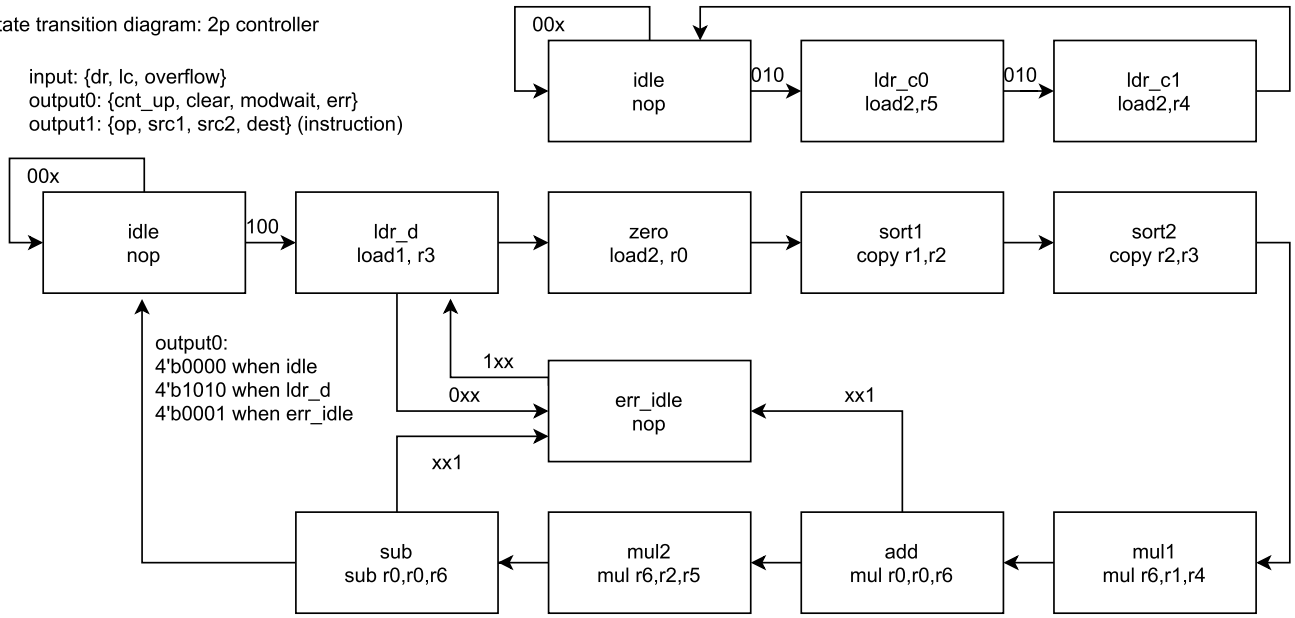
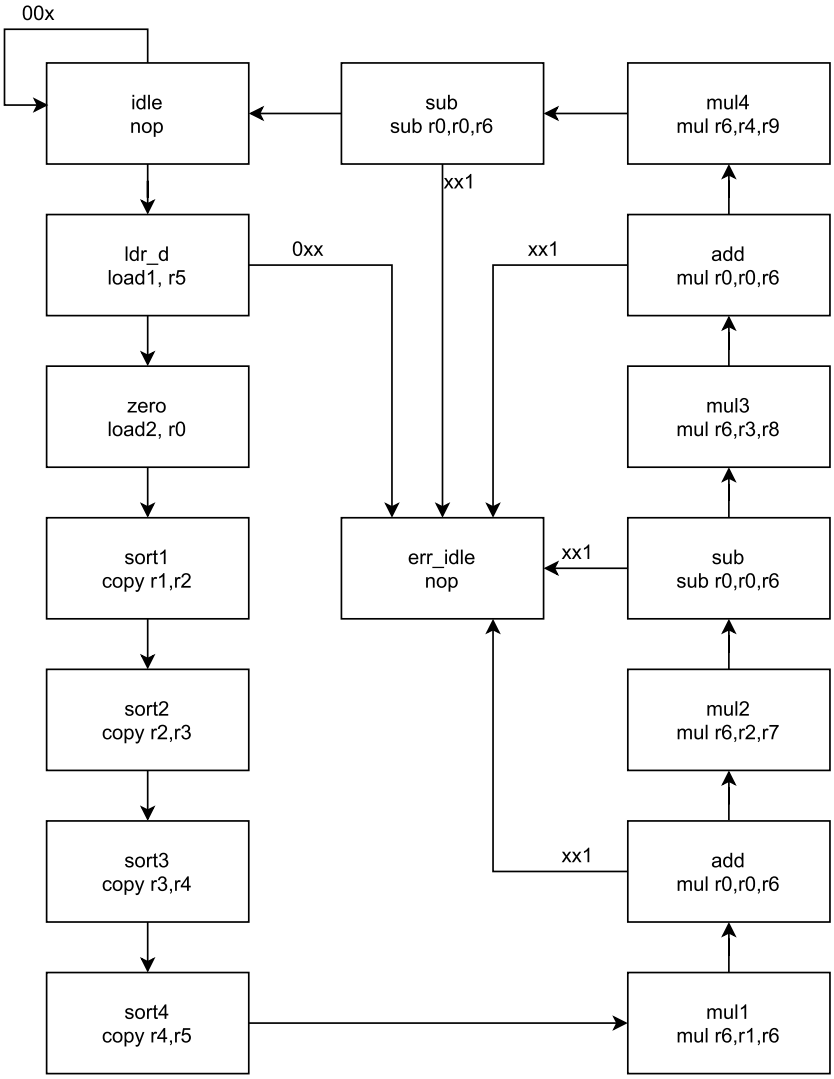
state transition diagram: 2p controller

