



5G Lattice ORAN Solution Stack 1.0 Demo

User Guide

FPGA-UG-02163-1.0

June 2022

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Acronyms in This Document	6
1. Introduction	7
2. Functional Description	8
3. Demo Setup	9
3.1. Hardware Requirements	9
3.2. Software Requirements	9
4. Setting up the Hardware	10
5. Installing the Linux Release	11
6. Running the Program	13
7. RSA and ECDH Algorithms	16
7.1. Authentication using RSA	16
7.2. ECDH for Key Exchange	17
8. Executing the Algorithm Based on User Preference	18
8.1. SHA384 PCIe to UART	18
8.2. SHA384 UART to PCIe	20
8.3. HMAC-384 PCIe to UART	24
8.4. HMAC 384 UART to PCIe	26
8.5. AES-GCM 256 Encryption	30
8.6. AES-GCM 256 Decryption	34
Appendix A – Programming the Flash	36
Programming SPI Flash on CertusPro-NX Board (with Macronix Flash)	36
Programming SPI Flash on CertusPro-NX Rev-B Board (with Winbond Flash)	40
Appendix B. Generating the Security Stack SoC Bit File	46
Top File Change for Bit File Generation	49
AES Mode Selection before Generating Bit File	50
Technical Support Assistance	53
Revision History	54

Figures

Figure 2.1. Block Diagram	8
Figure 4.1. Hardware Setup	10
Figure 5.1. Command to Navigate to Directory	11
Figure 6.1. Linkup of Device on Terminal Window	13
Figure 6.2. Linkup of Check on the Board	13
Figure 6.3. Start Communication	14
Figure 6.4. Executing clientdetected.sh Script	14
Figure 6.5. Script Run Successfully	15
Figure 7.1. Output of RSA Verification in Docklight	16
Figure 7.2. ECDH Key Exchange Flow Diagram	17
Figure 7.3. Docklight Output	17
Figure 8.1. User Data in userplaintext.txt	18
Figure 8.2. User Selection of Algorithm	18
Figure 8.3. Hash Function in Docklight	19
Figure 8.4. Hex Generation of Plain Text in FPGA_INPUT.hex	19
Figure 8.5. Hash Generation over Third Party Tool	19
Figure 8.6. Send Sequence Window of Packets	20
Figure 8.7. HEX Edit Mode	20
Figure 8.8. Copy Plain Text into Hex Edit Text	21
Figure 8.9. Open Communication Port in Docklight	22
Figure 8.10. User Selecting SHA384 UART to PCIe Algorithm	22
Figure 8.11. Hash Generation in output_data.txt	22
Figure 8.12. Hash Function in output_data.txt file	23
Figure 8.13. Hash Generation on Third Party Tool	23
Figure 8.14. User Data in userplaintext.txt	24
Figure 8.15. User Selecting HMAC-384 PCIe to UART Algorithm	24
Figure 8.16. Hash Function over Docklight	25
Figure 8.17. Hex Generation of Plain Text in FPGA_INPUT.hex	25
Figure 8.18. Hash Generation in Third Party Tool	25
Figure 8.19. HEX Edit Mode	26
Figure 8.20. Copying the Plain Text into Hex Edit Text	27
Figure 8.21. Packet in Send Sequence Name	27
Figure 8.22. User Selecting HMAC-384 UART to PCIe Algorithm	28
Figure 8.23. Sending Packet after Selecting Algorithm	28
Figure 8.24. Hash Generation in Outputdata.txt	29
Figure 8.25. Hash Function in outputdata.txt file	29
Figure 8.26. Hash Generation in Third Party Tool	29
Figure 8.27. Selecting HEX Edit Mode	30
Figure 8.28. Copying Plain Text into Hex Edit Text	31
Figure 8.29. Packet in Send Sequence Name	31
Figure 8.30. AES-GCM 256 Encryption Selection	32
Figure 8.31. Output File for AS-GCM Encryption	32
Figure 8.32. Tag Value Generated in Docklight	33
Figure 8.33. Input Decryption in inputdata.txt	34
Figure 8.34. User Selecting AES-GCM 256 Decryption Algorithm	34
Figure 8.35. Decryption over Docklight	35
Figure A.1. Lattice Radiant Programmer Getting Started Dialog Box	36
Figure A.2. Lattice Radiant Programmer- Main Interface	36
Figure A.3. Radiant Programmer- Device Selection	37
Figure A.4. Lattice Radiant Programmer- Device Properties	37
Figure A.5. Erase All (Lattice Radiant Programmer)	38

Figure A.6. Lattice Radiant Programmer-Bitstream Flashing.....	39
Figure A.1. Lattice Radiant Programmer Getting Started Dialog Box	40
Figure A.2. Lattice Radiant Programmer- Main Interface.....	40
Figure A.3. Radiant Programmer- Device Selection	41
Figure A.4. Edit Custom Device	41
Figure A.5. Custom SPI Flash Device Settings	42
Figure A.6. Edit Custom Device Settings	42
Figure A.7. Lattice Radiant Programmer- Device Properties	43
Figure A.8. Erase All (Lattice Radiant Programmer).....	44
Figure A.9. Lattice Radiant Programmer-Bitstream Flashing.....	45
Figure A.10. Opening the Complete Security Stack SoC Project	46
Figure A.11. Complete Security Stack SoC Project.....	46
Figure A.12. Loading the Application CPU Firmware	47
Figure A.13. Selecting the Application CPU Firmware	47
Figure A.14. Loading the Security CPU Firmware	48
Figure A.15. Selecting the Security CPU Firmware	48
Figure A.16. Security Stack Top File Location	49
Figure A.17. Top File Change for .bit File Generation	49
Figure A.18. OSE.vhd File in Project Directory	50
Figure A.19. Configuring the Generic Parameter for AES Mode Selection	50
Figure A.20. Opening the Security Stack_SoC Project	51
Figure A.21. Security Stack_SoC Project	51
Figure A.22. Exclude Block_cipher_hybrid_wrapper.vhd File from Project	52
Figure A.22. Security Stack_SoC Project after Synthesis and Bit File Generation.....	52

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
ECDH	Elliptic-curve Diffie–Hellman
GCM	Galois/Counter Mode
HMAC	Hash-based Message Authentication Code
ORAN	Open Radio Access Network
PCIe	Peripheral Component Interconnect Express
QSPI	Queued Serial Peripheral Interface
SHA	Secure Hashing Algorithm
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus

1. Introduction

This document describes the process for running the basic 5G Lattice ORAN™ Solution Stack. It is supported on Linux versions *16.04*, *18.04*, and *20.04*. For packet authentication, security covers algorithm for encryption and decryption of the packets including the crypto-256 and crypto-384 services to customers.

This basic demo includes the following Crypto algorithms:

- SHA384 PCIe to UART
- SHA384 UART to PCIe
- HMAC-384 PCIe to UART
- HMAC-384 UART to PCIe
- AES-GCM 256 Encryption
- AES-GCM 256 Decryption

2. Functional Description

In this project, the CertusPro™-NX is used with soft IPs to support fast packet encryption/decryption using either AES-CBC/GCM mode. Support for packet authentication is provided by ECC384, HMAC384 or RSA 3K/4K over PCIe Interface. Keys and other required configuration are set up through SMBus as a part of sideband communication.

CertusPro-NX is connected to the host PC using PCIe x1 endpoint IP. CertusPro-NX boots from the bitstreams stored at an external QSPI flash. It also connects to external components through SMBus. Furthermore, CertusPro-NX connects to an external USB through soft IP UART and an external converter UART to USB.

Figure 2.1 shows the system block diagram.

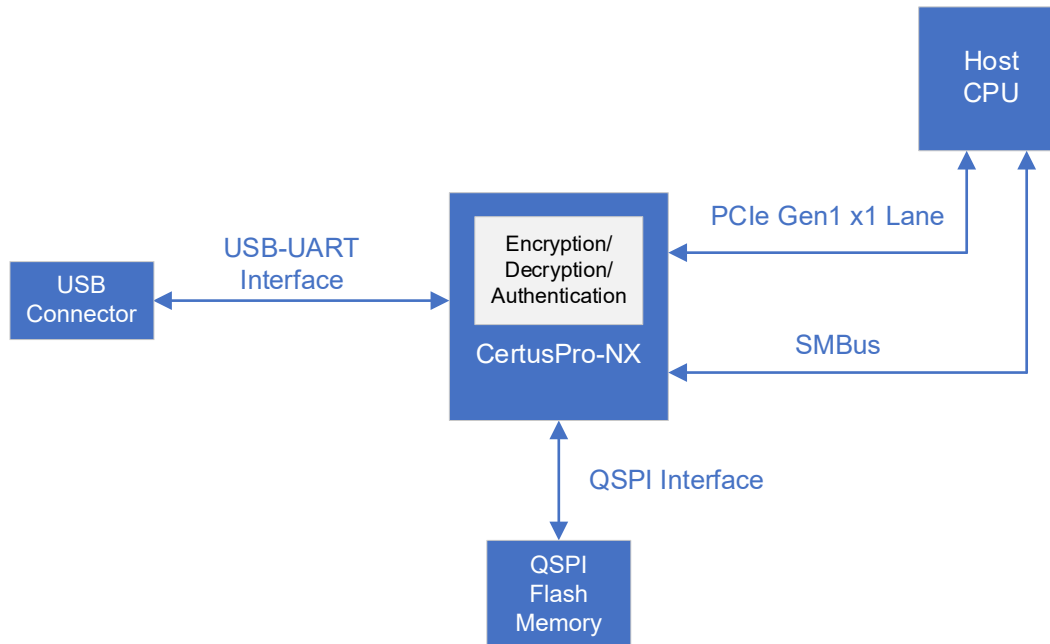


Figure 2.1. Block Diagram

3. Demo Setup

3.1. Hardware Requirements

The following hardware components are required:

- Host PC-1 running Linux Operating System of 16.04, 18.04 and 20.04
- Host PC-2 running Windows
- Lattice CertusPro™-NX Versa Evaluation Boards with 12 V power adapters(1 Main)
- USB Type-A (UART) cable for programming the bitstream file.
- Keyboard and Mouse
- PCIe Cable

3.2. Software Requirements

The following software programs are required:

- Lattice Radiant™ software version 3.1 or later
- Lattice Propel™ version 2.2 or later
- Docklight version 2.3

4. Setting up the Hardware

Figure 4.1 shows the hardware setup.

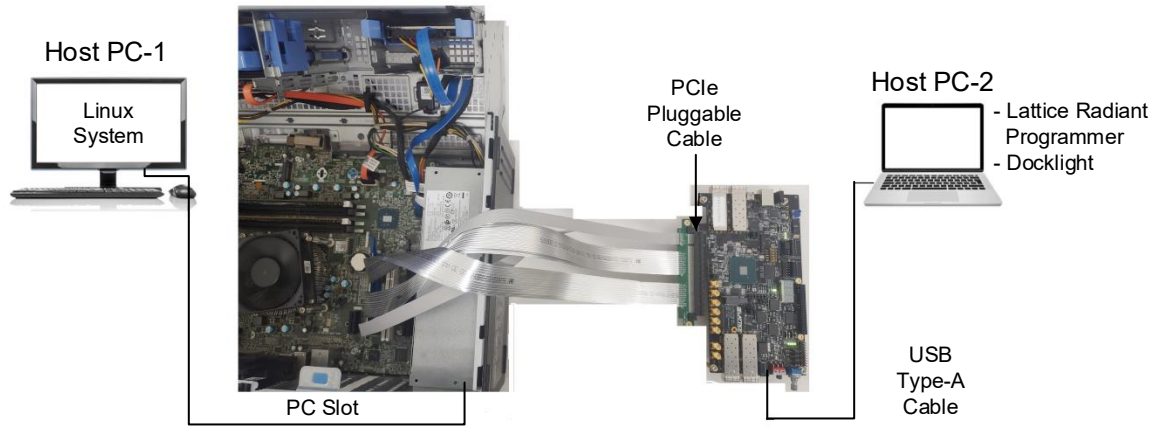


Figure 4.1. Hardware Setup

The following describes the hardware connection.

- The Host PC-1 runs the 5G ORAN script.
- The PCIe pluggable cable is connected from the CertusPro-NX versa board.
- The USB Type-A (UART) cable is connected to the board and the Host PC-2 for programming the bit file and to give input or print output over Docklight.

5. Installing the Linux Release

Note: The following installation procedure should be performed only when a new build is available or the build needs to be installed in a new PC.

To install the software:

1. Create a main folder in the home directory. In this demo, the folder is named: `5G_ORAN_REL`
2. Create a sub-folder and copy the provided build to this sub-folder.

Note: In Linux, you can create a folder using the commands below:

```
$ mkdir -p 5G_ORAN_REL
$ copy <Downloaded tar package > to folder 5G_ORAN_REL
$ Untar the file, by below command
tar -xvzf <Downloaded tar package >
```

3. Open the terminal.
4. Run the `cd` command to navigate to the build directory.
5. Open the *Readme* file and follow the indicated installation steps.

In the folder created in step 2, open the *package.sh* file.

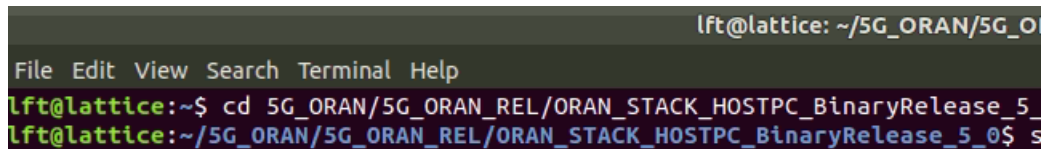


Figure 5.1. Command to Navigate to Directory

6. Run the commands below.

```
sudo apt update
sudo apt install build essential
```

7. Install *openssl*.

```
sudo apt-get install libssl-dev
```

8. Install *cmake*.

(Source Distribution) [Download the *cmake* package from <https://cmake.org/>]

Note: You can also go to the <preRequisites> folder.

Untar *cmake-3.23.1.tar.gz* by using the command below.

```
tar -xvzf cmake-3.23.1.tar.gz
cd cmake-3.23.1/
./bootstrap
sudo make
sudo make install
```

9. Install the SMBUS support.

```
sudo apt-get install libi2c-dev
```

10. Install `gcc 7.5`.

- a. Check the `gcc` version.

```
gcc --version
```

- b. If the version is not `gcc 7` or above, run the command below.

```
sudo apt-get install -y gcc-7
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 60 --
slave /usr/bin/g++ g++ /usr/bin/g++-7
```

- c. Verify the `gcc` version.

```
gcc --version
```

Note: You can also run the commands below.

```
sudo apt-get install -y software-properties-common
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt update
sudo apt install g++-7 -y
gcc -v
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 60 --slave
/usr/bin/g++ g++ /usr/bin/g++-7
```

11. Install `python3` and `pip3` on the Linux machine and run the commands below.

```
sudo apt-get install python3-pip
sudo python3 -m pip install pycryptodome
```

12. Install `openssl` version 1.1.1.

```
sudo apt install openssl 1.1.1
```

13. Go to the `preRequisites` folder and run the command below.

```
sudo cp -rf i2c* /usr/include/linux/
```

14. Make all script executable.

```
sudo chmod -R +x *.sh
```

15. Compile the application.

(For source code release only)

```
cd PCIE_Source_Code/
sudo ./compile.sh
```

(For binary code release only)

```
sudo make driver
sudo make libs
```

16. Run the command below to compile all project files.

(For source code release only)

```
cd ../
sudo ./run.sh
```

6. Running the Program

To run the program:

1. Make sure that the link between the board and the Host PC-1 is established by executing the `lspci` command. If the linkup is successful, the connected board is displayed as shown in [Figure 6.1](#).

```
lft@lattice:~$ lspci
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v6/7th Gen Core Processor Host Bridge/DRAM Registers (rev 05)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor PCIe Controller (x16) (rev 05)
00:02.0 VGA compatible controller: Intel Corporation HD Graphics 630 (rev 04)
00:14.0 USB controller: Intel Corporation 200 Series/Z370 Chipset Family USB 3.0 xHCI Controller
00:14.2 Signal processing controller: Intel Corporation 200 Series PCH Thermal Subsystem
00:15.0 Signal processing controller: Intel Corporation 200 Series PCH Serial IO I2C Controller #0
00:16.0 Communication controller: Intel Corporation 200 Series PCH CSME HECI #1
00:17.0 RAID bus controller: Intel Corporation SATA Controller [RAID mode]
00:1c.0 PCI bridge: Intel Corporation 200 Series PCH PCI Express Root Port #4 (rev f0)
00:1f.0 ISA bridge: Intel Corporation 200 Series PCH LPC Controller (Q270)
00:1f.2 Memory controller: Intel Corporation 200 Series/Z370 Chipset Family Power Management Controller
00:1f.3 Audio device: Intel Corporation 200 Series PCH HD Audio
00:1f.4 SMBus: Intel Corporation 200 Series/Z370 Chipset Family SMBus Controller
00:1f.6 Ethernet controller: Intel Corporation Ethernet Connection (5) I219-LM
01:00.0 ATA controller: Lattice Semiconductor Corporation Device 9c1e (rev 10)
02:00.0 PCI bridge: Texas Instruments XIO2001 PCI Express-to-PCI Bridge
```

Figure 6.1. Linkup of Device on Terminal Window

2. Linkup with the board. Ensure that it is connected to the Host PC-1 as indicated by a glowing LED D105.

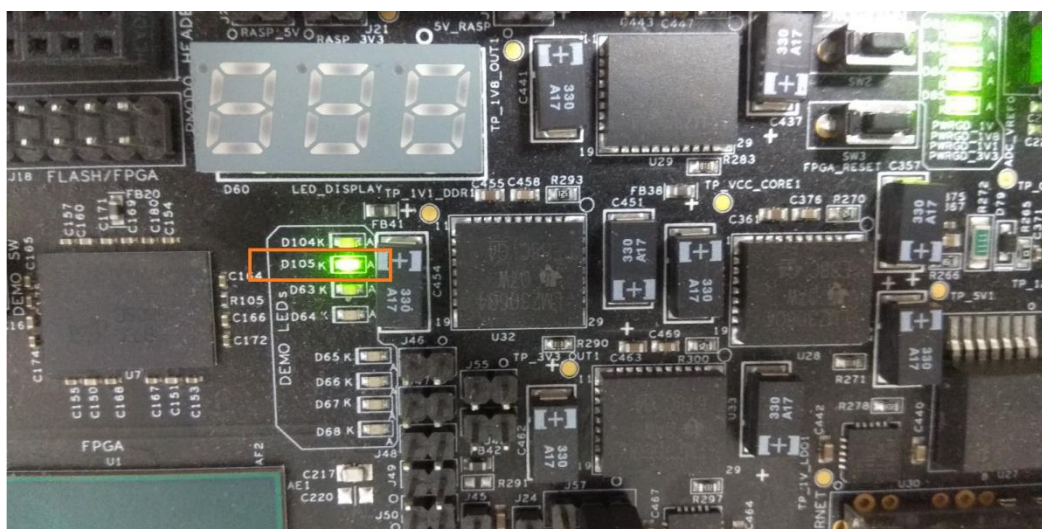


Figure 6.2. Linkup of Check on the Board

3. Open Input_data folder. Edit in `UserPlainText.txt` file and save.
`cd ../`
4. Go to the build directory and run the script by using the command below.
`sudo ./clientDetected.sh`
5. When prompted, enter the PC password to execute the script.
6. In parallel, open Docklight in the Host PC-2 and select the port to see the prints.
7. Select **Run > Start Communication** to open the communication port of Docklight.

