Project Report

Lightweight Cryptographic Cipher Bank for Resource Constrained Devices and Secure IoT

Submitted By-
Gaurav Bansal
B.Tech CSE (2017-21)
IIT Jammu
Intern, DSCI (Jun'20 – Aug'20)

Project Head- Mr. Vinayak Godse
Mentor- Ms. Shravani Shahapure

Acknowledgement

I would like to thank DSCI for giving me this internship opportunity. It was a great chance for my learning and professional development. I am using this opportunity to express my deepest gratitude and special thanks to the Vice President of DSCI, Mr. Vinayak Godse who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training. I express my deepest thanks to Ms. Shravani Shahapure for being my mentor and giving necessary advices and arranging all facilities to make my life easier.

I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

Contents

- Introduction
- Literature Survey of Ciphers
- Selected Ciphers Overview
- Hardware Implementation Details
- Individual ciphers designs
- Cipher Bank designs
- Extra Key Embedding Rounds Description and Designs
- Simulation Details and Results
- Performance Analysis of Designs
- Conclusion

Introduction

Internet of Things (IoT) is one of the fastest emerging areas in today’s world. Huge number of industries as well as researchers are coming up with new technologies and applications of IoT everyday. Projects like “Smart Vehicles”, “Smart Cars”, “Smart Cities” are in talks everywhere. A lot of resource constrained devices such as “RFID tags” are being extensively used in tracking, merchandise and what not. But the main problems with these devices are their limited Area, Power and Memory which makes it difficult to fit existing ciphers on these to provide security.

In recent years, a lot of “Lightweight Cryptographic Ciphers” have been designed for such kind of devices. This project aims to efficiently integrate multiple such kind of ciphers into one **Cipher Bank**. Not only this cipher bank is highly area optimized but it provides user a range of ciphers to select for encryption. Moreover, this cipher bank comes with two more variants one of which support extra level of security and other supports double encryption.

Literature Survey of Ciphers

This is the most important step of the project as the bank will consist of the ciphers we select in this step. There are two major criteria for the cipher selection:

- Minimum area possible of design.
- Good level of security.

Extensive literature survey and study has been done to select the ciphers. As there are two broad categories of lightweight ciphers: Block ciphers and Stream ciphers, both kinds of ciphers have been studied and the results are stored in this repository. Some of the studied ciphers include Hummingbird, Present, Simon, ANU, LED, Piccolo, Grain, Tea, DESXL etc.

For this project, 3 Block ciphers are selected: PRESENT-128, SIMON-64/128 and ANU-II.

Selected Ciphers Overview

Expected Bank Size: < 3850 GE

Expected Average Power:

32 mW on 10 MHz clock frequency

References:

1. PRESENT
2. SIMON
3. ANU-II

Cipher	Key Size	Block Size	Rounds	Approx. Area (GE)	Resistant Against
PRESENT	128 bits	64 bits	32	1886	Linear and Differential Cryptanalysis, Algebraic, Structural, Key Schedule Attacks.
SIMON	128 bits	64 bits	44	958	Cryptanalytical Techniques and other practical attacks.
ANU-II	128 bits	64 bits	25	1010	Cryptanalytical Techniques, Biclique Attack

PRESENT-128

- State of the Art lightweight Block cipher.
- SPN type network.
- Requires 32 clock cycles per encryption.

```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE,$K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,$K_{32}$)
```

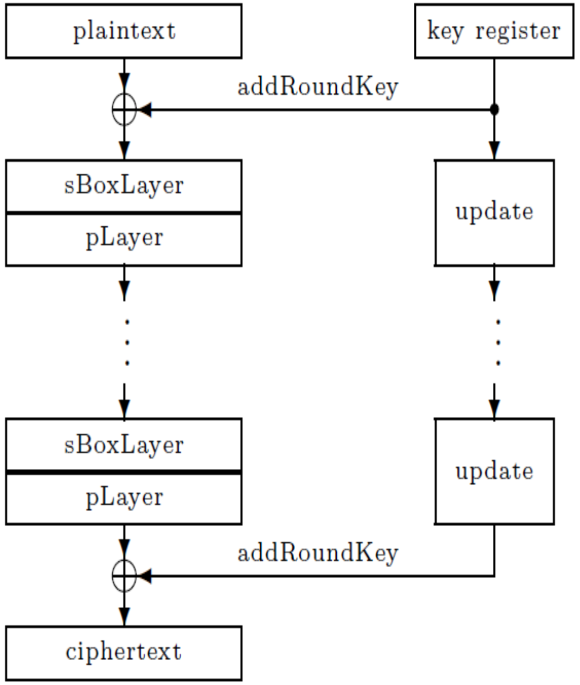


Fig. 1. A top-level algorithmic description of PRESENT.

SIMON-64/128

- Developed by NSA, one of the best area optimized and secure block cipher.
- Feistel Type network.
- Requires 44 clock cycles per encryption.

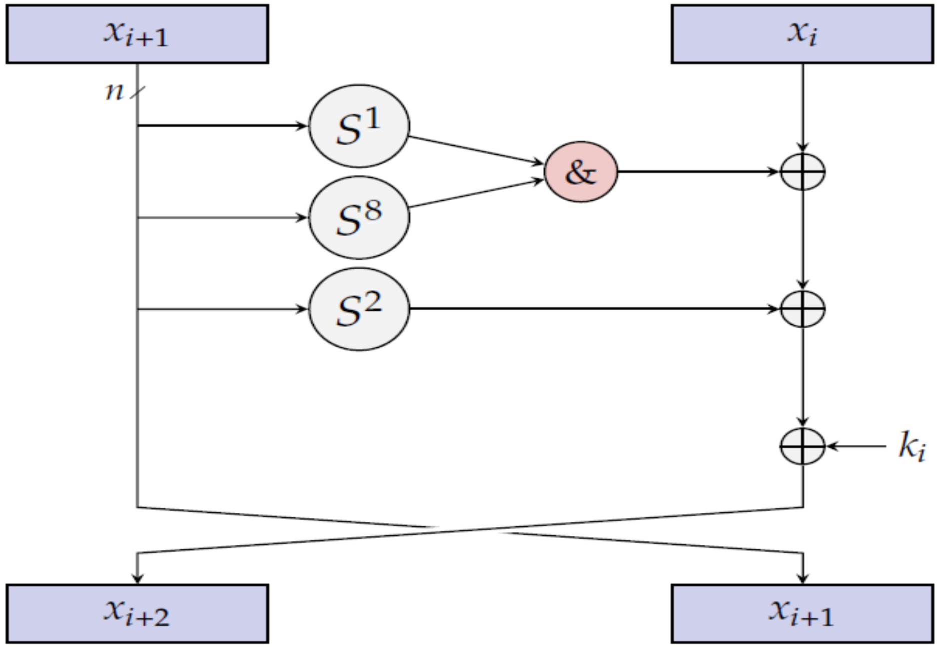


Figure 3.1: Feistel stepping of the SIMON round function.

ANU-II

- Recently developed area optimized block cipher with enhanced security.
- Feistel type network.
- Requires 25 clock cycles per encryption.

