2023/2024

Encadré par :

**Pr.EL MOUMNI**

Réalisé par :

**ARROUD Israe**

**ben el hamdia arij**

**moussafir salma**

Cryptography : DES decryption

**Introducrion :**

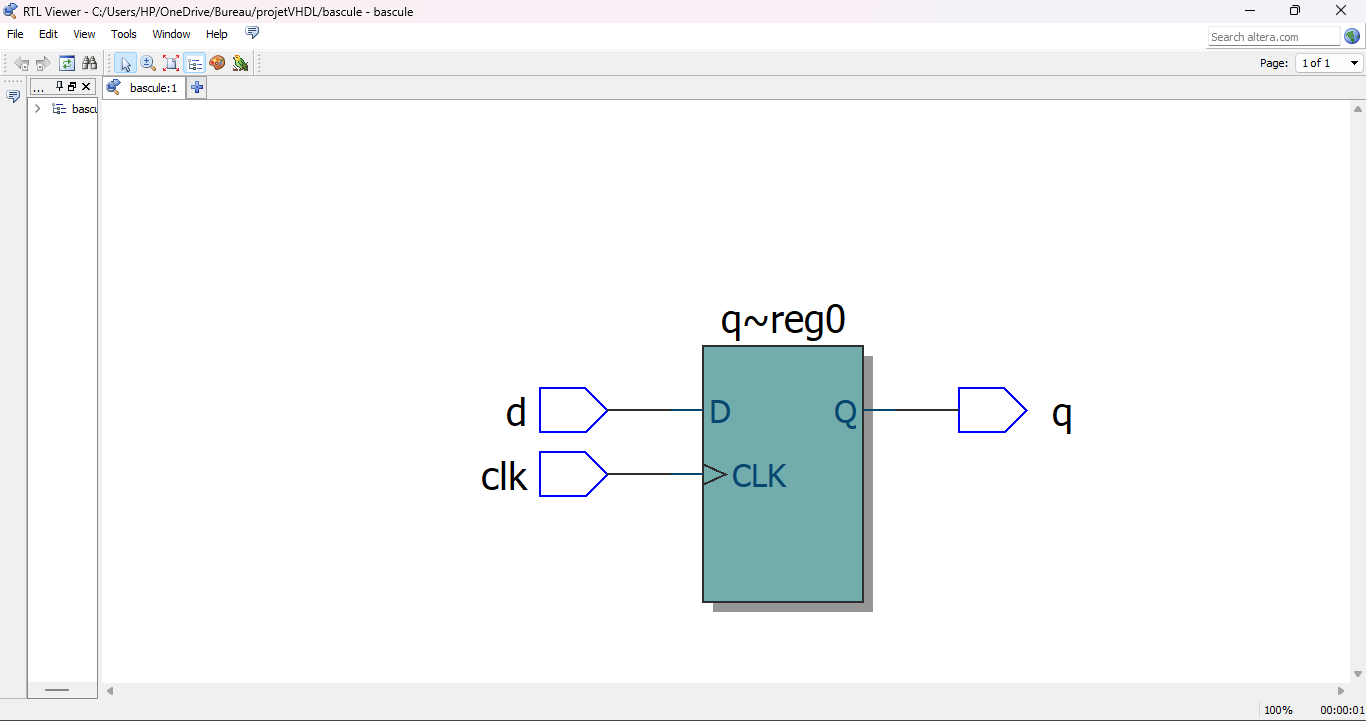
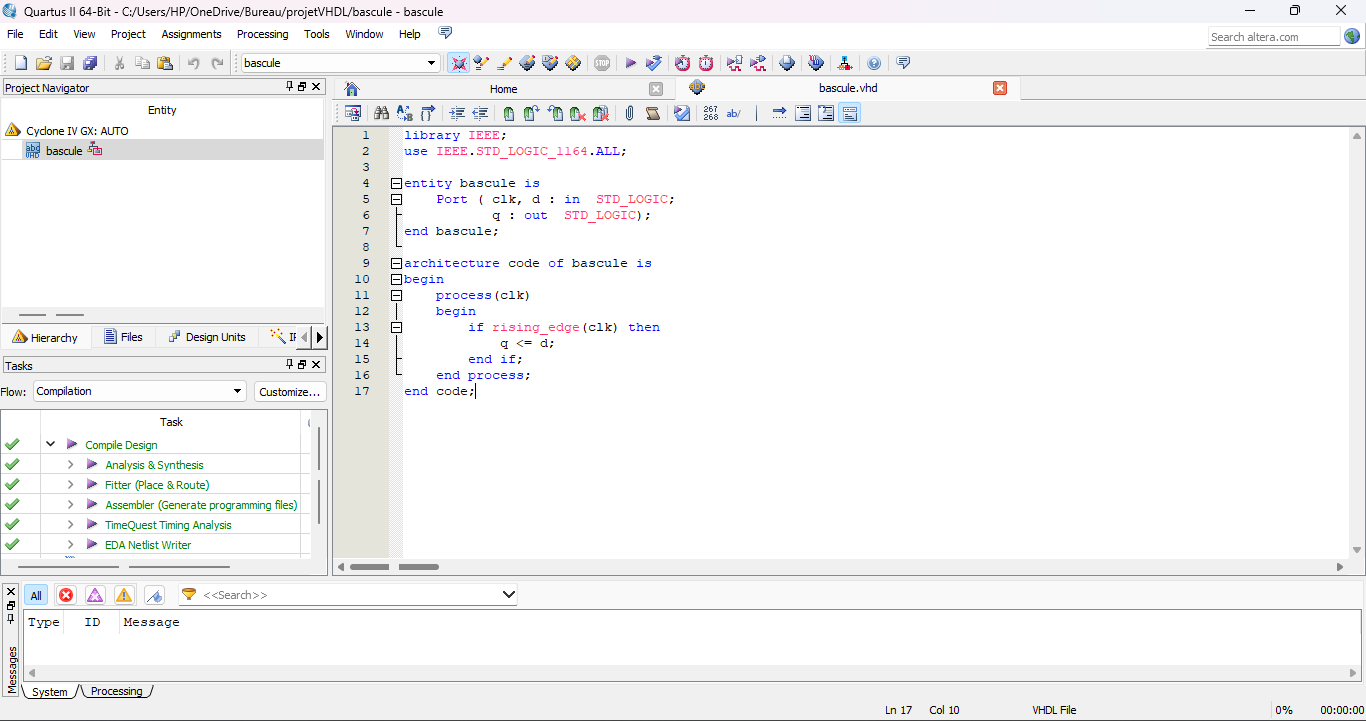
DES, or Data Encryption Standard, is a symmetric-key block cipher algorithm developed in the 1970s. It operates on 64-bit blocks of plaintext and uses a 56-bit key for encryption and decryption. DES consists of several key components including initial and final permutations, key generation, a Feistel network structure, and substitution-permutation (S-box) operations. While once widely used, DES is now considered relatively weak against modern cryptographic attacks due to its small key size. Nonetheless, implementing DES in VHDL provides valuable insights into both cryptography and hardware design.

**History :**

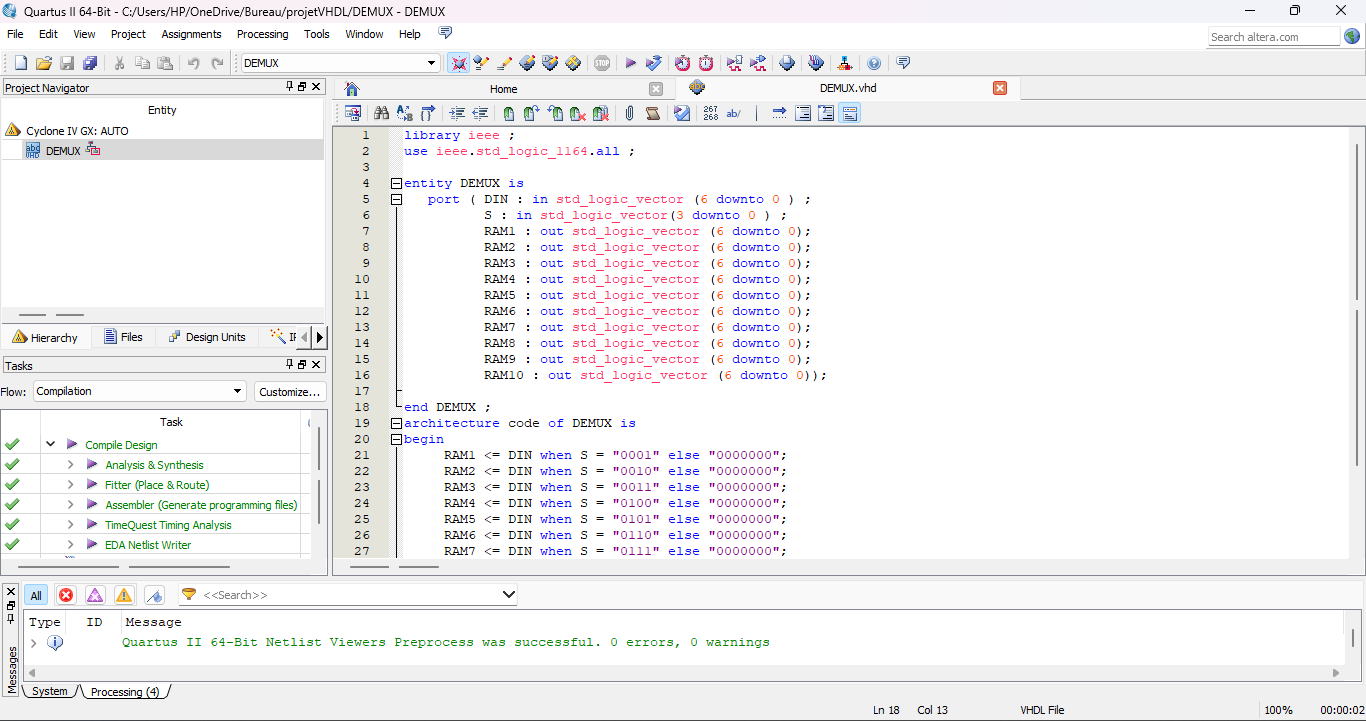
The Data Encryption Standard (DES) emerged in the 1970s as a pioneering effort to establish a standardized encryption algorithm. Developed by IBM engineers, with significant contributions from Horst Feistel, DES was adopted as a federal standard by the US government in 1977. Its introduction marked a pivotal moment in the history of cryptography, providing a widely accepted method for securing data transmission. However, as computing power advanced, DES's vulnerability to brute-force attacks became apparent due to its relatively small key size. This realization prompted the need for a more robust encryption standard, leading to the development of the Advanced Encryption Standard (AES) in the early 2000s. Despite its eventual deprecation, DES's legacy endures as a foundational milestone in the evolution of encryption technologies, shaping the landscape of modern cryptography and laying the groundwork for subsequent advancements.

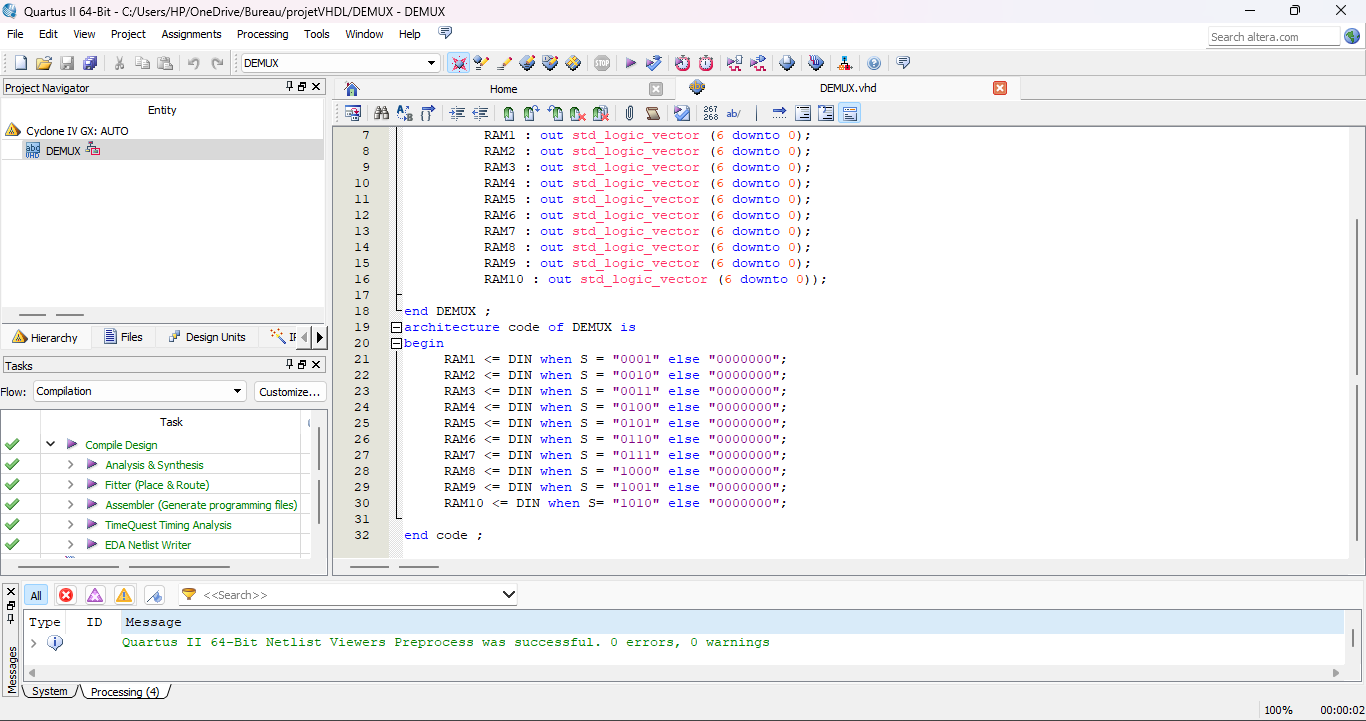
**The input code :**

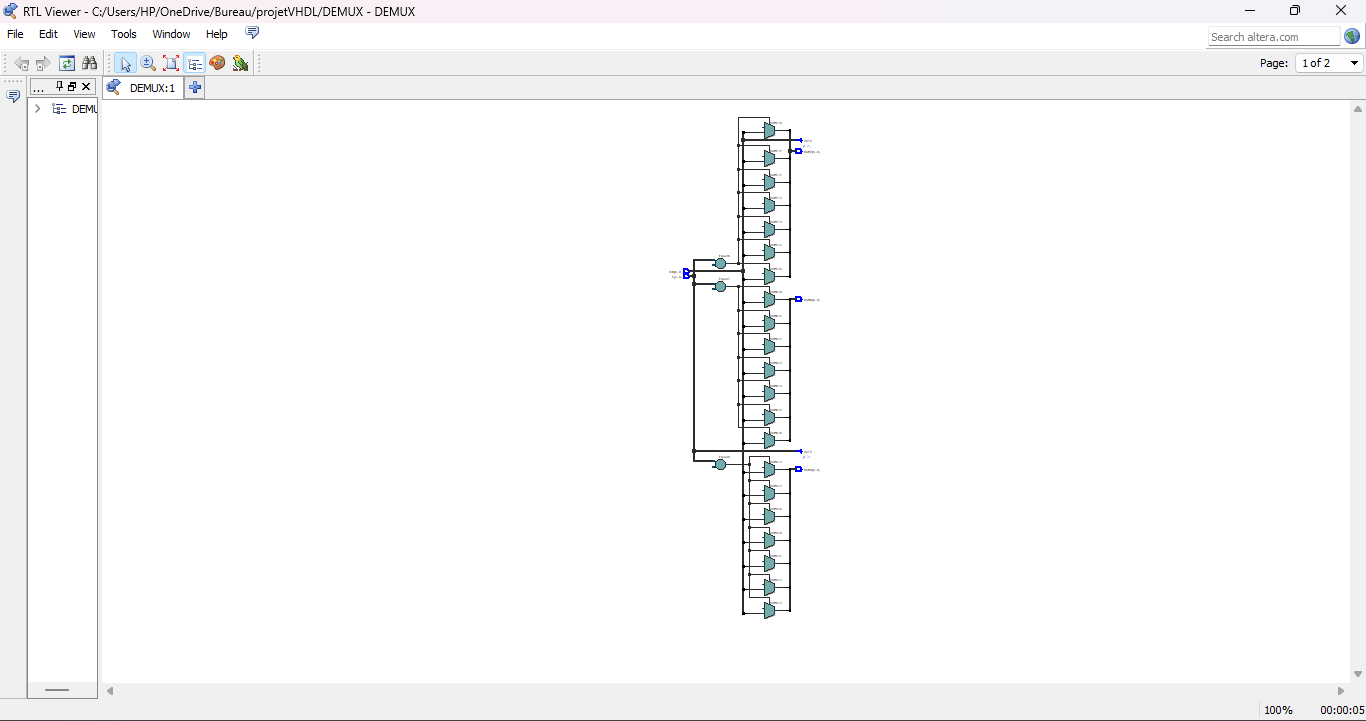
**Bascule D :** The D flip-flop is a fundamental building block in digital circuits used for storing binary data. It captures the input (d) and holds it until the next rising edge of the clock (clk), at which point it updates the output (q) with the input value.

In a DES implementation, D flip-flops might be used for storing intermediate values, holding data during computation, or synchronizing signals with the clock.

