AND OVER THERE WE HAVE THE LABYRINTH GUARDS.
ONE ALWAYS LIES, ONE ALWAYS TELLS THE TRUTH, AND
ONE STABS PEOPLE WHO ASK TRICKY QUESTIONS.
Last Class

- More combinational logic gates
  - NAND, NOR, XOR, XNOR
- Proofs of Logical Equivalence
  - e.g.
    \[(p \land q) \rightarrow (q \lor p)\] is a tautology

\[
(p \land q) \rightarrow (q \lor p) \equiv \neg(p \land q) \lor (q \lor p) \\
\equiv (\neg p \lor \neg q) \lor (q \lor p) \\
\equiv \neg p \lor (\neg q \lor (q \lor p)) \\
\equiv \neg p \lor ((\neg q \lor q) \lor p) \\
\equiv \neg p \lor (p \lor (\neg q \lor q)) \\
\equiv (\neg p \lor p) \lor (\neg q \lor q) \\
\equiv T \lor T \\
\equiv T
\]
**Boolean Logic: Circuits**

### Combinational Logic
- output = $F(\text{input})$

### Sequential Logic
- output$_t = F(\text{output}_{t-1}, \text{input}_t)$
  - output dependent on history
  - concept of a time step (clock, $t$)

### Boolean Algebra: Another notation for logic consisting of...
- a set of elements $B = \{0, 1\}$
- binary operations $\{ +, \cdot \}$ (OR, AND)
- and a unary operation $\{ ' \}$ (NOT)
Boolean Algebra

- **Usual notation used in circuit design**

- **Boolean algebra**
  - a set of elements \( B \) containing \( \{0, 1\} \)
  - binary operations \( \{+, \cdot\} \)
  - and a unary operation \( \{'\} \)
  - such that the following axioms hold:

For any \( a, b, c \) in \( B \):

1. **closure**:
   - \( a + b \) is in \( B \)
   - \( a \cdot b \) is in \( B \)

2. **commutativity**:
   - \( a + b = b + a \)
   - \( a \cdot b = b \cdot a \)

3. **associativity**:
   - \( (a + (b + c)) = (a + b) + c \)
   - \( (a \cdot (b \cdot c)) = (a \cdot b) \cdot c \)

4. **distributivity**:
   - \( a + (b \cdot c) = (a + b) \cdot (a + c) \)
   - \( a \cdot (b + c) = (a \cdot b) + (a \cdot c) \)

5. **identity**:
   - \( a + 0 = a \)
   - \( a \cdot 1 = a \)

6. **complementarity**:
   - \( a + a' = 1 \)
   - \( a \cdot a' = 0 \)

7. **null**:
   - \( a + 1 = 1 \)
   - \( a \cdot 0 = 0 \)

8. **idempotency**:
   - \( a + a = a \)
   - \( a \cdot a = a \)

9. **involution**:
   - \( (a')' = a \)
A Combinational Logic Example

Sessions of Class:
We would like to compute the number of lectures or quiz sections remaining at the start of a given day of the week.

– Inputs: Day of the Week, Lecture/Section flag
– Output: Number of sessions left

Examples: Input: (Wednesday, Lecture) Output: 2
Input: (Monday, Section) Output: 1
public int classesLeftInMorning(weekday, lecture_flag) {
    switch (weekday) {
        case SUNDAY:
        case MONDAY:
            return lecture_flag ? 3 : 1;
        case TUESDAY:
        case WEDNESDAY:
            return lecture_flag ? 2 : 1;
        case THURSDAY:
            return lecture_flag ? 1 : 1;
        case FRIDAY:
            return lecture_flag ? 1 : 0;
        case SATURDAY:
            return lecture_flag ? 0 : 0;
    }
}
Implementation with Combinational Logic

Encoding:
- How many bits for each input/output?
- Binary number for weekday
- One bit for each possible output (called “1-hot” encoding)
Defining Our Inputs!

Weekday Input:
- Binary number for weekday
- Sunday = 0, Monday = 1, ...
- We care about these in binary:

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Number</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>0</td>
<td>(000)₂</td>
</tr>
<tr>
<td>Monday</td>
<td>1</td>
<td>(001)₂</td>
</tr>
<tr>
<td>Tuesday</td>
<td>2</td>
<td>(010)₂</td>
</tr>
<tr>
<td>Wednesday</td>
<td>3</td>
<td>(011)₂</td>
</tr>
<tr>
<td>Thursday</td>
<td>4</td>
<td>(100)₂</td>
</tr>
<tr>
<td>Friday</td>
<td>5</td>
<td>(101)₂</td>
</tr>
<tr>
<td>Saturday</td>
<td>6</td>
<td>(110)₂</td>
</tr>
</tbody>
</table>
Converting to a Truth Table!

