Logic gate implementation and circuit minimization

Lila Kari

The University of Western Ontario
Why binary?

- Why are we bothering to use binary numbers?
- There is no theoretical reason why we could not build a “decimal computer”.
- Computers store information using electronic devices and the internal representation of a number must be implemented in terms of electronic quantities such as currents or voltage levels.
- Building a decimal (base-10) computer requires finding a device with 10 distinct and stable energy states that can be used to represent the 10 unique digits of the decimal system.
A decimal computer

For example, assume that there exists a device that can store electrical charges in the range of 0 to +45 volts. We could use it to build a decimal computer by letting certain voltage levels correspond to specific decimal digits.

<table>
<thead>
<tr>
<th>Voltage level</th>
<th>Corresponds to this decimal digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>+5</td>
<td>1</td>
</tr>
<tr>
<td>+10</td>
<td>2</td>
</tr>
<tr>
<td>+15</td>
<td>3</td>
</tr>
<tr>
<td>+20</td>
<td>4</td>
</tr>
<tr>
<td>+25</td>
<td>5</td>
</tr>
<tr>
<td>+30</td>
<td>6</td>
</tr>
<tr>
<td>+35</td>
<td>7</td>
</tr>
<tr>
<td>+40</td>
<td>8</td>
</tr>
<tr>
<td>+45</td>
<td>9</td>
</tr>
</tbody>
</table>
The reliability of binary representation

- While the decimal system is theoretically feasible, it is certainly not recommended.
- As electrical devices age, they become unreliable, and they may slowly drift or change their energy state over time.
- For example, the device representing the value 8 (40 volts) might lose 6 percent of its voltage. The voltage would drop to +37.5 volts.

Question: Does +37.5 represent 8 or 7?
Answer: Impossible to say.
Another 6 percent drop would cause a drop to +35, so 8 becomes a 7.
The reliability of binary representation

- While the decimal system is theoretically feasible, it is certainly not recommended.
- As electrical devices age, they become unreliable, and they may slowly drift or change their energy state over time.
- For example, the device representing the value 8 (40 volts) might lose 6 percent of its voltage. The voltage would drop to +37.5 volts.
- **Question:** Does +37.5 represent 8 or 7?
The reliability of binary representation

- While the decimal system is theoretically feasible, it is certainly not recommended.
- As electrical devices age, they become unreliable, and they may slowly drift or change their energy state over time.
- For example, the device representing the value 8 (40 volts) might lose 6 percent of its voltage. The voltage would drop to +37.5 volts.
- **Question:** Does +37.5 represent 8 or 7?
- **Answer:** Impossible to say.
While the decimal system is theoretically feasible, it is certainly not recommended.

As electrical devices age, they become unreliable, and they may slowly drift or change their energy state over time.

For example, the device representing the value 8 (40 volts) might lose 6 percent of its voltage. The voltage would drop to +37.5 volts.

Question: Does +37.5 represent 8 or 7?

Answer: Impossible to say.

Another 6 percent drop would cause a drop to +35, so 8 becomes a 7.
Decimal or binary computer?

- Building a reliable decimal machine, while theoretically feasible, can be an engineering nightmare.
- The exclusive use of binary is not for theoretical reasons, but for reliability.
The reliability of decimal representation

- Electrical systems tend to operate best in what is called a bistable environment in which there are only 2 (rather than 10) stable states, separated by a huge energy barrier.

- Back to our hypothetical electronic device that stored voltages in the range 0 to +45 volts, we could use the scheme:

  0 volts = 0 (full off)
  +45 volts = 1 (full on)

- In this case, 6 or even 12 percent drift cause no problem interpreting the value. In fact it takes almost a 50 percent change in voltage to create a problem.
Reliability of binary representation

- The use of binary for the internal representation of data significantly increases the inherent reliability of a computer.
- This single advantage makes it worth all the time needed to convert from decimal to binary for internal storage and from binary to decimal for external display of results.
Binary storage devices

Binary computers can be built out of any bistable device. Formally, it is possible to construct a binary computer using any device that meets the following four conditions:

- It has **two stable energy states** (one for a 0, one for a 1).
- These two states are **separated by a large energy barrier** (so a 0 does not accidentally become a 1, or the reverse).
- It is **possible to sense** what state the device is in (to see if it is storing a 0 or a 1) without permanently destroying the stored value.
- It is **possible to switch** from a 0 to a 1 and vice versa by applying a sufficient amount of energy.
Bistable devices

- There are many devices that meet these conditions, including some unexpected ones such as the familiar on-off light switch!
- Thus it would be possible to build a reliable (albeit very slow and bulky) binary computer out of ordinary light switches!
The transistor

- Today, the elementary building block for all modern computer systems is the transistor.
- A **transistor** is simply a **switch**, much like the light switch mentioned earlier.
- A transistor can be in an **off** state, which does not allow electricity to flow, or in an **on** state, in which electricity can pass unimpeded.
- However, unlike the light switch, a transistor is a solid-state device that has no mechanical or moving parts.
- The switching of a transistor from the OFF state to the ON state, or vice versa, is done electronically rather than mechanically. This allows it to be fast as well as extremely small.
The transistor

- Transistors are constructed from special materials called **semiconductors** such as silicon or gallium arsenide.
- A large number of transistors as well as the electrical conducting paths connecting them, can be printed photographically on a wafer of silicon to produce a device known as **integrated circuit**, or more commonly, a **chip**.
- The chip is sometimes mounted inside a ceramic housing called a **dual in-line package (DIP)** which contains the input-output connectors called **pins** that allow one chip to communicate and exchange information with other chips.
- The DIP itself is mounted on a **circuit board** that interconnects the different chips such as memory, processor, communications.
The use of photographic image rather than mechanical production techniques has numerous advantages. Since light can be focused very sharply, these integrated circuits can be manufactured in very high densities (number of transistors per square centimeter) and with a very high degree of accuracy (VLSI: "very large scale integration").

Another advantage of photographic production techniques is that it is possible to make a standard template, called a mask that describes the circuit. This mask can be used to produce a virtually unlimited number of copies of that chip, much like the idea of using a photographic negative to produce an unlimited number of copies.
The theoretical concepts underlying the physical behaviour of semiconductors and transistors are beyond the scope of this course. They are usually discussed in courses in physics or electrical engineering.

Instead we visualize a transistor using a highly simplified model, and then use this model to explain its behaviour.
The transistor: A simplified model
In this model, each transistor contains three lines: two input lines and one output line.

The first input line, called the control line, also referred to as the base, is used to open or close the switch inside the transistor.
How the transistor works

Power supply (+5 V)

1 (switch closed)

Measuring device

ON state
Binary 1

Power supply (+5 V)

0 (switch open)

Measuring device

OFF state
Binary 0
How the transistor works

- If we set the input value on the control line to a 1 by applying a sufficient amount of voltage, the switch closes and the transistor enters the ON state. In this state, voltage coming from the **in** line, called the **collector**, goes directly to the **out** line, called the **emitter**, and this voltage can be detected by a measuring device.
- This ON state can be used to represent the binary 1.
- If, instead, we set the input value of the control line to a 0 by not applying voltage, the switch opens and the transistor enters the OFF state.
- In this state, no voltage can get through the transistor, and none would be detected on the output line.
- The OFF state could be used to represent the value 0.
Transistors now, biocomputers tomorrow?

- This solid-state switching device, the transistor, forms the basis of construction of virtually all computers built today, and it is thus the fundamental building block for all high-level computers.
- However, there is no theoretical reason why we must use transistors as our elementary particles when designing computer systems.
- Just as magnetic cores were replaced by transistors, it is possible that transistors will ultimately be replaced by some newer technology (optical, biological, quantum) that is faster, smaller, and cheaper.
Logic gates

- Logic gates (inverter, OR, AND) are not abstract entities that exist only in textbooks and classroom discussions.
- On the contrary, logic gates are actual electronic devices that are the building blocks used in the design and construction of modern computer systems.
- The reason for using gates rather than transistors in circuit design is that a transistor is too elementary a device to act as the fundamental design component.
- Dealing with transistors requires a designer to deal with such low level issues as currents, voltages, and the laws of physics.
From transistors to logic gates

