Assignment 3: Hardware Design of a Quantum-Secure Cryptosystem

Shantanu Mukhopadhyay
[200541793]

Venkatesh Rajagopalan
[200540422]

October 16, 2024

1 Analyze the algorithm

Q1. Please describe the parameter set requested from you, write down the security level, polynomial degrees, and the coefficient's modulo value.

Ans: The security of the scheme is based on the hardness of Ring-Binary LWE (R-BinLWEEnc-II), a variant of learning with error problems. Our security analysis indicates a security level of **84 bits** for an implementation **polynomial of degree 255** and the **coefficient's modulus of 256** (n = 256, q = 256).

Q2. What is the resulting message, secret key, public key, and ciphertext size? Given the algorithm description, which polynomials refer to the secret key, encoded message, and decoded message, and which ones are the ciphertext polynomials?
Ans:
According to the reference:
Size of the Resulting message: 256 bits
Size of secret key: 256 bits
Size of public key: 2048 bits
Size of Ciphertext: 4096 bits

Polynomials referring to secret key: r2
Polynomials referring to encoded message: $\overline{m}$
Polynomials referring to decoded message: m
Polynomials referring to ciphertext: c1 and c2

2 Hardware Specifications

Q3: What is the hardware optimization goal assigned to you by the TA?
Ans: We were assigned an **area-cost** optimization metric for our FPGA implementation.

Q4: Check and make sure that the output of your hardware matches those given in the testbench. Did you achieve a functionally correct design?

Ans: Yes, we have achieved a functionally correct design with most of the test vectors matching the expected output.

When the range for the threshold function was in ranges (64,192), we found out that the following test vectors had slight mismatches marked by "_x_"

Expected: m[3] `256'hE258A9A14BF88AF77E50C_F_E0A27F35EE8C041F0AB_7_C52FBBCDC7328F93AD56FB;`
Actual: m[3] = `256'hE258A9A14BF88AF77E50C_E_E0A27F35EE8C041F0AB_6_C52FBBCDC7328F93AD56FB;`

Expected: m[4] = `256'hDFF7_7_C878F7F57D877523ACA99ADFB1FE09BE91E309D3459E575809B6106D9DF;`
Actual: m[4] = `256'hDFF7_3_C878F7F57D877523ACA99ADFB1FE09BE91E309D3459E575809B6106D9DF;`

These discrepancies arise due to the errors in the test vectors not matching the required threshold

function range, the given test vector expects us to use the range [64,192) and if we use the range [64,192] then there would be mismatch in the m[2]. We can rectify this by either using the ranges described in the assignment (64,192) and rectify our test vectors in the the testbench file or either changing the threshold function range to [64,192) resulting 1 and other to 0. After this we were able to achieve the functionally correct design.

3 Compile and Evaluate the Hardware

Q5: Synthesize, place, and route your design. What is the software and version you are using to carry out the synthesis, place, and route processes?

Ans: We have used the **Kintex-7 (XC7K355T : speed grade -3)** Family with project part : **xc7k355tffg901-3** to synthesize, place and route our design on **AMD-Xilinx Vivado v2023.2** software.

Q6: Did you achieve a design that is synthesized, placed, and routed with no errors? What is the hardware area cost (slices, register, LUTs, DSPs, BRAMs, etc.) of your design?

Ans: Yes, we were able to achieve a design which was successfully synthesized, placed and routed without any errors and any timing violations. The hardware area-cost pre- and post-optimization is given below:

Base-line synthesis Utilization values:

Name	Slice LUTs	Slice Registers	F7 Muxes	F8 Muxes	Bonded IOB	BUFGCTRL
RBLWE	8010	8660	798	380	23	1

