Trabalho Final

Jogo de Blackjack

- **Objetivos:**
    - Desenvolvimento, em VHDL, de um ***Circuito Digital Sintetizável*** que implemente um jogo de *Blackjack;*
    - Desenvolvimento, em VHDL, de um ***Testbench*** para a *Verificação*.
    - Análise da Cobertura de Código do Testbench;
    - Aplicação de *Constraints*;
    - Síntese Lógica do Projeto;
    - Análise dos Resultados Projeto.

- **Regras do Jogo:**

O Blackjack é um jogo de cartas, cujo jogador (*player*) enfrenta um adversário (*dealer*). O objetivo do jogo é obter o maior somatório de pontos, sem que esta contagem de pontos ultrapasse o valor 21. Os pontos do *player* e/ou do *dealer* são obtidos a partir do somatório dos valores das cartas de cada jogador.

O jogo começa com o *dealer* distribuindo duas cartas para cada um dos jogadores (distribuição de forma alternada). Após a distribuição das cartas iniciais, o *player* escolhe "*hit*" para receber mais uma carta, ou "*stay*" para não receber mais cartas. Após o jogador indicar "*stay*", este nunca mais poderá solicitar novas cartas.

Se o somatório de pontos do *player* ultrapassar a contagem de 21, este perde a partida automaticamente (independente da contagem de pontos do *dealer*). Caso isto não ocorra, o *player* espera pela análise do somatório das cartas do *dealer*. O *dealer* também pode solicitar "*hit*" ou "*stay*" de acordo com o somatório atual das suas cartas. Assim como o player, uma vez indicado "*stay*", esta opção não poderá mais ser revertida.

Caso o somatório de pontos do *dealer* ultrapasse a soma de 21 este perde o jogo, e o *player* é declarado vencedor. Caso nenhuma das duas contagens ultrapasse os 21 pontos, quem possuir o maior somatório de pontos é declarado vencedor da partida.

Caso, ambos os jogadores possuam a mesma contagem de pontos, o jogo é finalizado, e um empate é declarado.

Os valores individuais das cartas são apresentados na Tabela 1:

Carta	Valor
A	1 ou 11
2 até 9	Respectivo valor da carta
10, J, Q, K	10

Tabela 1 - Valores das cartas.

- **Especificações do Projeto:**

Você desenvolverá um console de um jogo de *Blackjack* eletrônico. O console possuirá quatro sinais de entrada: **RESET**, **HIT**, **STAY, DEBUG** e **SHOW**. Estas entradas permitem ao *player* iniciar e/ou reiniciar o jogo, controlar o pedido de cartas durante a partida, além de controlar a apresentação dos somatórios finais de pontos de cada um dos jogadores.

As cartas são oriundas de um circuito externo que as armazena. Você pode assumir que estas cartas já estão previamente embaralhadas.

O jogo é iniciado pela energização do console e pelo pressionamento do botão de **RESET**.

Após a inicialização do circuito, a máquina de estados finitos (FSM) irá começar a distribuir as cartas conforme foi descrito nas regras do jogo acima.

Após a distribuição das primeiras cartas, utilizando os botões do console, o *player* terá a escolha de solicitar mais cartas (**HIT**), ou ficar com a quantidade de cartas atuais (**STAY**). Esta última opção é irreversível, pois passa ao *dealer* o direito de jogar.

A máquina de estados deverá avaliar as cartas do *player* e do *dealer*, e ao final do jogo, determinar o resultado de vitória (**WIN**), derrota (**LOSE**) ou empate (**TIE**).

A máquina de estados deverá controlar automaticamente a quantidade de cartas entregues, bem como, a determinação de vitória, derrota ou empate, além da disponibilização dos respectivos resultados.

A Figura 1 apresenta as interfaces do projeto:

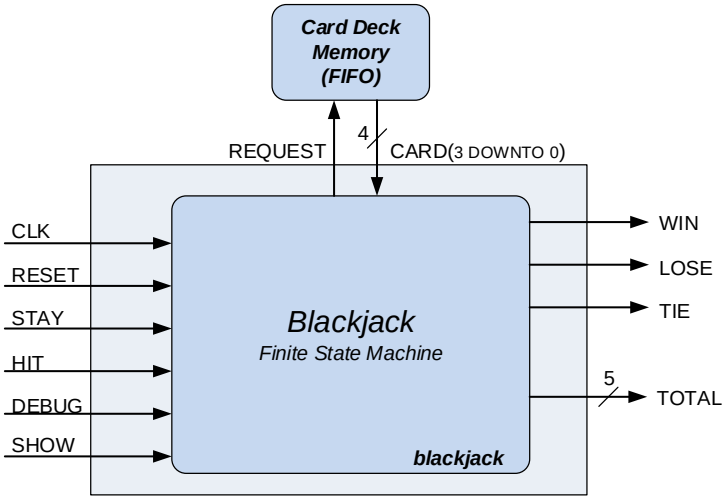

Figura 1 – Diagrama de Referência.

