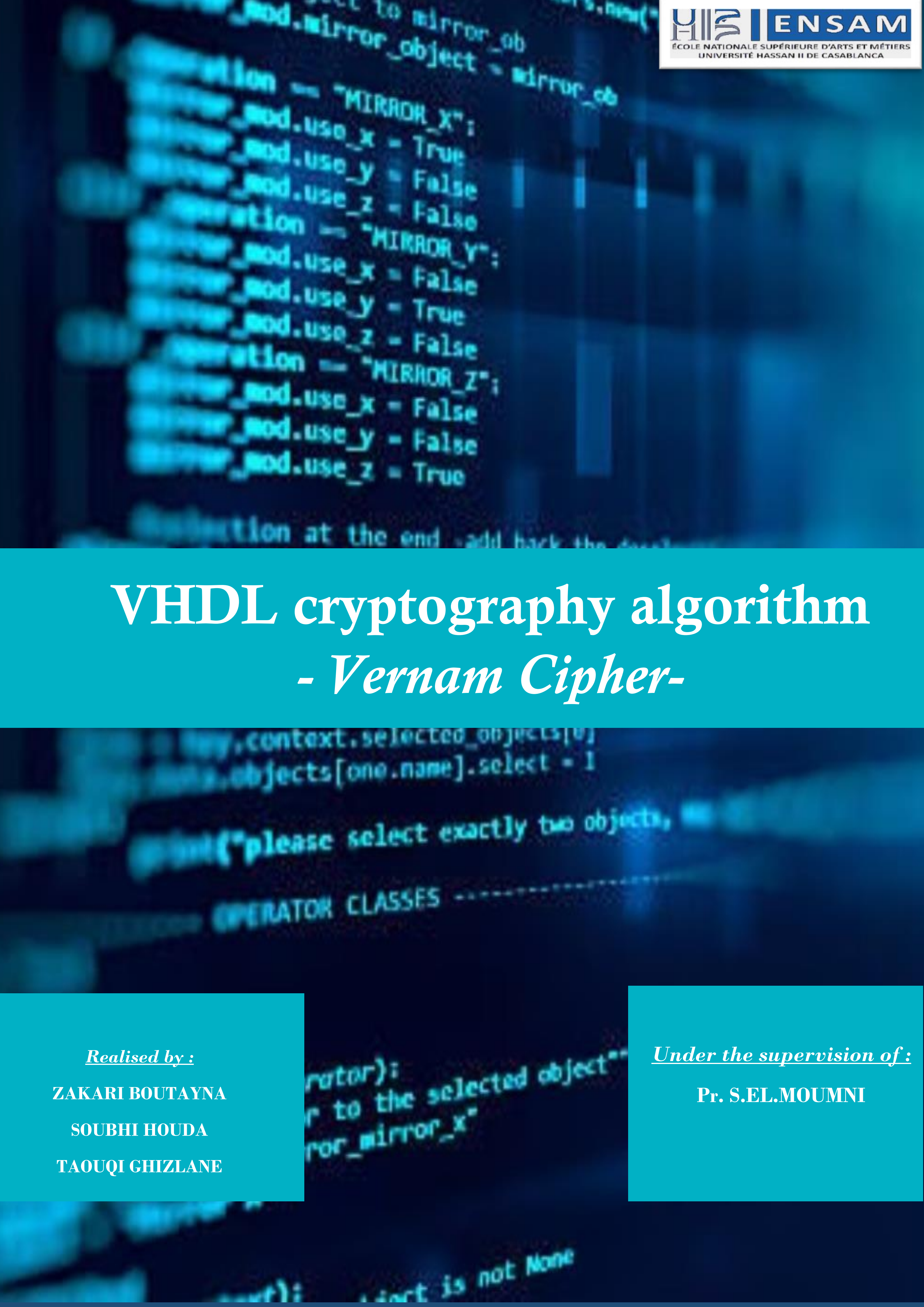


ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

VHDL cryptography algorithm

- Vernam Cipher-

Realised by :

ZAKARI BOUTAYNA

SOUBHI HOUDA

TAOUQI GHIZLANE

Under the supervision of :

Pr. S.EL.MOUMNI

Introduction to Vernam Cipher

The Vernam Cipher, also known as one-time pad, is a highly secure encryption technique that uses a random key as long as the message itself. This ensures perfect secrecy, as the encrypted message is completely indistinguishable from random noise without the proper key.

