UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DOCENTE: GEOMAR ANDRE SCHREINER
DISCENTES: ARTHUR EMANUEL DA SILVA (2211100029)
E JÉSSICA BRITO DA SILVA (20240002517)

IMPLEMENTAÇÃO DE BLACKJACK

SISTEMAS DIGITAIS

CHAPECÓ
Maio de 2025

SUMÁRIO

1 INTRODUÇÃO 2
1.1 RESUMO 2
1.2 DESCRIÇÃO DA APLICAÇÃO 2
2 ESTRATÉGIA DE RESOLUÇÃO 3
2.1 FSM - GERADOR DE CARTAS 3
2.2 DIAGRAMA DE TRANSIÇÃO DO GERADOR DE CARTAS 3
2.3 FSM - EXECUÇÃO DO JOGO BLACKJACK 4
2.4 DIAGRAMA DE TRANSIÇÃO DAS RODADAS DO JOGO 5
2.5 INTERAÇÃO ENTRE AS MÁQUINAS DE ESTADO DO PROBLEMA . . 6
3 RESOLUÇÃO 8
3.1 FASE DE PLANEJAMENTO 8
3.2 ENTIDADE CARDS 9
3.3 ENTIDADE BLACKJACK 9
3.4 ENTIDADE TOP-LEVEL E MAPEAMENTO FPGA 10
4 NOTAS E CONSIDERAÇÕES FINAIS 12

1 INTRODUÇÃO

1.1 RESUMO

Este relatório descreve o projeto e a implementação de uma Máquina de Estados Finitos (FSM) em VHDL para um jogo de Blackjack simplificado, executado em uma placa FPGA DE1 - Cyclone II. O sistema gerencia as regras do jogo para um jogador e um carteador, incluindo distribuição de cartas, decisões de "HIT"e "STAY", e determinação do vencedor. Uma FSM secundária foi desenvolvida para a geração de cartas em modos aleatório ou manual. A interação do jogador é feita por botões físicos (HIT, STAY, START), com o botão START funcionando também como reset. As pontuações e cartas são exibidas em displays de 7 segmentos. O projeto foi desenvolvido em VHDL e implementado fisicamente em uma FPGA, validando seu funcionamento conforme o esperado, demonstrando a aplicação de design digital em hardware reconfigurável.

1.2 DESCRIÇÃO DA APLICAÇÃO

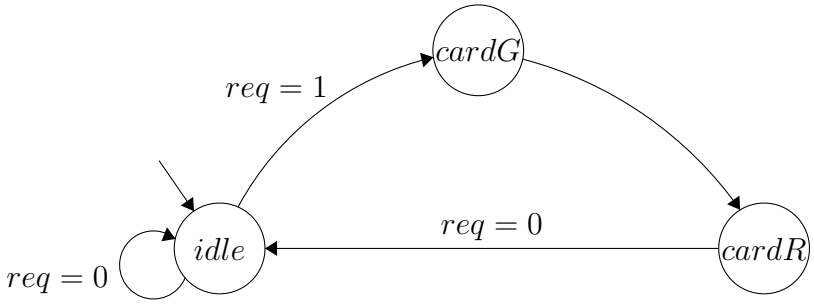
O jogo de Blackjack envolve um jogador e um carteador, com o objetivo de obter uma mão de cartas com pontuação superior à do adversário, sem ultrapassar 21 pontos. As regras incluem a distribuição inicial de duas cartas para cada participante , opções de "HIT"(pedir carta) ou "STAY"(manter a mão) para o jogador , e uma lógica automática para o carteador (pedir carta se a mão for menor que 17, e parar com 17 ou mais). A pontuação das cartas segue o padrão do jogo, com figuras valendo 10 e Ás valendo 1 ou 11 pontos. A arquitetura do sistema compreende duas Máquinas de Estados Finitos (FSMs) principais: uma para o controle do fluxo do jogo (blackjack) e outra para a geração e seleção de cartas (Random ou manual, para caso de testes). As entradas do sistema incluem botões para as ações do jogador (HIT, STAY, START), sendo o START também responsável pelo reset. As saídas compreendem indicadores de status do jogo (WIN, LOSE, TIE) e displays de 7 segmentos para a carta atual (em hexadecimal) e a soma total da mão do jogador (em decimal). Este trabalho visa demonstrar a aplicação prática de VHDL e FSMs no desenvolvimento de um sistema interativo em ambiente FPGA.

