愛知工業大学研究報告
第 50 号　平成 27 年

ASIC 設計手法を用いたビット幅拡張 CPU の設計

Chip Design for a CPU with 8-bit Wide Datapath by means of Application Specific Integrated Circuit

鈴木貴斗†，江口一彦††，五島敬史郎††
Takato Suzuki, Kazuhiko Eguchi, Keishiro Goshima

Abstract In recently, hardware description language (HDL) is the most powerful tools for the design and development of Large Integrate Circuit (LSI). LSI circuit using ASIC (Application Specific Integrated Circuit) technology has advantages for such as low consumption and short processing time. In this paper, we designed a CPU (Central Processing Unit) with large register memories and 8-bit wide detapath for the purpose of the increase processing capability. And we fabricated a custom LSI using the ASIC technology.

1. はじめに

昨今，身の回りにはコンピュータ製品が溢れ設計技術者の重要性がますます高まってきている．デジタル回路を用いた大規模集積回路（LSI)の現場では，設計効率化のため，HDL と呼ばれるハードウェア記述言語の利用が主として行われており，現在では C 言語を用いたハードウェア設計も可能となっている．高度に抽象化された HDL とデジタル回路動作との間には大きな理解の障壁があるのが事実である．

一方，Field Programmable Gate Array （FPGA)はリコンフィギャラブルな特徴を持ち，短期間で開発できる利点があるため急速に普及してきた．FPGA を用いたデジタル回路設計において FPGA 内部はブラックボックス化されており，内部でどのような回路構成になっているか確かめることは難しくなってきている．加えて，大規模・高性能化するにつれて消費電力・製造コストの増大などの欠点がある．そのため，低消費電力・過酷な条件下での動作などの特殊性能が求められる場合には，ASIC で設計される場合が多い(1) (2)．

また，高い動作周波数の動作状況下では，デジタル回路であってもアナログの設計要素が必須になってきている．HDL を用いた開発だけで無く，その裏で動いている回路の中身や動作を理解する必要があると考える．

†　愛知工業大学大学院　工学研究科
電気電子工学専攻　（豊田市）
††　愛知工業大学　工学部　電気学科
電子情報工学専攻　（豊田市）

また，高い動作周波数の動作状況下では，デジタル回路であってもアナログの設計要素が必須になってきている．HDL を用いた開発だけで無く，その裏で動いている回路の中身や動作を理解する必要があると考える(3)．

そこで，我々は HDL によるデジタル回路開発だけでは無く，ASIC を用いた設計方法を取り入れている．Application Specific Integrated Circuit (ASIC)設計手法では，HDL を用いつつデジタル回路の構成要素である Complementary MOS (CMOS)トランジスタの設計からスタンダードセル設計，配置，それに伴う信号遅延等を考慮した設計を行うことによって内部回路の理解の両立を目指した(4)．

現在までに我々は VLSI Design and Education Center (VDEC)を利用した ASIC 開発環境を構築し，4 ビット CPU のオープンソースを利用して ASIC 設計工程の下流設計の部分を行った．配置配線・レイアウト設計を行い，フェニテックセミコンダクター社のシャトル便を用いて実際にチップ製造まで行った．しかし，取り扱うレジスタメモリ容量が少ない事やデータパス幅が 4 ビットの為，処理に要する時間が多く必要とすることから用途が非常に限定されていた．

そこで本研究では、上記の CPU をベースとしてデータパス幅やレジスタメモリ容量の増大・処理時間の低減を目的とした新たな CPU の設計を行った．加えて今回の設計からは上流設計及び下流設計のすべての ASIC 設計工程を行うこととした．

