

PROJET VHDL

CHIFFREMENT ET DÉCHIFFREMENT DES MESSAGES TEXTE PAR DÉCALAGE

Rédigé par

El Badri Imane

Hajar Bentabet

Jouichat Salma

Encadré par

Mr. EL Moumni



I. Introduction :

VHDL est un langage de description de matériel destiné à représenter le comportement ainsi que l'architecture d'un système électronique numérique. Son nom complet est **VHSIC (Hardware Description Language)**.

L'intérêt d'une telle description réside dans son caractère exécutable : une spécification décrite en VHDL peut être vérifiée par simulation, avant que la conception détaillée ne soit terminée. En outre, les outils de conception assistée par ordinateur permettant de passer directement d'une description fonctionnelle en VHDL à un schéma en porte logique ont révolutionné les méthodes de conception des circuits numériques, [ASIC](#) ou [FPGA](#).

II. Vue d'ensemble du projet :

Notre projet se concentre sur le chiffrement et le déchiffrement par décalage de César, une technique classique de cryptographie symétrique. En utilisant VHDL, nous développons des circuits électroniques capables de mettre en œuvre ces opérations de manière efficace et sécurisée. L'approche modulaire du VHDL nous permet d'implémenter les étapes spécifiques de l'algorithme, offrant ainsi une solution adaptable et optimisée pour le traitement des messages texte.

Description du Principe de l'Algorithme de César :

L'algorithme de chiffrement de César, également connu sous le nom de chiffrement par décalage, est l'une des méthodes de cryptographie les plus anciennes et les plus simples. Il repose sur le décalage des lettres de l'alphabet d'un nombre fixe de positions. Pour chiffrer un texte, chaque lettre du texte clair est remplacée par la lettre située un certain nombre de positions plus loin dans l'alphabet. Le même décalage est utilisé pour toutes les lettres du message.

Étapes de l'Algorithme :

Initialisation : Définir la clé de chiffrement, qui est le nombre de positions de décalage. Par exemple, une clé de 3 signifie que chaque lettre sera remplacée par la lettre située trois positions plus loin dans l'alphabet.

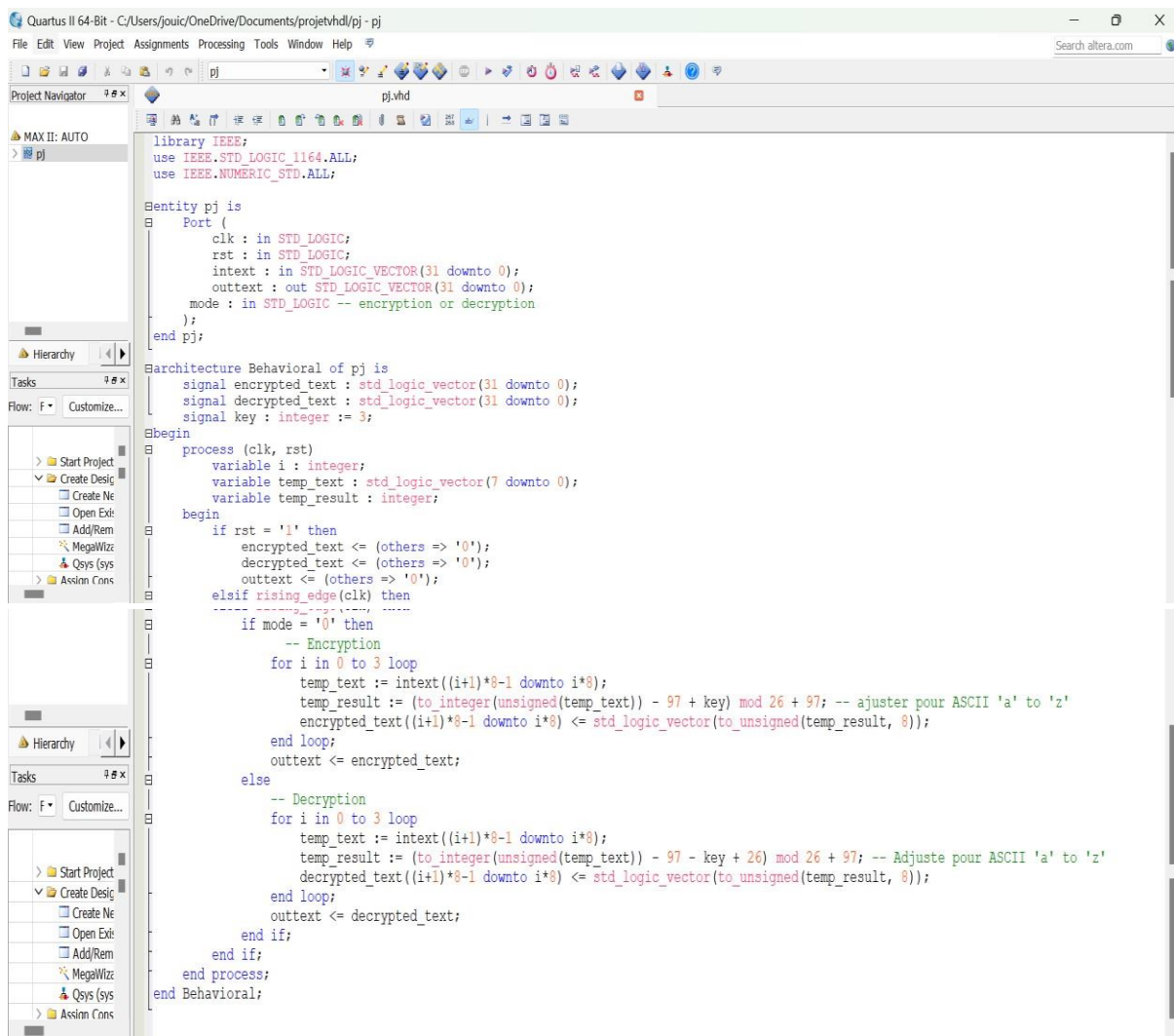
Chiffrement : Pour chaque caractère du texte clair : Convertir le caractère en sa valeur ASCII. Appliquer le décalage en ajoutant la clé à la valeur ASCII. Si le résultat dépasse la valeur ASCII de 'Z' (pour les majuscules) ou 'z' (pour les minuscules), revenir au début de l'alphabet en utilisant l'arithmétique modulaire. Convertir la nouvelle valeur ASCII en caractère.

