1. Consider the pseudo-C code below.

Inputs: uint16_t A, uint16_t B, bit hold
Outputs: uint32_t Y, bit done
Storage: uint32_t tmp

main()
while(1){
    while(hold);
    done = 0;
    temp = A;
    while(B != 0){
        if (B(0) == 1){
            Y += tmp;
        }
        tmp = tmp << 1;
        B = B >> 1;
    }
    done = 1;
}

a. What does this code implement?
b. Draw an HLSM to represent the system.
c. Draw the datapath for this system.
d. Connect the datapath to the control unit. Label all signals between them.

a. This is a shift-and-add multiplier.
b.
c. 

\[ \{0...0, A\} \]

\[ \begin{array}{c}
\text{tmp}_S \\
\text{tmp}_L \\
\text{tmp}_\text{REG} \\
+ \\
\text{Y}_\text{C} \\
\text{Y}_\text{L} \\
\text{Y}_\text{REG}
\end{array} \]

\[ \begin{array}{c}
B \\
\text{B}_\text{S} \\
\text{B}_\text{L} \\
\text{B}_\text{REG}
\end{array} \]

\[ \begin{array}{c}
\text{hold} \\
A \\
B \\
\text{B}_\text{is}_0 \\
\text{B}(0) \\
\text{B}_\text{S} \\
\text{B}_\text{L} \\
\text{tmp}_\text{S} \\
\text{tmp}_\text{L} \\
\text{Y}_\text{C} \\
\text{Y}_\text{L}
\end{array} \]

\[ \text{done} \]
2. Design a system that determines whether the given unsigned 8-bit number A is prime. The system starts with pushing the button S, and activates the output E when finished the computation. Output Y determines if A is prime. You may assume that you have a combinational divider unit that gets two inputs A1 and A2, and outputs the remainder R and quotient Q in 3 clock cycles.

a. Draw an HLSM to represent the system.

b. Draw the datapath for this system.

c. Connect the datapath to the control unit. Label all signals between them.

a. 

![HLSM Diagram]

b. 

![Datapath Diagram]
3. Consider the entire system of datapath and control flow as follows. All numbers are in signed 2’s complement format. The delay of all components are given in the table below. Find the maximum frequency. Assume no hold-time violations and any subtractor/adder has a ripple-carry structure, made up of only Half- or Full-adders.
<table>
<thead>
<tr>
<th>Component</th>
<th>Delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter (NOT)</td>
<td>5</td>
</tr>
<tr>
<td>Gate (other than NOT)</td>
<td>15</td>
</tr>
<tr>
<td>Half Adder</td>
<td>25</td>
</tr>
<tr>
<td>Full Adder</td>
<td>35</td>
</tr>
<tr>
<td>2:1 Mux</td>
<td>30</td>
</tr>
<tr>
<td>DFF&lt;sub&gt;Setup&lt;/sub&gt;</td>
<td>25</td>
</tr>
<tr>
<td>DFF&lt;sub&gt;clock-q&lt;/sub&gt;</td>
<td>50</td>
</tr>
<tr>
<td>Memory (read)</td>
<td>100</td>
</tr>
</tbody>
</table>

The critical path starts from the memory and ends with the sum register. The path consists of:

1. Memory: the read delay is 100ps
2. Subtractor: 1 NOT delay plus 8-bit ripple-carry adder → delay = 5 + 8×35 = 285ps
3. ABS unit: 1 NOT delay for A', followed by an 8-bit adder that calculates A' + 1, and a mux that chooses among them → delay = 5 + 8×25 + 30= 235ps. Note that an alternative implementation is to first multiplex among A and A', then add that with A[7], which results in the same delay.
4. Adder: One input is 32 bit, the other is 8 bit. Thus, we can use full adders for the first 8 bits, and half adders for the remaining 24 bits. Note that we don’t have c_in, so the first full adder can also be replaced by a half adder → delay = 25 + 7×35 + 24×25 = 870ps
5. Register: Composed of flip-flops → delay = setup-time = 25ps

Total delay = 100 + 285 + 235 + 870 + 25 = 1515ps

4. Design a system which reorders the data using a bubble sort algorithm with the smallest entry at memory index 0. Assume a memory block contains 256 entries. Assume your memory has two read ports and can write two entries at a time.

   a. Draw an HLSM to represent the system.
   b. Draw the datapath for this system
   c. Connect control and datapath
   d. Draw the FSM and label signals controlling transitions between the states
   e. Implement the MSB of FSM
a. Output: bit done - data sorted

Storage: d[256] - array of values to be sorted

C - check if sorted

b. [Diagrams and text explaining the flow and logic of the process]
c.

Control Unit

Datapath

i_inc
i clr
sig_swap
W_en_1
W_en_2
i_lt_254
R_en_1
R_en_2

done

S0

S1

Hold

S2

S3

S4

S5

Hold'

i_clr := 1

(i_lt_254) * C

(sig_swap)

C := 0
R_en_1 := 1
R_en_2 := 1
W_en_1 := 1
W_en_2 := 1

(i_lt_254) * C'

i_inc := 1

done := 0

(d.)
<table>
<thead>
<tr>
<th>S2</th>
<th>S1</th>
<th>S0</th>
<th>MSB_next</th>
<th>Hold</th>
<th>C</th>
<th>sig_swap</th>
<th>i_lt_254</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

![Diagram](image-url)