Implementação do Jogo Resta Dois

Gustavo Silva Reis
Faculdade de Computação - FACOM
Universidade Federal do Mato Grosso do Sul
Campo Grande, Brasil
gustavoreis2040@hotmail.com

Gabriel Medina Braga
Faculdade de Computação - FACOM
Universidade Federal do Mato Grosso do Sul
Campo Grande, Brasil
medina_cdz@hotmail.com

***Index Terms*—VHDL; FPGA; Resta Um, Resta Dois;**

I. Introdução

Inicialmente, a ideia do projeto era implementar o jogo Resta Um como uma máquina de estados. Porém, nos estágios iniciais de desenvolvimento notou-se que seria necessário ter conhecimento prévio sobre todos os estados possíveis do jogo; dado o tempo limite para o desenvolvimento do projeto, considerou-se irrazoável esta quantidade de estados. A partir destas considerações foi tomada a decisão de diminuir o tabuleiro, consequentemente mudando o jogo, este que foi apelidado de Resta Dois. As particularidades deste jogo são explicadas em detalhes na Seção II.

Os experimentos realizados serão explicitados na Seção III e por fim, será concluído se os resultados desejados foram alcançados.

II. Projeto Resta Dois

A primeira decisão de projeto foi tomar o tabuleiro no formato de uma cruz 5x5 como ilustrado na Figura 1. Desta maneira, a quantidade de estados possíveis tornou-se razoavelmente aceitável.

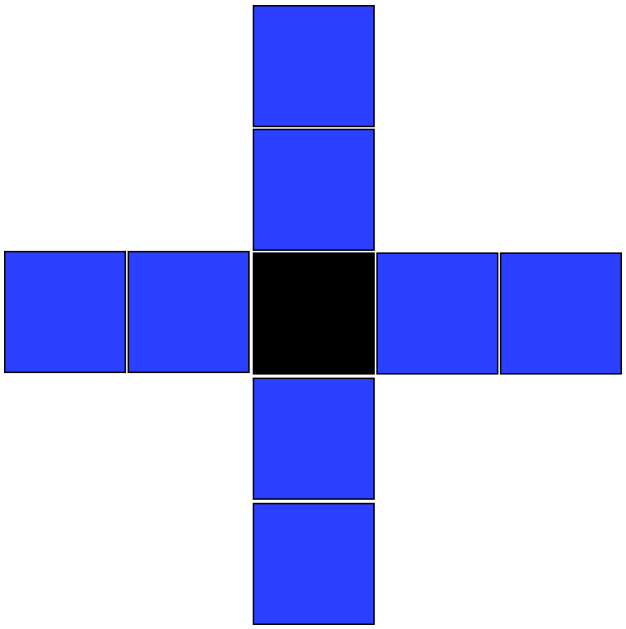

Fig. 1. Tabuleiro do Resta Dois

A. *Modelo*

Para identificar estes estados, foram abertas todas possibilidades manualmente, tomando cuidado de verificar quais estados eram equivalentes aos outros. Cada estado recebeu um nome de acordo com seu posicionamento dentro de seu ”nível”, por exemplo: o nível de um estado alcançado após executada a primeira jogada é igual a 1 e seu posicionamento pode ser ”A”, portanto o nome deste estado é N1A.

Para jogar, deve-se primeiro setar a posição inicial da jogada, esta que é composta por coordenadas ”xi” e ”yi”, e que podem ser visualizadas no display de 7 segmentos; para setar um valor é preciso colocar seu equivalente em binário através do switches, SW5 ao SW3 para xi e SW2 ao SW0 para yi, e confirmar este valor setando valor lógico alto para o switch SW9. Após confirmar a posição inicial da jogada, analogamente, é setada a posição final da jogada, de coordenadas ”xf” e ”yf”, a única diferença é que o SW8 é utilizado para confirmar seus valores. Ao executar estes passos, verifica-se que o tabuleiro muda de acordo, pois acabou de trocar de estado. Quando o jogo chega a um estado final, o display de 7 segmentos deixa de mostrar as coordenadas da última jogada e mostra uma mensagem que indica se o jogador perdeu ou ganhou. Ademais, é possível reiniciar o jogo a qualquer momento ao pressionar o push button ”BUTTON 0”.

B. *Desenvolvimento*

O projeto foi dividido em dois módulos, sendo estes:

1) VGA_RASTER: Este arquivo VHDL é o módulo principal do jogo onde basicamente tudo acontece.

