

CMOS Logic Gates

Fig. 1.8 p. 8, nMOS and pMOS FET

Fig. 1.9 p. 9, Fig. 1.19 p. 15 switch model

Fig. 1.10 p. 10, inverter; Table 1.1 inverter truth table

The inverter has several good characteristics. Both FET's are never ON at the same time (no static current implies no static power consumption) and nMOS passes 0, pMOS passes 1.

Fig. 1.21 p. 16, non-inverting buffer

This has poor performance because nMOS passes 1, pMOS passes 0. Don't use it!

Fig. 1.20 p. 16, transmission gate

Note that two gates have complementary inputs and therefore require two different control signals.

More complex logic gates are based upon the inverter and transmission gate.

Fig. 1.13 p. 11, inverter based gates; Table 1.3 p. 11 output states.

Fig. 1.14 p. 12 series parallel switches

Fig. 1.11 p. 10, CMOS NAND; Table 1.2 p. 11, NAND truth table

Note that the p-channel FETs in the pull up circuit implement the 1's in the truth table and the n-channel FETs in the pull down circuit implement the 0's in the truth table.

Fig. 1.12 p. 11, 3-input NAND

Fig. 1.15 p. 14, CMOS NOR; Table 1.4 p. 13, NOR truth table

Note that the p-channel FETs in the pull up circuit implement the 1's in the truth table and the n-channel FETs in the pull down circuit implement the 0's in the truth table.

Fig. 1.16 p. 13, 3-input NOR

Fig. 1.17 p. 14 a compound (complex) gate

The function

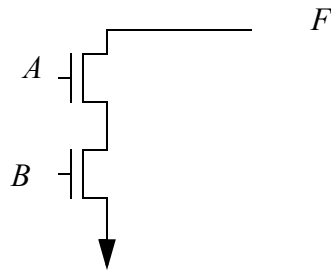
$$F = \overline{AB + CD}$$

is written in such a form so that it is easy to identify the zeroes of F , i.e. whenever

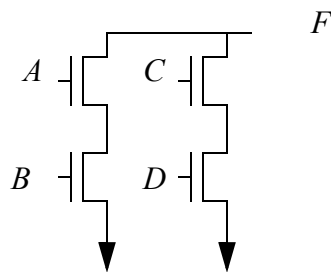
$$AB + CD$$

is true, F should be 0. We can use this to design the pull down circuit out of n-channel

FETs. Whenever A and B are both 1, there should be a path from F to ground. The “and”



combination can be implemented by a pair of n-channel FETs in series. Similarly, there should also be a path from F to ground whenever both C and D are 1. Another pair of n-channel FETs can be added in parallel with the first pair to accomplish this. In general,



whenever an “and” combination is required, a series combination of FETs can be used to implement it. An “or” combination can be implemented with a parallel combination.

The same procedure can be used for the pull up circuit to implement the 1’s of F , but first F must be written in a form that makes it clear what the 1’s are. Using DeMorgan’s laws,

$$\begin{aligned} F &= \overline{AB + CD} \\ &= (\overline{AB})(\overline{CD}) \\ &= (\overline{A} + \overline{B})(\overline{C} + \overline{D}) \end{aligned}$$

we see that whenever

$$(\overline{A} + \overline{B})(\overline{C} + \overline{D})$$

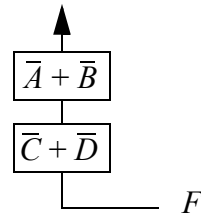
is true, F should be 1. We can regard the pull up circuit as a series combination of two sub-functions,

$$(\overline{A} + \overline{B})$$

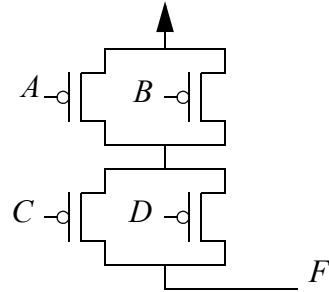
and

$$(\overline{C} + \overline{D}) .$$

Whenever both sub-functions are true, there should be a path from power to F.



Each sub-function can be implemented as a parallel combination of p-channel FETs.



We can now put the pull up circuit with the pull down circuit to form the complete CMOS logic gate.