2 ESTRATÉGIA DE RESOLUÇÃO

Para implementar em VHDL o jogo de Blackjack, utilizamos como resolução duas máquinas de estados finitas (finite state machine - FSM) que atuam em conjunto para a realização das rodadas do jogo. Em uma das FSM, reside a lógica para gerar e apresentar cartas que podem ser aleatórias ou manualmente colocadas, enquanto na segunda máquina está inserida a lógica por trás do estouro (isto é, a soma das cartas ultrapassar 21) ou a possível comparação das somas de mãos caso ambos carteador e jogador tenham escolhido enviar um sinal de STAY e o programa não tenha uma resposta imediata para um vencedor.

2.1 FSM - GERADOR DE CARTAS

Em primeiro lugar, construímos uma máquina de estados finita com objetivo de gerar cartas que podem ser aleatórias para os casos de implementação dinâmica do jogo ou manuais, exclusiva para casos de testagem relacionadas a acurácia do programa em VHDL. O comportamento do gerador de cartas pode ser observado na esquemática construída abaixo:



A máquina de estados para o gerador de cartas foi projetada de forma que o sinal reqManual seja usado apenas para testes de funcionalidade. Em uma condição de jogo normal, o reqManual estará sempre em '0'. Quando o sinal req é ativado a partir do estado idle, o sistema decide ir para o CardG, onde uma carta será criada de forma aleatória (reqManual = 0) ou inserida manualmente. No próximo ciclo de clock, essa carta será lida no estado read e então enviada como entrada para a FSM principal que gerencia o jogo.

2.2 DIAGRAMA DE TRANSIÇÃO DO GERADOR DE CARTAS

Para detalhar o comportamento da máquina de estados do gerador de cartas, a tabela abaixo mostra como ela transita entre os diferentes estados. Essa tabela define as regras para o funcionamento do programa em VHDL, com base nos sinais de entrada.

