Chapter 4

Gates and Circuits (with some transistors thrown in for good measure)
Abstractions and more abstractions ...

**Computers**
Made of lots of different circuits (CPU, memory, controllers, etc.)

**Circuits**
Made from gates combined to perform more complicated tasks

**Gates**
Devices that perform basic logical operations on electrical signals. They’re built out of transistors

**Transistors**
Very small electronic switches
How do we describe the behavior of gates and circuits?

1. **Boolean expressions**
   Uses Boolean algebra, a mathematical notation for expressing two-valued logic

2. **Logic diagrams**
   A graphical representation of a circuit; each gate has its own symbol

3. **Truth tables**
   A table showing all possible input value and the associated output values
4.2 Gates

There are six basic gates:

- NOT
- AND
- OR
- XOR
- NAND
- NOR

Typically, logic diagrams are black and white with gates distinguished only by their shape.

We use color for emphasis (and fun)
NOT Gate (a.k.a. inverter)

A NOT gate accepts one input signal (0 or 1) and returns the opposite signal as output.

**Boolean Expression**  |  **Logic Diagram Symbol**  |  **Truth Table**
---|---|---
$X = A'$  | ![Logic Diagram Symbol]  | 

<table>
<thead>
<tr>
<th>$A$</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure 4.1 Various representations of a NOT gate*
AND Gate

An AND gate accepts two input signals
If both are 1, the output is 1; otherwise, the output is 0

Figure 4.2 Various representations of an AND gate
OR Gate

An OR gate accepts two input signals
If both are 0, the output is 0; otherwise, the output is 1

Figure 4.3 Various representations of a OR gate
XOR Gate

If both inputs are the same, the output is 0; otherwise, the output is 1.

XOR is called the exclusive OR
Pronunciations: zor, ex-or
QUIZ: recognize the gate!

\[ X = A \cdot B \]

\[ \begin{array}{ccc}
A & B & X \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array} \]

\[ X = A + B \]

\[ \begin{array}{ccc}
A & B & X \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array} \]

\[ X = A' \]

\[ \begin{array}{ccc}
A & B & X \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array} \]
QUIZ: draw the gate symbols!

\[ X = A \cdot B \]

\[
\begin{array}{c|c|c}
A & B & X \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

\[ X = A + B \]

\[
\begin{array}{c|c|c}
A & B & X \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[ X = A' \]
**QUIZ: elementary properties**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A AND 0 = ?
A AND 1 = ?
Etc.
NAND Gate

If both inputs are 1, the output is 0; otherwise, the output is 1

Figure 4.5 Various representations of a NAND gate

<table>
<thead>
<tr>
<th>Boolean Expression</th>
<th>Logic Diagram Symbol</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X = (A \cdot B)’ )</td>
<td>![NAND Gate Diagram]</td>
<td>![NAND Truth Table]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
NOR Gate

The NOR gate accepts two input signals. If both are 0, the output is 1; otherwise, the output is 0.

![Diagram of a NOR gate with truth table]

**Figure 4.6** Various representations of a NOR gate
Review of Gates

A NOT gate **inverts** its single input
An AND gate produces **1** if both input values are **1**
An OR gate produces **0** if both input values are **0**
An XOR gate produces **0** if input values are the **same**
A NAND gate produces **0** if both inputs are **1**
A NOR gate produces a **1** if both inputs are **0**
Quiz

• What are the 3 ways we use to describe gates and circuits?
• Use the 3 ways to describe the NAND gate
  – Hint: Describe AND first!
Solve in notebook for next time:

End of chapter:
  – 1 through 10
  – 18 through 29
Gates with More Inputs

Gates can be designed to accept three or more input values. A three-input AND gate, for example, produces an output of 1 only if all input values are 1.

![Diagram of a three-input AND gate]

**Figure 4.7** Various representations of a three-input AND gate
QUIZ

Draw the gate symbols for:

- 4-input OR
- 5-input NAND
- 3-input NOR
- 4-input XOR

*Figure 4.7 Various representations of a three-input AND gate*
QUIZ

Draw the gate symbols for:
• 4-input OR
• 5-input NAND
• 3-input NOR
• 4-input XOR

How many lines does each of the truth tables have?
QUIZ

Draw the gate symbols for:
- 4-input OR
- 5-input NAND
- 3-input NOR
- 4-input XOR

How many lines does each of the truth tables have?