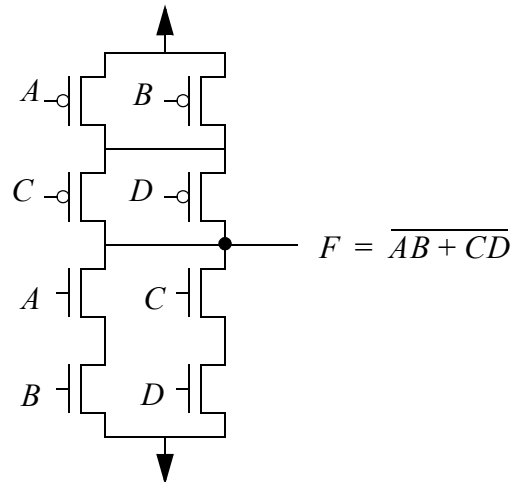
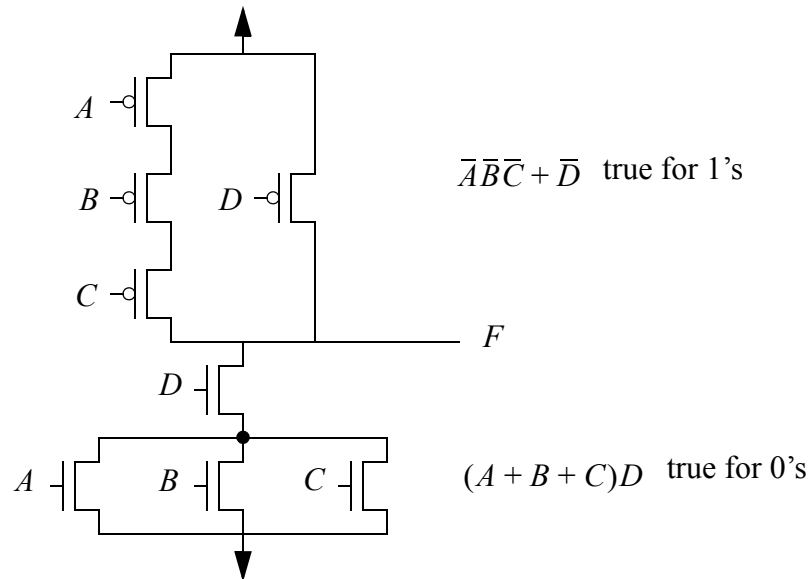


Fig. 1.18 p. 15 another complex gate

$$F = \overline{(A + B + C)D}$$

$$= \bar{A}\bar{B}\bar{C} + \bar{D}$$



It is not really necessary to derive both the pull up and pull down circuits. Once one has been derived, the other can be determined easily. The pull up and pull down circuits are duals of each other. Whenever there is a series combination in one, the other must have a parallel combination.

Fig. 1.26 p. 18, tristate inverter

The truth table is

EN	A	Y
0	0	Z
0	1	Z
1	0	1
1	1	0

The truth table in Table 1.5 p. 17 is for a transmission gate which is the same as a truth table for a NON-inverting tristate buffer. Beware that the transmission gate is not really a buffer since the output is connected to the input when the transmission gate is on.

Fig. 1.27 p. 19 MUX; Table 1.6 p. 19, truth table

This design style is significantly different than what was done for the inverter. In the inverter, the output is connected directly to power or ground through the FET switches. In the MUX design, the output is instead connected to one of the inputs through the FET switches. This design style is called “pass transistor logic” because the input passes directly through the switches to the output.

Pass transistor logic can frequently be used to reduce the transistor count in a circuit implementation. Consider a complex CMOS gate that implements the same function as the MUX but with inverted output.

