Registers

Flip-flops are the basic building blocks for registers
Parallel-load register (32-bit) has 2 operations
  c (control) = 0: hold
  c (control) = 1: load \( z_{31-0} = x_{31-0} \)
Block diagram

Don’t forget the clock!

Example: 4 bits, using D flip-flops
Registers

Need to choose inputs: hold or parallel load
Use 2-1 MUX
Hold: need to keep value constant
  D flip-flop sets Q to value of D
  feed Q back to 0 input of MUX
Parallel load: set flip-flop value to input
  feed input x to 1 input of MUX
Also need clock and control input
(Note that control is shown going through each MUX)
Registers: T flip-flops

Register with T flip-flops: for simplicity, implement with 2 flip-flops

How does T flip-flop hold the value?
\[ T = 0 \]

How does it load?
\[ T = 1 \] toggles
If \((x \neq Q)\), then toggle

MUX input 1:
\[ T = x \text{ XOR } Q \]

What other operations could we do?
Shift is simply another form of load!
Input is from current state, rather than external
Shift left: load bit \(i\) with value of bit \((i-1)\)
Shift right: load bit \(i\) with value of bit \((i+1)\)
Can add more control bits to select shift operations as well as regular load
Change MUX to 4-1:

<table>
<thead>
<tr>
<th>control</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>hold</td>
</tr>
<tr>
<td>01</td>
<td>parallel load</td>
</tr>
<tr>
<td>10</td>
<td>shift left logical 1 bit</td>
</tr>
<tr>
<td>11</td>
<td>shift right logical 1 bit</td>
</tr>
</tbody>
</table>