FPGA-Based Homomorphic Encryption Algorithm Deployment and Optimization

[1]Wasyihun Sema Admass,
[2]Appau Gideon Kofi Amo
[1,2]Department Software Engineering, University of Science and Technology of China,

Executive Summary

This project is to demonstrate a secure and efficient solution for privacy-preserving federated learning in healthcare by integrating additive semi-homomorphic encryption using the Paillier cryptosystem and FPGA-based hardware acceleration. Traditional machine learning methods risk data privacy by requiring centralized data storage. Federated learning mitigates this by keeping data decentralized while sharing only model updates. However, even model parameters may leak sensitive information during transmission. To address this, homomorphic encryption enables computations on encrypted data, protecting model updates during communication. Given the high computational demands of such cryptographic schemes, the project using FPGAs for acceleration. Using Vitis HLS, encryption and decryption cores are synthesized into RTL, optimized, and deployed on FPGA hardware via Vivado. The system supports secure aggregation of encrypted model updates, decryption, and redistribution, with validated results using healthcare datasets like breast cancer records. The final design is compiled into bitstream and hardware handoff files for FPGA programming, enabling a practical, secure federated learning environment with real-time potential.

1 INTRODUCTION

Federated learning has become a transformative paradigm in distributed machine learning in solving the growing concerns of data privacy and security in different application areas. In traditional centralized machine learning frameworks, raw data is often aggregated on a central server for model training, ha vulnerabilities such as data breaches and unauthorized access[1]. Federated learning can solve these problems by allowing multiple cleints/ praticipants to train their shared model without sharing their datasets, which means, only model parameters or updates are exchanged between clients and the central server. This decentralized approach significantly reduces privacy risks but introduces new challenges, particularly in ensuring secure interactions during joint modeling. Still the exchange of model parameters across untrusted network requires security attention to protect senstive information. To solve such problem homomorphic encryption

(HE) has gained prominence as a key solution to this challenge, allowing computations on encrypted data without requiring decryption[2]. In Federating learning based distributed machine learning frame work Homomorphic encryption specially additive semi-homomorphic encryption schemes like the Paillier cryptosystem can be integrated encrypt intermediate updates enabling secure parameter interaction and ensures that sensitive data remains confidential throughout the model training process for real time applications like, in healthcare, where patient data is highly regulated, HE can facilitate collaborative research across institutions without violating privacy laws. But, still computational overhead of Hommorphic encryption is major problem which needs to be optimized implemenation using hardware platforms.

Field-Programmable Gate Arrays (FPGAs) have emerged as a promising platform for deploying cryptographic algorithms, including homomorphic encryption, due to their inherent flexibility and energy efficiency[3]. FPGAs offer deterministic timing and lower latencies, making them ideal for real-time cryptographic tasks. FPGAs provide fine-grained control over hardware resources, enabling the implementation of custom logic tailored to specific cryptographic operations. This adaptability is crucial for optimizing homomorphic encryption, which involves complex mathematical transformations that benefit from dedicated circuits designed explicitly for these tasks. Field-Programmable Gate Arrays (FPGAs), known for their parallelism, reconfigurability, and energy efficiency, provide a compelling platform to accelerate homomorphic operations. This project aims to design, deploy, and optimize a hardware-accelerated homomorphic encryption module on FPGA, integrated into a federated learning pipeline tailored for privacy-preserving healthcare applications.

2 Objectives

2.1 General Objective

To design, implement, and optimize an FPGA-based privacy-preserving Federated Learning framework using Additive Semi-Homomorphic Encryption for secure and efficient healthcare data processing.

2.2 Specific Objectives

- Design and implement the Paillier-based additive semi-homomorphic encryption algorithm using FPGA
- Integrate the encryption module into a federated learning framework for distributed healthcare datasets
- Optimize the encryption algorithm for FPGA
- Evaluate performance metrics including latency, power, resource utilization (LUTs, FFs, BRAMs, DSPs), and encryption accuracy.

