▶ 2.13 CHAPTER SUMMARY

Section 2.1 introduced the idea of using a custom digital circuit to implement a system's desired functionality and defined combinational logic as a digital circuit whose outputs are a function of the circuit's present inputs. Section 2.2 provided a brief history of digital switches, starting from relays in the 1930s to today's CMOS transistors, with the main trend being the amazing pace at which switch size and delay have continued to shrink for the past several decades, leading to ICs capable of containing a billion transistors or more. Section 2.3 described the basic behavior of a CMOS transistor, just enough information to remove the mystery of how transistors work.

Section 2.4 introduced three fundamental building blocks for building digital circuits—AND gates, OR gates, and NOT gates (inverters), which are far easier to work with than transistors. Section 2.5 showed how Boolean algebra could be used to represent circuits built from AND, OR, and NOT gates, enabling us to build and manipulate circuits by using math—an extremely powerful concept. Section 2.6 introduced several different representations of Boolean functions, namely equations, circuits, and truth tables.

Section 2.7 described a straightforward three-step process for designing combinational circuits, and gave several examples of building real circuits using the three-step process.

Section 2.8 described why NAND and NOR gates are actually more commonly used than AND and OR gates in CMOS technology, and showed that any circuit built from AND, OR, and NOT gates could be built with NAND gates alone or NOR gates alone. That section also introduced two other commonly used gates, XOR and XNOR. Section 2.9 introduced two additional commonly used combinational building blocks, decoders and multiplexers.

Section 2.10 discussed how real gates actually have a small delay between the time that inputs change and the time that the gate's output changes. The section introduced active low inputs, and it also introduced some less commonly used combinational building blocks, demultiplexers and encoders. The section introduced schematic capture tools, which allow designers to draw circuits such that computer programs can read those circuits. The section also introduced simulation, which generates the output waveforms for designer-provided input waveforms, to help a designer verify that a circuit is correct.

▶ 2.14 EXERCISES

An asterisk (*) indicates an especially challenging problem.

SECTION 2.2: SWITCHES

2.1 A microprocessor in 1980 used about 10,000 transistors. How many of those microprocessors would fit in a modern chip having 3 billion transistors?

2.2 The first Pentium microprocessor had about 3 million transistors. How many of those microprocessors would fit in a modern chip having 3 billion transistors?

2.3 Describe the concept known as Moore's Law.

2.4 Assume for a particular year that a particular size chip using state-of-the-art technology can contain 1 billion transistors. Assuming Moore's Law holds, how many transistors will the same size chip be able to contain in ten years?

2.5 Assume a cell phone contains 50 million transistors. How big would such a cell phone be if the phone used vacuum tubes instead of transistors, assuming a vacuum tube has a volume of 1 cubic inch?

2.6 A modern desktop processor may contain 1 billion transistors in a chip area of 100 mm^2. If Moore's Law continues to apply, what would be the chip area for those 1 billion transistors after 9 years? What percentage is that area of the original area? Name a product into which the smaller chip might fit whereas the original chip would have been too big.

SECTION 2.3: THE CMOS TRANSISTOR

2.7 Describe the behavior of the CMOS transistor circuit shown in Figure 2.77, clearly indicating when the transistor circuit conducts.

2.8 If we apply a voltage to the gate of a CMOS transistor, why wouldn't the current flow from the gate to the transistor's source or drain?

2.9 Why does applying a positive voltage to the gate of a CMOS transistor cause the transistor to conduct between source and drain?

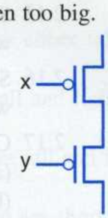


Figure 2.77 Circuit combining two CMOS transistors.

SECTION 2.4: BOOLEAN LOGIC GATES—BUILDING BLOCKS FOR DIGITAL CIRCUITS

2.10 Which Boolean operation—AND, OR, or NOT—is appropriate for each of the following:

(a) Detecting motion in any motion sensor surrounding a house (each motion sensor outputs 1 when motion is detected).

(b) Detecting that three buttons are being pressed simultaneously (each button outputs 1 when a button is being pressed).

(c) Detecting the absence of light from a light sensor (the light sensor outputs 1 when light is sensed).

2.11 Convert the following English problem statements to Boolean equations. Introduce Boolean variables as needed.

(a) A flood detector should turn on a pump if water is detected and the system is set to enabled.

