Chapter
13

Clocked Circuits

The transmission gate (TG) is used in digital CMOS circuit design to pass or not pass a signal, see Sec. 10.2.1. The schematic and logic symbol of the transmission gate (TG) are shown in Fig. 13.1. The gate is made up of the parallel connection of an NMOS and a PMOS device. Referring to the figure when S (for select) is high, we observe that the transmission gate passes the signal on the input to the output (noting the nodes we define as the input or output are interchangeable). The resistance between the input and the output can be estimated as $R_n \| R_p$. We begin this chapter with a description of the CMOS TG.

Figure 13.1 The transmission gate.

13.1 The CMOS TG

Since the NMOS pass gate (PG) passes logic lows well and the PMOS PG passes logic highs well, putting the two complementary MOSFETs in parallel, as seen in Fig. 13.1, produces a TG that passes both logic levels well. The propagation delay-times of the CMOS TG in the configuration seen in Fig. 13.2 (with a large load capacitance) are estimated as

$$t_{PHL} = t_{PLH} = 0.7 \cdot (R_n \| R_p) \cdot C_{load} \qquad (13.1)$$

The capacitance on the S input of the TG is the input capacitance of the NMOS device, or C_{inn} (= $1.5C_{oxn}$). The capacitance on the $\bar{S}$ input of the TG is the input capacitance of the PMOS device, or C_{inp}.

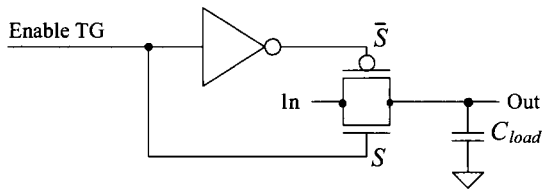


Figure 13.2 The transmission gate with control signals shown.

Increasing the widths of the MOSFETs used in the TG reduces the propagation delay-times from the input to the output of the TG when driving a specific load capacitance. However, the delay-times in turning the TG on, the select lines going high, increase because of the increase in input capacitance. This should be remembered when simulating. Using a voltage source in SPICE for the select lines, which can supply infinite current to charge the input capacitance of the TG, gives the designer a false sense that the delay through the TG is limited by R_n and R_p. Often, when simulating logic of any kind, the SPICE-generated control signals are sent through a chain of inverters so that the control signals more closely match what will actually control the logic on die (and the control signals have a finite source driving resistance).

Example 13.1

Estimate and simulate the delays through the TG circuit shown in Fig. 13.3 using the short-channel CMOS process and a load capacitance of 50 fF.

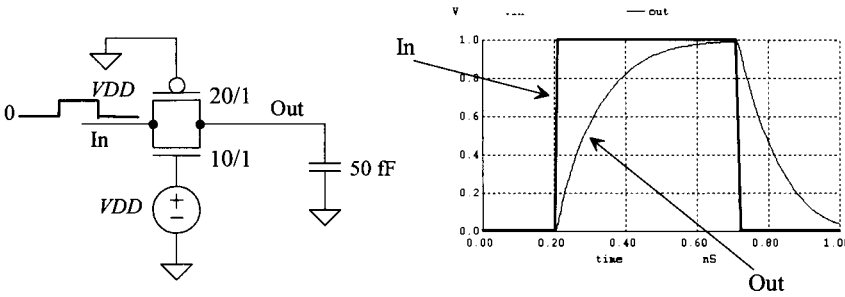


Figure 13.3 TG circuit discussed in Ex. 13.1.

From Table 10.2 $R_n \| R_p = 1.7\ \text{k}\Omega$ so that using Eq. (13.1) we get $t_{PLH} = t_{PHL} = 60$ ps. The SPICE simulation results are also seen in Fig. 13.3. ■

Example 13.2

Repeat Ex. 13.1 for the circuit seen in Fig. 13.4

The output load is initially at *VDD* and then discharged to ground when the TG turns on. The delay time calculation is exactly the same as the one used in Ex. 13.1. Also seen in the figure are the simulation results. ■

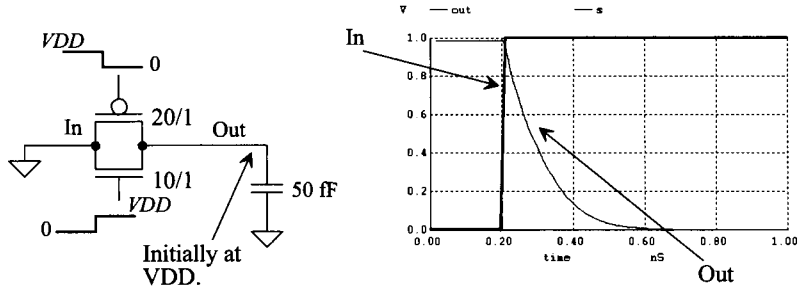


Figure 13.4 TG circuit discussed in Ex. 13.2.

Series Connection of TGs

Consider the series connection of the CMOS transmission gates shown in Fig. 13.5. The equivalent digital model is also depicted in this figure. As seen in Figs. 10.17, 10.21, and the associated discussion, the capacitance on each MOSFET's source/drain (assuming triode operation) is $C_{ox}/2$. The total capacitance on each internal node in a series connection of TGs is then the sum of the oxide capacitances from each adjacent PG, that is, $C_{oxn} + C_{oxp}$. The delay through the series connection of TGs can be estimated using

$$t_{PHL} = t_{PLH} = 0.7 \cdot N \cdot (R_n || R_p)(C_{load}) + 0.35 \cdot (R_n || R_p)(C_{oxn} + C_{oxp})(N)^2 \tag{13.2}$$

The first term in this equation is simply the time needed to charge C_{load} through the sum of the TG effective resistances, while the second term in the equation describes the RC transmission line effects.

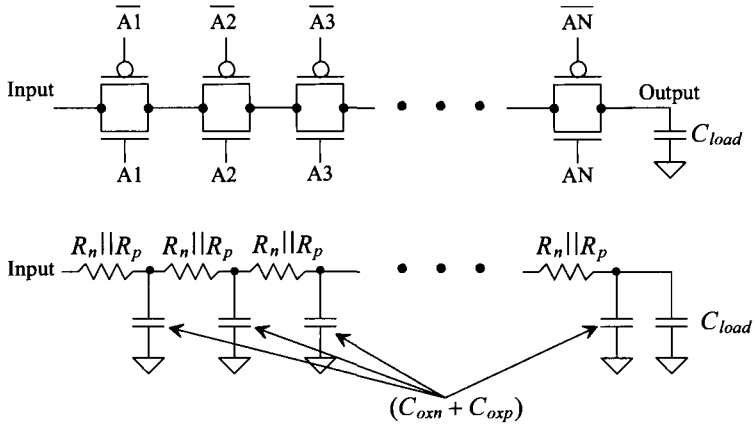


Figure 13.5 Series connection of TGs with digital model.

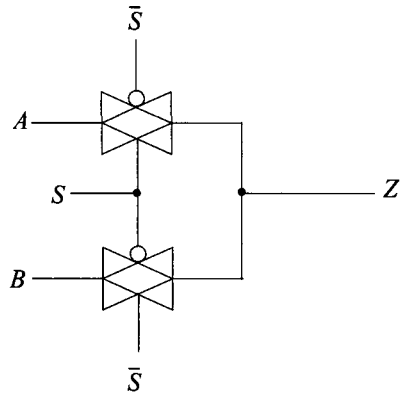


Figure 13.6 Path selector.

