2024年度プロジェクト学習成果発表会

Project No.4 暗号と数理とセキュリティ

Cryptography, Mathematics and Security

メンバー：後藤瞭介 Ryousuke Gotou　ディオガーディディラン基暉 Dylan Motoki Dioguardi　伊澤叡遵 Eijun Izawa　松浦実里 Minori Matsuura

担当教員：白勢政明 Masaaki Shirase　川越敏司 Toshiji Kawagoe　川口聡 Satoshi Kawaguchi　斎藤朝輝 Asaki Saito

■プロジェクト概要

本プロジェクトは、デジタル社会における情報セキュリティの課題を解決し、社会貢献を目指すことを目的とする。暗号技術の基礎となる数理を学び、必要な知識を習得した後、楕円曲線暗号をハードウェア、特にFPGAに実装し、その効率化に取り組んだ。楕円曲線暗号は、次世代型の自動運転システム、次期マイナンバーカードなどにも使用される予定であり、安全で効率的な社会基盤に寄与する為の応用が期待されている。

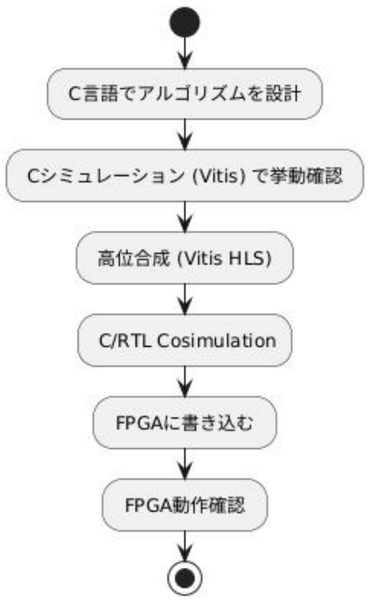
Project Overview

The purpose of this project is to contribute to society by solving information security issues in the digital society. After learning the mathematics underlying cryptography and acquiring the necessary knowledge, we implemented elliptic curve cryptography in hardware, especially FPGA, and worked on improving its efficiency. Elliptic curve cryptography is expected to be used in the next generation of automatic driving systems, the next generation of my number card, etc., and is expected to be applied to contribute to a secure and efficient social infrastructure.

■活動内容 Activity Details

前期では、暗号技術の基礎である数理の学習を行い、基盤となる知識を身に着けていった。後期は、それらを元に楕円曲線暗号のハードウェア実装へ向けて取り組んだ。ハードウェア実装のメリットは、低遅延かつ高効率で処理が可能で、CPUの負荷を減らす点にある。従来ハードウェア設計を行う場合は、直接ハードウェア記述言語を用いて開発されているが、今回は短期間での開発である為、高位合成という技術を用いて開発を行った。高位合成は、AMD社が提供するVitisというFPGAの設計と開発を行うためのソフトウェアを使用した。Vitis上で、C言語でアルゴリズムを設計、アルゴリズムのシミュレーション、高位合成、ハードウェア記述言語で記述された回路のシミュレーションを行った。FPGA本体での動作確認まではたどり着かなかったもの、楕円曲線暗号の主要な計算のシミュレーションは正確に行うことができた。

In the first semester, the students learned the mathematics underlying cryptography and acquired knowledge of the fundamentals. In the second semester, they worked toward hardware implementation of elliptic curve cryptography. The advantage of hardware implementation is that it enables low latency and high efficiency processing and reduces the CPU load. In the past, hardware design was developed directly using a hardware description language, but this time, since the development was to be done in a short period of time, a technique called high-level synthesis was used. For high-level synthesis, we used AMD's Vitis software for FPGA design and development. Although we did not reach the point of verifying the operation on the main unit, we were able to accurately simulate the main calculations of elliptic curve cryptography.



○ 暗号の高速化(アルゴリズム) increasing the speed

楕円曲線暗号の主要計算として、まずはシンプルなアルゴリズムを用いて作成を行った。しかし、使用する数が大きくなるにつれてCシミュレーションを行うのが困難になり高速化を行った。楕円曲線暗号における、暗号化の支配的な処理は２回のスカラー倍算、復号の支配的な処理は１回のスカラー倍算である。したがってスカラー倍の高速化は楕円曲線暗号そのものを高速化したといえる。

Scalar multiplication was initially implemented using a simple algorithm, but as the size of the numbers increased, simulation in C became challenging, necessitating optimization for speed. In elliptic curve cryptography, the dominant operation during encryption involves two scalar multiplications, while decryption requires one. Therefore, optimizing scalar multiplication effectively accelerates the entire elliptic curve cryptography process.

○ 高位合成 high-level synthesis

ハードウェア設計の手法の一つで、アルゴリズムや高水準言語（例：C言語）で記述されたコードを元に、ハードウェア記述言語（HDL）の回路設計データに自動変換すること。

A method of hardware design that automatically converts circuit design data in Hardware Description Language (HDL) based on algorithms or code written in a high-level language (e.g., C).