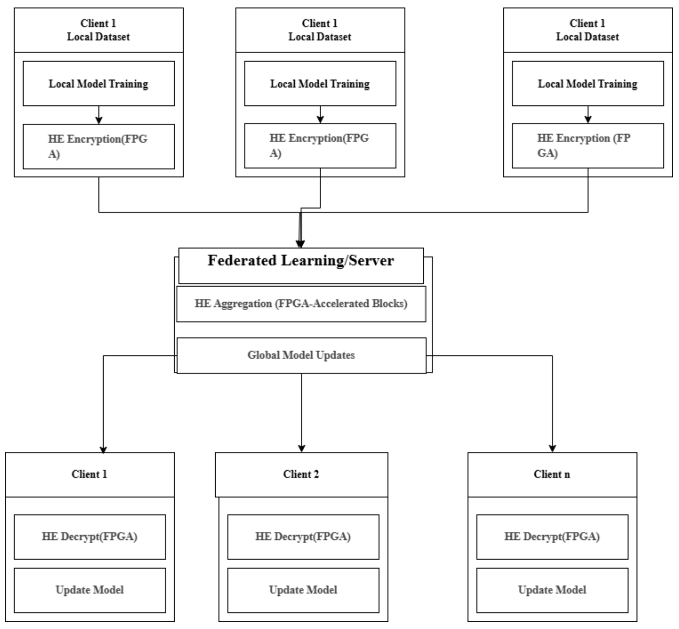


Figure 1: System Architecture for FPGA based Hemomorphic encryption

3 System Architecture

The system integrates a federated learning framework with an FPGA-accelerated additive hemomorphic encryption module. It consists of multiple client nodes performing local model training and a central aggregator node (server FPGA) responsible for securely aggregating encrypted updates.

As in the above diagram Clients train there model with their own private datasets to pass gradient descent updates model weights, and feed in to Hemomorphic encryption block and in hemomorphic block, Model weights are encrypted using Paillier HE (additive homomorphism) to preserve privacy. Then on FPGA Acceleration (Vitis Blocks) optimized Paillier's additive HE encryption/decryption performed.On Federated learning framework secure aggregation is performed, by receiving encrypted weights from all clients, Homomorphic aggregation by computing the weighted average of encrypted models with out decryption, then accelerate HE operation on FPGA Acceleration blocks.

Algorithm 1 Data Flow overview for Paillier HE and FPGA Acceleration

```
 1: Initialize:
 2: for each client (hospital $i = 1$ to $N$) do
 3:     Load local dataset $D_i$ ( CSV)
 4:     Initialize model $M_i$
 5: end for
 6: Generate Paillier keypair $(public\_key, private\_key)$
 7: Deploy encryption/decryption units on FPGA using Vitis HLS
 8: for each communication round $t = 1$ to $T$ do
 9:     for each client $i = 1$ to $N$ in parallel do
10:         Train local model $M_i$ on dataset $D_i$
11:         Obtain local model weights $W_i$
12:         if FPGA encryption is available then
13:             $[W_i] \leftarrow FPGA\_Encrypt(W_i, public\_key)$
14:         else
15:             $[W_i] \leftarrow Paillier\_Encrypt(W_i, public\_key)$
16:         end if
17:         Send $[W_i]$ to Federated Server over secure channel
18:     end for
19:     Initialize $[W_{\text{agg}}] \leftarrow [0]$
20:     for $i = 1$ to $N$ do
21:         $[W_{\text{agg}}] \leftarrow [W_{\text{agg}}] + [W_i]$
22:     end for
23:     Send $[W_{\text{agg}}]$ to FPGA for decryption
24:     $W_{\text{agg}} \leftarrow FPGA\_Decrypt([W_{\text{agg}}], private\_key)$
25:     Broadcast $W_{\text{agg}}$ to all clients
26:     for each client $i$ do
27:         Update local model $M_i$ with $W_{\text{agg}}$
28:     end for
29:     if Inference is required then
30:         Load $W_{\text{agg}}$ into FPGA CNN Inference Unit
31:         $Predict \leftarrow CNN\_Infer(new\_patient\_data)$
32:     end if
33: end for
```

3.1 FPGA Acceleration Blocks

In FPGA Acceleration block, paillier encryption/decryption and model inference is performed using Vitis High-Level Synthesis (HLS) for optimized hardware implementation. A hardware module designed using Xilinx Vitis High-Level Synthesis (HLS) for optimized performance.