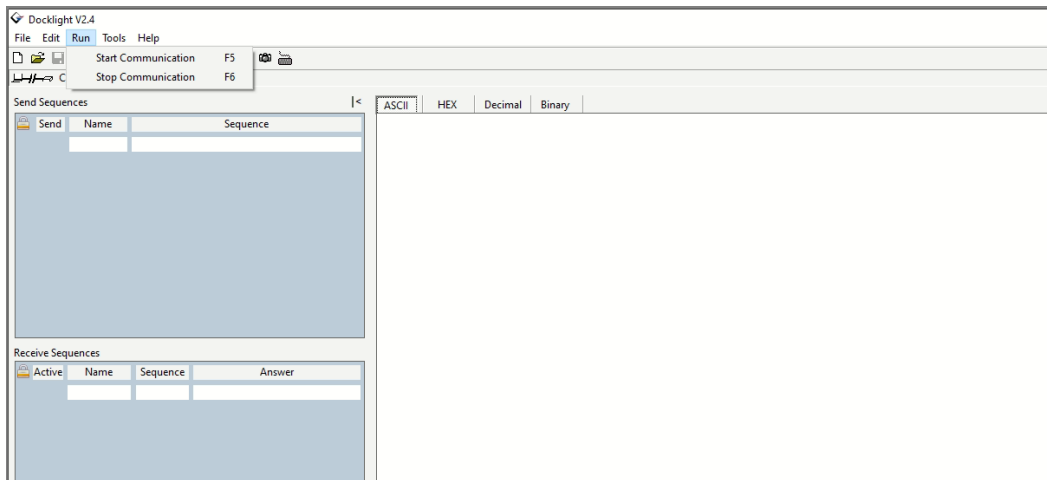


Figure 6.3. Start Communication

8. Enter the I2C SLAVE Address as an input.

```
lft@lattice:~/5G_ORAN/5G_ORAN_REL/ORAN_STACK_HOSTPC_BinaryRelease_5_0$ sudo ./clientDetected.sh
[sudo] password for lft:
Release Version: 5.0
Release Build Date: 20.06.2022
SLAVE_ADDR: 66
Smbus Driver present..
NR_ADAPTOR : 7
I2C Device [66] Detected.
Generating RSA private key, 3072 bit long modulus (2 primes)
.....++++
.....++++
e is 65537 (0x010001)
writing RSA key
.
.
.
.
.
.
.
Authentication Successful !!!
read EC key
writing EC key
.
Hello USER, select the Algorithm from below:
1. SHA 384 PC to UART
2. SHA 384 UART to PC
3. HMAC 384 PC to UART
4. HMAC 384 UART to PC
5. AES-GCM 256 ENCRYPTION
6. AES-GCM 256 DECRYPTION
7. Exit
```

Figure 6.4. Executing clientdetected.sh Script

9. If the script is run successfully, the **Signature Verified Successfully** message is displayed in Docklight as shown in [Figure 6.5](#).



7. RSA and ECDH Algorithms

7.1. Authentication using RSA

RSA algorithm is asymmetric cryptography algorithm. Asymmetric means that it works on two different keys. For example, public key and the private key.

After running the script on the Host PC, the user should perform authentication. After successful authentication, the algorithms can be executed.

To perform authentication:

1. Generate the signature.

Sign a message msg with the private key exponent d.

Calculate the message hash: $h = \text{hash}(\text{msg})$.

Encrypt h to calculate the signature: $s = \text{hd}(\text{modn})$.

The hash h should be in the range $[0 \dots n)$.

The signature s obtained is an integer in the range $[0 \dots n)$.

2. Verify the signature.

Verify signature s for the message msg with the public key exponent e.

Calculate the message hash: $h = \text{hash}(\text{msg})$.

Decrypt the signature: $h' = \text{se}(\text{modn})$.

Compare h with h' to check whether the signature is valid.

Below is the output of RSA verification on Docklight.



Figure 7.1. Output of RSA Verification in Docklight

7.2. ECDH for Key Exchange

After successful authentication, the shared secret is generated on both sides. The Host PC and FPGA exchange the created public key and make the shared secret. Below is the flow of the ECDH (Elliptic-curve Diffie–Hellman).

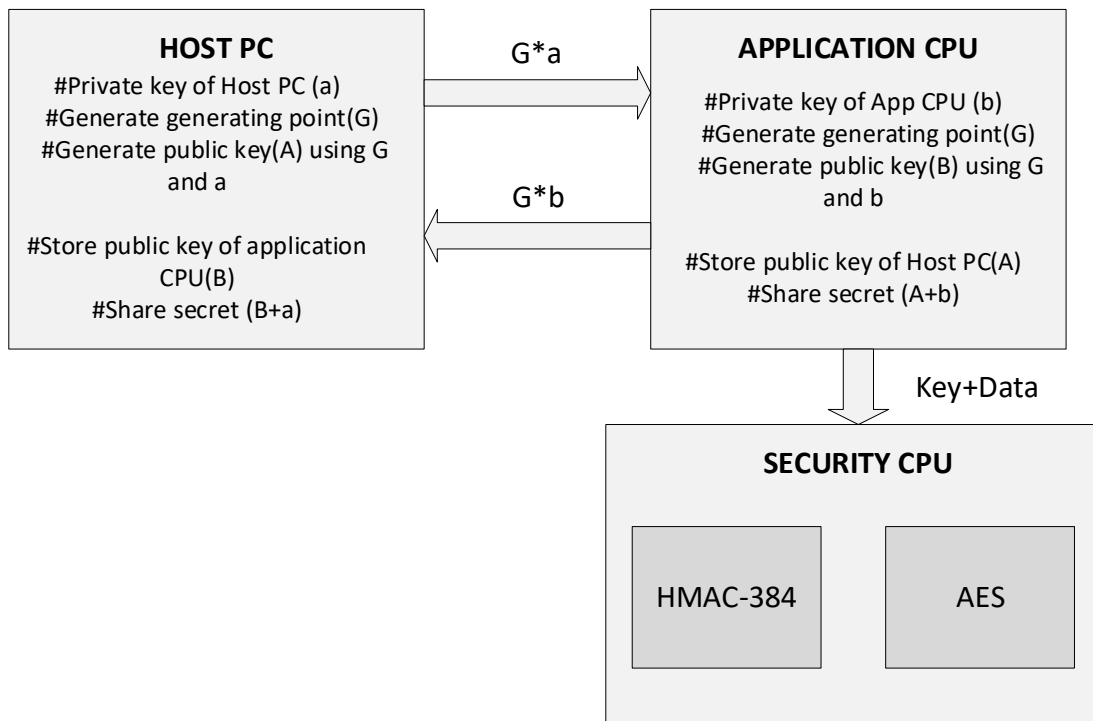


Figure 7.2. ECDH Key Exchange Flow Diagram

Figure 7.3 shows the output in Docklight.

```

Public key x :16429670e7f872eda1c908d27d02f4951b635982561bd2dc8617d94abb4a47e<LF>
Public key y :e59c1d8c1d8ab3646a9cbf1a5a55421416339ed708bc387cfcbe23891ec912cServo Busy = 0 <CR><LF>
Servo in service done <CR><LF>
Mode value written <CR><LF>
2 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000
91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000
91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000 91000000
ECDH Public Key
:0f06610ead5110eb53fdda1f53d697a62ecaa968ad7d4010e03e8efaa9d4205ff15d1fc08ab0a5eae1671183884585fc28e78e5c1f2343d77330504a351d7aa29e472b7fed91
<LF>
  
```

Figure 7.3. Docklight Output

8. Executing the Algorithm Based on User Preference

After successful authentication and establishment of shared secret, the user can execute algorithms based on his preference.

8.1. SHA384 PCIe to UART

To generate the SHA 384 PCIe to UART function:

1. Provide the User plain data in the *UserPlainText.txt* file in the *Input_data* folder. This is used in generating the hash function.

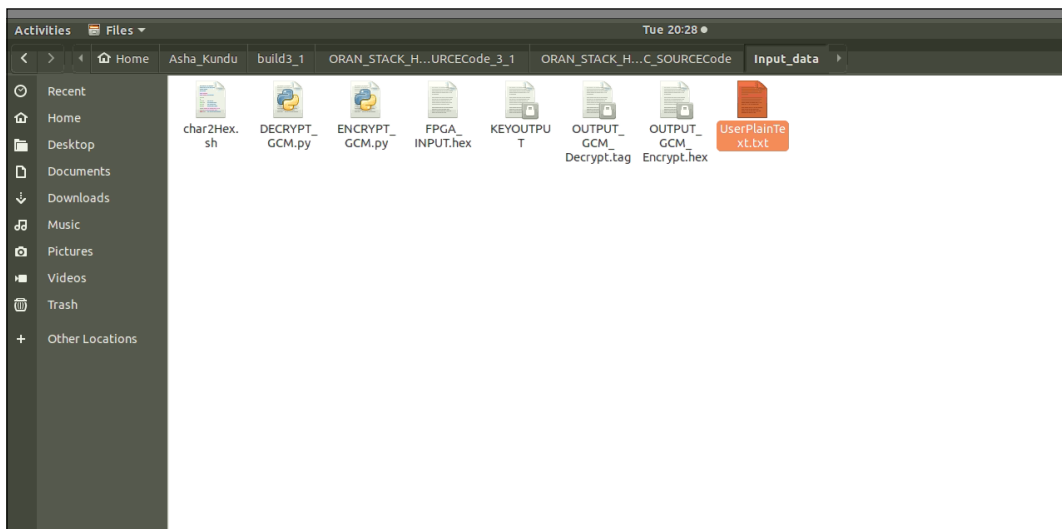


Figure 8.1. User Data in userplaintext.txt

2. Verify if the Docklight in the Host PC-2 communication port is open.
3. Enter 1 to select SHA 384 PCIe to UART Algorithm.

```

Hello USER, select the Algorithm from below:
1. SHA 384 PC to UART
2. SHA 384 UART to PC
3. HMAC 384 PC to UART
4. HMAC 384 UART to PC
5. AES-GCM 256 ENCRYPTION
6. AES-GCM 256 DECRYPTION
7. Exit
1
You have selected SHA384
Key= 69f6b8e2e714474f10d24f6836bb186480ba01a4af5ae8295bcf328c95cbd316
IV = 73a8ed90ca45d2f3780abfb1e7cabffa
AAD= 3591d066d5e9dbfe25affce9214a6c14
TAG = b'5fd851a7bf21a878e5e32680e8cd021d'

User Input File Exist...
main: SHA : read
FPGA linkup Read Ready status : 0x1
-----DMA Counter Read Operation-----
No. of descriptor will be 238
Burst Size is greater than 128*****Need to be handle*****
TotalWritelen :60832 Writelen:60832 RemainingBytes:0 WriteReqLen:60832
/home/lft/SG_ORAN/SG_ORAN_REL/ORAN_STACK_HOSTPC_BinaryRelease_5_0
libspdm_send_spdm_request[0] (0x4):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0xc):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0x30):
.
libspdm_receive_spdm_response[0] (0xff):
.
SPDM Handshaking executed successfully
    
```

Figure 8.2. User Selection of Algorithm

- After the algorithm is successfully completed, check the generated hash function in Docklight.

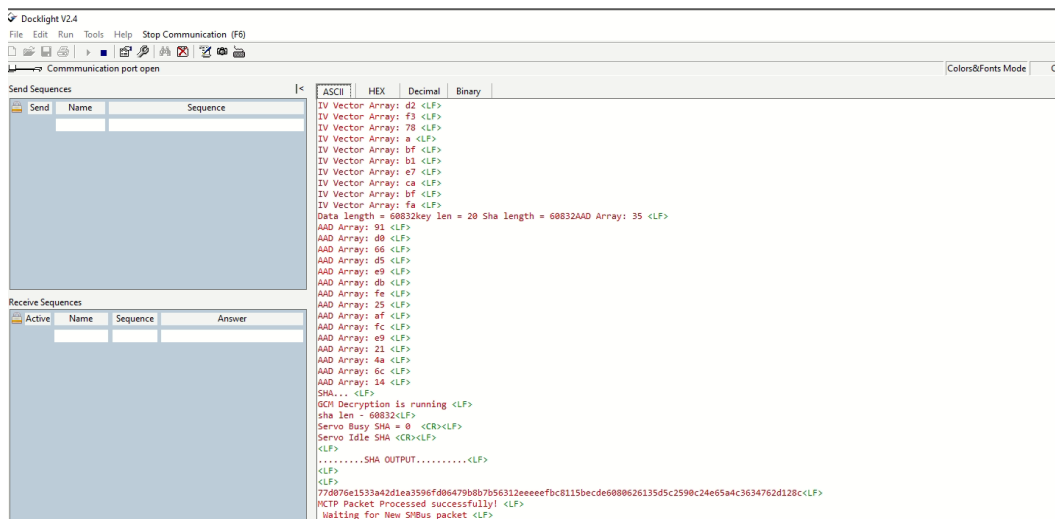


Figure 8.3. Hash Function in Docklight

- Open the Cryptogrium third party hash generator tool from <http://www.cryptogrium.com/sha384-online-hash-generator.html>.
- A file with the hex value of the plain user data is generated during the execution of *FPGA_INPUT.hex* in the *Output_data* folder.
- Copy the hex data from the file to the input of the third party tool and generate its hash function

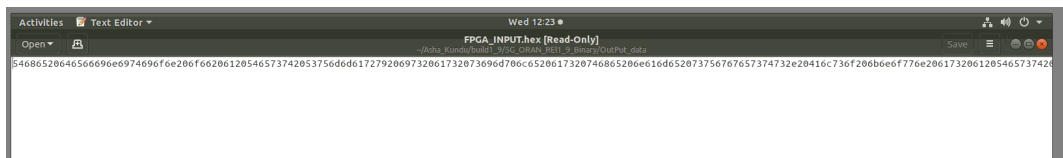


Figure 8.4. Hex Generation of Plain Text in FPGA_INPUT.hex

- Verify if the hash function generated in Docklight and in the third party tool are the same.

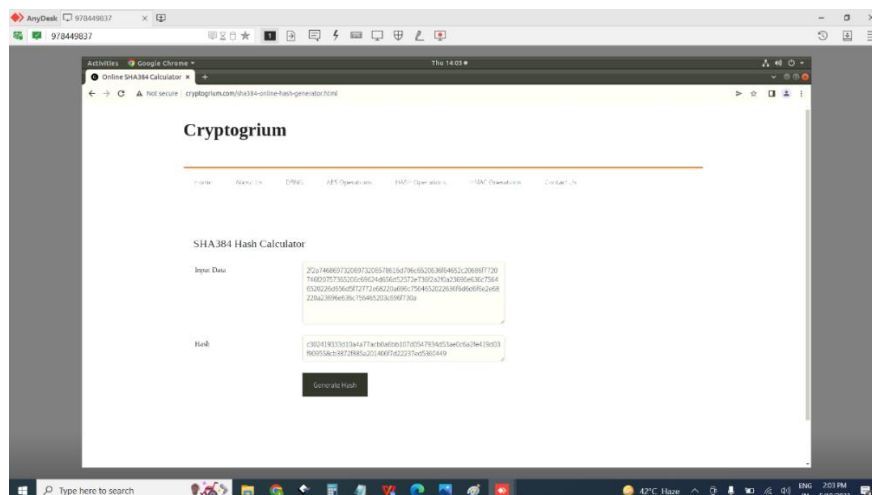


Figure 8.5. Hash Generation over Third Party Tool

8.2. SHA384 UART to PCIe

To generate the SHA384 UART to PCIe function:

1. Provide the User plain data in the form of hex value in Docklight. This is used in generating the hash function.
2. Make the packet of hex value in Docklight.

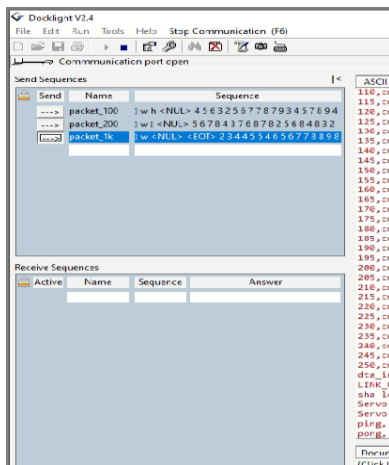


Figure 8.6. Send Sequence Window of Packets

3. Double click the blank block under the **Name** column in the **Send Sequences** block.
4. The **Edit Send Sequence** window opens. Enter the name of packet.
5. Select **HEX** in **Edit Mode**.