Describe in your own words each of the truth tables.

Figure 4.7 Various representations of a three-input AND gate
A computer represents numbers in 8-bit two’s complement. Design a circuit that will detect the number zero (the output of the circuit becomes 1 if and only if all 8 bits are 0):

```
0 0 0 0 0 0 0 0
```
Extra-credit QUIZ

A computer represents numbers in 8-bit two’s complement. Design a circuit that will detect the number -128.

Hint: -128 is 1000 0000 in two’s comp.

1 0 0 0 0 0 0 0 0
From gates to circuits

Find the logic diagram of the circuit described by the following truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Hint: The table is similar to which of the fundamental gates presented last time?
Having only one 0 in the output column, the circuit most resembles the **OR** gate!

It is **different** from the OR gate only in this respect: ...

Write the Boolean expression:

Draw the diagram:
For next time:

Read the entire Section 4.2 Gates.
Understand the examples given in the text and quizzes.

Comments and solutions for Ch.2 homework
Remember: There are 3 layers of computer abstraction that we examine in this chapter:

<table>
<thead>
<tr>
<th>Circuits</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Gates</td>
<td>√</td>
</tr>
<tr>
<td>Transistors</td>
<td></td>
</tr>
</tbody>
</table>
4.3 Constructing Gates

Transistor = device that acts either as a wire that conducts electricity or as a resistor that blocks the flow of electricity, depending on the voltage level of an input signal.

Acts like a switch, but w/o moving parts:
- Switch open
- Switch closed

Made of a semiconductor material
- Neither good conductor of electricity nor a good insulator
A transistor has three terminals:

- A collector/source, typically connected to the positive terminal of a power source (5 volts, 3.5 volts, etc.)
- An emitter/drain, typically connected to the “ground” (0 volts)
- A base/gate, which controls the flow of current between source and emitter
The names of transistor terminals
-setting the record straight FYI-

Bipolar Junction Transistor (BJT)
Collector, Base, Emitter

Field-Effect Transistor (FET)
Drain, Gate, Source
How transistors operate as switches

When 1 is applied on the base/gate, the switch closes

When 0 is applied on the base/gate, the switch opens
The easiest gates to create are the **NOT**, **NAND**, and **NOR**

<table>
<thead>
<tr>
<th>NOT gate</th>
<th>NAND gate</th>
<th>NOR gate</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="NOT gate diagram" /></td>
<td><img src="image2" alt="NAND gate diagram" /></td>
<td><img src="image3" alt="NOR gate diagram" /></td>
</tr>
</tbody>
</table>

We can explain their operation for any combination of inputs! We do this by replacing the transistors with switches!
If there is no path from output to Ground, $V_{out} = 1$
If there is a path from output to Ground, $V_{out} = 0$
QUIZ

The AND gate is obtained as a NAND followed by an inverter. Draw its transistor diagram!
**QUIZ**

Draw its switch diagram.

Show the states of all switches for \( V1 = 0 \) and \( V2 = 1 \).

<table>
<thead>
<tr>
<th>NOT gate</th>
<th>NAND gate</th>
<th>NOR gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source ( v_{in} )</td>
<td>Source ( v_1 ) ( v_2 )</td>
<td>Source ( v_1 ) ( v_2 )</td>
</tr>
<tr>
<td>Emitter ( v_{out} )</td>
<td>( v_{out} )</td>
<td>Emitter ( v_{out} )</td>
</tr>
<tr>
<td>Ground</td>
<td>Ground</td>
<td>Ground</td>
</tr>
</tbody>
</table>
4.4 Circuits

Combinational circuit
The input values explicitly determine the output

Sequential circuit
The output is a function of the input values and the existing state of the circuit

We describe the circuit operations using
  Boolean expressions
  Logic diagrams
  Truth tables
Combinational Circuits

Gates are combined into circuits by using the output of one gate as the input for another.

\[ X = AB + AC \]
Combinational Circuits

Three inputs require $2^3 = 8$ rows to describe all possible input combinations.