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Lecture?</th>
<th>c₀</th>
<th>c₁</th>
<th>c₂</th>
<th>c₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

case SUNDAY or MONDAY:
    return lecture_flag ? 3 : 1;
case TUESDAY or WEDNESDAY:
    return lecture_flag ? 2 : 1;
case THURSDAY:
    return lecture_flag ? 1 : 1;
case FRIDAY:
    return lecture_flag ? 1 : 0;
case SATURDAY:
    return lecture_flag ? 0 : 0;
Converting to a Truth Table!

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Lecture?</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

case SUNDAY or MONDAY:
    return lecture_flag ? 3 : 1;

case TUESDAY or WEDNESDAY:
    return lecture_flag ? 2 : 1;

case THURSDAY:
    return lecture_flag ? 1 : 1;

case FRIDAY:
    return lecture_flag ? 1 : 0;

case SATURDAY:
    return lecture_flag ? 0 : 0;
Let’s begin by finding an expression for $c_3$. To do this, we look at the rows where $c_3 = 1$ (true).

$$S_3 = \text{"DAY=SUN" \cdot L} + \text{"DAY=MON" \cdot L}$$
## Truth Table to Logic (Part 1)

<table>
<thead>
<tr>
<th>Day</th>
<th>$d_2d_1d_0$</th>
<th>$L$</th>
<th>$c_0$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Truths for Day:**
- **Day == SUN && L == 1**
- **Day == MON && L == 1**
### Truth Table to Logic (Part 1)

<table>
<thead>
<tr>
<th>(d_2d_1d_0)</th>
<th>L</th>
<th>(c_0)</th>
<th>(c_1)</th>
<th>(c_2)</th>
<th>(c_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Substituting DAY with the binary representation.

- \(d_2d_1d_0 = 000 \land L = 1\)
- \(d_2d_1d_0 = 001 \land L = 1\)
### Truth Table to Logic (Part 1)

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d₂d₁d₀</td>
<td>L</td>
<td>c₀</td>
<td>c₁</td>
<td>c₂</td>
<td>c₃</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Splitting up the bits of the day; so, we can write a formula.

\[ d_2 = 0 \text{ } \&\& \text{ } d_1 = 0 \text{ } \&\& \text{ } d_0 = 0 \text{ } \&\& \text{ } L = 1 \]

\[ d_2 = 0 \text{ } \&\& \text{ } d_1 = 0 \text{ } \&\& \text{ } d_0 = 1 \text{ } \&\& \text{ } L = 1 \]
### Truth Table to Logic (Part 1)

<table>
<thead>
<tr>
<th></th>
<th>(d_2d_1d_0)</th>
<th>L</th>
<th>(c_0)</th>
<th>(c_1)</th>
<th>(c_2)</th>
<th>(c_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Replacing with Boolean Algebra...

\[d_2 \cdot d_1 \cdot d_0 \cdot L\]
Truth Table to Logic (Part 1)

<table>
<thead>
<tr>
<th>d₂d₁d₀</th>
<th>L</th>
<th>c₀</th>
<th>c₁</th>
<th>c₂</th>
<th>c₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Either situation causes c₃ to be true. So, we “or” them.

\[ c₃ = d₂' \cdot d₁' \cdot d₀' \cdot L + d₂' \cdot d₁' \cdot d₀ \cdot L \]
Truth Table to Logic (Part 2)

<table>
<thead>
<tr>
<th></th>
<th>$d_2d_1d_0$</th>
<th>L</th>
<th>$c_0$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>$010$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>$011$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Now, we do $c_2$.

