Lab 10: Concurrent synchSMs (2 days)

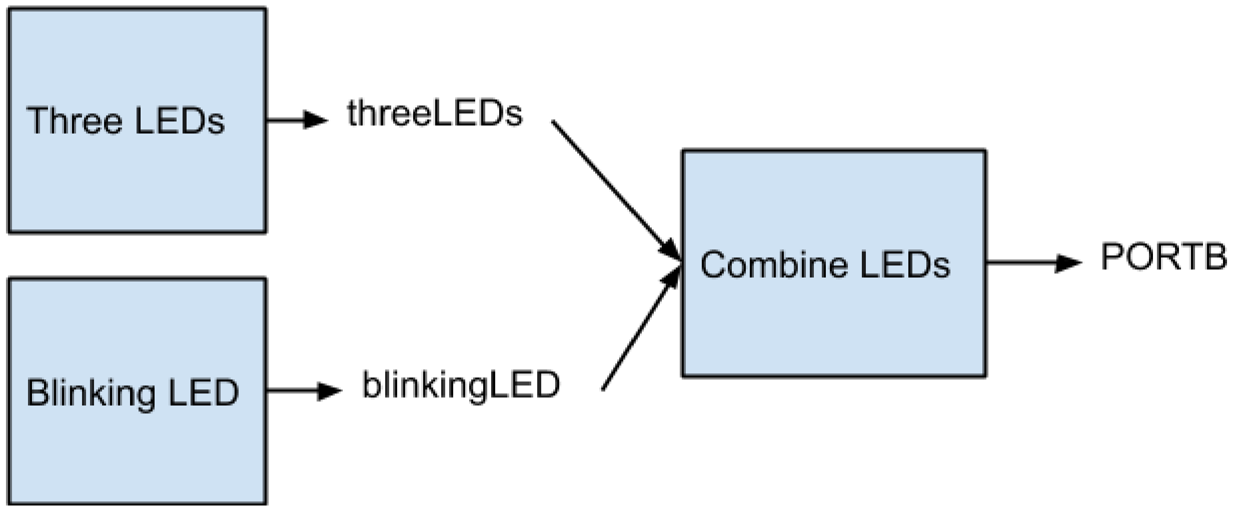
UCR EE/CS120B

Pre-lab

Have your board fully wired and have your synchSMs and your complete C code for Exercise 1. *Be sure to use the clean timer abstraction and the structured method for converting synchSMs to C.*

Exercises

1. Connect LEDs to `PB0`, `PB1`, `PB2`, and `PB3`.
    a. In one state machine (`ThreeLEDsSM`), output to a shared variable (`threeLEDs`) the following behavior: set only bit 0 to 1, then only bit 1, then only bit 2 in sequence for 1 second each.
    b. In a second state machine (`BlinkingLEDSM`), output to a shared variable (`blinkingLED`) the following behavior: set bit 3 to 1 for 1 second, then 0 for 1 second.
    c. In a third state machine (`CombineLEDsSM`), combine both shared variables and output to the `PORTB`.
    Note: only one SM is writing to outputs. Do this for the rest of the quarter.



Concurrency with different period-tasks can be achieved by maintaining the elapsed time since

the last tick for each task. A simple method ticks the timer at 1 ms and then counts X ticks to determine period X. (**Do *not* tick the timer at the GCD of the tasks**).
Video Demonstration: http://youtu.be/Snmt0VFE_Zs

2. Modify the above example so the `threeLEDs` light for 300 ms, while `blinkingLED`'s LED still blinks 1 second on and 1 second off.

Video Demonstration: http://youtu.be/i8f5JSteH-U

3. To the previous exercise's implementation, connect your speaker's red wire to `PB4` and black wire to ground. Add a third task that toggles `PB4` on for 2 ms and off for 2 ms as long as a switch on `PA2` is in the on position. **Don't use the PWM for this task.**

Video Demonstration: http://youtu.be/Ufrlc6xyPyQ

4. (**Challenge**) Extend the previous exercise to allow a user to adjust the sound frequency up or down using buttons connected to `PA0` (up) and `PA1` (down). Using our 1 ms timer abstraction, the fastest you'll be able to pulse is 1 ms on and 1 ms off, meaning 500 Hz. **Hint**: You'll probably want to introduce another synchSM that polls the buttons and sets a global variable storing the current frequency that in turn is read by the frequency generator task.

Video Demonstration: http://youtu.be/mt8eznAcp6o

Submission

Each student must submit their source files (`.c`) and test files (`.gdb`) according to instructions in the lab submission guidelines.

```
$ tar -czvf [cslogin]_lab2.tgz turnin/
```

Don't forget to commit and push to Github before you logout!