Boolean expression is: $X = AB + AC$
Another Combinational Circuit

Diagram:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>B + C</th>
<th>A(B + C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Another Combinational Circuit

Consider the following Boolean expression $A(B + C)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>B + C</th>
<th>A(B + C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Does this truth table look familiar?
Circuit equivalence

Two circuits that produce the same output for identical input

Boolean algebra allows us to apply provable mathematical principles to help design circuits

\[ A(B + C) = AB + BC \] (distributive law) so circuits must be equivalent
# Properties (laws) of Boolean Algebra

<table>
<thead>
<tr>
<th>Property</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commutative</td>
<td>$AB = BA$</td>
<td>$A + B = B + A$</td>
</tr>
<tr>
<td>Associative</td>
<td>$(AB)C = A(BC)$</td>
<td>$(A + B) + C = A + (B + C)$</td>
</tr>
<tr>
<td>Distributive</td>
<td>$A(B + C) = (AB) + (AC)$</td>
<td>$A + (BC) = (A + B)(A + C)$</td>
</tr>
<tr>
<td>Identity</td>
<td>$A1 = A$</td>
<td>$A + 0 = A$</td>
</tr>
<tr>
<td>Complement</td>
<td>$A(A') = 0$</td>
<td>$A + (A') = 1$</td>
</tr>
<tr>
<td>DeMorgan's law</td>
<td>$(AB)' = A' OR B'$</td>
<td>$(A + B)' = A'B'$</td>
</tr>
</tbody>
</table>

- **Null elements**: $A \cdot 0 = 0$ \hspace{1cm} $A + 1 = 1$
- **Idempotency**: $A \cdot A = A$ \hspace{1cm} $A + A = A$
- **Double complement**: $(A')' = A$
DeMorgan’s law applied directly to gates

(a) \( \overline{x_1 x_2} = \overline{x_1} + \overline{x_2} \)

(b) \( \overline{x_1 + x_2} = \overline{x_1} \overline{x_2} \)
Using double-complement to “set up” DeMorgan

Any two-level AND-OR circuit had a 2-level NAND-only equivalent!
DeMorgan’s law QUIZ

Apply DeMorgan’s Law directly on the gate diagrams below to obtain equivalent circuits:
Solve in notebook for next time:

End of chapter:
  – 11, 12
  – 46 through 50
  – 55
The XOR operation can be implemented with AND, OR and NOT gates:

\[ X = A \oplus B = A'B + AB' \]

How many transistors are required for the XOR gate?
Very Useful Combinational Circuit:
the Adder

At the digital logic level, addition is performed in binary

Addition operations are carried out by special circuits called adders
Half-Adder truth table

The result of adding two binary digits could produce a *carry value*

Recall that $1 + 1 = 10$ in base two

**Half adder**

A circuit that computes the sum of two bits and produces the correct carry bit

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Sum</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Do you recognize these outputs?
Half Adder

Circuit diagram

Boolean expressions

\[ \text{Sum} = A \oplus B \]
\[ \text{Carry} = A \cdot B \]

How many transistors are here?
Full Adder

This adder takes the *Carry-in* value into account!

Do you recognize these circuits?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Carry-in</th>
<th>Sum</th>
<th>Carry-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Adding multiple bits
- “Ripple Carry” Adder -
Very Useful Combinational Circuit: the Multiplexeer

MUX = A circuit that uses a few input control signals to determine which of several output data lines is routed to its output.

It is nothing but an electronic rotary switch!
Multiplexer symbol and truth table

The control lines S0, S1, and S2 determine which of eight other input lines (D0 ... D7) are routed to the output (F)

![Multiplexer symbol and truth table](image)

Figure 4.11 A block diagram of a multiplexer with three select control lines
“Lookup table” with MUX

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A(B + C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Connect the MUX input to implement a **prime number detector** (i.e. the output \( F \) is 1 iff \( S_2S_1S_0 \) are the binary code of a prime number)
4.5 Circuits as Memory
a.k.a. Sequential Circuits

A **sequential circuit** is one whose output depends not only on the **current** values of its inputs, but also on the past **sequence** of those inputs (history).

It can be used to **store** information, i.e. as **memory**.
Putting things in perspective: Delay-Line memories 😊

Mercury memory of UNIVAC I (1951)

Sources: Wikipedia, UNIVAC manual

How many Bytes of memory total? (see notes)
The S–R latch

There are several ways to build S–R latches using various kinds of gates, but there’s always feedback.

