平成14年度 卒業論文

ハードウェア記述言語を用いた行列演算専用回路設計

電気通信大学 電気通信学部 電子工学科

マイクロエレクトロニクス講座

9922014　小林 直樹

指導教官 齋藤 理一郎 助教授

提出日 平成15年 2月 6日

概要

物性の計算には，多くの場合膨大な計算量を要する．例えば，分子軌道計算には行列の固有値・固有ベクトルを求める計算を必要とするが，その計算量は行列の次数を N とすると $O(N^3)$ となり，$N \sim 10^4$ に及ぶ大規模な計算を実用時間内に行うことは非常に困難となっている．物性計算においてその計算時間を短縮することが大きな課題である．

計算時間短縮のためにはスーパーコンピュータをはじめとする並列コンピュータを用いた計算の並列化が考えられるが，演算に並列化できない部分があるとどんなにプロセッサ数を増やしても並列処理の効果は期待できない，プロセッサ間の通信が多い場合，そこで時間がかかる等の問題があるため，計算時間短縮にはならない場合が多い．

そこで，近年，汎用並列コンピュータにかわるアーキテクチャとして，ある問題に特化したコンピュータである専用コンピュータを用いる手法が $O(N)$ 法に代表される新アルゴリズムの開発とともに一般的になってきた．本研究では行列計算を対象とし，専用プロセッサを設計することによって計算時間短縮を試みる．

また従来，対象のアーキテクチャの性能評価を行うには大別して，ソフトウェアでエミュレーションを行う方法，あるいは実際に対象のアーキテクチャをハードウェアとして製作する等の手法が取られているが，前者では評価に膨大な時間を要する，後者では製作に多額の費用がかかる，開発から実際に評価を行うまでに相当な期間を要する，ハードウェアを変更して複数のアーキテクチャを試行することが困難である等の問題がある．

これらの問題を解決するシステムとして，近年，デジタル回路設計手法として一般的になってきたハードウェア記述言語 HDL(Hardware Description Language) の 1 つである VHDL を用いてプロセッサの機能設計を行い，これをプログラマブルデバイスである FPGA(Field Programmable Gate Array) を用いて実装評価するシステムを構築した．実際に，10 万ゲート相当の FPGA 2 個を用いた基盤 [3] において，LU 分解回路を実装した．プロセッサは 3 つの単精度浮動小数点数演算器を含み，3×3 行列の LU 分解を計算することができた．

目 次

第1章　序論

1.1　本研究の背景

量子力学をはじめとする科学計算においては，膨大な計算量を要する．物性計算を例にとると，行列の固有値，固有ベクトルを求める必要がある．その計算量は行列の次数を N とすると $O(N^3)$，つまり次数の 3 乗に比例し，科学計算で使われる 1000 次以上の大規模な計算においては PC では数日以上かかり，コンピュータの使用効率を下げてしまう．

この計算時間短縮の手法として，並列コンピュータを用いた計算の並列化や新しい行列計算アルゴリズムなどがあげられる．しかし，並列化の問題点としては，並列化できない演算部分があり，コンピュータ数を増やしても，その部分は計算時間の短縮はできない．また，コンピュータ間での通信にも時間がかかるため，その部分も短縮はできない．

また，新しい行列計算アルゴリズムとして，$O(N)$ 法などがあげられるが，このようなアルゴリズムでは，計算時間の短縮が期待できるが，厳密解が得られないという問題点がある．

そこで本研究では，計算時間短縮の手法として，ハードウェアを用いた専用計算機を用いる方法と取り上げる．この方法では，計算を構成するアルゴリズムをハードウェア的に動作させるために，デジタル集積回路の機能設計を行なう．そして，その機能をプロセッサに搭載することで，専用計算機として計算を行なう．

この専用計算機のプロセッサを設計し，搭載するところにおいて，計算アルゴリズムの複雑さと計算量の多さにより，機能設計に必要なデジタル回路のゲート数は非常に膨大なものとなる．そのため，この回路の設計手法として，ハードウェア記述言語である HDL(Hardware Description Laguage) を，近年用いるようになった．そして，何度でも書き込み可能なゲート素子（プログラマブルデバイス）である FPGA(Field Programmable Gate Array) を用いることで，集積回路の評価が比較的容易なものとなった．

1.2　今までの研究成果と本年度の課題

本研究は当初，(株) 画像技研 との共同研究として，科学計算を高速に行うために，専用の計算機を開発しようという目的で，1996 年度から始まった研究である．ここでは，前年度までの本研究の成果について述べる．

まず，'96 年度は，本研究室の中島 [1] と八木 [2] が，行列の固有値および固有ベクトルを求めるためのアルゴリズムであるハウスホルダ法をこの計算機に搭載するアルゴリズムとして採用した．このアルゴリズムを採用した理由としては，本研究室で行われている量子力学上の分子起動計算では行列計算を多用している．この計算において固有値，固有ベクトルを求めるため

に，多くの時間を要しているため，この計算時間を短縮するための手法として，ハウスホルダ法を採用した．さらに，ハードウェア上での三重対角化から逆反復法までの計算過程のモデルを提案した．

そして'97 年度は，松尾 [3] とグェン [4] が，計算アルゴリズムを実際に動作させるためのハードウェアを作製するための設計方法を決めた．研究室で設計を行うために，設計の容易さと開発コストを考慮しなければならない．そこで，近年ディジタル回路の設計手法として一般的になってきたハードウェア記述言語 HDL(Hardware Description Language) を採用した．そして，この言語により設計した機能をハードウェアとして動作させるために，プログラマブルデバイスである FPGA(Field Programmable Gate Array) を採用した．

本研究室では，これらを用いた開発環境を得るために，(株) インターリンク社より PeakFPGA を HDL 設計ツールとして購入し，(株) 日本アルテラ社のユニバーシティプログラムに参加し，FPGA の配置，配線ツールである Max+plusII の無償提供を受けた．そして，(株) アルティマから FLEX10-K シリーズのひとつである EPF10K100GC503-4 という FPGA を 2 個購入した．

この FPGA を使用した専用計算機を構築するためには，FPGA を搭載するための基板が必要である．そこで，松尾はこの FPGA2 個，かつ SRAM(Static Random Access Memory)，DRAM (Dynamic Random Access Memory) といったメモリを実装した基板を設計，製作した [3]．また，PC とこの基板間でのデータ通信が可能なインターフェースボードを製作した．そして，計算アルゴリズムを VHDL(VHSIC Hardware Description Language) によって記述し，シミュレーションによって，この計算アルゴリズムをハードウェアレベルで動作させるためのモデルを築いた．

'98 年度は，山岡 [5] と沼 [6] により，先に製作された基板を利用してハウスホルダ法のアルゴリズムを使い，実際に行列の固有値と固有ベクトルの計算をハードウェア上で動作させた．まず，基板と PC との間でデータ通信を行うための VHDL を設計し，PC と FPGA 間の通信を行った．そして，SRAM コントローラを設計し，計算の対象となるデータを SRAM に記憶させることができた．

次に，固有値計算を行うための準備として，積和器の設計を行った．この積和器は行列の計算を行う上で非常に重要な要素となっている．最後に，ハウスホルダ法の三重対角化からの逆反復法など 4 つのアルゴリズムを VHDL で設計し，実際に行列の固有値計算がハードウェア上で動作が可能となった．ただし，この動作には SRAM を用いているので，メモリ容量に制限がある．

'99 年度は沼 [6] により，新たに DRAM を用いて DRAM を制御するサブプログラムを作製して行列の固有値と固有ベクトルを求めるプログラムを作製した．DRAM は SRAM に比べてメモリ容量が 8 倍と大容量であるが，SRAM にアクセスするより，3 倍ほどのクロックを必要とし，リフレッシュと呼ばれる電荷を補充する動作が必要となる．

2000 年度は清水 [7] により，PC との通信に ISA バスを使用した FPGA 評価基板 [3] を用いての SRAM をデータメモリに指定しての高周波クロック (10MHz) での行列演算装置を開発した．クロック周波数 10MHz での行列の乗算器，高次数での乗算器が可能となった．

1.3 目的

本研究の目的はFPGA 評価基板 [3] を用いてLU 分解を計算する専用プロセッサの設計である．そのために，今まで作製された各種演算回路を実際に動作検証または改良し，それらをコンポーネントとして利用し演算回路設計を行う．また，作製したLU 分解回路を用いて連立一次方程式を解くことがこの研究の最終目的である．

1.4 本論文の構成

まず，第 2 章においてハードウェア記述言語 (HDL) を用いたLSI 設計手順を示すとともに，設計した計算システムを実装し評価するためのFPGA 搭載計算システム評価ボードについて説明する．

第 3 章ではLU 分解法について説明し，C 言語での計算方法を示し，VHDL により設計したLU 分解プロセッサについて説明する．

第 4 章では本論文の結論を述べる．

付録にはVHDL で作製した数値計算プログラム及びC 言語で作製したLU 分解計算実行用のプログラムを示し，研究に必要なソフトウェアの使用方法を説明する．

1.5 用語説明

以下において，本研究を行う上での必要最低限の用語を説明する．

- VHDL
  VHDL(VHSIC Hardware Description Language) はハードウェア記述言語のひとつで，これと論理合成ツール (MAX+PlusII など) を用いてハイレベル設計手法による論理回路の設計が，現在主流となってきている．
  VHDL の特徴は，このデジタル回路設計，シミュレーション，合成のすべてを実行するための幅広い構成を持っていること，そして，シミュレーションによる動作検証がしやすく，設計の変更に時間がかからないこと，言語記述がそれほど複雑ではないので，習得が容易である事が言える．
  また，HDL にはVerilog-HDL というのがあり，こちらはC 言語に近い構文の記述ができることや，各記述ブロック毎に各信号の入出力の指定が必要という特徴を持っている．

- FPGA
  FPGA(Field Programmable Gate Array) とは，PLD(Programmable Logic Device) の一種で，何度でもデジタル回路機能を書込みできるゲート素子である．本研究で使用するFPGA はアルテラ社製のFLEX10k100 である．これはSRAM をベースに作られているので電源を入れるたびにダウンロードする必要がある．

第2章　設計方法と使用装置

2.1　設計方法

本研究において VHDL による回路設計の流れ図を図 2.1 に示す．まず PeakFPGA というVHDL エディタ で回路機能 を記述し，HDL ファイル (*.vhd) を作製する．もし，VHDL 構文の記述がおかしければ，コンパイル中にエラーメッセージが出る．その場合は正しい文章に書き直し，再度コンパイルする．そして，この機能検証を実行するために，テストベンチをおこなう．これは，デバイスの入力ポートにデータを入力する動作を VHDL で記述したものをシミュレーションでデバイス内の各信号の動作を検証することである．このシミュレーションで信号の動作が正しくなければ，VHDL ファイルを書き直しシミュレーションに戻り，信号の動作が正しくなるまで書き直す．

本研究の場合には，FPGA でのピンの配置も VHDL ソース作製の時点で指定する必要がある．この場合には，entity の部分で port 文で入出力ピンを指定し，attribute 文でピン番号を指定する必要がある．

そして，その回路の動作が正しければ，PeakFPGA で論理合成を行なう．論理合成とは HDL によって記述された回路機能を AND や OR などの論理回路の形に変換することである．これにより HDL ファイルは EDIF(Electronic Design Interchange Format) ファイル (*.edf) に変換される．この EDIF ファイルはデジタル回路をテキストファイルで表したファイルで，回路を実際のデバイスに実装するための情報が含まれている．この EDIF ファイルを目的のデバイスにコンフィグレーションする為には，そのデバイスに合う形に変換しなければならない．そこで MAX+plusII というソフトウェアが必要になってくる．この MAX+plusII でデバイスを指定して再びコンパイルする．すると評価基板上にある FPGA，FLEX10K に乗る TTF(True Type Format) ファイル (*.ttf) が作製される．この TTF ファイルを松尾 [3] が作製した C プログラムで FPGA にコンフィグレーションする．コンフィグレーション後に実際に動作するか検証する必要がある．その為に，C++Builder を使って評価基板を制御して検証をする．作製の流れは図 2.1 に示す．

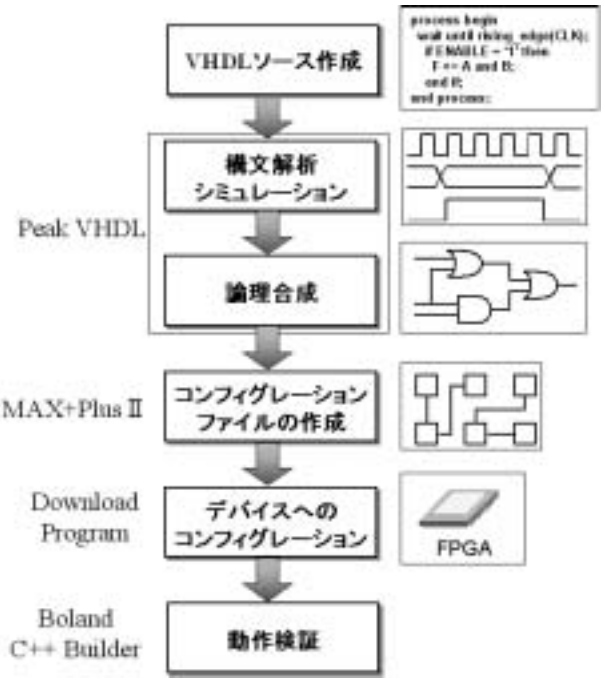

図 2.1: 本研究におけるデジタル回路設計の流れ[0]

2.2 ハードウェア

2.2.1 評価基板

本研究で使う評価基板は松尾 [3] によって作られた (図 2.2，図 2.3)．この評価基板の構成は，Field Progrmmable Gate Array (以降 FPGA) は FLEX10K が 2 つ，DRAM が各 FPGA に 2 個．SRAM が各 FPGA に 8 個．インターフェイスは 50pin フラットケーブル．評価基板上の FPGA は SRAM なので電源が供給されないと，データが消えてしまう．よって毎回コンフィグレーションをする必要がある．また評価基板上には DRAM(Dynamic Random Access Memory) が各 FPGA の両サイドにあり，片方のメモリーの容量は SRAM が 521KByte，DRAM が 16MByte ある．PC から送る単精度浮動小数点型は 32bit(4Byte) なので SRAM では 131072 個のデータ (512KByte) 格納出来る．

[0] u02koba/eps/flow1.ps

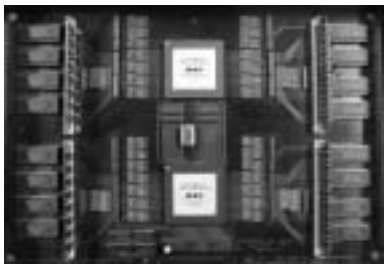

図 2.2: FPGA 評価基板 [1]

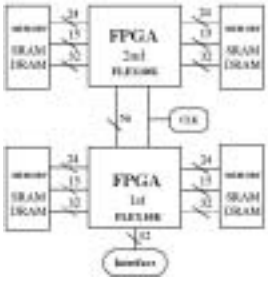

図 2.3: 評価基板ブロック図 [2]

表 2.1: SRAM と DRAM のスペック

	SRAM	DRAM
アクセス速度	数 ns 以下	60ns
アクセス手続き	2 回	6 回
データ容量 (片側)	512KByte	16MByte
記憶出来るデータ (4Byte) 数	131072 個	4194304 個
リフレッシュ	不要	必要

ここで SRAM のアクセス方法について簡単に述べる．まず SRAM にアクセスする為に必要な信号は $\overline{SCS}$，$\overline{SWE}$，$\overline{SOE}$ がある．書込む場合は SRAM のアドレスをセットした状態で $\overline{SCS}$ を立ち下げる．
これで SRAM のアドレスが指定されて，その後 (1 クロック後)$\overline{SWE}$ を立ち下げると指定したアドレスにデータが書込まれる．読み込む場合は書込みと同様に $\overline{SCS}$ を立ち下げてアドレスを指定する．それと同時に $\overline{SOE}$ を立ち下げることで SRAM からデータを読み込む事ができる．

