Digital Circuit Design and Language

Combinational / Sequential Logic

Chang, Ik Joon
Kyunghhee University
Combinational Logic

- The outputs are determined by the present inputs
- Consist of input/output variables and logic gates
- No memory / No feedback
Combinational Logic Design
Procedure (Gate-level Design)

1. Decide input and output signals
2. Derive the truth table based on the relationship between inputs and outputs
3. Obtain the simplified Boolean Algebra
4. Draw the logic diagram and verify the design (manually or by simulation)

cf. Behavioral Level and Data-flow Level Design
Sequential Logic

- Outputs are function of inputs and present states
- Present states are supplied by memory elements
Synchronous vs. Asynchronous

- **Synchronous**: Controlled by periodical clock signal
- **Asynchronous**: Controlled by non-periodical signal
Storage Elements: Latch vs. Flip-flop

- **Latch**: Operates with clock level (level-sensitive)
- **Flip-flop**: Controlled by clock transition (edge-sensitive)

What is Registers and RAM?

(a) Response to positive level

(b) Positive-edge response
For two cross-coupled NOR gates,

- S=1, R=0 then Q=1 (set)
- S=0, R=1 then Q=0 (reset)
- S=0, R=0 then no change (keep condition)
- S=1, R=1 Q=Q’=0 (undefined)

For two cross-coupled NAND gates, S and R are inverted compared to the above.
Eliminate indeterminate state in SR latch

- C=1, transparent
- C=0, locked (no output change)
Edge-Triggered D Flip-Flop

- $C=0$ : master disable, slave enable
- $C=1$ : master enable, slave disable
- Negative-edge or Positive-edge?
Edge-Triggered D Flip-Flop (Cont.)

- D-type positive edge triggered flip flop
  - Consist of 3 SR-latches
  - Q changes only when C becomes 0 to 1
Other Flip-flops – JK Flip-Flop

(a) Circuit diagram

(b) Graphic symbol

Table 5-1
Flip-Flop Characteristic Tables

| JK Flip-Flop |  |  
|--------------|---|---|
| $Q(t+1)$     |   |   |
| $J$ $K$      |   |   |
| 0 0          | $Q(t)$ | No change |
| 0 1          | 0   | Reset    |
| 1 0          | 1   | Set      |
| 1 1          | $Q'(t)$ | Complement |

- Performs three operations
- Set($J$), Reset($K$), Complement($J=K=1$)
- $D=JQ' + K'Q$
Other Flip-flops – T Flip-Flop

(a) From $JK$ flip-flop

(b) From $D$ flip-flop

(c) Graphic symbol

$D = TQ' + T'Q$

- Complementing flip-flop

<table>
<thead>
<tr>
<th>$T$</th>
<th>$Q(t+1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$Q(t)$ No change</td>
</tr>
<tr>
<td>1</td>
<td>$Q'(t)$ Complement</td>
</tr>
</tbody>
</table>

- Complementing flip-flop
- $D = TQ' + T'Q$
Analysis of Clocked Sequential Logic

$$A(t+1) = Ax + Bx$$

$$B(t+1) = A'x$$

$$Y = (A+B)x'$$

- Clocked sequential Logic can be described using finite state machines (Behavioral Model)
Circuit Diagram and State Equation

\[ A(t+1) = A(t)x(t) + B(t)x(t) \]

\[ B(t+1) = A'(t)x(t) \]

\[ Y = [A(t) + B(t)]x'(t) \]
<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  B</td>
<td>x</td>
<td>A  B</td>
<td>y</td>
</tr>
<tr>
<td>0  0</td>
<td>0</td>
<td>0  0</td>
<td>0</td>
</tr>
<tr>
<td>0  0</td>
<td>1</td>
<td>0  1</td>
<td>0</td>
</tr>
<tr>
<td>0  1</td>
<td>0</td>
<td>0  0</td>
<td>1</td>
</tr>
<tr>
<td>0  1</td>
<td>1</td>
<td>1  1</td>
<td>0</td>
</tr>
<tr>
<td>1  0</td>
<td>0</td>
<td>0  0</td>
<td>1</td>
</tr>
<tr>
<td>1  0</td>
<td>1</td>
<td>1  0</td>
<td>0</td>
</tr>
<tr>
<td>1  1</td>
<td>0</td>
<td>0  0</td>
<td>1</td>
</tr>
<tr>
<td>1  1</td>
<td>1</td>
<td>1  0</td>
<td>0</td>
</tr>
</tbody>
</table>
Gate-level Design (Circuit Diagram): Need to know hardware implementation

Behavioral Model (State Diagram): Does not require the background of hardware implementation
Finite State Machine Model: Mealy Machine vs. Moore Machine

(a) Mealy Machine

(b) Moore Machine
Moore vs. Mealy

Moore Machine: The outputs are determined by the present states only
- Outputs are changed at only clock edges
- Pros: Safe, Easy to debug
- Cons: Large memory size

Mealy Machine: The outputs are determined by both present states and inputs
- Outputs can be changed near clock edges
- Pros: Small memory size
- Cons: Complex

What is more preferred?
Design Procedure

Automated Design Steps

1) Derive a state diagram or state table
2) Reduce the number of states if necessary (State Reduction)
3) Assign binary code to the state (State Assignment)
4) Make a HDL description and run logic synthesis using ‘design compiler’ (→ netlist)

Manual Design Steps

1) Derive a state diagram or state table
2) Reduce the number of states if necessary (State Reduction)
3) Assign binary code to the state (State Assignment)
4) Choose the type of flip-flops to be used
5) Derive the flip-flop input equations and output equations
6) Draw the logic diagram
EQUIVALENT STATES

