CSE140-B: Review & Discussion Session for Midterm 3 (Updated)

2014/12/08 (19:00PM-19:50PM)

Hao Zhuang

Computer Science and Engineering Department,
University of California, San Diego
Preparations for Midterm 3

- **Timing**
  - Maximum frequency
    - Longest Path
  - Setup time violation
    - clock skew
  - Hold time violation
    - Shortest Path
    - clock skew

- **Standard Combinational Modules:**
  - Multiplexer, Decoder, etc
    - How to use them to implement Boolean function
    - Tips for the exam: always write down the input and output labels in your diagram
Preparations for Midterm 3

- System Design (covered this date)
  - Data subsystem
    - Table for operations (remember to write the state numbers)
    - Logic diagram
    - Control signals
  - Control subsystem
    - State diagram
    - Control signal for every state
    - One-hot state machine
Preparations for Midterm 3

- Useful Operation Modules
  - INC, DEC, SUB, ADD, COMP, SHL, SHR, LD, CLR, CS
Timing Analysis

This large adder is built from three full adders such that the carry out of the first adder is the carry in to the second adder, and the carry out of the second adder is the carry in to the third adder.

- What is the maximum operating frequency of the circuit?

Each full adder
- Propagation delays:
  - 20ps from Cin to Cout or to Sum(S)
  - 25ps from A or B to Cout
  - 30ps from A or B to S.
- Contamination delay
  - 15ps from Cin to either output
  - 22ps from A or B to either output.

Each flip-flop
- setup time of 30ps,
- hold time of 10ps,
- clock-to-Q propagation delay of 35ps
- clock-to-Q contamination delay of 21ps.

• What is the maximum operating frequency of the circuit?
We need to find longest path (critical path) between flip-flops

\[ t_{pcq} + t_{pd} + t_{setup} \leq T_c \]

Each full adder
- Propagation delays:
  - 20ps from Cin to Cout or to Sum(S)
  - 25ps from A or B to Cout
  - 30ps from A or B to S.
- Contamination delay
  - 15ps from Cin to either output
  - 22ps from A or B to either output.

Each flip-flop
- setup time of 30ps,
- hold time of 10ps,
- clock-to-Q propagation delay of 35ps
- clock-to-Q contamination delay of 21ps.
Maximum operating frequency (without clock skew)

We need to find longest path (critical path) between flip-flops

\[ t_{pcq} + t_{pd} + t_{setup} \leq T_c \]

\[ 35\text{ps} + (25\text{ps} + 20\text{ps} + 20\text{ps}) + 30\text{ps} \leq T_c \]

\[ T_c \geq 130\text{ps} \]

\[
\frac{1}{T_c} \leq \frac{1}{130\text{ps}}
\]

\[ f \leq 7.7\text{GHz} \]

Where

\[
\frac{1}{130 \times 10^{-12}\text{s}} = \frac{1}{130} \times 10^{12}\text{Hz}
\]

\[ = 7.7 \times 10^9\text{Hz} \]

\[ = 7.7\text{GHz} \]

Therefore

\[ f_{\text{max}} = 7.7\text{GHz} \]
Operating frequency with clock skew

How much clock skew can the circuit tolerate if it must operate at 8 GHz?
Operating frequency with clock skew

How much clock skew can the circuit tolerate if it must operate at 8 GHz?

\[ t_{pq} + t_{pd} + t_{setup} + t_{skew} \leq T_c \]

\[ 35\text{ps} + (25\text{ps} + 20\text{ps} + 20\text{ps}) + 30\text{ps} + t_{skew} \leq \frac{1}{8\text{GHz}} = 125\text{ps} \]

\[ t_{skew} \leq -5\text{ps} \]
Does this adder violate hold time constraint?

Each full adder
- Propagation delays:
  - 20ps from Cin to Cout or to Sum(S)
  - 25ps from A or B to Cout
  - 30ps from A or B to S.
- Contamination delay
  - 15ps from Cin to either output
  - 22ps from A or B to either output.

