**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: \(0011 \text{ (3)} \rightarrow 0110 \text{ (6)}\)
  - Shift right once same as dividing by 2

**Shift 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: \(F = \text{left shift}(C) + 32\)

<table>
<thead>
<tr>
<th>C</th>
<th>8-bit left shift</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00001</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00010</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00011</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00100</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00101</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00110</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>00111</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01000</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01001</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01010</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01011</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01100</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01101</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01110</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>01111</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10001</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10010</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10011</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10100</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10101</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10110</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>10111</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11000</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11001</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11010</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11011</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11100</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11101</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11110</td>
<td>00000000 (0)</td>
<td></td>
</tr>
<tr>
<td>11111</td>
<td>00000000 (0)</td>
<td></td>
</tr>
</tbody>
</table>

**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 two
  - 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 8-bit number by 8 positions is pointless -- you just lose all the bits)
- Could design using 8x1 muxes and lots of wires
  - Too many wires
- More elegant design
  - Chain three shifters: 4, 2, and 1
  - Can achieve any shift of 0, 2, 3, 4, 5, 6, or 7 positions 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 = equal? (a3 = a2 = a1 = a0)
    - Two bits are equal if: 1, 0 = 0, 0
    - eq = (a3b3 + a3'b3') (a2b2 + a2'b2') (a1b1 + a1'b1') (a0b0 + a0'b0')
    - Recall that XOR returns 1 if two input bits are different

By-hand example leads to idea for design

How does it work?

0110 = 0111 ?

01100111

– Each bit pair called a stage
– 4-bit equality comparator with inputs A and B

Frank Vahid

Copyright © 2006
Digital Design

Magnitude Comparator

- **N-bit magnitude comparator** indicates whether A=B, A>B, or A<B, for its two N-bit inputs A and B
  - How design? Consider how compare by hand. First compare a3 and b3. If equal, compare a2 and b2. And so on. Stop if comparison not equal — whichever’s bit is 1 is greater. If never see unequal bit pair, A=B.

Each stage:

- Simple circuit inside each stage, just a few gates (not shown)
- Bit inputs A and B
- Whether A>B, A=B, or A<B, for its two N-bits

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

Magnitude Comparator

- How does it work?

1011 = 1001 ?

11 00 10 11

(eq = (a3b3 + a3'b3') (a2b2 + a2'b2') (a1b1 + a1'b1') (a0b0 + a0'b0'))

Magnitude Comparator

- Final answer appears on the right
  - Takes time for answer to "ripple" from left to right
  - Thus called "carry-ripple style" after the carry-ripple adder

Even though there’s no "carry" involved

Magnitude Comparator

A=1011 B=1001

1011 1001 Equal

1011 1001 Equal

1011 1001 Unequal

So A > B

Magnitude Comparator

A=1011 B=1001

1011 1001 Equal

1011 1001 Equal

1011 1001 Unequal

So A > B
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>B, pass A through mux. Else, pass B.

Incrementer

- Counter design used incrementer
- Incrementer design
  - Could use carry-ripple adder with B input set to 00...001
  - But when adding 00...001 to another number, the leading 0's obviously don't need to be considered — so just two bits being added per column
  - Use half-adders (adds two bits) rather than full-adders (adds three bits)

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

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: 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 tC 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 decrementer (-1) – design like designed incrementer

**Up/Down-Counter**

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

**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 → 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 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: 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

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

Multiplier – Array Style

- Generalized representation of multiplication by hand

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 1

• 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)
1 point extra credit (Homework)

Design Challenge 2

• Design Challenge
  – Design a 4-bit up-counter with an additional output upper. upper outputs a 1 whenever the counter is within the upper half of the counter’s range, 8 to 15.

Due Next Lecture (as announced in class)
1 point extra credit (Homework)