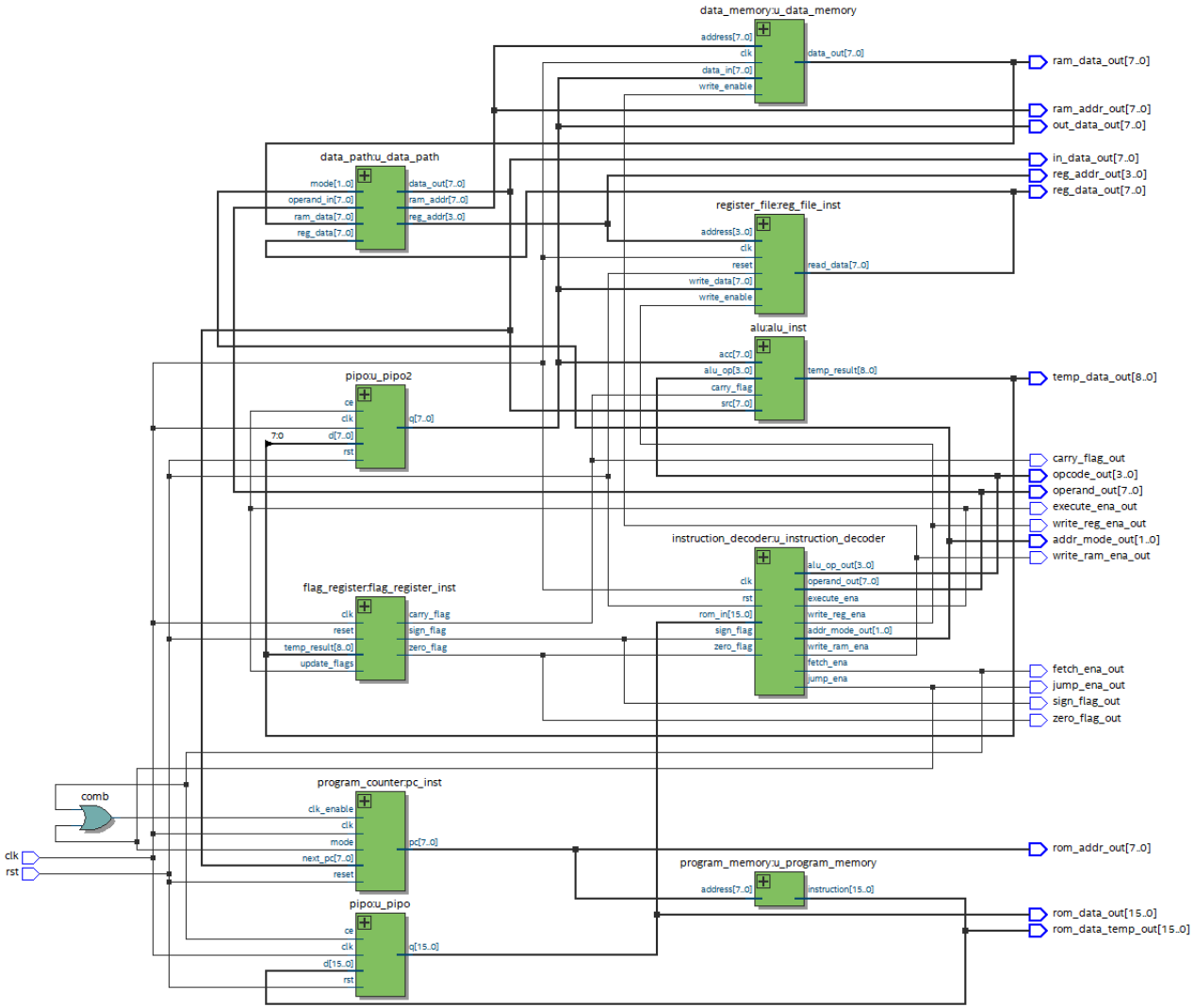
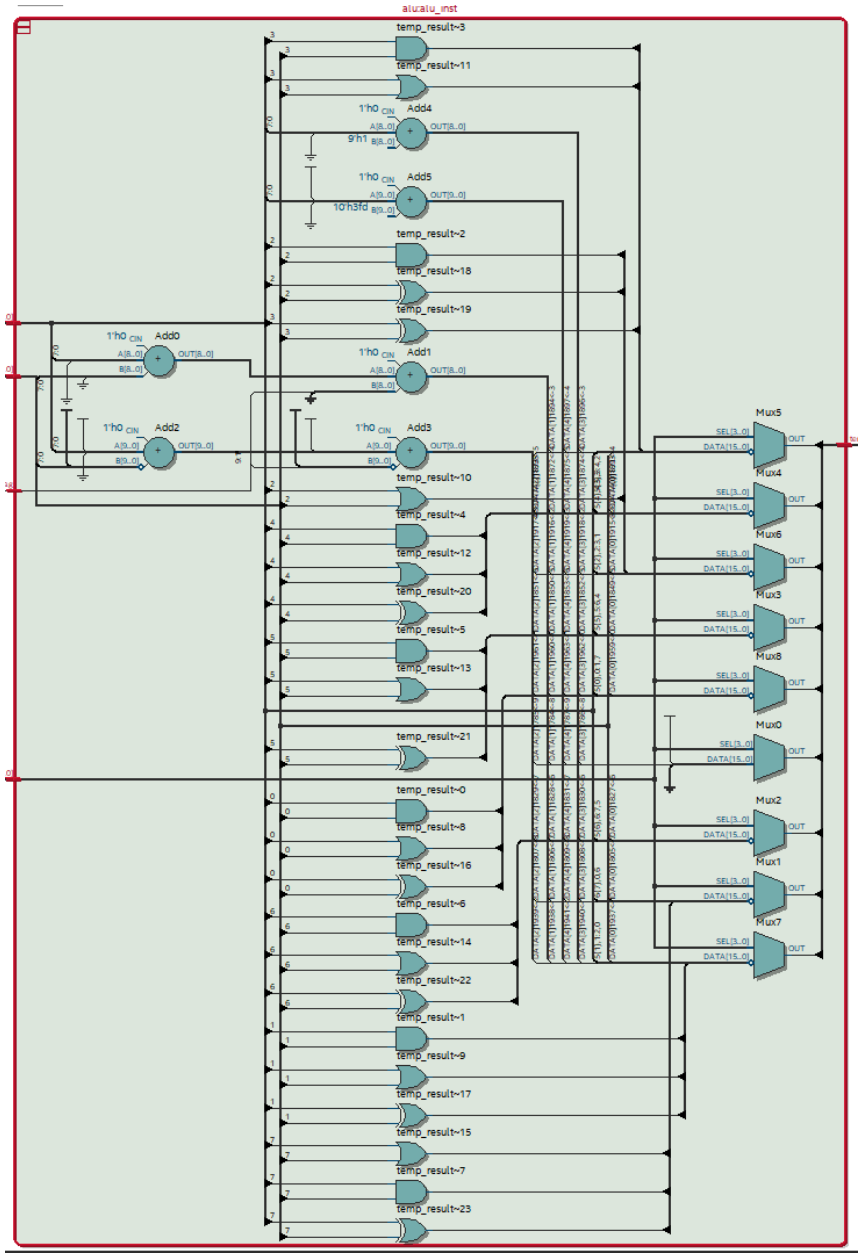