As indicated in the above diagram on software layers (C++/C simulation), Hardware design layer (Vivado design suit) and Hardware Synthesis Layer (HLS - High-Level

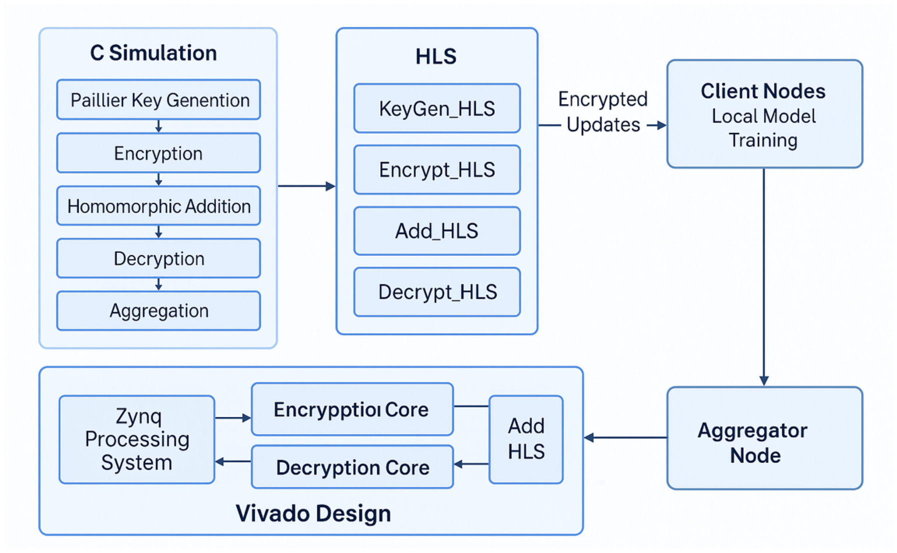
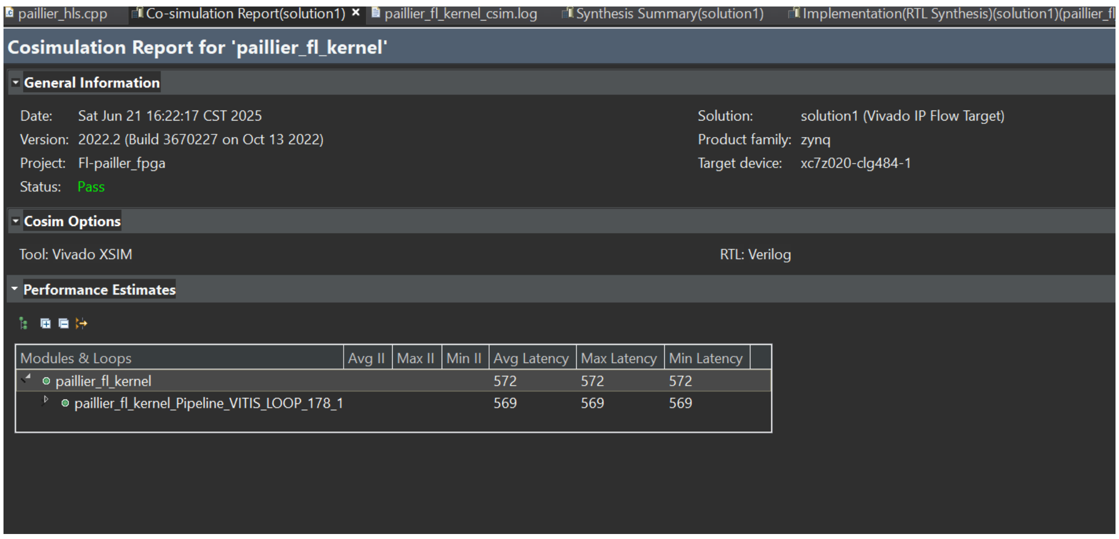


Figure 2: FPGA Acceleration Blocks

Synthesis Architecture)

3.1.1 Software layer (C++/C Simulation

In the C simulation validation of encryption, decryption, and aggregation logic before hardware deployment is performed. it simulates, Key Generation, encryption, hemomorphic addition, decryption, and agregation of the model are performed.