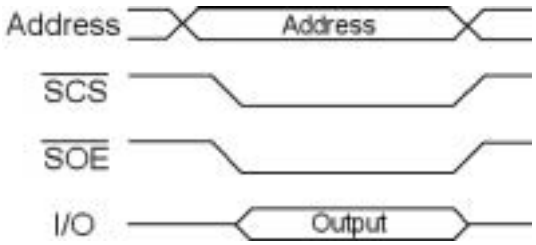

図 2.4: SRAM からの読込み [3]

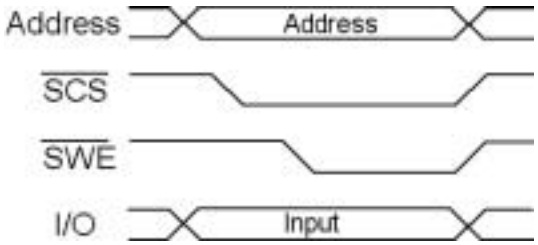

図 2.5: SRAM への書込み [4]

- ハイインピーダンス

[1] u02koba/eps/m-bord.ps
[2] u02koba/eps/m-bord2.ps
[3] u02koba/eps/sram_read.ps
[4] u02koba/eps/sram_write.ps

SRAM とのデータバス (信号名:DATA) は in:out 両方兼用である．SRAM からデータが送られる時，このデータバスをハイインピーダンスに設定しなければならない．PeakFPGA ではハイインピーダンスを ‘Z’ で表現している．また PC と評価基板との通信も A ポートが in:out 両方兼用なのでここも PC から評価基板にデータを送る時，A ポートをハイインピーダンスに設定する．

例 (VHDL 記述)

```
process begin
if OE = '1' then
DATA = "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";
end if;
end process
```

評価基板の詳しい内容は松尾 [3] の卒業論文に記述されている．

2.2.2 PC-評価基板間の通信インタフェイス

評価基板のコンフィグレーションと評価基板の制御を行う為に PC と評価基板の通信が必要になる．そこで松尾 [3] が ISA バス専用の通信インタフェイスを作製した．通信インタフェイスの詳しい内容は松尾 [3] の卒業論文に記述されている．

2.3 ソフトウェア

2.3.1 PeakFPGA

VHDL を記述するにあたり，シミュレーション，論理合成，シミュレーションが出来るエディタが必要になる．そこでインターリンク社より PeakFPGA を購入した．現在のバージョンは PeakFPGA5.20c である．

2.3.2 MAX+plusII

VHDL エディタにより論理合成して出来た EDF ファイルを目標の FPGA にコンフィグレーションするためにコンパイルする必要がある．そこで Altera 社の MAX+plusII を使って行うこととする．現在のバージョンは MAX+PlusII 10.0 である．

2.3.3 C++Builder

評価基板を制御する為に Boland 社の C++Builder を使って制御する．制御とは評価基板上の FPGA に信号を出力，PC と FPGA 間の伝送路の制御がある．詳しい使い方は付録参照.

2.3.4 pcAnywhere を使った遠隔操作でのコンフィグレーション

PC と評価基板との通信には松尾 [3] が作製した ISA バス専用のインターフェイスを用いているが，最近の PC には ISA バスが搭載されている PC は皆無に等しい．そこで ISA バス搭載 PC を評価基板制御 PC として独立させ，その PC を外部の PC から通信で制御させることとする．詳しい使い方は付録参照.

第3章　LU分解

3.1　LU分解について

3.1.1　LU分解とは

LU分解とは係数行列をL行列(下三角行列)とU行列(上三角行列)に分けて計算する方法(3.1)であり，連立一次方程式を解く手段の1つである．LU分解にかかる計算量は$O(n^3)$で，LU行列から連立一次方程式の解を求める計算量は$O(n^2)$である．連立一次方程式を解く最も基本的な方法はガウスの消去法であり，その計算量は$O(n^3)$である．LU分解がガウスの消去法より優れている点は，L行列とU行列の分解が係数行列のみに依存しているという点である．係数行列が同一ならば，どんな連立一次方程式を解く場合においてもLU分解法までは共通して計算でき，同一の係数行列を使って何度も連立一次方程式を解く場合，1度LU分解をしておけば後は$O(n^2)$の計算量で解を導くことがでる．
同一の係数行列を使って連立一次方程式をとくことは，逆行列を求める場合や熱伝導方程式を差分法で解く場合など珍しいことではない．

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \tag{3.1}$$

3.1.2　LU分解の計算方法

LU分解法のやり方には，外積形式ガウス法，内積形式ガウス法，クラウド法があり，本論文でFPGAを用いてLU分解を行う方法は外積ガウス法を使用する．
外積ガウス法では行列Aを順次,

$$\begin{aligned} A &= L_1 A_1 U_1 \\ &= L_1 L_2 A_2 U_2 U_1 \\ &\quad \cdots \\ &= L_1 L_2 \cdots L_n A_n U_n \cdots U_2 U_1 \\ &= LDU,\ L = L_1 \cdots L_n,\ D = A_n,\ L = U_n \cdots U_1 \end{aligned} \tag{3.2}$$

と分解していく．ここでDは対角行列であり，各L_kは下三角行列であり，各U_kは上三角行列でつぎの形をもつ．

$$L_k = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \mathbf{0} \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & l_{kk+1} & 1 & & \\ \mathbf{0} & & & \vdots & & \ddots & \\ & & & l_{nk} & & & 1 \end{bmatrix}, \quad U_k = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \mathbf{0} \\ & & 1 & & & & \\ & & & 1 & u_{kk+1} & \cdots & u_{kn} \\ & & & & 1 & & \\ \mathbf{0} & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \tag{3.3}$$

つまり，L_kとU_kは第k列ないし第k行だけが単位行列と異なっている．ここで，積$L_1 \cdots L_k$と積$U_k \cdots U_1$は

$$\begin{aligned} L_1 L_2 \cdots L_k &= L_1 + L_2 + \cdots + L_k - (k-1)I \\ U_k \cdots U_2 U_1 &= U_k + \cdots + U_2 + U_1 - (k-1)I \end{aligned} \tag{3.4}$$

という性質をもっている．外積形式のアルゴリズムを示すと図3.1になる．

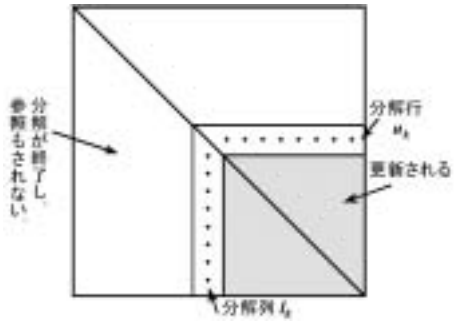

図 3.1: 外積形式ガウス法[5]

外積形式のアルゴリズムを数式表現すると次のようになる．まず,

$$\begin{aligned} A = A_0 &= \begin{bmatrix} a_{11} & \boldsymbol{u}^T \\ \boldsymbol{l_1} & A^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \\ \boldsymbol{l_1}/l_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} l_{11} & \mathbf{0} \\ \mathbf{0} & B_1 \end{bmatrix} \begin{bmatrix} 1 & \boldsymbol{u_1}^T/l_{11} \\ \mathbf{0} & I_{n-1} \end{bmatrix} \\ &= \begin{bmatrix} l_{11} & \boldsymbol{u_1}^T \\ \boldsymbol{l_1} & \boldsymbol{l_1 u_1}^T/l_{11} + B_1 \end{bmatrix} = L_1 A_1 U_1 \end{aligned} \tag{3.5}$$

これより，l_{11}，$B_1 = A^{(1)} - \boldsymbol{l_1 u_1}^T/l_{11}$である．つぎに$A_1$の分解をする．

[5] u02koba/eps/lu2.ps

$$A_1 = \begin{bmatrix} l_{11} & \mathbf{0} \\ \mathbf{0} & B_1 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \mathbf{0} \\ 0 & l_{22} & \boldsymbol{u_2}^T \\ \mathbf{0} & l_2 & A^{(2)} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & & \mathbf{0} \\ 0 & 1 & \\ \mathbf{0} & \boldsymbol{l_2}/l_{22} & I_{n-2} \end{bmatrix} \begin{bmatrix} l_{11} & & \\ & l_{22} & \\ & & B_2 \end{bmatrix} \begin{bmatrix} 1 & o & \mathbf{0} \\ & 1 & \boldsymbol{u_2}^T/l_{22} \\ \mathbf{0} & & I_{n-2} \end{bmatrix} = L_2 A_2 U_2 \tag{3.6}$$

これより，$l_{22} = b_{22}^{(1)} = B_1$ の第一対角要素であり，$B_2 = A^{(2)} - \boldsymbol{l_2}\boldsymbol{u_2}^T/l_{22}$．このまま三項計算するのが外積形式ガウス法である．

設計する集積回路で，行列 A を下三角行列 L と上三角行列 U の積に分解 (LU 分解) する計算方法は，

$$A = LU, L = (l_{ij}), U = (u_{ij}) \tag{3.7}$$

とする．ここで，対角要素 l_{ii} または u_{ii} を 1 にする．本論文では $l_{ii} = 1$ とする．この式を各要素ごとに展開すると，

$$\begin{aligned} a_{ij} = \sum_{k=1}^{n} l_{ik}u_{kj} = \sum_{k=1}^{i} l_{ik}u_{kj} = \sum_{k=1}^{i-1} l_{ik}u_{kj} + l_{ii}u_{ij} \qquad (i < j) \\ = \sum_{k=1}^{j} l_{ik}u_{kj} = \sum_{k=1}^{j-1} l_{ik}u_{kj} + l_{ij} \qquad (i \geqq j) \end{aligned} \tag{3.8}$$

つまり

$$\begin{cases} l_{ij} = a_{ij} - \displaystyle\sum_{k=1}^{j-1} l_{ik}u_{kj} & (i \geqq j) \\ u_{ij} = \left(a_{ij} - \displaystyle\sum_{k=1}^{i-1} l_{ik}u_{kj} \right) / l_{ii} & (i < j) \end{cases} \tag{3.9}$$

となる．

分解の手順としては，$j = 1$ とおいて $i = 1, \cdots, n$ に対し l_{i1} を求め，次に $i = 1$ とおいて，$j = 2, \cdots, n$ に対し u_{1j} を求める．つまり，図 3.2 のように分解していく．

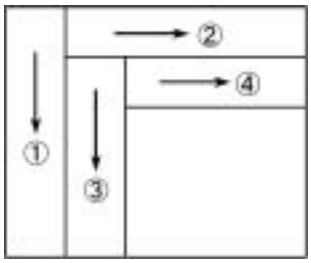

図 3.2: LU 分解の計算順序 [6]

3.2 C 言語での LU 分解

3.2.1 目的

集積回路上で LU 分解回路を実現するにあたり，FPGA 内に行列を書き込む配列レジスタを用意する．そこで，配列制御や計算方法などの確認のため Borland C++Builder5 を用いて LU 分解の計算を行う．

3.2.2 プログラム作製

LU 分解を外積形式ガウス法で計算する場合, 次のような三重ループが必要となる．

例 (三重のループ計算)

```
for(i = 0; i < n; i++){
    for(j = i +1 ; j < n; j++){
        a[j][i] /= a[i][i];
        for(k = i +1; k < n; k++){
            a[j][k] -= a[i][k] *a[j][i];
        }
    }
}
```

この三重ループプログラムを利用し，LU 分解プログラムを作製した (付録 D.2)．

3.2.3 結果

PC でランダムに作製した 3×3 行列の LU 分解を作製したプログラム (付録 D.2) で計算した結果を図 3.3 に記す．また，3.2.2 で記した三重ループプログラムの n の値は LU 分解する行列

[6] u02koba/eps/lu-keishiki.eps

の次数に依存しているので，その値を変えることにより LU 分解する行列の次数を変えることが可能である．

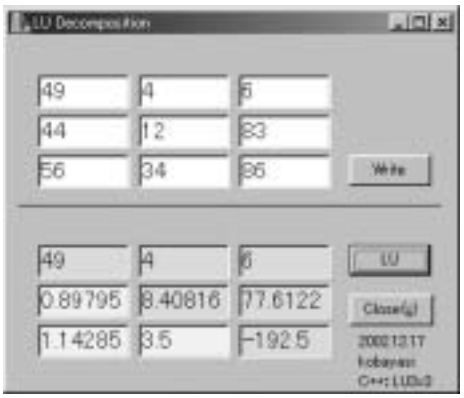

図 3.3: C++での LU 分解結果[7]

[7]u02koba/eps/LUonC.eps

3.3 集積回路上でのLU分解回路の実現

3.3.1 単精度浮動小数点演算器

数値計算を行うことは，主に浮動小数点の四則演算を組み合わせて行うことに帰着する．外積形式ガウス法を用いてLU分解を行う場合，減算器，乗算器，除算器を使用する．乗算器，除算器は山岡 [5] によって作製された．減算器は山岡 [5] が作製した加算器の一部を変更することにより実現した．

計算はIEEE-754の単精度浮動小数点規格 (符号1bit, 指数部8bit, 仮数部23bit) に準拠したデータタイプで行う．

また，減算器，乗算器，除算器はコンポーネントとして使用できるように設計した．

コンポーネントの作製

大規模回路設計では，機能ごとに回路をコンポーネント化し，それらを統合していくことで回路が構築される．回路の階層構造を適切に構築することで，設計を効率化することが可能である．(図3.4)

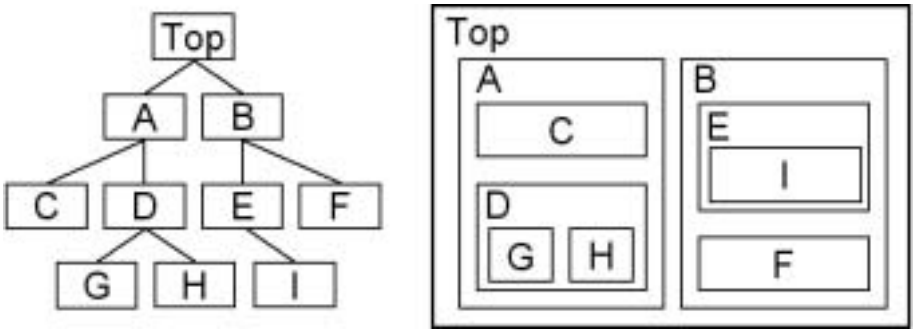

図 3.4: 階層構造の例[8]

コンポーネントは，ブロックに与えた名前がそのブロックを示す回路であり，信号の入力を示す入力ポート，回路からの信号の出力を示す出力ポート，入力および出力を兼用した入出力ポートによって他のブロックと接続する．半加算器をコンポーネント化する場合 (図3.5)，半加算器の入力ポートをAとB，出力ポートをSUMとCARをしてコンポーネントを作製する．この半加算器を使用するときはコンポーネントのポートを上位階層と接続することにより使用可能となる．

例 (半加算器のポート接続)

HA:hadder port map (A=>IN_A, B=>IN_B, SUM, CAR);

---

[8] u02koba/eps/kaiosu.eps

この例で，“HA” はインスタンス名でコンポーネントにインスタンス固有な識別名を与えるものである．“hadder” はコンポーネント名を示す．port map は，コンポーネントのポートをどのように接続するかを指定する．ポートを接続するとき，上位階層のポートとコンポーネント (下層) のポートの結合なので，ポート名が同一である必要はない．