Definition and Principles

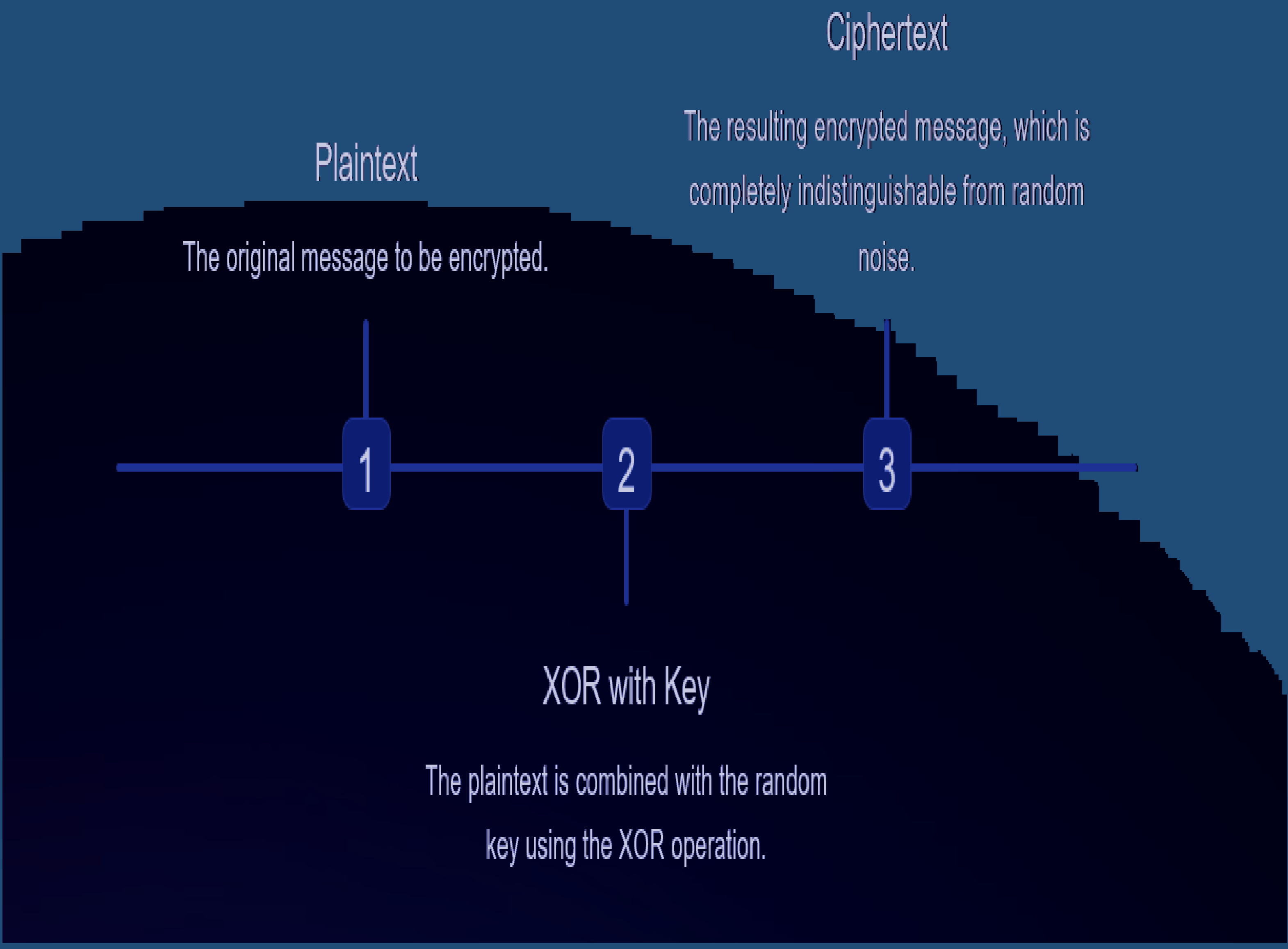
Definition

The Vernam Cipher is a symmetric-key cryptographic technique where the plaintext is combined with a random key of the same length using the XOR operation

Principles

The key must be truly random, as long as the message, and used only once. This ensures the encrypted message is completely unbreakable.

Encryption and Decryption Process



Plaintext

The original message to be encrypted.