$$Y = \overline{D_1 S + D_2 \bar{S}}$$

$$= (\bar{D}_1 + \bar{S})(\bar{D}_2 + S)$$

Fig. 1.28a p. 19 CMOS inverting MUX

Fig. 1.28b p. 19 tristate inverting MUX

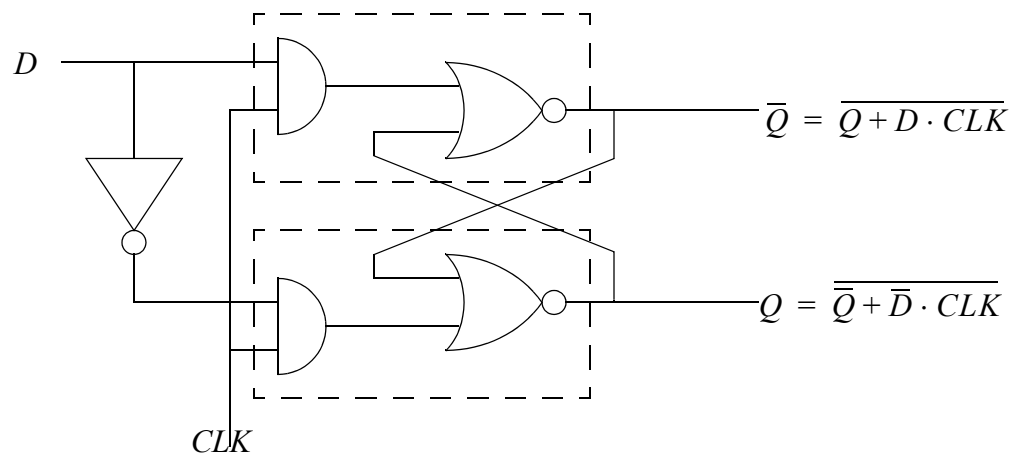
Fig. 1.29b p. 20 tristate inverting 4 to 1 MUX

The CMOS gate for the inverting MUX has twice as many transistors as the pass transistor implementation of the non-inverting MUX. So does the tristate implementation. But they both provide buffering that the transmission gate MUX does not.

Fig. 1.30 p. 21 latch







Fig. 1.31 p. 22 register

The above two examples show that pass transistor style design can also be used in clocked systems to reduce transistor count. Compare the design for the latch with one made from standard logic gates.

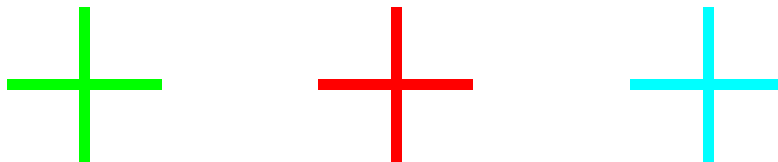


There are two complex gates with 6 FETs each plus an inverter for a total of 14 FETs. The latch on p. 21 has 8 FETs.

Stick Diagrams

COLOR	NAME	FUNCTION
green 	ndiff	source/drain
yellow 	pdiff	
red 	poly	gate
blue 	metal 1	interconnect
violet 	metal 2	
light blue 	metal 3	

Connections



crossing lines in same layer are always connected.



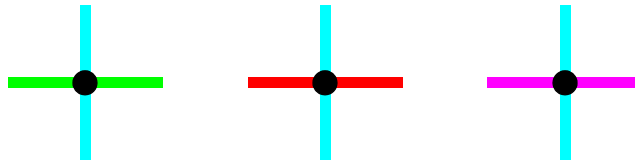
crossing lines in different layers are always unconnected



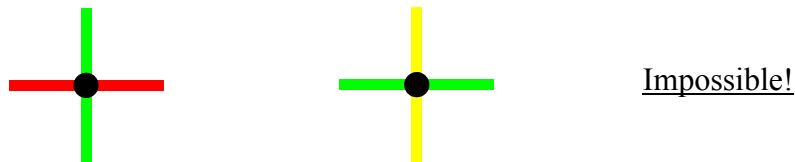
exception:

poly crossing diff always makes FET (even if you don't want one there).

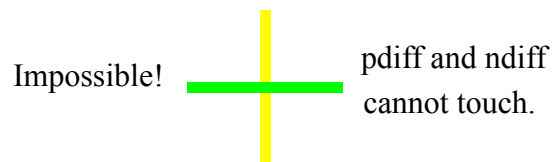
To get connections between layers, use contacts



Note: contacts possible between metal and any other layer, but that is all!
Metal must be one of the layers in the contact. For example,



one other restriction:



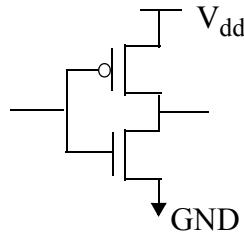
Rules of the layer “game”

1. Keep areas of all interconnect (not FET) to a minimum.
2. Keep interconnect lengths to a minimum in the following prioritized order

ndiff, pdiff	short lengths only
poly	short or intermediate lengths OK
metal 1, metal 2, etc.	any length OK
3. Use as few contacts as possible.
4. One of the layers in the contact must be metal.