- Entidade VGA_RASTER

```
entity vga_raster is
port(
     reset,clk: in bit;
     SW_I, SW_F: in bit;
     x,y: in bit_vector(2 downto 0);
     d0, d1, d2, d3: out bit_vector(0 to 6);
     VGA_HS,VGA_VS: OUT STD_LOGIC;
     VGA_R:out std_logic_vector(3 downto 0);
     VGA_G:out std_logic_vector(3 downto 0);
     VGA_B:out std_logic_vector(3 downto 0));
end vga_raster;
```

- Principais Estados

```
type States is (I,N1A,N1B,N1C,N1D,
     N2A,N2B,N2C,N2D,
     N3A,N3B,N3C,N3D,N3E,N3F,N3G,N3H,
     N4A,N4B,N4C,N4D,N4E,N4F,N4G,N4H,
     N5A,N5B,N5C,N5D,
     N6A,N6B,N6C,N6D);
```

- Processo P1: Contém um switch case cobrindo todos os estados possíveis.

```
p1:process(est_atual, SW_I,SW_F, xi,
                yi, xf, yf) is begin
    case est_atual is
    ...
    end case;
end process;
```

- Processo P2: Este processo contém quatro procedimentos essencias com a lógica necessária para que a troca entre estados seja efetivada.
  - Display_xi_yi controla dois displays de sete segmentos relacionados à posição x e y iniciais:

```
procedure display_xi_yi is begin
...
end procedure display_xi_yi;
```

  - Display_xf_yf controla outros dois displays de sete segmentos relacionados à posição x e y finais:

```
procedure display_xf_yf is begin
...
end procedure display_xf_yf;
```

  - DisplayDefeat controla os quatros displays de sete segmentos monstrando a palavra LOSE quando um estado de derrota é alcançado:

```
procedure displayDefeat is begin
...
end procedure displayDefeat;
```

  - DisplayVictory controla os quatros displays de sete segmentos monstrando a palavra WIN quando um estado de vitória é alcançado:

```
procedure displayVictory is begin
...
end procedure displayVictory;
```

2) VGA_RASTER_RESTA_DOIS: Este módulo contém a entidade de alto nível, responsável pela impressão das telas, de forma que o clock de impressão dos pixels na tela é realizado com uma frequência de 25,175 MHz.

- Entidade vga_raster_resta_dois

```
entity vga_raster_resta_dois is
port(clk: in bit;
 reset : in bit;
 SW_I_S, SW_F_S: in bit;
 X_S, Y_S: in bit_vector(2 downto 0);
 D0_S, D1_S: out bit_vector(0 to 6);
 D2_S, D3_S: out bit_vector(0 to 6);
 VGA_HS: OUT STD_LOGIC;
 VGA_VS: OUT STD_LOGIC;
 VGA_R:out std_logic_vector(3 downto 0);
 VGA_G:out std_logic_vector(3 downto 0);
 VGA_B:out std_logic_vector(3 downto 0));
end entity;
```

III. Experimentos e Resultados

Por mais que o tamanho do tabuleiro tenha sido reduzido em relação ao projeto inicial, o mesmo ainda apresentou uma quantidade apreciável de estados, o quê, consequentemente facilita o aparecimento de erros no código devido à necessidade de se replicar blocos de código. Tendo em vista esta situação, os experimentos consistiram em executar, na força bruta, toda linha de jogadas possível, fazendo com que todos os estados fossem visitados. Este procedimento foi necessário para verificar se não foram cometidos erros tanto nas trocas de estado, quanto na representação visual daquele estado do tabuleiro.

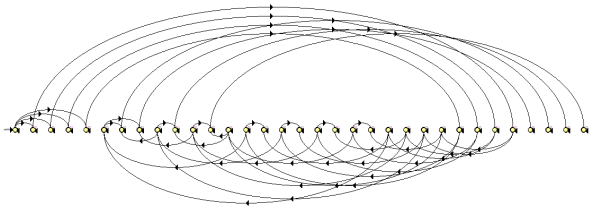

Fig. 2. Máquina de Estados gerada pelo Quartus

Curiosamente, após a compilação, a ferramenta Quartus II 13.0 gerou exatamente o diagrama dos estados com suas devidas transições.

1) Compilação do projeto: A compilação completa do projeto verificada na Figura 3, mostrou que o projeto ocupou 386 elementos lógicos de um total de 15.408 disponíveis, ou seja, apenas 2% do total. E, a quantidade de pinos utilizados foi de 52, cerca de 15% dos disponíveis.

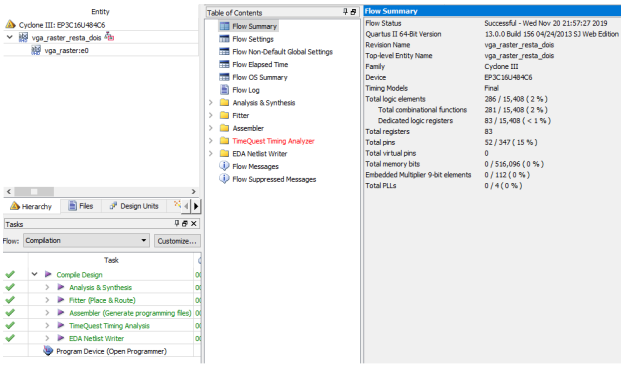

Fig. 3. Ocupação projeto na placa.