III. ANU-II A Block Cipher

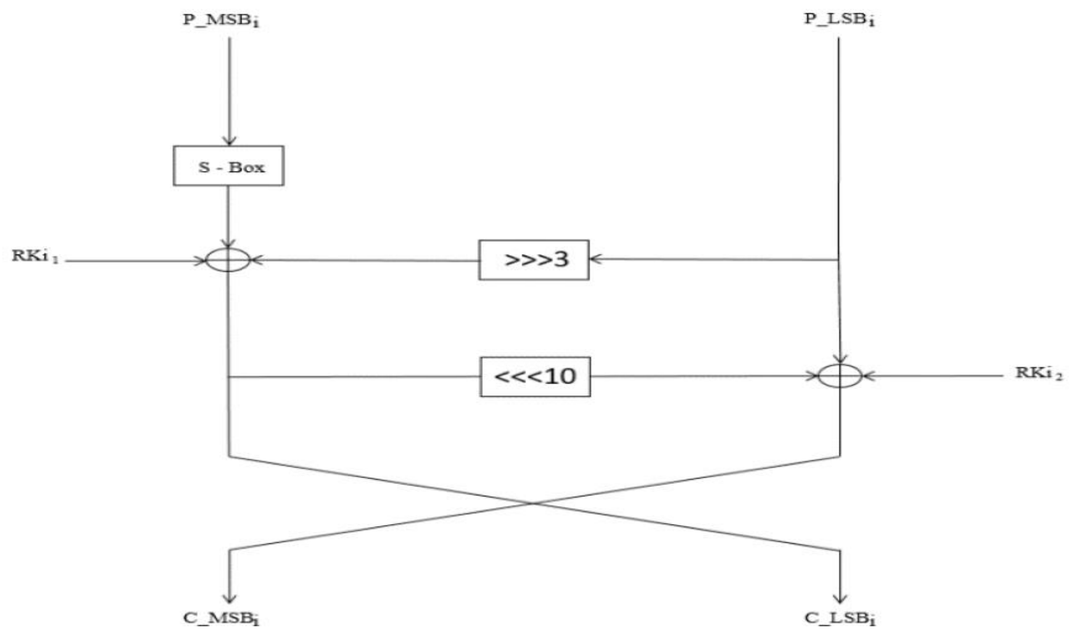


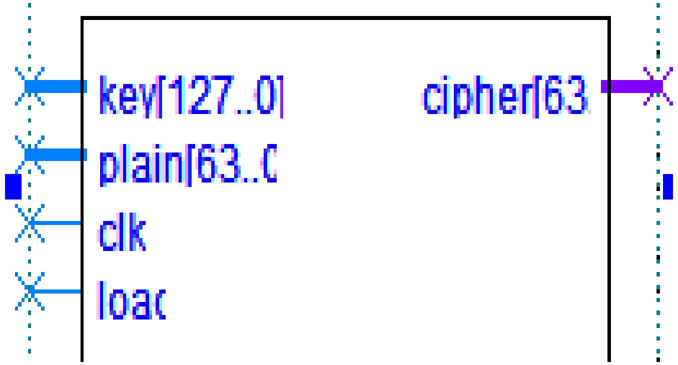
Fig. 1: A Block Cipher ANU-II

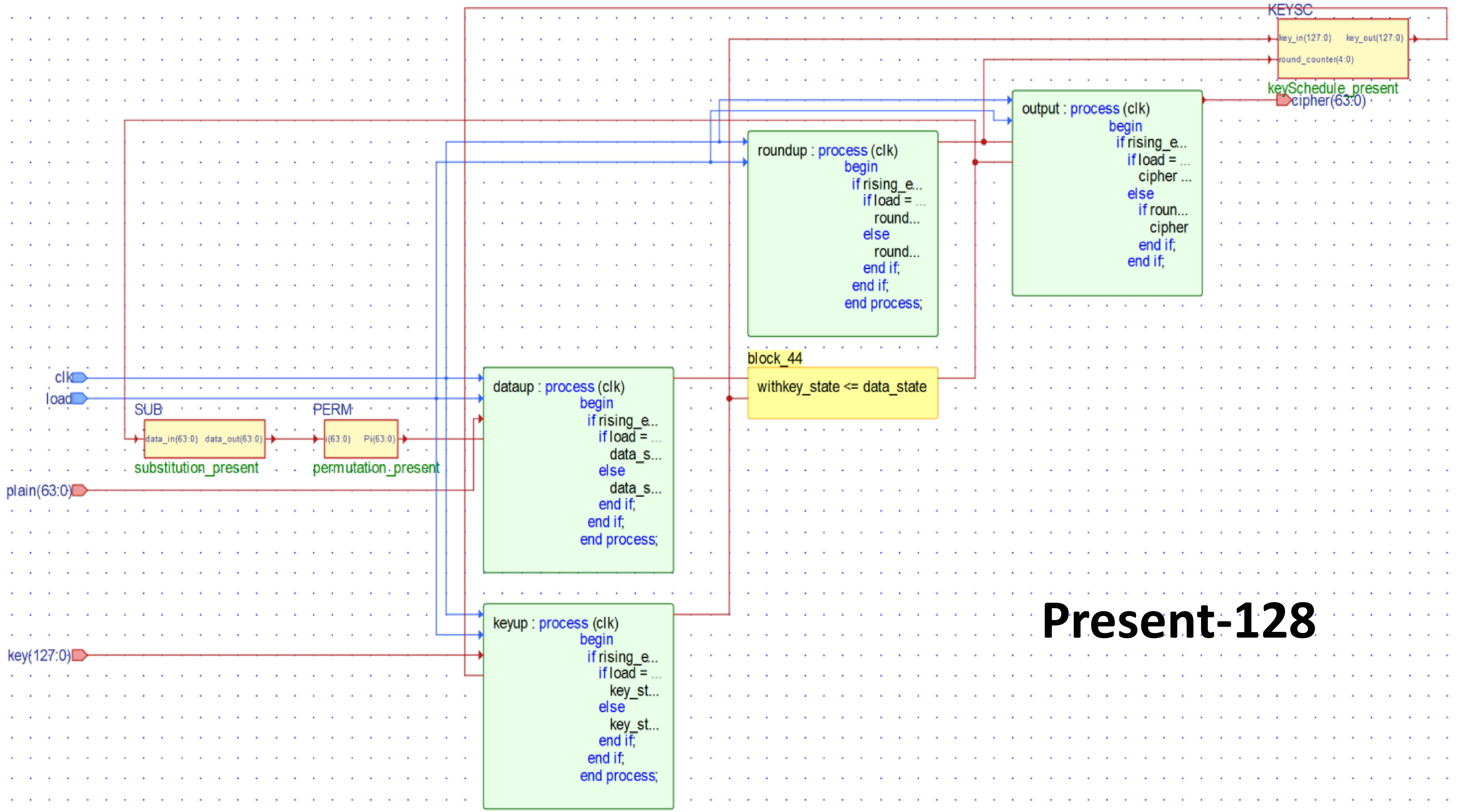
Hardware Implementation

- **VHDL** is used as the implementation language for all the designs.
- All designs consists of the processes which are Clocked.
- There are 4 processes in each design as follows:
  - Data Update: This process updates the data to be encrypted after each clock cycle or round with initial value as the ‘plaintext’ given by the user.
  - Key Update: This process updates the key to be used for encryption after each round with initial value as the ‘key’ given by the user. The key updates according to the key scheduling algorithms.
  - Round Update: This process increments the round counter by 1 after each clock cycle with initial value as ‘0’or ‘1’ depending upon the cipher. It’s value marks the completion of encryption.
  - Output: This process updates the output. Initially, the output is set to ‘0’ whenever new data is loaded and remains 0 till encryption. Once encryption is completed, the output is updated.

Individual Ciphers Designs

- key: 128 bit key supplied by the user.
- plain: 64 bit plaintext supplied by the user.
- clk: clock used in processes.
- load: This bit is set to ‘1’ when new data comes in and after loading it is set to ‘0’.
- cipher: 64 bit output encrypted plaintext.
- **Active-HDL Student Edition** is used to compile and generate a high-level circuit design of the implementation.





```
roundup : process (clk)
          begin
           if rising_e...
            if load = ...
             round...
            else
             round...
            end if;
           end if;
          end process;
```

