公立はこだて未来大学 2018 年度 システム情報科学実習
グループ報告書

Future University-Hakodate 2018 System Information Science Practice
Group Report

プロジェクト名

FUN-ECM プロジェクト

Project Name

FUN-ECM Project

グループ名

A グループ

Group Name

A Group

プロジェクト番号/Project No.

18

プロジェクトリーダ/Project Leader

1016228　青山和弘　Kazuhiro Aoyama

グループリーダ/Group Leader

1016228　青山和弘　Kazuhiro Aoyama

グループメンバ/Group Member

1016002　池田晴輝　Haruki Ikeda
1016010　小泉建敬　Takehiro Koizumi
1016019　矢和田航平　Kohei Yawata
1016133　上田隼人　Hayato Ueda
1016139　染川眞輝　Masaki Somekawa
1016228　青山和弘　Kazuhiro Aoyama
1016229　鳴海雄登　Yuto Narumi

指導教員

白勢政明　由良文孝

Advisor

Masaaki Shirase　Fumitaka Yura

提出日

2019 年 1 月 15 日

Date of Submission

1 15, 2019

概要

私達のプロジェクトの目的は，楕円曲線法 (ECM) を用いてより大きな桁数の素因数を発見することである．その背景には，約 40 年前に考案されて以来現在もデジタル署名などに利用されている RSA 暗号の安全性の評価がある．RSA 暗号が長年利用されているのは，大きい桁数の 2 つの素数からなる合成数を素因数分解することが難しいことにより安全性が保障されているためである．しかし，近年ではより高い安全性を持つ楕円曲線暗号が利用されてきている．そこで，前年度までに作成・改良された，楕円曲線法によってより大きい桁数の素因数分解を行うプログラムを運用し，大きい数の素因数分解をランキングしたサイトである ECMNET[1] や STUDIO KAMADA[2] へのランクインを目標としている．加えて，今年度は以下の 2 点を達成するために 2 つの班に分かれ活動を行った．

- 素因数分解を行うプログラムの更なる高速化
- FUN-ECM プロジェクトの活動広報 Web サイトを一新し，かつ高校生にも ECM を理解してもらえるような学習コンテンツを盛り込むこと

以降，前者を担当する班を高速化班，後者を担当する班を広報班と呼称する．
高速化班はプログラムを高速化する方法を検討し，前年度までに作成されたプログラムで使用されていた C 言語よりも高速な言語 FORTRAN で，同様の動作をするプログラムを作成することで高速化できるのではないかと考えた．高速化班には FORTRAN を使用したことのあるメンバがいたので，そのメンバを中心に FORTRAN の基礎学習を行った．また，FORTRAN でプログラムを作成することで高速化ができることを裏付けるために，FORTRAN・C 言語・Python の 3 つで同じ演算を行うプログラムを作成し，その実行速度を比較した．その結果，FORTRAN で作成したプログラムが最も早く動作すると分かった．そのため後期は素因数分解を行う FORTRAN 版プログラムの完成を目標に活動する予定であったが，学習コストや開発期間などの理由から断念した．そのため，高速化班から更に「システム班」と「ハードウェア班」の 2 つの班に分かれることとなった．
広報班はまず Web サイトを一新するために，Adobe Illustrator を用いて既存の Web サイトを構造化し，一目で確認できるようにした．これに基づきサイトの弱点をあぶり出し，新たな Web サイトのレイアウトを考案した．また，前期は他に HTML や Adobe Illustrator の学習，従来のロゴをベースにした新しいロゴの作成，サイトの大まかなカテゴリ分け，及び各ページのプレビュー作成を行った．後期は各メンバが HTML や Web デザインの知識を身につけ，実際にサイトの構築，サーバへのアップロードと各カテゴリの統合を行った．
システム班は，前年度までに作成された素因数分解プログラムの利用が不便であることを問題とし，利便性の向上を図るため funecm システムと呼称するシステムの構築を行った．結果として，プログラムの実行，及びその結果の確認を容易に行えるようなシステムを構築した．
ハードウェア班は，本プロジェクト本来の目的である「素因数分解を高速に行うこと」を達成するため，楕円曲線法を FPGA 上に実装することを目標に活動を行った．楕円曲線法を行う上で最も時間がかかる点 P のスカラー倍の操作について FPGA に実装した．

キーワード　素因数分解, 楕円曲線法, ECMNET, RSA 暗号, FORTRAN, FMLIB, FPGA

（※文責: 青山和弘）

Abstract

The purpose of our project is to find big numbers of prime factors using elliptic curve method (ECM). In the background, there is the RSA cryptography which has been used for digital signatures and the like since it was devised about 40 years ago. The reason why RSA encryption has been used for many years is that security is guaranteed because it is difficult to prime factorize a composite number composed of two prime numbers with a large number of digits. However, in recent years elliptic curve cryptography with higher security has been used. Therefore, by using a program that performs factoring decomposition of a larger number of digits by the elliptic curve method created and improved by the previous fiscal year, rank in to ECMNET[1] or STUDIO KAMADA[2] which is a site ranking a large number of prime factorization It is a target. In addition, this fiscal year, we divided into two groups and carried out activities to achieve the following two points.

- Further speed-up of program for prime factorization
- Activities of the FUN-ECM project Public information Including learning content that redesigns the website and allows high school students to understand ECM

Hereafter, the group responsible for the former is called a high-speed group and the group responsible for the latter is called a public relations group.

The high-speed group examines a method to speed up the program, and by creating a program that behaves similarly in a language FORTRAN, which is faster than the C language used in the program created up to the previous year, speedup we thought that we could do it. Since there was a member who had used FORTRAN in the speeding group, we conducted basic learning of FORTRAN mainly on its members. Moreover, in order to support that speed can be improved by creating a program in FORTRAN, we created a program that performs the same operation in FORTRAN, C language, and Python and compares its execution speed. As a result, I found that programs written in FORTRAN work most quickly. For this reason, we planned to work on completion of the FORTRAN version program which performs prime factorization in the latter term but abandoned for reasons such as learning cost and development period. Therefore, from the speeding up group, it was divided into two groups of "system group" and "hardware group".

In order to redesign the website first, the public relations group structured an existing Web site using Adobe Illustrator and made it possible to confirm at a glance. Based on this, he hid a weakness of the site and devised a layout of a new Web site. In addition, in the previous term, I did other HTML and Adobe Illustrator learning, create a new logo based on the traditional logo, roughly categorize the site, and make a preview of each page. In the latter term, each member acquired knowledge of HTML and Web design, actually constructed the site, uploaded to the server and integrated each category.

The system group created a system called a funecm system in order to improve convenience, with the problem that incomplete use of prime factorization program created up to the previous year is inconvenient. As a result, we constructed a system that makes it easy to execute the program and confirm the result.

In order to attain 'original prime factorization at high speed' which is the original purpose of this project, the hardware group worked with the goal of implementing the elliptic curve method on the FPGA. A scalar multiplication operation of the point P, which takes the longest time to perform the elliptic curve method, was implemented in an FPGA.

Keyword Elliptic Curve Method, prime factorization, ECMNET, RSA cryptosystem, FORTRAN, FMLIB

（※文責: 青山和弘）

目次

第 1 章　背景

1.1　プロジェクトの背景

暗号化技術は，情報の保護やコンピュータセキュリティにおいて欠かせない技術である．ファイルの暗号化の他に，HTTPS や，無線 LAN における通信など多くの場面で暗号化技術が利用されている．しかし，暗号化技術は常に進化する攻撃方法により解読の脅威に晒されている．様々な攻撃方法から安全な暗号アルゴリズムを作成するためには，作成する側が暗号解読の方法を知る必要がある．暗号の安全性評価には暗号解読の技術が利用されていて，暗号の強度は暗号解読に必要な情報量と計算量によって評価される．今回のプロジェクトでは，その暗号解読アルゴリズムの 1 つである，楕円曲線法 (ECM) を学ぶ．

現在，有名な公開鍵暗号の 1 つに RSA 暗号がある．RSA 暗号は，桁数が大きい合成数の素因数分解が困難であることを安全性の根拠とした暗号である．RSA 暗号を解読する時は合成数の元となる 2 つの素因数を見つけ出す必要がある．ECM では，与えられた曲線の点が無限遠点になることによって，因数が発見される．この性質を利用して RSA 暗号を解読する．

昨年度までに C 言語と多倍長整数などの任意の精度の算術ライブラリ GNU Multi-Precision Library (GMP) を用いた素因数分解プログラムが完成した．そこで今年度はより高速な計算処理を行えるとされる科学技術計算向きプログラミング言語 FORTRAN を用いて素因数分解プログラムを作成し，昨年度までに作成された C 言語で記述されたプログラムと速度比較を行うことでより高速な素因数分解を行うことを目指している．

（※文責: 鳴海雄登）

1.2　ECMNET，STUDIO KAMADA とは

ECMNET[1] とは，ECM を用いて発見した素因数の大きさをランキング形式で競う Web サイトである．ECMNET にランクインするためには，現在登録されている素因数よりも大きな素因数を見つける必要がある．このサイトは ECM を用いて特定の範囲でカニンガム数を素因数分解し，Cunningham project に貢献することを目標として掲げている．
以下にカニンガム数，Cunningham project に関しての詳細を記す．

カニンガム数

以下の条件を満たす数をカニンガム数という．なお ECMNET では変数 b,n の範囲を限定している．

$$b^n \pm 1 \ s.t.\ b,n \in N,\ \ b \text{ は累乗数でない} \tag{1.1}$$

Cunningham project

Cunningham project とは， b,n を表 1.1 の範囲に限定したカニンガム数を素因数分解するプロジェクトである．

表 1.1　Cunningham project における b, n の範囲

b	2	3	5	6	7	10	11	12
n の上限	1300	850	550	500	450	400	350	350

2018 年 12 月現在，ECMNET にランクインするためには，全期間の上位 50 位までで 68 桁以上の素因数を発見する必要がある．また，2018 年内のランキングにランクインすることを目的とした場合においても，61 桁以上の素因数を発見する必要がある．

STUDIO KAMADA[2] とは，鎌田誠氏により開設されたサイトである．こちらのサイトでは素因数が見つかっていない桁の大きい合成数が公開されており，これを素因数分解し，分解した素因数の大きさを競うサイトである．合成数の素因数分解を行う方法も多様であり，その中の 1 つとして楕円曲線法がある．以下に STUDIO KAMADA にて対象とされている合成数の説明をする．

レピュニット
: repeated unit (反復単位数) の略称であり，1,111,1111... のような 1 だけからなる自然数のこと．

レプディジット
: repeated digit の略称である．レピュニットを含むすべての桁の数字が同じ自然数から構成される自然数のこと．

ニアレプディジット
: near-repeated digit の略称であり，レプディジットの数字を 1 桁だけ他の数字に置き換えた自然数のこと．

クワージレプディジット
: レプディジットの数字を 2 個他の数字に置き換えた自然数のこと．

プラトウアンドデプレッション
: レプディジットの数字の両端を共通の数字に置き換えた自然数のこと．

ニアレプディジット回文数
: ニアレプディジットのうち，桁数が奇数であり，中央の数字だけが異なる数字のこと．

STUDIO KAMADA では，上記のようなニアレプディジット関連の数の完全な素因数分解を目的とされている. 有志で素因数分解を行った後，素因数が見つかった際にその数ともう一方の要素をサイトに投稿することで，段階的に上記の数の素因数分解を行っている． 2018 年 12 月現在，STUDIO KAMADA にて公開されている ECM で見つかった素因数のランキングにランクインするには 53 桁以上の素因数を発見する必要がある．

（※文責: 染川眞輝）

1.3　ECM の類似技術

本プロジェクトでは素因数分解を行うために楕円曲線を用いたが，楕円曲線はセキュリティのための技術 (楕円曲線暗号系) にも用いられている．本プロジェクトに関連する類似技術としてこれらを紹介する．

楕円曲線暗号系とは，以下の楕円曲線を用いた技術の総称である．歴史的には，Miller と Kobliz によって独立に楕円曲線暗号系が最初に開発された．

1. ECElGamal のような公開鍵暗号:
   暗号化に用いる公開鍵を公開できるため，不特定多数との秘密通信に適している．
2. ECDSA のようなディジタル署名
   ユーザ認証や改ざん検出のために用いられる．近年は SSL/TLS 通信において一般的に用いられている．また，欧米では V2X 通信において ECDSA の使用が決定した．
3. ECDHE のような鍵共有法
   送受信者間の共通鍵暗号の鍵の共有のために使用される．近年は SSL/TLS 通信において一般的に用いられている．

楕円曲線暗号系は RSA 暗号系と比較して，同じ安全性で鍵長が短くてよく，処理が高速であるという長所を有している．更に，ECDHE は前方秘匿性 (秘密情報が漏えいしても被害を被るのは 1 回の通信のみであるという性質) を有している．RSA 暗号系では前方秘匿性を実現できない．

楕円曲線を用いることで，ID ベース暗号や ID ベース署名，属性ベース暗号，検索可能暗号，タイムリリース暗号，グループ署名，代理人再暗号といった高機能暗号と呼ばれる新しい暗号技術を実現できる．これらは一部を除いて研究段階であるが，ブロックチェーンに必要な電子署名に ID ベース署名が用いられている実用例がある．

1. ID ベース暗号:
   メールアドレスや学籍番号といったユーザの ID を公開鍵とする公開鍵暗号.
2. ID ベース署名:
   メールアドレスや学籍番号といったユーザの ID を公開鍵とするディジタル署名.
3. 属性ベース暗号:
   学科，コース，学年，氏名，といったユーザの属性の論理式を公開鍵とする公開鍵暗号.
4. 検索可能暗号:
   暗号化されたデータから，復号せずに検索できる暗号方式.
5. タイムリリース暗号:
   復号日時を指定できる公開鍵暗号.
6. グループ署名:
   グループに属する人が行えるディジタル署名．署名者がグループの誰かであるかは秘匿される．
7. 代理人再暗号:
   ユーザ A によって暗号化されたデータを，復号せずに直接別のユーザ B が復号できるような暗号文に変換できる公開鍵暗号.