13.2 Applications of the Transmission Gate

In this section we present some of the applications of the TG.

Path Selector

The circuit shown in Fig. 13.6 is a two-input path selector. Logically, the output of the circuit can be written as

$$Z = AS + B\bar{S} \tag{13.3}$$

When the selector signal S is high, A is passed to the output while a low on S passes B to the output.

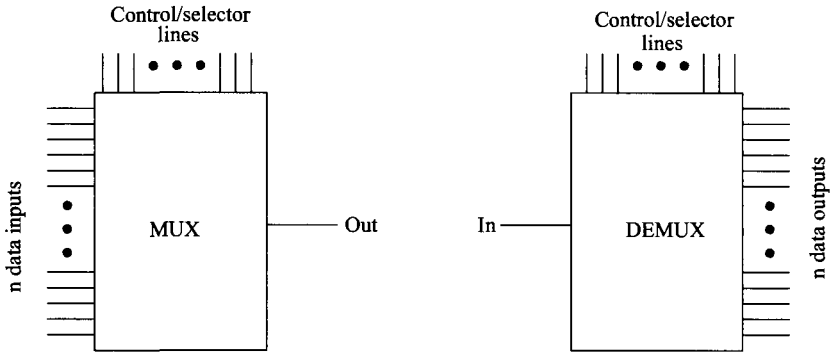


Figure 13.7 Block diagram of MUX/DEMUX.

This same idea can be used to implement multiplexers/demultiplexers (MUX/DEMUX). Consider the block diagrams of a MUX and DEMUX shown in Fig. 13.7. The number of control lines is related to the number of input lines by

$$2^m = n \tag{13.4}$$

where n is the number of inputs (outputs) to the MUX (DEMUX) and m is the number of control lines. A 4-to-1 MUX/DEMUX is shown in Fig. 13.8. Note that the MUX is bi-directional; that is, it can be used as a MUX or a DEMUX. The logic equation describing the operation of the MUX is given by

$$Z = A(S1 \cdot S2) + B(S1 \cdot \overline{S2}) + C(\overline{S1} \cdot S2) + D(\overline{S1} \cdot \overline{S2}) \tag{13.5}$$

Figure 13.9 shows the transistor-level implementation of the circuit in Fig. 13.8. Notice how the PMOS are grouped together (and laid out in the same n-well) while the NMOS are grouped together.

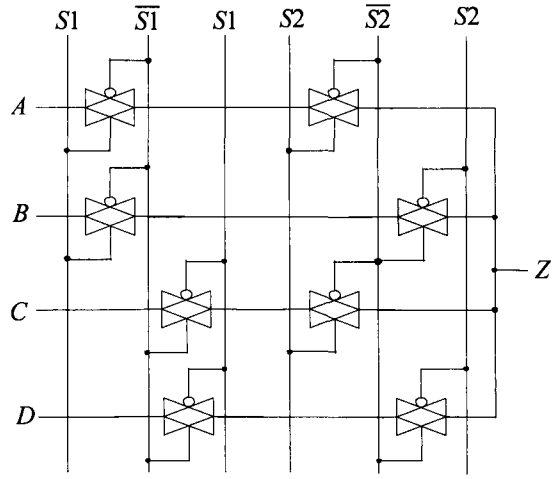


Figure 13.8 Circuit implementation of a 4-to-1 MUX.

Figure 13.10a shows an NMOS PG implementation of the 4-to-1 MUX. The pass transistor implementation is simpler, using fewer transistors, at the price of a threshold voltage drop from input to output when the input is a high (*VDD*). A simplified version of the circuit of Fig. 13.10a is shown in Fig. 13.10b. Here the MOSFETs connected to *S2* and $\overline{S2}$ are combined to reduce the total number of MOSFETs used. The reduction of the total number of MOSFETs used can be extended to an n-input (output) MUX (DEMUX) (see Fig. 16.43). Again, it should be remembered that a DEMUX can be formed using the circuits of Figs. 13.8 or 13.10 by switching the inputs with the outputs.

Static Gates

The TG can be used to form static logic gates. Consider the OR gate shown in Fig. 13.11. To understand the operation of the gate, consider the case when both *A* and *B* are low. Under these circumstances the pass transistor, M1 is off, and the TG is on. Since the input *B* is low, a low is passed to the output. If *A* is high, M1 is on and *A* is passed to the output. If *B* is high and *A* is low, *B* is passed to the output through the TG. If both *A* and *B* are high, the TG is off and M1 is on passing *A*, a high, to the output.

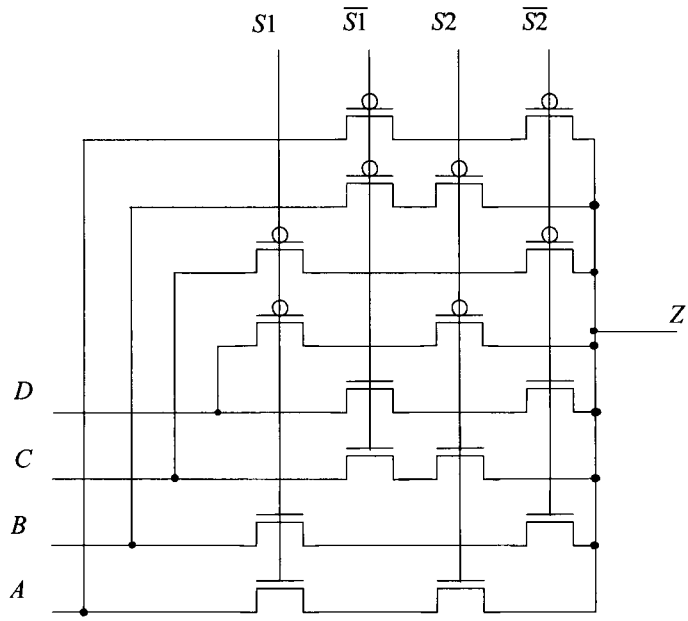


Figure 13.9 Transistor implementation of Fig. 13.8.

Figure 13.12 shows an XOR and an XNOR gate made using TGs. Consider the XOR gate with both A and B low. Under these circumstances, the top TG is on and its output is connected to A, a low. If both inputs are high, the bottom TG connects the output to $\overline{A}$, again a low. If A is high and B is low, the top TG is on and the output is connected to A, a high. Similarly, if A is low and B is high, the bottom TG is on and connects the output to $\overline{A}$, a high.

13.3 Latches and Flip-Flops

Basic Latches

Consider the set-reset latch (SR latch) shown in Fig. 13.13 made using NAND gates. The logic symbol and truth table are also shown in this figure. Consider the case when S is high and R is low. Forcing R low causes Q to go high. Since S is high and $Q$is high, the $\overline{Q}$ output is low. Now consider the case when both S and R are low. Under these circumstances, the latch's outputs are both high. This latch can easily be designed and laid out with the techniques of Ch. 12 (see also Fig. 15.6).

