CE-325: Digital System Design

Final Exam - Take Home Q5

Huzaifah Tariq Ahmed - ha07151

6th May 2024

1 Finite State Machine (FSM) Diagrams

1.1 TCP Client

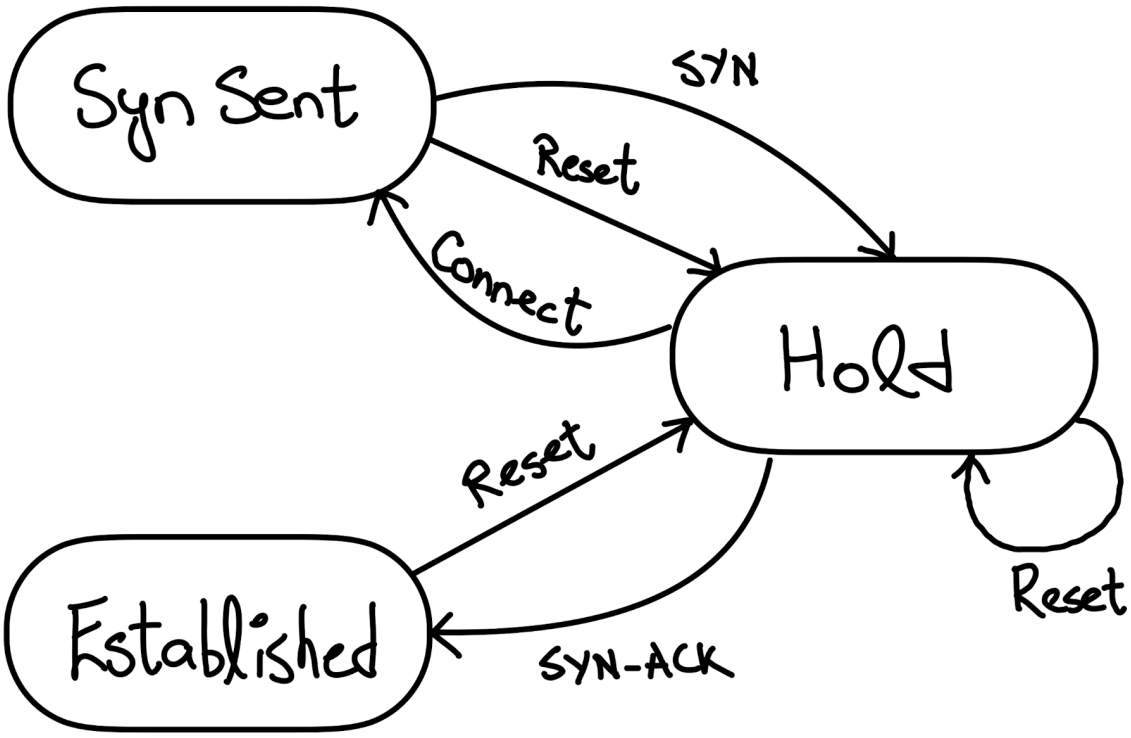

Figure 1: TCP Client FSM Diagram

1.2 TCP Server

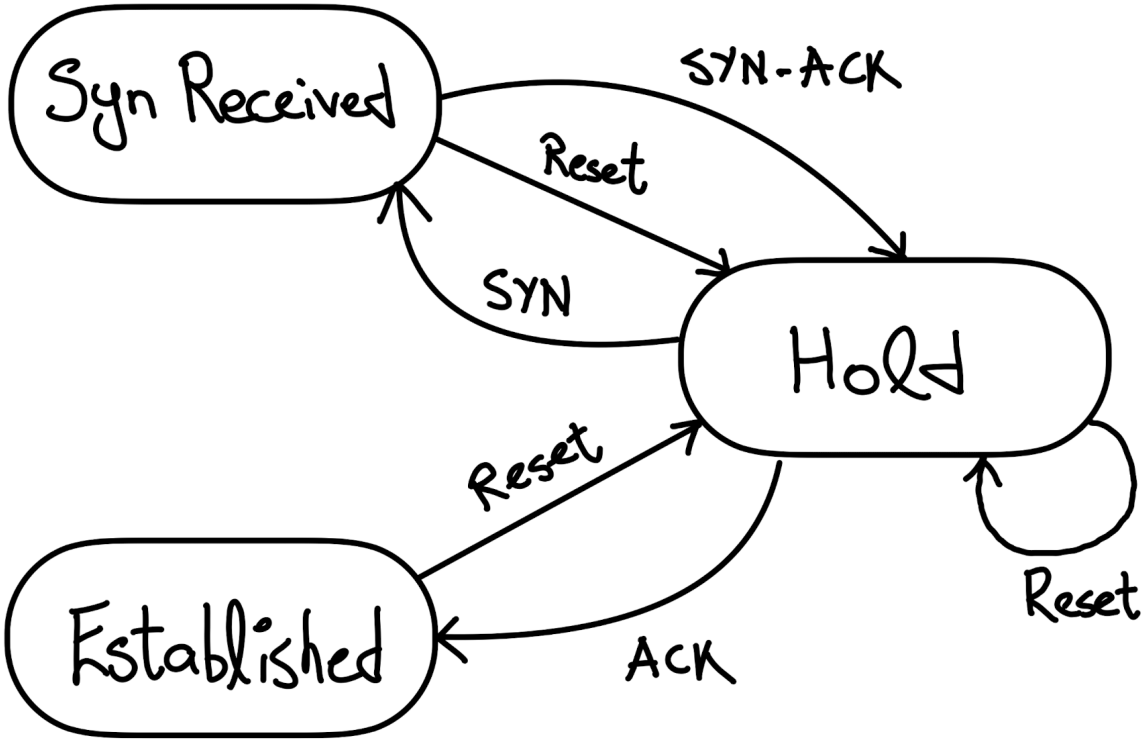

Figure 2: TCP Server FSM Diagram

2 FSM Design In Verilog

2.1 TCP Top Module

2.1.1 Code

```
1  `timescale 1ns / 1ps
2
3  module tcp(
4      reset,
5      clk,
6      connect
7      );
8
9      input reset, clk, connect;
10
11     wire SYN, ACK, SYN_ACK;
12
```

```
    client c1(reset,
              clk,
              connect,
              SYN_ACK,
              SYN,
              ACK
              );

    server s1(reset,
              clk,
              SYN,
              ACK,
              SYN_ACK
              );

endmodule
```

