

## VHDL-FPGA

# Distributeur de Café avec Gestion de la Monnaie

Ecole Nationale Supérieure d'arts et métiers

Université Hassan II de Casablanca

Réalisé par :

Soufiane Saliki

Yacine Bendnaiba

Oumaima Ait El Moudene

Encadré par :

Prof : S.EL MOUMNI

## Introduction Générale

Le projet "Distributeur de Café avec Gestion de la Monnaie" basé sur VHDL-FPGA et implémenté avec Quartus vise à concevoir un simulateur de distributeur de café interactif. L'utilisateur peut choisir parmi différents cafés en insérant des pièces de monnaie, et le système assure une gestion précise de la monnaie insérée. Les fonctionnalités clés incluent la sélection du café via des boutons dédiés, la reconnaissance des valeurs des pièces en tant que signaux d'entrée, l'affichage de l'état du distributeur via des voyants en tant que signaux de sortie, et la délivrance du café lorsque la monnaie est suffisante. La conception utilise Quartus et VHDL pour intégrer efficacement les aspects de contrôle, de calcul et d'affichage dans un simulateur de distributeur de café fonctionnel.



## LISTE DES FIGURES

Figure 1: Quartus .....	6
Figure 2: VHDL .....	7
Figure 3: Les Entrées et les sorties : .....	10
Figure 4: Ports d'entrée (input ports): .....	11
Figure 5: Ports de sortie (output ports): .....	12
Figure 6: Architecture de Gestion de la Monnaie pour le Distributeur .....	14
Figure 7: Processus de Gestion de la Monnaie (Premier Processus) .....	15
Figure 8: Processus de Gestion de la Monnaie (Deuxième Processus) .....	16
Figure 9: Processus de Gestion de la Monnaie (Troisième Processus) .....	17
Figure 10: Processus de Gestion de la Monnaie (Troisième Processus) .....	18
Figure 11: Processus de Gestion de la Monnaie (Cinquième Processus) .....	19

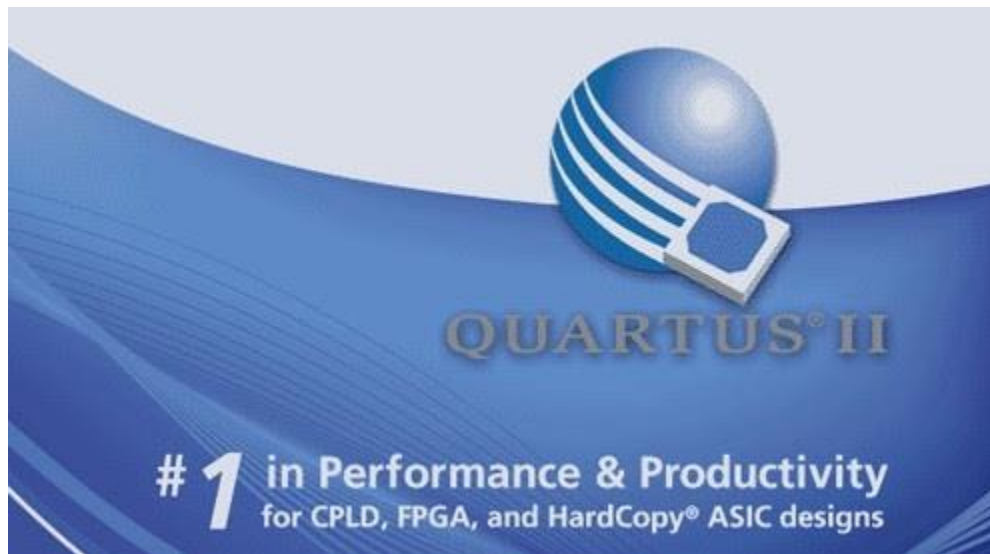
# SOMMAIRE

Introduction Générale .....	2
I. Les Outils utilisés.....	5
I. Quartus 2 .....	6
II. VHDL.....	7
II. La Réalisation .....	8
I. Cahier de charge.....	9
II. Les Entrées et les sorties : .....	10
1. Ports d'entrée (input ports): .....	11
2. Ports de sortie (output ports):.....	12
3. Description générale: .....	13
III. Architecture de Gestion de la Monnaie pour le Distributeur.....	14
1. Description générale: .....	14
2. Signaux internes:.....	14
3. Constantes: .....	14
IV. Processus de Gestion Financière du Distributeur Automatique .....	15
A. Processus de Gestion de la Monnaie (Premier Processus) :.....	15
B. Processus de Gestion de la Monnaie (Deuxième Processus) .....	16
C. Processus de Gestion de la Monnaie (Troisième Processus) .....	17
D. Processus de Gestion de la Monnaie (Quatrième Processus).....	18
E. Processus de Gestion de la Monnaie (Cinquième Processus).....	19
CONCLUSION .....	20