(b) A house energy monitor should sound an alarm if it is night and light is detected inside a room but motion is not detected.

(c) An irrigation system should open the sprinkler's water valve if the system is enabled and neither rain nor freezing temperatures are detected.

2.12 Evaluate the Boolean equation F = (a AND b) OR c OR d for the given values of variables a, b, c, and d:

(a) a=1, b=1, c=1, d=0

(b) a=0, b=1, c=1, d=0

(c) a=1, b=1, c=0, d=0

(d) a=1, b=0, c=0, d=0

2.13 Evaluate the Boolean equation F = a AND (b OR c) AND d for the given values of variables a, b, c, and d:

(a) a=1, b=1, c=0, d=1

(b) a=0, b=0, c=0, d=1

(c) a=1, b=0, c=0, d=0

(d) a=1, b=0, c=1, d=1

2.14 Evaluate the Boolean equation F = a AND (b OR (c AND d)) for the given values of variables a, b, c, and d:
(a) a=1, b=1, c=0, d=1
(b) a=0, b=0, c=0, d=1
(c) a=1, b=0, c=0, d=0
(d) a=1, b=0, c=1, d=1

2.15 Show the conduction paths and output value of the OR gate transistor circuit in Figure 2.12 when: (a) x = 1 and y = 0, (b) x = 1 and y = 1.

2.16 Show the conduction paths and output value of the AND gate transistor circuit in Figure 2.14 when: (a) x = 1 and y = 0, (b) x = 1 and y = 1.

2.17 Convert each of the following equations directly to gate-level circuits:
(a) F = ab' + bc + c'
(b) F = ab + b'c'd'
(c) F = ((a + b') * (c' + d)) + (c + d + e')

2.18 Convert each of the following equations directly to gate-level circuits:
(a) F = a'b' + b'c
(b) F = ab + bc + cd + de
(c) F = ((ab)' + (c)) + (d + ef)'

2.19 Convert each of the following equations directly to gate-level circuits:
(a) F = abc + a'bc
(b) F = a + bcd' + ae + f'
(c) F = (a + b) + (c' * (d + e + fg))

2.20 Design a system that sounds a buzzer inside a home whenever motion outside is detected at night. Assume a motion sensor has an output M that indicates whether motion is detected (M=1 means motion detected) and a light sensor with output L that indicates whether light is detected (L=1 means light is detected). The buzzer inside the home has a single input B that when 1 sounds the buzzer. Capture the desired system behavior using an equation, and then convert the equation to a circuit using AND, OR, and NOT gates.

2.21 A DJ ("disc jockey," meaning someone who plays music at a party) would like a system to automatically control a strobe light and disco ball in a dance hall depending on whether music is playing and people are dancing. A sound sensor has output S that when 1 indicates that music is playing, and a motion sensor has output M that when 1 indicates that people are dancing. The strobe light has an input L that when 1 turns the light on, and the disco ball has an input B that when 1 turns the ball on. The DJ wants the disco ball to turn on only when music is playing and nobody is dancing, and wants the strobe light to turn on only when music is playing and people are dancing. Create equations describing the desired behavior for B and for L, and then convert each to a circuit using AND, OR, and NOT gates,

2.22 Concisely describe the following situation using a Boolean equation. We want to fire a football coach (by setting F=1) if he is mean (represented by M=1). If he is not mean but has a losing season (represented by the Boolean variable L=1), we want to fire him anyway. Write an equation that translates the situation directly to a Boolean equation for F, without any simplification.

SECTION 2.5: BOOLEAN ALGEBRA

2.23 For the function F = a + a'b + acd + c':
(a) List all the variables.
(b) List all the literals.
(c) List all the product terms.

2.24 For the function $F = a'd' + a'c + b'cd' + cd$:
(a) List all the variables.
(b) List all the literals.
(c) List all the product terms.