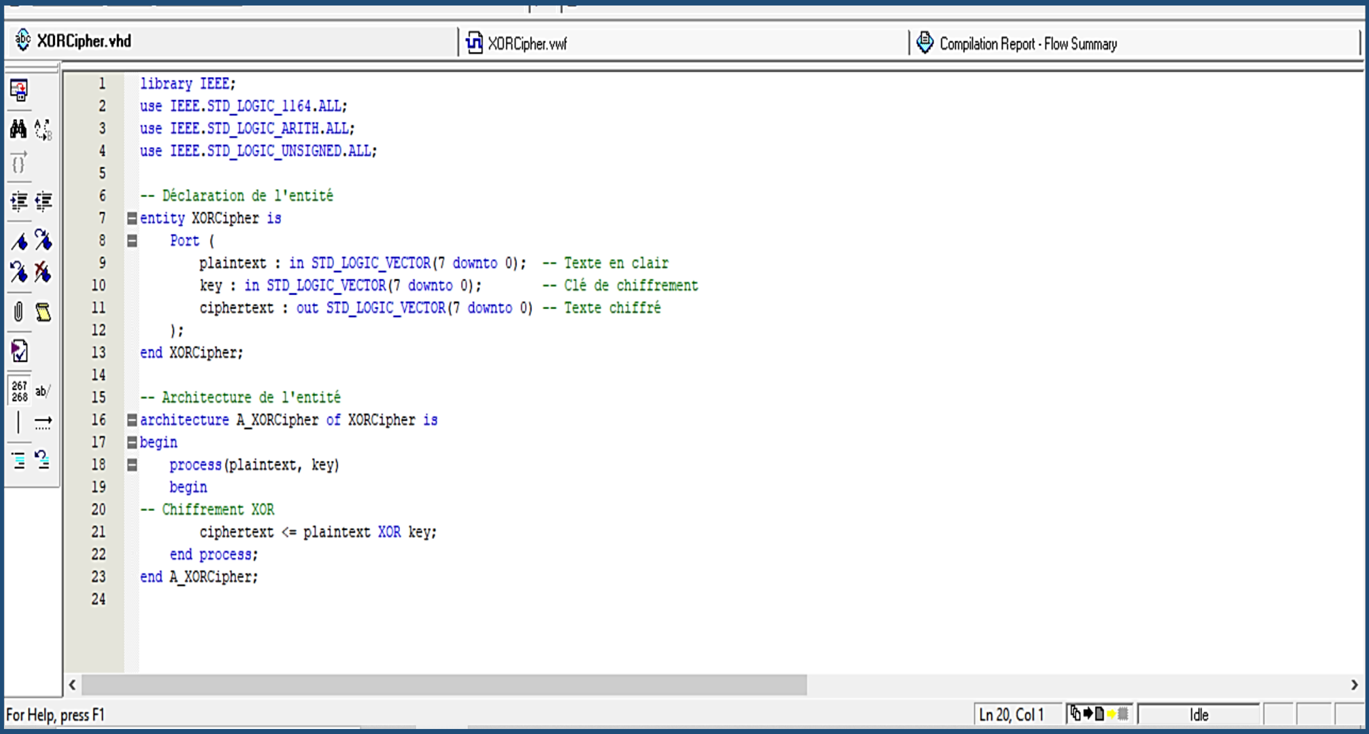
XOR with Key

The plaintext is combined with the random key using the XOR operation.

Ciphertext

The resulting encrypted message, which is completely indistinguishable from random noise.

Code VHDL

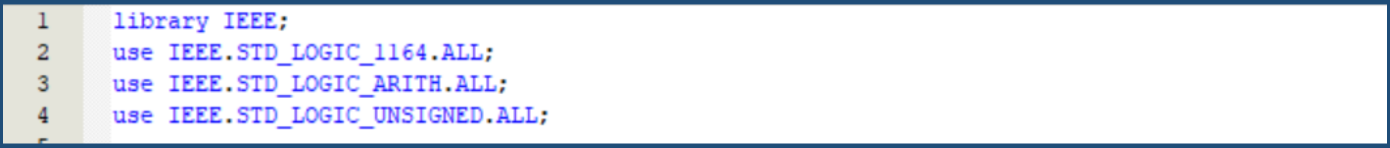


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Déclaration de l'entité
entity XORCipher is
    Port (
        plaintext : in STD_LOGIC_VECTOR(7 downto 0);  -- Texte en clair
        key : in STD_LOGIC_VECTOR(7 downto 0);        -- Clé de chiffrement
        ciphertext : out STD_LOGIC_VECTOR(7 downto 0) -- Texte chiffré
    );
end XORCipher;