なお，ECM とここで紹介したセキュリティ技術では目的は全く異なるが，支配的な処理は同じある．

表 1.2　支配的な処理と剰余に用いる数

	ECM	ECDSA	ECDHE	高機能暗号
支配的な処理	スカラー倍算	スカラー倍算	スカラー倍算	スカラー倍算とペアリング写像
剰余に用いる数	合成数	素数	素数	素数

なお，ペアリング写像とは楕円曲線上に定義される双線形性写像であり，Weil ペアリング，Tate ペアリング，Ate ペアリング等が有名である．ペアリング写像の効率的な計算法の提案者は，楕円曲線暗号系の開発者でもある Miller である．

（※文責: 小泉建敬）

1.4　課題の概要

本プロジェクトでは ECMNET にランクインすることと，STUDIOKAMADA に掲載されている未だ素因数分解されていないより長大桁の合成数を素因数分解することを目標としている．長大桁になるほどに計算時間は長くなるので高速化が重要であるが，昨年度においてプログラムがかなりの完成度を得たため新たな切り口での高速化を目指すこととなった．よって新たな試みとして FORTRAN という言語によるプログラムの書き直しによって更なる高速化を目指す．また，未来大学を志望している人にプロジェクトを理解してもらうため，Web サイトの改善によって FUN-ECM の情報と ECM についてのより分かりやすい説明を Web ページにて発信し，ECM のさらなる普及を目指す．

（※文責: 矢和田航平）

第 2 章　到達目標

2.1　今年度のプロジェクトにおける目的

本プロジェクトの今年度の最大の目的は FUN-ECM プロジェクトで作成したプログラムによって解決した素因数分解の結果が ECM NET，STUDIO KAMADA に記載されることである．そのためには既存のプログラムを改善が必要である．そこで，単位時間ごとの計算速度を向上させた．また，この目的を達成するために，既存のプログラムの高速化を行った．さらに論理回路について学習し，FPGA を用いて新たなハードウェアの実装も行った．

このプロジェクトの 2 つ目の目標として，ECM を世の中に広めるということも行った，これは，ECM 研究者を世の中に増やすことで，新たな素因数の発見のための競争が起こり，ゆくゆくは世の中の RSA 暗号より正確な安全性評価につながると考えたからである．また，FUN-ECM プロジェクトが ECM を広めるということには大きいな意味があると考えている．その理由は，本プロジェクトは大学生によるプロジェクトであるためである．よって，活動に興味を持つ対象者が，同年代の大学生や，これから大学進学を控える高校生などの若年層となる可能性が高いからである．若年層に ECM を広めることができれば，その後も加速度的に広まっていく可能性が高いのではないかと考えた．そこで本プロジェクトの活動内容の紹介，ECM の紹介などの内容を盛り込んだ新たな WEB サイトの作成を行った．

（※文責: 池田晴輝）

2.2　課題達成の為の分担

私達は課題達成のために高速化班と WEB サイト班の 2 つの班を編成した．高速化班の活動目的は，素因数分解プログラムの高速化を検討することである．WEB サイト班の活動目的は，ECM を用いた素因数分解の広報活動である．高速化班のメンバーは，青山，鳴海，染川，矢和田である．WEB サイト班のメンバーは，池田，小泉，上田である．WEB サイト班は前期及び後期を通して活動した．

しかし，高速化班は途中で解散し，そこからさらにシステム班とハードウェア班を編成し直した．理由として，昨年度までに作成された素因数分解プログラムとの比較を行うまでの工期に見通しがつかない．そして，C 言語で書かれている素因数分解プログラムを FORTRAN で書き直すのは高速化と言えない．という 2 つが挙がった．詳しくは高速化の検討の項で記述される．

新しく編成されたシステム班の活動目的は，既存の素因数分解プログラムを使いやすくすることである．ハードウェア班の活動目的は，FPGA で素因数分解プログラムを実装し高速な素因数分解を検討することである．システム班のメンバーは，鳴海，染川，矢和田である．ハードウェア班のメンバーは，青山である．なお，ハードウェア班の活動には教員の白勢先生も参加した．

最終的には，システム班，ハードウェア班，WEB サイト班の 3 つの班で本プロジェクトは活動した．

（※文責: 上田隼人）

第 3 章　活動内容

3.1　基礎学習

自分たちが 1 年間プロジェクト学習を行うにあたって，なぜ素因数分解の高度な技術が必要なのかを理解する必要があった．そのため，前期の 1 ヶ月間の時間を基礎学習期間として使用した．方式は，全体で自習を行う方式とプロジェクト担当教員の講義方式の二つで行われた．学習には『楕円曲線暗号入門 (2013 年度)』(伊豆哲也著) を用いた．ここで，素因数分解と暗号の関係性，楕円曲線法の必要性，楕円曲線法の手順と論理を理解した．前述を踏まえた上で，素数を見つけるプログラムのアルゴリズムがどのようなものになっているかを理解する時間を全体で設けた．

（※文責: 小泉建敬）

3.2　高速化班の活動

3.2.1　高速化の方法の検討

まず高速化班では，前年度までに作成された素因数分解プログラムをどのように高速化するかを検討した．前年度には楕円曲線離散対数問題の有効な解決方法である Baby-step Giant-step 法の実装による高速化が行われていたので、今年度は別のアプローチから高速化を図ろうとした。検討の結果，次の 2 つの方法が上がった．

- プログラムを改良して，複数の PC による並列分散処理を可能とする方法
- プログラムが C 言語によって作成されていることから，C 言語よりも高速に演算処理ができるプログラミング言語を使用して素因数分解プログラムを作成する方法

高速化班のメンバの一人が過去に FORTRAN を使用していたこともあり，前期は後者の方法を採用して高速化を図ることになった．なお，前者の方法については FORTRAN 版の素因数分解プログラムが完成したのち，時間に余裕があれば実験することにした．

（※文責: 青山和弘）

3.2.2　FORTRAN の学習

FORTRAN の中でも FORTRAN 90 というバージョンを用いてプログラミングをするにあたって，FORTRAN 90 のルールやその例外事項，FORTRAN 90 の関数の種類や動き，どのようなときにどのようなエラーが発生するのか，そしてどのようにすることでそのエラーが解消されるのかなどの FORTRAN 90 への知識が不足していた．そこで本学の情報ライブラリーで FORTRAN 90 の参考書を借りて基礎について学習し，ネットからプログラムを探しそれを FORTRAN 90 で作成して実行することになった．これによって具体的に以下のことが分かった．

- FOTRAN の概要

FORTRAN は 1950 年代に IBM のジョン・バッカスらの手で誕生した世界初の高級プログラミング言語であり，FORmura TRANslation の略である．FORTRAN，IBM1401 版 FORTRAN，FORTRAN Ⅱ，FORTRAN Ⅲ，FORTRAN Ⅳ，FORTRAN 66，FORTRAN 77，FORTRAN 90，FORTRAN 95，FORTRAN 2003，FORTRAN 2008 などがあり，長所としてもともと科学技術計算用プログラミング言語として設計されているため，各種組込み関数や複素数，そして強力な配列操作など，数値計算に便利な機能があらかじめ組み込まれている点と，バージョンアップにあたって概ね上位互換を保ちながらバージョンアップされているため，古いバージョンに沿って書かれたプログラムが新しいバージョンでコンパイル可能であることが多い点と，移植性に優れているため，作成したプログラムを変更することなく（あるいは最小限の変更で）PC からスパコンまで様々な環境で利用可能であることが多い点がある．短所として科学技術計算に特化した結果，他の機能をほとんど捨てているので科学技術計算以外が苦手であるという点がある．

- FORTRAN 90 のプログラムの書き方

FORTRAN 90 では，[program プログラム名] の記述からプログラムが始まり，[end program プログラム名] の記述までがプログラムとなり，その間に printf などの実行したい処理を書き込むこととなる．基本的な書き方のルールとしては，1 行は 132 文字までで大文字小文字の区別はない点や，!を記述することで以降からその行の末までがコメントとなる点，一行に入りきらない場合には&を末尾に配置することで次の行に継続することが可能である点，及び文字列の途中での継続は行末と次の行の最初にも&が必要となる点などが存在する．また，FORTRAN 90 には 4 つのプログラム単位があり，宣言部，実行部，副プログラム部の順番で記述される必要がある．プログラム単位は FORTRAN 90 においてファイル分割を許されている最小の単位でもあり，例えば主プログラムと外部サブルーチン 2 つで構成されるプログラムがあった場合に，全てを一つのファイルに記述することも可能だが，主プログラムと外部サブルーチン 2 つをそれぞれ別のファイルに記述することも可能である．主プログラムや外部副プログラムやモジュールプログラムの中で内部副プログラムを定義することもできるが，これは中に内部副プログラムを含むことはできない．4 つのプログラム単位の内訳は以下のとおりである．

1. 主プログラム
   実行の開始場所であり，[program] から [end program] までとなる．また，注意するべき点としてプログラムの実行は，主プログラムの実行部が上から順番に実行される点と，宣言部を実行部の中に書くことはできないという点がある．以下が記述の基本である．

```
program プログラム名
  宣言部
  実行部
  contains
          ▼
    subroutine　内部副プログラム名
      内部副プログラム宣言部
      内部副プログラム実行部
    end subroutine 内部副プログラム名
          ▲
    (※) 複数の内部副プログラムを使用したい場合は，▼から▲までを使用したい内部副プログラムの数だけ繰り返す．
  end program プログラム名
```

2. 外部副プログラム

関数もしくはサブルーチンでどこにも属さないものである．何にも含まれず，内部副プログラムを含むことができる．以下が記述の基本である．

```
subroutine サブルーチン名
  宣言部
  実行部
  contains
          ▼
    subroutine　内部副プログラム名
      内部副プログラム宣言部
      内部副プログラム実行部
    end subroutine 内部副プログラム名
          ▲
    (※) 複数の内部副プログラムを使用したい場合は，▼から▲までを使用したい内部副プログラムの数だけ繰り返す．
  end subroutine サブルーチン名
```

3. モジュール

モジュールは他のプログラム単位と共有できる宣言，型定義，手続，またはインタフェースを含んでいるプログラム単位である．また，モジュール内にモジュール副プログラムを定義し，更にその内部に内部副プログラムを持つことができる．以下が基本の記述の方法である．

```
module モジュール名
  宣言部
  実行部
  contains
    function モジュール副プログラム名
      モジュール副プログラム宣言部
      モジュール副プログラム実行部
      contains
            ▼
    subroutine　内部副プログラム名
      内部副プログラム宣言部
      内部副プログラム実行部
    end subroutine 内部副プログラム名
            ▲
    (※) 複数の内部副プログラムを使用したい場合は，▼から▲までを使用したい内
部副プログラムの数だけ繰り返す．
    end モジュール副プログラム名
 end module モジュール名
```

4. 初期値設定
   初期値設定プログラムは名前付き共通ブロックの変数の初期値を提供するプログラム単位である．名前を持つ必要はないが，1 つの実行形式プログラムの中に名前のない初期値設定プログラムは 1 つしか存在することができず，インタフェース宣言と実行分は存在できない．以下が基本の記述の方法である．

```
BLOCK DATA 初期値設定名
  宣言部
END BLOCK DATA 初期値設定名
```

- FORTRAN の型

FORTRAN 90 では変数は宣言してから利用する必要があり，変数名は英字で始まる任意の英数字とアンダースコアの組み合わせで最大 31 文字まで指定可能である．また，文字列は括弧を用いて長さを指定することができ，指定を行わなかった場合には長さは 1 となる．以下のように FORTRAN 90 にはいくつかの組み込み型が用意されている．

整数型　interger
実数型　real
倍精度実数型　double precision
複素数型　complex
倍精度複素数型　complex(kind(0d0))
論理型　logical
文字型　character

- FORTRAN の条件分岐

FORTRAN 90 では if 文を使うことによって条件分岐が可能である．以下にいくつかの例を示す．

条件が真の場合のみの処理の記述

```
if (条件) then
  条件が真の場合の処理
end if
```

処理が 1 文で済む場合

```
if (条件) 条件が真の場合の処理
```

偽の場合の処理も記述する場合

```
if (条件) then
  条件が真の場合の処理
else
  条件が偽の場合の処理
end if
```

条件が複数の場合

```
if (条件 1) then
  条件 1 が真の場合の処理
else if (条件 2) then
  条件 1 が偽で条件 2 が真の場合の処理
else if (条件 n) then
  条件 1 から条件 n-1 が偽で条件 n が真の場合の処理
else
  条件 1 から条件 n がすべて偽の場合の処理
end if
```

