EENG 284

–

Digital Design Lab

Lab7

–

1 Dimensional Cellular Automat

Lab Solutions

# Cellular Automata Module:

Table 1: The next state of a processing element depends on the current state and the number of alive neighbors.

|  |  |  |  |
| --- | --- | --- | --- |
| Cell | Current State | # Alive neighbors | Next State |
| C0 | Dead | 1 | Alive |
| C1 | Dead | 1 | Alive |
| C2 | Alive | 0 | Dead |
| C3 | Dead | 2 | Dead |
| C4 | Alive | 1 | Alive |
| C5 | Alive | 1 | Alive |
| C6 | Dead | 2 | Dead |
| C7 | Alive | 0 | Dead |

Verilog code for the cellularAutomata

module cellularAutomata(sButton, rButton, clk, reset, slideSwitches, currentLifeState, loadOrRunSlide );

input wire sButton, rButton;

output wire clk;

input wire reset;

input wire [16:0] slideSwitches;

input wire loadOrRunSlide;

output wire [16:0] currentLifeState;

wire srFeedback;

wire [7:0] rule;

wire [16:0] initialState;

assign rule = slideSwitches[7:0];

assign initialState = slideSwitches;

singleCell arrayCell0(clk, reset, rule, currentLifeState[0], initialState[0], {currentLifeState[1],currentLifeState[0],currentLifeState[16]}, loadOrRunSlide);

singleCell arrayCell(clk, reset, rule, currentLifeState[1], initialState[1], {currentLifeState[2],currentLifeState[1],currentLifeState[0]}, loadOrRunSlide);

singleCell arrayCell(clk, reset, rule, currentLifeState[2], initialState[2], {currentLifeState[3],currentLifeState[2],currentLifeState[1]}, loadOrRunSlide);

singleCell arrayCell16(clk, reset, rule, currentLifeState[16], initialState[16], {currentLifeState[0],currentLifeState[16],currentLifeState[15]}, loadOrRunSlide);

assign clk = ~(rButton | srFeedback);

assign srFeedback = ~(sButton | clk);

endmodule

# singleCell module:

Table 2: The input/output relationship for the nextState functionality in Figure 7.

|  |  |  |
| --- | --- | --- |
| {n+, n, n-} | nextState | Rule bit |
| 3’b000 | 0 | rule[0] |
| 3’b001 | 1 | rule[1] |
| 3’b010 | 0 | rule[2] |
| 3’b011 | 1 | rule[3] |
| 3’b100 | 1 | rule[4] |
| 3’b101 | 0 | rule[5] |
| 3’b110 | 1 | rule[6] |
| 3’b111 | 0 | rule[7] |

module singleCell(clk, reset, rule, currentState, initialState, neighborhoodState, loadOrRun);

input wire clk, reset;

input wire [7:0] rule;

output wire currentState;

input wire initialState;

input wire [2:0] neighborhoodState;

input wire loadOrRun;

reg nextLife;

wire din;

//------------------------------------------------------------------

// current pattern 111 110 101 100 011 010 001 000

// new state for center cell 1 0 0 1 0 0 0 1

//------------------------------------------------------------------

always @(\*)

case (neighborhoodState)

3'b000: nextLife = rule[0];

3'b001: nextLife = rule[1];

3'b010: nextLife = rule[2];

3'b011: nextLife = rule[3];

3'b100: nextLife = rule[4];

3'b101: nextLife = rule[5];

3'b110: nextLife = rule[6];

3'b111: nextLife = rule[7];

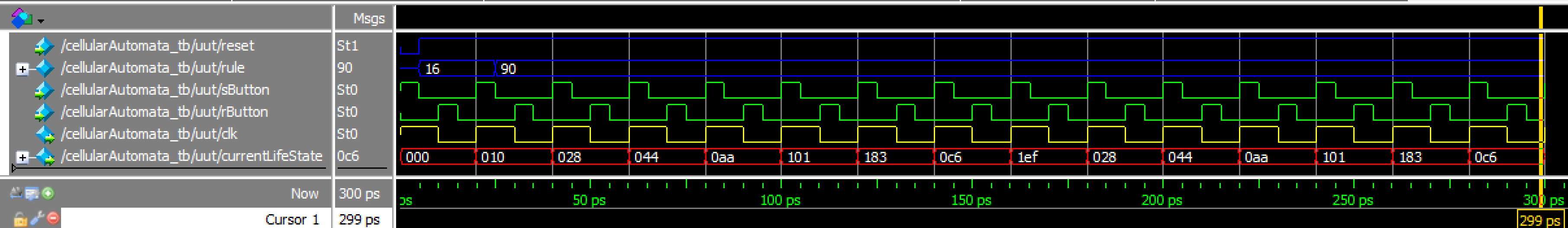
endcase

genericMux2x1 #(1) nextStateMux(nextLife, initialState, loadOrRun, din);

dffNegEdge arrayCell(clk, reset, din, currentState);

endmodule

**Overall Design:**



Pin Assignment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| slide[9] | PIN\_AE19 |  | clk  LEDG6 | PIN\_H8 |
| slide[8] | PIN\_Y11 |  | led [8]  LEDR8 | PIN\_H7 |
| slide[7] | PIN\_AC10 |  | led [7] | PIN\_K8 |
| slide[6] | PIN\_V10 |  | led [6] | PIN\_K10 |
| slide[5] | PIN\_AB10 |  | led [5] | PIN\_J7 |
| slide[4] | PIN\_W11 |  | led [4] | PIN\_J8 |
| slide[3] | PIN\_AC8 |  | led [3] | PIN\_G7 |
| slide[2] | PIN\_AD13 |  | led [2] | PIN\_G6 |
| slide[1] | PIN\_AE10 |  | led [1] | PIN\_F6 |
| slide[0] | PIN\_AC9 |  | led [0]  LEDR0 | PIN\_F7 |

|  |  |  |
| --- | --- | --- |
| sClk | Key[3] | PIN\_Y16 |
| rClk | Key[2] | PIN\_Y15 |
| reset | Key[1] | PIN\_P12 |