```
`timescale 1ns / 1ps
/*********************************************
********************************
*
* Module: Codebreaker
*
* Author: Eric Christie
* Class: ECEN 220, Section 1, Winter 2021
* Date: 3/30/21
*
* Description: Instances into a larger RC4
codebreaker module
*
*
*********************************************
********************************/

`default_nettype none
module Codebreaker(
    input wire logic clk, reset, start,
    output logic [15:0] key_display,
    output logic stopwatch_run,
draw_plaintext,
    input wire logic
done_drawing_plaintext,
    output logic [127:0]
plaintext_to_draw);

    /*assign key_display = 0;   //Ex 1
    assign stopwatch_run = 1;
    assign plaintext_to_draw =
{"HELLO123"};
    assign draw_plaintext = start;*/

    //Logic wires
    logic [23:0] key;
    logic [127:0] cyphertext;
    logic decrypted, begin_decryption;
    logic plaintext_is_ascii;

    assign key_display = {key[23:8]};

    assign plaintext_is_ascii =
((plaintext_to_draw[127:120] >= "A" &&
plaintext_to_draw[127:120] <= "Z") ||
(plaintext_to_draw[127:120] >= "0" &&
plaintext_to_draw[127:120] <= "9") ||
(plaintext_to_draw[127:120] == " ")) &&
((plaintext_to_draw[119:112] >= "A" &&
plaintext_to_draw[119:112] <= "Z") ||
(plaintext_to_draw[119:112] >= "0" &&
plaintext_to_draw[119:112] <= "9") ||
(plaintext_to_draw[119:112] == " ")) &&

((plaintext_to_draw[111:104] >= "A" &&
plaintext_to_draw[111:104] <= "Z") ||
(plaintext_to_draw[111:104] >= "0" &&
plaintext_to_draw[111:104] <= "9") ||
(plaintext_to_draw[111:104] == " ")) &&

((plaintext_to_draw[103:96] >= "A" &&
plaintext_to_draw[103:96] <= "Z") ||
(plaintext_to_draw[103:96] >= "0" &&
plaintext_to_draw[103:96] <= "9") ||
(plaintext_to_draw[103:96] == " ")) &&

((plaintext_to_draw[95:88] >= "A" &&
plaintext_to_draw[95:88] <= "Z") ||
(plaintext_to_draw[95:88] >= "0" &&
plaintext_to_draw[95:88] <= "9") ||
(plaintext_to_draw[95:88] == " ")) &&
((plaintext_to_draw[87:80] >= "A" &&
plaintext_to_draw[87:80] <= "Z") ||
(plaintext_to_draw[87:80] >= "0" &&
plaintext_to_draw[87:80] <= "9") ||
(plaintext_to_draw[87:80] == " ")) &&

((plaintext_to_draw[79:72] >= "A" &&
plaintext_to_draw[79:72] <= "Z") ||
(plaintext_to_draw[79:72] >= "0" &&
plaintext_to_draw[79:72] <= "9") ||
(plaintext_to_draw[79:72] == " ")) &&

((plaintext_to_draw[71:64] >= "A" &&
plaintext_to_draw[71:64] <= "Z") ||
(plaintext_to_draw[71:64] >= "0" &&
plaintext_to_draw[71:64] <= "9") ||
(plaintext_to_draw[71:64] == " ")) &&

((plaintext_to_draw[63:56] >= "A" &&
plaintext_to_draw[63:56] <= "Z") ||
(plaintext_to_draw[63:56] >= "0" &&
plaintext_to_draw[63:56] <= "9") ||
(plaintext_to_draw[63:56] == " ")) &&
((plaintext_to_draw[55:48] >= "A" &&
plaintext_to_draw[55:48] <= "Z") ||
(plaintext_to_draw[55:48] >= "0" &&
plaintext_to_draw[55:48] <= "9") ||
(plaintext_to_draw[55:48] == " ")) &&

((plaintext_to_draw[47:40] >= "A" &&
plaintext_to_draw[47:40] <= "Z") ||
(plaintext_to_draw[47:40] >= "0" &&
plaintext_to_draw[47:40] <= "9") ||
(plaintext_to_draw[47:40] == " ")) &&

((plaintext_to_draw[39:32] >= "A" &&
plaintext_to_draw[39:32] <= "Z") ||
(plaintext_to_draw[39:32] >= "0" &&
plaintext_to_draw[39:32] <= "9") ||
(plaintext_to_draw[39:32] == " ")) &&

((plaintext_to_draw[31:24] >= "A" &&
plaintext_to_draw[31:24] <= "Z") ||
(plaintext_to_draw[31:24] >= "0" &&
plaintext_to_draw[31:24] <= "9") ||
(plaintext_to_draw[31:24] == " ")) &&
((plaintext_to_draw[23:16] >= "A" &&
plaintext_to_draw[23:16] <= "Z") ||
(plaintext_to_draw[23:16] >= "0" &&
plaintext_to_draw[23:16] <= "9") ||
(plaintext_to_draw[23:16] == " ")) &&

((plaintext_to_draw[15:8] >= "A" &&
plaintext_to_draw[15:8] <= "Z") ||
(plaintext_to_draw[15:8] >= "0" &&
plaintext_to_draw[15:8] <= "9") ||
(plaintext_to_draw[15:8] == " ")) &&

((plaintext_to_draw[7:0] >= "A" &&
plaintext_to_draw[7:0] <= "Z") ||
(plaintext_to_draw[7:0] >= "0" &&
plaintext_to_draw[7:0] <= "9") ||
(plaintext_to_draw[7:0] == " "));

    //RC4 module    clk, reset, enable,
[24] key, [128] bytes_in, [128] bytes_out,
done
    decrypt_rc4 RC4(.clk(clk),
.reset(reset), .enable(begin_decryption),
.key(key), .bytes_in(cyphertext),
.bytes_out(plaintext_to_draw),
.done(decrypted));

    //assign key and cyphertext
    assign cyphertext =
128'ha13a3ab3071897088f3233a58d6238bb;

    //FSM
    typedef enum logic [2:0] {Wait,
Decrypting, Check, Drawing, Terminate,
ERR='X} StateType;
    StateType ns, cs;

    always_comb begin
        ns = ERR;
        begin_decryption = 0;
        stopwatch_run = 0;
        draw_plaintext = 0;

        if(reset) ns = Wait;
        else
            case(cs)
                Wait: begin
```

```
                    if(start) ns =
Decrypting;
                    else ns = Wait;
                end

                Decrypting: begin
                    begin_decryption = 1;
                    stopwatch_run = 1;
                    if(decrypted) ns =
Check;
                    else ns = Decrypting;
                end

                Check: begin
                    stopwatch_run = 1;
                    if(plaintext_is_ascii)
ns = Drawing;
                    else ns = Decrypting;

                end

                Drawing: begin
                    draw_plaintext = 1;
```

```
if(done_drawing_plaintext) ns = Terminate;
                    else ns = Drawing;
                end

                Terminate:
                    if(!reset) ns =
Terminate;
            endcase
    end

    always_ff @(posedge clk) begin
        cs <= ns;
        if((cs == Check) && (ns ==
Decrypting)) key <= key + 1;
        else if(cs == Wait) key = 0;
    end

endmodule
```