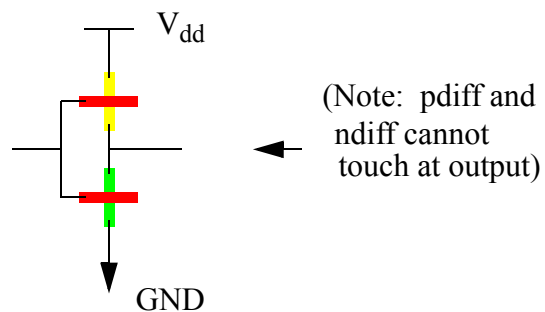
Layout Procedure

1. Make transistor circuit diagram.

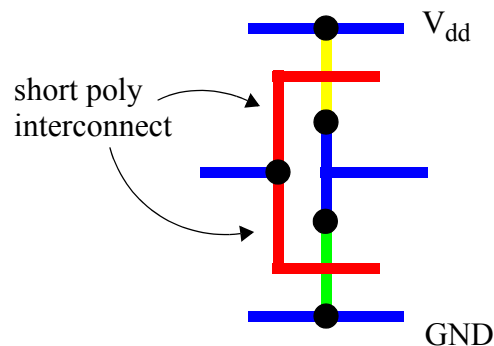


2. Make stick diagram (symbolic layout) to help plan layout.

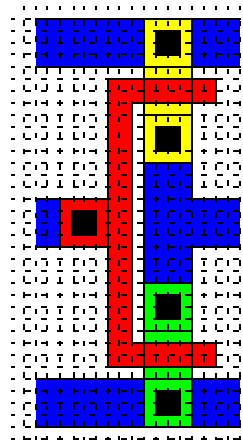
- a. For each transistor, make short diffusions for source and drain and cross with poly to make gate.



- b. Complete stick diagram by adding interconnect plus power and ground connections.
Note: short interconnect can be poly, but long interconnect must be metal. Do not use diffusion lines for interconnect.



3. Use the layout editor program to make layout by “fleshing out” stick diagram with correct spacing and line widths without design rule violations.



Symbolic Layout Examples

inside front cover

Fig. 1.43a p.33, inverter. The width of the lines has no real significance. Note that the transistors are rotated so that a straight poly line can be used to make the transistor gates. This makes for much more efficient layouts for CMOS gates.

Fig. 1.72 p. 65, NAND gate. Layout style from transistor diagram.

Fig. 1.43b p. 33, NAND gate. The “line of diffusion” layout style is more efficient since it allows transistors to be placed end to end sharing contacts and has much simpler gate connections with parallel poly lines.

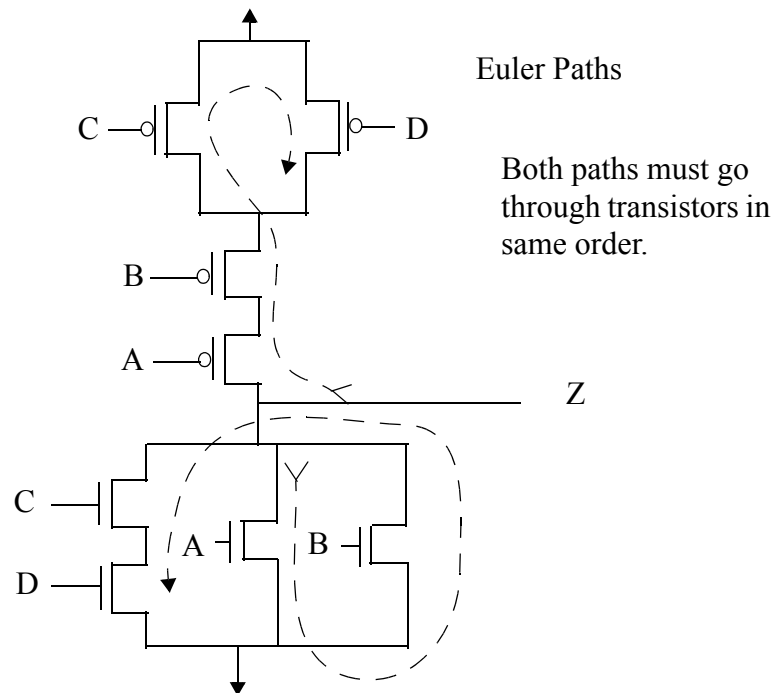
Fig. 8.55a, 8.55b p. 557, NOR gate. Two versions of the “line of diffusion” layout style. The one with the fewest contacts on the output node is faster.

The “line of diffusion” layout style can be extended to more complex gates by attempting to layout all of the FETs of each type end to end. This is possible if a path through the transistor circuit can be found that includes each transistor once and only once.

