Solutions

Note: solutions to simulation exercises are not included. Chapter 12 is not yet complete, and a few other solutions are presently missing. DH 8/4/04

Chapter 1

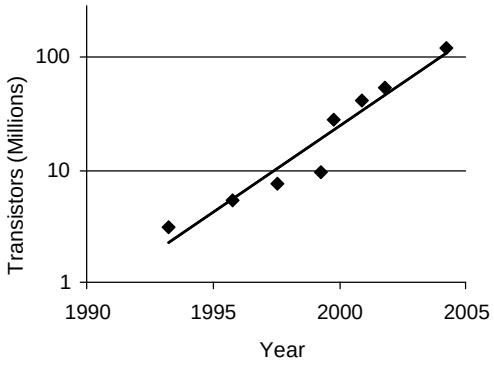
1.1 Starting with 42,000,000 transistors in 2000 and doubling every 26 months for 10 years gives $42\text{M} \bullet 2^{\left(\frac{10 \cdot 12}{26}\right)} \approx 1\text{B}$ transistors.

1.2 Some recent data includes:

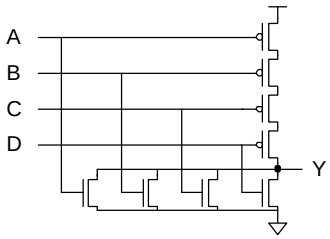
Table 1: Microprocessor transistor counts

Date	CPU	Transistors (millions)
3/22/93	Pentium	3.1
10/1/95	Pentium Pro	5.5
5/7/97	Pentium II	7.5
2/26/99	Pentium III	9.5
10/25/99	Pentium III	28
11/20/00	Pentium 4	42
8/27/01	Pentium 4	55
2/2/04	Pentium 4 HT	125

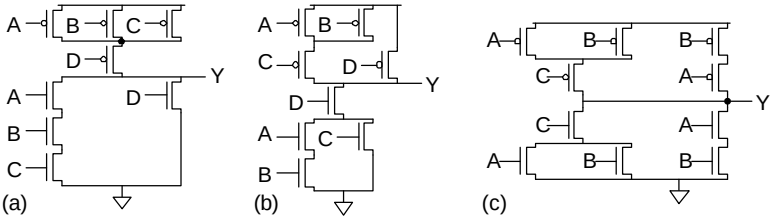
The transistor counts double approximately every 24 months.



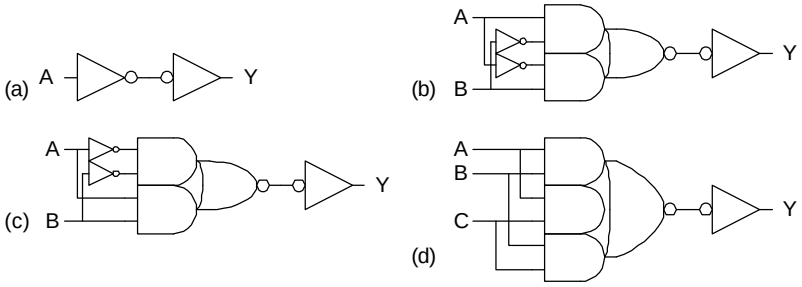
1.3



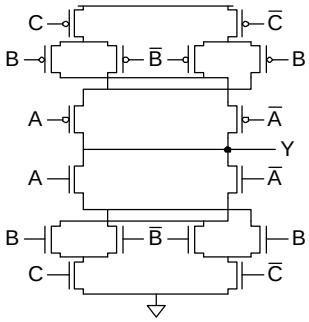
1.4



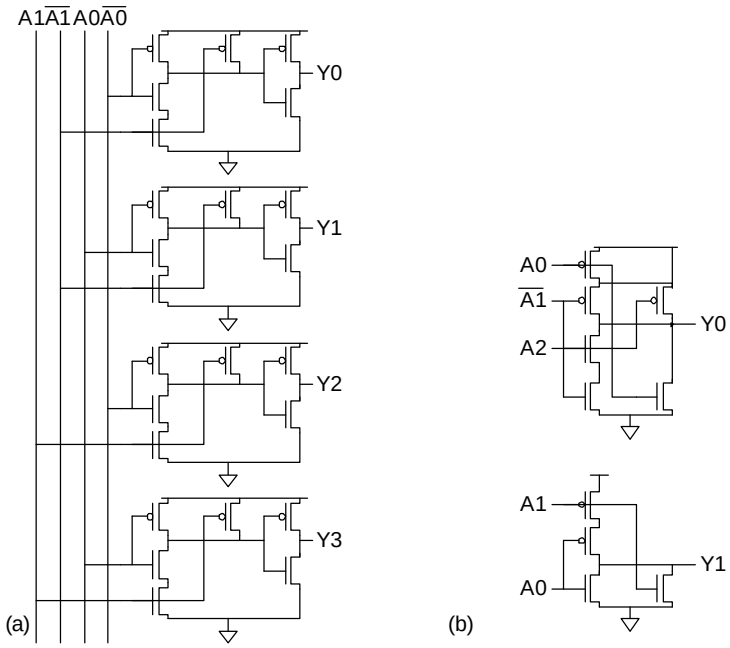
1.5


(a)
A
Y
(b)
A
B
Y
(c)
A
B
Y
(d)
A
B
C
Y

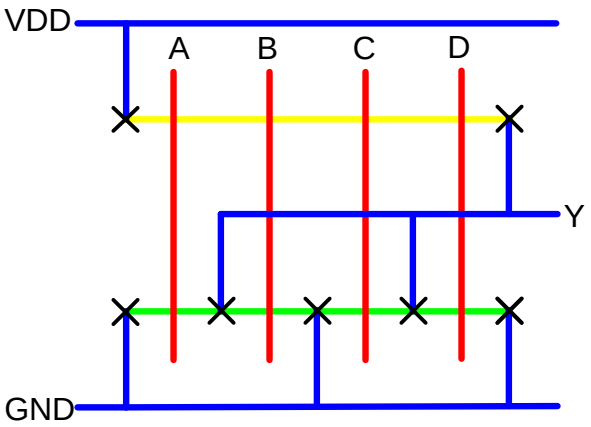
1.6


C
$\overline{C}$
B
$\overline{B}$
B
A
$\overline{A}$
Y
A
$\overline{A}$
B
$\overline{B}$
B
C
$\overline{C}$

1.7

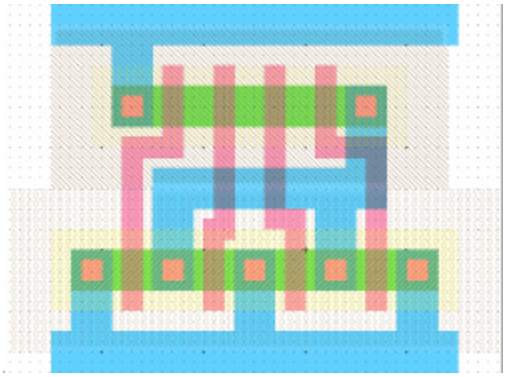


1.8

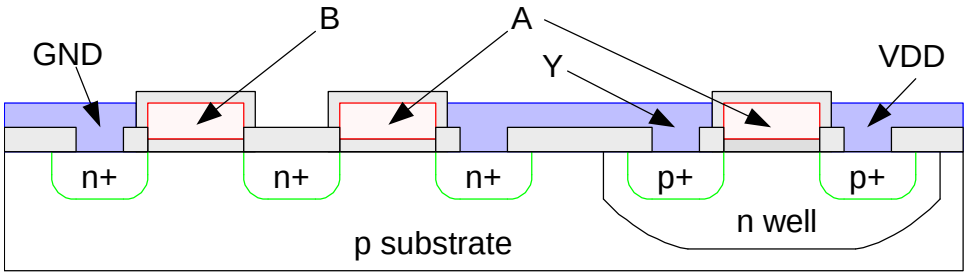


1.9 The minimum area is 5 tracks by 5 tracks (40 λ x 40 λ = 1600 λ^2).

1.10 The layout is 40 λ x 40 λ if minimum separation to adjacent metal is considered, exactly as the track count estimated.

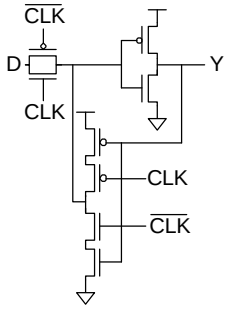


1.11

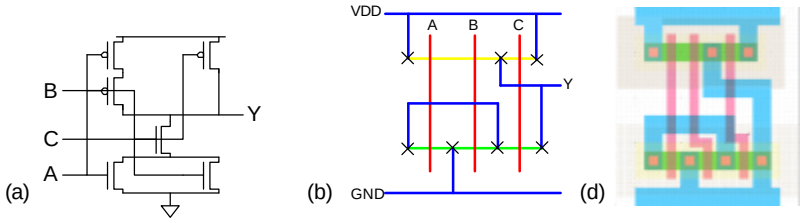


1.12 5 tracks wide by 6 tracks tall, or 1920 λ^2.

1.13 This latch is nearly identical save that the inverter and transmission gate feedback has been replaced by a tristate feedaback gate.



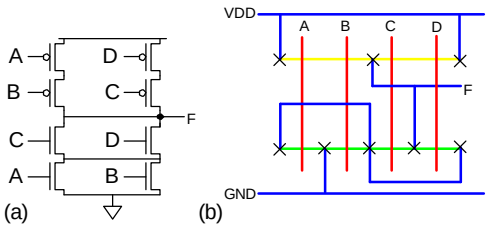
1.14



(c) 4 x 6 tracks = 32 λ x 48 λ = 1536 λ^2.

(e) The layout size matches the stick diagram.

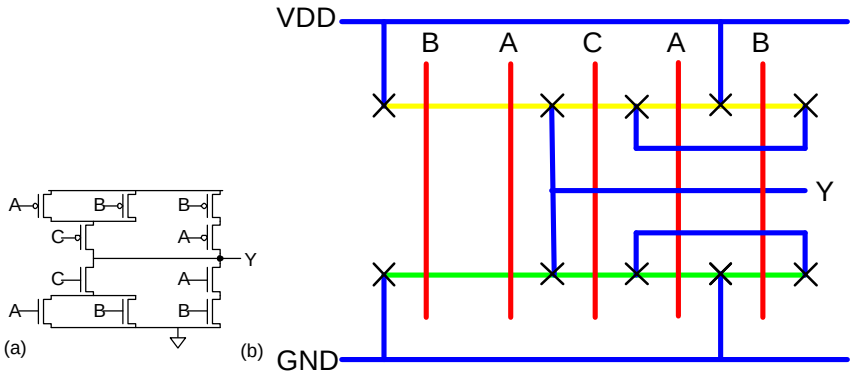
1.15



(c) 5 x 6 tracks = 40 λ x 48 λ = 1920 λ^2. (with a bit of care)

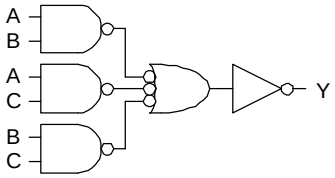
(d-e) The layout should be similar to the stick diagram.

1.16

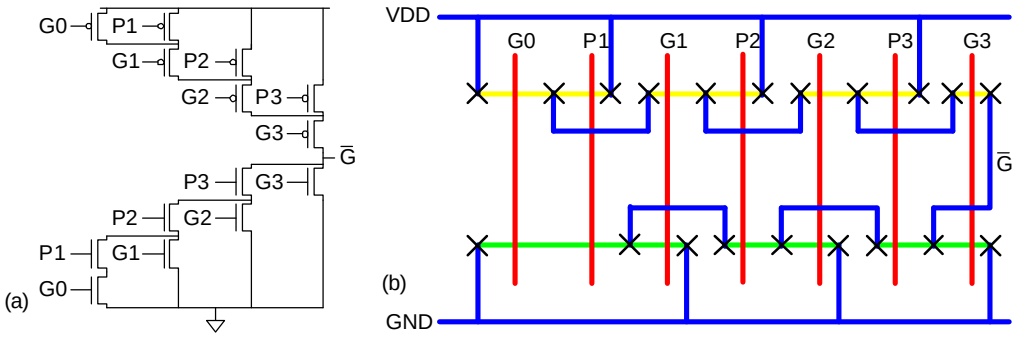


(c) 6 tracks wide x 7 tracks high = (48 x 56) = 2688 λ^2.

1.17 20 transistors, vs. 10 in 1.16(a).



1.18



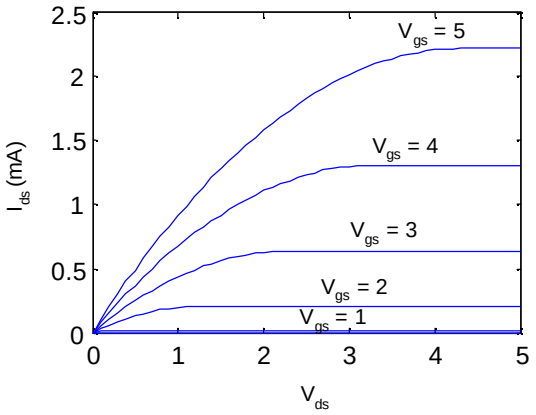
(c) The area of this stick diagram is 11 x 6 tracks = 4224 λ^2 if the polysilicon can be bent.

1.19 The lab solutions are available to instructors on the web.

Chapter 2

2.1

$$b = mC_{ox}\frac{W}{L} = (350)\left(\frac{3.9 \bullet 8.85 \cdot 10^{-14}}{100 \cdot 10^{-8}}\right)\left(\frac{W}{L}\right) = 120\frac{W}{L}mA/V^2$$



2.2 In (a), the transistor sees V_{gs} = V_{DD} and V_{ds} = V_{DS}. The current is

$$I_{DS1} = \frac{b}{2}\left(V_{DD} - V_t - \frac{V_{DS}}{2}\right)V_{DS}$$

In (b), the bottom transistor sees V_{gs} = V_{DD} and V_{ds} = V_1. The top transistor sees V_{gs} = V_{DD} - V_1 and V_{ds} = V_{DS} - V_1. The currents are

$$I_{DS2} = b\left(V_{DD} - V_t - \frac{V_1}{2}\right)V_1 = b\left((V_{DD} - V_1) - V_t - \frac{(V_{DS} - V_1)}{2}\right)(V_{DS} - V_1)$$

Solving for V_1, we find

$$V_1 = (V_{DD} - V_t) - \sqrt{(V_{DD} - V_t)^2 - \left(V_{DD} - V_t - \frac{V_{DS}}{2}\right)V_{DS}}$$

Substituting V_1 indo the I_{DS2} equation and simplifying gives I_{DS1} = I_{DS2}.

2.3 The body effect does not change (a) because V_{sb} = 0. The body effect raises the threshold of the top transistor in (b) because V_{sb} > 0. This lowers the current through the series transistors, so I_{DS1} > I_{DS2}.

2.4 $C_{\text{permicron}}$ = $\varepsilon L/t_{ox}$ = 3.9 * 8.85e-14 F/cm * 90e-7 cm / 16e-4 µm= 1.94 fF/µm.

2.5 The minimum size diffusion contact is 4 x 5 λ, or 1.2 x 1.5 μm. The area is 1.8 μm^2 and perimeter is 5.4 μm. Hence the total capacitance is

$$C_{db}(0V) = (1.8)(0.42) + (5.4)(0.33) = 2.54\,fF$$

At a drain voltage of VDD, the capacitance reduces to

$$C_{db}(5V) = (1.8)(0.42)\left(1+\frac{5}{0.98}\right)^{-0.44} + (5.4)(0.33)\left(1+\frac{5}{0.98}\right)^{-0.12} = 1.78\,fF$$

2.6 The new threshold voltage is found as

$$f_s = 2(0.026)\ln\frac{2\bullet 10^{17}}{1.45\bullet 10^{10}} = 0.85V$$

$$g = \frac{100\bullet 10^{-8}}{3.9\bullet 8.85\bullet 10^{-14}}\sqrt{2\left(1.6\bullet 10^{-19}\right)\left(11.7\bullet 8.85\bullet 10^{-14}\right)\left(2\bullet 10^{17}\right)} = 0.75V^{1/2}$$

$$V_t = 0.7 + g\left(\sqrt{f_s+4} - \sqrt{f_s}\right) = 1.66V$$

The threshold increases by 0.96 V.

2.7 No. Any number of transistors may be placed in series, although the delay increases with the square of the number of series transistors.

2.8 The threshold is increased by applying a negative body voltage so $V_{sb} > 0$.

2.9 (a) (1.2 - 0.3)2 / (1.2 - 0.4)2 = 1.26 (26%)

(b) $\dfrac{e^{\frac{-0.3}{\sqrt{1.4\bullet 0.026}}}}{e^{\frac{-0.4}{\sqrt{1.4\bullet 0.026}}}} = 15.6$

(c) $v_T = kT/q$ = 34 mV; $\dfrac{e^{\frac{-0.3}{\sqrt{1.4\bullet 0.034}}}}{e^{\frac{-0.4}{\sqrt{1.4\bullet 0.034}}}} = 8.2$; note, however, that the total leakage will normally be higher for both threshold voltages at high temperature.

2.10 The current through an ON transistor tends to decrease because the mobility goes down. The current through an OFF transistor increases because V_t decreases. A chip will operate faster at low temperature.

2.11 The nMOS will be off and will see $V_{ds} = V_{DD}$, so its leakage is

$$I_{leak} = I_{dsn} = b v_T^2 e^{1.8} e^{\frac{-V_t}{n v_T}} = 69\,pA$$

2.12 The question is misworded; it is only true if $n = 1.0$. If the voltage at the intermediate node is x, by KCL:

$$I_{leak} = b v_T^2 e^{1.8} e^{\frac{-V_t}{n v_T}} \left(1 - e^{\frac{-x}{v_T}}\right) = b v_T^2 e^{1.8} e^{\frac{-x - V_t}{n v_T}}$$

Now, solve for x using $n = 1.4$:

$$\left(1 - e^{\frac{-x}{v_T}}\right) = e^{\frac{-x}{n v_T}} \rightarrow x \approx 0.8 v_T$$

Substituting, the current is 0.56 times that of the inverter. If $n = 1.0$, x is about 0.7 v_T and the current is exactly half that of the inverter.

2.13 Assume $V_{DD} = 1.8$ V. For a single transistor with $n = 1.4$,

$$I_{leak} = I_{dsn} = b v_T^2 e^{1.8} e^{\frac{-V_t + h V_{DD}}{n v_T}} = 499\,pA$$

For two transistors in series, the intermediate voltage x and leakage current are found as:

$$I_{leak} = b v_T^2 e^{1.8} e^{\frac{-V_t + h x}{n v_T}} \left(1 - e^{\frac{-x}{v_T}}\right) = b v_T^2 e^{1.8} e^{\frac{h(V_{DD} - x) - V_t - x}{n v_T}}$$

$$e^{\frac{-V_t + h x}{n v_T}} \left(1 - e^{\frac{-x}{v_T}}\right) = e^{\frac{h(V_{DD} - x) - V_t - x}{n v_T}}$$

$$x = 69 \text{ mV}; \; I_{leak} = 69 \text{ pA}$$

In summary, accounting for DIBl leads to more overall leakage in both cases. However, the leakage through series transistors is much less than half of that through a single transistor because the bottom transistor sees a small Vds and much less DIBL. This is called the *stack effect*.