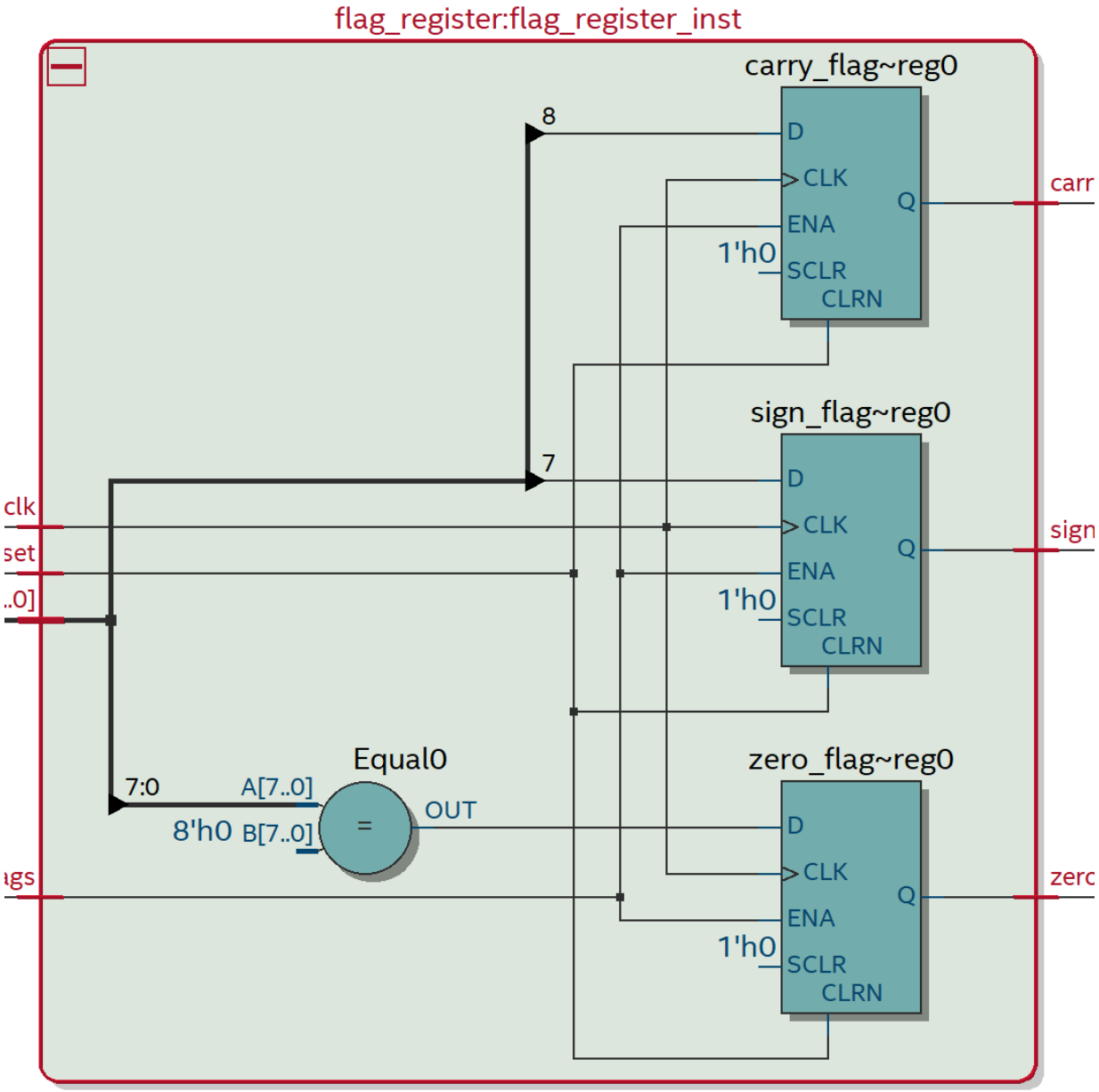
