設 計 自 動 化 71－7
（1994．6．24）

UDL／I第2期ソフトウェア開発
-合成系ソフトウェアのテスト-

神原弘之[1)]，横田吏司[1)]，遠藤真[2)]

1) 京都高度技術研究所
〒600 京都市下京区中堂寺南町 17 京都リサーチパーク

2) NTT LSI研究所
〒243 - 01 神奈川県厚木市森の里若宮3 - 1

あらまし
（社）日本電子工業振興協会の UDL/I (Unified Design Language for Integrated Circuit) 論理合成処理系の開発について，実施した動作テストの方法を中心に報告する．「UDL/I の言語仕様に忠実な論理合成を行う事」と「大規模な回路の UDL/I 記述を実用上問題なく合成できる事」の両者を保証するために行った UDL/I 論理合成処理系のテスト方法とその結果を報告する．

和文キーワード
論理合成，ソフトウェアのテスト，ハードウェア記述言語，ＣＡＤ

UDL/I 2nd Phase Development
- Testing Scheme for UDL/I Logic Synthesis System -

Hiroyuki Kanbara[1)], Satoshi Yokota[1)], Makoto Endo[2)]

1) Advanced Software Technology and Mechatronics Institute of Kyoto(ASTEM RI)
Kyoto Research Park, Chudoji Minami - machi 17, Shimogyou-ku, Kyoto 600 Japan

2) NTT LSI Laboratories
3-1, Morinosato Wakamiya, Atsugi - Shi, Kanagawa Pref., 243-01 Japan

Abstract
This paper reports the development projet of UDL/I(Unified Design Language for Integrated Circuit) logic synthesis system, focusing on testing scheme for UDL/I logic synthesis system. It is important to validate that the UDL/I logic synthesis sytem is implemented correctly to UDL/I Language Reference Manual and that the system can synthsize a large scale circuit practically.

英文 key words
Logic synthesis, Software test, Hardware Description Language, CAD

1. はじめに

（社）日本電子工業振興協会の UDL/I (Unified Design Language for Integrated Circuit) 論理合成処理系の開発について，実施した動作テストの方法を中心に報告する．

UDL/I は，論理合成系に与える回路の動作仕様を記述することを目的として設計されたハードウェア記述言語である．文法のみならず意味定義がその言語仕様書[1,2]で明確に記述されている．

1990 年 6 月より（社）日本電子工業振興協会の中に設置された UDL 開発委員会は，UDL/I 第 1 期言語仕様に基づいたUDL/I コンパイラとシミュレータからなる UDL/I 処理系の開発に着手した．2 年後の 1992 年 5 月，UDL/I 処理系は完成し UDL 開発委員会メンバにリリースされた．引き続き委員会は，第 2 期言語仕様への対応を中心とする UDL/I コンパイラとシミュレータの改良に加え，UDL/I 論理合成処理系の新規開発に 1992 年 9 月より取りかかった．1994 年 5 月に第 2 期言語仕様に対応した UDL/I コンパイラ，シミュレータ，UDL/I 論理合成処理系が完成しリリースされた．

1992 年 9 月より始めた UDL/I 論理合成処理系の新規開発では，UDL/I 論理合成処理系が

- UDL/I の言語仕様（文法と意味定義）に忠実な論理合成を行う事
- 大規模な回路の UDL/I 記述を実用上問題なく合成できる事

をテストを通じて保証することが重要であり，ワーキンググループでの議論を元にテスト方法を策定した．

本論文では，まずハードウェア記述言語 UDL/I の特徴と今回開発した UDL/I 論理合成処理系の外部仕様を紹介する．そしてテストデータの作成方法と実施したテストの方法と結果及び処理系の性能評価を報告する．

2. UDL/I 論理合成処理系の概要

2.1 ハードウェア記述言語：UDL/I について

1987 年（社）日本電子工業振興協会の中の LSI 設計用記述言語標準化委員会で，UDL/I の言語仕様の検討が行われてきた．1991 年UDL/I に第 1 版の言語仕様が策定された．その後も引き続き言語仕様のメンテナンスと拡張が行われ，1993 年に第 2 版の言語仕様に改訂された．

UDL/I は論理合成向けに設計されたレジスタトランスファレベルの ハードウェア記述言語である．単相クロックを用いた同期式回路設計に適しているが，非同期式回路も記述できる言語要素を備えている[3,4]．

