電気通信大学紀要16巻2号 pp.179-184（2004）〔報告〕

電子工学実験におけるFORTHの利用

齋藤正和　藁科崇
高橋光生　田中清臣

FORTH based logic circuits laboratory work

SAITO Masakazu, WARASHINA Takashi, TAKAHASHI Mitsuo, TANAKA Kiyoomi

Abstract

The purpose in the laboratory experiments of logic circuits design, is students to study how to carry out the software design and hardware design according to each function, and evaluate the design in consideration of the trade-off problem.
Therefore, paying attention to the modularity and hierarchy of logic circuits design, students assemble and test logic circuits, and they study design know-how about logic circuits.
As FORTH system has an interactive programming environment and its modularity is suitable to replace hardware function by software function, we adopt FORTH system for logic circuits laboratory work.
In this paper, new experiment environment of logic circuits design using FPGA, and laboratory work tools by FORTH system are presented.

1 はじめに

電子工学実験（今後実験と記述する）における実験課題としてディジタル回路がある。この実験課題では，ソフトウェアとハードウェアによる設計を行い，設計におけるモジュール化や階層化，機能分担のトレードオフ問題について考察する内容となっている[1]。実際の実験では，IC の組み合わせによる論理回路製作を行い，ハードウェアの試験をFORTH システムを利用して行っている。FORTH システムはPLD （Programmable Logic Device ）のプログラミングにも用いている。

実験では実際の設計・製作作業を通して，機能の複雑さに対する設計手法の違いや，製作段階における試験および修正のやり方といったことを学んでもらう。そして動作比較や設計製作の作業量といった観点を考慮してできばえを評価し，トレードオフ問題を考察する。

この論文では，現在計画を進めているディジタル回路実験でのFPGA （Field Programmable Gate Array ）の利用について述べる。また実験支援環境として，Web 上でのテキストの公開やシミュレータの提供などを行っており，コンテンツの一つとしてJava Applet によるFORTH インタプリタを製作して提供している。さらに，FORTH システム上に実現されたツール類について述べる。

2 実験環境と実験内容

2.1 新しい実験環境

FPGA を導入した新しい実験環境について説明する。実験者は従来のワイヤラッピングによる配線ではなく，プログラミングによってFPGA 上に実験回路を製作する。

また，現在利用している論理回路操作盤をハードウェアインタフェースとし，ホストコンピュータ側の各種ツールやソフトウェアインタフェースを使用する。ホストコンピュータのPCI スロットに増設するFPGA 実装基板をFig 1 に示す。

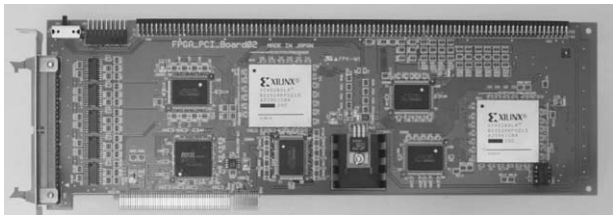

Fig 1: Newly developed PCI board with FPGA

Received on October 20, 2003.
Department of Electronic Engineering.

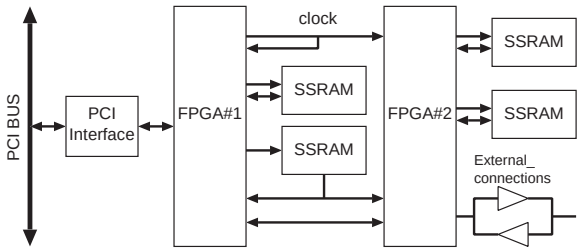

Fig 2: Block diagram of PCI board with FPGA.

このFPGA 実装基板はFPGA とSRAM を2 組実装しており，通常は一方をマイクロプログラム制御方式のFORTH プロセッサとして，もう一方を実験用回路製作用として，マイクロプログラムによって制御可能になっている。場合によっては2つのFPGA にまたがる回路も製作可能であり，基板上に実装されているメモリや外部端子などを用いて，様々な機能回路の製作を可能としている。

従来より使用している論理回路操作盤にはスイッチ・LED といった入出力装置があり，ハードウェアインタフェースとして利用している。現在実験回路の製作は，論理回路操作盤に接続して利用する実験回路組立基板上で行っている。これは基本的なNAND, NOR, NOT, DECODER, ENCODER, COUNTER, SHIFT REGISTOR, ALU, DAC などが実装されているほか，空き領域に新たな論理素子を増設することが可能である。FPGA を導入した新しい実験環境では，実験回路組立基板は新たに表示装置などを追加して利用する。

