```verilog
`timescale 1ns / 1ns // `timescale time_unit/time_precision

// Counter that changes the speed of counting based on switch inputs
module multiSpeedCounter4bit(input[9:0] SW, input CLOCK_50, output[6:0] HEX0);
    wire enable,reset;

    wire[1:0] speedSelect;
    wire[3:0] Q;
    wire[25:0] rDivider;

    reg[25:0] rDivLoad;

    // When simulating use divisor = 100000
    // Uploading to FPGA use divisor = 1
    // Used because simulating a 50 Mhz clock for
    // something like 10s means a very long run time
    parameter divisor = 1;

    // Input assignments
    assign speedSelect = SW[1:0];
    assign reset = SW[2];

    // Decide the parallel load into RateDivider
    always @(*)
    begin
        case(speedSelect)
            2'b00: // Full speed (50 MHz)
                rDivLoad = 0;
            2'b01: // 4 Hz (12.5M-1)
                rDivLoad = 12500000/divisor-1;
            2'b10: // 2 Hz (25M-1)
                rDivLoad = 25000000/divisor-1;
            2'b11: // 1 Hz (50M-1)
                rDivLoad = 50000000/divisor-1;
            default: // Just set to full speed
                rDivLoad = 0;
        endcase
    end

    // Instantiate rate divider
    rateDivider
    rD0(.clock(CLOCK_50),.reset(reset),.loadIn(rDivLoad),.rDivider(rDivider));
    assign enable = (rDivider==0) ? 1:0;

    // Actual counter
    displayCounter counter(.clock(CLOCK_50),.enable(enable),.reset(reset),.Q(Q));

    // Output
    displayHEX h0(.BIN(Q),.HEX(HEX0));
endmodule

// Active low, synchronous reset positive edge triggered counter with parallel load
// Counts down from loadIn to 0
module rateDivider(input[25:0] loadIn, input clock,reset, output[25:0] rDivider);
    reg[25:0] counter;

    always @(posedge clock)
    begin
        if(!reset) // Active low reset to loadIn
            counter <= loadIn;
        else if(counter==0) // Reset to initial value (1 cycle)
            counter <= loadIn;
        else // Count down
            counter <= counter-1;
    end

    assign rDivider = counter;
endmodule

```

```verilog
// Active low, synchronous reset positive edge triggered counter
// Counts from 0 to F (no reset at F required since Q is only 4 bits)
module displayCounter(input clock,enable,reset, output[3:0] Q);
  reg[3:0] q;

  always @(posedge clock)
  begin
    if(!reset) // Set to 0
      q <= 0;
    else if(enable) // Count
      q <= q+1;
  end

  assign Q = q;
endmodule

// Turns decimal to HEX
module displayHEX(input [3:0] BIN, output [6:0] HEX);
   wire x = BIN[3], y = BIN[2], z = BIN[1], w = BIN[0];
   //assign each segment with appropriate formula
   assign HEX[0] = ~((x|y|z|~w)&(x|~y|z|w)&(~x|y|~z|~w)&(~x|~y|z|~w));
   assign HEX[1] =
   ~((x|~y|z|~w)&(x|~y|~z|w)&(~x|y|~z|~w)&(~x|~y|z|w)&(~x|~y|~z|w)&(~x|~y|~z|~w));
   assign HEX[2] = ~((x|y|~z|w)&(~x|~y|z|w)&(~x|~y|~z|w)&(~x|~y|~z|~w));
   assign HEX[3] =
   ~((x|y|z|~w)&(x|~y|z|w)&(x|~y|~z|~w)&(~x|y|z|~w)&(~x|y|~z|w)&(~x|~y|~z|~w));
   assign HEX[4] =
   ~((x|y|z|~w)&(x|y|~z|~w)&(x|~y|z|w)&(x|~y|z|~w)&(x|~y|~z|~w)&(~x|y|z|~w));
   assign HEX[5] = ~((x|y|z|~w)&(x|y|~z|w)&(x|y|~z|~w)&(x|~y|~z|~w)&(~x|~y|z|~w));
   assign HEX[6] = ~((x|y|z|w)&(x|y|z|~w)&(x|~y|~z|~w)&(~x|~y|z|w));
endmodule
```