For $n = 1.0$, the leakage currents through a single transistor and pair of transistors are 13.5 pA and 0.9 pA, respectively.

2.14 Peter Pitfall is offering to license to you his patented noninverting buffer circuit shown in Figure 2.38. Graphically derive the transfer characteristics for this buffer. Why is it a bad circuit idea?

*** ugly

2.15 $V_{IL} = 0.3$; $V_{IH} = 1.05$; $V_{OL} = 0.15$; $V_{OH} = 1.2$; $NM_H = 0.15$; $NM_L = 0.15$

2.16 Set the currents through the transistors equal and solve the nasty quadratic for V_{out}. In region B, the nMOS is saturated and pMOS is linear:

$$\frac{b}{2}(V_{in} - V_t)^2 = b\left((V_{in} - V_{DD}) - \frac{(V_{out} - V_{DD})}{2} + V_t\right)(V_{out} - V_{DD})$$

$$V_{out} = (V_{in} + V_t) + \sqrt{(V_{in} + V_t)^2 - (V_{in} - V_t)^2 + V_{DD}(V_{DD} - 2V_{in} - 2V_t)}$$

In region D, the nMOS is linear and the pMOS is saturated:

$$\frac{b}{2}(V_{in} - V_{DD} + V_t)^2 = b\left(V_{in} - V_t - \frac{V_{out}}{2}\right)(V_{out})$$

$$V_{out} = (V_{in} - V_t) - \sqrt{(V_{in} - V_t)^2 - (V_{DD} - V_{in} - V_t)^2}$$

2.17 Either take the grungy derivative for the unity gain point or solve numerically for $V_{IL} = 0.46$ V, $V_{IH} = 0.54$ V, $V_{OL} = 0.04$ V, $V_{OH} = 0.96$ V, $NM_H = NM_L = 0.42$ V.

2.18 The switching point where both transistors are saturated (region C) is found by solving for equal currents:

$$\frac{b_n}{2}(V_{in} - V_{tn})^2 = \frac{b_p}{2}(V_{in} - V_{DD} - V_{tp})^2$$

$$V_{in}^2(b_n - b_p) + V_{in}(-2b_nV_{tn} + 2b_p(V_{DD} + V_{tp})) + (b_nV_{tn}^2 - b_p(V_{DD} + V_{tp})^2) = 0$$

$$V_{in} = \frac{b_nV_{tn} - b_p(V_{DD} + V_{tp}) + (V_{DD} + V_{tp} - V_{tn})\sqrt{b_nb_p}}{b_n - b_p}$$

$$= \frac{V_{DD} + V_{tp} + \sqrt{\frac{b_n}{b_p}}V_{tn}}{1 + \sqrt{\frac{b_n}{b_p}}}$$

The output voltage in region B is found by solving

$$\frac{b_n}{2}\left(V_{\text{in}}-V_{tn}\right)^2 = b_p\left(\left(V_{\text{in}}-V_{DD}\right)-\frac{\left(V_{\text{out}}-V_{DD}\right)}{2}-V_{tp}\right)\left(V_{\text{out}}-V_{DD}\right)$$

$$V_{\text{out}} = \left(V_{\text{in}}-V_{tp}\right)+\sqrt{\left(V_{\text{in}}-V_{tp}\right)^2-\frac{b_n}{b_p}\left(V_{\text{in}}-V_{tn}\right)^2+V_{DD}\left(V_{DD}-2V_{\text{in}}+2V_{tp}\right)}$$

and the output voltage in region D is

$$\frac{b_p}{2}\left(V_{\text{in}}-V_{DD}-V_{tp}\right)^2 = b_n\left(V_{\text{in}}-V_{tn}-\frac{V_{\text{out}}}{2}\right)\left(V_{\text{out}}\right)$$

$$V_{\text{out}} = \left(V_{\text{in}}-V_{tn}\right)-\sqrt{\left(V_{\text{in}}-V_{tn}\right)^2-\frac{b_p}{b_n}\left(V_{DD}-V_{\text{in}}+V_{tp}\right)^2}$$

2.19 Take derivatives or solve numerically for the unity gain points: V_{IL} = 0.43 V, V_{IH} = 0.50 V, V_{OL} = 0.04 V, V_{OH} = 0.97 V, NM_H = 0.39, NM_L = 0.47 V.

2.20 The nMOS is in the linear region and the pMOS is saturated. By KCL

$$b_n\left(V_{\text{DD}}-V_{tn}-\frac{\text{V}_{\text{out}}}{2}\right)V_{\text{out}} = \frac{b_p}{2}\left(V_{DD}+V_{tp}\right)^2$$

$$V_{\text{out}} = \left(V_{DD}-V_{tn}\right)-\sqrt{\left(V_{DD}-V_{tn}\right)^2-\frac{b_p}{b_n}\left(V_{DD}+V_{tp}\right)^2}$$

***see Uyemura p. 343 EQ 9.5 for solution to check

2.21 (a) 0; (b) $2|V_{tp}|$; (c) $|V_{tp}|$; (d) V_{DD} - V_{tn}

2.22 (a) 0; (b) 0.6; (c) 0.8; (d) 0.8

Chapter 3

3.1 First, the cost per wafer for each step and scan. 248nm – number of wafers for four years = 4*365*24*80 = 2,803,200. 157nm = 4*365*24*20 = 700,800. The cost per wafer is the (equipment cost)/(number of wafers) which is for 248nm \$10M/ 2,803,200 = \$3.56 and for 147nm is \$40M/700,800 = \$57.08. For a run through the equipment 10 times per completed wafer is \$35.60 and \$570.77 respectively.

Now for gross die per wafer. For a 300mm diameter wafer the area is roughly 70,650 mm^2 (π*(r^2/A – r/(sqrt(2*A))). For a 50mm^2 die in 90nm, there are 1366 gross die per wafer. Now for the tricky part (which was unspecified in the question and could

cause confusion). What is the area of the 50nm chip? The area of the core will shrink by $(90/50)^2$ = .3086. The best case is if the whole die shrinks by this factor. The shrunk die size is 50*.3086 = 15.43mm^2. This yields 4495 gross die per wafer.

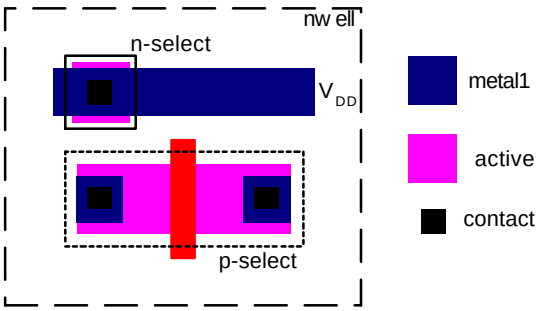
The cost per chip is \$35.60/1413 = \$0.026 and \$570.77/4578 = \$0.127 respectively for 90nm and 50nm. So roughly speaking, it costs \$0.10 per chip more at the 50nm node.

Obviously, there can be variations here. Another way of estimating the reduced die size is to estimate the pad area (if it's not specified as in this exercise) and take that out or the equation for the shrunk die size. A 50mm^2 chip is roughly 7mm on a side (assuming a square die). The I/O pad ring can be (approximately) between 0.5 and 1 mm per side. So the core area might range from 25mm^2 to 36mm^2. When shrunk, this core area might vary from 7.7 to 11.1mm^2 (2.77 and 3.33mm on a side respectively). Adding the pads back in (they don't scale very much), we get die sizes of 4.77 and 4.33 mm on a side. This yield possible areas of 18.7 to 22.8 mm^2, which in turn yields a cost of processing on the stepper of between \$0.155 and \$0.189. This is a rather more pessimistic (but realistic) value.

3.2 The answer to this question is based on the difference in dielectric constant between SiO_2 (k=3.9) and HfO_2 (k=20) [Section 3.4.1.3 page 137]. The oxide thickness would be 2 nm*(20/3.9) = 10.26 nm.

3.3 Polycide – only gate electrode treated with a refractory metal. Salicide – gate and source and drain are treated. The salicide should have higher performance as the resistance of source and drain regions should be lower. (Especially true at RF and for analog functions).

3.4 The pMOS transistor and well contact will be surrounded by the n-well. The pMOS transistor will have active surrounded by p-select while the well contact will have active surrounded by n-select. Contact and metal will be located in the well contact and at the source and drain of the pMOS transistor (and possibly the gate connection).



3.5 This question is poorly worded. The metals that were intended were silver and gold. (This information isn't in the book. The student would have to do a bit of web

searching.) Silver has better conductivity than copper and gold while having poorer conductivity than copper, has good immunity to oxidization. The reason for not using gold or silver is that they both have the property that they can migrate and enter the silicon. This alters CMOS device characteristics in undesirable ways. This question should probably be reworded in any new printing.

3.6 The following table summarizes the pitch requirements assuming contacts meet head to head. A wiring strategy might be to offset contacts, the pitch is then reduced by half the distance that the contact extends beond minimum metal width.

Layer	contact size with metal overlap	metal spacing	pitch
Metal1	4	3	7
Metal3	4	3	7
Metal6	5	5	10

3.7 The uncontacted transistor pitch is = 2*half the minimum poly width + the poly space over active = 2*0.5*2 + 3 = 5 λ. The contacted pitch is = 2*half the minimum poly width + 2 * poly to contact spacing + contact width = 2*0.5*2 + 2*2 + 2 = 8 λ.

The reason for this problem is to show that there is an appreciable difference in gate spacing (and therefore source/drain parasitics) between contacted source and drains and the case where you can eliminate the contact (e.g. in NAND structures). In the main this may not be important but if you were trying too eke out the maximum performance you might pay attention to this. In some advanced processes, the spacing between polysilicon increases to the point that the uncontacted pitch may be the same as the contacted pitch.

3.8 The vertical pitch is divided into three basic segments. First, we have to determine the spacing of the n-transistor to the GND bus. The next segment is defined by the n-transistor to p-transistor spacing. Finally, the p-transistor to V_{DD} bus spacing needs to be determined. (all spacings are center to center).

N-transistor to GND bus

First let us assume minimum metal1 widths. Next, the width of a metal contact is equal to the contact width plus twice the overlap of the metal over the contact = 2 + 2*1 = 4 λ. The minimum width of a transistor is the contact width plus 2*active overlap of contact = 2 + 2*1 = 4 λ (actually the same as a metal1 contact). So the spacing of the n-transistor to the GND bus will be half the GND bus width plus metal spacing plus half of the metal contact width = 0.5*3 + 3 + 0.5*4 = 6.5 λ .

N-transistor to P-transistor spacing

There are two cases: with a polysilicon contact to the gate and without. With the metal-to-polysilicon contact, the spacing will probably be half of the n-transistor

width plus the metal space plus the polysilicon contact width plus the metal space plus half the p-transistor width = 0.5*4 + 3 + 4 + 3 + 0.5*4 = 14 λ. The spacing without a contact is half the n-transistor width plus n-active to p-active spacing plus half the p-transistor width = 0.5*4 + 4 + 0.5*4 = 8 λ. However, the n-well must surround the pMOS transistor by 6 and be 6 away from the nMOS. This sets a minimum pitch of 0.5*4 + 6 + 6 + 0.5*4 = 16 for both cases.

P-transistor to V_{DD} bus

By symmetry, this is also 6.5 λ.

Summary

The total pitch is 2*6.5 + 16 = 29 λ. The total height of the inverter is 35 λ including the complete supply lines and spacing to an adjacent cell. In the case where the V_{DD} and GND busses are not minimum pitch, the vertical pitch and cell height increase appropriately.

In this inverter the substrate connections have not been included. They could be included in the horizontal plane so that the vertical pitch is not affected. If they are included under the metal power busses, the spacing on the transistors to the power busses may be altered. Normally, this is what is done the power bus can be sized up to account for the spacing. This helps power distribution and does not affect the pitch much.

In an SOI process, if the n to p spacing is 2 λ rather than 12 λ, the pitches are 2*6.5 + 14 = 27 λ and 2 * 6.5 + 6 = 19 λ respectively for interior poly connection and not.

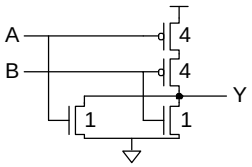
In older standard cell families (two and three level metal processes), the polysilicon contact was often eliminated and the contact to the gate was made above or below the cell in the routing channels. With modern standard cells, all connections to the cells are normally completed within the cell (up to metal2).

3.9 A fuse is a necked down segment of metal (Figure 3.24) that is designed to blow at a certain current density. We would normally set the width of the fuse to the minimum metal width – is this case 0.5 μm. At this width, the maximum current density is 500 μA. At a programming current of 10 times this – 5mA, the fuse should blow reliably. The "fat" conductor connecting to the fuse has to be at least 2.5 μm to carry the fuse current. Actually, the complete resistance from the programming source to the fuse has to be calculated to ensure that the fuse is the where the maximum voltage drop occurs.

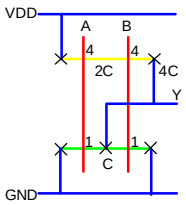
The length of the fuse segment should be between 1 and 2 μm. Why? It's a guess – in a real design, this would be prototyped at various lengths and the reliability of blowing the fuse could be determined for different lengths and different fuse currents. The fabrication vendor may be able to provide process-specific guidelines. One needs enough length to prevent any sputtered metal from bridging the thicker conductors.

Chapter 4

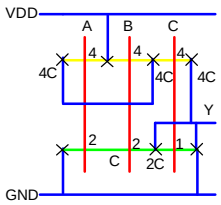
4.1 The rising delay is (R/2)*8C + R*(6C+5hC) = (10+5h)RC if both of the series pMOS transistors have their own contacted diffusion at the intermediate node. More realisitically, the diffusion will be shared, reducing the delay to (R/2)*4C + R*(6C+5hC) = (8+5h)RC. Neglecting the diffusion capacitance not on the path from Y to GND, the falling delay is R*(6C+5hC) = (6+5h)RC.



4.2 The rising delay is (R/2)*2C + R*(5C+5hC) = (6+5h)RC and the falling delay is R*(5C+5hC) = (5+5h)RC.



4.3 The rising delay is (R/2)*(8C) + (R)*(4C + 2C) = 10 RC and the falling delay is (R/2)*(C) + R(2C + 4C) = 6.5 RC. Note that these are only the parasitic delays; a real gate would have additional effort delay.

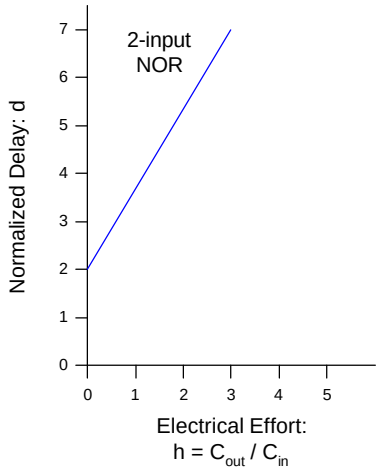


4.4 The output node has 3*n*C. Each internal node has 2*n*C. The resistance through each pMOS is R/*n*. Hence, the propagation delay is

$$t_{pd} = R(3nC) + \sum_{i=1}^{n-1}\left(\frac{iR}{n}\right)(2nC) = \left(n^2 + 2n\right)RC$$

4.5 The slope (logical effort) is 5/3 rather than 4/3. The y-intercept (parasitic delay) is

identical, at 2.

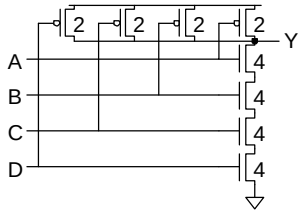


4.6 C_{in} = 12 units. g = 1. $p = p_{inv}$. Changing the size affects the capacitance but not logical effort or parastiic delay.

4.7 The delay can be improved because each stage should have equal effort and that effort should be about 4. This design has imbalanced delays and excessive efforts. The path effort is F = 12 * 6 * 9 = 648. The best number of stages is 4 or 5. One way to speed the circuit up is to add a buffer (two inverters) at the end. The gates should be resized to bear efforts of $f = 648^{1/5}$ = 3.65 each. Now the effort delay is only $D_F = 5f$ = 18.25, as compared to 12 + 6 + 9 = 27. The parasitic delay increases by $2p_{inv}$, but this is still a substantial speedup.

4.8 (a) 4 units. (b) (3/4 units).

4.9 g = 6/3 is the ratio of the input capacitance (4+2) to that of a unit inverter (2 + 1).



4.10 (a) should be faster than (b) because the NAND has the same parasitic delay but lower logical effort than the NOR. In each design, H = 6, B = 1, P = 1 + 2 = 3. For (a), G = (4/3) * 1 = (4/3). $F = GBH$ = 8. $f = 8^{1/2}$ = 2.8. $D = 2f + P$ = 8.6 τ. x = 6C * 1 / f = 2.14C. For (b), G = 1 *(5/3). $F = GBH$ = 10. $f = 10^{1/2}$ = 3.2. $D = 2f + P$ = 9.3 τ. x = 6C * (5/3) / f = 3.16C.

4.11 $D = N(GH)^{1/N} + P$. Compare in a spreadsheet. Design (b) is fastest for $H = 1$ or 5. Design (d) is fastest for $H = 20$ because it has a lower logical effort and more stages to drive the large path effort. (c) is always worse than (b) because it has greater logical effort, all else being equal.

Comparison of 6-input AND gates

Design	*G*	*P*	*N*	*D* (*H*=1)	*D* (*H*=5)	*D* (*H*=20)
(a)	8/3 * 1	6 + 1	2	10.3	14.3	21.6
(b)	5/3 * 5/3	3 + 2	2	8.3	12.5	19.9
(c)	4/3 * 7/3	2 + 3	2	8.5	12.9	20.8
(d)	5/3 * 1 * 4/3 * 1	3 + 1 + 2 + 1	4	11.8	14.3	17.3

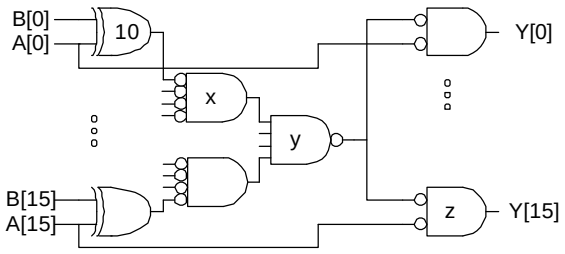
4.12 $H = (64 * 3) / 10 = 19.2$. $B = 32 / 2 = 16$. Compare several designs in a spreadsheet. The five-stage design is fastest, with a path effort of $F = GBH = 683$ and stage effort of $f = F^{1/5} = 3.69$. The gate sizes from end to start are: 192 * 1 / 3.69 = 52; 52 * (4/3) / 3.69 = 18.8; 18.8 * 1 / 3.69 = 5.1; 5.1 * (5/3) / 3.69 = 2.3; 2.3 * 1 / 3.69 = 0.625.