An alternative implementation of the SR latch is shown in Fig. 13.14 using NOR gates. Consider the case when S is high and R is low. For the NOR gate, a high input forces the output of the gate low. Therefore, the $\overline{Q}$ output is low whenever the S input is high. Similarly, whenever the R input is high, the Q output must be low. The case of both inputs being high causes both Q and $\overline{Q}$ to go low, or in other words the outputs of the latch are no longer complements.

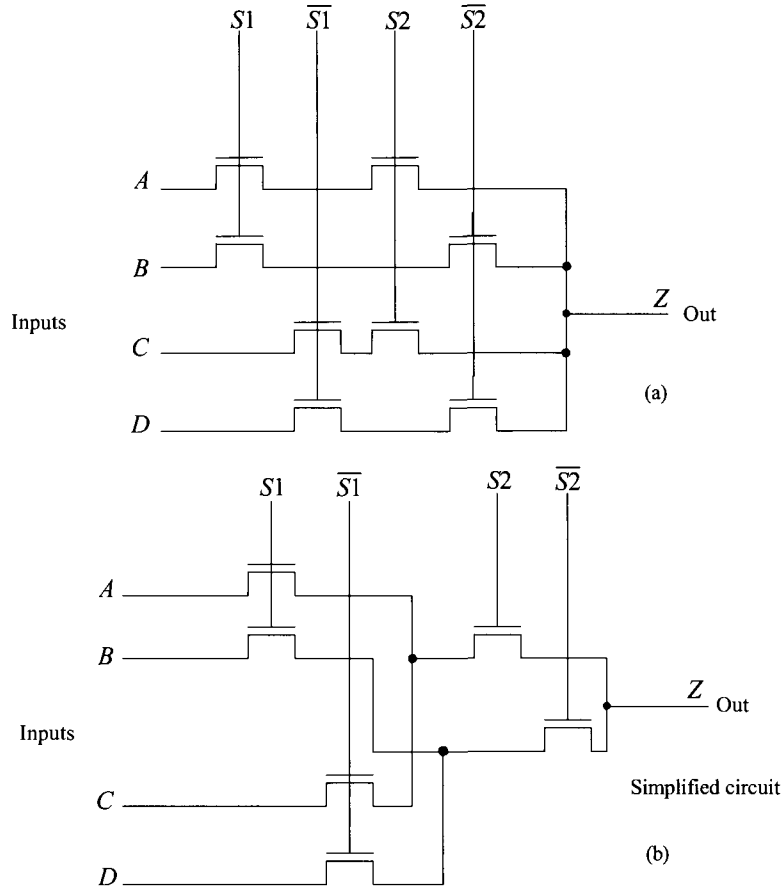


Figure 13.10 MUX/DEMUX using pass transistors.

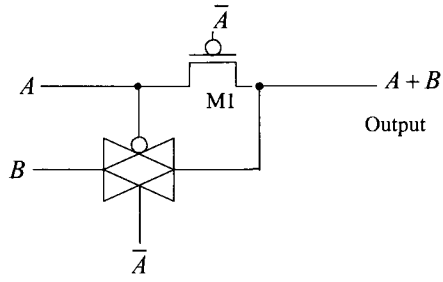


Figure 13.11 TG-based OR gate.

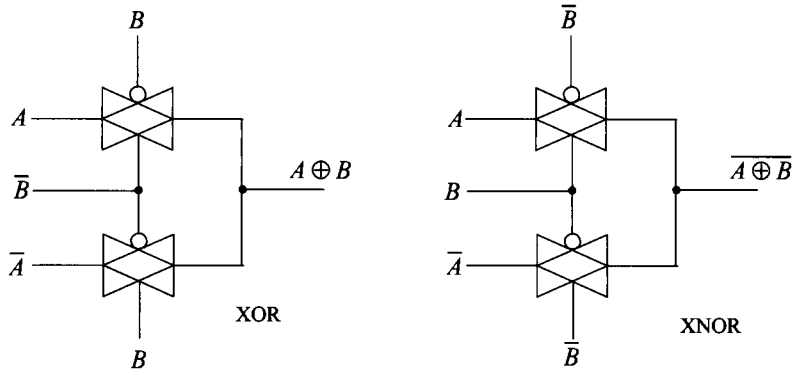
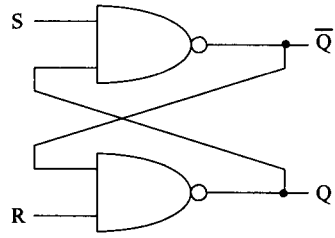


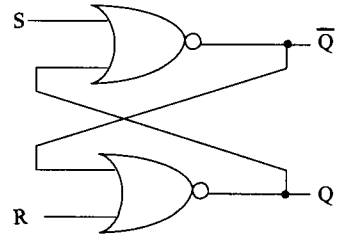
Figure 13.12 TG implementation of XOR/XNOR gate.



Truth table

S	R	Q	$\overline{Q}$
0	0	1	1
1	0	1	0
0	1	0	1
1	1	Q	$\overline{Q}$

Figure 13.13 Set-reset latch made using NAND gates.



Truth table

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	0	1	0
0	1	0	1
1	1	0	0

Figure 13.14 Set-reset latch made using NOR gates.

An Arbiter

One of the uses of the NAND latch is in an arbitration circuit (called an arbiter, Fig. 13.15). This circuit can be very useful in asynchronous circuit design or in clock synchronization circuits (see Ch. 19). To understand the operation of the arbiter in Fig. 13.15, let's consider the case when both In1 and In2 are low. Both NAND gates' outputs are high and the two inverters' outputs, that is Out1 and Out2, are low. When In1 goes high, the output of X1 goes low, while the output of X2 remains high. This causes Out1 to go high and Out2 to remain low. Notice that with the output of X1 going low the power supplied to the PMOS in the inverter connected to Out2 is removed (making it impossible for Out2 to go high). When In2 goes high, while In1 is already high, the fact that the output of X1 is a low keeps the output of X2 high and Out2 low. When In1 goes low, with In2 high, Out1 goes low and Out2 goes high. The usefulness of this circuit occurs when In1 and In2 transition high at nearly the same times. The way the inverters are powered by the NAND gates makes it impossible for both Out1 and Out2 to go high at the same time. *Thus an arbiter is useful for determining which input arrives first.*

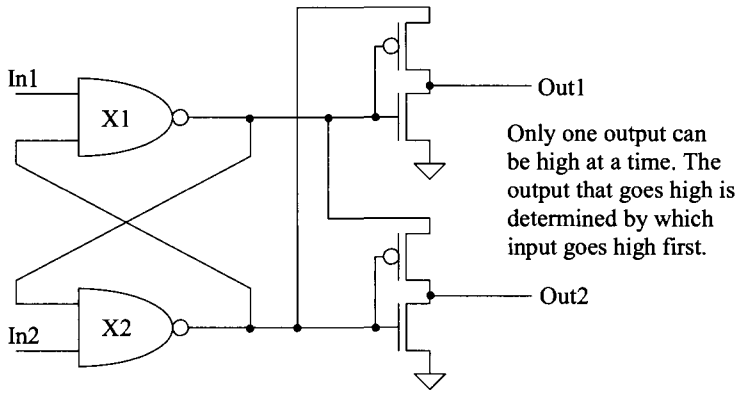


Figure 13.15 An arbiter made using NAND gates.