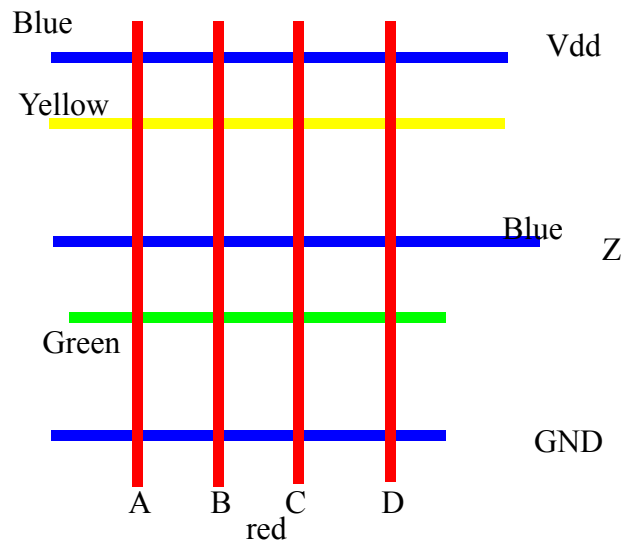
Fig. 8.49 p. 552, Finding circuit graph from the transistor circuit diagram.

Fig. 8.50 p. 552, Euler path from graph and corresponding layout

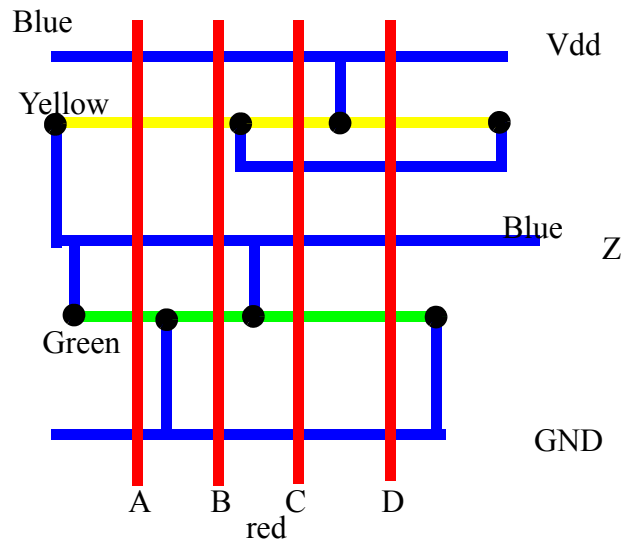
Simplified design process using Euler paths:



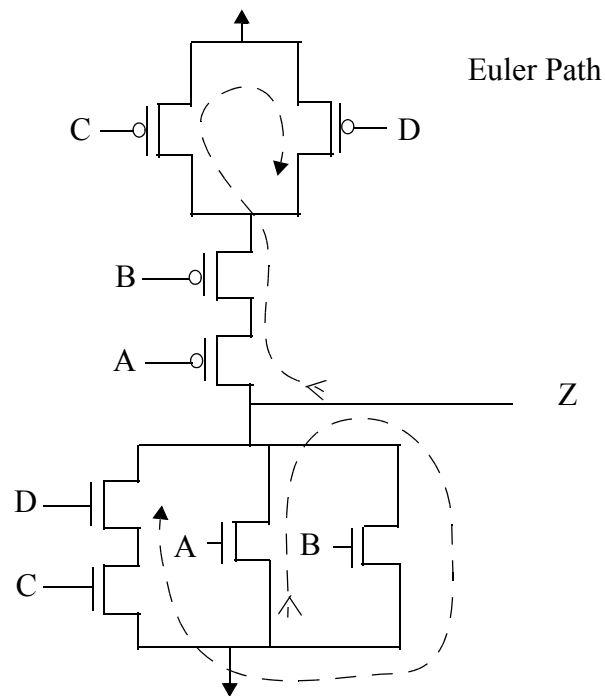
We know immediately that the stick diagram will look like this:



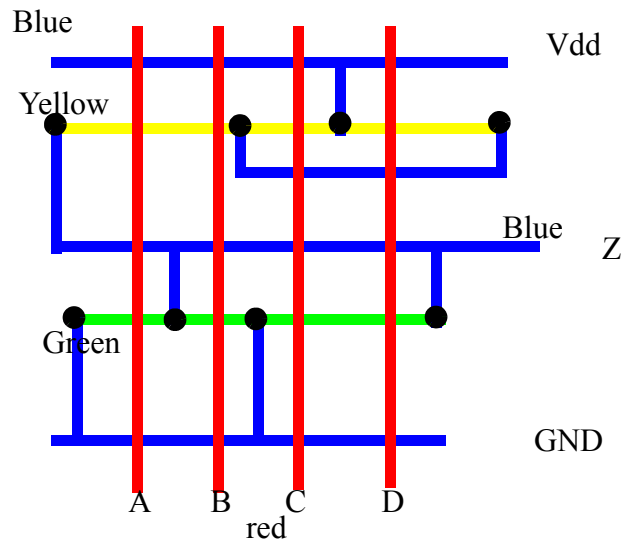
Then we add the metal interconnect to implement the rest of the circuit topology.