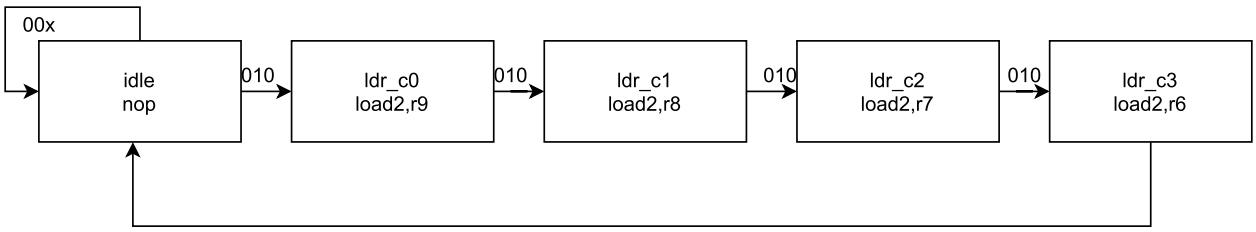
input: {dr, lc, overflow}
output0: {cnt_up, clear, modwait, err}
output1: {op, src1, src2, dest} (instruction)



persudo code; 4p FIR

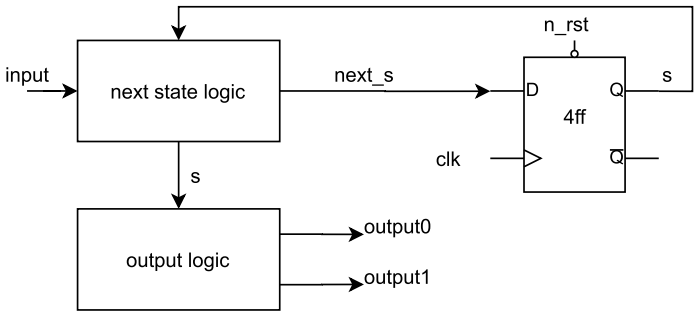
```
idle:
    if(!data_ready) goto idle
store:
    if(!data_ready) goto erridle
    reg[5] = data
    err = 0
zero:
    reg[0] = 0
stream:
    reg[1] = reg[2]
    reg[2] = reg[3]
    reg[3] = reg[4]
    reg[4] = reg[5]
mul1:
    reg[10] = reg[1] * reg[6]
add1:
    reg[0] = reg[0] + reg[10]
    if(V) goto erridle
mul2:
    reg[10] = reg[2] * reg[7]
sub2:
    reg[0] = reg[0] - reg[10]
    if(V) goto erridle
mul3:
    reg[10] = reg[3] * reg[8]
add3:
    reg[0] = reg[0] + reg[10]
    if(V) goto erridle
muls4:
    reg[10] = reg[4] * reg[9]
sub4:
    reg[0] = reg[0] - reg[10]
    if(V) goto erridle
    else goto idle
erridle:
    err = 1
    if(data_ready) goto store
    if(!data_ready) goto erridle
```





RTL diagram: controller

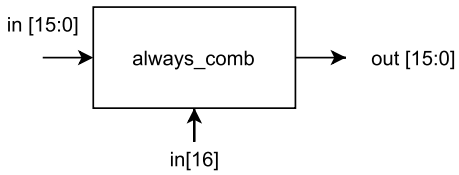
note: the output logic and next state logic has been explained in state transition diagram.
moore machine's output only depends on the current state.



input: {dr, lc, overflow}
output0: {cnt_up, clear, modwait, err}
output1: {op, src1, src2, dest} (instruction)

output0:
4'b0000 when idle
4'b1010 when ldr_d
4'b0001 when err_idle

output1 is the opcode, src1, src2, dest,
which is converted from the instruction
shown in state transition diagram.



note:
in[16] serves as EN signal of this module. when in[16] is 1, the input is a negative number.
thus the conversion is need. otherwise the output is in [15:0].
assign out = (in [16]) ? ~in [15:0] + 1 : in[15:0];