言語の設計は，処理系の開発と独立におこなわれたため，特定のＣＡＤツールに依存していない．UDL/I を用いる論理回路設計者あるいは UDL/I の処理系のＣＡＤ開発者の間でセマンティクスに関する共通の理解を得るため，言語仕様書で意味定義が明記されている．意味定義は，UDL/I の言語要素のフルセットから，それと等価なコアサブセットへの変換規則により行われている[5,6,7]．

UDL/I 記述とUDL/I 論理合成処理系で生成される回路の例を図 1 に示す．

```
REGISTER:      REG;
REG := IF .COND THEN
                    AT .CLK DO .IN END_DO
               END_IF;
IF .RST THEN
          RESET( REG );
END_IF;
```

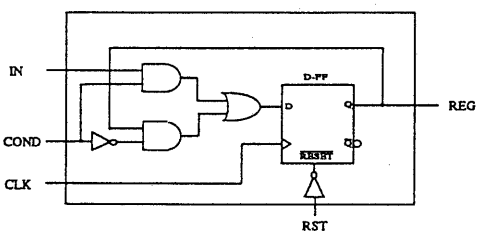

図 1 ：UDL/I 記述と合成された回路

2.2 UDL/I 論理合成処理系の外部仕様

（社）日本電子工業振興協会の UDL/I 処理系のシステム構成を図２に示す．

I）UDL/I コンパイラ
UDL/I 記述の文法チェックと中間フォーマットへの変換を行う
II）シミュレータ
UDL/I で記述された回路に，入力波形記述で表された入力波形が加えられたときの動作を模擬（シミュレーション）する．
III）UDL/I 論理合成処理系
UDL/I 記述を動作仕様とする回路を，セルライブラリ中の論理素子や記憶素子を用いて合成する．

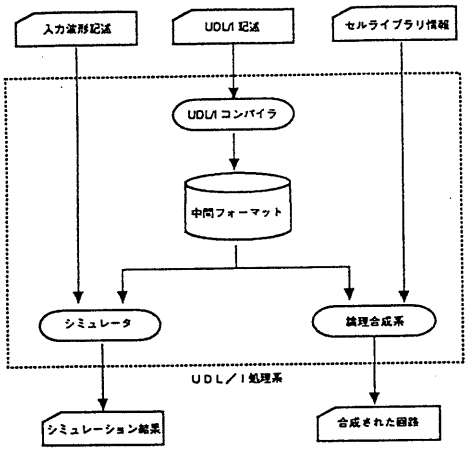

図２：UDL/I 処理系のシステム構成

UDL/I 論理合成処理系の合成の手順を図３に示す．各ステップの処理の概略は，以下の通りである．

1）MONAD への変換
中間フォーマット形式の UDL/I 記述を，UDL/I 論理合成処理系のデータ構造：MONAD[9] に変換する．
2）状態遷移機械の合成
UDL/I の <オートマトン記述> による状態遷移機械の記述について，各状態に２進のベクトルを割り当て，同期式の順序回路で実現する．
3）レジスタ合成
UDL/I 記述中のレジスタ，ラッチを，セルライブラリ中の D フリップフロップ，ラッチに置換する．
4）論理簡単化
UDL/I 記述中の論理式を，二段論理または多段論理で簡単化する．
5）テクノロジ・マッピング
UDL/I 記述中の論理式を，セルライブラリ中の論理素子を用いて組合せ回路で実現する．

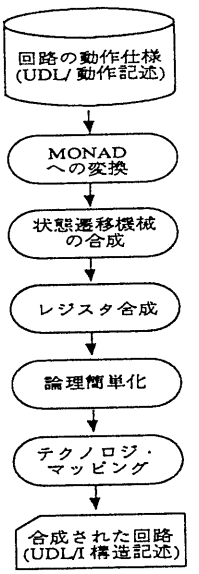

図３：合成の手順

UDL/I 論理合成処理系は，与えられた遅延制約の下で，合成される回路の面積の最小化をはかる．そのために

・状態遷移機械を順序回路で実現するときの状態割り当ての最適化
・論理式の簡単化
・テクノロジ・マッピング時の論理素子の組み合わせの最適化

を行う．なお，合成の段階でレイアウト後の配線領域の面積を見積もることは困難なため，回路の面積は，合成に用いられた素子の面積の総和とすることにした．

状態遷移機械を合成する際の状態割り当てには

- ワンホットコードの割り当て
- ランダムコードの割り当て
- Silicon Automation Systems 社の提案するFASTアルゴリズム[9]による状態割り当て
- Fintronic U.S.A 社の提案するアルゴリズムによる状態割り当て

