

TM-0802

繰り返し改善に基づく
アーキテクチャ設計支援法

野田泰徳(沖電気)

September, 1989

©1989, ICOT

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191--5
Telex ICOT J32964

Institute for New Generation Computer Technology

繰り返し改善に基づく アーキテクチャ設計支援法

作成された設計案の評価および改善の繰り返しプロセスとしてモデル化されるアーキテクチャ設計作業に対する一設計支援法について述べている。そして、この繰り返しプロセスモデルに基づいて試作した設計支援システムの概要を紹介し、システム実現上の問題点などに対して考察を加えている。

設計知識ベースを用いた本設計支援法は、繰り返し改善に基づく設計作業に対して有効な支援手段となる可能性がある。

野田 泰徳

沖電気工業(株) 技術本部 CAD部

1 論理装置のアーキテクチャ設計

アーキテクチャ設計とは、基本的な機能ユニット構成と処理方式とを定めるフェーズである。ここでは、全てについて新規設計を行うわけではなく、過去の豊富な設計例をもとに、これらをそのまま、或は適宜修正して設計を行うことがかなり多い。そこで本設計では、非常に高度な知識と創造性の発揮が要求され、最も経験豊富な設計者が担当することが一般的である。またその設計結果の評価は難しく、通常、設計した処理方式に基づく動作をシミュレータにより模擬し、その結果を観察することにより行われている。

ここでの設計活動は、設計仕様の設定、設計案の生成、設計案の評価、設計案の改善という四つのステップを考えることができる。

(1) 設計仕様の設定

このステップでは、設計対象に関する設計仕様の明確化が行われる。設計仕様としては各種考えられるが、これらは設計対象が果たすべき機能および要求される制約条件に大別できる。

a. 要求機能

ビット数、各命令の振舞い、など

b. 制約条件

性能(処理スピード)、コスト(回路規模)、開発期間、など

制約条件の設定においては、要求機能との関係においてその実現可能性の予測を行うことが必要であり、過去の豊富な設計経験に基づくヒューリスティクスが重要となる。

例えば、(このような簡単な命令機能だけなら**MIPSは可能であろう)、

(この性能をこのコストで実現するのは難しい)、

(この開発期間で実現するためにはゲートアレイを採用すべきである) など

(2) 設計案の生成(詳細化)

与えられた要求機能を制約の満足する範囲内で実現するために、仕様をブレイクダウン(詳細化)し設計案を生成するステップである。ここでは、制約を満たすような設計案を生成するために、ヒューリスティクスに基づく各種の設計者のアイデアにより詳細化方法が決定される。例えば、詳細化方法としては、以下の項目に対する取り組み方などがあげられる。

- ・ n 段パイプライン処理方式、
- ・ 命令/データキャッシュメモリ、
- ・ 命令形式 …… etc.

この詳細化方法の決定においては、各制約条件の厳しさに対応して、それを満足させるような方法に関する知識が用いられる。例えば、(この性能を満足させるためには、データおよび命令用に異なるキャッシュメモリを用いるべきである)などが有る。

(3) 設計案の評価

生成された設計案が制約条件を満足しているかどうかを評価するステップである。設計

案に対する各種シミュレーションや解析プログラムによる出力データをもとに評価を行うのが一般的である。処理スピードや回路規模などの制約条件に対する評価は、これらの出力データを用いて行う事が可能であるが、これらのデータの精度はそれほど高く無く、データの解釈において高度の知識が必要とされている。

(4) 設計案の改善(最適化)

制約の満足度を考慮して選択した詳細化方法に従って設計案を生成したにも拘らず、評価の結果から制約条件を満足していないことが明らかになった時、この設計案の部分的変更により制約条件に対する評価の改善(最適化)を狙うステップである。ここでは、評価結果の解析から設計案のどの辺りを変更したら改善できるかを絞り込み、設計者の設計経験にもとづいて改善の戦略を決定し、この戦略に従って具体的変更を実施している。そして、各制約条件を満足するまで評価および改善が繰り返されている。