-- Architecture de l'entité
architecture A_XORCipher of XORCipher is
begin
    process(plaintext, key)
    begin
-- Chiffrement XOR
        ciphertext <= plaintext XOR key;
    end process;
end A_XORCipher;
```

Bibliothèques

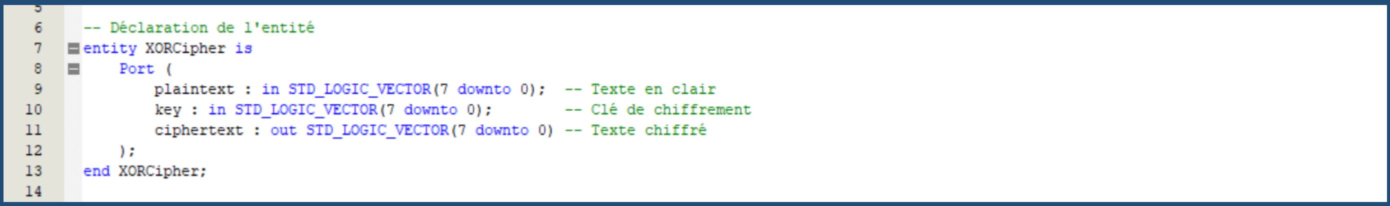


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

Explication :

- **library IEEE;** : Inclut la bibliothèque IEEE qui est standard pour la conception en VHDL.
- **use IEEE.STD_LOGIC_1164.ALL;** : Utilise le package STD_LOGIC_1164 qui définit les types de données standard **STD_LOGIC** et **STD_LOGIC_VECTOR**.
- **use IEEE.STD_LOGIC_ARITH.ALL;** : Utilise le package STD_LOGIC_ARITH pour les opérations arithmétiques sur les vecteurs logiques (bien que dans ce cas, il n'est pas vraiment nécessaire car on n'utilise pas d'opérations arithmétiques complexes).
- **use IEEE.STD_LOGIC_UNSIGNED.ALL;** : Utilise le package STD_LOGIC_UNSIGNED pour les opérations sur les vecteurs logiques traités comme des nombres non signés.

Entity Declaration

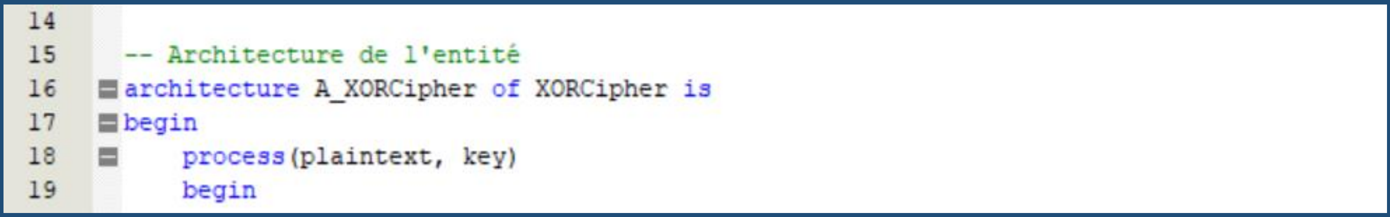


```vhdl
-- Déclaration de l'entité
entity XORCipher is
    Port (
        plaintext : in STD_LOGIC_VECTOR(7 downto 0);  -- Texte en clair
        key : in STD_LOGIC_VECTOR(7 downto 0);        -- Clé de chiffrement
        ciphertext : out STD_LOGIC_VECTOR(7 downto 0) -- Texte chiffré
    );
end XORCipher;
```

❖ **Explanation:**

- **entity XORCipher is: Declares the entity named XORCipher. An entity in VHDL defines the interface of a module.**
- **Port: Defines the input and output ports of the entity.**
- **plaintext : in STD_LOGIC_VECTOR(7 downto 0);: Declares an input port plaintext of type STD_LOGIC_VECTOR with 8 bits (bit vector ranging from 7 to 0).**
- **key : in STD_LOGIC_VECTOR(7 downto 0);: Declares an input port key of type STD_LOGIC_VECTOR with 8 bits.**
- **ciphertext : out STD_LOGIC_VECTOR(7 downto 0): Declares an output port ciphertext of type STD_LOGIC_VECTOR with 8 bits.**

Entity Architecture



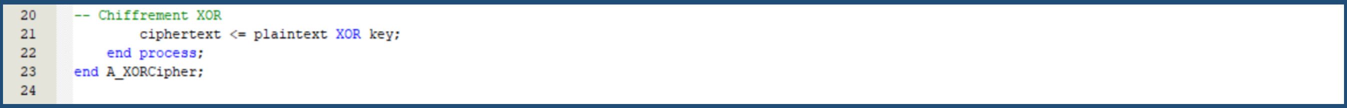
```vhdl