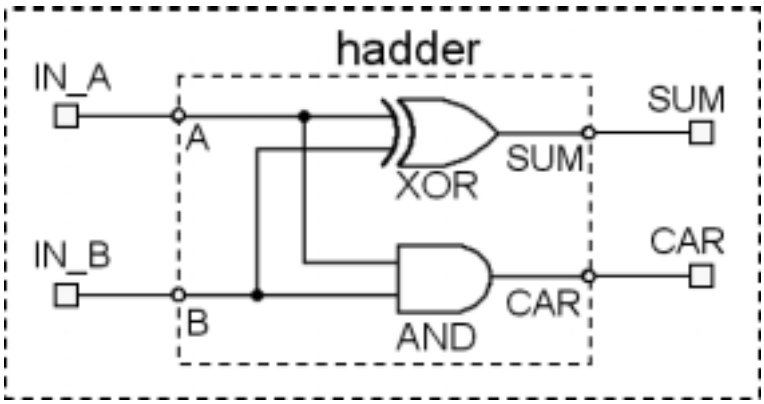

図 3.5: 半加算器のコンポーネント化 [9]

減算器

entity の構成を表 3.1 に，VHDL ソースを付録 A.1 に示す．

表 3.1: 減算器の entity 構成

信号名	用途	方向	型
CLK	クロック	in	`std_logic`
FA	被減数	in	`std_logic_vector(31 downto 0)`
FB	減数	in	`std_logic_vector(31 downto 0)`
Q	差	out	`std_logic_vector(31 downto 0)`

演算は 3 ステージで構成される同期パイプライン方式を用いている．ステージ構成を図 3.6 に示す．ここで，入力における E1，M1，E2，M2 はそれぞれ FA の指数部，仮数部，FB の指数部，仮数部を表す．

[9] u02koba/eps/hadder.eps

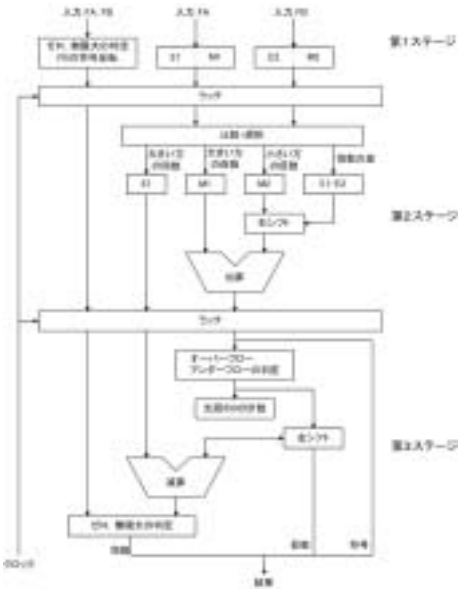

図 3.6: 浮動小数点減算器 [10]

VHDL ソースでは上から順にステージが構成されており，図 3.6 の構成に対応している．演算動作は組み合わせ回路で記述されており，また，ステージ間のラッチは process 文でシステムクロックに同期して行うように記述されている．

まず，第 1 ステージでは FA と FB のゼロ，無限大の判定，FB の符号反転を行う．第 2 ステージでは FA と FB の比較・選択をし，絶対値の小さい方の数の仮数部 (M2) を指数部の差 (E1-E2) だけ右ビットシフトし，M1 と M2 の加算を行う．第 3 ステージでは仮数部の加算結果のオーバーフロー，あるいはアンダーフローの判定をして結果を絶対値表現に戻し，正規化し，それに基づき指数部を調整して計算結果を外部に出力する．実際にこの手順で計算した結果を図 3.7 に示す．

図 3.7: 減算結果 [11]

[10] u02koba/eps/fsub.eps
[11] u02koba/eps/sub_kekka.eps

乗算器

entity の構成を表 3.2 に，VHDL ソースを付録 A.2 に示す.

表 3.2: 乗算器の entity 構成

信号名	用途	方向	型
CLK	クロック	in	`std_logic`
FA	被乗算	in	`std_logic_vector(31 downto 0)`
FB	乗算	in	`std_logic_vector(31 downto 0)`
Q	積	out	`std_logic_vector(31 downto 0)`

演算は 3 ステージで構成される同期パイプライン方式を用いている．ステージ構成を図 3.8 に示す．ここで，入力における E1，M1，E2，M2 はそれぞれ FA の指数部，仮数部，FB の指数部，仮数部を表す.

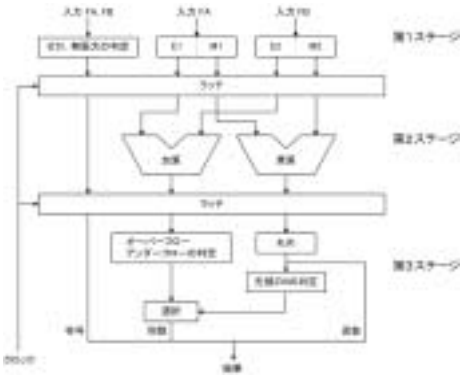

図 3.8: 浮動小数点乗算器 [12]

VHDL ソースでは上から順にステージが構成されており，図 3.8 の構成に対応している．演算動作は組み合わせ回路で記述されており，また，ステージ間のラッチは process 文でシステムクロックに同期して行うように記述されている.

まず，第 1 ステージでは指数部と仮数部を計算形式にし，また積の符号を判定する．第 2 ステージでは仮数部の乗算をし，不要な下位 22 ビットを切り捨て，また，並行して指数部の加算を行う．第 3 ステージでは仮数部の乗算結果の丸めをし，指数部の加算結果のオーバーフロー，あるいはアンダーフローの判定をして結果を調整し，最後に正規化して計算結果を外部に出力する．この手順で計算した結果を図 3.9 に示す.

[12] u02koba/eps/fmul.eps

図 3.9: 乗算結果 [13]

除算器

entity の構成を表 3.3 に，VHDL ソースを付録 A.3 に示す.

表 3.3: 除算器の entity 構成

信号名	用途	方向	型
CLK	クロック	in	`std_logic`
FA	被除数	in	`std_logic_vector(31 downto 0)`
FB	除数	in	`std_logic_vector(31 downto 0)`
Q	商	out	`std_logic_vector(31 downto 0)`

演算は 3 ステージで構成される同期パイプライン方式を用いている．ステージ構成を図 3.10 に示す．ここで，入力における E1，M1，E2，M2 はそれぞれ FA の指数部，仮数部，FB の指数部，仮数部を表す.

[13] u02koba/eps/mul_kekka.eps

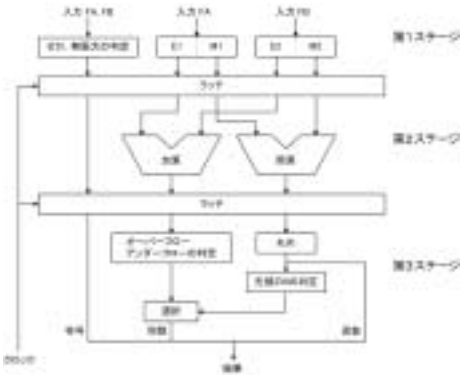

図 3.10: 浮動小数点除数器[14]

VHDL ソースでは上から順にステージが構成されており，図 3.10 の構成に対応している．演算動作は組み合わせ回路で記述されており，また，ステージ間のラッチは process 文でシステムクロックに同期して行うように記述されている．

ステージ構成はほぼ乗算器と同様であり，第 2 ステージにおける仮数部と指数部の演算のみが異なる．第 2 ステージでは仮数部の除算をし，また，並行して指数部の減算を行う．この手順で計算した結果を図 3.11 に示す．

図 3.11: 除算結果[15]

[14] u02koba/eps/fdiv.ps
[15] u02koba/eps/div_kekka.eps

3.3.2 制御モジュール

LU 分解を計算するための制御モジュールの役割を図 3.12 に簡潔に記す.

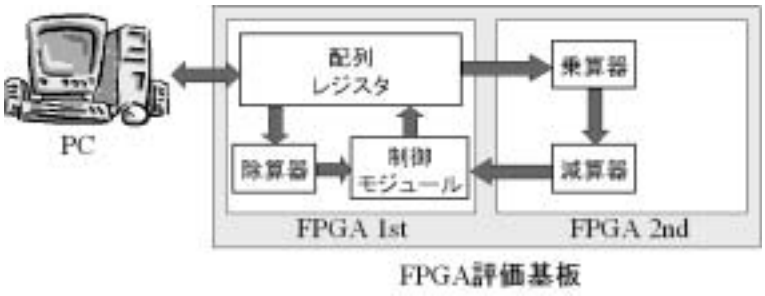

図 3.12: 制御モジュールとデータの流れ [16]

PC とのデータ転送を制御

PC からデータを転送する場合インターフェイスの信号を解析し信号の制御を行う.
まず，付録 A.4 で，OBF というのがインターフェイスボードからの信号である．この OBF はデータが PC からのデータが インターフェイスボードにラッチされると ‘11‘になる (図 3.13:ア). この信号がでたら，データをレジスタ，WRITE_DATA_REG に入れる．PC からのデータは 32bit だが，インターフェイスボードのバス幅が 16bit なので 2 回に別けている．そこで 1 回目のデータを下位 16bit，2 回目のデータを上位 16bit に入れている．この OBF は ACK_BUF を経て ACK に代入される用に記述されている (図 3.13:イ)．ACK とはインターフェイスボードの受取り信号であり，この信号が ‘11‘ になるとインターフェイスボードが OBF に ‘00‘ を返す (図 3.13:ウ)．これにより ACK にも ‘00‘ が入って (図 3.13:エ) 初期状態に戻る.

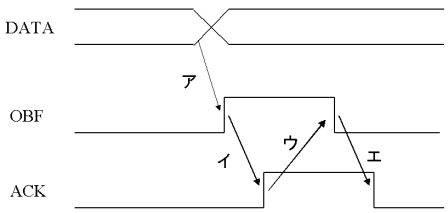

図 3.13: PC からのデータ転送タイミング図 [17]

結果を返すとき，PC から制御信号 ‘00000010‘ に続き，‘00000000‘を BL に連続で出す．FPGA では BL の 2 ビット目 (BL(1)) が下がるのを確認して，計算結果があるレジスタ，QQ の下位 16bit を PC に返す．ここで，OUT_ CNT が ‘1‘になり，続いて同じ信号が出た時今度は上位 16bit が PC に返される．ここでインターフェイスボードの信号の STB(STB_BUF) を ‘00‘にし

[16] u02koba/eps/lu-flow.eps
[17] u02koba/eps/input_frompc.eps

ないと (図 3.14:1) インターフェイスボードにラッチされない．またこの STB は ‘00’ になるとインターフェイスボードの信号，IBF を ‘11‘ (図 3.14:2) にする．このタイミングを見計らい，再び STB を’11’ に戻す (図 3.14:3)．IBF は自動に’00’(図 3.14:4) に戻るのでこの信号は再び考慮する必要はない．

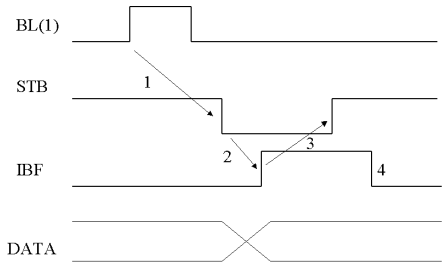

図 3.14: PC へのデータ転送タイミング図 [18]

演算器，配列レジスタ，FPGA 間通信の制御

- 除算器，配列レジスタの制御 (1stFPGA)(VHDL は付録 A.4 を参照)
  3×3 行列の LU 分解を行う場合第 1 ステップとして，配列レジスタから 2 値のデータを読み出し除算を行なう．この計算を行う手順はステートで記述しているので各ステートごとに説明する．

  - DSTART
    ステートの始まり．
    配列レジスタから 2 値のデータを取り出す．
  - DPUSH_DATA
    除算回路コンポーネントに取り出した 2 つのデータを送り，除算を開始する．
  - DWAIT1 ～ DWAIT6
    除算を始めてから計算終了までに 7CLK 必要なので，6CLK 分待機する．
  - DKEKKA_OUT
    除算回路コンポーネントから計算結果の取り出し．
  - DSAVE
    取り出した計算結果を一時的に別のレジスタに入れ，もう一度除算を行う場合は配列レジスタからデータを取り出し，[DPUSH_DATA］のステートに移行．除算を行なわない場合は［DWAIT7］へ移行．
  - DWAIT7
    待ち時間．

[18] u02koba/eps/flow1.ps

- DARRAY
  除算結果を配列レジスタに戻す.
- DWAIT8
  待ち時間.
- DTWO_STR
  2ndFPGA へのデータ転送を制御するステートメントを開始する.
- DSTOP
  ここで除算が終了する.

このステートにより，図 3.2 の (1)(3) の計算手順を行う.

- FPGA 間のデータ通信の制御 (1stFPGA)(VHDL は付録 A.4 を参照)
  LU 分解を行う場合，減算器，乗算器，除算器が必要であるが，それぞれのコンポーネントを 1 つの FPGA に入れることはロジック数の関係でできない．そこで，FPGA を 2 個使うことにより LU 分解を計算する．そのために FPGA 間のデータ通信が必要となる．この通信の制御はステートで制御しているので各ステートについて説明する.

  - START
    ステートの始まり.
    配列レジスタから 3 値のデータを取り出す.
  - RESET
    2ndFPGA にコントロールバスからリセット信号を送り，リセットする.
  - PUSH_DATA1
    1 番目のデータを 2ndFPGA に転送する．(データバス使用)
  - WAIT1
    待ち時間.
  - PUSH_DATA2
    2 番目のデータを 2ndFPGA に転送する．(データバス使用)
  - WAIT2
    待ち時間.
  - PUSH_DATA3
    3 番目のデータを 2ndFPGA に転送する．(データバス使用)
  - WAIT3
    待ち時間.
  - TSTART
    2ndFPGA で計算を開始するための信号をコントロールバスから送る.
  - WAIT4
    計算終了の信号が帰ってくるまで待つ.

- KEKKA_OUT
  1stFPGA に計算結果を送るための信号をコントロールバスから送る.
- RESULT
  2ndFPGA から計算結果を取り出す.
- SAVE
  取り出した計算結果を一時的に別のレジスタに入れ，もう一度計算する場合は配列レジスタからデータを取り出し,［RESET］のステートに移行．計算終了の場合は［WAIT5］へ移行.
- ARRAY
  計算結果を配列レジスタに戻す.
- WAIT5
  待ち時間.
- DIV_BK
  除算が必要であれば除算器制御ステートメントを開始し［WAIT6］に移行.
  必要なければそのまま［WAIT6］に移行.
- WAIT6
  待ち時間.
- STOP
  ここで 2ndFPGA での計算が終了する.

- 減算器，乗算器の制御 (2ndFPGA)(VHDL は付録 A.5 を参照)
  3×3 行列の LU 分解を行う第 2 ステップとして，1stFPGA から送られた 3 値のデータを乗算，減算を行なう．この計算を行う手順もステートで記述しているので各ステートごとに説明する.

  - START
    ステートの始まり.
  - FMUL
    1stFPGA から送られた 1,2 番目のデータを乗算回路コンポーネントに送り乗算を開始する.
  - WAIT1
    乗算終了までの待ち時間.
  - FSUB
    1stFPGA から送られた 3 番目のデータと乗算結果を減算回路コンポーネントに送り減算を開始する.
  - WAIT2,WAIT3
    減算終了までの待ち時間.

- QSUB
  減算結果の取り出し.
- STOP
  ここで減算，乗算が終了する.

このステートにより，図 3.2 の (2)(4) の計算手順を行う.