- FORTRAN の繰り返し文

FORTRAN 90 では繰り返し文を利用する場合は do 文を使用する．do 文は以下のように記述する．

```
do 変数=初期値, 最終値 [, 刻み幅]
  繰り返したい処理
  end do
```

また，ループを途中で抜け出したい場合には以下のように exit 文を利用する．

```
do 変数=初期値, 最終値 [, 刻み幅]
  繰り返したい処理
  if(条件式) exit
end do
```

また，ループの最初に戻りループを継続する場合には以下のように cycle 文を使用する．

```
do 変数=初期値, 最終値 [, 刻み幅]
  繰り返したい処理 A
  if(条件式) cycle
  繰り返したい処理 B
end do
```

上の例の場合条件を満たして cycle 文が実行された場合，処理 B は実行されずに do 文の最初に処理が移行する．
以上によって FORTRAN のテストを行うにあたって必要な最低限の知識を得ることができた．

（※文責: 矢和田航平）

3.2.3 FMLIB の導入

FUNECM プログラムの高速化を図るため FORTRAN でのプログラミングを行うこととなったが，FORTRAN で扱える整数は 8 バイトの値までであり，桁数が足りないことが分かった．そのため多倍長整数を扱う方法を模索し，FORTRAN 用の多倍長計算用パッケージである FMLib を採用することとなった．FMLib 導入に伴い，FMLib のマニュアルが英語で記述されていたためこれの日本語化を行った．

（※文責: 染川眞輝）

3.2.4 FORTRAN と他言語の性能比較

FORTRAN は一般的に科学技術計算分野では高速なプログラミング言語とされ，数多のプログラミング手法が開発された今日に至っても未だ高性能計算機等で用いられている．

しかし，昨今に渡り利用されてきたシステムをそのまま流用していることから FORTRAN が用いられているということも有り得るため，我々は昨年度までに作成されたプログラムを記述している C 言語と FORTRAN の比較を行うこととした．なお，利用している FORTRAN は FORTRAN90 であり，コンパイラはどちらも GNU Compiler を用いた．速度計測は Time コマンドを用いた．

参考までにライブラリを用いていない全て手書きの Python も速度比較を行った．

1. ライプニッツの公式を用いた円周率の導出

   ライプニッツ公式は以下の級数で表される．

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - ... = \frac{\pi}{4}$$

   これは初項が 1 で各項が奇数の逆数である交項級数が $\pi/4$ に収束することを意味している．総和の記号を用いると以下のようになる．

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}$$

   今回の比較ではこの式を用いて円周率の導出を $n = 1,000,000$ にて行った．これにより単純な繰り返し計算を行う速度を測定した．結果は以下の表に示す．

表 3.1　ライプニッツの公式による円周率導出の実行結果

	C 言語		FORTRAN		Python	
	Real	User	Real	User	Real	User
1	6.151	6.125	6.159	6.156	27.979	27.953
2	6.326	6.328	6.135	6.125	26.303	26.281
3	6.217	6.219	6.142	6.125	26.784	26.750
4	6.253	6.234	6.137	6.141	26.265	26.250
5	5.904	5.844	6.131	6.125	27.202	27.156
AVE	6.170	6.150	6.141	6.134	26.907	26.878

   表の通り C 言語と FORTRAN の実行速度は殆ど変わらなかったが，僅かに FORTRAN が高速であるという結果が得られた．

2. モンテカルロ法を用いた円周率の導出

   モンテカルロ法とはシミュレーションや数値計算を乱数を用いて行う手法の総称であるが，この比較ではその中でも円周率の近似値を計算するアルゴリズムを用いた．

   これは，まず 1×1 の正方形内にランダムに打点し，原点からの距離が 1 以下の点を数え X とする．この操作を N 回繰り返したときに，$4X/N$ を円周率の近似値とするという方法である．

   今回の比較ではこの式を用いて，円周率の導出を $1,000,000$ 点について行った．これにより条件分岐等の複雑な処理を伴う計算を行う速度を測定した．結果は以下の表に示す．

   表の通り FORTRAN は C 言語より高速であるという結果が得られた．

これら 2 つの比較から FORTRAN によるプログラムの記述によって高速化が図れるのではないかと推測し，我々は開発を始めた．

（※文責: 鳴海雄登）

表 3.2　ライプニッツの公式による円周率導出の実行結果

	C 言語		FORTRAN		Python	
	Real	User	Real	User	Real	User
1	3.228	3.219	1.740	1.734	48.145	48.109
2	3.209	3.203	1.742	1.734	46.932	46.906
3	3.226	3.219	1.739	1.734	47.823	47.781
4	3.217	3.219	1.738	1.734	47.302	47.234
5	3.568	3.563	1.740	1.734	46.239	46.188
AVE	3.290	3.285	1.740	1.734	47.288	47.244

3.2.5　高速化班の方針転換に関して

高速化班は前期に行っていた FUNECM プログラムを FORTRAN で記述することで高速化を行うという方針を断念することとなった．主な断念の理由としては，開発にかけられる時間が足りないことが挙げられる．時間が足りないと判断した要因は，FUNECM プログラムが数年かけて開発されたものであり，本年度内での FORTRAN による開発が不可能であることと，次年度以降に引き継いだ際に発生する FORTRAN の学習コストが高いことが挙げられる．FORTRAN を用いる利点が少ないことや，FORTRAN である必要性を考慮した際に学習コストに見合ったものではないと判断した．加えて前期に行った C 言語と FORTRAN において，同じ処理を実装した際の処理速度比較の結果から FORTRAN の処理速度が高速であると判断するには判断材料が乏しいという意見が出たためである．また以前に行われた研究によると C 言語と FORTRAN において 2 次元的に動的確保を行わない限り C 言語と FORTRAN の間に演算性能に大きな差がないことが言及されており，I/O 性能に関しても C 言語で記述されたプログラムと FORTRAN で記述されたプログラムの間に大きな性能の差がないことが言及されている．以上の点から FORTRAN で記述したプログラムが既存の C 言語で記述されたプログラムより高速であるという判断を撤回することとなった．加えて，記述言語を変えたことによって生じる可読性の低下なども問題点として挙げられた．可読性の低下から保守に支障をきたすといった問題を事前に防ぐという面でも FORTRAN での記述を断念することとなった．

（※文責: 染川眞輝）

3.3　広報班の活動

3.3.1　FUN-ECM の現在の Web サイトの構造化

Web サイトを新しく立ち上げるあたって，2 年前に FUN-ECM として作成した Web サイトを振り返った．そこで上がった問題点が 2 つあった．1 つ目はデザイン性の問題である．今回の Web サイト作成の目的は ECM を世の中に広めることであるため，私たちは，老若男女幅広い年代の方々が親しみやすく，操作しやすい Web サイトを目指した．結果として，デザインはシンプルを追求しようという方針で意見が合致した．そして，2 つ目に実際に掲載されている情報量よりも Web サイトを閲覧した際の情報量が多く見えてしまうという問題点があった．そこで，よりス

マートに，よりコンパクトに Web サイト作りができるのではないかと考え，従来の Web サイトを細かく分析した．その一貫作業として，構造化を行った．

構造化の説明に入る前に，Web サイトの構造について幾分か示したいと思う．一般に，Web サイトの構造は階層構造となっている．一番上の階層はトップページで，その次の階層には大きなカテゴリーがある．それぞれのカテゴリーの次には，より小さなカテゴリーが垂下している．これら一連の階層構造を一目で確認できるようにする作業のことを私たちは構造化と位置付けた．

具体的な作業内容の説明に入る．まず初めに Web サイト班のメンバー 3 人で平等に担当のカテゴリを割り振った．次に 3 人それぞれが旧 Web サイトを閲覧しながら担当カテゴリの階層構造を手書きで書き起こした．A4 用紙一面に収まるサイズで，文字のみではなく適宜図を挿入しつつ表記した．その後，Adobe Illustrator(アドビ イラストレーター) を用いて，3 人がそれぞれ作成した図を一面で見ることができるように情報の統合を行い，旧 Web サイト全体の階層構造図 (図 3.1 構造化) を完成させた．Adobe Illustrator(アドビ イラストレーター) は，アドビシステムズが販売するベクターイメージ編集ソフトウェア (ドローソフト) である．このソフトウェアを使用した理由は，文字サイズやフォント，図形などをオリジナルにカスタマイズして使用できるという特徴を持っていて，描画ツールとして優れていると判断したためである．結果的に，この階層構造図をもとに議論を繰り返すことで，旧 Web サイトの不要箇所，新 Web サイトで追加すべき項目を決定することができた．これは，上で述べたよりスマートでコンパクトな Web サイトを作成するために，大変効果的であったと同時に，欠かせない作業であったと考える (図 3.1).

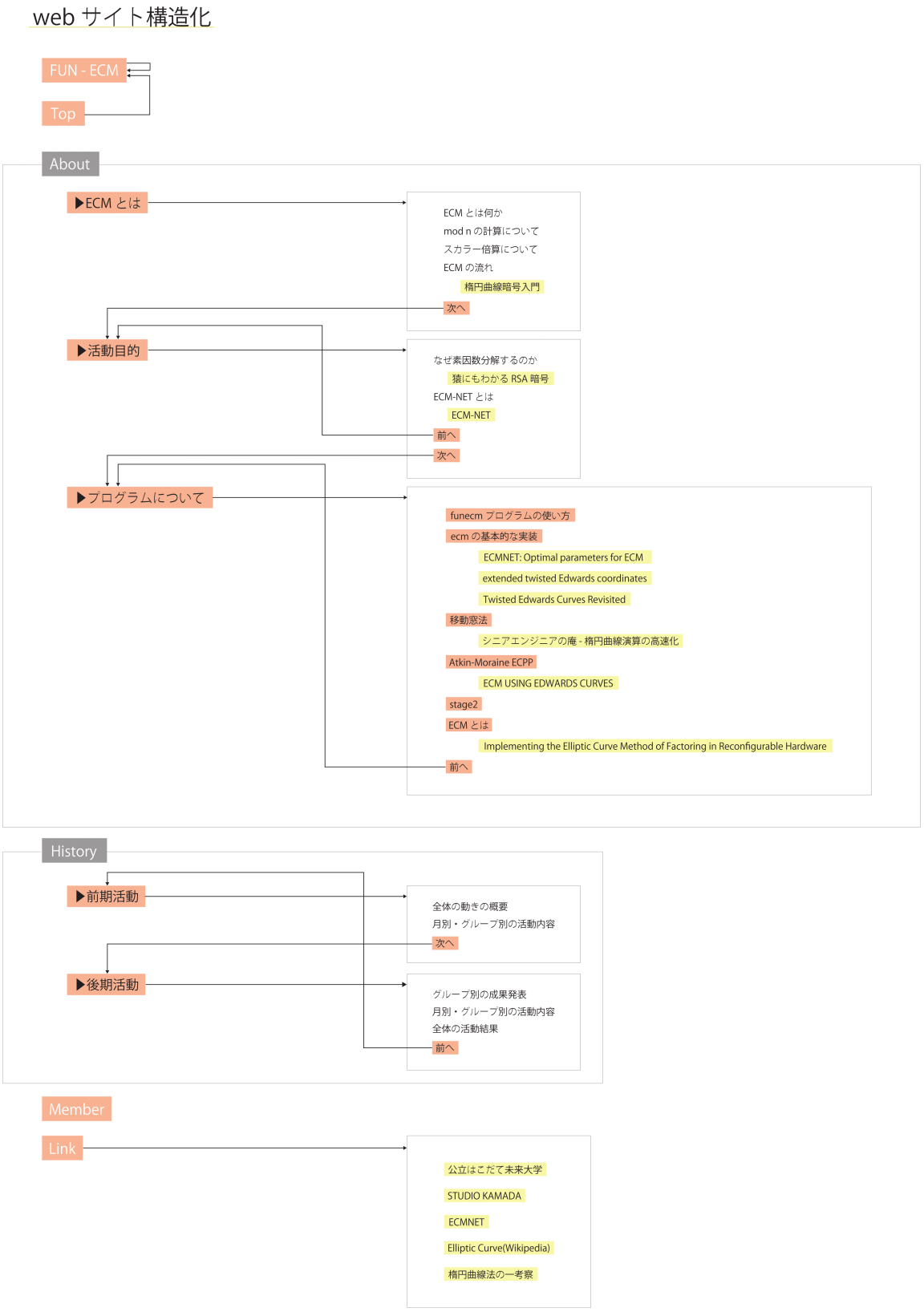

図 3.1　構造化

（※文責: 池田晴輝）

3.3.2 HTML の学習と書き込み

Adobe Illustrator を用いて作成した担当カテゴリのプレビューをもとに，実際の HTML のコーディングを行った．コーディングは池田が WORKS，ECM，上田が MEMBER，NEWS，小泉が HOME, PURPOSE，PROJECT というようにメンバーで担当カテゴリを分けて、基本的には個人作業として行った。しかし, 3 人で揃って作業を行うことが多かったため、お互いに不明点に関しては助け合い、情報共有も怠ることなくスムーズに作業を行うことができた．この点に関しては, お互いに良い刺激を及ぼしあえたと考えている．また，コーディングと並行して，Web 上でプログラミングを学べる学習サイト (Progate) を利用して，HTML の学習を行った．私たちが HTML の学習方法として Progate を選んだ理由としては初学者向けであるという点と無料で利用できるという 2 点が大きかった．実際に使ってみると説明のフェーズの後すぐに実際にコーディングを行うという作業が繰り返し行われる仕組みとなっていた．説明のフェーズではイラストが豊富に使われたスライドが盛り込まれていて, 新しい概念であってもイメージで頭の中にインプットできる仕掛けになっていた．実際のコーディングのフェーズではオンライン環境であればすぐにコンパイルが完了しプレビューが表示され, 間違っている部分もすぐに復習できるため, ゲーム感覚で次々に勉強を進めることができた．実際のコーディングをすぐに行えるという点が最も魅力的だった．無料範囲のみの利用となったが, ここで基礎を身に着けたことによって, 後の, header と footer の調整を円滑に進めることができたので, 意義のあるものだったと考えている．

