東海大学紀要情報通信学部
vol.3,No1,2010,pp.19-24

論文

ASIC16bitCPU開発におけるPBLの有効性

山崎 亮太*1　　上陰 敏弘*1　　虎口克尚*1　　清水 尚彦*2

Effectiveness of PBL in ASIC16bitCPU Development

by

Ryota YAMAZAKI, Tosihiro KAMIKAGE, Katunao TORAGUTI and Naohiko SHIMIZU

(received on May 8, 2010 & accepted on August 6, 2010)

Abstract

Experiment on a CPU system development is a good method to learn the detail of the software behavior, and it is required in recent years with popularity of the programmable logic. But the development requests learners a broad area of knowledge that cannot be acquired in the usual lectures. Therefore, we selected the PBL(Project-Based Learning) for the development to get the knowledge of the covering area. We executed the education program for the students who have the fundamental knowledge of the computer science to propose the system exhibiting at the ASIC design contest of PARTHENON society. The proposition of the system is excellent method to learn the foundation of the development and the principle of the CPU for students.

Keywords: ASIC design contest, PARTHENON, PBL(Project-Based Learning), CPU system development learning
キーワード: ASIC デザインコンテスト，パルテノン，PBL(プロジェクト・ベースド・ラーニング)，CPU システム開発学習

1. はじめに

CPU システム開発学習はソフトウェアの最も細分化した点の学習に有効であり，プログラマブルデバイスが普及し始めている近年では必須の学習である．CPU システム開発学習の問題点として，CPU 開発は規模が大きく，学習が難しいと言う点があげられる．そこで，我々は CPU システム開発の参考となるコンテスト標準モデルが用意されている特定非営利活動法人パルテノン研究会の ASIC デザインコンテスト [1] が学習に適した題材であると考え選択した．ASIC デザインコンテストの規定に従いシステム開発を行うことで，システムに機能制限をかけ，CPU 構成，システム構成を単純化し学習に適した規模の開発が行える．

我々はシステム開発学習方法として，PBL(Project-Based Learning) を用い，主に学生間で問題提議をし，解決方法の提案を教員を含め全員で行った．以下, 報告する．

2. CPU システム開発 PBL の特徴と目的

CPU システム開発 PBL は，開発プロセスを実践でき，開発手順を踏んで，問題提議から解決まで一貫して取り組める利点がある．開発を実践することで，学習の目標を成果物によって明確にすることができる．また，開発手順を踏むことで，CPU 設計に必要な CPU の構成や動作，コンパイラ設計に必要なアセンブラコードの知識やコンパイラのアルゴリズム，システム設計に必要な論理合成の知識を学ぶことができ，CPU システム開発に必要な広域にわたる知識に触れることができる．さらに，問題提議から解決まで参加者自身で取り組めるため，自発的に知識を身につけることができる．本論文における PBL の目的は CPU システム開発において広域にわたる知識を身につけることである．

3. ASIC デザインコンテストの概要

16bitCPU 及びコンパイラの開発を行い，論理合成後の消費電力と命令処理時間を競う．規定として CPU アーキテクチャは，命令・データのアドレス空間を分離したハーバードアーキテクチャであり，1 ワード 1 アドレスのワードマシンである．変数はメモリ上の 1 番地から順に格納される大域変数の a から z までと，関数引数の arg が許される．変数には添字をつけることができ，配列 a[i] は，a から i 番地のメモリ上の値を表す．関数は整数型か，void 型の foo という関数名 1 つだけが定義でき，局所変数は存在しない．仮引数は arg を 1 つだけ使用でき．演算では +，−，<，>，>=，<=，==，++，−− が，構文では while，if，else，for，return，halt が利用可能である．なお，halt 実行時にはプロセッサをストップする．コンテスト審査数値には，実行時間ならびに実行エネルギーの相乗平均を使用する．