paillier_hls.cpp | Co-simulation Report(solution1) | paillier_fl_kernel_csim.log | Synthesis Summary(solution1) | Implementation(RTL Synthesis)(solution1)(paillier_fl

Cosimulation Report for 'paillier_fl_kernel'

General Information

Date:	Sat Jun 21 16:22:17 CST 2025	Solution:	solution1 (Vivado IP Flow Target)
Version:	2022.2 (Build 3670227 on Oct 13 2022)	Product family:	zynq
Project:	Fl-pailler_fpga	Target device:	xc7z020-clg484-1
Status:	Pass		

Cosim Options

Tool: Vivado XSIM   RTL: Verilog

Performance Estimates

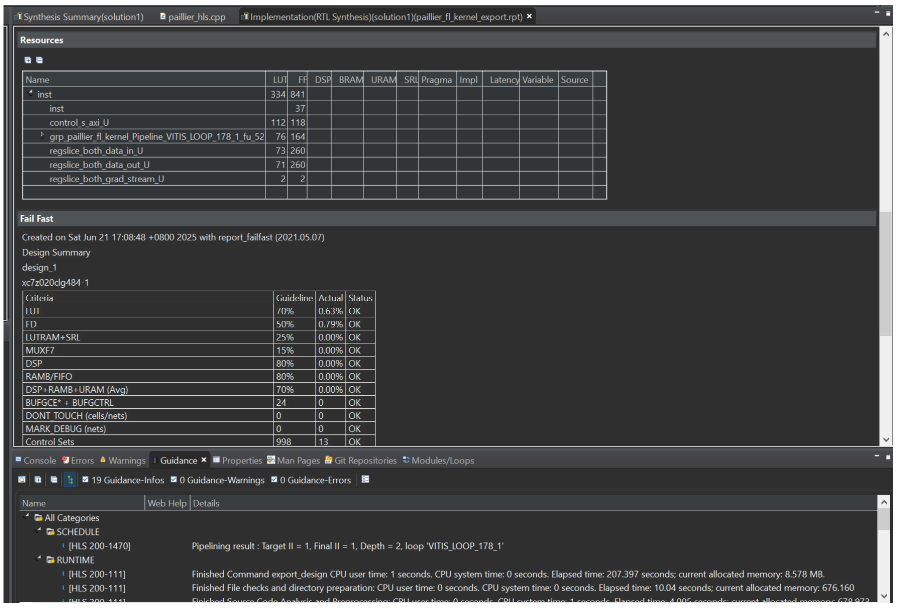
Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
paillier_fl_kernel				572	572	572
paillier_fl_kernel_Pipeline_VITIS_LOOP_178_1				569	569	569

Figure 3: C++/C simulation with latency reports

Finally simulation is verified and using vitis HLS to synthesize and export the design in to register transfer lever (RTL).

3.1.2 Hardware Synthesis Layer (HLS high level synthesis

The HLS is responsible for converting high level C/C++ simulation of the hemomorphic encryption algorithm in to register transfer level hardware using vitis HLS to insure that the encryption, decryption, agregation function are implemented efficientlly for FPGA, and allowing for performance tuning and resource optimization. In the Vitis HLS the four key modules are performed keyGen_HLS for generating public key and private key, Encrypt_HLS encrypt client updates using modular exponentiation, Add_HLS add encrypted value(homomorphic encryption), decrypt_HLS decrypt the agregated results, then loop pipeling, loop unrolling, bandwidth optimization , and resource directive techniques are applied for optimization of the model finally synthesizable RTL (verilog is generated)



Synthesis Summary(solution1)
paillier_hls.cpp
Implementation(RTL Synthesis)(solution1)(paillier_fl_kernel_export.rpt)

Resources

Name	LUT	FF	DSP	BRAM	URAM	SRL	Pragma	Impl	Latency	Variable	Source
inst	334	841									
inst		37									
control_s_axi_U	112	118									
grp_paillier_fl_kernel_Pipeline_VITIS_LOOP_178_1_fu_52	76	164									
regslice_both_data_in_U	73	260									
regslice_both_data_out_U	71	260									
regslice_both_grad_stream_U	2	2									

Fail Fast

Created on Sat Jun 21 17:08:48 +0800 2025 with report_failfast (2021.05.07)

