Chapter 6 Implementation of Logic Circuits with Standard Logic Chips

Once logic circuits have been designed with generic logic devices, the next step is to implement the design in physical hardware. The first approach we will study is to use *standard logic chips*, integrated circuits that contain a small number of logic devices. The second approach, using programmable logic hardware, will be addressed in Chapter 8.

To implement a design with logic chips, it is necessary to know what chips are available and what their limitations are. Real devices need to be connected to power, and sometimes there will be unused gates or inverters we'll have to deal with. With a generic design, a single output can drive dozens of inputs, but with real devices there is a limit. The outputs of real devices also don't change instantly when an input changes, so if the design is time critical, this delay must be analyzed and minimized.

6.1 Logic Families

In order to build a digital circuit using standard logic chips, it is necessary to understand *logic families*. A logic family is a set of integrated circuits that implement various logical operations using the same technology, voltage supply and digital signal levels. As long as a circuit is made from a common logic family, all the inputs and outputs will be compatible. Logic families are often grouped by technology as shown in Table 6-1.

Table 6-1 Some Popular Logic Families

Technology	Family	Description
TTL	LS	Low Power Schottky
	LV	Low Voltage
	ALS	Advanced Low Power Schottky
CMOS	HC/HCT	High Speed CMOS (HCT is TTL compatible)
	AC/ACT	Advanced CMOS (ACT is TTL compatible)
	AHC/AHCT	Advanced High Speed CMOS (AHCT is TTL compatible)

Some authors use the term *logic family* to refer to the *technology* and the term *logic subfamily* to refer to the *logic family*. This may seem confusing, but usually the meaning is clear from context.

The industry standard for logic device nomenclature today is the 5400/7400 series (originally developed by Texas Instruments); it is by far the most common and encompasses a great many logic families. The difference between the 5400 and the 7400 series is that the 5400 series devices operate over the military temperature range of -55°C to 125°C, and the less expensive 7400 series devices need only operate over the commercial temperature range of 0°C to 70°C.

Under this nomenclature, each device has a part number that conforms to the format shown in Figure 6-1.

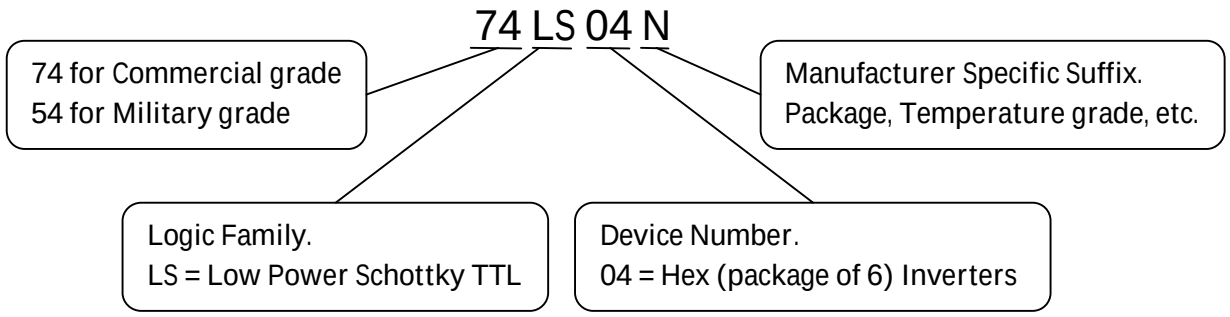
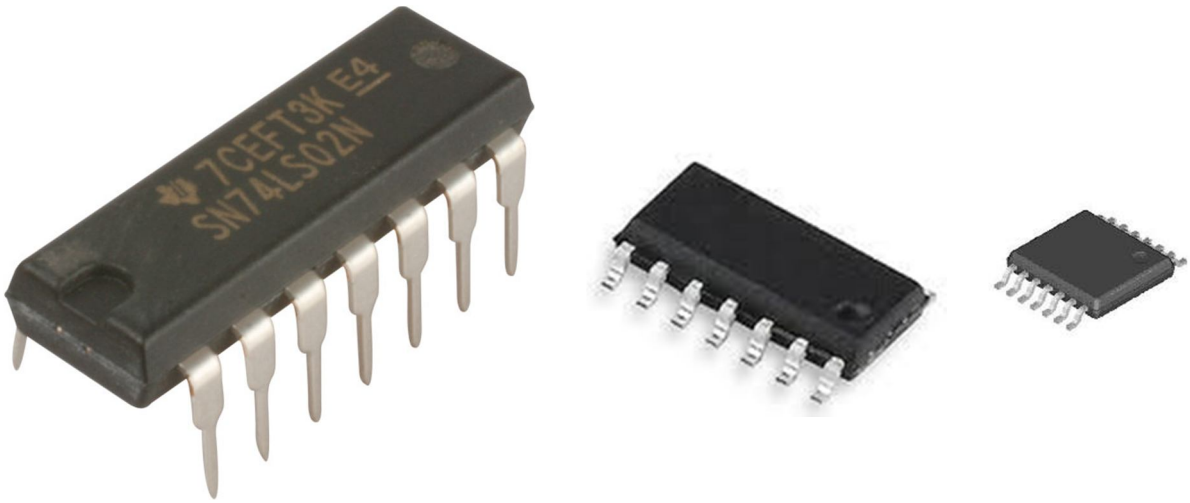


Figure 6-1 Part Number Format for Most Logic Devices.

