CS61CL Machine Structures

Lec 8 – State and Register Transfers

David Culler
Electrical Engineering and Computer Sciences
University of California, Berkeley
Review: Combinational Logic

• Any boolean function can be expressed as an acyclic connection of gates

• Often specified by a truth table

• Outputs are purely a function of the inputs
  – no history, no state
Examples: Logical Operations

\[ C = A \& B \]

A \rightarrow B \rightarrow C

\[ A_{31:0} \quad B_{31:0} \rightarrow C_{31:0} \]

A_{31:0} \quad B_{31:0} \rightarrow C_{31:0}

A_{31:0} \quad B_{31:0} \rightarrow C_{31:0}

A_{31:0} \rightarrow C_{31:0}
Example: Multiplexor

\[ C = S \oplus A : B \]

\[ C = (S \land A) \lor (\neg S \land B) \]
Example: Adder

<table>
<thead>
<tr>
<th>Ci</th>
<th>A</th>
<th>B</th>
<th>Co</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A B Ci

Co S

A B Ci

Co S

A B Ci

Co S

A B Ci

Co S

A B Ci

Co S

A31:0

B31:0

C31:0
Example: Arithmetic Logic Unit

A_{31:0} \quad B_{31:0}

S_{1:0}

C_{31:0}
Element of Time

- Logical change is not instantaneous
- Broader digital design methodology has to make it appear as such
  - Clocking, delay estimation, glitch avoidance
What makes Digital Systems tick?

Combinational Logic

clk

time
Administrative Issues

• HW 6 due tonight
• Project 2 dues Monday 10/26
  – bimodal check-off
  – testing tools available tomorrow
  – they are really picky
• Project 1 grading almost done
  – Friday
• HW 7 – discuss

• Midterm 2 on 11/9 as in original schedule
  – 11/11 is holiday
A Bit of state: D-type edge-triggered flip-flop

- The edge of the clock is used to sample the "D" input & send it to "Q" (positive edge triggering).
  - At all other times the output Q is independent of the input D (just stores previously sampled value).
  - The input must be stable for a short time before the clock edge.

\[ \begin{array}{c}
\text{clk} \\
\text{D} \\
\text{clk} \\
\text{Q}
\end{array} \]
Registers

- **Collections of flip-flops with similar controls and logic**
  - Stored values somehow related (e.g., form binary value)
  - Share clock, reset, and set lines
  - Similar logic at each stage
What “registers” do we need?

- “read” vs use the output
- “write” on the clock edge => Load
- Load Control
Register with Load Control

\[ R_{31} \rightarrow R_0 \]
Register File

Din
Dsel
D
Rs
R0
R1
R2
R31

Asel
B.sel

Bout
Aout
Towards a Data Path

Asel
Bsel
Dsel
Id

aluOP
Exercise a Data Path
What about RAM - Randomly Accessible Memory?

• Like a HUGE register file
  – dense, slower, low-cost storage cell (6T)
  – fewer ports
  – wider address lines
  – accessed over a “bus”

• Bus: means of composition in hardware system
  – logically related collection of wires
  – interfacing one or more sources to one or more destinations
Recall: Instruction Cycle

Instruction Fetch
Decode
Operand
Execute
Result
Next

PC 0B24

32 2 3 1

"add $1,$2,$3"
# Register Transfers

## MIPS Instruction Format

### R: Reg-Reg

<table>
<thead>
<tr>
<th>Field</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>6</td>
</tr>
<tr>
<td>rs</td>
<td>5</td>
</tr>
<tr>
<td>rt</td>
<td>5</td>
</tr>
<tr>
<td>rd</td>
<td>5</td>
</tr>
<tr>
<td>shamt</td>
<td>5</td>
</tr>
<tr>
<td>funct</td>
<td>6</td>
</tr>
</tbody>
</table>

### I: Reg-Imed

<table>
<thead>
<tr>
<th>Field</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>6</td>
</tr>
<tr>
<td>rs</td>
<td>5</td>
</tr>
<tr>
<td>rt</td>
<td>5</td>
</tr>
<tr>
<td>immediate</td>
<td>16</td>
</tr>
</tbody>
</table>

### J: Jump

<table>
<thead>
<tr>
<th>Field</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>6</td>
</tr>
<tr>
<td>immediate</td>
<td>26</td>
</tr>
</tbody>
</table>

### Reg-Reg instructions (op == 0)
- add, sub, and, or, nor, xor, slt
  \[ R[rd] := R[rs] \text{ funct } R[rt]; \text{ pc:=pc+4} \]
- sll, srl, sra
  \[ R[rd] := R[rt] \text{ shift } \text{ shamt} \]

### Reg-Immed (op != 0)
- addi, andi, ori, xori, lui
  \[ R[rt] := R[rs] \text{ op Im16} \]
- addiu, sli, sliu
- lw, lh, lhu, lb, lbu
- sw, sh, sb
  \[ R[rt] := \text{Mem}[ \text{R[rs]} + \text{signEx(Im16)} ] \]
  \[ \text{Mem}[ \text{R[rs]} + \text{signEx(Im16)} ] := R[rt] \]
Synchronous Circuit Design

- **Combinational Logic Blocks (CL)**
  - Acyclic
  - no internal state (no feedback)
  - output only a function of inputs

- **Registers (reg)**
  - collections of flip-flops

- **clock**
  - distributed to all flip-flops

- **ALL CYCLES GO THROUGH A REG!**