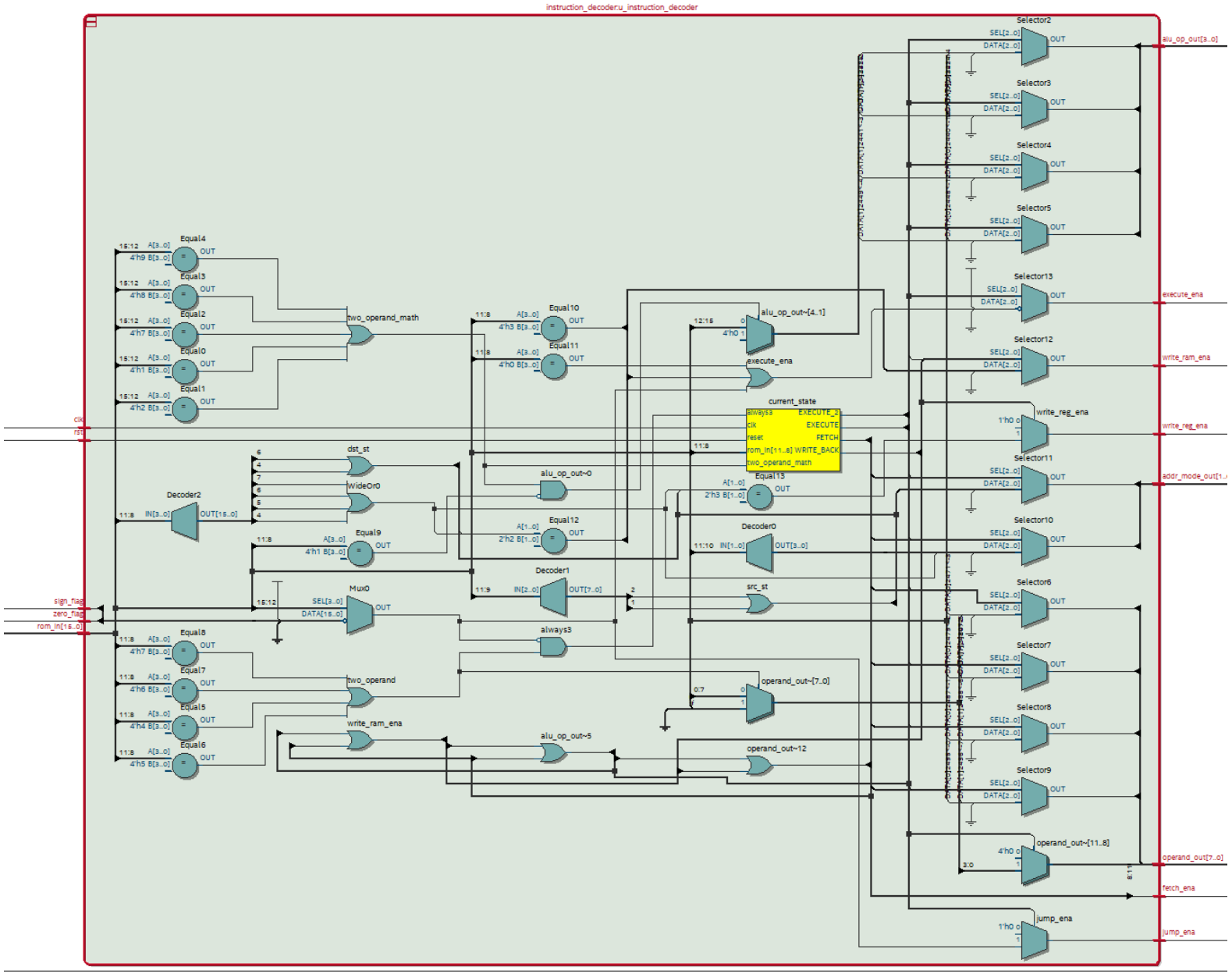
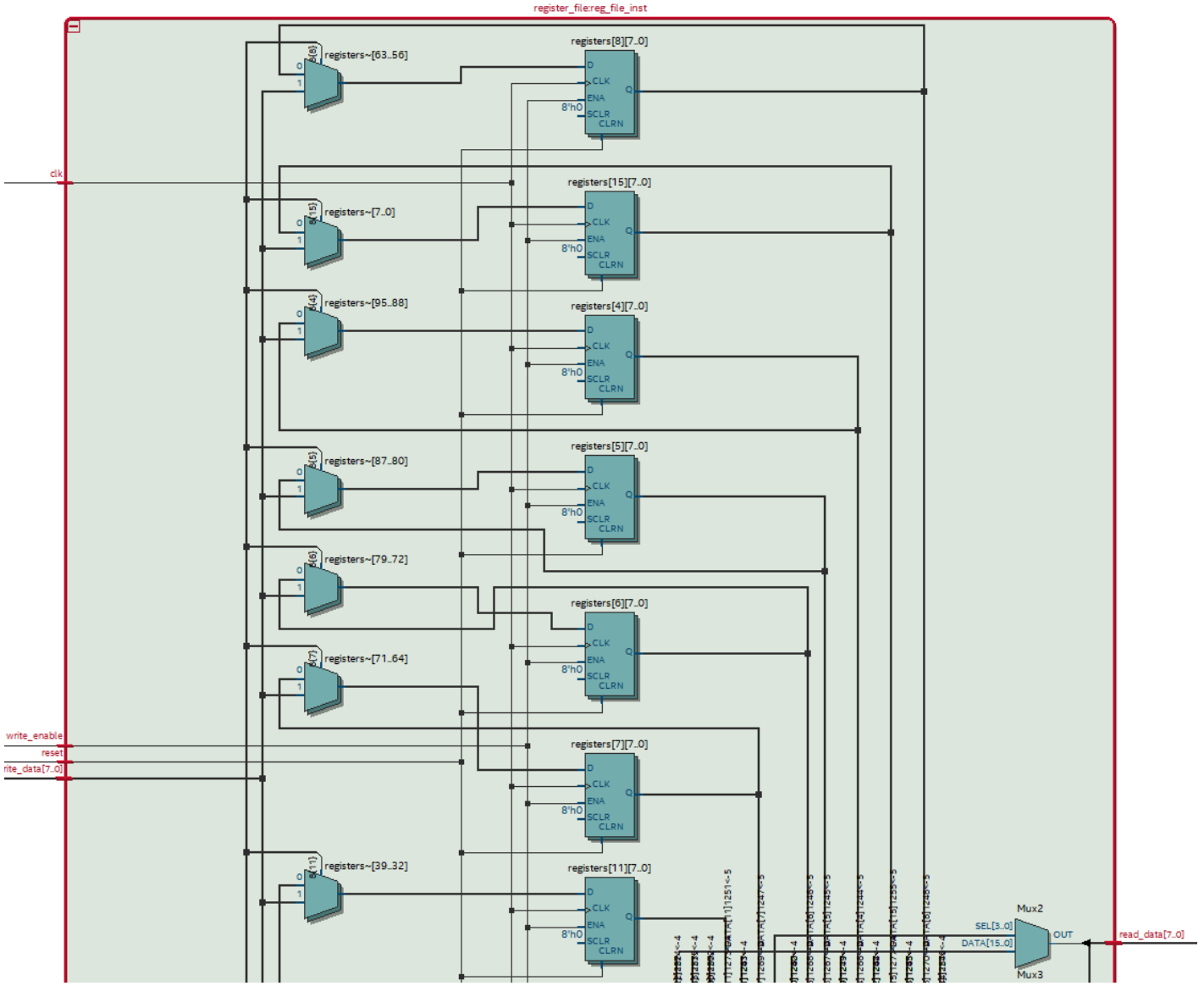

