Announcements

- Midterm next Thurs. 6-8pm sharp, 141 McCone

- Elad will hold a review session next Mon. evening at 7pm in 540A/B Cory

- No lecture next Thurs.
Counters

- Special sequential circuits (FSMs) that repeatedly sequence through a set of outputs.
- Examples:
  - Binary counter: 000, 001, 010, 011, 100, 101, 110, 111, 000,
  - Gray code counter: 000, 010, 110, 100, 101, 111, 011, 001, 000, 010, 110, ...
  - One-hot counter: 0001, 0010, 0100, 1000, 0001, 0010, ...
  - BCD counter: 0000, 0001, 0010, ..., 1001, 0000, 0001
  - Pseudo-random sequence generators: 10, 01, 00, 11, 10, 01, 00, ...
- Moore machines with “ring” structure in State Transition Diagram:

What Are They Used For? (1)

- Counters are commonly used in hardware designs because most (if not all) computations that we put into hardware include iteration (looping). Examples:
  - Shift-and-add multiplication scheme.
  - Bit serial communication circuits (must count one word’s worth of serial bits.)
- Other uses for counter:
  - Clock divider circuits
    \[ \frac{16\text{MHz}}{1/64} \]
    - Systematic inspection of data-structures
      - Example: Network packet parser/filter control.
What Are They Used For? (2)

- Counters simplify “controller” design by:
  - providing a specific number of cycles of action,
  - sometimes used with a decoder to generate a sequence of timed control signals.

Example Circuit using Counters for Control

- Bit-serial multiplier:
Example Circuit using Counters for Control

- Bit-serial multiplier \([n^2 \text{ cycles}, \text{one bit of result finalized every } n \text{ cycles}]:\)

- Control Algorithm:

```
repeat n cycles { // outer (i) loop
    repeat n cycles{ // inner (j) loop
        shiftA, selectSum, shiftHi
    } shiftB, shiftHi, shiftLOW, reset
```

Note: The occurrence of a control signal \(x\) means \(x=1\). The absence of \(x\) means \(x=0\).

Controller Using Counters

- State Transition Diagram:
  - Assume presence of two binary counters. An "i" counter for the outer loop and "j" counter for inner loop.

TC is asserted when the counter reaches its maximum count value. CE is "count enable". The counter increments its value on the rising edge of the clock if CE is asserted.
Controller using Counters

- Controller circuit implementation:

  ![Diagram of controller circuit]

- Outputs:

  - $C_{E_i} = q_2$
  - $C_{E_j} = q_1$
  - $R_{ST_i} = q_0$
  - $R_{ST_j} = q_2$
  - $shiftA = q_1$
  - $shiftB = q_2$
  - $shiftLOW = q_2$
  - $shiftHI = q_1 + q_2$
  - $reset = q_2$
  - $selectSUM = q_1$

How Do We Design Counters?

- For binary counters [most common case] incrementer circuit would work:

  ![Diagram of incrementer circuit]

  - In Verilog, a counter is specified as: $x = x + 1$;
    - This does not imply a full-blown adder
    - An incrementer is simpler than an adder
    - And a counter can be simpler yet.

- Can often use special optimizations for counters. But, if in doubt, think of a counter as an FSM and follow general procedure.
Synchronous Counters

All outputs change with clock edge.

• Binary Counter Design:
  Start with 3-bit version and generalize:

<table>
<thead>
<tr>
<th>c</th>
<th>b</th>
<th>a</th>
<th>c'</th>
<th>b'</th>
<th>a'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

a' = a'

b' = a ⊕ b

cb

a' 00 01 11 10

b' 0 0 1 1

c 1 0 1 0

c' = a'c + abc' + b'c
= c'(a' + b') + c'(ab)
= c'(ab)' + c'(ab)
= c ⊕ ab

Synchronous Counter

• How do we extend to n bits?
  - Extrapolate: c*: d* = d ⊕ abc, e* = e ⊕ abcd

• Good news: can be reasonably fast
  - Critical path is n-bit AND + XOR

• Bad news: hardware grows somewhat quickly with n
  - AND gate size grows with n
**Alternative Counter Design**

- CE is "count enable", allows external control of counting,
- TC is "terminal count", is asserted on highest value, allows cascading, external sensing of occurrence of max value.

- Good news: hardware simply linear with n
- Bad news: delay also linear with n...

---

**Up-Down Counter**

- c b a c' b' a'
- 0 0 1 0 0 0
- 0 1 0 0 0 1
- 0 1 1 0 1 0
- 1 0 0 1 1
- 1 0 1 1 0
- 1 1 0 1 0 1
- 1 1 1 1 0

*Down-count*
Odd Counts

- Extra combinational logic can be added to terminate count before max value is reached:
- Example: count to 12

  ![4-bit binary counter diagram]

  = 11?

- Alternative:

Ring Counters

- “one-hot” counters
  0001, 0010, 0100, 1000, 0001,
  ...

  ![Ring counter diagram]

  “Self-starting” version:

  ![Self-starting ring counter diagram]
Johnson Counter

Each stage is 2 of previous.

Look at output waveforms:

Forbidden in Synchronous Design

Asynchronous “Ripple” Counters

Each stage is 2 of previous.

Look at output waveforms:

Often called “asynchronous” counters.

A “T” flip-flop is a “toggle” flip-flop. Flips its state on cycles when T=1.