（※文責: 池田晴輝）

3.3.3 CSS の学習と書き込み

CSS（Cascading Style sheets，カスケーディング・スタイル・シート）とは，ウェブページのスタイルを指定する言語である．ワープロソフトなどで作成される文書も含めて，文書のスタイルを指定する技術全般をスタイルシートと言う．HTML や XHTML を用いて作成されたウェブページにスタイルを適用する場合には，スタイルシート言語の一つである CSS を一般的に利用する．そのため，私たちは既存のウェブサイトを改善し，新しいウェブサイトを作成するにあたって CSS を学習する必要があった．私たちは CSS の基礎を学習するために，Progate(https://prog-8.com/) というウェブサイトを用いて学習を行った．Progate はオンラインプログラミング学習サービスであり，初心者でも学びやすい学習環境が整っている．そこから，イラスト中心のスライドとコーディングを実践することで CSS の学習を進めた．CSS の学習を終えた後に，実際に新規ウェブサイトのコーディングを行った．コーディングとはプログラミングと違い，仕様書通りにプログラミング言語に置き換えることである．プログラミングは設計を含むのに対して，コーデイングは設計を含まないという違いがある．新規ウェブサイトのコーディングにあたって，ウェブサイト全体の大きさを変化させた際に，ウェブサイト全体のレイアウトが崩れないことを意識する必要があった．そのため，比率を利用した大きさの指定と実際にサイズを用いて指定する二つを場合によって使い分ける必要があった．また，HTML で記述された文章の表示は，基本的に上から下に対して縦に要素が配置される構造になっている．ウェブサイトを作成するにあたって，この構造を左から右に対して横に要素を指定する必要がある場面が多く存在する．ここで，要素が block，inline，inline-block のどれであるかを判断して適切なプロパティを採用する必要があった．この条件を理解することが難しいため，コーディングと CSS の学習は同時進行で行われた．ウェブサイトの

作成にあたっては HTML の学習と記述に比べて CSS の学習と記述に大幅に時間を費やしたと言える．

（※文責: 小泉建敬）

3.3.4 PHP の学習と書き込み

ウェブサイトには，素因数分解プログラムが素因数分解を行った結果を表示する NEWS ページが存在する．具体的には，素因数分解を行った日付，成功したか失敗したか，素因数分解の対象の数が記述されている．これらのデータはサーバの MySQL に保存されているので，表示するには PHP で MySQL を操作する必要がある．よって PHP でウェブサイトを構築する必要があった．NEWS ページを担当した上田は，PHP でのコーディング経験がないので，基礎学習を Progate というサイトを利用して行った．その後，PHP から MySQL への接続，及び接続の管理を PHP.net で学習した．

次に，PHP が正常に実行されているかを確認するために，ローカルサーバを構築した．使用したソフトウェアは，Web サーバソフトウェアの Apache である．Apache を使用するために，パッケージ化されたアプリケーションである XAMPP を利用した．これにより MySQL もインストールされた．MySQL には素因数分解のデータのバックアップをインポートしておいた．

実際の PHP の書き込みではデータベースのデータを，素因数分解を行った日付と，成功か失敗かと，素因数分解の対象の数をテーブルにして表示するという処理を行うようにした．

（※文責: 上田隼人）

3.3.5 JavaScript の学習と実装

楕円曲線法に関しての説明を行う ECM のページでは，実際の楕円曲線のグラフを挿入した．その際に，変数の値を閲覧者自身が設定し，変数に応じたグラフの動きの変化の様子を確認できる機能を追加した．そこで，JavaScript が必要となったため JSXGraph というクロスブラウザ JavaScript ライブラリを利用し，グラフを作成した．挿入に関しては Progate での学習をベースとした．ほかに JavaScript の使用を検討した部分として, WORKS ページのページ内遷移があげられる. 今回は HTML と CSS のみで対応したが, 今後使用していく中で処理速度の面などで問題が生じた場合には JavaScript の使用を検討すべき点である．

（※文責: 池田晴輝）

3.3.6 全体レイアウトの試案

ウェブサイトのレイアウトを決定する前に，池田，小泉，上田がそれぞれ様々なウェブサイトを閲覧し，良いと感じたサイトを挙げていった．挙げられたレイアウトには，画面上部にヘッダーがあり画面下部にフッターがあるものや，画面左部にグローバルナビゲーションがあるものがあった．その後，池田，小泉，上田の 3 人で話し合い，一番良いと思われるレイアウトを決定した．採用されたレイアウトは，画面上部にヘッダーがあり，画面下部にフッターが配置されているものである．ヘッダーにはホーム画面に遷移するロゴとグローバルナビゲーションが配置されている．

フッターには表示されているページの遷移状態と公立はこだて未来大学のウェブサイトに遷移する公立はこだて未来大学のロゴが配置されている．ページのボディには左右にマージンを取り，画面の右から左まで一杯に文字が表示されて見づらい状況になるのを回避するように構成されている．このレイアウトを採用した理由は，普段インターネットを利用する人が一番見慣れているであろう構成であり，使いやすいと判断したためである．

（※文責: 上田隼人）

3.3.7　ロゴの制作

FUN-ECM である本プロジェクトのウェブサイトには既にロゴが存在した (図 3.2)．しかし，そのロゴは文字だけで構成されており，ECM と一目でわかるものではなかった．よって新しくロゴを製作することになった．新しいロゴには，既存のロゴに代表的な楕円曲線を追加した．また，全体の色を青から黒に統一し，シックなものにした (図 3.3)．このロゴはウェブサイトのグローバルナビゲーションの左側と，発表用のスライドの右上にて使用された．ロゴを製作したのは広報班の小泉である．Adobe Illustrator を用いて制作した．その後，上田と微修正を重ねて完成させた．

図 3.2　旧ロゴ

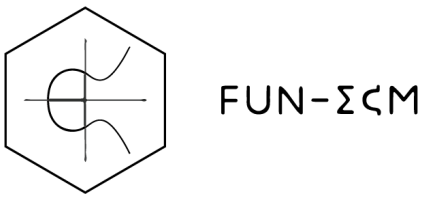

図 3.3　新ロゴ

（※文責: 上田隼人）

3.3.8　カテゴリ分け

ウェブサイトで主に伝えたいことを軸とし，それに付随する情報のカテゴリを決定した．ウェブサイトの軸は，公立はこだて未来大学に FUN-ECM というプロジェクトが存在することを伝えることである．カテゴリは，HOME・PURPOSE・WORKS・MEMBER・ECM・PROGRAM・NEWS である．

（※文責: 小泉建敬）

3.3.9　プレビューの作成

新規のウェブサイトを作成するにあたって，最初から HTML と CSS を用いてコーデイングを行うのは難しいと考えた．そのため，一度 Adove Illustrator を用いてウェブサイト全体のプレビューを作成することを考えた．Adove Illustrator を用いたプレビュー作成では，実際に作成するウェブサイトと同じ大きさのアートボードを用いて文字サイズはピクセル単位に変換して作成した．また，カラーにおいては，コーディングを行いやすいように CMYK を用いた．前述のレイアウト試案とカテゴリ分けを踏まえて小泉が HOME のプレビューを作成した．HOME のプレビューをもとに各カテゴリーごと，全ページのプレビューを作成した．担当したカテゴリは，小泉が HOME･PURPOSE･PROGRAM を，池田が WORKS･ECM を，上田が MEMBER･NEWS である．各々の担当ページのプレビュー作成を終了した後に，担当したカテゴリのプレビューの評価を行い，修正を行った．

（※文責: 小泉建敬）

3.3.10　各カテゴリのマージン統一

各々の担当ページを作成した後に，担当ページのマージンが実際にウェブサイトを使用した際に使いやすいかどうか検討した．初めの検討では文字サイズの修正を行った．各々が担当ページのコーディングを行ったため，タイトル，文章ともにそれぞれ大きさが異なっていた．どのページに遷移した時も見やすい文字サイズを指定して各々の担当ページに反映させた．次の検討では文字と文字の間，文字と画像の間，画像と画像の間の三つの間隔を調節した．各々の担当ページを見て明らかであったことは，マージンが小さく内容の繋がりが見えにくいことである．そのため，内容の繋がりがある箇所には見やすく小さいマージンを適用した．その反対に内容の繋がりがきれる箇所では大きめのマージンを適用した．このどちらにも共通して，文字と画像との間には大きめのマージンを適用して，画像を見やすく強調させる効果を付随させた．最後に上記 2 つを合わせて，各担当ページに見にくい点が存在しないかを確認した．

（※文責: 小泉建敬）

3.3.11　各カテゴリの header と footer の内容統一

作成するウェブページの担当が決めた後，各自ウェブページの製作に取り掛かった．ウェブページを作っていく中で各ページの内容を追加していくと，グローバルナビゲーションとフッターのレイアウトが崩れやすく，HTML と CSS が複雑になっていることに気が付いた．そこで，HTML5 から追加されたタグの「header」と「footer」を使用し，どのページでも内容とレイアウトが変わらないグローバルナビゲーションとフッターの部分を統一した．それにより，HTML と CSS が簡略化され，各ページ担当者はコーディングがしやすくなった．また，他人に上手くいかないところを教える際も，簡略化されたことにより問題点が発見しやすいので教えやすくなった．

（※文責: 上田隼人）

3.3.12　ファイルの統合

ページ担当者全員の作業が終わり次第，全てのファイルの統合を行った．そして，各ページから他のページに正常に移動できるか確認した．その後，ウェブサイト全体のフォント，文字の大きさを統一し，行間などの最終調整をした．最終成果発表会時点では，ファイルの統合を行い，全体の微修正を行ったところで終わった．

（※文責: 上田隼人）

3.3.13　サーバー経由でのアップロード

ウェブサイトをアップロードする Web サーバは XREA(エクスリア) というレンタルサーバである．プランは XREA Free で，料金は無料である．その代わりにウェブページの上部中央に広告が表示されてしまう．このレンタルサーバを選んだ理由は，データベースと PHP が利用可能であったためである．データベースは素因数分解を行った結果を格納するために必要になる．PHP については，当初の予定ではデータベースを JavaScript から操作しようとしていた．しかし，同じ操作を PHP にて行うとソースコードが短く，且つ，可読性が高いことが分かった．よって，レンタルサーバを選ぶ際の条件として，データベースと PHP が利用可能なものにした．

実際にファイルをアップロードする際には WinSCP というソフトウェアを利用して，FTP サーバ経由でファイルをアップロードした．

（※文責: 上田隼人）

3.4　システム班の活動

3.4.1　昨年度のプログラムの運用

システム班では前期に活動していた高速化班から引き継いで昨年度のプログラムの運用を行った．本報告書では，システム班の活動として記述する．

昨年度までに制作された素因数分解プログラムを用いた未解決の長大桁合成数の素因数分解を行った．素因数分解を行った合成数はいずれも「STUDIO KAMADA」にて公開されているニアレプディジット関連の数の素因数分解の過程で発生した cofactor であった．

楕円曲線法を用いた素因数分解では，その特徴上合成数の素因数の桁数を推測してそれに応じたパラメータ B1 と B2 を用いる．この B1 の値に応じて素因数分解プログラムを動作させる 1 回あたりの時間が変わり，B1 が大きいほど長い時間がかかることとなる．そこで，プロジェクト開始直後は同じ合成数から発見できる素因数の桁数を最初に推測したものから徐々に上方へ見直していたが，それを 50 桁と固定して様々な合成数の素因数分解を順次試みる方針へと転換した．この 50 桁という桁数の根拠は，我々が素因数分解を行っている環境では 50 桁と推定してパラメータを決定することにより，プログラムの実行時間が約 1 週間とキリが良くなるというのが理由のひとつである．また，「STUDIO KAMADA」にて公開されている未解決の合成数は 50 桁と推定して分解されていない場合が多く，この B1 を用いた素因数分解を行い続ければ成功する可能性が向上するのではないかというのがもう一つの理由である．

実際に B1 の採用方法を上記の方法に改めてから現在に到るまで 3 件の素因数分解に成功しているため，明確な根拠はないが素因数分解をより多く成功させるという点では正しい方針なのではないかと推測される．

また，運用中に発見したこのプログラムの改善すべき点を基に，次項で示す昨年度のプログラムの改良を行った．

表 3.3　素因数の桁数に対応した B1

桁数	B1
20	11,000
25	50,000
30	250,000
35	1,000,000
40	3,000,000
45	11,000,000
50	43,000,000
55	110,000,000
60	260,000,000
65	850,000,000
70	2,900,000,000
75	7,600,000,000
80	25,000,000,000