data_memory:u_data_memory
address[7..0]
clk
data_in[7..0]
write_enable
data_out[7..0]
ram_data_out[7..0]
ram_addr_out[7..0]
out_data_out[7..0]
data_path:u_data_path
mode[1..0]
operand_in[7..0]
ram_data[7..0]
reg_data[7..0]
data_out[7..0]
ram_addr[7..0]
reg_addr[3..0]
in_data_out[7..0]
reg_addr_out[3..0]
reg_data_out[7..0]
register_file:reg_file_inst
address[3..0]
clk
reset
write_data[7..0]
write_enable
read_data[7..0]
alu:alu_inst
acc[7..0]
alu_op[3..0]
carry_flag
src[7..0]
temp_result[8..0]
temp_data_out[8..0]
pipo:u_pipo2
ce
clk
d[7..0]
rst
q[7..0]
7:0
carry_flag_out
opcode_out[3..0]
operand_out[7..0]
execute_ena_out
write_reg_ena_out
addr_mode_out[1..0]
write_ram_ena_out
instruction_decoder:u_instruction_decoder
clk
rst
rom_in[15..0]
sign_flag
zero_flag
alu_op_out[3..0]
operand_out[7..0]
execute_ena
write_reg_ena
addr_mode_out[1..0]
write_ram_ena
fetch_ena
jump_ena
flag_register:flag_register_inst
clk
reset
temp_result[8..0]
update_flags
carry_flag
sign_flag
zero_flag
fetch_ena_out
jump_ena_out
sign_flag_out
zero_flag_out
comb
clk
rst
program_counter:pc_inst
clk_enable
clk
mode
next_pc[7..0]
reset
pc[7..0]
rom_addr_out[7..0]
program_memory:u_program_memory
address[7..0]
instruction[15..0]
pipo:u_pipo
ce
clk
d[15..0]
rst
q[15..0]
rom_data_out[15..0]
rom_data_temp_out[15..0]