Comparison of decoders

Design	G	P	N	D
NAND5 + INV	7/3	6	2	59.5
INV + NAND5 + INV	7/3	7	3	33.8
NAND3 + INV + NAND2 + INV	20/9	7	4	27.4
INV + NAND3 + INV + NAND2 + INV	20/9	8	5	26.4
NAND3 + INV + NAND2 + INV + INV + INV	20/9	9	6	26.8
NAND2 + INV + NAND2 + INV + NAND2 + INV	64/27	9	6	27.0

4.13 One reasonable design consists of XNOR functions to check bitwise equality, a 16-input AND to check equality of the input words, and an AND gate to choose Y or 0. Assuming an XOR gate has $g = p = 4$, the circuit has $G = 4 * (9/3) * (7/3) * (5/3) = 46.7$. Neglecting the branch on A that could be buffered if necessary, the path has $B = 16$ driving the final ANDs. $H = 10/10 = 1$. $F = GBH = 747$. $N = 4$. $f = 5.23$, high but not unreasonable (perhaps a five stage design would be better). $P = 4 + 4 + 4 + 2 = 14$. $D = Nf + P = 34.9\ \tau = 7$ FO4 delays. $z = 10 * (5/3) / 5.23 = 3.2$; $y = 16 *$

z * (7/3) / 5.23 = 22.8; $x = y$ * (9/3) / 5.23 = 13.1.



4.14 t_{pd} = 76 ps, 72 ps, 67 ps, 70 ps for the XL, X1, X2, and X4 NAND2 gates, respectively. The XL gate has slightly higher parasitic delay because the wiring capacitance is a greater fraction of the total. It also has a slightly higher logical effort, possibly because stray wire capacitance is counted in the input capacitance and forms a greater fraction of the total C_{in}.

4.15 Using average values of the intrinsic delay and K_{load}, we find d_{abs} = (0.029 + 4.55*C_{load}) ns. Substituting $h = C_{load}/C_{in}$, this becomes d_{abs} = (0.029 + 0.020h) ns. Normalizing by τ, d = 1.65h + 2.42. Thus the average logical effort is 1.65 and parasitic delay is 2.42.

4.16 X2: g = 1.55; p = 2.14. X4: g = 1.56; p = 2.17. The logical efforts are about 6% lower and parasitic delay 12% lower than in the X1 gate. Because of differing layout parasitics and slightly nonlinear I vs. W behavior, the parasitic delay and logical effort are not entirely independent of transistor size. However, the variations with size are small enough that the model is still useful.

4.17 g = 1.47, p = 3.08. The parasitic delay is substantially higher for the outer input (B) because it must discharge the internal parasitic capacitance. The logical effort is slightly lower for reasons discussed in Section 6.2.1.3.

4.18 Parasitic delay decreases relative to the estimates when sharing is considered, but increases when the internal diffusion and wire capacitance are considered.

4.19 NAND2: g = 5/4; NOR2: g = 7/4. The inverter has a 3:1 P/N ratio and 4 units of capacitance. The NAND has a 3:2 ratio and 5 units of capacitance, while the NOR has a 6:1 ratio and 7 units of capacitance.

4.20 NAND: g = (μ + k) / (μ + 1); NOR: g = (μk + 1) / (μ + 1). As μ increases, NOR gates get worse compared to NAND gates because the series pMOS devices become more expensive.

4.21 d = (4/3) * 3 + 2 = 6 τ = 1.2 FO4 inverter delays.

4.22 d = (4/3) * 3 + 2 * 0.75 = 5.5 τ = for p_{inv} = 0.75; g = 6.5 τ FO4 delays for p_{inv} = 1.25. The FO4 inverter delays are 4.75 and 5.25 τ, respectively. Hence, the delays are 1.16 and 1.24 FO4 delays, respectively, only a 4% change in normalized delay for a 25% change in parasitics. Hence, delay measured in FO4 delays is relatively insensi-

tive to variations in parasitics from one process to another.

4.23 The adder delay is 6.6 FO4 inverter delays, or about 133 ps in the 70 nm process.

4.24 F = (10 pF / 20 fF) = 500. $N = \log_4 F = 4.5$. Use a chain of four inverters with a stage (5 would also wor, but would produce the opposite polarity). $D = 4F^{1/4} + 4 = 22.9\ \tau = 4.58$ FO4 delays.

4.25 If the first upper inverter has size x and the lower 100-x and the second upper inverter has the same stage effort as the first (to achieve least delay), the least delays are: $D = 2(300/x)^{1/2} + 2 = 300/(100\text{-}x) + 1$. Hence $x = 49.4$, $D = 6.9\ \tau$, and the sizes are 49.4 and 121.7 for the upper inverters and 50.6 for the lower inverter. Such circuits are called *forks* and are discussed in depth in [Sutherland99].

4.26 The clock buffer from Exercise 4.25 is an example of a 1-2 fork. In general, if a 1–2 fork has a maximum input capacitance of C_1 and each of the two legs drives a load of C_2, what should the capacitance of each inverter be and how fast will the circuit operate? Express your answer in terms of p_{inv}.

Let the sizes be x and y in the 2-stage path and C_1 - x in the 1-stage path.

$$D = 2\sqrt{\tfrac{C_2}{x}} + 2p_{\text{inv}} = \frac{C_2}{C_1 - x} + p_{\text{inv}}$$

*** doesn't seem to have any closed-form solution

4.27 $P = aCV^2f = 0.1 * (150e^{-12} * 70) * (0.9)^2 * 450e^6 = 0.38$ W.

4.28 Dynamic power consumption will go down because it is quadratically dependent on V_{DD}. Static power will go up because subthreshold leakage is exponentially dependent on V_t.

4.29 Simplify using $V_{DD} >> v_T$:

(a)

$$I_1 = I_{ds0}e^{\frac{-V_t}{vT}}\left[1 - e^{\frac{-V_{DD}}{vT}}\right] \approx I_{ds0}e^{\frac{-V_t}{vT}}$$

$$I_2 = I_{ds0}e^{\frac{-V_t}{vT}}\left[1 - e^{\frac{-x}{vT}}\right] = I_{ds0}e^{\frac{-V_t - x}{vT}}\left[1 - e^{\frac{-V_{DD}+x}{vT}}\right]$$

$$I_2 \approx I_1\left[1 - e^{\frac{-x}{vT}}\right] = I_1 e^{\frac{-x}{vT}}$$

$$1 - e^{\frac{-x}{vT}} = e^{\frac{-x}{vT}} \Rightarrow e^{\frac{-x}{vT}} = \tfrac{1}{2} \Rightarrow I_2 / I_1 = 1/2$$

(b) Increasing η increases I_1 because the threshold is effectively reduced. The

change is:

(c) Increasing η increases I_2 because the threshold is effectively reduced. However, the relative increase is less than that of I_1. Solve numerically for the change in I_2 to be a factor of 2.25, with x = 83 mV.

d) First solve for x

$$I_2' = I_{ds0} e^{\frac{-V_t + hx}{v_T}} \left[1 - e^{\frac{-x}{v_T}}\right] = I_{ds0} e^{\frac{-V_t - x + h(V_{DD} - x)}{v_T}} \left[1 - e^{\frac{-(V_{DD} - x)}{v_T}}\right]$$

(The last term is approximately unity)

$$= I_1 e^{\frac{hx}{v_T}} \left[1 - e^{\frac{-x}{v_T}}\right] \approx I_1 e^{\frac{-x}{v_T}} e^{\frac{h(V_{DD} - x)}{v_T}}$$

$$1 = e^{\frac{-x}{v_T}} \left[e^{\frac{h(V_{DD} - 2x)}{v_T}} + 1\right]$$

(Assume $x << V_{DD}$)

$$x = v_T \ln\left(1 + e^{\Delta}\right) \approx \Delta v_T$$

Then substitute x to find the relative currents in stacked vs. unstacked transistors:

$$I_2' = I_1 e^{\frac{hx}{v_T}} \left[1 - e^{\frac{-x}{v_T}}\right] = I_1 e^{h\Delta} \left[1 - e^{-\Delta}\right]$$

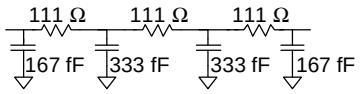
$$\frac{I_2'}{I_2} \approx 2e^{h\Delta}$$

$$\frac{I_2'}{I_1'} \approx e^{(h-1)\Delta}$$

(e) When DIBL is significant, we see from (d) that the current in stacked transistors is exponentially smaller because the bottom transistor sees a small drain voltage and

thus much less DIBL than a single transistor.

4.30 The wire width is 1.2 µm so the wire is 5000 µm/1.2 µm = 4167 squares in length. The total resistance is (0.08 Ω/sq)•(4167 sq) = 333 Ω. The total capacitance is (0.2 fF/µm)•(5000 µm) = 1 pF.



4.31 A unit inverter has a 4 λ = 0.36 µm wide nMOS transistor and an 8 λ = 0.72 µm wide pMOS transistor. Hence the unit inverter has an effective resistance of (2.5 kΩ•µm)/(0.36 µm) = 6.9 kΩ and a gate capacitance of (1.2 µm + 2.4 µm)•(2 fF/µm) = 7.2 fF. The Elmore delay is t_{pd} = (690 Ω)•(500 fF) + (690 Ω + 330 Ω)•(500 fF + 14 fF) = 0.72 ns.

4.32 $R = 0.05^{*}l/W$; $C = l^{*}C(W, S)$; $W + S$ = 1000 nm. $C(W, S)$ is found from Table 4.8. The C_{adj} term is doubled if the adjacent bits might switch in the opposite direciton. If neighbors are not switching, choose S = 320 nm and W = 680 nm. If neighbors are switching, choose S = 500 nm and W = 500 nm. In the first case, resistance dominates so the wide wire is fastest. In the second case, the coupling capacitance is exacerbated by the switching neighbors, so increasing the spacing is most useful.

4.33 The Elmore delay of each segment is

$$t_{pd-seg} = \frac{R}{W}\left(\frac{C_w l}{N} + C'W\right) + \left(\frac{R_w l}{N}\right)\left(\frac{C_w l}{2N} + C'W\right)$$

The total delay is N times greater:

$$t_{pd} = NRC' + L\left(R_w C'W + \frac{RC_w}{W}\right) + L^2\frac{R_w C_w}{2N}$$

Take the partial derivatives with respect to N and W and set them to 0 to minimize delay:

$$\frac{\partial t_{pd}}{\partial N} = RC' - l^2\frac{R_w C_w}{2N^2} = 0 \Rightarrow N = l\sqrt{\frac{R_w C_w}{2RC'}}$$

$$\frac{\partial t_{pd}}{\partial W} = l\left(R_w C' - \frac{RC_w}{W^2}\right) = 0 \Rightarrow W = \sqrt{\frac{RC_w}{R_w C'}}$$

Using these gives a delay per unit length of

$$\frac{t_{pd}}{l} = \left(2+\sqrt{2}\right)\sqrt{RC'R_wC_w}$$

4.34 The Elmore delay of each segment now includes the delay of the extra buffer:

$$t_{pd-seg} = kRC' + \frac{R}{kW_1}\left(\frac{C_w l}{N} + C'W_1\right) + \left(\frac{R_w l}{N}\right)\left(\frac{C_w l}{2N} + C'W_1\right)$$

The total delay is N times greater:

$$t_{pd} = NRC'\left(k+\frac{1}{k}\right) + l\left(R_wC'W_1 + \frac{RC_w}{kW_1}\right) + l^2\frac{R_wC_w}{2N}$$

Take partial derivatives with respect to *N*, *W*, and *k* and set them to 0 to minimize delay

$$\frac{\partial t_{pd}}{\partial N} = RC'\left(k+\frac{1}{k}\right) - l^2\frac{R_wC_w}{2N^2} = 0 \Rightarrow N = l\sqrt{\frac{R_wC_w}{2RC'\left(k+\frac{1}{k}\right)}}$$

$$\frac{\partial t_{pd}}{\partial W} = l\left(R_wC' - \frac{RC_w}{kW_1^2}\right) = 0 \Rightarrow W_1 = \sqrt{\frac{RC_w}{kR_wC'}}$$

$$\frac{\partial t_{pd}}{\partial k} = NRC' - \frac{1}{k^2}\left(NRC' + l\frac{RC_w}{W_1}\right) = 0 \Rightarrow k = \sqrt{1+\frac{lC_w}{NW_1C'}}$$

Note that this implies the first inverter is a factor of $\sqrt{k}$ smaller and the second is a factor of $\sqrt{k}$ larger than the single inverter for an inverting repeater. Substituting the equations for *N* and *W* into that for k gives the interesting result

$$k^4 - 4k^2 - 1 = 0 \Rightarrow k = \sqrt{2+\sqrt{5}} = 2.06$$

Substituting *k*, solve for delay per unit length:

$$\frac{t_{pd}}{l} = \left(\frac{2}{\sqrt{k}} + \sqrt{2\left(k+\frac{1}{k}\right)}\right)\sqrt{RC'R_wC_w} = 3.65\sqrt{RC'R_wC_w}$$

4.35 Compute the results with a spreadsheet:

$$D = \left(2+\sqrt{2}\right)\sqrt{R_w C_w (2.5k\Omega)(0.7+1.4fF)}$$

Characteristic velocity of repeated wires

Layer	Pitch (μm)	R_w	C_w	Delay (ps/mm)
1	**0.25**	**0.32**	**210**	**64**
1	**0.50**	**0.16**	**167**	**40**
2	**0.32**	**0.16**	**232**	**47**
2	**0.64**	**0.078**	**191**	**30**
4	**0.54**	**0.056**	**232**	**28**
4	**1.08**	**0.028**	**215**	**19**

4.36 Eight years corresponds to four process generations and a feature size of 180 / (sqrt(2))^4 = 45 nm. With constant field scaling, the gate delay will improve by a factor of four, so the frequency scales to 5.6 GHz. Historically, frequencies have scaled faster than that because aggressive circuit design also reduces the number of gate delays per cycle as well as the delay of each gate.

4.37 The gate delay component scales as S^{-1} to 250 ps. The delay of a repeated wire of reduced thickness scales as $S^{-1/2}$ to 354 ps. The path delay scales to 604 ps, a 66% speedup.

Chapter 5

5.1 Find the average propagation delay of a fanout-of-5 inverter by modifying the SPICE deck in Figure 5.10.

5.2 By what percentage does the delay of Exercise 5.1 change if the input is driven by a voltage step rather than a pair of shaping inverters?

5.3 By what percentage does the delay of Exercise 5.1 change if *X*5, the load on the load, is omitted?

5.4 Find the input and output logic levels and high and low noise margins for an inverter with a 3:1 *P/N* ratio.

5.5 What *P/N* ratio maximizes the smaller of the two noise margins for an inverter?

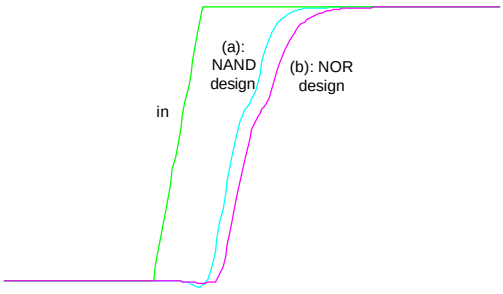
5.6 Generate a set of eight I-V curves like those of Figure 5.16 for nMOS and pMOS transistors in your process.

5.7 The `char.pl` Perl script runs a number of simulations to characterize a process. Look for SPICE models for a newer process on the MOSIS Web site. Use the script to add another column to Table 5.5 for the new process.

5.8 The `charlib.pl` script runs a number of simulations to extract logical effort and parasitic delay of gates in a specified process. Add another column to Table 5.8 for a new process.

5.9 Use the `charlib.pl` script to find the logical effort and parasitic delay of a 5-input NAND gate for the outermost input.

5.10 The deck is listed below. Transistor widths are chosen arbitrarily so that C is equivalent to 5 microns (55 λ). The delays are 130 and 150 ps, respectively (compared to 147 and 158 if τ = 17 ps in this process). Both are faster than expected because the logical efforts of real NANDs and NORs are slightly lower than we have predicted theoretically




```
* Include SPICE models for AMI 0.6u process
.include 'mosistsmc180.sp'

* Define lambda so dimensions can be expressed in lambda
.options scale=0.09u

* Define power supply
.param SUPPLY=1.8
.global vdd gnd

* Define library of logic gates
*   default sizes are given, but may be overridden in subckt calls
*   dimensions are given in lambda thanks to scale

* inverter
.subckt inv a y NW=4 PW=8
  Mn1 y a gnd gnd nmos L=2 W='NW' AS='5*NW' AD='5*NW' PS='NW+10' PD='NW+10'
  Mp1 y a vdd vdd pmos L=2 W='PW' AS='5*PW' AD='5*PW' PS='PW+10' PD='PW+10'
.ends

.subckt nand2 a b y NW=8 P=8
  Mn1 mid a gnd gnd nmos L=2 W='NW' AS='5*NW' AD='1.5*NW' PS='NW+10' PD='3'
```

```
  Mn2 y b mid gnd nmos L=2 W='NW' AS='1.5*NW' AD='5*NW' PS='3' PD='NW+10'
  Mp1 y a vdd vdd pmos L=2 W='PW' AS='5*PW' AD='3*PW' PS='PW+10' PD='6'
  Mp2 y b vdd vdd pmos L=2 W='PW' AS='5*PW' AD='3*PW' PS='PW+10' PD='6'
.ends

.subckt nor2 a b y NW=4 PW=16
  Mp1 mid a vdd vdd pmos L=2 W='PW' AS='5*PW' AD='1.5*PW' PS='PW+10' PD='3'
  Mp2 y b mid vdd pmos L=2 W='PW' AS='1.5*PW' AD='5*PW' PS='3' PD='PW+10'
  Mn1 y a gnd gnd nmos L=2 W='NW' AS='5*NW' AD='3*NW' PS='NW+10' PD='6'
  Mn2 y b gnd gnd nmos L=2 W='NW' AS='5*NW' AD='3*NW' PS='NW+10' PD='6'
.ends

* Define the power supply and input
Vdd vdd gnd 'SUPPLY'
Vin a gnd pulse 0 'SUPPLY' 0.3n 0.1n 0.1n 2n 5n
* Format for pulse:  v1 v2 tdelay trise tfall pulsewidth period

* Path 1: NAND design
X1       a vdd n1 nand2 NW='55/2' PW='55/2'         * NAND2
X2       n1 n3 inv NW='2.1*55/3' PW='2.1*55*2/3'    * 2.1x inverter
X3       n3 n4 inv NW='6*55/3' PW='6*55*2/3' * load

* Path 2: NOR design
X4       a n5 inv NW='55/3' PW='55*2/3'             * inverter1
X5       vdd n6 inv NW='55/3' PW='55*2/3'           * inverter2
X6       n5 n6 n7 nor2 NW='3.2*55/5' PW='3.2*55*4/5'* 3.2x NOR2
X7       n7 n8 inv NW='6*55/3' PW='6*55*2/3' * load

* Run transient analysis (step size and total duration)
.tran 1p 5n

* Plot result
.plot v(a) v(n3) v(n7)

* Measure delay
.measure tpdr1
+        TRIG v(a) VAL='SUPPLY/2' RISE=1
+        TARG v(n3) VAL='SUPPLY/2' RISE=1
.measure tpdf1
+        TRIG v(a) VAL='SUPPLY/2' FALL=1
+        TARG v(n3) VAL='SUPPLY/2' FALL=1
.measure tpd1 param='(tpdr1+tpdf1)/2'

.measure tpdr2
+        TRIG v(a) VAL='SUPPLY/2' RISE=1
+        TARG v(n7) VAL='SUPPLY/2' RISE=1
.measure tpdf2
+        TRIG v(a) VAL='SUPPLY/2' FALL=1
+        TARG v(n7) VAL='SUPPLY/2' FALL=1
.measure tpd2 param='(tpdr2+tpdf2)/2'

.end
```

5.11 Exercise 4.13 asks you to estimate the delay of a logic function. Simulate your design and compare your results to your estimate. Let one unit of capacitance be a minimum-sized transistor.