（※文責: 鳴海雄登）

3.4.2　昨年度のプログラムの改良

前項にて述べた昨年度のプログラムの運用により，このプログラムの改善すべき点を 3 点発見した．

まず 1 点目は，プログラムを動作させる際に分解したい合成数毎にコマンドを送信する必要がある点である．既存のプログラムでは一回の動作で一つの合成数の素因数分解を行うことができるが，学内ネットワークからのみ接続できるワークステーションでプログラムを動作させているため，複数の合成数を順次素因数分解したいと考えた．

2 点目は，プログラム終了を確認する手段がワークステーション上でのプロセスの動作状況を見ることのみであるという点である．プログラム動作直後に出力結果ファイルが作成される．その内容は素因数分解が成功した場合は最後に見つかった素因数が記述されるが，見つからなかった場合はその時点で書き込みが終了するため，出力結果ファイルの内容だけではプログラム終了の判断材料には不十分である．そこでプロセスの動作状況を確認するのだが，都度ワークステーションにアクセスすることなくプログラム終了を確認したいと考えた．

3 点目は，プログラムの動作やプログラムの終了確認をワークステーションへ SSH 接続することによって行うという点だ．動作開始や確認の都度 SSH クライアントを用いてワークステーションへ接続するのは非効率であると考えるため，Windows 上で動作する我々の素因数分解プログラ

ム専用クライアントが欲しいと考えた．

これらの改善を行うために，単一のプログラムではなく複数のプログラムを内包したシステムを構築することとした．この判断の根拠としては，既存のプログラムへ加える変更は最小限にすることによって，現在動作している部分への破壊を防ぐためである．さらに動作毎にソフトウェアを分けることによって保守管理が容易となる．

システムの構成要素は，ソフトウェア単位で分割すると 4 つである．今回のシステム化にあたってユーザとシステムのインタフェースとなるフロントエンド，ワークステーション上で素因数分解プログラムへ順次引数を渡し実行し 1 回のプログラム終了後に終了結果を WEB サーバへ転送するシェルスクリプト，昨年度までに作成された素因数分解プログラム，ワークステーションから受領した終了結果を表示する WEB サイトである．

ユーザとシステムのインタフェースとなるフロントエンドは，Windows 上で動作させることを前提とし C#により記述した．GUI 上にて素因数分解を行いたい合成数，合成数から発見される素因数の推測される桁数，処理結果を保存するファイルのファイル名の 3 点を入力して表形式の配列に追加し，表を CSV として出力し，それを学内 LAN に接続している時に限り SFTP でワークステーションへ転送する．また，転送後に SSH 接続して後述するシェルスクリプトを実行することも可能である．

ワークステーション上で動作するシェルスクリプトは，本プロジェクトで運用している RedHat Enterprise Linux 上で動作させることを前提としている．利用しているコマンドは Linux カーネルの他に 2 つあり，対話的なプログラムとのやり取りを行う expect コマンドと FTP クライアントである ftp コマンドである．セキュリティの観点と RedHat Enterprise Linux にクライアントが標準搭載されているという点から SFTP を利用してデータ通信を行う sftp コマンドを使用することが望ましいが，現在本プロジェクトで運用している外部 WEB サーバは SFTP 通信をする際に 1 ヶ月おきに認証が必要となるため，保守運用上の観点から外部サーバで追加設定を必要とせずに利用できる FTP 通信を採用した．実際の動作は，素因数分解の対象とする合成数と素因数分解をする際に用いるパラメータ，また処理結果を保存するファイル名を CSV から 1 行読み出し各変数へ代入し，それら変数を引数として素因数分解プログラムを実行する．実行後，素因数分解プログラムから出力された終了結果情報を外部サーバへポストし，CSV ファイルから次に素因数分解を実行する各変数を読み出す．

昨年度までに作成された素因数分解プログラムは，前述のワークステーション上で動作することを前提とし，C 言語により記述した．プログラムを動作させる際に引数として素因数分解を行う合成数と素因数分解を行う際に用いるパラメータ B1，B2，また素因数分解の結果を出力する際のファイル名を受け取り，素因数分解を試みる．従来のプログラムからの変更点は，プログラム動作終了時に，素因数分解を試みた合成数，終了した日付，素因数分解の成功の可否を包含した終了結果情報ファイルを出力する点だ．主な成果物としてプログラムから出力される素因数分解の結果は途中経過をすべて含んだ大きなファイルとなるため，WEB サイトで直近の結果を表示する際に，そのファイル自体から素因数分解の結果を抽出するよりも，プログラム動作段階では変数としてメモリ上に配置されている終了結果情報として望ましいもののみを包含したファイルを出力するほうが扱いやすいという観点からこのような変更を加えるに至った．

ワークステーションから受け取った終了結果情報を表示する WEB サイトは，前述の外部 WEB サーバ上で動作することを前提とし，PHP により記述した．終了結果情報ファイルは json 形式としたため連想配列として読み込み，それぞれの値をフィールド上に表示する．この終了結果情報ファイルは同一の名前をつけられたファイルであるため PHP スクリプト上で特別な指定をするこ

となく常に直近の素因数分解結果を WEB ページ上で参照することができる.

（※文責: 鳴海雄登）

3.5 ハードウェア班の活動

3.5.1 FPGA

FPGA(Field Programmable Gate Array) は PLD(Programmable Logic Device) の一種で, HDL(ハードウェア記述言語) によって内部に自由な論理回路を構成・再構成できる集積回路である.FPGA に含まれるプログラム可能な論理コンポーネント (論理ブロック) を相互接続する, 再構成可能な配線階層がある. コンポーネントを組み合わせ複雑な論理回路を構成したり, AND, OR など単純な論理回路を構成することもできる.

初期のデジタル回路は，テキサスインスツルメンツの 7400 シリーズのような AND,OR,NOT 等の単一機能を持った汎用ロジック IC を組み合わせ構成するのが一般的だった．しかし，この方法をとると回路の物理的規模が大きくなってしまう問題があった．このため，加算器，フリップフロップ，シフトレジスタのような論理回路を組み合わせた IC が作られるようになったが，それらを使用しても複雑な回路を構成しようとすると物理的規模が大きくなってしまう．これを解決しようと IC に様々な回路が組み込まれるようになり，結果として IC の規模が大きくなった．当初は IC に数個〜数十個の論理回路が組み込まれていたが，1980 年代には数千個もの論理回路が組み込まれる大規模なものになり，LSI(Large Scale Integration) と呼ばれるようになった．IC の進化に伴い，基板上に大量の IC を配置する必要があったデジタル回路が，数個の LSI を配置するだけで済むようになった．すると今度は目的とする LSI をどのように作るかという問題が浮上した．汎用 LSI に機能的過不足があっても容易に機能を足したり削ったりするのは難しいためである．専用 LSI を作れば解決するが，それには数千万〜数億円の初期コストがかかってしまう．

こうした流れと別に，1970 年代に最初から回路の要素 (AND や OR のような単一機能の数々) を盛り込んでおきユーザが手元で自由に配置･配線できる PLD が登場した．最初の PLD は規模が小さく汎用 IC 数個分でしかなかったが，1980 年代にこれを大規模なものにした CPLD(Complex-PLD) が登場すると汎用 IC 数百〜数千個分の回路を内部で構成できるようになった．PLD で重要なのは，製造のための初期コストが不要である点にある．これを受けてザイリンクスが，プログラム記憶要素として SRAM を採用し，何度でも変更できる大規模 PLD として開発したものが FPGA である.

現在 FPGA 市場を主導しているのはザイリンクスとインテルの 2 社である. ザイリンクスは 1985 年に世界で初めて FPGA を製品化した企業であり, 2009 年までは市場の 50% を占めていた. 一方，インテルはアルテラという 2010 年頃に急成長しザイリンクスと肩を並べると予想されていた企業を 2015 年に買収し市場に参入した.

FPGA の用途は多様で, デジタル信号処理, 画像処理, 暗号などが挙げられる. 暗号 (特に暗号解読) のような高い並列性が期待できる分野やアルゴリズムを FPGA 上で実装すると, C 言語などで作成したプログラムを PC 上で実行するのと比較して高速に実行できることが期待できる. また, 高速フーリエ変換や畳み込みのような高性能計算を行うのにも採用されている.

FPGA の動作を定義するためには, HDL, もしくは回路図で設計を行う. 大規模な回路を設計するときは HDL の方が適しているが, 回路図も設計の視覚化・確認が容易というメリットをもって

いる．

LSI や FPGA は，信号入力，信号出力，データ処理，データ保持，データ転送の 5 つの機能を最低限要する．それらの機能がいかに高速であるか，いかに高機能であるかより，LSI や FPGA のグレードが決定される．データをどのように処理するかはユーザのデザインに依存するが，それを実現するためにどんなエレメント，集積度，速度を FPGA が持っているかが重要となる．データ処理の機構は自分で CPU を実装してソフトウェア処理をするというのも可能である．

今回ターゲットボード (設計した回路を動作させる対象となる FPGA) として選んだのは，インテル (アルテラ) の製品である Cyclone シリーズのひとつ，Cyclone IV である．Cyclone シリーズは FPGA 市場の中でも低コストで低消費電力であり，インテルが提供する無償ソフトウェア Quartus で開発が可能である特徴がある．トランシーバや ARM プロセッサ，DDR3 SDRAM や PCI Express コントローラを内蔵した製品もある．Cyclone IV を選択した理由の一つとして，開発ボードとして入手できるためというのが挙げられる．

一方，ザイリンクスが提供しているのは，高集積・高性能・高機能な Virtex シリーズと，低価格・大量生産向けの Spartan シリーズである．前者は機能性，高速性を重視していたり大きな集積度を必要とする製品に採用される．後者は価格を抑えているため大量生産での使用という面でコストメリットがある．

（※文責: 上田隼人）

3.5.2 VHDL の学習

後期の活動として FPGA に FUNECM プログラムを実装することが挙げられた．そのため，デジタル回路設計用のハードウェア記述言語である VHDL に関する基礎学習を行った．

VHDL の概要

VHDL とは, デジタル回路設計用のはーでウェア記述言語の一種であり，標準化は IEEE/IEC によるものである．主だって FPGA や ASIC などの設計に用いられるものである．初出は 1981 年であり，名称の由来は VHSIC HDL（very high speed integrated circuits Hardware Description Language）の略称である．ハードウェア記述言語を用いることで，ハードウェア・アーキテクチャの設計からレイアウト設計，テストに至る一連の工程の中で，動作レベルやゲートレベルなどの記述を経たトップダウン設計を行うことが可能となり，生産設計性や信頼性が向上する．

VHDL の記述形式

ハードウェアの構造の基本はコンポーネント，ポート，信号からなるものである．コンポーネントはハードウェア記述のブロックに該当しゲートやチップ，ボードなど様々な大きさを持つブロックをコンポーネントと考えることができ，これはハードウェアを記述するレベルに依存することとなる．VHDL では，コンポーネントはライブラリ宣言，エンティティ宣言，アーキテクチャ本体から構成される．

エンティティ宣言はコンポーネントのポートを含め，外部からどのように見えるかを記述するための物である．エンティティ宣言において，エンティティ自身と宣言された各ポートは識別子により名前を与えられている．識別子は以下の規則に従う．

・並びはすべて英数字または'_' からなる．

・一文字目は英文字である．
・’_’ が続けて現れることや，最後の文字が’_’ となるものは不可．
・begin,end,entity 等 VHDL の予約語と一致するものはエンティティや信号，コンポーネントの名称として指定できない．
ポートの宣言は以下のように行う．

```
port（ポート名, ポート名: 方向指定 データタイプ 信号種別;
      ...
       ポート名, ポート名: 方向指定 データタイプ 信号種別);
```

・ポート名は入出力端子に相当し，識別子にあたる．
・ポートの方向指定を行う．
・ポートのデータタイプを指定する．
・信号種別には，’bus’ か’register’ を指定することができる．これらを宣言することでその信号の値をし決定する出力ドライバを切断することができる．’register’ ではドライバの出力が切断した際にその直前の値をキープするが，’bus’ では値をキープできない．
ポートの方向指定は in,out,inout,buffer を指定する．in では入力であることを示し，out では内部的に値を再利用できない出力であることを示す．inout では入出力であることを示す．buffer では，内部で値を再利用できる出力を示す．
VHDL では，あらかじめ定義された型として整数型を指定する Integer, 浮動小数点型を指定する Real,2 値の列挙型である Bit が存在する．同じく 2 値の列挙型ではあるが False,True を返す Boolean 型も存在する．加えて，ユーザが新たな型を宣言することも可能である．

アーキテクチャ本体はコンポーネントの動作や構造を記述するものである．各コードはエンティティ宣言とアーキテクチャ本体の 2 部から構成される．これが完全な記述とされる．以下に VHDL の記述形式を示す．

```
ライブラリ宣言部
導入ライブラリ宣言
エンティティ宣言部
entity エンティティ名 is
  port インタフェース・リスト;
end 識別子;
アーキテクチャ本体
architecture アーキテクチャ名 of エンティティ名 is
コンポーネント宣言
コンフィギュレーション定義
信号宣言
begin
 コンポーネントインスタンス文
 同時処理文 ( 動作の記述 )
end 識別子;
```