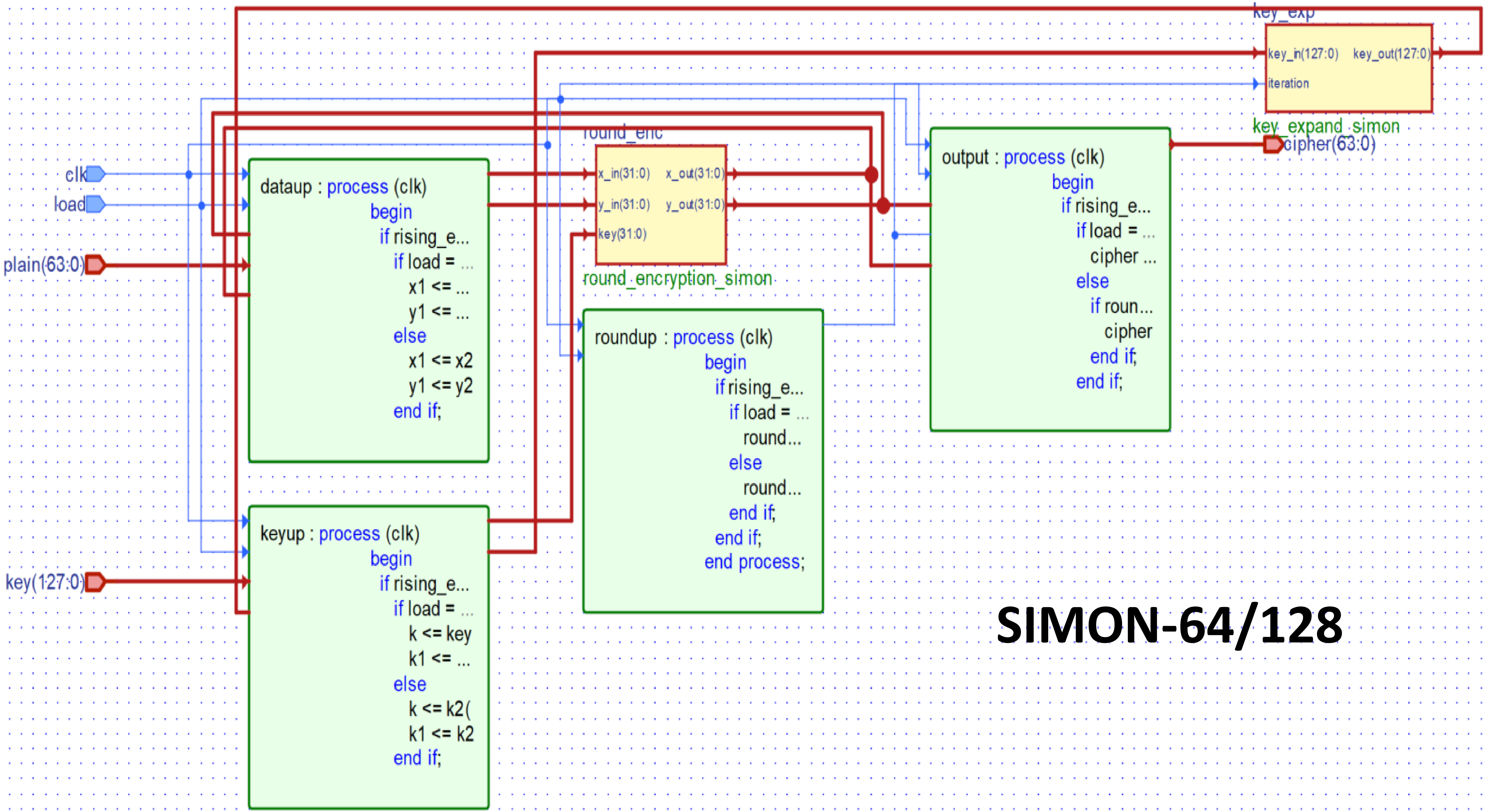
```
output : process (clk)
          begin
           if rising_e...
            if load = ...
             cipher ...
            else
             if roun...
              cipher
             end if;
            end if;
```

```
withkey_state <= data_state
```

```
dataup : process (clk)
          begin
           if rising_e...
            if load = ...
             data_s...
            else
             data_s...
            end if;
           end if;
          end process;
```

```
keyup : process (clk)
          begin
           if rising_e...
            if load = ...
             key_st...
            else
             key_st...
            end if;
           end if;
          end process;
```

Present-128



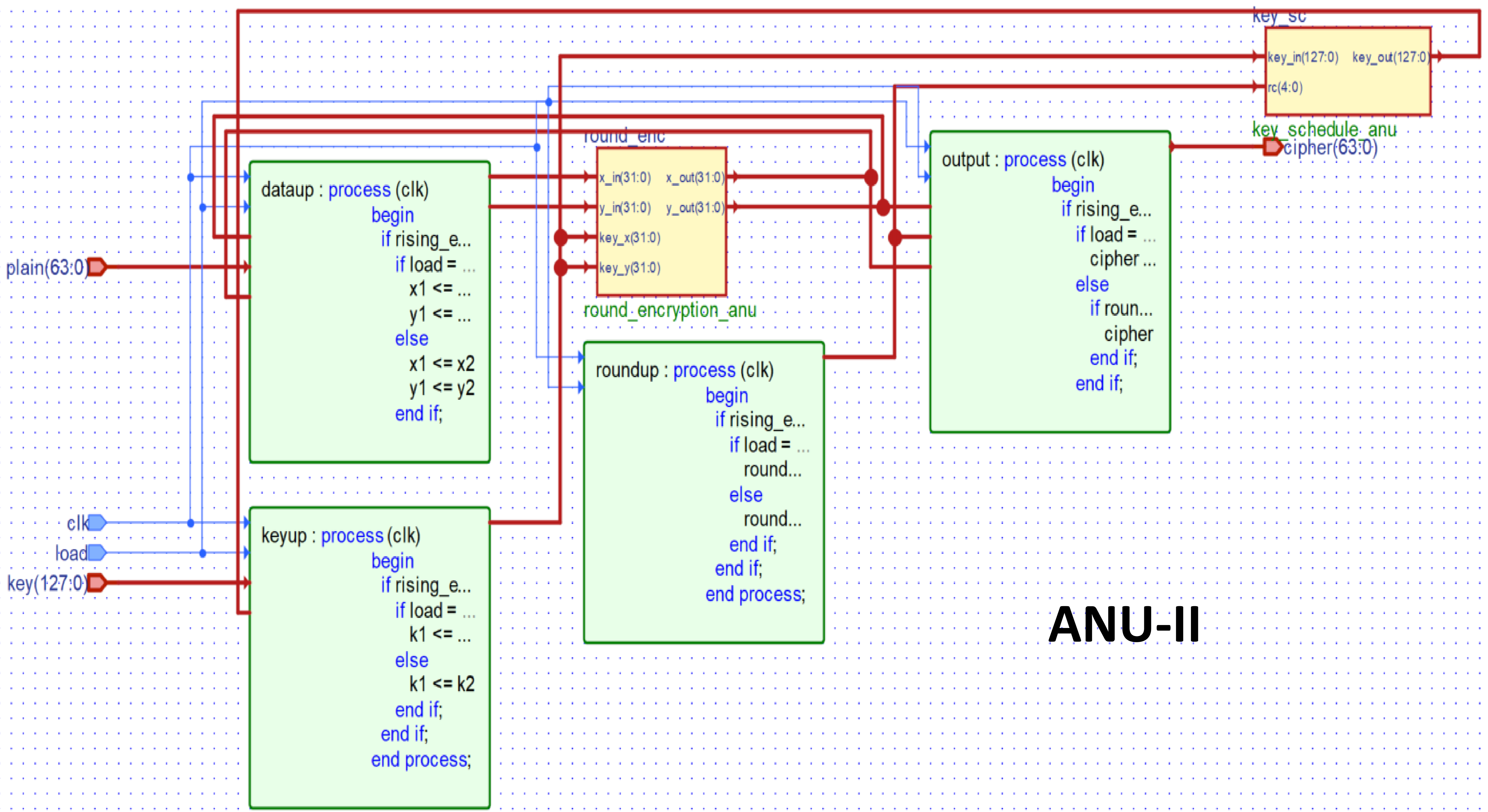
```
dataup : process (clk)
        begin
         if rising_e...
          if load = ...
           x1 <= ...
           y1 <= ...
          else
           x1 <= x2
           y1 <= y2
          end if;
```

```
keyup : process (clk)
        begin
         if rising_e...
          if load = ...
           k <= key
           k1 <= ...
          else
           k <= k2(
           k1 <= k2
          end if;
```

```
roundup : process (clk)
          begin
           if rising_e...
            if load = ...
             round...
            else
             round...
            end if;
           end if;
          end process;
```

```
output : process (clk)
         begin
          if rising_e...
           if load = ...
            cipher ...
           else
            if roun...
             cipher
            end if;
           end if;
```

SIMON-64/128



```
dataup : process (clk)
    begin
     if rising_e...
      if load = ...
        x1 <= ...
        y1 <= ...
      else
        x1 <= x2
        y1 <= y2
      end if;
```

```
keyup : process (clk)
    begin
     if rising_e...
      if load = ...
        k1 <= ...
      else
        k1 <= k2
      end if;
     end if;
    end process;
```

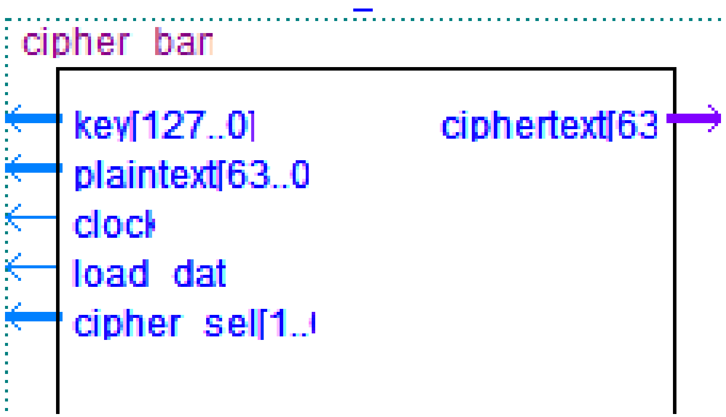
```
roundup : process (clk)
    begin
     if rising_e...
      if load = ...
        round...
      else
        round...
      end if;
     end if;
    end process;
```

