Computer Assignment #1

Computer Architecture

Seyyed Sina Alizadeh Tabatabai - 810101477
Majid Sadeghinezhad Saryazdi - 810101459
Fall 2024
Prof. : Dr.Safari

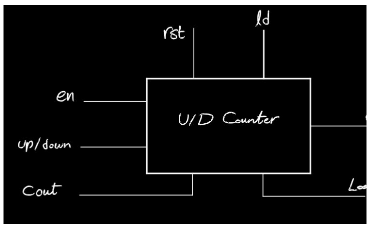
Abstract

The Goal for this project is designing hardware for solving the famous eight queens problem. The “Safety Check” module is designed to get the coordinates of two cells, check if two queens positioned in the said cells threaten each other, and give the “threat” signal as the output. This process is done for every cell that belongs to previous columns. The cell is considered unsafe the moment a threat arises, and it is considered safe if none of the previously positioned queens threaten the cell. If none of the rows in a column are safe, the hardware backtracks to change the position of the queen placed in the previous column. This whole process gets done until the position of the last column’s queen is determined. “done” signal becomes one after this step and the board contents are displayed column by column on the output (in the format 8’b00001000 where 1 is the position of the queen in the column) in eight clock cycles.

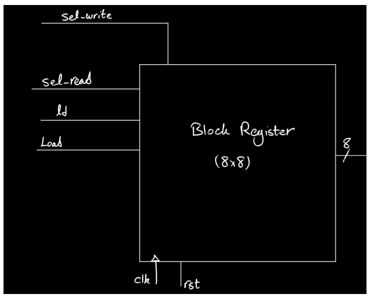
Data Path and Controller Design

- Data Path
- Needed Components:

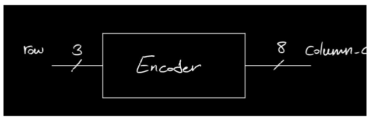
- U/D Counter



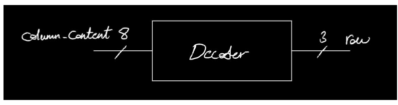
- 8x8 Register



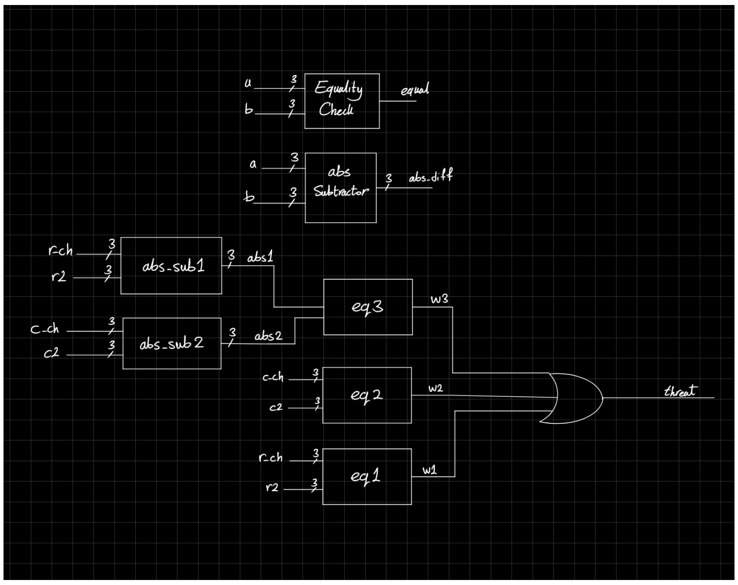
- Encoder



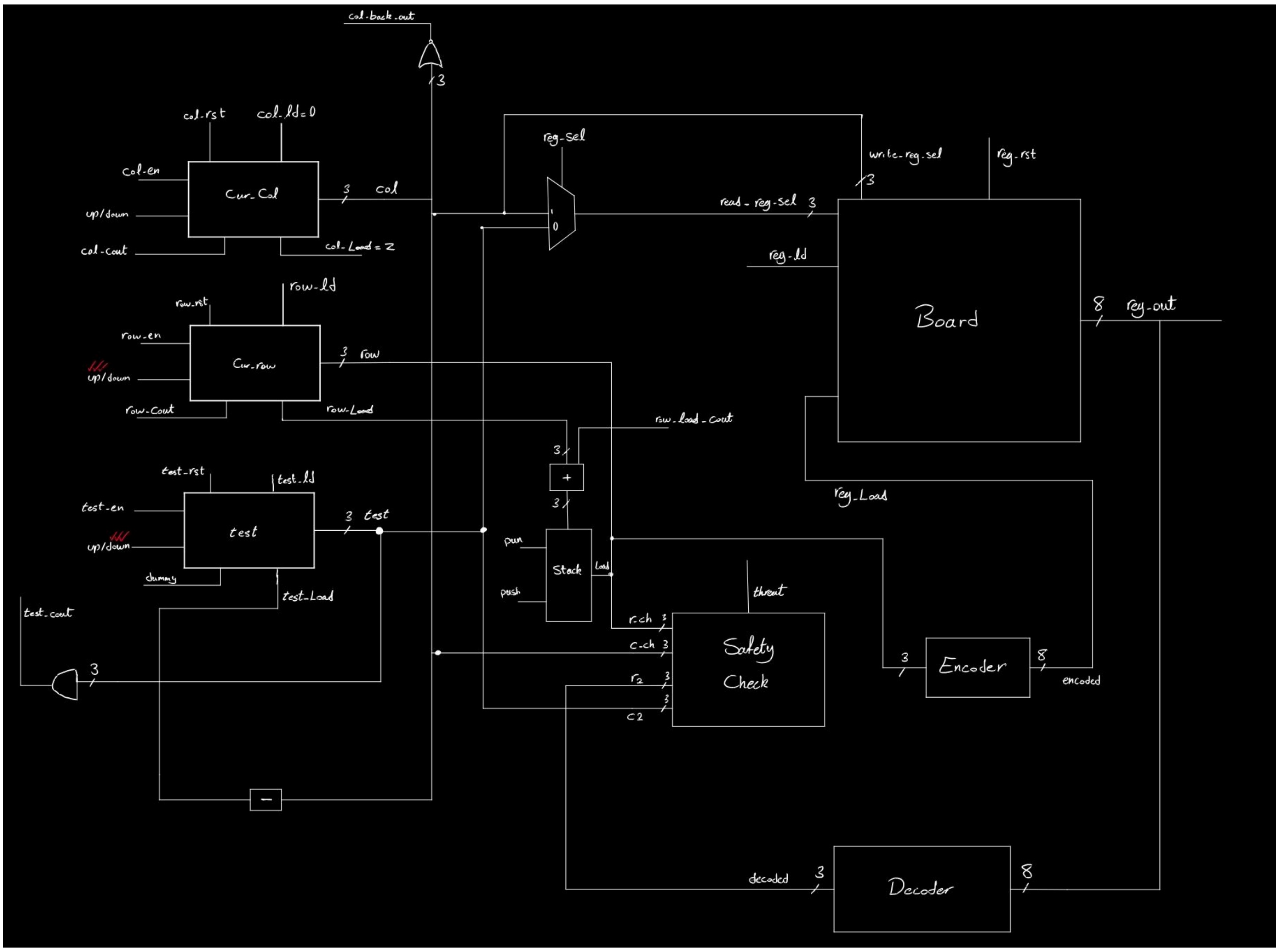
- Decoder



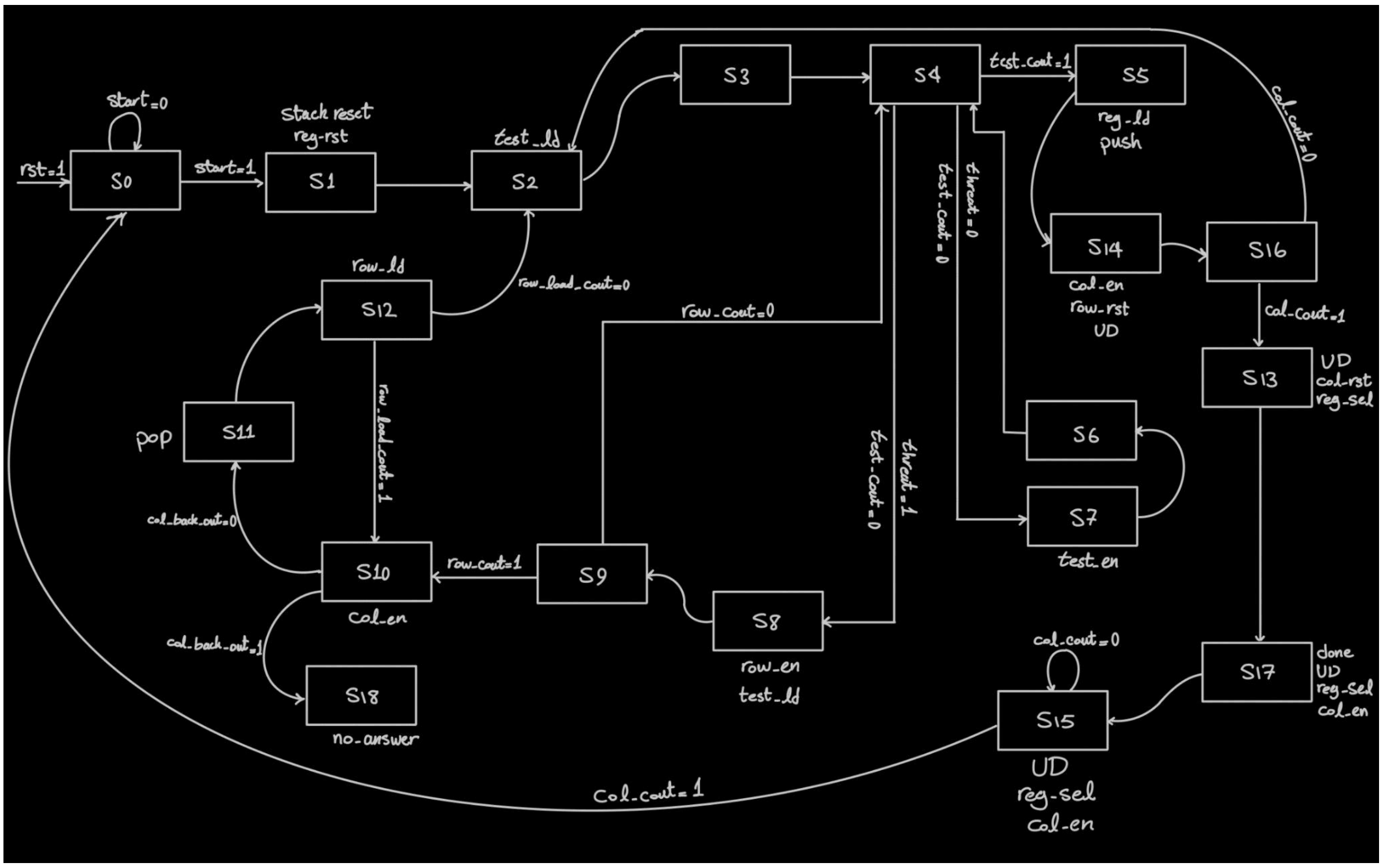
- Safety Check Module



- Data Path:



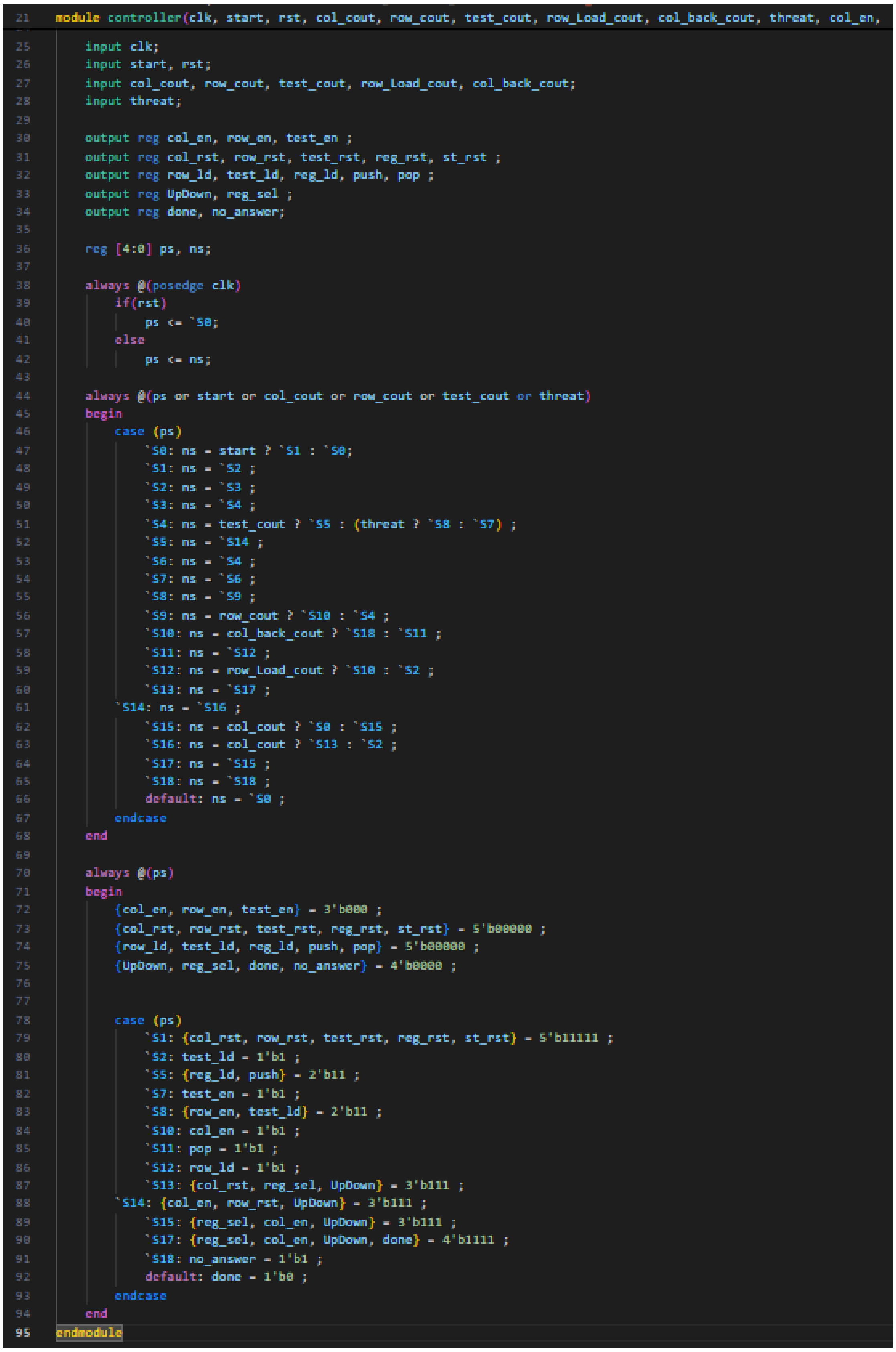
- Controller
- State Transition Signals:
  - start
  - test_cout
  - threat
  - col_cout
  - row_cout
  - row_load_cout
  - col_back_out
- Signals Changed in States:
  - rst
  - reg_rst
  - test_ld
  - reg_ld
  - col_en
  - row_rst
  - UD
  - col_rst
  - reg_sel
  - done
  - no_answer
  - row_ld
  - test_en
  - row_en
- FSM:



-

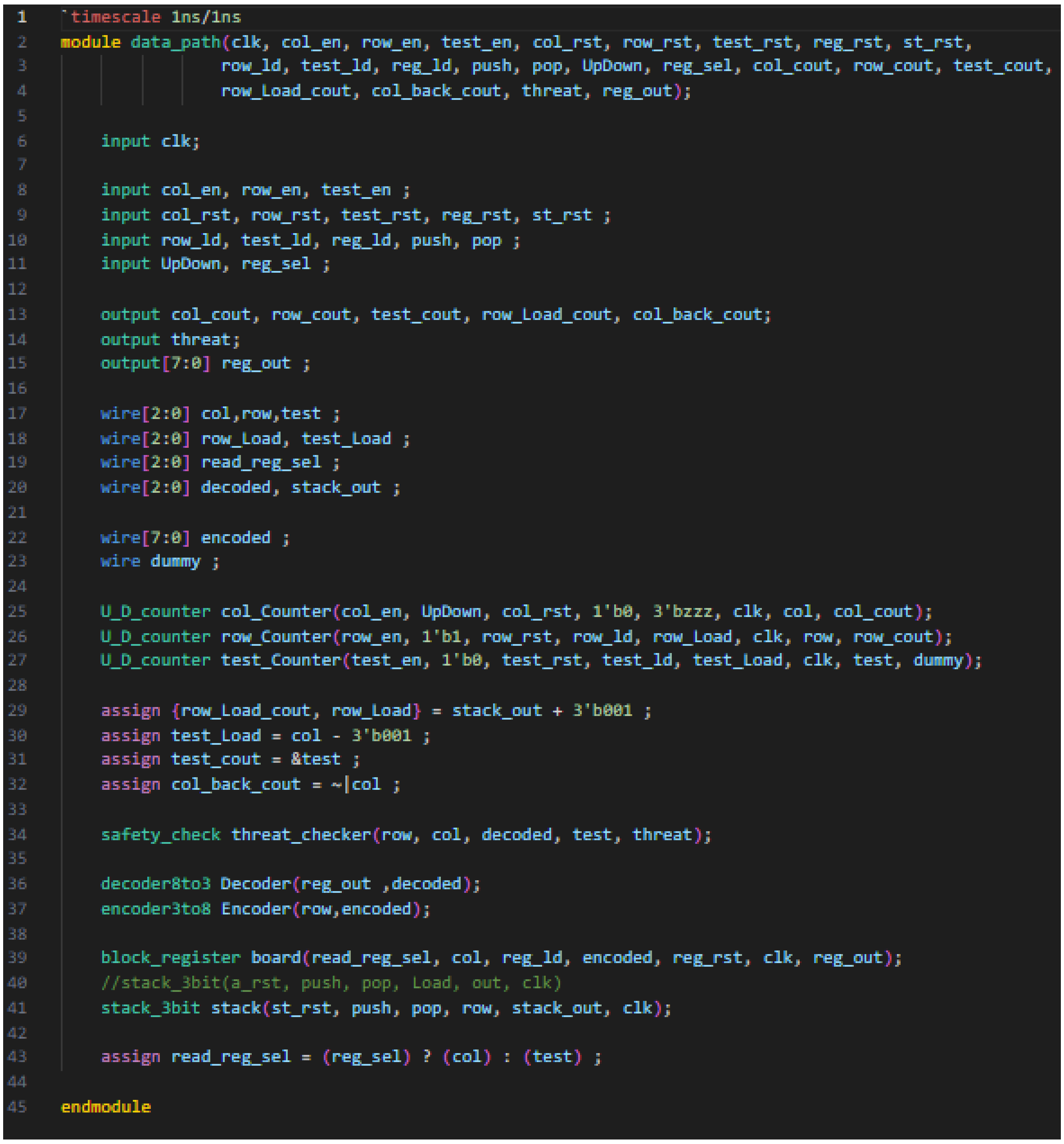
System Verilog Code

- Controller’s Always Blocks:



```
module controller(clk, start, rst, col_cout, row_cout, test_cout, row_Load_cout, col_back_cout, threat, col_en,
    input clk;
    input start, rst;
    input col_cout, row_cout, test_cout, row_Load_cout, col_back_cout;
    input threat;

    output reg col_en, row_en, test_en ;
    output reg col_rst, row_rst, test_rst, reg_rst, st_rst ;
    output reg row_ld, test_ld, reg_ld, push, pop ;
    output reg UpDown, reg_sel ;
    output reg done, no_answer;

    reg [4:0] ps, ns;

    always @(posedge clk)
        if(rst)
            ps <= `S0;
        else
            ps <= ns;

    always @(ps or start or col_cout or row_cout or test_cout or threat)
    begin
        case (ps)
            `S0: ns = start ? `S1 : `S0;
            `S1: ns = `S2 ;
            `S2: ns = `S3 ;
            `S3: ns = `S4 ;
            `S4: ns = test_cout ? `S5 : (threat ? `S8 : `S7) ;
            `S5: ns = `S14 ;
            `S6: ns = `S4 ;
            `S7: ns = `S6 ;
            `S8: ns = `S9 ;
            `S9: ns = row_cout ? `S10 : `S4 ;
            `S10: ns = col_back_cout ? `S18 : `S11 ;
            `S11: ns = `S12 ;
            `S12: ns = row_Load_cout ? `S10 : `S2 ;
            `S13: ns = `S17 ;
        `S14: ns = `S16 ;
            `S15: ns = col_cout ? `S0 : `S15 ;
            `S16: ns = col_cout ? `S13 : `S2 ;
            `S17: ns = `S15 ;
            `S18: ns = `S18 ;
            default: ns = `S0 ;
        endcase
    end

    always @(ps)
    begin
        {col_en, row_en, test_en} = 3'b000 ;
        {col_rst, row_rst, test_rst, reg_rst, st_rst} = 5'b00000 ;
        {row_ld, test_ld, reg_ld, push, pop} = 5'b00000 ;
        {UpDown, reg_sel, done, no_answer} = 4'b0000 ;


        case (ps)
            `S1: {col_rst, row_rst, test_rst, reg_rst, st_rst} = 5'b11111 ;
            `S2: test_ld = 1'b1 ;
            `S5: {reg_ld, push} = 2'b11 ;
            `S7: test_en = 1'b1 ;
            `S8: {row_en, test_ld} = 2'b11 ;
            `S10: col_en = 1'b1 ;
            `S11: pop = 1'b1 ;
            `S12: row_ld = 1'b1 ;
            `S13: {col_rst, reg_sel, UpDown} = 3'b111 ;
        `S14: {col_en, row_rst, UpDown} = 3'b111 ;
            `S15: {reg_sel, col_en, UpDown} = 3'b111 ;
            `S17: {reg_sel, col_en, UpDown, done} = 4'b1111 ;
            `S18: no_answer = 1'b1 ;
            default: done = 1'b0 ;
        endcase
    end
endmodule
```

