1. The figure shows the simulation of a D flip-flop. \( z \) is the output.

A. Which input is normally called "clear" or "reset"? \( b \)

B. Why is the value of the output unknown before 10 ns? We are in a "store" state, and we don't know what value is being stored.

C. If \( b \) falls to 0 at 68 ns, while \( a \) remains at 0 and \( c \) remains at 1, will the output change? no

D. (Five points.) Write a Verilog module (not the test fixture) to implement this circuit.

```verilog
module partF (input a, input b, input c, output reg z);
always @ (posedge a or posedge b)
if (b == 1) z <= 0;
else z <= c;
endmodule
```

E. (Two points.) Write a Verilog module to implement a circuit whose output becomes 0 immediately when the clear input goes low (not high); otherwise, the circuit is identical to the previous case.

```verilog
module partF (input a, input b, input c, output reg z);
always @ (posedge a or negedge b)
if (b == 0) z <= 0;
else z <= c;
endmodule
```
2. Using D flip-flops and logic gates, design a Mealy machine that detects the sequence 1001.

A. First, list all the states (s0, s1, etc.) that we need to define, and briefly define them.

s0: nothing detected
s1: first 1 detected
s2: 10 detected
s3: 100 detected

B. (Four points.) Write out the state diagram (not the circuit diagram).

```
0/0   1/0   1/1   0/0
  s0   s1    s2    s3
1/0
  s1
0/0
  s2
1/0
  s1
0/0
  s3
```

C. How many flip-flops do you need? 2

D. Write out the necessary truth table, showing how Next State and Output depend on Present State and Input.

<table>
<thead>
<tr>
<th></th>
<th>q1</th>
<th>q0</th>
<th>In</th>
<th>D1</th>
<th>D0</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
E. (Three points.) Write a logic equation for each bit's Next State and for the Output.

\[ D_0 = \neg q_1 \land \text{In} \lor q_1 \land \neg q_0 \land q_1 \land q_0 \land \text{In} \]

\[ D_1 = \neg \text{In} \land (q_1 \oplus q_0) \]

\[ \text{Out} = q_1 \land q_0 \land \text{In} \]

F. (Three points.) Write out the circuit diagram.

G. Write an expression for \( D_0 \) (as a function of \( q_0, q_1, \) and \( \text{In} \)) that requires only three logic gates (including NOTs). Your response for part G may be the same as part of your response for part E.

\[ D_0 = \text{In} \lor q_1 \land \neg q_0 \]

H. Write an expression for \( D_1 \) (as a function of \( q_0, q_1, \) and \( \text{In} \)) that requires only three logic gates (including NOTs). Your response for part H may be the same as part of your response for part E.

\[ D_1 = \neg \text{In} \land (q_1 \oplus q_0) \]