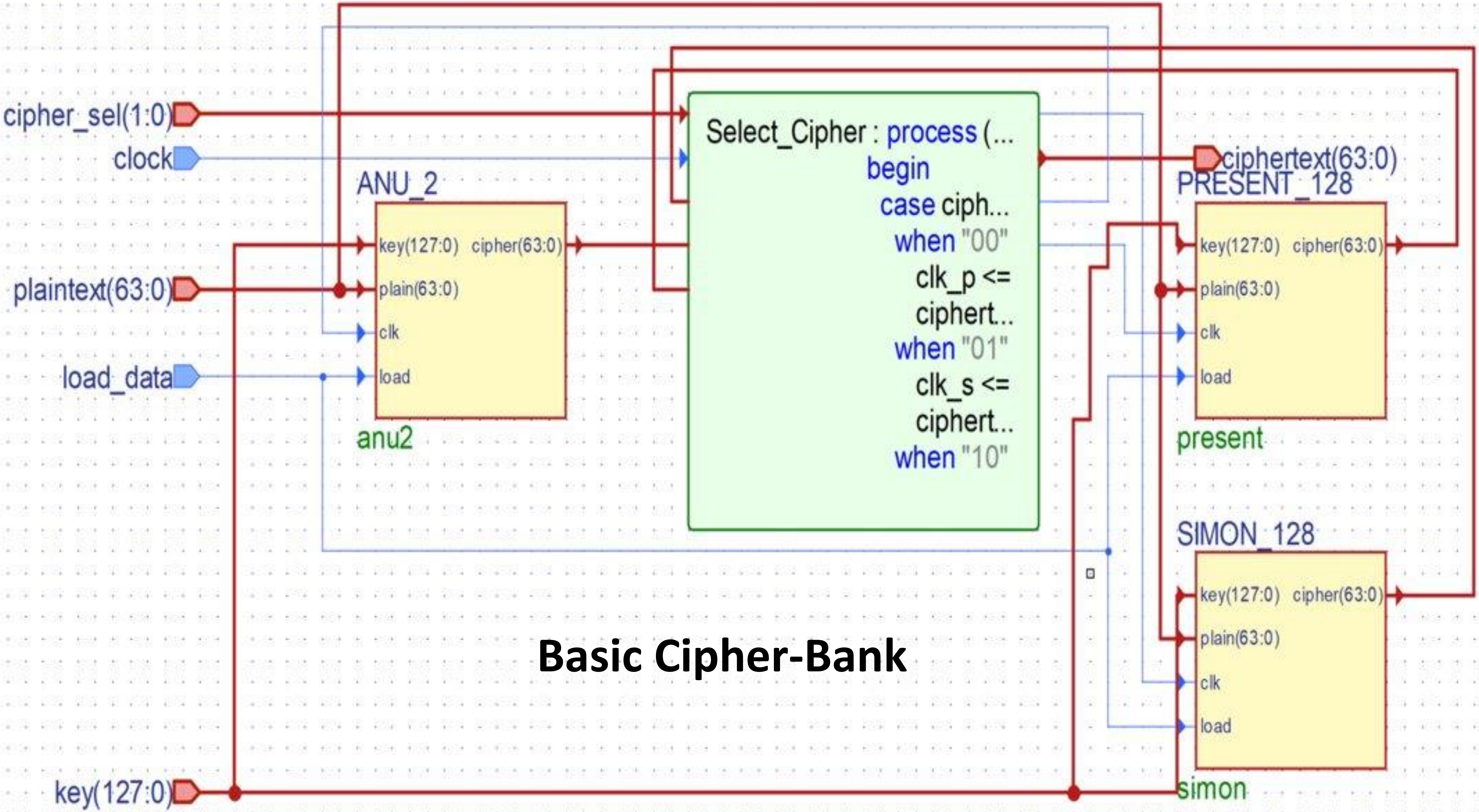
```
output : process (clk)
    begin
     if rising_e...
      if load = ...
        cipher ...
      else
        if roun...
          cipher
        end if;
      end if;
```

ANU-II

Cipher Bank Designs

- key: 128 bit key supplied by the user.
- plaintext: 64 bit plaintext supplied by the user.
- clock: clock used in processes.
- load_data: This bit is set to ‘1’ when new data comes in and after loading it is set to ‘0’.
- cipher_sel: 2 bit input used to select cipher. ‘00’ for Present, ‘01’ for Simon and ’10’ for ANU-II.
- ciphertext: 64 bit output encrypted plaintext.
- In the cascaded version, two keys are used: key_pr for Present and key_anu for ANU-II encryption.

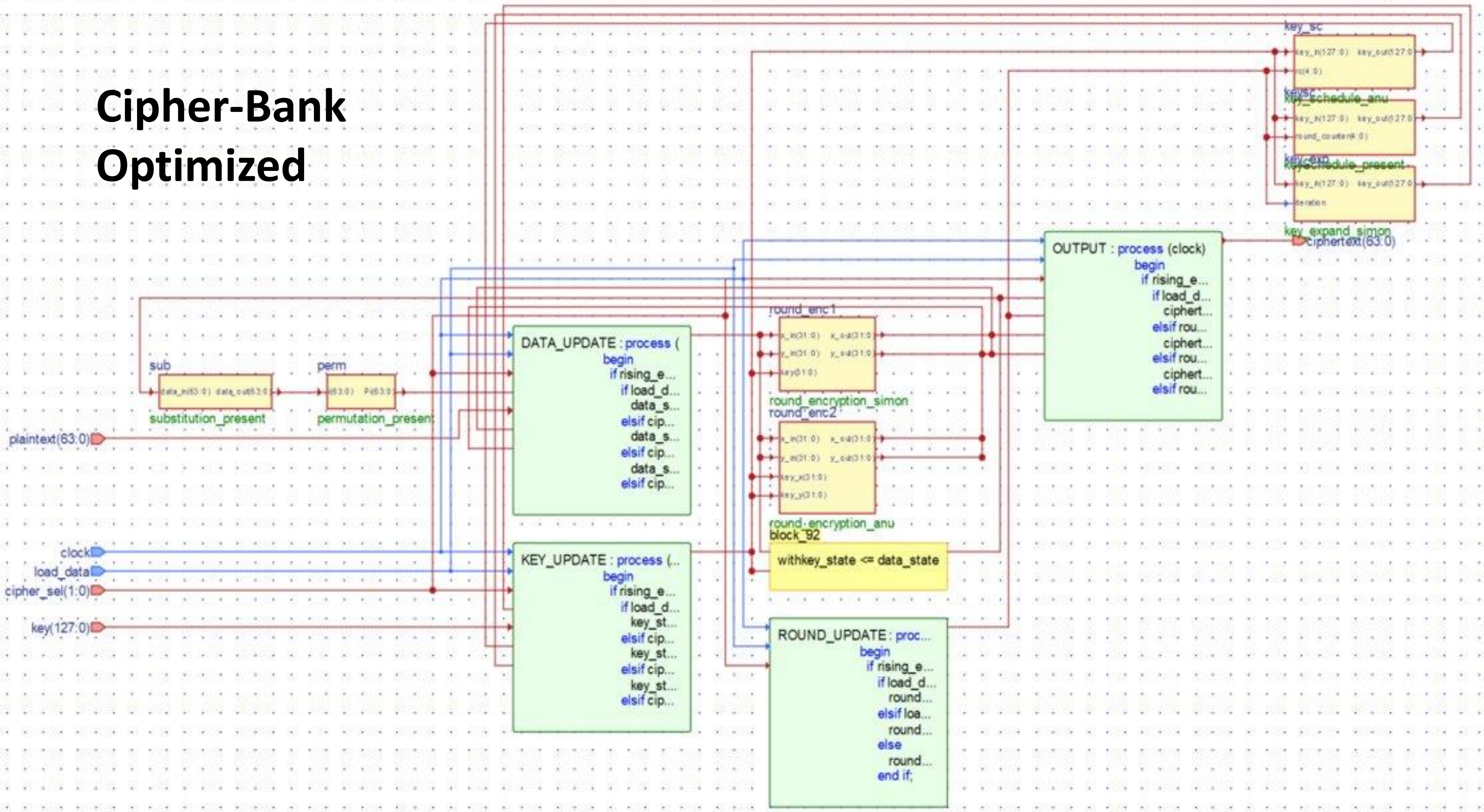




```
Select_Cipher : process (...
                  begin
                   case ciph...
                    when "00"
                      clk_p <=
                      ciphert...
                    when "01"
                      clk_s <=
                      ciphert...
                    when "10"
```

