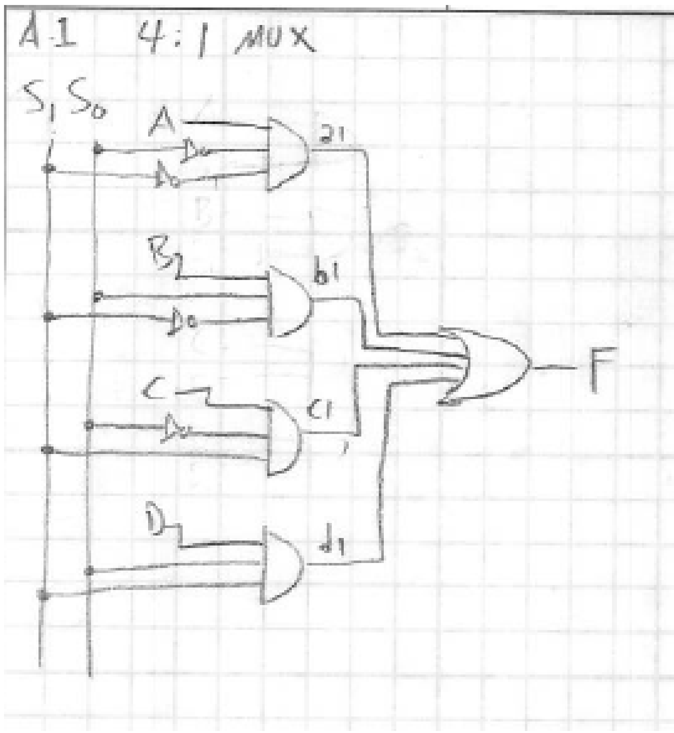
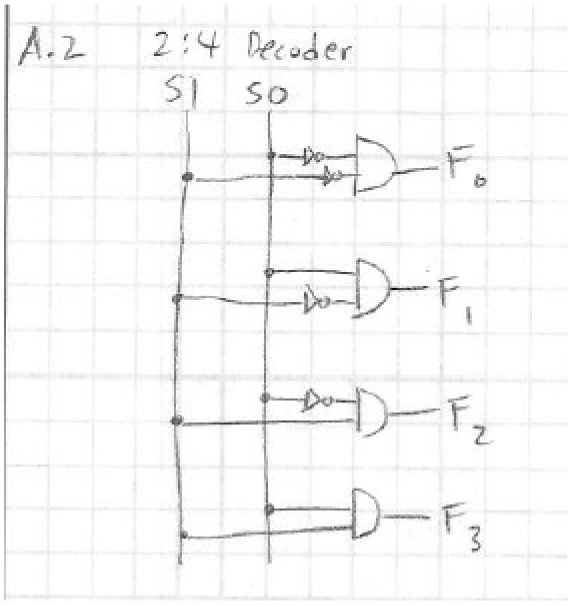
A.1 4:1 MUX

```
Module mux4_1(
    input a,
    input b,
    input c,
    input d,
    input[1:0] s,
    output f
    );

    wire a1, b1, c1, d1, ns1, ns0;

    not(ns1, sel[1]);
    not(ns0, sel[0]);

    and(a1, a, ns1, ns0);
    and(b1, sel[0], ns1);
    and(c1, ns0, sel[1]);
    and(d1, sel[0], sel[1]);
    or(f, a1, b1, c1, d1);
endmodule
```



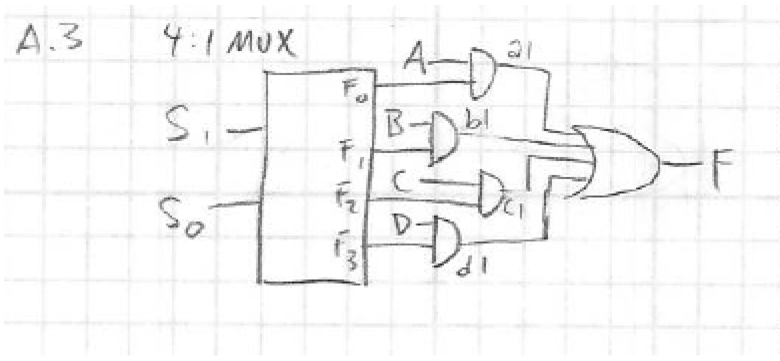
A.2 2:4 Decoder

```
    module decoder2_4(
            input[1:0] sel,
            output f0,
            output f1,
            output f2,
            output f3
            );

            wire ns1, ns0;
            not(ns1, sel[1]);
            not(ns0, sel[0]);
            and(f0, ns1, ns0);
            and(f1, ns1, sel[0]);
            and(f2, sel[1], ns0);
            and(f3, sel[1], sel[0]);
    endmodule
```