## I. Les Outils utilisés

## I. Quartus 2

✚ **Description** : Quartus II est un environnement de conception pour les FPGA et les CPLD d'Altera (maintenant Intel). Il prend en charge les langages VHDL et Verilog, offre des outils de synthèse, de placement, et de simulation, facilitant la conception, l'optimisation, et la programmation des circuits logiques programmables. Cet outil est largement utilisé dans l'industrie électronique pour le développement de systèmes embarqués.



*Figure 1: Quartus*

## II. VHDL

**Description :** VHDL (VHSIC Hardware Description Language) est un langage de description matériel utilisé pour modéliser, simuler et synthétiser des circuits électroniques. Il permet de décrire le comportement et la structure d'un système électronique, facilitant ainsi la conception de circuits intégrés, de FPGA et d'autres dispositifs logiques programmables. VHDL est particulièrement utilisé dans le domaine de la conception électronique numérique pour décrire et spécifier le fonctionnement des circuits.

The logo for VHDL (Very High Speed Integrated Circuit Hardware Description Language) is displayed. It features the acronym "VHDL" in a large, bold, black serif font. Below the acronym, the full name "Very High Speed Integrated Circuit Hardware Description Language" is written in a smaller, bold, black serif font, arranged in two lines.

**VHDL**  
**Very High Speed Integrated Circuit**  
**Hardware Description Language**

*Figure 2: VHDL*

## II. La Réalisation



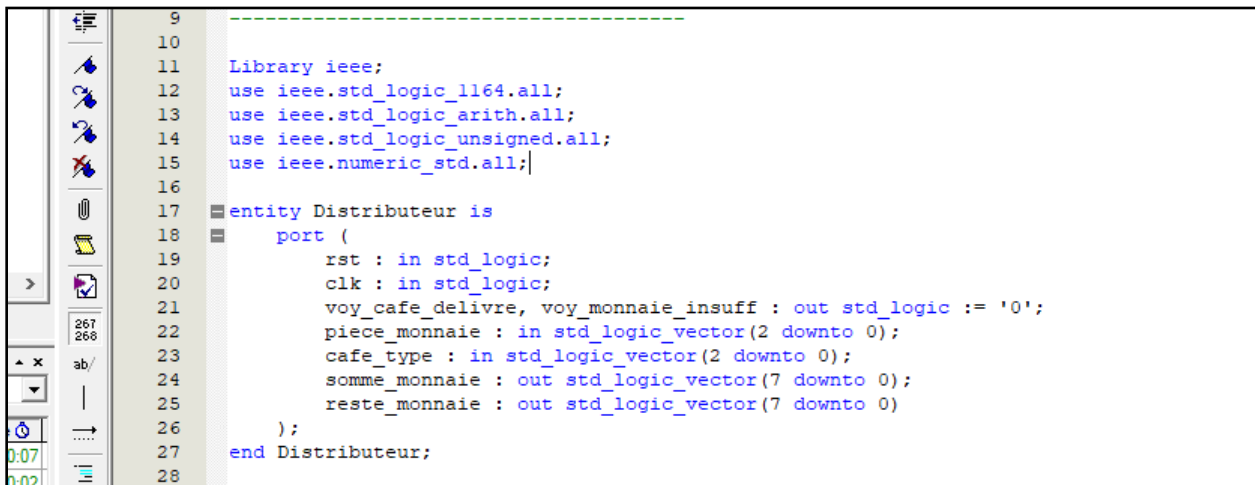
## I. Cahier de charge

Le cahier des charges pour le projet "Distributeur de Café avec Gestion de la Monnaie" spécifie les exigences principales pour la conception du simulateur utilisant Quartus et VHDL. Voici une description du cahier des charges adapté à Quartus VHDL :