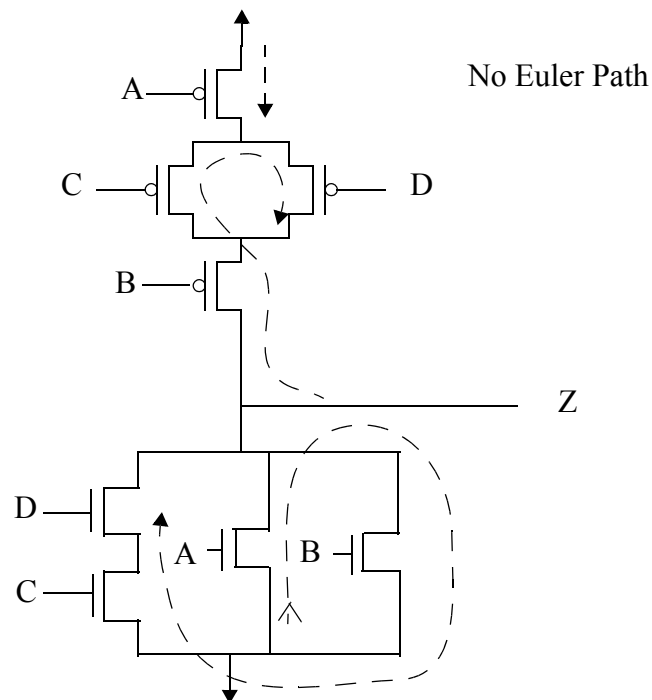
There may be several different valid stick diagrams for the same logic gate. By interchanging the nMOSFETs controlled by C and D, the Euler path can also be chosen as follows.



This results in an equally good but different stick diagram.

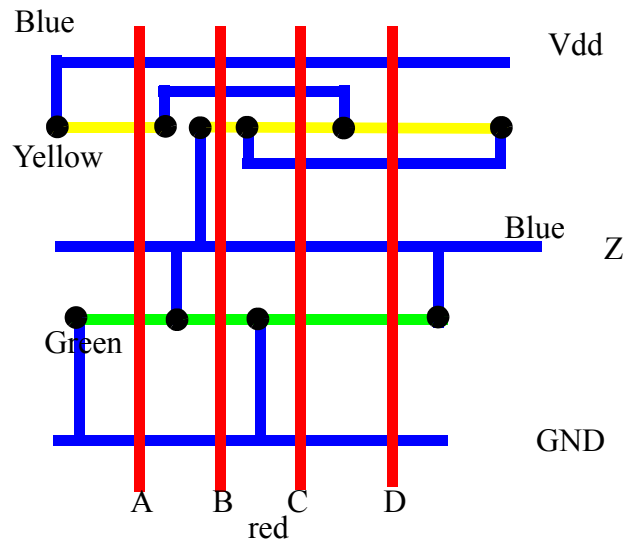


Not all circuits have an Euler path. By putting the pMOSFET controlled by A at the top, there is no Euler path and instead the two paths shown must be used.

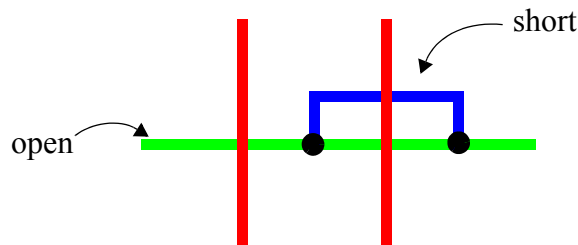


This results in a stick diagram with a gap in the p-diffusion. Note the extra contacts compared with the other designs of the same gate. Whenever an Euler path is possible, it gives

a more compact layout than layouts with gaps in the diffusion lines.