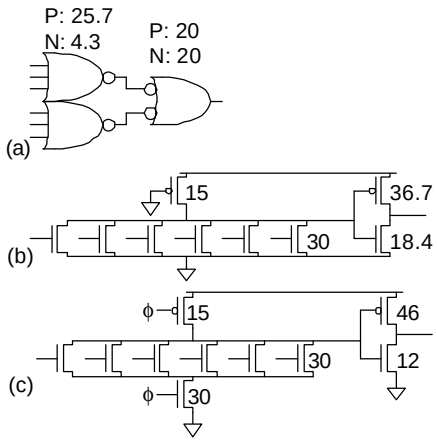
Chapter 6

6.1 In each case, $B = 1$ and $H = (60+30)/30 = 3$.

(a) NOR3 ($p = 3$) + NAND2 ($p = 2$). $G = (7/3)*(4/3) = 28/9$. $F = GBH = 28/3$. $f = F^{1/2} = 3.05$. Second stage size = $90*(4/3)/f = 39$. $D = 2f + P = 11.1$.

(b) Pseudo-nMOS NOR6 ($p = 52/9$) + static INV ($p = 1$). $G = (8/9)*(1) = 8/9$. $F = GBH = 8/3$. $f = F^{1/2} = 1.63$. Second stage size = $90*1/f = 55.1$. $D = 10.0$.

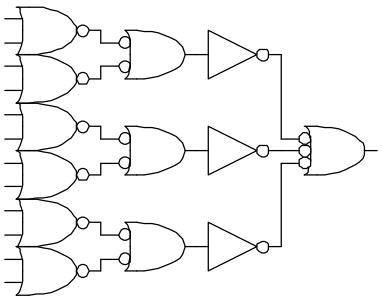
(c) Dynamic NOR6 ($p = 13/3$) + high-skew INV ($p = 5/6$). $G = (2/3)*(5/6) = 10/18$. $F = GBH = 5/3$. $f = F^{1/2} = 1.29$. Second stage size = $90*(5/6)/f = 58$. $D = 7.75$.



6.2 Your mileage may vary. $\tau = 15$ ps.

	rising	falling	average
static	200 ps	157 ps	178 ps = 11.8 t
pseudo-nMOS	93 ps	148 ps	121 ps = 8.1 t
domino	94 ps	n/a	94 ps = 6.3 t

6.3 There are many designs such as NOR2 + NAND2 + INV + NAND3.



6.4 See 6.35.

6.5 (a) For $0 \le A \le 1$, $B = 1$, $I(A)$ depends on the region in which the bottom transistor operates. The top transistor is always saturated because $V_{gs} = V_{ds}$.

$$I(A) = \begin{cases} \left(A - \frac{x}{2}\right)x & x < A \\ \frac{1}{2}A^2 & x \ge A \end{cases} = \tfrac{1}{2}(1-x)^2$$

Thus the bottom transistor is saturated for $A < 1/2$ and linear for $A > 1/2$. Solve for x in each of these two cases:

$$\tfrac{1}{2}A^2 = \tfrac{1}{2}(1-x)^2 \Rightarrow x = 1 - A \qquad A < \tfrac{1}{2}$$

$$\left(A - \tfrac{x}{2}\right)x = \tfrac{1}{2}(1-x)^2 \Rightarrow x = \frac{A + 1 - \sqrt{(A+1)^2 - 2}}{2} \qquad A \ge \tfrac{1}{2}$$

Substituting, we obtain an equation for I vs. A:

$$I(A) = \begin{cases} \frac{1}{2}A^2 & A < \frac{1}{2} \\ \dfrac{A^2 + (1-A)\sqrt{A^2 + 2A - 1}}{4} & A \ge \frac{1}{2} \end{cases}$$

For $0 \le B \le 1$, $A = 1$, the top transistor is always saturated because $V_{gs} = V_{ds}$. The bottom transistor is always linear because $V_{gs} > V_{ds}$. The current is

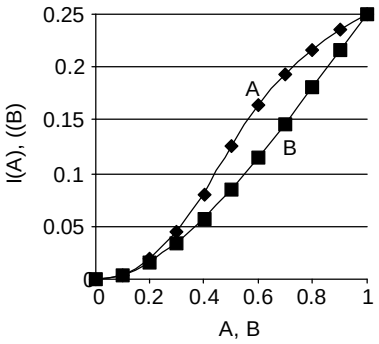
$$I(B) = \tfrac{1}{2}(B - x)^2 = \left(1 - \tfrac{x}{2}\right)x$$

Solve for x and $I(B)$:

$$x = \frac{B+1-\sqrt{(B+1)^2-2B^2}}{2}$$

$$I(B) = \frac{1+(B-1)\sqrt{-B^2+2B+1}}{4}$$

Plotting I vs. A and B, we find that the current is always higher when the lower transistor is switching than when the higher transistor is switching for a given input voltage. This plot may have been found more easily by numerical methods.



(b) The inner input of a NAND gate or any gate with series transistors has grater logical effort than the outer input because the inner transistor provides slightly less current while partially ON. This is because the intermediate node x rises as B rises, providing negative feedback that quadratically reduces the current through the top transistor as it turns ON.

6.6 g = 6/3 at the OR terminals and 5/3 at the AND terminal. p = 7/3.

6.7 ***

Simulate a 3-input NOR gate in your process. Determine the logical effort and parasitic delay from each input.

6.8 t_{pdr} = 0.0313 + 4.5288*0.0042h (in units of ns) = 2.52 + 1.53h (in units of τ)

t_{pdf} = 0.0195 + 2.8429*0.0042h (in units of ns) = 1.57 + 0.96h (in units of τ)

g_u = 1.53; p_u = 2.52; g_d = 0.96; p_d = 1.57

6.9 t_{pdr} = 0.0400 + 4.5253*0.0039h (in units of ns) = 3.22 + 1.42h (in units of τ)

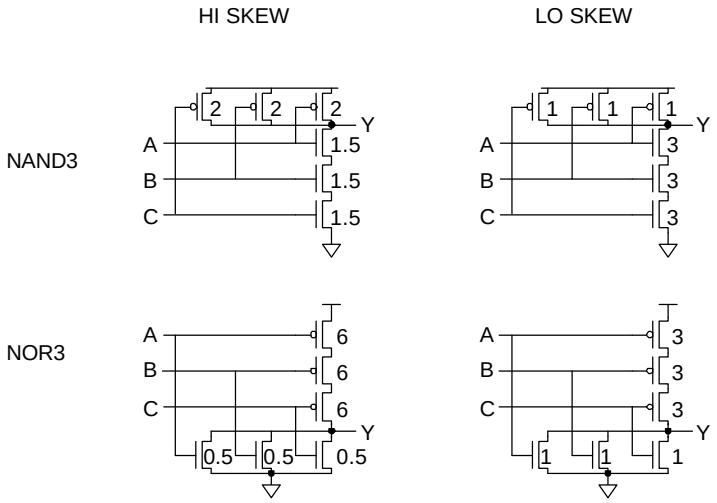
t_{pdf} = 0.0242 + 2.8470*0.0039h (in units of ns) = 1.95 + 0.90h (in units of τ)

g_u = 1.42; p_u = 3.22; g_d = 0.90; p_d = 1.95

As compared to input A, input B has a greater parasitic delay and slightly smaller logical effort. Input B must be the outer input, which must discharge the parasitic capacitance of the internal node, increasing its parasitic delay.

6.10 NAND3: HI-skew: $g_u = 7/6$; LO-skew: $g_d = 4/3$

NOR3: HI-skew: $g_u = 13/6$; LO-skew: $g_d = 4/3$



6.11 HI-skew: $p = 2$, $n = sk$, $g_u = (2 + ks)/3$, $g_d = (2 + ks)/3s$, $g_{avg} = (2 + k + ks + 2/s)/6$

LO-skew: $p = 2s$, $n = k$, $g_u = (2s + k)/3s$, $g_d = (2s + k)/3$, $g_{avg} = (2 + k + 2s + k/s)/6$

6.12 $n_{crit} = 1$. For $g_{crit} = 1.5$, $C_{in} = 4.5$, so $p_{crit} = 4.5\text{-}1 = 3.5$ on the critical input. For unit resistance, $R = 2/p_{crit} + 2*(2/p_{noncrit}) = 1$ -> $p_{noncrit} = 4/(1\text{-}2/p_{crit}) = 28/3$. If $n_{noncrit} = 1/2$, $g_{noncrit} = (p_{noncrit} + n_{noncrit}) / 3 = 3.28$.

6.13 Suppose a P/N ratio of k gives equal rise and fall times. If the pMOS device is of width p and the nMOS of width 1, then we find

$$t_{pdr} \propto \frac{k}{p}(1+p)$$

$$t_{pdf} \propto (1+p)$$

$$t_{pd} \propto \frac{\left(1+\frac{k}{p}\right)(1+p)}{2}$$

$$\frac{\partial t_{pd}}{p} \propto 1 - \frac{k}{p^2} = 0 \Rightarrow p = \sqrt{k}$$

6.14 $\rho(1, p/g)$ is the value of ρ satisfying $p/g + \rho(1 - \ln \rho) = 0$. Suppose we have a path with n_1 stages, a path effort F, and a path parasitic delay P. If we add N-n_1 bufferes of parasitic delay p and logical effort g, the best path delay is

$$D = N\left(Fg^{N-n_1}\right)^{1/N} + P + \left(N - n_1\right)p$$

Differentiate this path delay with respect to N to find the number of stages that minimizes delay. The best stage effort at this number of stages is $\rho(g, p) = (Fg^{N-n1})^{1/N}$. Substituting this into the derivative and simplify to find

$$\frac{\partial D}{\partial N} = \left(Fg^{N-n_1}\right)^{1/N}\left(1 + \frac{n_1 \ln g}{N} - \frac{\ln F}{N}\right) + p = 0$$

$$r(g,p)\left[1 - \ln \frac{r(g,p)}{g}\right] + p = 0$$

Assume the equality we are trying to prove is true and substitute it into the equation above to obtain

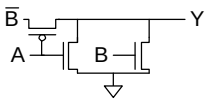
$$r(1, \tfrac{p}{g})\left[1 - \ln r(1, \tfrac{p}{g})\right] + \tfrac{p}{g} = 0$$

This is just the definition of ρ that we began with, so the substitution must have been valid and the equality is proven.

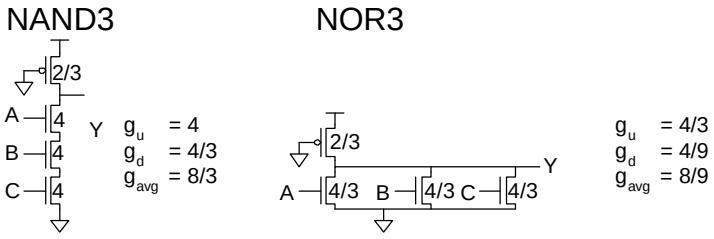
6.15 Accoring to Section 5.2.5 for the TSMC 180 nm process, a P/N ratio of 3.6:1 gives equal rising and falling delays of 84 ps, while a P/N ratio of 1.4:1 gives the minimum average delay of 73 ps, a 13% improvement (not to mention the savings in power and area). Recall that the minima is very flat; a ratio between 1.2:1 and 1.7:1 all produce a 73 ps average delay.

6.16 The ratio for equal rise and fall delays is slower and requires large pMOS transistors. The ratio for minimum average delay results in significantly different rising and falling delays. Some paths are primarily influenced by one of these two delays, so a long rising delay can be problematic. Choosing an intermediate P/N ratio can give average delay nearly equal to the minimum and transistor sizes smaller than those for equal delay while avoiding the very slow rising delay.

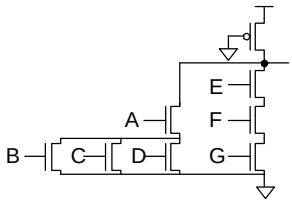
6.17 The 3-transistor NOR is nonrestoring.



6.18



6.19



6.20 Use 8 CMOS inverters driving an 8-input psuedo-nMOS NOR. $F = 1 * (8/9) * 6 = 16/3$. $P = 1 + 34/9$. $D = 2*(16/3)^{1/2} + 43/9 = 9.4\ \tau$.

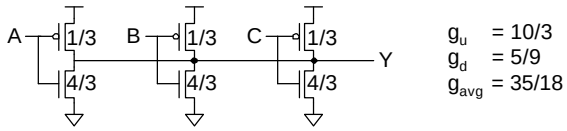
6.21 ***

Simulate a pseudo-nMOS inverter in which the pMOS transistor is half the width of the nMOS transistor. What are the rising, falling, and average logical efforts? What is V_{OL}?

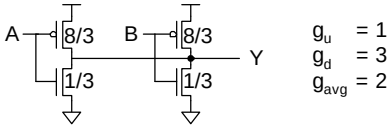
6.22 ***

Repeat Exercise 6.21 in the FS and SF process corners.

6.23 The average logical effort is 35/18, slightly less than 7/3 for a static CMOS NOR3.



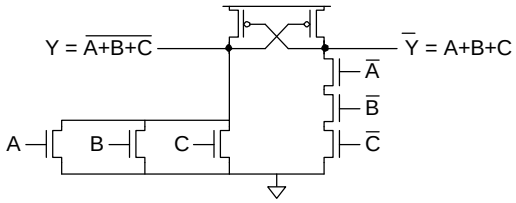
6.24 The average logical effort is 2, more than 4/3 for a static CMOS NOR2. Symmetric NANDs are a poor idea because the pMOS transistors are the slow ones.



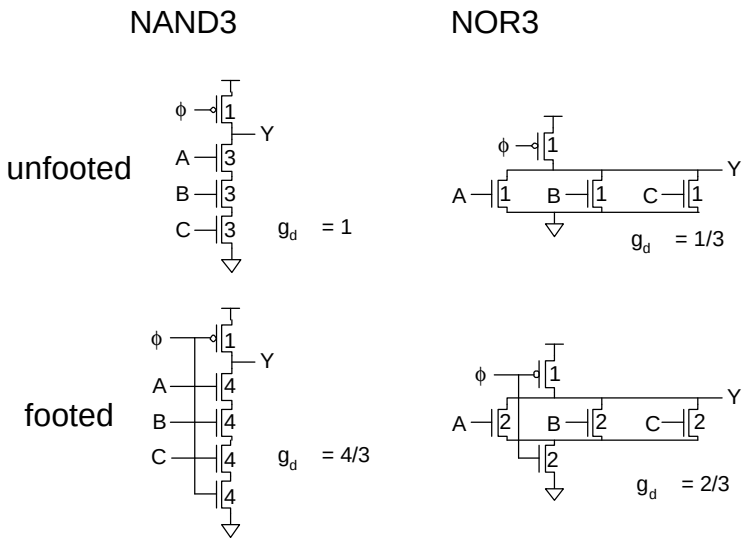
6.25 ***

Compare the average delays of a 2, 4, 8, and 16-input pseudo-NMOS and SFPL NOR gate driving a fanout of 4 identical gates.

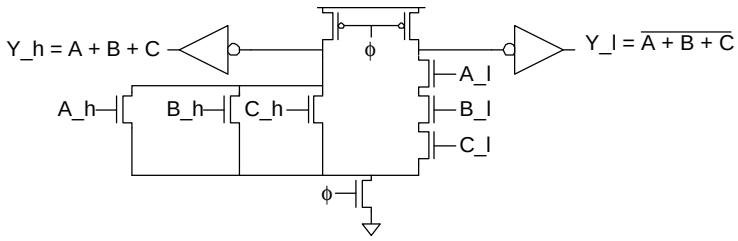
6.26



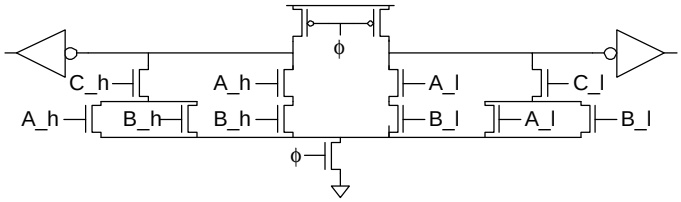
6.27



6.28

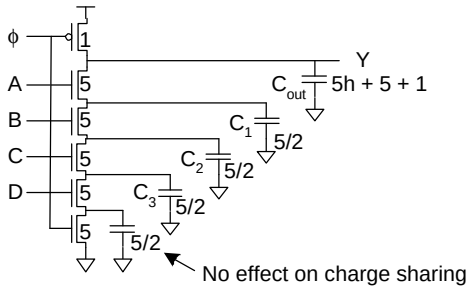


6.29

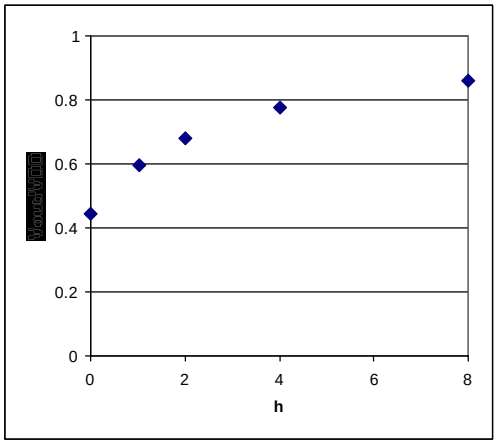


6.30 Your results should be similar to those of Figure 6.37.

6.31 The worst case is when A is low on one cycle, B, C, and D are high, and all the internal nodes become predischarged to 0. Then D falls low during precharge. Then A goes high during evaluation. The NAND has 11 units of capacitance on C_{out} precharged to V_{DD} and 7.5 units of internal capacitance (C_1, C_2, C_3) that will be initially low. The output will thus droop to 11/(11+7.5) V_{DD} = 0.59 V_{DD}.



