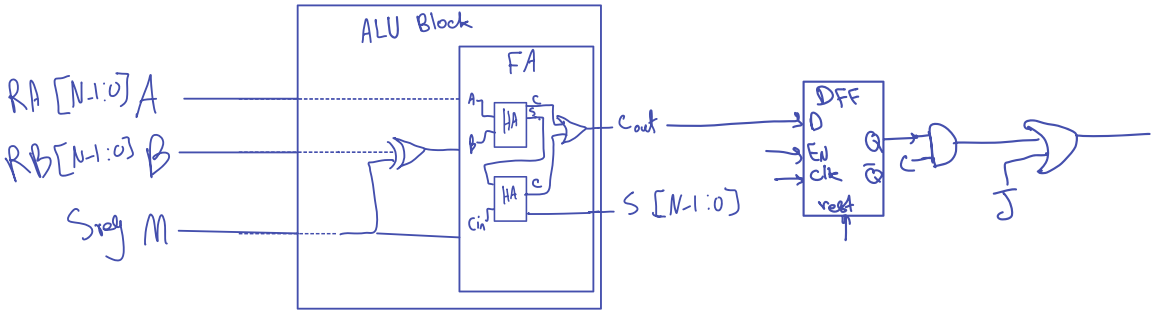
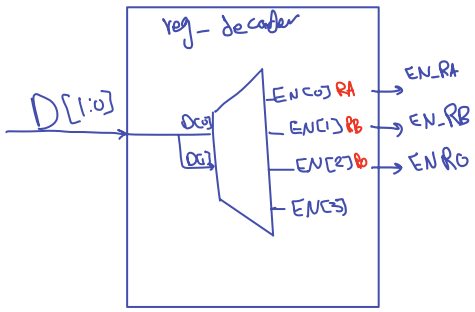
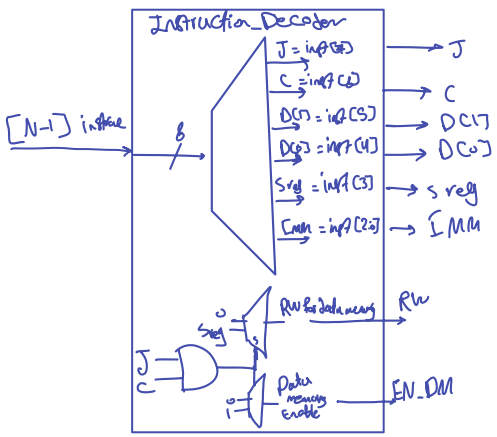
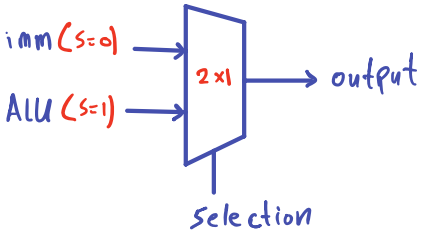
All drawings


ALU Block
FA
RA [N-1:0] A
RB [N-1:0] B
Sreg M
HA
HA
Cout
S [N-1:0]
Cin
DFF
D
EN
clk
Q
reset
C
J