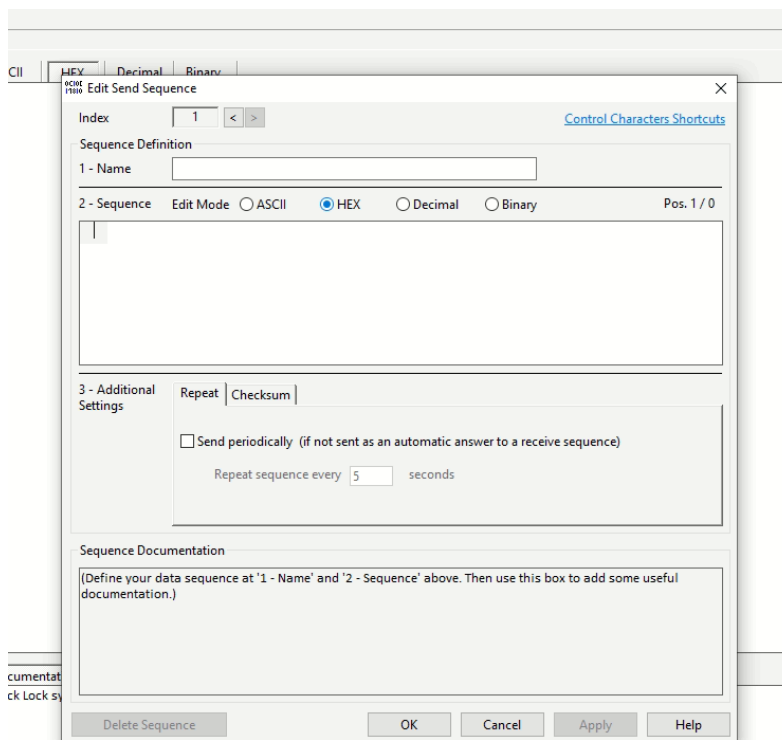


Figure 8.7. HEX Edit Mode

Note:

1 | 2 | 3 | 4 |

Format of packet -> CC|77|Total packet number converted in Hex| format of packet is from 1 to 4 index, the first and second always remain the same.

1024 packet converted into hex is 400. Starting from the right, last two digits are zero, which comes the third index and the remaining 4 comes in fourth place by 04.

For Example: CC|77|00|04

203 packet converted into hex is CB -> CC|77|CB|00 - CB is in the third index, the fourth index is zero as no other number is left.

6. Copy any plain data and convert that into hex value by any online tool (for example, <https://www.online-toolz.com/tools/text-hex-converter.php>).
7. Paste the hex data to the **Edit Mode** box in Docklight using the concept of CC|77|Number of total packet including CC, 77 AB, 00 in hex.
8. Copy the hex data from fifth index.

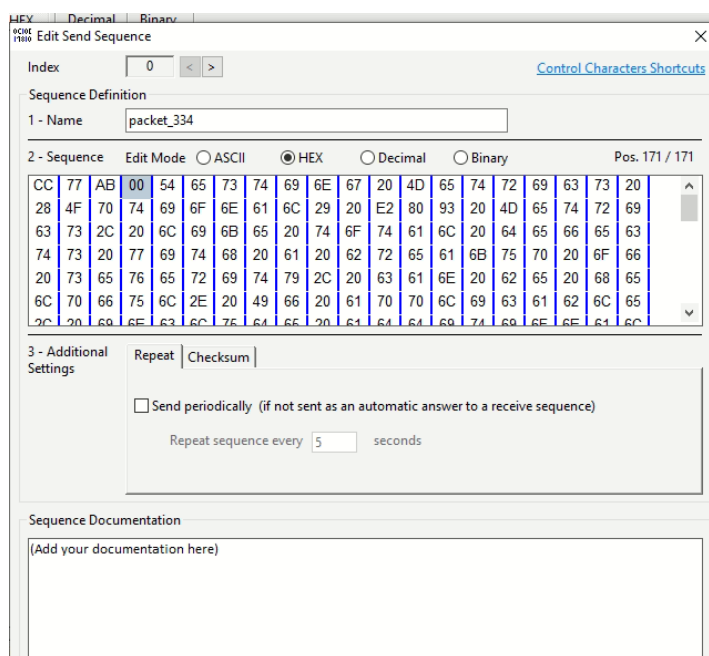


Figure 8.8. Copy Plain Text into Hex Edit Text

9. Click **OK**.
10. The packet name is displayed under the **Name** column in the **Send Sequences** block.
11. Select **Run > Start Communication** to open the communication port of Docklight.

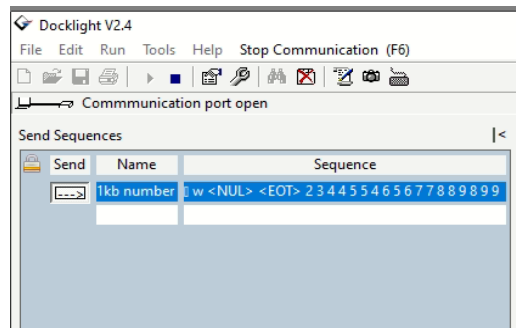


Figure 8.9. Open Communication Port in Docklight

12. Enter 2 to select SHA 384 UART to PCIe Algorithm

```
Hello USER, select the Algorithm from below:
1. SHA 384 PC to UART
2. SHA 384 UART to PC
3. HMAC 384 PC to UART
4. HMAC 384 UART to PC
5. AES-GCM 256 ENCRYPTION
6. AES-GCM 256 DECRYPTION
7. Exit
2
You have selected SHA384
Key= 69f6b8e2e714474f10d24f6836bb186480ba01a4f5ae8295bcf328c95cbd316
IV = 73a8ed98ca45d2f3789abfb1e7cabffa
AAD= 3591d066d5e9dbfe25affce9214a6c14

User Input File Exist...
libspdm_send_spdm_request[0] (0x4):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0xc):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0x30):
.
libspdm_receive_spdm_response[0] (0xff):
.
SPDM Handshaking executed successfully
main: SHA : write
Read status : 0x1
FPGA Data Write status : 0x1
count of 128 bit block received : 0x3
TestDMAWriteCounter:128 burst_size:64 file_size:64
```

Figure 8.10. User Selecting SHA384 UART to PCIe Algorithm

13. After selecting the algorithm, a *Please send data over UART in packet format* message is displayed in Docklight.

14. Send the packet.

15. The output is generated through PCIe DMA Read in *output.txt* file in the *output_Data* folder.

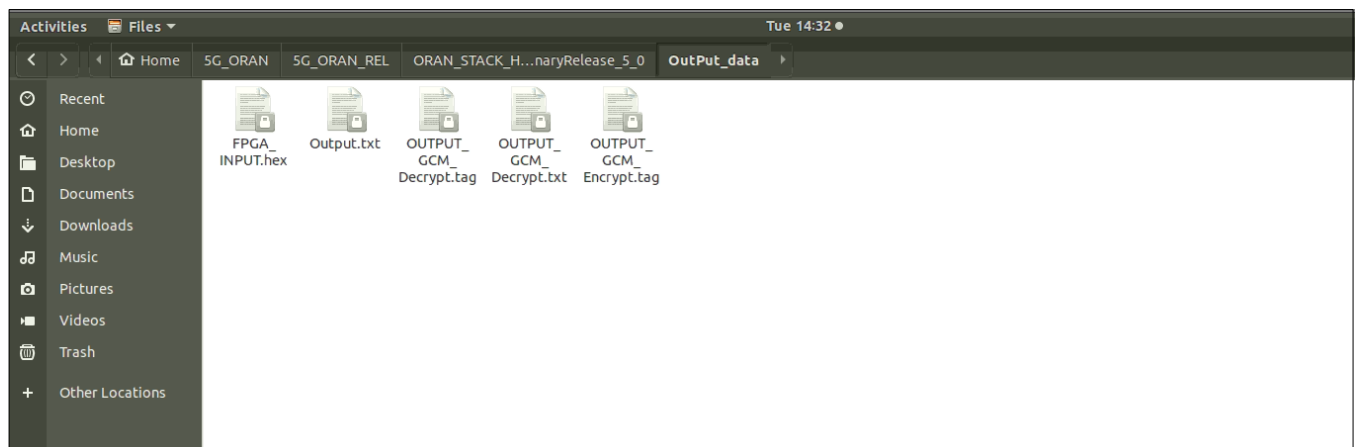


Figure 8.11. Hash Generation in output_data.txt

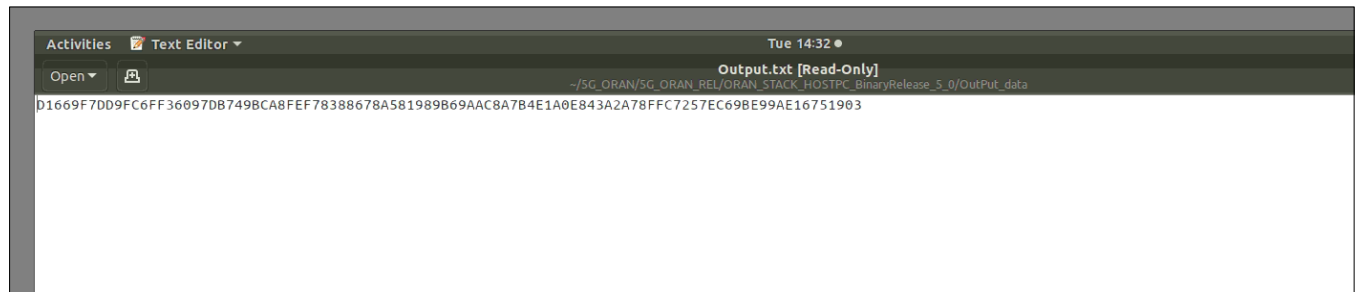


Figure 8.12. Hash Function in output_data.txt

16. Open the Cryptogrium third party hash generator tool from <http://www.cryptogrium.com/sha384-online-hash-generator.html>.
17. Copy the same hex value of input given as packets in Docklight.
18. Verify if the hash function generated in Docklight and in the third party tool are the same.

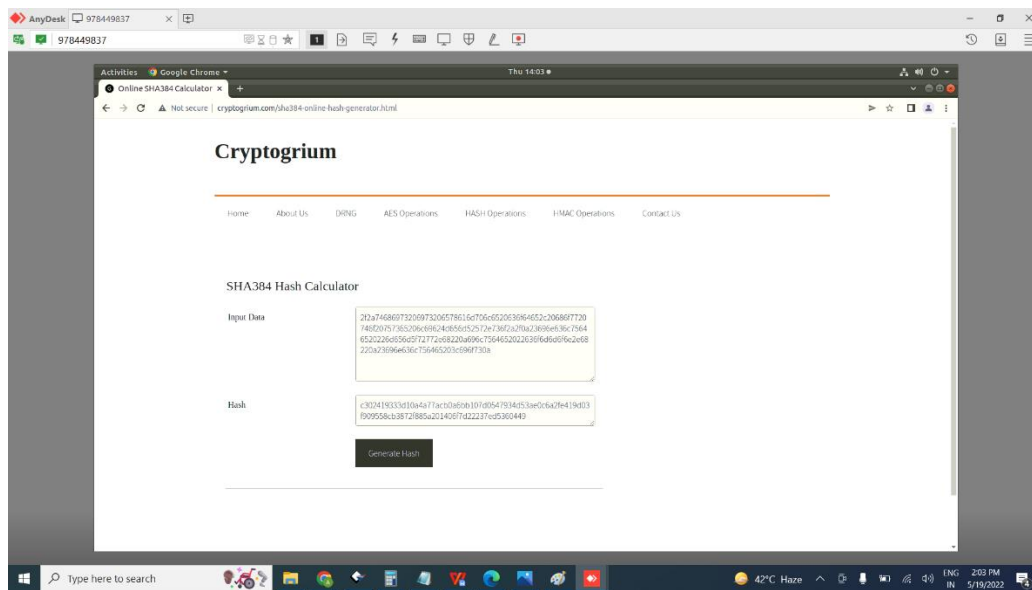


Figure 8.13. Hash Generation on Third Party Tool

8.3. HMAC-384 PCIe to UART

To generate the HMAC 384 PCIe to UART function:

1. Provide the user plain data in *UserPlainText.txt* in the *Input_data* folder. This is used in generating the hash function.

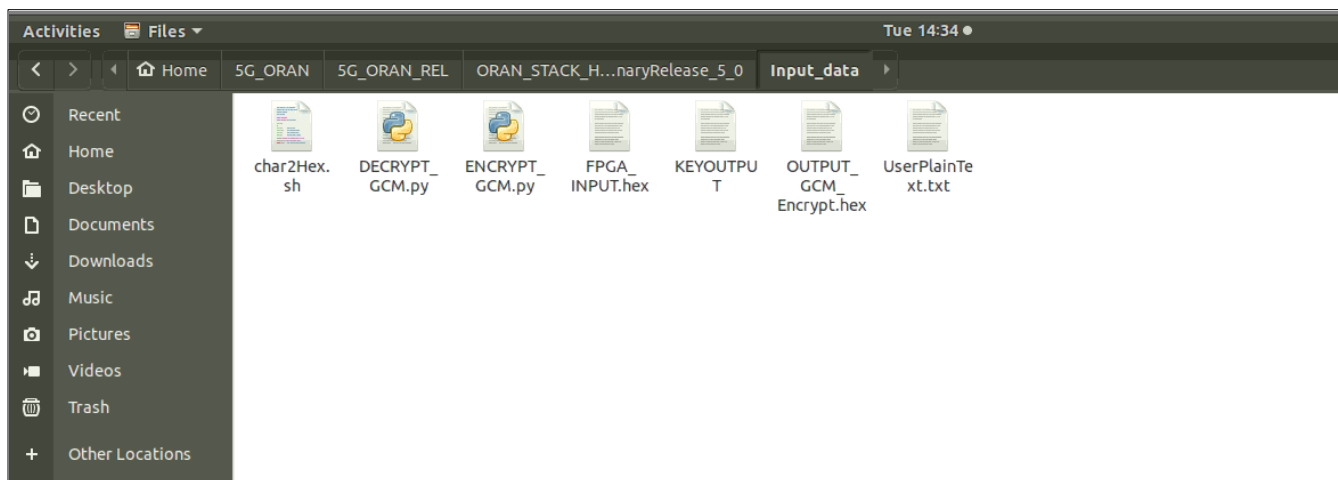


Figure 8.14. User Data in userplaintext.txt

2. Enter 3 to select **HMAC 384 PCIe to UART Algorithm** to generate the hash function in Docklight.

```
Hello USER, select the Algorithm from below:
1. SHA 384 PC to UART
2. SHA 384 UART to PC
3. HMAC 384 PC to UART
4. HMAC 384 UART to PC
5. AES-GCM 256 ENCRYPTION
6. AES-GCM 256 DECRYPTION
7. Exit
3
You have selected HMAC384
Key= 69f6b0e2e714474f10d24f6036bb186480ba01a4af5ae8295bcf328c95cbd316
IV = 73a8ed90ca45d2f3780abfb1e7cabffa
AAD= 3591d060d5e9dbfe25affce9214a6c14
TAG = b'5fd851a7bf21a878e5e32680e8cd021d'

User Input File Exist...
main: HMAC + read
FPGA linkup Read Ready status : 0x1
-----DMA Counter Read Operation-----
No. of descriptor will be 238
Burst Size is greater than 120*****Need to be handle*****
TotalWriteLen :60832 WriteLen:60832 RemainingBytes:0 WriteReqLen:60832
/home/lft/SG_ORAN/SG_ORAN_REL/ORAN_STACK_HOSTPC_BinaryRelease_5_0
libspdm_send_spdm_request[0] (0x4):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0xc):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0x30):
.
libspdm_receive_spdm_response[0] (0xff):
.
SPDM Handshaking executed successfully
```

Figure 8.15. User Selecting HMAC-384 PCIe to UART Algorithm

3. Check the generated hash function in Docklight after successfully executing the algorithm.

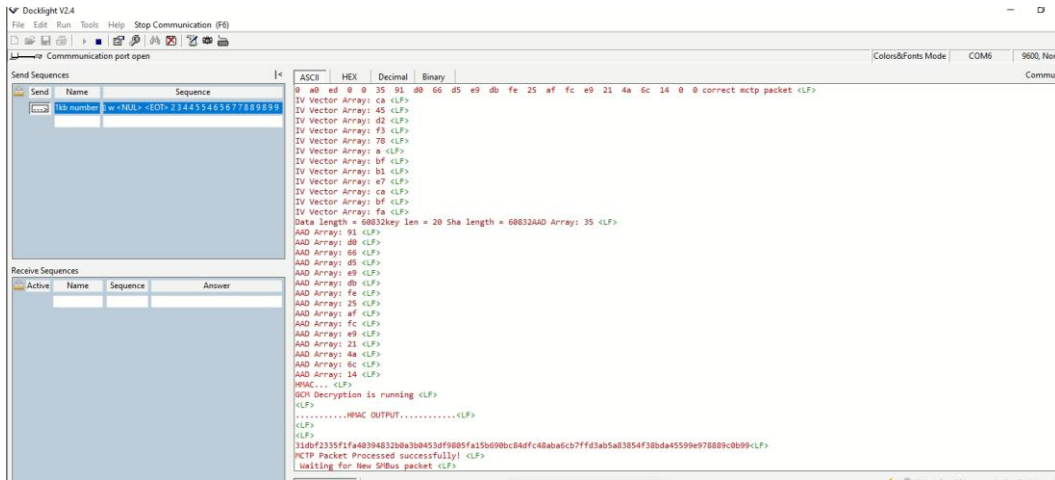


Figure 8.16. Hash Function over Docklight

- Open the Cryptogrium third party hash generator tool from <http://www.cryptogrium.com/sha-HMAC384-online-hash-generator.html>.
- A file with the hex value of the plain user data is generated during the execution of *FPGA_INPUT.hex* in the *Output_data* folder.
- Copy the hex data from the file to the input of the third party tool and generate its hash function.

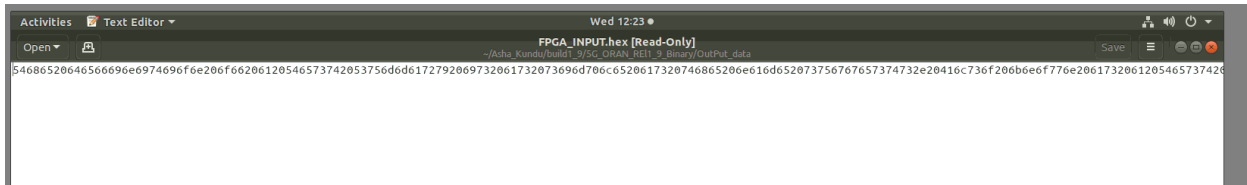


Figure 8.17. Hex Generation of Plain Text in FPGA_INPUT.hex

- Verify if the hash function generated in Docklight and in the third party tool are the same.