1. **Sélection de Café :** Le système doit être capable de prendre en charge plusieurs types de café, tels que Café Noir et Café au Lait, avec des boutons dédiés en tant que signaux d'entrée. Ces signaux seront associés à des processus VHDL pour gérer la sélection du café.
2. **Gestion de la Monnaie :** Les signaux d'entrée doivent permettre à l'utilisateur d'insérer des pièces de monnaie de différentes valeurs. Des processus VHDL doivent être mis en place pour reconnaître la valeur totale des pièces insérées, en utilisant des opérations arithmétiques appropriées.
3. **Affichage de l'État :** Les voyants en tant que signaux de sortie doivent être configurés pour indiquer l'état du distributeur. Un voyant doit signaler si la machine est prête à délivrer le café, tandis qu'un voyant distinct doit indiquer si la monnaie insérée est insuffisante, exacte ou excédentaire par rapport au coût du café sélectionné.
4. **Délivrance du Café :** La délivrance du café doit être conditionnée par la suffisance de la monnaie insérée. Un processus VHDL doit être conçu pour déclencher la délivrance du café sélectionné lorsque la monnaie est suffisante. De plus, si nécessaire, la machine doit générer le reste de la monnaie, impliquant des opérations arithmétiques appropriées.

En utilisant Quartus et VHDL, la conception du simulateur doit intégrer ces fonctionnalités de manière efficace en combinant le contrôle, le calcul et l'affichage pour créer un système interactif de gestion de café avec une gestion monétaire appropriée.

## II. Les Entrées et les sorties :



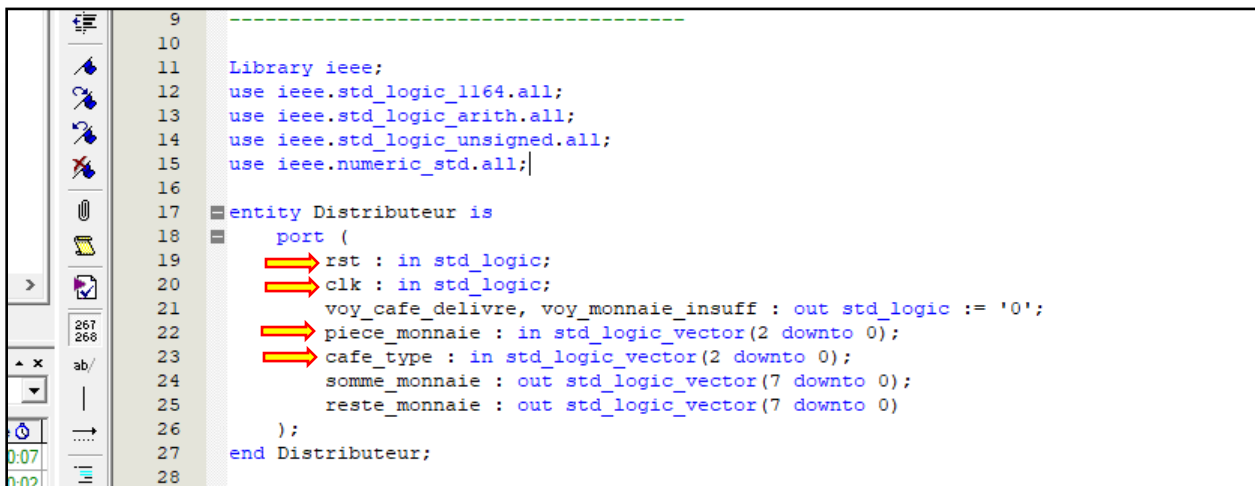
```
9  -----
10
11  Library ieee;
12  use ieee.std_logic_1164.all;
13  use ieee.std_logic_arith.all;
14  use ieee.std_logic_unsigned.all;
15  use ieee.numeric_std.all;
16
17  entity Distributeur is
18  port (
19      rst : in std_logic;
20      clk : in std_logic;
21      voy_cafe_delivre, voy_monnaie_insuff : out std_logic := '0';
22      piece_monnaie : in std_logic_vector(2 downto 0);
23      cafe_type : in std_logic_vector(2 downto 0);
24      somme_monnaie : out std_logic_vector(7 downto 0);
25      reste_monnaie : out std_logic_vector(7 downto 0)
26  );
27  end Distributeur;
28
```

*Figure 3: Les Entrées et les sorties :*

Ce code VHDL définit une entité nommée "Distributeur" qui représente un distributeur automatique. Voici une description des différents éléments de ce code :

## 1. Ports d'entrée (input ports):

- ✚ **rst** : Signal de réinitialisation. Il est de type `std_logic`, ce qui signifie qu'il peut prendre des valeurs '0' ou '1'.
- ✚ **clk** : Signal d'horloge. Il est de type `std_logic`, utilisé pour synchroniser les opérations du circuit avec le signal d'horloge.
- ✚ **piece\_monnaie** : Vecteur de signaux logiques de taille 3 bits, représentant la monnaie insérée dans le distributeur.
- ✚ **cafe\_type** : Vecteur de signaux logiques de taille 3 bits, représentant le type de café sélectionné.