2.2 実験内容

現在行っている実験の一つとして7 セグメント表示器の課題がある。この課題では7 セグメント表示器の表示方法を確かめ，実際に数値データの表示を行う。

7セグメント表示器で有効な表示結果を得るには，入力データを表示用データに変換する作業が必要である。このデコード処理は一つの完結した機能であり，74LS248（BCD-to-seven-segment decoders/drivers ）やソフトウェアプログラムが一つのモジュールとなる。この部分の処理をソフトウェアとハードウェアで切り替えて実験を行い，モジュール化やトレードオフ問題を考察する。

またデコード処理だけでなく，入力インターフェースやデータ送信経路などを考慮すると，コンピュータ上での数値入力やスイッチによる入力，74LS138 （3-line to 8-line decoders/demultiplexers ）や74LS148 （8-line to 3-line octal priority encoders ）を含んだ送信経路などさまざまな組合せが考えられる（Fig 3 ）。データの扱い易さや，回路の組み立て易さなどを実際に体験して検討や評価を行う。これらの内容は，FPGA を利用した新しい実験環境でも行える。

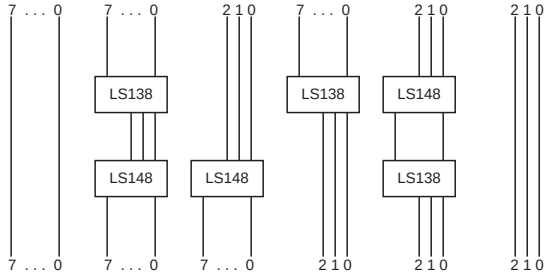

Fig 3: Design of various combination

3 実験におけるFORTH の利用

ここではなぜFORTH を利用するかを説明するとともに，FORTH の応用利用について説明する。

3.1 プログラミング言語として

ディジタル回路実験の目的の一つである機能のモジュール化や階層化といったことを，論理回路・プログラミング両方で共通に適応でき，機能分担を考える際に対応が明確に把握できることが望ましい。

プログラミングに精通している場合は，プロシージャやサブルーチンという単位で機能のモジュール化を考えることが容易である。FORTH ではFORTH ワード（あるいは単にワード）という既存の処理アルゴリズムを組み合わせてプログラミングするので，モジュール化を自然と考えることができる。

さらにFORTH システムの特徴として，コマンドラインより直接入力してワードを実行することができる。これによってテスト用プログラムを用意せずに実行と結果確認が即座に行え，デバッグ作業が簡単に行える。

このようなモジュール化を考えやすいプログラミング手法，インタプリタによる対話的で容易なプログラミング環境などの利点を考慮してFORTH を利用している。

また結果として，FORTH 自身のコンパクトさと柔軟性が多様な利用方法を可能としている。

3.2 実験用ツールとして

プログラミング言語としての利用は先に述べた通りであるが，その他にも実験用のツールとしても利用している。

実験回路の製作において一部PLD を利用しており，PLDへの書き込み，およびJEDEC フォーマットファイル（PLDの機能制御ビットパターン及び、試験パターン）への変換にもFORTH を利用している。ソースファイル内の記述はFORTH プログラミングそのものであり，利用者の目的に応じてコンパイラの変更を行うことができる。

外部機器への入出力処理は，通常ならば入出力処理用プログラミングが必要であるが，実験回路装置との入出力用FORTH ワードを用意しているので，コマンドラインよりこのFORTH ワードを直接実行することによって入出力を行うことができる。また必要に応じて入出力処理を含めてプログラミングすることも可能であり，実験内容に応じて多様な利用ができる。

4 FORTH の概要

現在利用しているFORTH (C 版FORTH)と，新たに整備したFORTH は次の3 つである。

1. C 版FORTH （実験での利用）
2. Java 版FORTH （Web 上での利用）
3. FPGA 版FORTH （FORTH プロセッサ）

FORTH の整備方針は次のとおりである。約80 個の基本ワードを実装環境に合わせて同一仕様に従い作成した。FORTH レベルのプログラミングは3 つのFORTHで共通に利用できる。

ここではC 版・Java 版をあわせてソフトウェア版と呼び，FPGA 版を対としてハードウェア版と呼ぶ。

4.1 ソフトウェア版FORTH

4.1.1 C 版FORTH

現在実験室で利用しているFORTH は1996 年にホストコンピュータに依存しないシステムとするためにC 言語によって移植されたものである[2]。

ファイル入出力やポート入出力の機能が組み込まれており，ユーザは単一のワードで簡単に使用できる。基本ワードなどを組み合わせた起動時に必要となるワードや数値データの集まりをFORTH 辞書と呼んでおり，起動時に読み込むFORTH 辞書はメモリダンプ形式で用意されている。このFORTH 辞書はJava 版・FPGA 版FORTHでも共通に利用できる。