2.　ASIC 設計手法を用いた開発工程

今回我々は ASIC 設計手法を用いて CPU の設計を行った. ASIC 設計では大きく 5 つの工程から成り立っている。上流設計と呼ばれる機能設計→論理設計、下流設計と呼ばれるレイアウト設計→チップ製造→動作検証である。現在デジタル LSI の設計は Electronic Design Automation (EDA) ツールの使用が必須である. そのため, 本研究では VDEC を通して各設計に必要な EDA ツールを入手し設計を行った. 設計に使用したツールは Icarus Verilog, Design Compiler, IC Compiler, Ismo, Stylus の 5 種類である. 機能設計では, Icarus Verilog を用い Verilog HDL で記述した設計対象のコンパイルと機能シミュレーションを行った. 論理設計では,Synopsys 社の Design Compiler を使用して Verilog HDL の記述をセルあるいはマクロセルという論理素子に変換する論理合成を行う. レイアウト設計では, チップコア部のレイアウト設計 (自動配線作業) と、チップコア以外の外部端子との接続を行う配線作業の 2 種類のレイアウト設計を行った。チップコア部のレイアウト設計では、Synopsys 社の IC Compiler を用い論理合成で出力されたネットリストをもとにチップコア部の自動配置配線を行う. チップコア以外のレイアウト設計では Jedat 社の Ismo を用いコア部の配置および, チップ全体のレイアウトを行う. 製造に関してはフェニテックセミコンダクター社の行っている大学等の研究・教育を目的としたシャトル便制度を用いて製作を依頼した. プロセスは 0.6μm である. 動作検証では, Inovys 社の LSI テスタ Personal Ocelot と Stylus というツールを用い製作したチップの遅延を考慮した性能評価を行った。表 1 に設計工程と各工程で使用するツールをまとめて示す.

表 1 ASIC 設計工程及び使用ツール

設計工程	使用ツール・施設
機能設計	Icarus Verilog
論理設計	Design Compiler (Synopsys 社)
レイアウト設計	IC Compiler (Synopsys 社)
	Ismo (Jedat 社)
チップ製造	フェニテックセミコンダクター社
動作検証	Stylus (Inovys 社)

3. ビット幅拡張 CPU の設計仕様及び動作概要

3・1 設計仕様

今回設計した CPU は演算機能として 8 ビットの演算装置 (ALU) とアキュムレータ(ACC), 汎用レジスタ 2 個(Reg1,Reg2)と 2 種類の演算結果フラグを備える. アドレス制御としてプログラムカウンタ(PC)と分岐用のアドレスレジスタ(AReg)を備える. 外部インターフェース機能として 8 ビットの入出力端子を備える. 実装しているすべての命令を 1 サイクルで実行する. 8 ビット(256 アドレス)のアドレス空間を持ち, 全てユーザプログラム領域とする. PC で, プログラムアドレスの制御を行い, PC の初期値(00h)がプログラム開始アドレスとなる. また, プログラムアドレスの値は常に PAD 端子から出力され, 外部メモリのアドレスとして使用される. 外部メモリからプログラムデータを PDT 端子を介して入力する事によって, プログラム制御を行う. 命令セットを表 2 に, CPU の構成図を図 1 に, 外部接続図を図 2 に示す[5][6].

表 2 命令セット

命令分類	命令
算術演算命令	ADD / SUB
論理演算命令	AND / OR / EXOR
ビット操作命令	ROL / ROR
ロード命令	LDA
ストア命令	STA
分岐命令	BRZ / BRC
無条件分岐命令	JMP
その他	NOP