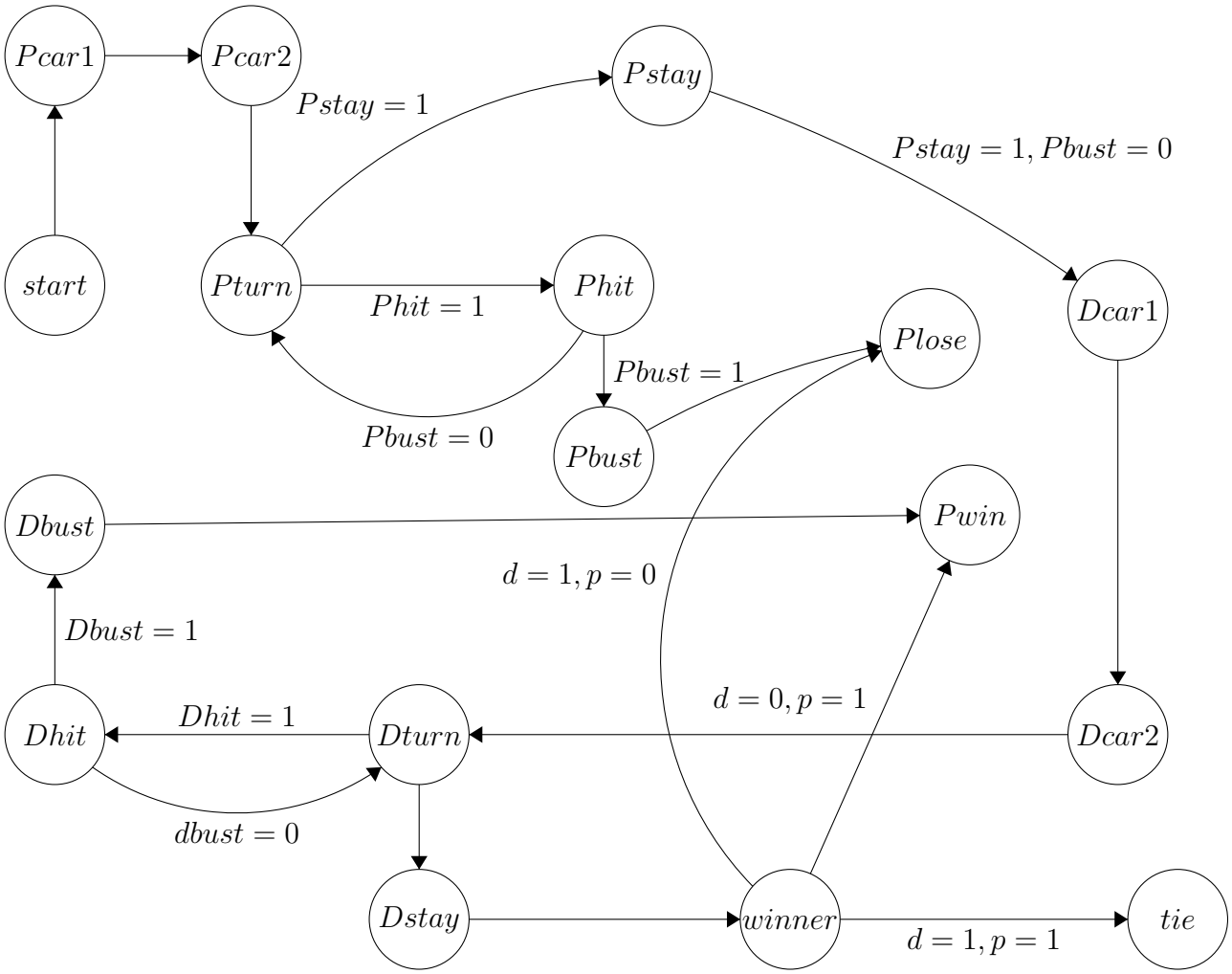
Tabela 1 – DIAGRAMA DE PRÓXIMO ESTADO

CURRENT	req	reqManual	NEXT
idle	0	x	idle
idle	1	x	CardG
CardG	0	x	idle

A tabela acima descreve as transições de estado para a FSM do gerador de cartas. Ela explica como o sistema se move dentro de um estado CURRENT, utilizando os sinais req e reqManual. O modo reqManual é prioritário para testar diferentes cenários, enquanto um req em '0' faz a FSM voltar para o estado idle. O estado read indica quando a carta está pronta para ser utilizada pelo restante do sistema.

2.3 FSM - EXECUÇÃO DO JOGO BLACKJACK

A projeção da segunda máquina de estados seguiu à risca as recomendações dadas pelo problema, ciente das limitações apresentadas para a execução do jogo na FPGA. Dessa forma, a proposta escolhida foi que o player jogue até que haja um estado de BUST ou STAY, passando a vez para o dealer que decide de forma automatizada se transita para um estado de HIT ou STAY. O jogo se inicia com a distribuição obrigatória de duas cartas, que não tem qualquer tipo de condição para serem dadas. A partir da soma de ambas, o jogador ou carteador decide no estado de TURN se deseja solicitar outra carta ou parar o jogo com o sinal de STAY.



Observa-se que só transitamos ao estado de WINNER quando ambos carteador e jogador enviam o sinal de STAY, no cenário em que o jogo teria acabado sem nenhum dos dois terem ultrapassado 21 na soma de suas cartas. Essa comparação está implicita na máquina de estados e o resultado dela é o sinal '1' que simboliza qual dos dois tem o maior somatório em mãos, ou ainda '1' para ambos carteador e jogador caso possuem a mesma soma, configurando um empate. A única outra possibilidade de empate é o cenário onde ambos jogador e carteador ultrapassam a soma de 21 pontos.