2.25 Let variables T represent being tall, H being heavy, and F being fast. Let's consider anyone who is not tall as short, not heavy as light, and not fast as slow. Write a Boolean equation to represent each of the following:
(a) You may ride a particular amusement park ride only if you are either tall and light, or short and heavy.
(b) You may NOT ride an amusement park ride if you are either tall and light, or short and heavy. Use algebra to simplify the equation to sum of products.
(c) You are eligible to play on a particular basketball team if you are tall and fast, or tall and slow. Simplify this equation.
(d) You are NOT eligible to play on a particular football team if you are short and slow, or if you are light. Simplify to sum-of-products form.
(e) You are eligible to play on both the basketball and football teams above, based on the above criteria. Hint: combine the two equations into one equation by ANDing them.

2.26 Let variables S represent a package being small, H being heavy, and E being expensive. Let's consider a package that is not small as big, not heavy as light, and not expensive as inexpensive. Write a Boolean equation to represent each of the following:
(a) Your company specializes in delivering packages that are both small and inexpensive (a package must be small AND inexpensive for us to deliver it); you'll also deliver packages that are big but only if they are expensive.
(b) A particular truck can be loaded with packages only if the packages are small and light, small and heavy, or big and light. Simplify the equation.
(c) Your above-mentioned company buys the above-mentioned truck. Write an equation that describes the packages your company can deliver. Hint: Appropriately combine the equations from the above two parts.

2.27 Use algebraic manipulation to convert the following equation to sum-of-products form:
$F = a(b + c)(d') + ac'(b + d)$

2.28 Use algebraic manipulation to convert the following equation to sum-of-products form:
$F = ab(c + d') + a(b' + c) + a(b + d)c$

2.29 Use DeMorgan's Law to find the inverse of the following equation: $F = abc + a'b$. Reduce to sum-of-products form. Hint: Start with $F' = (abc + a'b)'$

2.30 Use DeMorgan's Law to find the inverse of the following equation: $F = ac' + abd' + acd$. Reduce to sum-of-products form.

SECTION 2.6: REPRESENTATIONS OF BOOLEAN FUNCTIONS

2.31 Convert the following Boolean equations to a digital circuit:
(a) $F(a,b,c) = a'bc + ab$
(b) $F(a,b,c) = a'b$
(c) $F(a,b,c) = abc + ab + a + b + c$
(d) $F(a,b,c) = c'$

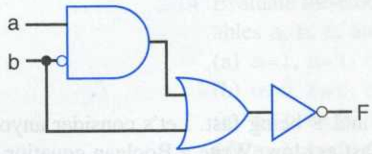


Figure 2.78 Combinational circuit for *F*.

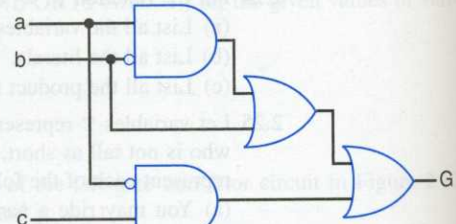


Figure 2.79 Combinational circuit for *G*.

2.32 Create a Boolean equation representation of the digital circuit in Figure 2.78.

2.33 Create a Boolean equation representation for the digital circuit in Figure 2.79.

2.34 Convert each of the Boolean equations in Exercise 2.31 to a truth table.

2.35 Convert each of the following Boolean equations to a truth table:
(a) `F(a,b,c) = a' + bc'`
(b) `F(a,b,c) = (ab)' + ac' + bc`
(c) `F(a,b,c) = ab + ac + ab'c' + c'`
(d) `F(a,b,c,d) = a'bc + d'`

2.36 Fill in Table 2.8's columns for the equation: `F = ab + b'`

TABLE 2.8 Truth table.

Inputs					Output
a	b	ab	b'	ab+b'	F
0	0				
0	1				
1	0				
1	1				

2.37 Convert the function F shown in the truth table in Table 2.9 to an equation. Don't minimize the equation.

TABLE 2.9 Truth table.

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2.38 Use algebraic manipulation to minimize the equation obtained in Exercise 2.37.

2.39 Convert the function F shown in the truth table in Table 2.10 to an equation. Don't minimize the equation.

TABLE 2.10 Truth table.

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2.40 Use algebraic manipulation to minimize the equation obtained in Exercise 2.39.

2.41 Convert the function F shown in the truth table in Table 2.11 to an equation. Don't minimize the equation.

TABLE 2.11 Truth table.

