Shifters, Comparators, Increasers, Counters, Multipliers

Shifters

- Shifting (e.g., left shifting 0011 yields 0110) useful for:
  - Manipulating bits
  - Converting serial data to parallel (remember earlier above-mirror display example with shift registers)
  - Shift left once same as multiplying by 2 (0011 (3) becomes 0110 (6))
  - Why? Essentially appending a 0 — Note that multiplying decimal number by 10 accomplished just by appending 0, i.e., by shifting left (55 becomes 550)
  - Shift right once same as dividing by 2

Shifter Example: Approximate Celsius to Fahrenheit Converter

- Convert 8-bit Celsius input to 8-bit Fahrenheit output
  - \( F = C \times \frac{9}{5} + 32 \)
  - Approximate: \( F = C \times 2 + 32 \)
  - Use left shift: \( F = \text{left_shift}(C) + 32 \)

Shifter Example: Temperature Averager

- Four registers storing a history of temperatures
- Want to output the average of those temperatures
- Add, then divide by four
  - Same as shift right by 2
  - Use three adders, and right shift by two

Barrel Shifter

- A shifter that can shift by any amount
  - 4-bit barrel left shift can shift left by 0, 1, 2, or 3 positions
  - 8-bit barrel left shifter can shift left by 0, 1, 2, 3, 4, 5, 6, or 7 positions
  - (Shifting an n-bit number by 8 positions is pointless — you just lose all the bits)
- Could design using \( 8 \times 1 \) muxes and lots of wires
  - Too many wires
- More elegant design
  - Chain three shifters: 4, 2, and 1
  - Can achieve any shift of 0..7 by enabling the correct combination of those three shifters, i.e., shifts should sum to desired amount
Comparators

- N-bit equality comparator: Outputs 1 if two N-bit numbers are equal
  - 4-bit equality comparator with inputs A and B
    - \( a0, a1, a2, a3, b0, b1, b2, b3 \)
      - Two bits are equal if \( (a0 = b0) \land (a1 = b1) \land (a2 = b2) \land (a3 = b3) \)
    - Recall that XNOR outputs 1 if its two input bits are the same
      - \( \text{eq} = (a0 \oplus b0) \land (a1 \oplus b1) \land (a2 \oplus b2) \land (a3 \oplus b3) \)
  - Even though \( \text{XNOR} \) outputs 1 if its two input bits are the same

Magnitude Comparator

- 4-bit equality comparator with inputs A and B
  - \( A = 1011 \) and \( B = 1001 \)
  - \( A \gt B \) if already determined in higher stage, or if higher stages equal but in this stage
  - \( A \lt B \) if already determined in higher stage, or if higher stages equal but in this stage
  - \( A = B \) if already determined in higher stage, or if higher stages equal but in this stage

Magnitude Comparator

- How does it work?
  - \( A = 1011 \) and \( B = 1001 \)
  - \( A \gt B \) if already determined in higher stage, or if higher stages equal but in this stage
  - \( A = B \) if already determined in higher stage, or if higher stages equal but in this stage
  - \( A \lt B \) if already determined in higher stage, or if higher stages equal but in this stage

Magnitude Comparator

- By-hand example leads to idea for design
  - Start at left, compare each bit pair, pass results to the right
  - Each bit pair called a stage
  - Each stage has 3 inputs indicating results of higher stage, passes results to lower stage
  - \( A \gt B \) if already determined in higher stage, or if higher stages equal but in this stage
  - \( A = B \) if already determined in higher stage, or if higher stages equal but in this stage
  - \( A \lt B \) if already determined in higher stage, or if higher stages equal but in this stage
**Magnitude Comparator Example: Minimum of Two Numbers**

- Design a combinational component that computes the minimum of two 8-bit numbers.
  - Solution: Use 8-bit magnitude comparator and 8-bit 2x1 mux
    - If A \( \leq \) B, pass A through mux. Else, pass B.

