# Adventure Game

Submitted by: Piyusha Jahagirdar

In this lab, we designed a Finite State Machine(FSM) that implemented and Adventure Game. We designed this game using VHDL and simulated or played the game using ModelSim. It took us 3.5 hours to complete this lab.

Figure 1 shows the State Transition Diagram for the Game

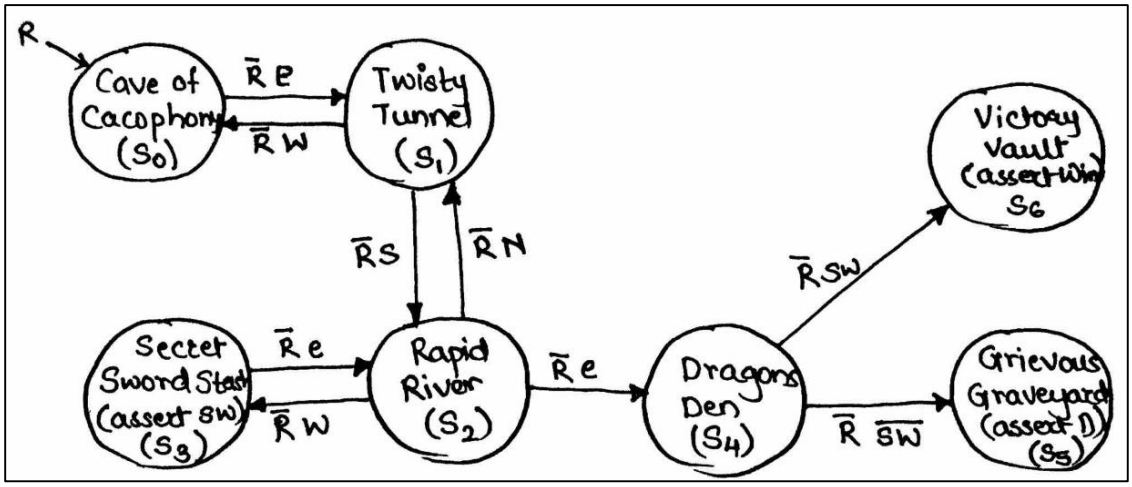


*Figure 1*

As we have designed the code using VHDL we have only one state transition diagram. When Reset is set to 0 the user can play the game. '$S_0$ to $S_6$' are used to denote the states of the FSM, 'SW' is the Sword variable, 'd' is the variable for Death, 'win' is the variable for Win and 'n, e, s, w' are the directions for the player movements in the various Rooms(states)

Table 1 is the state transition table for the FSM

| Current State | reset | clk | Direction | | | | sw | win | d | Next State |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | n | s | e | w | | | | |
| $S_0$ | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $S_0$ |
| $S_0$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | $S_1$ |
| $S_1$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | $S_0$ |
| $S_1$ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | $S_2$ |
| $S_2$ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $S_1$ |
| $S_2$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | $S_3$ |
| $S_2$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | $S_4$ |
| $S_3$ | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | $S_2$ |
| $S_4$ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | $S_5$ |
| $S_4$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | $S_6$ |

*Table 1*

Below is the VHDL code for the Game

```
-- Quartus II VHDL Template
-- Adventure Game

library ieee;
use ieee.std_logic_1164.all;
entity game is                                  -- entity declaration
        port(
                clk : in  std_logic;        --clock input
                n       : in    std_logic;      --input direction North
                s       : in    std_logic;      --input direction South
                e       : in    std_logic;      --input direction East
                w       : in    std_logic;      --input direction West
                reset   : in    std_logic;      --reset input
                win : out       std_logic;      --output variable for winner
                d       : out   std_logic       --output variable for death
        );
end entity;
architecture g of game is
        type state_type is (s0, s1, s2, s3,s4,s5,s6);     --Enumerated type for state
        signal state   : state_type;                      --Signal to hold the current state
        signal direction : std_logic_vector(3 downto 0);--Vector for direction
        signal sw : std_logic;                            --Signal for Sword
begin
        -- Logic to advance to the next state
        direction <= (n,s,e,w);
        process (clk, reset)
```

```
begin
        if reset = '1' then
                state <= s0;
        elsif (rising_edge(clk)) then
                case state is
                        when s0=>
                                if direction = "0010" then
                                        state <= s1;
                                else
                                        state <= s0;
                                end if;
                        when s1=>
                                if direction = "0001" then
                                        state <= s0;
                                elsif direction="0100" then
                                        state <= s2;
                                else
                                        state <= s1;
                                end if;
                        when s2=>
                                if direction = "1000" then
                                        state <= s1;
                                elsif direction="0001" then
                                        state <= s3;
                                elsif direction="0010" then
                                        state <= s4;
                                else
```

```
                    state <= s2;
                end if;
            when s3 =>
                sw <= '1';                  --Setting sword signal to 1
                if direction = "0010" then
                    state <= s2;
                else
                    state <= s3;
                end if;
            when s4 =>
                if sw ='1' then
                    state <= s6;
                    win <='1';              --Setting win output to 1
                else
                    state <= s5;
                    d <='1';        --Setting die output to 1
                end if;
            when others =>
                state <= s0;
        end case;
    end if;
end process;
end g;
```