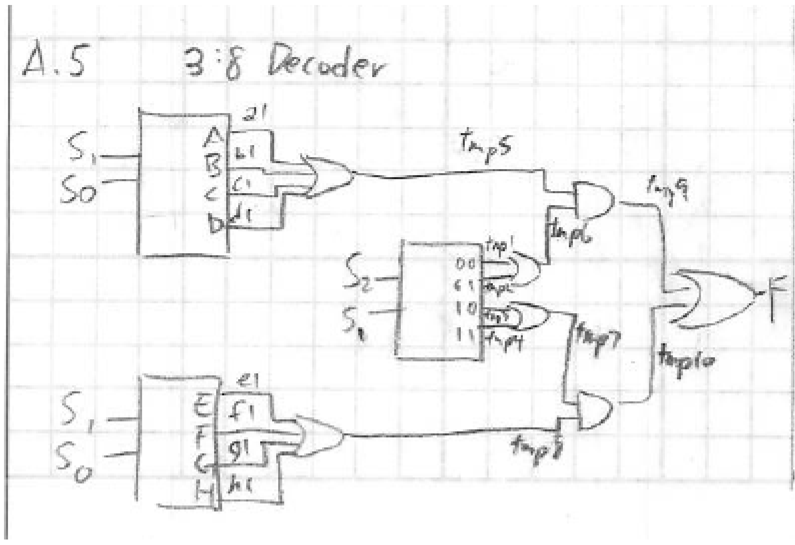
A.3 4:1 MUX

```
module mux4_1(
        input a,
        input b,
        input c,
        input d,
        input[1:0] sel
        output f
        );
        wire f0, f1, f2, f3, a1, b1, c1, d1;

        decoder2_4 D1(.sel(sel), .f0(f0), .f1(f1), .f2(f2), .f3(f3));
        and(a1, a, f0);
        and(b1, b, f1);
        and(c1, c, f2);
        and(d1, d, f3);
        or(f, a1, b1, c1, d1);
endmodule
```



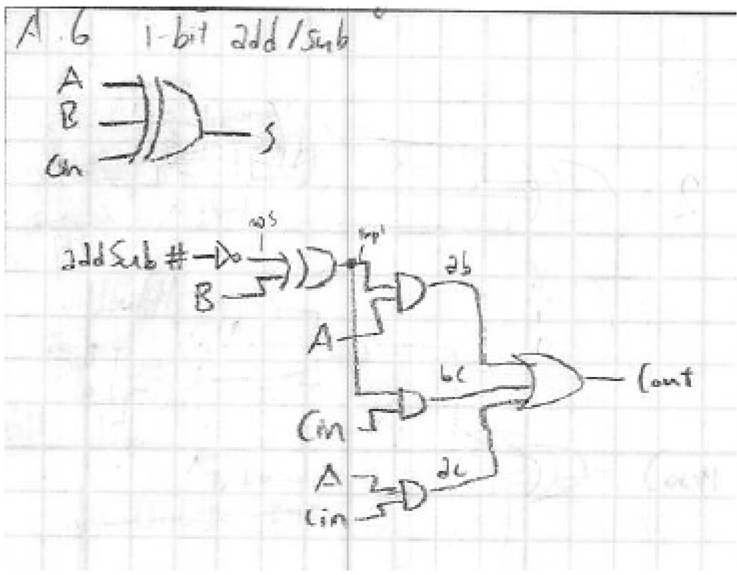
A.5 3:8 Decoder

```
module decoder3_8(
        input [2:0] sel,
        output f
        );

        wire a1, b1, c1, d1, e1, f1, g1, h1, tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7, tmp8,
                tmp9, tmp10;

        decoder2_4 D1(.sel(sel[1:0]), .f0(a1), .f1(b1), .f2(c1), .f3(d1));
        decoder2_4 D2(.sel(sel[1:0]), .f0(e1), .f2(f1), .f2(g1), .f3(h1));
        decoder2_4 D3(.sel(sel[2:1]),
.f0(tmp1), .f1(tmp2), .f2(tmp3), .f3(tmp4));

        or(tmp5, a1, b1, c1, d1);
        or(tmp6, tmp1, tmp2);
        or(tmp7, tmp3, tmp4);
        or(tmp8, e1, f1, g1, h1);

        and(tmp9, tmp5, tmp6);
        and(tmp10, tmp7, tmp8);

        or(f, tmp9, tmp10);
endmodule
```