**Incrementer**

- Can build faster incrementer using combinational logic design process
  - Capture truth table
  - Derive equation for each output
    - \( c_0 = a_3a_2a_1a_0 \)
    - ...\( c_0 = a_0a_1 \)
  - Results in small and fast circuit
  - Note: works for small \( N \) — larger \( N \) leads to exponential growth, like for N-bit adder

**Counters**

- **N-bit up-counter**: N-bit register that can increment (add 1) to its own value on each clock cycle
  - 0000, 0001, 0010, 0011, ..., 1110, 1111, 0000
  - Note how count “rolls over” from 1111 to 0000
    - Terminal (last) count, \( tc \), equals \( 1 \) during value just before rollover
  - Internal design
    - Register, incrementer, and N-input AND gate to detect terminal count

**Counter Example: Mode in Above-Mirror Display**

- Recall above-mirror display example from Chapter 2
  - Assumed component that incremented \( xy \) input each time button pressed: 00, 01, 10, 11, 00, 01, 10, 11, 00, ...
  - Can use 2-bit up-counter
    - Assumes \( mode=1 \) for just one clock cycle during each button press
    - Recall “Button press synchronizer” example from Chapter 3

**Counter Example: 1 Hz Pulse Generator Using 256 Hz Oscillator**

- Suppose have 256 Hz oscillator, but want 1 Hz pulse
  - 1 Hz is 1 pulse per second — useful for keeping time
  - Design using 8-bit up-counter, use \( tc \) output as pulse
    - Counts from 0 to 255 (256 counts), so pulses 1 Hz every 256 cycles
Down-Counter

- 4-bit down-counter
  - 1111, 1110, 1101, 1100, ..., 0011, 0010, 0001, 0000, 1111, ...
  - Terminal count is 0000
    - Use NOR gate to detect
    - Need decrements (+1) – design like designed incrementer

Up/Down-Counter

- Can count either up or down
  - Includes both incrementer and decrements
  - Use dir input to select, using 2x1: dir=0 means up
  - Likewise, dir selects appropriate terminal count value

Counter with Parallel Load

- Up-counter that can be loaded with external value
  - Designed using 2x1 mux – id input selects incremented value or external value
  - Load the internal register when loading external value or when counting

Counter Example: Light Sequencer

- Illuminate 8 lights from right to left, one at a time, one per second
- Use 3-bit up-counter to counter from 0 to 7
- Use 3x8 decoder to illuminate appropriate light
- Note: Used 3-bit counter with 3x8 decoder
  - NOT an 8-bit counter – why not?

Counter with Parallel Load

- Useful to create pulses at specific multiples of clock
  - Not just at N-bit counter’s natural wrap-around of 2^N
- Example: Pulse every 9 clock cycles
  - Use 4-bit down-counter with parallel load
  - Set parallel load input to 8 (1000)
  - Use terminal count to reload
    - When count reaches 0, next cycle loads 8.
  - Why load 8 and not 9? Because 0 is included in count sequence:
    - 8, 7, 6, 5, 4, 3, 2, 1, 0 \rightarrow 9 counts

Counter Example: New Year’s Eve Countdown Display

- Chapter 2 example previously used microprocessor to counter from 59 down to 0 in binary
- Can use 8-bit (or 7- or 6-bit) down-counter instead, initially loaded with 59
### Counter Example: 1 Hz Pulse Generator from 60 Hz Clock
- U.S. electricity standard uses 60 Hz signal
  - Device may convert that to 1 Hz signal to count seconds
- Use 6-bit up-counter
  - Can count from 0 to 63
  - Create simple logic to detect 59 (for 60 counts)
  - Use to clear the counter back to 0 (or to load 0)

### Timer
- A type of counter used to measure time
  - If we know the counter’s clock frequency and the count, we know the time that’s been counted
- Example: Compute car’s speed using two sensors
  - First sensor (a) clears and starts timer
  - Second sensor (b) stops timer
  - Assuming clock of 1kHz, timer output represents time to travel between sensors.
  - Knowing the distance, we can compute speed

### Multiplier – Array Style
- Generalized representation of multiplication by hand

### Multiplier – Array Style
- Multiplier design – array of AND gates

### Design Example
- Can build multiplier that mimics multiplication by hand
  - Notice that multiplying multiplicand by 1 is same as ANDing with 1

### Design Example
- Design a more accurate version of the Celsius to Fahrenheit converter. The new conversion circuit receives a digitized temperature in Celsius as a 16-bit binary number $C$ and outputs the temperature in Fahrenheit as a 16-bit output $F$. Our more accurate equation for calculating an approximate conversion from Celsius to Fahrenheit is: $F = C \times \frac{30}{16} + 32$. 
Design Challenge

- Design Challenge
  - Design a comparator that determines if three 4-bit numbers are equal, by connecting 4-bit magnitude comparators together and using additional logic if necessary.

Due Next Lecture (as announced in class)