Integrated circuits (ICs) can be packaged in many different ways. The oldest package is the DIP (*dual in-line package*) or PDIP (*plastic dual in-line package*) shown in Figure 6-2a. The pins of the package are designed to be inserted through holes in a printed circuit board and soldered, but the pins work well in sockets and prototype boards as well. When devices started to be mounted directly to the surface of printed circuit boards, however, a different (and smaller) package was required. The first was the *Small Outline Integrated Circuit* (SOIC or SO) package (Figure 6-2b). This package soon was available in *narrow*, *wide* and *micro*. Later came the *Small Outline Package* (SOP), and that was soon available in *thin* and *shrink* varieties. (There are even more IC packaging options, but they are more suitable for ICs with more pins than logic chips usually have.)



(a) (b) (c)

Figure 6-2 Packages (a) PDIP, (b) SOIC and (c) SOP.

Not all manufacturers supply all of the package types, and each manufacturer defines its own suffix codes to specify the package (although N usually implies a DIP package). This makes ordering parts particularly challenging, and it is important to take extra care to get the correct package.

The set of logic operations available for these logic families is rich and diverse, for example, there are inverters, buffers, nand-gates, nor-gates and-gates and or-gates. Nand gates, for example, have 2-input, 3-input, 4-input and 8-input versions, and if more inputs are needed, multiple logic devices can be combined. The device numbers for these basic logic functions are listed in Table 6-2.

Table 6-2 Basic 7400 Series Logic Devices

Device	Description
74xx00	Quad, 2-input NAND gate
74xx02	Quad, 2-input NOR gate
74xx04	Hex Inverter
74xx08	Quad, 2-input AND gate
74xx10	Triple, 3-input NAND gate
74xx11	Triple, 3-input AND gate

Device	Description
74xx20	Dual, 4-input NAND gate
74xx21	Dual, 4-input AND gate
74xx27	Triple, 3-input NOR gate
74xx28	Dual, 4-input NOR gate
74xx30	8-input NAND gate
74xx32	Quad, 2-input OR gate

6.2 Circuit Implementation with Standard Logic

When logic circuits are implemented in hardware with standard logic chips, three changes must be made to the schematic: First, all the chips in a design must have a designator (usually the letter U followed by a number). It is not unusual for a single chip to contain several logic devices. So to designate an individual gate or inverter within an IC, we add a letter (A, B, C, etc.) to the designator. Second, each connection to a device requires a pin number outside of the symbol. The pin assignments can be found in the *datasheet* (a document the gives all the specification for a given part). Third, the schematic must show connections to V_{CC} and ground. (The generic schematics in Chapter 5 do not show power and ground connections, but without them, no logic device will work). To illustrate, consider the circuit from Example 5-2, repeated for convenience in Figure 6-3.

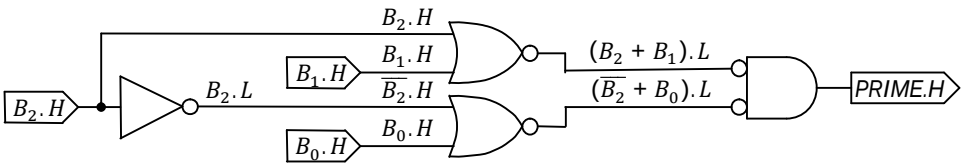


Figure 6-3 Generic Schematic from Example 5-2.

Figure 6-4 illustrates the changes that are necessary to the schematic. Note that each logic device has a designator and pin numbers and that one of them shows the connections to V_{CC} and ground.

