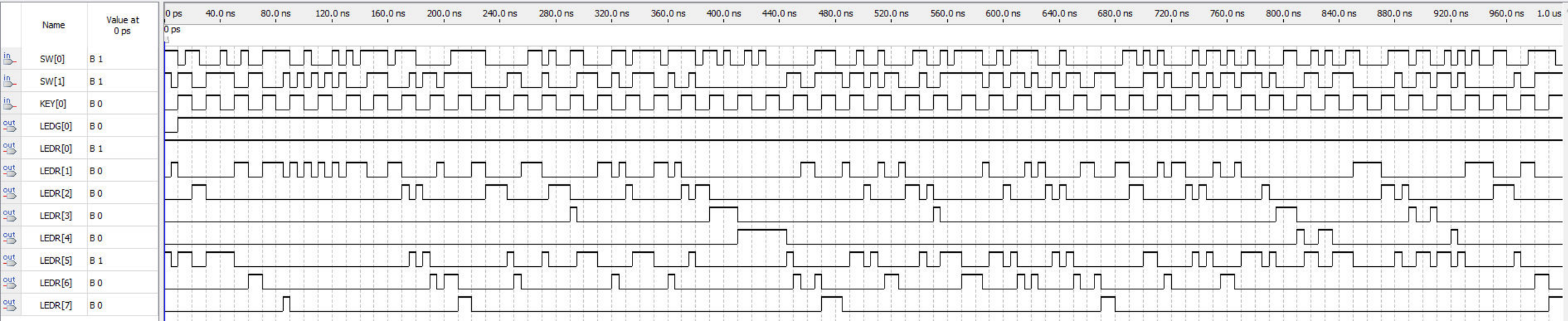
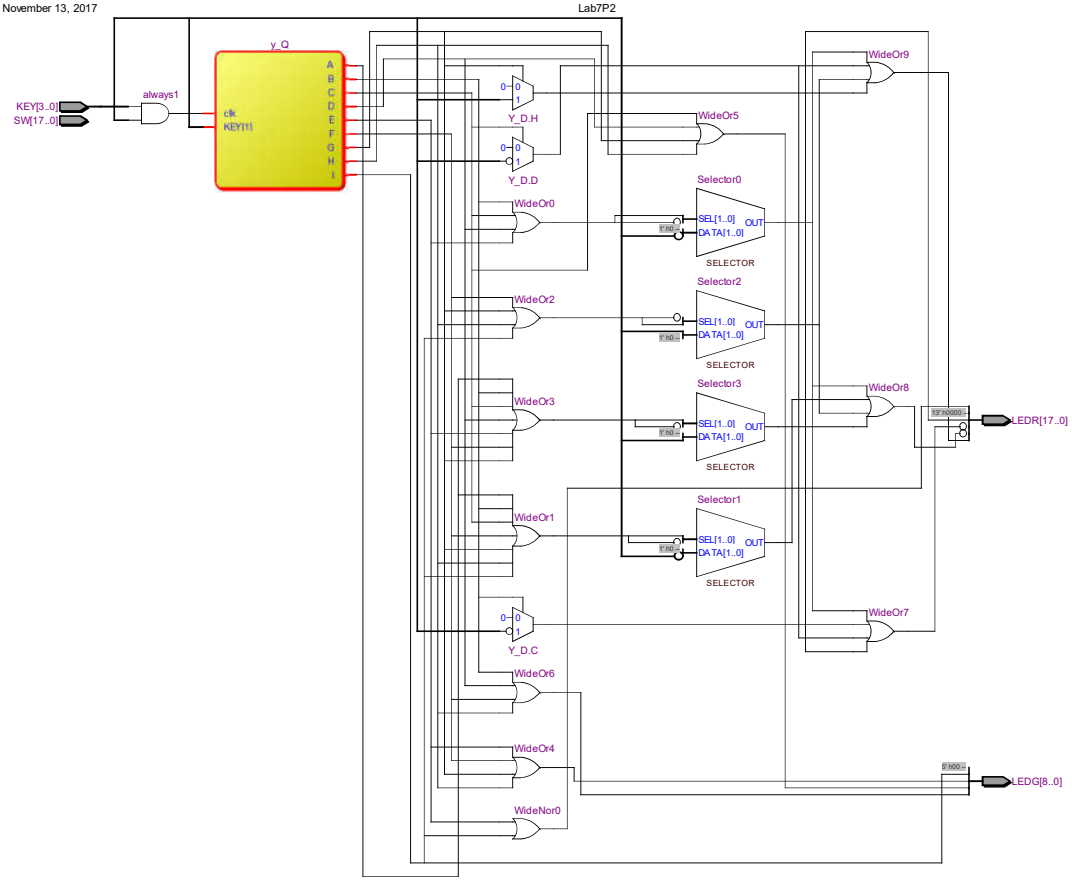
```
module part1 (SW, KEY, LEDR, LEDG);
  input [17:0] SW;
  input [3:0] KEY;
  output [17:0] LEDR;
  output [8:0] LEDG;

  wire [8:0] D, Y;
  wire w;

  assign w = SW[1];

  my_lpm_ff FF (~KEY[1], KEY[0], D[8:0], Y[8:0]);

  assign D[0] = ~(~(~Y[0] | Y[1] | Y[2] | Y[3] | Y[4] | Y[5] | Y[6] | Y[7] | Y[8]) | ~(D[1] |
  D[2] | D[3] | D[4] | D[5] | D[6] | D[7] | D[8]));

  assign D[1] = (~Y[0] | Y[5] | Y[6] | Y[7] | Y[8]) & ~w; // B
  assign D[2] = Y[1] & ~w; // C
  assign D[3] = Y[2] & ~w; // D
  assign D[4] = (Y[3] | Y[4]) & ~w; // E

  assign D[5] = (~Y[0] | Y[1] | Y[2] | Y[3] | Y[4]) & w; // F
  assign D[6] = Y[5] & w; // G
  assign D[7] = Y[6] & w; // H
  assign D[8] = (Y[7] | Y[8]) & w; // I

  assign LEDG = Y;
  assign LEDR[8:0] = D;
endmodule
```

```
module Lab7P2 (SW, KEY, LEDR, LEDG);
  input [17:0] SW;
  input [3:0] KEY;
  output [17:0] LEDR;
  output [8:0] LEDG;

  wire w;
  assign w = SW[1];

  wire Clock;
  assign Clock = KEY[0];
  reg z;

  reg [3:0] y_Q, Y_D; // y_Q represents current state, Y_D represents next state
  parameter A = 4'b0000, B = 4'b0001, C = 4'b0010, D = 4'b0011, E = 4'b0100, F = 4'b0101, G =
  4'b0110, H = 4'b0111, I = 4'b1000;
  always @(w, y_Q)
  begin: state_table
    case (y_Q)
      A: if (!w) Y_D = B;
        else Y_D = F;
      B: if (!w) Y_D = C;
        else Y_D = F;
      C: if (!w) Y_D = D;
        else Y_D = F;
