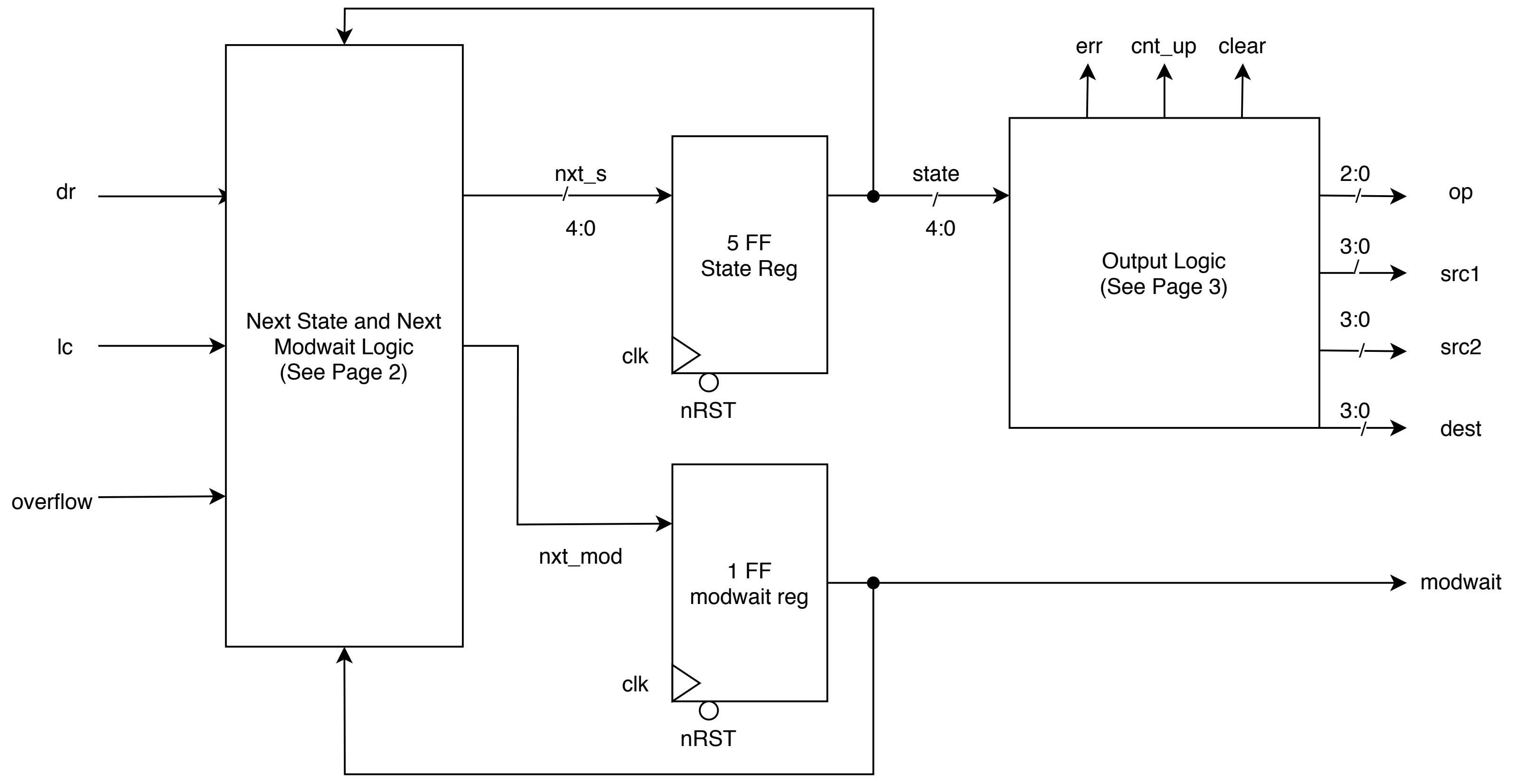
Controller RTL Diagram (6FF)



NEXT STATE LOGIC

```
nxt_s = state;

case (state)
    IDLE:  if (dr) nxt_s = STORE;
           else if (lc) nxt_s = LOAD_F0;
    STORE: if (dr) nxt_s = ZERO;
           else nxt_s = EIDLE;
    ZERO : nxt_s = SORT1;
    SORT1: nxt_s = SORT2;
    SORT2: nxt_s = SORT3;
    SORT3: nxt_s = SORT4;
    SORT4: nxt_s = MUL1;
    MUL1 : nxt_s = ADD1;
    ADD1 : if (overflow) nxt_s = EIDLE;
           else nxt_s = MUL2;
    MUL2 : nxt_s = SUB1;
    SUB1 : if (overflow) nxt_s = EIDLE;
           else nxt_s = MUL3;
    MUL3 : nxt_s = ADD2;
    ADD2 : if (overflow) nxt_s = EIDLE;
           else nxt_s = MUL4;
    MUL4 : nxt_s = SUB2;
    SUB2 : if (overflow) nxt_s = EIDLE;
           else nxt_s = IDLE;
    EIDLE: if (dr) nxt_s = STORE;
    LOAD_F0: nxt_s = WAIT1;
    WAIT1  : if (lc) nxt_s = LOAD_F1;
    LOAD_F1: nxt_s = WAIT2;
    WAIT2  : if (lc) nxt_s = LOAD_F2;
    LOAD_F2: nxt_s = WAIT3;
    WAIT3  : if (lc) nxt_s = LOAD_F3;
    LOAD_F3: nxt_s = IDLE;
    default: nxt_s = IDLE;
endcase
```

NEXT MODWAIT LOGIC

```
nxt_mod = 1;

case (state)
    IDLE:  if (!dr && !lc) nxt_mod = 0;
    STORE: if (!dr)        nxt_mod = 0;
    ADD1 : if (overflow)   nxt_mod = 0;
    SUB1 : if (overflow)   nxt_mod = 0;
    ADD2 : if (overflow)   nxt_mod = 0;
    SUB2 :                 nxt_mod = 0;
    EIDLE: if (!dr)        nxt_mod = 0;
    LOAD_F0:               nxt_mod = 0;
    LOAD_F1:               nxt_mod = 0;
    LOAD_F2:               nxt_mod = 0;
    LOAD_F3:               nxt_mod = 0;
    default: nxt_mod = 1;
endcase
```

```
OUTPUT LOGIC
err = state == EIDLE;
cnt_up = state == ZERO;
clear = state == LOAD_F0;
op = '0;
src1 = '0;
src2 = '0;
dest = '0;

case (state)
    STORE: begin
           op = LOAD1;
           dest = 4'd5;
           end
    ZERO : begin
           op = COPY;