2.4 DIAGRAMA DE TRANSIÇÃO DAS RODADAS DO JOGO

Como citado anteriormente, a lógica de transição do jogo foi alinhada às limitações do ambiente de execução, seguindo todo o possível para ainda ser fielmente um jogo de Blackjack. A Tabela 2 detalha o comportamento da Máquina de Estados Finitos (FSM) principal do jogo. Vale ressaltar que, para otimização e controle preciso do fluxo de dados, há um estado temporário (dummy) omitido na representação visual da máquina de estados, localizado imediatamente antes dos estados de obtenção de carta. Este estado intermediário é crucial para sincronizar a requisição e a leitura da carta com a máquina

de geração, garantindo que a lógica da FSM principal opere apenas com cartas válidas já disponibilizadas.

Tabela 2 – DIAGRAMA DE PRÓXIMO ESTADO

CURRENT	Phit	Pstay	Pbust	Dhit	Dstay	Dbust	p	d	NEXT
start	x	x	x	x	x	x	x	x	Pcar1
Pcar1	x	x	x	x	x	x	x	x	Pcar2
Pcar2	x	x	x	x	x	x	x	x	Pturn
Pturn	1	0	0	x	x	x	x	x	Phit
Phit	x	x	0	x	x	x	x	x	Pturn
Phit	x	0	1	x	x	x	x	x	Pbust
Pbust	x	x	x	x	x	x	0	1	Plose
Pstay	0	1	0	x	x	x	x	x	Dcar1
Dcar1	x	x	x	x	x	x	x	x	Dcar2
Dcar2	x	x	x	x	x	x	x	x	Dturn
Dturn	x	x	x	1	0	0	x	x	Dhit
Dhit	x	x	x	x	x	0	x	x	Dturn
Dhit	x	0	1	x	x	1	x	x	Dbust
Dbust	x	x	x	x	x	x	0	1	Pwin
Dhit	x	x	x	0	1	0	x	x	winner
winner	x	x	x	x	x	x	1	0	Pwin
winner	x	x	x	x	x	x	0	1	Plose
winner	x	x	x	x	x	x	0	0	tie
winner	x	x	x	x	x	x	1	1	tie

A tabela acima detalha as transições de estado para a FSM principal do Blackjack. Ela descreve como o jogo avança ao longo de um estado CURRENT, baseado nas ações do jogador (Phit, Pstay) e do dealer (Dhit, Dstay), bem como nos resultados do jogo (Pbust, Dbust, p, d). Essa formalização é essencial para a implementação precisa do fluxo de jogo em VHDL, garantindo que todas as regras e condições de vitória, derrota ou empate sejam corretamente representadas e executadas.

2.5 INTERAÇÃO ENTRE AS MÁQUINAS DE ESTADO DO PROBLEMA

A complexidade de um sistema como o Blackjack em um ambiente de FPGA exige uma abordagem modular, onde diferentes funcionalidades são delegadas a FSMs específicas. Neste projeto, a máquina de estados para o gerador de cartas funciona como um subsistema fundamental, fornecendo as cartas necessárias para a FSM principal que orquestra o jogo de Blackjack. A saída do gerador de cartas funcionará como entrada para a segunda máquina de estados, onde o jogo Blackjack efetivamente é executado.

O mecanismo de comunicação entre essas duas FSMs é iniciado pelo sinal de 'req'. Este sinal é ativado pela máquina de estados do Blackjack quando há a necessidade de uma nova carta, seja para a distribuição inicial ou para uma jogada de 'HIT' tanto do jogador

quanto do dealer. Ao acionar 'req', a FSM do Blackjack entra em um estado temporário — o já mencionado "estado dummy"— que não é explicitamente representado no diagrama visual para manter a clareza e legibilidade. Uma vez que a carta é disponibilizada, a FSM do Blackjack prossegue com suas transições de estado, avaliando a nova pontuação da mão e decidindo, por exemplo, se ocorreu um "bust"(estouro de 21) ou se a jogada pode continuar, garantindo uma sincronização eficiente e segura entre as duas entidades.