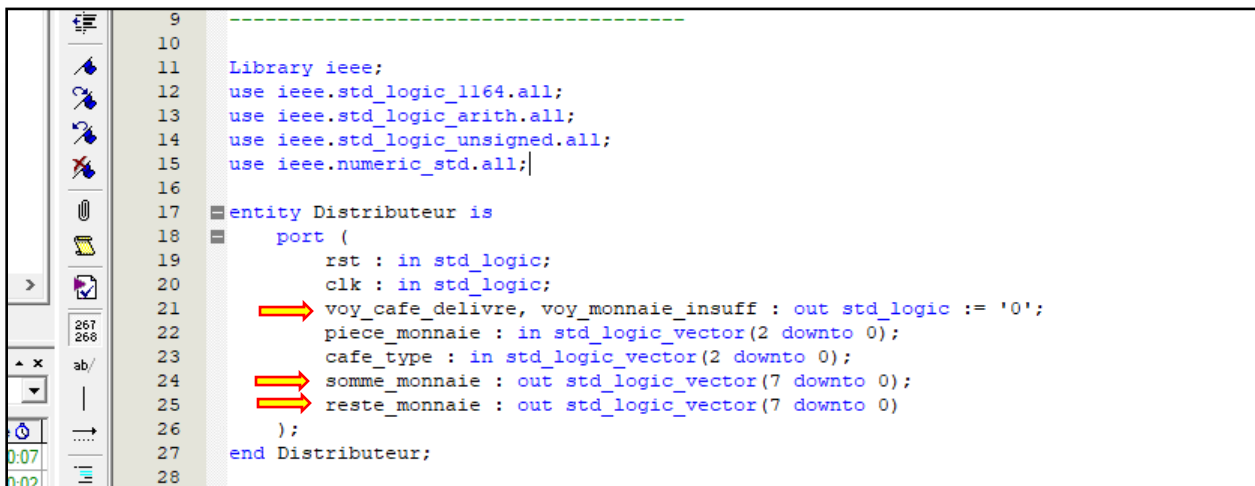


```
9 -----
10
11 Library ieee;
12 use ieee.std_logic_1164.all;
13 use ieee.std_logic_arith.all;
14 use ieee.std_logic_unsigned.all;
15 use ieee.numeric_std.all;
16
17 entity Distributeur is
18   port (
19     rst : in std_logic;
20     clk : in std_logic;
21     voy_cafe_delivre, voy_monnaie_insuff : out std_logic := '0';
22     piece_monnaie : in std_logic_vector(2 downto 0);
23     cafe_type : in std_logic_vector(2 downto 0);
24     somme_monnaie : out std_logic_vector(7 downto 0);
25     reste_monnaie : out std_logic_vector(7 downto 0)
26   );
27 end Distributeur;
28
```

*Figure 4: Ports d'entrée (input ports):*

## 2. Ports de sortie (output ports):

- ✚ **voy\_cafe\_delivre** : Signal de sortie indiquant si le café a été délivré ('1') ou non ('0').
- ✚ **voy\_monnaie\_insuff** : Signal de sortie indiquant si la somme de la monnaie insérée est insuffisante ('1') ou suffisante ('0') pour acheter le café.
- ✚ **somme\_monnaie** : Vecteur de signaux logiques de taille 8 bits représentant la somme totale de la monnaie insérée.
- ✚ **reste\_monnaie** : Vecteur de signaux logiques de taille 8 bits représentant la monnaie restante après l'achat du café.



```
9
10
11 Library ieee;
12 use ieee.std_logic_1164.all;
13 use ieee.std_logic_arith.all;
14 use ieee.std_logic_unsigned.all;
15 use ieee.numeric_std.all;
16
17 entity Distributeur is
18 port (
19     rst : in std_logic;
20     clk : in std_logic;
21     --> voy_cafe_delivre, voy_monnaie_insuff : out std_logic := '0';
22     piece_monnaie : in std_logic_vector(2 downto 0);
23     cafe_type : in std_logic_vector(2 downto 0);
24     --> somme_monnaie : out std_logic_vector(7 downto 0);
25     --> reste_monnaie : out std_logic_vector(7 downto 0)
26 );
27 end Distributeur;
28
```

Figure 5: Ports de sortie (output ports):