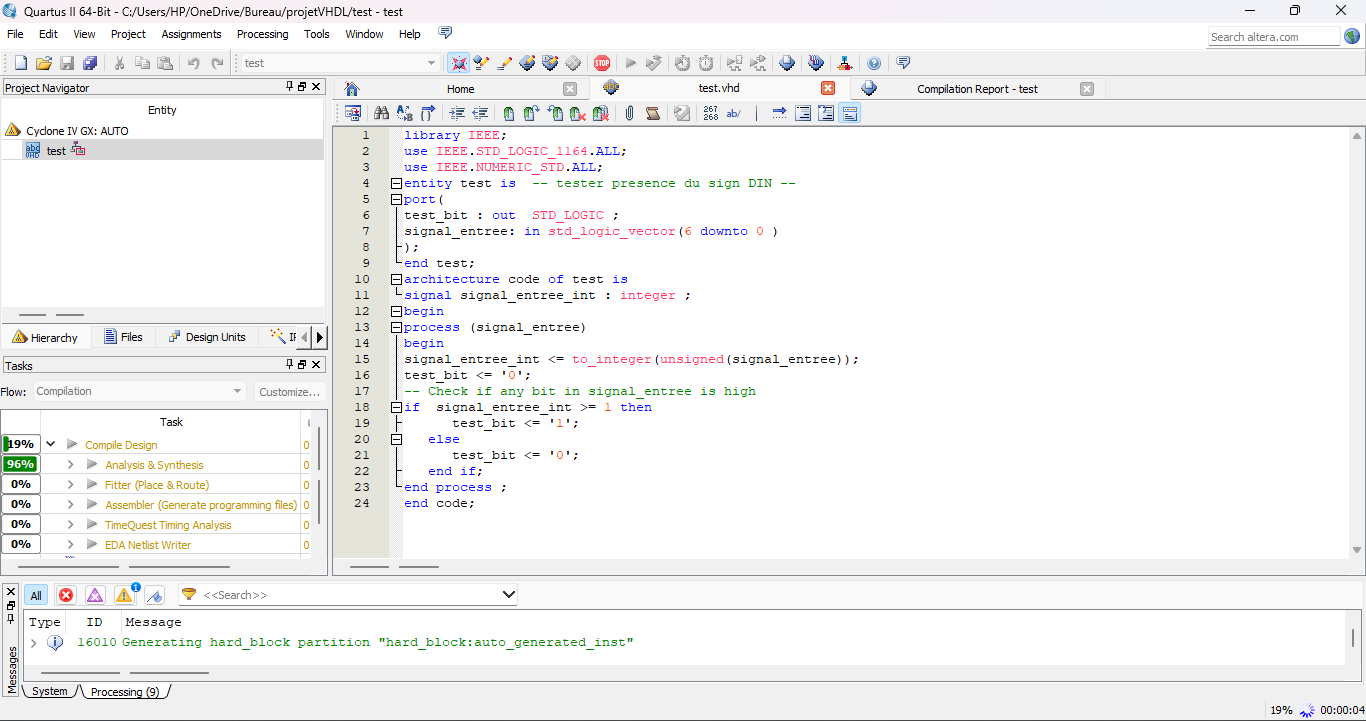
**DMUX 10-1 :** In DES, a demultiplexer could be used for routing data to different components based on control signals. For example, it could be used to direct input data to different rounds of the encryption algorithm or to select different keys for each round.

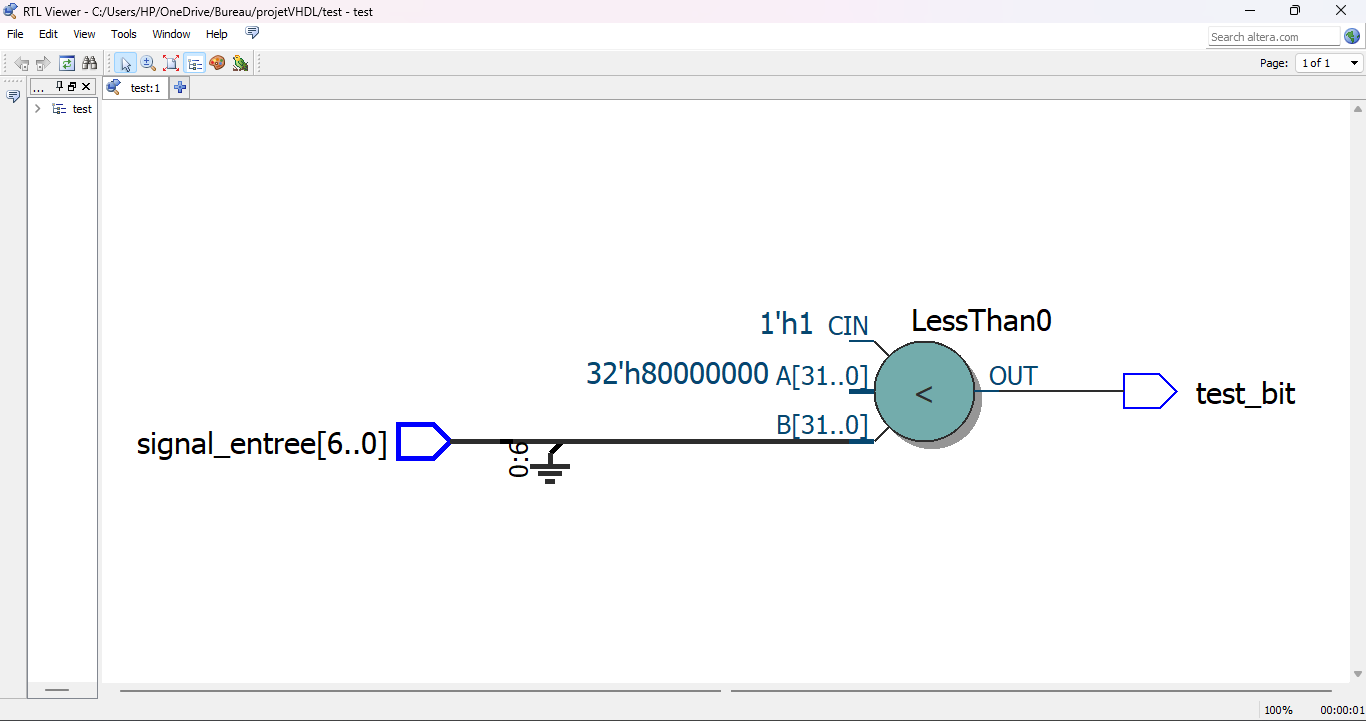
****

****

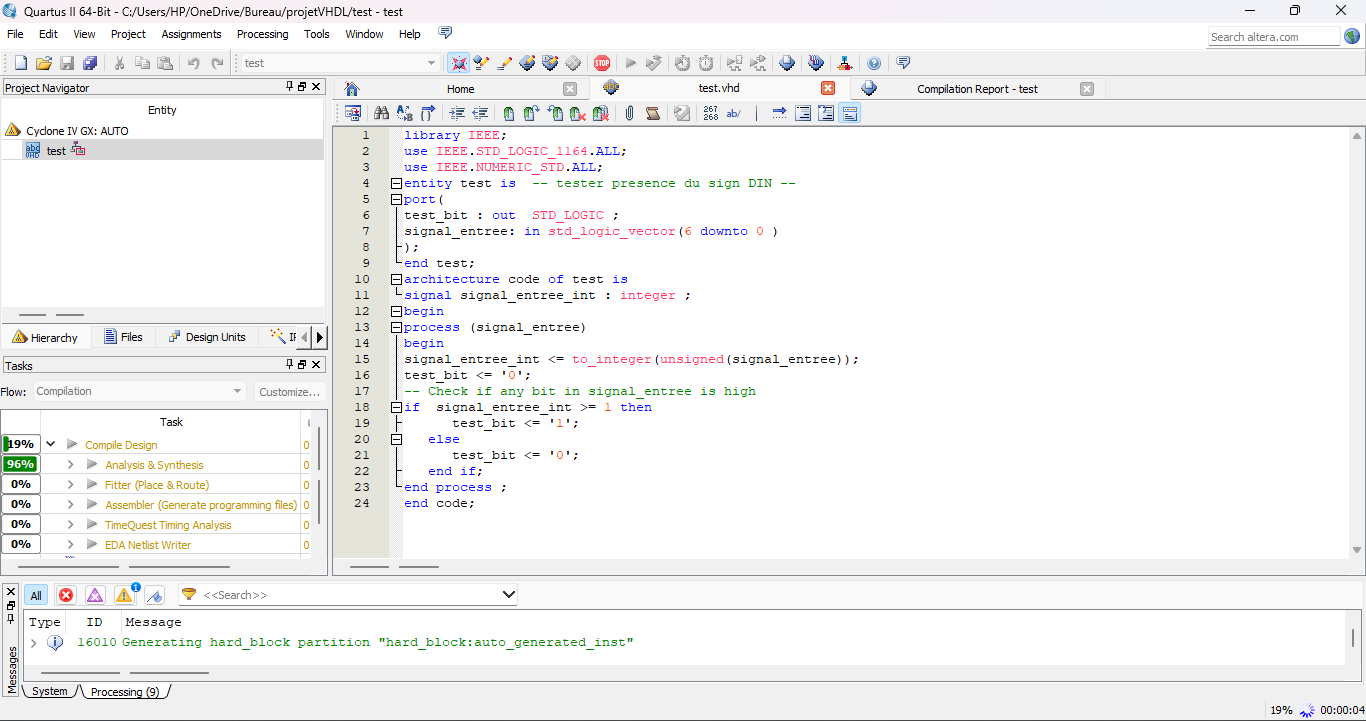


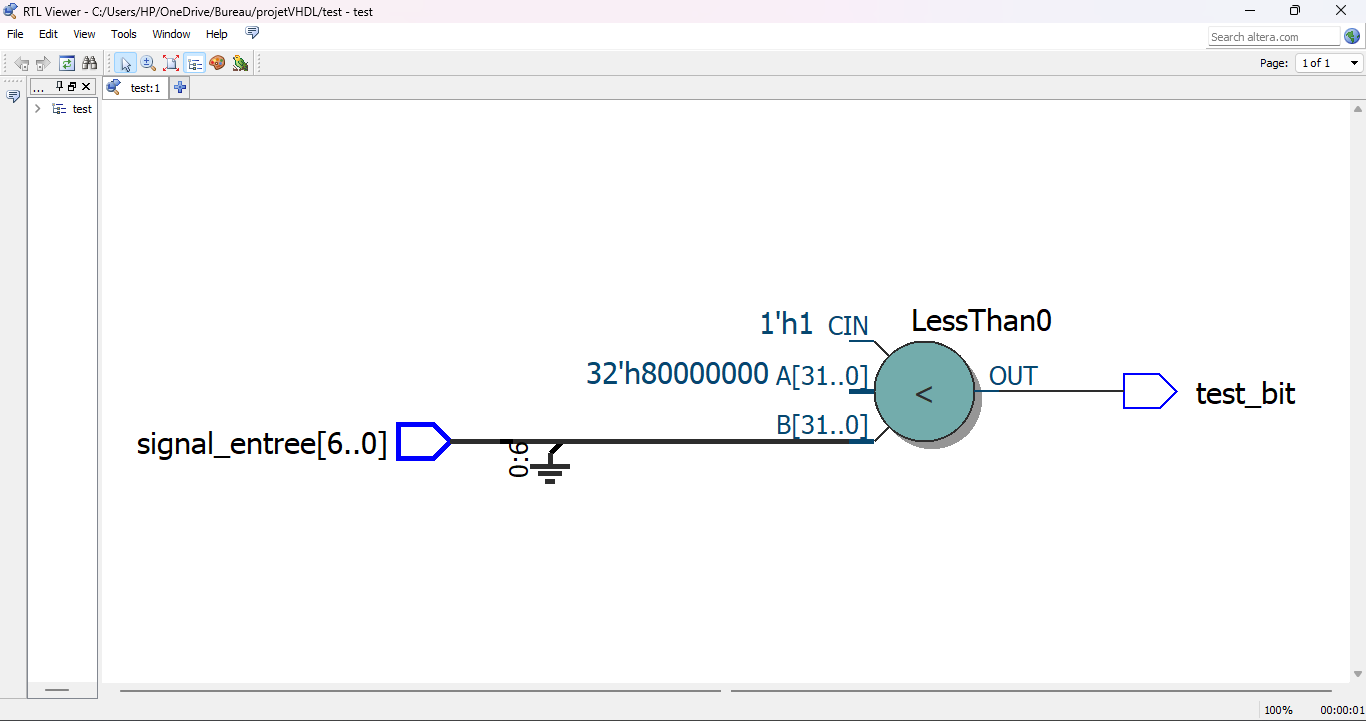
**Test :** In DES, a test component might be used for various purposes, such as verifying input data integrity, checking for specific patterns, or controlling certain aspects of the algorithm based on external conditions.



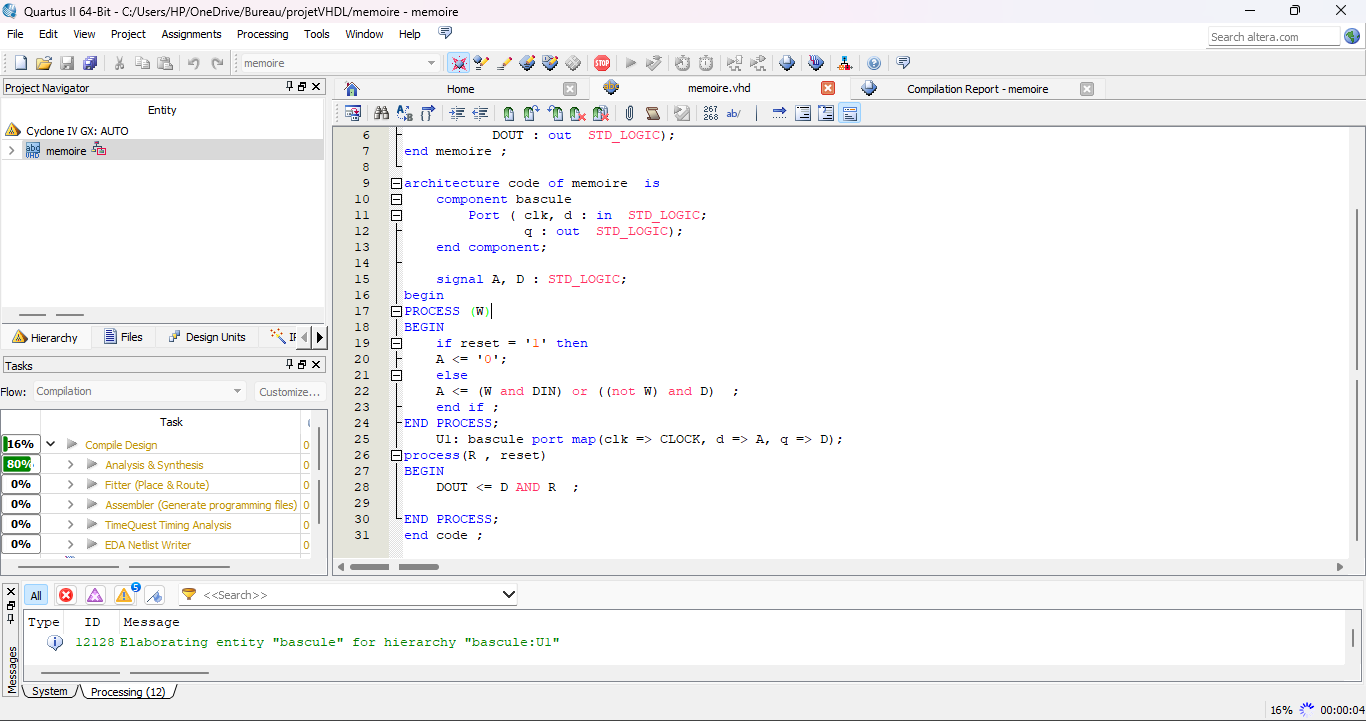
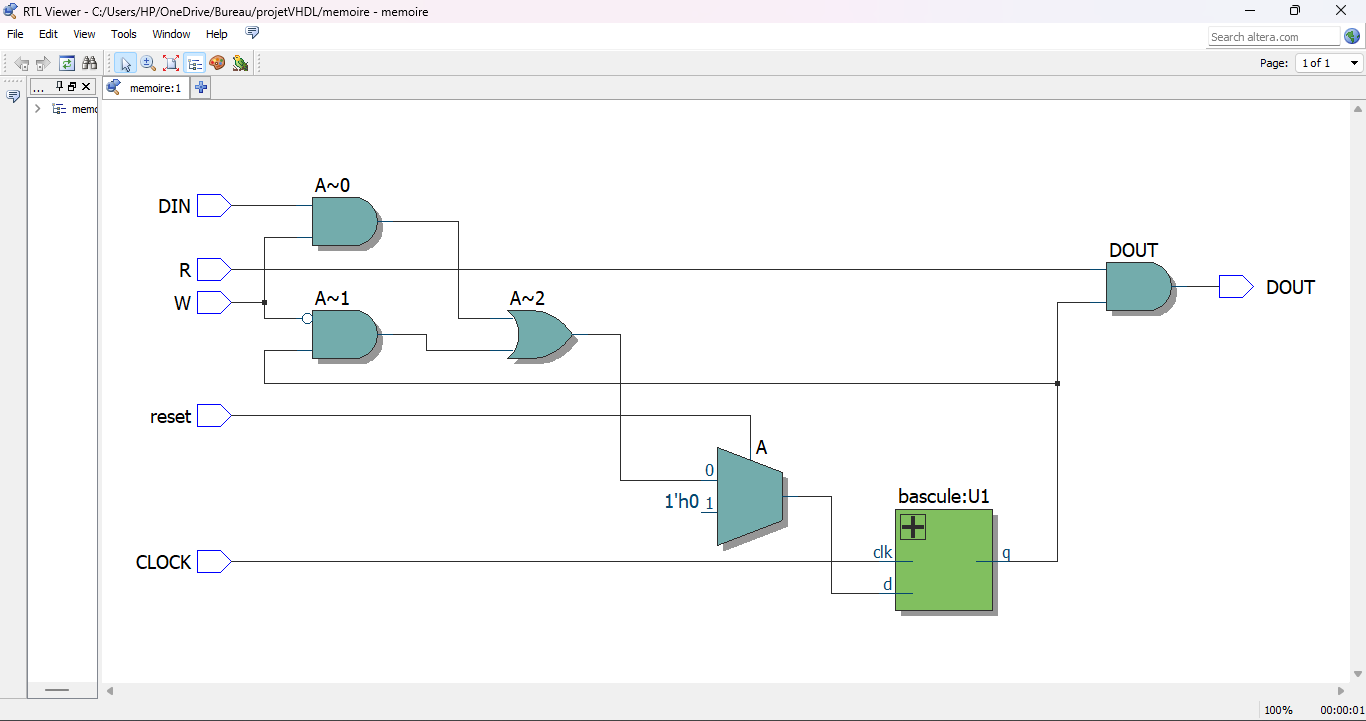