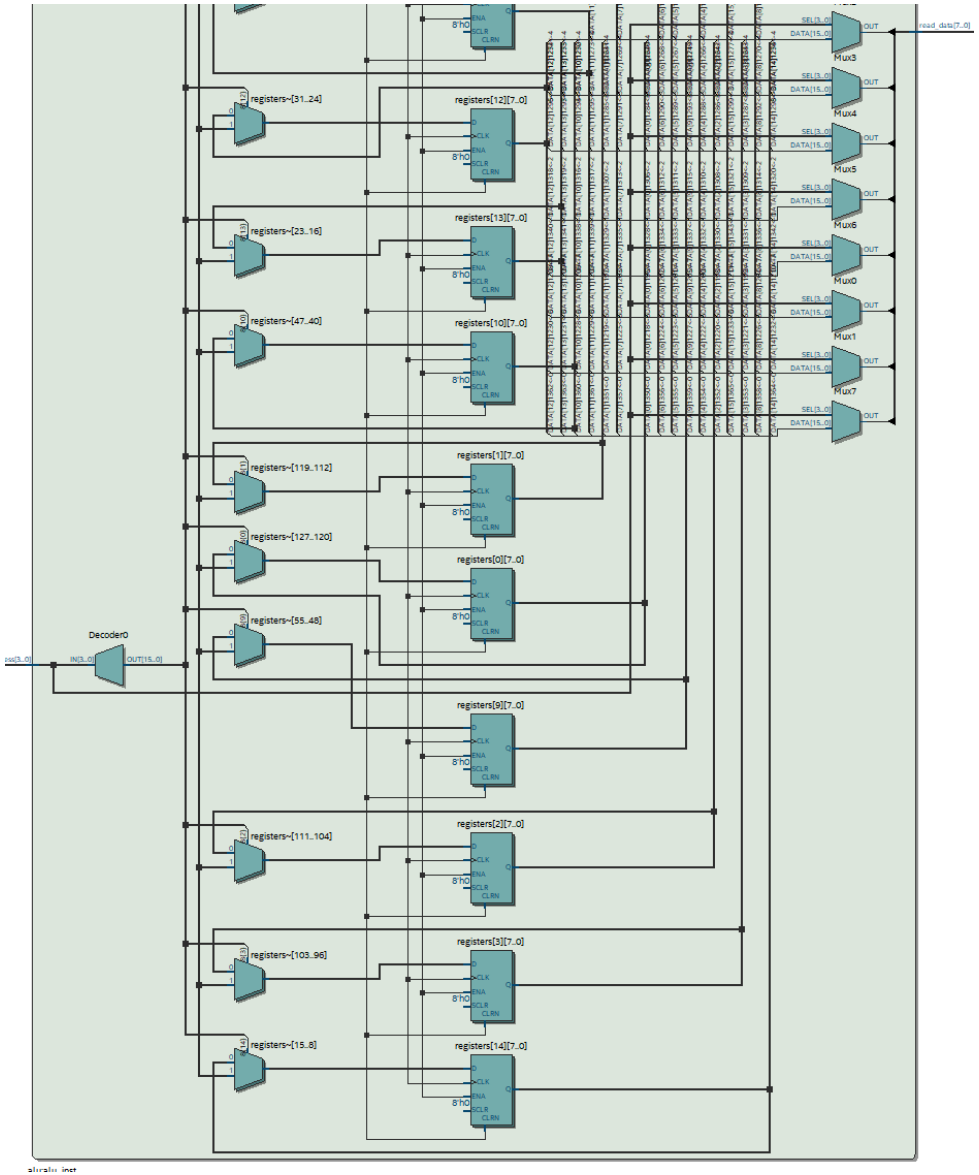

alu:alu_inst
temp_result~3
temp_result~11
1'h0 CIN
Add4
A[8..0]
B[8..0]
9'h1
OUT[8..0]
1'h0 CIN
Add5
A[9..0]
B[9..0]
10'h3fd
OUT[9..0]
temp_result~2
temp_result~18
temp_result~19
1'h0 CIN
Add0
A[8..0]
B[8..0]
OUT[8..0]
1'h0 CIN
Add1
A[8..0]
B[8..0]
OUT[8..0]
1'h0 CIN
Add2
A[9..0]
B[9..0]
OUT[9..0]
1'h0 CIN
Add3
A[9..0]
B[9..0]
OUT[9..0]
temp_result~10
temp_result~4
temp_result~12
temp_result~20
temp_result~5
temp_result~13
temp_result~21
temp_result~0
temp_result~8
temp_result~16
temp_result~6
temp_result~14
temp_result~22
temp_result~1
temp_result~9
temp_result~17
temp_result~15
temp_result~7
temp_result~23
Mux5
Mux4
Mux6
Mux3
Mux8
Mux0
Mux2
Mux1
Mux7
SEL[3..0]
DATA[15..0]
OUT


flag_register:flag_register_inst
carry_flag~reg0
sign_flag~reg0
zero_flag~reg0
Equal0
8
7
7:0
A[7..0]
8'h0
B[7..0]
=
OUT
D
CLK
ENA
SCLR
CLRN
Q
1'h0
clk
set
carr
sign
zerc

