The C code given below calculates the number of times the value \( b \) is found within an array \( A \) consisting of 256 8-bit values. For this code complete the following questions:

```c
Inputs: byte a[256], byte b, bit go
Outputs: byte freq, bit done

FREQUENCY:
while(1) {
    while(1) {  
        done = 0;
        i = 0;
        freq = 0;
        while( i < 256 ) {
            if( a[i] == b ) {
                freq = freq + 1;
            }
            i = i + 1;
        }
        done = 1;
    }
}
```

1. Convert C code into a high-level finite state machine.
2. Create the datapath for part 1.
3. Connect the datapath to the controller, label all signals.
4. Design the controller FSM. Create a state table showing inputs, current/next state, and outputs.
5. Minimize the FSM. List the maximal class of compatibility, and prove that a new set of states you choose satisfy the covering and closure conditions.
6. Use minimum bit change heuristic to encode the states resulting from part 5.
7. Implement the controller using T-FFs and the state encoding from part 6.
8. Show a gate level design of each element in your datapath. Use T-FF.
9. What is the critical path in your design (neglect memory access to fetch each of 256 8-bit data)? Assume each gate’s delay = 1ns * \#gate_inputs. For example, an inverter’s delay is 1ns, while three input NOR gate has 3ns delay. Quantify the delay of the critical path.
10. Optimize your design to have minimum TRANSISTOR count. Draw the new implementation; for repetitive blocks show the transistor level design of only one of the elements (e.g. if you have 8 T-FFs, show transistor implementation of only one of them).
11. Does your design in part 10 have any static-0 or static-1 hazards? If so, identify one of each type and show how to remove it.
12. What is the critical path of the design in part 10? Gate delay = 1ns * \#gate_inputs & wires have 2ns delay. Quantify the delay of the critical path.
13. Calculate the maximum clock frequency of both designs (parts 9. & 12.). For all T-FFs use setup time/hold time of 2ns and clock skew of 3ns. Justify your answer.
14. Now assume that array of 256 8-bit values is stored in DRAM. Show how you’d design a special purpose DRAM chip for this problem. Draw all components needed for the data access to work. Show timing diagrams illustrating how to store and fetch 8-bit data to/from DRAM. How many transistors total between DRAM and the design shown in part 10?
15. Assume that the time for any type of data transfer between the rest of your design and DRAM is 7 clock cycles (the clock cycle time you calculated in part 13). Given this, estimate how long will your design take to get the final result?
16. What dominates the performance and area of your designs? What parts of your design do you think would consume the most energy?