**Incrementer** :

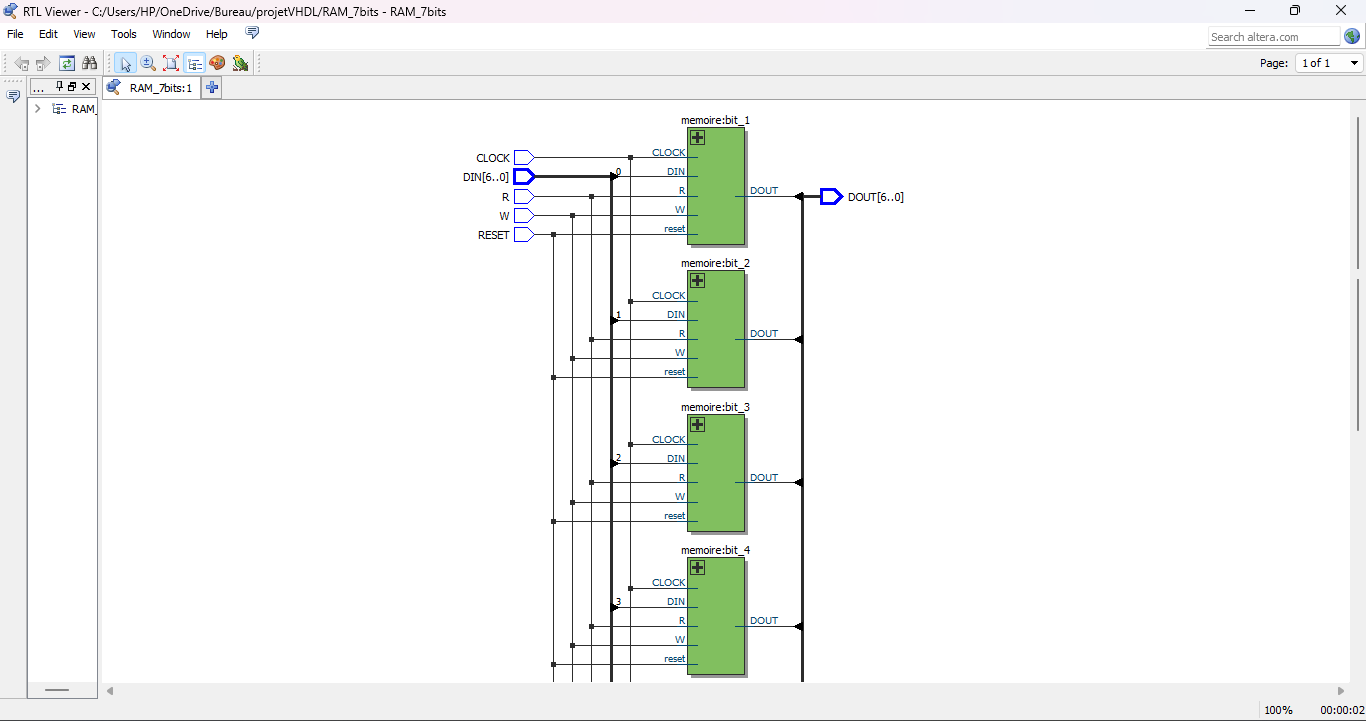
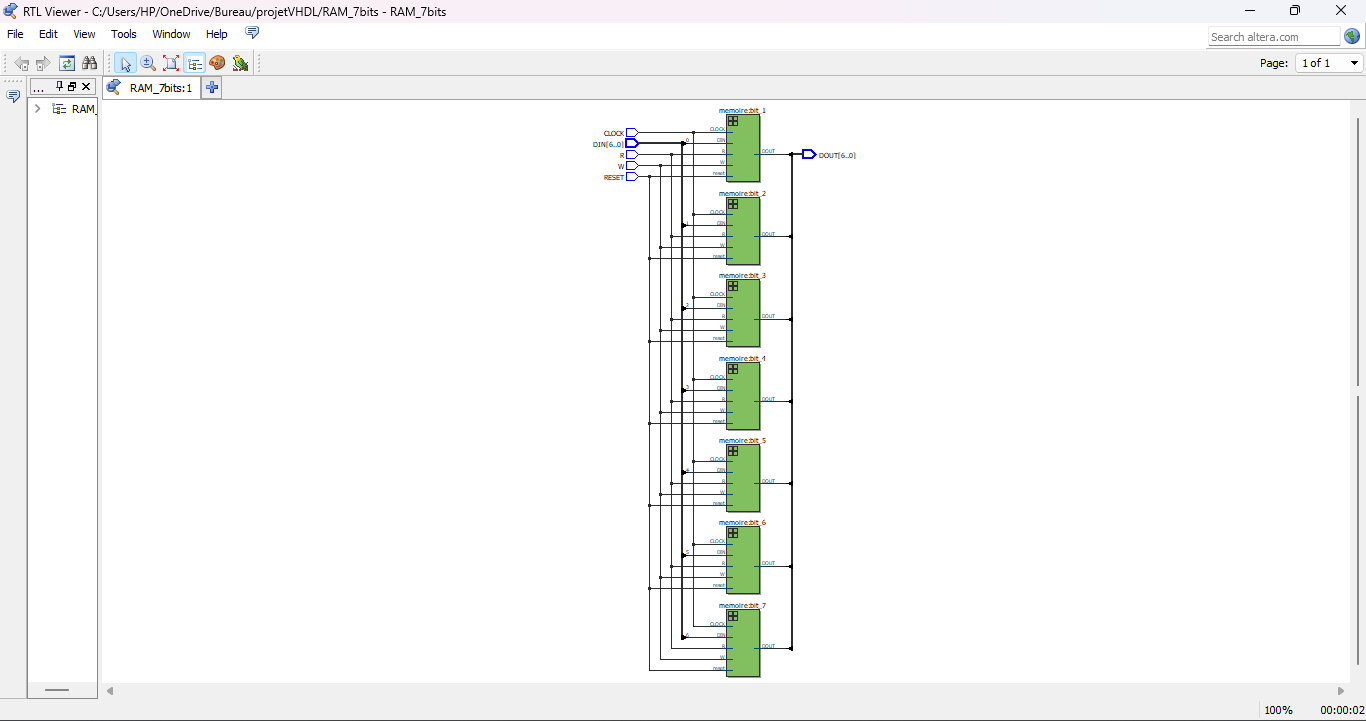
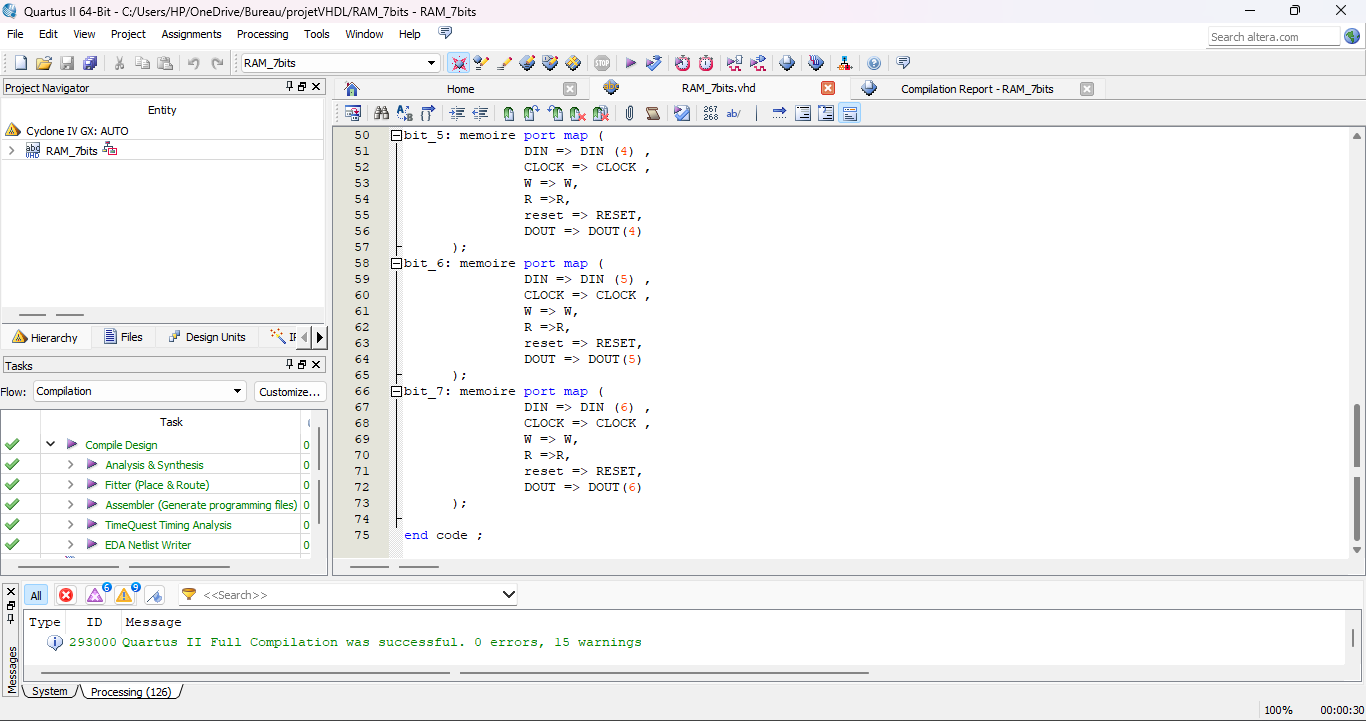
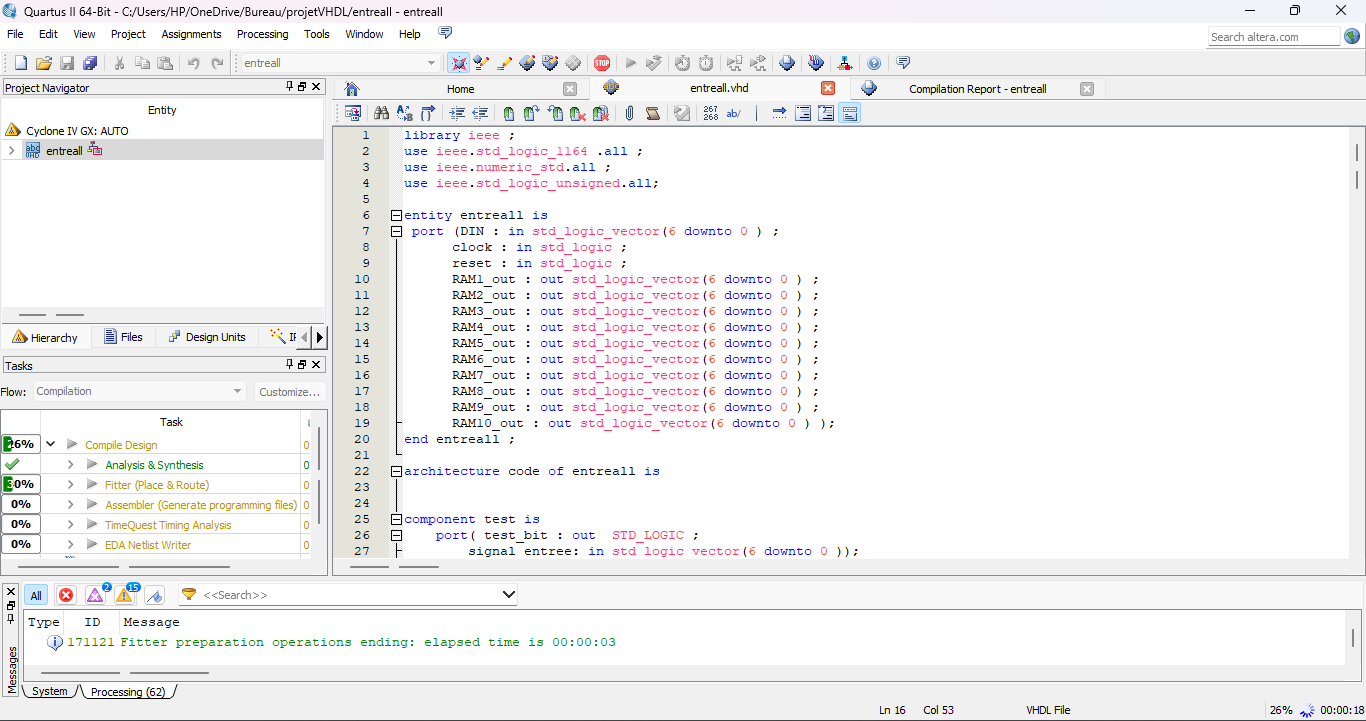
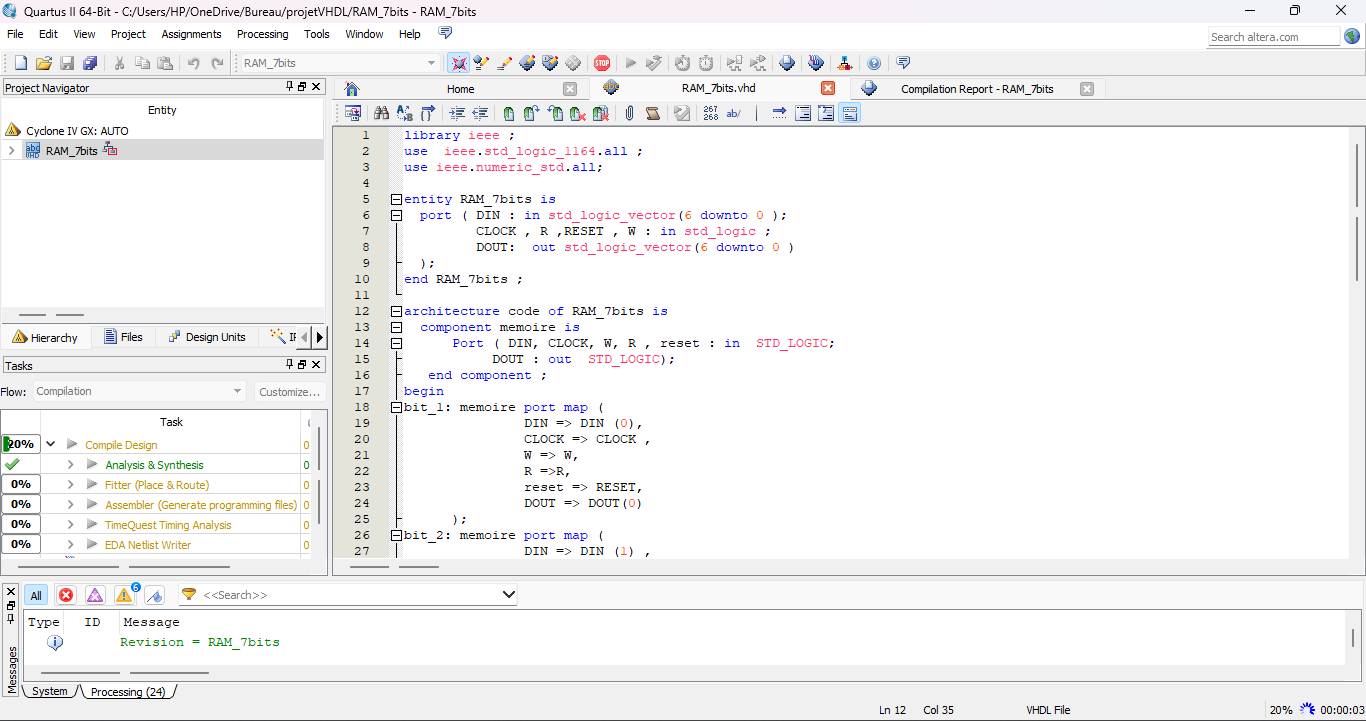
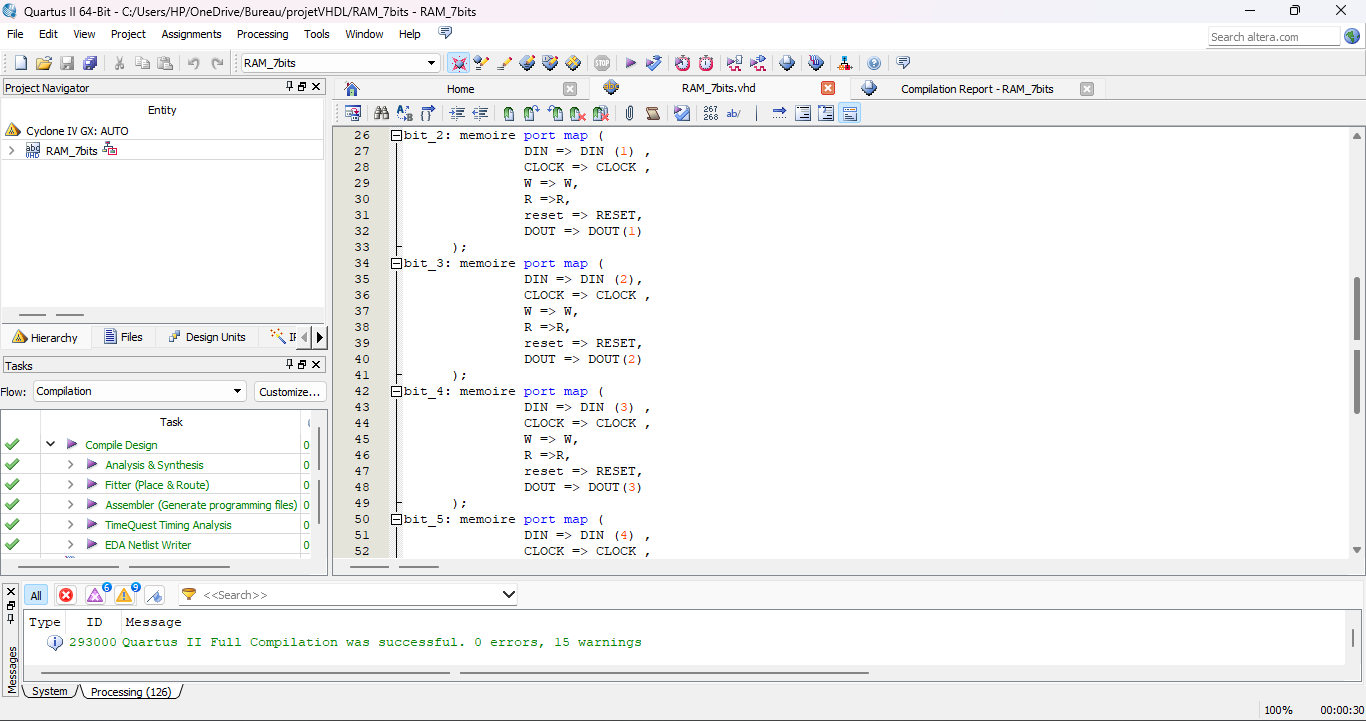
 In DES, an incrementer might be used for generating round numbers or addresses for accessing different parts of the algorithm, such as keys or data blocks.