      D: if (!w) Y_D = E;
        else Y_D = F;
      E: if (!w) Y_D = E;
        else Y_D = F;
      F: if (w) Y_D = G;
        else Y_D = B;
      G: if (w) Y_D = H;
        else Y_D = B;
      H: if (w) Y_D = I;
        else Y_D = B;
      I: if (w) Y_D = I;
        else Y_D = B;
      default: Y_D = 4'bxxxx;
    endcase
  end // state_table
  always @(posedge Clock)
  begin: state_FFs
    y_Q = Y_D;
  end // state_FFS

  always
  begin: zset
    case (y_Q)
      E: z = 1;
      I: z = 1;
      default: z = 0;
    endcase
  end
```

```
  assign LEDG[3:0] = y_Q;
  assign LEDR[3:0] = Y_D;

  assign LEDR[17] = z;

  test t1 (SW[17], SW[16], LEDR[17]);
endmodule


module test (A, B, Out);
  input A, B;
  output reg Out;

  always
    case (A)
      0: Out = 0;
      1: Out = B;
    endcase
endmodule
```


Name
Value at 0 ps
SW[0] B 1
SW[1] B 1
KEY[0] B 0
LEDG[0] B 0
LEDR[0] B 1
LEDR[1] B 0
LEDR[2] B 0
LEDR[3] B 0
LEDR[4] B 0
LEDR[5] B 1
LEDR[6] B 0
LEDR[7] B 0
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
360.0 ns
400.0 ns
440.0 ns
480.0 ns
520.0 ns
560.0 ns
600.0 ns
640.0 ns
680.0 ns
720.0 ns
760.0 ns
800.0 ns
840.0 ns
880.0 ns
920.0 ns
960.0 ns
1.0 us


y_Q
A
B
C
D
E
F
G
H
I
KEY[3..0]
SW[17..0]
always1
clk
KEY[1]
Y_D.H
Y_D.D
Y_D.C
WideOr0
WideOr1
WideOr2
WideOr3
WideOr4
WideOr5
WideOr6
WideOr7
WideOr8
WideOr9
WideNor0
Selector0
Selector1
Selector2
Selector3
SEL[1..0]
DATA[1..0]
OUT
SELECTOR
LEDR[17..0]
LEDG[8..0]