4. 実施過程

開発手順を踏むにあたり，まず全員の基礎知識を確認した．その後，開発をはじめ，CPU システムの提案を行った．CPU システム開発 PBL の実施期間は週 1 回，授業時間を９０分として 10 月 ～2 月まで行った．予備教育として，CPU 本体とコンパイラの学習を 2 ヶ月間とし，最初の週に問題提議の議論を行い，次の週で問題解決の議論を行った．まず，10 月は CPU 本体の学習を行い，CPU システムを実際に動かすことでシステム動作を学び，当システムの問題点などについて議論した．コンパイラの改良余地があることが判明したので 11 月からはコンパイラの

*1 情報理工学部 ソフトウェア開発工学科 4 年生
*2 情報通信学部組込みソフトウェア工学科 教授

学習を全員で始め，コンパイルシステムについての学習とシステムの改良余地についての議論を行った．12 月は学生の能力に合わせてグループを作り本格的なシステム開発に取り組んだ．1 月は CPU システム及びコンパイラシステムが完成したので検証を行った．しかし検証を行う上で論理合成に問題があったので 2 月には PARTHENON 論理合成検証システムの改良を行った．(Table1)

Table1　Execution process

October	The composition of SN/X is understood. Making to the pipeline is scheduled. Simulation on the desk. Finding of data hazard problem.
November	Understanding of yacc and lex. Find the problem of the register.
December	The member is grouped into CPU and the compiler. Parthenon Verification.
January	Completion of each pipeline.
February	Improvement of logic synthesis tool. Verification of system.

4.1　CPU(SN/X) システムについての学習

当プロジェクトを 10 月に始めた段階では，学生の CPU 開発に関する知識，及びコンテスト標準 CPU:SN/X[2) 3)] についての知識が無かった．そのため，まずは SN/X システムを動かすことから始め逐次気づいたことを議論し，今後の改良点の参考とした．第 1 週は今後のスケジュールなどについて話し合い，第 2 週は SN/X の動作を理解するため，指導者と共に机上でアセンブラコードを動かした．これによって SN/X の構成，CPU の基本的な構成を理解することができた．第 3 週は過去のコンテスト参加者の資料 [4) 5)] からパイプライン化を行えば早くなることが分かった．第 4 週までに参加者各自がパイプラインについての調査を行い，パイプライン化した SN/X の動作を指導者と共に机上でアセンブラコードを追って理解した．これによってパイプラインの改良点としてデータハザードの回避や回路規模の縮小および周期の短縮などが分かり，議論から今後の CPU 改良の方針が決定した．

4.2　コンパイルシステムについての学習

SN/X システムについての知識は身についたので 11 月にはコンパイラについての学習を始めた．コンパイラは yacc と lex と C で構成されており，非常に複雑なので，まずはコンパイル動作の解析を全員で行い, その上で改良の余地について話し合うこととした. 第 1 週はコンパイラの大まかな構成について調べ，第 2 週は実際に指導者と共に机上でコードのコンパイルを行った．これによってコンパイラの構成についての理解が進み，学生同士が教え合うことによって開発の基盤を固めた．第 3 週はどのように改良を加えれば性能が向上するのかを議論し，アセンブラコードに問題があることを発見した．第 4 週は自分で考えたアセンブラコードとコンパイルして生成されたアセンブラコードを比較し，レジスタに余分な書き込みが行われているなどの具体的な問題が分かり，今後のコンパイラ改良の方針が決定した．

4.3　グループに分かれての開発

10 月 11 月で決定した今後の方針を進めるには時間が足りないと考え，12 月は参加学生を CPU 設計担当グループとコンパイラ担当グループに分けた．第 1 週は学習過程にて決定した方針にしたがい，CPU 及びコンパイラの改良を行った．また，CPU 設計はプレフィックス加算器，コンパイラ改良には perl 言語についての知識が必要であることが分かり，第 2 週は担当グループごとの学習を行った．第 3 週はシステムの改良と更なる改良点の議論を行い，今後，いくつかの CPU モデルを作り，それぞれ検証を行って，よい結果の物を採用することとした．第 4 週は検証を行うために，パルテノン研究会から提供されている論理合成ツール「PARTHENON」の使用方法を調べた．また，コンパイラ開発を担当している学生により，コンパイラ改良過程にて命令コードを更に減らせるとの報告があり，その学生を主導として，報告に基づいたコンパイラ開発を行った．

