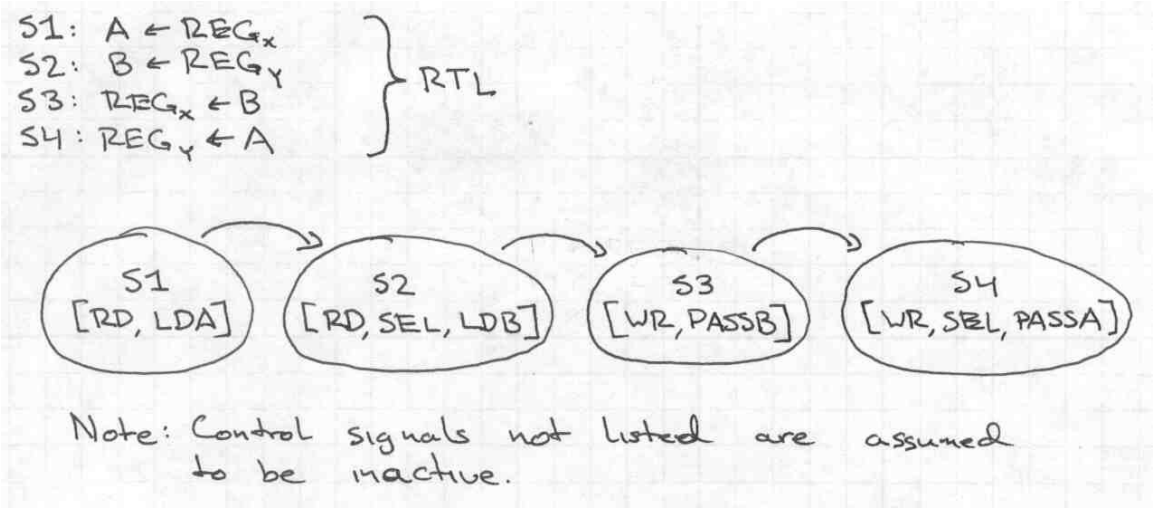
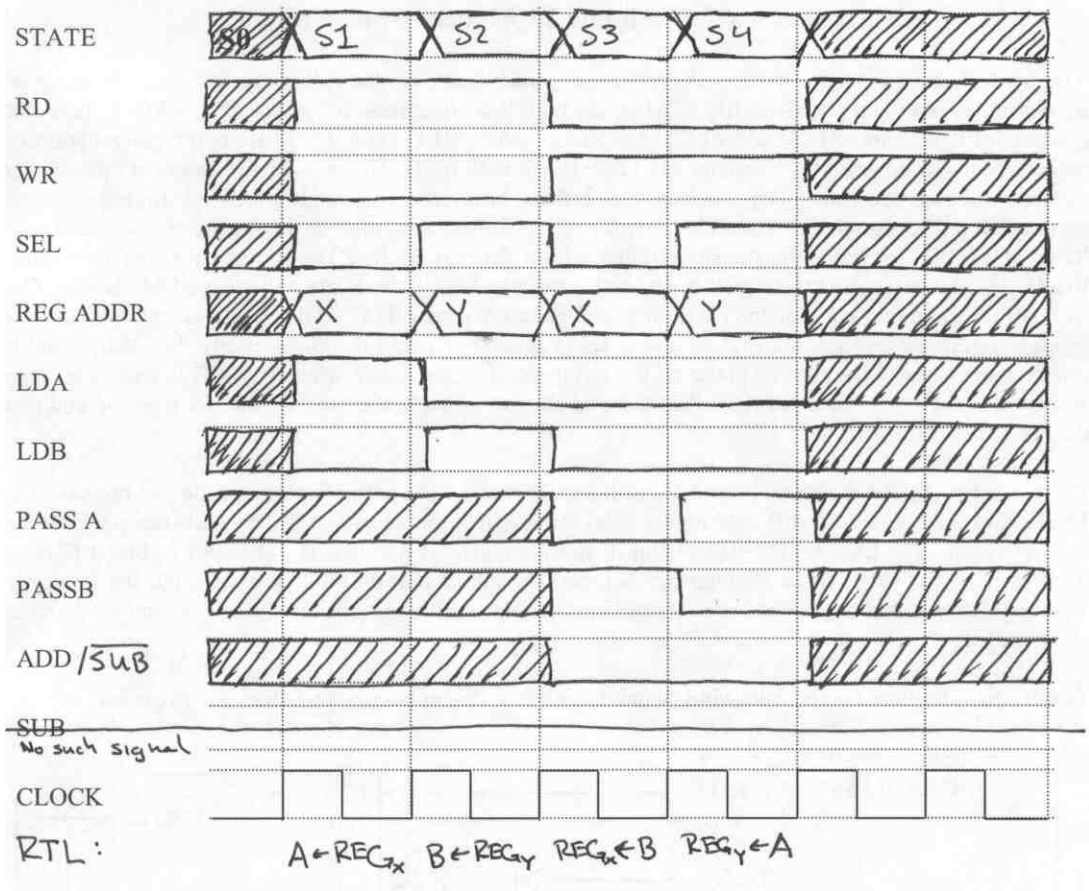
CS150 Spring 2004
Problem Set #7 Solutions

Greg Gibeling




```
module ControlFSM(Clock, Reset, RD, WR, SEL, LDA, LDB, ADD, PASSA, PASSB);
        input           Clock, Reset;
        output          RD, WR;
        output          SEL;
        output          LDA, LDB;
        output          ADD, PASSA, PASSB;

        parameter       STATE_S0 =                      3'h0,
                        STATE_S1 =                      3'h1,
                        STATE_S2 =                      3'b2,
                        STATE_S3 =                      3'h3,
                        STATE_S4 =                      3'h4,
                        STATE_... =                     ...;
                        // There could be lots of states...

        reg     [2:0]   CurrentState, NextState;
        reg             RD, WR;
        reg             SEL;
        reg             LDA, LDB;
        reg             ADD, PASSA, PASSB;

        always @ (posedge Clock) begin
                if (Reset) CurrentState <=              STATE_S0;
                else CurrentState <=                    NextState;
        end

        always @ (CurrentState or ...) begin
                NextState =                             CurrentState;
                RD =                                    1'b0;
                WR =                                    1'b0;
                SEL =                                   1'bx;
                LDA =                                   1'b0;
                LDB =                                   1'b0;
                ADD =                                   1'bx;
                PASSA =                                 1'bx;
                PASSB =                                 1'bx;

                case (CurrentState)
                        STATE_S0: begin
                                // Somehow we get into S1, for this
```

```
                        // problem we don't care how
                        if (...) NextState =   STATE_S1;
                    end