3 RESOLUÇÃO

3.1 FASE DE PLANEJAMENTO

Para a implementação do Blackjack, utilizamos o modelo proposto pelo exercício e que, no caso das máquinas de estado que construímos, se deu da seguinte maneira:

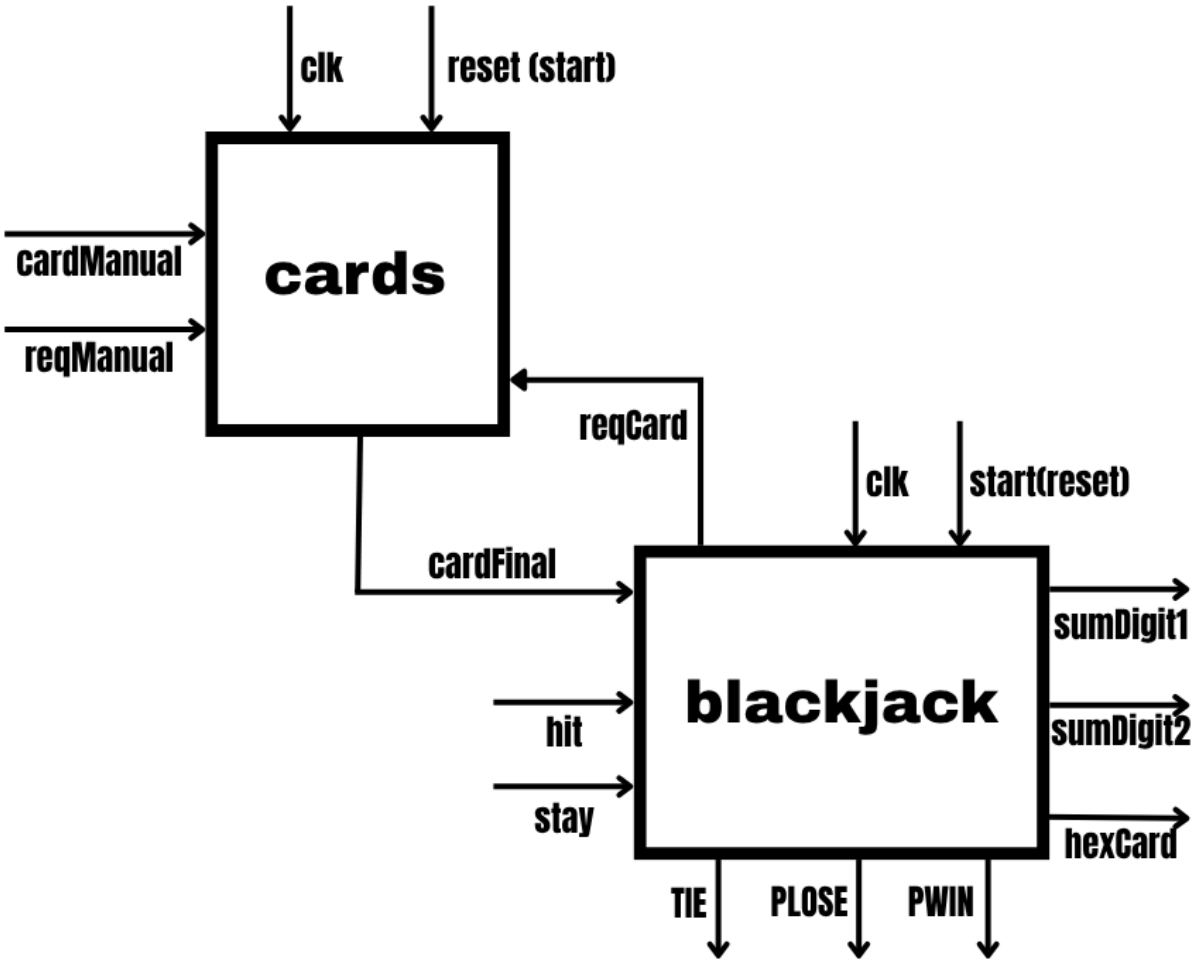


Figura 1 – Planejamento da resolução

As máquinas cards e blackjack interagem a partir dos sinais de req e da carta gerada. Quando o sinal de req é enviado pela máquina 2, a máquina 1 transita para o estado de cardG onde a carta será gerada. Enquanto isso, a máquina blackjack transita para o estado dummy para termos certeza de que ela receberá uma carta válida para dar prosseguimento ao jogo. Uma vez que a carta é enfim gerada, a máquina cards retorna a carta (um numero de 1-13) e a máquina blackjack agora consegue fazer alterações no somatório dos jogadores e conferir as possibilidades de estouro, de acordo com essa carta válida gerada pela outra máquina.

3.2 ENTIDADE CARDS

Para a primeira máquina de estados vista na Tabela 1, utilizamos uma entidade cards que gera um número aleatório a partir de uma SEED fixa ou colocar na saida cardFinal o valor armazenado na entrada cardManual caso o reqManual esteja ativo.

```
entity cards is
    port (
        clk         : in  std_logic;
        reset       : in  std_logic;
        reqCard     : in  std_logic;
        reqManual   : in  std_logic;
        cardManual  : in  std_logic_vector(3 downto 0);
        cardFinal   : out std_logic_vector(3 downto 0)
    );
end cards;
```

Figura 2 – Entidade geradora de cartas

Para a entidade cards, clk representa o clock (mesmo clock recebido pela FSM2), reset retorna a máquina imediatamente para o estado de idle (mesmo sinal conectado ao START da FSM2), reqCard é o sinal recebido pela FSM2 que define a transição de blackjack de idle para cardG, reqManual define se a cartaFinal será um número aleatório ou o valor de input colocado pelo cardManual e, por fim, cardFinal é a carta que enviamos para a máquina Blackjack, que fará operações a partir dessa carta recebida para atualizar os valores do player e dealer a partir dessa entrada.