4.4　動作検証

1 月の第 1 週は 2 段，3 段，4 段パイプラインの CPU モデル作りが完了し，コンパイラは 11 月の目標を越える改良がなされていた．ここで，論理合成検証ツールにて検証を行った．もっとも良い結果を得られた 2 段パイプライン CPU の使用を決定したが，過去のコンテスト参加者のシステムより劣っていることが分かり，今後の更なる改良の議論を第 2 週に行った．議論の結果，論理合成ツールに本コンテスト審査基準には必要の無いルールが適用されていることが分かり，今後の改良点とした．第 4 週までは論理合成ツールの仕組みを調べたが，改良できるほどの知識は得られなかった．

4.5　更なる改良

2 月の第 1 週は論理合成ツールの仕組みについて理解した学生が全体に説明を行った．第 2 週に論理合成ツールの改良を行い，論理合成ツールが完成した．システムの検証を行った結果，過去のコンテスト参加者を越える性能に達していたためプロジェクトを終了させ，完成品としてドキュメントを製作し提出を行った．

5. 成果物

5ヶ月間のCPU開発学習期間が終わりASICデザインコンテストに出品するCPUシステムが完成した．以降に今回のPBL学習過程にて主にシステム改良を行った点をあげる．

5.1 CPUアーキテクチャ

コンテストで審査されるエネルギー時間積は実行クロック数と比例関係にあるため，クロック数の削減により減らすことができる．そこでパイプラインの構築と，命令セットの改良を行い，クロック数を削減することでエネルギー時間積を減らすことにした．また，参加学生は学部3年生であり，CPU設計に関する知識は無い，そのため個々の能力によって最適なパイプラインの検証，加算器の変更，命令セットの改良，論理合成の改良を割り振り，個々の理解や成果を全体に知らせることで論点重複の無駄を減らし効率的なシステム全体の改良を行った．

5.1.1 SN/Xのパイプライン化

SN/Xはフェッチ(IF)，実行(EX)，メモリアクセス(ME)の3ステージ設計であり，命令は逐次実行されているためクロック数が比較的多かった．このSN/Xをパイプライン化し見掛け上1クロック1命令の実行を行い，大幅なステップ数の削減による最大遅延時間の短縮を行った．また，いくつかのパイプラインモデルを作り性能評価を行い，その中でも優秀な結果を残した2ステージ設計を採用した．(Table2，Fig1)

Table2　Each pipeline comparison beforee it modifies it

	Number of clocks (sort)	Consumption energy { μW/MHz }	Cycle {nS}	Performance order
Four step pipeline	16074	7917.6	45.8816	3
Three step pipeline	12522	5702.9	58.0425	2
Two steps Pipelline	9014	5002.9	62.9483	1
Normal SN/X	26216	5680.1	63.6	4

5.1.2 Parallel Prefix Adder（並列プレフィックス加算機）の変更

加算機を変更するにあたり4つの加算機のデータを調べ，その中から素子数と速度の兼ね合いが本体の設計に適していると考えてHan-Carlson Adder(Fig2)を実装することにした．Han-Carlson Adderは経由する素子が平均的で入力される値がバランス良く経由されるので，電力を抑えつつある程度の速度が得られると判断した．加算機の制作は回路と素子の式を元にしてSFLで行った．(Fig2)において本体で用いないキャリーは削除してある（最上位ビットの削除）．プレフィックス演算を行うにあたり，値をPとGに分けて計算するPgモジュールを並列にした．