Assuma que os sinais de entrada **HIT** e **STAY** estão conectados a botões externos, de forma que o *player* possa indicar sua opção de ação.

Você deve assumir que a **transição de nível lógico baixo para alto** nos sinais de HIT e STAY indicam uma solicitação.

A entrada de cartas (dados) possuirá 4 bits para a representação do valor da próxima carta a ser distribuída para a FSM. Você pode utilizar uma FIFO para a implementação deste elemento. Considere esta FIFO um elemento externo ao seu circuito lógico.

A entrada de RESET (ativo em nível lógico alto) leva o sistema para o seu estado inicial.

O sinal de entrada CLK é a referência de relógio para todo o sistema.

Os três sinais de saída (WIN, LOSE, TIE) são referentes ao jogo em andamento, porém só podem ser atualizados após a declaração de fim de jogo, isto é, durante a execução da partida estes sinais devem permanecer em nível lógico baixo.

Todos os sinais de saída são ativos em nível lógico alto.

Durante a execução o jogo, o barramento TOTAL apresentará somente a pontuação do player (neste caso, considere o sinal DEBUG em nível lógico baixo).

Caso o sinal DEBUG esteja em nível lógico alto (modo debug ativado), o barramento TOTAL apresentará as contagens conforme a tabela verdade apresentada na Tabela 2

Debug	Show	Total
1	0	Somatório Player
1	1	Somatório Dealer

Tabela 2 – Tabela Verdade do Debug.

Após a finalização da partida, o barramento TOTAL apresentará a pontuação do player se o botão SHOW estiver em nível lógico baixo, e a pontuação do *dealer* caso o botão SHOW esteja em nível lógico alto independente do estado do sinal DEBUG.

- **Suposições e Restrições:**
  - A interface de topo do projeto **deverá** ser respeitada;
  - Utilizar o nome **"blackjack"** para a entidade de topo;
  - Existe apenas um *player* e um *dealer*;
  - O *player* é o jogador que acessará os sinais da entidade HIT e STAY;
  - O *dealer* será a máquina;
  - **Não** é necessária atenção sobre o circuito de energização do console. Assuma que o controle referente a isto está completamente pronto;
  - O sinal de RESET pode ser aplicado a qualquer momento do jogo, porém este sinal deve ser síncrono com o sinal de clock do circuito (CLK).
  - Após um evento de RESET (**ativo alto**), o console deverá, **sincronamente**, voltar para o seu estado inicial;
  - As transições do circuito lógico deverão ser síncronas **a borda de subida** do sinal de *clock* (CLK);
  - Após o RESET a contagem de pontos de ambos os jogadores deverá ser zerada;
  - No estado subsequente ao RESET, a *FSM* **deverá** distribuir, alternadamente, duas cartas para cada jogador, isto é, a primeira carta deverá ser entregue ao *player*, a segunda ao *dealer*, a terceira ao *player* e a quarta ao *dealer*;
  - Os sinais de entrada HIT e STAY **não podem ser** pressionados simultaneamente, caso isto ocorra, a máquina de estados não deverá fazer nada;
  - O player poderá apertar o botão HIT quantas vezes quiser;
  - Se o *player* ultrapassar 21 pontos, o *player* obrigatoriamente perde a partida;
  - O *dealer* deverá solicitar mais cartas **sempre que seu somatório de pontos for igual ou inferior a 16 pontos**;
  - O *dealer* não deverá solicitar mais cartas **se seu somatório for superior a 16 pontos**;
  - Se *player* ultrapassar o somatório de 21 pontos, **este perde a partida** e o *dealer* vence, independente da sua contagem.