3.3.3 ロジックセル使用率

設計した各モジュールのロジックセル使用率を表3.4に示す．1個のFPGAに減算器，乗算器，除算器を入れるとほぼすべてのロジックセルを使用してしまうので，計算を制御するモジュールを入れることができない．

1個目のFPGAに除算器モジュールとPC-FPGA間通信，計算制御モジュールを入れ，2個目のFPGAに減算器，乗算器モジュールを入れることによりLU分解の計算が可能となる．

表3.4で1stFPGAでのロジックセルの値は除算器が含まれており，2ndFPGAの値は減算器，乗算器が含まれている．また，2×2行列でも3×3行列でも2ndFPGAでのLU分解計算モジュールは同じものを使用した．

表 3.4: ロジックセル使用率

モジュール名	ロジックセル数 (Max 4992)
減算器	1058 (21 %)
乗算器	2120 (42 %)
除算器	1688 (33 %)
2×2 行列 LU 分解 (1stFPGA)	2178 (43 %)
3×3 行列 LU 分解 (1stFPGA)	3550 (71 %)
行列 LU 分解 (2ndFPGA)	3149 (63 %)

3.3.4 結果

PCでランダムに作った単精度浮動小数点で3×3の正方行列を作り，評価基板上の動作クロックを10MHzで計算させた結果を図3.15に記す．また，この結果とPCで計算した結果を図3.16に示し結果が一致することを確認した．

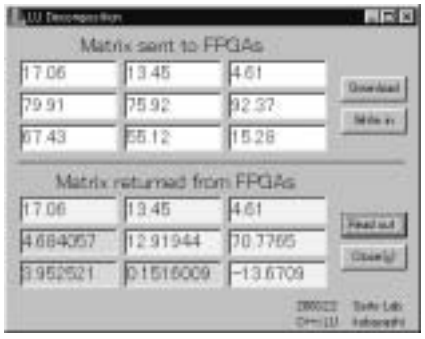

図 3.15: 評価基板での計算結果 [19]

図 3.16: PCでの計算結果 [20]

[19] u02koba/eps/LU-fpga.eps
[20] u02koba/eps/LU-com.eps

第4章　考察と今後への提言

2個のFPGAを使用し，各演算モジュールを用いて3×3行列のLU分解を10MHzのクロック数で計算する演算デバイスは完成した．しかし，評価基板に搭載されているメモリ(SRAM, DRAM)を用いて計算可能な行列の次数を増やすまでには至らず，またLU分解を用いて連立1次方程式を解くことにも至らなかった．

今後の課題は第一に計算できる行列の拡大である．そのためにSRAM，DRAMを使用し，またLU分解を計算する三重ループを行列の大きさに依存しないように改良する必要がある．または，現在使用している評価基板のFPGAはそれぞれ10万ゲートであり，1つの演算ジュールでも約20〜40％使用する．そこで，100万ゲート程度のFPGAを搭載した基板を作製もしくは購入することによりゲート数の問題は軽減し，多くの演算モジュールを入れることができるので高速に演算することも可能である．また，インターフェイス部分の改善とPCと評価基盤の通信速度の短縮も課題の1つである．現在使用している評価基板はのISAバスを介し通信していてバス幅が16bitの為，単精度浮動小数点(32bit)では2回，倍精度浮動小数点(64bit)では4回 もの通信が必要になってくる．さらに最近のPCにはISAバスが標準に搭載されているPCは少なくなってきた．そこでPCIバスもしくはUSBをインターフェイスとする新しい基板を使用するべきではべきではないだろうか．

謝辞

本研究および論文作成にあたり，終始懇切なる御指導御鞭撻を賜りました指導教官である齋藤理一郎助教授に心より御礼の言葉を申し上げます.

本研究およびセミナー等で御指導を賜りました木村忠正教授，湯郷成美教授，一色秀夫助手に深く感謝の意を表します.

また，本研究を行うにあたり，様々な資産を残していただいた八木将志様，中島瑞樹様，松尾竜馬様，グエンドゥックミン様，山岡寛明様，ホーフィクー様，沼知典様，清水信貴様，稲荷徹様，阿部正之様に多大なる感謝を致します．特に稲荷徹様，阿部正之様には丁寧に直接指導して頂きました．改めて感謝致します.

さらに，木村齋藤研究室，湯郷研究室の皆様方にも感謝致します.

また，事務業務をして頂きました渡辺美帆子様にも感謝いたします.

本研究にあたって，MAX+plusII を無償提供していただきました日本アルテラ (株) に感謝致します.

最後に，本研究を進めるにあたり多大な御理解，御配慮を頂いた勤務先である総合情報処理センターの教職員の皆様に感謝の意を表します.

参考文献

[1] 中島瑞樹, “超高速行列演算チップの開発” , 1996 年度卒業論文

[2] 八木将志, “大行列の対角化プログラムの並列化” ,1996 年度卒業論文

[3] 松尾竜馬, “行列計算専用大規模集積回路の開発” ,1997 年度卒業論文

[4] グェン・ドゥック・ミン, “ハードウェア記述言語を用いた行列計算専用プロセッサの設計” ,1997 年度卒業論文

[5] 山岡寛明, “FPGA を用いた行列計算専用プロセッサの設計” ,1998 年度卒業論文

[6] 沼知典, “書き換え可能なゲート素子を持つデバイスを用いた行列計算専用集積回路の設計” ,1999 年度修士論文

[7] 清水信貴, “ハードウェア記述言語を用いた専用デバイスの設計” ,2000 年度卒業論文

付 録A　プログラムソース

A.1　浮動小数点減算器 (u02koba/programs/VHDL/Fpsub.vhd)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity fpsub is
  port (
         CLK  : in std_logic;
         FA   : in std_logic_vector(31 downto 0);
         FB   : in std_logic_vector(31 downto 0);
         Q    : out std_logic_vector(31 downto 0)
       );

end fpsub;
architecture RTL of fpsub is

signal INF1, INF2, INF3, ZA1, ZA2, ZA3, ZB1, ZB2, ZB3 : std_logic;
signal SA1, SA2, SA3, SB1, SB2, SB3                   : std_logic;
signal EA1, EA2, EB1, EB2                             : std_logic_vector(8 downto 0);
signal EA3, EA4, EQ, ED1, ED2                         : std_logic_vector(7 downto 0);
signal MA1, MA2, MA3, MB1, MB2, MB5                   : std_logic_vector(22 downto 0);
signal MB3                                            : std_logic_vector(23 downto 0);
signal MA4, MB4                                       : std_logic_vector(25 downto 0);
signal MQ1, MQ2                                       : std_logic_vector(25 downto 0);
signal MQ3, MQ4                                       : std_logic_vector(24 downto 0);
signal MQ5                                            : std_logic_vector(22 downto 0);
signal V0, V1, V2, V3, V4, V5                         : std_logic_vector(24 downto 0);
signal VV0, VV1, VV2, VV3, VV4                        : std_logic_vector(24 downto 0);
signal VES1, VES2                                     : std_logic_vector(8 downto 0);

begin

INF1 <= '1' when FA(30 downto 23) = "11111111" or FB(30 downto 23) = "11111111" else '0';
ZA1 <= '1' when FA(30 downto 23) = "00000000" else '0';
ZB1 <= '1' when FB(30 downto 23) = "00000000" else '0';

EA1 <= '0' & FA(30 downto 23);
EB1 <= '0' & FB(30 downto 23);

process begin
wait until rising_edge( CLK );
SA1 <= FA(31);
    SB1 <= not FB(31);
EA2 <= EA1;
EB2 <= EB1;
MA1 <= FA(22 downto 0);
MB1 <= FB(22 downto 0);
INF2 <= INF1;
ZA2 <= ZA1;
ZB2 <= ZB1;
end process;

VES1 <= EA2 - EB2;
VES2 <= EB2 - EA2;

SA2 <= SA1 when VES1(8) = '0' else SB1;
```

```
SB2 <= SB1 when VES1(8) = '0' else SA1;

EA3 <= EA2(7 downto 0) when VES1(8) = '0' else EB2(7 downto 0);
ED1 <= VES1(7 downto 0) when VES1(8) = '0' else VES2(7 downto 0);

MA2 <= MA1 when VES1(8) = '0' else MB1;
MB2 <= MB1 when VES1(8) = '0' else MA1;

ZA3 <= ZA2 when VES1(8) = '0' else ZB2;
ZB3 <= ZB2 when VES1(8) = '0' else ZA2;

V0 <= "1" & MB2 & "0" when ED1(0) = '0' else "01" & MB2(22 downto 1) & "0";
V1 <= V0 when ED1(1) = '0' else "00" & V0(24 downto 2);
V2 <= V1 when ED1(2) = '0' else "0000" & V1(24 downto 4);
V3 <= V2 when ED1(3) = '0' else "00000000" & V2(24 downto 8);
V4 <= V3 when ED1(4) = '0' else "0000000000000000" & V3(24 downto 16);
V5 <= V4 + "00000000000000000000000001";
MB3 <= V5(24 downto 1) when ED1(7 downto 5) = "000" else "000000000000000000000000";

MA4 <= "000000000000000000000000" when ZA3 = '1' else
       "001" & MA2 when SA2 = '0' else
       "000000000000000000000000" - ("001" & MA2);

MB4 <= "000000000000000000000000" when ZB3 = '1' else
       "00" & MB3 when SB2 = '0' else
       "000000000000000000000000" - ("00" & MB3);

MQ1 <= MA4 + MB4;

process begin
wait until rising_edge( CLK );
EA4 <= EA3;
MQ2 <= MQ1;
INF3 <= INF2;
end process;

MQ3 <= MQ2(24 downto 0) when MQ2(25) = '0' else
       "0000000000000000000000000" - MQ2(24 downto 0);

MQ4 <= MQ3 + "00000000000000000000000001";

ED2 <=  "00000000" when MQ3(24) = '1' else
        "00000001" when MQ3(23) = '1' else
        "00000010" when MQ3(22) = '1' else
        "00000011" when MQ3(21) = '1' else
        "00000100" when MQ3(20) = '1' else
        "00000101" when MQ3(19) = '1' else
        "00000110" when MQ3(18) = '1' else
        "00000111" when MQ3(17) = '1' else
        "00001000" when MQ3(16) = '1' else
        "00001001" when MQ3(15) = '1' else
        "00001010" when MQ3(14) = '1' else
        "00001011" when MQ3(13) = '1' else
        "00001100" when MQ3(12) = '1' else
        "00001101" when MQ3(11) = '1' else
        "00001110" when MQ3(10) = '1' else
        "00001111" when MQ3( 9) = '1' else
        "00010000" when MQ3( 8) = '1' else
        "00010001" when MQ3( 7) = '1' else
        "00010010" when MQ3( 6) = '1' else
        "00010011" when MQ3( 5) = '1' else
        "00010100" when MQ3( 4) = '1' else
        "00010101" when MQ3( 3) = '1' else
        "00010110" when MQ3( 2) = '1' else
        "00010111" when MQ3( 1) = '1' else
        "00011000" when MQ3( 0) = '1' else
        "10000000";

VV0 <= MQ3 when ED2(0) = '0' else MQ3(23 downto 0) & "0";
VV1 <= VV0 when ED2(1) = '0' else VV0(22 downto 0) & "00";
VV2 <= VV1 when ED2(2) = '0' else VV1(20 downto 0) & "0000";
VV3 <= VV2 when ED2(3) = '0' else VV2(16 downto 0) & "00000000";
VV4 <= VV3 when ED2(4) = '0' else VV3( 8 downto 0) & "0000000000000000";
MQ5 <= VV4(23 downto 1) when MQ4(24) = '0' else MQ4(23 downto 1);

EQ <= "11111111" when INF3 = '1' else
```

```
      EA4 - ED2 + "00000001" when ED2(7) = '0' else
      "00000000";

Q <= MQ2(25) & EQ & MQ5;
end RTL;
```

A.2 浮動小数点乗算器 (u02koba/programs/VHDL/multiply.vhd)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity fpmul is
  port (    CLK  : in std_logic;
            FA   : in std_logic_vector(31 downto 0);
            FB   : in std_logic_vector(31 downto 0);
            Q    : out std_logic_vector(31 downto 0)
      );

end fpmul;
architecture RTL of fpmul is

signal MA1, MA2, MB1, MB2 : std_logic_vector(23 downto 0);
signal EA1, EA2, EB1, EB2 : std_logic_vector(9 downto 0);
signal MQ1, MQ2, MQ3      : std_logic_vector(25 downto 0);
signal EQ1, EQ2, EQ3, EQ4 : std_logic_vector(9 downto 0);
signal EQ5, EQ6, EQ       : std_logic_vector(7 downto 0);
signal MQ                 : std_logic_vector(22 downto 0);
signal S1, S2, SQ ,ZERO_FLAG_A1,ZERO_FLAG_A2,ZERO_FLAG_A3,
       ZERO_FLAG_B1,ZERO_FLAG_B2,ZERO_FLAG_B3: std_logic;

begin

MA1 <= '1' & FA(22 downto 0);
MB1 <= '1' & FB(22 downto 0);
EA1 <= "00" & FA(30 downto 23);
EB1 <= "00" & FB(30 downto 23);
S1  <= FA(31) xor FB(31);
ZERO_FLAG_A1 <= '1' when FA(22 downto 0) = "00000000000000000000000"
                and FA(30 downto 23) = "00000000" else
                '0';
ZERO_FLAG_B1 <= '1' when FB(22 downto 0) = "00000000000000000000000"
                    and FB(30 downto 23) = "00000000" else
                    '0';
process begin
wait until rising_edge( CLK );
MA2 <= MA1;
MB2 <= MB1;
EA2 <= EA1;
EB2 <= EB1;
S2  <= S1;
    ZERO_FLAG_A2 <= ZERO_FLAG_A1;
    ZERO_FLAG_B2 <= ZERO_FLAG_B1;

end process;

process( MA2, MB2 )
        variable TMQ : std_logic_vector(47 downto 0);
begin
       TMQ := MA2 * MB2;
       MQ1 <= TMQ(47 downto 22);

end process;

EQ1 <= "0011111111" when EA2(7 downto 0) = "11111111" or EB2(7 downto 0) = "11111111" else
       "0000000000" when EA2(7 downto 0) = "00000000" or EB2(7 downto 0) = "00000000" else
       EA2 + EB2 - "0001111111";

EQ2 <= "0011111111" when EA2(7 downto 0) = "11111111" or EB2(7 downto 0) = "11111111" else
```

```
       "0000000000" when EA2(7 downto 0) = "00000000" or EB2(7 downto 0) = "00000000" else
       EA2 + EB2 - "0001111110";

process begin
wait until rising_edge( CLK );
SQ <= S2;
MQ2 <= MQ1;
EQ3 <= EQ1;
EQ4 <= EQ2;
    ZERO_FLAG_A3 <= ZERO_FLAG_A2;
    ZERO_FLAG_B3 <= ZERO_FLAG_B2;

end process;

MQ3 <= MQ2 + "00000000000000000000000001" when MQ2(25) = '0' else
       MQ2 + "00000000000000000000000010";

EQ5 <= "00000000" when EQ3(9 downto 8) = "11" else
       "11111111" when EQ3(9 downto 8) = "01" else
       EQ3(7 downto 0);

EQ6 <= "00000000" when EQ4(9 downto 8) = "11" else
       "11111111" when EQ4(9 downto 8) = "01" else
       EQ4(7 downto 0);

MQ <= "00000000000000000000000" when ZERO_FLAG_A3 = '1' or ZERO_FLAG_B3 = '1' else
      MQ3(23 downto 1) when MQ3(25) = '0' else
      MQ3(24 downto 2);

EQ <= EQ5 when MQ3(25) = '0' else
      EQ6;

Q <= SQ & EQ & MQ;
end RTL;
```

A.3 浮動小数点除算器 (u02koba/programs/VHDL/Division.vhd)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity fpdiv is
  port (
            CLK  : in std_logic;
            FA   : in std_logic_vector(31 downto 0);
            FB   : in std_logic_vector(31 downto 0);
            Q    : out std_logic_vector(31 downto 0)
       );

end fpdiv;
architecture RTL of fpdiv is

signal MA, MB : std_logic_vector(23 downto 0);
signal MC, MCT, MC0: std_logic_vector(25 downto 0);
signal MQ : std_logic_vector(22 downto 0);
signal EA, EB, EC0, EC1: std_logic_vector(9 downto 0);
signal EC0A, EC1A, EQ : std_logic_vector(7 downto 0);
signal S, SS, SSS, SQ: std_logic;

begin

------------< Floating-Point Division Algorithms >
process begin --1st step--
wait until rising_edge( CLK ); -- CLOCK の立ち上がりに同期
MA <= '1' & FA(22 downto 0);   -- 被除数の仮数 bit を MA へ (1+23bit)
MB <= '1' & FB(22 downto 0);   -- 除数の仮数 bit を MB へ (1+23bit)
EA <= "00" & FA(30 downto 23); -- 被除数の指数 bit を EA へ (2+8bit)
EB <= "00" & FB(30 downto 23); -- 除数の指数 bit を EB へ (2+8bit)
```

