CS 3220
Project 1 - Tips and Tricks
Project 1 – “Watch”

- Let’s pretend our board is a wristwatch
- Has display and two buttons (we’ll use KEY[0] and KEY[1])
- Has three modes – clock display, clock set, and stopwatch

• Clock display mode
  - Display time on the HEX display in the HH:MM format
  - Show seconds in binary on LEDG
  - Time starts off at 00:00:00, hours shown in 24-hour format
  - Push KEY[0] to switch to stopwatch mode
  - Hold KEY[1] for 1s to enter clock-set mode (if released in less than 1s, ignore the key press)

• Clock-set mode
  - Display same as in clock display mode, but HEX and LEDG display blinking (off for 0.5s, then show time for 0.5s)
  - When KEY[1] pushed and released in less than 1s, increase time by 1 minute (and reset seconds to zero)
  - When KEY[1] is held for more than 1s, increase time by 1 hour for each second that KEY[1] is held pressed
  - Push KEY[0] to go to clock display mode
Project 1 – “Watch”

• Stopwatch mode
  – Show stopwatch time in the MM:SS format on HEX display, LEDG shows hundredths of a second in binary
  – Initially stopwatch is in the “initial” state (shows 00:00.00)
  – When KEY[1] is pushed while in the initial state, start stopwatch
  – When KEY[1] is pushed while stopwatch running, stop stopwatch (freeze the stopwatch time)
  – When KEY[1] is pushed while stopwatch is stopped, reset stopwatch to 00:00.00 and go back to “initial state
  – When KEY[0] is pushed, switch to clock display mode

• Changing the mode just changes what is shown and what keys do!
  – Stopwatch keeps running in all modes
    • If stopwatch running when we leave stopwatch mode, keep counting until we come back to stopwatch mode and push KEY[1] to stop
    • If stopwatch stopped when we leave stopwatch mode, remember stopwatch time and re-display when we are back to this mode
  – Clock keeps correct time in all modes!
Strategy

• Make a working clock - just counts time properly
  – Test by initializing with “interesting” times
• Add ability to increment hours and minutes
  – Test by initializing into clock-set mode
• Add switch between clock and clock set modes
• Add stopwatch counting – just counts properly
  – Test by initializing into started-stopwatch state
• Add ability to control stopwatch
• Now add ability to switch between modes!
always @(posedge clk) begin
  if(ClockSecCnt!=26'd49999999) ClockSecCnt<=ClockSecCnt+26'd1;
  else begin
    ClockSecCnt<=26'd0;
  end
  if(ClockSecs!=6'd59) ClockSecs<=ClockSecs+6'd1;
  else begin
    ClockSecs<=6'd0;
  end
  if(ClockMinsLo!=4'd9) ClockMinsLo<=ClockMinsLo+4'd1;
  else begin
    ClockMinsLo<=4'd0;
  end
  if(ClockMinsHi!=4'd5) ClockMinsHi<=ClockMinsHi+4'd1;
  else begin
    ClockMinsHi<=4'd0;
  end
  if({ClockHourHi,ClockHourLo}==8'h23)
    {ClockHourHi,ClockHourLo}<=8'h00;
  else if(ClockHourLo==4'd9)
    {ClockHourHi,ClockHourLo}<={ClockHourHi+4'd1,4'd0};
  else
    ClockHourLo<=ClockHourLo+4'd1;
end end end end end end
• Use KEY[3] and KEY[2] to test it
  – Will figure out the real thing with KEY[1] later
• How to detect a key press/release?
  – Not using “always @(posedge KEY[3])” !!!

```verilog
reg [3:0] PrevKey=4'b1111;
always @(posedge clk)
  PrevKey<=KEY;
wire Key3Press={PrevKey[3],KEY[3]}==2'b10;
wire Key3Release={PrevKey[3],KEY[3]}==2'b01;
```
• Clock already has increment-minutes logic
  – Happens every minute!
  – Just make it also happen when key pressed 😊

• But the previous code is not very nice for this
  – Must first go through seconds-increment code to get to minute-increment code
  – For hours, must go through seconds, minutes, and ten minutes increment logic to get to hours-increment
    • But inc-hour should leave minutes, seconds, etc. as they were!
wire sigclksec=(ClockSecCnt==49999999);
wire sigclkmin=((ClockSecs==59)&&sigclksec)||((ClockSet&&Key3Press));
wire sigclktenmin=(ClockMinsLo==4'd9)&&sigclkmin;
...

always @(posedge clk) begin
  if(sigclksec)
    ClockSecCnt<=0;
  else
    ClockSecCnt<=ClockSecCnt+1;
  if(sigclkmin)
    ClockSecs<=0;
  else if(sigclksec)
    ClockSecs<=ClockSecs+1;
  if(sigclktenmin)
    ClockMinsLo<=4'd0;
  else if(sigclkmin)
    ClockMinsLo<=ClockMinsLo+4'd1;
  ...

Switch between clock and clock-set

• When in clock-set, Key0Press changes to clock
• When in clock, holding KEY[1] …
  – How to detect this?
  – Another counter!
  – Start counting when KEY[1] pressed
  – Create a signal that is 1 when counter at exactly 1s
    • For only one clock cycle, then make signal zero again!
    • Remember that counter was at 1s already (in a reg)
  – Switch from clock-set to clock when signal is 1 and was-already-1 is 0
Stopwatch – almost exactly the same as clock
  – Except that it counts 1/100ths of a second…
Start/stop/reset
  – Three states, move between them on Key1Press
Increment hours and minutes properly
  – Similar to the 1s hold stuff we already discussed