CPS 104
Computer Organization and Programming
Lecture-10: Digital Logic Design.

Sep. 27, 1999

Dietolf (Dee) Ramm
http://www.cs.duke.edu/~dr/cps104.html
Overview of Today’s Lecture:

- Review: Truth tables, Boolean functions, Gates and Circuits
- Digital circuit examples: 2-1 MUX, Full Adder
- The ALU
- Shift Unit
- Memory elements: Latch, Data-FlipFlop

Read Appendix B
### Boolean Functions and Gates (Cont.)

- **Examples: Boolean functions:** NOT, AND, OR, XOR, ... 

<table>
<thead>
<tr>
<th>a</th>
<th>NOT(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>AND(a, b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>OR(a, b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>XOR(a, b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>XNOR(a, b)</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>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>NOR(a, b)</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>
Boolean Functions and Gates (Cont.)

- Gates are electronics devices that implements simple Boolean functions:

- Examples:

  - AND \((a, b)\)
    - ![AND Gate Diagram](image)
  - OR \((a, b)\)
    - ![OR Gate Diagram](image)
  - NOT \((a)\)
    - ![NOT Gate Diagram](image)
  - XOR \((a, b)\)
    - ![XOR Gate Diagram](image)
  - NAND \((a, b)\)
    - ![NAND Gate Diagram](image)
  - NOR \((a, b)\)
    - ![NOR Gate Diagram](image)
  - XNOR \((a, b)\)
    - ![XNOR Gate Diagram](image)
Boolean Functions, Gates and Circuits

- **Circuits** are made from a network of gates. (function compositions).
- **Example:**

  \[
  F = \neg a \cdot b + \neg b \cdot a
  \]

  \[
  \begin{array}{cc|c}
  a & b & \text{XOR}(a, b) \\
  \hline
  0 & 0 & 0 \\
  0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0 \\
  \end{array}
  \]
Circuit Example: 2x1 MUX

\[ Y = (A \times S) + (B \times \neg S) \]
Example 4x1 MUX

```
Example 4x1 MUX
```

```
Example 4x1 MUX
```
Circuit Example: Selector

\[\begin{array}{c|c|c|c|c}
I_1 & I_0 & Q_0 & Q_1 & Q_2 & Q_3 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 \\
\end{array}\]
**Full Adder**

