PCS3635 – Laboratório Digital I

Relatório da Experiência 6

Extensões ao Jogo do Tempo de Reação (parte 1)

Turma 3 – Bancada 3

Professores:
Paulo Sergio Cugnasca
Edson Midorikawa

Integrantes:
Arthur Pires da Fonseca – 10773096
Lucas Lopes de Paula Junior - 9344880

02 de março de 2020

INTRODUÇÃO

A partir desta experiência, o jogo do tempo de reação implementado na experiência 5 de Laboratório Digital 1 será modificado de forma a realizar novas funcionalidades.

Neste caso, iremos ampliar a quantidade de sinais que podem ser representados pelo *display* de 7 segmentos da placa DE0-CV.

1. PARTE EXPERIMENTAL

1.1. Atividade 1 – Revisão do código do Jogo do Tempo de Reação

Resgatamos o projeto qar da última experiência e alteramos a forma como a máquina de estados do Jogo do Tempo de Reação era acessada. Foi criada uma entidade nova de nome *"cont_inter_med"*, que é a unidade de controle principal do jogo e que faz a ponte entre os sinais recebidos da Interface Leds e Botões e os sinais enviados para o Medidor de Largura. Dessa forma é possível modularizar o código e entender melhor o seu fluxo de dados.

Paralelamente a isso, multiplexamos os sinais de saídas das principais entidades do projeto para os displays hexadecimais e os leds. Isso nos permite selecionar qual módulo estamos depurando e todos eles irão aparecer nos displays e leds de acordo com a seleção.

A tabela a seguir especifica quais foram as informações selecionadas para serem mostradas em cada *display* relacionados a cada módulo integrado ao projeto até agora.

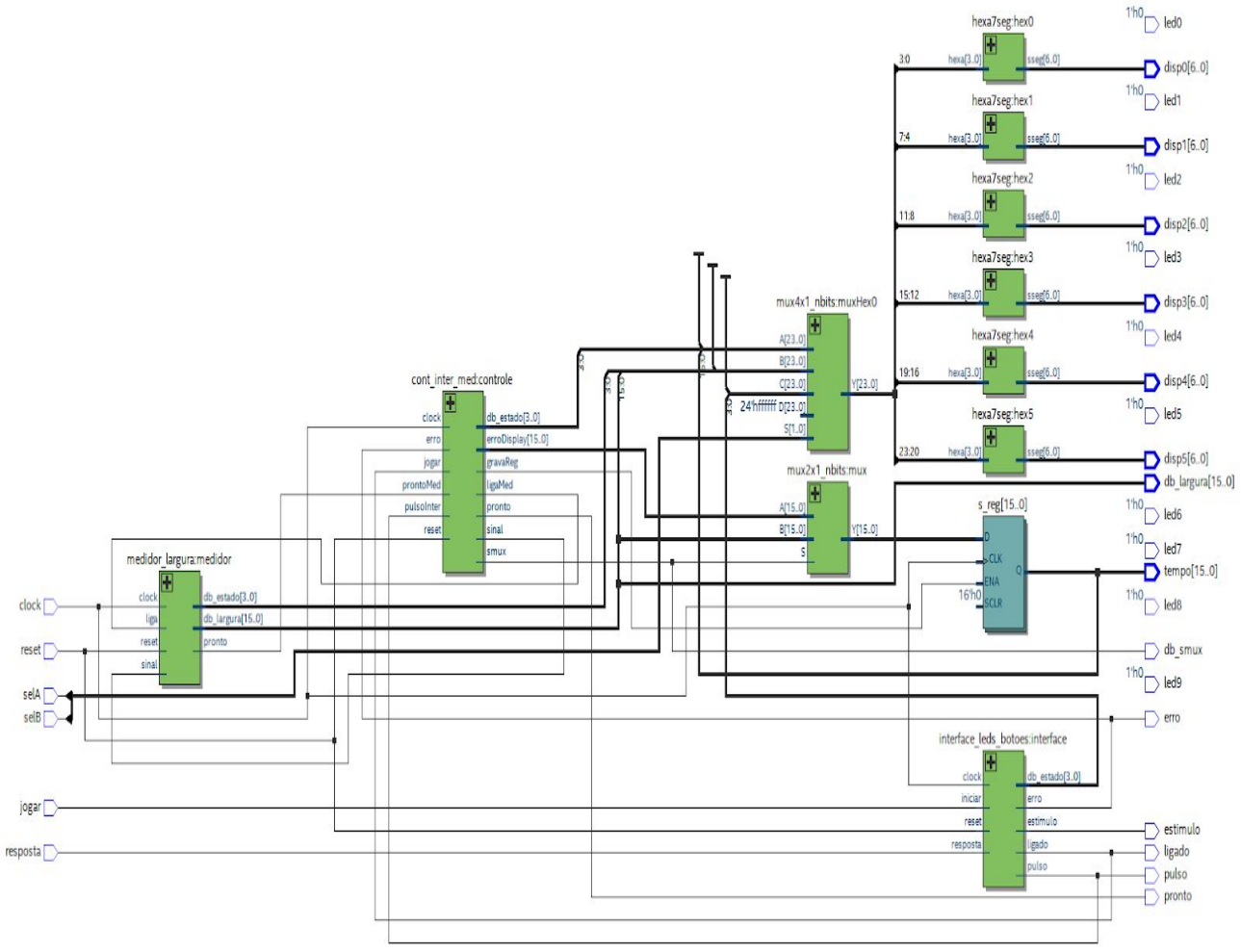
Módulo	**sel_mux**	**HEX5**	**HEX4**	**HEX3**	**HEX2**	**HEX1**	**HEX0**
Jogo do Tempo de Reação	00	estado do jogo		tempo de reação			
Medidor de Largura de Pulso	01	estado do medidor		saída do contador de largura de pulso			
Interface de *Leds* e botões	10	estado da interface					
	11						

