[EEL480] Laboratório de sistemas digitais
Relatório 02

Artur Amaral

15 de Outubro de 2021

1. Introdução

O seguinte relatório descreve a implementação de um jogo da forca em linguagem VHDL. Para escrita e teste do código, utilizou-se o software de desenvolvimento Quartus II Lite, da Intel. O projeto final foi aplicado em uma placa de prototipagem FPGA Altera DE2-115, cuja configuração é feita remotamente pelo serviço online Labsland.

2. Desenvolvimento do projeto

O projeto implementa uma máquina de estados, pois a lógica do jogo da forca é fundamentalmente sequencial. A palavra secreta escolhida foi **123321.**

2.1 Especificações de entradas e saídas

A interface com o sistema é feita através das entradas e saídas descritas na tabela abaixo. Em suma, as **9 chaves** representam as possibilidades de **entrada** do usuário, só podendo ser acionadas uma única vez durante a progresso de um jogo (O próprio comportamento de “switch” das entradas já induz a esse funcionamento). Dos **displays de 7 segmentos**, **6** são utilizados para exibir ao jogador os **caracteres que já foram acertados**, enquanto **1** deles é reservado para o fim de uma partida, quando exibe uma “**flag” indicativa do resultado do jogador** - “G” para ganhou, “P” para perdeu. Por fim, três LED’s são utilizados para indicar o número de “vidas” do jogador. A cada erro, um LED se apaga. O apagar de todos os LED’s - i.e. três erros - leva o jogador à derrota.

Entradas		
9 switches	{0,1,2,3,4,5,6,7,8,9}	SW: in std_logic_vector(17 downto 0);
Saídas		
3 leds	indicador de vidas	LEDG: out std_logic_vector(3 downto 0);
6 hex display	exibição da palavras	HEX1 : out std_logic_vector(6 downto 0); HEX2 : out std_logic_vector(6 downto 0); HEX3 : out std_logic_vector(6 downto 0); HEX4 : out std_logic_vector(6 downto 0); HEX5 : out std_logic_vector(6 downto 0); HEX6 : out std_logic_vector(6 downto 0);
1 hex display	"G" ganhou, "P" perdeu	HEX7 : out std_logic_vector(6 downto 0);

2.2 Especificações da máquina de estados

O conjunto de possíveis estados nessa configuração de jogo da forca é exibido na tabela abaixo, que, consequentemente lista o nome de todos os estados codificados no código VHDL.

Na tabela, foram divididos logicamente em colunas para facilitar a compreensão. Existem os estados iniciais, que representam situações de início e fim do jogo. A divisão em colunas do resto dos estados é marcada pelo número de vidas restantes. Para cada uma das possibilidades de vidas restantes, existem diversas possibilidade para a pergunta “Qual dos caracteres já foi acertado?”. E a reposta é descrita no nome do estado pelos números já acertados seguidos pela palavra “hit”.

Exemplificando: O estado 13_hit_two_lives significa que o jogador já acertou os números 1 e 3, mas já teve um erro, logo, possui duas vidas. Consequentemente, se sua próxima entrada for 2, ele irá para o estado win, mas se for qualquer outra, irá para o estado 13_hit_one_life, e assim sucessivamente.

Conjunto dos estados possíveis			
Estados terminais	**Restam 3 vidas**	**Restam 2 vidas**	**Resta 1 vida**
initial_state	1_hit_three_lives	no_hits_two_lives	no_hits_one_life
win	2_hit_three_lives	1_hit_two_lives	1_hit_one_life
lost	3_hit_three_lives	2_hit_two_lives	2_hit_one_life
	12_hit_three_lives	3_hit_two_lives	3_hit_one_life
	13_hit_three_lives	12_hit_two_lives	12_hit_one_life
	23_hit_three_lives	13_hit_two_lives	13_hit_one_life
		23_hit_two_lives	23_hit_one_life

Abaixo, segue a descrição completa da tabela de transição de estados do sistema. É pertinente analisá-la sob a perspectiva de um jogador, para comprovar que, apesar de extensa, é bastante intuitiva.