### 3. Description générale:

- L'entité "Distributeur" est destinée à être utilisée dans un environnement VHDL plus large.
- Le distributeur automatique est conçu pour accepter des pièces de monnaie (représentées par le port "piece\_monnaie") et permettre la sélection d'un type de café (représenté par le port "cafe\_type").
- Le distributeur génère des signaux de sortie indiquant si le café a été délivré avec succès ("voy\_cafe\_delivre") et si la somme de la monnaie insérée est insuffisante ("voy\_monnaie\_insuff").
- Les montants totaux de la monnaie insérée ("somme\_monnaie") et la monnaie restante après l'achat du café ("reste\_monnaie") sont également disponibles en sortie.
- Le signal de réinitialisation ("rst") est utilisé pour remettre le distributeur à un état initial.

### III. Architecture de Gestion de la Monnaie pour le Distributeur

```
27 end Distributeur;  
28  
29 architecture gestion_monnaie of Distributeur is  
30  
31   signal somme : std_logic_vector(7 downto 0) := (others => '0');  
32   signal reste : std_logic_vector(7 downto 0) := (others => 'Z');  
33   signal cafe_choix : std_logic_vector(2 downto 0) := (others => '0');  
34   -- prix des cafe|  
35   constant cafe_au_lait_prix : std_logic_vector(7 downto 0) := "00001010"; -- 10 DH  
36   constant Americano_prix : std_logic_vector(7 downto 0) := "00001100"; -- 12 DH  
37   constant Cappuccino_prix : std_logic_vector(7 downto 0) := "00000111"; -- 7 DH  
38   constant Espresso_prix : std_logic_vector(7 downto 0) := "00001001"; -- 9 DH  
39
```

*Figure 6: Architecture de Gestion de la Monnaie pour le Distributeur*

#### 1. Description générale:

L'architecture "gestion\_monnaie" du distributeur automatique (défini par l'entité "Distributeur") est responsable de la gestion des aspects financiers du système. Elle comprend des signaux internes pour suivre la somme totale de la monnaie insérée, la monnaie restante après l'achat, ainsi que le choix du café effectué par l'utilisateur. De plus, des constantes sont définies pour représenter les prix associés à chaque type de café disponible dans le distributeur.

#### 2. Signaux internes:

- ✚ **somme** : Vecteur de signaux logiques de taille 8 bits, représentant la somme totale de la monnaie insérée dans le distributeur. Il est initialisé à zéro.
- ✚ **reste** : Vecteur de signaux logiques de taille 8 bits, représentant la monnaie restante après l'achat d'un café. Il est initialement indéterminé ('Z').
- ✚ **cafe\_choix** : Vecteur de signaux logiques de taille 3 bits, représentant le choix du café fait par l'utilisateur. Il est initialisé à zéro.

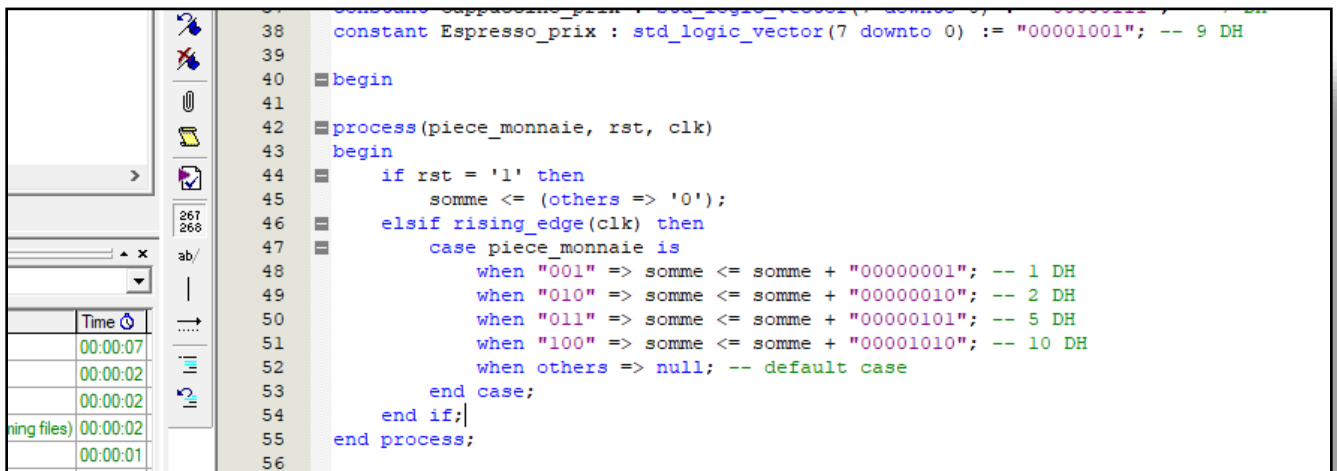
#### 3. Constantes:

- ✚ **cafe\_au\_lait\_prix** : Constante représentant le prix du café au lait en monnaie (10 DH).