Da mesma forma, devido à quantidade limitada de LEDs na placa FPGA, selecionamos alguns sinais de depuração provenientes dos módulos internos do circuito lógico para serem amostrados convenientemente, sendo selecionados por um multiplexador 4x1, o qual destina os sinais adequadamente segundo o valor do sinal "sel_mux".

Módulo	sel_mux	LEDR9	LEDR8	LEDR7	LEDR6	LEDR5	LEDR4	LEDR3	LEDR2	LEDR1	LEDR0
Jogo do Tempo de Reação	00	pronto									ligado
Medidor de Largura de Pulso	01	pronto					fim	db_contaCont	db_zeraCont	db_clock	
Interface de *Leds* e botões	10	pronto	erro	pulso	estímulo	ligado					
	11										

Os testes de depuração do circuito foram feitos de forma semelhante aos feitos no Quartus para averiguar o funcionamento do nosso projeto**.** Para cada módulo do circuito, deve-se conferir se os *displays* e LEDs estão sendo apresentados como esperado.

Abaixo o diagrama RTL do circuito modificado em laboratório.



Plano de testes

Nome do teste	Sequência de passos a serem realizados e resultados esperados
Jogada válida	Acionar "jogar", esperar o estímulo ligar e acionar "resposta". Os *displays* 0 a 3 deverão amostrar o tempo de reação quando o modo selecionado for o do Jogo do Tempo de Reação.
Jogada inválida	Acionar "jogar" , em seguida acionar "resposta" o mais rápido possível. Os displays 0 a 3 deverão amostrar "EAA0" no modo Jogo do Tempo de Reação, mas fornecerão o tempo de reação no modo Medidor de Largura de Pulso
Depuração	Selecionar com as chaves CH8 e CH9 os sinais para sel_mux e conferir se estão de acordo com as máquinas de estado de cada módulo.

Pinagem

Designação			**Analog Discovery**		
sinal		**pino**	**instrumento**	**função**	**sinal**
entradas	clock	GPIO_0_D0	Patterns	clock	DIO0
	reset	GPIO_0_D1	Static I/O	button 0/1	DIO1
	jogar	GPIO_0_D3	Static I/O	button 0/1	DIO2
	resposta	GPIO_0_D5	Static I/O	button 0/1	DIO7
saídas	pulso	GPIO_0_31	Scope	-	C1
	estímulo	GPIO_0_33	Static I/O	*LED*	DIO15
	erro	GPIO_0_35	Static I/O	*LED*	DIO14
	sinais de depuração	LEDR0 a LEDR9	-	-	-
	estados e informações adicionais	HEX00 a HEX56	-	-	-
	selA	SW9	-	-	-
	selB	SW8	-	-	-

1.2. Atividade 2 – Implementação do Jogo do Tempo de Reação

Utilizando o Intel Quartus Prime, sintetizamos o nosso projeto na placa FPGA e conectamos os seus terminais GPIO aos do Analog Discovery. Em seguida, executamos o plano de testes e não observamos qualquer divergência em relação ao observado na simulação realizada antes do laboratório.