実際のマシン上での処理部分はユーザから隠蔽されており，マシン別による処理の違いを気にする必要はない。そのため，ユーザ自身はターミナル入力やポート入出力などの具体的な処理を知らずとも，すべての機能を利用できる。実験時のプログラミングでは制御シーケンスに関するデータの数値処理と入出力処理だけに専念できる。

実験では主に実験回路とのデータ入出力とそのための前後処理が主体である。また，FORTH 辞書を作り直すためにクロスコンパイラとしての特殊な利用方法がある。

4.1.2 Java 版FORTH

Java 版FORTH の作成は，特別な環境や知識を必要とせずにFORTH システムを利用できることを目標としている。即実行可能なオブジェクトファイルで提供でき，ブラウザによるアクセスのみで実行可能なJava Applet とすることにより，ユーザの環境や知識への依存度を軽減することができた。またApplet の入力領域に直接入力を行うので，プログラムソースを記述するエディタソフトがなくても利用できる。

Java 版FORTH は電子工学実験のWeb ページで公開中であり，アクセスするだけでプログラミング環境を利用できる[3]。プログラミング環境が実験用のC 版FORTHと共通であり，プログラミング・デバッグが自宅や本学総合情報処理センターなどでも可能となり，時間外学習の支援が可能となった。

4.2 ハードウェア版FORTH

実働するハードウェアの実例として提示すると共に，FORTH ワード単位でのハードウェア処理を観察する機材として利用できるように整備したものである。また当初より命令コードの拡張や，マイクロプログラムの追加を考慮して設計している。

ハードウェア版FORTH はマイクロプログラムによって基本ワードが用意されており，FORTH 辞書はソフトウェア版FORTH と共通のものを利用する。これは保守管理という点で優れているばかりでなく，ソフトウェア版FORTH のプログラミングを，そのまま移植できることを意味する。

マイクロプログラムは編集・追加が可能であり，周辺回路の増設時や，特殊アルゴリズムの追加など，用途や目的に合わせて適時変更ができる。これは管理者側のみならずユーザ側も可能であり，新しいマイクロプログラムであっても既存のものと区別することなく利用できる。このためにFORTH システム上に実現されたマイクロプログラムアセンブラが用意されている。

FORTH レベルでのプログラミングは，ソフトウェア版のものが利用でき，直接アセンブラ言語を用いて記述するよりも容易にハードウェア制御のアルゴリズムを作成できる。そしてインタプリタ処理を省いた高速化プログラムへの変換も比較的容易である。また単に，FORTHプロセッサ内のアダー・シフタ・レジスタなどを部品として使用し，マイクロプログラムによる専用演算装置の製作も可能である。

このような環境はソフトウェアにおける可読性や保守性を考えたプログラミングと，高速化・最適化を重視したプログラミングを比較する実験機材として利用できる。そしてこの処理アルゴリズムを完全にハードウェア化した制御部の製作へとつなげられる。

5 実験用に整備されたツール

実験でFORTH を利用するに際して，プログラミングとデバッグの支援のためにユーティリティを用意してある。その中から主なものを紹介する。

5.1 逆コンパイラ DIS

すでに定義されているワードをFORTH レベルで逆コンパイルして出力する。これによってワードの内部処理を知ることができる。

あるワードの内部構成を詳しく調べたり，既存のワードの一部を変更したいときなど，FORTH レベルのソースコードを参照する際に利用される。

```
forth>DIS MAX
: MAX
  OVER OVER <
  IF
    SWAP
  ENDIF
  DROP
;
```

Fig 4: Control structure presented by DIS command

Fig 4 中5 行目のようにインデントを付加し，制御構造を再現して視覚的にわかりやすく表示する。

5.2 シンボリックダンプ SDMP

すでに用意されているワードにたいして，占有しているメモリ領域の情報を表示する。表示する情報は数値情報のみならず，呼び出しているワード名と付加情報を表示する。

出力される結果は，インタプリタが順次実行していく手順であり，分岐や反復の処理がどのように実現されているかなど，その詳細を知ることができる。

```
forth>SDMP MAX
08CA 83   MAX
08CB 4D41D8
08CE 08B9
08D0 0001
08D2 0463 OVER
08D4 0463 OVER
08D6 0415 <
08D8 0323 0BRANCH 08DE
08DC 0475 SWAP
08DE 046C DROP
08E0 02A1 ;S
```