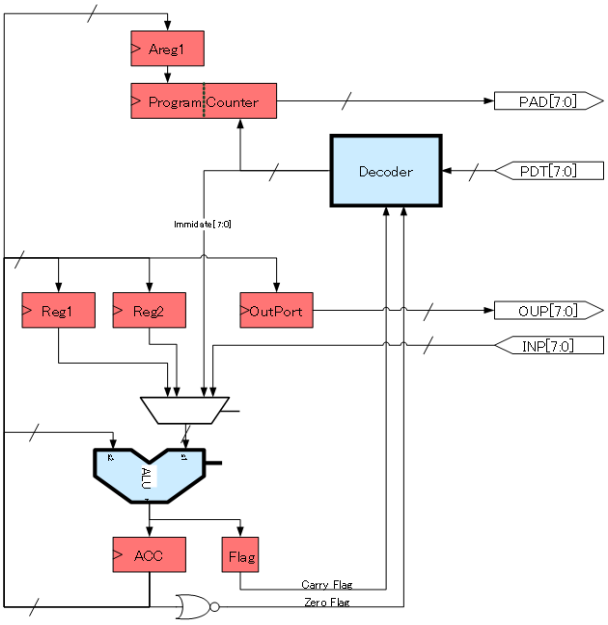

図.1 CPU 内部構造

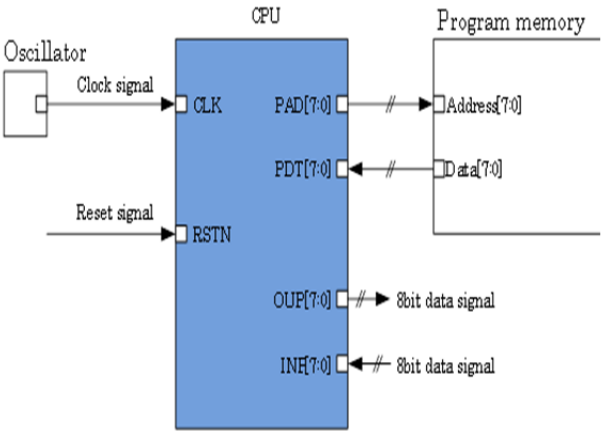

図 2　外部接続図

3・2　動作概要

算術演算命令を例に CPU の動作を述べる．算術演算命令の処理の流れは以下のようになる．

1)　外部命令メモリからデコーダへ命令コードを送る
2)　デコーダから CPU の構成要素に制御信号を送る
3)　セレクタが演算に用いるデータを決める
4)　ALU にデータを送る
5)　ALU が演算を行う
6)　演算結果を ACC とフラグ Flag に書き込む

図 3 に処理の 1 と 2，図 4 に処理の 3 と 4，図 5 に処理の 5 と 6 の流れを示す．

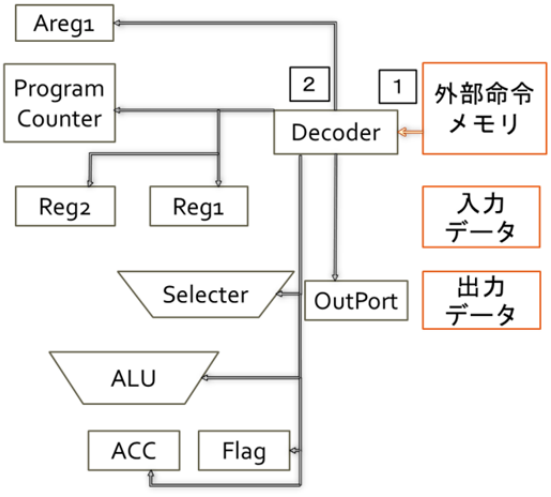

図 3　算術演算命令処理の流れ[1)，2)]

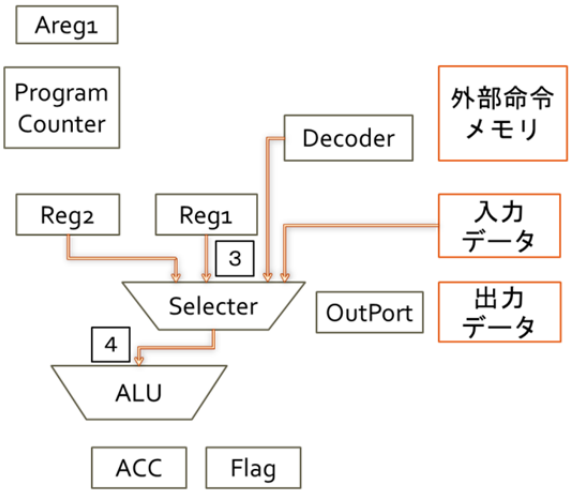

図 4　算術演算命令処理の流れ[3)，4)]

