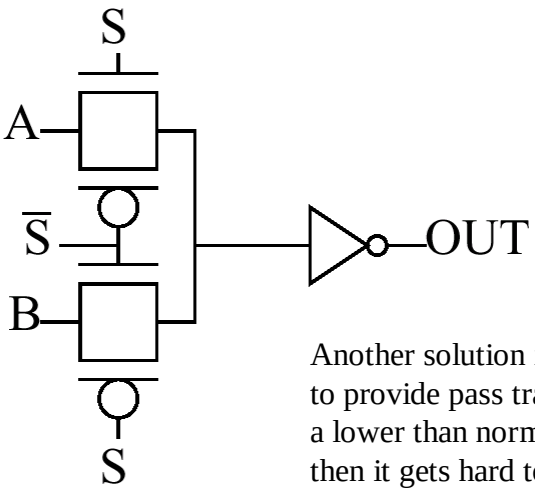
A 2/1 MUX is a complete logic family. Can build any logic function using 2/1 muxes.

Should add buffering to Provide drive



Unfortunately, power consumption will be a problem here because input to inverter will be at Vdd-Vt. This will mean the PMOS is not fully turned off. This problem gets worse the as you get deeper into submicron techologies.

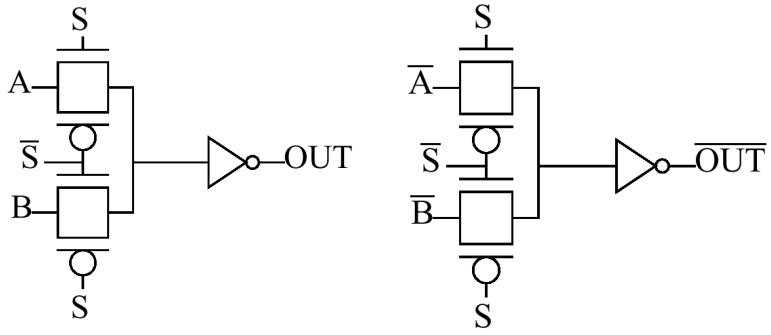
Use Transmission Gates



Another solution is for the process to provide pass transistors that have a lower than normal threshold (but then it gets hard to turn them off...).

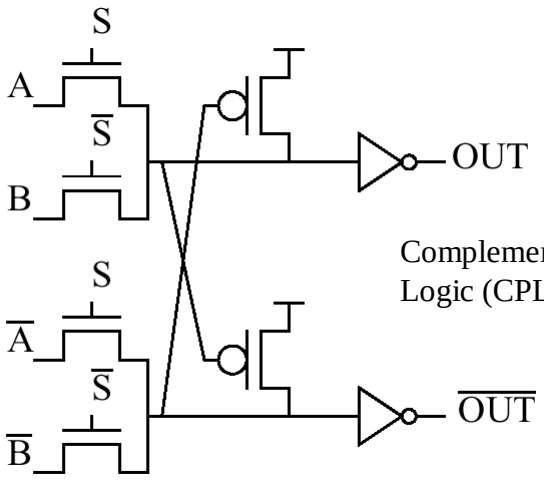
How to provide S, and NOT(S)?

When connecting 2/1 mux logic together to form larger gates, need both S, not(S). Can either add another inverter (slow), or duplicate logic to provide not(S).



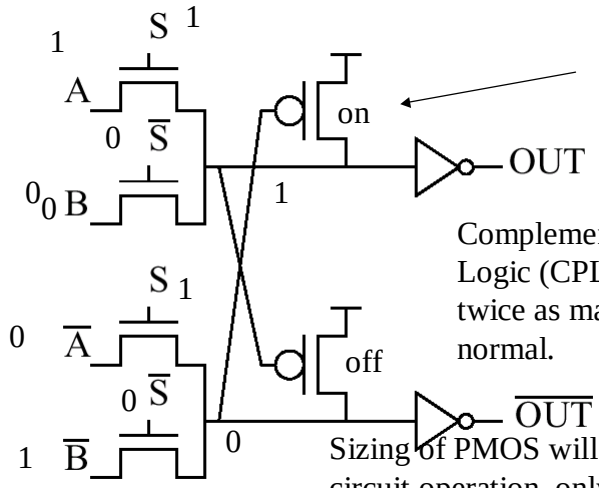
Transistor count has increased!!!! 12 tran. for 2/1 mux

Can we get rid of transmission gates?



Complementary Pass Transistor Logic (CPL).

Can we get rid of transmission gates?

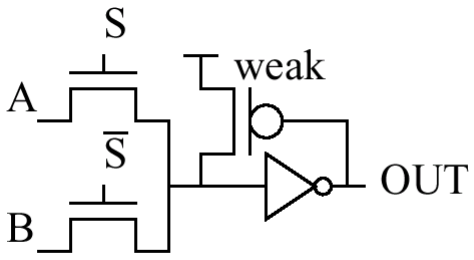


Pullups fully up to VDD (*level restorer*).