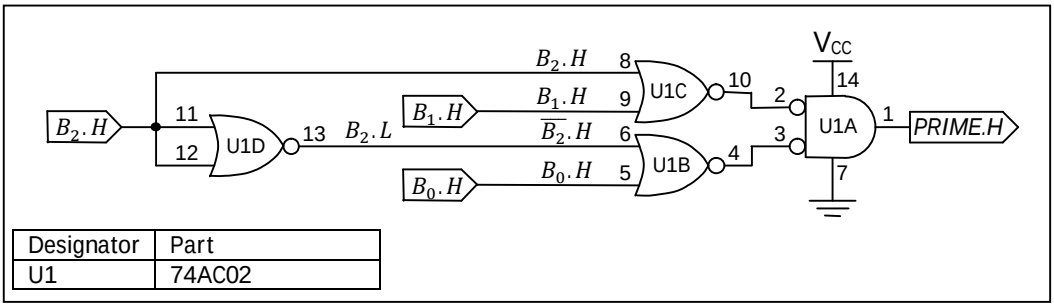


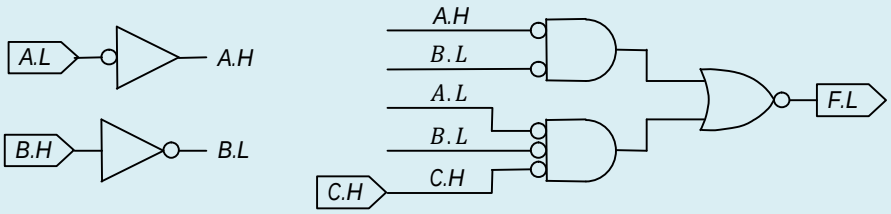
Figure 6-4 Schematic with Logic Devices Completely Specified.

The astute student may have also noticed that the inverter has been replaced by a nor-gate. This kind of subterfuge is common when a circuit is implemented with standard logic chips. Often there will be unused gates that can take the place of inverters or gates with fewer inputs, and often if the substitution is made, a lower chip count will result. For example, had we not made the substitution in Figure 6-4, an inverter (74AC04) would have been required.

If there are unused gates or inverters, it is necessary to connect the inputs to something, usually V_{CC}. The reason for using V_{CC} is historical. TTL input currents are about 20 times greater when an input is low than when it is high, so unused inputs were usually tied high to conserve power. With CMOS it doesn't matter whether an unused input is tied high or low, just so long as it isn't changing.

Example 6-1 Modify the schematic in Figure 5-8 so it can be implemented with Advanced CMOS logic with TTL voltage levels (74ACTxx) parts.

Solution: The schematic from Figure 5-8 is shown below.

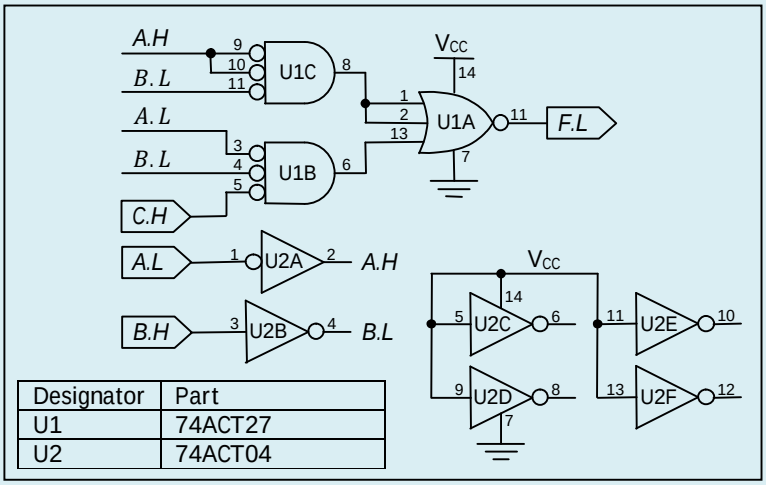


This design can be implemented in one of two ways. Both require a 74AC27, triple 3-input nor – gate. One way uses a 74AC04 and leaves four unused inverters. The other way uses a 74AC02 and leaves two unused 3-input nor-gates. For this example, we will use the 74AC04. The other implementation will be left as an exercise.

Since there are no 2-input nor-gates, we use 3-input nor gates with two inputs tied together instead. With pin numbers, device designators, unused inverters and connections to power, the schematic is as shown on the next page.

Continued on next page

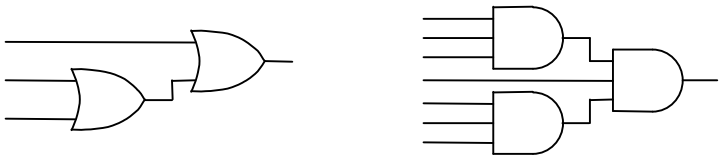
Example 6-1 continued.



Designator	Part
U1	74ACT27
U2	74ACT04

6.3 Gate Expansion

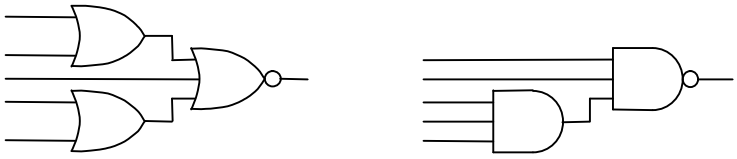
Occasionally, we encounter designs where there is no available gate with a sufficient number of inputs. For and-gates and or-gates, it is a simple matter to expand a gate by adding another gate of the same type. (See Figure 6-5).



(a) (b)

Figure 6-5 (a) Constructing a 3-input or-gate from two 2-input or-gates.
(b) Constructing a 7-input and-gate from three 3-input and gates.

A similar technique may be employed to expand a nor-gate by adding one or more or-gates, or to expand a nand-gate by adding one or more and-gates (Figure 6-6).