```
S <= FA(31) xor FB(31);         -- 被除数・除数の符号 bit の xor を取ることにより s に最終結果の符合
bit を入れる
end process;

process --2nd step--
variable TA, TB, REMAIN : std_logic_vector(24 downto 0); -- この provcess 内で使う変数の宣言
begin
wait until falling_edge( CLK ); -- CLOCK の立ち下がりに同期
    SS <= S; -- 符号 bit の最終結果を SS へ

    TA := '0' & MA; -- 減算結果の判断の為に 1bit 加える (1+24bit)
    TB := '0' & MB; -- 被除数の bit 数に合わせる (1+24bit)

---- 以下は for loop で仮数 bit の除算した結果の商を 26bit 出す ----
for index in 25 downto 0 loop -- index を 25 から 0 まで変えて loop (index はこの for loop の中のみ
に使用)
REMAIN := TA - TB; -- 仮数 bit の減算
   if REMAIN(24) = '1' then --TA < TB の場合
   MC(index) <= '0';             -- MC(index) に 0 を入れる
   TA := TA(23 downto 0) & '0'; -- TA を 1bit 上にシフト

        else -- TA > TB の場合
   MC(index) <= '1';                 -- MC(index) に 0 を入れる
   TA := REMAIN(23 downto 0) & '0'; -- 減算結果を 1bit 上にシフトして TA へ
   end if;
end loop;
---- 仮数 bit 除算終了 ----

    MCT <= MC;

---- 以下は指数 bit の減算 ----
if EA = "0011111111" or EB = "0000000000" then -- 被除数の指数 bit=MAX, 又は除数の指数 bit=MINI
の場合
EC0 <= "0011111111"; -- EC0 と
EC1 <= "0011111111"; -- EC1 を MAX

elsif EA = "0000000000" or EB = "0011111111" then -- 被除数の指数 bit=MINI, 又は除数の指数 bit=MAX
の場合
EC0 <= "0000000000"; -- EC0 と
EC1 <= "0000000000"; -- EC1 を MINI

else -- 指数 bit が MAX or MINI 以外の場合
EC0 <= EA - EB + "0001111110"; -- 指数の差に bias を加える (TA < TB の場合)
EC1 <= EA - EB + "0001111111"; -- 指数の差に bias を加える (TA > TB の場合)
end if;
---- 指数 bit の加算終了 ----
end process;

process begin --3rd step--
wait until rising_edge( CLK ); -- CLOCK の立ち上がりに同期
SSS <= SS; -- 符号 bit の最終結果を SSS へ

if MCT(25) = '0' then                              -- ┐
MC0 <= MCT + "00000000000000000000000001"; --  |
else                                               --  | 仮数 bit の最終 bit の繰上げを考慮
MC0 <= MCT + "00000000000000000000000010"; --  |
end if;                                            -- ┘

---- 以下の case 文 2 つは指数 bit 加算のオーバーフロー, アンダーフローの判断 ----
case EC0(9 downto 8) is -- TA < TB の場合
when "11" => EC0A <= "00000000"; -- EA << EB の場合
when "01" => EC0A <= "11111111"; -- EA >> EB の場合
when others => EC0A <= EC0(7 downto 0); -- それ以外,EC0 を EC0A へ (8bit)
end case;

case EC1(9 downto 8) is -- TA > TB の場合
when "11" => EC1A <= "00000000"; -- EA << EB の場合
when "01" => EC1A <= "11111111"; -- EA >> EB の場合
```

```
when others => EC1A <= EC1(7 downto 0); -- それ以外,EC1 を EC1A へ (8bit)
end case;
---- 指数 bit 加算のオーバーフロー, アンダーフロー判断終了 ----
end process;

process begin --4th step--
wait until falling_edge( CLK ); -- CLOCK の立ち下がりに同期
SQ <= SSS; -- 符号 bit の最終結果を SQ へ

if MC0(25) = '0' then  -- TA < TB の場合
MQ <= MC0(23 downto 1); -- 仮数 bit の最終除算結果を MQ へ
EQ <= EC0A; -- 指数 bit の最終除算結果を EQ へ

else  -- TA > TB の場合
MQ <= MC0(24 downto 2); -- 仮数 bit の最終除算結果を MQ へ
EQ <= EC1A; -- 指数 bit の最終除算結果を EQ へ
end if;
end process;

Q <= SQ & EQ & MQ; -- 最終除算結果の float 型

end RTL;
```

A.4 メインプログラム(1stFPGA)(u02koba/programs/VHDL/fpga1.vhd)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

library metamor;
use metamor.attributes.all;
entity first is

port (
       CLK       : in    std_logic;
       A         : inout std_logic_vector(15 downto 0);
       BL        : in    std_logic_vector( 7 downto 0);
       BH        : out   std_logic_vector( 7 downto 0);
       CL        : in    std_logic_vector( 5 downto 0);
       OBF       : in    std_logic_vector( 1 downto 0);
       IBF       : in    std_logic_vector( 1 downto 0);
       ACK       : out   std_logic_vector( 1 downto 0);
       STB       : out   std_logic_vector( 1 downto 0);
       DATA_BUS  : inout std_logic_vector(31 downto 0);
       ADRS_BUS  : out   std_logic_vector(16 downto 0);
       CTRL_BUS  : out   std_logic_vector( 4 downto 0);
       CALC_DONE : in    std_logic;
       OE_ALU    : out   std_logic
    );

  attribute pinnum of A   : signal is  "BC23,BB24,BC25,BB26,BC27,BB28,BC29,BB30,
                                        BC31,BB32,BC33,BB34,BC35,BB36,BC37,BB38";
  attribute pinnum of BL  : signal is  "BC13,BB14,BC15,BB16,BC17,BB18,BC19,BB20";
  attribute pinnum of BH  : signal is  "BC5,BB6,BC7,BB8,BC9,BB10,BC11,BB12";
  attribute pinnum of CLK : signal is  "D22";
  attribute pinnum of CL  : signal is  "AU23,AV24,AU25,AU33,AV34,AU35";
  attribute pinnum of OBF : signal is  "AV18,AV28";
  attribute pinnum of IBF : signal is  "AV20,AV30";
  attribute pinnum of ACK : signal is  "AU19,AU29";
  attribute pinnum of STB : signal is  "AU21,AU31";

  attribute pinnum of DATA_BUS  : signal is  "A5,B6,A7,B8,A9,B10,A11,B12,A13,B14,
                                              A15,B16,A17,B18,A19,B20,A23,B24,A25,
                                              B26,A27,B28,A29,B30,A31,B32,A33,B34,
                                              A35,B36,A37,B38";
  attribute pinnum of ADRS_BUS  : signal is  "F16,G19,F20,G21,F22,G23,F24,G25,F26,
                                              G29,F30,G31,F32,G33,F34,G35,F36";
  attribute pinnum of CTRL_BUS  : signal is  "G11,F12,G13,F14,G15";
```

```
  attribute pinnum of CALC_DONE : signal is  "F10";
  attribute pinnum of OE_ALU    : signal is  "G9";


end first;
architecture RTL of first is

component fpdiv is
port (
         CLK  : in  std_logic;
         FA   : in  std_logic_vector(31 downto 0);
         FB   : in  std_logic_vector(31 downto 0);
         Q    : out std_logic_vector(31 downto 0)
      );
end component;

signal KEKKA                 : std_logic_vector(31 downto 0 );
signal WRITE_DATA_REG_L      : std_logic_vector(31 downto 0 );
signal READ_DATA_REG_L       : std_logic_vector(31 downto 0 );
signal WRITE_DATA_ACTIVE_BUF : std_logic;
signal A_REG2                : std_logic_vector(15 downto 0 );
signal ACK_BUF               : std_logic_vector( 1 downto 0 );
signal STB_BUF               : std_logic_vector( 1 downto 0 );
signal WRITE_DATA_ACTIVE     : std_logic;
signal READ_DATA_ACTIVE      : std_logic;
signal IN_CNT                : std_logic;
signal OUT_CNT               : std_logic;

type STATE_TYPE1 is
     (DSTOP,DWAIT1,DWAIT2,DWAIT3,DWAIT4,DWAIT5,DWAIT6,DWAIT7,DWAIT8,DWAIT9,
      DWAIT72,DWAIT82,DWAIT92,DPUSH_DATA,DSTART,DSTART2,DKEKKA_OUT,
      DSAVE,DARRAY,DTWO_STR,DARRAY2,DTWO_STR2);
signal CURRENT_STATE1 : STATE_TYPE1;
signal NEXT_STATE1    : STATE_TYPE1;

type STATE_TYPE2 is
     (STOP,PUSH_DATA1,PUSH_DATA2,PUSH_DATA3,KEKKA_OUT,WAIT1,WAIT2,WAIT3,
     WAIT4,WAIT5,WAIT52,WAIT6,START,START2,RESET,TSTR,RESULT,SAVE,
     ARRAY1,ARRAY2,DIV_BK);
signal CURRENT_STATE2 : STATE_TYPE2;
signal NEXT_STATE2    : STATE_TYPE2;

subtype VECTOR  is std_logic_vector(31 downto 0);
type WORD is array (0 to 8) of VECTOR ;
signal MEM : WORD;

signal CNT1         : integer := 0 ;
signal CNT2         : integer := 0 ;
signal DATA_BUS_BUF : std_logic_vector(31 downto 0);
signal OE_BUF       : std_logic;

constant cRESET     : std_logic_vector( 4 downto 0):="00000";
constant cMEM_1     : std_logic_vector( 4 downto 0):="00001";
constant cMEM_2     : std_logic_vector( 4 downto 0):="00010";
constant cMEM_3     : std_logic_vector( 4 downto 0):="00011";
constant cCLK_STR   : std_logic_vector( 4 downto 0):="00100";
constant cKEKKA_OUT : std_logic_vector( 4 downto 0):="01000";
constant cSTOP      : std_logic_vector( 4 downto 0):="10000";

signal KA      : std_logic_vector(31 downto 0);
signal KB      : std_logic_vector(31 downto 0);
signal TWO_KA  : std_logic_vector(31 downto 0);
signal TWO_KB  : std_logic_vector(31 downto 0);
signal TWO_KC  : std_logic_vector(31 downto 0);
signal DIV_FA  : std_logic_vector(31 downto 0);
signal DIV_FB  : std_logic_vector(31 downto 0);
signal DIV_Q   : std_logic_vector(31 downto 0);
signal C_REG   : std_logic_vector(31 downto 0);
signal C_REG1  : std_logic_vector(31 downto 0);
signal C_REG2  : std_logic_vector(31 downto 0);
```

```
signal D_REG   : std_logic_vector(31 downto 0);
signal D_REG1  : std_logic_vector(31 downto 0);
signal D_REG2  : std_logic_vector(31 downto 0);
signal D_REG3  : std_logic_vector(31 downto 0);
signal D_REG4  : std_logic_vector(31 downto 0);
signal DIV_CNT1 : std_logic_vector(1 downto 0) :="00";
signal DIV_CNT2 : std_logic :='0';
signal DIV_STR : std_logic :='0';
signal DIV_END1 : std_logic :='0';
signal DIV_END2 : std_logic :='0';
signal TWO_CNT : std_logic_vector(2 downto 0) :="000";
signal TWO_CNT2 : std_logic :='0';
signal TWO_STR : std_logic :='0';
signal TWO_STR2 : std_logic :='0';
signal TWO_END1 : std_logic :='0';
signal TWO_END2 : std_logic :='0';


begin

fpdivision : fpdiv  port map (CLK, FA=>DIV_FA, FB=>DIV_FB, Q=>DIV_Q);

DATA_BUS <= "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ" when OE_BUF = '1' else DATA_BUS_BUF;
OE_ALU <= OE_BUF;

A <= "ZZZZZZZZZZZZZZZZ" when ACK_BUF = "00" else A_REG2;

ACK_BUF <= OBF;
ACK <= ACK_BUF;
STB <= STB_BUF;

--------入力信号制御
process ( BL(0),BL(3),OBF,IN_CNT) begin
  if OBF'event and OBF = "11" then
      if BL(3) = '1' then
          if IN_CNT = '0' then
              IN_CNT <= '1';
              WRITE_DATA_ACTIVE_BUF <= '0';
              WRITE_DATA_REG_L(15 downto 0) <= A;

          elsif IN_CNT = '1' then
              IN_CNT <= '0';
              WRITE_DATA_REG_L(31 downto 16) <= A;
              WRITE_DATA_ACTIVE_BUF <= '1';
          end if;
      end if;
  end if;

  if BL(0) = '1' then
      IN_CNT <= '0';
      WRITE_DATA_ACTIVE_BUF <= '0';
  end if;
end process;

process (CLK) begin
  if falling_edge (CLK) then
      WRITE_DATA_ACTIVE <= WRITE_DATA_ACTIVE_BUF;
  end if;
end process;

---------レジスタのアドレス制御
process(WRITE_DATA_ACTIVE,BL(0)) begin
  if rising_edge(WRITE_DATA_ACTIVE ) then
      MEM(CNT1) <= WRITE_DATA_REG_L;
      CNT1 <= CNT1 + 1;

  elsif rising_edge(CLK) then
      if DIV_END1 = '1' then
          MEM(3) <= C_REG1;
          MEM(6) <= C_REG2;

      elsif DIV_END2 = '1' then
          MEM(7) <= C_REG1;
```

```
      elsif TWO_END1 = '1' then
          MEM(4) <= D_REG1;
          MEM(5) <= D_REG2;
          MEM(7) <= D_REG3;
          MEM(8) <= D_REG4;

     elsif TWO_END2 = '1' then
          MEM(8) <= D_REG1;
     end if ;
  end if;

  if BL(0) = '1'then
      CNT1 <= 0;
  end if;

end process;


---------除算制御
process (CLK)begin
 if falling_edge(CLK) then
     case CURRENT_STATE1 is

        when DSTART =>
              KA <= MEM(0);
              KB <= MEM(3);
              NEXT_STATE1 <= DPUSH_DATA;

        when DSTART2 =>
              KA <= MEM(4);
              KB <= MEM(7);
              NEXT_STATE1 <= DPUSH_DATA;

        when DPUSH_DATA =>
              DIV_FA <= KB;
              DIV_FB <= KA;
              NEXT_STATE1 <= DWAIT1;

        when DWAIT1 =>
              NEXT_STATE1 <= DWAIT2;

        when DWAIT2 =>
              NEXT_STATE1 <= DWAIT3;

        when DWAIT3 =>
              NEXT_STATE1 <= DWAIT4;

        when DWAIT4 =>
              NEXT_STATE1 <= DWAIT5;

        when DWAIT5 =>
              NEXT_STATE1 <= DWAIT6;

        when DWAIT6 =>
              NEXT_STATE1 <= DKEKKA_OUT;

        when DKEKKA_OUT =>
              C_REG <= DIV_Q;
              NEXT_STATE1 <= DSAVE;

        when DSAVE =>
            if DIV_CNT1 = "00" then
              C_REG1 <= C_REG;
              DIV_CNT1 <= "01";
              KB <= MEM(6);
              NEXT_STATE1 <= DPUSH_DATA;

            elsif DIV_CNT1 = "01" then
              C_REG2 <= C_REG;
              DIV_CNT1 <= "10";
              NEXT_STATE1 <= DWAIT7;

            elsif DIV_CNT1 ="10" then
              C_REG1 <= C_REG;
              NEXT_STATE1 <= DWAIT72;
            end if;
```

```
        when DWAIT7 =>
              NEXT_STATE1 <= DARRAY;

        when DARRAY =>
              DIV_END1 <= '1';
              NEXT_STATE1 <= DWAIT8;

        when DWAIT8 =>
              DIV_END1 <= '0';
              NEXT_STATE1 <= DTWO_STR;

        when DTWO_STR =>
              TWO_STR <= '1';
              NEXT_STATE1 <= DWAIT9;

        when DWAIT9 =>
              TWO_STR <= '0';
              NEXT_STATE1 <= DSTOP;

        when DWAIT72 =>
              NEXT_STATE1 <= DARRAY2;

        when DARRAY2 =>
              DIV_END2 <= '1';
              NEXT_STATE1 <= DWAIT82;

        when DWAIT82 =>
              DIV_END2 <= '0';
              NEXT_STATE1 <= DTWO_STR2;

        when DTWO_STR2 =>
              TWO_STR2 <= '1';
              NEXT_STATE1 <= DWAIT92;

        when DWAIT92 =>
              TWO_STR2 <= '0';
              NEXT_STATE1 <= DSTOP;

        when DSTOP =>
              NEXT_STATE1 <= DSTOP;

     end case;
  end if;

  if BL(0) = '1'then
      DIV_END1 <= '0';
      DIV_END2 <= '0';
      TWO_STR <= '0';
      DIV_CNT1 <= "00";
      DIV_CNT2 <= '0';
  end if;
end process;

----------2nd FPGA 制御（乗算, 減算）
process   (CLK) begin
  if falling_edge(CLK) then
      case CURRENT_STATE2 is
          when START =>
              TWO_KA <= MEM(1);
              TWO_KB <= MEM(3);
              TWO_KC <= MEM(4);
              NEXT_STATE2 <= RESET;

          when START2 =>
              TWO_KA <= MEM(5);
              TWO_KB <= MEM(7);
              TWO_KC <= MEM(8);
              NEXT_STATE2 <= RESET;

          when RESET =>
              OE_BUF <= '1';
              CTRL_BUS <=cRESET;
              NEXT_STATE2 <= PUSH_DATA1;

          when PUSH_DATA1 =>
```

```
    DATA_BUS_BUF <= TWO_KA;
    OE_BUF <= '0';
    CTRL_BUS <= "00001";
    NEXT_STATE2<= WAIT1;

when WAIT1 =>
    OE_BUF <= '1';
    NEXT_STATE2<= PUSH_DATA2;

when PUSH_DATA2 =>
    DATA_BUS_BUF <= TWO_KB;
    OE_BUF <= '0';
    CTRL_BUS <= "00010";
    NEXT_STATE2<= WAIT2;

when WAIT2 =>
    OE_BUF <= '1';
    NEXT_STATE2<= PUSH_DATA3;

when PUSH_DATA3 =>
    DATA_BUS_BUF <= TWO_KC;
    OE_BUF <= '0';
    CTRL_BUS <= "00011";
    NEXT_STATE2<= WAIT3;

when WAIT3 =>
    NEXT_STATE2<= TSTR;

when TSTR =>
    CTRL_BUS <= "00100";
    NEXT_STATE2<= WAIT4;

when WAIT4 =>
    if CALC_DONE = '1' then
        NEXT_STATE2 <= KEKKA_OUT;

    else
        NEXT_STATE2 <= WAIT4;
    end if ;

when KEKKA_OUT =>
    CTRL_BUS<= cKEKKA_OUT;
    OE_BUF <= '1';
    NEXT_STATE2<= RESULT;

when RESULT =>
    D_REG <=  DATA_BUS;
    CTRL_BUS <= cSTOP;
    NEXT_STATE2<= SAVE;

when SAVE =>
    if TWO_CNT = "000" then
        D_REG1 <= D_REG;
        TWO_KA <= MEM(2);
        TWO_KC <= MEM(5);
        TWO_CNT <= "001";
        NEXT_STATE2 <= RESET;

    elsif TWO_CNT = "001" then
        D_REG2 <= D_REG;
        TWO_KA <= MEM(1);
        TWO_KB <= MEM(6);
        TWO_KC <= MEM(7);
        TWO_CNT <= "010";
        NEXT_STATE2 <= RESET;

    elsif TWO_CNT = "010" then
        D_REG3 <= D_REG;
        TWO_KA <= MEM(2);
        TWO_KC <= MEM(8);
        TWO_CNT <= "011";
        NEXT_STATE2 <= RESET;

    elsif TWO_CNT = "011" then
        D_REG4 <= D_REG;
        TWO_CNT <= "100";
```

```
                NEXT_STATE2<= ARRAY1;

            elsif TWO_CNT = "100" then
                D_REG1 <= D_REG;
                NEXT_STATE2<= ARRAY2;
            end if;

        when ARRAY1 =>
            TWO_END1 <= '1';
            NEXT_STATE2<= WAIT5;

        when WAIT5 =>
            TWO_END1 <= '0';
            NEXT_STATE2<= DIV_BK;

        when DIV_BK =>
            DIV_STR <= '1';
            NEXT_STATE2<= WAIT6;

        when WAIT6 =>
            DIV_STR <= '0';
            NEXT_STATE2<= STOP;

        when ARRAY2 =>
            TWO_END2 <= '1';
            NEXT_STATE2<= WAIT52;

        when WAIT52 =>
            TWO_END2 <= '0';
            NEXT_STATE2<= STOP;

        when STOP =>
            NEXT_STATE2 <= STOP;

        when others =>
            NEXT_STATE2 <= STOP;
    end case;
  end if;

  if BL(0) = '1' then
      TWO_CNT <= "000";
      TWO_CNT2 <= '0';
      TWO_END1 <= '0';
      TWO_END2 <= '0';
  end if;

end process;

----------計算スタート制御
process (CLK,BL(0),BL(2))begin
  if BL(0) = '1' then
      CURRENT_STATE1 <= DSTOP;
      CURRENT_STATE2 <= STOP;

  elsif rising_edge(CLK) then
      if BL(2) = '1' then
          CURRENT_STATE1 <= DSTART;
          CURRENT_STATE2 <= NEXT_STATE2;

      elsif DIV_STR = '1' then
          CURRENT_STATE1 <= DSTART2;
          CURRENT_STATE2 <= NEXT_STATE2;

      elsif TWO_STR = '1' then
          CURRENT_STATE1 <= NEXT_STATE1;
          CURRENT_STATE2 <= START;

      elsif TWO_STR2 = '1' then
          CURRENT_STATE1 <= NEXT_STATE1;
          CURRENT_STATE2 <= START2;

      else
          CURRENT_STATE1 <= NEXT_STATE1;
          CURRENT_STATE2 <= NEXT_STATE2;
      end if;
  end if;
```

```
end process;

-------データ出力
process (IBF, BL(0),BL(7),OUT_CNT) begin
  if BL(0) = '1' then
      OUT_CNT <= '0';
      A_REG2 <= "0000000000000000";
      STB_BUF <= "11";
      READ_DATA_ACTIVE <= '0';

  elsif falling_edge(BL(7)) then
      if OUT_CNT = '0' then
          OUT_CNT <= '1';
          STB_BUF <= "00";
          A_REG2 <= READ_DATA_REG_L (15 downto 0);

      else
          OUT_CNT <= '0';
          STB_BUF <= "00";
          A_REG2 <= READ_DATA_REG_L (31 downto 16);
      end if;
  end if;

  if IBF = "11" then
      STB_BUF <= "00";
      READ_DATA_ACTIVE <= '1';
  end if;
end process;

-------出力データ制御
process (BL(1),BL(0)) begin
  if rising_edge(BL(1)) then
      READ_DATA_REG_L <= MEM(CNT2);
      CNT2 <= CNT2 + 1;
  end if;

  if BL(0) = '1' then
      CNT2 <= 0;
  end if;
end process;
end RTL;
```