Basic Cipher-Bank

Cipher-Bank Optimized



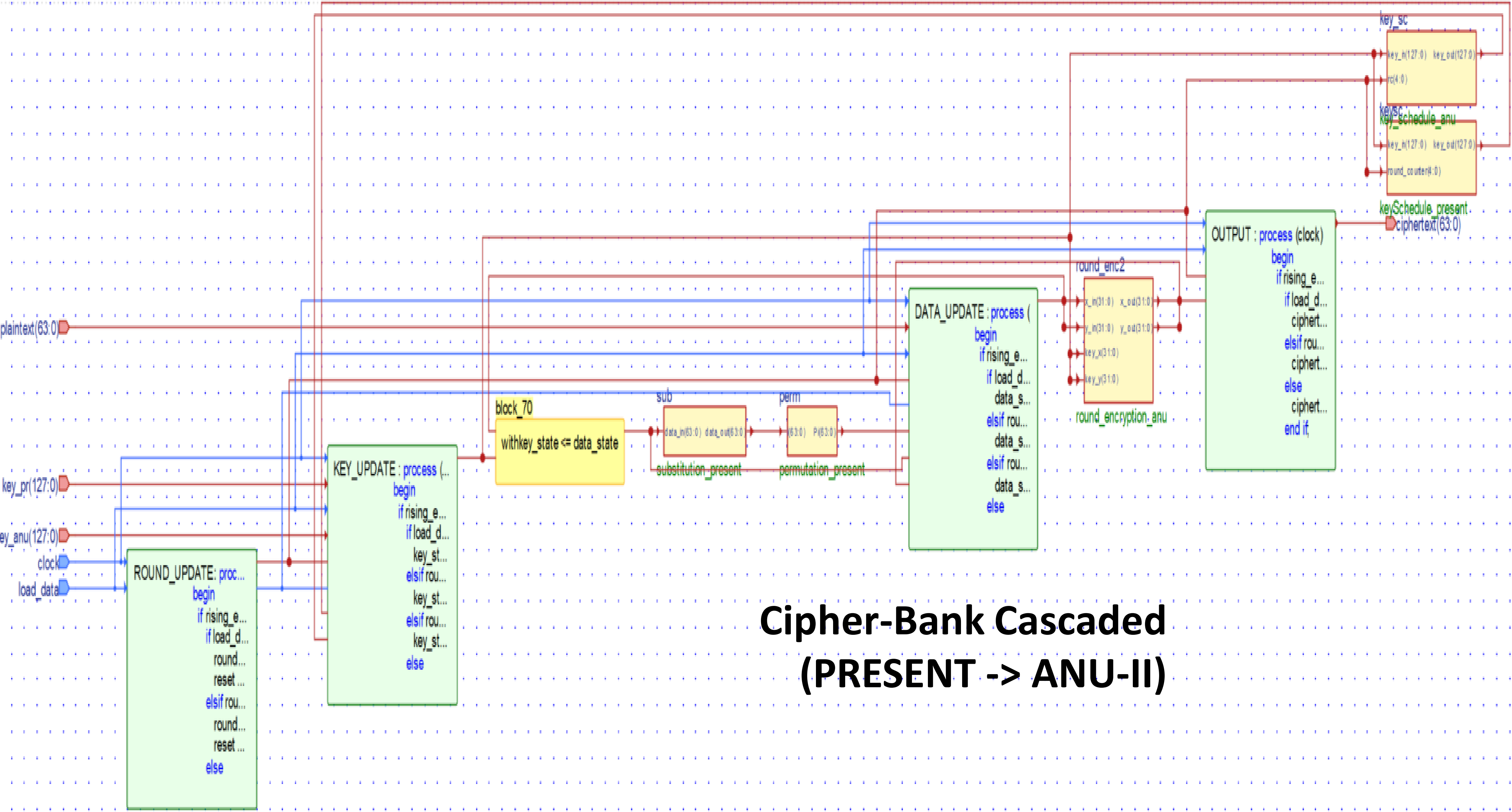
```
OUTPUT : process (clock)
        begin
          if rising_e...
            if load_d...
              ciphert...
            elsif rou...
              ciphert...
            elsif rou...
              ciphert...
            elsif rou...
```

```
DATA_UPDATE : process (
        begin
          if rising_e...
            if load_d...
              data_s...
            elsif cip...
              data_s...
            elsif cip...
              data_s...
            elsif cip...
```

```
withkey_state <= data_state
```

```
KEY_UPDATE : process (...
        begin
          if rising_e...
            if load_d...
              key_st...
            elsif cip...
              key_st...
            elsif cip...
              key_st...
            elsif cip...
```

```
ROUND_UPDATE : proc...
        begin
          if rising_e...
            if load_d...
              round...
            elsif loa...
              round...
            else
              round...
            end if;
```



Cipher-Bank Cascaded (PRESENT -> ANU-II)

Extra Key Embedding Rounds

K -128 bits

P - 64 bits

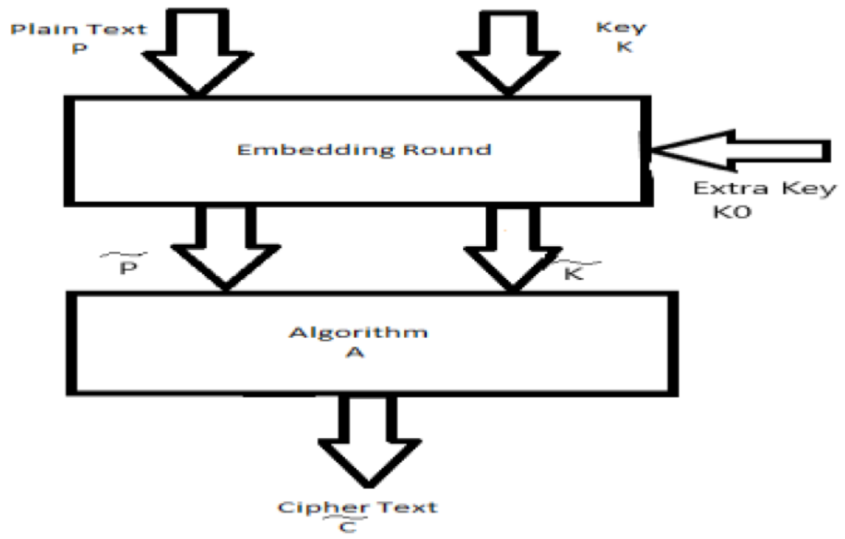
K_0 - 64 bits

K_1, K_2 - 64 bits

P_1, P_2 - 32 bits

For f_3 and f_4 – k_0 is divided in two equal parts of 32 bits.

Function	Description
f_1	Conjugation of Bits
f_2	10 bits Right Circular Shift
f_3	10 bits Left Circular Shift
f_4	Mirror Reflection of bits