(a) (b)

Figure 6-6 (a) Constructing a 5-input nor-gate by adding two 2-input or-gates to a 3-input nor-gate.
(b) Constructing a 5-input nand-gate by adding a 3-input and gate to a 3-input nand gate.

There is another option. We can add parentheses to the original logic expression (and perhaps rearrange terms) so that no sum or product exceeds the number of the available gate inputs. For

example, suppose a circuit needs to generate an active low output, *F.L*, if either signal *A.H* matches *B.H* or signal *C.H* matches *D.H*. Using techniques from Chapter 4, we find:

$$F = AB + \bar{A}\bar{B} + CD + \bar{C}\bar{D} \tag{6.1}$$

The straightforward implementation requires a 4-input nor-gate, but suppose, for the sake of argument, that 4-input nor-gates are not available. If parentheses are added and the design steps from Chapter 5 are then followed, the 4-input nor-gate is not necessary.

First, we re-write the equation by adding parentheses (a clever rearrangement may save inverters, but don't worry about that.)

$$F = (AB + CD) + \bar{A}\bar{B} + \bar{C}\bar{D} \tag{6.2}$$

Now it is clear that *F* is the sum of three terms, and one of those terms is itself a sum. If we proceed with the design technique from Chapter 5, we obtain the generic schematic in Figure 6-7. Note that this schematic can now be implemented with a 3-input nor-gate instead of a 4-input nor gate. Fortunately, the rearrangement of terms made it possible to eliminate all the inverters as well.

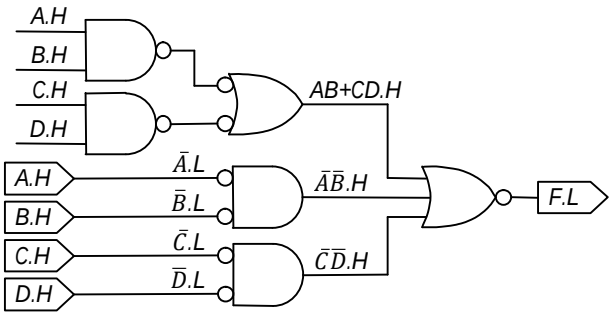


Figure 6-7 Generic Schematic for $F = (AB + CD) + \bar{A}\bar{B} + \bar{C}\bar{D}$.

The hardware implementation of this design is left as an exercise.

6.4 Propagation Delay

Up until now, we have assumed that the output of a gate or inverter immediately reflects the state of its inputs, but that is not true. All real logic devices have *propagation delay*, the time it takes for a change in input to be reflected on the output. Propagation delay depends on many things: the particular device, the logic family, the manufacturer, the temperature, and how well the foundry was running when the device was produced. Propagation delay also depends on whether the change in input causes a high-to-low or low-to-high transition on the output. When the output changes from high to low, the propagation delay is denoted t_{PHL}. When the output changes from low to high, the propagation delay is denoted t_{PLH}.

Most datasheets for logic devices give minimum, typical and maximum propagation delays, such as those shown in Figure 6-8. Note that this part (a 74AC02) will run either on 3.3V or 5V and the

propagation delays are different in each case. Also, this datasheet gives separate propagation delays for an ambient temperature of 25°C and the full range of -40°C – 85°C.

AC Electrical Characteristics

Symbol	Parameter	V_{CC} (V)	T_A = 25°C, C_L = 50pF			T_A = –40°C to 85°C, C_L = 50pF		Units
			Min	Typ	Max	Min	Max	
t_{PLH}	Propagation Delay	3.3	2.0	7.0	9.5	2.0	10.0	ns
		5.0	1.5	6.0	8.0	1.5	8.5	
t_{PHL}	Propagation Delay	3.3	1.5	5.5	8.0	1.0	8.5	ns
		5.0	1.5	4.5	6.5	1.0	7.0	

Figure 6-8 Excerpt from a 74AC02 Datasheet (Fairchild Semiconductor)

For the purposes of circuit design, we must always assume the worst case propagation delay. For example, if the device in Figure 6-8 runs on 5V but is not constrained to run at room temperature (25°C), then we would choose the maximum propagation delay the rightmost column and conclude that the maximum propagation delays are t_{PLH} = 8.5ns and t_{PHL} = 7.0ns.

Datasheets always indicate what load circuitry is attached to an output when the propagation delay is measured. This load is designed to mimic other inputs attached to the output under test. Figure 6-8 shows that, for the 74AC02, the load circuitry consists of a 50pF capacitor. (We'll get to capacitors in Section 6.5.) For this logic family, it turns out that an output can be connected to more than ten other inputs before it will exceed this load. The point here is that unless an output is overloaded, we can rely on the propagation delay data in the datasheet.

So, now that we know how to find the maximum propagation delay through an individual gate, the next question is: what is the maximum propagation delay through the entire logic circuit? To answer that, we must be able to compute the propagation delay from each input, through a signal path to an output. The slowest such path sets the propagation delay for the entire circuit.