○ FPGA

Field Programmable Gate Arrayの略称であり、実際に使用する現場（フィールド）で、デバイスの外部から設計情報を書き込むことが可能な論理回路（ゲート）が集積されたデバイス。

Abbreviation for Field Programmable Gate Array, a device that integrates logic circuits (gates) that allow design information to be written from outside the device in the field where it is actually used.

■現状の成果 Activity Details

Vitisを使用したCシミュレーションの結果をもとに、次は実際のFPGAハードウェア実装を進めた。シミュレーションで得られた処理時間と、実際のハードウェアでの実行時間が一致するかどうかを検証し、ハードウェア実装における効率性を確認する。この比較検証を通じて、性能向上に取り組んだ。

Based on the results of the C simulation using Vitis, the next step was to proceed with the actual FPGA hardware implementation. We verified whether the processing time obtained from the simulation matched the execution time in the actual hardware to confirm the efficiency in the hardware implementation. Through this comparative verification, we worked to improve performance.

Before

$O(2^{\lambda})$

1.15×10^{77}回の処理

※ λ=128 の場合

After

$O(\lambda)$

512回の処理

※ λ=128 の場合

```
-- ==============================================================
-- Generated by Vitis HLS v2024.1
-- Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
-- Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.
-- ==============================================================

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity point_mult is
port (
    ap_clk : IN STD_LOGIC;
    ap_rst : IN STD_LOGIC;
    ap_start : IN STD_LOGIC;
    ap_done : OUT STD_LOGIC;
    ap_idle : OUT STD_LOGIC;
    ap_ready : OUT STD_LOGIC;
    P : IN STD_LOGIC_VECTOR (2047 downto 0);
    n : IN STD_LOGIC_VECTOR (1023 downto 0);
    ap_return : OUT STD_LOGIC_VECTOR (2047 downto 0) );
end;


architecture behav of point_mult is
    attribute CORE_GENERATION_INFO : STRING;
    attribute CORE_GENERATION_INFO of behav : architecture is
    "point_mult_point_mult,hls_ip_2024_1,
{HLS_INPUT_TYPE=cxx,HLS_INPUT_FLOAT=0,HLS_INPUT_FIXED=0,HLS_INPUT_PART=xcvu
9p-flga2104-2-
i,HLS_INPUT_CLOCK=10.000000,HLS_INPUT_ARCH=others,HLS_SYN_CLOCK=7.284000,HL
S_SYN_LAT=-1,HLS_SYN_TPT=none,HLS_SYN_MEM=0,HLS_SYN_DSP=0,HLS_SYN_FF=731
26,HLS_SYN_LUT=71220,HLS_VERSION=2024_1}";
    constant ap_const_logic_1 : STD_LOGIC := '1';
    constant ap_const_logic_0 : STD_LOGIC := '0';
    constant ap_ST_fsm_state1 : STD_LOGIC_VECTOR (7 downto 0) := "00000001";
    constant ap_ST_fsm_state2 : STD_LOGIC_VECTOR (7 downto 0) := "00000010";
    constant ap_ST_fsm_state3 : STD_LOGIC_VECTOR (7 downto 0) := "00000100";
    constant ap_ST_fsm_state4 : STD_LOGIC_VECTOR (7 downto 0) := "00001000";
    constant ap_ST_fsm_state5 : STD_LOGIC_VECTOR (7 downto 0) := "00010000";
    constant ap_ST_fsm_state6 : STD_LOGIC_VECTOR (7 downto 0) := "00100000";
    constant ap_ST_fsm_state7 : STD_LOGIC_VECTOR (7 downto 0) := "01000000";
```

■今後の課題 Future Tasks

Vitisを使用したCシミュレーションの結果をもとに、今後は実際のFPGAハードウェア実装を進めていくことごが必要である。シミュレーションで得られた処理時間と、実際のハードウェアでの実行時間が一致するかどうか検証し、ハードウェア実装における効率性の確認を行う。また、この比較検証を通じて性能向上を目指す。加えて、Vitisのヒートマップ機能を活用して、コードのボトルネックを特定し、最適なアルゴリズムを適用することで、さらなるパフォーマンス向上を図る。これらの成果を基に、将来の応用に向けた基盤を築くことを目指す。

Based on the results of the C simulation using Vitis, it is necessary to proceed with the actual FPGA hardware implementation in the future. We will verify whether the processing time obtained from the simulation matches the actual hardware execution time, and confirm the efficiency of the hardware implementation. We also aim to improve performance through this comparative verification. In addition, Vitis' heat-map function will be used to identify code bottlenecks and apply optimal algorithms to further improve performance. Based on these results, we aim to lay the foundation for future applications.