$\text{DAY} = \text{TUE if } L = 1$

\[
d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L
\]

$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$
Truth Table to Logic (Part 3)

<table>
<thead>
<tr>
<th></th>
<th>d_2d_1d_0</th>
<th>L</th>
<th>c_0</th>
<th>c_1</th>
<th>c_2</th>
<th>c_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Now, we do c_3:

\[ d_2 \cdot d_1 \cdot d_0 \cdot L + d_2 \cdot d_1 \cdot d_0 \cdot L \]

\[ c_3 = d_2 \cdot d_1 \cdot d_0 \cdot L + d_2 \cdot d_1 \cdot d_0 \cdot L \]
### Truth Table to Logic (Part 3)

<table>
<thead>
<tr>
<th></th>
<th>$d_2d_1d_0$</th>
<th>$L$</th>
<th>$c_0$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2 \cdot d_1' \cdot d_0$</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2 \cdot d_1' \cdot d_0' \cdot L$</td>
</tr>
<tr>
<td>FRI</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$d_2 \cdot d_1' \cdot d_0' \cdot L$</td>
</tr>
<tr>
<td>SAT</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0' \cdot L$</td>
</tr>
<tr>
<td>SAT</td>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$c_2 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0' \cdot L$</td>
</tr>
</tbody>
</table>
### Truth Table to Logic (Part 3)

<table>
<thead>
<tr>
<th>$d_2d_1d_0$</th>
<th>L</th>
<th>$c_0$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>Now, we do $c_1$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN 000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>SUN 000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>MON 001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>MON 001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>TUE 010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>TUE 010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>WED 011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>WED 011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>THU 100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2 \cdot d_1' \cdot d_0'$</td>
</tr>
<tr>
<td>FRI 101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$d_2 \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
<tr>
<td>FRI 101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2 \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
<tr>
<td>SAT 110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$d_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
<tr>
<td>- 111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$c_2 = d_2' \cdot d_1 \cdot d_0 \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
</tbody>
</table>

No matter what $L$ is, we always say it’s 1. So, we don’t need $L$ in the expression.
Truth Table to Logic (Part 3)

<table>
<thead>
<tr>
<th>$d_2d_1d_0$</th>
<th>L</th>
<th>$c_0$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>Now, we do $c_1$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$d_2' \cdot d_1' \cdot d_0' \cdot L'$</td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2 \cdot d_1' \cdot d_0'$</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$d_2 \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$d_2 \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
<tr>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$</td>
</tr>
<tr>
<td>111</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$c_2 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$</td>
</tr>
</tbody>
</table>

No matter what $L$ is, we always say it’s 1. So, we don’t need $L$ in the expression.
### Truth Table to Logic (Part 4)

<table>
<thead>
<tr>
<th></th>
<th>$d_2d_1d_0$</th>
<th>$L$</th>
<th>$c_0$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUN</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUN</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MON</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TUE</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WED</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>THU</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Finally, we do $c_0$:

- $d_2 \cdot d_1 \cdot d_0 \cdot L'$
- $d_2 \cdot d_1 \cdot d_0'$
- $d_2 \cdot d_1 \cdot d_0$
Truth Table to Logic (Part 4)

\[ c_0 = d_2 \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \]

\[ c_1 = d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0' \cdot L + d_2 \cdot d_1' \cdot d_0' \cdot L + d_2 \cdot d_1' \cdot d_0' \cdot L 
\]

\[ c_2 = d_2' \cdot d_1' \cdot d_0 \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L 
\]

\[ c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L 
\]

Here's \( c_3 \) as a circuit:
Boolean Algebra

• **Usual notation used in circuit design**

• **Boolean algebra**
  – a set of elements $B$ containing \{0, 1\}
  – binary operations { + , • }
  – and a unary operation { ’ }
  – such that the following axioms hold:

For any $a, b, c$ in $B$:
1. **closure**: $a + b$ is in $B$
2. **commutativity**: $a + b = b + a$
3. **associativity**: $a + (b + c) = (a + b) + c$
4. **distributivity**: $a + (b \cdot c) = (a + b) \cdot (a + c)$
5. **identity**: $a + 0 = a$
6. **complementarity**: $a + a' = 1$
7. **null**: $a + 1 = 1$
8. **idempotency**: $a + a = a$
9. **involution**: $(a')' = a$

$a \cdot b$ is in $B$
$a \cdot b = b \cdot a$
$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
$a \cdot 1 = a$
$a \cdot a' = 0$
$a \cdot 0 = 0$
$a \cdot a = a$
Simplification using Boolean Algebra

uniting:
10. \( a \cdot b + a \cdot b' = a \)
10D. \( (a + b) \cdot (a + b') = a \)

absorption:
11. \( a + a \cdot b = a \)
11D. \( a \cdot (a + b) = a \)
12. \( (a + b') \cdot b = a \cdot b \)
12D. \( (a \cdot b') + b = a + b \)

factoring:
13. \( (a + b) \cdot (a' + c) = a \cdot c + a' \cdot b \)
13D. \( a \cdot b + a' \cdot c = \)
\[ a \cdot c + a' \cdot b \]

consensus:
14. \( (a \cdot b) + (b \cdot c) + (a' \cdot c) = a \cdot b + a' \cdot c \)
14D. \( (a + b) \cdot (b + c) \cdot (a' + c) = \)
\[ (a + b) \cdot (a' + c) \]

de Morgan’s:
15. \( (a + b + \ldots)' = a' \cdot b' \cdot \ldots \)
15D. \( (a \cdot b \cdot \ldots)' = a' + b' + \ldots \)
Proving Theorems