Fig 5: Threaded code and symbolic text presented by SDMP command

Fig 5 では左より，アドレス，内容，ワード名と表示されている。付加情報（Fig 5 中9 行目の08DE ）より分岐先アドレスが把握できる。

5.3 トレース機能

プログラミングのデバッグの際に，実行中のデータ変化や処理の流れがわかると都合が良い。特にFORTH ではデータをスタックに積むので，実行中のスタック状態の把握が重要である。

これらを考慮して，インタプリタにトレース機能を組み込んである。開始アドレスと終了アドレスを設定することによって，実行中のスタック状態やワードの実行状態を知ることができる（Fig.6）。

5.4 ポート入出力

外部との入出力用に> C とC > が用意されている。インタプリタ上から直接使用したり，前後処理を伴うワードに組み込んで使用する。

> C はデバイスポートに8 ビットを出力し，C > はデバイスポートより8 ビットの入力を得る。FORTH と実験回路との入出力にはこの2 つのワードを用いる。Java 版やFPGA 版FORTH のように動作環境が異なる場合，あるいはデバイスポートの種類が異なる場合でもFORTH レベルのプログラミングソースが共通に利用できるように実装してある。

5.5 実験用プログラムの入出力

プログラムの読み込みはLOAD で行う。ファイル内の文字列を連続したFORTH プログラムとみなし，インタプリタが解釈実行を行う。プログラミング量が多いPLD アセンブラや，FORTH プロセッサ用のマイクロプログラムアセンブラでは，ファイル内にプログラミングソースを記述しておき，ファイル読み込みによって一連の処理を行っている。

実行結果の保存はSAVE で行い，ディスプレイ出力される文字列をファイルへ書き込む。DIS によるソースファイルやディスプレイ出力結果の保存にも利用される。

5.6 マイクロプログラムアセンブラ

FORTH プロセッサのアセンブル用のワードを用意してある。ニーモニックをワードで定義してあり，インタプリタで処理して一つの命令コードとなる。アセンブル

```
forth>5 9 MAX
08D2 0465 003F  9F2C:0009 0005 3520 3920  9FFC:1605 1700 0000 0000  OVER
08D4 0465 003F  9F2A:0005 0009 0005 3520  9FFC:1605 1700 0000 0000  OVER
08D6 0417 0035  9F28:0009 0005 0009 0005  9FFC:1605 1700 0000 0000  <
08D8 0325 0017  9F2A:0001 0009 0005 3520  9FFC:1605 1700 0000 0000  0BRANCH
08DC 0477 0041  9F2C:0009 0005 3520 3920  9FFC:1605 1700 0000 0000  SWAP
08EE 046E 0040  9F2C:0005 0009 3520 3920  9FFC:1605 1700 0000 0000  DROP
08E0 02A3 0006  9F2E:0009 3520 3920 4D41  9FFC:1605 1700 0000 0000  ;S
IP   W    TKN   SP   tos  nos  ..   ...   RP   tos  nos  ..   ...   Word
```

Fig 6: Traced result by interpreter

ソース自体がFORTH プログラミングなので，変数や定数が利用できるとともに，演算記述なども限定されずに使用できる。

記述形式はSource/Destination/Operation の順であり，動作条件を任意に設定することができる。

```
:H  do0=     ( n --- f )
  False  WR1      ,LOAD ( False -> WR1     )
  TOS    WR2      ,MOVE ( Check TOS value )
  True   WR1  Z,  ,LOAD ( True  -> WR1     )
  WR1    TOS      ,MOVE ( Flag  -> TOS     )
  NEXT
```

Fig 7: Example of micro programming

ワードを利用して定数や変数による即値情報の設定もでき，アセンブラ自体もFORTH ワードで整備されているので，ユーザの用途に応じて変更ができる。

5.7 クロスコンパイラ

定義内容の改良，不具合の修正，機能向上，異なる環境への移植などのためにFORTH 辞書の変更が起こる。この変更作業を容易にするために，ソース形式のFORTH辞書からメモリダンプ形式のFORTH 辞書に変換するクロスコンパイラを作成した。

動作中のFORTH における辞書管理機構を巧妙に利用し，FORTH メモリ上に新しい辞書を展開し，アドレスポインタを変更して16 進ダンプにより新しいFORTH 辞書を生成している。なお，先に述べた逆コンパイラDIS によってFORTH メモリ上の定義済みワードをFORTH プログラムのソース形式として出力することにより，クロスコンパイル用のソースファイルを得ることができる。