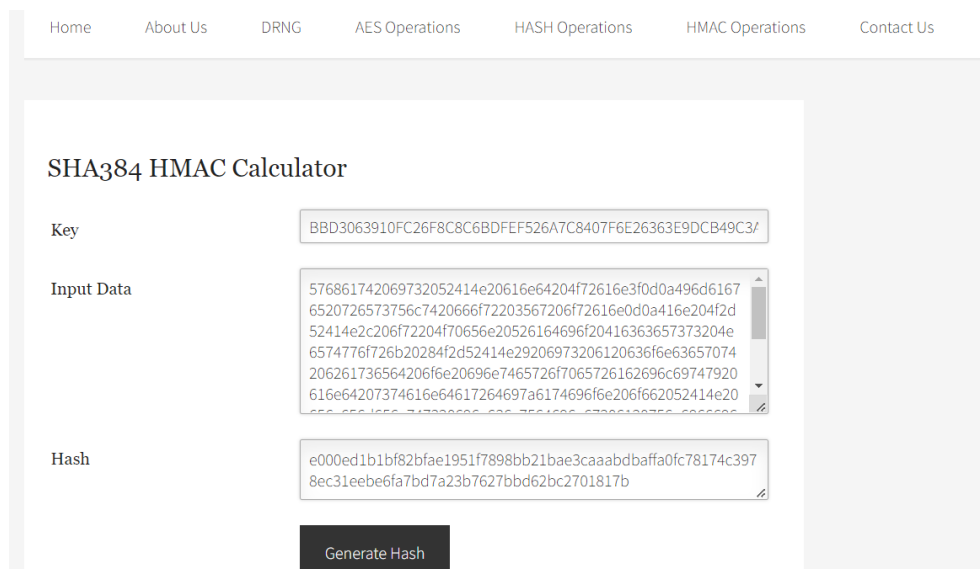


Figure 8.18. Hash Generation in Third Party Tool

8.4. HMAC 384 UART to PCIe

To generate the HMAC 384 UART to PCIe function:

1. Convert the user plain data into hex value in Docklight. This is used in generating the hash function.
2. Double click the blank block under the **Name** column in the **Send Sequences** block.
3. The **Edit Send Sequence** window opens. Enter the name of the packet.
4. Select **HEX** in **Edit Mode**.

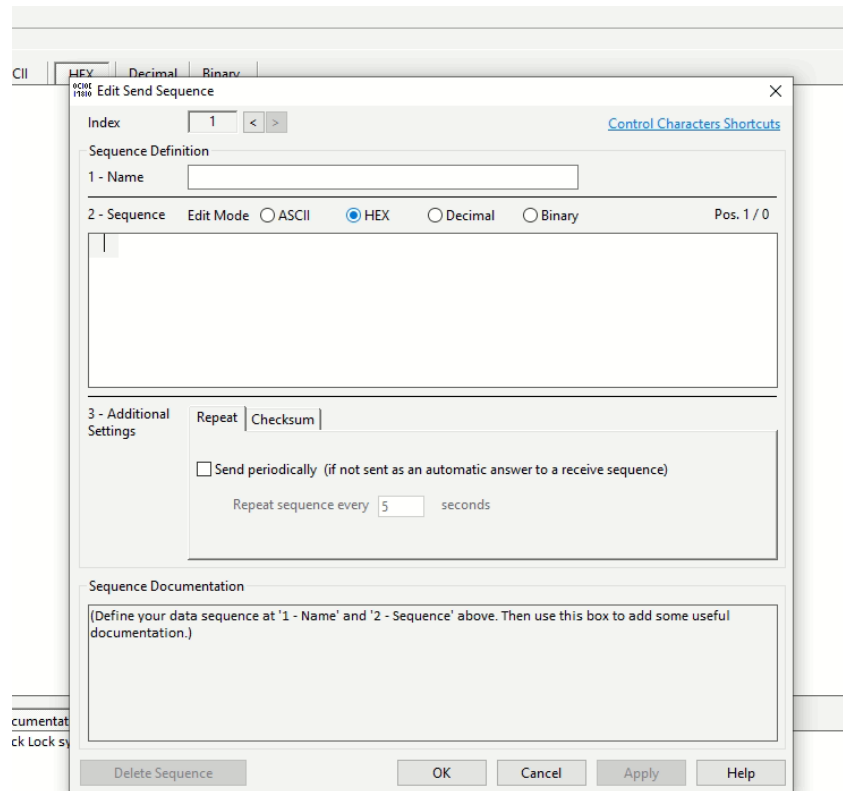


Figure 8.19. HEX Edit Mode

5. Copy any plain data and convert that into hex value by any online tool (for example, <https://www.online-toolz.com/tools/text-hex-convertor.php>).
6. Paste the hex data to the **Edit Mode** box in Docklight using the concept of CC|77|Number of total packet including CC, 77 AB, 00 in hex.
7. Copy the hex data from fifth index.

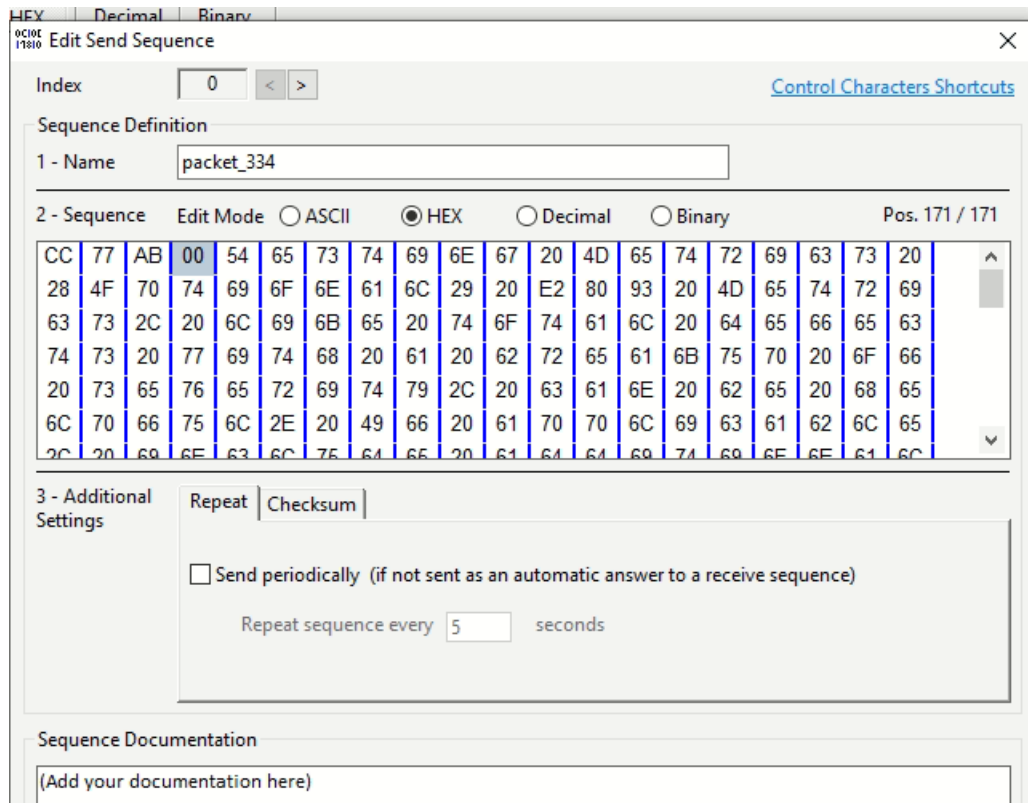


Figure 8.20. Copying the Plain Text into Hex Edit Text

8. Click **OK**.
9. The packet name is displayed under **Name**.
10. Select **Run > Start Communication** to open the communication port of Docklight.

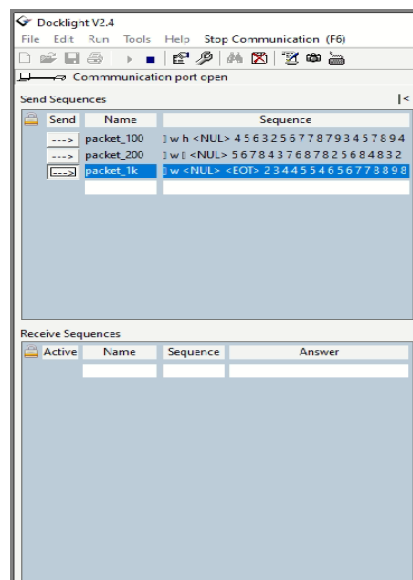


Figure 8.21. Packet in Send Sequence Name

11. Enter 4 to select HMAC 384 UART to PCIe Algorithm.

```

Hello USER, select the Algorithm from below:
1. SHA 384 PC to UART
2. SHA 384 UART to PC
3. HMAC 384 PC to UART
4. HMAC 384 UART to PC
5. AES-GCM 256 ENCRYPTION
6. AES-GCM 256 DECRYPTION
7. Exit
4
You have selected HMAC384
Key= 69f6b2e714474f10d24f6836bb186480ba01a4f5ae8295bcf328c95cbd316
IV = 73abed09ca45d2f3780abfb1efcabffa
AAD= 3591d06d5e9dbfe25affce9214a6c14

User Input File Exist...
libspdm_send_spdm_request[0] (0x4):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0xc):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0x30):
.
libspdm_receive_spdm_response[0] (0xff):
SPDM Handshaking executed successfully
main: HMAC : write
-----DMA Counter Write Operation-----
Read status : 0x0
Read status : 0x1
FPGA Data Write status : 0x1
count of 128 bit block received : 0x3
TestDMAWriteCounter:128 burst size:64 file size:64

```

Figure 8.22. User Selecting HMAC-384 UART to PCIe Algorithm

12. After selecting the algorithm, a *Please send data over UART in packet format* message is displayed in Docklight.

13. Send the packet.

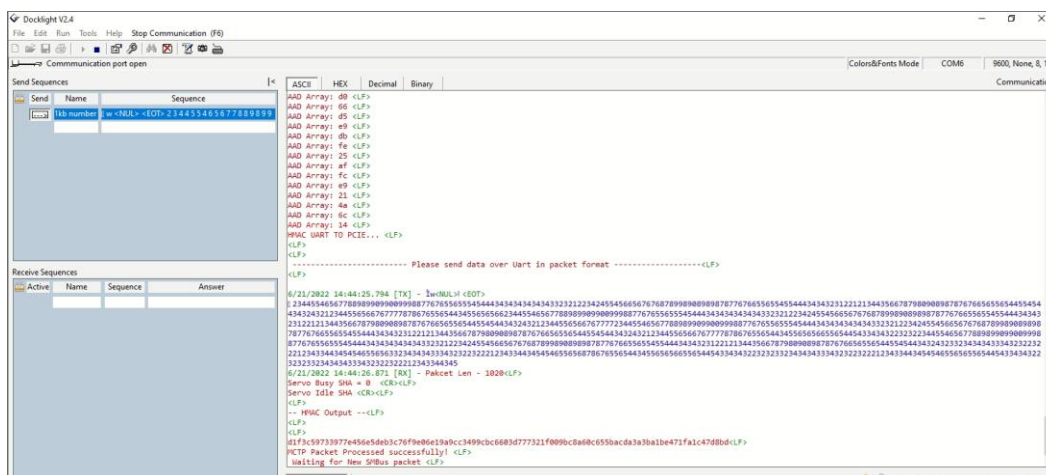


Figure 8.23. Sending Packet after Selecting Algorithm

14. Output is generated through the PCIe DMA Read in the *output.txt* file in the *output_Data* folder.

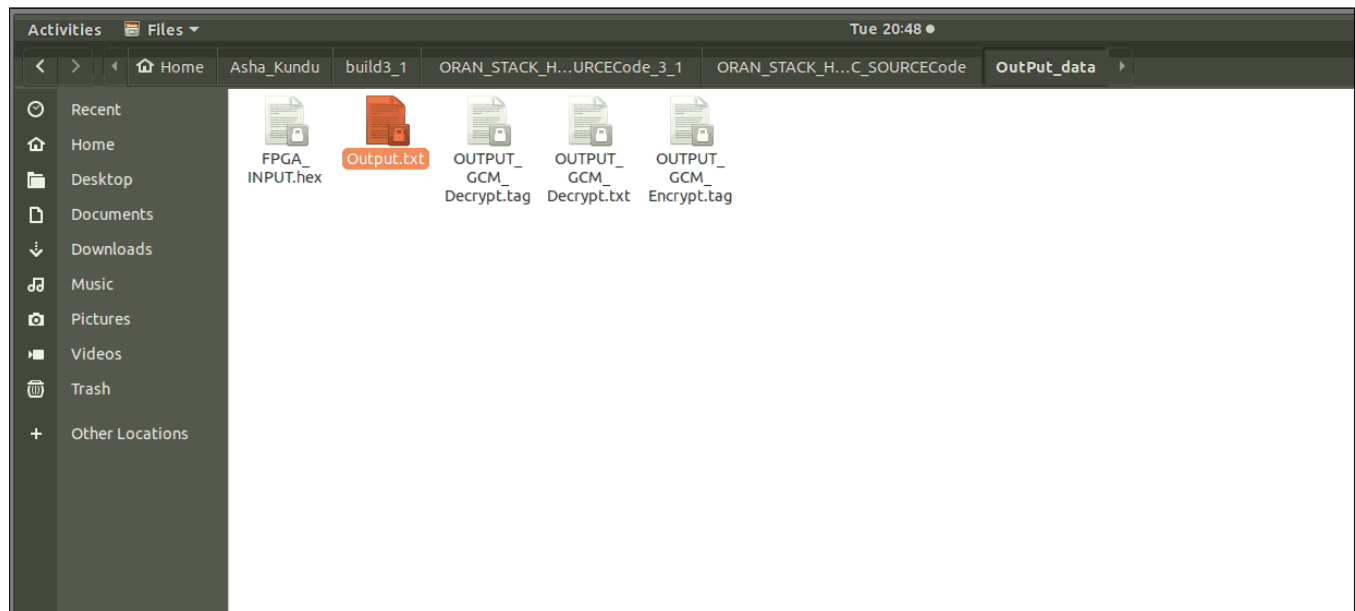


Figure 8.24. Hash Generation in Outputdata.txt



Figure 8.25. Hash Function in outputdata.txt file

15. Open the Cryptogrium third party hash generator tool from <http://www.cryptogrium.com/HMAC384online-hash-generator.html>. Copy the same hex value given as input in Docklight.
16. Copy the key from *Input_data* folder in the *Keyoutput* file.
17. Verify if the hash function generated over Docklight is same with the tool.

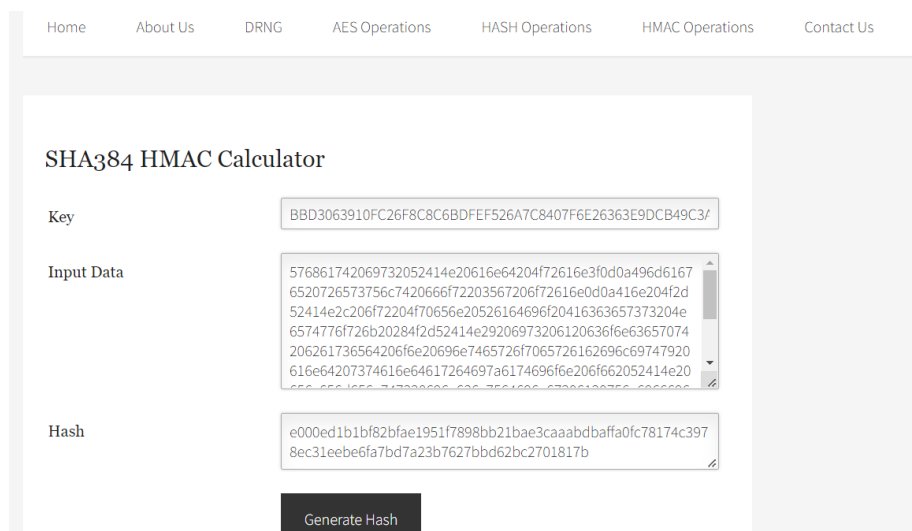


Figure 8.26. Hash Generation in Third Party Tool

8.5. AES-GCM 256 Encryption

To generate the AES-GCM 256 Encryption function:

1. Convert the user plain data to hex value in Docklight.
2. Double click the blank block under the **Name** column in the **Send Sequences** block.
3. The **Edit Send Sequence** window opens. Enter the name of packet.
4. Select **HEX** in **Edit Mode**.

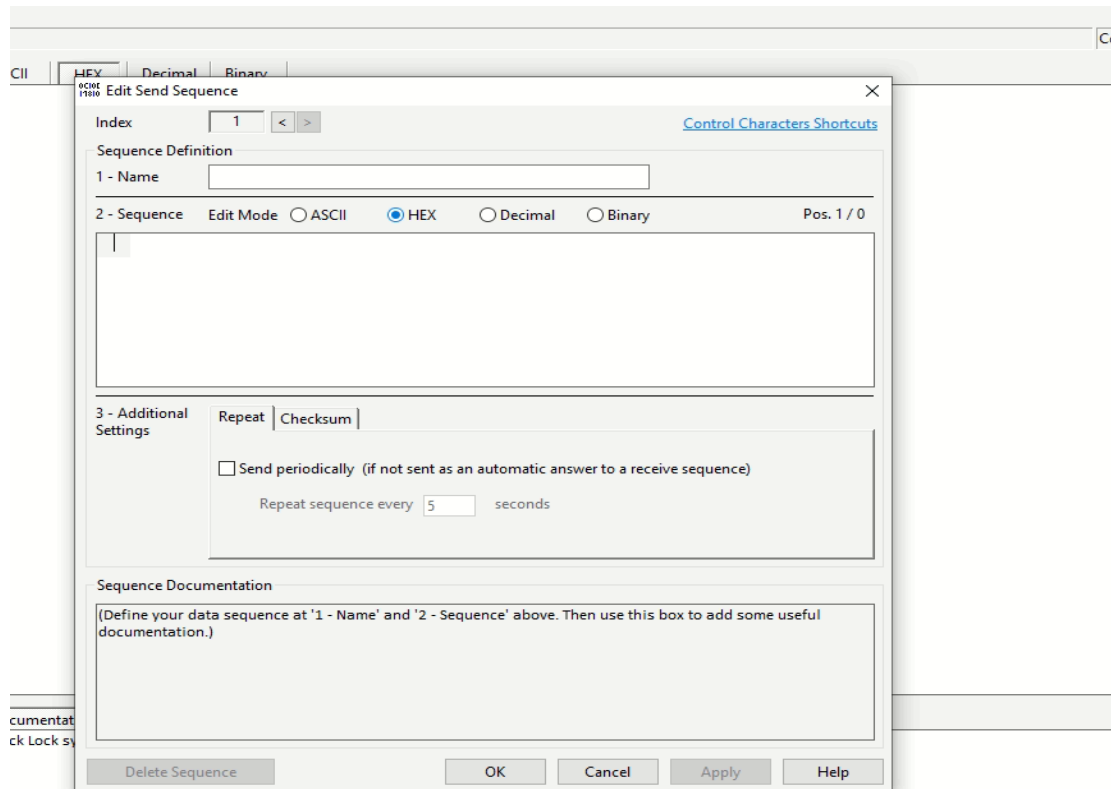


Figure 8.27. Selecting HEX Edit Mode

5. Select any plain data and convert it into hex value using any online tool.
6. Copy the packet to the **Edit Mode** box using the concept of CC|77|Number of total packet including CC, 77 AB, 00 in hex.
7. Copy the packet.