A.6 1 bit addder / subtracter

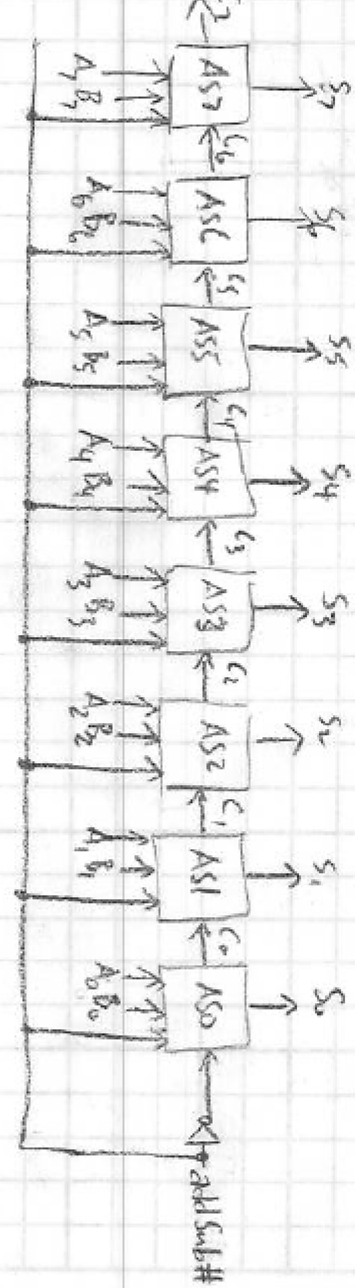
```
module addSub(
        input a,
        input b,
        input cin,
        input aS,
        output s,
        output cout
        );

        wire ab, bc, ac, tmp1, naS;

        xor(s, a, b, cin);
        not(naS, aS);
        xor(tmp1, naS, b);
        and(ab, tmp1, a);
        and(bc, cin, tmp1);
        and(ac, a, cin);
        or(cout, ab, bc, ac);
endmodule
```



A.7 8 bit adder / subtracter

```
module addSub8(
        input [7:0] a,
        input [7:0] b,
        input aS,
        output [7:0] s,
        output cout
        );

        wire c0, c1, c2, c3, c4, c5, c6, naS;

        not(naS, aS);

        addSub AS0(.a(a[0]), .b(b[0]), .cin(naS), .aS(aS), .s(s[0]), .cout(c0));
        addSub AS1(.a(a[1]), .b(b[1]), .cin(c0), .aS(aS), .s(s[1]), .cout(c1));
        addSub AS2(.a(a[2]), .b(b[2]), .cin(c1), .aS(aS), .s(s[2]), .cout(c2));
        addSub AS3(.a(a[3]), .b(b[3]), .cin(c2), .aS(aS), .s(s[3]), .cout(c3));
        addSub AS4(.a(a[4]), .b(b[4]), .cin(c3), .aS(aS), .s(s[4]), .cout(c4));
        addSub AS5(.a(a[5]), .b(b[5]), .cin(c4), .aS(aS), .s(s[5]), .cout(c5));
        addSub AS6(.a(a[6]), .b(b[6]), .cin(c5), .aS(aS), .s(s[6]), .cout(c6));
        addSub AS7(.a(a[7]), .b(b[7]), .cin(c6), .aS(aS), .s(s[7]), .cout(cout));

        endmodule
```

A 7   8bit add/sub



B.1 4:1 MUX

```
module mux4_1(q, sel, a, b, c, d);
        input a, b, c, d;
        input [1:0] sel;
        output q;

        assign q = (a & ~sel[1] & ~sel[0]) | (b & ~sel[1] & sel[0]) | (c & sel[1] & ~sel[0]) | (d &
                sel[1] & sel[0]);
endmodule
```

B.2 4:1 MUX

```
module mux4_1(q, sel, a, b, c, d);
        input a, b, c, d;
        input [1:0] sel;
        output q;

        assign q = (sel == 2'b00)?a:
                        (sel == 2'b01)?b:
                        (sel == 2'b10)?c:
                        d;
        endmodule
```

B.3 4:1 Mux

```
module mux4_1(q, sel, a, b, c, d);
        parameter WID = 16;
        input[WID – 1:0] a, b, c, d;
        input [1:0] sel;
        output[WID – 1:0] q;

        assign q =      (a & {WID{~sel[1]}} & {WID{~sel[0]}}) |
                        (b & {WID{~sel[1]}} & {WID{sel[0]}}) |
                        (c & {WID{sel[1]}} & {WID{~sel[0]}}) |
                        (d & {WID{sel[1]}} & {WID{sel[0]}});
endmodule
```

B.4 4:1 MUX

```
module mux4_1(q, sel, a, b, c, d);
        parameter WID = 16;
        input[WID -1:0] a, b, c, d;
        input [1:0] sel;
        output[WID -1:0] q;

        assign q =      (sel == 2'b00)?a:
                        (sel == 2'b01)?b:
                        (sel == 2'b10)?c:
                        d;
        endmodule
```

B.5 8 bit ALU

```
module alu8(A, B, Sel, Q);
        input[7:0] A, B;
        input[1:0] Sel;
        output [7:0] Q;

        assign Q =      (Sel == 2'b00)?A:
                        (Sel == 2'b01)?(A+B):
                        (Sel == 2'b10)?(A&B):
                        (~A);
        endmodule
```

B.8 8 bit adder/subtracter

```
module addSub8(A, B, MODE, Q);
        input [7:0] A, B;
        input MODE;
        output [8:0] Q;

        assign Q = (MODE == 1'b0)?({1'b0, A} + {1'b0, B}):({1'b0, A} – {1'b0, B});
        endmodule
```