6.32 The droop is (6+5h)/(6+5h+7.5). Charge sharing is less serious at high fanout.

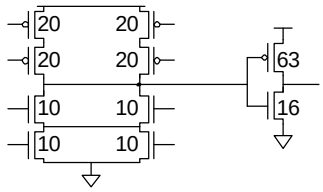


6.33 With a secondary precharge transistor, one of the internal nodes is guaranteed to be high rather than low. Thus 11 + 2.5 = 13.5 units of capcitance are high and 5 units are low, reducing the charge sharing noise to 13.5 / (13.5 + 5) V_{DD} = 0.73 V_{DD}.

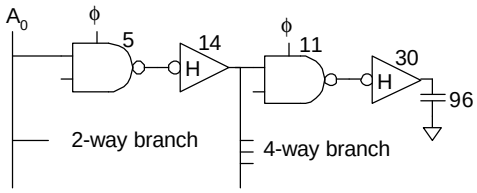
6.34 ***

Perform a simulation of your circuits from Exercise 6.31. Explain any discrepancies.

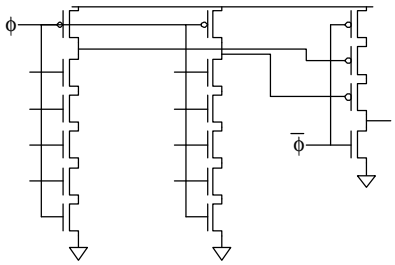
6.35 H = 500 / 30 = 16.7. Consider a two stage design: OR-OR-AND-INVERT + HI-skew INV. G = 2 * 5/6 = 5/3. P = 4 + 5/6 = 29/6. $F = GBH$ = 27.8. $f = F^{1/2}$ = 5.27. $D = 2f + P$ = 15.4 τ. The inverter size is 500 * (5/6) / 5.27 = 79.



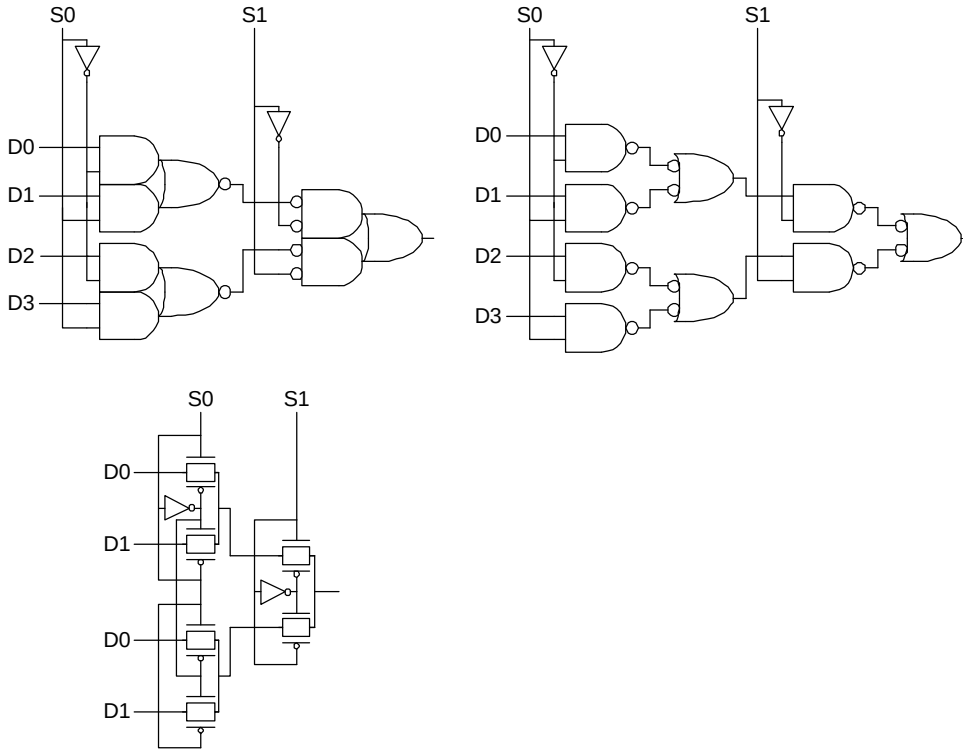
6.36 One design is: Dynamic NAND2 - HI skew INV - Dynamic NAND2 - HI skew INV. G = 1 * (5/6) * 1 * (5/6) = 25/36. F = (25/36) * 8 * 9.6 = 53.3. P = 4/3 + 5/6 + 4/3 + 5/6 = 4.3. $f = F^{1/4}$ = 2.7. $D = 4f + P$ = 15.1 τ.



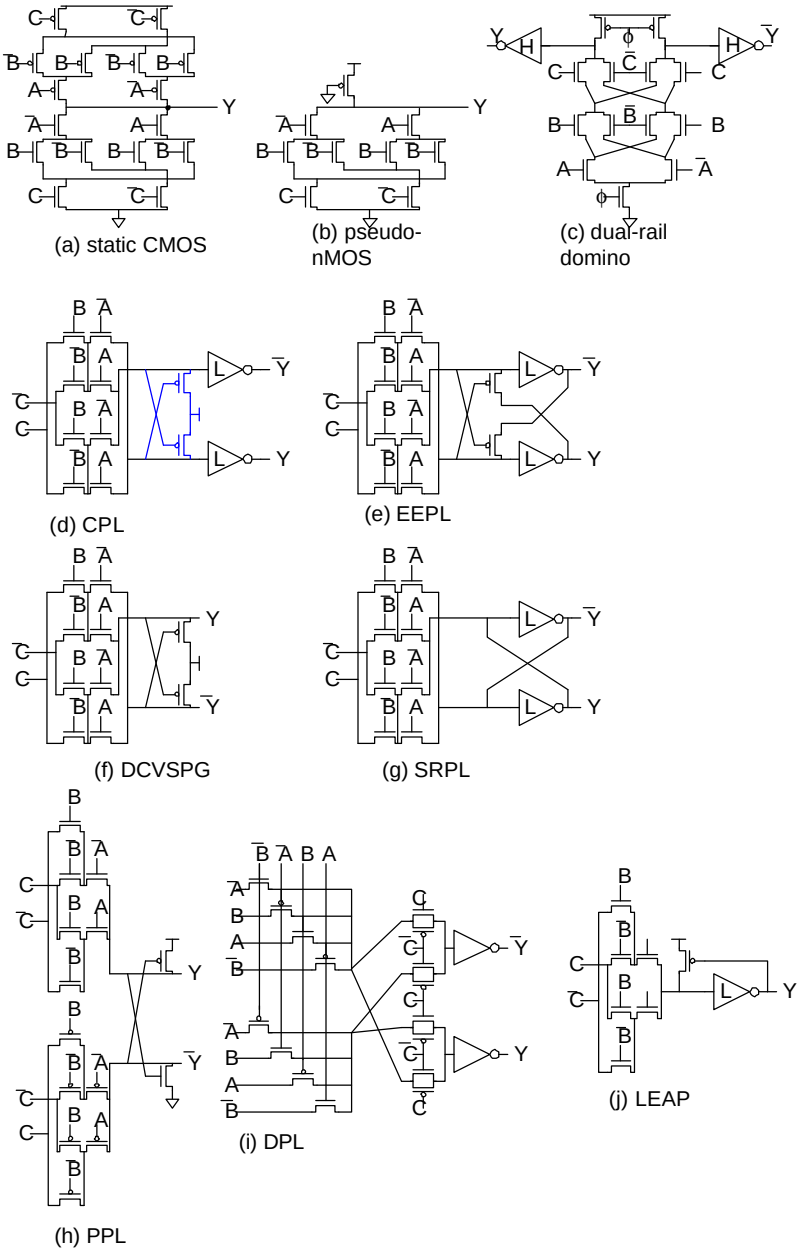
6.37



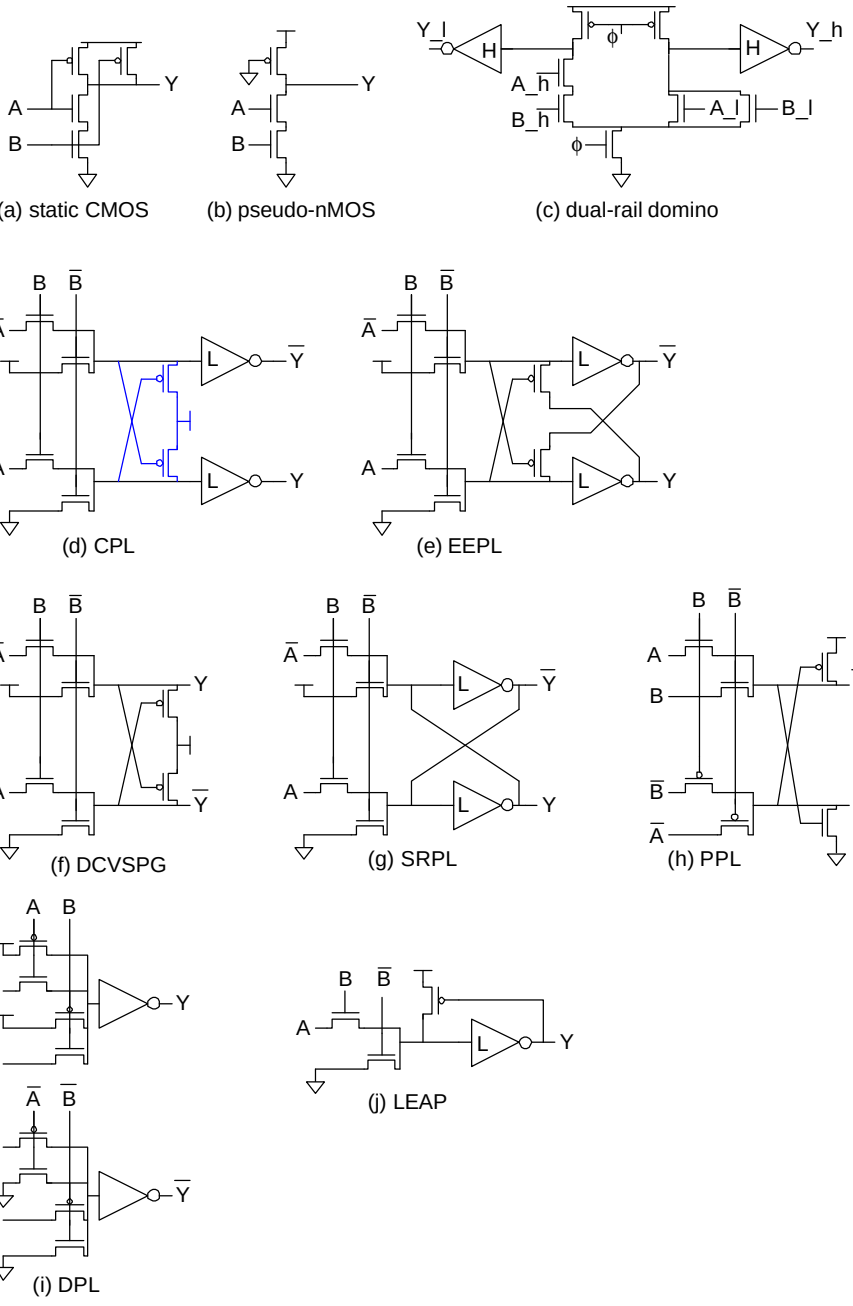
6.38 The static designs with and without complex AAOI gates use 28 and 40 transistors, respectively. The nonrestoring transmission gate design uses 16 transistors, though adding inverters on the inputs would raise that to 24 and make the mux restoring but inverting. Adding an output inverter to make the mux noninverting brings the transistor count to 26.


S0
S1
D0
D1
D2
D3
S0
S1
D0
D1
D2
D3
S0
S1
D0
D1
D0
D1

6.39

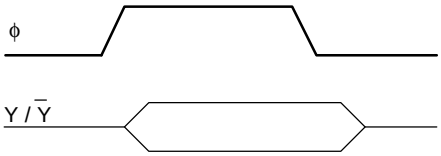

(a) static CMOS
(b) pseudo-nMOS
(c) dual-rail domino
(d) CPL
(e) EEPL
(f) DCVSPG
(g) SRPL
(h) PPL
(i) DPL
(j) LEAP

6.40


A
B
Y
(a) static CMOS
(b) pseudo-nMOS
Y_l
H
A_h
B_h
ϕ
Y_h
A_l
B_l
(c) dual-rail domino
L
(d) CPL
(e) EEPL
(f) DCVSPG
(g) SRPL
(h) PPL
(i) DPL
(j) LEAP

6.41 *** simulation

6.42 When the clock is low, the two outputs equalize at $V_{DD}/2$. When the clock rises, one side pulls down, fully turning ON the pMOS transistor to pull the other side up. This gate saves precharge power relative to dynamic logic because the precharge equalizes the two outputs rather than drawing power from the rail. The partial swing may lead to faster transistions. However, it consumes extra power early in evaluation because of contention between the partially ON pMOS transistor and the ON pulldown stack.



6.43 n/a

Chapter 7

7.1 (a) t_{pd} = 500 - (50 + 65) = 385 ps; (b) t_{pd} = 500 - 2(40) = 420 ps; (c) t_{pd} = 500 - 40 = 460 ps.

7.2 (a) t_{pd} = 500 - (50 + 65 + 50) = 335 ps; (b) t_{pd} = 500 - 2(40) = 420 ps; (c) t_{pd} = 500 - (50 + 25 - 80 + 50) = 455 ps.

7.3 (a) t_{cd} = 30 - 35 = 0; (b) t_{cd} = 30 - 35 = 0; (c) t_{cd} = 30 - 35 - 60 = 0; (d) t_{cd} = 30 - 35 + 80 = 75 ps.

7.4 (a) t_{cd} = 30 - 35 + 50 = 45 ps; (b) t_{cd} = 30 - 35 + 50 = 45 ps; (c) t_{cd} = 30 - 35 + 50 - 60 = 0; (d) t_{cd} = 30 - 35 + 80 + 50 =125 ps.

7.5 (a) t_{borrow} = 0; (b) t_{borrow} = 250 - 25 = 225 ps; (c) t_{borrow} = 250 - 25 - 60 = 165 ps; (d) t_{borrow} = 80 - 25 = 55 ps.

7.6 (a) t_{borrow} = 0; (b) t_{borrow} = 250 - 25 - 50 = 175 ps; (c) t_{borrow} = 250 - 25 - 60 -50 = 115 ps; (d) t_{borrow} = 80 - 25 - 50 = 5 ps.

7.7 If the pulse is wide and the data arrives while the pulsed latch is transparent, the latch contributes its D-to-Q delay just like a regular transparent latch. If the pulse is narrow, the data will have to setup before the earliest skewed falling edge. This is at time $t_{\text{setup}} - t_{\text{pw}} + t_{\text{skew}}$ before the latest rising edge of the pulse. After the rising edge, the latch contributes a clk-to-Q delay. Hence, the total sequencing overhead is $t_{pcq} + t_{\text{setup}} - t_{\text{pw}} + t_{\text{skew}}$.

7.8 $t_{\text{setup-flop}} = t_{\text{setup-latch}} + t_{\text{nonoverlap}}$; $t_{\text{hold-flop}} = t_{\text{hold}} - t_{\text{nonoverlap}}$; $t_{\text{pcq-flop}} = t_{\text{pcq}}$.

7.9 (a) 1200 ps: no latches borrow time, no setup violations. 1000 ps: 50 ps borrowed

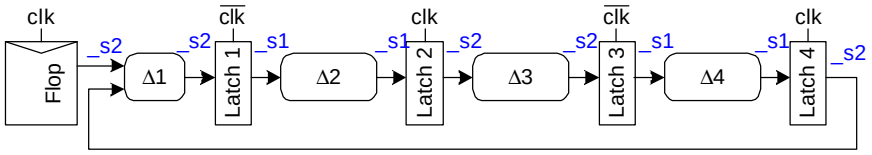
through L1, 130 ps through L2, 80 ps through L3. 800 ps: 150 ps borrowed through L1, 330 ps borrowed through L2, L3 misses setup time.

(b) 1200 ps: no latches borrow time, no setup violations. 1000 ps: 100 ps borrowed through L2, 50 ps through L4. 800 ps: 200 ps borrowed through L2, 200 ps borrowed through L3, 350 ps borrowed through L4, 250 ps borrowed through L1, L2 then misses setup time.

7.10 (a) 700 ps; (b) 825 ps; (c) 1200 ps.

7.11 (a) 700 ps; (b) 825 ps; (c) 1200 ps. The transparent latches are skew-tolerant and moderate amounts of skew do not slow the cycle time.

7.12



7.13 *** simulation

7.14 *** simulation

7.15 t_{pd} = 500 - 2(40) = 420 ps.

7.16 t_{pd} = 500 - 2(40 + 50) = 320 ps.

7.17 t_{pd} = 500 ps. Skew-tolerant domino with no latches has no sequencing overhead.

7.18 t_{pd} = 500 ps. The path can tolerate moderate amounts of skew without degradation.

7.19 t_{borrow} = 125 ps - 50 ps - t_{hold} = 75 ps - t_{hold}.

7.20 t_{borrow} = (0.65-0.25)*500 - 50 ps - t_{hold}= 150 ps - t_{hold}.

7.21 *** simulation

7.22 *** simulation

7.23 Solve for T_c:

$$100 \text{ years} = \frac{T_c e^{\frac{T_c}{54 \text{ ps}}}}{(10^7)(21 \text{ ps})} \Rightarrow T_c = 1811 \text{ ps}$$

7.24 *** simulation

7.25 If the output goes metastable near $V_{DD}/2$, the flip-flop will indeed produce a good high output. However, when it resolves from metastability, the output might suddenly fall low. Because the resolution time can be unbounded, the clock-to-Q delay of the synchronizer is also unbounded. The problem with synchronizers is not that their output takes on an illegal logic level for a finite period of time (all logic gates

do that while switching), but rather that the delay for the output to settle to a correct value cannot be bounded. With high probability it will eventually resolve, but without knowing more about the internal characteristics of the flip-flop, it is dangerous to make assumptions about the probability.

Chapter 8

8.1 Selection of a gate array cell comes down to selecting the number of transistors to place in series in a cell, remembering that if a cell uses less transistors, then the extra transistors are not utilized. We can categorize individual gates by the number of series transistors they require (i.e. an n-strip below a p-strip, as in Figure 8.28). The following table summarizes the usage in particular chip in the exercise:

Cell Type	Series transistors	Circuit	Percentage
D-latch	5	Figure 7.17f with clock inverter	
D-flipflop	10	Two D-latches	
Scannable D-flipflop	15-16	Allowing for input multiplexer	30
4 input gate	4		10
3 input gate	3		10
2 input gate	2		40
buffers	various		10

Clearly if we were to center on a D-flop, and use a five series transistor cell, at least 60% of the gates would waste transistors (2,3,4 input gates). As an aside, a scannable D-register would need three blocks if they were 5 transistor series blocks.