Design Summary

design_1

xc7z020clg484-1

Criteria	Guideline	Actual	Status
LUT	70%	0.63%	OK
FD	50%	0.79%	OK
LUTRAM+SRL	25%	0.00%	OK
MUXF7	15%	0.00%	OK
DSP	80%	0.00%	OK
RAMB/FIFO	80%	0.00%	OK
DSP+RAMB+URAM (Avg)	70%	0.00%	OK
BUFGCE* + BUFGCTRL	24	0	OK
DONT_TOUCH (cells/nets)	0	0	OK
MARK_DEBUG (nets)	0	0	OK
Control Sets	998	13	OK

19 Guidance-Infos 0 Guidance-Warnings 0 Guidance-Errors

```
Name                     Web Help  Details
All Categories
  SCHEDULE
    [HLS 200-1470]                 Pipelining result : Target II = 1, Final II = 1, Depth = 2, loop 'VITIS_LOOP_178_1'
  RUNTIME
    [HLS 200-111]                  Finished Command export_design CPU user time: 1 seconds. CPU system time: 0 seconds. Elapsed time: 207.397 seconds; current allocated memory: 8.578 MB.
    [HLS 200-111]                  Finished File checks and directory preparation: CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 10.04 seconds; current allocated memory: 676.160
    [HLS 200-111]                  Finished Source Code Analysis and Preprocessing: CPU user time: 0 seconds. CPU system time: 1 seconds. Elapsed time: 4.005 seconds; current allocated memory: 678.973
```

Figure 4: Synthesized HLS report of RTLs

3.1.3 Hardware design layer (Vivado design suit)

This level focuses on designing a complate hardware pipeline that can support secure federated learning operations on FPGA fabric by integrating optimized encryption/decryption HLS modules i to FPGA in to vivado.

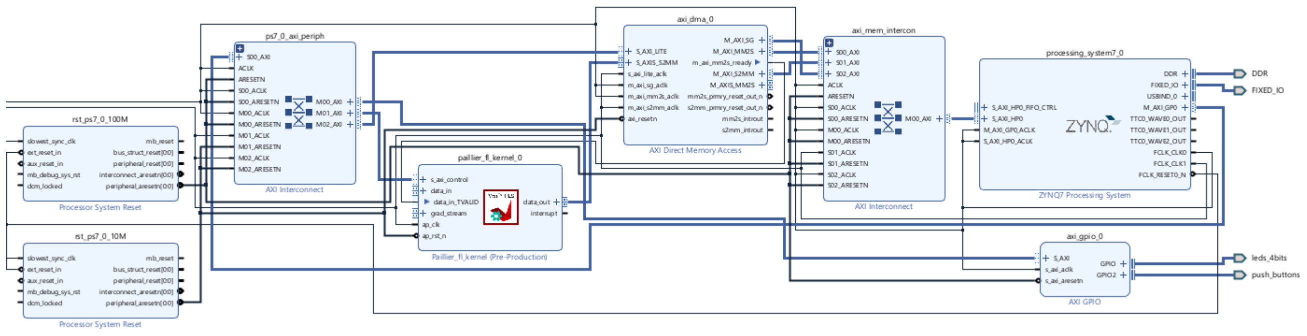


Figure 5: Hardware Level Design for FPGA based Federated learning HE

In the above diagram the following component are included

- zynq Processing system(PS) which contains AM cortex dual core processor, interfaces with Memory,coordinates and control with HLS IP cores genrated as RTL, Runs the host-side application to trigger encryption/decryption/aggregation functions

- HLS IP Cores (PL - Programmable Logic):

Imported from Vitis HLS: KeyGen_HLS Paillier key generation, Encrypt_HLS Paillier encryption engine, Decrypt_HLS Paillier decryption engine, Add_HLS Homomorphic addition unit.

- AXI Interconnect

The AXI interconnect is used for connecting HLS IP core with ZYNQ processor system with AXI-light and AXI-stream.

- Memory/Storage: DDR for large input/output data and supports DMA for transfering data to/from FPGA, UART/JTAG for debuging, GPIO for status control signals.

After complating the hardware design using vivado, bitstream file is generated to configure the FPGA with the custom encryption/degreption, agregation logic, then the bitstream is used for runtime execution.

