1. Problem 1
Capture the following system behaviour as an HLSM. The system counts the number of events on a single-bit input B and always outputs that number unsigned on a 6-bit output C, which is initially 0. An event is a change from 0 to 1 or from 1 to 0. Assume that system count rolls over when the maximum value of C is reached.

Solution:

Alternative solution:

2. Problem 2
Use the RTL design process to design a system that outputs the average of the most recent two data input samples. The system has an 8-bit unsigned data input I and an 8-bit unsigned output. The data input is sampled when a single bit input S changes from 0 to 1. Choose the internal bit widths that prevent overflow.

Solution:
Step 1 - Capture a high-level state machine

Inputs: I (8 bits), S (bit)
Outputs: avg (8 bits)
Local Registers: Prevreg (8 bits), Ireg (8 bits), avgreg (8 bits)

Prevreg := 0
Ireg := 0
avgreg := 0

Step 2 - Create a datapath

Note: A solution more consistent with the chapter’s methodology would use a separate clear and ld signal for each register. In this particular example, a single clr and a single load line happens to work.

Step 3 - Connect the datapath to a controller
3. Problem 3
Create a high-level state machine for a digital bath-water controller. The system has a 3-bit input ratio indicating the desired ratio of cold water to hot water, and a bit input on indicating that the water should flow. The system has two 4-bit outputs hflow and cflow, controlling the hot water flow rate and the cold water flow rate. The sum of these two rates should always equal 16. Your high-level state machine should determine the output values for hflow and cflow such that the ratio of hot water to cold water is as close as possible to the desired ratio, while the total flow is always 16.

Hint: As there are only 8 possible ratios, a reasonable solution may use one state for each ratio.

Solution:

Inputs: ratio (3 bits), on (bit)
Outputs: hflow (6 bits), cflow (4 bits)

Note: Multiple solutions are possible but the basic idea is the use of an Init state to route transitions. More than one clock cycle is required to change the ratio.

4. Problem 4
Convert the following C-like code, which calculates the greatest common divisor (GCD) of the two 8-bit numbers, a and b, into a high-level state machine. Then, use the RTL design process to convert the high-level state machine to a controller and a datapath. Design the datapath to structure, but design the controller to the point of an FSM only.

Inputs: byte a, byte b, bit go
Outputs: byte gcd, bit done
while(1) {
    while( !go );
    done = 0;
    while( a != b ) {
        if( a > b ) {
            a = a - b;
        } else {
            b = b - a;
        } 
    }
    gcd = a;
    done = 1;
}

Solution:
Step 1 - Capture a high-level state machine

Inputs: go (bit), a, b (8 bits)
Outputs: done (bit), gcd (8 bits)
Local Registers: a_reg (8 bits), b_reg (8 bits)

Step 2 - Create a datapath

Step 3 - Connect the datapath to a controller
Step 4 - Derive the controller’s FSM

Inputs: go, done, a_gt_b, a_eq_b (bit)
Outputs: done, a ld, a sel, b ld, b sel, gcd ld (bit)