Déchiffrement : Pour chaque caractère du texte chiffré : Convertir le caractère en sa valeur ASCII. Appliquer le décalage inverse en soustrayant la clé de la valeur ASCII. Si le résultat

est inférieur à la valeur ASCII de 'A' (pour les majuscules) ou 'a' (pour les minuscules), revenir à la fin de l'alphabet en utilisant l'arithmétique modulaire. Convertir la nouvelle valeur ASCII en caractère.

Gestion des Limites : Utiliser l'arithmétique modulaire pour s'assurer que les opérations de décalage restent dans les limites des caractères valides (de 'A' à 'Z' et de 'a' à 'z').

III. Algorithmme :



```
Quartus II 64-Bit - C:/Users/jouic/OneDrive/Documents/projetvhdl/pj - pj
File Edit View Project Assignments Processing Tools Window Help
Search altera.com

Project Navigator
MAX II: AUTO
> pj
pj.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

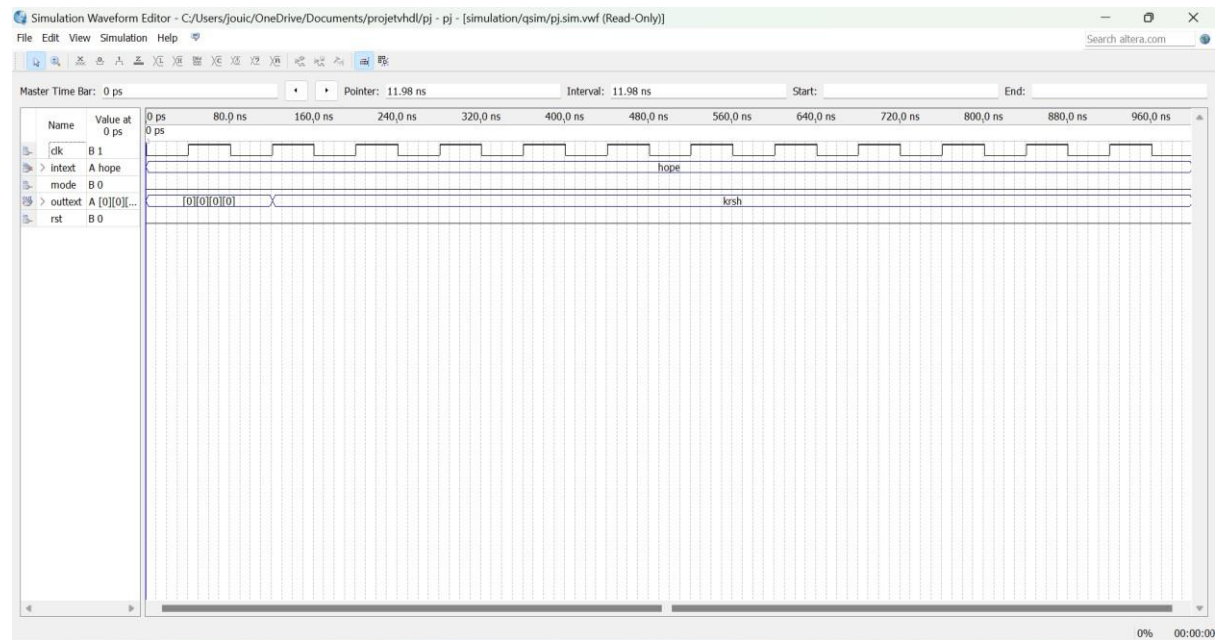
entity pj is
    Port (
        clk : in STD_LOGIC;
        rst : in STD_LOGIC;
        intext : in STD_LOGIC_VECTOR(31 downto 0);
        outtext : out STD_LOGIC_VECTOR(31 downto 0);
        mode : in STD_LOGIC -- encryption or decryption
    );
end pj;

Architecture Behavioral of pj is
    signal encrypted_text : std_logic_vector(31 downto 0);
    signal decrypted_text : std_logic_vector(31 downto 0);
    signal key : integer := 3;

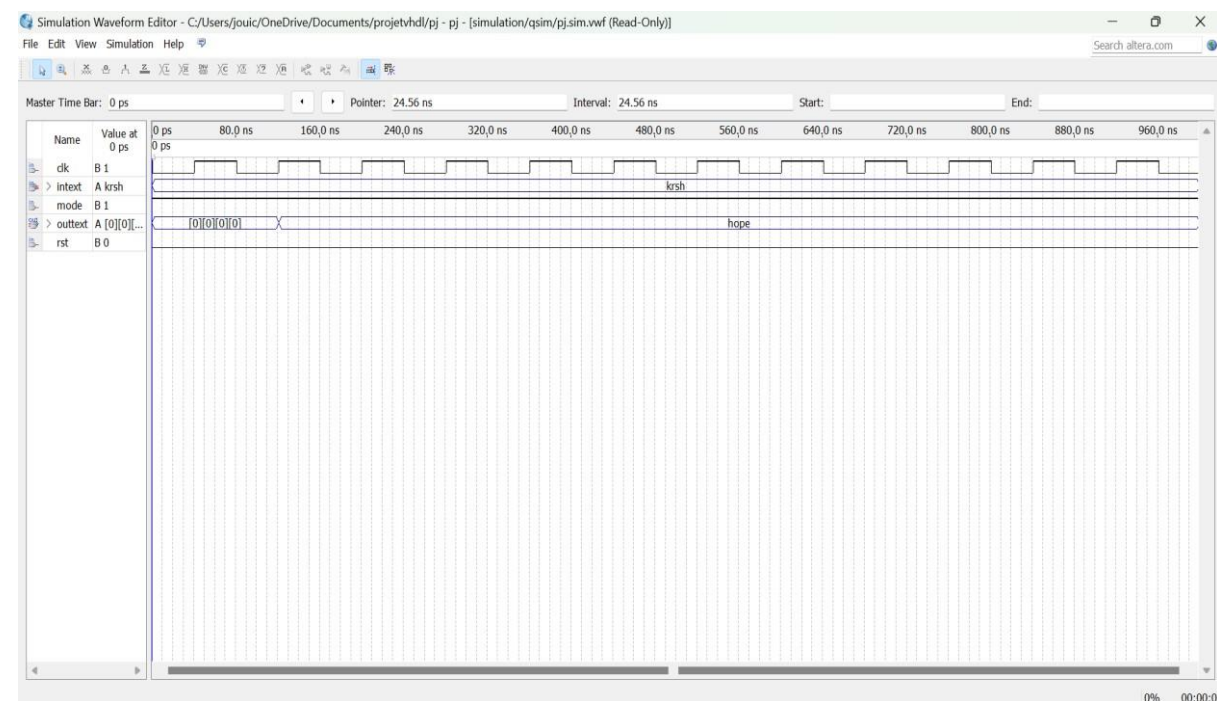
begin
    process (clk, rst)
        variable i : integer;
        variable temp_text : std_logic_vector(7 downto 0);
        variable temp_result : integer;
    begin
        if rst = '1' then
            encrypted_text <= (others => '0');
            decrypted_text <= (others => '0');
            outtext <= (others => '0');
        elsif rising_edge(clk) then
