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

EECS 150 R. H. Katz
Fall 2005

Problem Set # 5
(Assigned 5 October, Due 14 October)

SOLUTIONS

1. Consider the design of an elevator controller. The building has three floors, an up button on the first floor, up and down buttons on the second floor, a down button on the third floor, and three buttons inside the elevator indicating the floor to go to. Note that more than one button inside the elevator may have been pressed and active at the same time. While you can make assumptions, the behavior of the system must be reasonable. For example, pressing the "Floor 2" button with the elevator on the second floor causes the elevator to remain there with its door open. Also if the elevator is moving from the second to the third floor, pressing the first floor button inside the elevator should have no effect.
   (a) 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 state machine?

   One possible solution

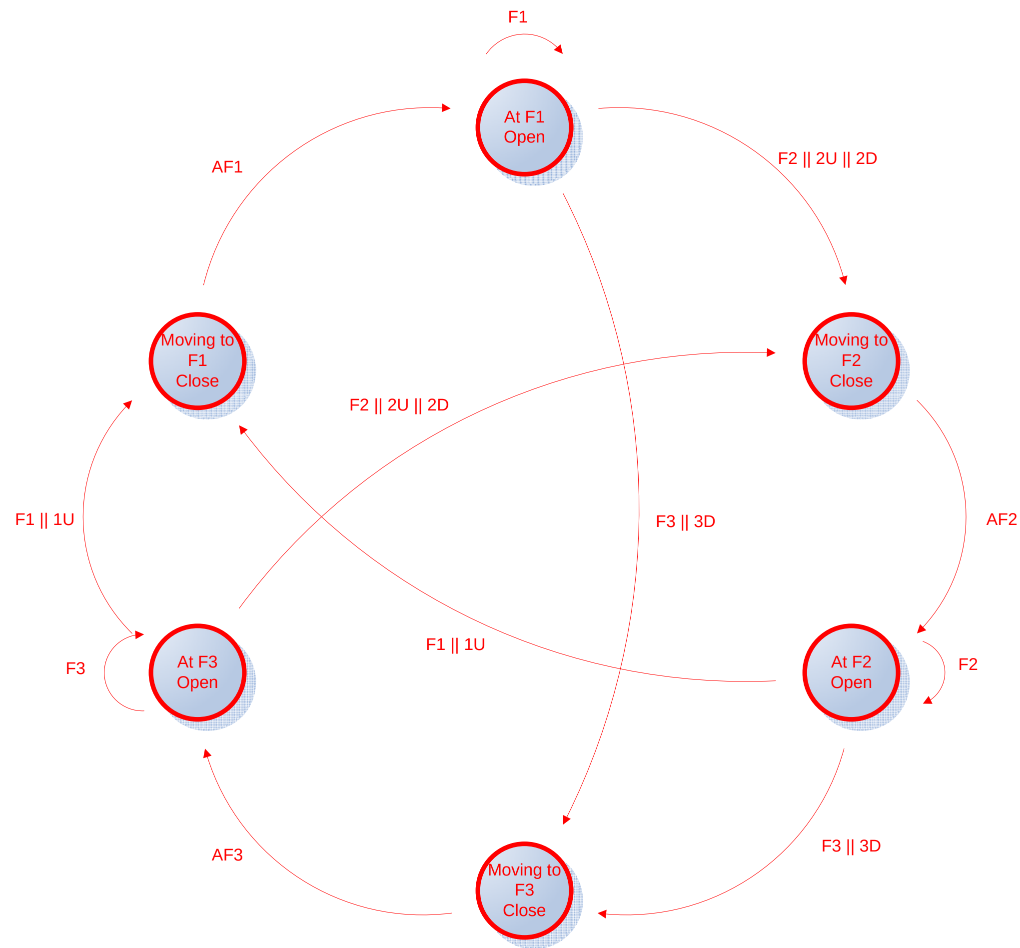
   Inputs: F1, F2, F3 (Buttons inside the elevator)
   1U, 2U, 2D, 3D (Buttons on each floor)
   AF1, AF2, AF3 (Sensors to tell when Elevator Arrives at Floor)

   Outputs: Open (Whether door is open or closed)

   For a complex system, timers could be used to wait a few seconds before the door closes after user selects the floor. Registers can be used to store the floor selection of the passenger if multiple floors were selected.