- ✚ **Americano\_prix** : Constante représentant le prix de l'Americano en monnaie (12 DH).
- ✚ **Cappuccino\_prix** : Constante représentant le prix du Cappuccino en monnaie (7 DH).
- ✚ **Espresso\_prix** : Constante représentant le prix de l'Espresso en monnaie (9 DH).

## IV. Processus de Gestion Financière du Distributeur Automatique

### A. **Processus de Gestion de la Monnaie (Premier Processus) :**



```

38  constant Cappuccino_prix : std_logic_vector(7 downto 0) := "00001101"; -- 7 DH
39  constant Espresso_prix : std_logic_vector(7 downto 0) := "00001001"; -- 9 DH
40  begin
41  process(piece_monnaie, rst, clk)
42  begin
43      if rst = '1' then
44          somme <= (others => '0');
45      elsif rising_edge(clk) then
46          case piece_monnaie is
47              when "001" => somme <= somme + "00000001"; -- 1 DH
48              when "010" => somme <= somme + "00000010"; -- 2 DH
49              when "011" => somme <= somme + "00000101"; -- 5 DH
50              when "100" => somme <= somme + "00001010"; -- 10 DH
51              when others => null; -- default case
52          end case;
53      end if;
54  end process;
55
56

```

*Figure 7:Processus de Gestion de la Monnaie (Premier Processus)*

#### 1. **Sensibilité aux Signaux :**

- Ce processus est déclenché chaque fois qu'il y a un front montant sur le signal d'horloge (clk).
- Il est également sensible au signal de réinitialisation (rst).

#### 2. **Réinitialisation (Reset) :**

- Si le signal de réinitialisation (rst) est à l'état logique '1', la variable interne somme est remise à zéro (tous les bits à '0').

#### 3. **Incrémentation de la Somme :**

- Lorsqu'il n'y a pas de réinitialisation (rst à '0') et qu'un front montant est détecté sur le signal d'horloge (clk), le processus examine la valeur du vecteur piece\_monnaie.
- En fonction de la valeur de piece\_monnaie, la variable somme est mise à jour en ajoutant la valeur correspondante :

- "001" : Ajoute 1 à la somme (somme <= somme + "000000001").
- "010" : Ajoute 2 à la somme (somme <= somme + "000000010").
- "011" : Ajoute 5 à la somme (somme <= somme + "00000101").
- "100" : Ajoute 10 à la somme (somme <= somme + "00001010").
- Si la valeur de `piece_monnaie` ne correspond à aucun des cas spécifiés, aucune action n'est effectuée (default case).

## B. Processus de Gestion de la Monnaie (Deuxième Processus)

```

56
57 process(caffe_type, rst)
58 begin
59     if rst = '1' then
60         caffe_choix <= (others => '0');
61     else
62         caffe_choix <= caffe_type;
63     end if;
64 end process;
65

```

Figure 8: Processus de Gestion de la Monnaie (Deuxième Processus)

### 1. Sensibilité aux Signaux :

- Ce processus est déclenché chaque fois qu'il y a un changement d'état sur le signal de réinitialisation (`rst`).
- Il est également déclenché en cas de changement d'état sur le signal `caffe_type`.