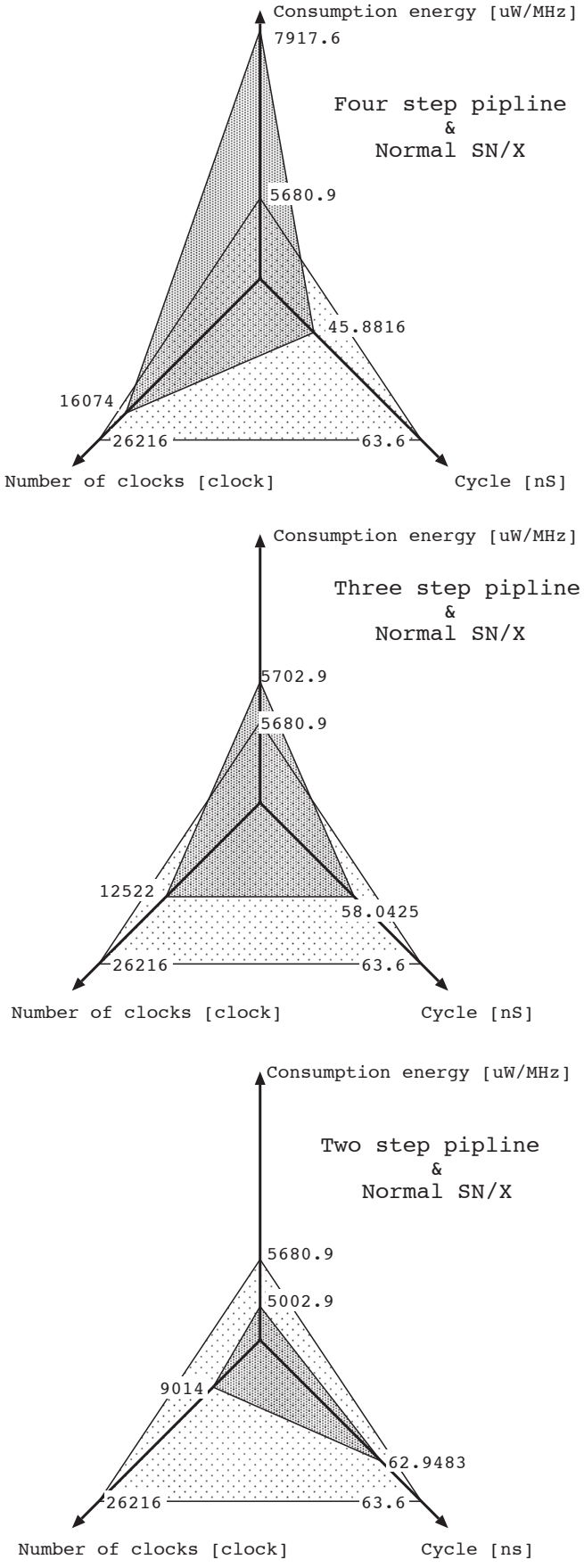

Fig1　Each pipeline comparison beforee it modifies

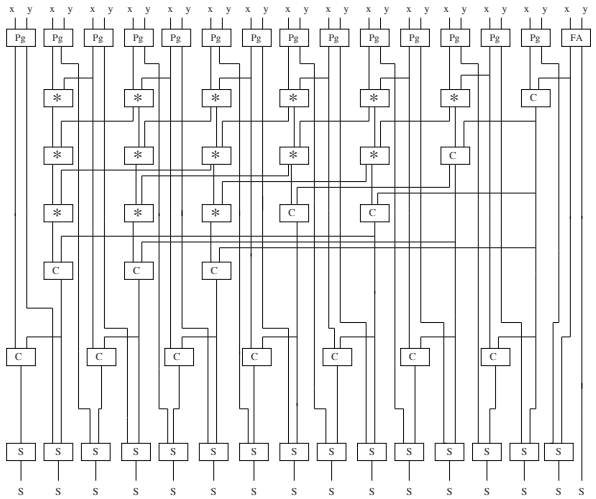

Fig2　Composition of adding machine

5.1.3　処理時間均等化

2 段パイプライン設計では，EX ステージに処理が集中してしまい，最大遅延時間が延びる結果となってしまったので，周期の短縮化を行った．EX ステージではデコード処理，レジスタからの値の呼び出し，呼び出した値に基づく命令の実行，実行結果のレジスタへの書き込み，データメモリの読み書き（論理遅延＝0），分岐時の次ステージの呼び出しを実行していた．