Devido o fato do jogo ter que imprimir na tela os estados do jogo validasse, conforme Figura 4, que dos 281 elementos lógicos combinacionais 276 está ligado ao vga_raster. O mesmo vale para o uso de registradores no geral, dos 83 utilizados 79 foram destinados para o vga_raster.

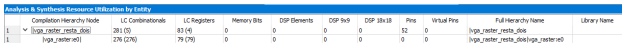

Fig. 4. Recursos alocados por entidade.

Analisando a Figura 5, é possível ver a dimensão do vga_raster confirmando o porquê dele ter ocupado a maior parte dos elementos lógicos do projeto. Apesar de uma leitura minunciosa do código na parte referente ao processo P2, não obtivemos exito na retirada de todos os latchs que foram inferidos pelo Quartus II.

Fig. 5. Vga Raster.

2) Estados Finais: Exitem duas configurações quando o jogo chega ao fim.

- Vitória caracterizado pela mensagem ”WIN” no displays de sete segmentos, conforme Figura 6, e por uma imagem na tela que configure apenas dois quadrados azuis, conforme Figura 7.

Fig. 6. Win no Display.

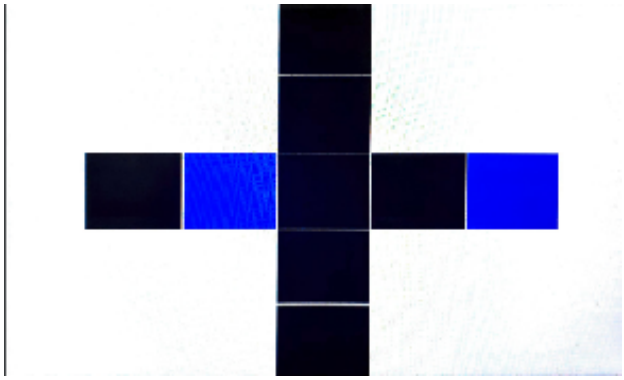

Fig. 7. Estado de Vitória.

- Derrota caracterizado pela mensagem ”LOSE” no display de sete segmentos, conforme Figura 8, e por uma imagem na tela que configure apenas quatro quadrados azuis, conforme Figura 9.

Fig. 8. Lose no Display.

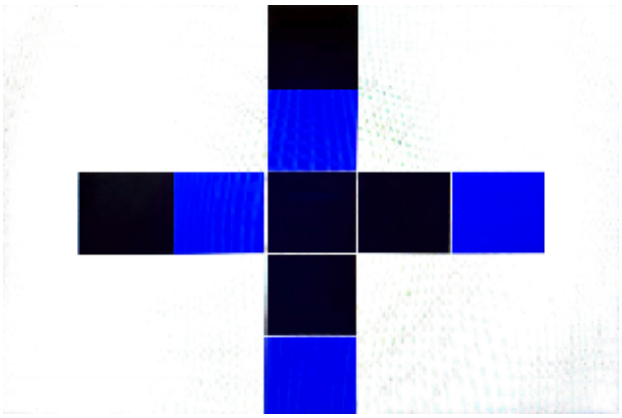

Fig. 9. Estado de Derrota.

A. *Simulação*

A simulação do projeto foi feita com o auxílio do software ModelSim-Altera 10.1d. O intuito da simulação foi de verificar se a troca de estados acontecia corretamente e se o display de sete segmentos estava sendo configurado apropriadamente. Não foram feitas muitas simulações pois rapidamente verificou-se que toda a máquina de estados e configurações dos displays de 7 segmentos estavam corretas, por conseguinte, os experimentos passaram a ser realizados apenas na placa FPGA DE0, pois desta maneira, era possível verificar se o interfaceamento VGA, que é um dos pontos mais críticos deste projeto, estava funcionando.

O script de simulação que se enconstra no diretório do código fonte, simula apenas a troca do estado inicial I para o estado N1C.

IV. Considerações finais

A priori, o objetivo do projeto foi implementar o jogo Resta Um, mas devido aos motivos citados na Seção II, foi desenvolvida uma versão jogável de um novo jogo batizado de Resta Dois. A escolha de se projetar um jogo de tabuleiro deveu-se ao fato de que era sabido que tal trabalho colocaria em prática conceitos importantes vistos em aula, tais como:

modelagem de um problema em máquina de estados, interfaceamento VGA, testbench, entre outros.

Após realizados os experimentos detalhados na Seção III, conclui-se que o projeto foi um sucesso, sendo essa nova versão totalmente desenvolvida e jogável.

V. Referências Bibliográficas

- https://ava.ufms.br/pluginfile.php/85056/mod_resource/content/1/DE0_User_manual_2012.pdf