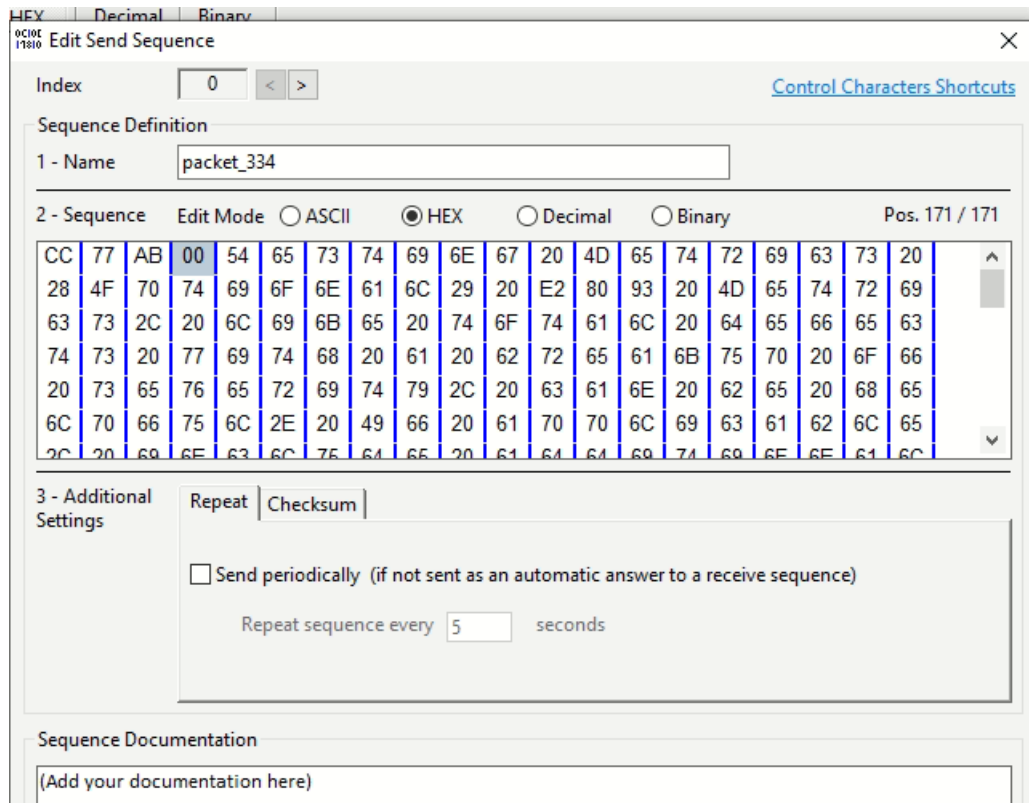


Figure 8.28. Copying Plain Text into Hex Edit Text

8. Click **OK**. The packet name is displayed under the **Name** column.

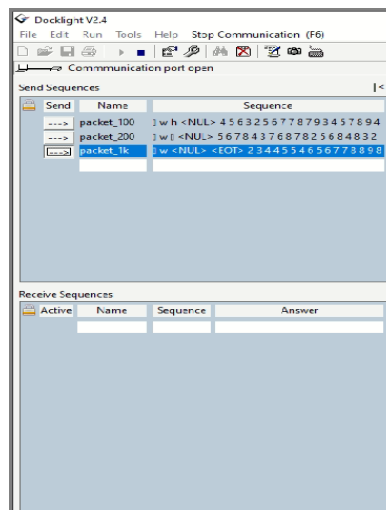


Figure 8.29. Packet in Send Sequence Name

9. Select option 5 for AES-GCM -256 Encryption

```

6. AES-GCM 256 DECRYPTION
7. Exit
8
You have selected AES - GCM 256 for ENCRYPTION
Key= 69f6b8e2e71447f10d24f683bb186480ba01a4af5ae8295bcf328c95cbd316
IV = 73a8ed90ca45d2f3780abfb1e7cabffa
AAD= 3591d060d5e9b0f25affce9214a0c14

User Input File Exist...
llbispdm_send_spdm_request[0] (0x4):
llbispdm_receive_spdm_response[0] (0xff):
llbispdm_send_spdm_request[0] (0xc):
llbispdm_receive_spdm_response[0] (0xff):
llbispdm_send_spdm_request[0] (0x30):
llbispdm_receive_spdm_response[0] (0xff):

SPDM Handshaking executed successfully
main: AES : write
-----DMA Counter Write Operation-----
Read status : 0x1
FPGA Data Write status : 0x1
count of 128 bit block received : 0x40
TestDMAWriteCounter:128 burst_size:1040 file_size:1040
tag: f0b871be841c76fc045438cc065380a
key: b'69f6b8e2e71447f10d24f683bb186480ba01a4af5ae8295bcf328c95cbd316'
iv: b'73a8ed90ca45d2f3780abfb1e7cabffa'
aad: b'3591d060d5e9b0f25affce9214a0c14'
File size is : 2048 bytes
data: b'144a229af302b4bb0e9c4658b9b64a12686dc36891c90fc3c64b84b7a08db4eb24fe9266a108e34263163b58880407d6a0b8303f5d8cedb886a09a5c259b25bac1b80ce2a5d96e05aa82c5766b6d7
58f5c86b23df66eadeb91a398bda26726baee1583c3b765f8bf0042a110ec5fb626a3661b545abc039f6572082a93970c4f9ba9dcf8d2db78b5ca3fc84423e059fd828eda66291265709fae38f065cc7e27ed0d00edca
ee0071a932d45cfc495c4e0b4d4f2e03fa22ba00b2372e0fd1fda351d404d4d2c3aa3acfc9b0edc45ed55163c913cfbb70d723312d4d3cdec0f5edcb994de3f6c4e4de23a55f5b0b87f1f1f9e200e27d68a905
bee45a8d4af8f5c55d90a24af11c0deaada2448492803afa7f8c7914cb4b86fa850a9db654f6e96dcffbb26ee08964d8fca3f0eb4937dea5f79cbe307f165fcc775a8f8c09530c51d1fe955c7f01d43f88a
93b2b6fd7707841aa205a91726fa32497eba0bda3eb52f4b78c8c7d855584fd14819e085222398834861aa91407ec1d0ac7286e9f8b7b7e017156d459efffb7b4c0b502432f75ff5f19648b043bfcf6b5dcceae33
cd25e19ead8ac34193261be48146e0d3e393d0f9118eb298b32c296188625bb4c78dbf771eb3843c3420822c13581eb4b3ae0fddcbe7d104370330c1e3bc386a3a25381a7e2d885179130b4e1f51cc851f085c51
1859a421a1b170551fedf0ed3014f1100523a7765793a504616073f2ae76deae5a3f2911b04529d003c1492181311aa4b4519119ee8a5009347aaab5922cf4e87448b-a0f61f50b0785a8a2f4950eb0b56354f4f
acc023ad4ed1c23c148db1bb2869f25ff5725383fcafe1e7c0c216aeb1632255643990cba107d54da4ef38e0ede7524af9c9558bfc3003a161eaeab3364492f32e224ae245e18ddae4af82a06f0bf325ca163ef151e07
296a9af360d09e35d45e72c3449e3df9874abf3c8ecdec2fbd4dbf0340197c68bc550737d6cadbaef9eb486d2d8d73d92d13b3e5db71900eb241a464b5e2d5629b7298bf0231c2dcf65c2507bd3095107d514e852572
f03560fa02d91a9134fcr2327cd02126bd354a89c3c74b126bba786a6de38087a602afe7b90cfa691a50455ac732143f90834916d431df7f73cd0ceae8d8129a471bcb08b184a7145a8e29d9361222ccde17e9a18e22
250e29101333b033ba09b73e03989f07a3c5df58ad062b0c0936434a45ff0c7b73aef4aee9e78adcfdd52ba43654974a5f72ad0df869002d9ba2719b5d35a21a8eace104f113ab4d4e0c4b6d4cb423ef0446effab
48716aa52245446ca04559c0f7cb4c045cc1e2203ecf5dbcc5256b19cd405adfb82408589ec03e58cd5188804d6be0a226ce9054769ddc847c0bc09'
tag: b'f0b871be841c76fc045438cc065380a'
None

```

Figure 8.30. AES-GCM 256 Encryption Selection

10. After selecting the algorithm, a *Please send data over UART in packet format* message is displayed in Docklight.
11. Send the generated packet.
12. Output is generated on Output_GCM_Decrypt.txt.

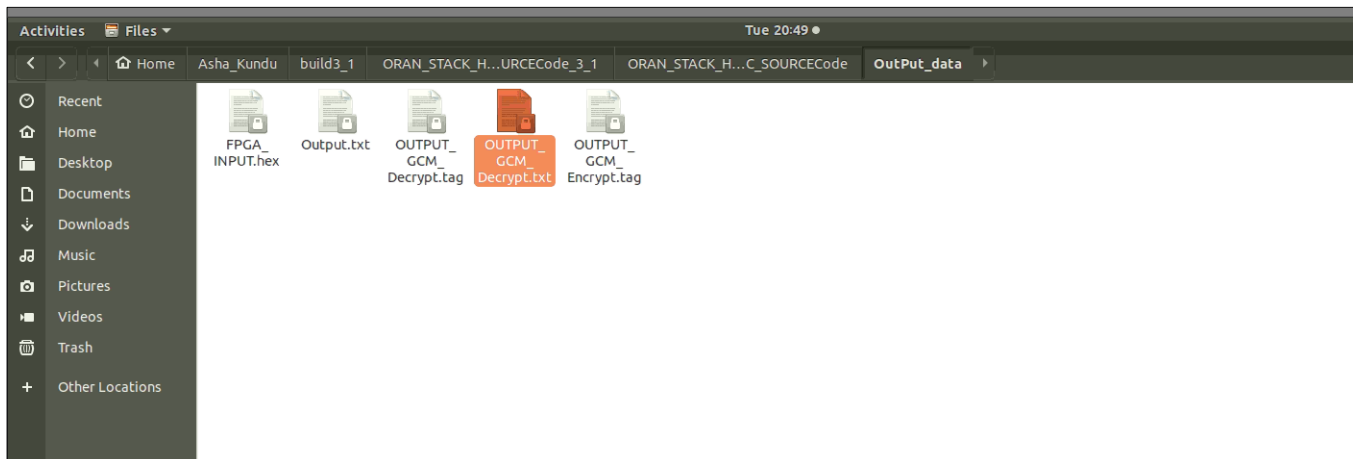


Figure 8.31. Output File for AS-GCM Encryption

13. Match the Docklight Tag value with the *OUTPUT_GCM_Decrypt.tag* file in the *Output_data* folder.

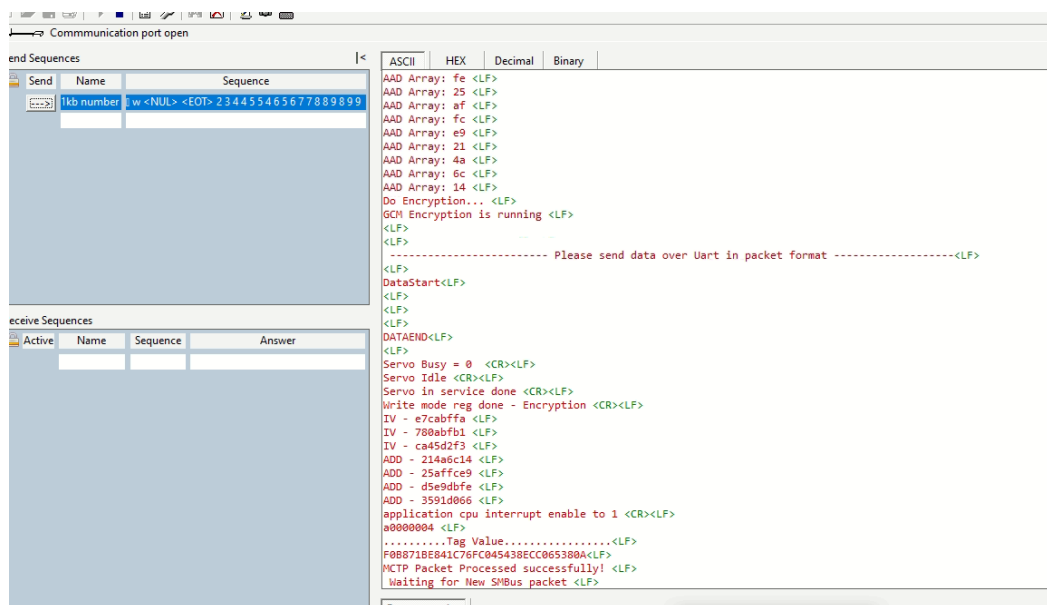


Figure 8.32. Tag Value Generated in Docklight

- Match the UART data given in hex with the *Output_GCM_Decrypt.txt* file in the *Output_data* folder

8.6. AES-GCM 256 Decryption

To generate the AES-GCM 256 Decryption function:

1. Provide the user plain data in *UserPlainData.txt* file in the *Input_data* folder.

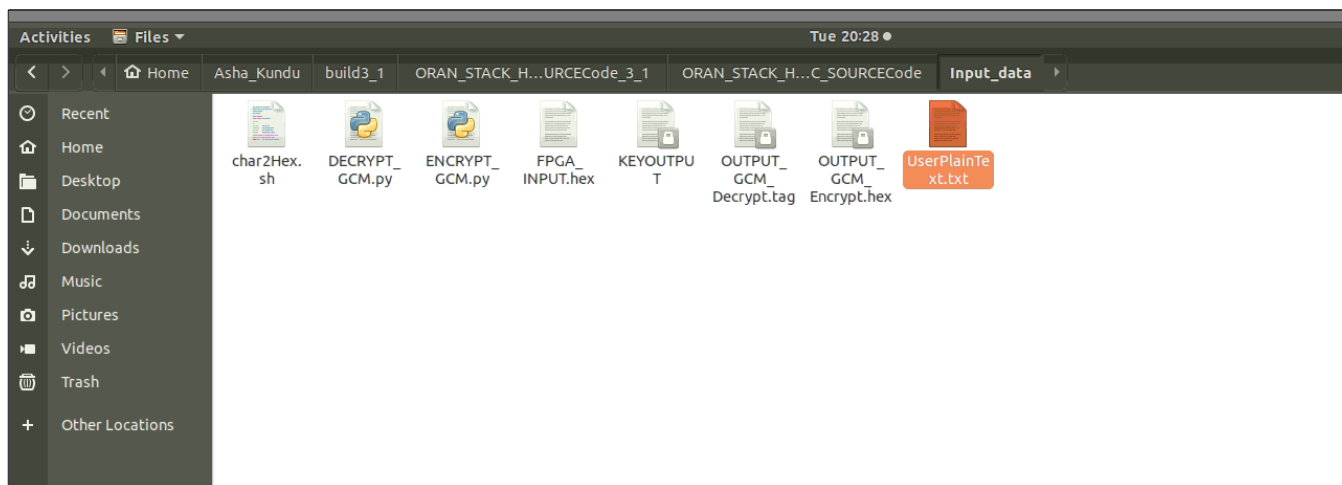


Figure 8.33. Input Decryption in inputdata.txt

2. Enter 6 to select AES-GCM 256 Decryption PCIe to UART Algorithm to generate decryption in Docklight.

```
1. SHA 384 PC to UART
2. SHA 384 UART to PC
3. HMAC 384 PC to UART
4. HMAC 384 UART to PC
5. AES-GCM 256 ENCRYPTION
6. AES-GCM 256 DECRYPTION
7. Exit
6
You have selected AES - GCM 256 for DECRYPTION
Key= 69f6b8e2e714474f10d24f6836bb186480ba01a4af5ae8295bcf328c95cbd316
IV = 73a8ed90ca45d2f3780abfb1e7cabfffa
AAD= 3591d066d5e9dbfe25affce9214a6c14
TAG = b'5fd851a7bf21a878e5e32680e8cd021d'

User Input File Exist...
main: AES : read
FPGA linkup Read Ready status : 0x1
-----DMA Counter Read Operation-----
No. of descriptor will be 238
Burst Size is greater than 120****Need to be handle*****
TotalWritelen : 60832 Writelen:60832 RemainingBytes:0 WriteReqLen:60832
/home/lft/SG_ORAN/SG_ORAN_REL/ORAN_STACK_HOSTPC_BinaryRelease_5_0
libspdm_send_spdm_request[0] (0x4):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0xc):
.
libspdm_receive_spdm_response[0] (0xff):
libspdm_send_spdm_request[0] (0x30):
.
libspdm_receive_spdm_response[0] (0xff):
SPDM Handshaking executed successfully
```

Figure 8.34. User Selecting AES-GCM 256 Decryption Algorithm

3. Check the generated decrypted function in Docklight after successfully executing the algorithm.

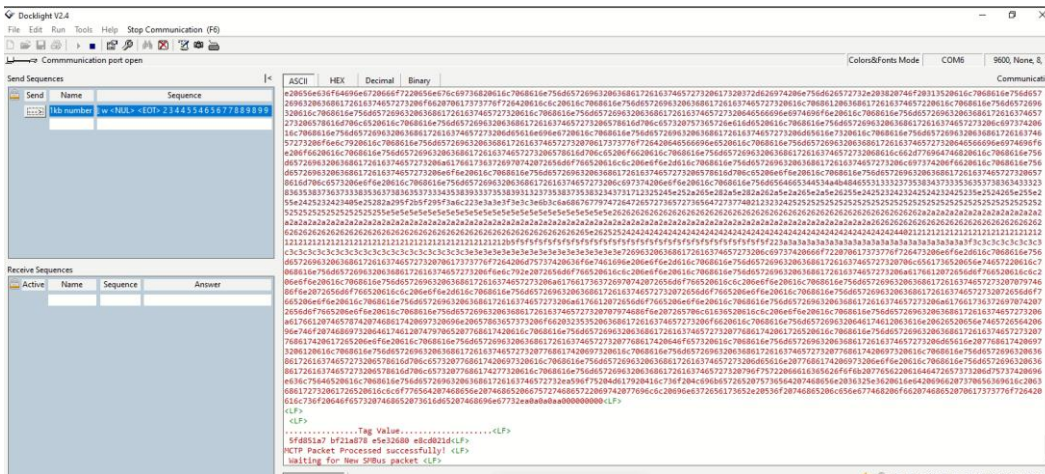


Figure 8.35. Decryption over Docklight

4. Open the *Output_GCM_Encrypt.tag* file in *Output_data* folder.
5. Match the data tag value in Docklight.
6. Open the *FPGA_INPUT.hex* file in the *Output_data* folder.
7. Match the Docklight data given in hex with the *FPGA_INPUT.hex* file.