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

2.42 Use algebraic manipulation to minimize the equation obtained in Exercise 2.41.

2.43 Create a truth table for the circuit of Figure 2.78.

2.44 Create a truth table for the circuit of Figure 2.79.

2.45 Convert the function F shown in the truth table in Table 2.9 to a digital circuit.

2.46 Convert the function F shown in the truth table in Table 2.10 to a digital circuit.

2.47 Convert the function F shown in the truth table in Table 2.11 to a digital circuit.

2.48 Convert the following Boolean equations to canonical sum-of-minterms form:
(a) `F(a,b,c) = a'bc + ab`
(b) `F(a,b,c) = a'b`
(c) `F(a,b,c) = abc + ab + a + b + c`
(d) `F(a,b,c) = c'`

2.49 Determine whether the Boolean functions `F = (a + b)'*a` and `G = a + b'` are equivalent, using (a) algebraic manipulation and (b) truth tables.

2.50 Determine whether the Boolean functions `F = ab'` and `G = (a' + ab)'` are equivalent, using (a) algebraic manipulation and (b) truth tables.

2.51 Determine whether the Boolean function `G = a'b'c + ab'c + abc' + abc` is equivalent to the function represented by the circuit in Figure 2.80.

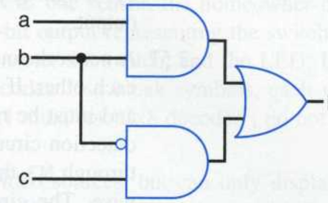


Figure 2.80 Combinational circuit for *H*.

2.52 Determine whether the two circuits in Figure 2.81 are equivalent circuits, using (a) algebraic manipulation and (b) truth tables.

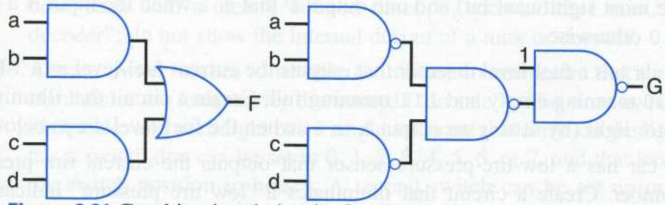


Figure 2.81 Combinational circuits for *F* and *G*.

2.53 * Figure 2.82 shows two circuits whose inputs are unlabeled.

(a) Determine whether the two circuits are equivalent. Hint: Try all possible labelings of the inputs for both circuits.

(b) How many circuit comparisons would need to be performed to determine whether two circuits with 10 unlabeled inputs are equivalent?

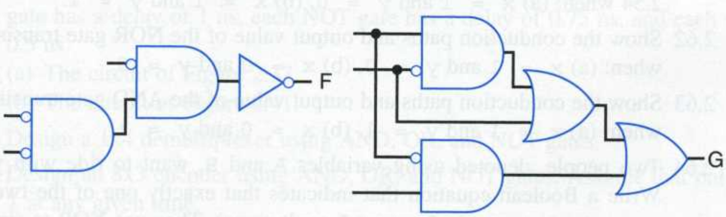


Figure 2.82 Combinational circuits for *F* and *G*.

SECTION 2.7: COMBINATIONAL LOGIC DESIGN PROCESS

2.54 A museum has three rooms, each with a motion sensor (`m0`, `m1`, and `m2`) that outputs `1` when motion is detected. At night, the only person in the museum is one security guard who walks from room to room. Create a circuit that sounds an alarm (by setting an output `A` to `1`) if motion is ever detected in more than one room at a time (i.e., in two or three rooms), meaning there must be one or more intruders in the museum. Start with a truth table.

2.55 Create a circuit for the museum of Exercise 2.54 that detects whether the guard is properly patrolling the museum, detected by *exactly* one motion sensor being `1`. (If no motion sensor is `1`, the guard may be sitting, sleeping, or absent.)

2.56 Consider the museum security alarm function of Exercise 2.54, but for a museum with 10 rooms. A truth table is not a good starting point (too many rows), nor is an equation describing when the alarm should sound (too many terms). However, the inverse of the alarm function can be straightforwardly captured as an equation. Design the circuit for the 10-room security system by designing the inverse of the function, and then just adding an inverter before the circuit's output.