To ensure that the arbiter makes the decision quickly (there isn't a long delay or *metastability* when the two inputs are arriving at the same time), two inverters can be added in series with the NAND gate outputs. This addition of inverters simply increases the gain of the NAND gates (their VTCs get steeper at the switching point, see Fig. 12.3 in the last chapter).

Flip-Flops and Flow-through Latches

A flow-through latch is a storage circuit whose output changes with a clock signal level (so a flow-through latch is often called a level-sensitive latch). A flip-flop (FF) is a storage circuit that changes states on the rising or falling edge of a clock signal.

Most FFs in CMOS IC design are based on the cross-coupled inverters seen in Fig. 13.16. Also seen in this figure are the VTCs for the two inverters. I1's characteristics are drawn with its input on the x-axis and its output on the y-axis (the normal curves as

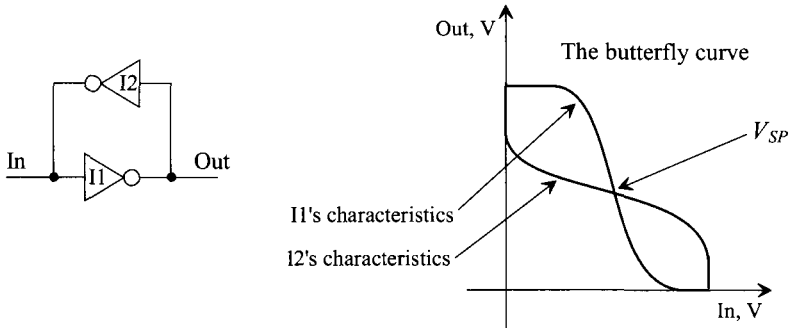


Figure 13.16 A cross-coupled inverter (a latch) and the input output characteristics.

seen in Fig. 11.2 back in Ch. 11). I2's input is labeled Out in the figure while its output is labeled In (and so its input is drawn as the y-axis and its output is drawn as the x-axis). The big concern when designing a FF with cross-coupled inverters is *metastability* (meaning changing stability). When the input signal moves to the switching-point of the inverters, both In and Out will be at V_{SP} (a stable state). This isn't where we want the circuit to operate because V_{SP} isn't a valid logic level voltage. Driving the input higher or lower enables the positive feedback present in the cross-coupled inverters to drive the In/Out nodes to valid logic levels (the other stable states). To avoid metastability, we can 1) use longer length inverters (which increases the gains of the inverters at the cost of longer delays), 2) use two inverters in series with the outputs of each inverter in Fig. 13.16 (again, to boost the gain) so that three inverters are used between In and Out, 3) ensure the In signal is driven with good logic levels (and a low impedance source to overdrive the output of I2), or 4) add a switch on the output of I2 to disconnect it when the In signal is changing to ensure that the input signal doesn't have to "fight" with the output of I2 to drive the signal to a valid logic level (the input signal range must still swing to valid logic levels). Figure 13.17 shows how metastability can result in a long delay before the outputs move to valid logic states. In this simulation the two capacitors are charged to 500 mV. It takes approximately 2.5 ns for the In/Out signals to change states.

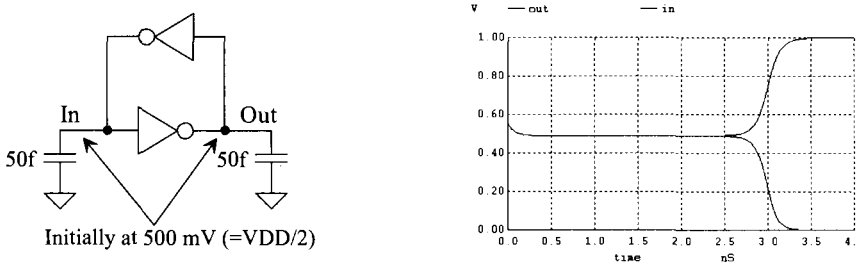


Figure 13.17 The delay associated with metastability.

Note that trying to move the switching point voltages of each inverter to differing levels makes the problem of metastability worse. When each inverter's V_{SP} is the same the gain around the loop is the highest (and so it's possible noise alone can cause the inverters to snap to a valid logic state).

Figure 13.18 shows a level-sensitive latch. When the clock signal is high, the data or D input passes through the NMOS PG and drives the cross-coupled inverters. The feedback inverter's lengths are made long to ensure that the source driving the D input can "overpower" the feedback inverter and allow the output to switch states. This circuit, while simple, has several fundamental issues. To begin, we know that an NMOS pass gate's output will swing from 0 to $VDD - V_{THN}$. To optimize the noise margins, we lower the V_{SP} of I1. Here we reduce the width of the PMOS from 20 to 10. Another concern is the contention current that flows when the D input is fighting with the output of I2 for control of I1's input. By increasing the lengths in I2 to 10, we lower this wasted current.

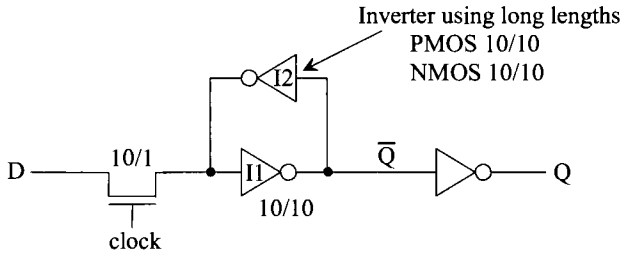


Figure 13.18 A level-sensitive latch.

Figure 13.19 shows the input and output simulation results for the latch in Fig. 13.18. When the clock signal is high, the input flows to the output (with a delay). When the clock signal goes low, the value on the D input is captured and remains on the Q output until the clock signal goes back high again. Notice that at 8 ns, when the clock signal goes high, the delay between the D input going high and the Q output going high is considerable.

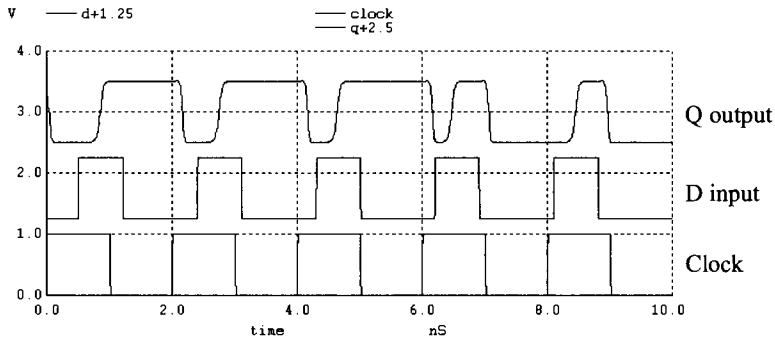


Figure 13.19 Simulating the level-sensitive latch in Fig. 13.18.

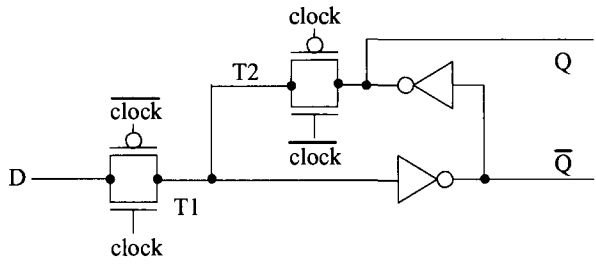


Figure 13.20 A higher performance level-sensitive latch.

