CMSC 313 Lecture 21

• Last time: Karnaugh Map Examples
• Higher Dimension Karnaugh Maps
• Quine-McCluskey Algorithm
• Introduction to Sequential Logic
• Flip-Flops
Five-Variable K-Map

- Visualize two 4-variable K-maps stacked one on top of the other; groupings are made in three dimensional cubes.

\[ F = \overline{A} \overline{C} D E + \overline{A} B D E + B E \]
Six-Variable K-Map

- Visualize four 4-variable K-maps stacked one on top of the other; groupings are made in three dimensional cubes.

\[ G = \overline{B} \overline{C} \overline{E} F + \overline{A} B \overline{D} E \]
Truth Table with Don’t Cares

- A truth table representation of a single function with don’t cares.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$d$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>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>0</td>
<td>1</td>
<td>1</td>
<td>$d$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$d$</td>
</tr>
</tbody>
</table>
Tabular (Quine-McCluskey) Reduction

- Tabular reduction begins by grouping minterms for which $F$ is nonzero according to the number of 1’s in each minterm. Don’t cares are considered to be nonzero.

- The next step forms a consensus (the logical form of a cross product) between each pair of adjacent groups for all terms that differ in only one variable.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

(a) Initial setup

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>_</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>1</td>
<td>_</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>0</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>1</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>_</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>_</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>_</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>_</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>_</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>1</td>
<td>_</td>
<td>1</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>_</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
</tbody>
</table>

(b) After first reduction

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>_</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>_</td>
<td>1</td>
<td>1</td>
<td>_</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>_</td>
<td>1</td>
<td>_</td>
<td>1</td>
<td>_</td>
<td>_</td>
</tr>
</tbody>
</table>

(c) After second reduction
Table of Choice

- The prime implicants form a set that completely covers the function, although not necessarily minimally.
- A *table of choice* is used to obtain a minimal cover set.

<table>
<thead>
<tr>
<th>Prime Implicants</th>
<th>Minterms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0001 0011 0101 0110 0111 1010 1101</td>
</tr>
<tr>
<td>0 0 0 _</td>
<td>√</td>
</tr>
<tr>
<td>* 0 1 1 _</td>
<td></td>
</tr>
<tr>
<td>* 1 0 1 _</td>
<td></td>
</tr>
<tr>
<td>0 _ _ 1</td>
<td>√ √ √ √</td>
</tr>
<tr>
<td>_ _ 1 1</td>
<td>√</td>
</tr>
<tr>
<td>* _ 1 _ 1</td>
<td>√ √ √ √</td>
</tr>
</tbody>
</table>
Reduced Table of Choice

- In a reduced table of choice, the essential prime implicants and the minterms they cover are removed, producing the *eligible set*.
- \( F = \overline{ABC} + \overline{ABC} + BD + \overline{AD} \)

<table>
<thead>
<tr>
<th>Eligible Set</th>
<th>Minterms</th>
<th>Set 1</th>
<th>Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 000_</td>
<td>0001</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Y 0__1</td>
<td>0011</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Z __11</td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>
**Multiple Output Truth Table**

- The power of tabular reduction comes into play for multiple functions, in which minterms can be shared among the functions.

<table>
<thead>
<tr>
<th>Minterm</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F₀</th>
<th>F₁</th>
<th>F₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>m₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m₁</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>m₂</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m₃</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m₄</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>m₅</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m₆</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m₇</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Multiple Output Table of Choice

\[ F_0(A,B,C) = \overline{A}\overline{B}\overline{C} + BC \]
\[ F_1(A,B,C) = \overline{A}C + A\overline{C} + BC \]
\[ F_2(A,B,C) = B \]

<table>
<thead>
<tr>
<th>Prime Implicants</th>
<th>Min-terms</th>
<th>( F_0(A,B,C) )</th>
<th>( F_1(A,B,C) )</th>
<th>( F_2(A,B,C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_0 )</td>
<td>* 0 0 0</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( F_1 )</td>
<td>* 0 1</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>( F_1 )</td>
<td>* 1 0</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>( F_2 )</td>
<td>* 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( F_{1,2} )</td>
<td>1 1</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>( F_{0,1,2} )</td>
<td>* 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sequential Logic

- The combinational logic circuits we have been studying so far have no memory. The outputs always follow the inputs.
- There is a need for circuits with memory, which behave differently depending upon their previous state.
- An example is a vending machine, which must remember how many and what kinds of coins have been inserted. The machine should behave according to not only the current coin inserted, but also upon how many and what kinds of coins have been inserted previously.
- These are referred to as finite state machines, because they can have at most a finite number of states.
Classical Model of a Finite State Machine

- An FSM is composed of a combinational logic unit and delay elements (called flip-flops) in a feedback path, which maintains state information.

![Diagram of FSM](image-url)
S-R Flip-Flop

- The S-R flip-flop is an active high (positive logic) device.

<table>
<thead>
<tr>
<th>$Q_t$</th>
<th>$S_t$</th>
<th>$R_t$</th>
<th>$Q_{i+1}$</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 (disallowed)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</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>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 (disallowed)</td>
</tr>
</tbody>
</table>

Timing Behavior
NAND Implementation of S-R Flip-Flop

\[
\begin{array}{c}
\text{S} & \text{Q} & \overline{Q} \\
\text{R} & Q & \overline{R}
\end{array}
\quad = \quad
\begin{array}{c}
\text{S} & \text{Q} & \overline{Q} \\
\text{R} & Q & \overline{R}
\end{array}
\quad = \quad
\begin{array}{c}
\overline{R} & Q & \overline{Q} \\
\overline{S} & \overline{Q}
\end{array}
\]
SR Latch using NAND GATES
A Hazard

• It is desirable to be able to “turn off” the flip-flop so it does not respond to such hazards.
A Clock Waveform: The Clock Paces the System

- In a positive logic system, the “action” happens when the clock is high, or positive. The low part of the clock cycle allows propagation between subcircuits, so their inputs settle at the correct value when the clock next goes high.
Clocked S-R Flip-Flop

- The clock signal, CLK, enables the S and R inputs to the flip-flop.
Clocked D Flip-Flop

- The clocked D flip-flop, sometimes called a *latch*, has a potential problem: If D changes while the clock is high, the output will also change. The *Master-Slave* flip-flop (next slide) addresses this problem.

![Clocked D Flip-Flop Diagram](image)

**Timing Behavior**

- $Q$ and $\bar{Q}$ change only when the clock is high.
- $D$ changes during the clock high period.
- The delay time, $\Delta t$, is the time it takes for the output to change.
Master-Slave Flip-Flop

- The rising edge of the clock loads new data into the master, while the slave continues to hold previous data. The falling edge of the clock loads the new master data into the slave.
Next Time

• Edge-triggered Flip Flops
• Introduction to Finite State Machines