SEQLAB AND HCL

CS429H - Spring 2011
Christian Miller
We give you a bigger, better Y86 simulator
You modify it to include two new instructions
  iaddl and leave
You do this by modifying the HCL description
HCL

- A toy hardware description language
  - Fake, actually compiles into a C program
- Looks a lot like C
- Does not execute sequentially, but simultaneously
  - Think logic gates, not assembly
  - Do not create loops!
Data

• Only two types: `bool` (a single bit) and `int` (32 bits)

• `boolsig` and `intsig` tie into the simulator
  • That means don’t edit them

• `Equals` doesn’t assign, it renames
  • Basically attaches names to wires
## Expression Syntax

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Logic value 0</td>
</tr>
<tr>
<td>1</td>
<td>Logic value 1</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Named Boolean signal</td>
</tr>
<tr>
<td><code>int-expr in {int-expr_1, int-expr_2, ..., int-expr_k}</code></td>
<td>Set membership test</td>
</tr>
<tr>
<td><code>int-expr_1 == int-expr_2</code></td>
<td>Equality test</td>
</tr>
<tr>
<td><code>int-expr_1 != int-expr_2</code></td>
<td>Not equal test</td>
</tr>
<tr>
<td><code>int-expr_1 &lt; int-expr_2</code></td>
<td>Less than test</td>
</tr>
<tr>
<td><code>int-expr_1 &lt;= int-expr_2</code></td>
<td>Less than or equal test</td>
</tr>
<tr>
<td><code>int-expr_1 &gt; int-expr_2</code></td>
<td>Greater than test</td>
</tr>
<tr>
<td><code>int-expr_1 &gt;= int-expr_2</code></td>
<td>Greater than or equal test</td>
</tr>
<tr>
<td><code>! bool-expr</code></td>
<td>NOT</td>
</tr>
<tr>
<td><code>bool-expr_1 &amp;&amp; bool-expr_2</code></td>
<td>AND</td>
</tr>
<tr>
<td>`bool-expr_1</td>
<td></td>
</tr>
</tbody>
</table>
Expression semantics

- Can be nested using parentheses
- Set membership returns true if something is in a given set
- Expressions basically compile into logic tables
- No partial evaluation, everything is evaluated
Case syntax

```
[  
  bool-expr_1 : int-expr_1  
  bool-expr_2 : int-expr_2  
  . . . 
  bool-expr_k : int-expr_k  
]
```
Case semantics

- Think switch statement
- Effectively compiles into a mux (output selector)
- Internally wrangled to evaluate in order
  - You can throw in “1 :” as a default at the end
Go forth

- You have an embarrassingly long time to do this.
- It can be done in 15 lines... easily.