Respostas às perguntas:

1. Como a estratégia de multiplexação de displays e leds pode ser usada nos procedimentos de teste e depuração de um circuito digital?

A multiplexação permite que diferentes funcionalidades de um circuito sejam testadas, podendo-se trocar sistematicamente (com um botão, por exemplo) quais sinais devem ser amostrados nos *displays* e *leds* dependendo de em que situação (modo) se quer verificá-lo.

2. Houve alguma situação em que foi necessário modificar o conjunto de sinais apresentados no projeto da experiência?

Sim, modificamos nosso VHDL de forma a deixar a apresentação de cada módulo mais enxuta. Os *displays* em que não se planejou apresentar qualquer dado novo receberam entrada cheia de ‘1’s, para que o *display* se mantivesse desligado, sem apresentar informações.

1.3 Atividade 3 - Desafio

O circuito foi modificado de forma a acrescentar um marco de pontuação para o jogo, a qual é definida como a soma dos tempos de reação medidos durante uma partida. Vence a sequência de jogos aquele que tiver a menor somatória desses pontos.

Querendo-se implementar essa ideia, as tabelas que relacionam os módulos do circuito e os *displays* e LEDs do FPGA tiveram que ser modificadas.

Módulo	sel_mux	HEX5	HEX4	HEX3	HEX2	HEX1	HEX0
Jogo do Tempo de Reação	00	estado do jogo		tempo de reação			
Medidor de Largura de Pulso	01	estado do medidor		saída do contador de largura de pulso			
Interface de *Leds* e botões	10	estado da interface					

Pontuação do jogo	11		vai-um	pontuação do jogo

Módulo	sel_mux	LEDR9	LEDR8	LEDR7	LEDR6	LEDR5	LEDR4	LEDR3	LEDR2	LEDR1	LEDR0
Jogo do Tempo de Reação	00	pronto									ligado
Medidor de Largura de Pulso	01	pronto					fim	db_contaCont	db_zeraCont	db_clock	
Interface de *Leds* e botões	10	pronto	erro	pulso	estímulo	ligado					
Pontuação do jogo	11	db_saidasomadordecimal(9 downto 0)									

Resultados alcançados

Pontos positivos: a multiplexação permite que sejam depurados todos os módulos que quisermos reaproveitando os componentes leds e displays de 7 segmentos da placa. Isso será uma vantagem nas próximas experiências.

Pontos negativos: é necessário entender muito bem em qual módulo está a depuração e o que significam seus sinais. A organização dos papéis no laboratório de forma a compreender o que se observa na placa FPGA é de execução difícil e exige muita atenção.

Lições aprendidas: quanto mais modularizado é o projeto mais fácil e intuitivo fica a conexão de sinais para depuração. (Dividir para conquistar!)

Apêndice

A partir da próxima página deixamos o código da nossa entidade principal “exp6_T3BB3.vhd”. Nele podem ser vistos todas as principais modificações feitas durante o laboratório.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.math_real.all;

entity exp6_T3BB3 is
    port (
        jogar, resposta, reset, clock             :  in    std_logic;
        selA                                      :  in    std_logic;
        selB                                      :  in    std_logic;
        ligado, pulso, estimulo, erro, pronto     :  out   std_logic;
        disp0, disp1, disp2, disp3, disp4, disp5  :  out   std_logic_vector(6 downto 0);
        db_largura                                :  out   std_logic_vector(15 downto 0);
        db_smux                                   :  out   std_logic;
        led0, led1, led2, led3, led4              :  out   std_logic;
        led5, led6, led7, led8, led9              :  out   std_logic;
        tempo                                     :  out   std_logic_vector(15 downto 0)
    );
end exp6_T3BB3;

architecture arch of exp6_T3BB3 is
    -- componente interface leds e botoes
    component interface_leds_botoes is
        port (
            clock, reset: in std_logic;
            iniciar, resposta: in std_logic;
            ligado, estimulo, pulso: out std_logic;
            db_estado : out std_logic_vector(3 downto 0);
            erro, pronto: out std_logic
        );
    end component;

