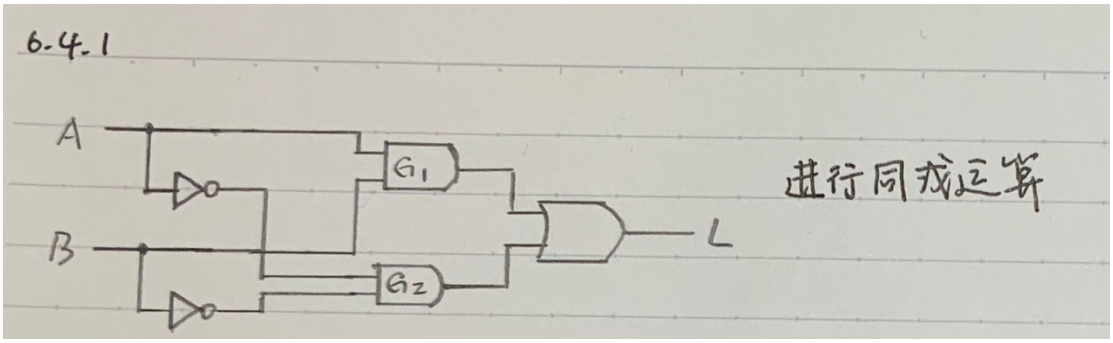
由于 modelsim 损坏，暂时无仿真（云服务器上正在下载）

6.4.1



6.6.3

```
module Encoder8to3_bh(EI, I, Y, GS, EO);
input EI;
input [7:0] I;
output reg [2:0] Y;
output reg GS, EO;

always@(EI, I)
begin
   if(!EI) begin Y = 3'd0; GS = 0; EO=0; end
   else
      begin
         GS = 1; EO = 0
         casex(1)
         8b'1xxx_xxxx: Y = 3d'7;
         8b'01xx_xxxx: Y = 3d'6;
         8b'001x_xxxx: Y = 3d'5;
         8b'0001_xxxx: Y = 3d'4;
         8b'0000_1xxx: Y = 3d'3;
         8b'0000_01xx: Y = 3d'2;
         8b'0000_001x: Y = 3d'1;
         8b'0000_0001: Y = 3d'0;
         default: begin Y = 3'd0; GS = 0; EO = 1; end
      endcase
end

endmodule
```

6.4.2

```
module D_FF(Q,D,CP,Rd); //module D_FF with asynchronous reset output Q;
input D,CP,Rd;
reg Q;
always @(posedge CP or negedge Rd)
if (~Rd) Q <= 1'b0;
else Q <= D; endmodule


module ripplecounter (Q0,Q1,Q2,Q3,CP,Rd); //Ripple counter output
Q0,Q1,Q2,Q3;
input CP, Rd;
input [3:0]D;
output reg [3:0]D;

always@(posedge clk)
   D_FF(Q0,~Q0,CP,~Rd);
   D_FF(Q1,~Q1,CP,~Rd);
   D_FF(Q2,~Q2,CP,~Rd);
   D_FF(Q3,~Q3,CP,~Rd);
end
endmodule
```

6.4.3

```
module JK_FF(J,K,CP,Q,Qnot); //参见例 6.6.6 output Q,Qnot;
    1J >C1 1K
    1J >C1 1K
    input J,K,CP;
    reg Q;
    assign Qnot = ~ Q ; always @(posedge CP)
    case ({J,K}) //根据 J、K 的值进行选择 2'b00: Q <= Q;
    2'b01: Q <= 1'b0;
    2'b10: Q <= 1'b1;
    2'b11: Q <= ~ Q;
    endcase
endmodule

module Up_Down (Q,CP,Up_Down); //IEEE 1364-1995 Syntax
    input CP,Up_Down;
    output [2:0]Q;
    wire UP,CP1,CP2;
    wire S1,S2,S3, S5,S6,S7;
    always@(posedge CP)
        JK_FF(1,1,CP,Q0,S1);
        and(S2,Q0,Up_Down);
        and(S3,S1,UP);
        or(CP1,S2,S3);
        always@(posedge CP1)
            JK_FF@(1,1,CP1,Q1,S5);
            and(S6,Q1,Up_Down);
            and(S7,S5,UP);
            or(CP2,S6,S7);
            always@(posedge CP2);
                JK_FF(1,1,CP2,Q2,~Q2);
            end
        end
    end
endmodule
```

6.6.4

```
module Step_motor(
    input M,CP,nCR, // Input Ports
    output A,B,C // Output Ports
    );
    reg [2:0] current_state, next_state;
    // Parameter Declarations
    parameter [2:0] S0=3'b000, S1=3'b001, S2=3'b010, S3=3'b011,
             S4=3'b100, S5=3'b101, S6=3'b110, S7=3'b111;
    case(current_state)
       S0: next_state <= S1;
       S1: if(M) next_state <= S3; else next_state <= S5;
       S2: if(M) next_state <= S6; else next_state <= S3;
       S3: if(M) next_state <= S2; else next_state <= S1;
       S4: if(M) next_state <= S5; else next_state <= S6;
       S5: if(M) next_state <= S1; else next_state <= S4;
       S6: if(M) next_state <= S4; else next_state <= S2;
       S7: next_state <= S6;
    endcase
endmodule
```

6.7.1

```
module OneDigitComparator(
    input A,B,
    output F1,F2,F3;//F1 for A>B, F2 for A==B and F3 for A<B
)
    reg G1,G2,G3;

    and(G1,A,~B);
    and(G2,~A,B);
    nor(G3,G1,G2);

    if (G1)
endmodule

module TwoDigitComparator(
    input A1,A2,B1,B2,
    output F1,F2,F3;
)
    reg G1,G2,G3,G4,G5;
    OneDigitComparator (A1,B1,F11,F12,F13);
    OneDigitComparator (A2,B2,F21,F22,F23);

    and(G1,F12,F13);
    and(G2,F12,F23);
    and(G1,F12,F23);
    or(G4,F11,G1);
    or(G5,F13,G2);
endmodule
```