While we can guess, a little bit of analysis might help. The pitch of a contacted transistor is 8λ (Exercise 3.7). (However see Exercise 8.5 where this gets blown out to 14λ if interior poly-contacts are required. For this exercise we'll stick with 8λ.) A break in the active area adds 3λ (Table 3.2 Rule 2.2). So the pitch of various gate arrays is as follows:

Series Transistors	Pitch	
2	2*8+3	19

3	3*8+3	27
4	4*8+3	35
5	5*8+3	43

We can construct a table as follows that charts usage per 1000 gates. A scannable D flipflip is assumed to use 15 series transistors. We ignore buffers.

5 series transistors

Cell	Total Cells	Area Used	Area Wasted
DFF	3*300=900	900*43	0
4 input gates	100	100*43	100*8
3 input gates	100	100*43	100*16
2 input gates	400	400*43	400*24
Total Area		64500	12000
Percentage Wastage			18.6%

4 series transistors

Cell	Total Cells	Area Used	Area Wasted
DFF	4*300=1200	1200*35	1200*8
4 input gates	100	100*35	0
3 input gates	100	100*35	100*8
2 input gates	400	400*35	400*16
Total Area		63000	16800
Percentage Wastage			26.7%

3 series transistors

Cell	Total Cells	Area Used	Area Wasted
DFF	5*300=1500	1500*27	0
4 input gates	100*2	200*27	200*8
3 input gates	100	100*27	0
2 input gates	400	400*27	400*8
Total Area		59400	4800
Percentage Wastage			8.1%

2 series transistors

Cell	Total Cells	Area Used	Area Wasted
DFF	8*300=2400	2400*19	300*8
4 input gates	100*2	200*19	0
3 input gates	100*2	200*19	200*8
2 input gates	400	400*19	0
Total Area		60800	4000
Percentage Wastage			6.6%

Based on this analysis, a three series transistor gate array looks the lowest area.Note that it does not have the best utilization (lowest area wasted), but three series transistors are denser than two because there is less space between transistors.

If we use a Sea-of-gates structure, the pitch is 8 λ but each gate has an extra series transistor (in general) to isolate the gate. This the table looks as follows:

Sea-of-gates

Cell	Total Cells	Area Used	Area Wasted
DFF	300*16 = 4800	4800*8 = 38400	300*8=2400
4 input gates	100*5	500*8=4000	100*8 = 800
3 input gates	100*4	400*8=3200	100*8 = 800
2 input gates	400*3	1200*8=9600	400*8 = 3200
Total Area		55200	7200
Percentage Wastage			13%

This is close to the 3-input case, but takes the guesswork out of estimating gate mixes. This is the reason SOG is widely used. In the end, this problem is looking for some reasoning why one cell size is better than another. Any well reasoned argument is probably acceptable.

8.2 An XOR gate is a 1-bit multiplier. A basic FIR filter is the sum of products of the delayed samples of the input and the coefficients. So for a 288 tap FIR we would have Y(t) = h1*X(t) + h2*X(t+1) + … +h288*X(t+277).

One way to buil this is to use a 288 by 1-bit serial shift register. The output of each register would be connected to one input of a 2 input XOR gate. The other XOR input would be connected to the coefficient. The outputs of the 288 XOR gates then have to be summed. This can be done with a tree of adders (see Chapter 10). We can implement the adder on terms of 3:2 compressors (just a full adder). So 288 bits compress with the first rank of adders to 192 to 128 to 86 to 58 to 39 to 26 to 18, at which point a parallel carry add would be completed.

8.3 [Admittedly, this exercise may require some knowledge of Chapter 11 although the design we use has been presented in Chapter 7. A detailed design is possible with a simulator and process files, but it's OK to just capture the basic principles.]

If we summarize the attributes we need for a control RAM cell for an FPGA, we would like it to be small. In addition, as the RAM cells are dispersed across the chip, it probably would be advisable to design a cell with the lowest wiring overhead. Finally, we want a circuit that is robust and easy to use in an FPGA.

A conventional RAM cell was a write line, a read line and data and complement data lines. Data is read or written using the data lines. To read the RAM cell fairly complicated sense amplifiers are required and there is normally a complicated precharge and timing sequence required to read the cell (Section 11.2.1). We would

prefer a RAM cell that operated with full logic levels.

A single-ended RAM cell that is often used as a register cell is probably the best choice. A typical circuit is shown in Figure 7.17j. This circuit has a single ports for data-in, data-out, write and read. In addition, all signals are full logic levels with the exception of the data-out signal which has to be held high with a pMOS load (or precharged and then read). This is probably OK as the global read operation is only used for testing or to infrequently read out the control RAM contents. It does not have to be fast.

Design starts with the write operation. The switching point of the "input" inverter is a balance between the write zero and one operations. This is achieved by using a single nMOS pass transistor to overwrite a pair of asymmetric inverters. When trying to write a zero, the driving inverter n-transistor and the memory cell write n-transistor have to overcome the p-transistor pullup of the feedback inverter in the memory cell. The circuit is shown below. We can arbitrarily size the weak-feedback inverter so that the pull down circuit triggers the input inverter.

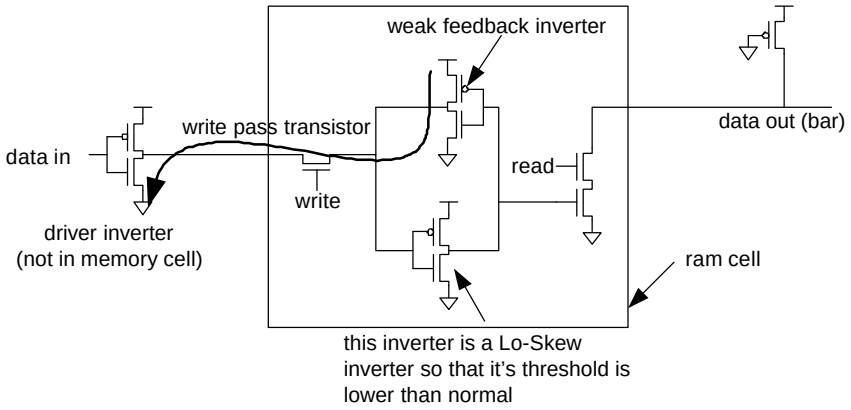


Figure E8.2 – Write Zero operation for single-ended RAM cell

Writing a one is somewhat constrained by the fact that the write n-transistor can only pull up to a threshold below V_{DD} (V_{DD}-V_{tn}). This means that the trip point for the RAM inverter has to be set well below this. This is achieved by having a LO-skewed inverter (Section 2.5.2). This involves sizing the n-transistor in the inverter up until the input switch point is comfortably below the V_{DD}-V_{tn} voltage.

Once the cell can be written, the read operation may be considered. If we use a pMOS load in what is effectively a two input pseudo-nMOS NAND gate or one leg of a multiplexer, the n-transistor pull-downs have to be able to pull the output to near zero when both transistors are turned on. Assuming the pulldown n-transistors are minimum size, this involves lengthening the pMOS pullup until acceptable operation over voltage, temperature and process are achieved.

8.4 FPGA routing blocks cascade the switches used in routing blocks. Therefore every

routing block that a “wire” passes through adds the delay of the switch (Figure 8.23a and Figure 8.24). Transmission gates can reduce the delay for small numbers of cascaded transmission gates but the delay builds as a square law (The square law increase in delay mentioned in Section 4.6.4 also applies here.) A tristate inverter costs a load dependent delay per stage (rather like the repeaters mentioned in Section 4.6.4). So the tradeoff involves determining which on average one uses. Or perhaps a mix of styles might be used. [Footnote: An FPGA company (now defunct) entered the FPGA market based on the fact that they had a patent on using a tristate inverter as a routing switch. It was reasoned that as processors got faster and routing (real wires) got slower, this would be a market edge.]

8.5 [Yikes, a term project!! I think I meant “design in principle”…. Also SUBM rules assumed.]

Exercise 3.8 calculated the vertical pitch for minimum sized n and p-transistors (4λ). Here the pMOS is 6λ, so the vertical pitch (without substrate contacts) would be 29 λ.

Adding a substrate contact alters the n-transistor to VSS and p-transistor to VDD spacing. Taking the n to VSS spacing first, we have a 4λ VSS contact, a 4λ transistor and a 4λ spacing, so the center to center separation is 8λ compared to 6.5 λ without contacts. The p to VDD spacing will be 9λ . So the pitch can be 8+8+8+9 = 33λ.

The horizontal pitch is determined by the figure below.

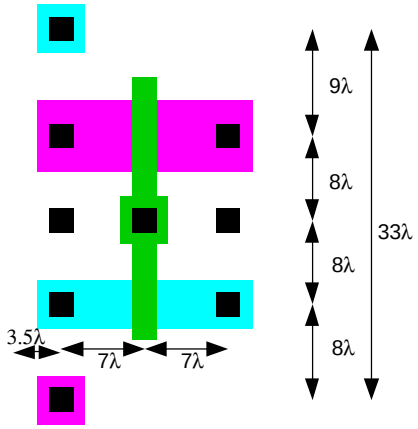


Figure E8.3 Horizontal Standard Cell Pitch

We need to contact the gate and be able to run a vertical metal pitch over the adjacent source and drain. It’s likely that the source and drain require a metal2/metal1 contact as well, so the pitch has to be the contacted pitch. This is 1/2*4 + 1/2*4 + 3 = 7 λ per half-pitch or 14 λ between transistors. At the end of each cell we also leave this space to allow easy abutment (3.5 λ to centre of space from centre of

source/drain contact) .

8.6 This is an exercise that can be widely interpreted. The following is just my approach. Some latitude is needed here for solutions. The main thing to look for is some organization to deal with the problem.

I usually start with subroutines to generate rectangles on a given layer and with particular dimensions in the desired output language. So this might look something like this in the C language:

```
gen_rect (fp, layer,  xlower, ylower, xupper, yupper);
{
/* file_pointer points to a file that has been opened —
this stores the resulting cell. omitting this in C and
using printf allows the output to be presented on the
standard output */
  float xlower, ylower, xupper, yupper;
  char *layer;
  int *fp;
  fprintf (fp, "rect %s %d %d %d %d\n", layer, xlower,
           ylower, xupper, yupper);
};
```

Then routines are written to generate transistors, wires and contacts. Annotations for signal names may also be required.

```
gen_wire (fp, layer, w, x1, y1, x2, y2);
{
  int *fp;
  char *layer;
  float w,x1,y1,x2,y2;
  gen_rect(fp,layer,x1-w/2,y1-w/2,x2+w/2,y2+w/2);
};


gen_contact (fp,type,x,y);
{
  int *fp,*type;
  float x,y;
```

```
  if (char_equal (type, “m1n”)) {
    gen_rect(fp,”METAL1”,x-1.5,y-1.5,x+1.5,y+1.5);
    gen_rect(fp,”ACTIVE”,x-1.5,y-1.5,x+1.5,y+1.5);
    gen_rect(fp,”N_SELECT”,x-2.5,y-2.5,x+2.5,y+2.5);
} } else if (char_equal (type,”m1p”)) {
    gen_rect(fp,”METAL1”,x-1.5,y-1.5,x+1.5,y+1.5);
    gen_rect(fp,”ACTIVE”,x-1.5,y-1.5,x+1.5,y+1.5);
    gen_rect(fp,”P_SELECT”,x-2.5,y-2.5,x+2.5,y+2.5);
} } else if (char_equal (type,”m1poly”)) {
    gen_rect(fp,”METAL1”,x-1.5,y-1.5,x+1.5,y+1.5);
    gen_rect(fp,”POLY”,x-1.5,y-1.5,x+1.5,y+1.5);
  };
  gen_rect(fp,”CONTACT”,x-1,y-1,x+1,y+1);
};

gen_tran (fp,type,w,l,x,y);
{
  int *fp;
  char type,t_type,c_type;
  float w,l,x,y,e;
  e = 2; /* poly gate extension */
  sd = 6; /* SD extension */
  if (char_equal (type, “n”)) {
    t_type = “N_SELECT”;
    c_type = “m1n”;
  } else {
    t_type = “P_SELECT”;
    c_type = “m1p”;
  }
  gen_rect(fp,”ACTIVE”,x-sd,y-w/2,x+sd,y+w/2);
  gen_rect(fp,”N_SELECT”,x-sd-2,y-w/2-2,x+sd+2,y+w/
```

```
                              2+2);
    gen_rect(fp,"POLY",x-l/2,y-w/2-e,x+l/2,y+w/2+e);
    gen_contact(fp,c_type,x-3);
    gen_contact(fp,c_type,x+3);
};
```

Finally generate a "unit" inverter.

```
gen_unit_inv (fp,x,y);
{
  int *fp;
  float x,y;

  /* place transistors, VDD, GND wires, wire up transis
     tors*/
  gen_wire (fp,"METAL1",……)
  …
};
```

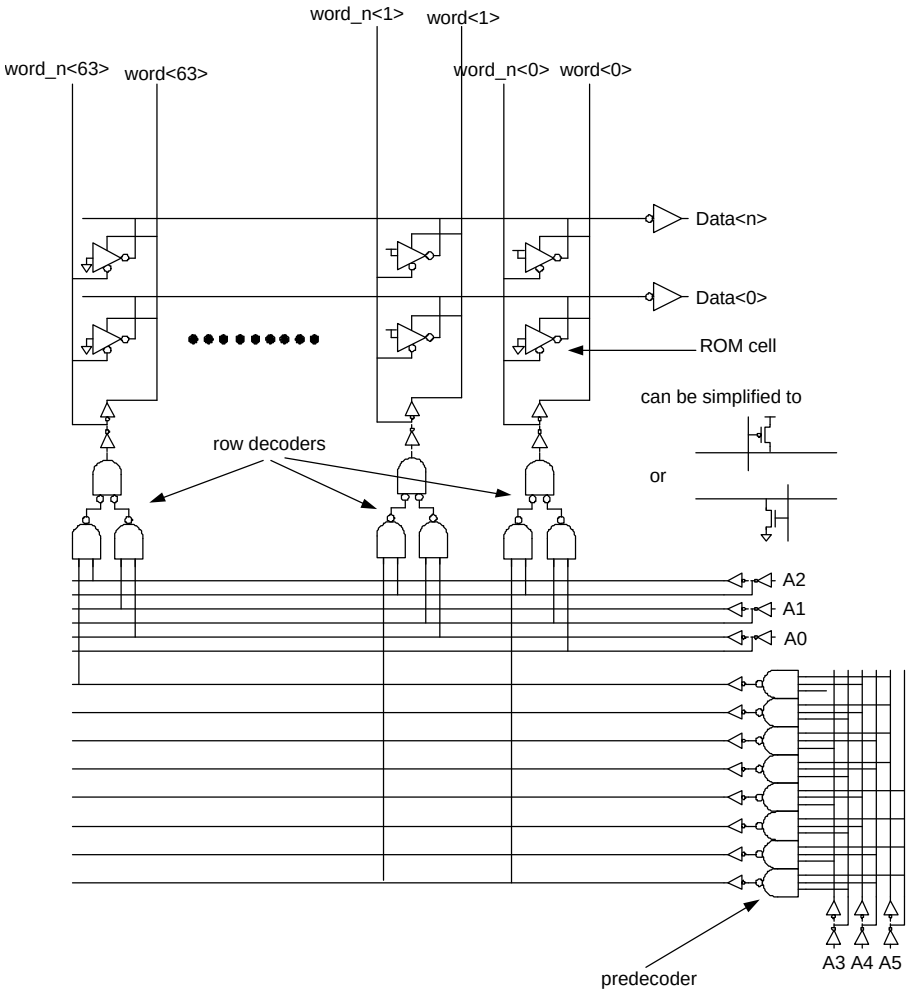
And then replicate.

```
gen_inv (fp,size);
{
  int *fp;
  float x,y;
  int size;

  /* code to iteratively place inverters and connect up
  */
  gen_unit_inv (fp, x, y);
}
```

The above code is only included to indicate an approach.

8.7 This is a little more complicated (than Exercise 8.6). Approaches here will vary widely, so it is fairly pointless to specify code (even though we did do it in the previous example). The main thing would be to look for a credible layout. My approach would be similar to the previous example. Write code for the primitives and then hierarchically build these up to the ROM.



8.8 Similar to Exercise 8.7 except than instead of layout, we are specifying code. Again look for credible HDL.

8.9 Using a standard cell library usually means that only normal logic gates, inverters and tristate buffers are available. The sense amp would just be an inverter. The col-

umn decode address buffers remain the same (inverters). The row decoder has to be implemented in terms of normal logic gates. The horizontal pitch depends on what is used for the ROM cell itself (see next). So we will delay a decision on the row decoder until the ROM cell is decide upon.

The simplest cell that may be used for the ROM cell is a tristate buffer. The tristate buffer has the input connected to either VDD or GND to program the cell. The output is connected to the bit line. The enable and enable_bar are routed from the row decoder (word and word_bar). If you have control over the standard cell library one can eliminate the extraneous transistors. Connection to VDD or GND can be left to the automatic router or completed inside the cell. Remember if this is done, the connection will usually have to be via a diffusion wire so that the gate is not directly to the supply rail (for gate breakdown reasons). For either the full tristate buffer or depleted cell, the horizontal pitch is two transistors, meaning that any cell with two series (or paralleled) transistors will "pitch match" in the vertical plane (assuming transistors are running horizontal).

Now that the ROM cell is decided, the row decoder may be designed. Is essence one "builds down" from the ROM cell. First we need a buffers (inverters) for the word and word_bar signals. These take the first two rows. Then a stack comprised of a two input NOR gate and two, two input NAND gates. The NAND gates decode address bits A0-A2 and one of eight predecoded lines of A3-A5. The predecode lines are located with the address buffers in the lower right.

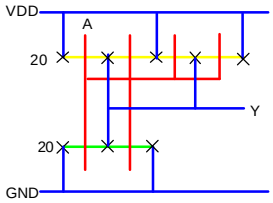
8.10 Power = Voltage*Current. We know the voltage and need to calculate the current. Many design systems allow the current of voltage sources to be measured directly. If you do not have a design system that does, this insert a small resistor in series with the power or ground lead (say 0.1 ohms). The voltage across the resistor then yields the current. Inserting the resistor in the ground lead is easier as the voltage is to ground but may be difficult to get to if the schematic has been design with a "global" ground (i.e. the ground signal does not appear explicitly in the schematic). See also section 5.5.4.

8.11 Another largish exercise. The main portions of the code appear in the text (sans mistakes). The gross test of a working solution at the end of this exercise is a simulation showing a sine wave. The good thing about this is that artefacts are easily spotted by eye. A nice clean sine wave means the design was probably done correctly. The latter part of the problem (comparing with the ROM based design) depends obviously on the student having done that problem.

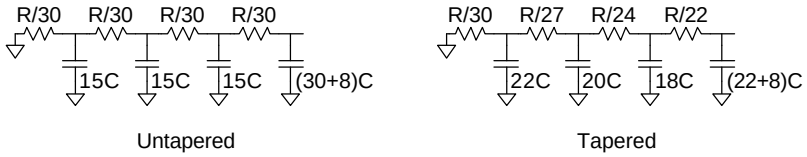
8.12 Here just look for the items discussed in Section 8.6. This as close as it gets to an essay answer…

8.13 Using Equation 8.7, the (yielded) gross die per wafer for the first process is 1500 (1914*.8*.98) and the die cost is $1.47. For the scaled process there are 2227 yielded die (2841*.8*.98) which cost $1.35. So it is probably worth moving considering that the yield probably improves as well (smaller die).

