Datapath Components

Adders

- Adds two N-bit binary numbers
  - 2-bit adder: adds two 2-bit numbers, outputs 3-bit result
  - e.g., $01 + 11 = 100$ \( (1 + 3 = 4) \)
  - Can design using combinational logic design process, but doesn’t work well for reasonable-size N
    - *Why not?*

Truth table too big

- 2-bit adder’s truth table shown
  - Has \( 2^{2+1} = 16 \) rows
- 8-bit adder: \( 2^{8+8} = 65,536 \) rows
- 16-bit adder: \( 2^{16+16} = 4 \) billion rows
- 32-bit adder: ...
- Big truth table with numerous 1s/0s yields big logic
  - Plot shows number of transistors for N-bit adders, using state-of-the-art automated combinational design tool
  - *How many of transistors are needed for a 16-bit adder?*
Datapath Components

Alternative Adder Design Method
- Mimic how people do addition by hand
- One column at a time
  - Compute sum, add carry to next column

Alternative Method: Imitate Adding by Hand

Create component for each column
- Adds that column’s bits, generates sum and carry bits

Datapath Components

Half-Adder

- Half-adder: Adds 2 bits, generates sum and carry
- Design using combinational design process from Ch 2

Step 1: Capture the function

Step 2: Convert to equations

Step 3: Create the circuit

Full-Adder

- Full-adder: Adds 3 bits, generates sum and carry
- Design using combinational design process from Ch 2

Step 1: Capture the function

Step 2: Convert to equations

Step 3: Create the circuit
Datapath Components

Carry-Ripple Adder

- Using half-adder and full-adders, we can build adder that adds like we would by hand
- Called a carry-ripple adder
  - 4-bit adder shown: Adds two 4-bit numbers, generates 5-bit output
    - 5-bit output can be considered 4-bit "sum" plus 1-bit "carry out"
  - Can easily build any size adder

```
\[
\begin{align*}
\text{a0} & \quad \text{b0} \\
\text{co} & \quad \text{co} \\
\text{s0} & \quad \text{s0} \\
\text{c0} & \quad \text{c0} \\
\end{align*}
\]
```

4-bit adder

```
\[
\begin{align*}
\text{a3} & \quad \text{a2} & \quad \text{a1} & \quad \text{a0} \\
\text{b3} & \quad \text{b2} & \quad \text{b1} & \quad \text{b0} \\
\text{s3} & \quad \text{s2} & \quad \text{s1} & \quad \text{s0} \\
\text{co} & \quad \text{co} & \quad \text{co} & \quad \text{co} \\
\end{align*}
\]
```

Datapath Components

Carry-Ripple Adder’s Behavior

- Assume all inputs initially 0
- 0111+0001 (answer should be 01000)
- Output after 2 ns (1 FA delay)

```
\[
\begin{align*}
\text{a0} & \quad \text{b0} & \quad \text{co} & \quad \text{s0} \\
\text{c0} & \quad \text{c0} & \quad \text{c0} & \quad \text{c0} \\
\end{align*}
\]
```

Correct answer appears after 4 delays
Datapath Components

Cascading Adders

- 4-bit adder
- 8-bit adder

Incrementer

- Counter design used incrementer
- Incrementer design
  - Could use carry-ripple adder with B input set to 00...001
  - But when adding 00...001 to another number, the leading 0's obviously don't need to be considered -- so just two bits being added per column
  - Use half-adders (adds two bits) rather than full-adders (adds three bits)

Carries:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Unused:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) 4-bit adder

(b) 8-bit adder

(a) Incrementer

(b) Incrementer