2.57 A network router connects multiple computers together and allows them to send messages to each other. If two or more computers send messages simultaneously, the messages "collide" and must be re-sent. Using the combinational design process of Table 2.5, create a collision detection circuit for a router that connects 4 computers. The circuit has 4 inputs labeled `M0` through `M3` that are `1` when the corresponding computer is sending a message and `0` otherwise. The circuit has one output labeled `C` that is `1` when a collision is detected and `0` otherwise.

2.58 Using the combinational design process of Table 2.5, create a 4-bit prime number detector. The circuit has four inputs—`N3`, `N2`, `N1`, and `N0`—that correspond to a 4-bit number (`N3` is the most significant bit) and one output `P` that is `1` when the input is a prime number and that is `0` otherwise.

2.59 A car has a fuel-level detector that outputs the current fuel-level as a 3-bit binary number, with `000` meaning empty and `111` meaning full. Create a circuit that illuminates a "low fuel" indicator light (by setting an output `L` to `1`) when the fuel level drops below level 3.

2.60 A car has a low-tire-pressure sensor that outputs the current tire pressure as a 5-bit binary number. Create a circuit that illuminates a "low tire pressure" indicator light (by setting an output `T` to `1`) when the tire pressure drops below 16. Hint: you might find it easier to create a circuit that detects the inverse function. You can then just append an inverter to the output of that circuit.

SECTION 2.8: MORE GATES

2.61 Show the conduction paths and output value of the NAND gate transistor circuit in Figure 2.54 when: (a) `x = 1` and `y = 0`, (b) `x = 1` and `y = 1`.

2.62 Show the conduction paths and output value of the NOR gate transistor circuit in Figure 2.54 when: (a) `x = 1` and `y = 0`, (b) `x = 0` and `y = 0`.

2.63 Show the conduction paths and output value of the AND gate transistor circuit in Figure 2.55 when: (a) `x = 1` and `y = 1`, (b) `x = 0` and `y = 1`.

2.64 Two people, denoted using variables `A` and `B`, want to ride with you on your motorcycle. Write a Boolean equation that indicates that exactly one of the two people can come (`A=1` means `A` can come; `A=0` means `A` can't come). Then use XOR to simplify your equation.

2.65 Simplify the following equation by using XOR wherever possible: `F = a'b + ab' + cd' + c'd + ac`.

2.66 Use 2-input XOR gates to create a circuit that outputs a `1` when the number of `1`s on inputs `a`, `b`, `c`, `d` is odd.

2.67 Use 2-input XOR or XNOR gates to create a circuit that detects whether an even number of the inputs `a`, `b`, `c`, `d` are `1`s.

SECTION 2.9: DECODERS AND MUXES

2.68 Design a 3x8 decoder using AND, OR, and NOT gates.

2.69 Design a 4x16 decoder using AND, OR, and NOT gates.

2.70 Design a 3x8 decoder with enable using AND, OR, and NOT gates.

2.71 Design an 8x1 multiplexer using AND, OR, and NOT gates.

2.72 Design a 16x1 multiplexer using AND, OR, and NOT gates.

2.73 Design a 4-bit 4x1 multiplexer using four 4x1 multiplexers.

2.74 A house has four external doors, each with a sensor that outputs `1` if its door is open. Inside the house is a single LED that a homeowner wishes to use to indicate whether a door is open or closed. Because the LED can only show the status of one sensor, the homeowner buys a switch that can be set to 0, 1, 2, or 3 and that has a 2-bit output representing the switch position in binary. Create a circuit to connect the four sensors, the switch, and the LED. Use at least one mux (a single mux or an *N*-bit mux) or decoder. Use block symbols, each with a clearly defined function, such as "2x1 mux," "8-bit 2x1 mux," or "3x8 decoder"; do not show the internal design of a mux or decoder.

2.75 A video system can accept video from one of two video sources, but can only display one source at a given time. Each source outputs a stream of digitized video on its own 8-bit output. A switch with a single-bit output chooses which of the two 8-bit streams will be passed on a display's single 8-bit input. Create a circuit to connect the two video sources, the switch, and the display. Use at least one mux (a single mux or an *N*-bit mux) or decoder. Use block symbols, each with a clearly defined function, such as "2x1 mux," "8-bit 2x1 mux," or "3x8 decoder"; do not show the internal design of a mux or decoder.