-- Architecture de l'entité
architecture A_XORCipher of XORCipher is
begin
    process(plaintext, key)
    begin
```

❖ **Explanation**:

- **architecture A_XORCipher of XORCipher is:** Declares the architecture named A_XORCipher associated with the entity XORCipher. The architecture describes the internal behavior of the entity.
- **begin**: Begins the body of the architecture.
- **process(plaintext, key):** Declares a process that is sensitive to the signals plaintext and key. Each time one of these signals changes, the process is executed.
- **begin:** Begins the body of the process.
- **ciphertext <= plaintext XOR key;:** Performs the XOR operation between plaintext and key, and assigns the result to ciphertext.
- **end process;:** Ends the process.
- **end A_XORCipher;:** Ends the architecture.

Opération XOR : ciphertext <= plaintext XOR key :



```
-- Chiffrement XOR
        ciphertext <= plaintext XOR key;
    end process;
end A_XORCipher;
```

❖ **Explanation:**

- **ciphertext: This is the output port of the XORCipher entity. It stores the result of the encryption operation.**
- **<=: This is the assignment operator in VHDL. It assigns the value of the expression on the right side of the operator to the signal on the left side.**
- **plaintext: This is an input port of the XORCipher entity. It contains the plaintext to be encrypted.**
- **XOR: This is the XOR (exclusive OR) logical operator in VHDL. This operator compares each bit of the two operands and produces a bit result that is 1 if and only if the compared bits are different; otherwise, the result is 0.**
- **key: This is an input port of the XORCipher entity. It contains the key used for encryption.**

How the XOR Operation Works :

The XOR operation is performed bit by bit between the plaintext and the key. For example, if the plaintext is 10101010 and the key is 11001100, the result (ciphertext) will be calculated as follows:

Plaintext Bit	**Key Bit**	**XOR Result**
1	1	0
0	1	1
1	0	1
0	0	0
1	1	0
0	1	1
1	0	1
0	0	0

Thus, 10101010 XOR 11001100 gives 01100110.

End of the Process: end process;

❖ ***Explanation:***

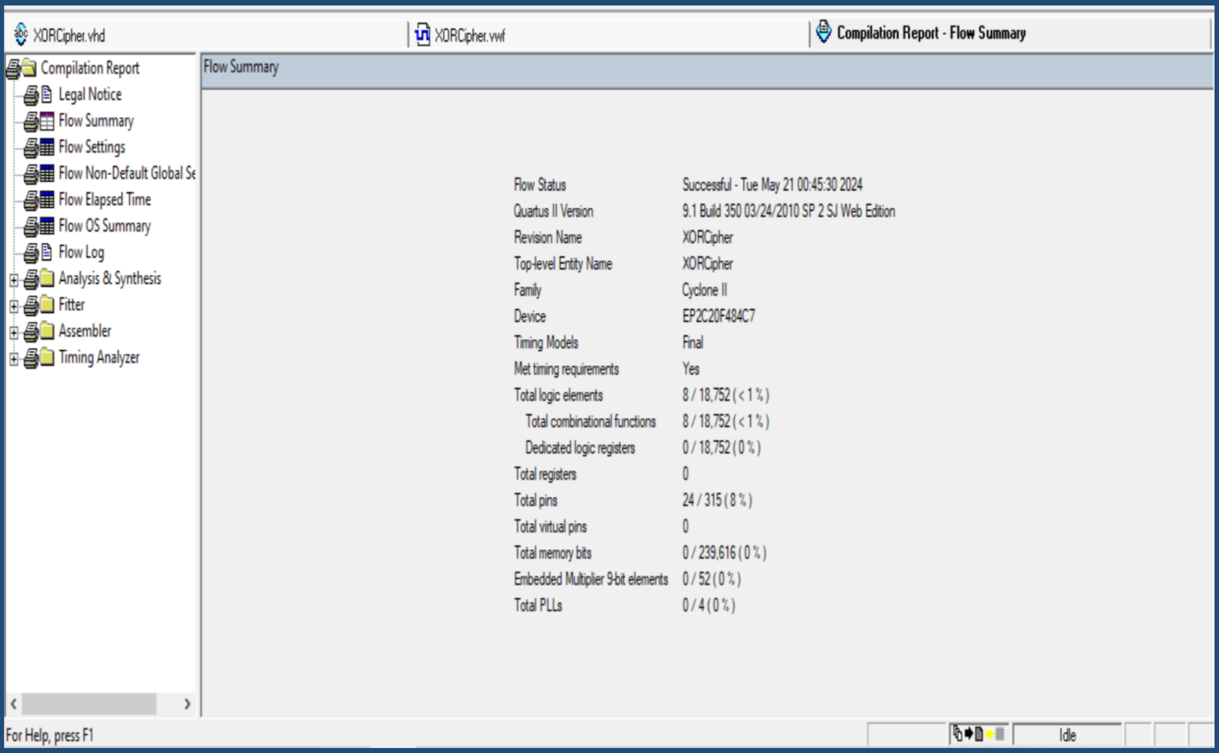
- ***This line marks the end of the process in VHDL. The process is a block of code that executes whenever there is a change in the signals it is sensitive to (in this case, plaintext and key).***

End of the Architecture: end A_XORCipher;