- Instead, transistors can be grouped together to form more powerful building blocks (logic gates), which allow us to think and design at a higher level.
- Instead of dealing with the complex physical rules associated with discrete electrical devices, we can use the power and expressiveness of mathematics and logic to build computers.
- This is another example of the use of abstraction in computer science.
Logic gate implementation: The NOT gate (inverter)

- A NOT gate requires a single transistor.
How the inverter works

- The collector is connected to the power supply (logical 1)
- The emitter is connected to the ground (logical 0)
- If the input is set to 1, the transistor is in the ON state and will pass the current through the ground. In this case the output voltage of the gate is 0.
- If the input is set to 0, the transistor is in the OFF state, and it blocks passage of the current to the ground. Instead, the current is transmitted to the output line, producing an output of 1.
The opposite of AND

To construct the opposite of an AND gate, we must use two **two transistors** connected in **series**.

![Diagram of an AND gate with two transistors connected in series]
The NAND gate

- The collector line of transistor 1 is connected to the power supply (logical 1).
- The emitter line of transistor 2 is connected to ground (logical 0).
- If both input lines, Input-1 and Input-2, are set to 1, then both transistors are in the ON state, and the output will be connected to the ground, resulting in a value of 0 on the output line.
- If either (or both) Input-1 or Input-2 is 0, the corresponding transistor is in the OFF state and does not allow the current to pass, resulting in a 1 on the output line.
- Thus the output of this gate is the opposite of AND and represented a gate called NAND gate.
To construct an AND gate, we must use a NAND gate and an inverter.
The opposite of OR

- To construct the opposite of an OR gate we must use two transistors connected in parallel.
The NOR gate

- If either or both of the lines Input-1 and Input-2 are set to 1, then the corresponding transistor is in the ON state, and the output is connected to the ground, producing an output value of 0.
- Only if both input lines are 0, effectively shutting off both transistors, will the output line contain a 1.
- Again, this is the exact opposite of the definition of OR, and this gate is called NOR gate.
The OR gate

- To construct an **OR gate** we must use a NOR gate and an inverter.
Thus, when we draw the logical gate symbols, we are diagramming actual electrical components composed internally of one or more transistors.

This seemingly minor change in viewpoint from transistors to logic gates has a profound effect on how computer hardware is designed and built.

From this point on, in a discussion of hardware design one no longer needs to deal with anything “electrical”. One no longer requires a knowledge of transistors or resistors. One no longer needs to be a physicist or an electrical engineer.

Instead, our building blocks will be the three gates just described, AND, OR and NOT, and our circuit construction rules will be the rules of Boolean logic.
Minimization of circuits

- The efficiency and cost of a combinatorial circuit depends on the number and arrangement of its gates.
- The process of designing a combinatorial circuit begins with the table specifying the output for each combination of input values.
- We can always use sum-of-product expansion of a circuit to find a set of logic gates that will implement this circuit.
- However, sum-of-products expansions may contain many more terms than are necessary.
- Terms in a sum-of-products expansions that differ in just one variable, so that in one term this variable occurs and in the other term the complement of the variable occurs, can be combined.
• For instance, consider the circuit that has output 1 iff
  \( x = y = z = 1 \) or \( x = z = 1 \) and \( y = 0 \).
• For instance, consider the circuit that has output 1 iff $x = y = z = 1$ or $x = z = 1$ and $y = 0$.
• The sum-of-products expansion is $xyz + x\overline{y}z$.
• For instance, consider the circuit that has output 1 iff 
  \( x = y = z = 1 \) or \( x = z = 1 \) and \( y = 0 \).
• The sum-of-products expansion is \( xyz + x\bar{y}z \).
• The two products in this expansion differ in exactly one 
  variable, namely \( y \).
• For instance, consider the circuit that has output 1 iff \( x = y = z = 1 \) or \( x = z = 1 \) and \( y = 0 \).

• The sum-of-products expansion is \( xyz + x\bar{y}z \).

• The two products in this expansion differ in exactly one variable, namely \( y \).