3.3 ENTIDADE BLACKJACK

No caso da segunda máquina de estados vista na Tabela 2, utilizamos uma entidade blackjack que faz alterações em seus sinais playerValue e dealerValue de acordo com as cartas recebidas pela FSM 1. Essa entidade é responsável por receber essas cartas e realizar operações que garantem o funcionamento dentro das regras do jogo. Aqui está situada toda a lógica de vitória, derrota, estouro e tratamento de casos excepcionais (i.e estouros com Áses na mão).

```
entity blackjack is
    port (
        HIT : in  std_logic;
        STAY : in  std_logic;
        START : in  std_logic;
        CLK : in  std_logic;
        CARD : in  std_logic_vector(3 downto 0);

        REQCARD : out  std_logic;
        PWIN : out std_logic;
        PLOSE : out std_logic;
        TIE : out std_logic;
        hexCard : out std_logic_vector(6 downto 0); --carta pescada
        sumDigit1 : out std_logic_vector(6 downto 0); --soma
        sumDigit2 : out std_logic_vector(6 downto 0) --soma
    );
end blackjack;
```

Figura 3 – Entidade executora do jogo

Como mencionado anteriormente, os sinais de CLK e START(reset) são divididos entre as duas máquinas, o sinal de HIT leva o req para 1, dando início ao processo de gerar a carta para o jogo, STAY faz com que o jogador ou dealer pare de puxar cartas a fim de evitar o estouro, CARD é a carta recebida pela FSM1, PWIN/PLOSE/TIE são os estados de resultado do jogo, que podem ser atingidos por meio de diferentes condições definidas na arquitetura da Blackjack e, por mim, as saídas de display exibem a última carta pescada pelo jogador ou dealer em valor hexadecimal (hexCard) e a soma do jogador/dealer de acordo com a atualização depois de pescar cada carta (SumDigit1 e SumDigit2).