Figure 13.20 shows a modification of the latch in Fig. 13.18. A TG is used on the input of the latch instead of a PG. This improves the noise performance of the circuit (at the cost of an additional clock signal, that is, the complement of clock). We've also added a TG in series with the feedback inverter. When T1 is on, the Q output follows the D input. When T1 is on, T2 is off so that the input doesn't have to fight with the feedback inverter. When T1 turns off, T2 turns on and the value of D at that instance is stored in the inverters. Figure 13.21 shows the simulation results using this latch with the input signals used to generate Fig. 13.19. Notice, for example, the improvement in performance over what's seen in Fig. 13.19 when the clock transitions high at 8 ns.

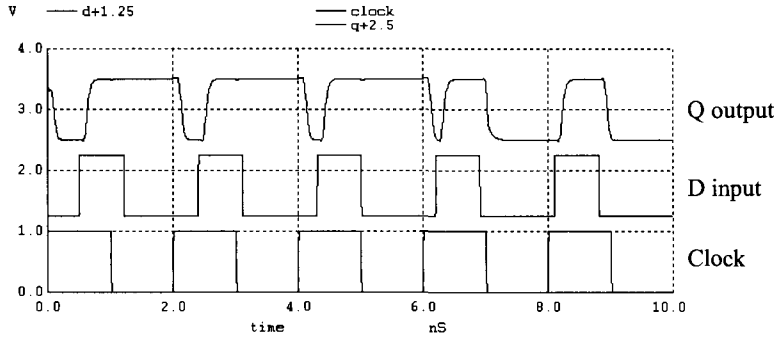


Figure 13.21 Simulating the level-sensitive latch in Fig. 13.20.

An Edge-Triggered D-FF

Notice that the value of the D input in the circuit seen in Fig. 13.20 is stored or captured when clock goes low. To implement an edge-triggered FF, we can use two of these level-sensitive latches in cascade. When the clock is low, the first stage tracks the D input and the second stage holds the previous output. When clock goes high, the first stage captures the input and transfers it to the second stage. The first stage is often called the "master" latch, while the second stage is the "slave" latch. Figure 13.22 shows an implementation of an edge-triggered D-FF. When clock is low, T1 and T4 are on. T2 and T3 are off. The D input flows through to point A and its complement to point B. When clock goes high, T1 and T4 shut off while T2 and T3 turn on. This causes the value of the D input, when

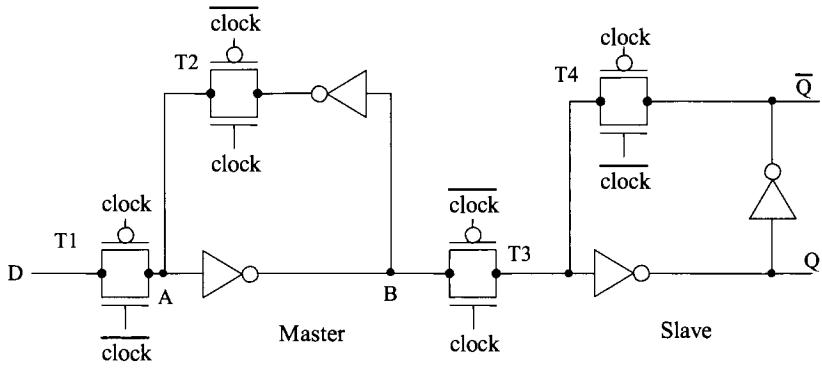


Figure 13.22 An edge-triggered D-FF.

clock transitioned high, to be captured and passed to the Q output. When clock goes back low, T1 and T4 turn on. The value of D on the Q output is then circulated around the two inverters. The Q output can change states again when clock goes back high. Figure 13.23 shows results of simulating this FF with SPICE. Note the output only changes on the rising edge of the clock signal.

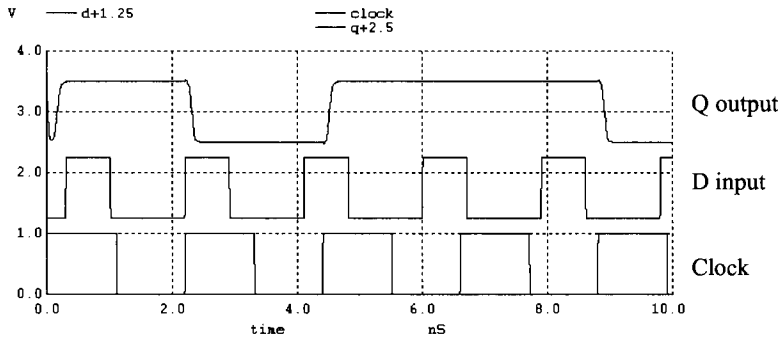


Figure 13.23 Simulating the operation of the edge-triggered FF in Fig. 13.22.

Figure 13.24 shows the addition of NAND gates to the circuit of Fig. 13.22 for clear and set inputs. When clear goes low, the outputs of N1 and N4 go high. If T3 is on or T4 is on, the outputs of the FF are forced to Q = 0 (and thus its complement is set to a 1). If the set input goes low, then Q is forced to a 1. If both clear and set are pulled low, then both the Q output and its complement are driven high. This feature can useful in some situations.

Note that it is **very important** for the clock signals we use to have quick rise times. If they don't, then the FF may not function correctly. Often the clock signals are applied to a string of inverters both to sharpen their transition times and to generate the complement clock required for the TGs.

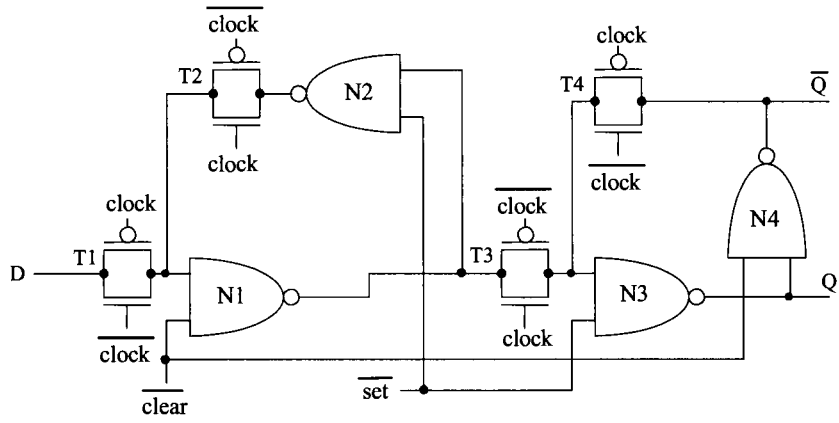


Figure 13.24 An edge-triggered FF with asynchrounous set and clear.

Flip-Flop Timing

The data must be *set up* or present on the D input of the FF (see Fig. 13.22) a certain time before we apply the clock signal. This time is defined as the setup time of the FF. To understand the origin of this time, consider the time it takes the signal at D to propagate through T1 and the inverter to node B. Before the clock pulse can be applied, the logic level D must be settled on node B. Consider the waveforms of Fig. 13.25. The time between D going high (or low) and the clock rising edge is termed the setup time of the FF and is labeled, t_s.