A.5 メインプログラム(2ndFPGA)(u02koba/programs/VHDL/fpga2.vhd)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

library metamor;
use metamor.attributes.all;
entity second is

port (
       CLK       : in    std_logic;
       DATA_BUS  : inout std_logic_vector(31 downto 0);
       ADRS_BUS  : in    std_logic_vector(16 downto 0);
       CTRL_BUS  : in    std_logic_vector(4 downto 0);
       CALC_DONE : out   std_logic;
       OE_ALU    : in    std_logic
      );

attribute pinnum of CLK       : signal is "AY22";
attribute pinnum of DATA_BUS  : signal is "BC5,BB6,BC7,BB8,BC9,BB10,BC11,BB12,
                                           BC13,BB14,BC15,BB16,BC17,BB18,BC19,
                                           BB20,BC23,BB24,BC25,BB26,BC27,BB28,
                                           BC29,BB30,BC31,BB32,BC33,BB34,BC35,
                                           BB36,BC37,BB38";
attribute pinnum of ADRS_BUS  : signal is "AU15,AV18,AU19,AV20,AU21,AV22,AU23,
                                           AV24,AU25,AV28,AU29,AV30,AU31,AV32,
                                           AU33,AV34,AU35";
```

```
attribute pinnum of CTRL_BUS  : signal is "AV10,AU11,AV12,AU13,AV14";
attribute pinnum of CALC_DONE : signal is "AU9";
attribute pinnum of OE_ALU    : signal is "AV8";

end second;
architecture RTL of second is

component fpmul is
port(
        CLK : in  std_logic;
        FA  : in  std_logic_vector(31 downto 0);
        FB  : in  std_logic_vector(31 downto 0);
        Q   : out std_logic_vector(31 downto 0)
    );
end component;

component fpsub is
port(
        CLK  : in std_logic;
        FA   : in std_logic_vector(31 downto 0);
        FB   : in std_logic_vector(31 downto 0);
        Q    : out std_logic_vector(31 downto 0)
     );
end component;

signal MUL_FA        : std_logic_vector (31 downto 0);
signal MUL_FB        : std_logic_vector (31 downto 0);
signal MUL_Q         : std_logic_vector (31 downto 0);

signal SUB_FA        : std_logic_vector (31 downto 0);
signal SUB_FB        : std_logic_vector (31 downto 0);
signal SUB_Q         : std_logic_vector (31 downto 0);

signal DATA_BUS_BUF  : std_logic_vector (31 downto 0);
signal DATA_BUS_REG1 : std_logic_vector (31 downto 0);
signal DATA_BUS_REG2 : std_logic_vector (31 downto 0);
signal DATA_BUS_REG3 : std_logic_vector (31 downto 0);

constant cRESET     :std_logic_vector:= "00000";
constant cMEM_1     :std_logic_vector:= "00001";
constant cMEM_2     :std_logic_vector:= "00010";
constant cMEM_3     :std_logic_vector:= "00011";
constant cCLK_STR   :std_logic_vector:= "00100";
constant cKEKKA_OUT :std_logic_vector:= "01000";
constant cSTOP      :std_logic_vector:= "10000";

type STATE_TYPE2 is
     (STOP,KEKKA_OUT,FSUB,QSUB,FMUL,QMUL,WAIT1,WAIT2,WAIT3,WAIT4,WAIT5,
      WAIT6,WAIT7,START,RESET);
signal CURRENT_STATE2 : STATE_TYPE2;
signal NEXT_STATE2 : STATE_TYPE2;

signal sagua             : std_logic_vector (31 downto 0);
signal CTRL_BUS_BUF      : std_logic_vector ( 4 downto 0);
signal DATA_BUS_REG      : std_logic_vector (31 downto 0);
signal WRITE_DATA_ACTIVE : std_logic;
signal CS                : std_logic;
signal CS2               : std_logic;

begin

multiplier : fpmul  port map (CLK, FA=>MUL_FA, FB=>MUL_FB, Q=>MUL_Q);
subtract   : fpsub  port map (CLK, FA=>SUB_FA, FB=>SUB_FB, Q=>SUB_Q);

CTRL_BUS_BUF <= CTRL_BUS;

DATA_BUS <= "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ" when OE_ALU = '0' else DATA_BUS_BUF;

process begin
  if CTRL_BUS = cRESET then
      SAGUA <= (others => '0');
```

```
  elsif OE_ALU= '0' then
      SAGUA <= DATA_BUS;

  end if;
end process;

process(CTRL_BUS_BUF,CLK) begin
  if rising_edge(CLK)  then
      case CTRL_BUS_BUF is
          when cRESET =>
              DATA_BUS_BUF <= (others => '0');
              DATA_BUS_REG1 <= (others => '0');
              DATA_BUS_REG2 <= (others => '0');
              DATA_BUS_REG3 <= (others => '0');
              CS <= '0';

          when  cMEM_1 =>
              DATA_BUS_REG1 <= SAGUA;

          when  cMEM_2 =>
              DATA_BUS_REG2 <= SAGUA;

          when  cMEM_3 =>
              DATA_BUS_REG3 <= SAGUA;

          when  cCLK_STR =>
                CS <= '1';

          when cKEKKA_OUT =>
              DATA_BUS_BUF <= SUB_Q;

          when cSTOP =>
              NULL;

          when others =>
              NULL;
      end case;
  end if;
end process;

process (CLK)begin
  if rising_edge(CLK) then
      if CTRL_BUS = cRESET then
          CALC_DONE <= '0';

      elsif  WRITE_DATA_ACTIVE = '1' then
          CALC_DONE <= '1';
      end if;
  end if;
end process;

process (CLK) begin
  if falling_edge(CLK) then
      case CURRENT_STATE2 is
          when START =>
              NEXT_STATE2 <= FMUL;

          when FMUL =>
              MUL_FA <= DATA_BUS_REG1;
              MUL_FB <= DATA_BUS_REG2;
              NEXT_STATE2 <= WAIT1;

          when WAIT1 =>
              NEXT_STATE2 <= FSUB;

          when FSUB =>
              SUB_FA <= DATA_BUS_REG3;
              SUB_FB <= MUL_Q;
              NEXT_STATE2 <= WAIT2;

          when WAIT2 =>
              NEXT_STATE2 <= WAIT3;

          when WAIT3 =>
              NEXT_STATE2 <= QSUB;
```

```
          when QSUB =>
              WRITE_DATA_ACTIVE <= '1';
              NEXT_STATE2 <= STOP;

          when STOP =>
              WRITE_DATA_ACTIVE <= '0';
              NEXT_STATE2 <= STOP;

          when others =>
              NEXT_STATE2 <= STOP;

      end case;
  end if;

  if CTRL_BUS = cRESET then
      WRITE_DATA_ACTIVE <= '0';
  end if;

end process;

process(CLK,CS) begin
  if rising_edge(CLK) then
      if CS = '1'then
          if CS2 = '0'then
              CURRENT_STATE2 <= START;
              CS2 <= '1';

          elsif CTRL_BUS = cRESET then
              CURRENT_STATE2 <= STOP;
              CS2 <='0';

          else
              CURRENT_STATE2 <= NEXT_STATE2;
          end if;
      end if;
  end if;
end process;
end RTL;
```

付 録B　回路設計をする上でのソフトウェアの使用法

ここでは，回路設計をする上でのソフトウェアの使用法して，Accolade 社の PeakFPGA と Altera 社の Max+PLUSII の使用方法と，FPGA のダウンロード方法について説明する．

B.1　PeakFPGA

B.1.1　VHDL ファイル作製における注意点

この研究で PeakFPGA を使用する場合，VHDL ファイルの作製方法について説明する．
まず最初に，

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

library metamor;
use metamor.attributes.all;
```

を記述し，VHDL のライブラリを呼び出す．また，metamor のライブラリを呼び出すときには，PeakFPGA で Synopsys library を呼び出すようにしなければならない．

