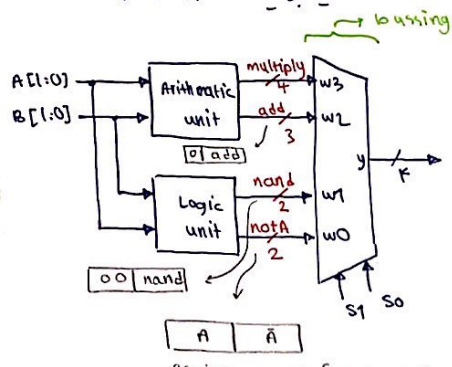

operation	Description	Sel1	Sel0	
not	A′	0	0	2
nand	(A·B′)	0	1	2
add	A+B	1	0	3
multiply	A*B	1	1	4

assign w0 = { A, ~A }

verilog code for 4to1 mux:

```
module multiplexer4to4 ( input [3:0] w3 ,input [3:0] w2, input [3:0] w1,
input [3:0] w0, input [1:0] sel, output [3:0] y );

  assign y = sel[1] ? (sel[0] ? w3 : w2) : (sel[0] ? w1 : w0) ;
endmodule
```

sel[1]	sel[0]	
0	0	w0
0	1	w1
1	0	w2
1	1	w3

verilog code for logic unit using mux and assign:

```
module alu( input [1:0] A, input [1:0] B , input [1:0] sel , output [3:0] Y );

  wire addw[3:0] ;
  wire multw [3:0] ;
  wire nandw [3:0] ;
  wire notw [3:0] ;
  wire w0 [1:0] ;
  wire w1 [1:0] ;
  wire cout ;
  wire plus [1:0];
  assign w0[0] =0 ;
  assign w0[1] =0
   "  {cout, plus} = A+B;
   "  w1[1] =0
   "  w1[0] = cout;
   "  addw = { w1, plus};
   "  nandw = { w0, ~(A & B)} ;
   "  notw = { A , ~A }
multphier 2x2 multy (A, B, multw); (1)
```

```
mux4: multiplexer4to4 muxy( multw, addw,
nandw, notw, sel, Y );

endmodule
```

```
module multiply2x2 ( input [1:0] A, input[1:0] B , output [3:0] f );
  wire [3:0]c;
  and
      g1( f[0], A[0], B[0]) ,
      g2 ( c[0], A[1], B[0] ) ,
      g3 ( c[1], A[0], B[1] ),
      g4 ( c[2], B[1], A[1]));

assign { c[3], f[1]} =  c[0] + c[1] ;
assign { f[3], f[2]} =  c[3] + c[2] ;
                          ↳ carry
  endmodule
```