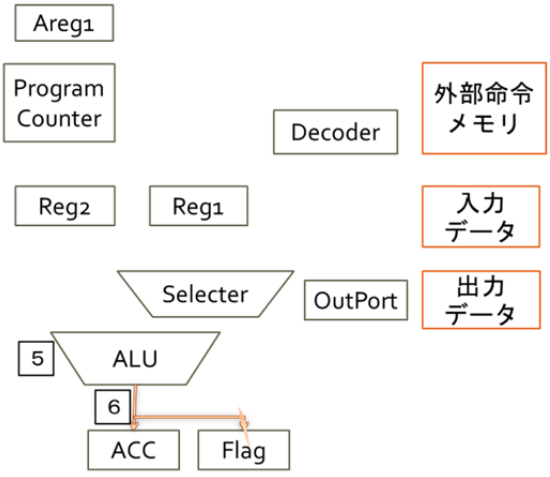

図 5　算術演算命令処理の流れ[5)，6]]

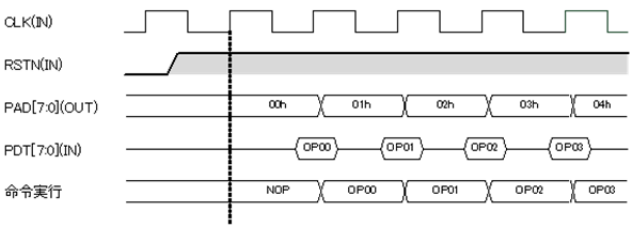

図 6　初期化およびタイミングチャート

初期化および初期化からのタイミングチャートについては，リセット端子 RSTN を一定期間「0」レベルにする事で CPU を初期化する．これにより 8 ビット CPU 内部のレジスタ，出力端子等が全て初期状態となる．続いて RSTN を「0」から「1」レベルにする事で初期化が解除され，CPU は CLK の立上りに同期して動作を開始する．図 6 に初期化および初期化解除後のタイミングチャートを示す。

また，命令アドレスが不規則に変わる分岐命令実行時のタイミングチャートについて図 7 に示す．条件分岐命令（BRZ/BRC)で条件が一致した場合あるいは，無条件分岐命令（JMP)実行時の動作を示す．条件分岐命令で条件が一致しない場合は，次の命令を実行する．

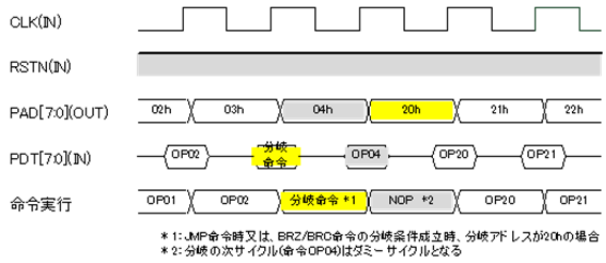

図 7　分岐命令実行時のタイミングチャート

4. 各設計工程の詳細

4・1　機能設計

機能設計では，Verilog HDL を用いて設計を行った．上記の仕様を満たすために, Verilog HDL の入出力ポートやレジスタ，ALU 等のデータ幅を 4 ビットから 8 ビットにした．加えて，Icarus Verilog を用いて論理シミュレーションを行い論理的に動作が正常であると確認した．[8)]

4・2　論理設計

フェニテックセミコンダクター社のプロセスデザインキット(PDK)より論理合成に必要なデータベースファイルを利用した．この PDK には、フェニテック社が動作を保障した論理素子および、論理素子の物理的パラメータが収められている。この PDK のデータを利用する事によって回路論理合成を行う際にある程度の CPU の動作周波数を予測し、クロック周波数の制約を決めることが出来る。このクロック制約の決め方は最初に動作周波数を与え論理合成を行った後, タイミング解析を行い 1 クロックの間に処理が実行できるかの時間的マージンを調整する。時間的マージンを厳しく、あるいは緩和していくことで最適な動作周波数決めていった．その結果 1 クロック 20ns の制約を与えることとした. 図 8 に論理設計の結果の一例として ALU の論理回路図を示す.

