EECS150 - Digital Design
Lecture 17 - Sequential Circuits 3
(Counters)

March 19&21, 2002
John Wawrzynek
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, 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 to STD:

```
S3 --S0-- S1
    |     |
  ---S2---
```
What are they used?

- Examples from this semester:
  - Clock divider for UART circuit
    \[
    \begin{array}{c}
    \text{16MHz} \\
    \text{\div 64}
    \end{array}
    \]
  - UART control (count to 8 between bit capture).
  - Bit-serial multiplier control circuitry (from HW and quiz)

- 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:

- Control Algorithm:

```plaintext
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 assured with 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:

\[
\begin{align*}
\text{CE}_i &= q_2 \\
\text{CE}_j &= q_1 \\
\text{RST}_i &= q_0 \\
\text{RST}_j &= q_2 \\
\text{shiftA} &= q_1 \\
\text{shiftB} &= q_2 \\
\text{shiftLOW} &= q_2 \\
\text{shiftHI} &= q_1 + q_2 \\
\text{reset} &= q_2 \\
\text{selectSUM} &= q_1
\end{align*}
\]
How do we design counters?

• For binary counters (most common case) incremented circuit would work:

• In Verilog, a counter is specified as: \( x = x + 1; \)
  – This does \textit{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 \( \div 2 \) of previous.
- Look at output waveforms:
  - Often called “asynchronous” counters.

![Diagram of 4-bit binary ripple counter with T flip-flops and D flip-flops](image)

Fig. 6-8 4-Bit Binary Ripple Counter
Synchronous Counters

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

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

\[ a^+ = a' \]
\[ b^+ = a \oplus b \]

\[ c^+ = a'c + abc' + b'c \]
\[ = c(a'+b') + c'(ab) \]
\[ = c(ab)' + c'(ab) \]
\[ = c \oplus ab \]
Synchronous Counters

- 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?
- Delay grows $\alpha n$

- Generation of TC signals very similar to generation of carry signals in adder.
- “Parallel Prefix” circuit reduces delay:
Binary Counter from JK FFs

Count enable

A₀

A₁

A₂

A₃

CLK

To next stage

Fig. 6-12  4-Bit Synchronous Binary Counter
Up-Down Counter

\[
\begin{array}{cccc}
\text{c} & \text{b} & \text{a} & \text{c'} \text{b'} \text{a'} \\
0 & 0 & 0 & 1 1 1 \\
0 & 0 & 1 & 0 0 0 \\
0 & 1 & 0 & 0 0 1 \\
0 & 1 & 1 & 0 1 0 \\
1 & 0 & 0 & 0 1 1 \\
1 & 0 & 1 & 1 0 0 \\
1 & 1 & 0 & 1 0 1 \\
1 & 1 & 1 & 1 1 0 \\
\end{array}
\]

Down-count
With Parallel Load

Fig. 6-14 4-Bit Binary Counter with Parallel Load
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 ?

= 11 ?
```

- Alternative:

```
4-bit binary counter

load

4

TC
```
Ring Counters

• “one-hot” counters
0001, 0010, 0100, 1000, 0001, …

• What are these good for?

“Self-starting” version:
Ring Counters

(a) Ring-counter (initial value = 1000)

(b) Counter and decoder

(c) Sequence of four timing signals

Fig. 6-17 Generation of Timing Signals
Johnson Counter

(a) Four-stage switch-tail ring counter

(b) Count sequence and required decoding

<table>
<thead>
<tr>
<th>Sequence number</th>
<th>Flip-flop outputs</th>
<th>AND gate required for output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0 0 0</td>
<td>$A'E'$</td>
</tr>
<tr>
<td>2</td>
<td>1 0 0 0 0</td>
<td>$AB'$</td>
</tr>
<tr>
<td>3</td>
<td>1 1 0 0 0</td>
<td>$BC'$</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 0 0</td>
<td>$CE'$</td>
</tr>
<tr>
<td>5</td>
<td>1 1 1 1</td>
<td>$AE$</td>
</tr>
<tr>
<td>6</td>
<td>0 1 1 1</td>
<td>$A'B$</td>
</tr>
<tr>
<td>7</td>
<td>0 0 1 1</td>
<td>$B'C$</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 1</td>
<td>$C'E$</td>
</tr>
</tbody>
</table>
Register Summary

- All register (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”)

![Shift-Register Diagram]

- $x_3$, $x_2$, $x_1$, $x_0$, LD
- FF
- clk
- out
Universal Sift-register

Fig. 6-7  4-Bit Universal Shift Register