    -- componente controlador entre a interface e medidor
component cont_inter_med is
    port (
        jogar              :     in    std_logic;
        pulsoInter         :     in    std_logic; --recebe pronto da interface
        prontoMed          :     in    std_logic; --recebe erro da interface
        reset, clock       :     in    std_logic; --clock e reset
        erro               :     in    std_logic;
        smux               :     out   std_logic;
        sinal              :     out   std_logic; --conecta no liga do medidor
        ligaMed            :     out   std_logic;
        gravaReg           :     out   std_logic;
        pronto             :     out   std_logic;
        db_estado          :     out   std_logic_vector(3 downto 0);
        erroDisplay        :     out   std_logic_vector(15 downto 0)
    );
end component;

    -- componente medidor de largura de pulso
    component medidor_largura is --entidade medidor_largura
        port ( clock, reset  :  in    std_logic; --entradas clock e reset
            liga, sinal    :  in    std_logic; --entradas liga e sinal
            fim            :  out   std_logic; --saida fim
            pronto         :  out   std_logic; --saida pronto
            db_largura     :  out   std_logic_vector(15 downto 0); --saida db_largura de 16 bits
            db_estado      :  out   std_logic_vector(3 downto 0);
            db_clock       :  out   std_logic; --saida de clock
            db_zeraCont    :  out   std_logic; --saida db_zeraCont
            db_contaCont   :  out   std_logic --saida db_contaCont
           );
    end component;

    -- compomente mux 2x1 de N bits
    component mux2x1_nbits is
        generic (
            N: integer := 16
        );
        port (
            A, B: in  std_logic_vector (N-1 downto 0);
            S:    in  std_logic;
            Y:    out std_logic_vector (N-1 downto 0)
        );
    end component;

    -- componente mux 4x1 de N bits
    component mux4x1_nbits is
     generic (
            N: integer := 24
     );
     port (
            A,B,C,D: in  std_logic_vector (N-1 downto 0);
            S:       in  std_logic_vector (1 downto 0);
            Y:       out std_logic_vector (N-1 downto 0)
     );
    end component;

    component hexa7seg is
     port (
            hexa : in std_logic_vector(3 downto 0);
            sseg : out std_logic_vector(6 downto 0)
     );
    end component;

    signal s_erro, s_prontoI, s_prontoM, s_prontoJR, s_pulso, s_liga, s_ligado, s_estimulo, s_sinal : std_logic;
    signal s_smux, s_gravaReg, s_db_zeraCont, s_db_clock, s_db_contaCont, s_fim : std_logic;
    signal sel_mux :  std_logic_vector(1 downto 0);
    signal s_erroDisplay, s_db_largura, s_tempo, s_reg : std_logic_vector(15 downto 0);
    signal s_mux0, s_mux1, s_mux2, s_mux3, s_mux5 : std_logic_vector(3 downto 0);
    signal estadoMedidor, estadoInterface, estadoJogo : std_logic_vector(3 downto 0);
    --signal s_led0, s_led1, s_led2, s_led3, s_led4, s_led5, s_led6, s_led7, s_led8, s_led9 : std_logic;
    signal s_leds : std_logic_vector(9 downto 0);
    signal leds_JR, leds_ML, leds_ITLB, leds_pontuacao : std_logic_vector(9 downto 0);
```