2.2 TCP Client

2.2.1 Code

```
`timescale 1ns / 1ps

module client(
    reset,
    clk,
    connect,
    SYN_ACK,
    SYN,
    ACK
    );

    input reset,
          connect,
          clk,
          SYN_ACK;

    output reg SYN, ACK;

    parameter hold = 3'b1xx;
    parameter syn_sent  = 3'b010;
    parameter established  = 3'b011;

    reg [2:0] state,
              next_state;

    always @ (posedge clk)
    begin
        if (reset)
```

```
29            begin state <= hold;
30                            SYN <= 0;
31                  ACK <= 0;
32            end
33          else state <= next_state;
34      end
35
36
37      always @ (state or connect or SYN_ACK or posedge clk)
38      begin
39          casex(state)
40              hold: begin
41                      if (reset) next_state <= hold;
42                      else if (connect & SYN_ACK) next_state <= established;
43                      else if (connect) next_state <= syn_sent;
44                      else next_state <= hold;
45                    end
46
47              syn_sent:  begin
48                           if (reset) next_state <= hold;
49                           else if (connect)
50                             begin next_state <= hold;
51                                   SYN <= 1;
52                             end
53                           else next_state <= hold;
54                         end
55
56              established:  begin
57                      if (reset) next_state <= hold;
58                      else if (connect & SYN_ACK)
59                        begin ACK <= 1;
60                           next_state <= established;
61                        end
62                      else next_state <= hold;
63                    end
64
65              default: next_state <= hold;
66          endcase
67      end
68  endmodule
```

2.3 TCP Server

2.3.1 Code

```
1  `timescale 1ns / 1ps
2
3  module server(
4      reset,
```

```
    clk,
    SYN,
    ACK,
    SYN_ACK
    );

    input reset,
          clk,
          SYN,
          ACK;

    output reg SYN_ACK;

    parameter hold = 3'b1xx;
    parameter syn_recieved  = 3'b010;
    parameter established  = 3'b011;

    reg [2:0] state,
              next_state;

    always @ (posedge clk)
    begin
        if (reset) state <= hold;
        else state <= next_state;
    end

    always @ (state or SYN or ACK or posedge clk)
    begin
        casex (state)

            hold:   begin
                      if (reset) next_state <= hold;
                      else if (SYN & ~ACK) next_state <= syn_recieved;
                      else if (SYN &  ACK) next_state <= established;
                      else next_state <= hold;
                    end

            syn_recieved:    begin
                              if (reset) next_state <= hold;
                              else if (SYN & ~ACK)
                                begin next_state <= hold;
                                      SYN_ACK = 1;
                                end
                              else next_state <= hold;
                             end

            established:    begin
                              if (reset) next_state <= hold;
                              else if (SYN & ACK) next_state <= established;
                              else next_state <= established;
```

```
                                end

            default: next_state <= hold;
        endcase
    end
endmodule
```

2.4 Testbench

2.4.1 Code

```
`timescale 1ns / 1ps

`include "tcp.v"
`include "client.v"
`include "server.v"

module tcp_tb ();
  reg reset,
      clk,
      connect;

  tcp dut(reset,
          clk,
          connect);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
    clk = 0;
    reset = 1;
    connect = 1;

    #10
    reset = 0;

    #100
    reset = 1;

    #10 $stop;
  end

  always #5 clk = ~clk;


endmodule
```

2.5 Simulation Results

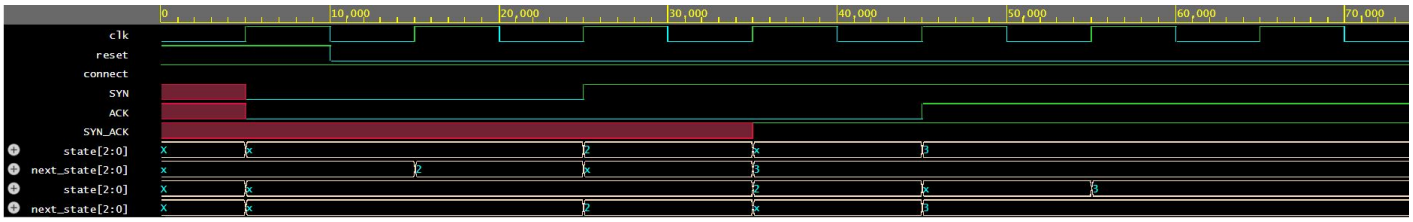

Figure 3: Simulation Result

In these simulation Results the first state is of client and the one below it is of server. Same is the case for next state. X signifies hold state, in both the server and client. 2 signifies syn sent state in client and syn received in server. 3 signifies the established state in both the server and the client. This simulation illustrates the generation of the SYN signal when the client enters the syn sent state. Upon receiving this signal, the server transitions to the syn received state and responds with the SYN ACK signal. Upon receiving the acknowledgment, the client enters the established state and sends an ACK signal. Upon receiving this signal, the server also moves to the established state.