Two States, $s_i$ and $s_j$ Equivalent IFF
- Give Exact Same Output Sequence, For Any Possible Input Sequence $X$
State Reduction Algorithm

For N states, at least \( \log_2 N \) Flip-Flops are necessary

- To reduce # of flip-flop, we need to reduce # of states

State Reduction Algorithm

- When two states are equivalent, one can be removed
- Repeat this until we are not able to find equivalent ones
State Reduction

Reducing the State Table

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x = 0$</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
</tr>
<tr>
<td>$c$</td>
<td>$a$</td>
<td>$d$</td>
</tr>
<tr>
<td>$d$</td>
<td>$e$</td>
<td>$f$</td>
</tr>
<tr>
<td>$e$</td>
<td>$a$</td>
<td>$f$</td>
</tr>
<tr>
<td>$f$</td>
<td>$g$</td>
<td>$f$</td>
</tr>
<tr>
<td>$g$</td>
<td>$a$</td>
<td>$f$</td>
</tr>
</tbody>
</table>
State Reduction (Cont.)

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x = 0$</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
</tr>
<tr>
<td>$c$</td>
<td>$a$</td>
<td>$d$</td>
</tr>
<tr>
<td>$d$</td>
<td>$e$</td>
<td>$d$</td>
</tr>
<tr>
<td>$e$</td>
<td>$a$</td>
<td>$d$</td>
</tr>
</tbody>
</table>
STATE ASSIGNMENT

One-Hot State Encoding
- One Flip-Flop per State
- Simple Decoding Logic

Minimal State Encoding
- Use $\lceil \log_2(\#\text{states}) \rceil$ Flip-Flops
- Assignment of State Codes
  - Impacts Amount of Combinational Logic
BCD TO EXCESS-3 CODE CONVERTER

\[
\begin{align*}
0100 & \quad 0101 \\
+0011 & \quad +0011 \\
0111 & \quad 1000
\end{align*}
\]

LSB received first

<table>
<thead>
<tr>
<th>PS</th>
<th>X = 0</th>
<th>X = 1</th>
<th>X = 0</th>
<th>X = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₀</td>
<td>S₁</td>
<td>S₂</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S₁</td>
<td>S₃</td>
<td>S₄</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S₂</td>
<td>S₄</td>
<td>S₄</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S₃</td>
<td>S₅</td>
<td>S₅</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S₄</td>
<td>S₅</td>
<td>S₆</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S₅</td>
<td>S₀</td>
<td>S₀</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S₆</td>
<td>S₀</td>
<td>–</td>
<td>1</td>
<td>–</td>
</tr>
</tbody>
</table>
STATE ASSIGNMENT GUIDELINES

Assign Adjacent State Codes

- States with same NS for given input
- States that are NS of same state
- States having same output for given input

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$X = 0$</td>
<td>$X = 1$</td>
</tr>
<tr>
<td>$S_0$</td>
<td>$S_1$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>$S_3$</td>
<td>$S_4$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$S_4$</td>
<td>$S_4$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$S_5$</td>
<td>$S_5$</td>
</tr>
<tr>
<td>$S_4$</td>
<td>$S_5$</td>
<td>$S_6$</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$S_0$</td>
<td>$S_0$</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$S_0$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
ASSIGNMENT MAP AND TRANSITION TABLE

State Assignment Guidelines
- \((1,2), (3,4), (5,6), (0,1,4,6), (2,3,5)\)

<table>
<thead>
<tr>
<th>(Q_2Q_3)</th>
<th>(Q_1)</th>
<th>(Q_1^+ Q_2^+ Q_3^+) (X = 0)</th>
<th>(X = 1)</th>
<th>(Z) (X = 0)</th>
<th>(X = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>100</td>
<td>101</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>111</td>
<td>110</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>110</td>
<td>110</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>011</td>
<td>011</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>011</td>
<td>010</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>000</td>
<td>000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>xxx</td>
<td>xxx</td>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>xxx</td>
<td>xxx</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
REALIZATION OF CODE CONVERTER
MAXIMUM FREQUENCY OF OPERATION
TIMING CONDITIONS

- Clock period must be long enough to satisfy flip-flop setup time:
  \[ t_{ck} \geq t_{p\text{max}} + t_{c\text{max}} + t_{su} \]

- Clock period should be long enough to satisfy flip-flop hold time
  \[ t_{p\text{min}} + t_{c\text{min}} \geq t_h \]

- External input \[ t_x \geq t_{c\text{max}} + t_{su} \]
- External input \[ t_y \geq t_h - t_{c\text{min}} \]
TIMING FOR EXTERNAL INPUT CHANGE
What’s max frequency of operation of circuit?

- Min and max delays of inverter are 1ns and 3ns
- \( t_{p_{\text{min}}} \) and \( t_{p_{\text{max}}} \) are 5ns and 8ns
- Setup and hold times are 4ns and 2ns
TIMING EXAMPLE

What are safe regions for changes in $X$?

- Min and max delays of comb ckt are 2ns and 4ns
- $t_{p_{\text{min}}}$ and $t_{p_{\text{max}}}$ of flip-flop are 5ns and 10ns
- Setup and hold times are 8ns and 3ns
Typical Synchronous Design

- Synchronous Design
  - Clock Used to Synchronize Operation of All Flip-Flops
- Control Section: Finite State Machine
TIMING CHART

Falling-Edge Devices

State Change Initiated Here

Uncertain

Clock

Switching Transients

Control Signal

Clock·CS