EN202
Projet Blackjack

Filière Électronique

Semestre 7

Besson Thomas

Encadrant : Y. Bornat

Table des matières

1 Introduction

Ce projet consiste à implémenter en **VHDL** le jeu du **Blackjack**, en utilisant une **machine à états finis**. L'objectif est d'étudier à la fois la logique séquentielle du jeu et son architecture matérielle, comprenant une **unité de contrôle** et un **chemin de données**.

2 Règles du Blackjack

2.1 Objectif du jeu

Obtenir une main dont la valeur est la plus proche possible de 21 sans le dépasser. Le joueur affronte uniquement le croupier, pas les autres joueurs.

2.2 Valeur des cartes

Cartes	Valeur
2 à 10	Valeur faciale
Valet, Dame, Roi	10
As	1 *ou* 11 (selon le cas le plus avantageux)

2.3 Déroulement d'une donne

1. **Mise** : chaque joueur place sa mise.
2. **Distribution** : deux cartes sont distribuées au joueur et au croupier.
3. **Tour du joueur** : choix entre *hit* (tirer) ou *stand* (rester).
4. **Tour du croupier** : tire jusqu'à au moins 17.
5. **Résolution** : comparaison des totaux pour déterminer le résultat.

3 Structure et Architecture du Projet

3.1 Vue d'ensemble

Le système est constitué de deux parties principales :

- **Unité de contrôle (FSM)** : gère les transitions et signaux logiques ;
- **Chemin de données** : assure les calculs, mémorisations et comparaisons.

3.2 Blocs matériels principaux

- **RAM64** : mémoire des cartes jouées ;
- **ACC (Accumulateur)** : somme totale des cartes tirées ;
- **Compteur mod 52** : sélection de la carte courante ;
- **Registres ZERO et PERDU** : indicateurs d'erreur et de perte ;
- **Afficheurs** : visualisation des cartes et des totaux.

3.3 États principaux de la FSM

1. **DÉMARRAGE** : initialisation et attente d'un nouveau jeu.
2. **COMPTER** : activation du compteur et lecture d'une carte.
3. **ACCUMULER** : ajout de la carte au total, mise à jour des registres.
4. **ÉCRIRE** : vérification de dépassement (score ≥ 22) et écriture mémoire.

Les transitions se font uniquement sur le **front montant de l'horloge**.

4 Fonctionnement du Process et des États en VHDL

4.1 Bloc `process`

En VHDL, un `process(clk, reset)` s'exécute à chaque changement d'un signal de la liste de sensibilité :

- si `reset = '0'`, retour à l'état initial ;
- sur front montant de l'horloge (`rising_edge(clk)`), exécution d'un cycle FSM.

Chaque cycle lit l'état courant, exécute le code associé, puis prépare la transition vers l'état suivant.

4.2 Bloc `case` et transitions

L'instruction `case state is` sélectionne le bloc d'instructions selon la valeur du signal `state`. Chaque bloc `when <état>` décrit un état de la FSM.

- Le code du bloc courant est exécuté pendant le cycle actif.
- Une modification de `state` ne prend effet qu'au **cycle suivant**.

```
process(clk, reset)
begin
  if reset = '0' then
    state <= start;
  elsif rising_edge(clk) then
    case state is
      when start =>
        output <= cardA + cardB;
        if output > "10101" then
          state <= busts;
        else
          state <= play;
        end if;

      when busts =>
        bust <= '1'; win <= '0'; stand <= '0'; hit <= '0';
    end case;
  end if;
end process;
```

Listing 1 – Structure simplifiée d'une FSM VHDL

Ainsi, une transition comme `state <= busts;` ne prendra effet qu'à la prochaine montée d'horloge.

5 Signaux et Logique Combinatoire

5.1 Signaux principaux

Signal	Fonction
`rst_compteur`	Réinitialise le compteur
`ce_compteur`	Active le défilement des cartes
`ld_acc`	Charge le registre accumulateur
`we_ram`	Active l'écriture en mémoire
`dépassement`	Indique une perte (score ≥ 22)

5.2 Vérifications combinatoires

- `ACC = 0` : carte invalide ;
- `ACC` $\geq$ 22 : perte automatique.

5.3 Affichage

Trois afficheurs montrent :

- la carte tirée ;
- la somme courante (`ACC1`) ;
- le total complet (`ACC2`).

6 Synthèse et Conclusion

Le projet **Blackjack en VHDL** illustre une conception claire et modulaire :

- la **FSM synchrone** gère les transitions d'états via l'horloge ;
- le **chemin de données** réalise les calculs et comparaisons combinatoires ;
- les interactions joueur (tirer, se tenir) sont traduites en signaux logiques.

Cette architecture sépare proprement le contrôle et les opérations, rendant le système fiable, évolutif et facile à tester. Une bonne compréhension du comportement séquentiel du `process` et du `case` est essentielle pour garantir le fonctionnement correct de la machine.