```
105     signal jogoReacao, medidorLargura, interfaceLB, pontuacao : std_logic_vector(23 downto 0);
106     signal vaiUm : std_logic_vector(3 downto 0);
107     signal somaTempo : std_logic_vector(15 downto 0);
108     signal sinalzaoDisplays : std_logic_vector(23 downto 0);
109
110   begin
111
112     --instanciacao do componente interface e ligacao dos fios/sinais
113     interface : interface_leds_botoes port map (
114        clock => clock,
115        reset => reset,
116        iniciar => jogar,
117        resposta => resposta,
118        ligado => s_ligado,
119        estimulo => s_estimulo,
120        db_estado => estadoInterface,
121        erro => s_erro,
122        pronto => s_prontoI,
123        pulso => s_pulso
124     );
125
126     --instanciacao do componente controle entre interface e medidor
127     controle : cont_inter_med port map (
128        jogar => s_ligado,
129        pulsoInter => s_pulso,
130        prontoMed => s_prontoM,
131        reset => reset,
132        clock => clock,
133        erro => s_erro,
134        smux => s_smux,
135        sinal => s_sinal,
136        ligaMed => s_liga,
137        gravaReg => s_gravaReg,
138        pronto => s_prontoJR,
139        db_estado => estadoJogo,
140        erroDisplay => s_erroDisplay
141     );
142
143     pronto <= s_prontoJR;
144
145     --instanciacao do componente medidor de largura de pulso
146     medidor : medidor_largura port map (
147        liga => s_liga,
148        sinal => s_sinal,
149        clock => clock,
150        reset => reset,
151        fim => s_fim,
152        pronto => s_prontoM,
153        db_largura => s_db_largura,
154        db_estado => estadoMedidor,
155        db_clock => s_db_clock,
156        db_zeraCont => s_db_zeraCont,
157        db_contaCont => s_db_contaCont
158     );
```

```
159
160     mux : mux2x1_nbits port map(
161        A => s_erroDisplay,
162        B => s_db_largura,
163        Y => s_tempo,
164        S => s_smux
165     );
166
167     reg : process(clock, s_gravaReg)
168     begin
169        if clock'event and clock = '1' then
170           if s_gravaReg = '1' then
171              s_reg <= s_tempo;
172           end if;
173        end if;
174     end process;
175
176     jogoReacao <= estadoJogo & "1111" & s_reg;
177     medidorLargura <= estadoMedidor & "1111" & s_db_largura;
178     interfaceLB <= estadoInterface & X"FFFFF";
179     pontuacao <= "1111" & vaiUm & somaTempo;
180
181     vaiUm <= (others => '1');
182     somaTempo <= (others => '1');
183
184     muxHex0 : mux4x1_nbits port map(
185        A => jogoReacao,
186        B => medidorLargura,
187        C => interfaceLB,
188        D => pontuacao,
189        S(0) => selB,
190        S(1) => selA,
191        Y => sinalzaoDisplays
192     );
193
194     hex0 : hexa7seg port map(
195        hexa => sinalzaoDisplays(3 downto 0),
196        sseg => disp0
197     );
198
199     hex1 : hexa7seg port map(
200        hexa => sinalzaoDisplays(7 downto 4),
201        sseg => disp1
202     );
203
204     hex2 : hexa7seg port map(
205        hexa => sinalzaoDisplays(11 downto 8),
206        sseg => disp2
207     );
208
209     hex3 : hexa7seg port map(
210        hexa => sinalzaoDisplays(15 downto 12),
211        sseg => disp3
212     );
```

```
214     hex4 : hexa7seg port map(
215        hexa => sinalzaoDisplays(19 downto 16),
216        sseg => disp4
217     );
218
219     hex5 : hexa7seg port map(
220        hexa => sinalzaoDisplays(23 downto 20),
221        sseg => disp5
222     );
223
224
225     sel_mux(0) <= selB;
226     sel_mux(1) <= selA;
227
228     leds_JR <= S_prontoJR & X"00" & s_ligado;
229     leds_ML <= s_prontoM & "0000" & s_fim & s_db_contaCont & s_db_zeraCont & s_db_clock & '0';
230     leds_ITLB <= s_prontoI & s_erro & s_pulso & s_estimulo & s_ligado & B"0_0000";
231
232     with sel_mux select s_leds <=
233        leds_JR when "00",
234        leds_ML when "01",
235        leds_ITLB when "10",
236        leds_pontuacao when "11";
237
238
239     tempo <= s_reg;
240     ligado <= s_ligado;
241     pulso <= s_pulso;
242     erro <= s_erro;
243     estimulo <= s_estimulo;
244     db_largura <= s_db_largura;
245     db_smux <= s_smux;
246  end architecture;
```