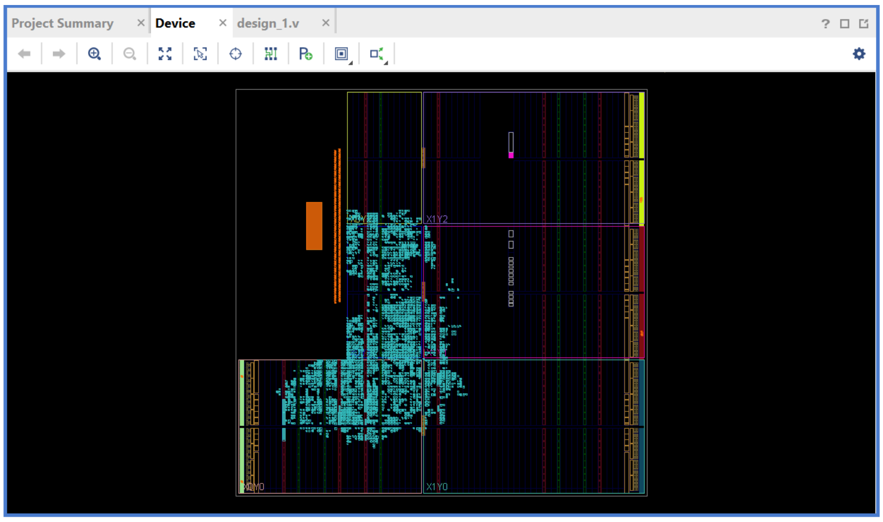
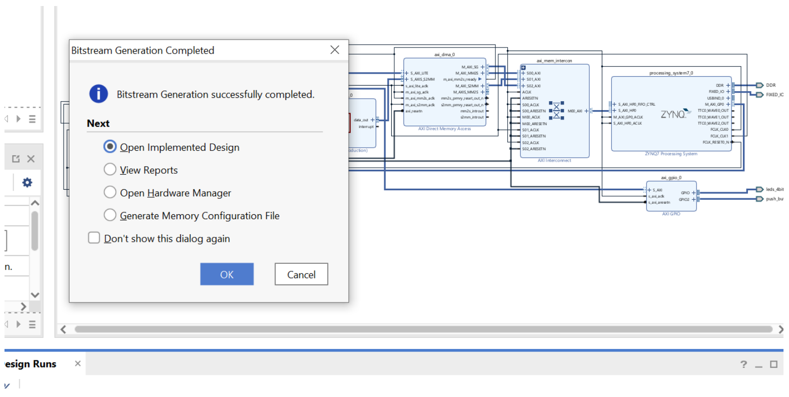


Figure 6: Designed hardware board generated from Vivado


Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

[✓ Open Implemented Design]

View Reports

Open Hardware Manager

Generate Memory Configuration File

Don't show this dialog again

Figure 7: Bit stream Generation from The designed hardware

After complating the design on vivado and generating bit streams, then the hardware is synthesised and converted to a gate level representation, then the implementation maps and routes the design in to FPGA fabric, the the hardware is exported asâxsa file which helps to program FPGA board.