からいずれかを選択できる．

論理の簡単化は，多段論理の簡単化または二段論理の簡単化を行う．多段論理の簡単化では，論理式を因数式で表わしたときのリテラル数を最小化することを試みる．簡単化は

- 米国のカリフォルニア・バークレー校が開発した多段論理簡単化ツール：MISII

または

- Silicon Automation Systems 社の提案するMULTI アルゴリズムによる多段論理簡単化ツール：SMULTI

のどちらかを選択して行うことができる．

二段論理の簡単化は，論理式を積和形で表現した時の積項数の最小化を行う．アルゴリズムは，米国のカリフォルニア・バークレー校が開発した二段論理簡単化ツール：ESPRESSO のアルゴリズムを改良したもの用いる．

UDL/I 論理合成処理系がマッピング可能な論理素子は，単純ゲート（NAND，NOR，AND，OR，XOR，XNOR），複合ゲート（AND-OR-Invert，OR-AND-Invert），インバータ，バッファ，セレクタ，デコーダ及び定数セルである．記憶素子は，D フリップフロップまたは J-K フリップフロップ及びラッチをマッピングできる．UDL/I のデータ形には RAM と ROM があるが，これらは RAM または ROM を含むモジュールのまま残し，合成は行わない．

マッピング時に考慮するセルライブラリの素子の特性は，遅延時間，素子の面積，ファンイン，ファンアウト制約，フリップフロップのセットアップ時間制約，ホールド時間制約である．遅延時間は，素子の負荷容量に対して線形に遅延時間が変化するモデルを使用して計算を行う．

設計者は，合成される回路が順序回路である場合，クロック周期とクロック位相を合成時の制約として指定できる．組合せ回路の場合は，入力から出力までのパス遅延の最大値を指定できる．

組合せ回路部分のテクノロジ・マッピングには，グラフ・マッチングに基づく Silicon Automation Systems 社の提案するアルゴリズムを採用した．

2.3 開発の方法

NTT 社から UDL 開発委員会に対して開示された論理合成フレームワーク：SYNTHESISKIT[9][10]に対し，下記の機能を追加することで UDL/I 論理合成処理系を実現した．

・UDL/I 第 2 期言語仕様への対応
・状態遷移機械合成機能の追加
・論理最適化機能の追加
・テクノロジーマッピング機能の追加
・ネットリスト生成機能の追加

動作環境は，SUN Microsystems 社の SPARC-station2 で，ＯＳは SUN-OS Ver.4.1.3である

新規に追加する機能を実現するツールの開発は，Silicon Automation Systems 社と Fintronic U.S.A 社が担当した．（財）京都高度技術研究所は，外部仕様の策定と新規開発ツールを含めた UDL/I 論理合成処理系のシステムの統合と検収を行った．

UDL/I 論理合成処理系は，第 2.0.3 版の言語仕様をサポートしている．

3. テストの概要

3.1 テストの目標

UDL/I 論理合成処理系のテストの目的は，以下の2点である．

・UDL/I の言語仕様（文法と意味定義）に忠実な論理合成を行う事

・大規模な回路の UDL/I 記述を実用上問題なく合成できる事

以下では，このテスト目標を達するために用意したテストデータとテストの実施方法について述べる．

3.2 テストデータ

「UDL/I の言語仕様（文法と意味定義）に忠実な論理合成を行う事」をテストするため，UDL/I の言語仕様書の BNF 定義をまんべんなくカバーするようなテストデータを作成した．1つのテストデータは，数十から数百ステップのUDL/I 記述とUDL/I 記述のテストパターン（入力波形と出力期待値）からなる．

BNF 定義をもらさずカバーする UDL/I 記述は，BNF 規則の右辺の非終端子を，その非終端子を左辺に持つ BNF 規則の右辺の非終端子で置き換えることで生成した．例えば，UDL/I の<代入文> のBNF 定義は，以下の通りである．

<代入文> ::= <連結> := <式> ;