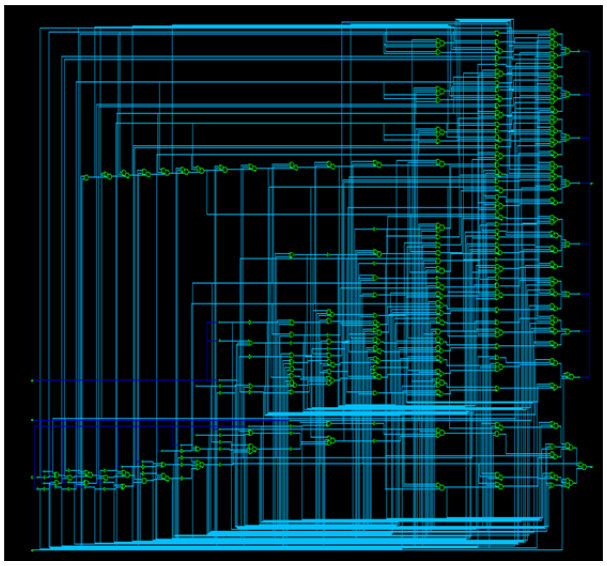

図 8　ALU の論理回路図

4・3　レイアウト設計

レイアウト設計では、まず CPU コア部のレイアウト設計から行う。このコア部のレイアウト設計では Synopsys 社の IC Compiler を利用した．このツールは論理合成後のネットリストからセルを自動的に配線するために用いた。配線は 2 層での配線の成功が見込めなかったため 3 層で行った. スタンダードセルの電源は 1 及び 2 層目で行い、信号線は 3 つの全ての層を使用した．基本的に縦方向信号線を 2 層目で行い，横方向の信号線を 1 層目で行い，どうしても配線できない場合のみ 3 層目を用いて配線を行った．電源変動による不安定動作を避けるために、チップコア回りと中央部に電源ラインを引き回した。チップコア部の面積は幅 788μm 縦 693μm となった．実際には，チップ内で CPU の入出力端子から外部に配線する必要がある．CPU コアのトランジスタは微小なため、そのまま入出力部に信号線を引き出しても駆動電力が足りなく外部からのノイズ・静電気に非常に弱い。そのため、CPU コアの入出力部から外部端子の間には、すべての端子に入出力を強化するバッファー素子及び静電気による内部破壊を避けるために ESD 素子を付けた。これらの CPU コア以外のチップ全体のレイアウトでは Jeadat 社の Ismo を使用した. 図 9 にチップコア部のレイアウト設計の結果を示し，図 10 にチップ全体のレイアウト設計の結果を示す.

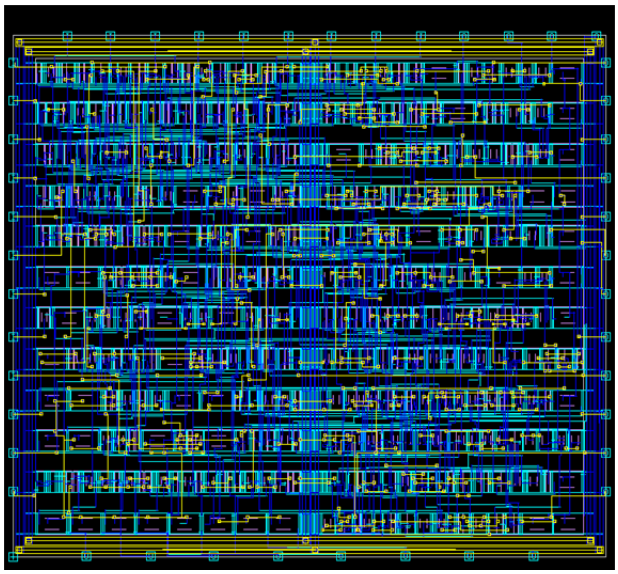

図 9　8 ビット CPU レイアウト図

4・4　製造

製造に関しては当研究室に製造する装置がないため, フェニテックセミコンダクター社の行っている大学等の研究・教育を目的としたシャトル便制度を用いて製作を依頼した．プロセスは 0.6μm である．図 11 に実際に製造されたチップ写真を示す．[9)]