IF ステージでは命令メモリ（論理遅延＝0）から命令の引き出し，PC へのカウント加算，次の IF ステージの呼び出し，EX ステージの呼び出しを実行しており，IF ステージは EX ステージと比べ処理時間が短い．

EX ステージで実行している処理のうち，デコード処理は IF ステージで実行してもハザードは発生しないため，IF ステージでデコード処理を実行できるようにし，処理時間の均等化による最大遅延時間の短縮化を行った．これによってパイプラインレジスタの改良が可能になり，デコード処理により不要となった命令ビットの削除と，削除部分へデコード値を組み込むことによって消費電力の肥大化を回避した．(Fig.3)

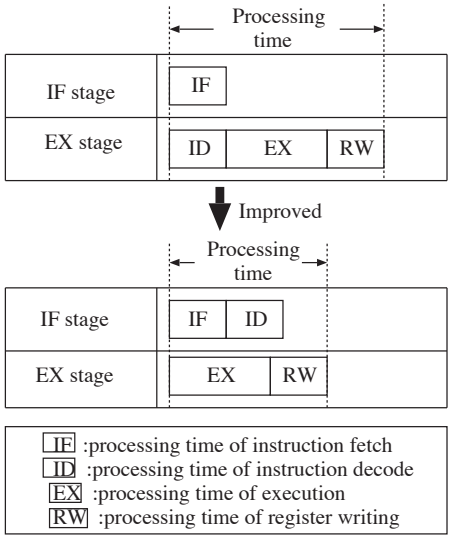

Fig3　Change in processing time

5.2　コンパイラ

パルテノン研究会の提供するコンパイラは，汎用的な設計となっているため，コンテストの規定に最も合う設計となるよう様々な改良を行った．まず，既存のコンパイラが生成するアセンブラコードの机上デバッグを行う事でコンテストの規定において余分な部分を抽出した．その時，コンパイラが使用しない不要な命令を削除を行い，CPU アーキテクチャの命令セットの変更を行った．また，処理性能を向上させる追加命令を考え，実装をした．その後，より少ないアセンブラコードを生成するようにコンパイラを改良した．最後に，コンパイルして生成されたアセンブラコードから，更に余分なコードを削除するスクリプトを作成することで実行クロック数を削減した．

5.2.1　アセンブラコードの最適化

コンパイルして作られたアセンブラコードのままでは，余分な ld 命令が多くある．それを削除するために，Perl を用いて，アセンブラコードの最適化を行った．最適化の方法としては，大きく分けて２つの処理を行っている．１つめは，アセンブラコード上の不要な ld 命令の削除を行う ld 命令削除処理で，２つめは，条件分岐で$0 レジスタを使用するようにコードの最適化を行う$0 変換処理である．(fig4)

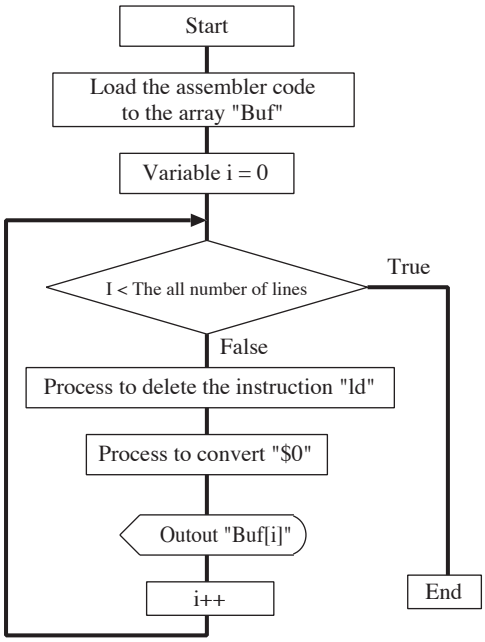

Fig4　Flow of assembler code optimization

ld 命令削除処理は，アセンブラコードを最初から１行ずつ読み出し，ld 命令を見つけると，第１引数のレジスタが ld 命令より前に使用されている行まで１行ずつ戻る．その時，ld 命令の第３引数であるレジスタが別の命令で第１引数となっていないかを判断し，第一引数となっていない時に，ld 命令を削除する．