2.76 A store owner wishes to be able to indicate to customers that the items in one of the store's eight aisles are temporarily discounted ("on sale"). The store owner thus mounts a light above each aisle, and each light has a single-bit input that turns on the light when `1`. The store owner has a switch that can be set to 0, 1, 2, 3, 4, 5, 6, or 7, and that has a 3-bit output representing the switch position in binary. A second switch can be set up or down and has a single-bit output that is `1` when the switch is up; the store owner can set this switch down if no aisles are currently discounted. Use at least one mux (a single mux or an *N*-bit mux) or decoder. Use block symbols, each with a clearly defined function, such as "2x1 mux," "8-bit 2x1 mux," or "3x8 decoder"; do not show the internal design of a mux or decoder.

SECTION 2.10: ADDITIONAL CONSIDERATIONS

2.77 Determine the critical path of the following specified circuits. Assume that each AND and OR gate has a delay of 1 ns, each NOT gate has a delay of 0.75 ns, and each wire has a delay of 0.5 ns.

(a) The circuit of Figure 2.37.

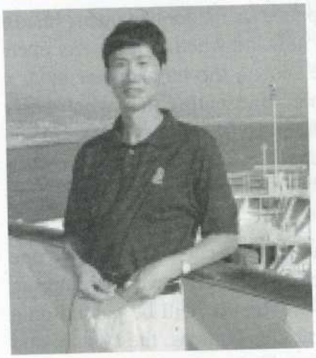
(b) The circuit of Figure 2.41.

2.78 Design a 1x4 demultiplexer using AND, OR, and NOT gates.

2.79 Design an 8x3 encoder using AND, OR, and NOT gates. Assume that only one input will be `1` at any given time.

2.80 Design a 4x2 priority encoder using AND, OR, and NOT gates. If every input is `0`, the encoder output should be `00`.

▶ DESIGNER PROFILE



Samson enjoyed physics and math in college, and focused his advanced studies on integrated circuit (IC) design, believing the industry to have a great future. Years later, he realizes his belief was true: "Looking back 20 years in high tech, we have experienced four major revolutions: the PC revolution, digital revolution, communication revolution, and Internet revolution—all four enabled by the IC industry. The impact of these revolutions to our daily life is profound."

He has found his job to be "very challenging, interesting, and exciting. I continually learn new skills to keep up, and to do my job more efficiently."

One of Samson's key design projects was for digital television, namely, high-definition TV (HDTV), involving companies like Zenith, Philips, and Intel. In particular, he led the 12-person design team that built Intel's first liquid crystal on silicon (LCoS) chip for rear-projection HDTV. "Traditional LCoS chips are analog. They apply different analog voltages on each pixel of the display chip so it can produce an image. But analog LCoS is very sensitive to noise and temperature variation. We used digital signals to do pulse width modulation on each pixel." Samson is quite proud of his team's accomplishments: "Our HDTV picture quality was much better."

Samson also worked on the 200-member design team for Intel's Pentium II processor. That was a very different experience. "For the smaller team project, each person had more responsibility, and overall efficiency was high. For the large team project, each person worked on a specific part of the project—the chip was divided into clusters, each cluster into units, and each unit had a leader. We relied heavily on design flows and methodologies."

Samson has seen the industry's peaks and valleys during the past two decades: "Like any industry, the IC job market has its ups and downs." He believes the industry survives the low points in large part due to innovation. "Brand names sell products, but without innovation, markets go elsewhere. So we have to be very innovative, creating new products so that we are always ahead in the global competition."

But "innovation doesn't grow on trees," Samson points out. "There are two kinds of innovations. The first is invention, which requires a good understanding of the physics behind technology. For example, to make an analog TV into a digital TV, we must know how human eyes perceive video images, which parts can be digitized, how digital images can be produced on a silicon chip, etc. The second kind of innovation reuses existing technology for a new application. For example, we can reuse advanced space technologies in a new non-space product serving a bigger market. e-Bay is another example—it reused Internet technology for online auctions. Innovations lead to new products, and thus new jobs for many years."

Thus, Samson points out that "The industry is counting on new engineers from college to be innovative, so they can continue to drive the high-tech industry forward. When you graduate from college, it's up to *you* to make things better."