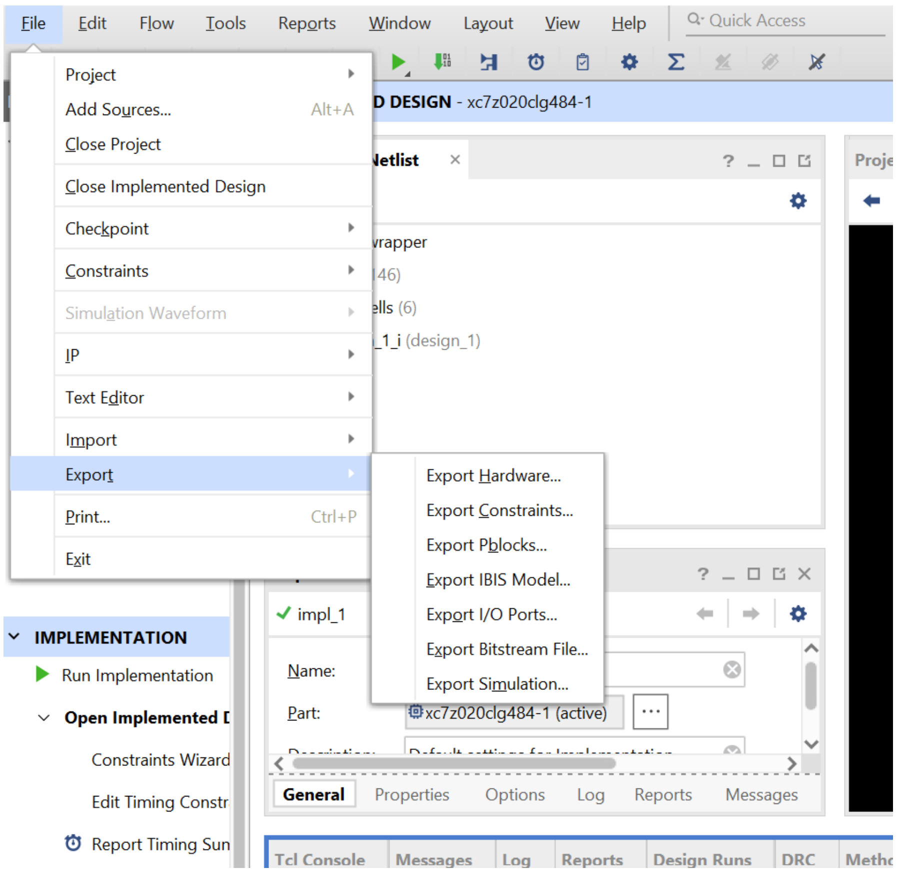
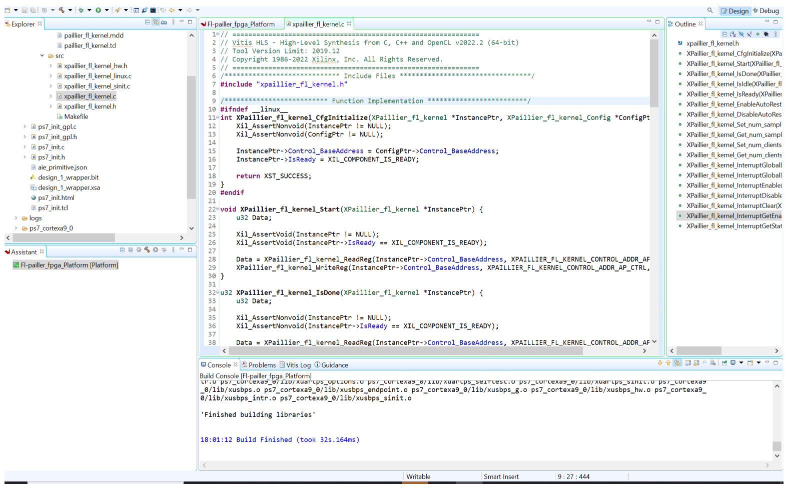


Figure 8: Exporting the hardware to program the FPGA board

Finally, after generating the hardware from vivado, hardware handoff file (.xsa) file is generated for software development in Vitis IDE which allow to program FPGA board and develop host applications that communicate with the custom hard ware blocks.



```c
// ==============================================================
// Vitis HLS - High-Level Synthesis from C, C++ and OpenCL v2022.2 (64-bit)
// Tool Version Limit: 2019.12
// Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
// ==============================================================
/***************************** Include Files *********************************/
#include "xpaillier_fl_kernel.h"

/************************** Function Implementation *************************/
#ifndef __linux__
int XPaillier_fl_kernel_CfgInitialize(XPaillier_fl_kernel *InstancePtr, XPaillier_fl_kernel_Config *ConfigPt
    Xil_AssertNonvoid(InstancePtr != NULL);
    Xil_AssertNonvoid(ConfigPtr != NULL);

    InstancePtr->Control_BaseAddress = ConfigPtr->Control_BaseAddress;
    InstancePtr->IsReady = XIL_COMPONENT_IS_READY;

    return XST_SUCCESS;
}
#endif

void XPaillier_fl_kernel_Start(XPaillier_fl_kernel *InstancePtr) {
    u32 Data;

    Xil_AssertVoid(InstancePtr != NULL);
    Xil_AssertVoid(InstancePtr->IsReady == XIL_COMPONENT_IS_READY);

    Data = XPaillier_fl_kernel_ReadReg(InstancePtr->Control_BaseAddress, XPAILLIER_FL_KERNEL_CONTROL_ADDR_AF
    XPaillier_fl_kernel_WriteReg(InstancePtr->Control_BaseAddress, XPAILLIER_FL_KERNEL_CONTROL_ADDR_AP_CTRL,
}

u32 XPaillier_fl_kernel_IsDone(XPaillier_fl_kernel *InstancePtr) {
    u32 Data;

    Xil_AssertNonvoid(InstancePtr != NULL);
    Xil_AssertNonvoid(InstancePtr->IsReady == XIL_COMPONENT_IS_READY);

    Data = XPaillier_fl_kernel_ReadReg(InstancePtr->Control_BaseAddress, XPAILLIER_FL_KERNEL_CONTROL_ADDR_AF
```