How many transistors are here?
If $X$ is 1, we say that the circuit is storing a 1; if $X$ is 0, the circuit is storing a 0.

As long as $S = R = 1$, an S-R latch stores a single binary digit, 1 or 0.

The design guarantees that the two outputs $X$ and $Y$ are (almost always) complements of each other.

The value of $X$ at any point in time is considered to be the state of the circuit.
S – R latch

Real-life check: since we make S = 0 or R = 0, we say that the signals S and R are **active low**.

They are normally denoted S’ and R’.

Accordingly, this type of S – R latch is called “non-S non-R”.

To make X = 1, make S = 0, while keeping R = 1.

To make X = 0, make R = 0, while keeping S = 1.
S – R latch “forbidden” inputs

What happens if both S and R are activated (made 0) at the same time?
Integrated Circuit (a.k.a. IC or chip) = A piece of silicon on which multiple gates have been embedded

Silicon pieces are mounted on a plastic or ceramic package with pins along the edges that can be soldered onto circuit boards or inserted into appropriate sockets
Integrated Circuits

Integrated circuits (IC) are classified by the number of gates contained in them.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Name</th>
<th>Number of Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSI</td>
<td>Small-Scale Integration</td>
<td>1 to 10</td>
</tr>
<tr>
<td>MSI</td>
<td>Medium-Scale Integration</td>
<td>10 to 100</td>
</tr>
<tr>
<td>LSI</td>
<td>Large-Scale Integration</td>
<td>100 to 100,000</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very-Large-Scale Integration</td>
<td>more than 100,000</td>
</tr>
</tbody>
</table>
Integrated Circuits

Figure 4.13 An SSI chip containing NAND gates

VLSI chip: AMD Phenom II CPU contains 768 million transistors

How many transistors are here?
CPU Chips

The most important integrated circuit in any computer is the Central Processing Unit, or CPU.

Each CPU chip has a large number of pins through which essentially all communication in a computer system occurs.
Ethical Issues

Email Privacy

*Explain why privacy is an illusion.*

*Who can read your email?*

*Do you send personal email from work?*

*Does everyone in your family use email?*
All the world knows my name. What is it and why do people know it?
Do you know?

What is the name of the study of materials smaller than 100 nanometers?

Did DeMorgan discover DeMorgan's laws?

Who did the 4th Infantry Division take to war with them?

What is a virtual charity event?
Chapter review questions

• Identify the basic gates and describe the behavior of each
• Describe how gates are implemented using transistors
• Combine basic gates into circuits
• Describe the behavior of a gate or circuit using Boolean expressions, truth tables, and logic diagrams
Chapter review questions

• Compare and contrast a half adder and a full adder
• Describe how a multiplexer works
• Explain how an S-R latch operates
• Describe the characteristics of the four generations of integrated circuits
Work in notebook: End-of-chapter 62, 63, 64
Read and take notes: Ethical issues, Email privacy, Trivia

**Homework**

Due Wednesday, Feb. 22, at beginning of lab, before midterm:

End-of-chapter 39, 40, 56, 59, 60, 65, 66, 67, 68
Review for Midterm – Ch.4
Use a MUXes as “lookup tables” to implement the 1-bit adder

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Carry-in</th>
<th>Sum</th>
<th>Carry-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram of MUXes connected as lookup tables for the 1-bit adder.
Show how a MUX with only 4 data inputs works

Assume $S1 = 0$, $S0 = 1$. Draw the equivalent circuit in this case and explain the value of the output $q$. 

![MUX diagram](image)
Extra-credit

How many transistors are needed to build this MUX?
Show how an S-R latch works

Unlike the one presented last class, this latch is made of **NOR** gates. Assume **R = 0, S = 1**. Show that:

- the latch is “set”, i.e. the output/state **X** is 1
- the other output has the correct value of 0 (complement of **X**)

![Diagram of S-R latch with NOR gates]
Show how an S-R latch works (continued)

To do in the notebook:
Examine the other 3 combinations of the inputs $R$, $S$ and explain if the latch operates correctly or not.
What is the “forbidden” combination of inputs for this version of the S-R latch?