Computing the propagation delay through a path is not difficult if the circuit was designed as described in Chapter 5. It takes three steps: a) calculate the propagation delay if a change on the input causes the output to become active (asserted), b) calculate the propagation delay if a change on the input causes the output to become inactive, and c) select the maximum of the two.

To calculate part (a), add the t_{PHL} of each gate with a bubble on its output and t_{PLH} of each gate without an output bubble. If there is an inverter, add t_{PHL} if it connects to a gate with an input bubble or t_{PLH} if it connects to a gate without an input bubble. Repeat this procedure for part (b) but exchange t_{PHL} with t_{PLH}. As an illustration, consider the schematic diagram in Figure 6-4. There are four signal paths from the inputs to the output:

Path Number	Signal Path
1	B_2.*H*→U1C→U1A→*PRIME.H*
2	B_1.*H*→U1C→U1A→*PRIME.H*
3	B_2.*H*→U1D→U1B→U1A→*PRIME.H*
4	B_0.*H*→ U1B→U1A→*PRIME.H*

Figure 6-9 Signal Paths for the Schematic Shown in Figure 6-4

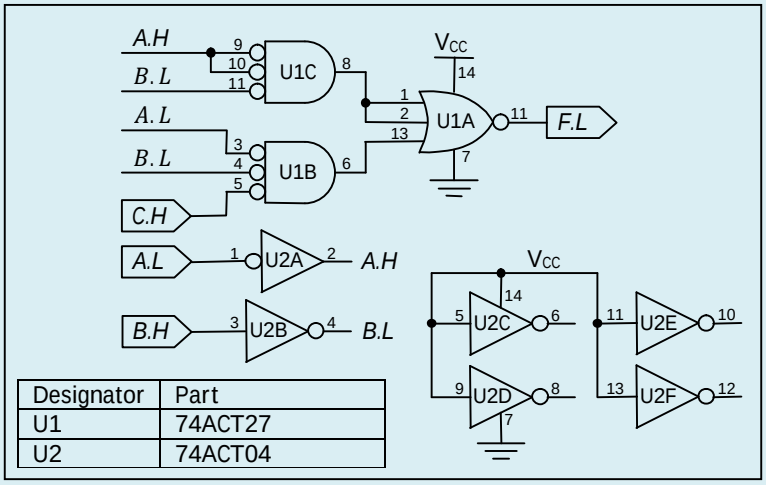
Clearly path 3 is the longer path because it contains one extra gate, and since the objective is to find the longest path, we can ignore the others. The propagation delays for each gate in the circuit are t_{PLH} = 8.5ns and t_{PHL} = 7.0ns because the entire circuit consists of 74AC02 nor-gates. Figure 6-10 shows the calculation of propagation delay using the technique outlined earlier.

Device Description	Activation Delay	Deactivation Delay
U1D (inverter not connected to an input bubble)	8.5ns (t_{PLH})	7.0ns (t_{PHL})
U1B (gate with output bubble)	7.0ns (t_{PHL})	8.5ns(t_{PLH})
U1A (gate with no output bubble)	8.5ns(t_{PLH})	7.0ns (t_{PHL})
Total:	24.0ns	22.5ns

Figure 6-10 Calculation of the Activation and Deactivation Propagation Delays for *PRIME.H*.

The propagation delay for the entire digital logic circuit is the maximum of these two totals, which in this case is 24ns.

Example 6.2. Find the maximum propagation delay through the circuit in Example 6.1. Assume the propagation delay through the 74ACT04 is t_{PHL} = 8.5 ns and t_{PLH} = 9.0 ns, and that the propagation delay through the 74ACT27 is t_{PHL} = 10.0 ns and t_{PLH} = 10.5 ns. The schematic is repeated here for convenience.



Designator	Part
U1	74ACT27
U2	74ACT04

Solution: Clearly the paths that go through the inverters are longer than those that do not, so we will consider only the paths *A.L*→U2A→U1C→U1A→*F.L* and *B.H*→U2B→U1B→U1A→*F.L*.
To calculate the delay for the first path, we have:

Device Description	Activation Delay	Deactivation Delay
U2A (inverter connected to an input bubble)	8.5ns (t_{PHL})	9.0ns (t_{PLH})
U1C (gate with no output bubble)	10.5ns (t_{PLH})	10.0ns (t_{PHL})
U1A (gate with output bubble)	10.0ns (t_{PHL})	10.5ns (t_{PLH})
Total:	29.0ns	29.5ns

The second path has exactly the same configuration, so it has the same propagation delay. The maximum propagation delay, then, is the larger of the activation and deactivation delay, which in this case is 29.5 ns.