6 FORTH を利用した機能分担の実現

ディジタル回路のトレードオフ問題を扱う実験の機材として，任意な機能分担を実現する機材が望ましい。またそのための手続きや処理はできるだけ容易であり，対応が明確に理解できることが重要である。

6.1 容易な機能分担の切り替え

実験ではある機能をソフトウェアとハードウェアによって実現し，それを切り替えて観測や試験を行っている。

ソフトウェアで実現する場合にはFORTH プログラミングによって行う。FORTH のプログラムはワード単位で実行されるので，ハードウェアとの機能分担はワード単位で行う。ワードの内部はアドレスポインタの並びであり，このアドレスポインタを変更することで呼び出す下位ワードを変更することができる。

ソフトウェアからハードウェアへ機能分担を切り替える際には，このアドレスポインタの変更で実現できる。ユーザはハードウェアとのインタフェース処理をワードとして定義し，アドレスポインタを変更する。この際に必要となるワードの情報は必要に応じてDIS やSDMP などを用いて調べる。

アドレスポインタによる機能分担の切り替えは基本ワード（算術演算子を含む予約語），既存のハイレベルワード，新たなユーザ定義ワードなど区別無く適用できる。データの入出力にはスタックを利用する。データの存在する場所は変化するが，常にスタックトップを対象とした同じ手続きで処理することができる。またハードウェア版FORTH を利用することによってデータ受渡しや演算処理などの機能が即利用可能となり，制御アルゴリズムの作成のみによる容易なハードウェア製作が可能になる。

6.2 多様な機能分担

ハードウェア版FORTH での制御アルゴリズム作成には，FORTH プログラミングとマイクロプログラミングの2 通りの方法がある。さらにハードウェア版FORTHの外部に機能回路を用意して，そこで処理をさせるという方法もある。それぞれは保守性や処理速度といった点がトレードオフとなっている。ソフトウェア版FORTHとハードウェア版FORTH では同じFORTH プログラミングが動作するが，外部との通信手段や操作性，処理速度といった動作環境に依存するトレードオフがある。

従来からのハードウェア製作は，PLD によるコントローラ製作と，ラッピング配線による回路製作で行っており非常に労力を要する作業である。ハードウェア記述言語の活用により配線作業が無くなり，また，FORTHプロセッサを利用する場合，シーケンス制御部分が用意されることになるので，制御アルゴリズムの作成のみで済む。これによって，従来よりも設計や検証により多くの時間を費せるようになる。

FORTH プログラミング移植の簡単な手段からハードウェア記述言語による設計まで，ソフトウェアとハードウェアによる機能分担をより段階的に調節でき，学生実験として過度に複雑にならずに実現することができる。機能モジュールの比較対象を多く実現できることで，従来よりもトレードオフ問題についてより多様な実験が行えるようになる。

7 おわりに

ディジタル回路の実験用に統合的にFORTH 環境を整備した。ツール類の整備，FORTH プロセッサの製作，実験室以外でのプログラミング環境を提供することを目的としたJava Applet によるFORTH インタプリタの作成などを報告した。

FORTH システムの特徴であるスレディッドコードや

スタックは容易な機能切り替えの方法を実現する。そして従来では画一的で単調であったハードウェア製作のアプローチに対して，FORTH プロセッサ・マイクロプログラムの利用による柔軟な手段を提供できる。

今後は実験での活用方法や，Web 上の遠隔実験との相互運用環境を整備し，実験室と遠隔実験サービスを密接に結びつけたグローバルな実験環境の構築を目指していくつもりである。

最後に，藁科は平成8 年度電気通信大学奨励研究費の支援を受け試作研究を行い，平成13 年度の理工系教育高度化設備費“ ディジタル回路「設計・製作」実習教育設備 ”によって電子工学実験授業のために設備が調えられたことを記して，関係各位に深謝する。

参考文献

[1] 電子工学実験参考資料: 電気通信大学実験工学研究室，2002.

[2] 高橋光生，田中清臣：移植性を向上させたFORTH の内部構造: 電気通信大学紀要第11 巻第2 号p.161-167，1998.

[3] http://www-lab.ee.uec.ac.jp/

[4] 井上外志雄著: 標準FORTH，共立出版，1985.

[5] The Programmable Logic Data Book，Xilinx inc，1994.

[6] Phillip J.Koopman,Jr.(田中清臣監訳) : スタックコンピュータ，共立出版，1994.