<代入文>の右辺の非終端子<連結> と<式> の BNF 定義は

<連結> ::= <オペランド> {!! <オペランド> }
<式> ::= <単純式> [<遅延>]
　　| <begin_end 式>
　　| <条件式>

である．<代入文> が UDL/I 論理合成処理系で正しく処理されることを確かめるために，以下の8種類の <代入文> の UDL/I 記述を作成した．

1. <オペランド> := <単純式> ;
2. <オペランド> := <単純式> <遅延>;
3. <オペランド> := <begin_end 式>;
4. <オペランド> := <条件式>;
5. <オペランド> !! <オペランド> := <単純式> ;
6. <オペランド> !! <オペランド> := <単純式> <遅延>;
7. <オペランド> !! <オペランド> := <begin_end 式> ;
8. <オペランド> !! <オペランド> := <条件式>;

{!! <オペランド> }は，0回以上の繰り返しを意味するが，テストデータの作成では0回または1回の繰り返しの場合のみを作成した．

このようにして作成したテストデータのうち，論理合成処理系の入力となる回路の動作仕様を記述したものは 584 個あった．これらの UDL/I 記述は，言語仕様を幅広くカバーするものの，「大規模な回路の UDL/I 記述を実用上問題なく合成できる事」をテストするには

・状態数が数個以上の状態遷移機械の記述が少ない

・大規模な組合せ回路の記述がない

ことが問題であった．

上記のテストデータの不十分な面を補うため，MCNC/ISCAS のベンチマークデータを UDL/I 記述に変換してテストに用いた．その内訳は，

多段論理組合せ回路：56 個
二段論理組合せ回路：20 個
順序回路：43 個

である．

UDL/I 論理合成処理系のテストと現実的な評価を行うため，ASIC ベンダー数社の 0.8 μm プロセスを参考に，テスト用のセルライブラリのデータを作成した．ライブラリのセットは，参考にしたライブラリすべてに用意されていてかつUDL/I 論理合成処理系のテクノロジ・マッパがマッピング可能な論理素子と記憶素子を選んだ．このライブラリのセットを表1に示す．ライブラリの各素子のデータは，遅延時間，素子の面積，ファンイン，ファンアウト制約，フリップフロップのセットアップ時間制約，ホールド時間制約を含んでいる．

素子	種類
インバータ	駆動能力が1倍，2倍，4倍
バッファ	駆動能力が1倍，2倍，4倍
3 - Stateバッファ	
NAND/AND	2入力，3入力，4入力
NOR/OR	2入力，3入力，4入力
EXOR/EXNOR	2入力，3入力
Dフリップフロップ	Reset，Reset & Set，Reset なし
J-Kフリップフロップ	Reset，Reset & Set，Reset なし
ラッチ	Reset，Reset なし
複合ゲート	AND-OR-Invert，OR-AND-Invert
セレクタ	2to1，4to1
デコーダ	2to4，3to8

表1　テスト用ライブラリのセット

3.3 動作テストの実施

UDL/I の文法を幅広くカバーする５８４個のテストデータを用いたテストでは，UDL/I 記述から合成された回路について，元の UDL/I 記述のテストパタンを用いてUDL/I 処理系のシミュレータでシミュレーションを行った．そして，シミュレーション結果がテストパタンの出力期待値と一致することを確認した．一致しなかった場合，図３の論理合成の各ステップの途中のデータを逆コンパイラにかけて等価な UDL/I 記述に変換し，その UDL/I 記述をシミュレーションすることで，どのステップで誤った合成処理が行われているかを調べた．

MCNC/ISCAS の順序回路のベンチマークデータから作成された UDL/I の状態遷移機械の記述について，合成された順序回路が記述された通りに状態遷移することを確かめるテストは，シミュレータのインタラクティブコマンドの機能を利用して行った．合成された順序回路のシミュレーションを一時中断して，状態値を保持するフリップフロップにある状態値を強制的に代入かつ順序回路の入力信号値をセットした上，１クロックだけシミュレーションを実行した結果，フリップフロップが正しい次状態値になっていることを全状態について確認した．

MCNC/ISCAS の組合せ回路について，簡単化された論理式が元の論理式と等価であることは，信号値０と１の網羅的な入力波形を与えたときのシミュレーション結果が簡単化の前後で同じことを確認した．ただし入力数の大きいデータについては，シミュレーション時間あるいはシミュレーション結果のファイルサイズの問題で，入力波形は 1000 ステップ程度に縮小しなければならなかった．