6.5 Sum of Products and Product of Sums

Sometimes, the simplest logic expression is not the one with the smallest propagation delay. If propagation delay is important, it is often best to express a logical expression as a *Sum of Products* (*SOP*) or a *Product of Sums* (*POS*) before implementing it. As the name implies, a sum of products is a logical expression consisting of a sum of terms, with each term being the product of one or more (possibly complemented) Boolean variables. For example $\bar{A}B\bar{C}+\bar{A}CD$ is a sum of products, while the equivalent (and simpler) expression, $\bar{A}(B\bar{C}+CD)$, is not. A product of sums, on the other hand, is a logical expression consisting of a product of factors, with each factor being the sum of one or more variables. As an example, $(A+\bar{B})(A+C+\bar{D})$ is a product of sums, but the expressions $(A+\bar{B})(\overline{C+D})$ and $(A+\bar{B})(C+\bar{B}D)$ are not.

The most important property of *SOP* and *POS* expressions is that when they are implemented in hardware, the longest input/output signal path is limited to an inverter and two gates.

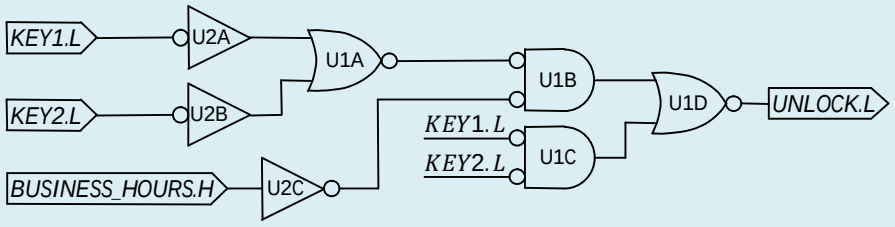
Example 6.3 Implement the logic expression for *UNLOCK* in Example 4.8 using only nand-gates, nor-gates and inverters. Then implement the equivalent *SOP* expression and compare the propagation delays. Assume KEY1, KEY2 and UNLOCK are active low, BUSINESS_HOURS is active high and that the propagation delays are as shown below:

Device	T_{PHL}	T_{PLH}
Inverter	7.0ns	7.5ns
2-input gate	9.0ns	9.5ns
3-input gate	10.0ns	10.5ns

Solution: The expression for UNLOCK from Example 4.8 is repeated here for convenience:

$$UNLOCK = (KEY1+KEY2)\cdot BUSINESS\_HOURS + KEY1\cdot KEY2$$

Using the technique from Chapter 5, this expression is implemented as shown below:



The two longest paths for this implementation are: *KEY1.L*→U2A→U1A→U1B→U1D→*UNLOCK.L* and *KEY2.L*→U2B→U1A→U1B→U1D→*UNLOCK.L*. Both paths contain the same devices in the same order, so we need analyze only one. Computing the delay for the first path, we find it to be 35ns:

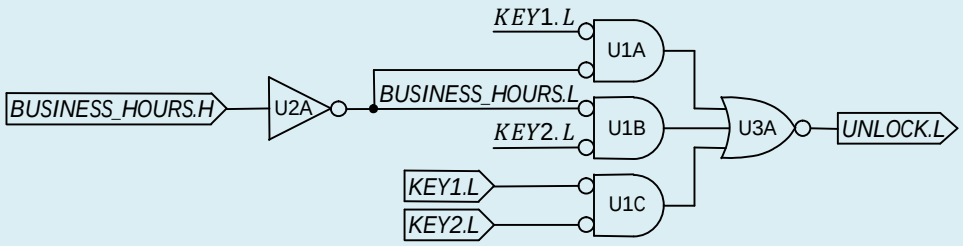
Device	Activation Delay	Deactivation Delay
U2A (inverter not connected to an input bubble)	7.5ns (t_{PLH})	7.0ns (t_{PHL})
U1A (gate with output bubble)	9.0ns (t_{PHL})	9.5ns (t_{PLH})
U1B (gate with no output bubble)	9.5ns (t_{PLH})	9.0ns (t_{PHL})
U1D (gate with output bubble)	9.0ns (t_{PHL})	9.5ns (t_{PLH})
Total:	35.0ns	35.0ns

Continued on next page...

Using property 4.21a, we find the equivalent SOP expression:

$$UNLOCK = KEY1 \cdot BUSINESS\_HOURS + KEY2 \cdot BUSINESS\_HOURS + KEY1 \cdot KEY2$$

The implementation of the SOP expression is shown below:



The longest paths is: *BUSINESS_HOURS.H*→U2A→U1A→U3A→*UNLOCK.L*. (The path through U1B is identical). Computing the delays, we find that the total propagation delay is 27ns:

Device	Activation Delay	Deactivation Delay
U2A (inverter connected to an input bubble)	7.0ns (t_{PHL})	7.5ns (t_{PLH})
U1C (gate with no output bubble)	9.5ns (t_{PLH})	9.0ns (t_{PHL})
U1A (3-input gate with output bubble)	10.0ns (t_{PHL})	10.5ns (t_{PLH})
Total:	26.5ns	27.0ns

The SOP implementation has a total propagation delay of 27ns, compared to 35ns in the original implementation.