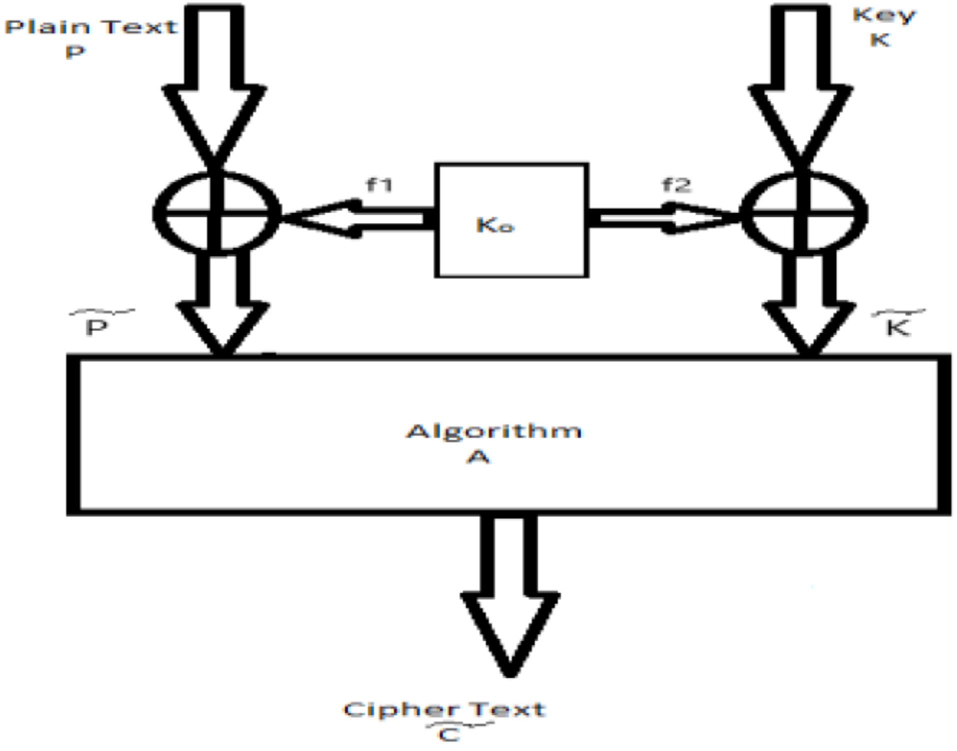


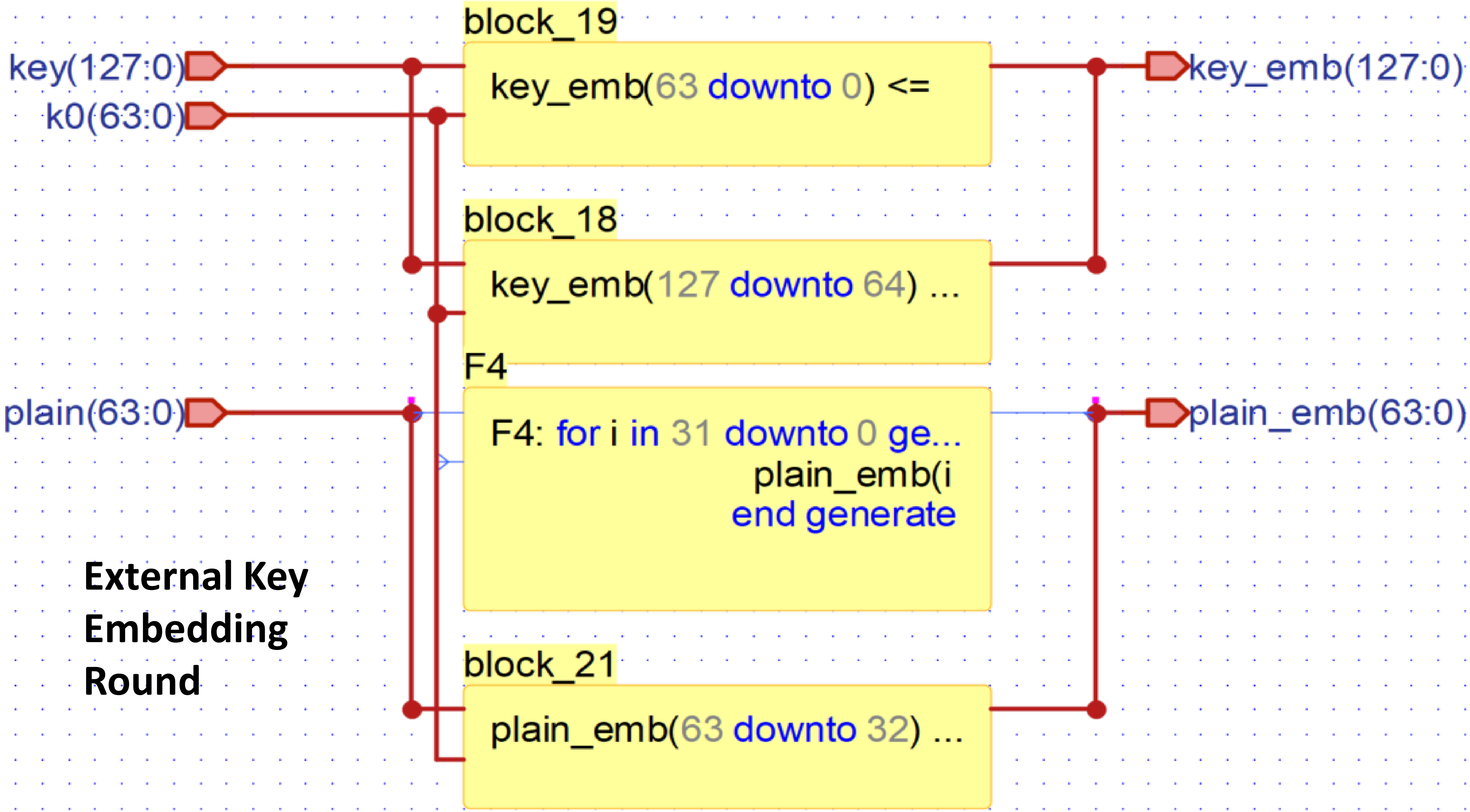
$$\check{K} = (K_1 \oplus f_1 (K_0), K_2 \oplus f_2(K_0)$$

$$\check{P} = (P_1 \oplus f_3(K_0), P_2 \oplus f_4(K_0))$$

External Key Embedding Round

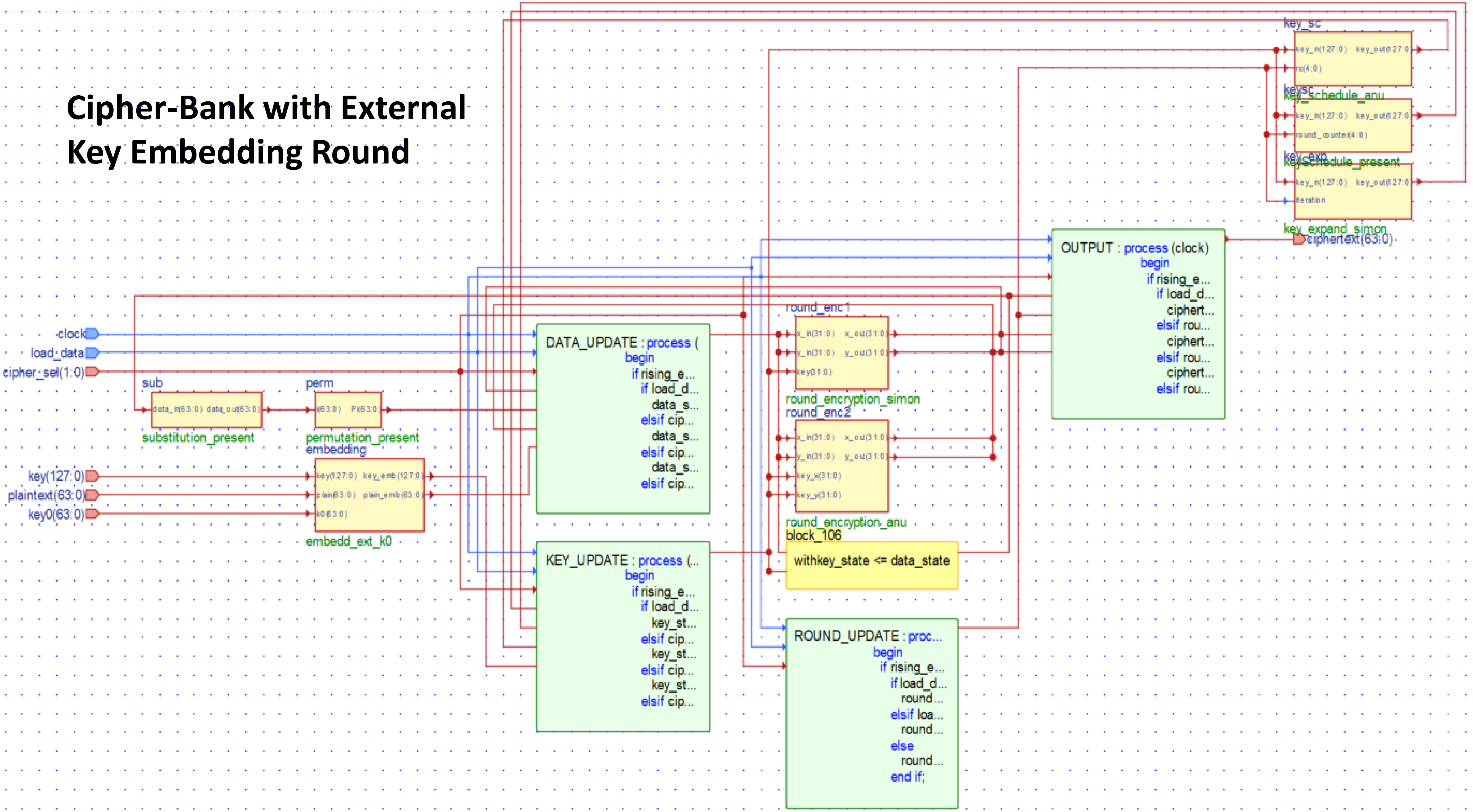
Round Key K_0 is provided by the user.