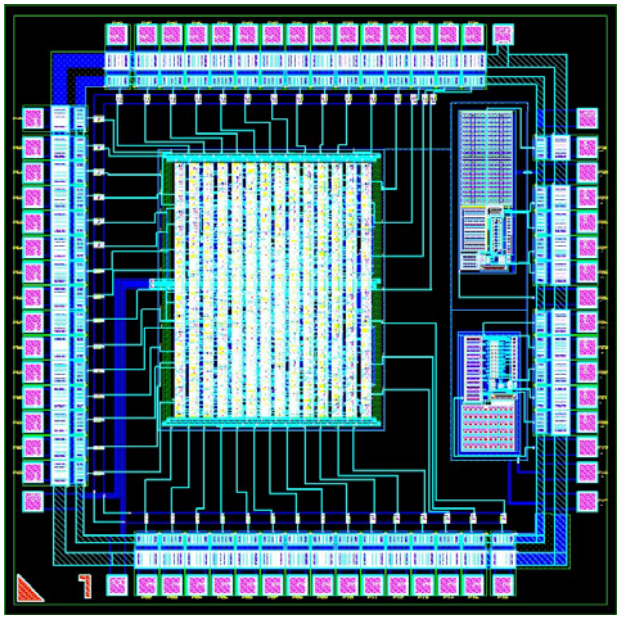

図 10　チップ全体のレイアウト

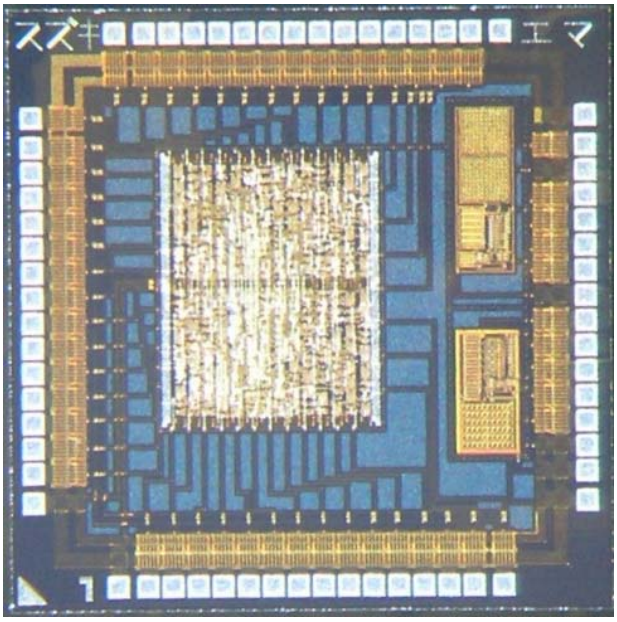

図 11　製造したチップ写真

4・5　動作検証

動作検証では Inovys 社の LSI テスタである Personal Ocelot と Stylus というツールを用いて以下の 3 点についての検証を行った.

・DC テスト：入出力端子の電圧レベルテスト
・AC テスト：入出力信号のタイミングテスト
・機能テスト：論理機能が正常であるか否かのテスト

DC テストに関しては「H (high レベル)」と「L (low レベル)」の電圧レベルを決めて検証を行った. AC テストに関しては入力信号ごとに入れるタイミングや, 出力信号を観測するタイミングを決めて検証を行った. 機能テストに関しては，設計仕様で定めた命令セット（13 種類）を全て 1 回実行するテストプログラムと、一番動作不良が生じやすい桁あふれの検証プログラムの 2 種類のテストプログラムに絞り機能テストの検証を行った. 今回この検証は 4 ビット CPU でのみ行った. 図 12 に動作が正常に終了した動作検証の結果画面を示し，図 13 に動作が正常に終了しなかった場合の結果画面を示す.

