(2 + q + v)(c + q + v)(c + q + v)

Product of sums:

Sum of products: abc + abv + abc + avc

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>abc</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<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>1</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<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>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>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Truth Tables To Logic Equations

Precedence: AND takes precedence over OR.

\[ a \land b = (ab) \]
\[ a \lor (b \land c) = (a \lor b) \land c \]
\[ a \lor (b \lor c) = (a \lor b) \lor c \]
\[ a \land (b \lor c) = (a \land b) \lor c \]
\[ a \land b = a \lor b \]
\[ a \lor b = a \lor b \]
\[ a \land 0 = 0 \]
\[ a \lor 0 = a \]
\[ a \land 1 = a \]
\[ a \lor 1 = 1 \]
\[ a \land a = a \]
\[ a \lor a = a \]

Boolean Algebra

Proving Logic Equations

Example: \((a \lor b) \land c = a \lor (b \land c)\)

Boolean algebra: tool to manipulate logic equations

Operations: AND, OR, complement

An algebra on a set of two elements: \{0, 1\}

5 wiring circuits

5 gate diagrams

5 truth tables

5 logic equations

Multiple levels of representation:

Combinational logic
Can be used to simplify logic equations.

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

Given: the input is always between 0 and 4.

Don't Cares

(details in section next week)

- Quine-McCluskey
- Karnaugh maps

Symmetric techniques:

One can use Boolean algebra to simplify equations.

\[ q^p = \frac{2q^p + 2q^p}{2q^p + 2q^p} + 2q^p + 2q^p \]

Earlier example:

<table>
<thead>
<tr>
<th>Input 0</th>
<th>Input 1</th>
<th>Output 0</th>
<th>Output 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>

Representational: 3-bit binary input

Increment input by 1, compute result mod 5

Word Problems

Universal: NAND and NOR

Using just NAND or just NOR gates.
How do we change the state? A = B

Part I: State-holding devices

A simple device:

SR latch

SR Latch

Sequential Circuits

combinational logic
• Generate output from inputs using
• Write stable outputs to state-holding elements
• Read stable inputs from state-holding elements

called state-holding elements

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[diagram of SR latch]

In a stable state, A = B

stable state

What happens?

[diagram of SR latch with add 1 mod 5]

What if I want to keep counting?
Clock frequency = 1/clock period.

Clock signal has a fixed cycle time (a.k.a. cycle period).

Introduce a free-running signal: the clock.

Part II: modifying state-holding elements

How does the output behave?

CLK = 

Q

D

S

R

Q

D

S

R

Q

D

State-holding element

If outputs change here, negative edge-triggered

If outputs change here, positive edge-triggered

Need to control when the output changes.

...immediately.

When Q changes, 0 changes.

Edge Triggered Clocking

Lot of other choices...  (EE 438)

Input must be stable just before the clock edge

D Latch
Designing a Finite-State Machine

Example: 1-bit Counter

Circuit:

Truth-table:

Moore: output is a function of state only
Mealy: output is a function of state and input

Two Types:
- Current State
- Input

Output and next state depend on:
- Internal state
- Externally visible outputs
- External inputs

Basic Idea: A circuit has

Master-Slave Flip-Flop
What's the clock period?

Logic Equations and Circuit

\[ s \cdot \bar{q} + \bar{s} \cdot q = s \cdot \bar{q} \cdot q + s \cdot \bar{q} + \bar{s} \cdot q = i \]

\[ s \cdot \bar{q} + \bar{s} \cdot q + \bar{s} = z \]

State Table

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>I</th>
<th>S1</th>
<th>S0</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>T</td>
<td>T</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>T</td>
<td>0</td>
<td>1</td>
<td>T</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>T</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>I</td>
<td>S</td>
<td>0</td>
<td>S</td>
</tr>
<tr>
<td>0</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>T</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Next state

Output: \( z \)

Input: \( a \) and \( b \)

Two states: \( S0 \) (carry is zero), \( S1 \) (carry is 1)

State Assignment

<table>
<thead>
<tr>
<th>Sp/q</th>
<th>S0</th>
<th>S1</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>T</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>T</td>
<td>I</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Pick encoding of states. We have two states, so use

The Serial Adder