In the above code we have used the switch case to move along the various states(rooms) in the game. Once it enters a particular state the direction vector decides which room will the player next move to. Once the player reaches State $S_3$ the sw(sword) variable is set to one. If the player reache the state $S_4$ without the sword(sw=0) then the player dies(d=1). But if he has the sword then he fights the dragon and wins the game(win=1).

Figure 2 shows the simulation for winning the game

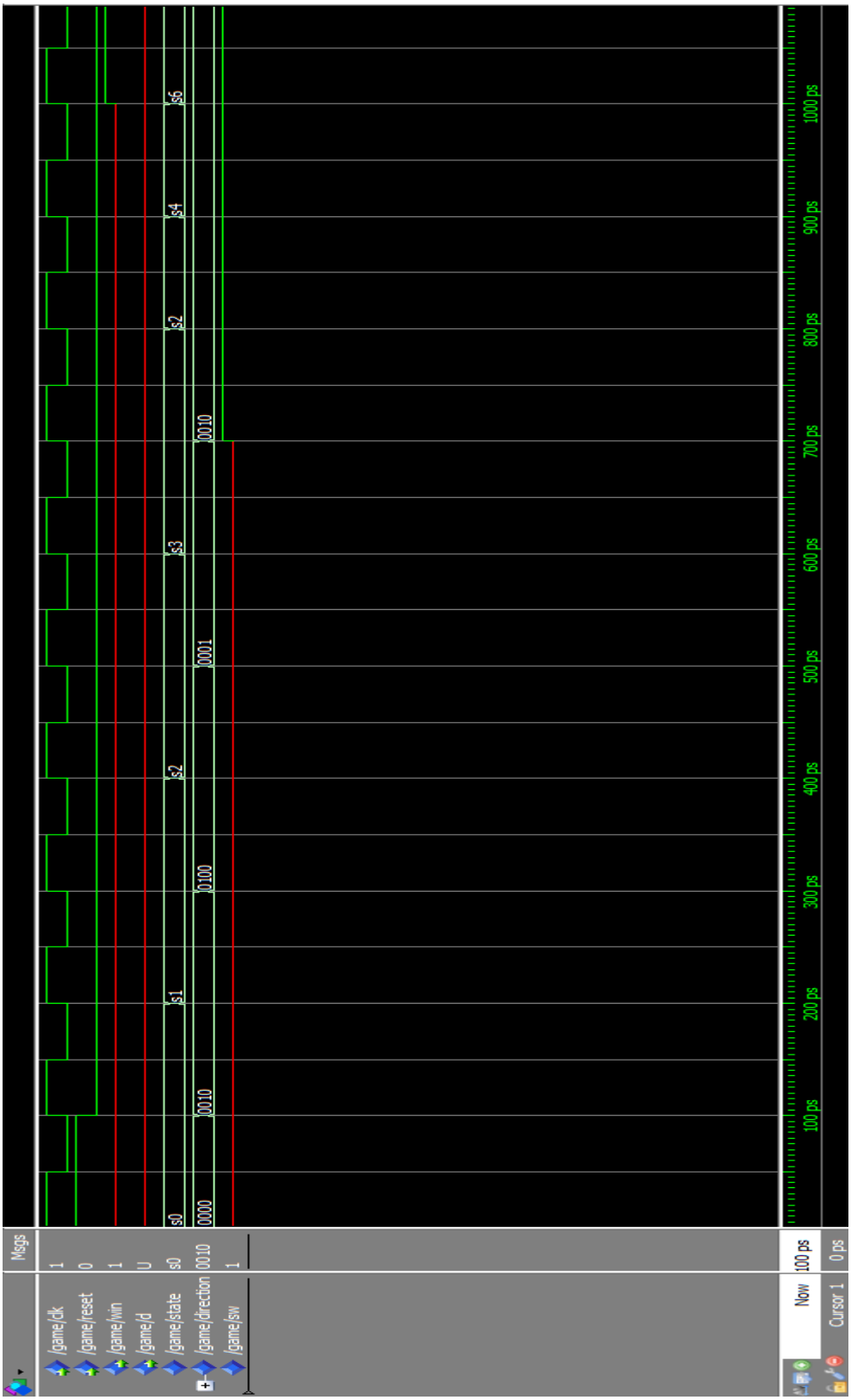


*Figure 2*

Figure 3 shows the simulation for losing the game

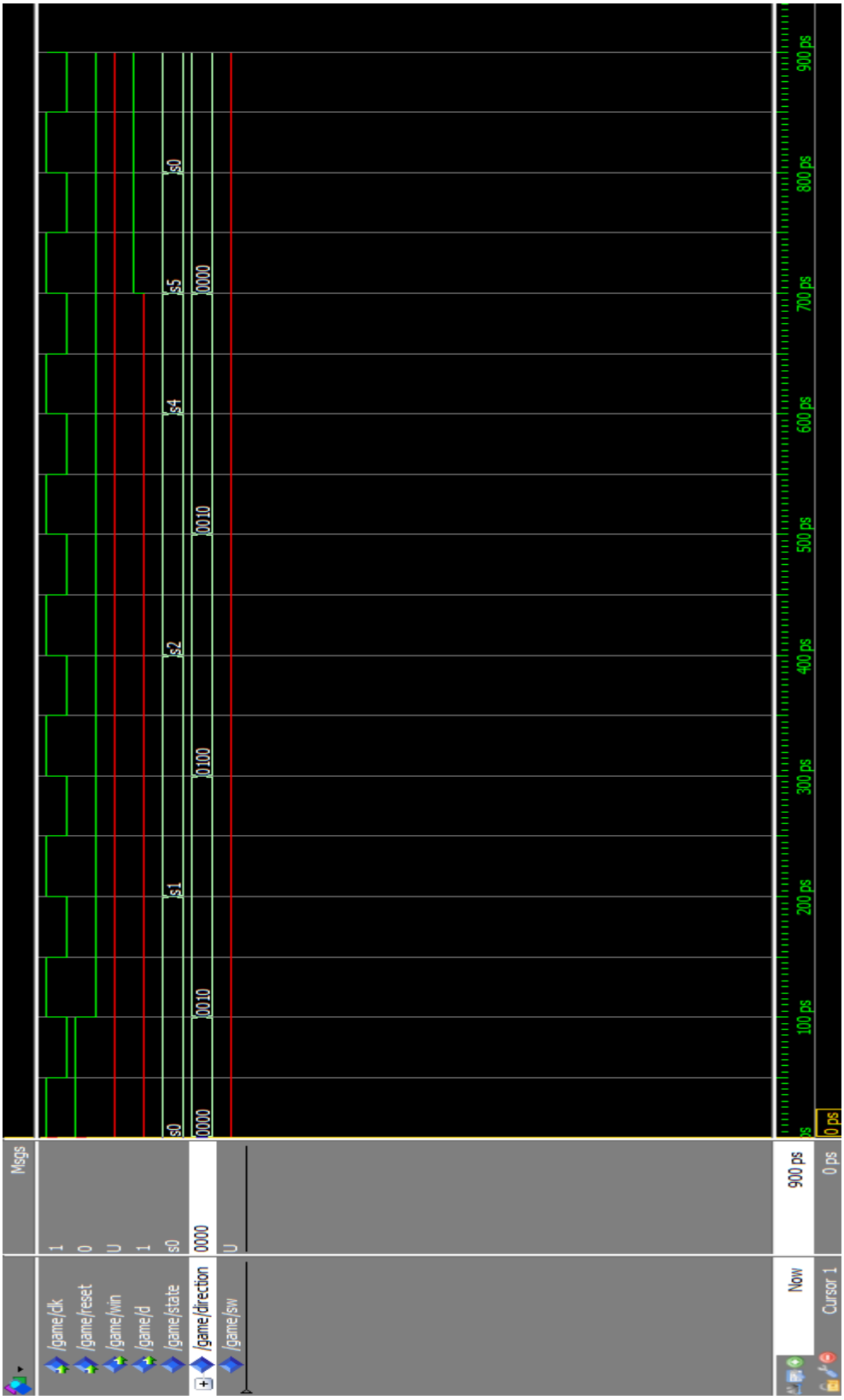


*Figure 3*