8.14 The order of contacts affects the parasitic delay of the gate. For example, if the GND wire were contacted to the middle of the nMOS pair and the Y wires to the outside, there would be twice as much n-diffusion capacitance.



8.15 RC models of the two circuits are shown below. The Elmore delay of the untapered stack is [15*(1/30) + 15*(2/30) + 15*(3/30) + 38*(4/30)]*RC* = 8.07 *RC*. The Elmore delay of the tapered stack is [22*(1/30) + 20*(1/27 + 1/30) + 18*(1/24 + 1/27 + 1/30) + 30*(1/22 + 1/24 + 1/27 + 1/30)]*RC* = 8.88 *RC*. The untapered design is faster.



8.16 For the untapered gate, the logical effort is 4/3 = 1.33 from any input. The parasitic delays are (38)*(4/30) / 3 = 1.69 from input D, [(38)*(4/30) + 15*(3/30)] / 3 = 2.19 from input C, [(38)*(4/30) + 15*(3/30) + 15*(2/30)] / 3 = 2.52 from input B, and [(38)*(4/30) + 15*(3/30) + 15*(2/30) + 15*(1/30)] / 3 = 2.69 from input A.

For the tapered gate, the effective resistance is R/30 + R/27 + R/24 + R/22 = R/6.35. A unit inverter with the same resistance would have an input capacitance of 3*6.35 = 19.05C. The logical effort is the ratio of the input capacitance of the gate to the input capacitance of an inverter with the same drive. Thus, the logical effort is 22/19.05 = 1.15 from input D, 24/19.05 = 1.26 from input C, 27/19.05 = 1.42 from input B, and 30/19.05 = 1.57 from input A.

The parasitic delays are (30)*(1/6.35) / 3 = 1.57 from D, [30*(1/6.35) + 18*(1/24 + 1/27 + 1/30)] / 3 = 2.25 from C, [30*(1/6.35) + 18*(1/24 + 1/27 + 1/30) + 20*(1/27 + 1/30)] / 3 = 2.72 from B, and [30*(1/6.35) + 18*(1/24 + 1/27 + 1/30) + 20*(1/27 + 1/30) + 22*(1/30)] / 3 = 2.96 from A.

Chapter 9

9.1 Cooling a circuit improves the mobility of the transistors which in turn improves the

speed. Raising V_{DD} has the same effect. These two tests together probably point to a path that is too slow at normal temperature and voltage. Re-simulating the path ensuring to include all parasitics (at especially the slow process corner), should reveal the problem.

9.2 If we take the parameters of Exercise 8.13, a 10 mm * 10 mm die costs $10.05 assuming an 8" wafer, 80% die yield (high!!) and 98% package yield. The package cost is significant at $5. So it is probably best to test at the wafer level.

9.3 Absolutely not! Any discrepancy between a golden model and the design should be tracked down and explained and eliminated. Often small deviations hide much larger problems.

9.4 This is straight from the text – Section 9.5.1.1. A Stuck-at-1 means that a node is shorted to V_{DD}. A Stuck-at-0 means a node is shorted to GND.

9.5 Again straight from the text (pp 590). Figure 9.10 is an example.

9.6 Wires can touch (especially if there is a dust particle over two wires or separation rules are broken) – this leads to shorts.

Wires can neck down (say due to overetching) – this can lead to opens.

Contacts or vias can be faulty leading to opens.

Gate oxide can have pin holes that leads to shorts.

9.7 Right out of the text. Controllability – Section 9.5.3. Observability – Section 9.5.2. Fault Coverage – Section 9.5.4

9.8 A high fault coverage means that the set of test vectors is effectively capable of finding as many faults as possible. Testing costs money, so the smaller a test set is while being effective, the lower the cost of the chip.

9.9 Another question straight out of the book (these are too easy…). Section 9.6.2. Basically, a scan design is implemented by turning all D flip-flops into scannable D flip-flops. This usually involves adding a two input multiplexer to the existing D flip-flop designs that are used (this isn't done manually, but using library elements).

Once scan flip flops are inserted, the task remains to divide the flip-flops into scan chains.

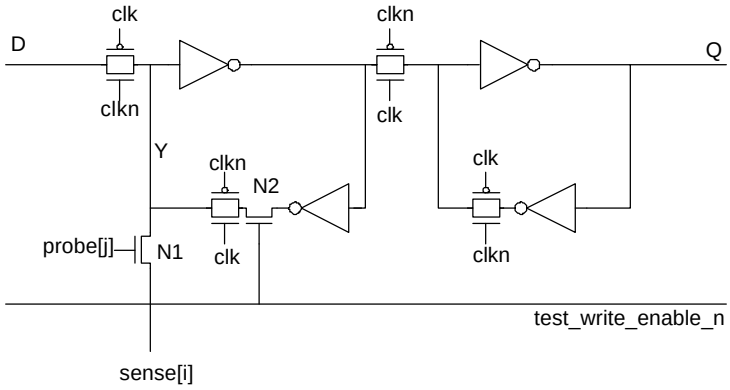
9.10 BIST builds on scan by surrounding logic blocks with a pseudo random sequence generator on the logic inputs and a scan chain on the output of the logic. These two functions (in addition to the normal operation of the flip flops) may be combined into the one structure (Figure 9.23)). BIST can reduce the number of external test vectors needed if it is compatible with the system test methodology. It can also perform high-speed testing with a low-speed tester. It costs area on chip.

9.11 The point that is trying to be illustrated here is that there are some areas where we do not want to encumber a flip-flop with extra circuitry. This is the case for high speed flip-flips used in dividers (irregardless of circuit design). So no scan elements.

Just test by observing the frequency of the MSB of the counter (lowest frequency) with a frequency counter. This is more classed as an analog block.

9.12 This register was featured in the second edition.

Transistors N1 and N2 are added to a regular static D flip-flop. Transistor N2 is used to prevent the master stage of the D flip-flop from writing. Setting signal probe[j] allows node Y to be read or written via signal sense[i]. If test_write_enable_n is true, the cell is read. If test_write_enable_n is false, the cell may be written (providing the D flip-flop master inverter is LO-skewed). Be careful of the single nMOS pass-gates



9.13 Essentially, this is a slice through Figure 9.24. The 16-bit datapath has a 16-bit LFSR on the input and a 16 bit signature analyser on the output. The sequence to test is as follows:

Initialize LFSR (i.e. set flip flops to zero)

Place signature analyzer in “analyze” mode

Cycle LFSR through a “large” number of vectors – can be exhaustive.

Shift signature analyzer out and observe syndrome – check whether it matches the simulated value. If it does your circuit is OK, if not, it’s faulty.

9.14 The data input, address and control (read/write controls and clocks) are muxed with test generators. The test structure for the address can be a counter. The data generator can be a simple logic structure that generates “all 0’s”, “all 1’s” and “alternating 1’s and 0’s”. The control generator generate a simple control sequence. A comparator compares the RAM data with what is expected. Typical operation might be as follows:

Stage 1: Write Data

Set data generator to “all 0’s”

Loop Counter through address range and write data to RAM

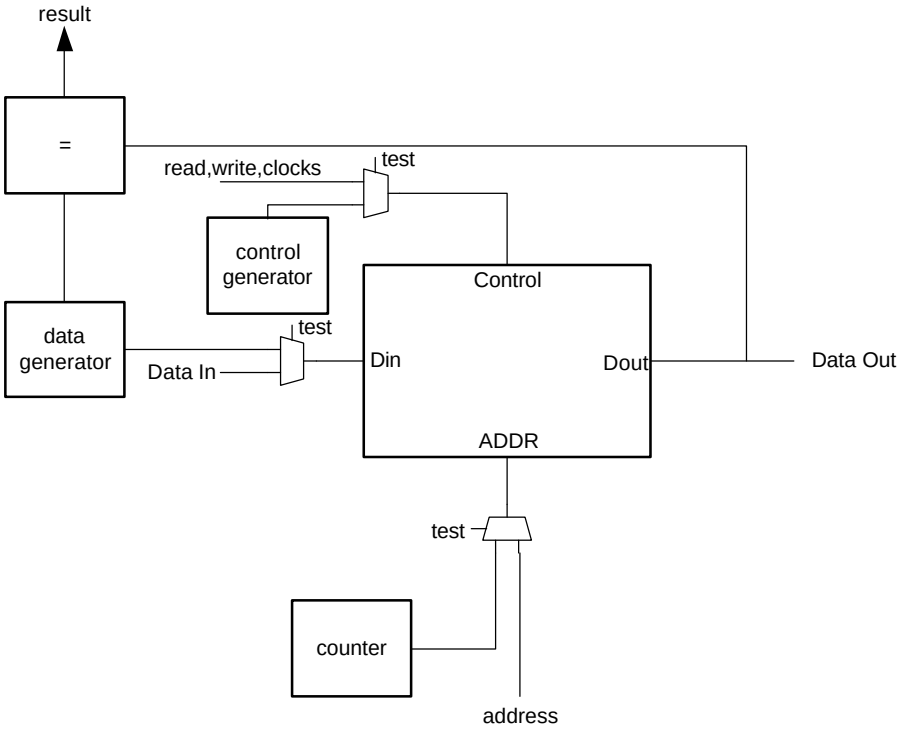
Stage 2: Check RAM

Set data generator to “all 0’s”

Loop Counter through address range and read RAM

Check RAM output at each step

The same would be done for “all 1’s” and “alternating 1’s and 0’s”.



9.15 The software radio consists of an IQ conversion unit, four microprocessors with multipliers and four memories. At the SOC level, we would start by adding the required Wrapper Serial Port (WSP) to each block. The decision then may be made as to whether a Wrapper Parallel Port (WPP) is required. This would depend on whether the intelligence for the block could be implemented internally or externally. On a case by case basis, let us look at each module.

The IQ conversion unit consists of an NCO and IQ multipliers. The inputs are an I and Q signal and control values for an internal NCO. The output is the sum of the products of the NCO and IQ inputs. The NCO (Figure 9.25) can be tested autonomously using a signature analyzer. It would be possible to extend this to the full module by placing LFSRs on the I and Q inputs. A fault analysis would indicate how many vectors would have to be run to achieve an acceptable fault coverage. So

we probably do not need a WPP here.

The microprocessor has a sequencer that can be used to set up tests autonomously. So it probably does not need a WPP port.

The memories do not have any innate intelligence, so a WPP port may be used here to test the memories in parallel from a central RAM test unit (not unlike the design in the previous example). So one test unit tests four RAMs. Including the test unit in each RAM would mean that no WPP would be required.

Overall no WPPs are required at all – the time to serially shift data in and out just affects testing time – so they probably would go in for the RAMs.

In terms of TAM design, one could select the Daisy-chained TAM. But this is likely to impact test time (but good if you want to minimize pin count). The local TAM controller option is likely to be good as it minimizes pins and the local controllers default to very simple circuits for the processor and IQ converter. The basic thing here is that any of the designs work – we just want some good reasons such as reducing test time, complexity or pin count.

Chapter 10

10.1 *** simulation

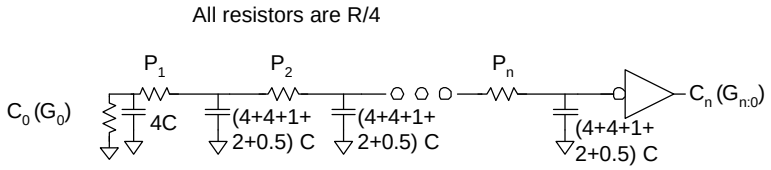
10.2 Overflow for signed numbers only occurs when adding numbers with the same sign (positive or negative). The numbers overflow (V) if the sign of the result Y does not match the sign of the inputs A and B:

$$V = A_{N-1}B_{N-1}\overline{Y}_{N-1} + \overline{A}_{N-1}\overline{B}_{N-1}Y_{N-1}$$

10.3 $$V = A_{N-1}(B_{N-1} \oplus SUB)\overline{Y}_{N-1} + \overline{A}_{N-1}(\overline{B}_{N-1} \oplus SUB)Y_{N-1}$$

10.4 The dynamic chain drives a known load capacitance, so its delay can be treated entirely as a parasitic delay. then the output inverter contributes logical effort and additional parasitic delay. The input capacitance is 4. The output resistance of the inverter is R for the critical rising output, equal to that of a unit inverter. Hence, the logical effort is g = 4/3. The output inverter has a parasitic delay of 5/6. The parasitic delay of the dynamic stage is computed using the Elmore delay model and added on to make:

$$p = \frac{5}{6} + \frac{\left(\frac{R}{4}\right)(4C) + \sum_{i=1}^{n}(n+1)\left(\frac{R}{4}\right)(11.5C)}{3RC} = \frac{5}{6} + \frac{1 + \frac{11.5n}{8}(n+3)}{3} = \frac{11.5}{24}n^2 + \frac{11.5}{8}n + \frac{7}{6}$$



10.5 Assuming the side loads are negligible so that each carry chain drives another identical chain and has $h = 1$, the stage delay is $g + p$. The number of stages is inversely proportional to n. Hence the delay per bit scales as:

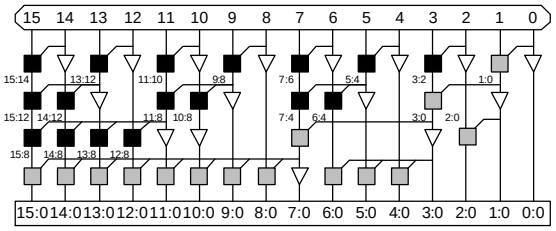
$$d = \frac{1}{n}\left[\frac{11.5}{24}n^2 + \frac{11.5}{8}n + \frac{7}{6} + \frac{4}{3}\right]$$

Taking the derivative of delay with respect to the length of each chain n and setting that equal to zero gives allows us to solve for the best chain length. Because the parasitic capacitance is large, the best delay is achieved with short carry chains ($n = 2$ or 3).

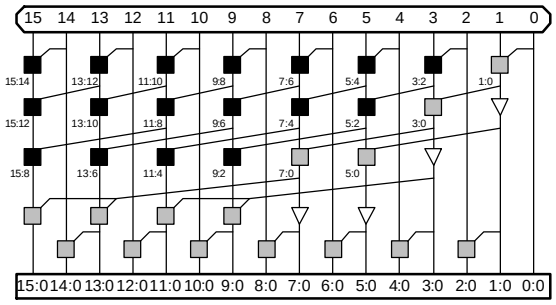
$$\frac{\partial}{\partial n}d = \frac{11.5}{24} - \frac{15}{6n^2} = 0 \Rightarrow n = 2.28$$

10.6 8 stages for 32-bit, 11 stages for 64-bit addition.

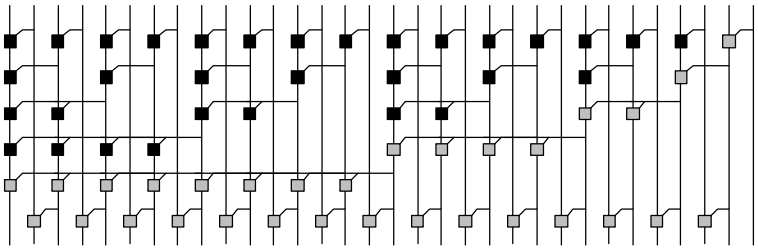
10.7



10.8



10.9



10.10

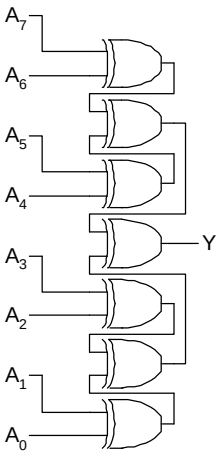
$$\begin{aligned} C_{out} &= \overline{(A \oplus B)\bar{C} + \overline{(A \oplus B)}\bar{A}} \\ &= \overline{A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}} \\ &= MAJ(A, B, C) \end{aligned}$$

10.11

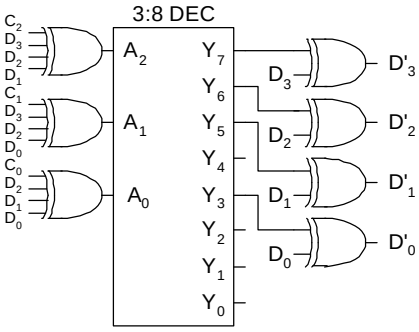
$$\begin{aligned} H_{i:j} &= G_{i:k} + G_{i-1:k} + P_{i-1:k-1}H_{k-1:j} \\ &= G_{i:k} + G_{i-1:k} + P_{i-1:k}P_{k-1:k-1}H_{k-1:j} \\ &= G_{i:k} + G_{i-1:k} + P_{i-1:k}G_{k-1:j} \\ &= G_{i:k} + G_{i-1:k} + G_{i-1:j} \\ &= G_{i:j} + G_{i-1:j} \\ I_{i:j} &= P_{i-1:k-1}P_{k-2:j-1} \\ &= P_{i-1:j-1} \end{aligned}$$

10.12 $-B = \overline{B} + 1$. Thus, the design of Figure 10.53 can be used if the B input is complemented and $c_0 = 1$.

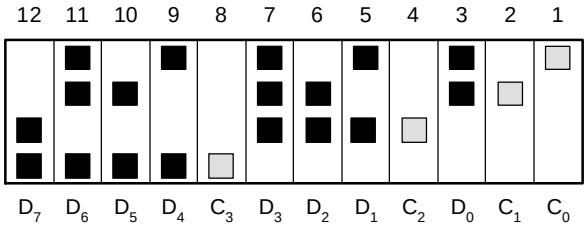
10.13



10.14 Use multiple-input XOR gates to compute the syndrome. Use a decoder to identify which bit needs correcting (000 means none need correcting). Use XOR gates to flip the bit that needs to be corrected to produce the outputs D'.



10.15 4 check bits suffice for up to 2^4-1 = 11 data bits.

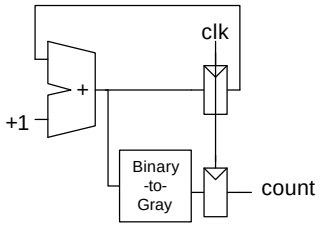


$$C_0 = D_6 \oplus D_4 \oplus D_3 \oplus D_1 \oplus D_0$$
$$C_1 = D_6 \oplus D_5 \oplus D_3 \oplus D_2 \oplus D_0$$
$$C_2 = D_7 \oplus D_3 \oplus D_2 \oplus D_1$$
$$C_3 = D_7 \oplus D_6 \oplus D_5 \oplus D_4$$

10.16 0: 0000; 1: 0001; 2: 0011; 3: 0010; 4: 0110; 5: 0111; 6: 0101; 7: 0100; 8: 1100; 9: 1101; 10: 1111; 11: 1110; 12: 1010; 13: 1011; 14: 1001; 15: 1000.