Complementary Pass Transistor Logic (CPL). Dual rail logic, twice as many signals as normal.

Sizing of PMOS will not cause incorrect circuit operation, only effect speed of falling transistions.

Getting rid of Complementary Gate

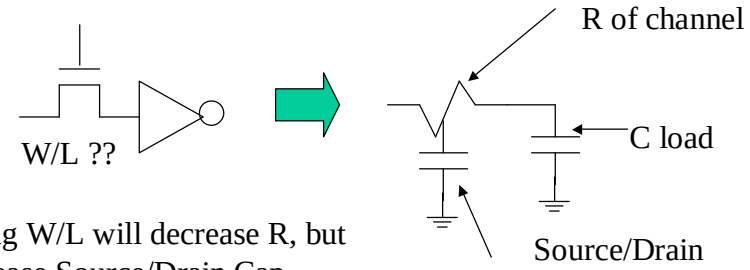


Must be careful to size PMOS weak enough to be overdriven by input. Performance degrades in submicron technologies.

Summary of Pass Transistor Logic

- Works well for mux logic
- Best choice is CPL with level restoring PMOS pullups
  - Can get rid of level restoring PMOS pullups with special 'low threshold' pass transistors. Requires support at process level.
- Layout can be difficult for pass transistors
- CPL dual rail requirements doubles routing requirements, and the area of most current designs are already more wiring limited than transistor limited.

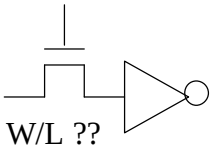
Sizing of Pass Transistors



Increasing W/L will decrease R, but will increase Source/Drain Cap

Delay = Rpass * (Cpass + Cload)
= Rpass*Cpass + Rpass*Cload

Sizing of Pass Transistors



W/L ??

Delay = Rpass * (Cpass + Cload)

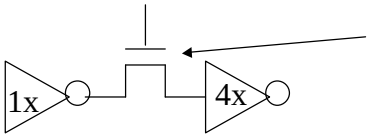
= Rpass*Cpass + Rpass*Cload

Constant with increasing W.

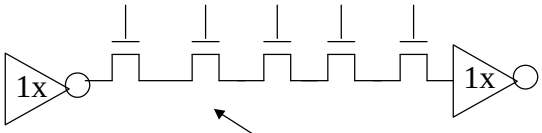
Decreasing with increasing W.

If Cpass << Cload, then get overall speed increase from increasing W. If Cpass > Cload, then little benefit from increasing W

Example

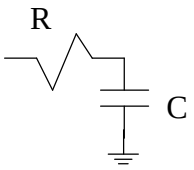


Larger than minimum size will result in higher speed.



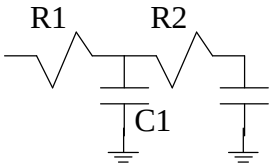
How do we size these transistors???

Elmore Delay

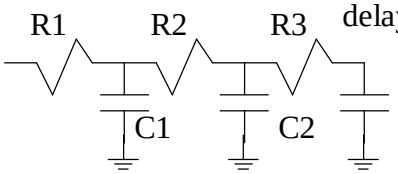


delay = R*C

First order approximation



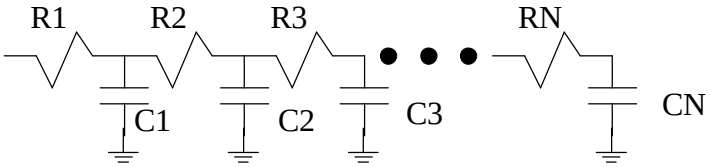
delay = R1*(C1+C2) + R2(C2)



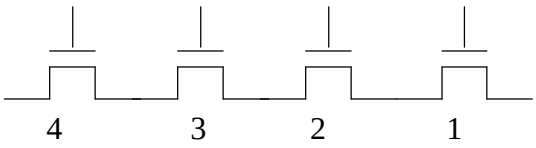
delay = R1*(C1+C2+C3) + R2(C2+C3) + R3C3

etc....

Elmore Delay (cont)



delay (approx) = $\sum Ri \sum Ci$ (i = 1 to N).



Progressive sizing works well.

Manchester Carry Chain (Pass transistor chain)

See. Figure 7.11 in Rabaey Textbook, pg 394

Sum = A xor B xor Ci
Cout = (A and B) or (B and Cin) or (A and Ci)

Define Generate, Propagate signals:

Generate true if FA block generates a carry.
Gi = A and B

Propagate true if FA block propagates a carry
P = A or B

Sum = P xor Ci
Cout = G + (P and Cin) Carry chain is critical path.