Estado atual	**Entrada**	**Próximo estado**
initial_state	1	1_hit_three_lives
initial_state	2	2_hit_three_lives
initial_state	3	3_hit_three_lives
initial_state	other	no_hits_two_lives
no_hits_two_lives	1	1_hit_two_lives
no_hits_two_lives	2	2_hit_two_lives

no_hits_two_lives	3	3_hit_two_lives
no_hits_two_lives	other	no_hits_one_life
no_hits_one_life	1	1_hit_one_life
no_hits_one_life	2	2_hit_one_life
no_hits_one_life	3	3_hit_one_life
no_hits_one_life	other	lost
1_hit_three_lives	2	12_hit_three_lives
1_hit_three_lives	3	13_hit_three_lives
1_hit_three_lives	other	1_hit_two_lives
1_hit_two_lives	2	12_hit_two_lives
1_hit_two_lives	3	13_hit_two_lives
1_hit_two_lives	other	1_hit_one_life
1_hit_one_life	2	12_hit_one_life
1_hit_one_life	3	13_hit_one_life
1_hit_one_life	other	lost
2_hit_three_lives	1	12_hit_three_lives
2_hit_three_lives	3	23_hit_three_lives
2_hit_three_lives	other	2_hit_two_lives
2_hit_two_lives	1	12_hit_two_lives
2_hit_two_lives	3	13_hit_two_lives
2_hit_two_lives	other	2_hit_one_life
2_hit_one_life	1	12_hit_one_life
2_hit_one_life	3	13_hit_one_life
2_hit_one_life	other	lost
3_hit_three_lives	1	13_hit_three_lives
3_hit_three_lives	2	23_hit_three_lives
3_hit_three_lives	other	3_hit_two_lives
3_hit_two_lives	1	13_hit_two_lives
3_hit_two_lives	2	23_hit_two_lives
3_hit_two_lives	other	3_hit_one_life
3_hit_one_life	1	13_hit_one_life
3_hit_one_life	2	23_hit_one_life
3_hit_one_life	other	lost
12_hit_three_lives	3	win
12_hit_three_lives	other	12_hit_two_lives
12_hit_two_lives	3	win
12_hit_two_lives	other	12_hit_one_life

12_hit_one_life	3	win
12_hit_one_life	other	lost
13_hit_three_lives	2	win
13_hit_three_lives	other	13_hit_two_lives
13_hit_two_lives	2	win
13_hit_two_lives	other	13_hit_one_life
13_hit_one_life	2	win
13_hit_one_life	other	lost
23_hit_three_lives	1	win
23_hit_three_lives	other	23_hit_two_lives
23_hit_two_lives	1	win
23_hit_two_lives	other	23_hit_one_life
23_hit_one_life	1	win
23_hit_one_life	other	lost