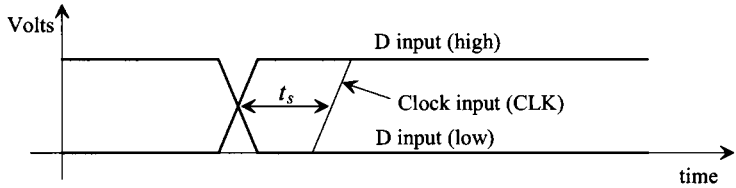


Figure 13.25 Illustrating D FF setup time.

The wanted D input must be applied t_s before the clock pulse is applied. Now the question becomes "How long does the wanted D input have to remain applied to, or *held* on, the input of the FF after the clock pulse is applied?" This time is called the hold time, t_h, and is illustrated in Fig. 13.26. Shown in this figure, t_h is a positive number. However, inspection of Fig. 13.22 shows that if the D input is removed slightly before the clock pulse is applied, node B will remain unchanged because of the propagation delay from the D input to node B. Analysis of this FF would yield a negative hold time. In other words, for the point B to charge to D, a time labeled t_s is needed. Once point B is charged, the D input can be changed as long as the clock signal occurs within t_h.

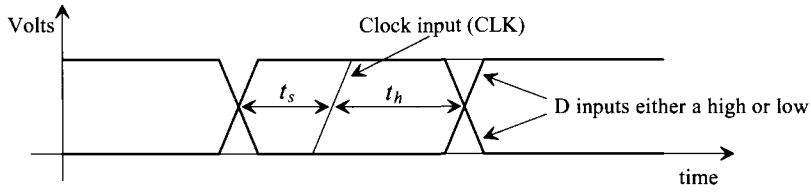


Figure 13.26 Illustrating D FF hold time.

One final important comment regarding the clock input of an FF is (again) in order. If the clock input risetime is slow, the FF will not function properly. There will not be an abrupt transition between the sets of transmission gates turning on and off. The result will be logic levels at indeterminate states. What is usually done to eliminate this problem is to buffer the clock input through several inverters. This has the effect of speeding up the leading and trailing edges of a slow input pulse and presenting a lower input capacitance on the clock input to whatever is driving the FF. The main disadvantage is the increase in delay times, t_{PHL} and t_{PLH} (defined by clock to output), of the FF. In general, the FF of Fig 13.22 should not be laid out without buffering the clock inputs.

The minimum pulse width of the clock, set, or clear inputs in Figs. 13.22 and 13.24 is labeled t_w. The minimum width is determined by the delay through (referring to Fig. 13.20) two NAND gates and a TG. The last timing definition we will comment on here is the recovery time, that is, the time between removing the set or clear inputs and a valid clock input. This variable is labeled t_{rec}.

13.4 Examples

In this section we give examples of delay calculations and comments on how to decrease the delays and the associated performance costs. Throughout this section we use the 50 nm short-channel process.

Example 13.3

Estimate the input capacitance and the delay through the circuit seen in Fig. 13.27. Compare the hand calculation estimates to simulation results.

Using the information seen in Table 10.2, we can calculate the first inverter's input capacitance, and thus the input capacitance of the circuit,

$$C_{in1} = \frac{3}{2} \cdot (C_{oxn1} + C_{oxp1}) = 0.938\,fF + 1.875\,fF = 2.81\,fF$$

Since the widths of the second inverter are 10 times larger than the widths of the first inverter, the input capacitance of the second inverter is 28.1 fF.

The delay through the first inverter, that is, the delay from the node labeled "In" to the node labeled "N1" in Fig. 13.27 is calculated as

$$t_{PHL1} + t_{PLH1} = 0.7 \cdot (3.4k + 3.4k) \cdot (0.625 + 1.25 + 9.38 + 18.75)\,fF = 143\,ps$$

noting, because $t_{PHL} = t_{PLH}$, that $t_{PHL1} = t_{PLH1} = 71.5\,ps$.

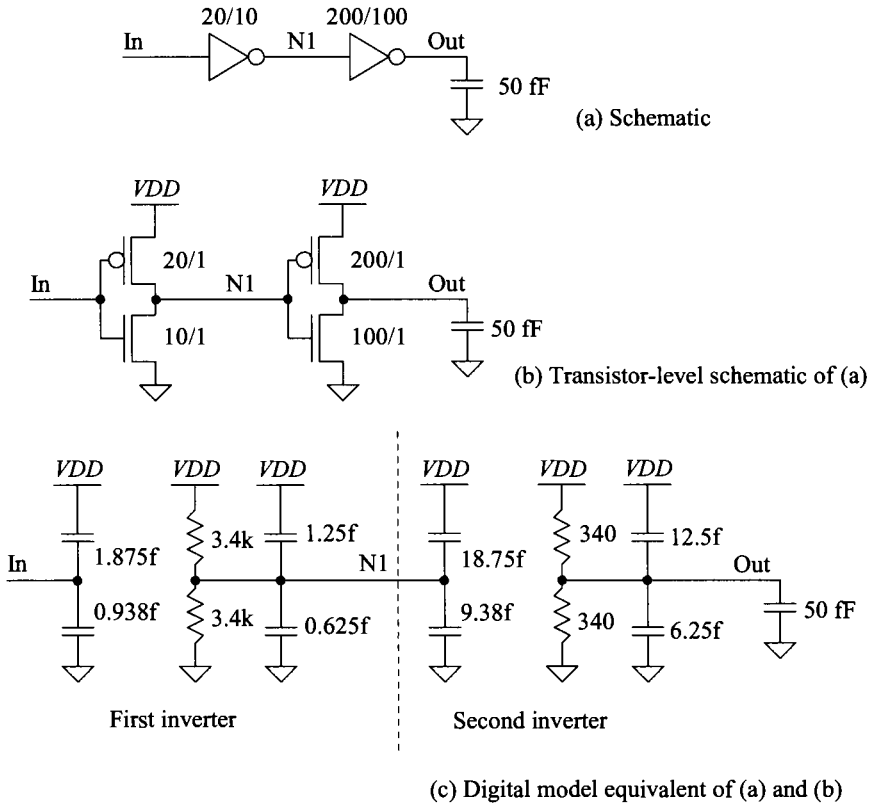


Figure 13.27 Circuit used in Ex. 13.3.

The delay through the second stage, that is, from N1 to the output is calculated as

$$t_{PHL2} + t_{PLH2} = 0.7 \cdot (340 + 340) \cdot 68.75\,fF = 33\,ps$$

Again as seen in the first inverter, the effective switching resistance of the PMOS is equal to the NMOS device's effective resistance so $t_{PHL2} = t_{PLH2} = 16.5\,ps$.

The overall delay through both inverters is

$$t_{PHL} = t_{PLH} = 88\,ps$$

Figure 13.28 shows the SPICE simulation results. The delays, from the simulations, are approximately 120 ps.