他のコンポーネントとの接続などがなければコンポーネント宣言，コンフィギュレーション定義，コンポーネントインスタンス文を省略することができる．階層的に設計された回路においては，上位階層から下位階層の回路呼び出すように記述する．このような記述を構造化記述と呼び，コンポーネント宣言，コンフィギュレーション記述，コンポーネントインスタンス文を用いて記述する．以下に記述方法を記す．

```
architecture アーキテクチャ名 of エンティティ名 is
  – コンポーネント宣言
  component コンポーネント名
    ポート文
    ...
  end component;
  ...
  – コンフィギュレーション定義
  for ラベル : コンポーネント名 use entity ライブラリ名. エンティティ名 (アーキテクチャ名);
    ...
begin
  – コンポーネントインスタンス文
  ラベル : コンポーネント名 port map( 信号リスト );
  ...
end アーキテクチャ名 ;
```

コンポーネント宣言はその階層で呼び出す階層のコンポーネントが使用する信号ポートを明示する．コンフィギュレーション定義ではコンポーネント宣言で指定したコンポーネントがどのライブラリに何というエンティティ名とアーキテクチャ名で定義されているかを関連付けを行う．そして

コンポーネントインスタンス文により下位階層の部品を呼び出す．
VHDL では signal,variable,constant の 3 種のオブジェクトクラスが存在する．signal は回路において配線イメージに相当し，architecture, package, entity 内で宣言され，その内部でグローバルである．信号代入文”¡=”で代入される．代入された値は同時処理を行うために直ちに代入されるわけでなく，代入の評価と実際の代入は別々に行われる．variable は回路の配線などに直接関係するものではなく，ハードウェアの動作をモデル化する際に使用するものである．これは，process,function,pricedure 内で宣言され，内部でしか参照できないローカルなものである．編入代入文”:=”で代入し，代入された値は直ちに変数へ代入され次の行意向で変更後の値を使用できる．constant は，architecture, package, entity, process, function, procedure 内で宣言で きるグローバルな定数であり，その値は宣言時に一度しか定義できないものである．

（※文責: 染川眞輝）

3.5.3 Quartus

Quartus は，アルテラが開発し現在はインテルが提供している FPGA，CPLD 用の統合開発ツールである．HDL を用いた回路設計だけでなく，回路のコンパイル，論理合成，論理ブロックの配置・配線，論理シミュレーション，消費電力の解析，ピンの配置など，開発フローに必要な機能を備えている．

現在提供されているのは Quartus Prime と呼ばれるものである．Stratix10, Arria10, Cyclone10 GX デバイスファミリーから始まる次世代 FPGA 及び SoC の先進的機能をサポートしているプロ・エディション，従来のデバイスファミリー及び Cyclone10 LP デバイスファミリーをサポートしているスタンダード・エディションがあるが，今年度購入した FPGA が搭載しているのは Cyclone IV であったため，非対応のプロ・エディションは購入しても意味がなかった．Cyclone IV に対応しているスタンダード・エディションの購入を検討したが，有償版であり，後期から使用することを考えると購入しても活用しきれないと判断し取りやめた．

以上の理由から，今回は低コスト FPGA デバイスファミリーをサポートしている無償版の Quartus Prime ライトエディションを使用し，回路の設計・開発からコンパイルまで行った．

（※文責: 青山和弘）

3.5.4 各種マニュアルの日本語化

FPGA を用いて FUNECM を構成するにあたって，使用している FPGA のハードウェアに独自の仕様や教本としている本と食い違う部分などがないか付属しているマニュアルに目を通す必要があった．しかし，マニュアルは英語で書かれているので翻訳と理解を一人で行うのは時間がかかると思われた．そこでプログラムに実際に取り掛かるメンバーとマニュアルを翻訳して読みやすい状態にするメンバーとで別れて作業を行った．マニュアルの翻訳にあたっては翻訳サイトを使用して仮に翻訳した物を，原文と見比べて確認して文脈などが不自然な部分を修正した。そして，図や表との位置関係が変わらないように Word ソフトで再配置した．この時，念のためにマニュアルの他にセットアップ作業の方法などを記したファーストガイドや一番初めに試しにプログラムを作る場合の手順について記したスタートアップガイドも同様の手順で翻訳した．

（※文責: 矢和田航平）

3.5.5　楕円曲線法を実現する回路の検討

前年度までに作成されたプログラムを FORTRAN で書き換えることによる高速化に大きな効果が期待できないこと, 及び今年度中に完了させることが困難と予測されることから, 後期よりFPGA 上に楕円曲線法を実装することを目的にハードウェア班が結成された.

環境構築, 基礎学習を完了させた後, 実装する回路についての検討を開始した. 目的とする長大桁数の素因数分解を行うために, 実装する必要があると挙げられた機能は以下の通りである.

- 128bit のデータを扱えるレジスタを 128 個内蔵したレジスタファイル
- 加算器
- 乗算器
- データ選択用のマルチプレクサ
- 桁上がりの状態を格納しておくレジスタ

（※文責: 青山和弘）

3.5.6　スカラー値を生成するリスト

ECM は分解したい合成数を N 任意の楕円曲線の適当な点を P として十分に大きな n に対して nP を計算し nP の x 座標の分母と N の最大公約数を取れば，ある確率で N の素因数が得られるという事実を利用している．このプロジェクトでは十分に大きな n には 43,000,000 までのすべての素数ごとの 43,000,000 を超えない限界のべき乗の総乗を用いている．この特定の値までのすべての素数ごとの特定の値を超えない限界のべき乗の総乗とは例えば特定の値が 10 であれば，10 までの素数は 2 と 3 と 5 と 7 であるので 2 の 10 を超えない限界のべき乗である 8 と 3 のの 10 を超えない限界のべき乗である 9 と 5 と 7 の総乗となる．この n を用いた nP を計算するにあたって，総乗をする前の 43,000,000 までの素数ごとの 43,000,000 を超えない限界のべき乗のリストを作成した．このリストを使い，総乗して n を算出してから P の n 倍の計算を行うのではなく，このリストの数を上から順に P のスカラー倍を行った．これは例えば n を 10 までの 10 を超えない限界のべき乗の総乗である 2520 だとすると，P の 2520 倍を行うのではなくまず 8 倍の計算を行ってから算出された点の 9 倍を行うというような手順を経て結果的に P の 2520 倍を求めるというような手法である．この手法を用いることによって P の n 倍の計算を直接行うよりも高速に計算を行うことができる.

（※文責: 矢和田航平）

3.5.7　楕円曲線法のハードウェア実装

楕円曲線法を行う手順を簡潔に説明すると次のようになる.

1. 任意の点 $P(x, y)$ を決定し, 楕円曲線 $y^2 = x^3 + ax + b$ を 1 つ決定する.
2. 十分大きな値 B を決定し, $Q = BP$ となる点 Q を求める
3. 点 Q の x 座標の分母と, 素因数分解の対象 N との最大公約数を求める (1 以外の値が求まっ

たらそれが N の素因数である)

手順 1 については点 P の座標を乱数で決定し, その点が存在する任意の楕円曲線を求めればよい．手順 3 についても, 最大公約数を求めるのは PARI/GP などを用いれば導出は容易である．これらを FPGA に実装した場合，確かに早く求めることは出来るが元が早いので大きな高速化には繋がらない．実装の手間と得られる効果を考慮して，今回は手順 1, 3 については FPGA には実装せず PC 上で何らかの方法により行うことに決定した．

手順 2 の点 Q を求めるための点 P のスカラー倍算は, 処理に大きな時間がかかる．また，『3.7.6 スカラー値を生成するリスト』で述べた方法によって求められた十分大きな値 B を導出するためのリスト中の要素を点 P に繰り返し乗算する手順は並列化が可能である．つまり，FPGA の並列演算を高速に処理できる特性を利用すれば，手順 2 を大きく高速化することが出来るのではないかと考えた．以上の理由から，今回は手順 2 を高速に処理するための回路を FPGA 上に実装することに決定した．

（※文責: 青山和弘）

3.5.8　設計した回路

『3.5.5　楕円曲線法を実現する回路の検討』で述べた機能に加え, その他必要となる機能について教員と相談を行い以下の図 3.4 に示すような回路を設計した．

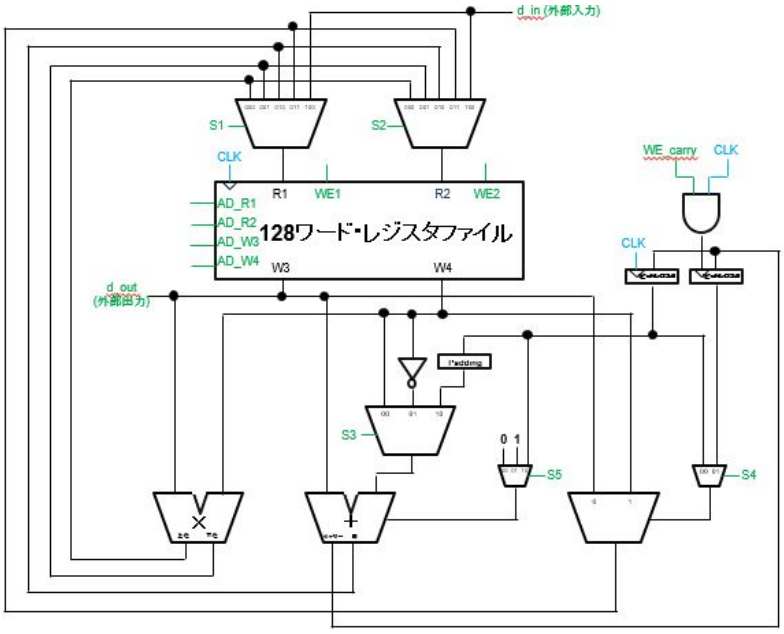

図 3.4　設計した回路

この回路図に基づき, VHDL で機能を記述した．

（※文責: 青山和弘）

3.6 中間発表

3.6.1 発表準備

ポスター

初めに，前年度のプロジェクトで作成されたポスターを参考に構成を決定した．次に概要，基礎学習，高速化班，WEB サイト班の 4 つの項目に分け，作成を分担した．ポスターの作成には Adobe Illustrator というソフトウェアを使用した．ポスターが完成次第，高速化班・WEB サイト班でレビューを行い，誤字脱字やフォントの違い等を修正した．

（※文責: 矢和田航平）

スライド

本プロジェクトの内容を説明するのにポスターと共にプレゼンテーション資料としてスライドを作成することにした．高速化班，広報班から最低一人ずつ配属されるようにポスター作成とスライド作成に人員を割り振り，他メンバーと話し合いながら発表内容を考えた．
スライドの作成には「Microsoft PowerPoint」を使用し，「Slack」を使用してバージョン管理を行った．また，模擬発表を何度か行い，修正点のあぶり出しを行うことで，わかりやすく見やすいスライドを目指した．

（※文責: 鳴海雄登）

発表原稿

前述のプレゼンテーション資料の作成と並行して発表用原稿の作成を行った．ECM について理解してもらえるようするために，基礎学習に関する説明に重点を置いた．作成した原稿はプレゼンテーション資料作成者とともに表現のすり合わせを行い，伝わりにくい表現，並びに資料との表記ゆれ等の修正を行った．加えて，担当教員にも文章などの添削をしていただき，伝わりにくいと思われる部分の修正を行った．

（※文責: 染川眞輝）

3.6.2 発表

中間発表会では前半に上田・小泉・鳴海の 3 人，後半に青山・池田・染川・矢和田の 4 人が，事前に作成したスライドをメインとして，あらかじめ相談し決定しておいた分担に従い，交代しながら発表した．

頂いた評価を後日集計したところ，「説明が聞き取りにくい，わかりにくい」といった発表自体に関する意見や，「プログラムの高速化や広報活動を行う目的が不明瞭である」といった指摘が見られた．前者に関しては発表練習の時間をあまり設けることが出来なかったことが原因だと思われるので，後期はもっと余裕を持って準備に取り掛かり練習回数を増やしたいと考えている．後者については ECM についての説明をどう分かりやすく行うかにポスターのスペースや発表時間を割か

なければいけなかったために起こったのではないかと思われる．後期はレビューを複数回行って，余分なところがないかをよく吟味したいと考えている．

（※文責: 青山和弘）

3.7　成果発表会

3.7.1　発表準備

ポスター

初めに，前期の中間発表で使用したプロジェクトのポスターを見直した．前期の中間発表で使用したポスターの欠点を列挙して一つ一つ修正を行った．前期の中間発表で使用したポスターの主な欠点としては，情報の関係性が視覚的に伝わりにくいこと，イラストによる図解が少ないことである．この前期との比較を経由して，後期では新しいポスターを作成した．前期の中間発表で使用したポスターは 2 枚である．ポスターの 1 つは，概要と基礎学習の紹介をしている．概要はプロジェクト名の由来，本プロジェクトの目的，プロジェクトの背景などである．基礎学習は学習目的，学習成果などである．ポスターの 2 つ目は sub ポスターの役割を担い，高速化班と広報班の活動内容の紹介を行っている．しかしながら，後期では高速化班がシステム班と FPGA 班の 2 つに分かれて活動を行ったため，ポスターの枚数を 3 枚にする必要があった．メインポスターの役割は前期と同様で，概要と基礎学習の紹介を行った．ポスターの 2 つ目と 3 つ目は sub ポスターの役割を担った．ポスターの 2 つ目では広報班の活動内容を紹介した．広報班の活動内容としては，再構築までのフローチャートと新規ウェブサイトのプレビューを情報の関係性を視覚的にわかるように図解した．ポスターの 3 つ目ではシステム班と FPGA 班の活動内容を紹介した．しかし，このポスターにおいては紹介内容のカテゴリにまとまりがなく，図解ができていないものになってしまった．また，各文章やイラストのマージンが人の知覚として見やすいものになっていなかった．原因としては，ポスター作成を後期活動の班分けごとに作成したことが挙げられる．実際の成果発表会の意見としても同様のことが挙げられた．

（※文責: 小泉建敬）

スライド

本プロジェクトの内容を説明するのにポスターと共にプレゼンテーション資料としてスライドを作成することにした．高速化班，ハードウェア班，広報班から最低一人ずつ配属されるようにポスター作成とスライド作成に人員を割り振り，他メンバーと話し合いながら発表内容を考えた．

スライドの作成には「Microsoft PowerPoint」を使用し，「Slack」を使用してバージョン管理を行った．また，模擬発表を何度か行い，修正点のあぶり出しを行うことで，わかりやすく見やすいスライドを目指した．

発表後の見学者からの意見を振り返ってみると，「白と黒をベースにまとめられたスライドが見やすかった」との主旨のコメントが多数寄せられていた. この点は来年度以降の後輩に

も伝えていきたいと思う.
また,「発表のスタートが遅かった」との意見がいくつか寄せられていたが, その点に関しては伝えたい内容の要点を絞る作業が甘かったことが考えられる. 理論面の説明は長くなってしまうのはスライド作成者が実際に自分の成果を周りに伝えたいと思っているあまりや無負えないことであるが, 本当に伝えるべき点はどこなのか, スライド班だけにとどまらず事前に全体で話し合いを行うのも一つの方法だなと思った.