2.3 Código

```
library ieee;
use ieee.std_logic_1164.all;

entity hangman_testbench is

port(
    SW: in std_logic_vector(17 downto 0);
    LEDG: out std_logic_vector(2 downto 0);
    HEX0 : out std_logic_vector(6 downto 0);
    HEX1 : out std_logic_vector(6 downto 0);
    HEX2 : out std_logic_vector(6 downto 0);
    HEX3 : out std_logic_vector(6 downto 0);
    HEX4 : out std_logic_vector(6 downto 0);
    HEX5 : out std_logic_vector(6 downto 0);
    HEX6 : out std_logic_vector(6 downto 0);
    HEX7 : out std_logic_vector(6 downto 0);
    CLOCK_50: in std_logic
);

end hangman_testbench;

--
architecture behavioral of hangman_testbench is

type state_type is
(
initial_state,
```

```
no_hits_two_lives,
no_hits_one_life,
hit_1_three_lives,
hit_1_two_lives,
hit_1_one_life,
hit_2_three_lives,
hit_2_two_lives,
hit_2_one_life,
hit_3_three_lives,
hit_3_two_lives,
hit_3_one_life,
hit_12_three_lives,
hit_12_two_lives,
hit_12_one_life,
hit_13_three_lives,
hit_13_two_lives,
hit_13_one_life,
hit_23_three_lives,
hit_23_two_lives,
hit_23_one_life,
win,
lost
);

signal pr_state, nx_state: state_type;
signal clk, rst: std_logic;
signal kbd: std_logic_vector(9 downto 0);
signal delayedKbd: std_logic_vector(9 downto 0);
signal kbdDiff: std_logic_vector(9 downto 0);
signal disp1: std_logic_vector(6 downto 0);

begin

    clk <= CLOCK_50;
    rst <= SW(17);
    kbd <= SW(9 downto 0);
    kbdDiff <= kbd XOR delayedKbd;

    process(clk, rst)

    begin

        if (rst = '1') then
               pr_state <= initial_state;
        elsif (clk'EVENT and clk = '1') then
               pr_state <= nx_state;
               delayedKbd <= kbd;
        end if;

    end process;
```

```
-- State control
process (kbdDiff, pr_state)
begin

    case pr_state is
          when initial_state =>

                HEX0 <= "0000000";
                HEX1 <= "0111111";
                HEX2 <= "0111111";
                HEX3 <= "0111111";
                HEX4 <= "0111111";
                HEX5 <= "0111111";
                HEX6 <= "0111111";
                HEX7 <= "0000000";
                LEDG <= "111";

                if (kbdDiff(1) = '1') then
                      nx_state <= hit_1_three_lives;
                elsif (kbdDiff(2) = '1') then
                      nx_state <= hit_2_three_lives;
                elsif (kbdDiff(3) = '1') then
                      nx_state <= hit_3_three_lives;
                elsif (kbdDiff = "0000000000") then
                      nx_state <= pr_state;
                else
                      nx_state <= no_hits_two_lives;
                end if;

          when no_hits_two_lives =>

                LEDG <= "011";

                if (kbdDiff(1) = '1') then
                      nx_state <= hit_1_two_lives;
                elsif (kbdDiff(2) = '1') then
                      nx_state <= hit_2_two_lives;
                elsif (kbdDiff(3) = '1') then
                      nx_state <= hit_3_two_lives;
                elsif (kbdDiff = "0000000000") then
                      nx_state <= pr_state;
                else
                      nx_state <= no_hits_one_life;
                end if;

          when no_hits_one_life =>

                LEDG <= "001";

                if (kbdDiff(1) = '1') then
                      nx_state <= hit_1_one_life;
```

```
                elsif (kbdDiff(2) = '1') then
                        nx_state <= hit_2_one_life;
                elsif (kbdDiff(3) = '1') then
                        nx_state <= hit_3_one_life;
                elsif (kbdDiff = "0000000000") then
                        nx_state <= pr_state;
                else
                        nx_state <= lost;
                end if;
        ------------------------
        -- OK!
                when hit_1_three_lives =>

                        HEX1 <= "1111001";
                        HEX6 <= "1111001";
                        LEDG <= "111";

                        if (kbdDiff(2) = '1') then
                                nx_state <= hit_12_three_lives;
                        elsif (kbdDiff(3) = '1') then