Each flip-flop
- setup time of 30ps,
- hold time of 10ps,
- clock-to-Q propagation delay of 35ps
- clock-to-Q contamination delay of 21ps.
Does this adder violates hold time constraint?

**NO**

\[ t_{ccq} + t_{cd} \geq t_{\text{hold}} \]

21ps + 15ps > 10ps

Each full adder

- **Propagation delays:**
  - 20ps from Cin to Cout or to Sum(S)
  - 25ps from A or B to Cout
  - 30ps from A or B to S.

- **Contamination delay**
  - 15ps from Cin to either output
  - 22ps from A or B to either output.

Each flip-flop

- setup time of 30ps,
- hold time of 10ps,
- clock-to-Q propagation delay of 35ps
- clock-to-Q contamination delay of 21ps.
Operating frequency with clock skew

How much clock skew can the circuit tolerate before it might experience a hold time violation?

\[ t_{ccq} + t_{cd} \geq t_{hold} \]

Where is \( t_{skew} \)?
Operating frequency with clock skew

How much clock skew can the circuit tolerate before it might experience a hold time violation?

\[
t_{ccq} + t_{cd} \geq t_{hold} + t_{skew}
\]

\[
21\text{ps} + 15\text{ps} \geq 10\text{ps} + t_{skew}
\]

\[
t_{skew} \leq 26\text{ps}
\]
Another Example

- Flip-flop: clock-to-Q maximum delay $t_{pcq} = 40\text{ps}$, clock-to-Q minimum delay $t_{ccq} = 30\text{ps}$, setup time $t_{setup} = 50\text{ps}$, hold time $t_{hold} = 60\text{ps}$
- Logic gate (each AND, OR, Inverter): propagation delay $t_{pd} = 35\text{ps}$, contamination delay $t_{cd} = 25\text{ps}$.

- Propagation delay analysis: How to find the longest path between flip flops
- Contamination delay analysis: How to find the shortest path between flip flops
- Understand the formulas
  - $t_{pcq} + t_{pd} + t_{setup} \leq T_c$ and $t_{pcq} + t_{pd} + t_{setup} + t_{skew} \leq T_c$
  - $t_{ccq} + t_{cd} \geq t_{hold}$ and $t_{ccq} + t_{cd} \geq t_{hold} + t_{skew}$
Another Example

- Flip-flop: clock-to-Q maximum delay $t_{pcq} = 40$ps, clock-to-Q minimum delay $t_{ccq} = 30$ps, setup time $t_{setup} = 50$ps, hold time $t_{hold} = 60$ps

- Logic gate (each AND, OR, Inverter): propagation delay $t_{pd} = 35$ps, contamination delay $t_{cd} = 25$ps.

Check

https://www.youtube.com/watch?v=JElzhGPYp68&feature=youtu.be
STANDARD COMBINATIONAL MODULES
Decoder and Multiplexer (MUX)

**Decoder**
- Check Homework 4’s Problem 4 and 6.1.
- Exercise’s Problem II

**Multiplexer (MUX)**
- Check Homework 4’s Problem 5, 6.2, and 6.2
- Exercise’s Problem III
- And
  - Multiplexers (I)
    https://www.youtube.com/watch?v=iCYNPkJ39H8
  - Multiplexers (II)
    https://www.youtube.com/watch?v=Swb3majVmFQ

**Decoder**

- **Definition:** A digital module that converts a binary address to the assertion of the addressed device
  - N inputs, $2^N$ outputs
  - One-hot outputs: only one output HIGH (1) at most

![Decoder Diagram](image)

- n to $2^n$ decoder function:
  - $y_i = 1$ if $E = 1$ & $(I_2, I_1, I_0) = i$
  - $y_i = 0$ otherwise
Definition: A digital module that converts a binary address to the assertion of the addressed device
- N inputs, $2^N$ outputs
- One-hot outputs: only one output HIGH (1) at most

