question | Like **36** views

question on lab 10 when input = 0x7 (2 cars and a pedestrian present)

Below is a snippet from my lab 10 - I'm wondering where I went wrong when the input = 0x7, cars on west and south roads, and a pedestrian present.
I put a for-loop in when I detected the input = 0x7, and cycled through each of my 11 states (not 9 - added 2 extra states - is that fine - or do you have to implement it in 9 states to make it work with the grader?) So it output to PB5-0 and PF3, PF1 for each state, before reading the sensor input again.
However, the grader never gets passed this check of the output sequence for Input = 0x7 before failing (first test). I chose delay = 50 since 50 times .01 sec (10ms) = .5sec, or the less than 1 second I thought was recommended for the time to wait in each state.

At first I had implemented it in 9 states, and then I thought it would be easier with 11 states, since each of the goWest or goSouth states could then have 2 waitToGoTo--the next states. Perhaps that is incorrect? The reason I changed it was because if you are in goWest for example, and then move to waitWest, you could be waiting because either a pedestrian had arrived and you moved to the wait state, or a car on the south road had arrived and you moved to the wait state, so I added those 2 extra wait states dependent on whether you were moving because of a sensor detect a pedestrian or a car.

So I'm not sure if that is a problem, but I did implement it that way. If that is fine, then my question is in regard to when I cycle through the outputs of these 11 states, the grader chokes. I observed in the simulator that all the LEDs were as I had expected them to be (but not as the grader expected them to be).

I thought the lab had also stated that you should not put a for loop within your while loop, but I did, once I detected Input on sensors = 0x7, so that I would not change the Input again until I cycled through the 11 states and output to the LEDs. Then I allowed the Input to read the sensors again.

Here is my code snippet. Any help appreciated! And I know I'm behind most people in this class for one reason or another, in that I'm working on lab 10 today, April 10th! I hope to catch up though before the course ends. Thanks.

```
// west facing red light connected to PB5
// west facing yellow light connected to PB4
// west facing green light connected to PB3
// south facing red light connected to PB2
// south facing yellow light connected to PB1
// south facing green light connected to PB0
// pedestrian / walker - WALK green light connected to PF3
// pedestrian / walker - DONT WALK - SOLID red light connected to PF1
// pedestrian / walker - HURRY WALK - FLASH red light ON and OFF TWICE connected to PF1
// pedestrian / walker present on either west or south road sensor detector connected to PE2 (1=walker present)
// west facing car detector connected to PE1 (1=car present)
// south facing car detector connected to PE0 (1=car present)

  while(1){
    LIGHT = FSM[S].Out >> 2;  // set west and south road traffic LED lights (PB5-0) - output based on current state
        GPIO_PORTF_DATA_R = ((FSM[S].Out & 0x2) << 2) | ((FSM[S].Out & 0x1) << 1); // set walk/don't walk leds (PF3 and PF1)
    SysTick_Wait10ms(FSM[S].Time);
    Input = SENSOR;     // read sensors
        if (Input == 0x7) {
            for (i = 0; i < 10; i++) {
                S = FSM[S].Next[Input];
                LIGHT = FSM[S].Out >> 2;  // set west and south road traffic LED lights (PB5-0) - output based on current state
                GPIO_PORTF_DATA_R = ((FSM[S].Out & 0x2) << 2) | ((FSM[S].Out & 0x1) << 1); // set walk/don't walk leds (PF3 and PF1)
                SysTick_Wait10ms(FSM[S].Time);
            }
            Input = SENSOR;     // read sensors
        }
        S = FSM[S].Next[Input];
  }

struct State {
  unsigned long Out;
  unsigned long Time;
  unsigned long Next[11];};
typedef const struct State STyp;
#define goWest   0
#define waitWestToGoSouth 1
#define waitWestToGoWalk   2
#define goSouth 3
#define waitSouthToGoWest   4
#define waitSouthToGoWalk 5
#define goWalk   6
#define hurryWalk 7
#define noWalkNoCars1  8
#define dontWalkNoCars  9
#define noWalkNoCars2  10

STyp FSM[11]={
 {0x31,50,{goWest, waitWestToGoSouth, goWest, waitWestToGoSouth, waitWestToGoWalk, waitWestToGoSouth, waitWestToGoWalk, waitWestToGoSouth}},

 {0x45,50,{goSouth, goSouth, goSouth, goSouth, goSouth, goSouth, goSouth, waitWestToGoWalk}},
 {0x51,50,{goWalk, goWalk, goWalk, goWalk, goWalk, goWalk, goWalk, goSouth}},
 {0x85,50,{goSouth, goSouth, waitSouthToGoWest, waitSouthToGoWest, waitSouthToGoWalk, waitSouthToGoWalk, waitSouthToGoWest,
waitSouthToGoWest}},
 {0x29,50,{goWest, goWest, goWest, goWest, goWest, goWest, goWest, waitSouthToGoWalk}},
 {0x89,50,{goWalk, goWalk, goWalk, goWalk, goWalk, goWalk, goWalk, goWalk}},
 {0x92,50,{hurryWalk, hurryWalk, hurryWalk, hurryWalk, goWalk, goWalk, goWalk, hurryWalk}},
 {0x91,50,{noWalkNoCars1, noWalkNoCars1, noWalkNoCars1, noWalkNoCars1, noWalkNoCars1, noWalkNoCars1, noWalkNoCars1, noWalkNoCars1 }},
 {0x90,50,{dontWalkNoCars, dontWalkNoCars, dontWalkNoCars, dontWalkNoCars, dontWalkNoCars, dontWalkNoCars, dontWalkNoCars, dontWalkNoCars }},

 {0x91,50,{noWalkNoCars2, noWalkNoCars2, noWalkNoCars2, noWalkNoCars2, noWalkNoCars2, noWalkNoCars2, noWalkNoCars2, noWalkNoCars2 }},
 {0x90,50,{noWalkNoCars2, goSouth, goWest, goWest, goWalk, goSouth, goWest, goWest }}};
```