• They can be combined as follows:

\[
xyz + x\bar{y}z = (y + \bar{y})(xz) = 1 \cdot (xz) = xz.
\]

Hence, \( xz \) is a Boolean expression with fewer operators that represents the circuit.
• For instance, consider the circuit that has output 1 iff 
  \( x = y = z = 1 \) or \( x = z = 1 \) and \( y = 0 \).

• The sum-of-products expansion is \( xyz + x\bar{y} z \).

• The two products in this expansion differ in exactly one variable, namely \( y \).

• They can be combined as follows:

  \[
  xyz + x\bar{y} z = (y + \bar{y})(xz) = 1 \cdot (xz) = xz.
  \]

• Hence, \( xz \) is a Boolean expression with fewer operators that represents the circuit.
Two circuits with the same output

\begin{align*}
x & \rightarrow \quad \text{AND gate} \\
y & \rightarrow \quad \text{AND gate} \\
z & \rightarrow \quad \text{AND gate} \\
\end{align*}

\begin{align*}
x & \rightarrow \quad \text{AND gate} \\
y & \rightarrow \quad \text{AND gate} \\
z & \rightarrow \quad \text{AND gate} \\
\end{align*}

\begin{align*}
\text{AND gate} & \rightarrow \quad \text{OR gate} \\
\text{AND gate} & \rightarrow \quad \text{OR gate} \\
\text{AND gate} & \rightarrow \quad \text{OR gate} \\
\end{align*}

\begin{align*}
\text{AND gate} & \rightarrow \quad \text{OR gate} \\
\text{AND gate} & \rightarrow \quad \text{OR gate} \\
\text{AND gate} & \rightarrow \quad \text{OR gate} \\
\end{align*}
Karnaugh maps

- To reduce the number of terms in a Boolean expression representing a circuit, it is necessary to find terms to combine.
- There is a graphical method, called Karnaugh map, for finding terms to combine for Boolean functions involving a relatively small number of variables. This method was introduced by Maurice Karnaugh in 1953.

Maurice Karnaugh, American physicist, (1924-)
Karnaugh maps in two variables

- There are 4 possible minterms in the sum-of-products expansion of a Boolean function in 2 variables, \( x \) and \( y \).
- A Karnaugh map for a Boolean function in these 2 variables consists of four squares where a 1 is placed in the square representing a minterm if this minterm is present in the expansion.
- Squares are said to be adjacent if they differ in exactly one literal. For instance, the square representing \( \overline{x} \ y \) is adjacent to the squares representing \( x \ y \) and \( \overline{x} \ \overline{y} \).
Examples

Find the Karnaugh maps for

(a) \( xy + \overline{x} \ y \)

(b) \( xy \overline{y} + \overline{x} \ y \)

(c) \( xy \overline{y} + \overline{x} \ y + \overline{x} \overline{y} \).
K-maps for the previous example

(a) 

(b) 

(c)
How to identify terms that can be combined

- Whenever there are 1s in two adjacent squares in the Karnaugh map, the minterms represented by these squares can be combined into a product involving just one of the variables.
- For instance, $x\bar{y}$ and $\bar{x} \bar{y}$ are represented by adjacent squares and can be combined into $\bar{y}$ since $x\bar{y} + \bar{x} \bar{y} = (x + \bar{x})\bar{y} = \bar{y}$.
- If 1s are in all four squares, the four minterms can be combined into one term, namely the Boolean expression 1.
- We mark blocks of squares in the Karnaugh map that represent minterms that can be combined and then find the corresponding sum of products.
How to identify terms that can be combined

The goal is

- To identify the largest possible blocks, and
- To cover all the 1s with the fewest blocks,
- Using the largest blocks first
- Always using the largest possible blocks.
Example

Simplify the sum-of-products expansions from the preceding example:
Simplify the sum-of-products expansions from the preceding example:

\[(a) \quad xy + xy = y\]

\[(b) \quad x y + xy = xy + xy\]

\[(c) \quad x y + xy + xy = x + y\]
Example

Simplify the sum-of-products expansions from the preceding example:

\[(a)\quad xy + \overline{x} \, y = y\]
Example

Simplify the sum-of-products expansions from the preceding example:

\[(a) \ xy + \bar{x} y = y\]
\[(b) \ x\bar{y} + \bar{x} \ y = x\bar{y} + \bar{x} y\]
Example