Apply to implement Boolean function
- Decoder produces minterms when E=1.
- We can use an OR gate to collect the minterms to cover the On-set.
- For the Don’t Care-Set, we can just ignore the terms
Exercise Problem II: $f_3(a, b, c, d) = \Sigma m(12, 15)$
- Implement $f_3$ using 4-16 decoder and OR gates
Exercise Problem II: \( f_3(a, b, c, d) = \Sigma m(12, 15) \)
- Implement \( f_3 \) using 4-16 decoder and OR gates

Decoder Examples
Exercise Problem II: $f_3(a, b, c, d) = \Sigma m(12, 15)$
- Implement $f_3$ using 4-16 decoder and OR gates

Label them in exam!
Exercise Problem II: \( f_3(a, b, c, d) = \Sigma m(12, 15) \)
- Implement \( f_3 \) using 3-8 decoder and OR gates
Exercise Problem II: $f_3(a, b, c, d) = \Sigma m(12, 15)$
- Implement $f_3$ using 3-8 decoder and OR gates

Label them in exam!
Decoder Examples

- Exercise Problem II: $f_3(a, b, c, d) = \Sigma m(12, 15)$
  - Implement $f_3$ using 2-4 decoder and OR gates
Decoder Examples

- Exercise Problem II: \( f_3(a, b, c, d) = \Sigma m(12, 15) \)
  - Implement \( f_3 \) using 2-4 decoder and OR gates

Label them in exam!
- Selects one of N inputs to connect to the output.
- \( \log_2 N \)-bit select input – control input
Exercise Problem III: $f(a, b, c) = \Sigma m(0, 3, 5, 7) + \Sigma d(6)$,
- implement the function using a minimal network of 2:1 multiplexers.
Exercise Problem III: \( f(a, b, c) = \Sigma m(0, 3, 5, 7) + \Sigma d(6) \),

- implement the function using a minimal network of 2:1 multiplexers.

### Truth Table

<table>
<thead>
<tr>
<th>id</th>
<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>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise Problem III: \( f(a, b, c) = \Sigma m(0, 3, 5, 7) + \Sigma d(6) \),
- implement the function using a minimal network of 2:1 multiplexers.

<table>
<thead>
<tr>
<th>( bc = )</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a=1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ I_0 = D_0(b,c) = b'c' + bc \]

\[ I_1 = D_1(b,c) = c \]
Exercise Problem III: \( f(a, b, c) = \sum m(0, 3, 5, 7) + \sum d(6) \),
- implement the function using a minimal network of 2:1 multiplexers.

<table>
<thead>
<tr>
<th>( b )</th>
<th>( c=0 )</th>
<th>( c=1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise Problem III: \( f(a, b, c) = \Sigma m(0, 3, 5, 7) + \Sigma d(6), \)
- implement the function using a minimal network of 2:1 multiplexers.

\[
\begin{array}{c|c|c}
\text{b} & \text{c=0} & \text{c=1} \\
\hline
0 & 1 & 0 & \text{I0=D(c)=c'} \\
1 & 0 & 1 & \text{I1=D(c)=c} \\
\end{array}
\]
MUX example

Remember to label the ports, including enable bit, inputs, selectors(addresses) and outputs.
SYSTEM DESIGN
Exercise 6: Problem 4

- Last year’s exercise
  [http://cseweb.ucsd.edu/classes/fa14/cse140b/exams/exercise.pdf](http://cseweb.ucsd.edu/classes/fa14/cse140b/exams/exercise.pdf)

- Corrections for the problem IV

```plaintext
Alg(X,Y, start, U, done);
Input X[7:0], Y[7:0], start;
Output U[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];
S1: If start’ goto S1|| done <=1 ;
S2: A ← X || B ← Y || C ← (00000000) || done ← 0;
S3: A ← Add(A, B) || C ← Inc(C);
S4: If A'[7] goto S3;
S5: U ← C || done ← 1 || goto S1;
End Alg
```
System Design: Implement the following algorithm

