CSE140: Components and Design Techniques for Digital Systems

Tajana Simunic Rosing
Midterm Statistics for Sections A & B

mean: 86.48  
std: 12.80  
min: 40  
max: 100

Excellent job!!!

Regrade requests:
• Review graded midterm during this class
• Write your request on the 1st page of the midterm
• Turn it back in at the end of this class
• We may add or subtract points
• No other regrade requests will be honored!
Where we are now....

• What we covered last time:
  – Combinational circuits

• What we’ll do next
  – Finish multipliers, introduce dividers and ALU
  – Latches and FlipFlops (FFs)

• Upcoming deadlines:
  – ZyBook today: Sec 4.1-2, Thursday: **Sec 4.3-6 (Note change!)**
  – HW#4 assigned, due next Tuesday
  – Quiz #4 Thursday

• Textbook references:
  – chap 3, Sec 6.3
CSE140: Components and Design Techniques for Digital Systems

ALU components & ALUs

Tajana Simunic Rosing
Multiplication of positive binary numbers

- Generalized representation of multiplication by hand

\[
\begin{array}{cccc}
  a_3 & a_2 & a_1 & a_0 \\
  x & b_3 & b_2 & b_1 & b_0 \\
  \hline
  b_0a_3 & b_0a_2 & b_0a_1 & b_0a_0 \\
  b_1a_3 & b_1a_2 & b_1a_1 & b_1a_0 & 0 \\
  b_2a_3 & b_2a_2 & b_2a_1 & b_2a_0 & 0 & 0 \\
  + & b_3a_3 & b_3a_2 & b_3a_1 & b_3a_0 & 0 & 0 & 0 \\
  \hline
  p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
\]

For demo see:
http://courses.cs.vt.edu/~cs1104/BuildingBlocks/multiply.010.html
Multiplier – Array Style

- Multiplier design – array of AND gates
Division of positive binary numbers

• Repeated subtraction
  – Set quotient to 0
  – Repeat while dividend >= divisor
    • Subtract divisor from dividend
    • Add 1 to quotient
  – When dividend < divisor:
    • Reminder = dividend
    • Quotient is correct

Example:
• Dividend: 101; Divisor: 10

<table>
<thead>
<tr>
<th>Dividend</th>
<th>Quotient</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 101</td>
<td>0 +</td>
</tr>
<tr>
<td>- 2 10</td>
<td>1</td>
</tr>
<tr>
<td>3 11</td>
<td>1 +</td>
</tr>
<tr>
<td>- 2 10</td>
<td>1</td>
</tr>
<tr>
<td>1 101</td>
<td>2</td>
</tr>
</tbody>
</table>

For demo see: 
http://courses.cs.vt.edu/~cs1104/BuildingBlocks/Binary.Divide.html

Sources: TSR, Katz, Boriello & Vahid
**4 x 4 Divider**

\[ A/B = Q + R/B \]

**Algorithm:**
\[ R' = 0 \]
for \( i = N-1 \) to 0
\[ R = \{ R' \ll 1. A_i \} \]
\[ D = R - B \]
if \( D < 0 \), \( Q_i=0 \), \( R' = R \)
else \( Q_i=1 \), \( R' = D \)
\( R' = R \)
Arithmetic Logic Unit (ALU)

+ = odd

2's complement

Zero Extend

$F_{2:0}$ Function

<table>
<thead>
<tr>
<th>$F_{2:0}$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>A &amp; B</td>
</tr>
<tr>
<td>001</td>
<td>A or B</td>
</tr>
<tr>
<td>010</td>
<td>A + B</td>
</tr>
<tr>
<td>011</td>
<td>Not used</td>
</tr>
<tr>
<td>100</td>
<td>A &amp; ~B</td>
</tr>
<tr>
<td>101</td>
<td>A or ~B</td>
</tr>
<tr>
<td>110</td>
<td>A - B</td>
</tr>
<tr>
<td>111</td>
<td>A &lt; B</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
Another ALU Design Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Bin</th>
<th>Chop</th>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>01</td>
<td>00</td>
<td>AND</td>
<td>a*b</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>01</td>
<td>01</td>
<td>XOR</td>
<td>(a&amp;b)\oplus(b&amp;\overline{b})</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>10</td>
<td>+</td>
<td>a+b</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>11</td>
<td>2's comp</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>-</td>
<td>a-b</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>10</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td>-</td>
<td>-a+b</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td>a&lt;b</td>
<td></td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
CSE140: Components and Design Techniques for Digital Systems

Sequential Circuit Introduction
Latches and Flip-Flops

Tajana Simunic Rosing
What is a sequential circuit?

A circuit whose output depends on current inputs and past outputs

A circuit with memory

\[ y_i = f_i(S^t, X) \]
\[ s_{i+1} = g_i(S^t, X) \]
Why do we need circuits with ‘memory’?

- Circuits with memory can be used to store data
- Systems have circuits that run a sequence of tasks
Simplest memory element

(SRAM)

"remember"

"data"

"load"

load = 1

NMOS OK

"stored value"

2 trans
Flight attendant call button

- Flight attendant call button
  - Press call: light turns on
    - Stays on after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

- SR latch implementation
  - Call=1 : sets Q to 1 and keeps it at 1
  - Cancel=1 : resets Q to 0

Sources: TSR, Katz, Boriello & Vahid
- \( S = 1, R = 0 \): Set
  then \( Q = 1 \) and \( \overline{Q} = 0 \)

- \( S = 0, R = 1 \): Reset
  then \( Q = 1 \) and \( \overline{Q} = 0 \)

Sources: TSR, Katz, Boriello & Vahid
SR Latch Analysis

- $S = 0, R = 0$:  
  then $Q = Q_{\text{prev}}$
  - **Memory!**

- $S = 1, R = 1$:  
  then $Q = 0, \bar{Q} = 0$
  - **Invalid State**
  $\bar{Q} \neq \text{NOT } Q$

$Q_{\text{prev}} = 0$

$Q_{\text{prev}} = 1$