                                nx_state <= hit_13_three_lives;
                        elsif (kbdDiff = "0000000000") then
                                nx_state <= pr_state;
                        else
                                nx_state <= hit_1_two_lives;
                        end if;

                when hit_1_two_lives =>

                        HEX1 <= "1111001";
                        HEX6 <= "1111001";
                        LEDG <= "011";

                        if (kbdDiff(2) = '1') then
                                nx_state <= hit_12_two_lives;
                        elsif (kbdDiff(3) = '1') then
                                nx_state <= hit_13_two_lives;
                        elsif (kbdDiff = "0000000000") then
                                nx_state <= pr_state;
                        else
                                nx_state <= hit_1_one_life;
                        end if;


                when hit_1_one_life =>

                        HEX1 <= "1111001";
                        HEX6 <= "1111001";
                        LEDG <= "001";

                        if (kbdDiff(2) = '1') then
```

```
                nx_state <= hit_12_one_life;
        elsif (kbdDiff(3) = '1') then
                nx_state <= hit_13_one_life;
        elsif (kbdDiff = "0000000000") then
                nx_state <= pr_state;
        else
                nx_state <= lost;
        end if;


-----------------------
-- OK!
when hit_2_three_lives =>

        LEDG <= "111";

        HEX2 <= "0100100";
        HEX5 <= "0100100";

        if (kbdDiff(1) = '1') then
                nx_state <= hit_12_three_lives;
        elsif (kbdDiff(3) = '1') then
                nx_state <= hit_23_three_lives;
        elsif (kbdDiff = "0000000000") then
                nx_state <= pr_state;
        else
                nx_state <= hit_1_two_lives;
        end if;


when hit_2_two_lives =>

        LEDG <= "011";

        HEX2 <= "0100100";
        HEX5 <= "0100100";

        if (kbdDiff(1) = '1') then
                nx_state <= hit_12_two_lives;
        elsif (kbdDiff(3) = '1') then
                nx_state <= hit_23_two_lives;
        elsif (kbdDiff = "0000000000") then
                nx_state <= pr_state;
        else
                nx_state <= hit_2_one_life;
        end if;

when hit_2_one_life =>

        LEDG <= "001";
```

```
		HEX2 <= "0100100";
		HEX5 <= "0100100";

		if (kbdDiff(1) = '1') then
			nx_state <= hit_12_one_life;
		elsif (kbdDiff(3) = '1') then
			nx_state <= hit_23_one_life;
		elsif (kbdDiff = "0000000000") then
			nx_state <= pr_state;
		else
			nx_state <= lost;
		end if;

	---

	when hit_3_three_lives =>

		LEDG <= "111";

		HEX3 <= "0110000";
		HEX4 <= "0110000";

		if (kbdDiff(1) = '1') then
			nx_state <= hit_13_three_lives;
		elsif (kbdDiff(2) = '1') then
			nx_state <= hit_23_three_lives;
		elsif (kbdDiff = "0000000000") then
			nx_state <= pr_state;
		else
			nx_state <= hit_3_two_lives;
		end if;

	when hit_3_two_lives =>

		LEDG <= "011";

		HEX3 <= "0110000";
		HEX4 <= "0110000";

		if (kbdDiff(1) = '1') then
			nx_state <= hit_13_two_lives;
		elsif (kbdDiff(2) = '1') then
			nx_state <= hit_23_two_lives;
		elsif (kbdDiff = "0000000000") then
			nx_state <= pr_state;
		else
			nx_state <= hit_3_one_life;
		end if;

	when hit_3_one_life =>
```

```
                LEDG <= "001";

                HEX3 <= "0110000";
                HEX4 <= "0110000";

                if (kbdDiff(1) = '1') then
                        nx_state <= hit_13_one_life;
                elsif (kbdDiff(2) = '1') then
                        nx_state <= hit_23_one_life;
                elsif (kbdDiff = "0000000000") then
                        nx_state <= pr_state;
                else
                        nx_state <= lost;
                end if;

        when hit_12_three_lives =>

                LEDG <= "111";

                HEX1 <= "1111001";
                HEX2 <= "0100100";
                HEX5 <= "0100100";
                HEX6 <= "1111001";