以上の各ステップを、そのフローに着目して整理すると図-1のようになる。ここで、設計のフィードバックにおけるAは設計案の改善を行った後の再評価を、Bはこれ以上の改善が望めないため他の詳細化方法により新たな設計案を作りそれをもとに改善を進めることを、Cはどのような詳細化を行っても制約を満足するような改善が望めないため制約条件そのものの緩和を図ることを意味している。

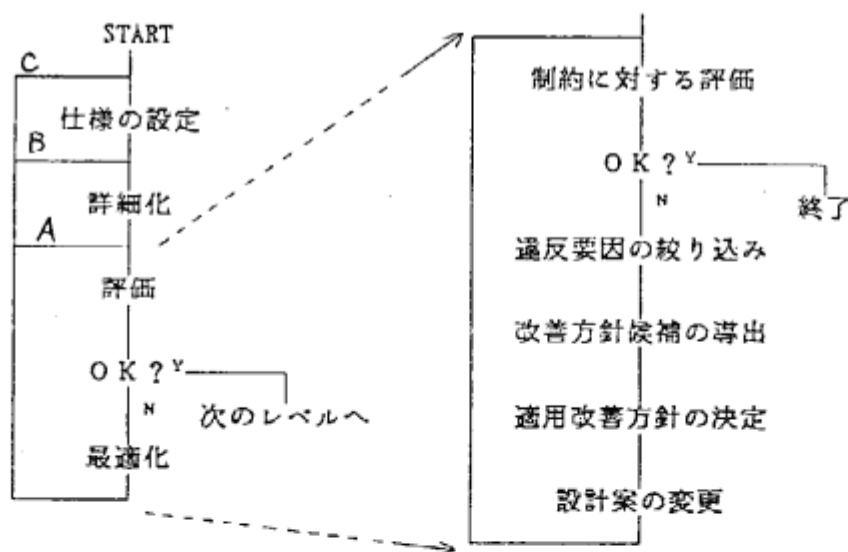


図1 改善の繰り返しプロセス

2 設計支援用実験システム

ここでは、改善の繰り返しプロセスに基づくモデルをもとに、アーキテクチャ設計の支援をターゲットとして試作した実験システムについて記す。

先ず試作した実験システムの構成を図2に示す。評価モジュールは、要求される制約条件に対する設計案の評価を実施する。解析モジュールでは、評価結果において制約違反が生じた時、その違反要因を導きこれを解消するような改善方針の候補が導かれる。そして、改善モジュールでは、これらの改善方針候補の中から実施すべき方針を決定し、これに従って設計案の変更を行う。ここで、これらの一連の処理は、知識ベースに蓄えられている改善知識に基づいて行われている。