   S_F1: At Floor One, Open Door. Wait for user to press selection.
   S_MF1: In Transit to Floor 1. Door closed. Wait for arrival at selected floor via AF1 (Arrival at F1)
   …

Draw a symbolic state diagram for your design, labeling all state transitions.

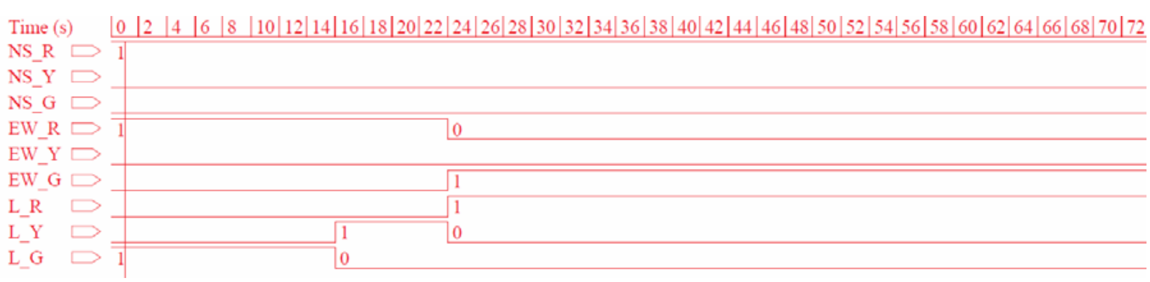


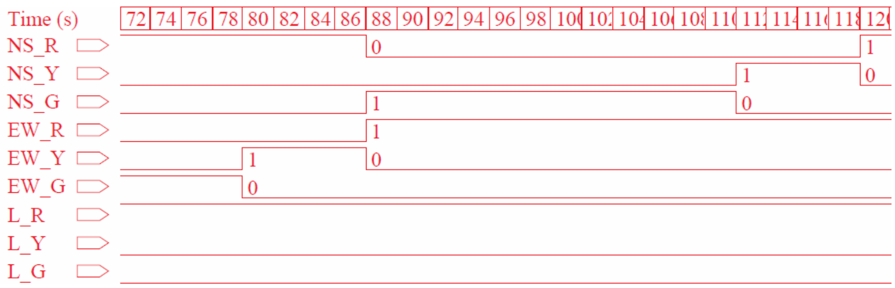
(b) Write "sketch" Verilog code for a Moore Machine implementation of this state diagram.

```
Always @ (*) begin
        NS = CS;
        Case (CS)
                S_F1:   begin
                        If (F2 || 2U || 2D)
                                NS = S_MF2;
                        If(F3 || 3D)
                                NS = S_MF3;
                        End

                S_F2:   begin
                        If(F1 || 1U)
                                NS = S_MF1;
                        If(F3 || 3D)
                                NS = S_MF3;
                        End

                S_F3:   begin
                        If(F1 || 1U)
                                NS = S_MF1;
```

```
                    If (F2 || 2U || 2D)
                            NS = S_MF2;
                    End

                S_MF1: begin
                    If (AF1)
                            NS = S_F1;
                    End

                S_MF2: begin
                    If (AF2)
                            NS=S_F2;
                    End

                S_MF3: begin
                    If (AF3)
                            NS=S_F3;
                    End
        Endcase
End

Always @ (posedge clk) begin
        If (Reset)
                CS<=S_F1;
        Else
                CS<=NS;
End
Assign Door_Open = (CS==S_F1) || (CS==S_F2) || (CS==S_F3);
```

2. Consider the following variation on the traffic light controller problem. A North-South road intersects an East-West road. In addition to the Red/Yellow/Green traffic lights, the N-S road has green left-turn arrows. The arrows work as follows. With the traffic lights red in all direction, the N-S left turn arrows are illuminated Green. Then they turn yellow and finally they turn red. At this point, the N-S lights cycle Green/Yellow/Red. In the N-S direction, the Green Arrow time is 16 seconds and the Yellow Arrow time is 8 s. Overlapping with this is Red light time, which is 88 s. The Green light time is 24 s and the Yellow light time is 8 s. The Red Arrow time is what is left after the other arrows have been illuminated within the N-S cycle. The E-W lights are: Red 56 s, Green 56 s, and Yellow 8 s.
    (a) Draw a simple timing chart that shows the behavior of the N-S and E-W traffic lights and the Left Turn Arrow lights.