Remember that a sum or product may consist of just one term or factor. So, for example, *A+BC* is an *SOP* expression and *(A+B)CD* is a *POS* expression. Curiously, *ABC* is both an *SOP* and a *POS* expression; it is the sum of one product, *ABC*, and it is the product of three sums, *A*, *B* and *C*.

Every logic expression has at least one equivalent *SOP* and *POS* form. Recall from Chapter 4 that any logic expression can be written as a sum of minterms or as a product of maxterms. These are, by definition, SOP and POS expressions respectively, but they are usually not the *simplest* SOP or POS forms. We will discuss a technique to find the simplest SOP or POS form for any logic expression in Chapter 7.

6.6 Fan-Out

In Section 6.4, we alluded to the fact that a digital output has a maximum number of inputs that it can drive. This number of inputs is called the *fan-out*.

For technologies other than CMOS, fan-out is primarily limited by the ability of an output to source or sink current from the other inputs. The guaranteed output current of a device is denoted I_{OH} or I_{OL} in the datasheet, depending on whether the output is high or low. (Current is always considered to flow into a device, so I_{OL} is positive and I_{OH} is negative.) The maximum current required for an input is denoted I_{IH} or I_{IL}, depending on whether the input is high or low. The two relationships that must be maintained between these currents are given by inequalities 6.3 and 6.4.

$$I_{OL} \geq -\sum_{\text{connected inputs}} I_{IL} \tag{6.3}$$

$$-I_{OH} \geq \sum_{\text{connected inputs}} I_{IH} \tag{6.4}$$

These relationships simply mean that the maximum output current must exceed the sum of the input currents. Consider, for example, the schematic fragment shown in Figure 6-11. The output of the inverter U1B is connected to four inputs, one on U1 and three on U2.

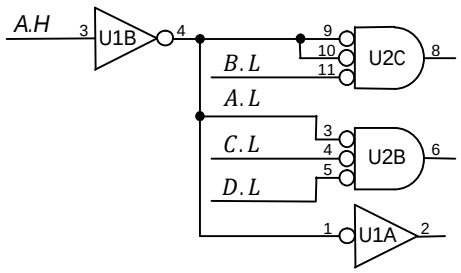


Figure 6-11 Fan-out Calculation

Suppose U1 is an ON semiconductor 74LS04 and U2 is a Texas Instruments 74LS27. Based on their datasheets, the input and output current specifications are as shown in Table 6-1.

Table 6-1 Input/Output Current Specifications for Various Devices

Parameter	(ON) 74LS04	(TI) 74LS27
I_{OH}	–0.4 mA	–0.4 mA
I_{OL}	8.0 mA	16.0 mA
I_{IH} (maximum)	20.0 uA	40.0 uA
I_{IL} (maximum)	–0.4 mA	–1.6 mA

From these specifications we can determine whether or not the output of the inverter (U1B) is overloaded. Applying Inequality 6.3 we see that

$$8\text{mA} \geq -(-1.6\text{mA} - 1.6\text{mA} - 1.6\text{mA} - 0.4\text{mA}) = 5.2\text{mA}. \tag{6.5}$$

From this we can infer that the output is not overloaded when it is low. Next we must check that the output is not overloaded when it's high. The application of Inequality 6.4 gives us

$$0.4\text{mA} \geq 40\mu\text{A} + 40\mu\text{A} + 40\mu\text{A} + 20\mu\text{A} = 0.14\text{mA}. \tag{6.6}$$

From this we can see that the output is likewise not overloaded when it is high.

There is an easier way to verify that none of the outputs in a circuit are overloaded, but it requires that all the chips be in the same logic family and come from the same manufacturer (or at least have the same I_{OH}, I_{OL}, I_{IH} and I_{IL}). If this is so, Inequalities 6.3 and 6.4 reduce to

$$I_{OL} \geq -nI_{IL} \tag{6.7}$$

and

$$-I_{OH} \geq nI_{IH} \tag{6.8}$$

where *n* is the number of inputs connected to the output. Both inequalities are satisfied if

$$n \leq \min\left(-\frac{I_{OL}}{I_{IL}}, -\frac{I_{OH}}{I_{IH}}\right) \tag{6.9}$$

The value of *n* is the fan-out for the logic family, and as long as the number of connections to an output are less than or equal to *n*, there is no danger that the output will be overloaded. Since the number of connections is integral, *n* is typically rounded down to the next lower integer.

For TTL logic families, any value of *n* that satisfies Inequality 6.7 also satisfies Inequality 6.8. This means that Inequality 6.9 can be simplified to:

$$n \leq -\frac{I_{OL}}{I_{IL}}. \tag{6.10}$$

For CMOS logic families, $I_{IL} \approx 0$ and $I_{IH} \approx 0$, so the value of *n* derived from Equation 6.9 is larger than the number of connections likely to occur in any practical circuit. There is another consideration, however, and that is the *capacitance*.

Recall from Chapter 1 that electric potential, or voltage, is created by separating charge, and to get more voltage it is necessary to separate more charge. The ratio of charge to voltage is constant and is called capacitance. Capacitance is measured in Farads. (A Farad is, by definition, a Coulomb per Volt.)