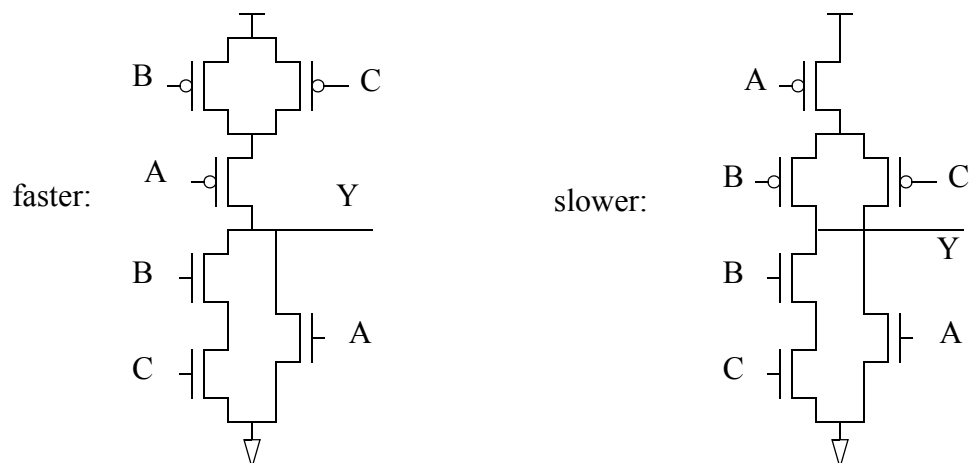
After the metal lines are routed, one can check the stick diagram for obvious errors. A few examples are shown below. Leaving the end of a diffusion line unconnected causes an



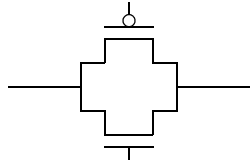
open circuit fault. Strapping a metal line over a single transistor causes a short fault across the transistor.

Fig. 8.51 p.553, Cascaded gates with common inputs

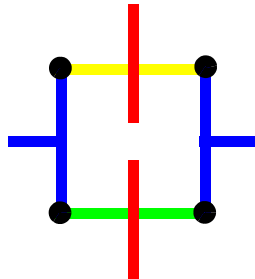
For faster gates, whenever there is a choice, always reduce the loading (contacts) on nodes closer to the output node. For example,



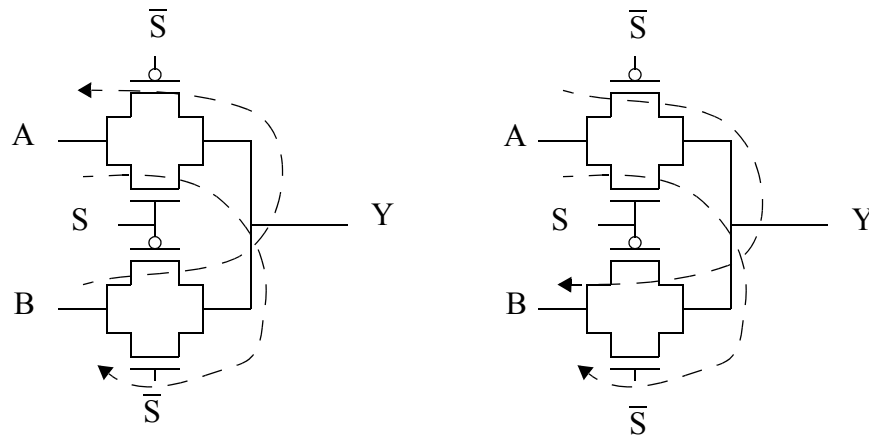
Recall transmission gate definition:



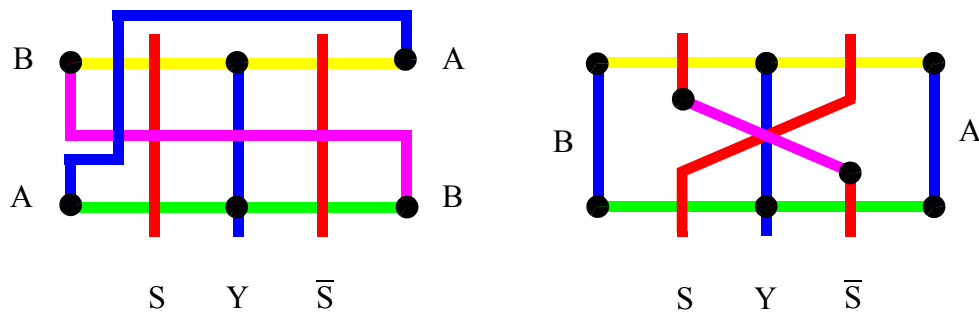
Transmission gates do not easily fit into the “line of diffusion” layout style because a the vertical poly line must be split for the two complimentary inputs.