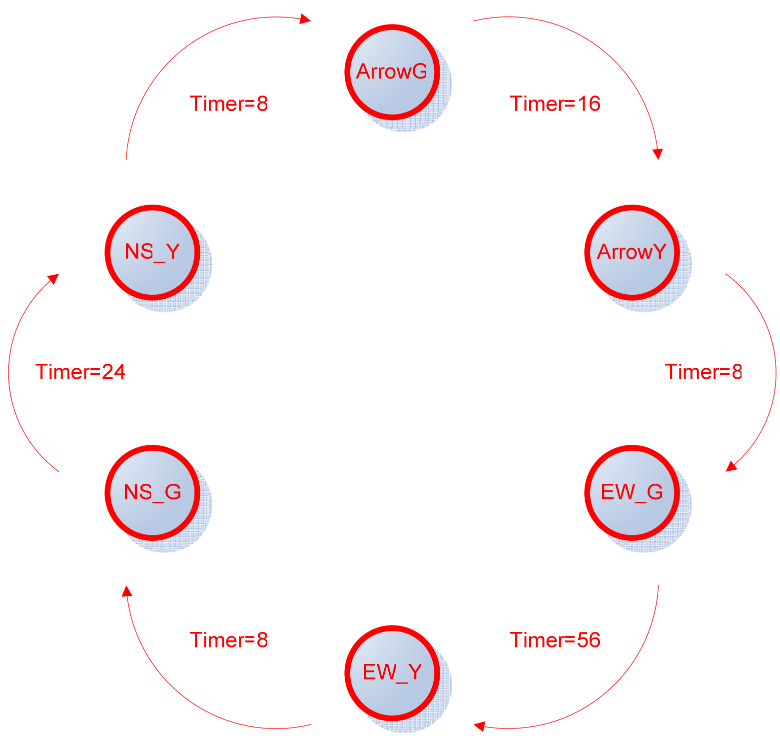
(b) Identify your inputs and outputs. What additional circuitry, like timers and flip-flops, do you need outside of the state machine?

Input: Timer

Output: NS_R, NS_Y, NS_G, EW_R, EW_Y, EW_G, L_R, L_Y, L_G (Signal for Each Light)

Need a timer to keep track of the elapsed time. The timer is formed with a counter with a one second clock. A comparator is then used to check the timer against the time for each state.

(c) Draw a symbolic state diagram. Make clear your assumptions, consistent with the specification above.



(d) Write "sketch" Verilog code for a Moore Machine implementation of this state diagram.

```
Always @ (*) begin
        NS = CS;
        Case (CS)
                Timer_Reset = 1'b0;
```

```
                NS = CS;

                ArrowG: begin
                        if (Timer==16)  begin
                                NS = ArrowY;
                                Timer_Reset = 1’b1;
                        End
                End
                ArrowY: begin
                        if (Timer==8)  begin
                                NS = EW_G;
                                Timer_Reset = 1’b1;
                        End
                End

                EW_G: begin
                        if (Timer==56)  begin
                                NS = EW_Y;
                                Timer_Reset = 1’b1;
                        End
                End

                EW_Y: begin
                        if (Timer==8)  begin
                                NS = NS_G;
                                Timer_Reset = 1’b1;
                        End
                End

                NS_G: begin
                        if (Timer==24)  begin
                                NS = NS_Y;
                                Timer_Reset = 1’b1;
                        End
                End

                NS_Y: begin
                        if (Timer==8)  begin
                                NS = ArrowG;
                                Timer_Reset = 1’b1;
                        End
                End

        EndCase
end

Always @ (posedge clk) begin
        If (Reset)
                CS<=S_F1;
        Else
                CS<=NS;
End

Assign Arrow_Green = (CS==ArrowG);
Assign Arrow_Yellow = (CS==ArrowY);
Assign Arrow_Red = ~(ArrowY || ArrowG);
```