そして，次の entity という部分で，FPGA の入出力ポートとピンの設定をする．

```
entity count is
    port ( CLK : in std_logic;
           A   : inout std_logic_vector(15 downto 0);
           BL  : in std_logic_vector(7 downto 0);
           BH  : out std_logic_vector(7 downto 0);
           CL  : in std_logic   );

attribute pinnum of CLK : signal is ‘‘D22’’;
attribute pinnum of A   : signal is ‘‘BC23,BB24,BC25,BB26,BC27,BB28,BC29,BB30, ..
attribute pinnum of BL  : signal is ‘‘BC13,BB14,BC15,BB16,BC17,BB18,BC19,BB20’’;
attribute pinnum of BH  : signal is ‘‘BC5,BB6,BC7,BB8,BC9,BB10,BC11,BB12’’;
attribute pinnum of CL  : signal is ‘‘AU23’’;

end count;
```

まず，entity 文でモジュール名を指定する．これは FPGA にダウンロードするときのファイル名にもなるので，長い名前はつけない方が良い．そして，port 文で FPGA のポートを指定する．FPGA にデータを入力する場合には in，データを出力する場合には out，両方の場合には inout を指定する．

そして attribute 文でピン番号を指定する．

```
architecture RTL of count is

signal CLK_CNT : std_logic_vector(7 downto 0);
signal RESET   : std_logic;

begin

end RTL;
```

次に architecture 文で回路の設計要素を記述する．count の部分は entity の名前と同じにする．begin 文の前で，内部信号 signal を宣言し，begin 文以降は回路の動作を記述する．

B.1.2 VHDL ファイルの作製

まず，VHDL ファイルの作製方法について説明する．

- 最初に PeakFPGA を起動する．

図 B.1: PeakFPGA を開く [21]

- “File” にある “New Project” を選択．

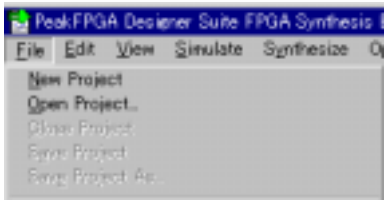

図 B.2: “メニュー” の “ファイル” を選択 [22]

- 下のウインドウが表われたら，右クリックを押して”Add Module” を選択，ダイアログボックスが表われたら，”The Project has not been saved. Save it now?” と出てくるので，”OK” を押す，するとダイアログボックスが表われるので，ACC(*.acc) ファイルに名前をつけて保存する．この ACC ファイルはプロジェクトのファイルなので，例えば内積のプログラムを作る時は naiseki1.acc 等，分かりやすい名前を付けることが必要である．

[21] u02koba/eps/peakfpga.eps
[22] u02koba/eps/select.eps

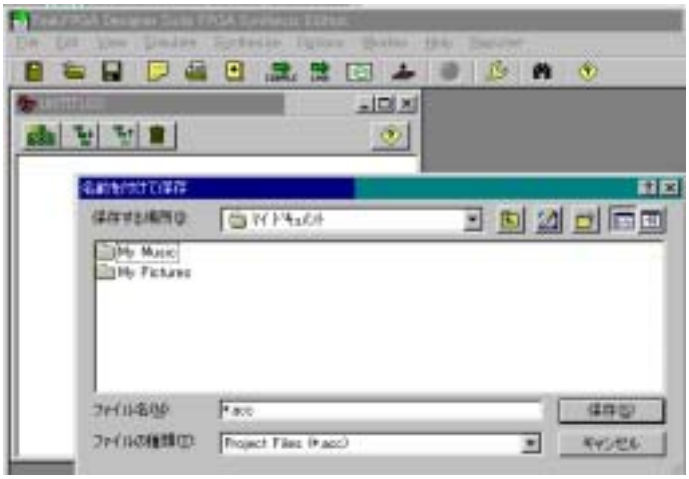

図 B.3: “開くを選択”[23]

- すると，今度は “開く” というダイアログボックスが表われるが，まだ VHDL ソースファイルを作っていないのでキャンセルを押す．そして，再び “File” を選択し，その中の “New Module” を選ぶ．すると，図 B.4 が現れ，その中の “Cleate Blank Module” を選択する．すると名前を求めてくるので，そこにモジュール名を記述する．これによって VHDL が記述出来る VHD(*.vhd) ファイルが作製される．再び追加したい時も，同じ事を繰り返せば良い．

図 B.4: “NewModule” を選択 [24]

- VHDL で記述した VHDL ファイルを構文解析するには，上部のメニューにある”COMPILE” というボタンを押す．するとコンパイルが始まり，中央にウインドウが表われる．構文に間違いがあったらここにエラーの原因とその行が示されるので，先の VHDL が書かれているウインドウから間違いを直す．

[23] u02koba/eps/open.eps
[24] u02koba/eps/newmodule.eps

- コンパイルが正しければ，次は論理合成を行なう．メニューの”Option” から”Synthesize” を選ぶ．すると，下のウインドウが表われる．ここで，右側にある”Device Family” から”Altera all Devices (EDIF)” を選択する．そして，左下にある”Include Synopsys Library” をチェックする．

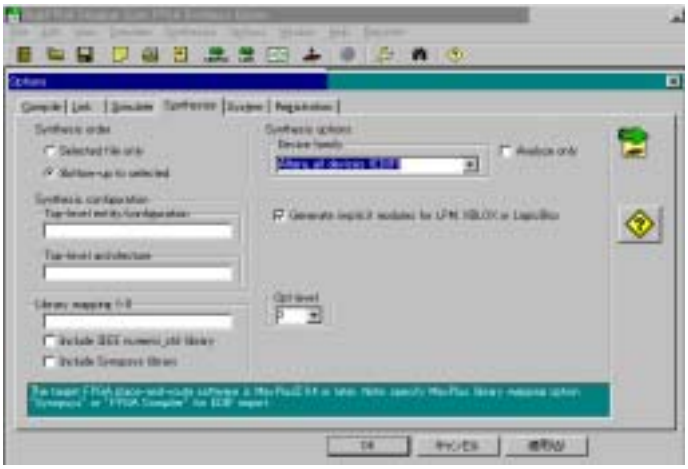

図 B.5: デバイスの選択 [25]

- そして，上部のデバイスマーク，もしくは “メニュー” の “Synthesize” を選ぶ．すると，別のウインドウで論理合成が行なわれる．論理合成が正しく行なわれたならば，モジュール名のついた EDF ファイルが作成される．

図 B.6: 論理合成 [26]

- このプロジェクトファイルの中で，VHDL ファイルを 2 つ以上合成して，ひとつのモジュールを作ることができる．このとき，それぞれのファイルは component または function で

[25] u02koba/eps/syn.eps
[26] u02koba/eps/ronri.eps

ファイルがリンクできていなければならない．そのリンクができているかを確かめるには，“Rebuild Hierarchy” というボタンを押して，リンクし直し，“Show Hieralchy” というボタンを押して component や function ができているかを確かめる．VHDL ファイルを手直しした時は必ず “Rebuild Hierarchy” を押すことを心掛けた方が良い．

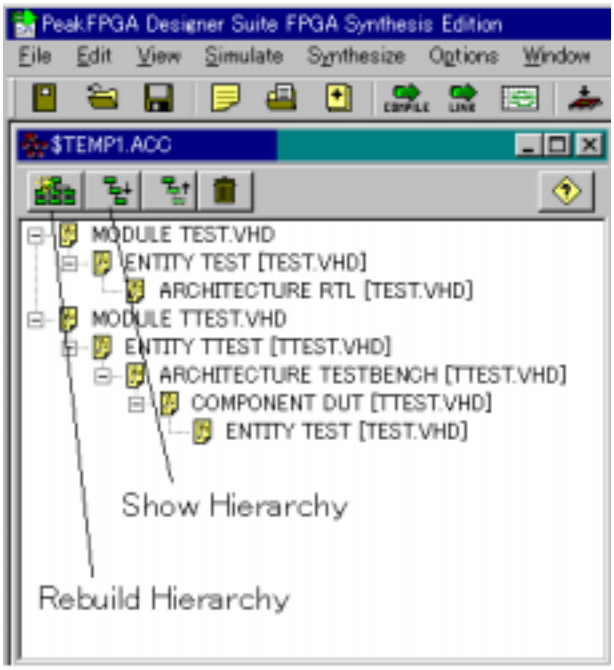

図 B.7: リンクの検査 [27]

B.1.3 VHDL で記述する時の注意点

ここでは VHDL を使う時の注意点を説明する．

std_logic を使った計算

例えば
signal A : std_logic_vector(16 downto 0);
で定義した信号を作る．これに CNT という信号が ‘1‘ の時に ”00000000000000001” を足すときには，
if CNT = ’1’ then
A = A + “00000000000000001”;
end if;

[27] u02koba/eps/link.eps

と書きたくなるが，これでは間違いである．CNT が ‘1‘ の時はプログラム上ではある 1 点かも知れないが，ハードウェア上では ‘1‘ になった状態の連続と見なされる．よって正しい結果は得られない．そこで CNT が ‘1‘ になった瞬間にある結果を実行させるという風に書かなければならない．

```
if rising_edge(CNT) then
A = A + “0000000000000001”;
end  if;
```

CNT が立ち上がる事はハードウェア上では 1 点しかかないのでこれで正しい動作が行われる．

信号の定義

これは std_logic_vector で定義した信号の計算をするとき，信号の幅を 32bit 以上で定義した信号の計算は Max+Plus 2 のコンパイルエラーを生じる．よって先ほど述べた用な信号の計算を行う場合は 31bit 幅で定義しなければならない．

```
signal A : std_logic_vector(31 downto 0);
```

B.1.4 PeakFPGA でのシミュレーション方法

シミュレーションはプロジェクトウインドウにテストベンチ用の VHDL ファイルを挿入することでできる．その VHDL ソースには，テストベンチの対象となるソースと同じ port を宣言し，それと同じ signal も宣言する．そして，その port に使われている信号を並べた DUT 文を begin 文の後に挿入する．proecss には入力ポートにたいする信号を記述する．このテストベンチ用の VHDL ファイルもテストするファイルにリンクしているので，必ず “Rebuild Hierarchy” を押してリンクさせる必要がある．例として，シミュレーションをする VHDL ソースとシミュレーションの VHDL ソースを以下に示す．このファイルは BL ポートの入力信号により出力信号の A,BH 信号を変化させる VHDL ソースである．

```
-----------------
-- ソースファイル
-----------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

library metamor;
use metamor.attributes.all;
 entity test is

  port (
         CLK : in std_logic;--クロック
         A   : inout std_logic_vector (15 downto 0);--A レジスタ（2 方向）
         BL  : in std_logic_vector ( 7 downto 0);--B レジスタ（入力）
         BH  : out std_logic_vector ( 7 downto 0 )--B レジスタ（出力）


 );

  attribute pinnum of A   : signal is  "BC23,BB24,BC25,BB26,BC27,BB28,BC29,BB30,BC31,
                                        BB32,BC33,BB34,BC35,BB36,BC37,BB38";
  attribute pinnum of BL   : signal is  "BC13,BB14,BC15,BB16,BC17,BB18,BC19,BB20";
  attribute pinnum of BH   : signal is  "BC5,BB6,BC7,BB8,BC9,BB10,BC11,BB12";
  attribute pinnum of CLK : signal is  "D22";
end test;

architecture RTL of test is


begin
 process (BL,CLK) begin
 if falling_edge(CLK) then
   case BL is--BL 信号の場合別け
        when "00000000" =>-- BL が’’00000000’’ の時
             A <= "0000000000000000";--A に’’0000000000000000’’
を出力
             BH <= "00000000";--BH に’’00000000’’ を出力
        when "00000001" =>-- BL が’’00000001’’ の時
             A <= "0000000000000001";--A に’’0000000000000001’’
を出力
             BH <= "00000000";--BH に’’00000000’’ を出力
        when "00000010" =>-- BL が’’00000010’’ の時
             A <= "0000000000000010";--A に’’0000000000000010’’
を出力
             BH <= "00000000";--BH に’’00000000’’ を出力
        when "00000011" =>-- BL が’’00000011’’ の時
             A <= "0000000000000011";--A に’’0000000000000011’’
を出力
             BH <= "00000000";--BH に’’00000000’’ を出力
        when others =>-- BL がその他の時
```

```
            BH <= "11111111";--BH に’’11111111’’ を出力
    end case;
  end if;
  end process;
end RTL;


-----------------
--シミュレートファイル
-----------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

library metamor;
use metamor.attributes.all;
 entity ttest is
 end ttest;

architecture TESTBENCH of ttest is
 component test is
  port(
         CLK : in std_logic;--クロック
         A   : inout std_logic_vector (15 downto 0);--A レジスタ（2 方向）
         BL  : in std_logic_vector ( 7 downto 0);--B レジスタ（入力）
         BH  : out std_logic_vector ( 7 downto 0 )--B レジスタ（出力）
  );
 end component;


  signal CLK : std_logic;--クロック
  signal A   : std_logic_vector (15 downto 0);--A レジスタ（2 方向）
  signal BL  : std_logic_vector ( 7 downto 0);--B レジスタ（入力）
  signal BH  : std_logic_vector ( 7 downto 0 );--B レジスタ
（出力）

begin

  DUT : test port map (CLK,A,BL,BH);-- 上で宣言した通りにポー
トマップを宣言
process begin-- クロックの作製，周期は 20ns
  CLK <= ’0’;
 wait for 10 ns;--Low は 10ns ずつ
  CLK <= ’1’;
 wait for 10 ns;--High は 10ns ずつ
end process;

process begin --以下にシミュレート内容を記述
  wait for 100 ns;
    BL <= "00000000";
  wait for 100 ns;
    BL <= "00000001";
  wait for 100 ns;
    BL <= "00000010";
  wait for 100 ns;
    BL <= "00000011";
  wait for 100 ns;
    BL <= "11111111";
  wait for 1000 ns;
end process;

end TESTBENCH;
```

シミュレーションの仕方について説明する．まず，先に述べたプロジェクトウインドウにテストベンチの VHDL ソースを追加する．そして，Rebulid したあとに，テストベンチのファイルを選択して上部のメニューの “Simulation” から”Load Selected” を選ぶ．すると，構文解析と同じウインドウが表われ，コンパイルが始まる．コンパイルが成功すると，図 B.8 のようなウインドウが表われる．

図 B.8: 信号の選択画面 [28]

図 B.8 はシミューレートしたい信号を全体の信号から選択するウィンドウである．図 B.8 のウインドウが表われたら，シミュレーションしたい内部信号を選択し，”close” を押す．そして，“メニュー” にあるスパナのアイコンを押して，シミュレーションする動作時間を決める．それを決めたら，“メニュー” のボタンのような “Go” アイコンを押して，シミュレーションを開始する．シミュレーションに成功すると，図 B.9 の画面となる．図 B.9 より実際のソースファイルの通りに動作している事が分かる．

図 B.9: シミュレーション成功 [29]

[28] u02koba/eps/simyu_face.eps
[29] u02koba/eps/logic.eps

B.2 Max+PLUS2

論理合成が終わったら，次はFPGAに搭載できるファイルに置き換えなければならない．ここではその方法について説明する．まず，Max+PLUS2を起動する．そして，“File”の“Open”で論理合成したEDFファイルを選択する．

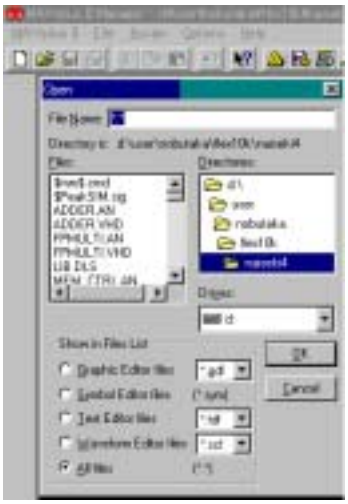

図 B.10: Max+Plus2 オープニング [30]

次に“Assign”の“Device”を選択し，ダウンロードするFPGAを選択する．本研究では，EPF10KGC503-4を使用しているので，それを選択する．そして“’Devaice Options”を選び，チェックをすべて外す．