3.4 性能テストの実施

多段論理最適化の性能については

簡単化前と簡単化後のリテラル数

簡単化に要する処理時間

をMCNC/ISCAS の組合せ回路のベンチマークデータについて測定した．

テクノロジ・マッピングまで行った合成結果については，Synopsys 社の Design Compiler v3.0c の合成結果と比較した．MCNC/ISCAS ベンチマークデータは，Verilog-HDL 記述に変換して Design Compiler に入力した．比較を正確に行うため，UDL/I 論理合成処理系のテスト用に作成した表１のセルライブラリと同じものを Design Compiler 用に作成した．比較では，回路面積最小化を指定した時の合成された回路の面積と最大パス遅延の値，ならびに遅延最小化を指定した時の最大パス遅延の値と回路面積を求めた．

3.5 自動テストシステム

以上で述べた動作テストと性能テストを，UDL/I 論理合成処理系が更新される度に円滑に行うため，自動テストシステムを（財）京都高度技術研究所は作成した．このテストシステムは，

・テスト手順（起動するツール名，起動の際指定するオプション，ツールの起動順序）

・用いるテストデータのセット

・論理合成処理系のバージョン

を指定すると，テストデータ毎にテスト手順に従ってテストを実施する．ツールが出力するメッセージや返り値などの情報をテストシステムは収集し，ツールが正常動作していることを確認しながらテストを進める．複数のツールを順番に起動する時にあるツールが異常終了した場合などは，そのテストデータについてはテストの実施を中断する．

テスト結果は

・テストの手順
・論理合成処理系のバージョン

を指定すると，テストデータ毎に起動したツールの動作結果を表示する．

4. 本処理系の評価

4.1 動作テストの状況

UDL/I の言語仕様を幅広くカバーするテストデータを用いた動作テストの進捗状況を図4に示す．図4は，テストデータ５８４個のうち，合成された回路のシミュレーション結果がテストパタンの出力期待値と一致したデータの個数を，論理合成処理系の各バージョン毎に表わしている．93年10月に動作テストを開始した時点では，約半数のテストデータしかテストを通過しなかった．その後のバグ修正とテストの繰り返しにより品質を改善し，７ヶ月間で動作テストを終了することができた．

4.2 性能テストの結果

多段論理最適化について，MISII を用いた場合と，MULTI アルゴリズムを用いた場合の，MCNC/ISCAS ベンチマークデータのリテラル数と簡単化に要した CPU 時間を表2に示す．計算機はメモリ容量 32 MB の SPARCstation2 を用いた．

表3には，UDL/I 論理合成処理系と Design Compiler について，面積最小化指定時の使用セル面積の総和と，遅延最小化指定時の最大パス遅延の値を示す．

5. まとめ

UDL/I 論理合成処理系の開発について，実施したテストの方法と結果を中心に報告した．言語仕様を幅広くカバーするテストデータと MCNC/ISCAS ベンチマークデータを併用することで，比較的短期間に品質を大きく改善することができた．また UDL/I 論理合成処理系の合成結果は，市販の論理合成系と比較して遜色のないことが確認された．

謝辞

テストデータの作成および各サイトでの動作テストに協力してだいたUDL 開発委員会のメンバの皆様に感謝します．また UDL/I 論理合成処理系の外部仕様とテスト方法を定めるにあたり，活発な議論を頂いた論理合成仕様WGのメンバの皆様に感謝します．Design Compiler の合成結果を求める際にご協力頂いた日本シノプシス社の桜井様に感謝します．