3.4 ENTIDADE TOP-LEVEL E MAPEAMENTO FPGA

Para que as entidades acima funcionem em paralelo, utilizamos uma terceira entidade que tem como função unicamente ligar ambas. Suas entradas e saídas podem ser observadas abaixo:

```
entity implementacao is
    port (
        clk         : in  std_logic;
        start       : in  std_logic;
        hit         : in  std_logic;
        stay        : in  std_logic;
        reqManual   : in  std_logic;
        cartaManual : in  std_logic_vector(3 downto 0);
        Pwin        : out std_logic;
        Plose       : out std_logic;
        TIE         : out std_logic;
        hex_disp    : out std_logic_vector(7 downto 0);
        sum1_disp    : out std_logic_vector(7 downto 0);
        sum2_disp    : out std_logic_vector(7 downto 0)
    );
end implementacao;
```

Figura 4 – Entidade top level do problema

Na top level criada, os mesmos clock e reset/start são divididos para as FSM1 e FSM2. hit e stay são entradas de Blackjack, reqManual e cartaManual são entradas do gerador de cartas. As saídas da top-level são todas da FSM2, os sinais de reqCard e a carta gerada pela FSM1 são controlados por meio de signals declarados na arquitetura da top-level. A partir dessa entidade, fizemos o mapeamento para a implementação na FPGA que se deu da seguinte maneira:

variáveis VHDL	mapeamento FPGA
clk	KEY(0)
reset / start	KEY(1)
SW(1)	stay
SW(5)	reqManual
SW(0)	hit
SW(9), SW(8), SW(7), SW(6)	cartaManual
LEDR(0)	Plose
LEDR(1)	TIE
LEDR(2)	Pwin
HEX0	sum1disp
HEX1	sum2disp
HEX3	hexdisp

Tabela 3 – Mapeamento na placa FPGA

4 NOTAS E CONSIDERAÇÕES FINAIS

O presente trabalho funciona para os cenários de solicitar e receber uma carta, atualizar corretamente os valores da mão do jogador e carteador. O código lida com a possibilidade de valores do Ás (decide de forma inteligente se deve somá-lo como 11 ou 1), lida com as possibilidades de estouro com e sem ás, partindo de qualquer tipo de valor de carta recebido, e exibe corretamente nos LEDs correspondentes o resultado da rodada jogada.

A projeção da segunda máquina de estados seguiu à risca as recomendações dadas pelo problema, ciente das limitações apresentadas para a execução do jogo na FPGA, onde principalmente os displays são limitados, fazendo com que haja a necessidade de jogar completamente a rodada do jogador e só depois de haver um STAY, isto é, o jogador escolhe terminar sua vez sem que tenha estourado o limite da soma em 21, é que se começa a jogada do dealer. Dessa forma, uma vez que o jogador estoura, o carteador sempre vence.

Para melhor legibilidade visual, abstraímos os estados dummies do problema que funcionam da seguinte maneira: um estado genérico solicita uma carta e transita para um estado dummy correspondente. Dentro desse estado, ele fica em espera enquanto a carta estiver inválida (0000). Quando a carta é corretamente recebida e interpretada pelo sistema, ele transita para um novo estado de soma correspondente, onde serão feitas alterações nos valores de jogador e dealer de acordo com a situação.

A arquitetura de Blackjack não lida com a possibilidade de receber uma carta inválida, isto é, a entrada '0000' ao enviar o sinal de reqManual. Quando um sinal de reqManual é enviado com o sinal inválido no input, a máquina entra em um estado de loop pois ela permanece no estado dummy até que a carta seja diferente de 0, o que não irá acontecer já que o sinal de req é colocado em 0 na transição para o estado dummy.