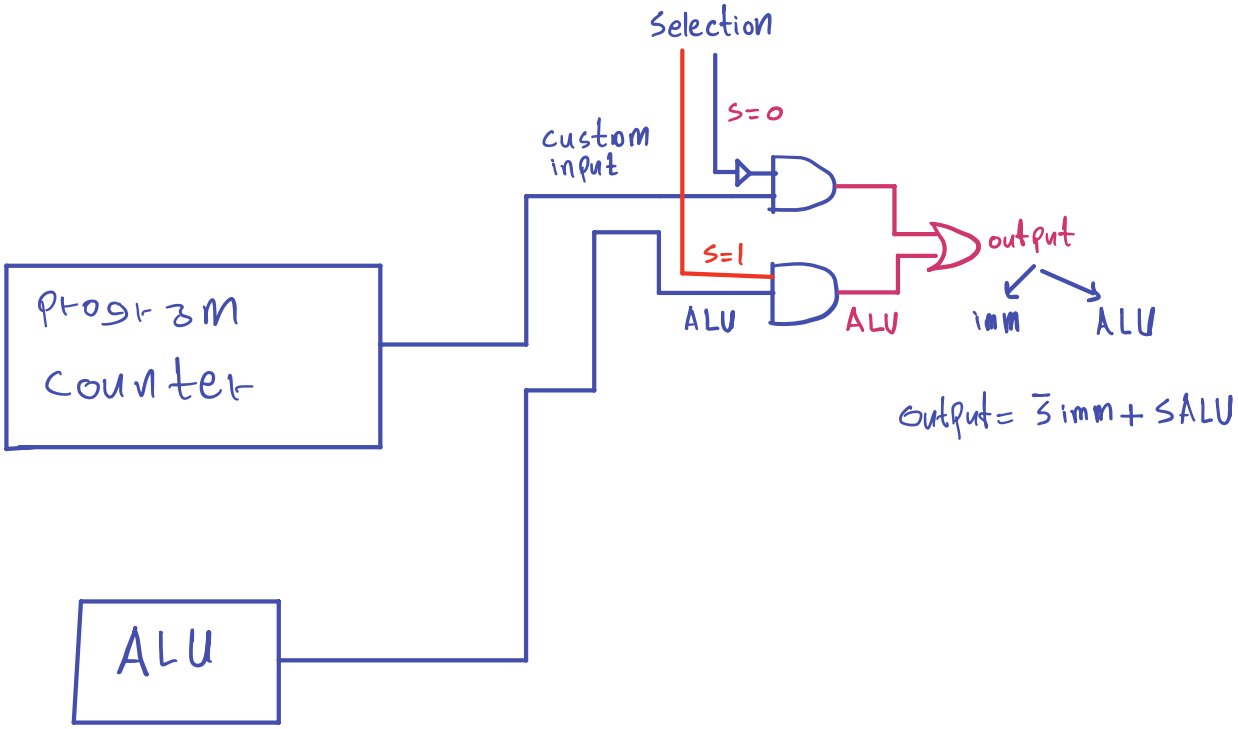

reg_decoder
D[1:0]
EN[0] RA
EN[1] RB
EN[2] RD
EN[3]
EN_RA
EN_RB
EN_RD


Instruction_Decoder
[N-1] instruc
J
C
DC[1]
DC[0]
Sreg
Imm
RW
EN_DM
Sreg
Data memory enable

Multiplexer

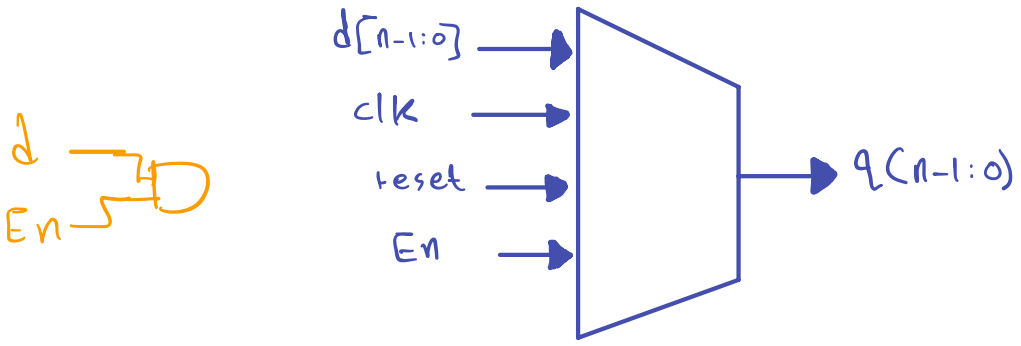


S	output
0	imm
1	ALU



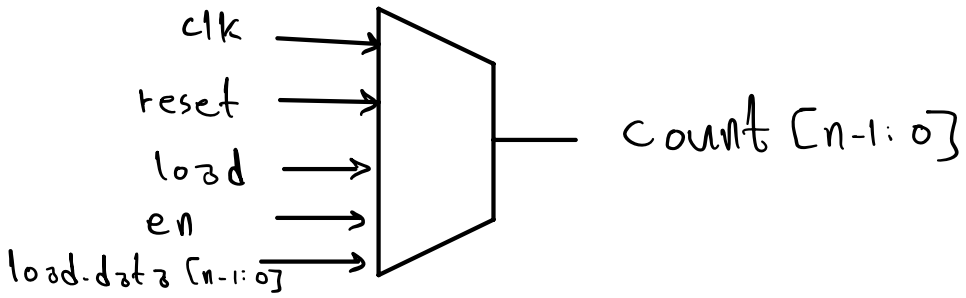
output = $\bar{S}$imm + SALU

Register



- if reset is low, the multiplexer selects the reset value (zero) to output to the register
- if Enable is high, the and gate allows the input data (d) to pass to the multiplexer.
- clock
  on the clock edge, the register updates its value based on the multiplexer output.
- output (q)
  The stored value is output from the register, reflecting the last loaded value or reset state