（※文責: 池田晴輝）

発表原稿

今回はスライドの文章を詳しく作成したため，原稿はハードウェア班の分を除いて用意しなかった．ハードウェア班についてはメンバが 2 名しかいないうえ，各々が担当していたことが大きく異なるために説明が難しくなる可能性があると考えたため原稿を用意したが，発表当日は設営の関係上原稿が使用できずあまり意味はなかった．
実際に発表を行ってみると，分かりやすく説明しようとして話し言葉が目立ってしまったり，スライドを見過ぎてしまったりという傾向が見られた．特に本プロジェクトの場合，楕円曲線上での計算方法や，楕円曲線法の手順などで複雑な説明が必要になってしまう．発表の場で原稿を見過ぎたり，作成した原稿を予め一字一句覚えておいたりというのはいいこととは言えないが，説明の大まかな流れを原稿として作成し発表練習を繰り返し行っておくことが必要だと感じた．

（※文責: 青山和弘）

3.7.2 発表

最終発表会は前半に上田・小泉・鳴海・矢和田の 4 人，後半に青山・池田・染川の 3 人の布陣で臨んだ．事前に作成したスライドをメインとして，あらかじめ相談し決定しておいた分担に従い，交代しながら発表した．頂いた評価を後日集計したところ，「スライドが白と黒で統一されていて見やすかった」,「難しい数学の内容を具体例を用いてわかりやすく説明している点がよかった」など，前向きな意見が寄せられた一方で「説明が早口で聞き取りにくかった」,「発表中にスライドを見すぎている」といった発表に関するご指摘も多く頂いた．これらの意見を踏まえると資料や発表内容の事前準備は十分に行われていたものの，実際の発表技術面でのマイナスなご指摘が多かったと考えることができる．よって今後は，実際の発表練習を交代で発表を行い何項目か観点を決めてレビューを行う形式にすべきだと思った．これは来年度への引継ぎ事項として加える．

（※文責: 池田晴輝）

第 4 章　プロジェクト内のインターワーキング

- 青山和弘（プロジェクトリーダー、高速化班、ハードウェア班・グループリーダー）
  (1) 楕円曲線法の基礎について学習した
  (2) FORTRAN について学習した
  (3) 高速化班の他のメンバと協力し，FORTRAN で多倍長整数を扱う方法について模索した
  (4) 中間発表会に向けてスケジュール，進捗の管理を行った
  (5) 中間報告書の文責を決定し，自分も担当箇所について執筆した
  (6) ハードウェア班を結成するにあたって, FPGA, VHDL, Quartus など必要なことについての基礎を再学習した
  (7) ECM を行うための回路を設計・開発した
  (8) 成果発表会に向けてスケジュール, 進捗の管理を行った
  (9) 最終報告書の文責を決定し, 自分も担当箇所について執筆した
  (10) 鳴海, 池田と協力し成果発表会用のスライドを作成した
  (11) ハードウェア班分のスライドについて, 発表原稿を作成した
  (12) プロジェクト報告書を執筆した

（※文責: 青山和弘）

- 鳴海雄登（高速化班・グループリーダー、システム班グループリーダー）
  (1) 楕円曲線法の基礎について学習した
  (2) FORTRAN について高速化班のメンバにアドバイスをするとともに，自分も学習した
  (3) 高速化班の他のメンバと協力し，FORTRAN で多倍長整数を扱う方法について模索した
  (4) 染川と協力し，来年度に向けて FORTRAN の多倍長計算パッケージ FMLIB のマニュアルを日本語化した
  (5) 染川・池田と協力し，中間発表会に向けてスライド，及び発表原稿を作成した
  (6) 中間報告書の担当箇所について執筆した

（※文責: 鳴海雄登）

- 染川眞輝（高速化班、システム班、ハードウェア班）
  (1) ECM の基礎について学習した
  (2) FORTRAN について学習した
  (3) 高速化班の他のメンバと協力し，FORTRAN で多倍長整数を扱う方法について模索した
  (4) 鳴海と協力し，来年度に向けて FORTRAN の多倍長計算パッケージ FMLIB のマニュアルを日本語化した
  (5) 鳴海，池田と協力し，中間発表会に向けてスライド，及び発表原稿を作成した
  (6) 中間報告書の担当箇所について執筆した

(7) 鳴海と協力して FUNECM プログラムの構造解析を行った
(8) VHDL の基礎を学んだ
(9) 矢和田と協力し最終報告会に向けてポスターを制作した
(10) グループ報告書の担当箇所を執筆した

（※文責: 染川眞輝）

- 矢和田航平（高速化班、ハードウェア班）
  (1) 他のメンバーと共に楕円曲線法の基礎について勉強した
  (2) 高速化班の他のメンバーと共に FORTRAN の基礎について勉強した
  (3) ポスター班でポスター第一版の高速化班部分を担当した
  (4) ポスター第二版の基礎学習部分の打ち込みと英語化と高速化班の部分の手直しを担当した
  (5) ポスター第三版の基礎学習部分の手直しを担当した
  (6) 報告書の割り振られた部分を担当した
  (7) 中間発表後半での発表の一部分を担当した
  (8) FPGA のマニュアル類の日本語化作業を担当した
  (9) 43,000,000 までの素数ごとの 43,000,000 を超えない限界のべき乗のリストの作成を担当した
  (10) ハードウェア班のポスター作製を担当した
  (11) 最終発表前半の発表の一部分を担当した

（※文責: 矢和田航平）

- 池田晴輝（広報班・グループリーダー）
  (1) 楕円曲線法の基礎について学習した
  (2) HTML・CSS について学習した
  (3) 広報班の他のメンバと協力し，既存の Web サイトを構造化した
  (4) 広報班の他のメンバとともに，既存の Web サイトの問題点，改善点について議論した
  (5) 広報班の他のメンバと協力し，新 Web サイトの目的を決定した
  (6) 広報班の他のメンバと協力し，新 Web サイトのレイアウトを試案した
  (7) Adobe Illustrator を用いて，作成を担当する Web ページのプレビューを作成した
  (8) 広報班の他のメンバと協力し，新 Web サイトの構築を開始した
  (9) 鳴海・染川と協力し，中間発表会に向けてスライド，及び発表原稿を作成した
  (10) 中間報告書の担当箇所について執筆した
  (11) javascript の学習を行った
  (12) 新規ウェブサイトのラベリングを行った
  (13) 新規ウェブサイトのコーディングを行った
  (14) 各カテゴリーのマージン統一を行った
  (15) 各カテゴリーの文字サイズの統一を行った
  (16) 各カテゴリーのカラーの統一を行った
  (17) ページ遷移を確認した
  (18) サーバーを決定した
  (19) ファイルを統合した

(20) 新規ウェブサイトのアップロードを行った
(21) 新規ウェブサイトの今後の展望の検討を行った
(22) 鳴海・染川と協力し，最終発表会に向けてスライド，及び発表原稿を作成した
(23) 最終報告書の担当箇所について執筆した

（※文責: 池田晴輝）

- 上田隼人（広報班）
  (1) 楕円曲線法の基礎について学習した
  (2) HTML・CSS について学習した
  (3) 広報班の他のメンバと協力し，既存の Web サイトを構造化した
  (4) 広報班の他のメンバとともに，既存の Web サイトの問題点，改善点について議論した
  (5) 広報班の他のメンバと協力し，新 Web サイトの目的を決定した
  (6) 広報班の他のメンバと協力し，新 Web サイトのレイアウトを試案した
  (7) Adobe Illustrator を用いて，作成を担当する Web ページのプレビューを作成した
  (8) 広報班の他のメンバと協力し，新 Web サイトの構築を開始した
  (9) 小泉・矢和田と協力し，中間発表会に向けてポスターを作成した
  (10) 中間報告書の担当箇所について執筆した
  (11) 新規ロゴを修正した
  (12) 新 Web サイトのタグを整理をした
  (13) 新 Web サイトのコーディングをした
  (14) PHP の学習をした
  (15) MySQL の学習をした
  (16) ローカルサーバの構築をした
  (17) 各カテゴリのマージン統一をした
  (18) 各カテゴリの文字サイズの統一をした
  (19) 各カテゴリのカラーの統一をした
  (20) ページ遷移の確認をした
  (21) 新 Web サイトのファイルの統合をした
  (22) ファイルをアップロードするサーバの決定をした
  (23) サーバの挙動確認をした
  (24) 新 Web サイトのアップロードをした
  (25) 新 Web サイトの今後の展望の検討をした
  (26) 最終成果発表会に向けてポスターを作成した
  (27) 最終報告書の担当箇所について執筆した

（※文責: 上田隼人）

- 小泉建敬（広報班）
  (1) 楕円曲線法の基礎について学習した
  (2) HTML・CSS について広報班の他のメンバにアドバイスするとともに，自分も学習した
  (3) 広報班の他のメンバと協力し，既存の Web サイトを構造化した
  (4) 広報班の他のメンバとともに，既存の Web サイトの問題点，改善点について議論した
  (5) 広報班の他のメンバと協力し，新 Web サイトの目的を決定した

(6) 広報班の他のメンバと協力し，新 Web サイトのレイアウトを試案した
(7) Web サイトの新しいロゴを制作した
(8) Adobe Illustrator を用いて，作成を担当する Web ページのプレビューを作成した
(9) 広報班の他のメンバと協力し，新 Web サイトの構築を開始した
(10) 上田・矢和田と協力し，中間発表会に向けてポスターを作成した
(11) 中間報告書の担当箇所について執筆した
(12) javascript の学習
(13) 新規ウェブサイトのラベリング
(14) 新規ウェブサイトのコーディング
(15) PHP の学習
(16) サーバーの挙動確認
(17) 各カテゴリーのマージン統一
(18) 各カテゴリーの文字サイズの統一
(19) 各カテゴリーのカラーの統一
(20) ページ遷移の確認
(21) サーバーの決定
(22) 新規ロゴの修正
(23) MySQL の学習
(24) 成果発表会に向けてポスターを作成した。
(25) 最終報告書の担当箇所について執筆した
(26) ファイルの統合
(27) 新規ウェブサイトのアップロード
(28) 新規ウェブサイトの今後の展望の検討
(29) ローカルサーバーの構築

（※文責: 小泉建敬）

第 5 章　活動の結果

5.1　高速化班

高速化班の成果として，まず現状のプログラムの高速化方法に FORTRAN を用いるというアイデアを提案し，実際に高速化に繋がるであろう根拠を発見したことである．楕円曲線法の基礎を学習した我々は現状のプログラムの高速化案をいくつか提案した，その中で高速化の可能性があり実装が可能であるものとして FORTRAN で素因数分解プログラムを作成するという案を提案した．その後実際に FORTRAN が科学技術計算分野で高速な計算が可能であることを示した．

また，FORTRAN 単体では多倍長変数を用いることができなかったため，FORTRAN で多倍長変数を用いる方法を模索した．その結果 FMLib というライブラリを用いることによって扱えるようになることを発見したため，ライブラリの導入とマニュアルの日本語化を行った．

（※文責: 鳴海雄登）

5.2　広報班

既存の WEB サイトをもとに不要情報の削除，追加情報の決定，情報のカテゴリ分けを行った．その後，WEB ページの実装に向け，担当カテゴリの分担を行った．池田が WORKS，ECM，上田が MEMBER，NEWS，小泉が PURPOSE，PROJECT を担当することとなった．最終的に Adobe Illustrator を用いて各自，担当カテゴリのプレビューを完成させることに成功した．ここまでの作業を通して HTML・WEB デザインの知識を身に着けることが課題となった．

（※文責: 池田晴輝）

5.3　システム班

3.4.1 節の通り，昨年度までに制作された素因数分解プログラムを改良することにより，実際に運用が容易になった．まず，我々がユーザとしてこのシステムを利用する際に，従来ならば素因数分解を行うごとにワークステーションに SSH 接続を行い，プログラムの終了を予測して何度もアクセスする必要がなくなったことが最も運用面で改善された点といっても過言ではない．また，現在は合成数を 5 つずつ入力しているが，長期に渡ってワークステーションにアクセスできないことが事前にわかっていれば，そのアクセスできない期間からその間に素因数分解を行える回数を推定し，その回数分，素因数分解を行いたい合成数を入力することによって，常にプログラムを動作状態にすることができる．そして，合成数の待ち行列はフロントエンドソフトウェアを実行した PC 上に残っているため，WEB ページ上で直近に分解を試みた合成数と照合することによって現在の進捗具合をワークステーションにアクセスすることなく確認することができる．

これらの改良によって，今後の素因数分解は簡便に行えると考えられる．

（※文責: 鳴海雄登）

5.4 ハードウェア班

実際に設計した図 3.4 の回路の仕様について以下に示す.