参考文献

1) UDL/I標準化委員会：UDL/I 言語仕様 第 1.0m 版，日本電子工業振興協会，1992.
2) UDL/I標準化委員会：UDL/I 言語仕様 第 2.0.3 版，日本電子工業振興協会，1993.
3) 星野，唐津：UDL/I，情報処理学会誌，Vol.33，No.11，pp.1244-1249，1992.
4) 唐津，星野，石浦，安浦：論理合成時代のハードウェア記述言語：UDL/I，電子情通信学会論文誌，Vol.J74-A，No.2，pp.170-178，1991.
5) N. Ishiura，H. Yasuura and S.Yajima：NES:The Behavioral Model for the Formal Semantics of a Hardware Design language UDL/I，Proc.27th DAC，1990.
6) H.Yasuura and N.Ishiura：Formal Semantics of UDL/I and Its Application to CAD/DA tools，Proc.of ICCD'90，1990.
7) 神原，安浦，P. Kukkal，H. KOBAYASHI，野地，小栗：ハードウェア記述言語の比較，情報処理学会誌，Vol.33，No.11，pp.1269-1283，1992.
8) 遠藤：UDL/I 第２期処理系開発 - 合成系ソフトウェアの開発 -，情報処理学会設計自動化研究会報告 71-6，1994.
9) Ashish Sirasao：FAST : A State Assignment Heuristic for Efficient Multilevel Logic Implementation of Finite State Machines，1993
10) 遠藤，高原：テンプレート化論理合成手法，情報処理学会設計自動化研究会報告 66-5，1993

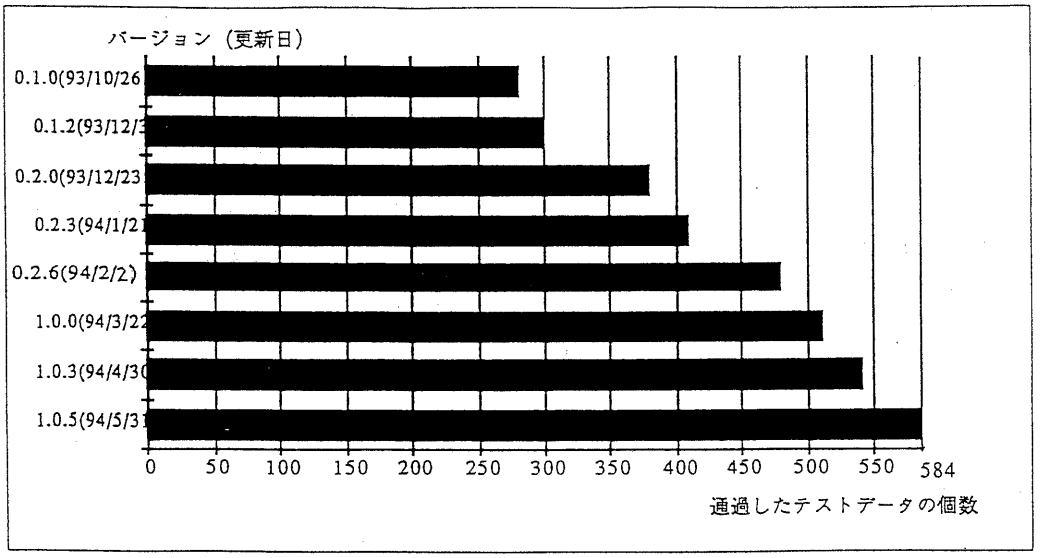

図4　動作テストの実施状況

表2 UDL/I 論理合成処理系の多段論理最適化の性能

データ名	元の	MULTI アルゴリズム		MISII	
		簡単化後の	処理時間	簡単化後の	処理時間
	リテラル数	リテラル数	(秒)	リテラル数	(秒)
5xp1	367	125	7.9	164	15.8
9sym	600	72	17.1	291	103.5
bw	374	182	30.6	225	9.5
rd53	173	42	2.1	95	5.1
f51m	316	170	8.6	167	11.3
sao2	497	202	11.8	227	36.4
frg2	2852	1012	93.4	1248	218.2
mux	165	54	2.1	100	13.9
c3540	2190	1368	647	メモリ不足のため終了せず	
alu4	1211	908	3204	929	68.8
apec6	1254	825	25	885	39.8

表3 UDL/I 論理合成処理系と Design Compiler の合成結果

データ名	面積の最小化指定時		最大パス遅延の最小化指定時	
	UDL/I 論理合成処理系	Design Compiler 3.0c	UDL/I 論理合成処理系	Design Compiler 3.0c
	セル面積の総和	セル面積の総和	最大パス遅延	最大パス遅延
5xp1	154	234	2.44	1.83
9sym	137	110	4.56	2.69
bw	393	348	2.76	1.38
rd53	85	43	1.34	1.6
f51m	215	274	3.56	2.05
sao2	346	287	2.75	1.92
frg2	2040	1547	3.16	2.89
mux	102	65	2.27	1.46
c3540	2099	2065	2.68	7.52
alu4	1574	1344	15.3	6.07
apec6	1454	1429	1.08	2.58