### 2. Réinitialisation (Reset) :

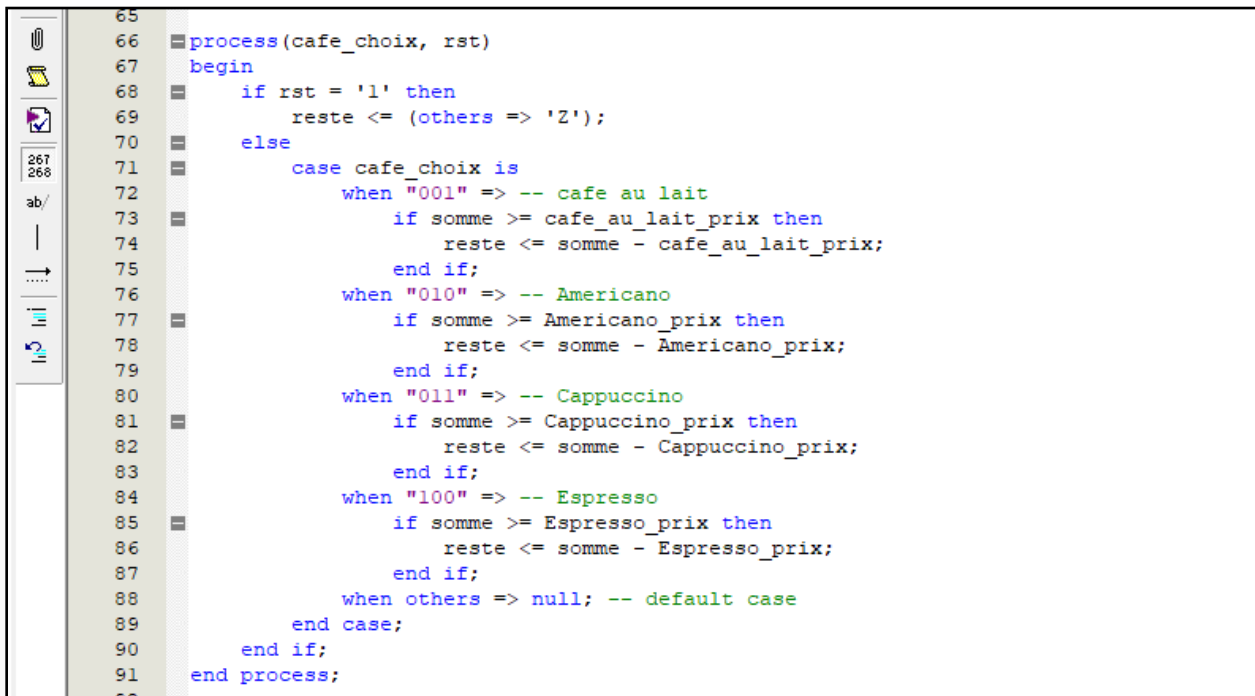
- Si le signal de réinitialisation (`rst`) est à l'état logique '1', le choix du café (`caffe_choix`) est remis à zéro (tous les bits à '0').
- 

### 3. Mise à Jour du Choix du Café :



- Si le signal de réinitialisation (`rst`) est à l'état logique '0', le processus met à jour la variable interne `cafe_choix` avec la valeur actuelle du signal `cafe_type`.
- Ainsi, lorsque l'utilisateur effectue un nouveau choix de café, la variable `cafe_choix` est mise à jour pour refléter ce choix.

### C. Processus de Gestion de la Monnaie (Troisième Processus)



```

65
66 process(cafe_choix, rst)
67 begin
68     if rst = '1' then
69         reste <= (others => 'Z');
70     else
71         case cafe_choix is
72             when "001" => -- café au lait
73                 if somme >= cafe_au_lait_prix then
74                     reste <= somme - cafe_au_lait_prix;
75                 end if;
76             when "010" => -- Americano
77                 if somme >= Americano_prix then
78                     reste <= somme - Americano_prix;
79                 end if;
80             when "011" => -- Cappuccino
81                 if somme >= Cappuccino_prix then
82                     reste <= somme - Cappuccino_prix;
83                 end if;
84             when "100" => -- Espresso
85                 if somme >= Espresso_prix then
86                     reste <= somme - Espresso_prix;
87                 end if;
88             when others => null; -- default case
89         end case;
90     end if;
91 end process;
92

```

Figure 9: Processus de Gestion de la Monnaie (Troisième Processus)

#### 1. Sensibilité aux Signaux :

- ✚ Ce processus est déclenché chaque fois qu'il y a un changement d'état sur le signal de réinitialisation (`rst`).
- ✚ Il est également déclenché en cas de changement d'état sur le signal `cafe_choix`.

#### 2. Réinitialisation (Reset) :

- Si le signal de réinitialisation (`rst`) est à l'état logique '1', la variable interne `reste` est initialisée à 'Z'.

