This is most definitely a Mealy machine.

Verilog to simulate this FSM:

```verilog
module Problem1;
  reg [3:0] ROM[0:3];
  reg [2:0] Q;
  reg [1:0] A;
  reg Out;
  reg Clock;

  always begin
    Clock = ¬Clock;
    #('NC);
  end

  always @(posedge Clock) begin
    if (Q != #1 ROM[0][0]) Q = 1;
  end

  always @(A or Q) begin
    if (Out != #2 ROM[1][0]) Out = 1;
  end

  initial begin
    $readmemh("ROMTable.txt", ROM);
    Clock = 0;
    A = 1;
    Q = 0;
    #('NC + 2);
  end
endmodule
```

See attached waveform from ModelSim.
This setup time violation will make the suggestions in (a), (b) and (c) all completely ineffective. Thanks to the fact that all the schemes have only one VE_Clocked register any of them might result in a metastability problem.

Part d:

EECS 150
Fall 2003
Problem Set #6
Solutions
Problem 3 C Code

So in C you can only do one subtract at a time, for the project they should have MANY
subtracts at once, at least 8 subtractions in parallel.
We used 64 absolute difference modules in parallel and then 63 adders to sum up the
error.
We then do 81 of these finds per 8x8 kernel. 81 finds because there are 9x9 possible
positions for an 8x8 kernel in a 16x16 cell.
For our memory we used 96 blockrams, 32 for the kernel and 16 for the cell. It requires
that many because of the bandwidth. To do 64 subtractions per cycle we need 64 bytes of
kernel data and 64 bytes of cell data per clock cycle. That means we need at LEAST 32
blockrams for the kernel and 32 for the cell (each blockram has two 1Byte ports).
It should be possible to do 8 absolute differences per cycle and then take 8x9x9 = 648
cycles per cell, but that complicates the memory structure a LOT because you must be
constantly reading from SDRAM into the blockrams, whereas with 64 absolute
differences per cycle it is possible to fill the blockrams and then use all the data in them
in an alternating pattern.

#include <stdio.h>

void main(int argc, char* argv[]) {
    int block, error, i, j, minerror, mini, min;
    unsigned char image[2][256][256];
    FILE *fileptr;
    
    if (argc != 3) {
        printf("Motion\ntnimage: Motion prevframe frame\n");
        return;
    }
    
    fileptr = fopen(argv[1], "r");
    fscan(fileptr, "%d", &width);
    fscan(fileptr, "%d", &height);
    for (y = 0; y < height; y++)
        for (x = 0; x < width; x++)
            fscan(fileptr, "%d", &image[0][y][x]);
    fclose(fileptr);
    
    fileptr = fopen(argv[2], "r");
    fscan(fileptr, "%d", &Width);
    fscan(fileptr, "%d", &Height);
    for (y = 0; y < height; y++)
        for (x = 0; x < width; x++)
            fscan(fileptr, "%d", &image[1][y][x]);
    fclose(fileptr);
    
    for (block = 0; block < 8; block++)
        for (i = 0; i < 9; i++)
            for (j = 0; j < 9; j++)
                error = 0;
                for (x = 0; x < 8; x++)
                    for (y = 0; y < 8; y++)
                        temp = image[0][128 + y][block*8 +
24 + x] - image[1][128 + y + j][block*8 + x + j];
                        error += (temp < 0) ? -temp : temp;
                if (((i == 0) & (j == 0)) || (error <
minerror))
                    minerror = error;
                    mini = i;
                    min = j;
    
    printf("Block %d at (%d,%d) moved to (%d,%d) error was %d\n", block, (block*8 + 4), 128, (block*8 + mini, 120 + min), error);
}