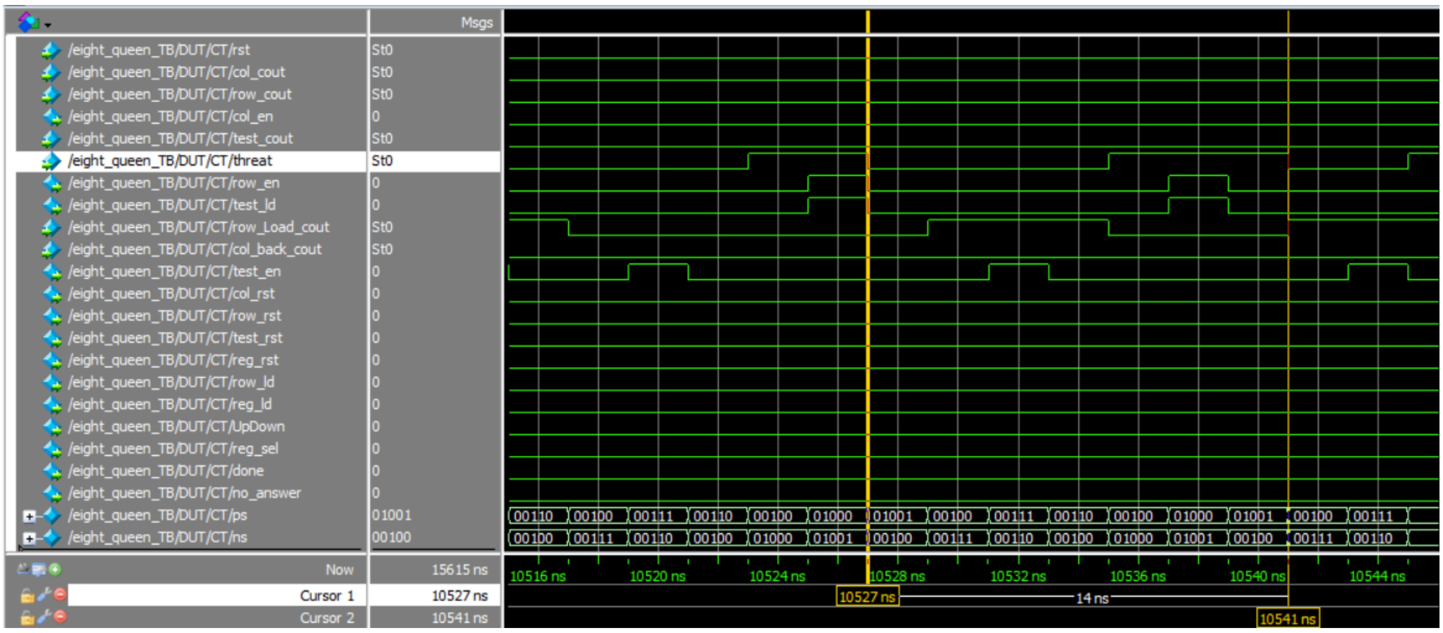
- Data_Path’s Code:



```verilog
`timescale 1ns/1ns
module data_path(clk, col_en, row_en, test_en, col_rst, row_rst, test_rst, reg_rst, st_rst,
                row_ld, test_ld, reg_ld, push, pop, UpDown, reg_sel, col_cout, row_cout, test_cout,
                row_Load_cout, col_back_cout, threat, reg_out);

    input clk;

    input col_en, row_en, test_en ;
    input col_rst, row_rst, test_rst, reg_rst, st_rst ;
    input row_ld, test_ld, reg_ld, push, pop ;
    input UpDown, reg_sel ;

    output col_cout, row_cout, test_cout, row_Load_cout, col_back_cout;
    output threat;
    output[7:0] reg_out ;

    wire[2:0] col,row,test ;
    wire[2:0] row_Load, test_Load ;
    wire[2:0] read_reg_sel ;
    wire[2:0] decoded, stack_out ;

    wire[7:0] encoded ;
    wire dummy ;

    U_D_counter col_Counter(col_en, UpDown, col_rst, 1'b0, 3'bzzz, clk, col, col_cout);
    U_D_counter row_Counter(row_en, 1'b1, row_rst, row_ld, row_Load, clk, row, row_cout);
    U_D_counter test_Counter(test_en, 1'b0, test_rst, test_ld, test_Load, clk, test, dummy);

    assign {row_Load_cout, row_Load} = stack_out + 3'b001 ;
    assign test_Load = col - 3'b001 ;
    assign test_cout = &test ;
    assign col_back_cout = ~|col ;

    safety_check threat_checker(row, col, decoded, test, threat);

    decoder8to3 Decoder(reg_out ,decoded);
    encoder3to8 Encoder(row,encoded);

    block_register board(read_reg_sel, col, reg_ld, encoded, reg_rst, clk, reg_out);
    //stack_3bit(a_rst, push, pop, Load, out, clk)
    stack_3bit stack(st_rst, push, pop, row, stack_out, clk);

    assign read_reg_sel = (reg_sel) ? (col) : (test) ;

endmodule
```