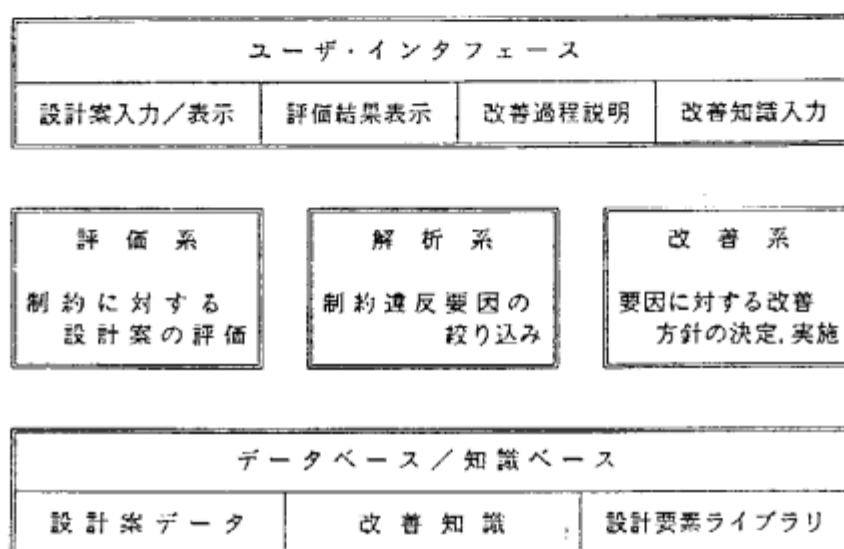


図2 システム構成

(1) 設計データ

アーキテクチャ設計における設計案を表現する設計データとして、対象計算機のデータベース構造および命令定義を考える。データベース構造には、計算機を構成する主要なコンポーネントの接続関係が表現される。命令定義には、実現されるべき命令毎に、データベース上でどのようなデータ転送を行ってその機能が実現されるかが表現される。ここで、設計対象としてパイプライン処理方式計算機を仮定しており、このデータベース転送は各命令をステージという実行単位に分割して記述される。図3にデータベース構造および命令定義の例を示す。

(2) 評価モジュール

アーキテクチャ設計案の評価は、サンプルプログラムに対するシミュレーションを実施することにより行われる。このシミュレーションは、命令定義の記述に従いサンプルプログラムをパイプライン処理に基づいて可能な限り並列に実行させることにより行わ

れる。この時シミュレータは、各命令の機能を実現するために必要なデータ転送が、複数命令の同時実行にも関わらず全体として矛盾なく行われるように管理している。

(3) 解析モジュール

設計案が制約条件に違反している時、その違反要因を導き、これを解消するための改善戦略の候補を導出するモジュールである。

違反要因の導出は、次のように、シミュレーションを実施した際の種々のデータを解析して行われる。性能(処理スピード)に関する制約について違反している時には、例えばデータ転送において利用要求の競合が多発しているコンポーネントを違反要因と推定する。これは「性能が低下するのは、パイプライン処理における複数命令の実行においてコンポーネントの利用競合が発生した時、後から実行が開始された命令において実行待ち(インターロック)が発生するため」というヒューリスティクスに基づいている。またコスト(回路規模)について違反している時には、例えば利用要求の度合いが低いコンポーネントを違反要因として導いている。これは、利用要求の度合いが低いコンポーネントは、他のコンポーネントでその機能を代替することが比較的容易であり、それによりこれを削除することが可能となってコストを下げることができる」というヒューリスティクスに基づいている。

次に本モジュールでは、これら要因としてのコンポーネント毎に、改善知識が蓄えられている設計知識ベースから違反要因を解消するための戦略(改善プランと呼ぶ)の候補を導き出す。この改善プランの候補は、改善すべき制約項目、要因の種類および要因と考えられるコンポーネント名を用いて設計知識ベース中から特定される。表1に改善知識の一例を示す。

(4) 改善モジュール

導かれた改善プランの候補の中から、現設計案に対して実際に適用できるものを抽出し、この中から実施すべき改善プランを決定する。そして、このプランに基づいて設計案の変更を実施するモジュールである。

設計知識ベースに格納されている改善知識は、改善プラン毎に適用可能かどうかを判定する手続きと設計データに対する具体的変更手続きを持つ。判定手続きには、改善プランの適用を可能とするために設計データ(ここではデータベース構造)が具備すべき条件が記述されている。また変更手続きには、データベース構造をどのように変更するか、そして、それに伴い命令定義がどのように変更されねばならないか、が記述されている。ここで、現設計案を作成するにあたり各コンポーネントをどのような目的で利用していたか等の情報(設計意図の一部)が必要となるケースが有り、この時には設計者に問うことにより情報を得ている。

図4に改善プラン適用の過程の一例を示す。ここでは性能の改善において、利用要求の競合が多発しているコンポーネントが表示され、それらの各コンポーネントについて改善プランの候補を知識ベースから探索し各々についてその適用の可能性をチェックした結果、4個のコンポーネントについて適用可能な改善手続きが存在することが示されている。

(5) 設計プロセス管理モジュール

改善の繰り返しにより種々の設計案が生成されるが、本モジュールは、これらを統一的に管理するとともに、改善の繰り返しのプロセス自体を管理して効率的に設計解へ到達できるように支援するものである。