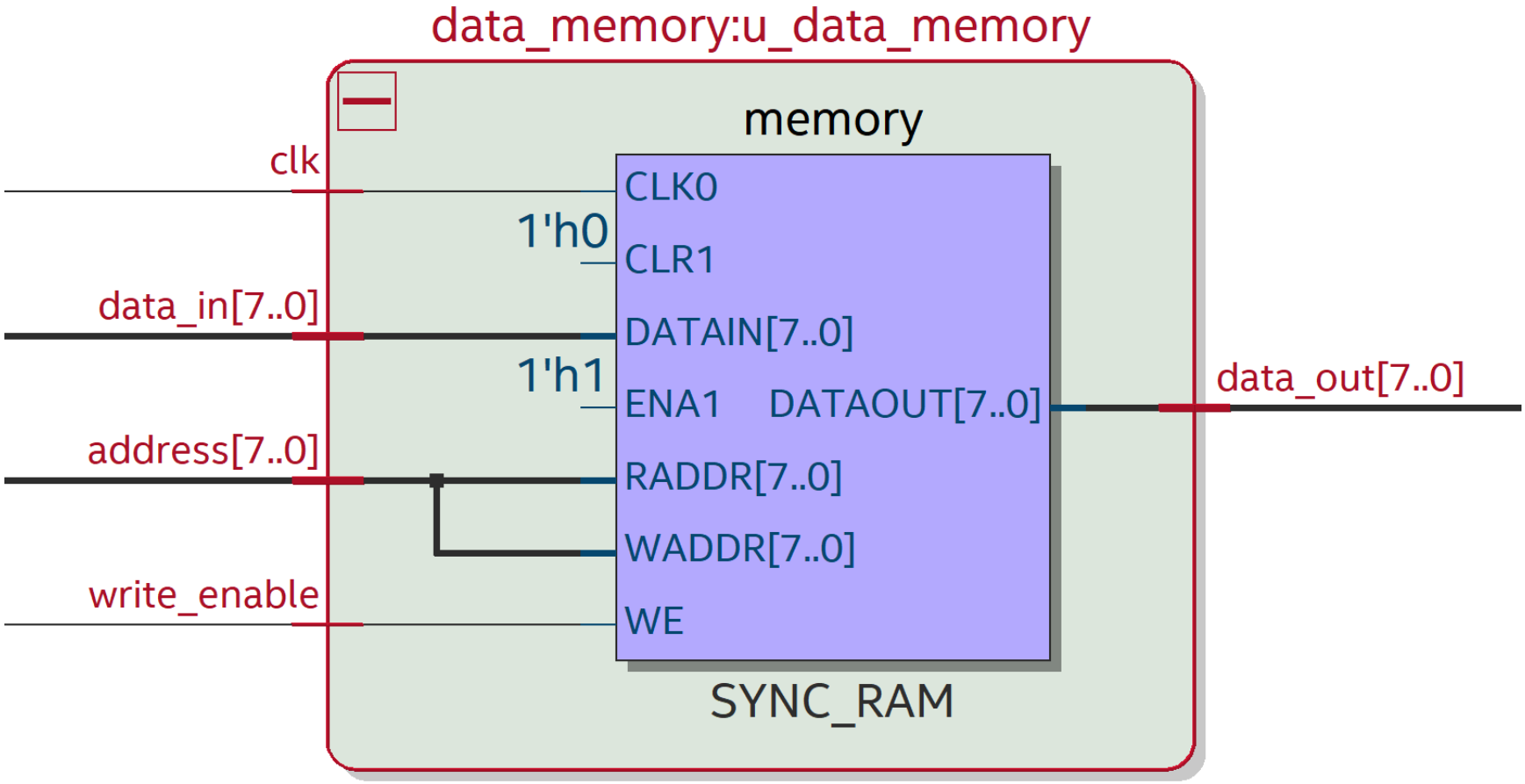

instruction_decoder:u_instruction_decoder
clk
rst
sign_flag
zero_flag
rom_in[15..0]
Equal4
Equal3
Equal2
Equal0
Equal1
two_operand_math
Equal10
Equal11
alu_op_out~[4..1]
execute_ena
current_state
always3
EXECUTE_S
EXECUTE
FETCH
WRITE_BACK
two_operand_math
Equal13
dst_st
WideOr0
alu_op_out~0
Decoder2
Equal9
Equal12
Decoder0
Decoder1
MUX0
src_st
always3
Equal8
Equal7
Equal5
Equal6
two_operand
write_ram_ena
alu_op_out~5
operand_out~[7..0]
operand_out~12
Selector2
Selector3
Selector4
Selector5
Selector13
Selector12
write_reg_ena
Selector11
Selector10
Selector6
Selector7
Selector8
Selector9
operand_out~[11..8]
jump_ena
alu_op_out[3..0]
execute_ena
write_ram_ena
write_reg_ena
addr_mode_out[1..
operand_out[7..0]
fetch_ena
jump_ena

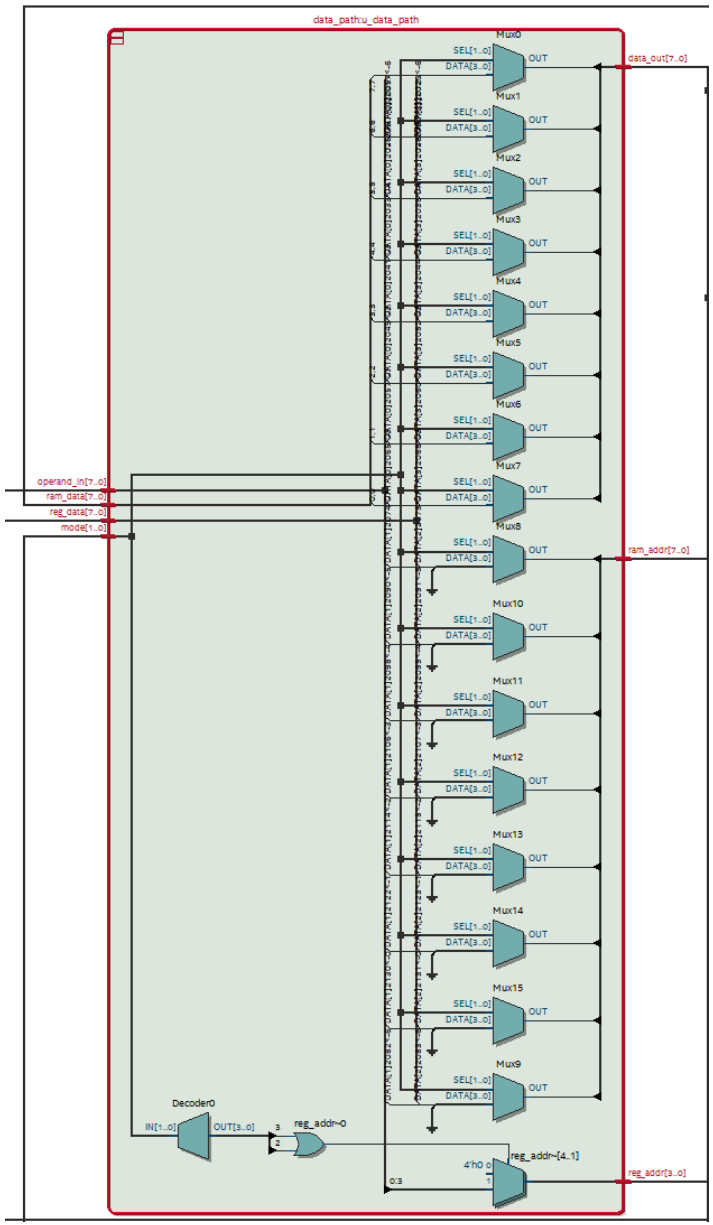

register_file:reg_file_inst
registers~[63..56]
registers[8][7..0]
clk
registers~[7..0]
registers[15][7..0]
registers~[95..88]
registers[4][7..0]
registers~[87..80]
registers[5][7..0]
registers~[79..72]
registers[6][7..0]
registers~[71..64]
write_enable
reset
registers[7][7..0]
registers~[39..32]
registers[11][7..0]
D
CLK
ENA
SCLR
CLRN
Q
8'h0
Mux2
SEL[3..0]
DATA[15..0]
OUT
Mux3
read_data[7..0]