Recall the circuit for the transmission gate multiplexer.



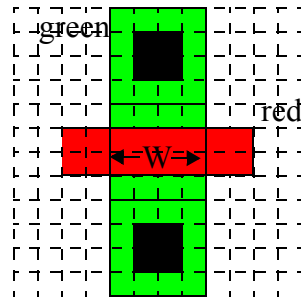
Euler paths through the pFETs and the nFETS give the following stick diagrams.



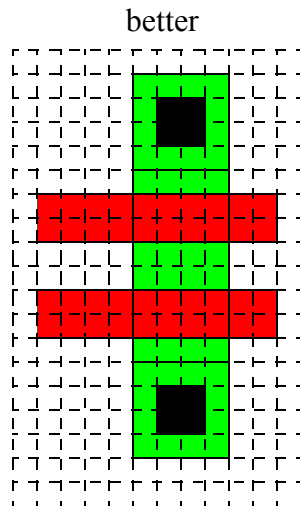
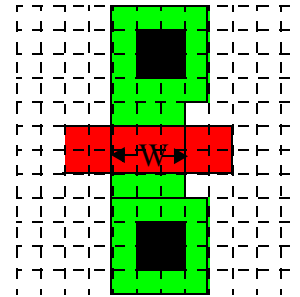
Crossovers are necessary in either metal or poly.

Making Layout from Stick Diagrams

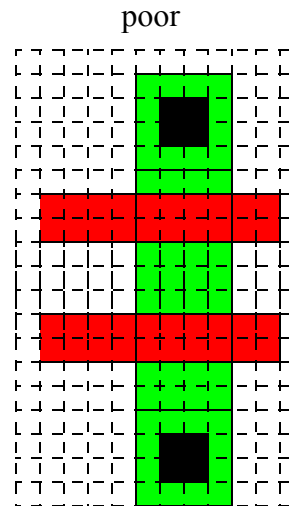
Stick diagrams (symbolic layout) show which layers are used and the relative position of the layers, but they do not accurately depict the width of the layers or the spacing between them. The line widths and spacings must be chosen carefully when constructing layouts. Common mistakes made by naive designers:



Make channel at least as wide as contacts



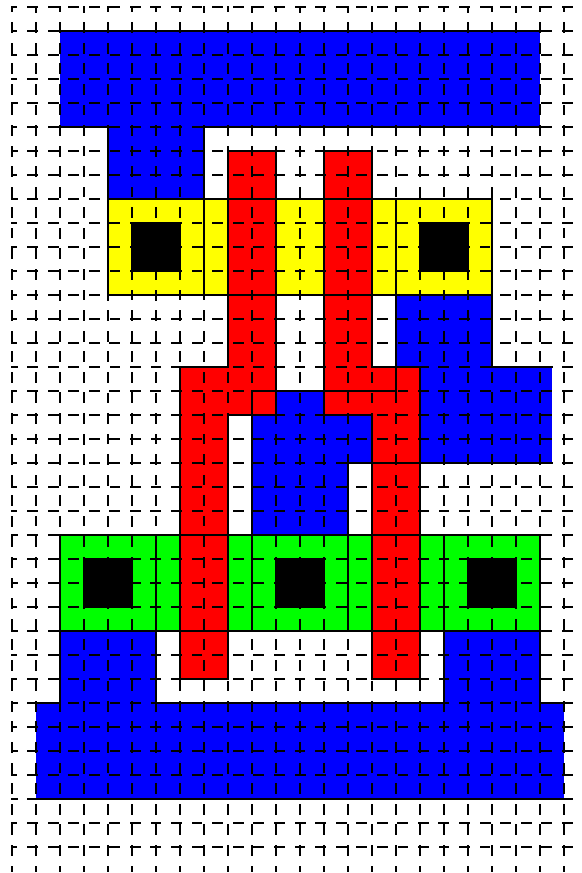
Use minimum space between series transistors and between contacts and transistors



This means that real layout can look somewhat different from the stick diagrams. For example, in the NOR gate, the poly lines in the layout are not straight even though they are drawn that way in the stick diagram. This is necessary to minimize the distance between the series pFETs while accommodating the contact between the nFETs.

Fig. 8.55a p. 557 NOR gate stick diagram.

NOR gate layout.



Note that the layout looks like a filled out stick diagram, but the spatial dimensions are no longer arbitrary. Also the exact shapes may be slightly different as for example the “dog legs” in the poly lines.