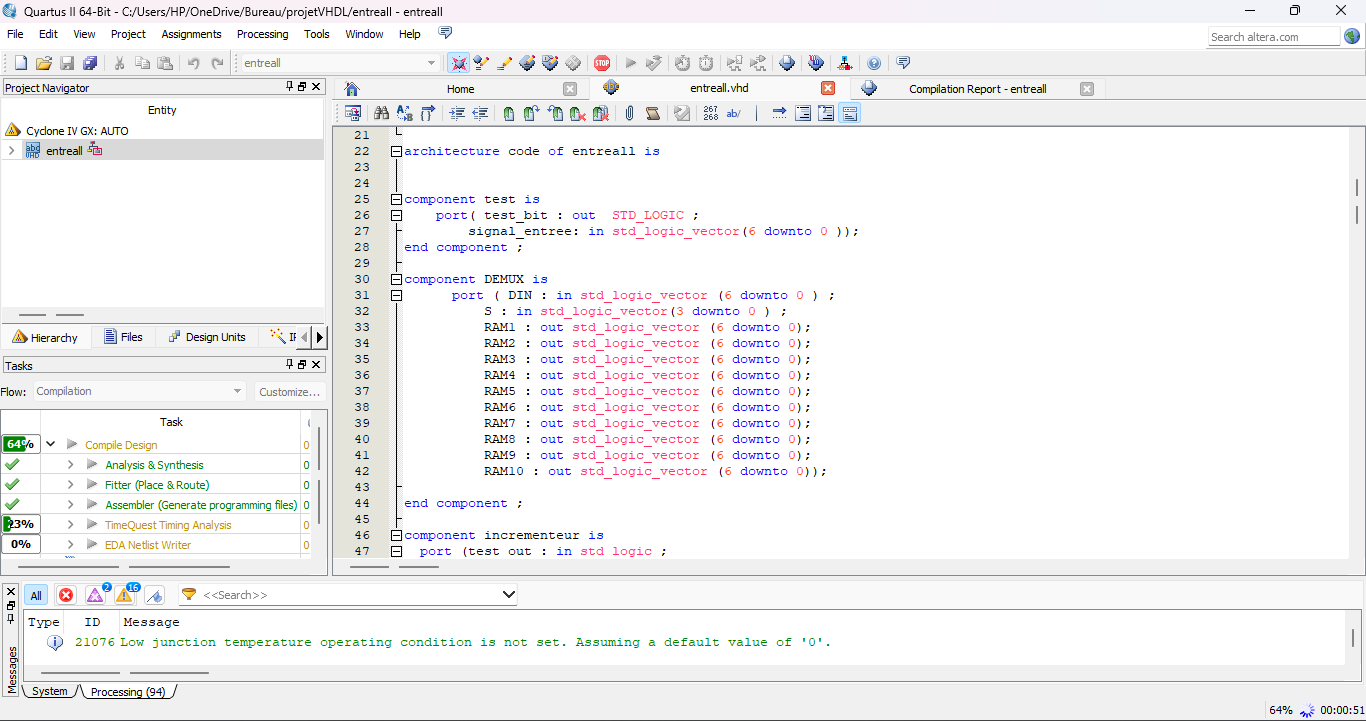
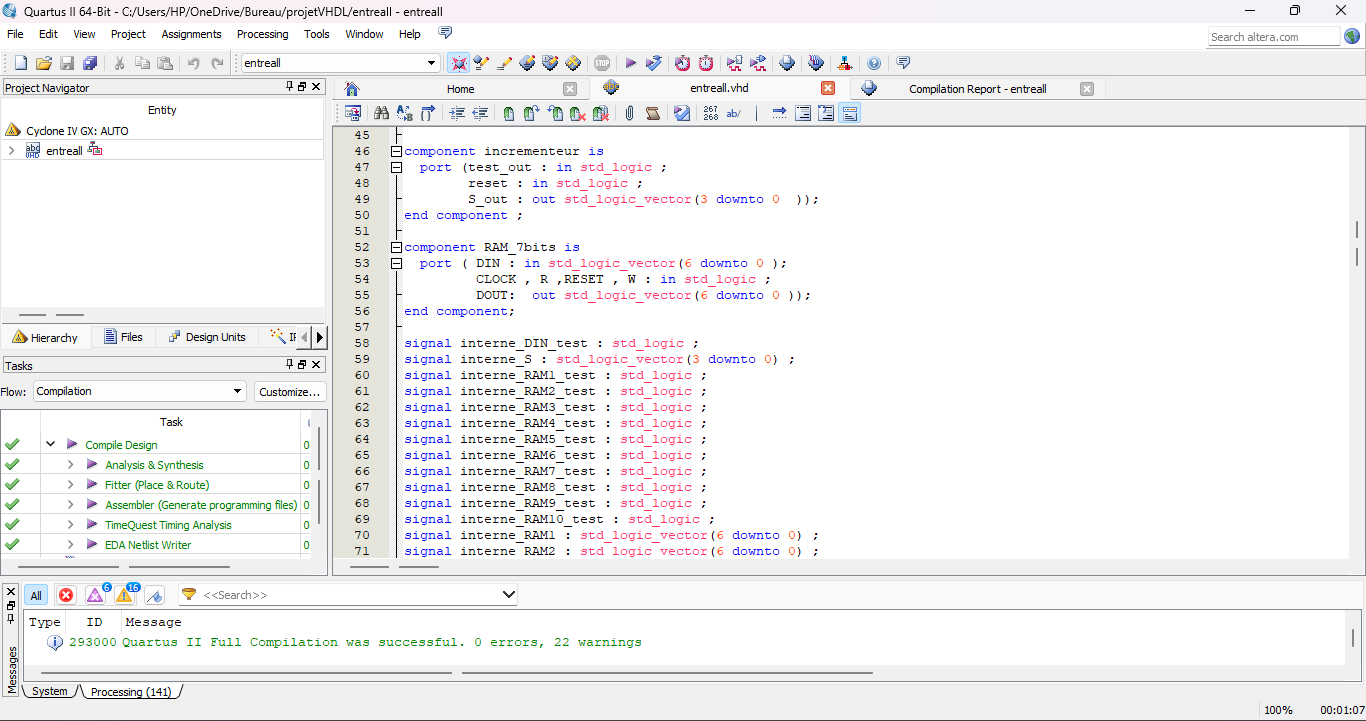
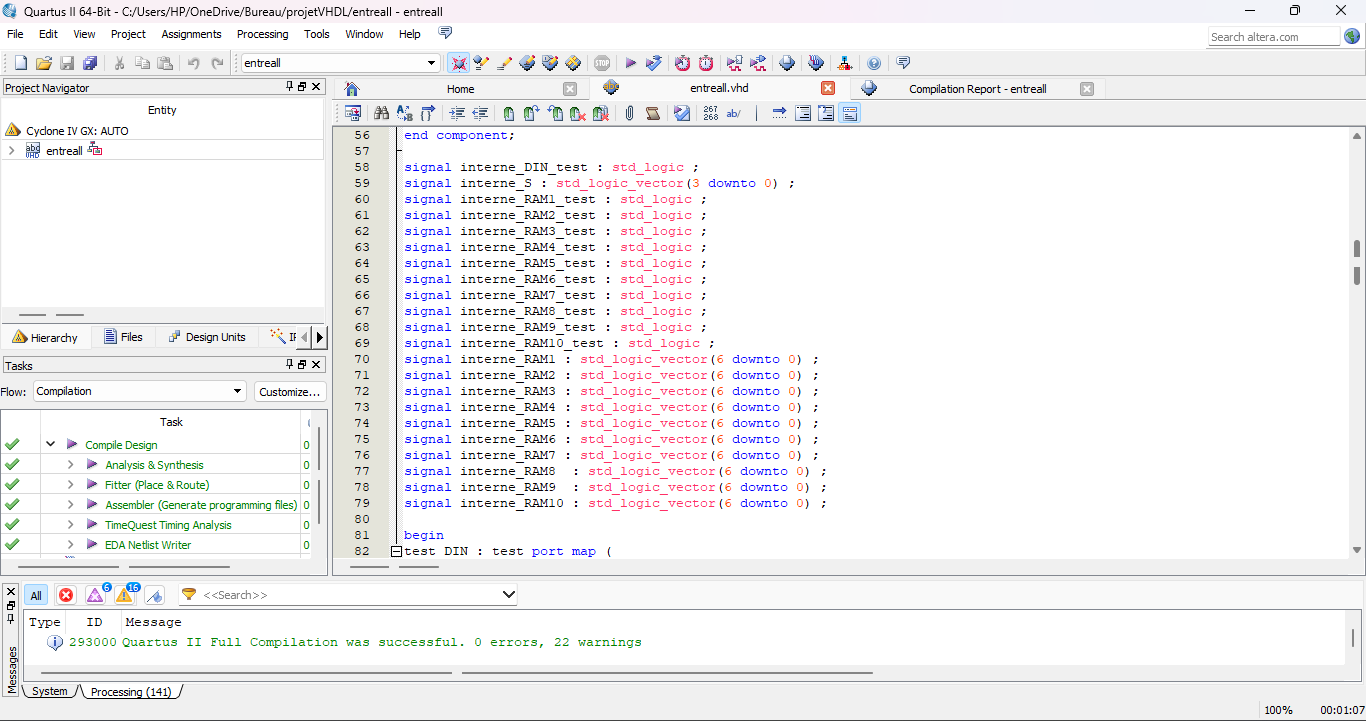
**Memory :**

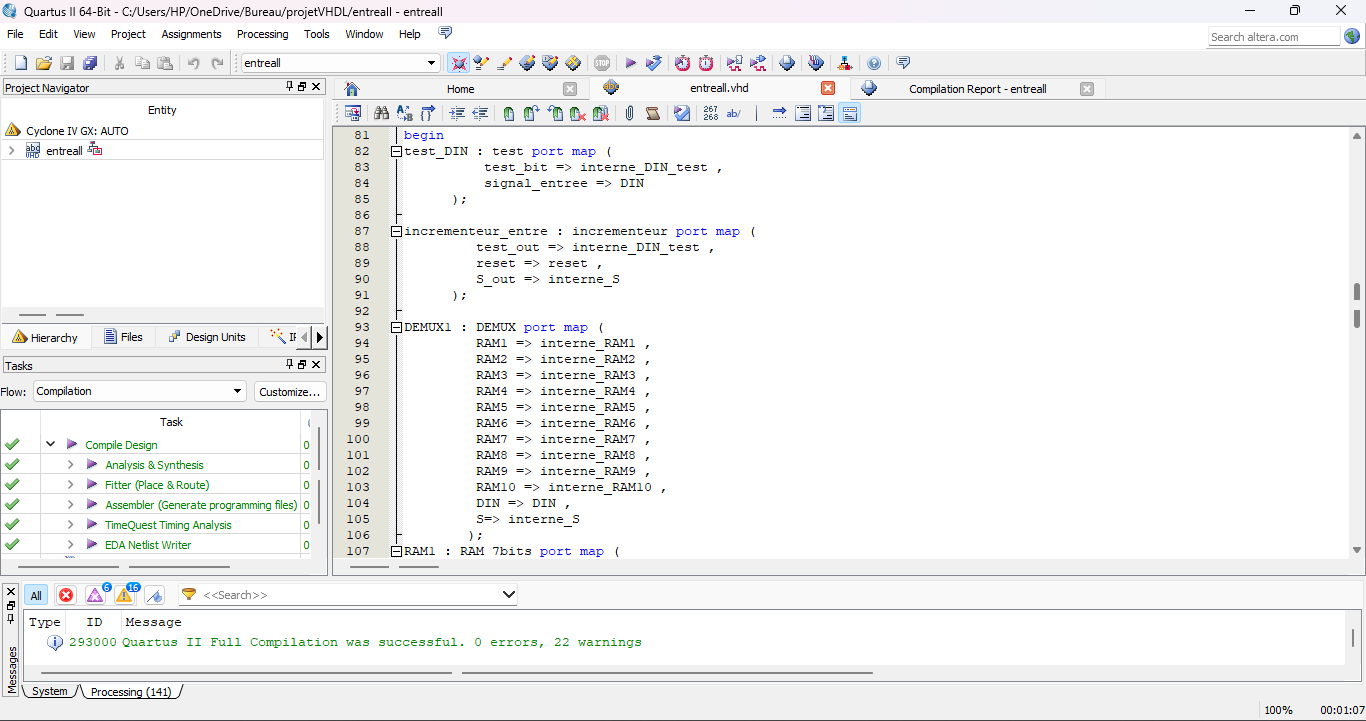
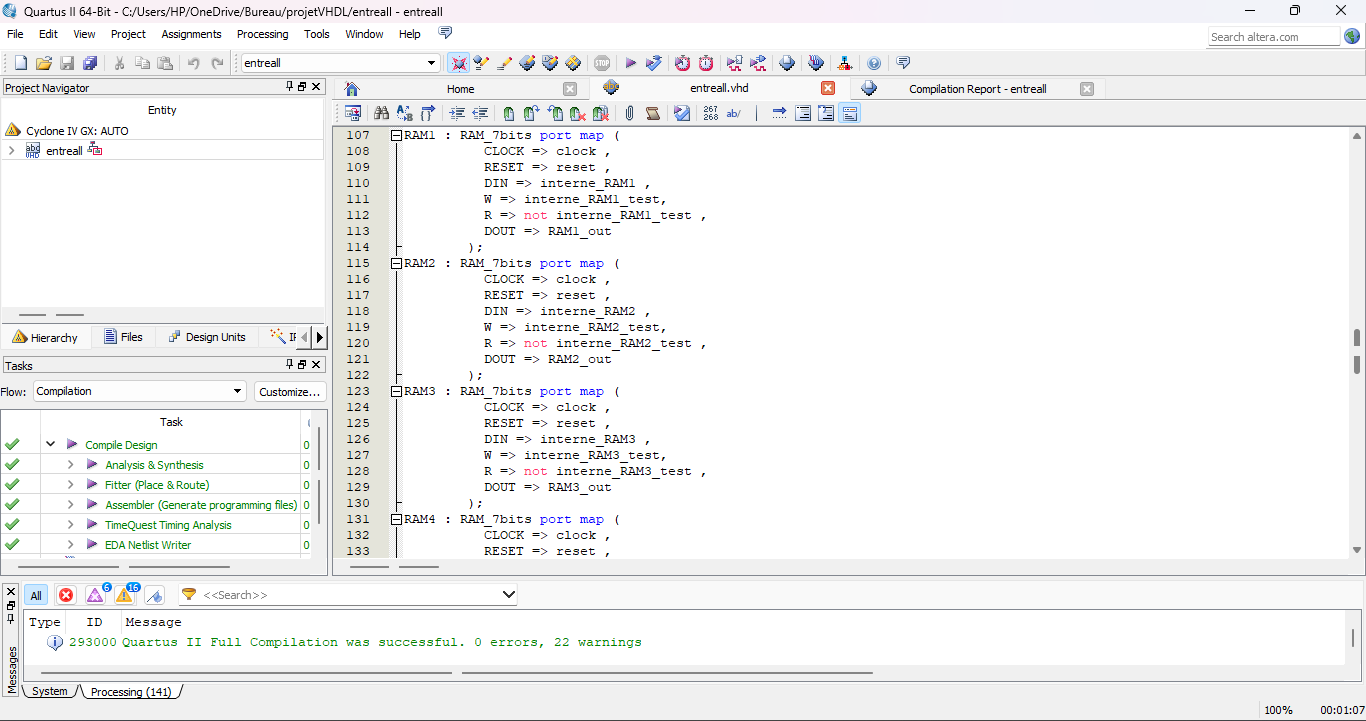
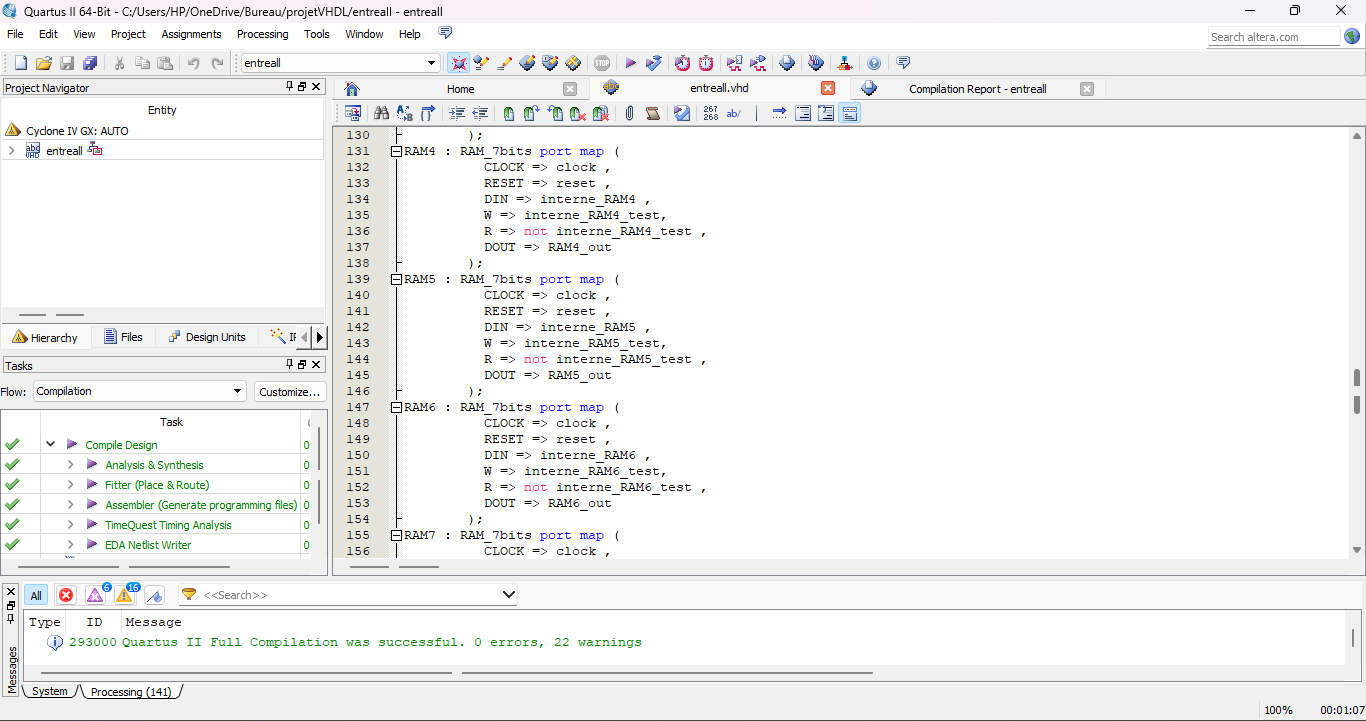
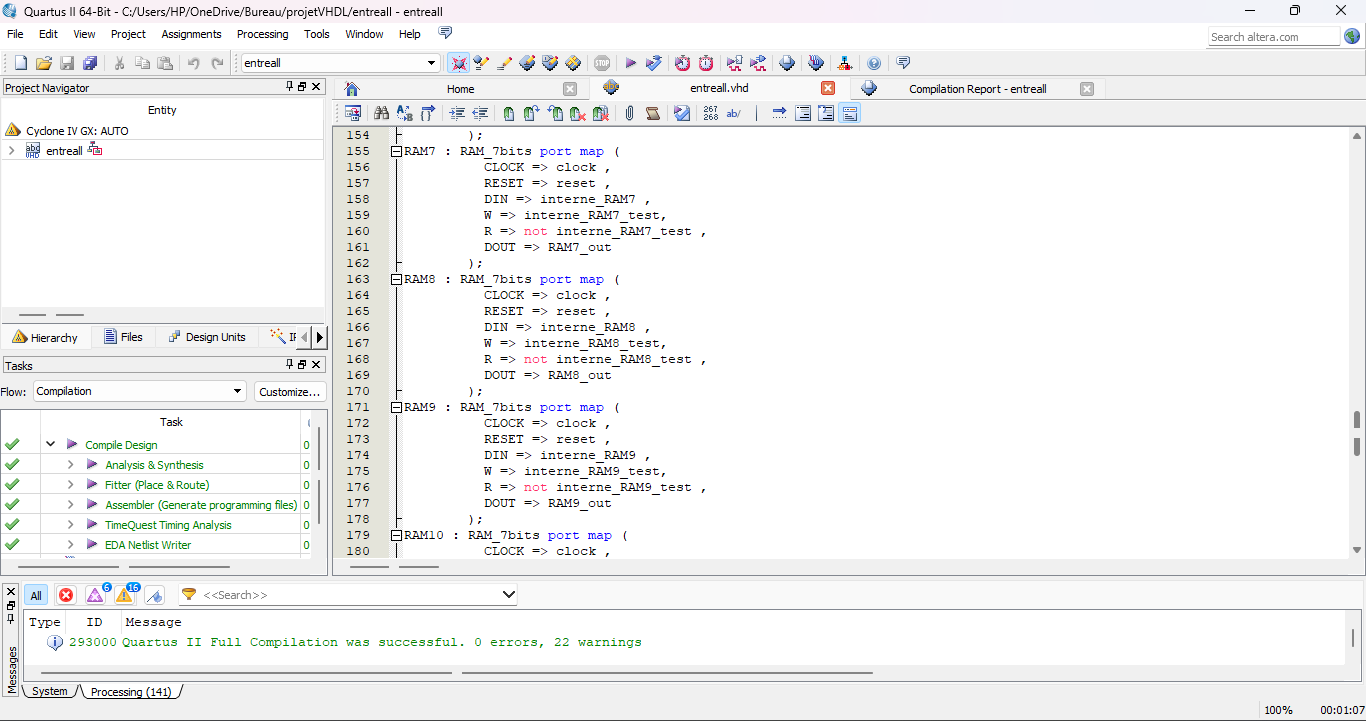
In DES, memory components could be used for storing key schedules, intermediate data, or lookup tables used in the encryption or decryption process