図5に設計案の改善を繰り返した時のデータベース構造の変化の履歴の例を示す。また図6は、この時の各制約に対する評価結果の履歴である。設計者はこの評価履歴を参考にして、制約条件の満足度のようすを把握することが出来る。更に、改善の途中においてこれ以上の改善が期待できないと判断されたときには、以前の古いバージョンの設計案へバックトラックし、その時点で選択しなかった他の改善プランを適用することにより、設計解の探索を効率的に継続することが可能となっている。

3 設計支援システムとしての立場からの議論

(1) 複数制約の同時改善の問題

設計案の評価の結果、複数の制約について違反が有ることが明らかになった時、これらの制約の改善を同時に考慮することは困難(複数制約間の因果関係を設計知識として整理することが實際上困難)であり、改善対象として、どの制約を先ず考慮するかは利用者の判断に委ねられている

(2) トータルの評価の効率的改善

最終的に複数の制約項目を満足させることが必要であり、これを効率的に(少ない改善操作で)実現するため、改善対象の制約項目と他の制約項目に対する影響とを総合的に評価して適用すべき改善プランを決定するメカニズムが必要となる。

(3) 試行錯誤的繰り返し改善過程の実現

ここでは特定の制約に着目した改善を考えているため、改善の繰り返しにおいてトータルの評価が悪化することもあり得る。そこで、任意のバージョンの設計案へバックトラックし、この時点からの改善を再び行うことのできる機構が特に有用となる。

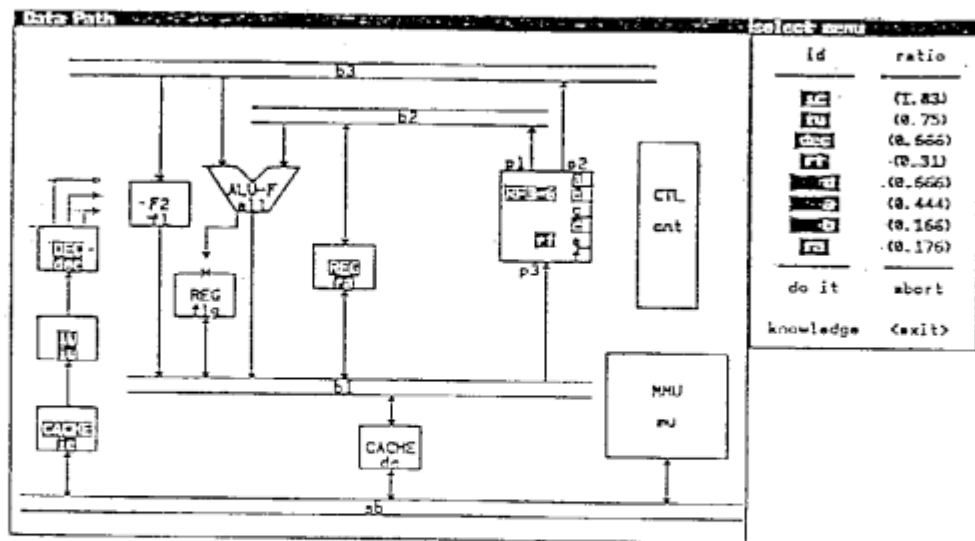
(4) より上位の改善戦略の取り扱い

ある戦略のもとに複数の改善プランを連続的に適用することが有効なケースが有り得る。そこで、連続した一連の改善プランの適用を制御するためのメタレベルの改善戦略を扱えることが望まれる。