- Se o *dealer* ultrapassar o somatório de 21 pontos, **este perde a partida** e o *player* vence.
- Se o *player* e o *dealer*, ao final do jogo, possuírem o mesmo somatório de pontos, o jogo terminará empatado.
- A frequência do relógio de entrada (CLK) será de 200MHz.

- **Fluxograma das Tarefas:**

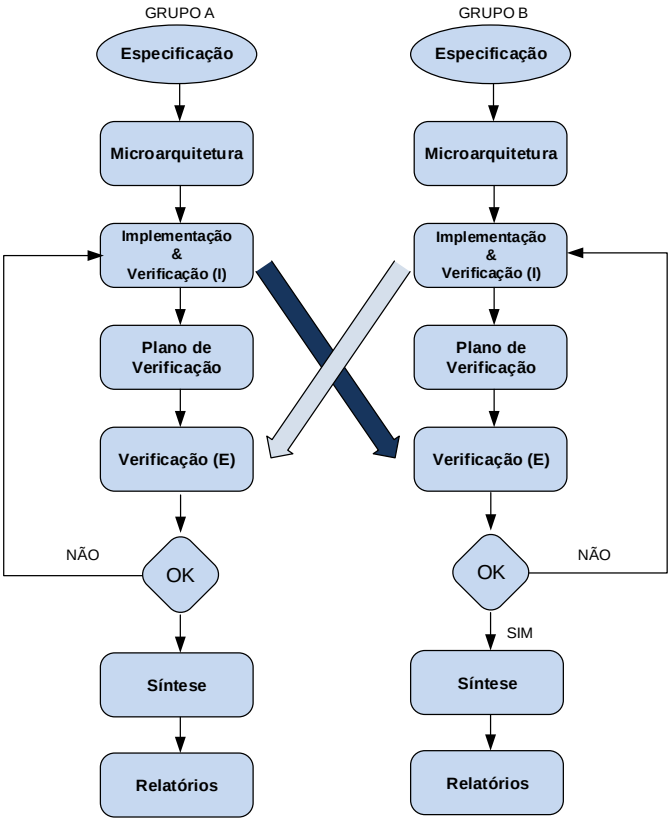

Figura 2 – Cronograma de Etapas.

- **Definições do Projeto 2018/2:**

As características funcionais e estruturais não definidas nesta especificação deverão ser tratadas na reunião de projeto. Estas definições serão publicadas em um documento anexo que estará disponível no moodle da disciplina para ser seguida por todas as duplas de trabalho.

- **Cronograma das Tarefas:**

Etapa	Descrição	Aulas
Especificação	Apresentação e Análise da Especificação; Reunião de Projeto; *Spec Freeze*; Definição dos Grupos de Trabalho e Duplas Parceiras;	**04/10**
Implementação	Desenvolvimento da Microarquitetura; Codificação do RTL; Codificação do Testbench (I - Interno ao Grupo); Verificação do Projeto da própria dupla; Cobertura de Código (Code Coverage); RTL Freeze; Troca dos Códigos;	**09/10 à 30/10**
Verificação	Plano de Verificação; Codificação do Testbench (E – Externo ao Grupo); Verificação do Projeto dupla parceira; Apresentação dos Relatórios de Verificação;	**30/10 à 20/11**
Síntese Lógica	Aplicação de Constraints; Síntese Lógica;	**20/11 à 27/11**
Relatórios	Geração e Análise dos Relatórios de Síntese;	**27/11**
Entrega do Projeto	Entrega do Artigo.	**29/11**

Tabela 3 – Descrição das Etapas.

- **Entregas:**

Etapa	Descrição	Data
Especificação	Lista das duplas e suas respectivas duplas parceiras;	**04/10**
Implementação	Microarquitetura; Código da FSM (*.vhd); Testbench – Interno (*.vhd); Relatório da Cobertura de Código (Code Coverage); Entrega dos Códigos para Dupla Parceira	**30/10**
Verificação	Plano de Verificação; Testbench – Externo (*.vhd); Relatório de Verificação; Apresentação do Relatório de Verificação para Dupla Parceira	**06/11** **13/11** **20/11**
Versão Final (Correções)	Código da FSM corrigido (*.vhd);	**27/11**
Síntese Lógica e Relatórios	Constraints (*.sdc); Síntese Lógica (script); Relatórios de Síntese;	**27/11**
Entrega do Projeto	Relatório Final (**em formato de artigo**);	**29/11**

Tabela 4 – Entregas das Etapas.

Atualizado em 04/10/2018.