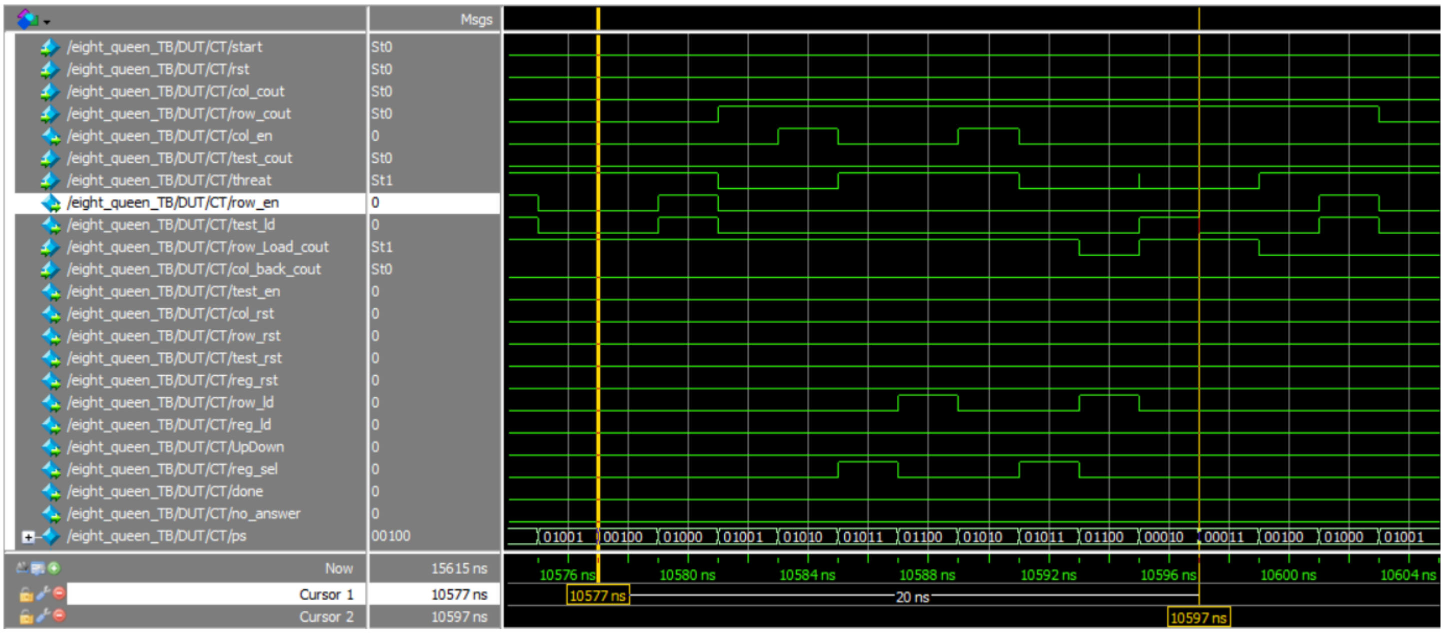
Results

- Threat Detection Example In Waveform



```
Msgs
/eight_queen_TB/DUT/CT/rst              St0
/eight_queen_TB/DUT/CT/col_cout         St0
/eight_queen_TB/DUT/CT/row_cout         St0
/eight_queen_TB/DUT/CT/col_en           0
/eight_queen_TB/DUT/CT/test_cout        St0
/eight_queen_TB/DUT/CT/threat           St0
/eight_queen_TB/DUT/CT/row_en           0
/eight_queen_TB/DUT/CT/test_ld          0
/eight_queen_TB/DUT/CT/row_Load_cout    St0
/eight_queen_TB/DUT/CT/col_back_cout    St0
/eight_queen_TB/DUT/CT/test_en          0
/eight_queen_TB/DUT/CT/col_rst          0
/eight_queen_TB/DUT/CT/row_rst          0
/eight_queen_TB/DUT/CT/test_rst         0
/eight_queen_TB/DUT/CT/reg_rst          0
/eight_queen_TB/DUT/CT/row_ld           0
/eight_queen_TB/DUT/CT/reg_ld           0
/eight_queen_TB/DUT/CT/UpDown           0
/eight_queen_TB/DUT/CT/reg_sel          0
/eight_queen_TB/DUT/CT/done             0
/eight_queen_TB/DUT/CT/no_answer        0
/eight_queen_TB/DUT/CT/ps               01001
/eight_queen_TB/DUT/CT/ns               00100

ps: 00110 00100 00111 00110 00100 01000 01001 00100 00111 00110 00100 01000 01001 00100 00111
ns: 00100 00111 00110 00100 01000 01001 00100 00111 00110 00100 01000 01001 00100 00111 00110

Now       15615 ns
Cursor 1  10527 ns
Cursor 2  10541 ns

10516 ns  10520 ns  10524 ns  10528 ns  10532 ns  10536 ns  10540 ns  10544 ns
10527 ns  14 ns  10541 ns
```