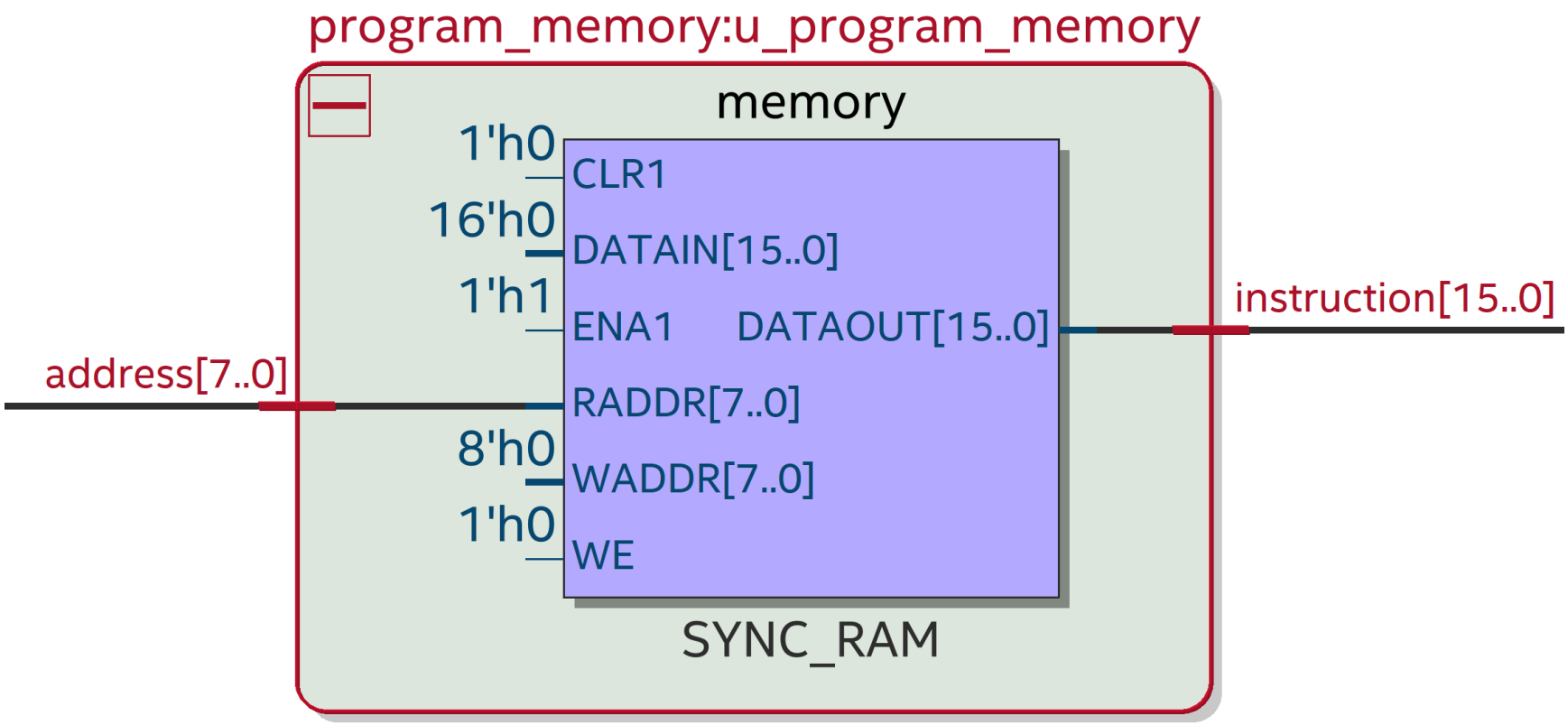
Reg_file 1/2

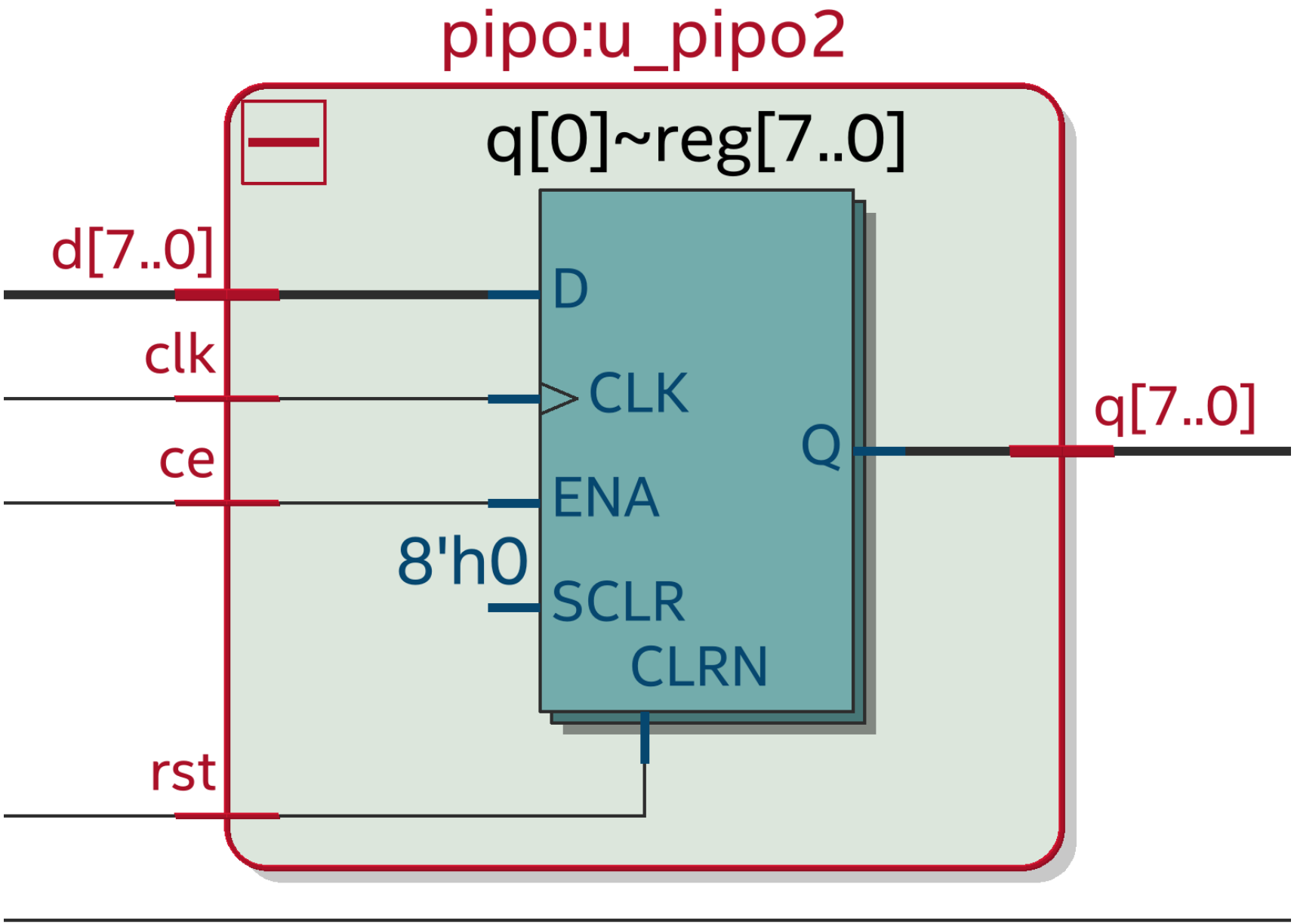

Decoder0
registers~[31..24]
registers~[23..16]
registers~[47..40]
registers~[119..112]
registers~[127..120]
registers~[55..48]
registers~[111..104]
registers~[103..96]
registers~[15..8]
registers[12][7..0]
registers[13][7..0]
registers[10][7..0]
registers[1][7..0]
registers[0][7..0]
registers[9][7..0]
registers[2][7..0]
registers[3][7..0]
registers[14][7..0]
Mux3
Mux4
Mux5
Mux6
Mux0
Mux1
Mux7

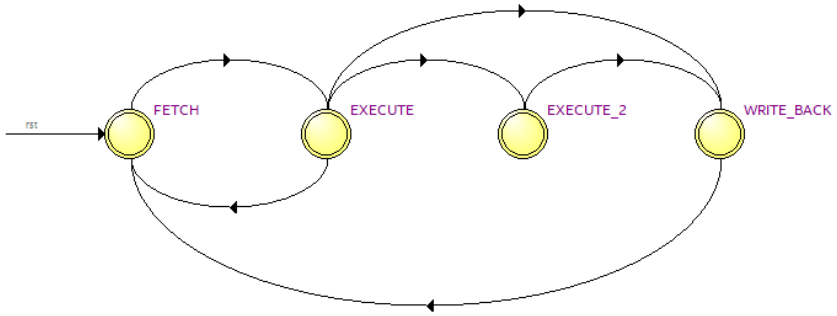
Reg_file 2/2


data_memory:u_data_memory
memory
clk
1'h0
CLK0
CLR1
data_in[7..0]
DATAIN[7..0]
1'h1
ENA1
DATAOUT[7..0]
data_out[7..0]
address[7..0]
RADDR[7..0]
WADDR[7..0]
write_enable
WE
SYNC_RAM


data_path:u_data_path
operand_in[7..0]
ram_data[7..0]
reg_data[7..0]
mode[1..0]
data_out[7..0]
ram_addr[7..0]
reg_addr[3..0]
Mux0
Mux1
Mux2
Mux3
Mux4
Mux5
Mux6
Mux7
Mux8
Mux10
Mux11
Mux12
Mux13
Mux14
Mux15
Mux9
