Counters

- Special sequential circuits (FSMs) that sequence through a set outputs.
- Examples:
  - Binary counter: 000, 001, 010, 011, 100, 101, 110, 111, 000, 001, ...
  - Gray code counter: 000, 010, 011, 110, 111, 101, 100, 001, 000, 010, 100, ...
  - One-hot counter: 0000, 0010, 0100, 0101, 1000, 1001, 1010, ...
  - 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 for UART circuit
  - 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

- State Transition Diagram:
  - Assume presence of two counters. An “i” counter for the outer loop and “j” counter for inner loop.
  - 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:
  - Outputs:
    - \( CE_i = q_0 \)
    - \( CE_j = q_1 \)
    - \( RST_i = q_2 \)
    - \( RST_j = q_2 \)
    - \( shiftA = q_3 \)
    - \( shiftB = q_4 \)
    - \( shiftLOW = q_5 \)
    - \( selectSUM = q_6 \)
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 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 $\frac{1}{2}$ of previous.
- Look at output waveforms:
  - Often called “asynchronous” counters.

### Synchronous Counters

- Binary Counter Design:
  - Start with 3-bit version and generalize:
    - \[ a \oplus b = a'b + ab' \]
    - \[ c \oplus a'b + ab'c = c + (a'b + ab')c \]
    - \[ c = a'(b + abc) + c'abd \]

- How do we extend to n-bits?
  - Extrapolate \( a^n \oplus b^n = a^n b^n + a^n b^n c \)

- 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.

### Binary Counter from JK FFs

- Generation of TC signals very similar to generation of carry signals in adder.
- “Parallel Prefix” circuit reduces delay:
  - \[ \text{Delay grows } \alpha n \]
Up-Down Counter

Down-count

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

Ring Counters
- "one-hot" counters
- What are these good for?

Johnson Counter

With Parallel Load

"Self-starting" version:

Ring Counters
- "one-hot" counters
- What are these good for?
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”)

Universal Shift-register

- 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”)