Counter

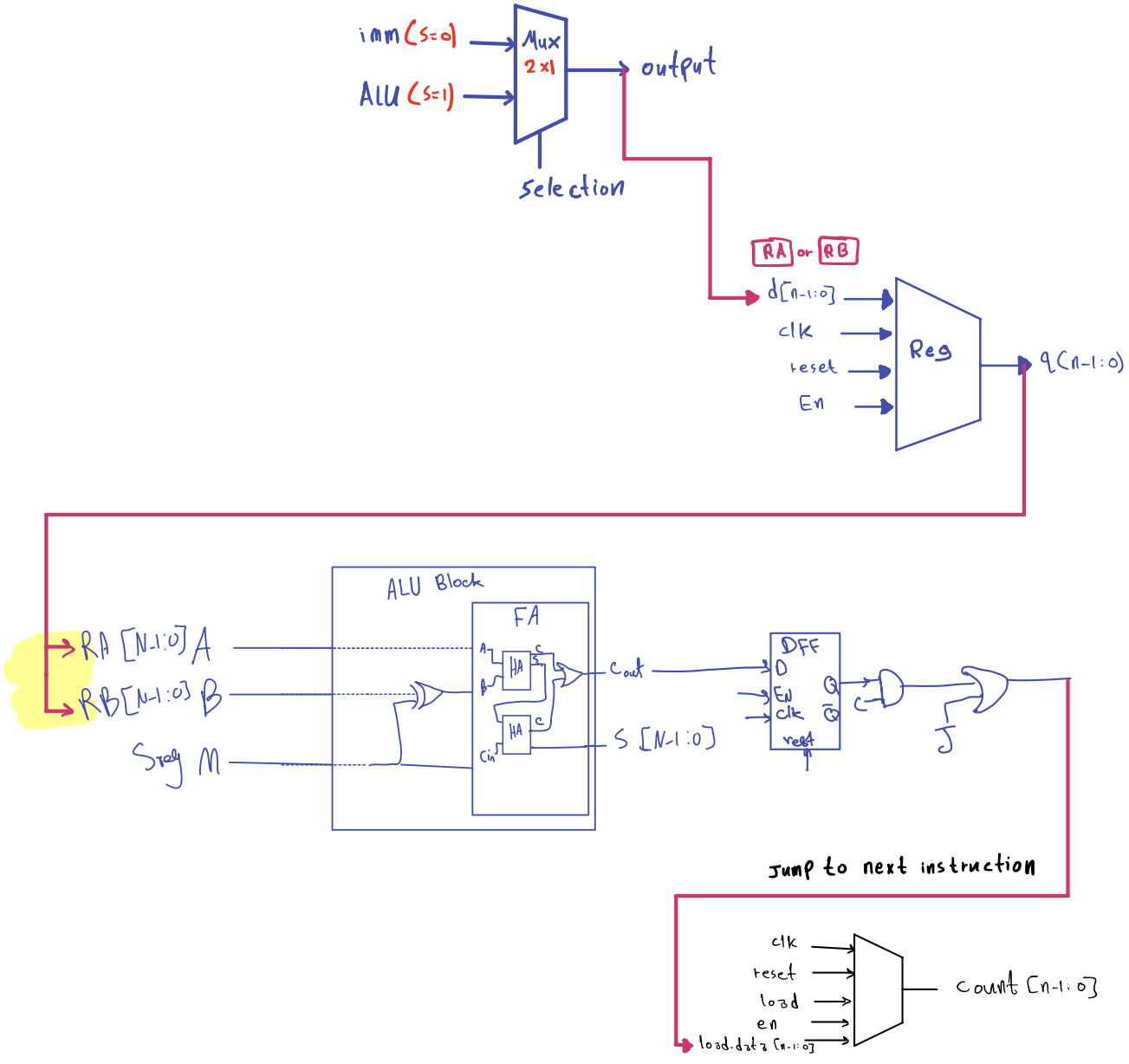


if reset is low → the counter is reset to 0

if load is high, the counter value is updated with load_data

if en is high, the counter increments by 1

[All operations occur on the rising edge of the clock]


imm (S=0)
ALU (S=1)
Mux
2x1
output
selection
RA or RB
d[n-1:0]
clk
reset
En
Reg
q[n-1:0]
ALU Block
FA
HA
HA
RA [N-1:0] A
RB [N-1:0] B
Sreg M
Cout
S [N-1:0]
DFF
D
En
clk
reset
Q
J
jump to next instruction
clk
reset
load
en
load_data [n-1:0]
count [n-1:0]