Appendix A – Programming the Flash

Programming SPI Flash on CertusPro-NX Board (with Macronix Flash)

This section provides the procedure for programming Bitstream into the SPI Flash on the CertusPro-NX Versa Board for the 5G ORAN Demo System.

To program SPI Flash in Lattice Radiant Programmer:

1. Turn ON the CertusPro-NX Versa Board by giving power supply through the PCIe cable /12-V adapter.
Connect the USB cable for programming through Lattice Radiant Programmer.
2. Start Lattice Radiant Programmer. In the **Getting Started** dialog box, select **Create a new blank project**.

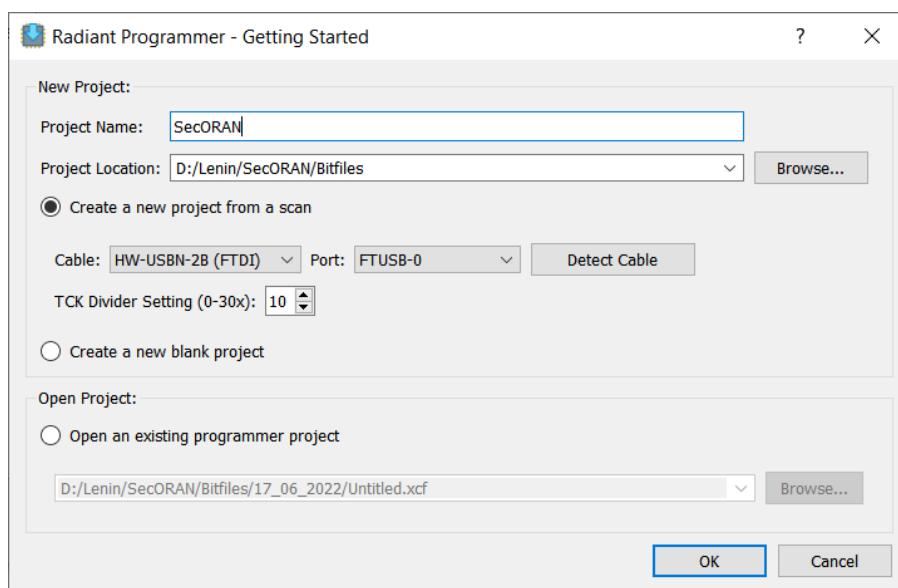


Figure A.1. Lattice Radiant Programmer Getting Started Dialog Box

3. Click **OK**.

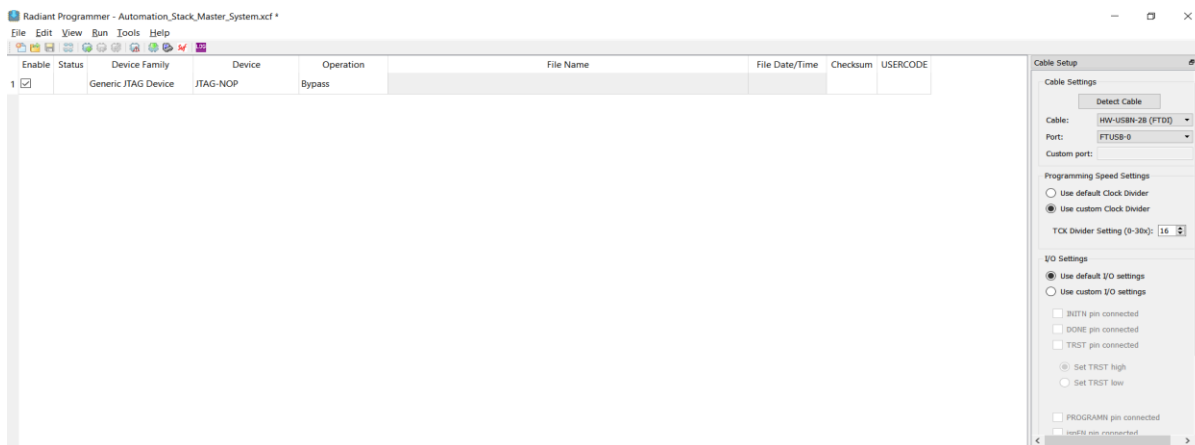


Figure A.2. Lattice Radiant Programmer- Main Interface

- In the main interface, select **LFCPNX** for **Device Family** and **LFCPNX-100** for **Device** or detect automatically as shown in [Figure A.3](#).

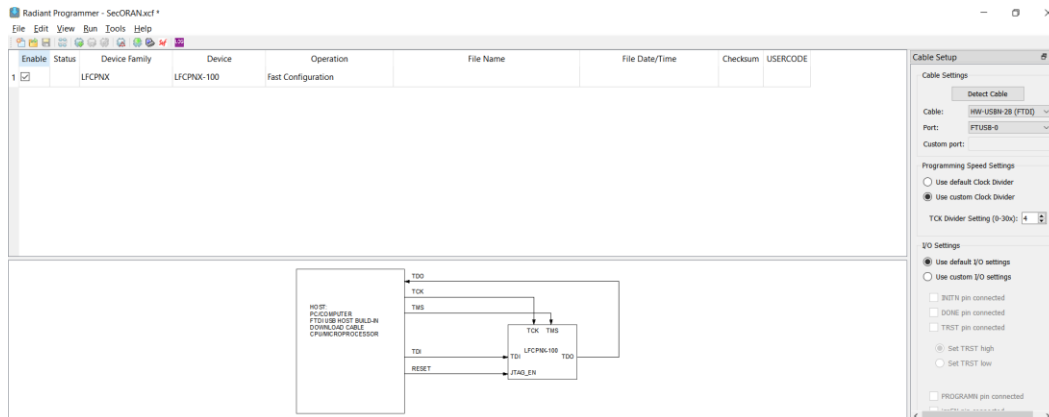


Figure A.3. Radiant Programmer- Device Selection

- Right-click and select **Device Properties**.

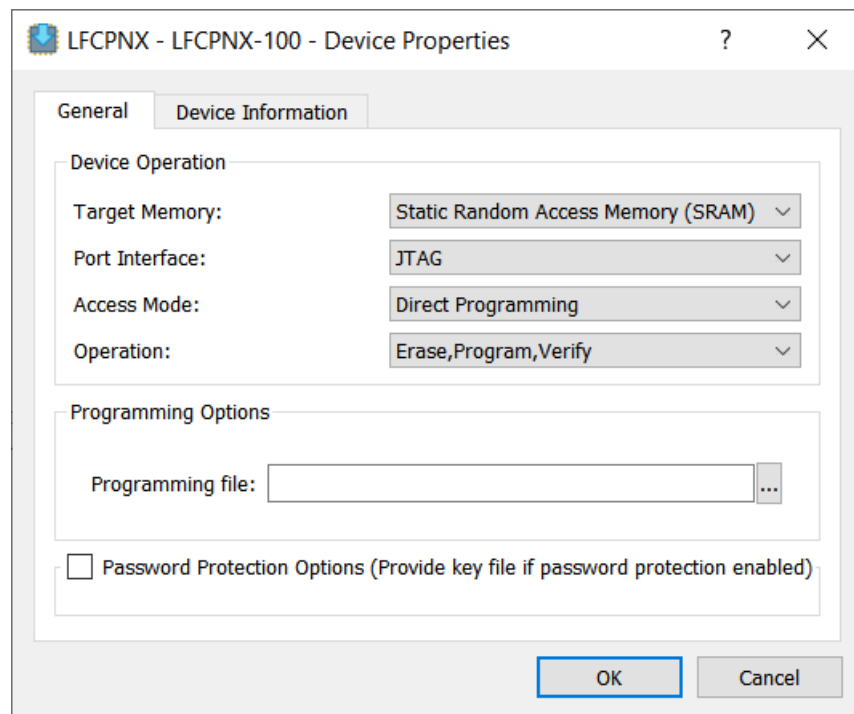


Figure A.4. Lattice Radiant Programmer- Device Properties

6. Before programming, erase the Flash memory.

To erase Flash memory:

- a. Under **Device Operation**, select the options below:

Target Memory – External SPI Flash Memory (SPI FLASH)

Port Interface – JTAG2SPI

Access Mode – Direct Programming

Operation – Erase all

- b. Under **SPI Flash Options**, select the options below:

Family – SPI Serial Flash

Vendor – Macronix

Device – MX25L51245G

Package – 8-land WSON

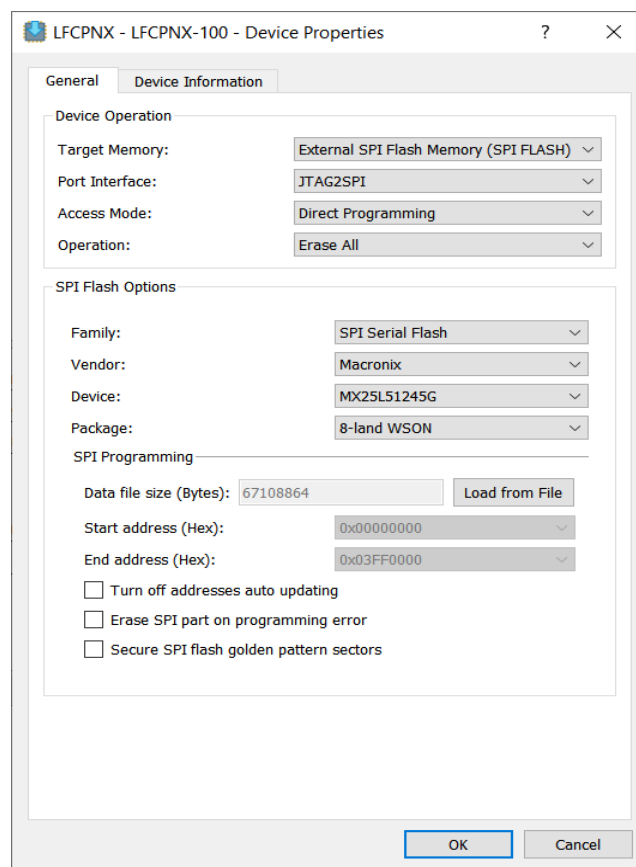


Figure A.5. Erase All (Lattice Radiant Programmer)

7. Click **OK**.
8. Click **Program Device** and wait for the process to successfully complete. This erases the flash memory.

9. After erasing the Flash, power cycle the board and apply the settings below:
 - a. Under **Device Operation**, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)**
 - Port Interface – JTAG2SPI**
 - Access Mode – Direct Programming**
 - Operation – Erase, Program, Verify**
 - b. Under **SPI Flash Options**, select the options below:
 - Family – SPI Serial Flash**
 - Vendor – Macronix**
 - Device – MX25L51245G**
 - Package – 8-land WSON**
10. To program the bitstream file, select the options as shown in [Figure A.6](#).

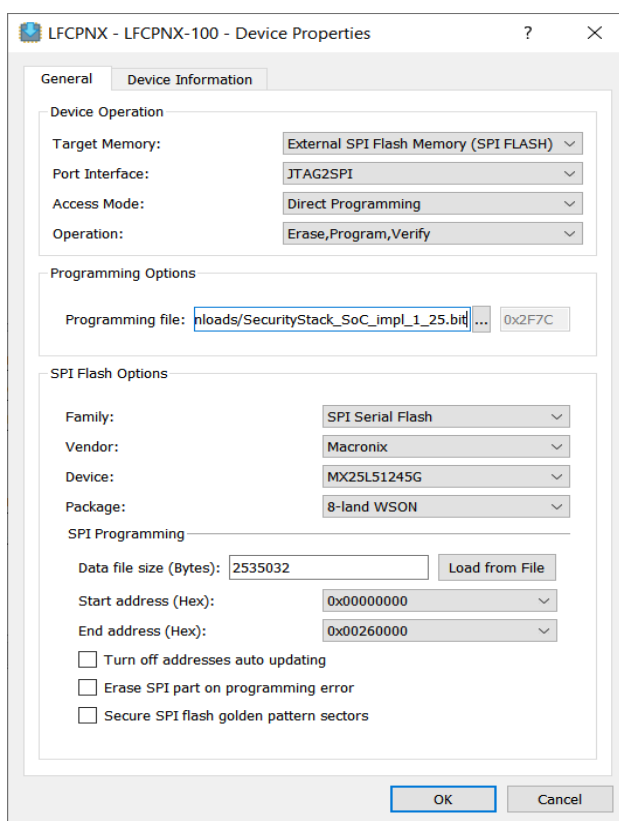


Figure A.6. Lattice Radiant Programmer-Bitstream Flashing

11. Select the *.bit* file that needs to be loaded in the **Programming file** option
12. Click **Load from File** to update the Data file size (Bytes) value.
13. Ensure that the following addresses are correct:
 - Start Address (Hex) – 0x00000000
 - End Address (Hex) – 0x00200000
14. Click **Program Device** and wait for the process to successfully complete.
15. Power cycle the CertusPro-NX Versa Board.

Programming SPI Flash on CertusPro-NX Rev-B Board (with Winbond Flash)

This section provides the procedure for programming files into the SPI Flash on the CertusPro-NX Versa Board Rev B with Winbond flash.

To program SPI Flash in Lattice Radiant Programmer:

1. Turn ON the CertusPro-NX Versa Board by giving power supply through the PCIe cable /12-V adapter.
Connect the USB cable for programming through the Lattice Radiant Programmer.
2. Start Lattice Radiant Programmer. In the **Getting Started** dialog box, select **Create a new blank project**.

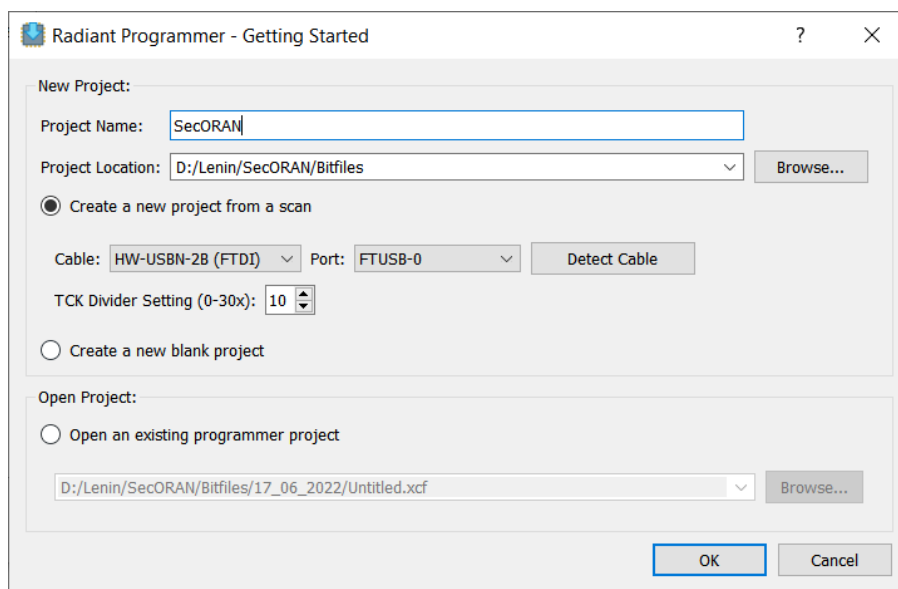


Figure A.1. Lattice Radiant Programmer Getting Started Dialog Box

3. Click **OK**.

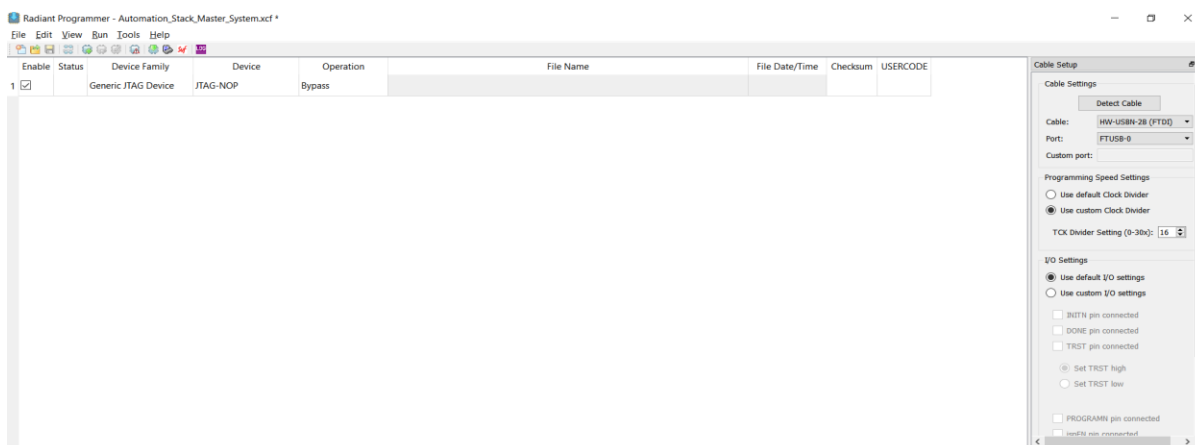
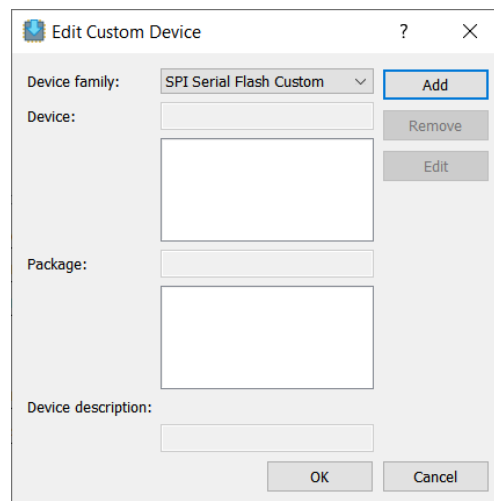


Figure A.2. Lattice Radiant Programmer- Main Interface

- The screenshot shows the Radiant Programmer interface for the LFCPNX-100 device. The main window displays the device name, operation (Fast Configuration), and a table with columns for File Name, File Date/Time, Checksum, and USERCODE. The I/O Settings panel on the right shows the following configuration:

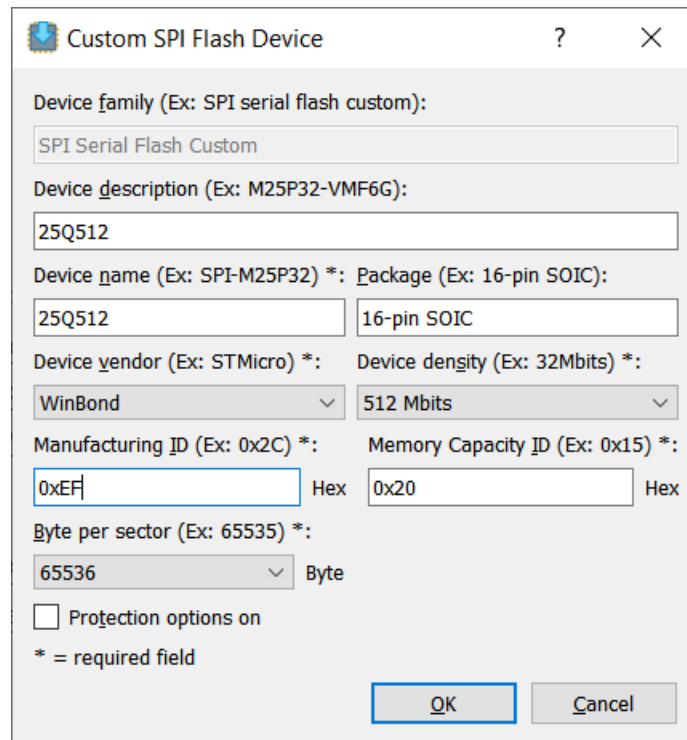
 - Cable Setup:**
 - Cable: HW-USB-2B (FTDI)
 - Port: JTAG-0
 - Custom port: (empty)
 - Programming Speed Settings:**
 - ☐ Use default Clock Divider
 - ☒ Use custom Clock Divider
 - TCK Divider Setting (0-30x): 10x
 - I/O Settings:**
 - ☒ Use default I/O settings
 - ☐ Use custom I/O settings
 - ☐ INITN pin connected
 - ☐ DONE pin connected
 - ☐ TRST pin connected
 - ☒ Set TRST high
 - ☐ Set TRST low
 - ☐ PROGRAM pin connected

5. Select **Tools > Custom Flash Device**.
6. The **Edit Custom Device** dialog box opens.



7. Click **Add**.

8. The **Custom SPI Flash Device** dialog box opens.
9. Apply the settings shown in [Figure A.5](#).

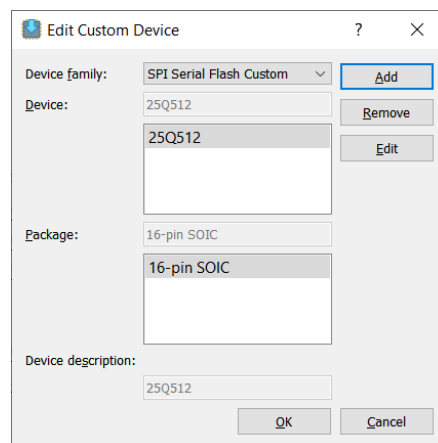


The dialog box titled "Custom SPI Flash Device" contains the following fields and controls:

- Device family (Ex: SPI serial flash custom):** A text box containing "SPI Serial Flash Custom".
- Device description (Ex: M25P32-VMF6G):** A text box containing "25Q512".
- Device name (Ex: SPI-M25P32) *:** A text box containing "25Q512".
- Package (Ex: 16-pin SOIC):** A text box containing "16-pin SOIC".
- Device vendor (Ex: STMicro) *:** A dropdown menu showing "WinBond".
- Device density (Ex: 32Mbits) *:** A dropdown menu showing "512 Mbits".
- Manufacturing ID (Ex: 0x2C) *:** A text box containing "0xEF" with a "Hex" label.
- Memory Capacity ID (Ex: 0x15) *:** A text box containing "0x20" with a "Hex" label.
- Byte per sector (Ex: 65535) *:** A dropdown menu showing "65536" with a "Byte" label.
- Protection options on:** An unchecked checkbox.
- * = required field**
- Buttons:** "OK" and "Cancel".

Figure A.5. Custom SPI Flash Device Settings

10. Click **OK**.
11. The **Edit Custom Device** dialog box opens.



The dialog box titled "Edit Custom Device" contains the following fields and controls:

- Device family:** A dropdown menu showing "SPI Serial Flash Custom" with an "Add" button.
- Device:** A list box containing "25Q512" with "Remove" and "Edit" buttons.
- Package:** A list box containing "16-pin SOIC".
- Device description:** A text box containing "25Q512".
- Buttons:** "OK" and "Cancel".

Figure A.6. Edit Custom Device Settings

12. Select Device **25Q512**.
13. Click **OK**.

14. Right-click and select **Device Properties**.

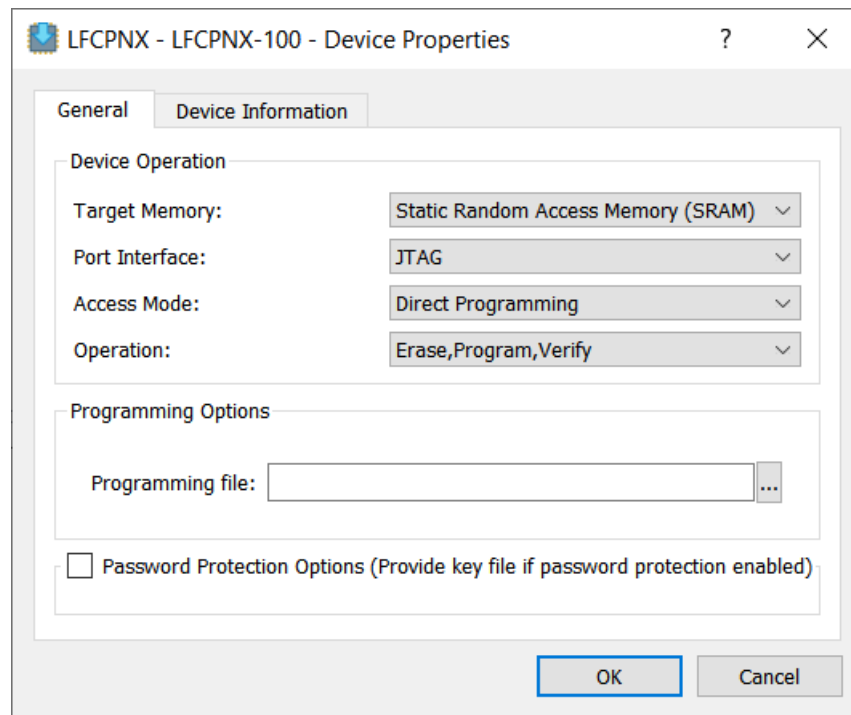


Figure A.7. Lattice Radiant Programmer- Device Properties

15. Before programming, erase the Flash memory.

To erase Flash memory:

- a. Under **Device Operation**, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)**
 - Port Interface – JTAG2SPI**
 - Access Mode – Direct Programming**
 - Operation – Erase all**
- b. Under **SPI Flash Options**, select the options below:
 - Family – SPI Serial Flash**
 - Vendor – Macronix**
 - Device – MX25L51245G**
 - Package – 8-land WSON**

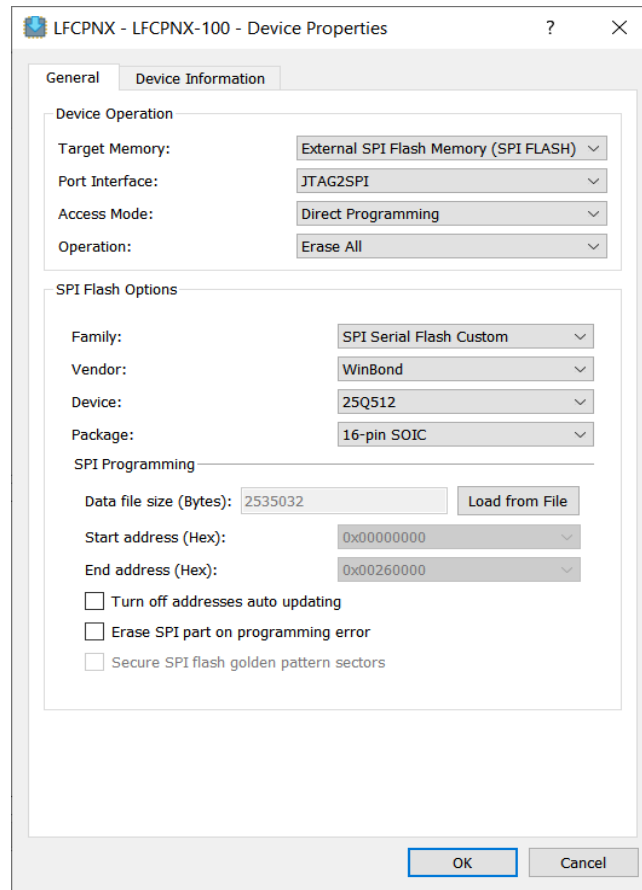


Figure A.8. Erase All (Lattice Radiant Programmer)

16. Click **OK**.
17. Click **Program Device**. This erases the flash memory.
18. After erasing the Flash, power cycle the board and apply the settings below:
 - a. Under **Device Operation**, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)**
 - Port Interface – JTAG2SPI**
 - Access Mode – Direct Programming**
 - Operation – Erase, Program, Verify**
 - b. Under **SPI Flash Options**, select the options below:
 - Family – SPI Serial Flash**
 - Vendor – Winbond**
 - Device – 25Q512**
 - Package – 16-pin SOIC**

19. To program the bitstream file, select the options as shown in Figure A.6.

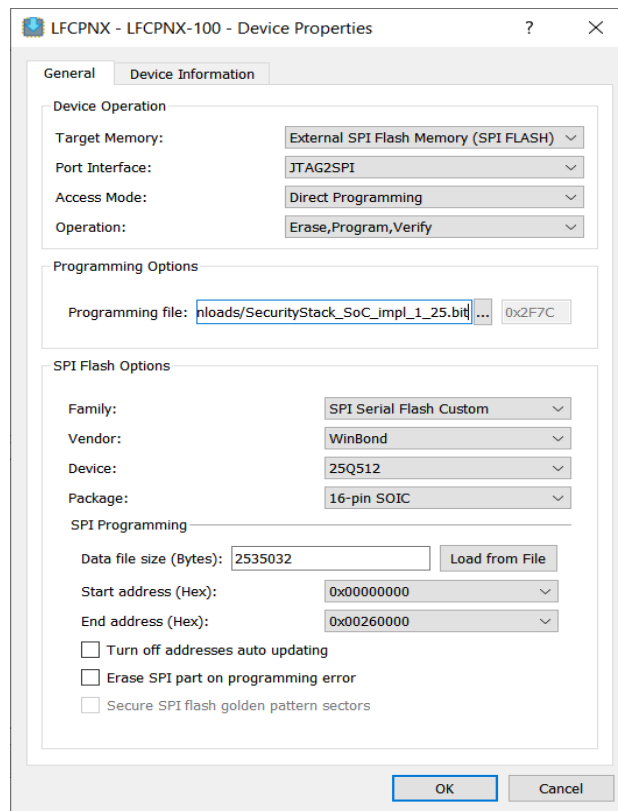


Figure A.9. Lattice Radiant Programmer-Bitstream Flashing

20. Select the *.bit* file that needs to be loaded in the **Programming file** option.
21. Click **Load from File** to update the Data file size (Bytes) value.
22. Ensure that the following addresses are correct:
Start Address (Hex) – 0x00000000
End Address (Hex) – 0x00260000
23. Click **Program Device** and wait for the process to successfully complete. This erases the flash memory.
24. Power cycle the CertusPro NX Versa Board.

Appendix B. Generating the Security Stack SoC Bit File

Note: Make sure that Lattice Control Pack for CRE (3.1.0.43.2_Radiant_Ctrl_Pack_CRE.exe) is installed in the PC where Lattice Radiant Software is installed to avoid Synthesis errors due to CRE IP and CRE OSC IP. Also, make sure that the Lattice Radiant License is updated for the CRE, PCIe, SMBus, and OSE IPs to avoid Map and Bitstream generation errors.

To generate the Security Stack SoC bit file:

1. Go to the Security Stack SoC project directory and open the *SecurityStack_SoC.sbx* file in the *SecurityStack_SoC_75Mhz/SecurityStack_SoC* folder using Lattice Propel™ Builder 2.2.

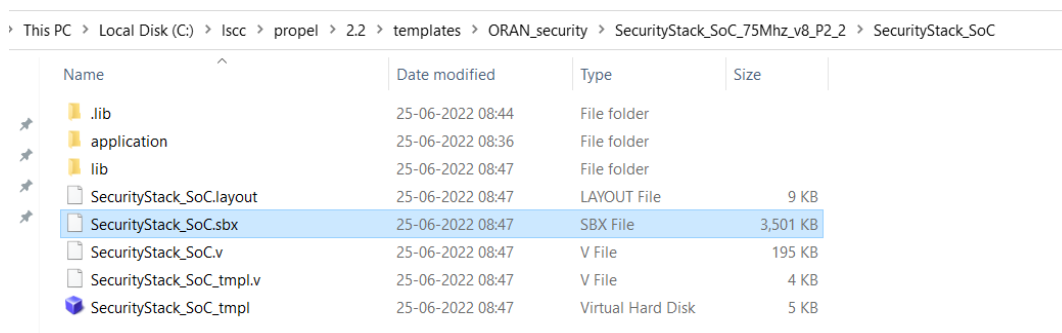


Figure A.10. Opening the Complete Security Stack SoC Project

2. The complete Security Stack SoC project opens in Lattice Propel Builder 2.2 as shown in Figure A.11. Double click on both CPUs. Disable the Simulation Mode and enable the Debug Mode only in the Application CPU.

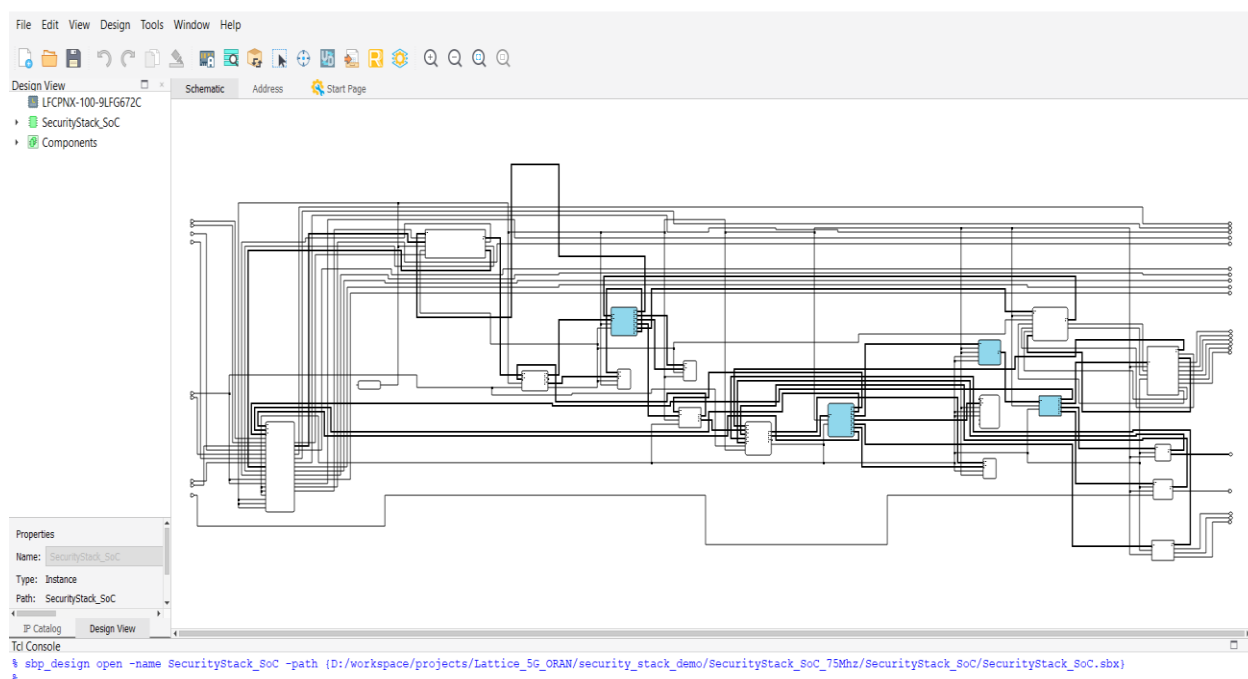


Figure A.11. Complete Security Stack SoC Project

- Load the Application CPU firmware in the Application CPU system memory (module instantiation name=app_system) as shown in Figure A.12.

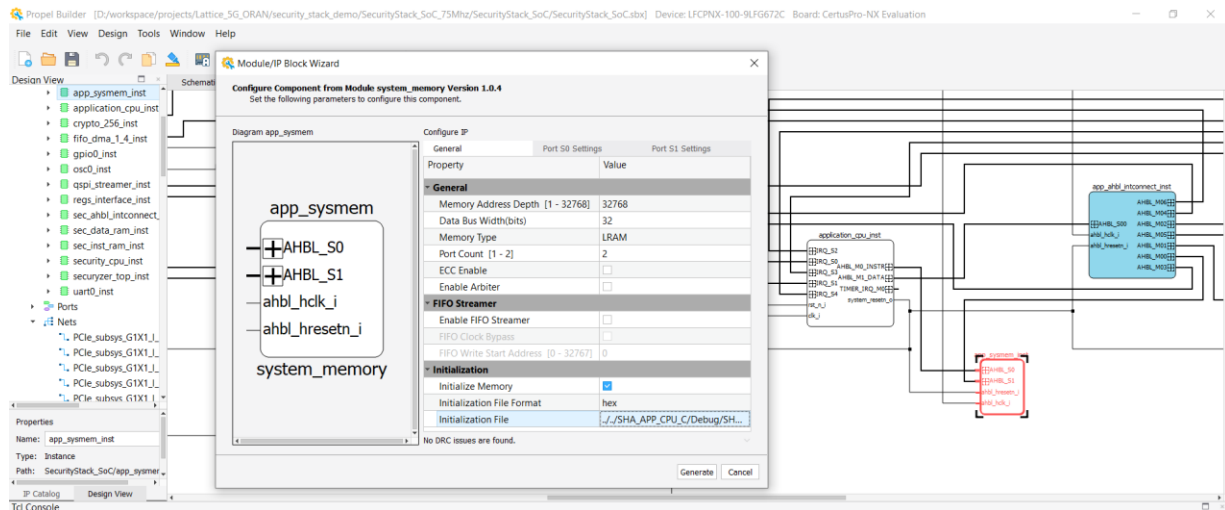


Figure A.12. Loading the Application CPU Firmware

Select the Application CPU firmware file (*.mem) from the debug location of the Application CPU's C project created with the complete design.

