

University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS 150
Spring 2007

R. H. Katz

Problem Set # 5
(Assigned 20 February, Due 2 March)

1. (*Verilog Synthesis*) Show the hardware, down to the level of multiplexers and flip-flops, which is likely to be generated by Verilog synthesis from the following snippet of Verilog behavioral description. It is perfectly fine to use Synplify to do your homework for you!

```
always @(a or c) begin
    if (c) begin
        value = a;
    end
    b = value;
end
```

2. (*Verilog Synthesis*) Show the hardware, down to the level of multiplexers and flip-flops, which is likely to be generated by Verilog synthesis from the following snippets of Verilog behavioral description. It is perfectly fine to use Synplify to do your homework for you!

```
(a) module mux3 (y, a, b, c, sel);
    output      y;
    input  [1:0] sel;
    input      a, b, c;
    reg        y;
    always @(a or b or c or sel)
        case (sel)
            2'b00: y = a;
            2'b01: y = b;
            2'b10: y = c;
            2'b11: y = 2'bxx;
        endcase
endmodule
```

```
(b) module mux3 (y, a, b, c, sel);
    output      y;
    input  [1:0] sel;
    input      a, b, c;
    reg        y;
    always @(a or b or c or sel)
        case (sel)
            2'b00: y = a;
            2'b01: y = b;
            2'b10: y = c;
            default: y = 2'bxx;
        endcase
endmodule
```

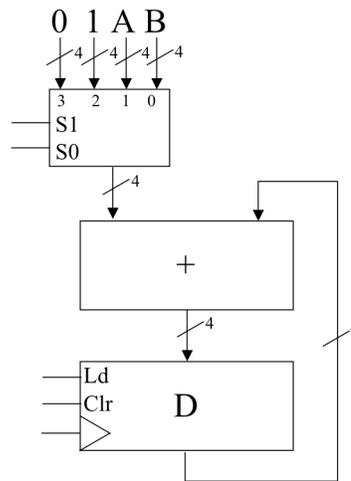
```
(c) module mux3 (y, a, b, c, sel);
    output      y;
```

```

input  [1:0] sel;
input   a, b, c;
reg     y;
always @(a or b or c or sel)
    case (sel)
        2'b00: y = a;
        2'b01: y = b;
        2'b10: y = c;
    endcase
endmodule

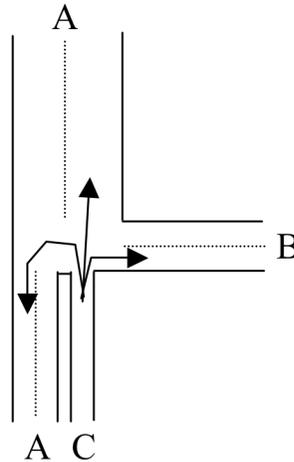
```

3. (*Control State Machines*) Below is a portion of a 4-bit datapath with four 4-bit input sources (constants 0, 1, and variables A, B), a 4-bit adder and a 4-bit register with synchronous load (Ld) and clear (Clr). This means that when the Clk undergoes a positive transition, if Ld is asserted, a new value is saved in D and if Clr is asserted, a zero is stored there. Use this datapath to compute the value $2A + B + 1$ and save it in D.



- Write high-level behavioral Verilog, independent of the datapath, to perform this computation.
 - Using the datapath in the figure, draw a Moore machine state diagram to perform the computation. The outputs from this state machine are the mux selection signals and the D load and clear signals.
 - Write behavioral Verilog that corresponds to your answer to part (b).
 - Draw a timing waveform, showing how your machine sequences through its states, asserting its control signals. Indicate on this chart what values are available at the inputs to the adder and the D register.
4. (*State Machine Word Problem*) Consider a variation on Professor Katz's ever popular traffic light problem. In this case, three streets come together in an unusual T-intersection, indicated by the figure below. Street A is a wide two-way street that runs North-South. Street B is a wide two-way street that runs East-West but intersections A in a T-intersection. Street C is a narrow one-way street heading North that parallels A but is separated from it by a barrier. Design a traffic light Moore machine to the following specification:
- It should be possible to turn from A onto B and vice versa without colliding with traffic coming from C.
 - The traffic from C should be able to continue onto A heading N or heading S without colliding with traffic from A or from B.

Write Verilog for the state machine with the minimum number of states that meets this specification. Your Verilog should have as its outputs the lights to illuminate for Streets A, B, and C. Indicate any additional assumptions you are making. The streets intersect as shown below:



5. *(State Machine Word Problem)* Consider the design of a dual elevator controller. The building has two elevators that move between two floors. The first floor has a single up button and the second floor a single down button. Inside each elevator are also two buttons, up and down. Normally, an elevator will move to a given floor when instructed to do so by the control, opening its door for a twenty seconds to let passengers on and off. Inputs to the system include sensors that indicate whether the elevator is at a given floor and if a passenger moving on or off the elevator is obstructing its doors. Furthermore, elevators can either be moving between floors or sitting idle on one of the floors. If both elevators are “idle” and on the same floor, choose an algorithm to decide which one to use to respond when a user “calls” for an elevator. Consider the cases when the elevators are both on the same floor, one each on two different floors, as well as moving in either (or the same!) direction. For example, if one elevator is already moving a floor where someone has pushed the call button, then allow that elevator to service the call rather than send the other elevator as well. This will require that you maintain the current state of the system in terms of where the elevators are, what they are doing, and what calls are pending, but these need not be encoded solely in the states of the state machine: you can also have some information saved in associated registers and other datapath components (recall the combination lock presented in lecture and in laboratory). While you can make assumptions, the behavior of the system must be reasonable. For example, pressing the up button with the elevator on the second floor causes the elevator to simply remain there with its door open. Also if the elevator is moving from the first to the second floor, pressing the first floor button inside the elevator should have no effect.
- Indicate any assumptions that you are making.
 - Identify your inputs, outputs, and name and describe your states. What additional circuitry, like timers, flip-flops, comparators, etc., do you need outside of the control state machine to completely specify the elevator control system?
 - Draw a symbolic state diagram for your design, labeling all state transitions.
 - Write Verilog code for a Moore Machine implementation of the state diagram from part (c).