Assumptions - Romulus-N 1.3 Hardware Implementation

Aadam and Hawa Dirie

December 2021

Contents

1 Source List Assumptions

Unclear on bottom-up. In mine it goes from the lower modules and wrappers or other files that instantiate that communicate with modules below show up as you go from the bottom to the top of the *source_list.txt* file.

2 Targeted Device

Targeted Device where all synthesis and implementation was run was on the Artix 7 family of FPGA denoted xc7a35tcsg324-1.

3 Datapath Assumptions

- Current implementation of Datapath assumes that PostProcessor will handle masking of output blocks provided from Datapath other than Tag (In Romulus-N 1.3, Tag is always 128 bits so no need to worry about clearing certain bytes from words). This function is defined in the algorithm as a *lsb* or *msb* operation (due to endianness, this operation is slightly different)
- Current implementation of Datapath assumes Control module will need to provide an encoded signal $len8$. This $len8$ signal is essentially a one byte encoding of the number of the number of bytes of each input block of the type plaintext or ciphertext. The current assumption is that we can leverage the use of bdi_size which is provided by the pre-processor in the LWC API in order to provide this signal

4 Test-Bench Assumptions

Here we describe our assumptions made for our attempted verification of the entirety of the datapath, as well as our successful verification of Skinny-128-384+, the Tweakable Block Cipher (TBC) as utilized in the Romulus-N 1.3 variant.

4.1 E_K_Controller_TB

The E_K_Controller_TB successfully verifies our Skinny-128-384+. This was done with the help of some generics in order to manually encode an entire Tweakey since the TBC used within the CryptoCore only sets the tweakey arbitrarily based on certain inputs. This TestBench verifies the test vectors given in the Romulus-N 1.3 Specification where after 40 rounds we were able to match the plaintext and tweakey input to the correct ciphertext output. However, this testbench only operates successfully with the generic of Testing set to 1.

4.2 CipherCore_Datapath_TB

Here were the detailed assumptions made in the design of this TestBench was that we attempted to verify the datapath with

- CipherCore Testbench assumes only complete blocks of associated data and other potential data inputs that have variable sizes plaintext and ciphertext
- CipherCore Testbench assumes that number of blocks are even for both associated data, plaintext, and ciphertext blocks
- CipherCore Testbench does not attempt to verify manually through the use of a more advanced testbench each word of blocks produced in the output phase, instead it is done through manual verification aided with modified software implementations of Romulus-N 1.3 provided by designers

5 ASM Assumptions

- The ASM chart, unlike the attempted verification of the CipherCore Datapath, assumes that the input blocks of the type associated data, plaintext, and ciphertext (for decryption) need to have padding
- The ASM chart for main controller in its current implementation also assumes that padding or clearing of certain bytes from bdo during output of data blocks are handled by post-processor or another kind of module. I did not have enough time to implement it fully, made a very very rough draft on paper but have a more polished version of that in the actual code for the controller.