For example, a typical CMOS input has an input capacitance (C_{IN}) of about 4.5pF (picofarads). This means that if 4.5pC (picocoulombs) of charge is separated from ground and placed on an input, its voltage will rise by 1V. Likewise, if twice that amount, 9pC, is placed on the input, its voltage will rise by 2V. Depending on V_{CC}, about 15-30pC of charge must be transferred to or from an input to make it go high or low, respectively. When multiple inputs are connected together, the effective capacitance seen by the output is the sum of the input capacitances. The greater that sum, the more charge that has to be moved per transition, and the longer it takes to do it.

The propagation delay times listed in CMOS datasheets specify a load capacitance, C_L, (typically 50pF). If the effective capacitance is less than C_L, the propagation delays advertised in the datasheet will be valid. If not, the circuit will still work, but will be slower. We cannot know exactly how much slower based only on the information in the datasheet, so if the digital circuit is time-critical, it is best to maintain the following relationship:

$$C_L \geq \sum_{\text{connected inputs}} C_{IN} \tag{6.11}$$

(This relationship is an oversimplification. The summation should also include the capacitance for traces on the printed circuit board, which can be substantial if they are long, but calculating that capacitance is beyond the scope of this text.)

Again, if a common logic family from the same manufacturer is used, or at least if C_{IN} and C_L are the same, Inequality 6.11 reduces to

$$C_L \geq nC_{IN} \tag{6.12}$$

where *n* is the number of inputs connected to the output. Solving for *n*, we get a limit on the number of inputs that can be connected to an output without compromising its propagation delay:

$$n \leq \frac{C_L}{C_{IN}} \tag{6.13}$$

If a design violates the Inequalities 6.3, 6.4 or 6.11, what can be done to reduce the number of inputs connected to a particular output? The short answer is: make more outputs. This can be accomplished by duplicating gates or inverters as shown in Figure 6-12.

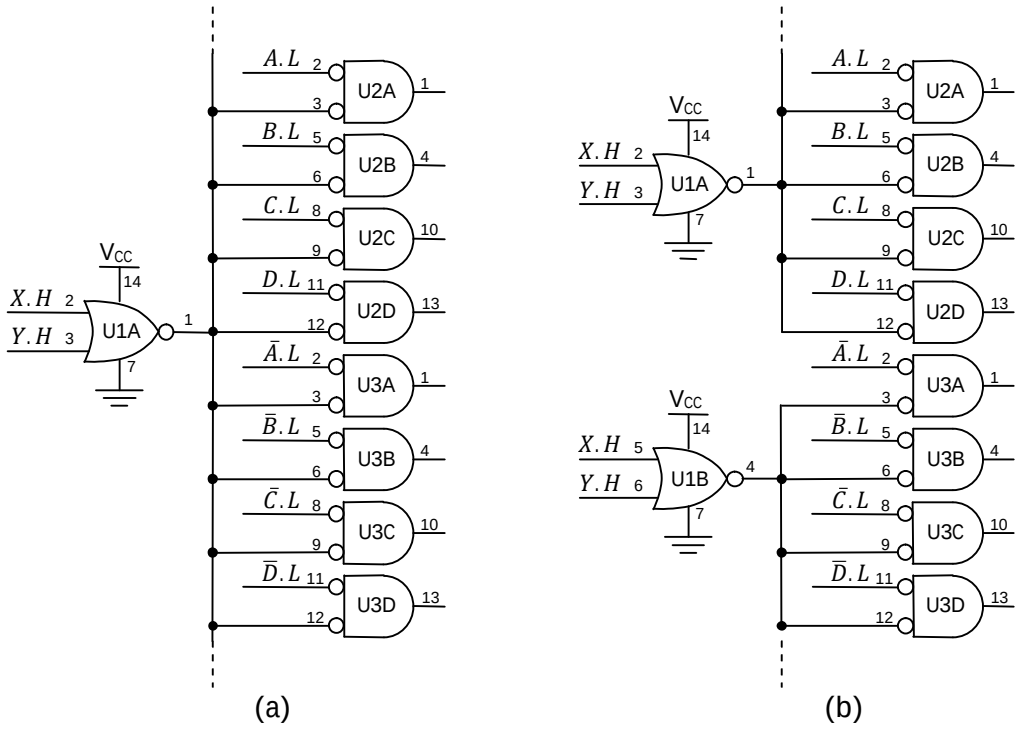


Figure 6-12 Fan-out (a) Too Many Inputs per Output, (b) Number of Inputs per Output cut in Half.

Exercises

1. Implement the circuit in Example 6-1 using a 74AC27 and a 74AC02.
2. Implement the circuit in Figure 6-7 using Advanced Low Power Schottky.
3. Is ($A+\bar{B}+\bar{C}$) an SOP expression? Is it a POS expression? Is it both? Explain.

Bibliography