                    STATE_S1: begin
                        RD =                1'b1;
                        SEL =               1'b0;
                        LDA =               1'b1;
                        NextState =         STATE_S2;
                    end
                    STATE_S2: begin
                        RD =                1'b1;
                        SEL =               1'b1;
                        LDB =               1'b1;
                        NextState =         STATE_S3;
                    end
                    STATE_S3: begin
                        WR =                1'b1;
                        SEL =               1'b0;
                        ADD =               1'b0;
                        PASSA =             1'b0;
                        PASSB =             1'b1;
                        NextState =         STATE_S4;
                    end
                    STATE_S4: begin
                        WR =                1'b1;
                        SEL =               1'b1;
                        ADD =               1'b0;
                        PASSA =             1'b1;
                        PASSB =             1'b0;
                        // Normaly we would say where to go
                        // next but for this problem we don't
                        // really care or know
                        NextState =         ...;
                    end
                    default: begin
                        NextState =         3'hX;
                        RD =                1'bx;
                        WR =                1'bx;
                        SEL =               1'bx;
                        LDA =               1'bx;
                        LDB =               1'bx;
                        ADD =               1'bx;
                        PASSA =             1'bx;
                        PASSB =             1'bx;
                    end
                endcase
        end

endmodule
```


STATE
S0
S1
S2
S3
S4
RD
WR
SEL
REG ADDR
X
Y
X
Y
LDA
LDB
PASS A
PASSB
ADD /SUB
SUB
No such signal
CLOCK
RTL :
A←REGx B←REGy REGx←B REGy←A