lab10

1 day ago by Karen West

the students' answer, *where students collectively construct a single answer*

Click to start off the wiki answer

the instructors' answer, *where instructors collectively construct a single answer*

From Chapter & Lab 10 :

1) Because we think being in a state is defined by the output pattern, we think you should implement a Moore and not a Mealy machine. However, your scheme should use a linked data structure stored in ROM.

2) There should be a 1-1 mapping between FSM graph and data structure. For a Moore machine, this means each state in the graph has a name, an output, a time to wait, and 8 next state links (one for each input). The data structure has exactly these components: a name, an output, a time to wait, and 8 next state pointers (one for each input). There is no more or no less information in the data structure then the information in the state graph.

*3) There can be no conditional branches in program, other than the **while** in **SysTick_Wait** and the **for**in **SysTick_Wait10ms**. This will simplify debugging make the FSM engine trivial.*

4) The state graph defines exactly what the system does in a clear and unambiguous fashion. In other words, do not embed functionality (e.g., flash 3 times) into the software that is not explicitly defined in the state graph.

5) Each state has the same format of each state. This means every state has exact one name, one 8-bit output (could be stored as one or two fields in the struct), one time to wait, and 8 next indices.

If you see 3) , it says no conditionals or loops in the program. By keeping a loop, you are violating the concept of FSM where each state has one set of inputs, one set of outputs and a delay time. Your FSM outputs of the current state are defined by FSM inputs from the same state. This is a Mealy FSM. You need to implement a Moore FSM. A Moore FSM is characterized by output depending only on the current state and not the current input.
Only the next state depends on the current input.

1 day ago by Chinmaya Dattathri

followup discussions *for lingering questions and comments*

Resolved [✓ Unresolved]

Karen West 1 day ago
Thank you for your response. I will look into it.

Anonymous 2 hours ago Hi Karen,
Chinmaya's answer is true but a bit formal in nature. For our problem it amounts too --- you shouldn't need to do any testing in the main loop (that's what makes it a Mealy State machine) to decide what the output should be, it all just comes from the state table and looks like:

```
while(1){
TRAFFICLIGHT = // from fsm[State].Out or a subset
PEDESTRIANLIGHT =  // from fsm[State].Out or a subset or as a seperate Struct item
//  wait some period of time
Input = SENSOR_PORTA >> 2; // read sensors shift down from bits 4-2
State = fsm[State].Next[Input];
}
```

You need 9/10/11 states depending on whether or not you have an all Red state and how many times you flash Red for Pedestrian "hurry up".

Hope this helps.