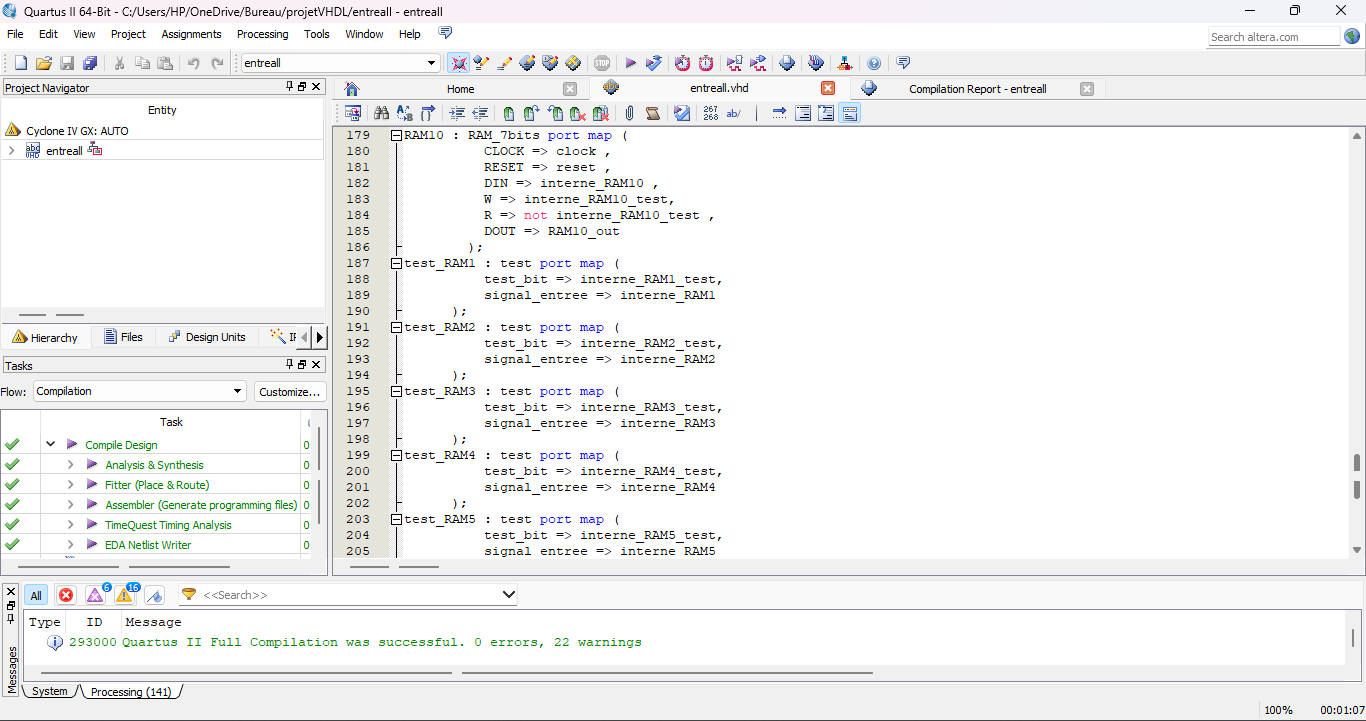


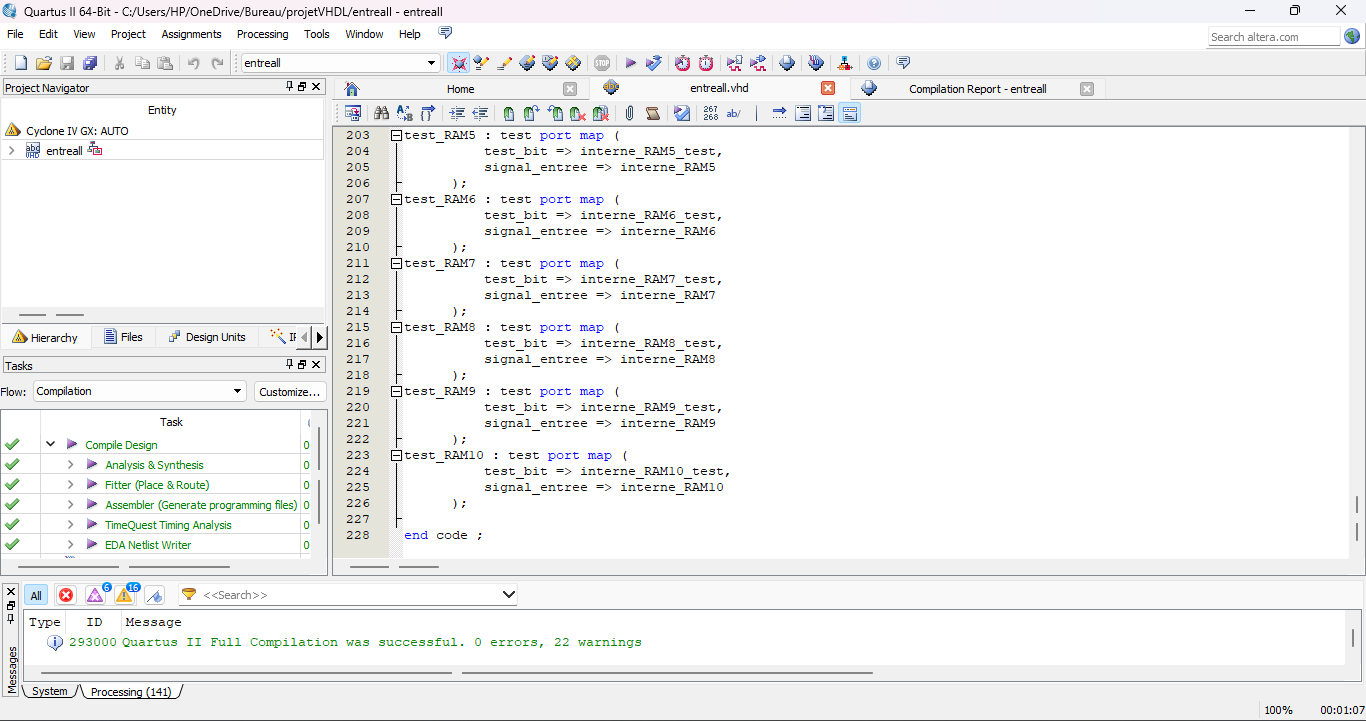
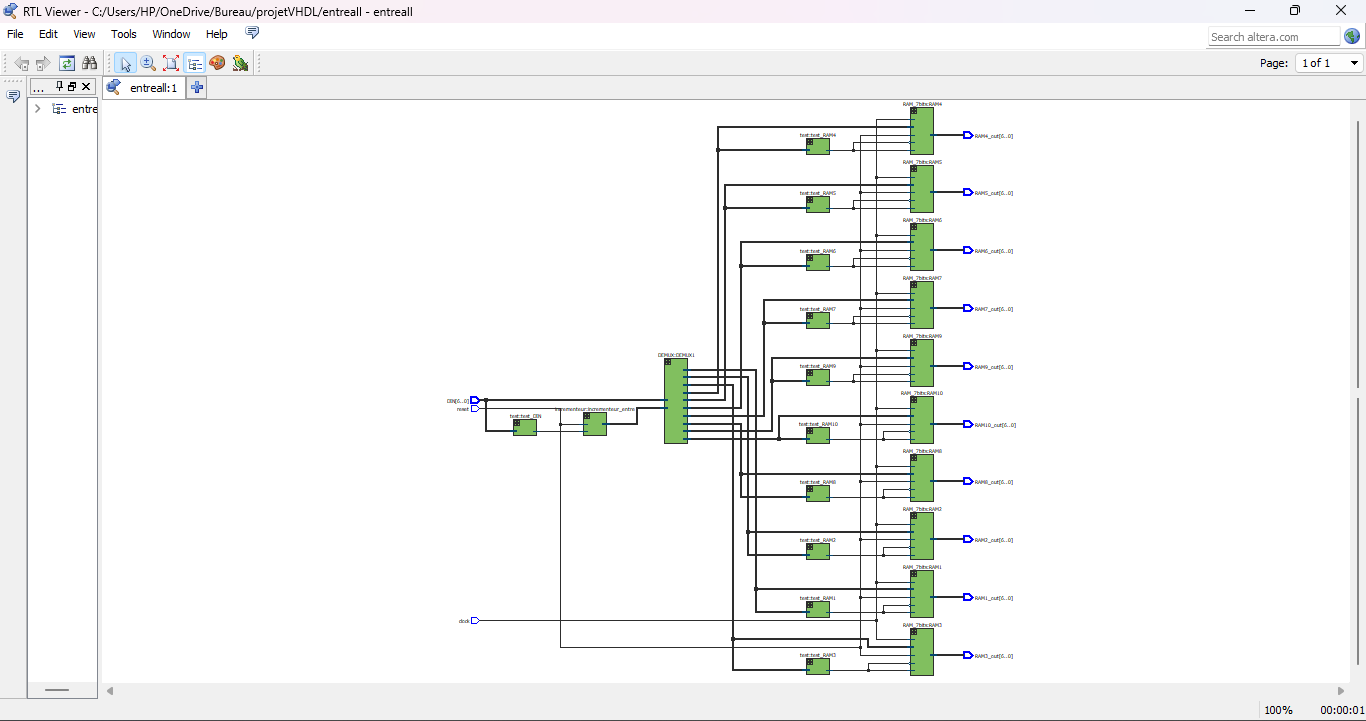
**RAM 7bits :**

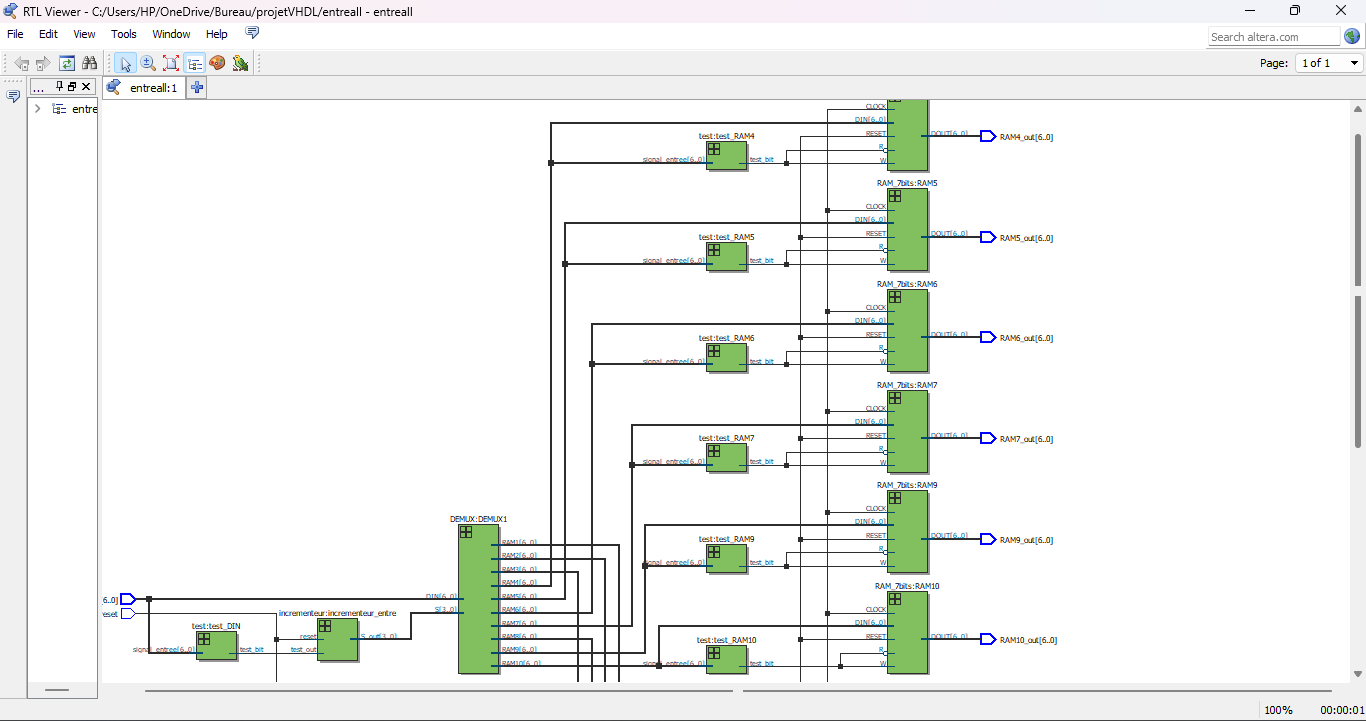
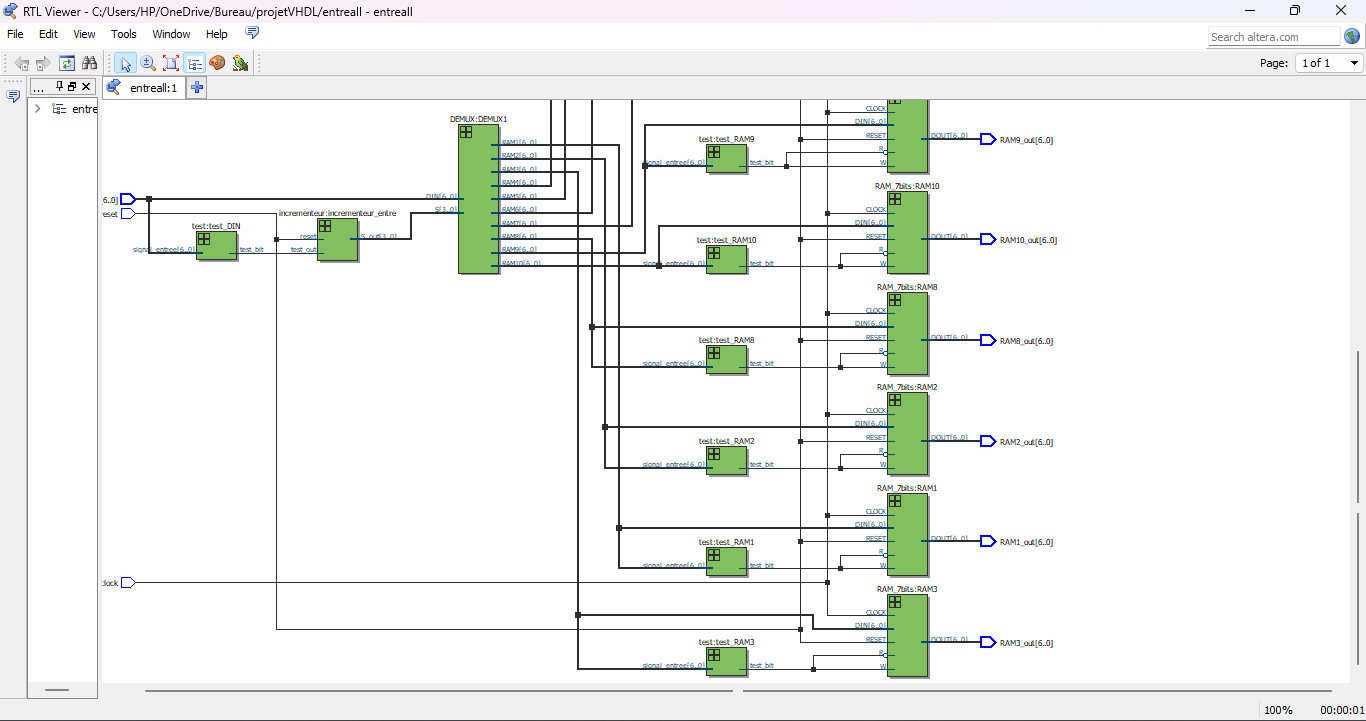
******The entire code :**