Assign NS_Green = (CS==NS_G);
Assign NS_Yellow = (CS==NS_Y);
Assign NS_Red = ~(NS_Green || NS_Yellow);

Assign EW_Green = (CS==EW_G);
Assign EW_Yellow = (CS==EW_Y);
Assign EW_Red = ~(EW_Green || EW_Yellow);

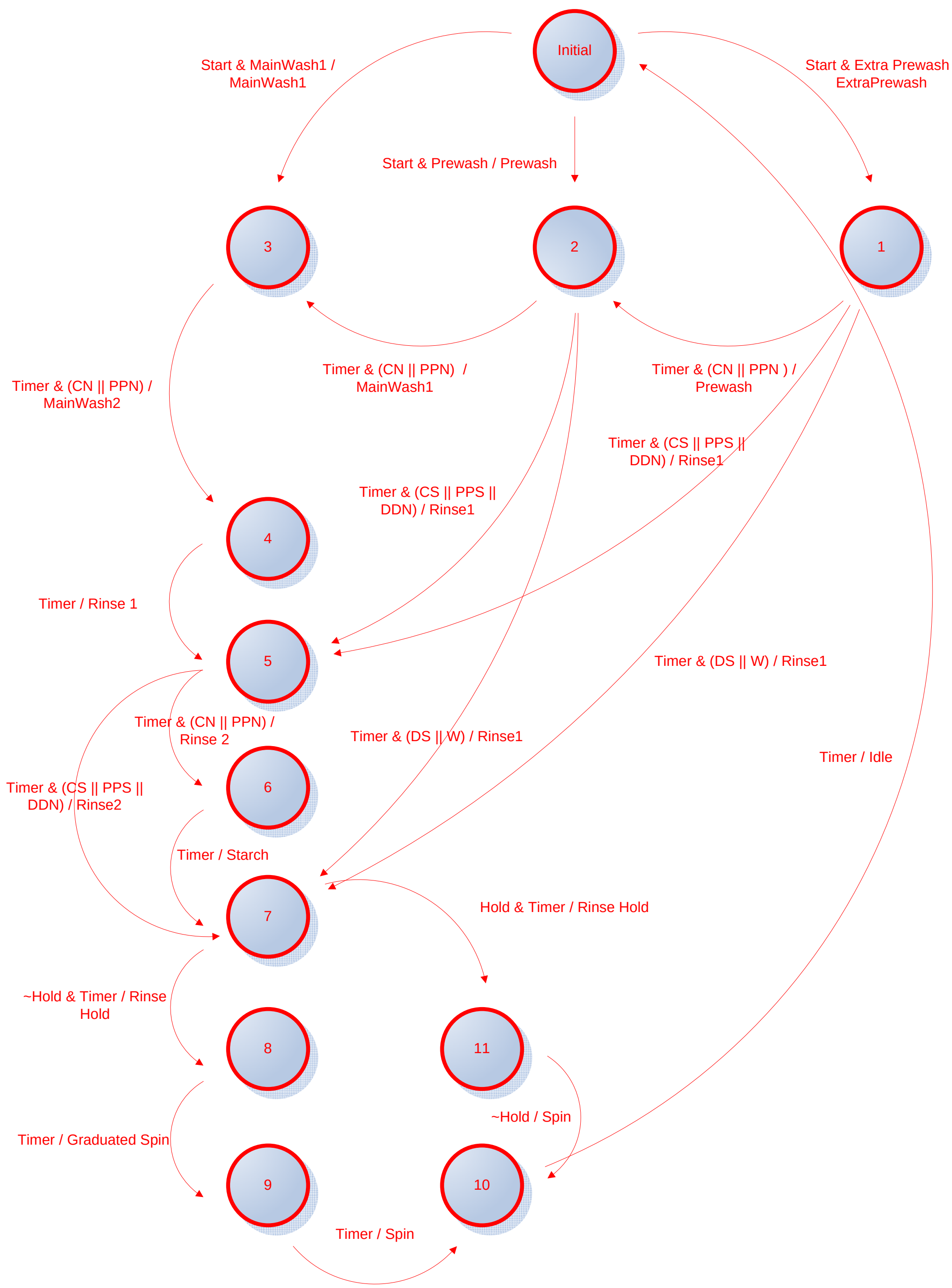
3. Professor Katz has a complicated washing machine at home. It can advance through the following states in the following sequence: Extra Prewash, Prewash, Main Wash 1, Main Wash 2, Rinse 1, Rinse 2, Rinse 3, Starch, Rinse Hold, Graduated Spin, and Spin. The user selectively positions a dial to Extra Prewash, Prewash, or Main Wash 1 to indicate the initial state for the wash. When the Start button is pressed, the cycle begins in the selected initial state. The machine has a "program control" to indicate the kind of fabrics being washed: Cotton Normal, Cotton Short, Permanent Press Normal, Permanent Press Short, Delicates Normal, Delicates Short, and Woolens. Normal cotton and permanent press programs cycle through every state following the initial state. Short cotton and permanent press programs and the Delicates Normal program pass through Main Wash 1, skip Main Wash 2, enter Rinse 1 and 2, and skip Rinse 3. Delicates Short and Woolens are similar but also skip the second rinse. Finally if the Short Spin/Rinse Hold button is depressed, the program holds in the Rinse Hold state until the button is released, and then advances directly to Spin skipping the Graduated Spin.
   (a) Identify your inputs, outputs, and name and describe your states. What additional circuitry, like timers and flip-flops, do you need outside of the state machine?