           src1 = 4'd11;
           dest = 4'd0;
           end
    SORT1: begin
           op = COPY;
           src1 = 4'd2;
           dest = 4'd1;
           end
    SORT2: begin
           op = COPY;
           src1 = 4'd3;
           dest = 4'd2;
           end
    SORT3: begin
           op = COPY;
           src1 = 4'd4;
           dest = 4'd3;
           end
    SORT4: begin
           op = COPY;
           src1 = 4'd5;
           dest = 4'd4;
           end
    MUL1 : begin
           op = MUL;
           src1 = 4'd1;
           src2 = 4'd6;
           dest = 4'd10;
           end
    ADD1 : begin
           op = ADD;
           src1 = 4'd0;
           src2 = 4'd10;
           dest = 4'd0;
           end
    MUL2 : begin
           op = MUL;
           src1 = 4'd2;
           src2 = 4'd7;
           dest = 4'd10;
           end
    SUB1 : begin
           op = SUB;
           src1 = 4'd0;
           src2 = 4'd10;
           dest = 4'd0;
           end
    MUL3 : begin
           op = MUL;
           src1 = 4'd3;
           src2 = 4'd8;
           dest = 4'd10;
           end
    ADD2 :  begin
            op = ADD;
            src1 = 4'd0;
            src2 = 4'd10;
            dest = 4'd0;
            end
    MUL4 :  begin
            op = MUL;
            src1 = 4'd4;
            src2 = 4'd9;
            dest = 4'd10;
            end
    SUB2  : begin
            op = SUB;
            src1 = 4'd0;
            src2 = 4'd10;
            dest = 4'd0;
            end
    LOAD_F0: begin
             op = LOAD2;
             dest = 4'd9;
             end
    LOAD_F1: begin
             op = LOAD2;
             dest = 4'd8;
             end
    LOAD_F2: begin
             op = LOAD2;
             dest = 4'd7;
             end
    LOAD_F3: begin
             op = LOAD2;
             dest = 4'd6;
             end
endcase
```