Build Console [Fl-pailler_fpga_Platform]

```
_0/lib/xusbps.o ps7_cortexa9_0/lib/xusbps_endpoint.o ps7_cortexa9_0/lib/xusbps_g.o ps7_cortexa9_0/lib/xusbps_hw.o ps7_cortexa9_
0/lib/xusbps_intr.o ps7_cortexa9_0/lib/xusbps_sinit.o

'Finished building libraries'


18:01:12 Build Finished (took 32s.164ms)
```

Figure 9: HLS Host code to program FPGA board on Vitis IDE

3.2 Tools and Platforms

Hardware and software tools are used to develop FPGA-based federated learning system with additive semi-homomorphic encryption for healthcare , and in the table bellow shows the general tools and platforms used in the project.

Layer	**Tool / Platform**	**Purpose**
Simulation	C/C++, Python	Validate Paillier logic and FL flow
Synthesis	Vitis HLS	Convert C to RTL, optimize encryption cores
FPGA Integration	Vivado Design Suite	Integrate and connect IPs, implement design
Testing	PYNQ / JTAG / UART	Board-level debugging and verification
Hardware	Zynq-7000 (ZedBoard) / ZCU104	SoC platform with ARM + FPGA
Dataset	Breast Cancer Dataset (CSV)	Healthcare data for FL training

Table 1: Tools and Platforms Used Across System Layers

3.3 Conclusion

In this project we try to demonstrate the implementation of privacy preserving federated learning system using FPGA accelerated additive semi-hemomorphic encryption by integrating paillier encryption scheme in to federating learning framework and optimizing it using high-level synthesis (HLS) and vivado design tools, to develop secure model updates without compromising performance. The use of FPGA improves the efficiency of encryption, decryption, and aggregation operation, overcoming the computational problem associated with homomorphic encryption. The layered architecture from software simulation to hardware synthesis and deployment ensures that the system is both functionally accurate and resource-efficient.

Applied to a healthcare setting using datasets like breast cancer records, the system enables secure, decentralized model training while protecting sensitive patient data. This work not only highlights the feasibility of deploying cryptographic machine learning workflows on FPGA but also teaches us to contributes toward building real-time, privacy-focused AI solutions in the medical domain.

REFERENCES

[1] N. M. Hijazi, M. Aloqaily, M. Guizani, B. Ouni, and F. Karray, "Secure federated learning with fully homomorphic encryption for iot communications," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 4289–4300, 2024. DOI: 10.1109/JIOT.2023.3302065.

[2] Z. Zuo, N. Su, B. Li, and T. Zhang, "Pack: Towards communication-efficient homomorphic encryption in federated learning," in *Proceedings of the 2024 ACM Symposium on Cloud Computing*, ser. SoCC '24, Redmond, WA, USA: Association for Computing Machinery, 2024, pp. 470–486, ISBN: 9798400712869. DOI: 10.1145/3698038.3698557. [Online]. Available: https://doi.org/10.1145/3698038.3698557.

[3] S. Sinha Roy, F. Turan, K. Jarvinen, F. Vercauteren, and I. Verbauwhede, "Fpga-based high-performance parallel architecture for homomorphic computing on encrypted data," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 387–398. DOI: 10.1109/HPCA.2019.00052.