Using the laws of Boolean Algebra:

prove the Uniting theorem:

\[ X \cdot Y + X \cdot Y' = X \]

\[ X \cdot Y + X \cdot Y' = X \cdot (Y + Y') \quad \text{dult} \]
\[ = X \cdot 1 \quad \text{compl} \]
\[ = X \quad \text{ident} \]

prove the Absorption theorem:

\[ X + X \cdot Y = X \]

\[ X + X \cdot Y = X \cdot 1 + X \cdot Y \quad \text{idnt} \]
\[ = X \cdot (1 + Y) \quad \text{dist} \]
\[ = X \cdot (Y + 1) \quad \text{un} \]
\[ = X \cdot 1 \quad \text{null} \]
\[ = X \quad \text{null} \]
Proving Theorems

Using the laws of Boolean Algebra:

prove the Uniting theorem:

\[ X \cdot Y + X \cdot Y' = X \]

- distributivity
- complementarity
- identity

prove the theorem:

\[ X + X \cdot Y = X \]

- identity
- distributivity
- commutativity
- null
- identity
Proving Theorems

Using truth table:

For example, de Morgan’s Law:

\[(X + Y)' = X' \cdot Y'\]
NOR is equivalent to AND with inputs complemented

\[(X \cdot Y)' = X' + Y'\]
NAND is equivalent to OR with inputs complemented

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X'</th>
<th>Y'</th>
<th>(X + Y)'</th>
<th>X' • Y'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X'</th>
<th>Y'</th>
<th>(X • Y)'</th>
<th>X' + Y'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Simplifying using Boolean Algebra

c3 = d2\cdot d1\cdot d0\cdot L + d2\cdot d1\cdot d0\cdot L
  = d2\cdot d1\cdot (d0' + d0)\cdot L
  = d2\cdot d1\cdot 1\cdot L
  = d2\cdot d1\cdot L
1-bit Binary Adder

\[
\begin{array}{c}
A \\
+ B \\
\hline
S \\
(\text{C}_{\text{OUT}}) \\
\hline
\end{array}
\]

\[
\begin{align*}
0 + 0 &= 0 \text{ (with } C_{\text{OUT}} = 0) \\
0 + 1 &= 1 \text{ (with } C_{\text{OUT}} = 0) \\
1 + 0 &= 1 \text{ (with } C_{\text{OUT}} = 0) \\
1 + 1 &= 0 \text{ (with } C_{\text{OUT}} = 1) \\
\end{align*}
\]
### 1-bit Binary Adder

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 + 0 = 0 (with $C_{OUT} = 0$)</td>
<td></td>
</tr>
<tr>
<td>+ B</td>
<td>0 + 1 = 1 (with $C_{OUT} = 0$)</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>1 + 0 = 1 (with $C_{OUT} = 0$)</td>
<td></td>
</tr>
<tr>
<td>(C_{OUT})</td>
<td>1 + 1 = 0 (with $C_{OUT} = 1$)</td>
<td></td>
</tr>
</tbody>
</table>

Idea: To chain these together, let's add a carry-in
1-bit Binary Adder

<table>
<thead>
<tr>
<th>A</th>
<th>+ B</th>
<th>S</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>

Idea: To chain these together, let’s add a carry-in.
1-bit Binary Adder

\[
\begin{array}{c}
&A & 0 + 0 = 0 \text{ (with } C_{OUT} = 0) \\
+ & B & 0 + 1 = 1 \text{ (with } C_{OUT} = 0) \\
&S & 1 + 0 = 1 \text{ (with } C_{OUT} = 0) \\
&(C_{OUT}) & 1 + 1 = 0 \text{ (with } C_{OUT} = 1) \\
\end{array}
\]

Idea: To chain these together, let’s add a carry-in

\[
\begin{array}{c}
\text{A} \\
\text{B} \\
\text{S} \\
\text{C}_{IN} \\
\text{C}_{OUT} \\
\end{array}
\]
1-bit Binary Adder