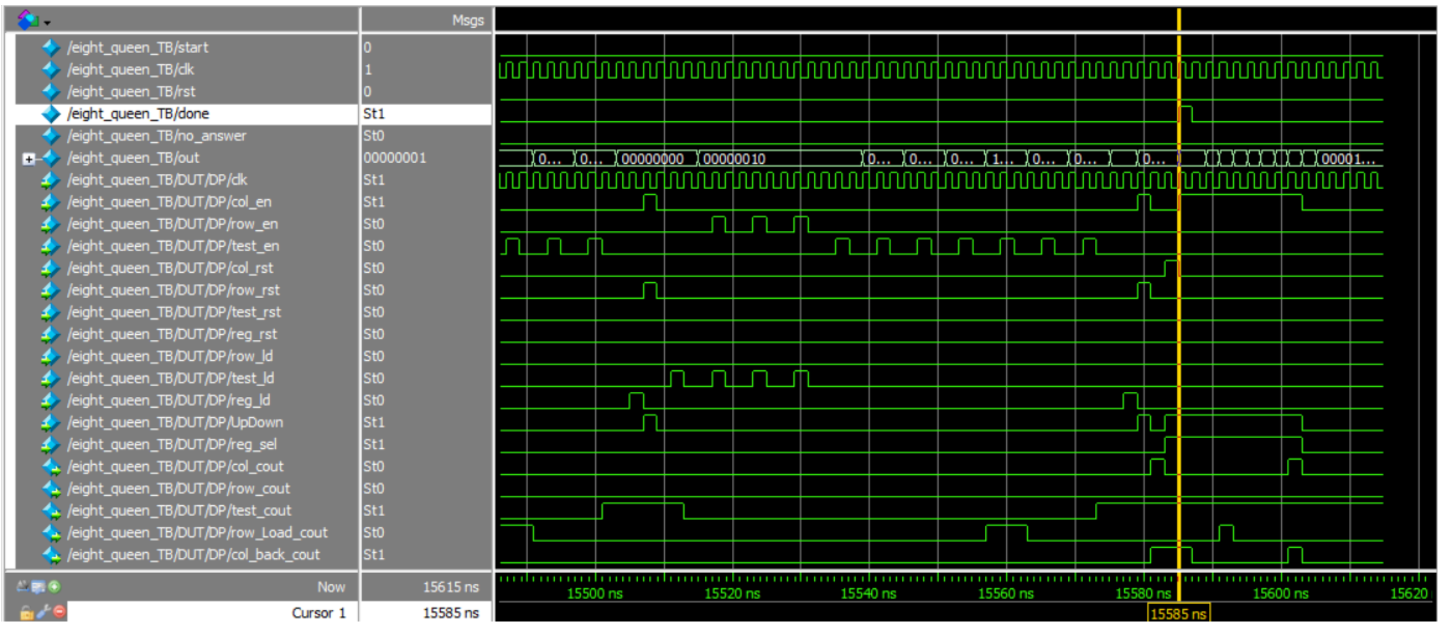
- Backtrack Example In Waveform



```
Msgs
/eight_queen_TB/DUT/CT/start            St0
/eight_queen_TB/DUT/CT/rst              St0
/eight_queen_TB/DUT/CT/col_cout         St0
/eight_queen_TB/DUT/CT/row_cout         St0
/eight_queen_TB/DUT/CT/col_en           0
/eight_queen_TB/DUT/CT/test_cout        St0
/eight_queen_TB/DUT/CT/threat           St1
/eight_queen_TB/DUT/CT/row_en           0
/eight_queen_TB/DUT/CT/test_ld          0
/eight_queen_TB/DUT/CT/row_Load_cout    St1
/eight_queen_TB/DUT/CT/col_back_cout    St0
/eight_queen_TB/DUT/CT/test_en          0
/eight_queen_TB/DUT/CT/col_rst          0
/eight_queen_TB/DUT/CT/row_rst          0
/eight_queen_TB/DUT/CT/test_rst         0
/eight_queen_TB/DUT/CT/reg_rst          0
/eight_queen_TB/DUT/CT/row_ld           0
/eight_queen_TB/DUT/CT/reg_ld           0
/eight_queen_TB/DUT/CT/UpDown           0
/eight_queen_TB/DUT/CT/reg_sel          0
/eight_queen_TB/DUT/CT/done             0
/eight_queen_TB/DUT/CT/no_answer        0
/eight_queen_TB/DUT/CT/ps               00100

ps: 01001 00100 01000 01001 01010 01011 01100 01010 01011 01100 00010 00011 00100 01000 01001

Now       15615 ns
Cursor 1  10577 ns
Cursor 2  10597 ns

10576 ns  10580 ns  10584 ns  10588 ns  10592 ns  10596 ns  10600 ns  10604 ns
10577 ns  20 ns  10597 ns
```