❖ ***Explanation:***

- ***This line marks the end of the architecture A_XORCipher. It concludes the behavioral description of the XORCipher entity.***

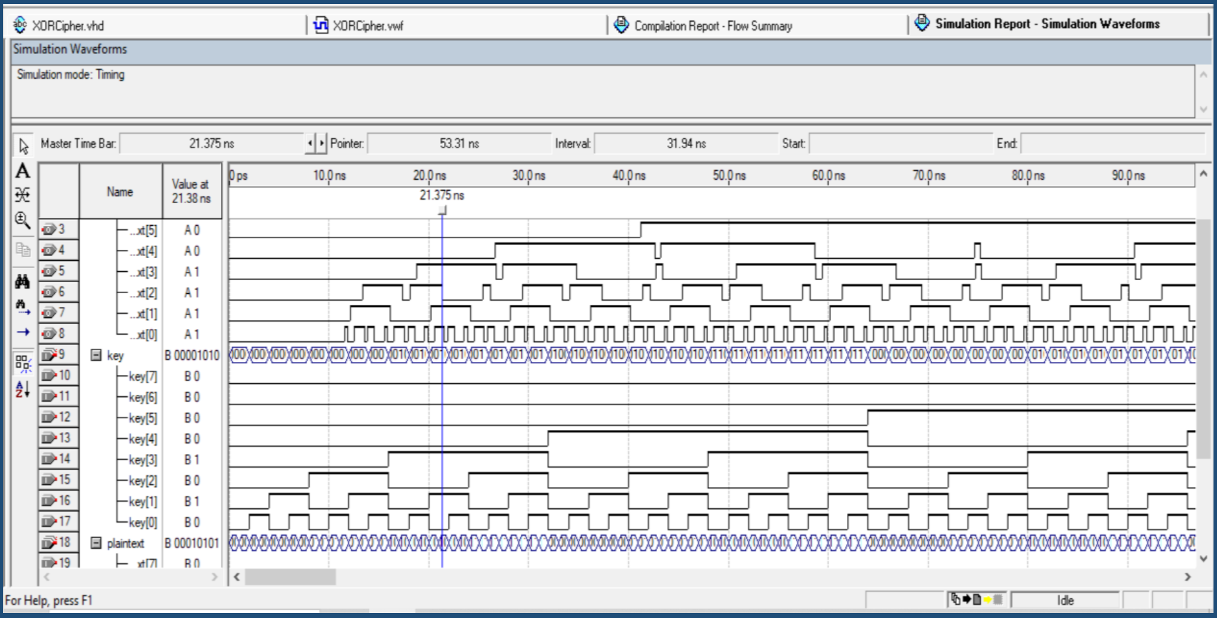
Flow Summary



Flow Summary

Flow Status	Successful - Tue May 21 00:45:30 2024
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	XORCipher
Top-level Entity Name	XORCipher
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	8 / 18,752 (< 1 %)
Total combinational functions	8 / 18,752 (< 1 %)
Dedicated logic registers	0 / 18,752 (0 %)
Total registers	0
Total pins	24 / 315 (8 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Stimulation Waveforms


XORCipher.vhd | XORCipher.vwf | Compilation Report - Flow Summary | **Simulation Report - Simulation Waveforms**

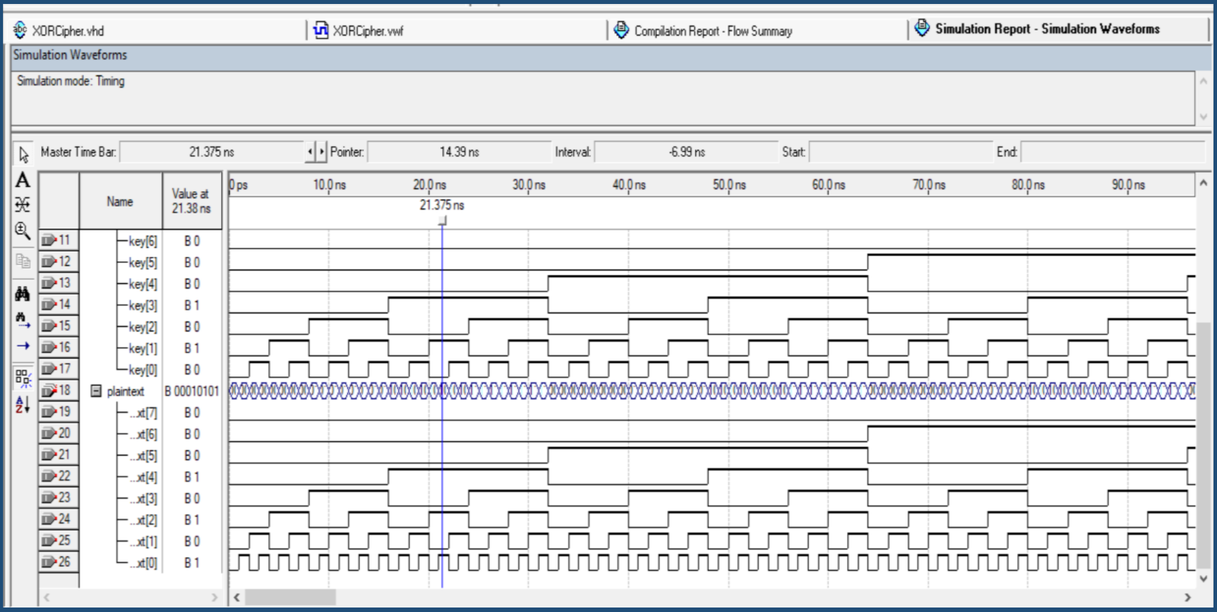
Simulation Waveforms

Simulation mode: Timing

Master Time Bar: 21.375 ns | Pointer: 53.31 ns | Interval: 31.94 ns | Start: | End:

	Name	Value at 21.38 ns
3	...xt[5]	A 0
4	...xt[4]	A 0
5	...xt[3]	A 1
6	...xt[2]	A 1
7	...xt[1]	A 1
8	...xt[0]	A 1
9	key	B 00001010
10	key[7]	B 0
11	key[6]	B 0
12	key[5]	B 0
13	key[4]	B 0
14	key[3]	B 1
15	key[2]	B 0
16	key[1]	B 1
17	key[0]	B 0
18	plaintext	B 00010101
