Switching Theory / Schakeltechniek

Lab session

Sequential circuits

Highlights previous session

Number systems
- From binary to decimal:
  \[11010 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 26\]
- From decimal to binary:
  \[
  \begin{align*}
  47 / 2 &= 23 & \text{remainder} & 1 \\
  23 / 2 &= 11 & \text{"} & 1 \\
  11 / 2 &= 5 & \text{"} & 1 \\
  5 / 2 &= 2 & \text{"} & 1 \\
  2 / 2 &= 1 & \text{"} & 0 \\
  1 / 2 &= 0 & \text{"} & 1
  \end{align*}
  \]
  47 in binary?
- 2's complement:
  \[
  \begin{array}{c}
  \text{dec:} & +4 & \text{dec:} & -4 \\
  \text{2's c:} & 0100 & \text{2's c:} & 1100
  \end{array}
  \]
  check: \[32 + 8 + 4 + 2 + 1 = 47\]

Circuits with memory

Memory:
- behavior of a circuit is determined by the current inputs plus inputs from the past
- Lock only opens when a designated sequence of inputs is received
- Combinational circuits do not have memory
A simple memory element

- values are stable: it remembers a 0 or 1 value
- difficult to change a value

Truth table:

<table>
<thead>
<tr>
<th>c</th>
<th>x</th>
<th>c + x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
</tr>
</tbody>
</table>

Idea:
- reset Q (Q becomes 0) if R is 1
- set Q (Q becomes 1) if S is 1

The R-S latch

Truth table:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q̅</th>
</tr>
</thead>
</table>
| 0 | 0 | Q   | Q̅   | hold
| 0 | 1 | 0   | 1    | reset / set
| 1 | 0 | 1   | 0    | forbidden input

Gated R-S latch

Changes only possible if clk is high

But ... ?

Clock:
- independent periodic reference signal
- allows control over changes in memory values

D flip-flop

Q copies the value on input D at each rising clock edge
→ (positive/leading) edge triggered ff
Parity checker: state diagram

Design a circuit that provides a high output if and only if the number of 1s received on its input is odd.

Idea: 2 states (even and odd)

- States: xyz
- State names: [0] even, [1] odd
- Output values: [uv] st state transition, initial state

Parity checker: transition table

<table>
<thead>
<tr>
<th>Q</th>
<th>i</th>
<th>Q'</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>even</td>
<td>0</td>
<td>even</td>
<td>0</td>
</tr>
<tr>
<td>even</td>
<td>1</td>
<td>odd</td>
<td>0</td>
</tr>
<tr>
<td>odd</td>
<td>0</td>
<td>odd</td>
<td>1</td>
</tr>
<tr>
<td>odd</td>
<td>1</td>
<td>even</td>
<td>1</td>
</tr>
</tbody>
</table>

- Q: current state
- Q': next state

Parity checker: implementation

- Truth table:
<table>
<thead>
<tr>
<th>Q</th>
<th>i</th>
<th>D</th>
<th>o</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- Expressions:
  \[ D = i \oplus Q \]
  \[ o = Q \]
Sequential circuits: design trajectory

- state diagram
- transition table
- binary coding of states and i/o, and selection of flip-flops
- truth table
- expressions and optimisation (K-maps, multi-level)
- implementation

Example: vending machine

design a controller for a machine that accepts 5c and 10c coins, and provides coffee if it has received (at least) 15c

inputs coded:
\[ x y = 5c \ 10c \]
11 is invalid input

Vending machine: transition table

<table>
<thead>
<tr>
<th>Q</th>
<th>5c 10c</th>
<th>Q'</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0c</td>
<td>00</td>
<td>0c</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>10c</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5c</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Vending machine: truth table

Transition table contains exactly the same information as the state diagram

Vending machine: truth table

State coding:
- 0c = 00
- 5c = 01
- 10c = 10
- 15c = 11

2 D flip-flops

<table>
<thead>
<tr>
<th>Q0Q1 5c10c</th>
<th>D0D1  o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>1 1 x x 0</td>
<td>1 1 x x</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>1 1 x x 0</td>
<td>1 1 x x</td>
</tr>
</tbody>
</table>
**Vending machine: K-maps**

\[
D0 = Q0 + 10c + 5cQ1 \\
D1 = 10cQ0 + 5cQ1 + 5cQ0 + 5cQ1 \\
o = Q0Q1
\]

**Vending machine: implementation**

**Moore machines**