This PC > Local Disk (C:) > Iscc > propel > 2.2 > templates > ORAN_security > ORAN_APP_CPU > Debug

Name	Date modified	Type	Size
sources.mk	25-06-2022 08:44	MK File	1 KB
ORAN_APP_CPU.mem	25-06-2022 04:09	MEM File	42 KB
ORAN_APP_CPU.map	25-06-2022 04:09	MAP File	101 KB
ORAN_APP_CPU.lst	25-06-2022 04:09	LST File	573 KB
ORAN_APP_CPU.elf	25-06-2022 04:09	ELF File	197 KB
ORAN_APP_CPU.bin	25-06-2022 04:09	BIN File	19 KB
objects.mk	25-06-2022 08:44	MK File	1 KB
makefile	25-06-2022 08:44	File	3 KB

Figure A.13. Selecting the Application CPU Firmware

- Click **Generate** and then **Finish**.
- Load the Security CPU firmware in the Security CPU instruction RAM (module instantiation name=sec_inst_ram) as shown in Figure A.14.

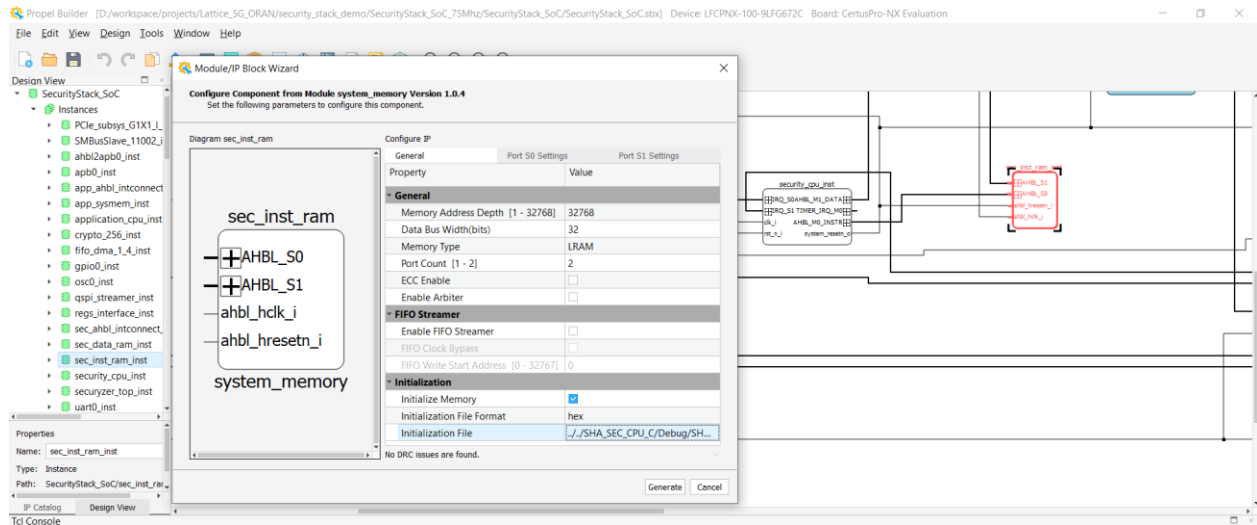


Figure A.14. Loading the Security CPU Firmware

Select the Security CPU firmware file (*.mem) from the debug location of the Security CPU's C project.

This PC > Local Disk (C:) > Iscc > propel > 2.2 > templates > ORAN_security > ORAN_SEC_CPU > Debug

Name	Date modified	Type	Size
sources.mk	25-06-2022 08:44	MK File	1 KB
ORAN_SEC_CPU.mem	25-06-2022 04:09	MEM File	36 KB
ORAN_SEC_CPU.map	25-06-2022 04:09	MAP File	111 KB
ORAN_SEC_CPU.lst	25-06-2022 04:09	LST File	406 KB
ORAN_SEC_CPU.elf	25-06-2022 04:09	ELF File	122 KB
ORAN_SEC_CPU.bin	25-06-2022 04:09	BIN File	16 KB
objects.mk	25-06-2022 08:44	MK File	1 KB
makefile	25-06-2022 08:44	File	3 KB

Figure A.15. Selecting the Security CPU Firmware

- Click **Generate** and then **Finish** as shown in [Figure A.14](#).
- Once both CPU firmware are loaded, save the design.
- Click **Validate Design** and select options for generating the base address, running DRC, and generating the platform.

Top File Change for Bit File Generation

To change top file for generating the .bit file:

1. After validation, the *SecurityStack_SoC.v* Verilog is generated in the project directory *SecurityStack_SoC_75MHz/SecurityStack_SoC*.

Name	Date modified	Type	Size
.lib	01-05-2022 11:42	File folder	
application	01-05-2022 11:42	File folder	
lib	03-05-2022 22:28	File folder	
SecurityStack_SoC	03-05-2022 22:28	LAYOUT File	13 KB
SecurityStack_SoC	03-05-2022 22:28	SBX File	3,725 KB
SecurityStack_SoC	03-05-2022 22:28	V File	217 KB
SecurityStack_SoC_tmpl	03-05-2022 22:28	V File	4 KB
SecurityStack_SoC_tmpl	03-05-2022 22:28	VHD File	6 KB

Figure A.16. Security Stack Top File Location

2. Open the top file and comment line no 59 as shown in Figure A.17.

Note: This step is required only when the user wants to regenerate the bit file with the new mem files of the RISC V CPUs.

```

34 // SOLE LIABILITY, AND LICENSEE'S SOLE REMEDY, IS SET FORTH ABOVE.
35 // LATTICE DOES NOT WARRANT OR REPRESENT THAT THIS FILE, ITS CONTENTS OR
36 // USE THEREOF DOES NOT INFRINGE ON THIRD PARTIES' INTELLECTUAL PROPERTY
37 // RIGHTS, INCLUDING ANY PATENT. IT IS THE USER'S RESPONSIBILITY TO VERIFY
38 // THE USER SOFTWARE DESIGN FOR CONSISTENCY AND FUNCTIONALITY THROUGH THE
39 // USE OF FORMAL SOFTWARE VALIDATION METHODS.
40 // -----
41
42 /* synthesis translate_off*/
43 `define SBP_SIMULATION
44 /* synthesis translate_on*/
45 `ifndef SBP_SIMULATION
46 `define SBP_SYNTHESIS
47 `endif
48
49 `include "../lib/latticesemi.com/ip/CRE_OSC_75Mhz/1.1.0/rtl/CRE_OSC_75Mhz.v"
50 //`include "../lib/latticesemi.com/ip/OSF_top/1.0.1/rtl/OSF_top.v"
51 `include "../lib/latticesemi.com/ip/PCIe_subsys_G1X1_1/2.1.0.13/rtl/PCIe_subsys_G1X1_1.v"
52 `include "../lib/latticesemi.com/ip/RSTN_Sync/1.1.0/rtl/RSTN_Sync.v"
53 `include "../lib/latticesemi.com/ip/SMBusSlave_11002/1.1.0.02/rtl/SMBusSlave_11002.v"
54 `include "../lib/latticesemi.com/ip/app_sysmem/1.1.2/rtl/app_sysmem.v"
55 `include "../lib/latticesemi.com/ip/application_cpu/2.2.0/rtl/application_cpu.v"
56 `include "../lib/latticesemi.com/ip/crypto_256/1.1.2/rtl/crypto_256.v"
57 `include "../lib/latticesemi.com/ip/regs_interface/1.0.3/rtl/regs_interface.v"
58 `include "../lib/latticesemi.com/ip/sec_inst_ram/1.1.2/rtl/sec_inst_ram.v"
59 `include "../lib/latticesemi.com/ip/security_cpu/2.2.0/rtl/security_cpu.v"
60 `include "../lib/latticesemi.com/ip/uart0/1.3.0/rtl/uart0.v"
61 `include "../lib/latticesemi.com/module/ahbl2apb0/1.1.0/rtl/ahbl2apb0.v"
62 `include "../lib/latticesemi.com/module/apb0/1.1.0/rtl/apb0.v"
63 `include "../lib/latticesemi.com/module/app_ahbl_intconnect/1.3.0/rtl/app_ahbl_intconnect.v"
64 `include "../lib/latticesemi.com/module/sec_ahbl_intconnect/1.3.0/rtl/sec_ahbl_intconnect.v"
65
66

```

Figure A.17. Top File Change for .bit File Generation

AES Mode Selection before Generating Bit File

To select AES Mode:

1. To select either AES-CBC 256 Mode or AES-GCM 256 Mode, go to the Security Stack_SoC project directory and open *Security_stack_SoC_75Mhz/ /OSE_rtl_E/OSE.vhd*.

This PC > Local Disk (C:) > Iscc > propel > 2.2 > templates > ORAN_security > SecurityStack_SoC_75Mhz_v8_P2_2 > OSE_rtl_E

Name	Date modified	Type	Size
acpu_lib	25-06-2022 04:09	Virtual Hard Disk	288 KB
ahbl_interconnect	25-06-2022 04:09	Virtual Hard Disk	10 KB
block_cipher_hybrid_wrapper_CBC	25-06-2022 04:09	Virtual Hard Disk	580 KB
block_cipher_hybrid_wrapper_GCM	25-06-2022 04:09	Virtual Hard Disk	614 KB
hash_function_ahb_wrapper	25-06-2022 04:09	Virtual Hard Disk	140 KB
OSE	25-06-2022 04:09	Virtual Hard Disk	25 KB
OSE_pkc_memories_e.v	25-06-2022 04:09	V File	23 KB
OSE_top.sv	25-06-2022 04:09	SV File	5 KB

Figure A.18. OSE.vhd File in Project Directory

2. Configure the generic parameter *BC_CONFIGURATION* = 1 for AES-CBC256 Mode or *BC_CONFIGURATION* = 2 for AES-GCM256 Mode as shown in Figure A.19.

Note: The FPGA default setting is AES-GCM256 Mode.

```

8
9 entity OSE is
10 generic(
11     BC_CONFIGURATION : natural := 2;
12     OSE_AXISTREAM_DATA_WIDTH : natural := 128
13 );

```

Figure A.19. Configuring the Generic Parameter for AES Mode Selection

3. Open the project using Lattice Radiant 3.1 software tool. Go to the *security_stack_SoC_75MHz* project directory and double click the *SecurityStack_SoC.rdf* file as shown in Figure A.20.

Name	Date modified	Type	Size
.externalToolBuilders	25-06-2022 08:36	File folder	
impl_1	25-06-2022 11:10	File folder	
misc	25-06-2022 08:47	File folder	
OSE_rtl_E	25-06-2022 08:36	File folder	
SecurityStack_SoC	25-06-2022 08:47	File folder	
SecurityStack_SoC_tcr.dir	25-06-2022 08:36	File folder	
sge	25-06-2022 08:47	File folder	
source	25-06-2022 08:36	File folder	
toprlcopy	25-06-2022 08:36	File folder	
verification	25-06-2022 08:36	File folder	
.ng_run_manager	25-06-2022 04:09	Configuration setti...	2 KB
.project	25-06-2022 04:09	PROJECT File	1 KB
.recovery	25-06-2022 08:48	RECOVERY File	2 KB
.setting	25-06-2022 04:09	Configuration setti...	1 KB
.socproject	25-06-2022 04:09	SOCPROJECT File	1 KB
bht_ini.bin	25-06-2022 08:47	BIN File	3 KB
drc	25-06-2022 04:09	Text Document	1 KB
radiant_setup_template.tcl	25-06-2022 08:47	TCL File	2 KB
reginit.bin	25-06-2022 08:47	BIN File	2 KB
reportview	25-06-2022 04:09	XML Document	1 KB
SecurityStack_SoC.pdc	25-06-2022 04:09	PDC File	4 KB
SecurityStack_SoC.rdf	25-06-2022 04:09	RDF File	2 KB
SecurityStack_SoC	25-06-2022 04:09	Text Document	1 KB
SecurityStack_SoC_tcl	25-06-2022 08:47	Microsoft Edge HT...	8 KB
SecurityStack_SoC1.sty	25-06-2022 08:48	STY File	17 KB
transcript	25-06-2022 04:09	File	1 KB

4. The complete project opens in the Lattice Radiant 3.1 tool without any errors, as shown in [Figure A.21](#).



- To generate bit file in AES-CBC Mode, exclude the *block_cipher_hybrid_wrapper_GCM.vhd* file from the project by right-clicking the file and selecting **Exclude from implementation**.

Similarly, to generate bit file in AES-GCM Mode, exclude the *block_cipher_hybrid_wrapper_CBC.vhd* as shown in Figure A.22.

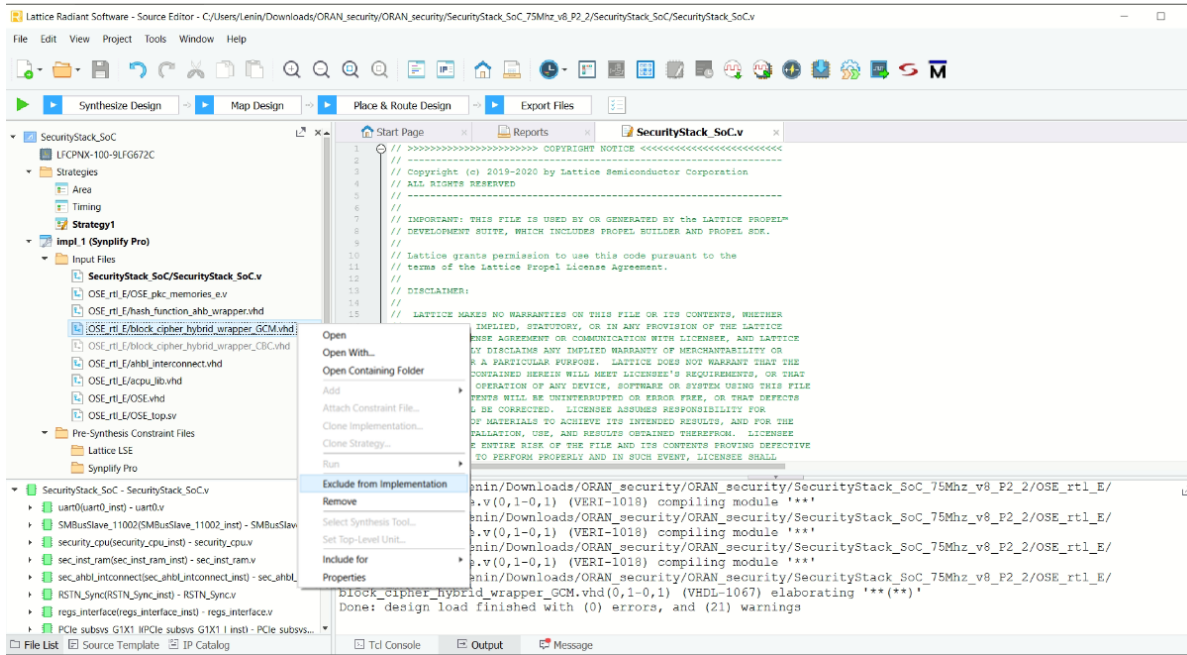


Figure A.22. Exclude Block_cipher_hybrid_wrapper.vhd File from Project

- Click **Run** to generate the bit file.
- The bit file is generated and saved in the project *impl_1* directory with green mark as shown in Figure A.23.

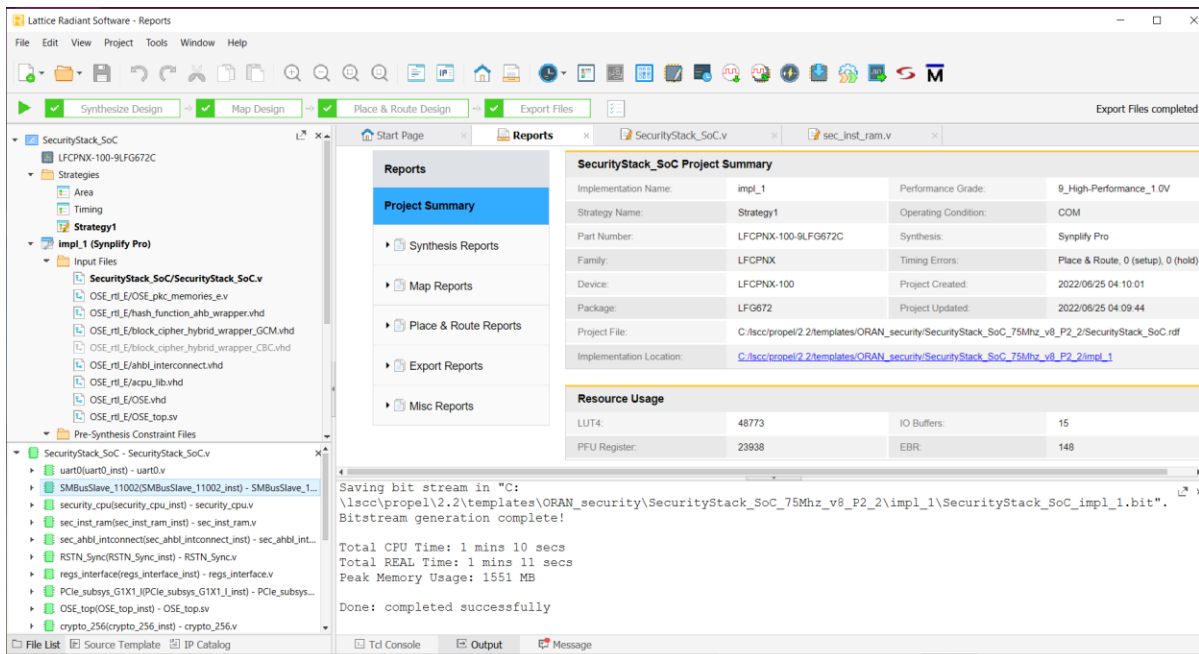


Figure A.23. Security Stack_SoC Project after Synthesis and Bit File Generation

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.0, June 2022

Section	Change Summary
	Initial release.



www.latticesemi.com