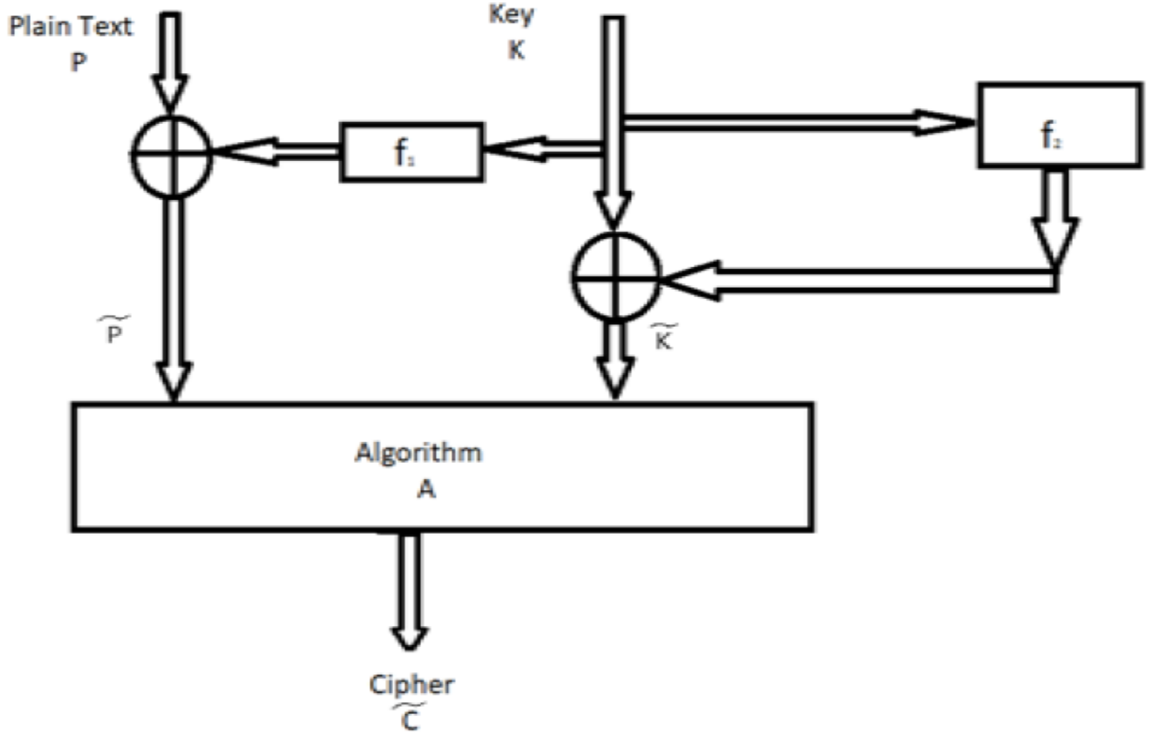
External Key Embedding Round

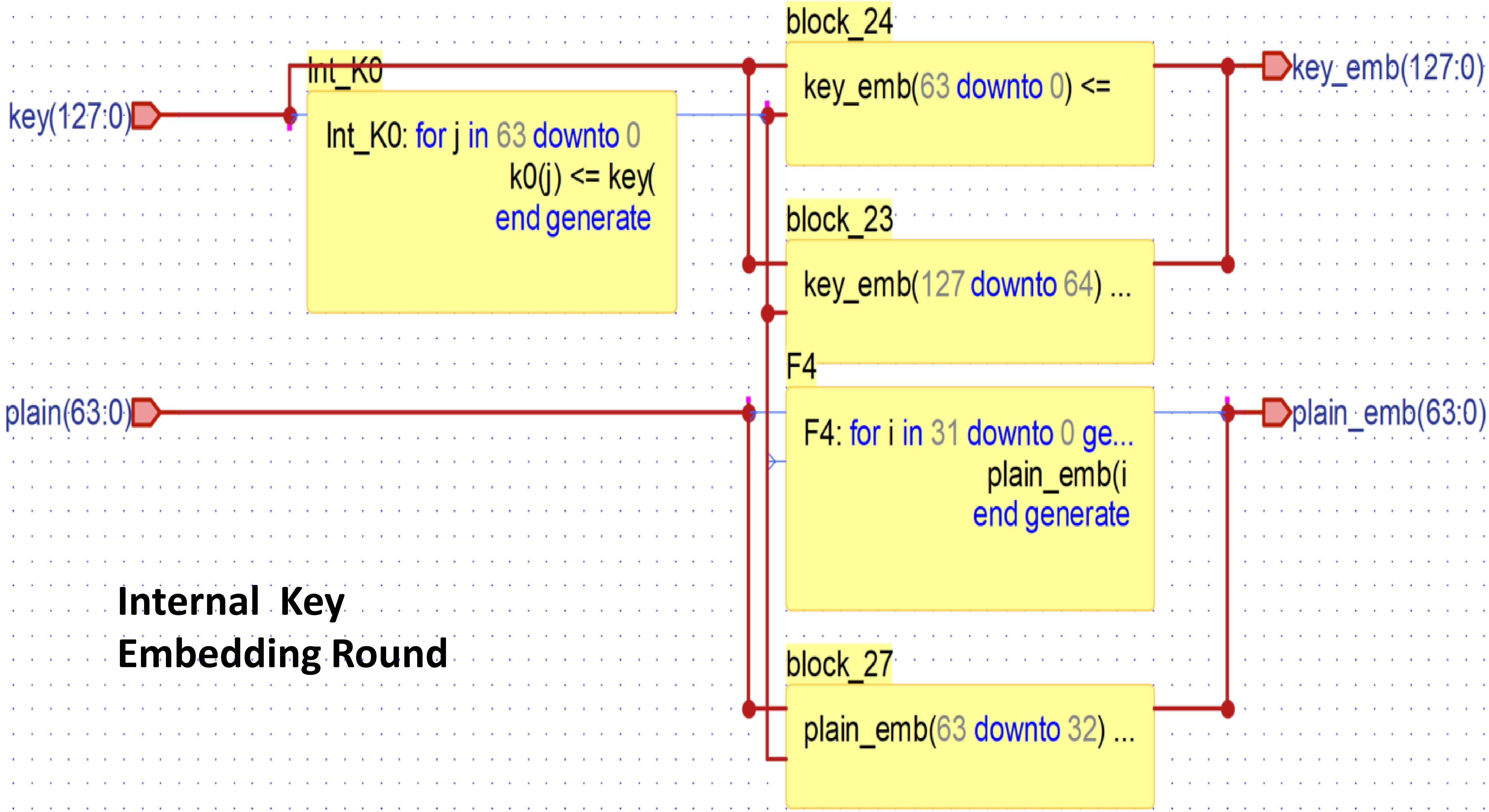
Cipher-Bank with External Key Embedding Round



Internal Key Embedding Round

Round Key K_0 is extracted from key K by taking its **odd position** bits.





```
Int_K0: for j in 63 downto 0
        k0(j) <= key(
        end generate
```

```
key_emb(63 downto 0) <=
```

