Results - Romulus-N 1.3 Hardware Implementation

Aadam and Hawa Dirie

December 2021

Contents

Acronyms

FFs Flip Flops

$LUTs$ Lookup Tables

$Slices$ Slices

1 Resource Utilization

Somewhat unclear on LUT-FF Pairs and how they are designated within the reports. So within this folder we included both the synthesis and implementation reports. This is all based upon Synthesis and Implementation when the datapath is set to the top. Similiarly to my peer student Omar, when doing synthesis on the preliminary CryptoCore which interfaced between the datapath and the controller we realized that we were getting lower utilization which does not make sense. SO the tool must have been reducing utilization within those blocks as signals and full logic has not been implemented.

1.1 LUTs

877 Lookup Tables ($LUTs$)

1.2 Slices

Total of 333 Slices ($Slices$), 212 SLICEL and 121 SLICEM.

1.3 Flip Flops

1388 Flip Flops (FFs)

1.4 LUT-FF pairs

Not exactly sure from report (Not Familiar with how this is designated).

1.5 BRAMS

None as per spec

1.6 DSP Units

None as per spec

1.7 I/O Pins

128 I/O Pins

2 Timing Parameters

2.1 Minimum Clock Period

$4.25ns$

2.2 Maximum Clock Frequency

235.2 MHz

2.3 Maximum Throughput

2.3.1 Associated Data Blocks

Approximately 716.8 $Mbit/s$

2.3.2 PlainText Blocks

Approximately 654.4 $Mbit/s$

2.3.3 CipherText Blocks

Approximately 654.4 $Mbit/s$

3 Observations and Conclusions

Some Observations was that as more files are added to vivado, it seems that the synthesis tool has a hard time designating real resource utilization. We attempted to modify the compilation order to no avail as we noticed that it would end in the same result. Moreover, since this is the preliminary results based off of the datapath alone, we know that it is most likely be around these values. As for the results, we attach the screenshot of the simulation verifying our Skinny-128-384+ implementation in Figure.1 with test vectors Figure.2. We also attach a simulation screenshot of our CipherCore_Datapath_TB that attempts to verify the entirety of the datapath however falls short as it seems like some underlying function like the Rho function or Tweakey encoding has an issue which causes subsequent Skinny TBC call to yield non-correct values. However, all other functionality works such as loading message blocks, key, Nonce, and Tag as shown in Figure.3.

4 Result Figures

Figure 1: Simulation Verifying Skinny-128-384+ Implementation

```
/* Skinny-128-384+ */
Key:        df889548cfc7ea52d296339301797449
            ab588a34a47f1ab2dfe9c8293fbea9a5
            ab1afac2611012cd8cef952618c3ebe8
Plaintext:  a3994b66ad85a3459f44e92b08f550cb
Ciphertext: ff38d1d24c864c4352a853690fe36e5e
```

Figure 2: Skinny-128-384+ Test Vectors Verified in Simulation

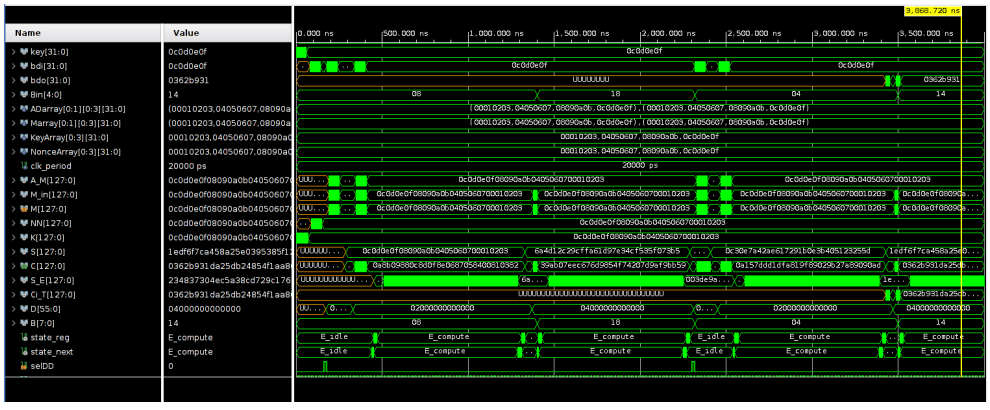

Figure 3: Simulation of CipherCore_Datapath_TB