Alg(X,Y, start, U, done);
Input X[7:0], Y[7:0], start;
Output U[7:0], done;
Local-object A[7:0], B[7:0], C[7:0];

S1: If start' goto S1|| done <=1;
S2: A ⇐ X || B ⇐ Y ||
    C ⇐ (00000000) || done ⇐ 0;
S3: A ⇐ Add(A, B) || C ⇐ Inc(C);
S4: If A'[7] goto S3;
S5: U ⇐ C || done ⇐ 1 || goto S1;

End Alg

*Inc(C) means C+1
### Data subsystem: use a table to list the instructions

Always write down these state numbers, they help us assign control signals for every state (during control subsystem design)

<table>
<thead>
<tr>
<th>State</th>
<th>Statement</th>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>A &lt;= X</td>
<td>A&lt;=Load(X)</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>B &lt;= Y</td>
<td>B&lt;=Load(Y)</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>C &lt;= 0</td>
<td>C&lt;=clear(C)</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>A &lt;= Add(A,B)</td>
<td>A&lt;=Add(A,B)</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>C &lt;= Inc(C)</td>
<td>C&lt;=Inc(C)</td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>U &lt;= C</td>
<td>Wires</td>
<td></td>
</tr>
</tbody>
</table>

S1: If start’ goto S1 || done <=1;
S2: A <= X || B <= Y || C <= (00000000) || done <= 0;
S3: A <= Add(A, B) || C <= Inc(C);
S4: If A'[7] goto S3;
S5: U <= C || done <= 1 || goto S1;
Data Subsystem Design: Storage & Function Components

<table>
<thead>
<tr>
<th>State</th>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>A&lt;=Load(X)</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>B&lt;=Load(Y)</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>C&lt;=clear(C)</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>A&lt;=Add(A,B)</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>C&lt;=Inc(C)</td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>Wires</td>
<td></td>
</tr>
</tbody>
</table>
Data Subsystem Design: Wiring and Map Clock Signals

<table>
<thead>
<tr>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>A &lt;= Load(X)</td>
</tr>
<tr>
<td>S2</td>
<td>B &lt;= Load(Y)</td>
</tr>
<tr>
<td>S2</td>
<td>C &lt;= clear(C)</td>
</tr>
<tr>
<td>S3</td>
<td>A &lt;= Add(A, B)</td>
</tr>
<tr>
<td>S3</td>
<td>C &lt;= Inc(C)</td>
</tr>
<tr>
<td>S5</td>
<td>Wires</td>
</tr>
</tbody>
</table>
Data Subsystem Design: Wiring and Map Clock Signals

<table>
<thead>
<tr>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;=Load(X)</td>
<td>C1=1, C0=1</td>
</tr>
<tr>
<td>B&lt;=Load(Y)</td>
<td>C2=1</td>
</tr>
<tr>
<td>C&lt;=clear(C)</td>
<td>C3=1</td>
</tr>
<tr>
<td>A&lt;=Add(A,B)</td>
<td>C0 = 0, C1=1</td>
</tr>
<tr>
<td>C&lt;=Inc(C)</td>
<td>C4=1</td>
</tr>
<tr>
<td>Wires</td>
<td>done</td>
</tr>
</tbody>
</table>
Data Subsystem Design: Wiring and Map Clock Signals

<table>
<thead>
<tr>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 A&lt;=Load(X)</td>
<td>C1=1, C0=1</td>
</tr>
<tr>
<td>S2 B&lt;=Load(Y)</td>
<td>C2=1</td>
</tr>
<tr>
<td>S2 C&lt;=clear(C)</td>
<td>C3=1</td>
</tr>
<tr>
<td>S3 A&lt;=Add(A,B)</td>
<td>C0=0, C1=1</td>
</tr>
<tr>
<td>S3 C&lt;=Inc(C)</td>
<td>C4=1</td>
</tr>
<tr>
<td>S5 Wires</td>
<td>done</td>
</tr>
</tbody>
</table>