****

****

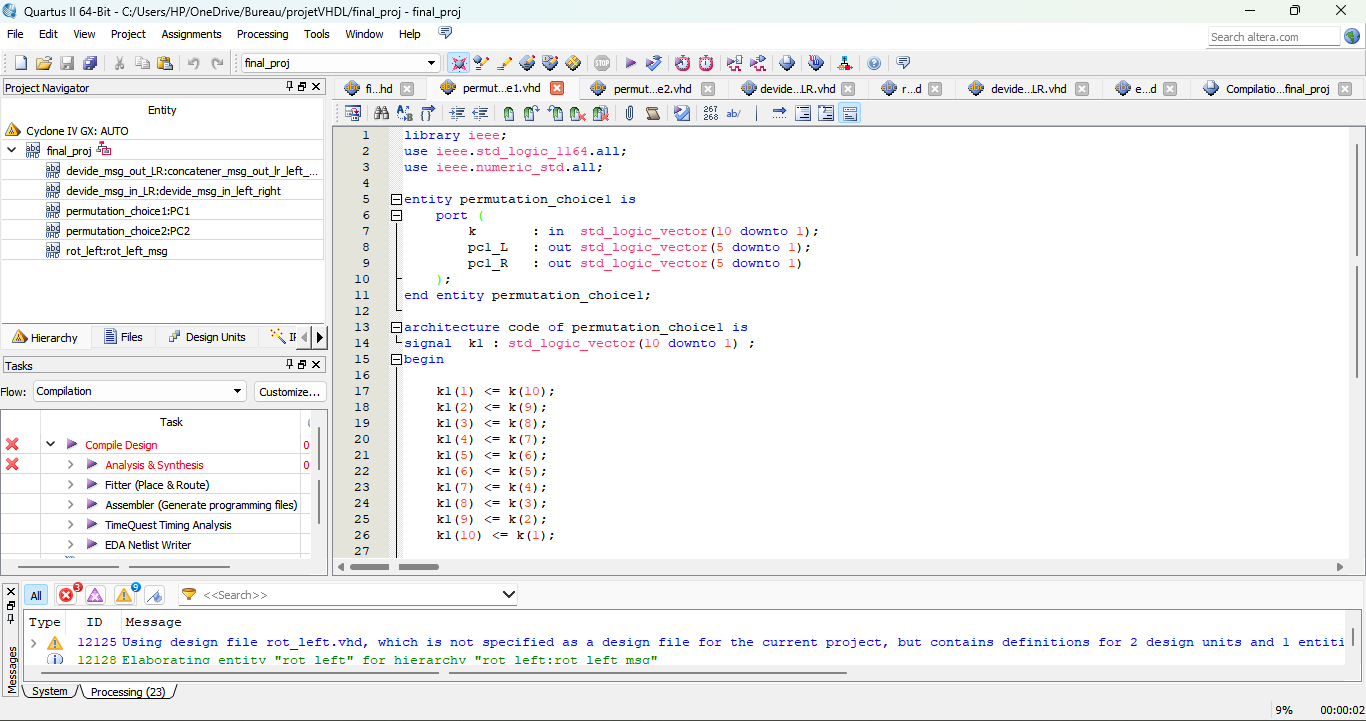
****

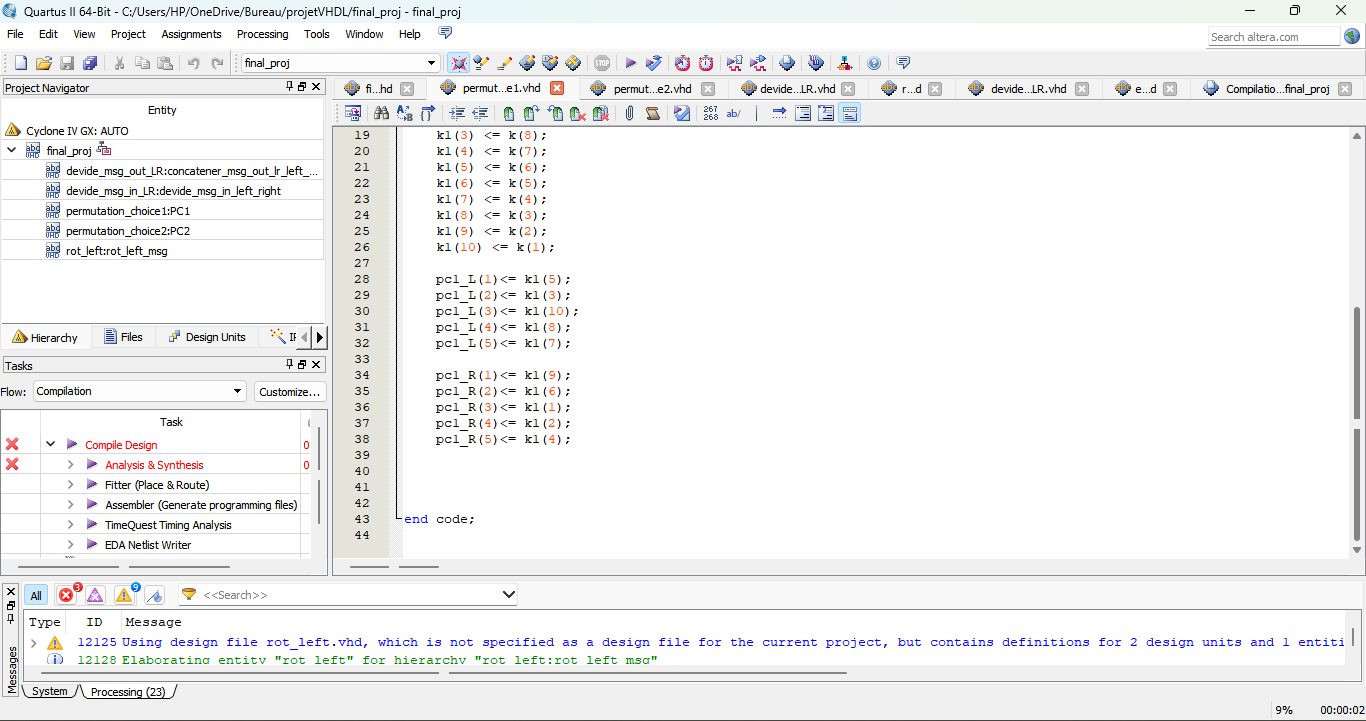
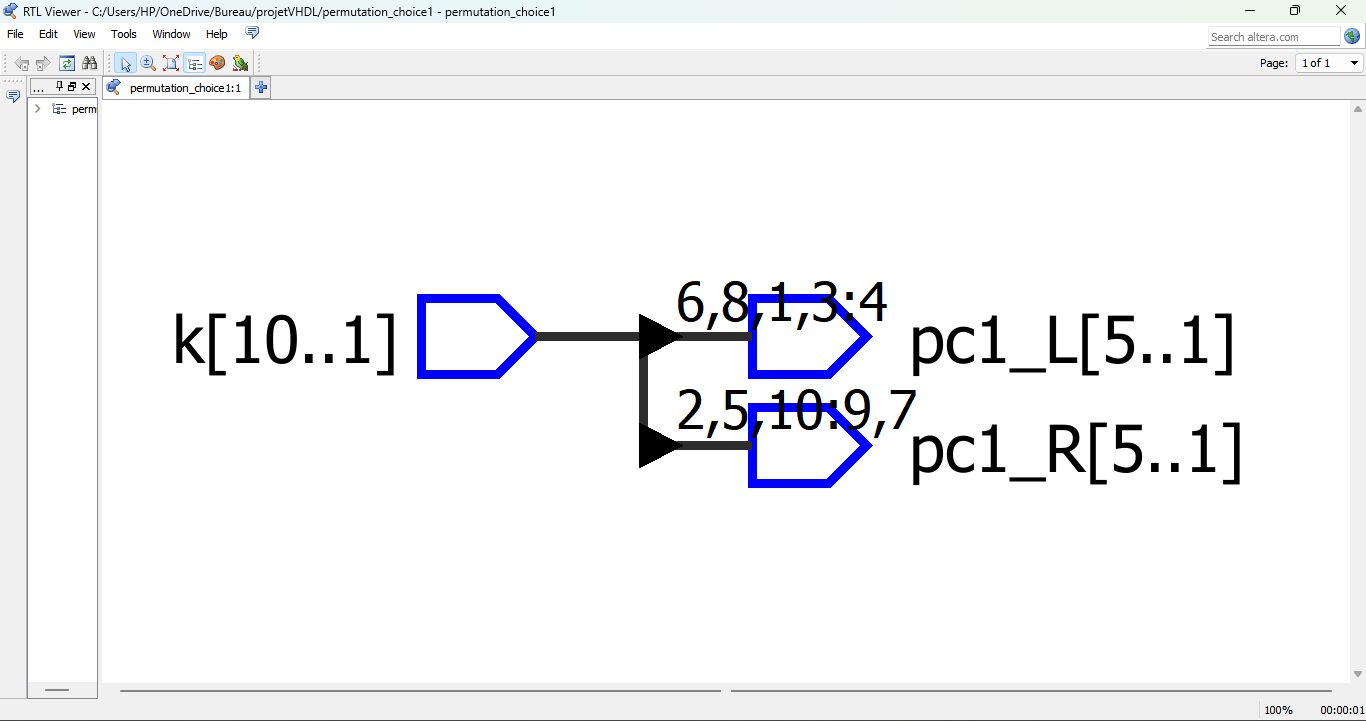
****

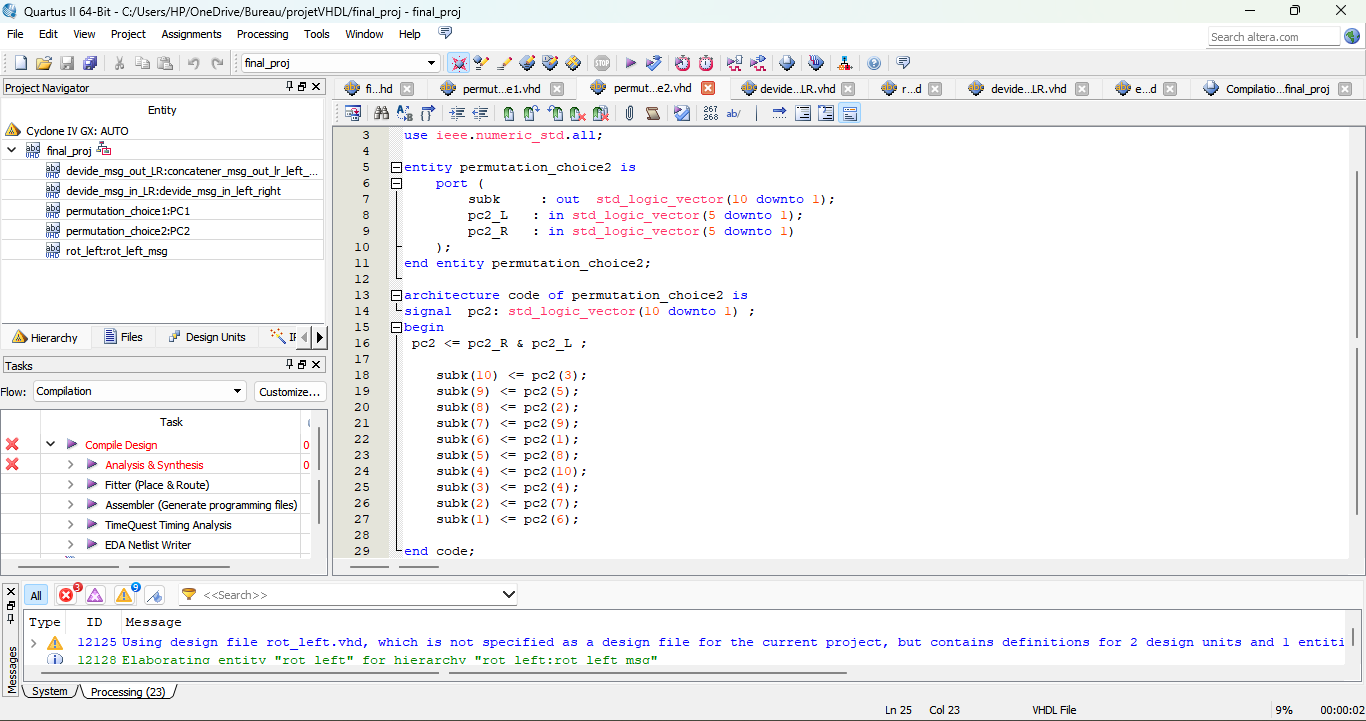
****

**DES code :**

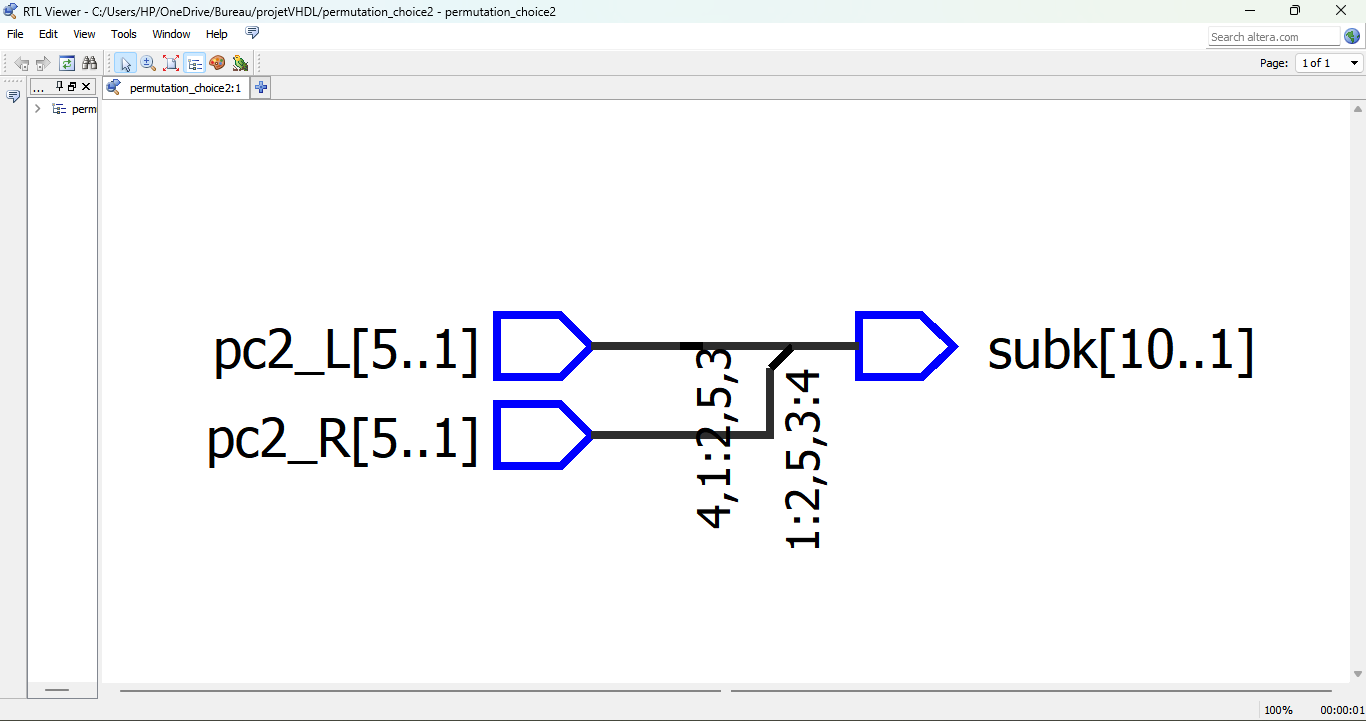
* The entity permutation\_choice1 takes a 10-bit input vector k, reverses its bit order to form k1, and then maps selected bits from k1 to two 5-bit output vectors, pc1\_L and pc1\_R. This kind of bit manipulation is often used in cryptographic algorithms and other digital processing tasks where specific bit patterns are required.