SEL[1..0]
DATA[3..0]
OUT
Decoder0
IN[1..0]
OUT[3..0]
reg_addr~0
reg_addr~[4..1]
4'h0
0:3


program_memory:u_program_memory
memory
1'h0
CLR1
16'h0
DATAIN[15..0]
1'h1
ENA1
DATAOUT[15..0]
instruction[15..0]
address[7..0]
RADDR[7..0]
8'h0
WADDR[7..0]
1'h0
WE
SYNC_RAM


pipo:u_pipo2
q[0]~reg[7..0]
d[7..0]
clk
ce
rst
D
CLK
ENA
8'h0
SCLR
CLRN
Q
q[7..0]



	Source State	Destination State	Condition
1	EXECUTE	FETCH	(!always3).(!two_operand_math) + (!always3).(two_operand_math).(rom_in[8]).(!rom_in[9]).(!rom_in[10]).(!rom_in[11])
2	EXECUTE	EXECUTE_2	(two_operand_math).(!rom_in[8]) + (two_operand_math).(rom_in[8]).(!rom_in[9]).(!rom_in[10]).(rom_in[11]) + (two_operand_math).(rom_in[8]).(!rom_in[9]).(rom_in[10]) + (two_operand_math).(rom_in[8]).(rom_in[9])
3	EXECUTE	WRITE_BACK	(always3).(!two_operand_math) + (always3).(two_operand_math).(rom_in[8]).(!rom_in[9]).(!rom_in[10]).(!rom_in[11])
4	EXECUTE_2	WRITE_BACK	
5	FETCH	EXECUTE	
6	WRITE_BACK	FETCH	