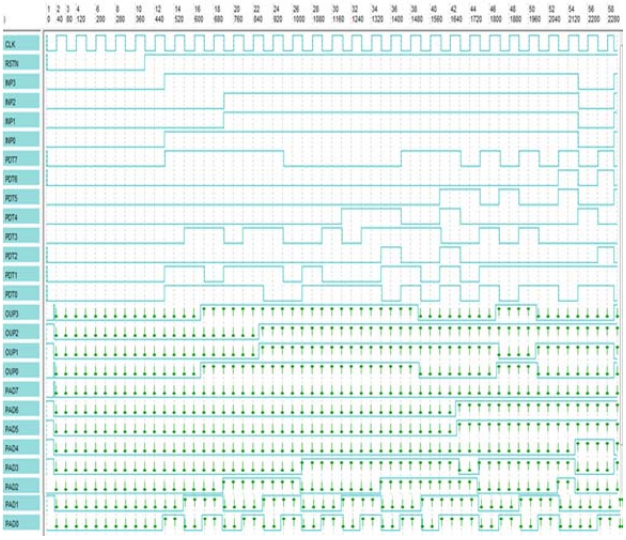

図 12　正常に終了した動作検証画面

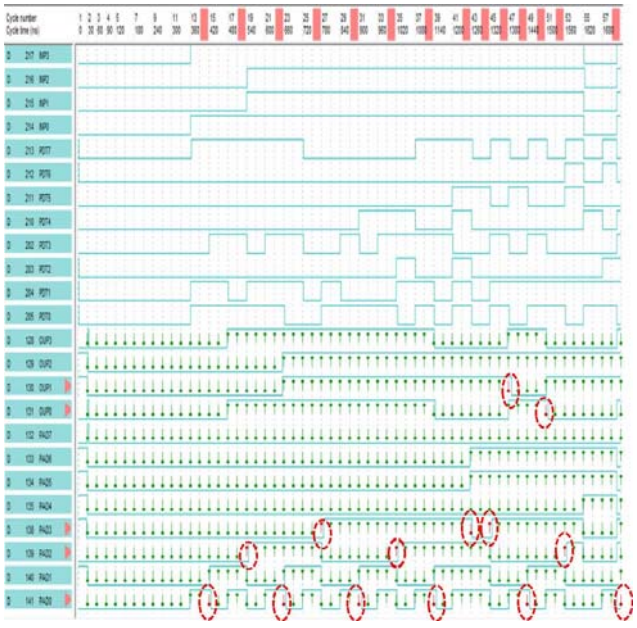

図 13　正常に終了しなかった動作検証画面

図 12 および図 13 において各信号の上下が DC テストにおける「H」と「L」を表す. また，図 12 および図 13 中のマークの位置が AC テストにおけるタイミングを表し，上下の向きがテストプログラムの期待値，緑の場合が正常，赤の場合が異常を示す. 図 12 の正常動作の場合は、

テストプログラムで設定した期待値と信号結果がすべて一致している。一方、図 13 では、テストプログラムの期待値と実際の出力信号が一致していない箇所(赤矢印)が表示されている事が分かる。DC テストに関しては各出力信号が「H」と「L」に変化しているため出力電圧は正常であるといえる．AC テストに関しては正常に動作する動作スピードの最大はクロック周期 80ns，動作周波数に換算して 12.5MHz であった．

