Introduction to FHDL
A Sample Circuit

A sample circuit diagram with labels for a, b, ax, bx, gi1, gi2, gq, i1, i2, and q.
Equivalent FHDL

Test1: circuit
inputs a,b
outputs q

gq: or (i1,i2),q
gi2: and (a,b),i2
gi1: and (ax,bx),i1
gax: not a,ax
gbx: not b,bx
endcircuit
Statements have the form:
\(<\text{NAME}\>\) : \(<\text{OP-CODE}\>\) \(<\text{OPERANDS}\>\)
- \(<\text{OP-CODE}\>\) must be surrounded by spaces
- \(<\text{NAME}\>\): is sometimes optional

Circuits must begin with:
\(<\text{NAME}\>\): circuit

Circuits must end with:
endcircuit
Coding Rules II

- Primary Inputs must be declared using the statement:
  
  \[
  \text{inputs} \ <\NAME 1>, <\NAME 2>, \ldots
  \]

- Primary Outputs must be declared using the statement:
  
  \[
  \text{outputs} \ <\NAME 1>, <\NAME 2>, \ldots
  \]
The Remainder of the Circuit is Gate Specifications of the form:

\(<\text{Name}>\) : \(<\text{GateType}>\) (\(<\text{Inputs}>\), (\(<\text{Outputs}>\))

- If \(<\text{Inputs}>\) or \(<\text{Outputs}>\) is a single name, Omit the Parenthesis

- Nets are declared implicitly

- The “wire” statement can alter net properties
**Circuit Must Be Parsed**

- Internal Tables are Constructed from Specifications
- Circuit-Processing programs use Parsed form of Circuit
- Many Different Structuring Techniques can be Used
### Possible Result of Parsing Gates

#### Gate Table

<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>Type</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>gq</td>
<td>or</td>
<td>3,4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>gi2</td>
<td>and</td>
<td>0,1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>gi1</td>
<td>and</td>
<td>5,6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>gaprime</td>
<td>not</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>gbprime</td>
<td>not</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
## Possible Result of Parsing Nets

### Net Table

<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>Type</th>
<th>Fanout</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td>PI</td>
<td>1,3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>PI</td>
<td>1,4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>q</td>
<td>PO</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>i1</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>i2</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>aprime</td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>bprime</td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
WINFHDL Style

Linked Data Structures
Access to Structures

- Circuit Structure
- Net List
- Gate List
- Primary Inputs
- Primary Outputs
Introduction to FHDL
A Sample Circuit
Equivalent FHDL

Test1: circuit
  inputs a, b
  outputs q
  gq: or (i1, i2), q
  gi2: and (a, b), i2
  gi1: and (ax, bx), i1
  gax: not a, ax
  gbx: not b, bx
  endcircuit
Coding Rules

- Statements have the form:
  \(<\text{NAME}\> : \ <\text{OP-CODE}\> \ <\text{OPERANDS}\>
  - \(<\text{OP-CODE}\>\) must be surrounded by spaces
  - \(<\text{NAME}\>:\) is sometimes optional

- Circuits must begin with:
  \(<\text{NAME}\>: \text{ circuit}\)

- Circuits must end with:
  \(\text{endcircuit}\)
Primary Inputs must be declared using the statement:

```
inputs <NAME 1>,<NAME 2>, ...
```

Primary Outputs must be declared using the statement:

```
outputs <NAME 1>,<NAME 2>, ...
```
Coding Rules III

◆ The Remainder of the Circuit is Gate Specifications of the form:
  <Name>: <GateType> (<Inputs>),(<Outputs>)
◆ If <Inputs> or <Outputs> is a single name, Omit the Parenthesis
◆ Nets are declared implicitly
◆ The “wire” statement can alter net properties
Circuit Must Be Parsed

- Internal Tables are Constructed from Specifications
- Circuit-Processing programs use Parsed form of Circuit
- Many Different Structuring Techniques can be Used
# Possible Result of Parsing Gates

## Gate Table

<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>Type</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>gq</td>
<td>or</td>
<td>3,4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>gi2</td>
<td>and</td>
<td>0,1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>gi1</td>
<td>and</td>
<td>5,6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>gaprime</td>
<td>not</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>gbprime</td>
<td>not</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
### Possible Result of Parsing Nets

<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>Type</th>
<th>Fanout</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td>PI</td>
<td>1,3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>PI</td>
<td>1,4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>q</td>
<td>PO</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>i1</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>i2</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>aprime</td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>bprime</td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
WINFHDL Style

Linked Data Structures
Access to Structures

Circuit Structure

Net List

Gate List

Primary Inputs

Primary Outputs