Simplify the sum-of-products expansions from the preceding example:

(a) \(xy + \overline{x} y = y\)
(b) \(x\overline{y} + \overline{x} y = x\overline{y} + \overline{x} y\)
(c) \(x\overline{y} + \overline{x} y + \overline{x} \overline{y} = \overline{x} + \overline{y}\).
Karnaugh maps in three variables

- A Karnaugh map of three variables is a rectangle divided into eight squares.
- The squares represent the eight possible minterms in three variables.
- Two squares are said to be adjacent if the minterms they represent differ exactly in one literal.
- We consider the lines and columns wrapped around.
Karnaugh map in 3 variables

This Karnaugh map can be thought of as a cylinder. One the cylinder, two squares have a common border iff they are adjacent.
How to identify terms to be combined

• Blocks of **two adjacent squares** represent pairs of minterms that can be combined into a product of two literals.

• **$2 \times 2$ and $4 \times 1$ blocks** of squares represent minterms that can be combined into a single literal.

• The block of all **eight squares** represents a product of no literals, namely the function 1.
• The goal is to identify the largest possible blocks in the map and
• Cover all the 1s in the map
• Using the least number of blocks, and
• Using the largest blocks first.
• The largest possible blocks are always chosen.
• Note that there may be more than one way to do this.
Karnaugh maps in 3 variables

\( y\bar{z} = \bar{x}y\bar{z} + xy\bar{z} \)

\( \bar{x}z = \bar{x}yz + \bar{x}y\bar{z} \)

\( z = xy\bar{z} + xy\bar{z} + \bar{x}y\bar{z} + xy\bar{z} \)

\( x = xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z} + xy\bar{z} \)

\( 1 = xy\bar{z} + xy\bar{z} + xy\bar{z} + \bar{x}y\bar{z} + \bar{x}y\bar{z} + xy\bar{z} + \bar{x}y\bar{z} \)
Karnaugh maps in four variables

<table>
<thead>
<tr>
<th></th>
<th>$yz$</th>
<th>$y\bar{z}$</th>
<th>$\bar{y}\bar{z}$</th>
<th>$\bar{y}z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$wx$</td>
<td>$wxyz$</td>
<td>$wxy\bar{z}$</td>
<td>$wx\bar{y}z$</td>
<td>$wx\bar{y}z$</td>
</tr>
<tr>
<td>$\bar{w}x$</td>
<td>$\bar{w}xyz$</td>
<td>$\bar{w}xy\bar{z}$</td>
<td>$\bar{w}x\bar{y}z$</td>
<td>$\bar{w}x\bar{y}z$</td>
</tr>
<tr>
<td>$\bar{w}x$</td>
<td>$\bar{w}xyz$</td>
<td>$\bar{w}xy\bar{z}$</td>
<td>$\bar{w}x\bar{y}z$</td>
<td>$\bar{w}x\bar{y}z$</td>
</tr>
<tr>
<td>$\bar{w}x$</td>
<td>$\bar{w}xyz$</td>
<td>$\bar{w}xy\bar{z}$</td>
<td>$\bar{w}x\bar{y}z$</td>
<td>$\bar{w}x\bar{y}z$</td>
</tr>
</tbody>
</table>
Karnaugh maps in four variables

(a) \( w\overline{x}z = w\overline{x}yz + w\overline{x}\overline{y}z \)

(b) \( \overline{w}\overline{x} = \overline{w}\overline{x}yz + \overline{w}\overline{x}\overline{y}\overline{z} + \overline{w}\overline{x}\overline{y}z + \overline{w}\overline{x}yz \)
Karnaugh maps in four variables

\[ \bar{z} = wxyz + wx\bar{y}z + \bar{w}xyz + \bar{w}x\bar{y}z \]

(c)

\[ \bar{z} = wxyz + wx\bar{y}z + \bar{w}xyz + \bar{w}x\bar{y}z + w\bar{xy}\bar{z} + \bar{w}\bar{xy}\bar{z} + \bar{w}x\bar{y}\bar{z} + w\bar{x}\bar{y}\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}x\bar{y}\bar{z} \]

(d)
The figures in this set of notes are from