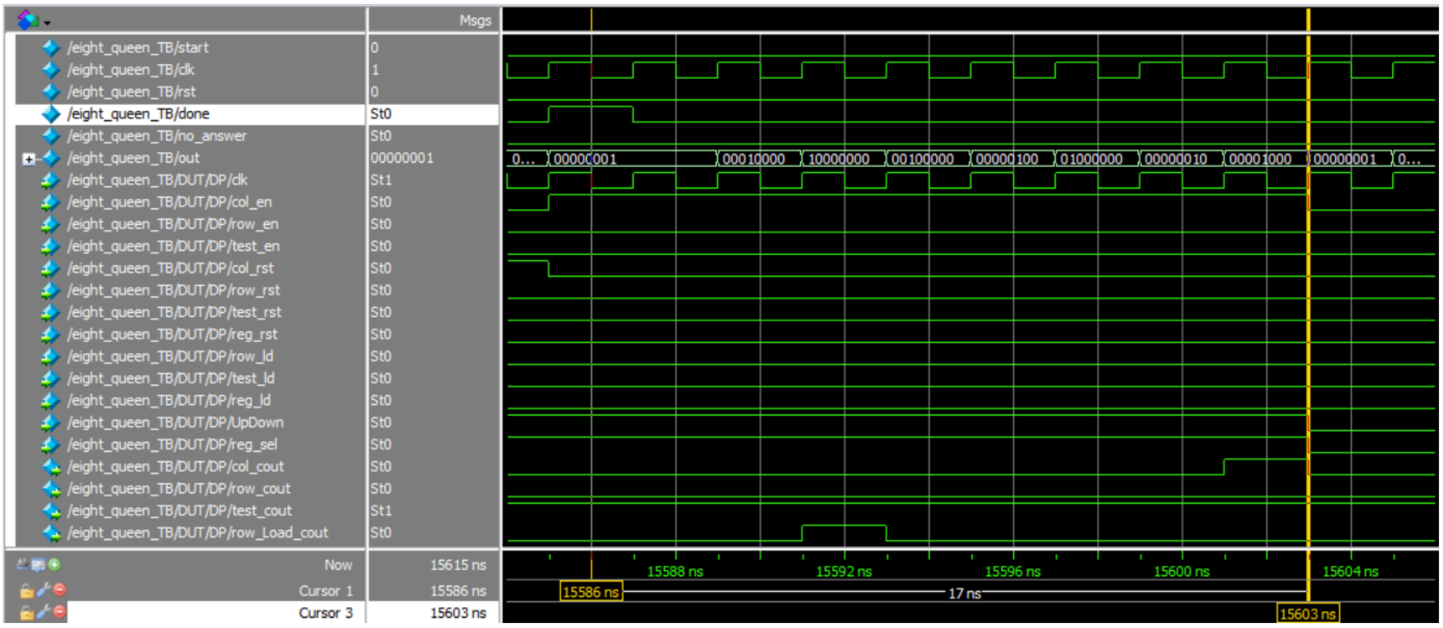
In this part of the waveform, we can see the transition S4->S8->S9->S10>S11->S12->S10->S11->S12->S2 and the change in the corresponding signals in the same way designed in the FSM.

- Done Signal



```
                                     Msgs
/eight_queen_TB/start                0
/eight_queen_TB/clk                  1
/eight_queen_TB/rst                  0
/eight_queen_TB/done                 St1
/eight_queen_TB/no_answer            St0
/eight_queen_TB/out                  00000001
/eight_queen_TB/DUT/DP/clk           St1
/eight_queen_TB/DUT/DP/col_en        St1
/eight_queen_TB/DUT/DP/row_en        St0
/eight_queen_TB/DUT/DP/test_en       St0
/eight_queen_TB/DUT/DP/col_rst       St0
/eight_queen_TB/DUT/DP/row_rst       St0
/eight_queen_TB/DUT/DP/test_rst      St0
/eight_queen_TB/DUT/DP/reg_rst       St0
/eight_queen_TB/DUT/DP/row_ld        St0
/eight_queen_TB/DUT/DP/test_ld       St0
/eight_queen_TB/DUT/DP/reg_ld        St0
/eight_queen_TB/DUT/DP/UpDown        St1
/eight_queen_TB/DUT/DP/reg_sel       St1
/eight_queen_TB/DUT/DP/col_cout      St0
/eight_queen_TB/DUT/DP/row_cout      St0
/eight_queen_TB/DUT/DP/test_cout     St1
/eight_queen_TB/DUT/DP/row_Load_cout St0
/eight_queen_TB/DUT/DP/col_back_cout St1
Now                                  15615 ns
Cursor 1                             15585 ns
```

- Output



```
                                     Msgs
/eight_queen_TB/start                0
/eight_queen_TB/clk                  1
/eight_queen_TB/rst                  0
/eight_queen_TB/done                 St0
/eight_queen_TB/no_answer            St0
/eight_queen_TB/out                  00000001
/eight_queen_TB/DUT/DP/clk           St1
/eight_queen_TB/DUT/DP/col_en        St0
/eight_queen_TB/DUT/DP/row_en        St0
/eight_queen_TB/DUT/DP/test_en       St0
/eight_queen_TB/DUT/DP/col_rst       St0
/eight_queen_TB/DUT/DP/row_rst       St0
/eight_queen_TB/DUT/DP/test_rst      St0
/eight_queen_TB/DUT/DP/reg_rst       St0
/eight_queen_TB/DUT/DP/row_ld        St0
/eight_queen_TB/DUT/DP/test_ld       St0
/eight_queen_TB/DUT/DP/reg_ld        St0
/eight_queen_TB/DUT/DP/UpDown        St0
/eight_queen_TB/DUT/DP/reg_sel       St0
/eight_queen_TB/DUT/DP/col_cout      St0
/eight_queen_TB/DUT/DP/row_cout      St0
/eight_queen_TB/DUT/DP/test_cout     St1
/eight_queen_TB/DUT/DP/row_Load_cout St0
Now                                  15615 ns
Cursor 1                             15586 ns
Cursor 3                             15603 ns
```

which translates into:

		Q					
					Q		
			Q				
	Q						
							Q
				Q			
						Q	
Q							