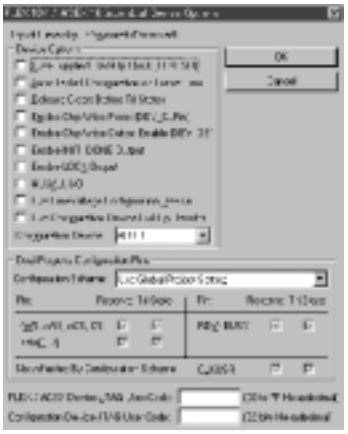

図 B.11: オプション画面 [31]

[30] u02koba/eps/max_open.eps
[31] u02koba/eps/max_edit.eps

次に，メニューの “MAX+plusII から “Compiler” を選択する．すると次のような画面が出てくるので，“start” ボタンを押す．

図 B.12: コンパイル開始[32]

コンパイルが終わると，TTF ファイルが作成されるので，あとはこれを FPGA にダウンロードすればよい．

[32] u02koba/eps/max_com.eps

B.3　FPGA へのコンフィグレーション

まず DOS プロンプトを開き作成した TTF ファイルのあるフォルダに移る．もし C ドライブの中の FLEX10K フォルダの naiseki フォルダの中に TTF があるなら，

cd c:\FLEX10K\naiseki(バックスラッシュは円記号の事)

という様に入力します．そして目的のフォルダに移ったら，

flex10k ***.ttf(*** は TTF ファイルの名前)

これでコンフィグレーションが開始されます．

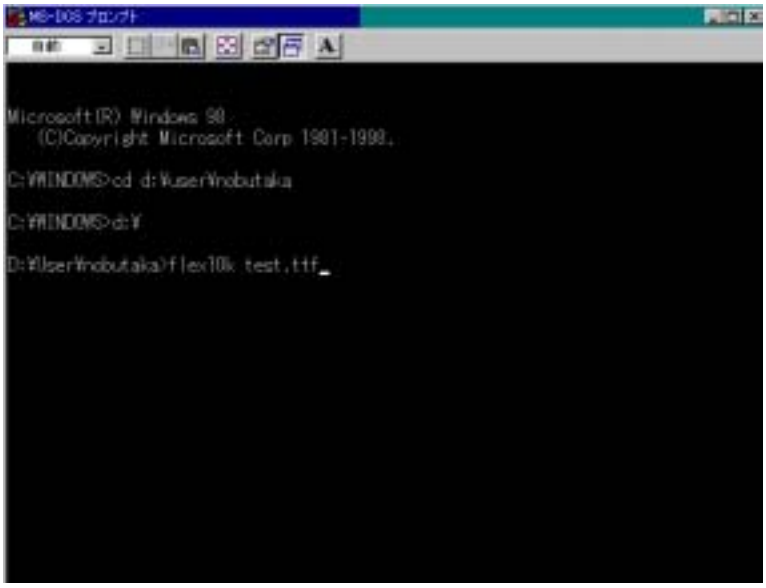

図 B.13: コンフィグレーション [33]

[33] u02koba/eps/dos.eps

付 録C　本研究に置けるC++Builderの使い方

C.1　使用目的

本研究において評価ボードを使用するにあたって，FPGA を外部から制御する必要がある．通信は松尾 [3] が作製した PPI8255 を 2 個搭載した，ISA バス専用の通信ボードを用いる (詳しくは [3] 参照)．評価基板の制御は C++Builder を使って行う．

C.2　使用方法

C.2.1　ヘッダーファイルの設定

まず C++Builder の自分の作業フォルダーに以下のファイルを入れる．

BASE8255.h
PORT95.dcu
PORT95.hpp
PORT95.OBJ
PORT95.PAS

BASE8255.h とはヘッダファイルであり，この中に基板を制御する為のポートが宣言されている．PORT95.hpp の中に実際の基板制御関数を宣言しているのでこの 2 つのファイルを読み込まなければ実際に基板を制御出来ない．他のファイルは基板制御関数の実際の動作をパスカル等記述している．

図 C.1: C++BUilder のプロジェクトマネージャー画面[34]

次に C++Builder を起動させて，C++Builder のメニューのプロジェクトマネージャーを開き図 C.1 の様に***.EXE に PORT95.OBJ を追加する．
最後にメインのソース，***.cpp に以下をヘッダーファイルとして追加する．

BASE8255.h
PORT95.hpp

これで C++Builder 上で制御命令文の使用が可能となる．

C.2.2 制御命令文

前節で行った設定で以下の命令が使用出来る

PortWiteByte(ポート名, 書込む数値)
PortReadByte(ポート名)
PortWiteWord(ポート名, 書込む数値)
PortReadWord(ポート名)

Byte は 8bit，Word は 16bit のことである．

ポート名　ポート名には A(A_H,A_L),B_H,B_L,CTRL_H,CTRL_L がある．詳しくは巻末の PPI8255 の動作表参照．

- A(A_H,A_L)
  A は入出力両方に使用する．16bit
- B_H
  B_H は評価基板から PC への入力とする．8bit
- B_L
  B_L は評価基板への PC からの出力とする．8bit
- CTRL_H
  A,B 両方の High ポートのコントロールワード設定．8bit

[34] u02koba/eps/C++_face.eps

- CTRL_L
  A,B 両方の Low ポートのコントロールワード設定．8bit

書込む数値　数値は16進数で表記する．例えばB_Lに “00000010” を出力したい場合はPortWrite-Byte(B_L,0X02) と書く．‘0X’ とは 16 進数という意味である．

C.2.3　整数型と単精度浮動小数点

PC と評価基板との通信にはA ポートを使っての16bit が最高の幅を持つ伝送路になる．C++Builder でこの研究に使う主な数値の型には int 型 (整数型) と float 型 (単精度浮動小数点) がありどちらも 32bit である．よって伝送路の幅を通る為には，16bit ずつ別けて通す必要がある．

- 32bit から 16bit への変換
  C のポインタを使う．
  float data1;
  data1 を float 型で定義
  ddata1= (unsigned short&*)data1;
  data1 に入っているデータがある場所のアドレスを ddata1 に返す．
  recordl1 =*ddata1;
  ddata1 に入っているアドレスが指す場所のデータを record1l に入れる．つまり 32bit の内，下位 16bit を入れる．
  recordh1 =*(ddata1+1);
  (ddata1+1) に入っているアドレスが指す場所のデータを record1l に入れる．つまり 32bit の内，上位 16bit を入れる．

図 C.2: ポインタを使った変換[35]

- 16bit から 32bit への変換
  unsigned short res[1];

[35]u02koba/eps/divide.eps

res[0],res[1] の 2 つの配列を unsigned short 型（16bit) で定義
res[0]=PortReadWord(P_A);
res[1]=PortReadWord(P_A);
2 つの配列に分割した 16bit のデータを入力
Memo3-Lines-Add("FPGA="+FloatToStr(*(float*)res));
res に入ったデータを 1 つにつなぎ出力

- 単精度浮動少数点
  単精度浮動小数点は 32bit である.

図 C.3: 単精度浮動小数点 [36]

32bit の内訳は

  - 上位 1bit
    1 の時正の数を表し 0 の時負を表す.

  - 次の 8bit
    指数部を表す. この 8bit から offset と呼ばれる 8bit の数 '01111111' を引いた物が真の指数部となる.

  - 残り 23bit
    仮数部を表す. ここには暗黙の 1 と呼ばれる物が付いており, この 1 に 23bit の内, 左から 2^{-1}, $2^{(-2)}$, $2^{(-3)}$.....$2^{(-23)}$ と, 重みが付いている数値を足して仮数部としている.
    例を挙げると

    '11000000111000000000000000000000'

    の浮動小数点型は

[36] u02koba/eps/float.eps

$$\mathbf{(-1)*2^{(129-offset(127))}*(1+(1*2^{-1})+(1*2^{(-2)}))=-7}$$

と言うことになる．0 を表す時は特別で，32bit 全て 0 にすればいい．これは厳密には 0 ではないが，形式的に 0 になる．

- 整数型整数型は単なる 2 進数表記で表し，負の数は補数で表現する．

付 録D　C++Builder のソースプログラム

D.1　FPGA 計算用プログラム (u02koba/programs/C++/LU-FPGA.cpp)

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "flaot_adder.h"
#include"BASE8255.h"
#include"port95.hpp"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
 PortWriteWord(P_CTRL_L,0xc0);
 PortWriteWord(P_CTRL_H,0xc2);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  PortWriteByte(P_BL,0x01);
  PortWriteByte(P_BL,0x00);

  data1=float(random(100))+float(random(50))/(float(random(100))+1);
  Edit1->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);


  data1= float(random(100))+float(random(50))/(float(random(100))+1);
  Edit2->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

  data1= float(random(100))+float(random(50))/(float(random(100))+1);
  Edit3->Text=FloatToStrF(data1,ffGeneral,7,0);
```

```
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

  data1 = float(random(100))+float(random(50))/(float(random(100))+1);
  Edit4->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

  data1=float(random(100))+float(random(50))/(float(random(100))+1);
  Edit5->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);


  data1= float(random(100))+float(random(50))/(float(random(100))+1);
  Edit6->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

  data1= float(random(100))+float(random(50))/(float(random(100))+1);
  Edit7->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

  data1 = float(random(100))+float(random(50))/(float(random(100))+1);
  Edit8->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

  data1= float(random(100))+float(random(50))/(float(random(100))+1);
  Edit9->Text=FloatToStrF(data1,ffGeneral,7,0);
  ddata1= (unsigned short *)&data1;
  recordl1 =*ddata1;
  recordh1 =*(ddata1+1);
  PortWriteByte(P_BL,0x08);
  PortWriteWord(P_A,recordl1);
  PortWriteWord(P_A,recordh1);
  PortWriteByte(P_BL,0x00);

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button3Click(TObject *Sender)
```

```
{
  unsigned short res[2];

  PortWriteByte(P_BL,0x01); //始める時は stop
  PortWriteByte(P_BL,0x00);
  PortWriteByte(P_BL,0x04); //計算開始  start
  PortWriteByte(P_BL,0x00);
  long tick=GetTickCount();
    while(GetTickCount()-tick < 100);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit10->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit11->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit12->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit13->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit14->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
```

```
   res[1]=PortReadWord(P_A);
        Edit15->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit16->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit17->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

   PortWriteByte(P_BL,0x02);
   PortWriteByte(P_BL,0x00);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[0]=PortReadWord(P_A);
   PortWriteByte(P_BL,0x80);
   PortWriteByte(P_BL,0x00);
   res[1]=PortReadWord(P_A);
        Edit18->Text=FloatToStrF(*(float*)res,ffGeneral,7,0);

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button5Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
system("flex10k2 second.ttf");

system("flex10k1 first.ttf");

}
//---------------------------------------------------------------------------
```

D.2 LU分解プログラム (u02koba/programs/C++/LU-com.cpp)

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include <stdio.h>
#include "flaot_adder.h"
#include"BASE8255.h"
#include"port95.hpp"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
 PortWriteWord(P_CTRL_L,0xc0);
 PortWriteWord(P_CTRL_H,0xc2);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{

 data[0][0]= random(100);
 Edit1->Text=FloatToStr(data[0][0]);
 data[0][1]= random(100);
 Edit2->Text=FloatToStr(data[0][1]);
 data[0][2]= random(100);
 Edit3->Text=FloatToStr(data[0][2]);
 data[1][0]= random(100);
 Edit4->Text=FloatToStr(data[1][0]);
 data[1][1]= random(100);
 Edit5->Text=FloatToStr(data[1][1]);
 data[1][2]= random(100);
 Edit6->Text=FloatToStr(data[1][2]);
 data[2][0]= random(100);
 Edit7->Text=FloatToStr(data[2][0]);
 data[2][1]= random(100);
 Edit8->Text=FloatToStr(data[2][1]);
 data[2][2]= random(100);
 Edit9->Text=FloatToStr(data[2][2]);

  int i,j,k,n;
  n = 3;

  for(i = 0; i < n; i++){
      for(j = i +1 ; j < n; j++){
          data[j][i] /= data[i][i];
          for(k = i +1; k < n; k++){
              data[j][k] -= data[i][k] *data[j][i];
          }
      }
  }

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button3Click(TObject *Sender)
{

        Edit10->Text=FloatToStrF(data[0][0],ffGeneral,7,0);
        Edit11->Text=FloatToStrF(data[0][1],ffGeneral,7,0);
        Edit12->Text=FloatToStrF(data[0][2],ffGeneral,7,0);
        Edit13->Text=FloatToStrF(data[1][0],ffGeneral,7,0);
        Edit14->Text=FloatToStrF(data[1][1],ffGeneral,7,0);
        Edit15->Text=FloatToStrF(data[1][2],ffGeneral,7,0);
        Edit16->Text=FloatToStrF(data[2][0],ffGeneral,7,0);
        Edit17->Text=FloatToStrF(data[2][1],ffGeneral,7,0);
        Edit18->Text=FloatToStrF(data[2][2],ffGeneral,7,0);

}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button5Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

付 録E　PC Anywhereの使用法

E.1　使用目的

PC と評価基板との通信には松尾 [3] が作製した ISA バス専用のインターフェイスを用いているが，最近の PC には ISA バスが搭載されている PC は皆無に等しい．そこで ISA バス搭載 PC を評価基板制御 PC として独立させ，その PC を外部の PC から通信で制御して，コンフィグレーションを行うことにする．

E.1.1　遠隔操作による利点

遠隔操作を行うことによる利点は，コンフィグレーションを複数のマシーンで行うことができる点である．これはホストになった PC に複数のユーザーが遠隔操作を同時に行うことができる．これにより複数の人が評価基板を共有できるということになった．しかし同時にそこにアクセスすることも可能だが，評価基板は同時には使用できないため，注意が必要である．

E.1.2　使用方法

評価基板をコンフィグレーションできる ISA　バス搭載 PC を PC Anywhere の設定画面でネットワークのホストをダブルクリックしてホストに設定する (図 E.1)．またホストのアイコンを右クリックしプロパティの中の設定で起動時にホストに設定する事も可能である．

図 E.1: ホスト設定 [37]

次にリモート制御する PC において図 E.1 のメニューのツールにあるネットワークのオプションを開く (図 E.2)．そこでネットワーク上にある探索するホストのネットワーク上の名前を設定する．ここでは NOBUTAKA，SUGIYAMA というホストを探すと言うことである．ここにホストに設定する PC の名前を入れて図 E.1 のリモート制御を選択して NETWORK をクリックすれば遠隔操作が可能になる．

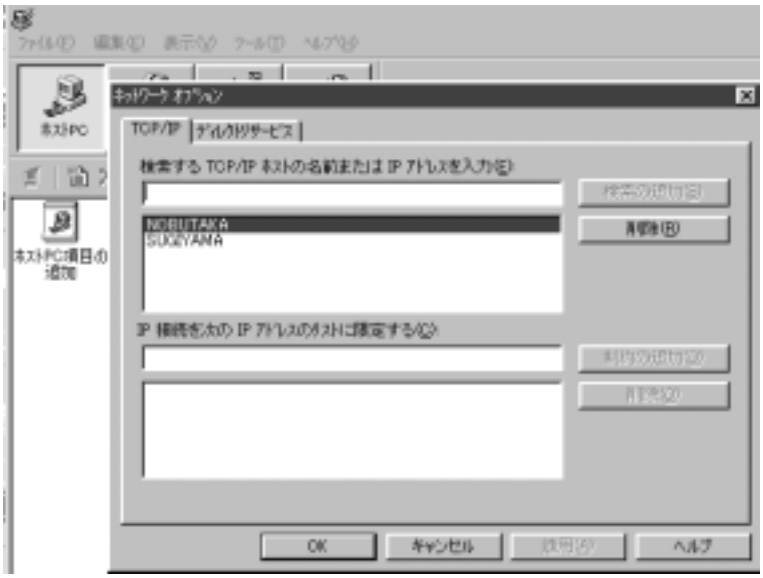

図 E.2: ホスト設定 [38]

[37] u02koba/eps/anyware_face.eps
[38] u02koba/eps/ip_settei.eps