$0 変換処理は，アセンブラコードを最初から１行ずつ読み出し，bz 命令を見つけると，一つ前の行が３つの引数を必要とす

る R 形式の命令だった時，命令と bz 命令の第 1 引数を$0 に変更する．

5.3 PARTHENON システム

PARTHENON システムの AUTO コマンドにて論理合成を行った．AUTO コマンドは，回路の最適化を行うために DEMO ライブラリにおける ons スクリプトを実行している．ons スクリプトは論理回路の簡略化を行うが，簡略化を行うと素子の減少に対して最大遅延時間は増加する．しかしエネルギー時間積には最大遅延時間が大きく関係しているので，新規に ons スクリプトを実行しないライブラリを作成し実行することで最大遅延時間の短縮を図った．

5.4 改良後の動作検証

以下のプログラムで動作検証を行い，正常に動作がされることを確認した．さらに課題プログラム以外も正常に動作することを確認するために新たに動作検証用プログラムを作り検証を行った．検証を行った結果いずれも正常に動作していることを確認した．(Fig5，Fig6)

- 再帰処理課題プログラム (recur.sc)
- バブルソート課題プログラム (sort.sc)

```
1:          n=20;
2:          for(i=0; i<n; i++) z[i]=n-i;
3:          for(i=0; i<n-1; i++)
4:             for(j=n-1; j>i; j−−)
5:                if(z[j]<z[j-1]) {
6:                   w=z[j];
7:                   z[j]=z[j-1];
8:                   z[j-1]=w;
9:                }
10:         halt;
```

Fig5　バブルソート課題プログラム (sort.sc)

```
1:          int foo(arg) {
2:           if(arg<2) return arg;
3:           else return foo(arg-2)+foo(arg-1);
4:          }
5:          foo(10);
6:          halt;
```

Fig6　再帰処理課題プログラム (recur.sc)

5.5 旧システムと新システムとの比較

コンテストにおいて性能評価基準であるエネルギー時間積を用いて比較を行った．

実行時間の相乗平均値 $T[s]$ は，

$$\left\{T = \sqrt{A \times B} \times D[s] \right. \tag{1}$$

である．また，実行エネルギーの相乗平均値 $E[J]$ は，

$$\left\{E = T \times (P/D)[J] \right. \tag{2}$$

したがって，エネルギー・時間積 $M[J \cdot s]$ は，

$$\left\{M = T \times E \right. \tag{3}$$

改良の結果として消費電力の大幅な減少は無かったが，最大遅延時間は大きく減少した．

さらに評価式において消費電力が 1 乗に対し，最大遅延時間が 2 乗で計算されているのでエネルギー時間積は大きく減少している．

我々の設計した CPU システムのエネルギー時間積は 0.0962[J·s] であり，コンテスト標準モデルのエネルギー時間積 3.06[J·s] の 31.8 倍の性能向上を実現した．(Table3，Table4，Fig7)

Table3　Comparison of numbers of clocks

	”recur.sc” Number of clocks	”sort.sc” Number of clocks
Before SN/X	347	26216
After SN/X	155	3985

Table4　Comparison of performances

	Power consumption[μ W/MHz]	Delay time[nS]	Energy time product[j·s]
Before SN/X	5680.1	63.6	3.06
After SN/X	5074.5	30.69	0.0962