Table 1: Baseline Synthesis Utilization

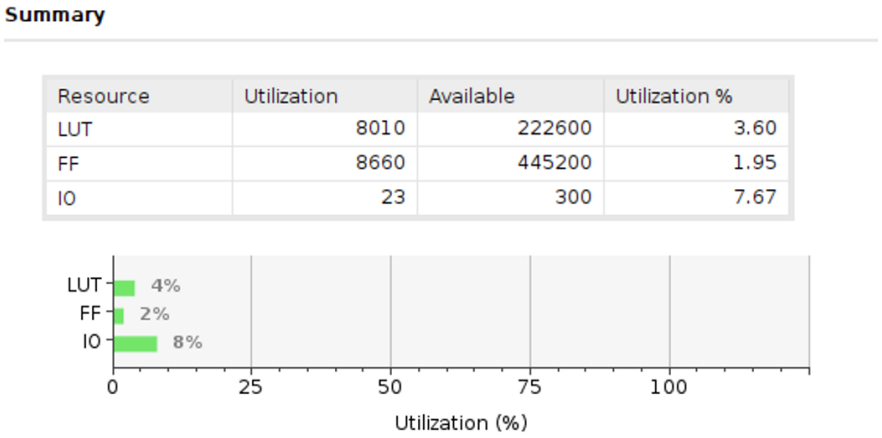

Figure 1: Baseline Utilization Summary

Ref Name	Used	Functional Category
FDRE	8656	Flop & Latch
LUT3	4377	LUT
LUT6	3205	LUT
LUT4	2338	LUT
MUXF7	798	MuxFx
MUXF8	380	MuxFx
LUT 5	120	LUT
LUT2	67	LUT
IBUF	21	IO
FDCE	4	Flop & Latch
CARRY4	4	CarryLogic
LUT1	3	LUT
OBUF	2	IO
BUFG	1	Clock

Table 2: Baseline Synthesis Primitives

Achieved optimized utilization values:

Name	Slice LUTs	Slice Registers	F7 Muxes	F8 Muxes	Slice	LUT as Logic	Bonded IOB	BUFGCTRL
RBLWE	4758	8524	730	4	3071	4758	23	1

Table 3: Optimized Post-route and Implementation Utilization

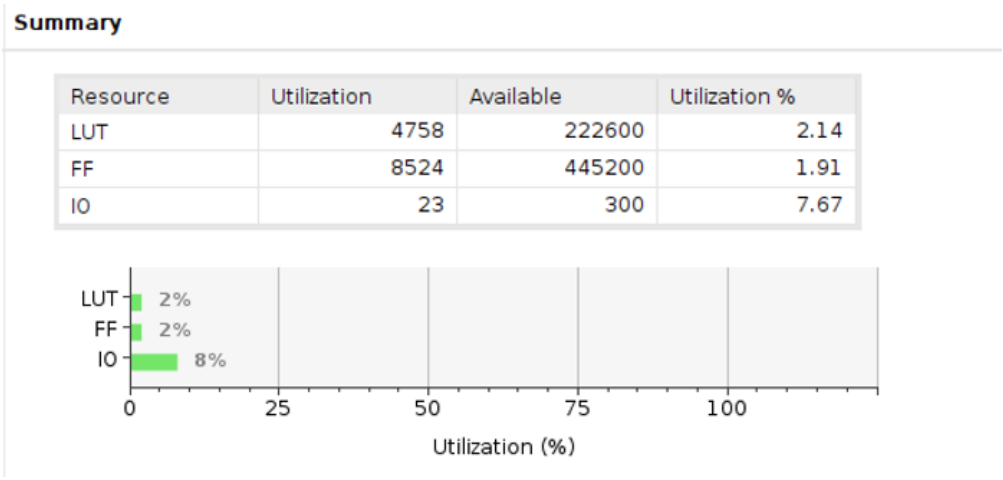

Figure 2: Optimized Utilization Summary

Ref Name	Used	Functional Category
FDRE	8520	Flop & Latch
LUT4	2402	LUT
LUT6	1973	LUT
MUXF7	730	MuxFx
LUT5	442	LUT
LUT3	140	LUT
LUT2	111	LUT
IBUF	21	IO
MUXF8	4	MuxFx
FDCE	4	Flop & Latch
CARRY4	4	CarryLogic
OBUF	2	IO
LUT1	2	LUT
BUFG	1	Clock