            if mode = '0' then
                -- Encryption
                for i in 0 to 3 loop
                    temp_text := intext((i+1)*8-1 downto i*8);
                    temp_result := (to_integer(unsigned(temp_text)) - 97 + key) mod 26 + 97; -- ajuster pour ASCII 'a' to 'z'
                    encrypted_text((i+1)*8-1 downto i*8) <= std_logic_vector(to_unsigned(temp_result, 8));
                end loop;
                outtext <= encrypted_text;
            else
                -- Decryption
                for i in 0 to 3 loop
                    temp_text := intext((i+1)*8-1 downto i*8);
                    temp_result := (to_integer(unsigned(temp_text)) - 97 - key + 26) mod 26 + 97; -- Adjuste pour ASCII 'a' to 'z'
                    decrypted_text((i+1)*8-1 downto i*8) <= std_logic_vector(to_unsigned(temp_result, 8));
                end loop;
                outtext <= decrypted_text;
            end if;
        end if;
    end process;
end Behavioral;
```

IV. Stimulation :

Encryption mode



Decryption mode



V. Conclusion :

En conclusion, notre projet de conception de circuits pour le chiffrement et le déchiffrement par décalage, réalisé en utilisant le langage VHDL, démontre l'efficacité et la flexibilité de cette approche dans le domaine de la cryptographie. En combinant des concepts de cryptographie classique avec des techniques de conception matérielles modernes, nous avons pu créer des circuits électroniques capables de garantir la confidentialité des données de manière rapide et sécurisée. Cette expérience illustre le potentiel du VHDL dans la création de systèmes électroniques sophistiqués répondant aux exigences croissantes en matière de sécurité informatique.