   Inputs: ExtraPrewash, Prewash, MainWash1, Start, Hold
   CN, CS, PPN, PPS, DN, DS, W (Type of Material)

   Outputs: State Machine In

   Need a timer that signals the end of the current cycle.

   (b) Draw a symbolic state diagram for your design, labeling all state transitions. Indicate any additional assumptions you are making.



(c) Write "sketch" Verilog code for a Mealy Machine implementation of this state diagram.

```
Always @ (*) Begin
        CS = NS;

        Case (CS)
                Initial:
```

```
		If (Start) begin
			If (ExtraPrewash) begin
				NS = 1;
				State = ExtraPrewash
			End
			If (Prewash) begin
				NS = 2;
				State = Prewash;
			End
			If (MainWash1) begin
				NS = 3;
				State = MainWash1;
			End
		End
	1:	if (Timer) begin
			If (CN || PPN) begin
				NS = 2;
				State = Prewash;
			End
			If (CS || PPS || DDN) begin
				NS = 5;
				State = Rinse1;
			End
			If (DS || W) begin
				NS = 7;
				State = Rinse1;
			End
		End
	2:	if (Timer) begin
			If (CN || PPN) begin
				NS = 3;
				State = MainWash1;
			End
			If (CS || PPS || DDN) begin
				NS = 5;
				State = Rinse1;
			End
			If (DS || W) begin
				NS = 7;
				State = Rinse1;
			End
		End
	3:	If (Timer && (CN || PPN) begin
			NS = 4;
			State = MainWash2;
		End
	4:	If (Timer) begin
			NS = 5;
			State = Rinse1;
		End
	5:	If (Timer) begin
			If (CN || PPN) begin
				NS = 6;
				State = Rinse2;
			End
			If (CS || PPS || DDN) begin
```

```
                    NS = 7;
                    State = Rinse2;
                End
            End
    6:      If (Timer) begin
                NS = 7;
                State = Starch;
            End
    7:      If (~Hold && Timer) begin
                NS = 8
                State = RinseHold;
            End
            If (Hold && Timer) begin
                NS = 11;
                State = RinseHold;
            End
    8:      If (Timer) begin
                NS = 9
                State = GraduatedSpin;
            End
    9:      If (Timer) begin
                NS = 10;
                State = Spin;
            End
    10:     If (Timer) begin
                NS = Initial;
                State = Idle;
            End
    11:     If (~Hold) begin
                NS = 10
                State = Spin;
            End

EndCase
End
```