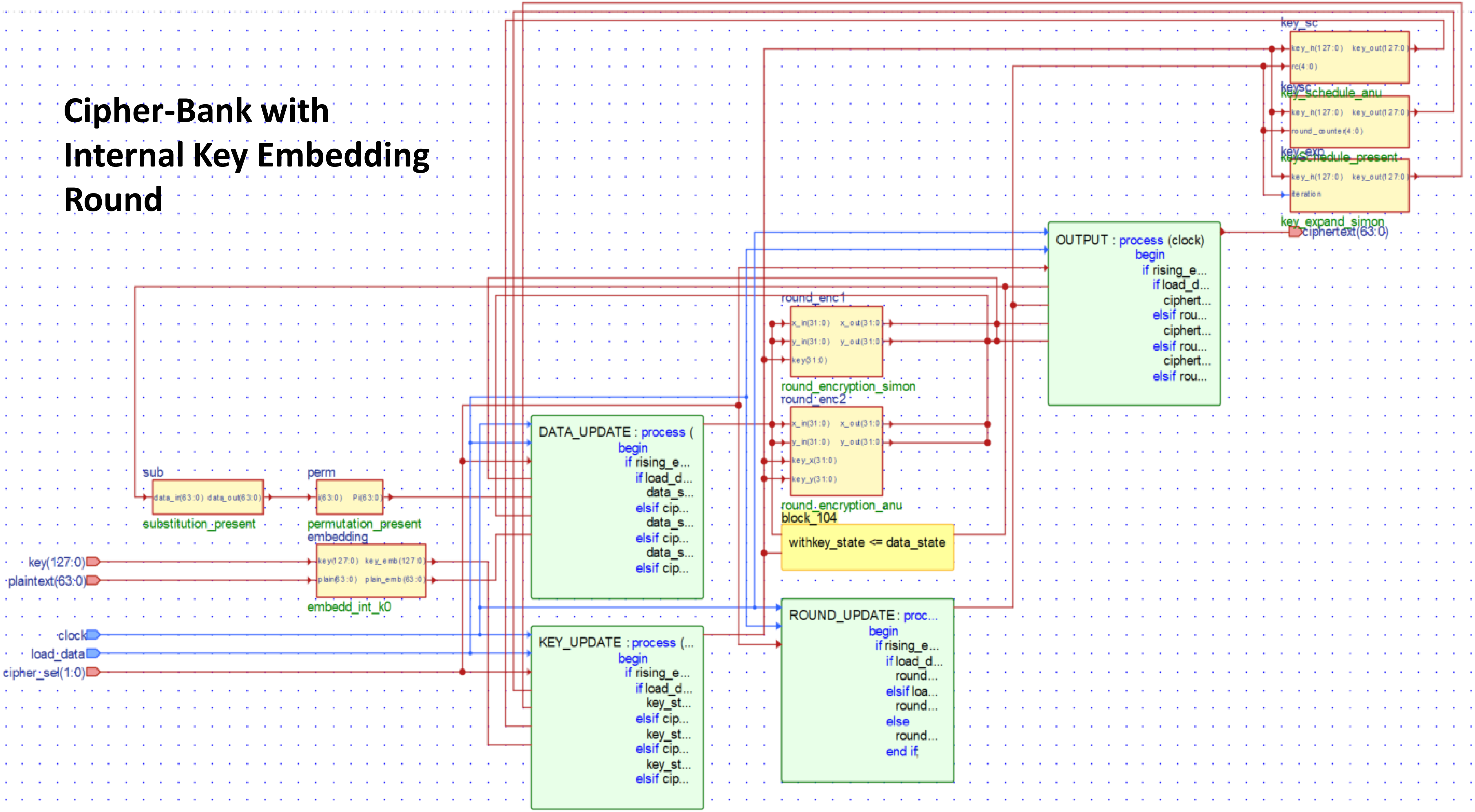
```
key_emb(127 downto 64) ...
```

```
F4: for i in 31 downto 0 ge...
        plain_emb(i
        end generate
```

```
plain_emb(63 downto 32) ...
```

Internal Key Embedding Round

Cipher-Bank with Internal Key Embedding Round



Simulation and Correctness Verification

- **Active-HDL Student Edition** is used for simulation and verification of the implemented VHDL designs.
- Testbenches are implemented in VHDL for all the designs verification.
- 100 MHz frequency clock is used for cipher encryption.
- Testcases are generated for all the 3 ciphers through High Level Language code.
- For all the ciphers, **Python** is used as HLL for the software implementation and testcase generation.
- Some basic testcases are taken directly from the research papers from cipher designers to test software implementation.

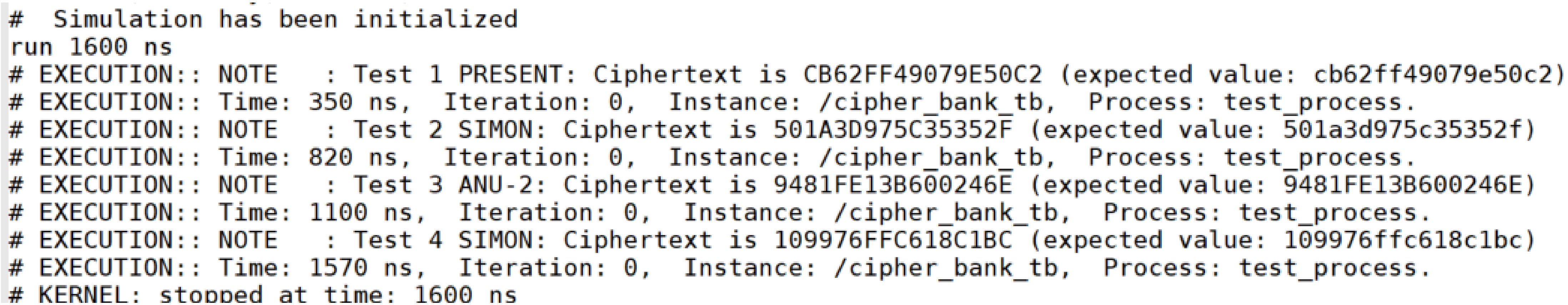
Simulation Results

Time Taken per 64 bits encryption on 100MHz clock

PRESENT-128:   340 ns

SIMON-64/128: 460 ns

ANU-II: 270 ns



```
#  Simulation has been initialized
run 1600 ns
# EXECUTION:: NOTE   : Test 1 PRESENT: Ciphertext is CB62FF49079E50C2 (expected value: cb62ff49079e50c2)
# EXECUTION:: Time: 350 ns,  Iteration: 0,  Instance: /cipher_bank_tb,  Process: test_process.
# EXECUTION:: NOTE   : Test 2 SIMON: Ciphertext is 501A3D975C35352F (expected value: 501a3d975c35352f)
# EXECUTION:: Time: 820 ns,  Iteration: 0,  Instance: /cipher_bank_tb,  Process: test_process.
# EXECUTION:: NOTE   : Test 3 ANU-2: Ciphertext is 9481FE13B600246E (expected value: 9481FE13B600246E)
# EXECUTION:: Time: 1100 ns,  Iteration: 0,  Instance: /cipher_bank_tb,  Process: test_process.
# EXECUTION:: NOTE   : Test 4 SIMON: Ciphertext is 109976FFC618C1BC (expected value: 109976ffc618c1bc)
# EXECUTION:: Time: 1570 ns,  Iteration: 0,  Instance: /cipher_bank_tb,  Process: test_process.
# KERNEL: stopped at time: 1600 ns
```

Performance Analysis of Designs

- Designs are primarily analysed on the basis of **Area** and **Power**.
- **Intel Quartus Prime Lite Edition** is used for synthesis and analysis.
- The analysis is targeted for **Intel MAX 10 FPGA** device family whose specifications are:
  - Device: MAX10 10M50DAF672C7G
  - Core Voltage: 1.2 V
  - Pin Count: 500
  - Junction Temperature: 0 - 85°C

  This same device is used for analysis of all the designs.
- A Synopsys Design Constraint(**.sdc**) file is created for synthesis with clock constraint of frequency 100 MHz or 10 ns Time Period.
- Area in GEs reported in the results is an approximation and other values are accurate reported by the tool.

Components Performance Results

Cipher	Area (LUT)	Approx. Area (GE)	Registers Used	Power (mW)	Total Thermal Power(mW)
PRESENT-128	400	1886	261	32.53	131
SIMON- 64/128	377	958	262	30.94	129.41
ANU-II	344	1010	261	33	131.48

Embedding Round	Area (LUT)	Approx. Area (GE)	Power (mW)	Total Thermal Power(mW)
External Key	193	580	1.79	100.14
Internal Key	193	580	1.57	99.91

Cipher Bank Performance Results

Cipher- Bank	Area (LUT)	Approx. Area (GE)	Registers Used	Power (mW)	Total Thermal Power(mW)
Basic Design	1313	3939	784	35.86	134.35
Optimized	**872**	2616	262	**36.73**	135.22
Cascaded	717	2151	263	41.9	140.41
With External Embedding	1066	3198	262	42	140.51
With Internal Embedding	1065	3195	262	38.77	137.27

Conclusion

As the project aims to design a lightweight cryptographic cipher bank, multiple variants of cipher bank is developed and presented in report. It started with the basic design which has been optimized in terms of area and hence comes a highly area optimized variant of the bank. The analysis results clearly shows that the area optimized version is **33.5% smaller** in area than the basic design. Then a cascaded version of the cipher bank is presented which is like a double encryption scheme. Then extra level of security is added to the bank by giving extra key embedding rounds. With a small increase in area, security gets enhanced from 2^{128} to 2^{192}.

This cipher bank can be further improved by adding more ciphers keeping smaller area. The designs presented can be optimized in power consumption. This cipher bank has a wide variety of use cases as this is a general purpose bank which can be deployed in any resource constrained device.