4-bit ALU :-

- Arithmatic logical unit performs op^n such as "Add", substract", Mult., div., logical op^n (AND, OR, NAND, NOR, NOT, XOR, XNOR) along with shifting (left or Right), etc.
- It acts just like a calculator
- operation stages include :-
  1. Basic Arithmatic & logical op^n of 2-bit operand
  2. operation controlled selection by usually using a Mux.

  ⇒ operation selection produces
  1. A 4-bit o/p result Y
  2. carry-flag (for arithmatic overflow or borrow)
  3. zero-flag (set when Y=0)

- All the process are computed in ||rl and mux selects any one o/p based on S-lines.
- CKts to be used
  1. Adder (A+B)
  2. Subtractor (A-B)
  3. Logic gate blocks (AND, OR, NOT ---)
  4. shifters (eg:- Barrel shifter)

- **Connections :-**

⇒ outputs of 1st stage are connected to a 1st stage Mux and controlled by sel. lines

eg :-

S ⇒	000	001	010	011	100	101	110	111
op^n ⇒	A+B	A-B	A&B	A|B	A^B	$\bar{A}$	A<<B	A>>B

⇒ If more op^n are added then Larger Mux should be chosen

eg :- 10 - 15 op^n ⇒ 16x1 Mux

16 - 31 op^n ⇒ 32x1 Mux

⇒ particularly, carry & burrow to be connected to a 2nd stage Mux which gives carry-flag as o/p

⇒ Another zero-flag from 1st stage Mux if y=0

- **Step-wise code**

① Module declaration and design (preferred Behavioural - case based Model)

② ↳ uses always @ (*) ⇒ executes whenever i/ps change, making ckt purely comb.

③ Initialise carry = 0 (for non arithmatic op^n)

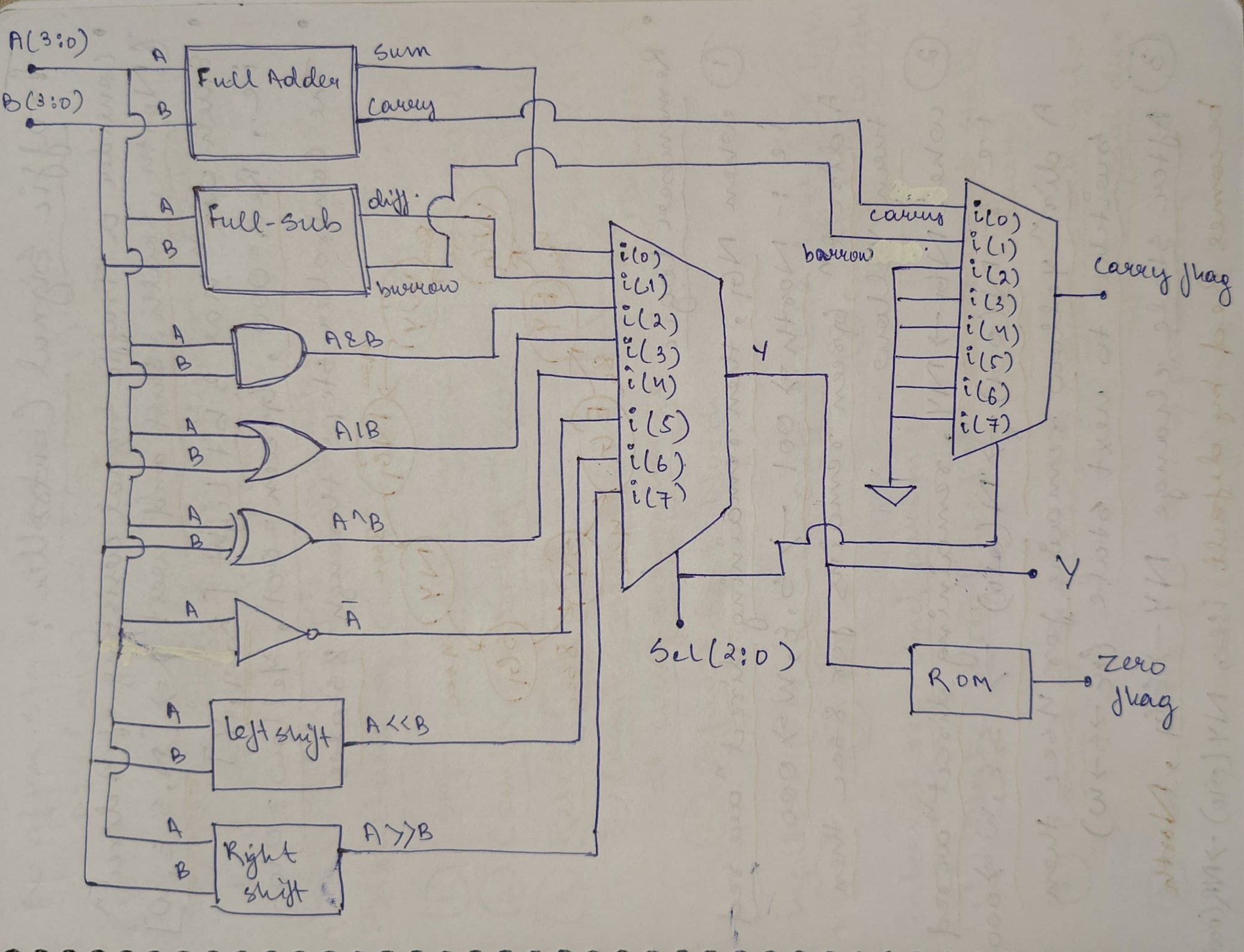
④ case statement

① 000 : {carry, y} = a+b

② 001 : {carry, y} = a-b

③ 010 : {a&b} ------ (same as table)

⑤ zero flag gen :- `zero = (y==4'b0000)?1'b1:1'b0;`


A(3:0)
B(3:0)
A
B
Full Adder
sum
carry
Full-Sub
diff.
borrow
A&B
A|B
A^B
Ā
left shift
A<<B
Right shift
A>>B
i(0)
i(1)
i(2)
i(3)
i(4)
i(5)
i(6)
i(7)
Y
Sel(2:0)
carry
borrow
carry flag
ROM
zero flag