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{Cin} & \text{Sum} \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{Cin} & \text{Cout} \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
01101100 \\
+00101100 \\
\hline
10011001 \\
\end{array}
\]
Example: 4-bit adder

```
Example: 4-bit adder

\[
\begin{array}{cccc}
a_0 & a_1 & a_2 & a_3 \\
\end{array}
\quad
\begin{array}{cccc}
b_0 & b_1 & b_2 & b_3 \\
\end{array}
\quad
\begin{array}{cccc}
S_0 & S_1 & S_2 & S_3 \\
\end{array}
\quad
\begin{array}{cccc}
C_{out} &  \\
\end{array}
\end{array}
\]
Example: Adder/Subtractor

Add/Sub

C_{out} → Full Adder → Full Adder → Full Adder → Full Adder → C_{out}

b3 a3 b2 a2 b1 a1 b0 a0

S0 S1 S2 S3
Overflow

Example 1:

\[
\begin{align*}
0100000_2 & \quad (= 53_{10}) \\
0110101_2 & \quad (= 42_{10}) \\
+0101010_2 & \quad (= -33_{10}) \\
\hline
1011111_2 & \quad (= -33_{10})
\end{align*}
\]

Example 2:

\[
\begin{align*}
1000000_2 & \quad (= -43_{10}) \\
1010101_2 & \quad (= -54_{10}) \\
+1001010_2 & \quad (= 31_{10}) \\
\hline
0011111_2 & \quad (= 31_{10})
\end{align*}
\]

Example 3:

\[
\begin{align*}
1100000_2 & \quad (= 53_{10}) \\
0110101_2 & \quad (= -22_{10}) \\
+1101010_2 & \quad (= 31_{10}) \\
\hline
0011111_2 & \quad (= 31_{10})
\end{align*}
\]

Example 4:

\[
\begin{align*}
0000000_2 & \quad (= 21_{10}) \\
0010101_2 & \quad (= 42_{10}) \\
+0101010_2 & \quad (= 63_{10}) \\
\hline
0111111_2 & \quad (= 63_{10})
\end{align*}
\]
Add/Subtract With Overflow detection

Add/Sub

Overflow

Full Adder

Full Adder

Full Adder

Full Adder

S_{n-1}  

S_{n-2}  

S_1  

S_0  

b_{n-1}  

a_{n-1}  

b_{n-2}  

a_{n-2}  

b_1  

a_1  

b_0  

a_0
ALU Slice

<table>
<thead>
<tr>
<th>A</th>
<th>F</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>a + b</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>a - b</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>NOT b</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>a OR b</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>a AND b</td>
</tr>
</tbody>
</table>
The ALU

Overflow

= Zero

Overflow

= Zero

Q_{n-1} Q_{n-2} Q_1 Q_0

b_{n-1} a_{n-1} b_{n-2} a_{n-2} b_1 a_1 b_0 a_0

ALU control

ALU Slice

ALU Slice

ALU Slice

ALU Slice

©GK&DR fall 1999
Memory Elements

- All the circuit we looked at so far are **combinational circuits**: the output is a Boolean function of the inputs.
- We need circuits that can remember values. (registers)
- The output of the circuit is a function of the input AND a function of a stored values (state).
- Circuits with memory are called **sequential circuits**.
Rest-Set Latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>–</td>
</tr>
</tbody>
</table>
Rest-Set Latch (cont.)
Rest-Set Latch (cont.)

CPS104  Lec10.20  ©GK&DR fall 1999
Positive Edge Data-Latch

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>Q</td>
</tr>
</tbody>
</table>
Negative Edge D-Latch

<table>
<thead>
<tr>
<th>D</th>
<th>E</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>Q</td>
</tr>
</tbody>
</table>
Master-Slave Data-Flip-Flop

- On \( C \) down, \( D \) is transferred to the master stage and the slave is is stable.

- On \( C \) up, the Master stage is transferred into the slave stage (output), and the master stage is stable.
DFF Timing

Data Clock

M

Q

©GK&DR fall 1999

CPS104 Lec10.24
The Tri-State driver is like a (one directional) switch:
- When the Enable is on (E=1) it transfers the input to the output.
- When the Enable is off (E=0) it disconnects the output.

Z :- High Impedance
Bus Connections

- The Bus: Many to many connections.
- Mutual exclusion: At most one Enable is on!
Register Cells on a bus

One can “source” and “sink” from any cell on the bus by activating the right controls (WE and RE).
3-Port Register Cell

Data-In

Bus-C

Bus-B

Bus-A

DinEnable

OutA

OutB

©GK&DR fall 1999
3-Port Register

EC  EA  EB

Bus-C
Bus-B

Bit-2

Bus-C
Bus-B

Bit-1

Bus-C
Bus-B

Bit-0

Bus-C
Bus-B

Bus-A

Bus-A

Bus-A

Bit-0

Bit-1

Bit-2
Address Decode circuit

Register address: 01
Summary

- So far we saw how to take a Boolean function and generate a circuit that “realize” the function.
- We learned to construct circuits that can add and subtract.
- We learned about the ALU: a circuit that can add, subtract, detect overflow, compare, and do bit-wise operations (AND, OR, NOT)
- Saw how to construct a shifter circuit.
- Learned about the memory elements: RS-Latch, D-Latches and D-Flip-flops.
- Learned about Tri-State drivers and BUS Communication. (many-to many)
- Learned about how to construct a register file.
- Saw how control signals can modify what the circuit will do with inputs.
  - Examples: ALU, Shift, Register read-write, ...