Table 4: Optuimized Post-Route Implmentation Primitives

Vivado settings used for area or utilization optimization:
for **Synthesis: -flatten_hierarchy** to '*rebuilt*' and **-directive*** to '*AreaOptimized_medium*'
Implementation: -directive to '*ExploreArea*'

For Simulation to see all the decryption output message signals on transcript and wave, we set the **Simulation.xsim.simulate.runtime*** to *11,930,920,000 ps*.

Q7: What is the maximum achievable frequency of your design? How many clock cycles it takes to process one decryption?

Frequency = 1/clock period = 1/10ns = **100 MHz**
Time at which the first data c1, c2 and r2 is loaded = 180 ns
Time at which the first decrypted output is displayed = 198905 ns
Thus time taken to process on decryption = 198,725 ns
One clock cycle = 10 ns
Number of clock cyles taken to process one decryption = (time taken to process one decryption/ time for one clock cycle) = 198725/10 = **19872.5 clock cycles**

Q8: What is the corresponding latency of your design? What is the throughput of your design (in terms of number of decryptions per second)?

Ans: The latency of the design would be described as the time taken to load the value and do the process of decryption and output the decrypted message:
Time at which the first data c1, c2 and r2 is loaded = 180 ns
Time at which the first decrypted output is displayed = 198,905 ns
Latency = 198,905 - 180 ns = 198,725 ns

Throughput: Number of decryptions per second: Our design takes 198,725 ns to run 1 decryptions. Hence the number of decryption in a second is:
Throughput = 1 /Latency = 1/198725 ns = 5032 decryptions per second

4 Conclusions

Q9: Write a summary of the assignment. Among the things to consider are: what did you learn, what are the challenges of the assignment, what are the straightforward parts, and what could be improved?

Ans. Summary: The assignment focuses us on building a functionally correct cryptographic hardware design for post-quantum encryption scheme specifically using the the Ring - Binary Learning with Error set (R-BinLWEEnc-II) focusing on the hardware of the decryption block, which takes the two cipher-texts '*c1*' and '*c2*', the secret key '*r2*' provided in the testbench file and perform the decryption process to find the correct decrypted plain-text '*m*'. We have used Verilog as our hardware description language to model the decryption block, and have simulated, synthesized, place and routed the design on a Kintex-7 FPGA (XC7K355T : speed grade -3) board using Xilinx Vidado software and have optimized area-cost factor using the tools while maintaining all timing constraints.
Learnings:
We learned how the actual encryption and decryption process of the Ring Binary Learning with Error actually works and how it is not susceptible to quantum computer attacks since learning with error follows well known lattice problem namely Shortest-Vector-Path and this being a NP hard problem which would be very difficult for a quantum computer to break. We also learned how to design such decryption model and synthesize, place and route implementation on some common FPGA boards.
Challenges faced:
1. From a hardware design perspective had to figure out how the data and FSM machine would work after going through the reference paper.
2. The intermediate calculations with shifting and in-place reductions were challenging part of coding the design and without knowing the correct intermediate values made tedious manual calculations to verify time-consuming and tiresome.
3. There were software installation problems on local machine.
4. Figuring out synthesis issues for our design after successful simulation was a bit difficult requiring us to reach out to the TA.
Straightforward parts:
1. Implementing modulo-256 for ring operations and threshold function was easy.
2. Calculations regarding the report were straightforward once we had the synthesis and implementation values in front of us.
Further Improvements possible:
1. We could have improved our area utilization by exploring the big modules in our design and their look-up tables to explore where we can optimize area-cost by using mix of different gates. 2. Along with the area we could make our design a bit more faster by looking at optimizing our states and making our throughput higher.

Q10: Finally, please also report the time you spent on this assignment.
Ans: Total estimated time spent on this assignment was **26 hours**.