10.17 One way to do this is with a finite state machine, in which the state indicates the present count. The FSM could be described in a hardware description language with a case statement indicating the order of states. This technique does not generalize to N-bit counters very easily.

Another approach is to use an ordinary binary counter in conjunction with a binary-to-Gray code converter (N-1 XOR gates). The converter output must also be registered to prevent glitches in the binary counter from appearing as glitches in the Gray code outputs.



10.18

Inputs			Partial Product	Booth Selects		
x_{2i+1}	x_{2i}	x_{2i-1}	PP_i	*POS*	NEG_i	*DOUBLE*
0	0	0	0	0	0	0
0	0	1	*Y*	1	0	0
0	1	0	*Y*	1	0	0
0	1	1	*2Y*	1	0	1

1	0	0	$-2Y$	0	1	1
1	0	1	$-Y$	0	1	0
1	1	0	$-Y$	0	1	0
1	1	1	-0 (= 0)	0	0	0

POS = $\overline{x}_{2i+1}(x_{2i} + x_{2i-1})$; NEG = $x_{2i+1}(\overline{x}_{2i} + \overline{x}_{2i-1})$;

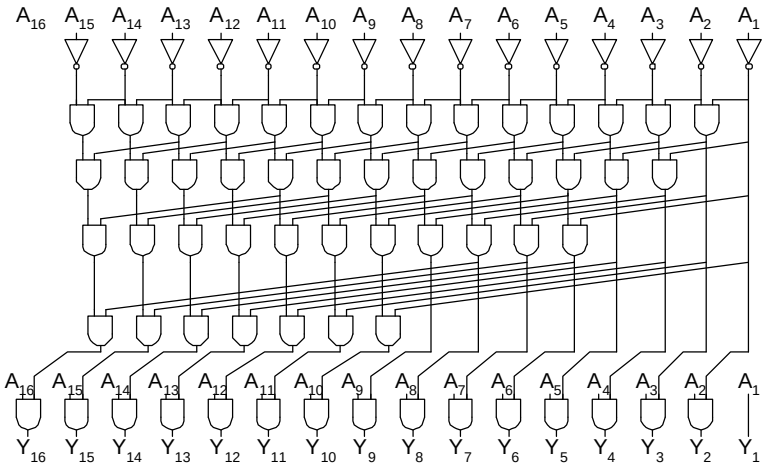
DOUBLE = $\overline{x}_{2i+1}x_{2i}x_{2i-1} + x_{2i+1}\overline{x}_{2i}\overline{x}_{2i-1}$

*** show encoder and selector from Chandrakasan01?

10.19 X0, X1, and X2 indicate exactly zero, one, or two 1's in a group. Y1, Y2, and Y3 are one-hot vectors indicating the first, second, and third 1.

$$X0_{i:i} = \overline{A_i}$$
$$X1_{i:i} = A_i \qquad \text{bitwise precomputation}$$
$$X2_{i:i} = 0$$
$$X0_{i:j} = X0_{i:k} \bullet X0_{k-1:j}$$
$$X1_{i:j} = X1_{i:k} \bullet X0_{k-1:j} + X0_{i:k} \bullet X1_{k-1:j} \qquad \text{group logic}$$
$$X2_{i:j} = X1_{i:k} \bullet X1_{k-1:j} + X2_{i:k} \bullet X0_{k-1:j} + X0_{i:k} \bullet X2_{k-1:j}$$
$$Y1_i = A_i X0_{i-1:1}$$
$$Y2_i = A_i X1_{i-1:1} \qquad \text{output logic}$$
$$Y3_i = A_i X2_{i-1:1}$$

10.20



10.21 Assume the branching effort on each A input is approximate 2 because it drives two gates (the initial inverter and the final AND). A path from input to output passes through an inverter and five AND gates, each made from a NAND and an inverter. There are four two-way branches within the network. Hence, $B = 32$. $G = 1^{6*}(4/3)^5 = 4.2$. $H = 1$. $P = 1*6 + 2*5 = 16$. $F = GBH = 135$. $N = 11$. $f = F^{1/N} = 1.56$. $D = Nf + P = 33.2\ \tau$. Note that the stage effort is lower than that desirable for a fast circuit. The circuit might be redesigned with NANDs and NORs in place of ANDs to reduce the number of stages and the delay.

10.22 The following equations are a slight modification of EQ 10.50. Use the base case $X_{1:1} = 1$, $W_{1:1} = 0$.

$$X_{i:i} = \overline{A_i \overline{A}_{i-1}} \qquad \text{bitwise precomputation}$$

$$W_{i:i} = A_i \overline{A}_{i-1}$$

$$X_{i:j} = X_{i:k} \bullet X_{k-1:j} \qquad \text{group logic}$$

$$W_{i:j} = W_{i:k} \bullet X_{k-1:j} + X_{i:k} \bullet W_{k-1:j}$$

$$Y_i = W_{i:1} \bullet W_{i-1:1} \qquad \text{output logic}$$

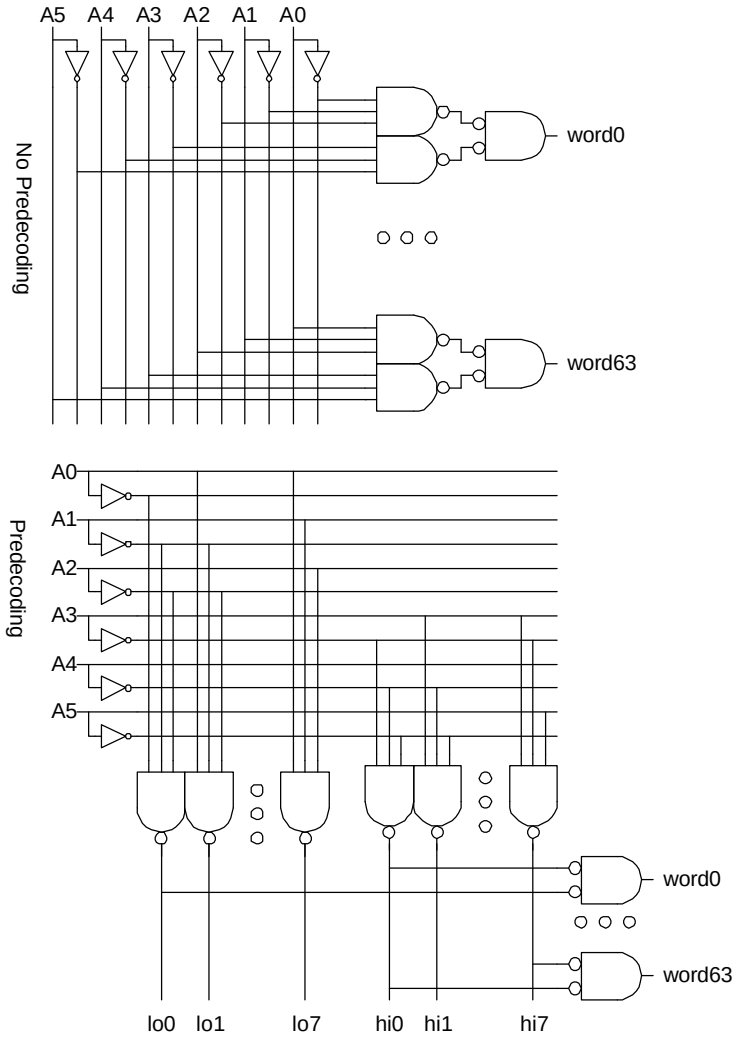
Chapter 11

11.1 If the array is organized as 128 rows by 128 columns, each column multiplexer must choose among (128/8) = 16 inputs.

11.2 The dimensions are (128 columns * 1.3 µm/col* 1.1) x (128 rows * 1.44 µm/row* 1.1) = 183 µm x 203 µm.

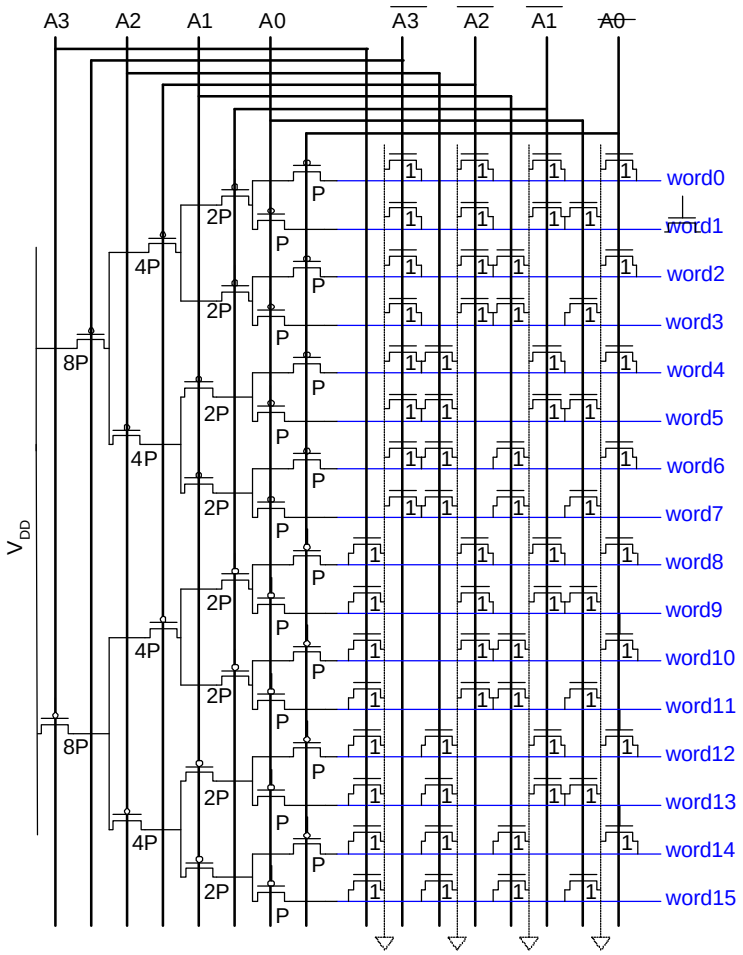
11.3 The design with predecoding uses 16 3-input NANDs while the design without uses 128. Both designs have the same path effort. Hence, the layout of the prede-

coded design tends to be more convenient.



11.4 For unit pull-up resistance, solve $2R/P + 2R/(2P) + 2R/(4P) + 2R/(8P) = R$ to find $P = 15/4$. The logical effort is $(1 + P) / 3 = 19/12$ and the branching effort is $N/2 =$

8.



11.5 (a) B = 512. H = 20. A 10-input NAND gate has a logical effort of 12/3, so estimate that the path logical effort is about 4. Hence $F = GBH$ = 40960. The best number of stages is $\log_4 F$ = 7.66, so try an 8-stage design: NAND3-INV-NAND2-INV-NAND2-INV-INV-INV. This design has an actual logical effort of G = (5/3) * (4/3) * (4/3) = 2.96, so the actual path effort is 30340. The path parasitic delay is P = 3 + 1 + 2 + 1 + 2 + 1 + 1 + 1 = 12. $D = NF^{1/N} + P$ = 41.1 τ.

(b) The best number of stages for a domino path is typically comparable to the best number for a static path because both the best stage effort and the path effort decrease for domino. Using the same design, the footless domino path has a path logical effort of G = 1 * (5/6) * (2/3) * (5/6) * (2/3) * (5/6) * (1/3) * (5/6) = 0.060 and a path effort of F = 610. The path parasitic delay is P = 4/3 + 5/6 + 3/3 + 5/6 + 3/

3 + 5/6 + 1/3 + 5/6 = 7. $D = NF^{1/N} + P = 24.8\ \tau$.

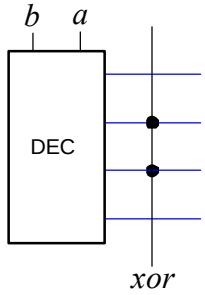
11.6 Design the footless domino decoder from Exercise 11.5(b) using self-resetting domino gates. Assume the inputs are available in true and complementary form as pulses with a duration of 3 FO4 inverters and can each drive 48 λ of gate width. Indicate transistor sizes and estimate the delay of the decoder.

***

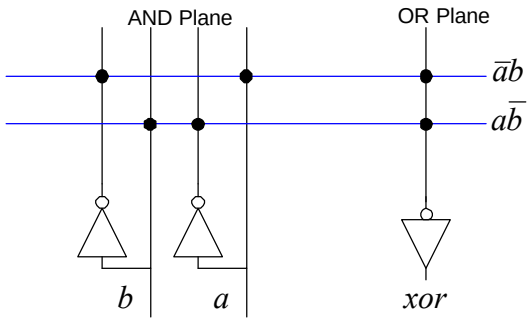
11.7 $H = 2^m$. $B = 2^{n-1}$ because each input affects half the rows. For a conservative estimate, assume that the decoder consists of an n-input NAND gate followed by a string of inverters. The path logical effort is thus G = $(n+2)/3$, so the path effort is $F = GBH = 2^{n+m}(n+2)/6$. The best numer of stages is $N = \log_4 F \sim (n+m)/2$. The parasitic delay of the n-input NAND and N-1 inverters is $P = n + (N-1)$. Hence, the path delay can be estimated as D = $((n+m)/2)\ (2^{n+m}(n+2)/6)\text{^}(2/(n+m)) + n + (N-1)$

11.8 In an open bitline, the sense amplifer compares the voltage on a bitline from the active subarray to the voltage on a bitline from a quiescent subarray. Power supply noise between subarrays makes sensing a small swing impossible. In a closed bitline, the two sense amplifier inputs come from bitlines in the same subarray, but only one of the two is activated. This design requires somewhat more layout area but eliminates most supply noise problems. It is still sensitive to coupling that affects one sense amplifier input more than the other. Twisted bitlines route the folded bitlines in such a way that each one sees exactly the same coupling capacitances, hence making coupling noise common mode as well. This is necessary in modern DRAM designs and costs slightly more area to perform the twists.

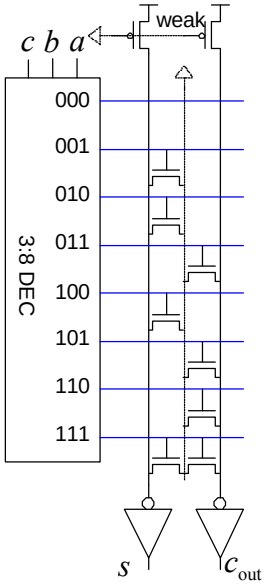
11.9



11.10



11.11



11.12 NAND ROMs use series rather than parallel transistors and one-cold rather than one-hot wordlines. They tend to be smaller than NOR ROMs because they do not require contacts between the series transistors, but they are also slower because of the series transistors.

11.13 The ROM cell is smaller than the SRAM cell. It presents one unit of capacitance for the transistor. Assume the wire capacitance is 1/2 as much, so each cell presents 1.5C on the wordline. It has only a single transistor in the pulldown path on the bitline so the resistance is R. Hence, the logical effort is 1/2, as compared to 2 for the SRAM cell.

The bitline has a capacitance of C/2 from the half contact. If the wire capacitance is again C/2, the total bitline capacitance is 2^n C. Because the cell has a resistance R, the delay is 2^n RC and the parasitic delay is $2^n/3$.

The ROM can use the same decoder as the SRAM, with a logical effort of $(n+2)/3$ and parasitic delay of n. Assume the bitline drives a load equal to that seen by the address so the path electrical effort is $H = 1$.

Putting this all together, the path effort is $F = GBH = 2^N(n+2)/6$. The path parasitic delay is $n + 2^n/3$. The path delay is $D = 2N + 4\log_4[(n+2)/6] + n + 2^n/3$.

Your modeling and loading assumptions may vary somewhat. The assumptions about wire capacitance have a large effect on the model.

Chapter 12

12.1 $P_{max} = (110\text{-}50) / (10 + 2) = 5$ W.

12.2 ***

Explain how an electrostatic discharge event could cause latchup on a CMOS chip.

12.3 H-trees ideally have zero skew and relatively low metal resource requirements, but in practice see significant skews, even locally, because of mismatches in loading, processing, and environment among the branches. Clock grids have low local skew because they short together nearby points, but can have large global skew and require lots of metal and associated capacitance. The hybrid tree/grid achieves low local skew because of the shorting without using as much metal as a full clock grid.

12.4 Calculate the bias-point and small-signal low-frequency gain of the common source amplifier from Figure 12.46 if $V_t = 0.7$, $\beta = 240\ \mu A/V^2$, and the nMOS output impedance is infinite. Let $V_{BIAS} = 3\ V$, $V_{DD} = 15\ V$, and $R_L = 10\ k\Omega$.

At the bias point of $V_{GS} = 3$, $I_{DS} = 240 \cdot 10^{-6} \cdot (3\text{-}0.7)^2 / 2 = 0.65$ mA and thus the output voltage is $V_{OUT} = 15 - I_{DS}R_L = 8.5$V. $g_m = 240 \cdot 10^{-6} \cdot (3\text{-}0.7) = 0.55$ mA/V. $v_{out} = -g_m R_L v_{in}$, or the gain $A = v_{out}/v_{in} = -g_m R_L = -5.5$.

***

12.5 Prove EQ (12.24).

12.6 Calculate the output impedance of the Wilson current mirror in Figure 12.97.

12.7 Design a current source that sinks 200 μA, using the process parameters from the example in Section 12.6.4. What is the minimum drain voltage over which your

current source operates?

12.8 Find the output impedance of your current source from Exercise 12.7. By what fraction does the current change as the output node changes by 1 V?

12.9 Simulate the operational amplifier of Figure 12.63. Using minimum-size transistors and a 10 kΩ resistor, what is the gain?

12.10 What changes would you make to the amplifier from Exercise 12.9 to increase the gain? What are the tradeoffs involved? What gain can you achieve using reasonable changes?

12.11 Prove EQ (12.31).

12.12 Use SPICE to find the transconductance and output resistance of a minimum-size transistor in your process biased at $V_{gs} = V_{ds} = V_{DD}/2$. What is the $g_m r_o$ product?

12.13 Repeat Exercise 12.12 for a transistor with 2x minimum channel length. How does the product change?

12.14 In Section 12.6.8 on resistor string DACs, it was mentioned that a similar DAC can be implemented with capacitors. Design the architecture of a 4-bit capacitor DAC.

12.15 A bias generator is required to generate 16 steps from 0 to 100 μA to bias an amplifier. Design a CMOS DAC to do this, assuming the presence of a 50 μA reference current.

12.16 Differential circuits provide good noise immunity and have the advantage of dual rail inputs and outputs. Pseudo-differential circuits based on CMOS inverter amplifiers can be implemented by using two signal paths that process each signal, but do not provide for the same level of noise immunity. Design a single stage of a pipeline ADC that uses this style of differential circuit.

12.17 To reduce clock and decoder skew in a current-mode DAC, a latch is often included in the current cell. Design the circuit for such a cell, demonstrating where the latch would be placed. If this is a slave latch, where would the master latch be located?