To decrease the delay through the circuit, the simplest solution is to increase the widths of the first inverter. Of course, this increases the circuit's input capacitance. Note that the *analysis* of this circuit has nothing to do with the *design* equations used for implementing a buffer, Sec. 11.4. ■

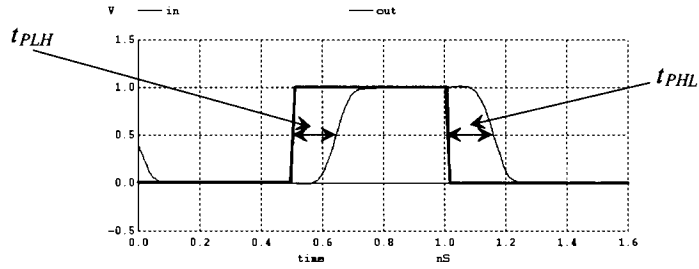


Figure 13.28 Simulating the circuit in Ex. 13.3 and Fig. 13.27.

Example 13.4

In Ex. 13.3 the devices were sized to give (ideally) equal propagation delays. Repeat Ex. 13.3 for the circuit seen in Fig. 13.29.

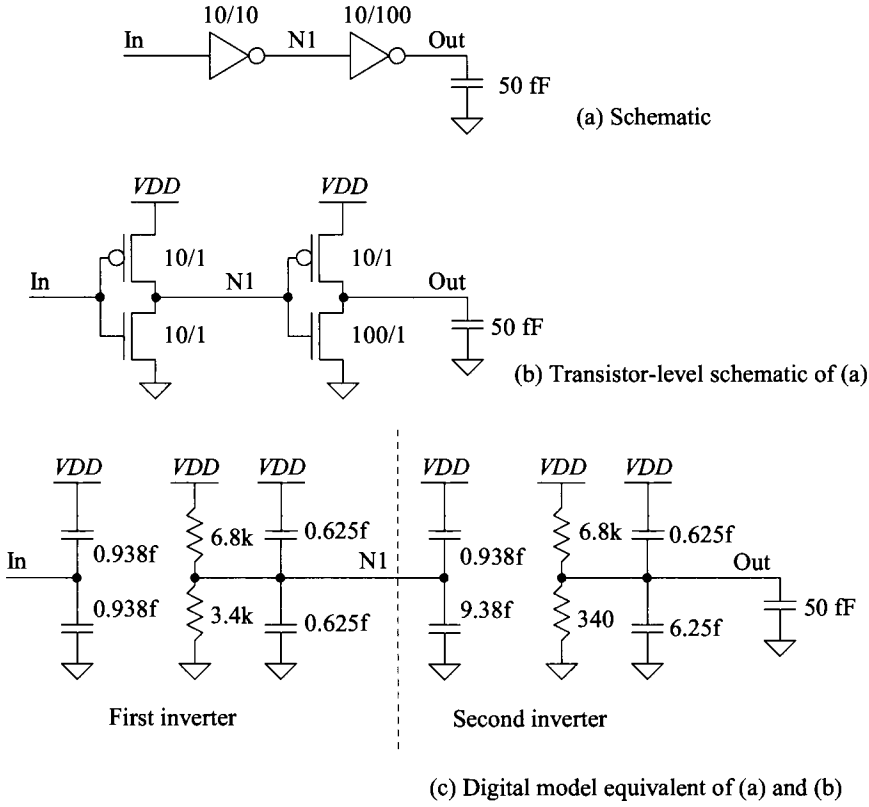


Figure 13.29 Circuit used in Ex. 13.4.

The PMOS device has a width of 10 which is half of the width used for the data in Table 10.2. We can write the effective switching resistance and oxide capacitance for this device as

$$R_p = 6.8k \text{ and } C_{oxp} = 0.625\,fF$$

As seen in Fig. 13.29, the input capacitance of the circuit is 1.876 fF.

Calculating t_{PLH} begins by noting that when the input goes high the NMOS device in the first inverter turns on. This turns the PMOS device in the second inverter on. The overall delay is calculated as

$$t_{PLH} = \overbrace{0.7 \cdot 3.4k \cdot 11.57\,fF}^{\text{First inverter's delay}} + \overbrace{0.7 \cdot 6.8k \cdot 56.88\,fF}^{\text{Second inverter's delay}} \approx 300\,ps$$

Similarly, when the input goes low, the PMOS in the first inverter turns on and the NMOS in the second inverter turns on, pulling the output low

$$t_{PHL} = \overbrace{0.7 \cdot 6.8k \cdot 11.57\,fF}^{\text{First inverter's delay}} + \overbrace{0.7 \cdot 340 \cdot 56.88\,fF}^{\text{Second inverter's delay}} \approx 70\,ps$$

The simulation results are seen in Fig. 13.30. ■

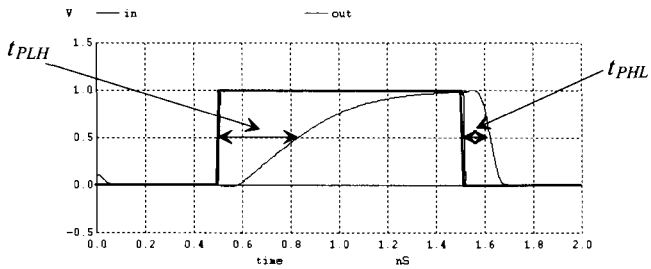


Figure 13.30 Simulating the operation of the circuit in Fig. 13.29.

Example 13.5

Estimate the delay through the circuit seen in Fig. 13.31 using 20/1 PMOS and 10/1 NMOS devices.

Since the load capacitance, 50 fF, is much larger than the output capacitance of the NAND gate, we don't include the NAND gate's capacitance contributions in the digital model seen in Fig. 13.31c. When the input goes high, the NMOS device in the inverter turns on and the PMOS device in the NAND gate turns on (causing the output to go high). The low-to-high delay time is calculated as

$$t_{PLH} = 0.7 \cdot 3.4k \cdot 4.7f + 0.7 \cdot 3.4k \cdot 50f = 130\,ps$$

The high-to-low delay time is calculated, noting the two NMOS in series when pulling the output low, as