****

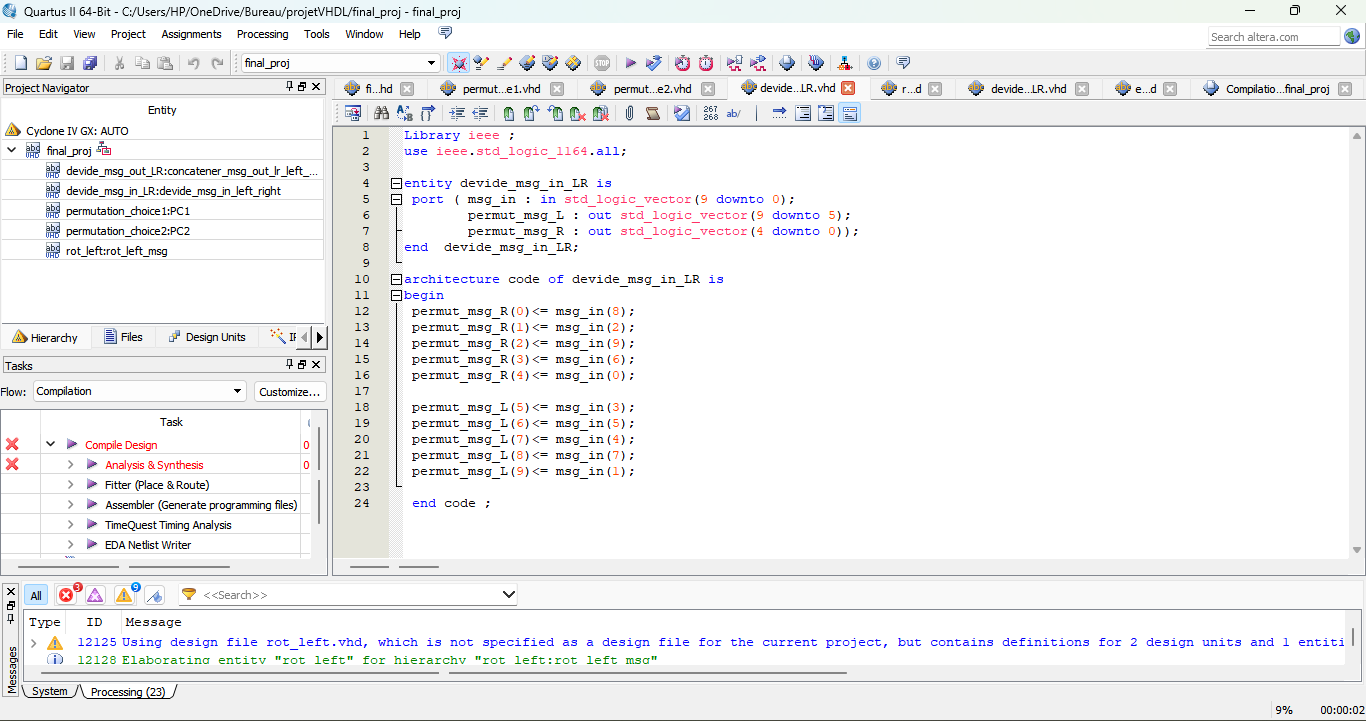
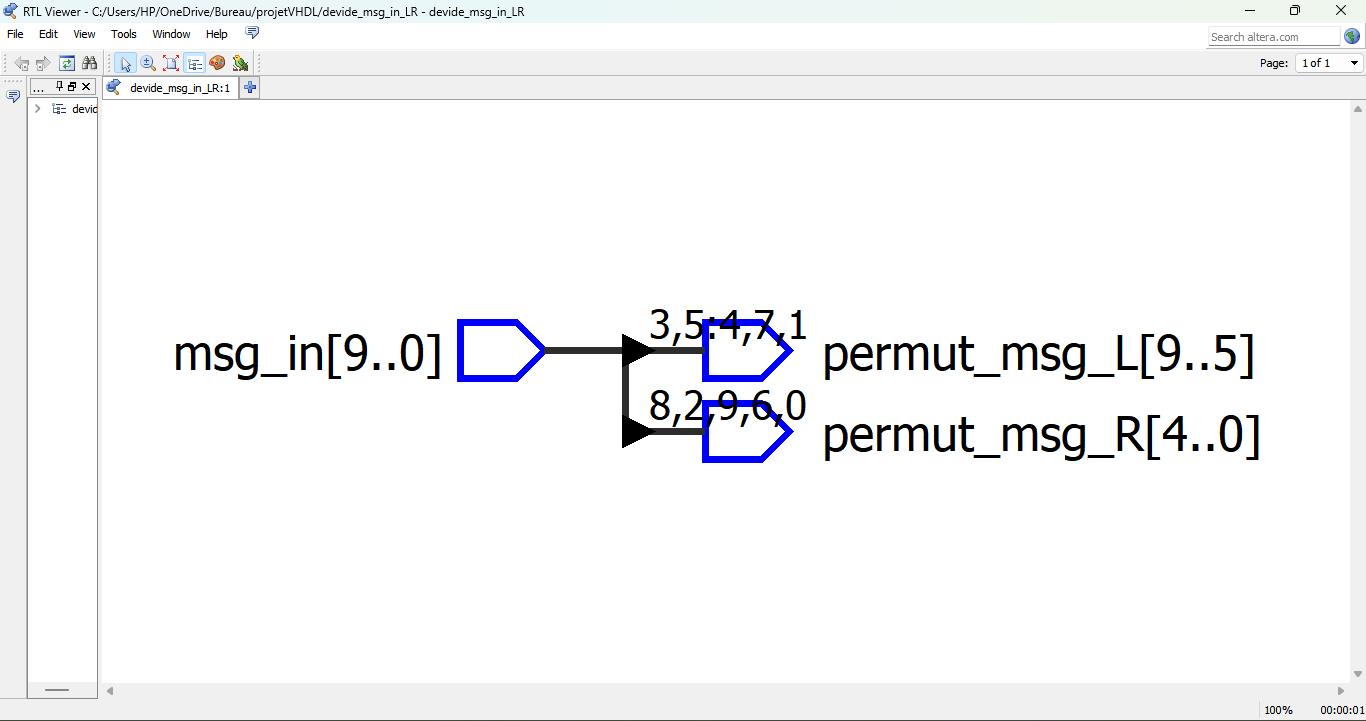
****

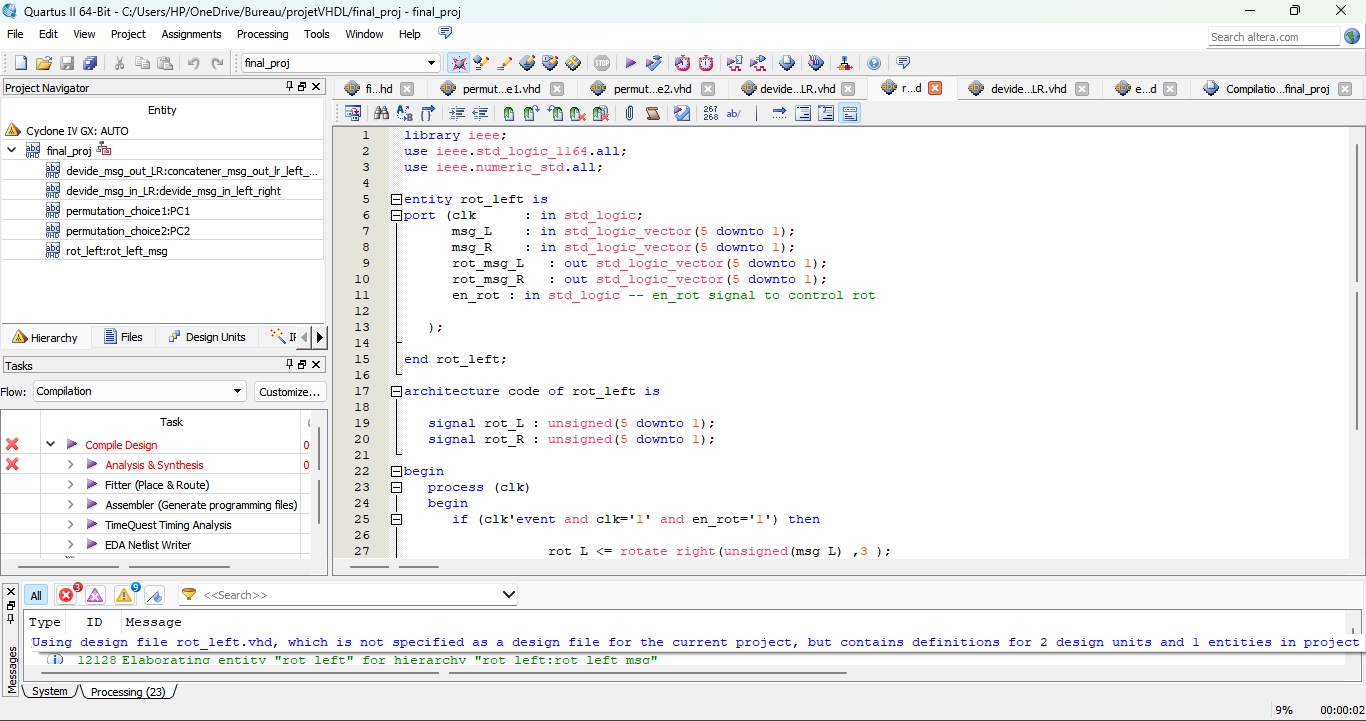
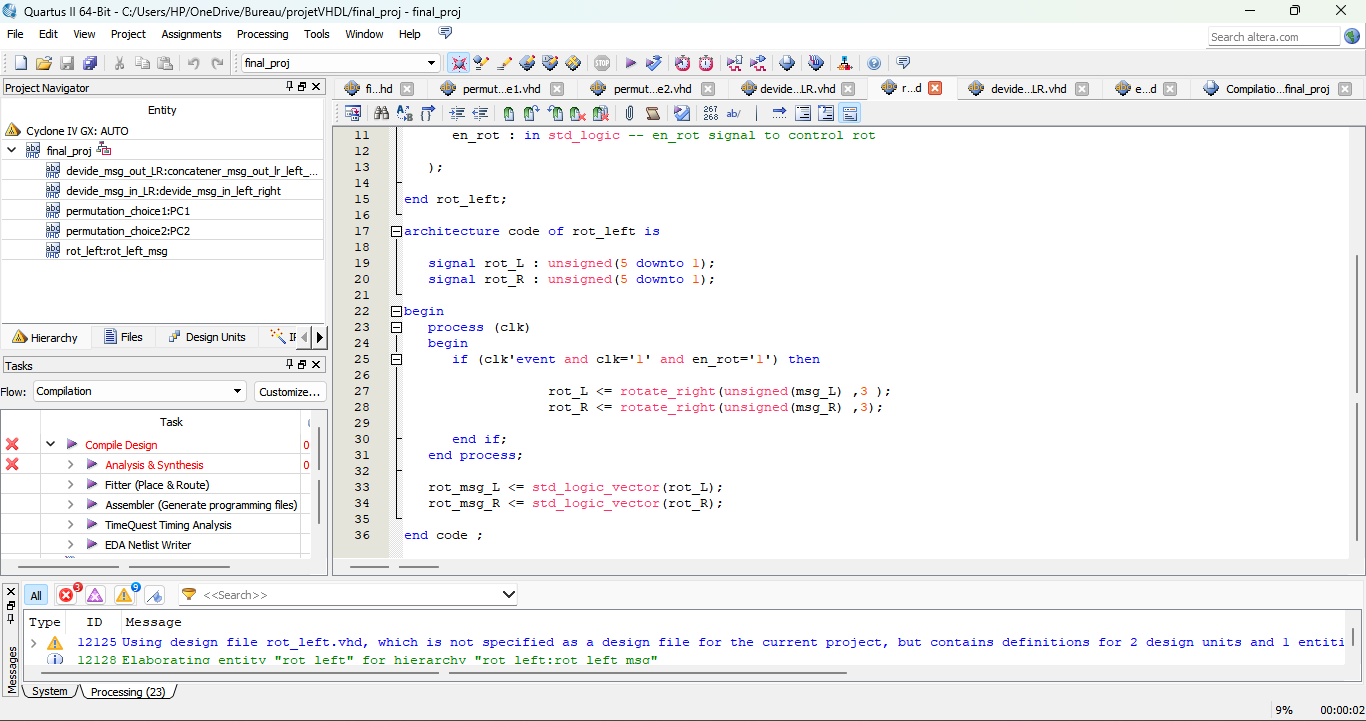
****

* This code defines an entity permutation\_choice2 that concatenates two 5-bit input vectors, pc2\_L and pc2\_R, into a single 10-bit signal pc2. It then permutes the bits of pc2 to generate a 10-bit output vector subk based on a specified bit selection pattern. This entity is useful for reordering bits from two smaller input vectors into a larger output vector, typically for use in cryptographic or digital processing applications.

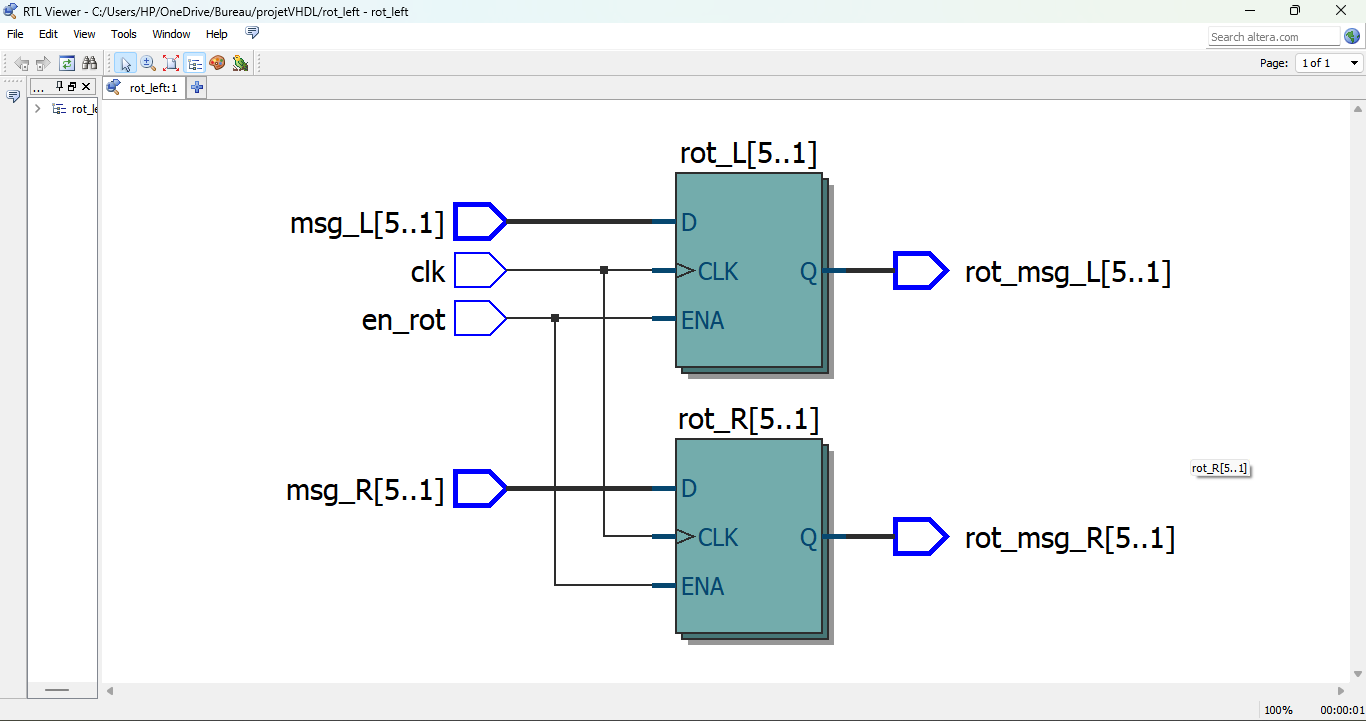


* This code defines an entity `devide\_msg\_in\_LR` that takes a 10-bit input vector `msg\_in` and produces two output vectors, `permut\_msg\_L` and `permut\_msg\_R`, by permuting specific bits of `msg\_in`. This entity is used to rearrange the bits of the input vector into two separate output vectors, typically for further processing.