2:1 MUX

Register A

Register B

Adder

Counter C

Data Subsystem

Control Subsystem
Data Subsystem Design: Identify Control Path Components

<table>
<thead>
<tr>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 A &lt;= Load(X)</td>
<td>C1=1, C0=1</td>
</tr>
<tr>
<td>S2 B &lt;= Load(Y)</td>
<td>C2=1</td>
</tr>
<tr>
<td>S2 C &lt;= clear(C)</td>
<td>C3=1</td>
</tr>
<tr>
<td>S3 A &lt;= Add(A,B)</td>
<td>C0 = 0, C1=1</td>
</tr>
<tr>
<td>S3 C &lt;= Inc(C)</td>
<td>C4=1</td>
</tr>
<tr>
<td>S5 Wires</td>
<td>done</td>
</tr>
</tbody>
</table>

2:1 MUX

Register A

Register B

Adder

Counter C

Control Subsystem
<table>
<thead>
<tr>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 A&lt;=Load(X)</td>
<td>C1=1, C0=1</td>
</tr>
<tr>
<td>S2 B&lt;=Load(Y)</td>
<td>C2=1</td>
</tr>
<tr>
<td>S2 C&lt;=clear(C)</td>
<td>C3=1</td>
</tr>
<tr>
<td>S2 A&lt;=Add(A,B)</td>
<td>C0 =0, C1=1</td>
</tr>
<tr>
<td>S3 C&lt;=Inc(C)</td>
<td>C4=1</td>
</tr>
<tr>
<td>S4 Wires</td>
<td>done</td>
</tr>
</tbody>
</table>
Control subsystem: State Diagram

S1: If start’ goto S1 || done <=1;
S2: A ← X || B ← Y || C ← (00000000) || done ← 0;
S3: A ← Add(A, B) || C ← Inc(C);
S4: If A’[7] goto S3;
S5: U ← C || done ← 1 || goto S1;
Control Subsystem: List control signals for every state

<table>
<thead>
<tr>
<th>Operations</th>
<th>Control signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 A&lt;=Load(X)</td>
<td>C0=1, C1=1</td>
</tr>
<tr>
<td>S2 B&lt;=Load(Y)</td>
<td>C2=1</td>
</tr>
<tr>
<td>S2 C&lt;=clear(C)</td>
<td>C3=1</td>
</tr>
<tr>
<td>S3 A&lt;=Add(A,B)</td>
<td>C0=0, C1=1</td>
</tr>
<tr>
<td>S3 C&lt;=Inc(C)</td>
<td>C4=1</td>
</tr>
<tr>
<td>S5 Wires</td>
<td>done</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C0 (MUX)</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Control Subsystem: List control signals for every state

Only when C1 is 1, we need to know what C0 is. Otherwise, Register A does not load anything from MUX.
Use a flip-flop to replace each state

Start

Start'

S1

S2

S3

S4

S5

A[7]

A'[7]
Control Subsystem: One-Hot State Machine

We may use a Demux to distribute the outward edges.
Control Subsystem: One-Hot State Machine

We use an OR gate to collect all inward edges.
Control Subsystem: One-Hot State Machine
Control Subsystem: One-Hot State Machine

Start’ → S1 → S2 → S3 → S4 → S5 → done

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

output the control signals

References

* YouTube video by Dao
  - Multiplexers (I) https://www.youtube.com/watch?v=iCYNPKVujH8
  - Multiplexers (II) https://www.youtube.com/watch?v=Swb3majVmFQ
  - Data subsystem https://www.youtube.com/watch?v=6TnT_GpYZAI
  - Control Subsystem: https://www.youtube.com/watch?v=mByGoCGESRY&feature=youtu.be
  - One-hot state machine: https://www.youtube.com/watch?v=bdUQRl1Ofa8

* Also read the corrections and FAQ @Piazza 189
  - https://piazza.com/class/hyd8lwowxmr4v6?cid=189