Counters

• Special sequential circuits (FSMs) that sequence though a set outputs.
• Examples:
  – binary counter: 000, 001, 010, 011, 100, 101, 110, 111, 000, 001, ...
  – gray code counter: 000, 010, 100, 101, 110, 111, 011, 001, 000, 010, 110, ...
  – one-hot counter: 0001, 0010, 0100, 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 to STD:

What are they used?
• Examples from this semester:
  – Clock divider circuits
  – Network packet parser/filter control.
  – Bit-serial multiplier control circuitry (from HW)
  – In general: counters simplify controller design by
    • providing a specific number of cycles of action,
    • sometimes used in with a decoder to generate a sequence of control
      signals.

Controller using Counters
• Bit-serial multiplier:

    A register
    shiftA

    B register
    shiftB

    HI register
    shiftHI

    LOW register
    shiftLOW

• 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 counters. An "i" counter for the outer loop and "j" counter for inner loop.

  ![State Transition Diagram](image)

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

Controller using Counters

- **Controller circuit implementation:**
  - Outputs:
    - \( CE_i = q_2 \)
    - \( CE_j = q_1 \)
    - \( RST_i = q_0 \)
    - \( RST_j = q_2 \)
    - \( shiftA = q_1 \)
    - \( shiftB = q_2 \)
    - \( shiftLOW = q_2 \)
    - \( shiftHI = q_1 + q_2 \)
    - \( reset = q_2 \)
    - \( selectSUM = q_1 \)

How do we design counters?

- For binary counters (most common case) incrementer circuit would work:

  ![Incrementer Circuit](image)

  - In Verilog, a counter is specified as: \( x = x + 1 \);
    - This does not imply an adder
    - An incrementer is simpler than an adder
    - And a counter is simpler yet.

  - In general, the best way to understand counter design is to think of them as FSMs, and follow general procedure. But before that ...

“Ripple” counters

- Each stage is \( \cdot 2 \) of previous.
- Look at output waveforms:

  ![Ripple Counters Waveforms](image)

  - Often called “asynchronous” counters.
  - A "T" flip-flop is a "toggle" flip-flop. Flips it state on cycles when \( T = 1 \).
Synchronous Counters

All outputs change with clock edge.

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

  \[
  \begin{array}{c|c|c|c}
  a & b & c & a' + b' + c' \\
  \hline
  0 & 0 & 0 & 0' \\
  0 & 0 & 1 & 1 \\
  0 & 1 & 0 & 0 \\
  0 & 1 & 1 & 1 \\
  1 & 0 & 0 & 0 \\
  1 & 0 & 1 & 1 \\
  1 & 1 & 0 & 1 \\
  1 & 1 & 1 & 0 \\
  \end{array}
  \]

  \[
  \begin{align*}
  a' &= a' \\
  b' &= a' + b \\
  c' &= a'c + abc' + b'c \\
  &= a'c + (a'b')c + (c'ab) \\
  &= a'c + c'ab \\
  &= c(ab)' + c'(ab) \\
  &= c(ab)'
  \end{align*}
  \]

- How do we extend to n-bits?
  - Extrapolate \( c' \): \( d' = d' \oplus abc, e' = e' \oplus abcd \)
  - Has difficulty scaling (AND gate inputs grow with n)
  - 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.

Synchronous Counters

- How does this one scale?
  - Generation of TC signals very similar to generation of carry signals in adder.
  - "Parallel Prefix" circuit reduces delay:

Up-Down Counter

- Generation of TC signals very similar to generation of carry signals in adder.
- "Parallel Prefix" circuit reduces delay:
Odd Counts

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

```
reset 4-bit binary counter
        = 11 ?
```

Alternative:

```
load 4-bit binary counter TC
```

Ring Counters

- "one-hot" counters 0001, 0010, 0100, 1000, 0001, ...
- What are these good for?

```
reset 4-bit binary counter load 4-bit binary counter
```
}

Ring Counters

- "Self-starting" version:

```
reset
```

Johnson Counter

(a) Four-stage switch-to-ring counter

```
CLK
```

Sequence number Flip-flop outputs AND gate required for output

<table>
<thead>
<tr>
<th>Number</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Count sequence and required decoding

```
Fig. 6-17 Generation of Timing Signals
```

```
Fig. 6-18 Construction of a Johnson Counter
```
Register Summary

- All registers (this semester) based on Flip-flops:
  - Load-enable is a popular option:

Xilinx flip-flops employ a clock enable (CE) for same purpose.

Shift-_registers

- Parallel load shift register:
  - "Parallel-to-serial converter"
  - Also, works as "Serial-to-parallel converter", if q values are connected out.
  - Also get used as controllers (ala "ring counters")