### 3. Calcul du Reste en Fonction du Choix du Café :

- ✚ Si le signal de réinitialisation (`rst`) est à l'état logique '0', le processus utilise une structure `case` pour déterminer le type de café choisi (`cafe_choix`).
- ✚ En fonction du type de café choisi, le processus vérifie si la somme totale (`somme`) est suffisante pour acheter le café en comparant avec le prix du café correspondant (`cafe_au_lait_prix`, `Americano_prix`, `Cappuccino_prix`, `Espresso_prix`).
- ✚ Si la somme est suffisante, le reste est calculé en soustrayant le prix du café de la somme totale. Sinon, la variable `reste` reste indéterminée ('Z').
- ✚ Un cas par défaut (`others => null`) est inclus, mais il n'effectue aucune action spécifique

### D. Processus de Gestion de la Monnaie (Quatrième Processus)

```
92
93 process(reste, rst)
94 begin
95     if rst = '1' then
96         voy_monnaie_insuff <= '0';
97         voy_cafe_delivre <= '0';
98     else
99         if somme > "00000000" and cafe_choix /= "000" then
100             if reste = "ZZZZZZZZ" then
101                 voy_monnaie_insuff <= '1';
102             else
103                 voy_cafe_delivre <= '1';
104                 voy_monnaie_insuff <= '0';
105             end if;
106         end if;
107     end if;
108 end process;
109
110 --
```

*Figure 10: Processus de Gestion de la Monnaie (Troisième Processus)*

#### 1. Sensibilité aux Signaux :

- Ce processus est déclenché chaque fois qu'il y a un changement d'état sur le signal de réinitialisation (`rst`).
- Il est également déclenché en cas de changement d'état sur le signal `reste`.

## 2. Réinitialisation (Reset) :

- Si le signal de réinitialisation (`rst`) est à l'état logique '1', les signaux de sortie `voy_monnaie_insuff` et `voy_cafe_delivre` sont remis à zéro.

## 3. Génération des Signaux de Sortie :

- Si le signal de réinitialisation (`rst`) est à l'état logique '0', le processus examine la somme totale (`somme`), le choix du café (`cafe_choix`), et le reste après l'achat (`reste`).
- Si la somme totale est supérieure à zéro et que le choix du café n'est pas nul, le processus vérifie le contenu de la variable `reste`.
  - Si `reste` est égal à "ZZZZZZZZ", cela signifie que la monnaie insérée n'est pas suffisante pour acheter le café choisi, et le signal `voy_monnaie_insuff` est activé ('1').
  - Sinon, le café a été délivré avec succès, et le signal `voy_cafe_delivre` est activé ('1'). Le signal `voy_monnaie_insuff` est également désactivé ('0').
- Si la somme totale est nulle ou que le choix du café est nul, les signaux de sortie sont remis à zéro, indiquant l'absence d'opération significative.

## E. Processus de Gestion de la Monnaie (Cinquième Processus)

```
--|
process(somme, reste, rst)
begin
    if rst = '1' then
        somme_monnaie <= (others => '0');
        reste_monnaie <= (others => '0');
    else
        somme_monnaie <= somme;
        reste_monnaie <= reste;
    end if;
end process;

end gestion_monnaie;
```

Figure 11: Processus de Gestion de la Monnaie (Cinquième Processus)

### **1. Sensibilité aux Signaux :**

- Ce processus est déclenché chaque fois qu'il y a un changement d'état sur le signal de réinitialisation (`rst`).
- Il est également déclenché en cas de changement d'état sur les signaux `somme` et `reste`.

### **2. Réinitialisation (Reset) :**

- Si le signal de réinitialisation (`rst`) est à l'état logique '1', les variables de sortie `somme_monnaie` et `reste_monnaie` sont remises à zéro (tous les bits à '0').

### **3. Mise à Jour des Sorties Monétaires :**

- Si le signal de réinitialisation (`rst`) est à l'état logique '0', le processus met à jour les sorties monétaires `somme_monnaie` et `reste_monnaie` en fonction des variables internes `somme` et `reste`.
- Ces sorties sont mises à jour avec les valeurs actuelles de `somme` et `reste`

## CONCLUSION

En conclusion, le projet "Distributeur de Café avec Gestion de la Monnaie" offre une solution interactive et efficace grâce à l'utilisation de VHDL-FPGA et de l'environnement Quartus. En permettant aux utilisateurs de choisir leur café en insérant des pièces de monnaie, le système garantit une gestion précise des transactions monétaires. Les fonctionnalités clés, telles que la sélection du café, la reconnaissance des pièces, l'affichage de l'état du distributeur et la délivrance du café, sont soigneusement intégrées pour offrir une expérience utilisateur complète. L'utilisation de Quartus et VHDL témoigne de l'efficacité dans la conception et la mise en œuvre des aspects de contrôle, de calcul et d'affichage, faisant de ce simulateur de distributeur de café une solution bien conçue et fonctionnelle. Ce projet démontre l'application réussie de la technologie FPGA pour répondre aux besoins d'interactivité et de gestion monétaire dans un environnement de distribution de café simulé.