                if (kbdDiff(3) = '1') then
                        nx_state <= win;
                elsif (kbdDiff = "0000000000") then
                        nx_state <= pr_state;
                else
                        nx_state <= hit_12_two_lives;
                end if;


        when hit_12_two_lives =>

                LEDG <= "011";

                HEX1 <= "1111001";
                HEX2 <= "0100100";
                HEX5 <= "0100100";
                HEX6 <= "1111001";

                if (kbdDiff(3) = '1') then
                                nx_state <= win;
                        elsif (kbdDiff = "0000000000") then
                                nx_state <= pr_state;
                        else
                                nx_state <= hit_12_one_life;
                end if;

        when hit_12_one_life =>
```

```
		LEDG <= "001";

		HEX1 <= "1111001";
		HEX2 <= "0100100";
		HEX5 <= "0100100";
		HEX6 <= "1111001";

		if (kbdDiff(3) = '1') then
				nx_state <= win;
			elsif (kbdDiff = "0000000000") then
				nx_state <= pr_state;
			else
				nx_state <= lost;
		end if;

	when hit_13_three_lives =>

		LEDG <= "111";

		HEX1 <= "1111001";
		HEX3 <= "0110000";
		HEX4 <= "0110000";
		HEX6 <= "1111001";

		if (kbdDiff(2) = '1') then
				nx_state <= win;
			elsif (kbdDiff = "0000000000") then
				nx_state <= pr_state;
			else
				nx_state <= hit_13_two_lives;
		end if;

	when hit_13_two_lives =>

		LEDG <= "011";

		HEX1 <= "1111001";
		HEX3 <= "0110000";
		HEX4 <= "0110000";
		HEX6 <= "1111001";

		if (kbdDiff(2) = '1') then
				nx_state <= win;
			elsif (kbdDiff = "0000000000") then
				nx_state <= pr_state;
			else
				nx_state <= hit_13_one_life;
		end if;

	when hit_13_one_life =>
```

```
		LEDG <= "001";

		HEX1 <= "1111001";
		HEX3 <= "0110000";
		HEX4 <= "0110000";
		HEX6 <= "1111001";

		if (kbdDiff(2) = '1') then
				nx_state <= win;
			elsif (kbdDiff = "0000000000") then
				nx_state <= pr_state;
			else
				nx_state <= lost;
		end if;

	when hit_23_three_lives =>

		LEDG <= "111";

		HEX2 <= "0100100";
		HEX3 <= "0110000";
		HEX4 <= "0110000";
		HEX5 <= "0100100";

		if (kbdDiff(1) = '1') then
				nx_state <= win;
			elsif (kbdDiff = "0000000000") then
				nx_state <= pr_state;
			else
				nx_state <= hit_23_two_lives;
		end if;

	when hit_23_two_lives =>

		LEDG <= "011";

		HEX2 <= "0100100";
		HEX3 <= "0110000";
		HEX4 <= "0110000";
		HEX5 <= "0100100";

		if (kbdDiff(1) = '1') then
				nx_state <= win;
			elsif (kbdDiff = "0000000000") then
				nx_state <= pr_state;
			else
				nx_state <= hit_23_one_life;
		end if;

	when hit_23_one_life =>
```

```
                LEDG <= "001";

                HEX2 <= "0100100";
                HEX3 <= "0110000";
                HEX4 <= "0110000";
                HEX5 <= "0100100";

                if (kbdDiff(1) = '1') then
                        nx_state <= win;
                elsif (kbdDiff = "0000000000") then
                        nx_state <= pr_state;
                else
                        nx_state <= lost;
                end if;

        when win =>

                LEDG <= "111";

                HEX0 <= "1111111";
                HEX1 <= "1111001";
                HEX2 <= "0100100";
                HEX3 <= "0110000";
                HEX4 <= "0110000";
                HEX5 <= "0100100";
                HEX6 <= "1111001";

                HEX7 <= "0000010";


        when lost =>

                LEDG <= "000";

                HEX0 <= "0111111";
                HEX1 <= "0111111";
                HEX2 <= "0111111";
                HEX3 <= "0111111";
                HEX4 <= "0111111";
                HEX5 <= "0111111";
                HEX6 <= "0111111";

                HEX7 <= "0000011";

                LEDG <= "000";

    end case;

end process;
```

```
end behavioral;
```

3. Conclusão

Dada a proposta do roteiro, considero que a implementação do projeto foi um sucesso. Funcionou bem no Labsland e considero que sua solução foi simples, o que é um ponto bastante positivo da implementação.