- **Inputs:** A, B, Carry-in
- **Outputs:** Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C_IN</th>
<th>C_OUT</th>
<th>S</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
1-bit Binary Adder

- **Inputs**: A, B, Carry-in
- **Outputs**: Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C\text{\textsubscript{IN}}</th>
<th>C\text{\textsubscript{OUT}}</th>
<th>S</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Derive an expression for S:

\[ S = A' \cdot B' \cdot C\text{\textsubscript{IN}} + A' \cdot B \cdot C\text{\textsubscript{IN}}' + A \cdot B' \cdot C\text{\textsubscript{IN}}' + A \cdot B \cdot C\text{\textsubscript{IN}} \]
1-bit Binary Adder

- **Inputs:** A, B, Carry-in
- **Outputs:** Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C_{IN}</th>
<th>C_{OUT}</th>
<th>S</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Derive an expression for C_{OUT}**

\[ C_{OUT} = A' \cdot B \cdot C_{IN} + A \cdot B' \cdot C_{IN} + A \cdot B \cdot C_{IN}' + A' \cdot B' \cdot C_{IN} \]

**S = A' \cdot B' \cdot C_{IN} + A' \cdot B \cdot C_{IN}' + A \cdot B' \cdot C_{IN}' + A \cdot B \cdot C_{IN}**
1-bit Binary Adder

- **Inputs:** A, B, Carry-in
- **Outputs:** Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C\textsubscript{IN}</th>
<th>C\textsubscript{OUT}</th>
<th>S</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ S = A' \cdot B \cdot C\textsubscript{IN} + A' \cdot B \cdot C\textsubscript{IN}' + A \cdot B' \cdot C\textsubscript{IN}' + A \cdot B \cdot C\textsubscript{IN} \]

\[ C\textsubscript{OUT} = A' \cdot B \cdot C\textsubscript{IN} + A \cdot B' \cdot C\textsubscript{IN} + A \cdot B \cdot C\textsubscript{IN}' + A \cdot B \cdot C\textsubscript{IN} \]
The theorems of Boolean algebra can simplify expressions

– e.g., full adder’s carry-out function

\[
\begin{align*}
\text{Cout} & = A' B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = A' B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = A' B \text{ Cin} + A B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = (A' + A) B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = (1) B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = B \text{ Cin} + A (B' + B) \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = B \text{ Cin} + A (1) \text{ Cin} + A B \text{ Cin'} + A B \text{ Cin} \\
& = B \text{ Cin} + A \text{ Cin} + A B (\text{ Cin'} + \text{ Cin}) \\
& = B \text{ Cin} + A \text{ Cin} + A B (1) \\
& = B \text{ Cin} + A \text{ Cin} + A B
\end{align*}
\]
Multi-bit Ripple-Carry Adder

1-Bit Adder

\[ \begin{align*}
A & \rightarrow \text{Cout} \\
B & \rightarrow \text{Cout} \\
\text{Cin} & \rightarrow \text{Cout} \\
A & \rightarrow \text{Sum} \\
B & \rightarrow \text{Sum} \\
\text{Cin} & \rightarrow \text{Sum}
\end{align*} \]

\[ \begin{align*}
A_2 & \rightarrow \text{Sum}_2 \\
B_2 & \rightarrow \text{Sum}_2 \\
\text{Cout} & \rightarrow \text{Sum}_2 \\
\text{Cin} & \rightarrow \text{Sum}_2 \\
A_1 & \rightarrow \text{Sum}_1 \\
B_1 & \rightarrow \text{Sum}_1 \\
\text{Cout} & \rightarrow \text{Sum}_1 \\
\text{Cin} & \rightarrow \text{Sum}_1 \\
A_0 & \rightarrow \text{Sum}_0 \\
B_0 & \rightarrow \text{Sum}_0 \\
\text{Cout} & \rightarrow \text{Sum}_0 \\
\text{Cin} & \rightarrow \text{Sum}_0
\end{align*} \]
Mapping Truth Tables to Logic Gates

Given a truth table:

1. Write the Boolean expression
2. Minimize the Boolean expression
3. Draw as gates
4. Map to available gates

\[ \begin{array}{ccc|c}
A & B & C & F \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array} \]

\[ F = A'BC' + A'BC + AB'C + ABC \]

\[ = A'B(C' + C) + AC(B' + B) \]

\[ = A'B + AC \]

\[ = A'B + AC \]

\[ \begin{array}{c}
\text{notA} \\
\text{B} \\
\text{A} \\
\text{C} \\
\end{array} \quad \begin{array}{c}
\text{notA} \\
\text{B} \\
\text{A} \\
\text{C} \\
\end{array} \]

\[ \begin{array}{c}
\text{notA} \\
\text{B} \\
\text{A} \\
\text{C} \\
\end{array} \quad \begin{array}{c}
\text{notA} \\
\text{B} \\
\text{A} \\
\text{C} \\
\end{array} \]

\[ \begin{array}{c}
\text{notA} \\
\text{B} \\
\text{A} \\
\text{C} \\
\end{array} \quad \begin{array}{c}
\text{notA} \\
\text{B} \\
\text{A} \\
\text{C} \\
\end{array} \]
Canonical Forms

• Truth table is the unique signature of a Boolean Function

• The same truth table can have many gate realizations
  – We’ve seen this already
  – Depends on how good we are at Boolean simplification

• Canonical forms
  – Standard forms for a Boolean expression
  – We all come up with the same expression
Sum-of-Products Canonical Form

- AKA Disjunctive Normal Form (DNF)
- AKA Minterm Expansion

F = A’B’C + A’BC + AB’C + ABC’ + ABC'

1. Read T rows off truth table
2. Convert to Boolean Algebra
3. Add the minterms together
Sum-of-Products Canonical Form

Product term (or minterm)

- ANDed product of literals – input combination for which output is true
- each variable appears exactly once, true or inverted (but not both)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>minterms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A’B’C’</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A’B’C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A’BC’</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>A’BC</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>AB’C’</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>AB’C</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>ABC’</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ABC</td>
</tr>
</tbody>
</table>

F in canonical form:

\[
F(A, B, C) = A’B’C + A’BC + AB’C + ABC’ + ABC
\]

canonical form ≠ minimal form

\[
F(A, B, C) = A’B’C + A’BC + AB’C + ABC + ABC’
\]

\[
= (A’B’ + A’B + AB’ + AB)C + ABC’
\]

\[
= ((A’ + A)(B’ + B))C + ABC’
\]

\[
= C + ABC’
\]

\[
= ABC’ + C
\]

\[
= AB + C
\]
Product-of-Sums Canonical Form

- AKA Conjunctive Normal Form (CNF)
- AKA Maxterm Expansion

F =

1. Read F rows off truth table
2. Negate all bits
3. Convert to Boolean Algebra
4. Multiply the maxterms together

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</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>1</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>1</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>
Product-of-Sums Canonical Form

- AKA **Conjunctive Normal Form (CNF)**
- AKA **Maxterm Expansion**

\[
F = (A + B + C)(A + B' + C)(A' + B + C)
\]

1. **Read F rows off truth table**
2. **Negate all bits**
3. **Convert to Boolean Algebra**
4. **Multiply the maxterms together**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</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>1</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>1</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>
Product-of-Sums: Why does this procedure work?

Useful Facts:
- We know \((F')' = F\)
- We know how to get a minterm expansion for \(F'\)

\[
\begin{array}{c|c|c|c|c}
A & B & C & F \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[F' = A'B'C' + A'BC' + AB'C'\]
Product-of-Sums: Why does this procedure work?

Useful Facts:

- We know \((F')' = F\)
- We know how to get a minterm expansion for \(F'\)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</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>1</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>1</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>

\[
F' = A'B'C' + A'BC' + AB'C'
\]

Taking the complement of both sides...

\[
(F')' = (A'B'C' + A'BC' + AB'C')'
\]

And using DeMorgan/Comp....

\[
F = (A'B'C')' (A'BC')' (AB'C')'
\]

\[
F = (A + B + C)(A + B' + C)(A' + B + C)
\]
Product-of-Sums Canonical Form

Sum term (or maxterm)
- ORed sum of literals – input combination for which output is false
- each variable appears exactly once, true or inverted (but not both)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>maxterms</th>
<th>F in canonical form:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A+B+C</td>
<td>F(A, B, C) = (A + B + C) (A + B’ + C) (A’ + B + C)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A+B+C’</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A+B’+C</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>A+B’+C’</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>A’+B+C</td>
<td>canonical form ≠ minimal form</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>A’+B+C’</td>
<td>F(A, B, C) = (A + B + C) (A + B’ + C) (A’ + B + C)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>A’+B’+C</td>
<td>= (A + B + C) (A + B’ + C)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A’+B’+C’</td>
<td>(A + B + C) (A’ + B + C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= (A + C) (B + C)</td>
</tr>
</tbody>
</table>