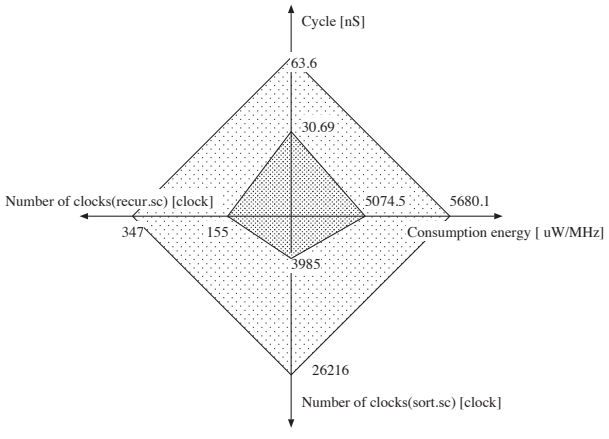

Fig7　Flow of assembler code optimization

6. 考察

5 ヶ月間の CPU 開発 PBL の結果，学生たちは CPU システム開発の基礎能力を身につけた．それは，10 月の段階では CPU の動作も分からず，システムを動かしていた時と比べ，プロジェクト終了時までに数種類のシステムを用意していたことやコンパイラ及び論理合成の工夫から伺える．

CPU システム開発において PBL は，学生たちの積極的な議論を促し，自ら提案した改良など問題を調べ解決することにより CPU システムの知識を得ることができた．

PBL 学習の成果として学生たちの設計した CPU システムのエネルギー時間積は 0.0962[J·s] でありコンテスト標準モデルのエネルギー時間積 3.06[J·s] の 31.8 倍の性能向上を実現した．これは CPU 本体とコンパイラ，そして，論理合成システムの 3 つの改良があったためであり，学生が十分にそれらを理解し，CPU システム開発を理解した結果である．

具体的に CPU システム開発 PBL を通し，どう言った知識が身についたか以下に例にあげる．

例 1

CPU 本体を改良するにあたり，現在の CPU では必ず使われている技術であるパイプラインに触れることができ，その結果 CPU 内部の状態遷移を理解し，改良を加えることができた．また CPU の HDL 記述に velirog より高位レベルの設計言語である SFL を用いることで，ハードウェア記述言語の基本的な記述方式を身に付けることができた．

例 2

コンパイラを改良するにあたり，コンパイラがどのような仕組みでソースコードを変換しているかを理解し，アセンブラ言語の理解も深めた．また，コンパイラの大部分で C 言語が用いられていたことから，C 言語の理解も深まった．

当 PBL では，ソフトウェア・ハードウェアの両面を一つのプロジェクトを行うことで実現した．しかし，今回の CPU システム開発 PBL では，CPU システム開発をより学習向けにするために既存の CPU システムの改良を行ったが，当 PBL を行った学生の意見では，次は自分達で一から CPU システム開発したいとの意見が出た．この様に学生が，知識，技術を身に付け，更に自主的に CPU システム開発を行いたいと思うようになったことから CPU システム開発における PBL は有効だと分かった．

7. 謝辞

本研究は，PARTHENON 研究会主催の ASIC デザインコンテストを主題にして行った．この様な機会をもうけていただいた PARTHENON 研究会に深く感謝いたします．

参考文献

1) パルテノン研究会,“ASIC デザインコンテスト”, http://www-lab09.kuee.kyoto-u.ac.jp/parthenon/contest/index.html (参照 2009-10-7)
2) 清水尚彦, “コンピュータ設計の基礎知識”，IP ARCH．Inc，http://www.ip-arch.jp (参照 2009-10-7)
3) 清水尚彦, “コンピュータ設計の基礎知識-ハードウェア・アーキテクチャ・コンパイラの設計と実装-”，PP.206，共立出版，2003.
4) 藤田歩，山本小太郎，大金淳一朗，樋口拓哉，北嶌卓哉,“ちえのわ”，第 34 回パルテノン研究会，Vol34，PP.81～88，2009.
5) 会津大学, 五十嵐翔一, 吉田幸裕, “あらばしり”，第 34 回パルテノン研究会，Vol34，PP.55～64，2009.