128 ワード・レジスタファイル

STUDIO KAMADA で公開されている問題を解くためには約 190 桁の数値を扱える必要がある. 10 進数で 190 桁の数値は, 2 進数で表すと 640bit になる. そこで, 1 つのレジスタが扱えるデータを 1 ワード=128bit としたとき, 5 ワード, つまりレジスタが 5 つあれば素因数分解の対象を格納することが出来る. 他に計算過程で出た数値などを格納する分を考慮し, 128bit のレジスタを 128 個内蔵した「128 ワード・レジスタファイル」が必要となった. 入力としてクロック信号, レジスタのアドレス, 書き込み制御信号, レジスタに書き込む 128bit のデータを受け取り, 入力されたアドレスに対応したレジスタに格納されている 128bit のデータを出力する. また，出力のうち一つは状態確認のために外部出力もされる.

乗算器

入力された 128bit のデータ 2 つを乗算した結果である 256bit のデータを出力する．なお，この出力は前後半の 128bit ずつに分解され，それぞれがマルチプレクサを通してレジスタファイルにフィードバックされる.

加算器

基本は入力された 128bit のデータ 2 つを加算した結果である 129bit のデータを出力する. この出力のうち先頭の 1bit がキャリーアウトとしてキャリーレジスタに送られ，後半の 128bit がマルチプレクサを通してレジスタファイルにフィードバックされる. また, 減算や桁上がりの処理などを行うことができる.

キャリーレジスタ

加算器の出力から受け取ったキャリーアウトを格納するレジスタである. これに基づき加算器で桁上がり処理を行ったりする.「直前の桁上がりの状態」が必要になるときがあるので, キャリーレジスタは 2 つ用意してある.

選択用マルチプレクサ

C 言語などでの if 文のような動作をするためのマルチプレクサである.2 つのレジスタファイル出力のうちどちらかを出力する.

また, 制御信号を組み合わせることによって以下の命令が実行できる.

- 加算：add0, addc, add_carry
- 減算：sub1, subc, subc_setc1
- 乗算：mul
- 選択：select_c, select_c1
- 入出力：load, store

「加算」「減算」はレジスタファイルからの加算器入力とキャリーレジスタに格納されている値から演算を行う命令である.「乗算」はレジスタファイルからの乗算器入力を利用して演算を行う命令である.「選択」は前述の通り選択用マルチプレクサを用いてレジスタファイルからのマルチプレクサ入力のうちどちらかを選び出力させる命令である.「入出力」はレジスタファイルに書き込む値を外部から入力するか，状態確認のためにレジスタファイルの出力のうち一方を外部に出力す

る命令である．

これらの命令を用いて，『3.7.7　楕円曲線法のハードウェア実装』で述べた，楕円曲線法における点 P のスカラー倍を行う操作を実行する．

（※文責: 青山和弘）

第 6 章　まとめ

6.1　前期活動の成果

6.1.1　高速化班

高速化班の成果として，まず C 言語と FORTRAN での処理速度比較が挙げられる．昨年度のプログラムから高速化を行うにあたり，より高速である言語である FORTRAN を利用する運びとなった．実際に FORTARN が C 言語より高速であるか比較を行い，FORTRAN の方が C 言語よりも高速であるとい結果を得ることができた．

速度比較の結果から FORTRAN でプログラムを記述することで高速化を行えると判断したため，ECM の基礎学習と並行して FORTRAN の基礎学習を行った．加えて，FORTRAN で多倍長整数の計算のために，FMLib の導入を行った．

また，昨年度までに制作された素因数分解プログラムを用いて STUDIO KAMADA に公開されている未解決の長大桁整数の素因数分解を行った．前期では 1 件の未解決の長大桁整数の解決に至った．

（※文責: 染川眞輝）

6.1.2　広報班

広報班の前期活動の成果は，既存のウェブサイトを構造化し，無駄な部分や追加した方が良いカテゴリを話し合った．それに基づいて新しく作成するウェブサイトのカテゴリを決定した．次にウェブサイト全体のレイアウトを考案し，ページのプレビューを Adobe Illustrator を用いて作成した．同時に，ロゴの制作も行った．ページのプレビューが完成した後，後期に向けて HTML と CSS の学習を開始した．

（※文責: 上田隼人）

6.2　後期活動の成果

6.2.1　広報班

前期は Web サイトの目的を定めて，大枠を作成するなど構想を練る作業が中心であったのに対し，後期は実際のコーディングやサーバーへのアップロードなど実務が中心となった．これを言い換えると，後期の活動は前期に比べると成果が目に見える形にあるものが多かった．しかし，前期とは打って変わって，連日個人ワークが続いた．そこで，基本的にはメンバー 3 人で足並みをそろえて同時に作業を行うこととした．その結果，3 人でお互いに協力し合い，周りからの刺激を受けつつ，毎週着実に進捗することができた．そして後期活動期間内にサーバーを通じて新 Web サイトをアップロードすることに成功した．残された改善の余地は複数あるものの，当初の目的であった老若男女幅広い年代の方々が親しみやすく，操作しやすい Web サイトの作成は，概ね完了する

ことができたと考えている．残された改善点の詳細は，6.4.1(今後の展望の広報班分野) に示す．

（※文責: 池田晴輝）

6.2.2　システム班

システム班は後期から活動している班であるため 5.3 節システム班の活動の結果に準ずるが，既存のプログラムに機能を追加し，システム化を行うことで円滑なプログラム運用を可能とした．

3.4.1 節のように，素因数分解をプロジェクト始動から継続して行っているが，システム改良後は改良したシステムを運用して素因数分解を試みている．その結果，今年度は 3 件の素因数分解に成功し，それらは「STUDIO KAMADA」へ投稿し掲載されている．いずれもニアレプディジット関連の数の素因数分解過程の最大素因数を見つけている．我々の制作した素因数分解プログラムを我々の環境で運用している状態では，概ね 140 桁から 160 桁程度の合成数に対して，推定 50 桁程度の素因数を発見するように素因数分解を行うことが最も効率的に素因数分解を行えるということが推測された．この推測はハードウェア班で制作した FPGA による ECM を用いた素因数分解回路の設計にも参考値として採用された．

（※文責: 鳴海雄登）

6.2.3　ハードウェア班

班員が 2 名だったので，作業の分担を行った．

1 人は本年度購入した FPGA, 及び対応している NiosII の英版マニュアルを日本語に訳した他に, 楕円曲線法の途中で点 P にスカラー倍する十分大きな値 B を無駄な演算を行うことなく求めるための要素リストを作成するプログラムを作成した．このプログラムが PARI/GP 上で正しく動作することを確認している．

もう 1 人は FPGA の特性などの知識や VHDL の書き方について基礎学習を行ったあと，教員と相談を行いながら楕円曲線法を行うための回路の設計を行った．設計方針と大まかな回路図が定まると，Quartus 上で機能の記述，論理ブロック (コンポーネント) の配線を行った．結果として，論理的には正しくスカラー倍算を実行できる回路を開発した．

（※文責: 青山和弘）

6.3　後期の展望

6.3.1　高速化班

後期の展望として FORTRAN を用いての ECM プログラムの STAGE 1 の作成や，FMLIB の説明書の日本語訳などがあげられる．

（※文責: 矢和田航平）

6.3.2 広報班

前期に広報班では，ウェブサイトを作成する準備を着々と進めた．そのため，後期では前期の活動内容をどのように実行していくかが課題となった．具体的には，ウェブサイトのプレビューを再現するのにどのような論理構造で HTML，CSS を記述するかなどである．また，実際に作成したウェブサイトが自分たちの目的と一致しているかを考える必要もあった．サーバーを経由してウェブサイトを開設することまでが一つのタスクであるため，どのサーバーを使用するかを考える必要もある．

（※文責: 小泉建敬）

6.4 今後の展望

6.4.1 広報班

今回の Web サイト作成の目的は ECM を世の中に広めることを達成するため，閲覧対象者である高校生に向け，ECM に親しみを持ってもらうための ECM 講座を新たなカテゴリとして作成したい．義務教育の範囲の学習で多くの方々が身に着けているであろう素因数分解の知識をトリガーとして，ECM に関する基礎学習ができるような内容で，さらに楽しいアニメーションなども付加できれば，なお良いと考えている．ゆくゆくはアプリケーションの作成も視野に入れている．

また，もう一つの重要な作業としてスマホ用サイトの構築を考えている．今や，年齢に関わらず電子端末の保有率が上がっているため，スマートフォンでのサイトアクセスに対応することは不可欠である．来年度以降 UI/UX に関してさらに詳しく学習し，スマホ用サイトを作成することで，さらに老若男女幅広い年代の方々が親しみやすく，操作しやすい Web サイトになると考えている．

（※文責: 池田晴輝）

6.4.2 システム班

3.4.1 節に記述した，既存のプログラムを用いた素因数分解は今後も継続していく目論見だ．

3.4.2 節に記述した，既存のプログラムの改良は，まだいくつか改善点が考えられる．1 つは現状のソフトウェアでは再利用性に乏しいという点だ．これは接続先情報や，ディレクトリの構造などの設定がコード上に記されているため，我々が現状維持的に利用する分には問題はないが，何らかの事情で現状から変更点が発生した場合にコードから変更を行う必要がある．そのため設定ファイル等で項目の変更を行うことができれば保守運用上のユーザビリティの向上につながると考える．もう 1 つの改善点は，素因数分解の統計自動化だ．現状では過去に行った素因数分解の統計を手動で行い，6.2.2 項で述べたように，我々のソフトウェアで素因数分解を行う上でより最適な条件を割り出している．また，過去の素因数分解の情報を元に，以前に分解を試みた合成数以外の合成数を分解することとしている．しかし，素因数分解を試みるたびにこういった素因数分解を行ったデータは累積していくため，これを自動的に統計することができれば，素因数分解を行うという作業に割く時間を減らすことができ，今後のプロジェクト活動にて別の作業をする機会を増やすことにつながると考える．この統計についていくつか手法を思案しているが，その中で最も現実的で

あると考えるのは，ワークステーション上にデータベースサーバを構築し，素因数分解を行う度にデータベースの更新を行うというものだ．しかしこの手法の問題点は，現状ワークステーションでは素因数分解プログラムにほぼすべてのリソースを割いているが，データベースサーバを構築することによって，プログラムの実行時間に影響を及ぼすのではないか，またそのデータベースを学内ネットワークからのみ接続できるといった点で問題が残ってしまう．これらいくつかの問題を解決し，FUN-ECM システムとも呼べる今年度の成果物を完成させるということが今後の展望である．

（※文責: 鳴海雄登）

6.4.3 ハードウェア班

設計した図 3.4 の回路をコンパイルしたが, ターゲットボードの規模とあっていなかったためかエラー (Out of memory) となった. そこで 128 ワード・レジスタファイル内のレジスタ 1 つあたりに格納できるデータを 4bit までにした小規模な回路を作成しコンパイルしたところ問題がなく, また論理シミュレーションも正しいと思われる動作をしているのを確認した. よって, 128bit 版の回路についてターゲットボードを変更し再度コンパイルを行い, 動作確認を行うのが課題である.

また, 制御信号を組み合わせることで命令セットを実行できると述べたが, これは現在手動で制御信号を設定する必要があるため利便性に欠けている. そこで, 制御信号のパターンを格納しておく命令メモリを作成することで扱いやすくなると考えられるので, 来年度以降の課題とする.

（※文責: 青山和弘）

参考文献

[1] ECMNET. https://members.loria.fr/PZimmermann/ecmnet/, (最終アクセス 2018 年 7 月 5 日)

[2] STUDIO KAMADA. http://stdkmd.com/, （最終アクセス 2018 年 7 月 5 日）

[3] LL(Lightweight Language) の実行速度を比較してみた - hykt の日記. http://d.hatena.ne.jp/hykt/20120930, (最終アクセス 2018 年 7 月 11 日)

[4] 伊豆哲也 『楕円曲線暗号入門』(2013)

[5] ザイリンクス FPGA 講座 - Xilinx https://japan.xilinx.com/japan/fpga-koza.html （最終アクセス 2019 年 1 月 11 日）