19	...xt[7]	B 0

For Help, press F1 | Idle


XORCipher.vhd | XORCipher.vwf | Compilation Report - Flow Summary | **Simulation Report - Simulation Waveforms**

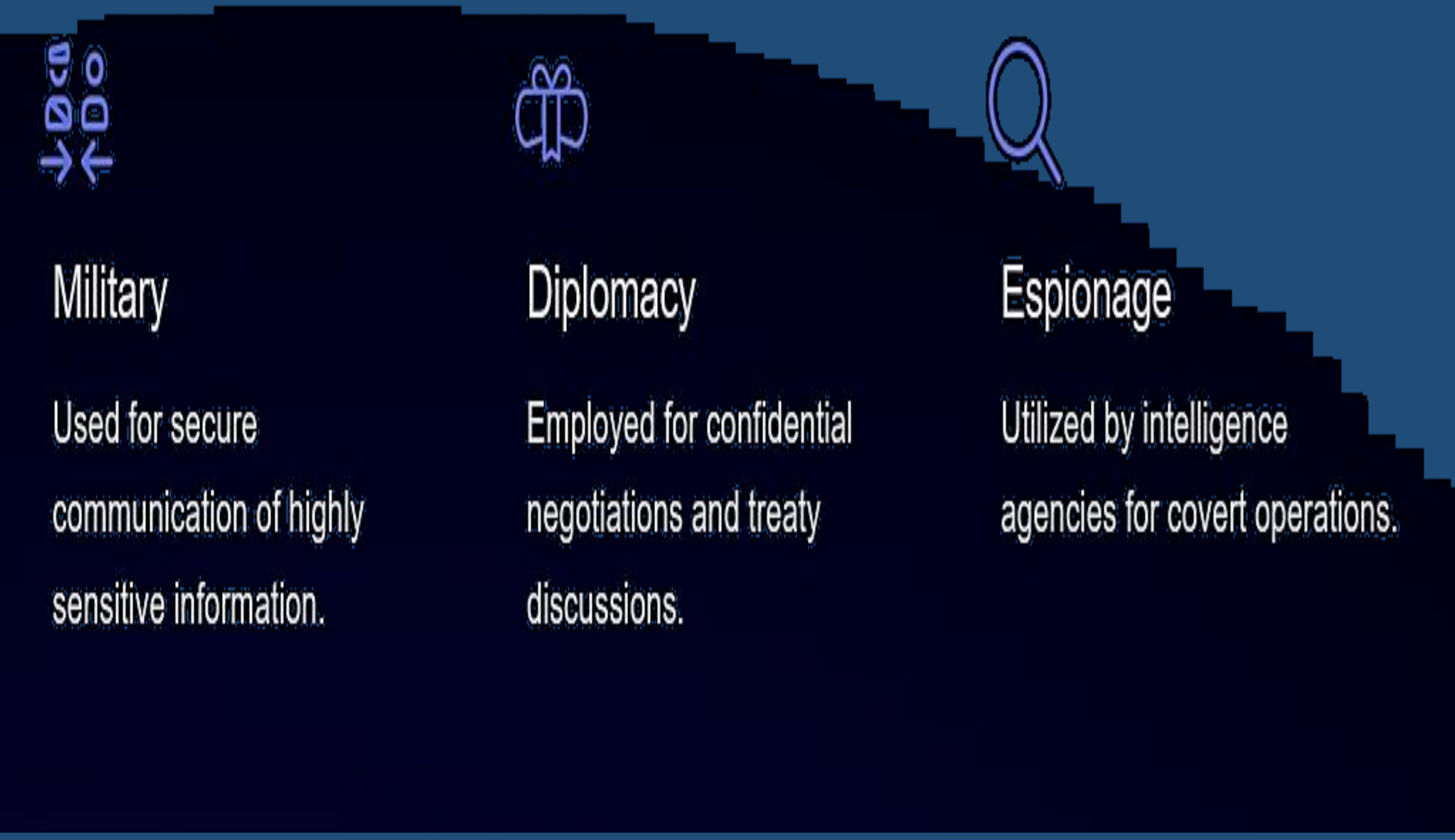
Simulation Waveforms

Simulation mode: Timing

Master Time Bar: 21.375 ns | Pointer: 14.39 ns | Interval: -6.99 ns | Start: | End:

	Name	Value at 21.38 ns
11	key[6]	B 0
12	key[5]	B 0
13	key[4]	B 0
14	key[3]	B 1
15	key[2]	B 0
16	key[1]	B 1
17	key[0]	B 0
18	plaintext	B 00010101
19	...xt[7]	B 0
20	...xt[6]	B 0
21	...xt[5]	B 0
22	...xt[4]	B 1
23	...xt[3]	B 0
24	...xt[2]	B 1
25	...xt[1]	B 0
26	...xt[0]	B 1

Applications & Use cases



Military

Used for secure communication of highly sensitive information.

Diplomacy

Employed for confidential negotiations and treaty discussions.

Espionage

Utilized by intelligence agencies for covert operations.

Conclusion & Future Considerations

1

Quantum Computing

The Vernam Cipher may be vulnerable to future quantum computing advances.

2

Post-Quantum Cryptography

Research is ongoing to develop new encryption techniques resistant to quantum attacks.

3

Continued Importance

The Vernam Cipher remains a crucial tool for the most sensitive communications.