$$t_{PHL} = 0.7 \cdot 3.4k \cdot 4.7f + 0.7 \cdot 6.8k \cdot 50f = 250\,ps$$

Figure 13.32 shows the simulation results. ■

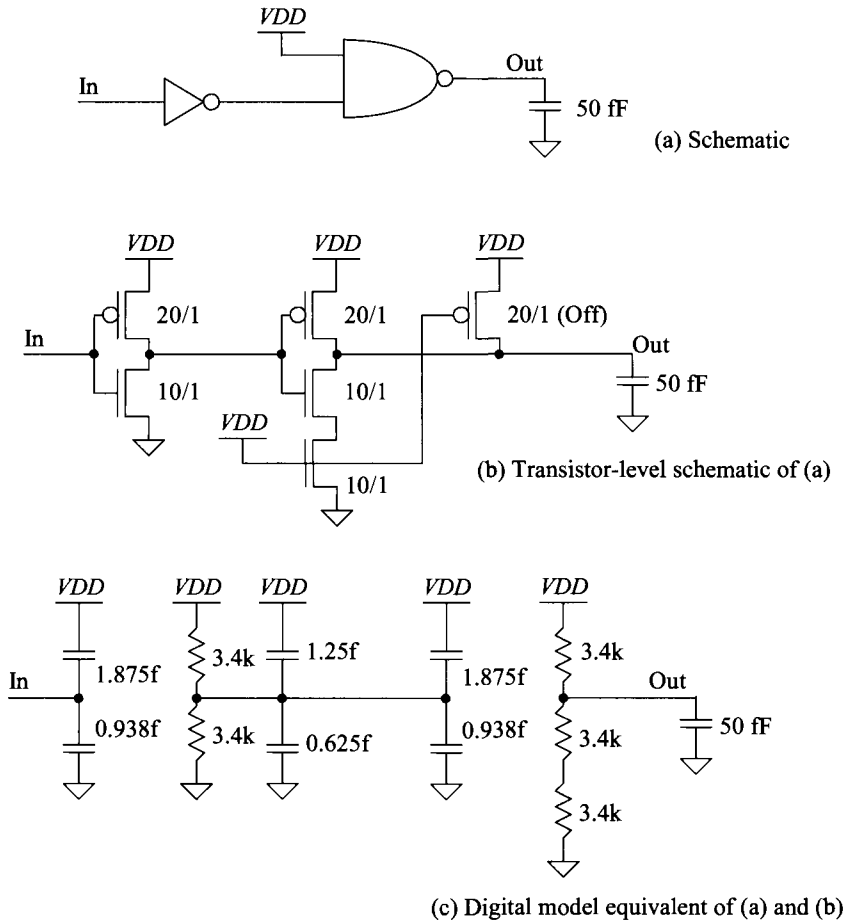


Figure 13.31 Circuit used in Ex. 13.5.

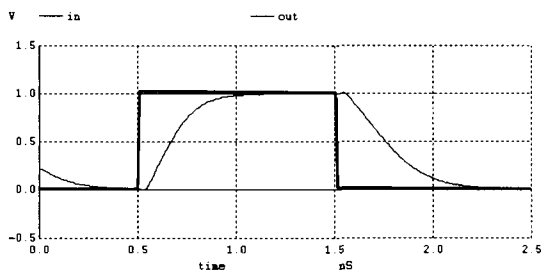


Figure 13.32 Simulating the circuit in Fig. 13.31.

Example 13.6

The circuit seen in Fig. 13.33 is the input portion of the edge triggered D-FF seen in Fig. 13.22. Estimate the delay from the D input to points A and B. Compare the estimate to SPICE simulations. Note that this delay is important because it directly determines the setup time of the FF.

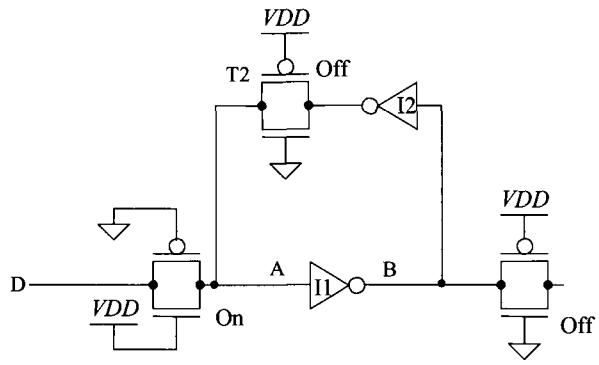


Figure 13.33 Section of a D-FF used in Ex. 13.5.

Node A in Fig. 13.33 is driven through the on TG. The resistance of the TG is $R_n||R_p$. The capacitance on node A is

$$C_A = \overbrace{\frac{C_{oxn}}{2}+\frac{C_{oxp}}{2}}^{\text{Capacitance from the on TG}} + \overbrace{\frac{3}{2}\cdot(C_{oxn}+C_{oxp})}^{\text{Input capacitance of the inverter}} = 3.75\,fF$$

The first term is the capacitance on node A due to the on TG (see Fig. 10.7). The second term is the input capacitance of the inverter. Note that we have not included the capacitive loading from the off TG. This causes the actual delays to be longer than what we calculate here. (A good exercise at this point is to estimate the capacitive loading of the TGs and include them in the calculations here.) The capacitance on node B is made up of an inverter input capacitance and an inverter output capacitance or

$$C_B = \overbrace{\frac{3}{2}\cdot(C_{oxn}+C_{oxp})}^{\text{Inverter input capacitance}} + \overbrace{(C_{oxn}+C_{oxp})}^{\text{Inverter output capacitance}} = 4.7\,fF$$

The delay from the D input to point A is estimated as

$$t_{PLHA} = t_{PHLA} = 0.7\cdot R_n||R_p\cdot C_A = 4.5\,ps$$

The delay from point A to point B through the inverter is

$$t_{PLHB} + t_{PHLB} = 0.7\cdot(R_n+R_p)\cdot C_B = 22.5\,ps$$

The propagation delay from D to B is then

$$t_{PLH} = t_{PHL} = 15.75\ ps$$

The simulation results are seen in Fig. 13.34. The delay to point A is approximately 30 ps, while the delay to point B is about 90 ps. It's important to reiterate the importance of simulations when determining delays. Hand calculations, as this example shows, have limitations. Hand calculations are still important, however, because they reveal the location of the dominant delays in a digital circuit. ■

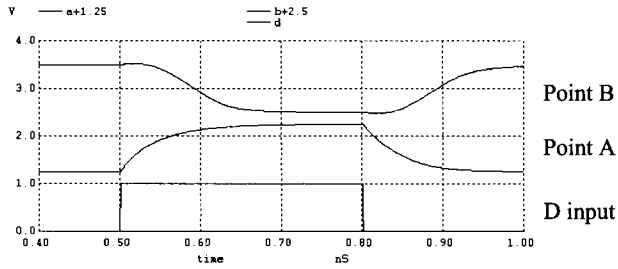


Figure 13.34 Simulating the delays through the latch seen in Fig. 13.33.

ADDITIONAL READING

[1] M. I. Elmasry, *Digital MOS Integrated Circuits II,* IEEE Press, 1992. ISBN 0-87942-275-0, IEEE order number: PC0269-1.

[2] J. P. Uyemura, *Circuit Design for Digital CMOS VLSI,* Kluwer Academic Publishers, 1992.

[3] M. Shoji, *CMOS Digital Circuit Technology,* Prentice-Hall, 1988. ISBN 0-13-138850-9.

PROBLEMS

Unless otherwise stated, use the 50 nm, short-channel CMOS process with the parameters seen in Table 10.2.

13.1 Estimate and simulate the delay through 10 TGs connected a 50 fF load capacitance.

13.2 Design and simulate the operation of a half adder circuit using TGs.

13.3 Sketch the schematic of an 8-to-1 DEMUX using NMOS PGs. Estimate the delay through the DEMUX when the output is connected to a 50 fF load capacitance.

13.4 Verify, using SPICE, that the circuit seen in Fig. 13.12 operates as an XOR gate.

13.5 Simulate the operation of an SR latch made with NAND gates. Show all four possible input logic combinations.

13.6 Simulate the operation of the arbiter seen in Fig. 13.15. Show how two inputs arriving at nearly the same time results in only one output going high.

13.7 Show, using simulations, how making the feedback inverter, I2, in Fig. 13.18 stronger (decrease the lengths) can result in the output having either a long delay or not fully switching.

13.8 Redesign the FF in Fig. 13.22 without T2 and T4 present. Simulate the operation of your design. Show, by using a limited amplitude on the D input, how point B can have a metastability problems.

13.9 In your own words describe setup and hold times. Use the D-FF in Fig. 13.22 and simulations to help support your clear descriptions.