5. 8 ビット CPU と 4 ビット CPU の処理時間の比較

本研究はデータパス幅の増大・処理時間の低減を目的とした新たな CPU の設計を目的としているため，本研究で設計した 8 ビット CPU と当研究室で以前設計を行った 4 ビット CPU とで目的の 2 点について比較を行った．1 つ目のデータパス幅の増大については Verilog HDL 記述内のデータパス部分やレジスタ部分を 4 ビットから 8 ビットに拡張しているため，データパス幅は増大している．2 つ目の処理時間の低減については演算を行うデータを 8 ビットとし，命令セットにある命令すべてを行うというテストプログラムを行いすべての処理が終了するまでの時間を比較した． 8 ビット CPU は 24 クロックサイクルでテストプログラムを終了し，4 ビット CPU は 43 クロックサイクルでテストプログラムを終了した．8 ビット CPU は 4 ビット CPU に比べ 44.2%少ない時間でテストを終えた．この理由として，4 ビット CPU では算術演算,論理演算,ビット操作命令を行う際,上位 4 ビットと下位 4 ビットを分割して演算を行う必要があるため，処理数に違いが表れた．表 3 に命令毎のクロックサイクル数の違いを示す.

表 3　命令による処理数の違い

	4 ビット CPU	8 ビット CPU
クロック サイクル数(算術)	7	3
クロック サイクル数(論理)	6	3
クッロク サイクル数(操作)	6	3
クロック サイクル数(その他)	1～2	1～2

6. まとめ

本研究では当研究室で作成した 4 ビット CPU をベースとしてデータパス幅やレジスタメモリ容量の増大・処理時間の低減を目的とした新たな CPU の設計を行った．以下に結果をまとめる

・データパス幅やレジスタメモリ容量の増大

Verilog HDL 記述のデータパス幅を 4 ビットから 8 ビットに拡張した．レジスタメモリ容量は 2 倍に増大した．

・処理時間の低減

4 ビット CPU は LSI テスタを用い動作周波数 12.5MHz であると測定した．8 ビット CPU は，LSI テスタを用いた検証ができていないので正確な動作周波数は不明だが,論理設計の段階で 1 クロックサイクル 20ns という制約を与えてあるため理論上は 50MHz となり,4 ビット CPU の約 4 倍で動作すると考えられる．また，命令の種類によっても変化するが、クロックサイクルに対する処理数は減少する．表 4 に 8 ビット CPU と 4 ビット CPU の動作周波数と命令処理毎に必要なクロックサイクル数から算出した処理時間を示す.

表 4　命令毎の処理時間

	8 ビット CPU	4 ビット CPU
処理時間(算術)	60ns	560ns
処理時間(論理)	60ns	480ns
処理時間(操作)	60ns	480ns
処理時間(その他)	20ns～40ns	80ns～160ns

以上のことから，当初の目的であったデータパス幅やレジスタメモリ容量の増大・処理時間の低減を目的とした AISIC 設計手法を用いた CPU の設計ができたといえる．

謝辞

本研究は，愛知工業大学 H26 年度共同研究 B，ALSI デザイン(株)，フェニテックセミコンダクター(株)，及び東京大学大規模集積システム設計教育研究センター(VDEC)を通じシノプシス株式会社の協力で行われたものである．

参考文献

1)　今井　正治：ASIC 技術の基礎と応用，電子情報通信学会　(1994)

2)　菅野　卓雄，堀口　勝治：ULSI 設計技術，電子情報通信学会　(1993)

3)　松尾　和典　他：熊本電波高専　研究紀要　第 34 号　11p　(2007)

4)　VDEC 監修:デジタル集積回路の設計と試作，　培風館　(2001)

5)　岩出　秀平，清水　徹：実用プロセッサ技術，　ムイスリ出版　(2009)

6)　David A. Patterson，John L. Hennessy：コンピュータの構成と設計，　日経 BP 社(2011)

7)　深山　正幸, 北川　章夫, 秋田　純一, 鈴木　正國：HDL による VLSI 設計-Verilog HDL と VHDL による CPU 設計-　共立出版　(2002)

8) 電子情報通信学会　総合大会　P-201　2014 年 3 月　“ASIC 設計手法を用いたプロセッサのビット幅拡張と並列処理による高速化”　鈴木貴斗　他

9) 電気学会 電子回路研究会 ECT-14-066, 2014 年 10 月　“ASIC 設計手法を用いた CPU の機能改善”　鈴木貴斗、江口一彦、五島敬史郎、山田明宏

（受理 平成 27 年 3 月 19 日）