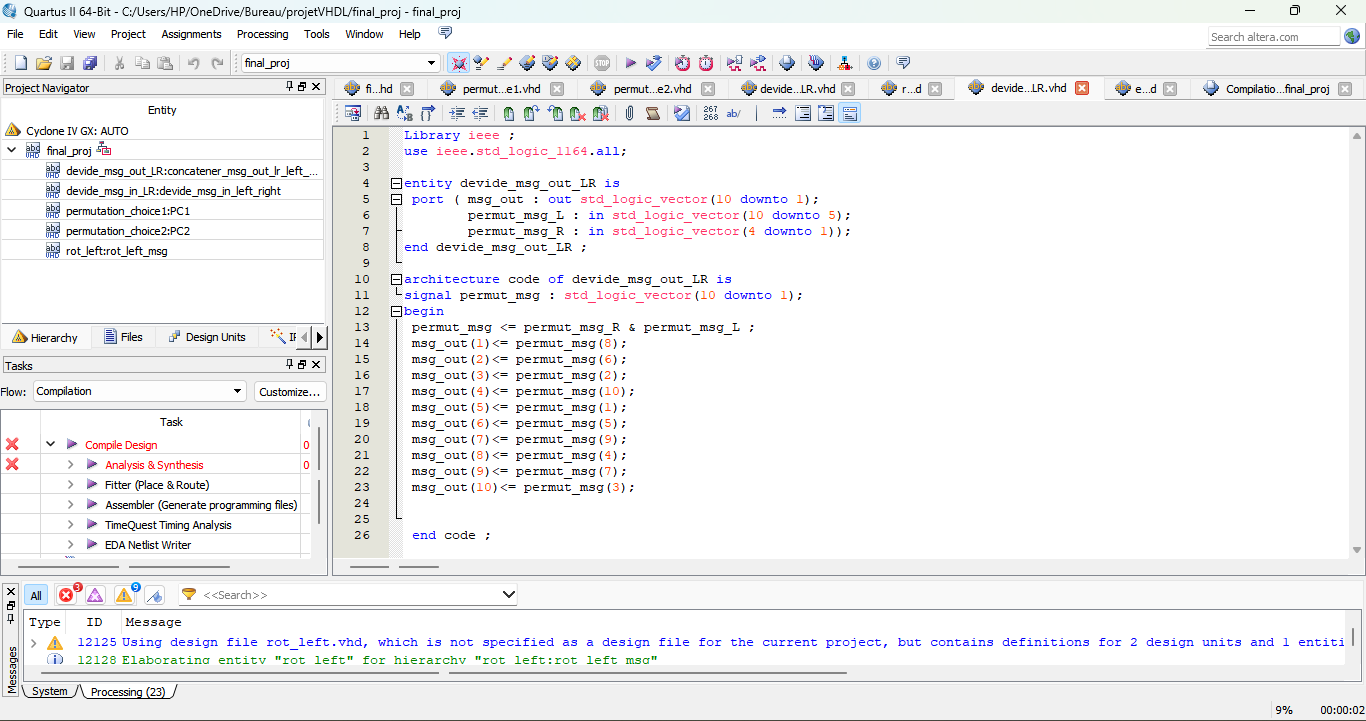
****

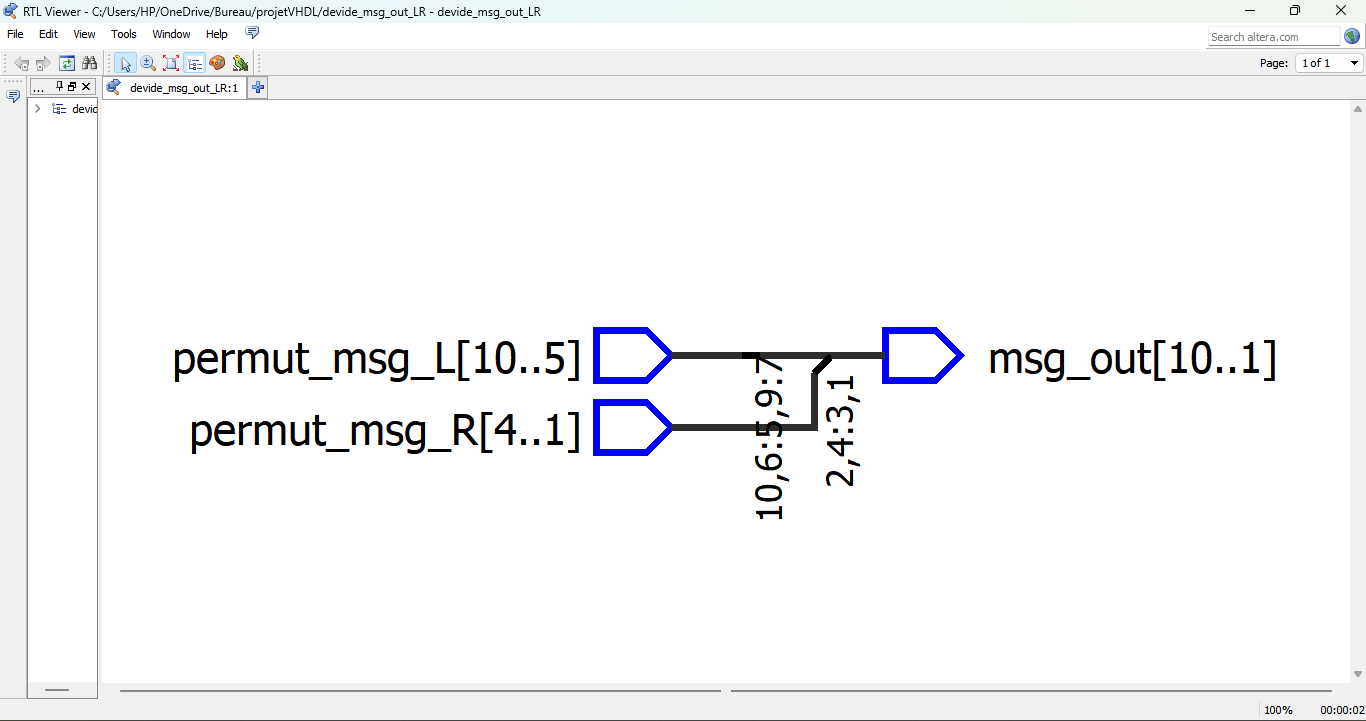
****

* This code defines an entity `rot\_left` that performs a 3-bit right rotation on two 5-bit input vectors, `msg\_L` and `msg\_R`, when enabled by the `en\_rot` signal on the rising edge of the clock (`clk`). The rotated results are output as `rot\_msg\_L` and `rot\_msg\_R`.

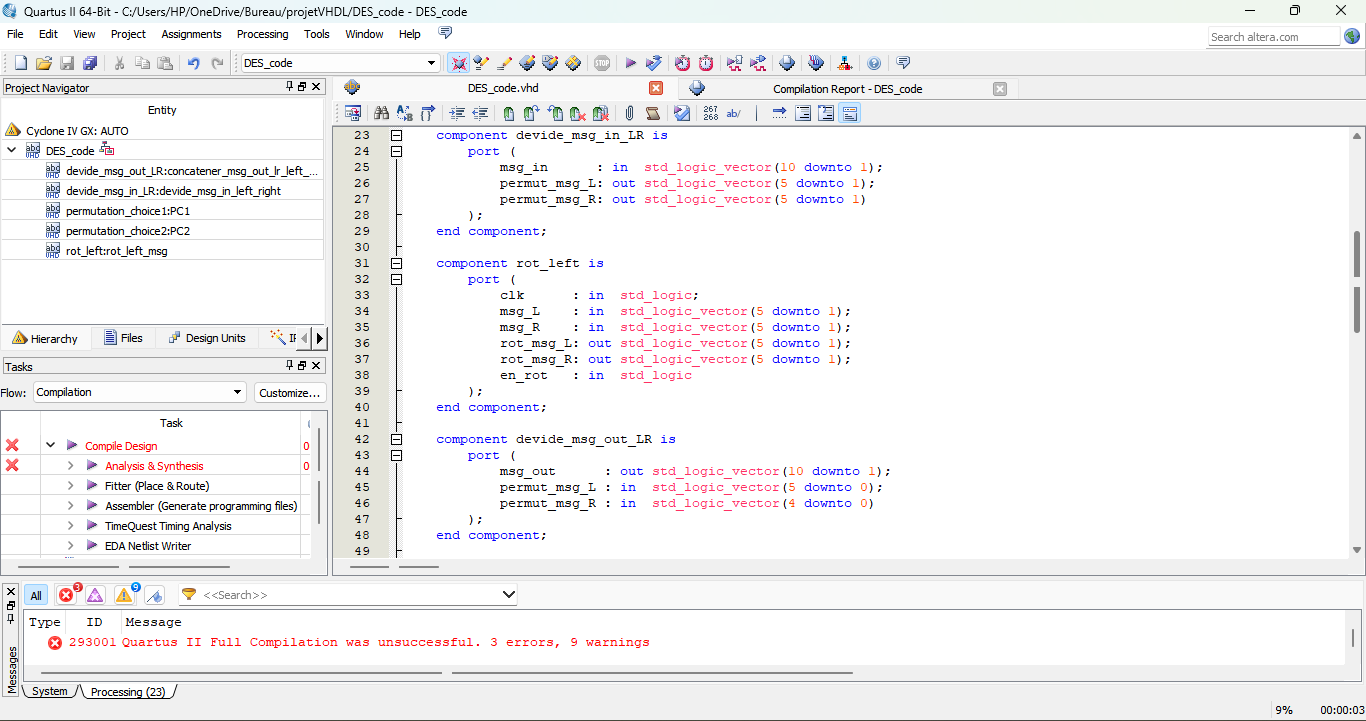
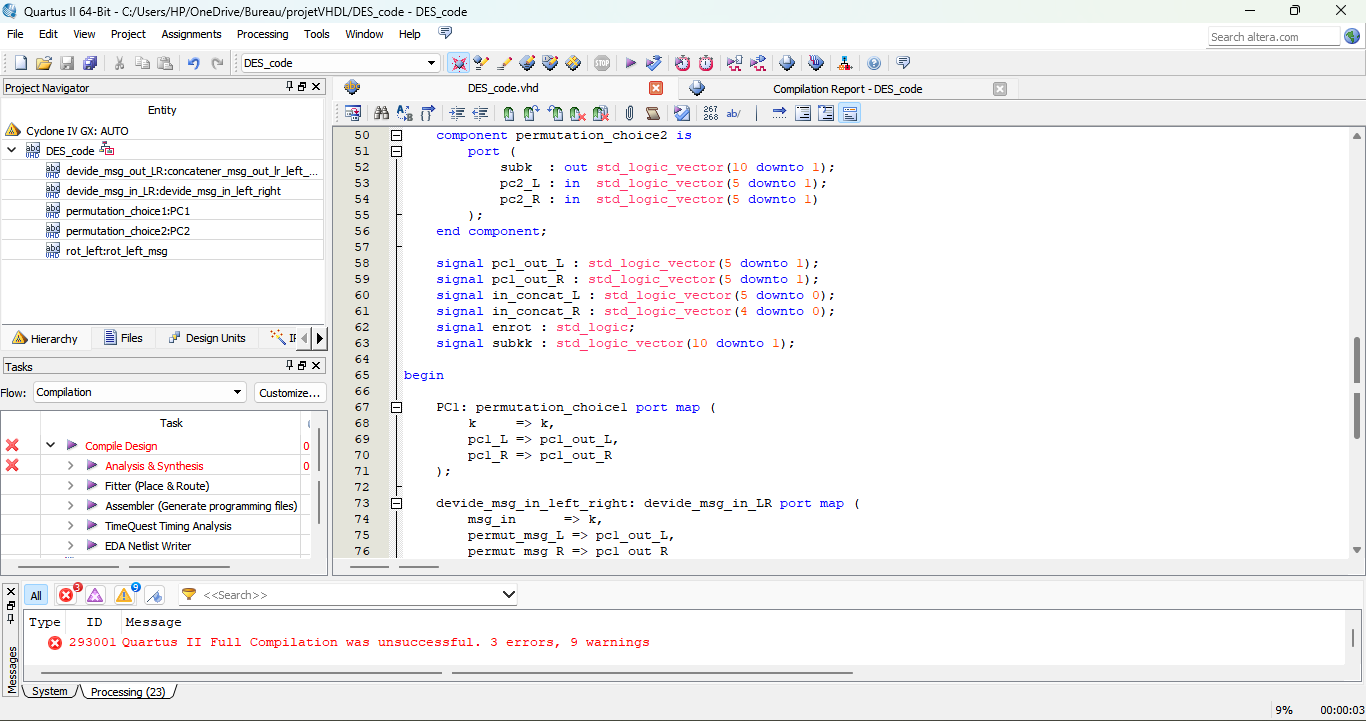
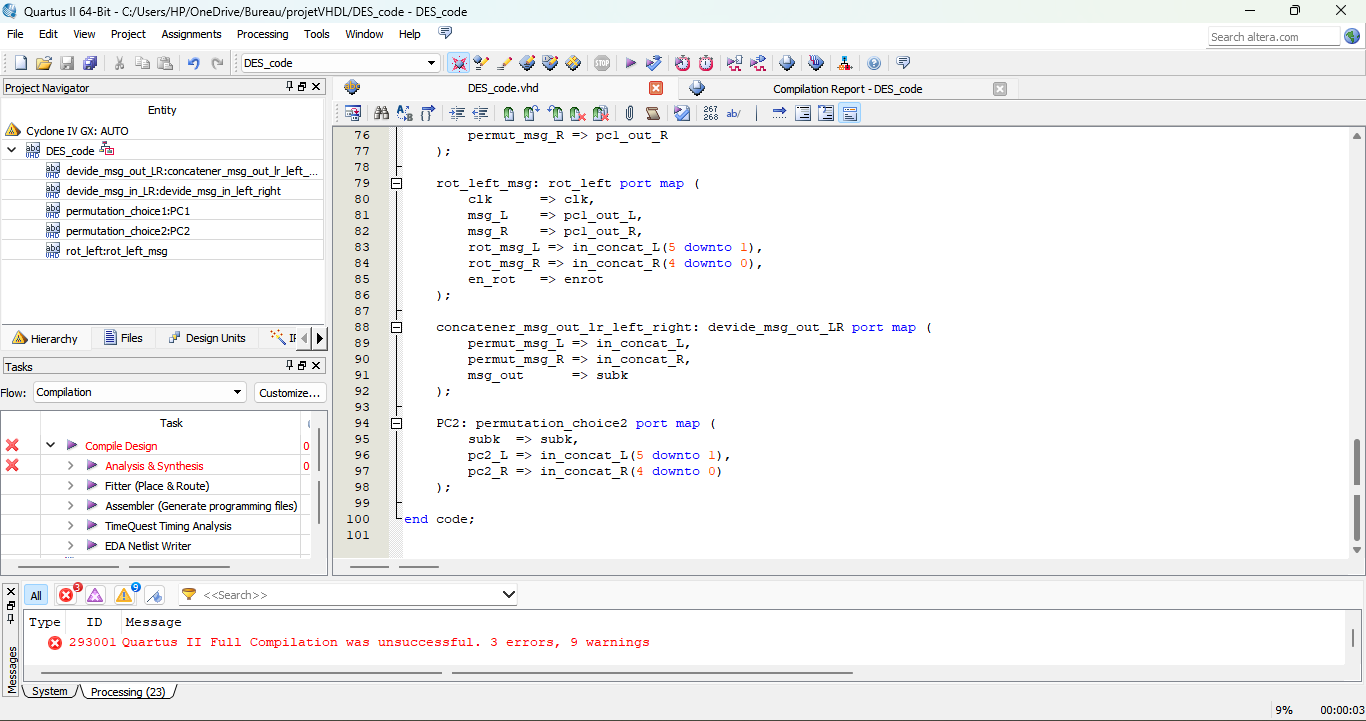
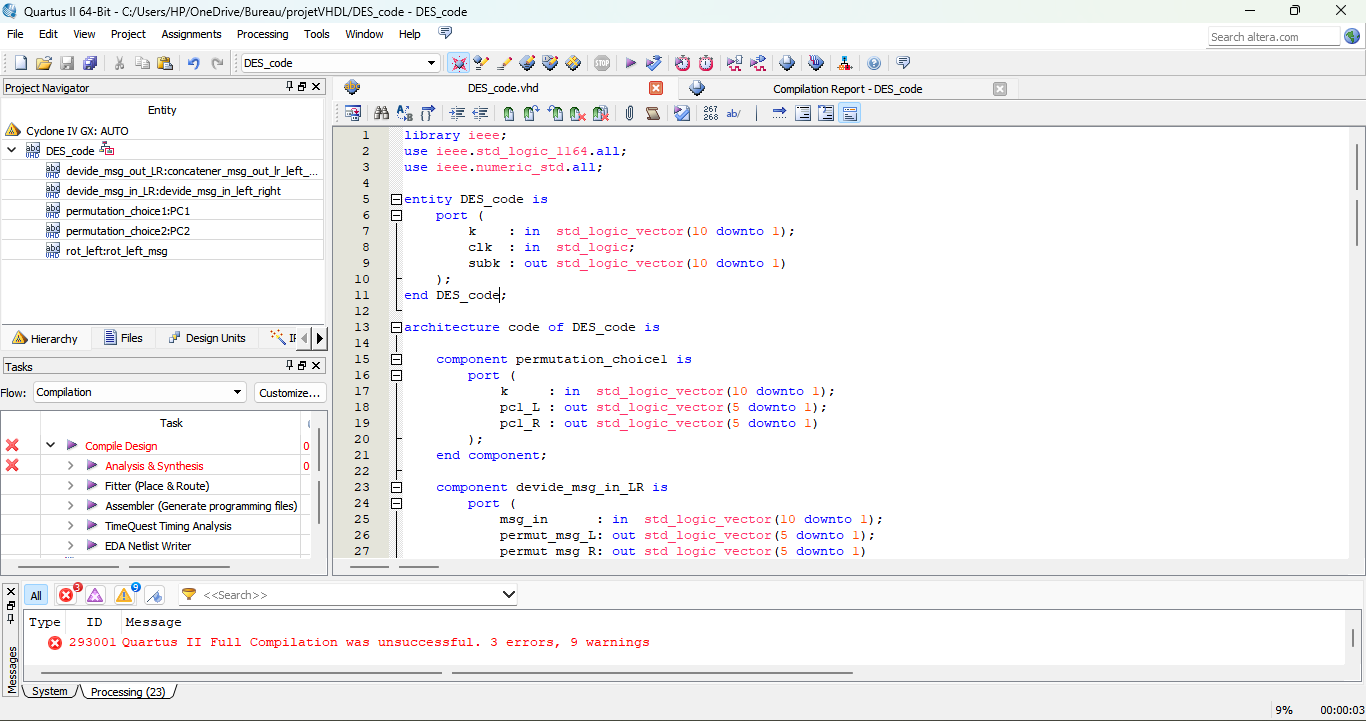
****

* This code defines an entity devide\_msg\_out\_LR that combines two input vectors permut\_msg\_L and permut\_msg\_R into a single vector msg\_out, and then permutes its bits according to a specified pattern. This entity is used to reorder and output the bits from the input vectors in a specific arrangement.

****

****

* The VHDL code implements the DES (Data Encryption Standard) algorithm. It instantiates several components like permutation modules (`permutation\_choice1` and `permutation\_choice2`), bit manipulation modules (`devide\_msg\_in\_LR` and `devide\_msg\_out\_LR`), and a bit rotation module (`rot\_left`). These components are interconnected to perform key scheduling and encryption operations according to the DES algorithm.

****

**Conclusion :**

In conclusion, the provided VHDL code segments collectively form a portion of an EDS (Encryption and Decryption System) implemented in VHDL. The code segments represent various components and functionalities necessary for the implementation of the Data Encryption Standard (DES) algorithm. These components include key scheduling, bit permutation, bit manipulation, and bit rotation, all of which are fundamental operations within the DES algorithm.

When integrated together, these components contribute to the overall encryption and decryption process within the EDS. By following the DES algorithm specifications and utilizing these VHDL implementations, the EDS can securely encrypt and decrypt data, ensuring confidentiality and integrity in data transmission and storage applications.

Finally, we would like to extend our deepest gratitude to Professor **EL** **MOUMNI Soufiane** for the opportunity to have worked on this interesting project and under their their mentorship.