表-1 改善知識の一例

改善項目	要因項目	箇所	優先度	改善方針	適用可能性チェック手続き	設計案変更手続き	コストへの影響	性能への影響	仕様書No
性能	インタロックが多い	バス	1	バスの再構成	要因であるバスをよよく利用する転送ルートの中で、他の転送ルートと完全に独立な関係にあるものがあるか	独立関係にある転送ルート及びファシリティをバスから分離させ、命令定義におけるその転送ルートを変更する	+(バス分)	UP (?)	1
			2	別のバスの利用	要因であるバスをよよく利用する転送ルートの中で、迂回ルートに代える事によって同一クロック転送ができ、且つその迂回ルートで利用するファシリティの競合度の少ないといったものがあるか	命令定義における一部命令(人手決定)の転送ルートを迂回ルートに変更する	±(0)	±(?)	2
			2	バイパスルートの利用	バイパス化できるデータバス構造になっているか	要因であるバスをよよく利用する転送ルート(人手決定)のファシリティ間にバイパスルートを設け、命令定義におけるその転送ルートをバイパスルートに変更する	+(バイパス分)	UP (?)	3
	レジスタ	レジスタ	1	レジスタの追加	要因であるレジスタを並列に加えることができる構造になっているか	要因であるレジスタと同一種のレジスタを並列位置に付加し、命令定義における一部命令(人手決定)の転送ルートを同一種レジスタ回りに変更する	+(レジスタ分)	UP (?)	4
			2	利用レジスタの変更	要因であるレジスタの代替となるレジスタが並列位置にあり、且つそのレジスタの競合度が他のファシリティの利用度よりかなり低いものがあるか	命令定義における一部命令(人手決定)の転送ルートを代替レジスタ回りに変更する	±(0)	±(?)	5

制約違反の要因の導出 (performance の違反)



適用可能改善方針の絞り込み
(performance の改善)

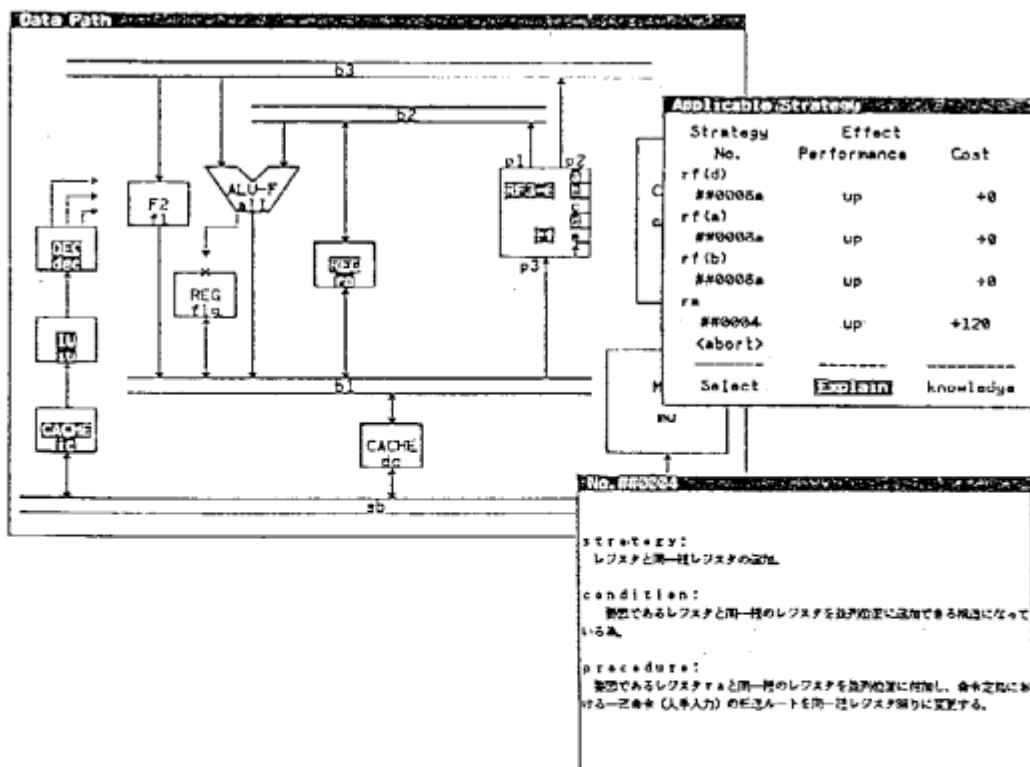


図-4. 改善プラン適用の過程の一例

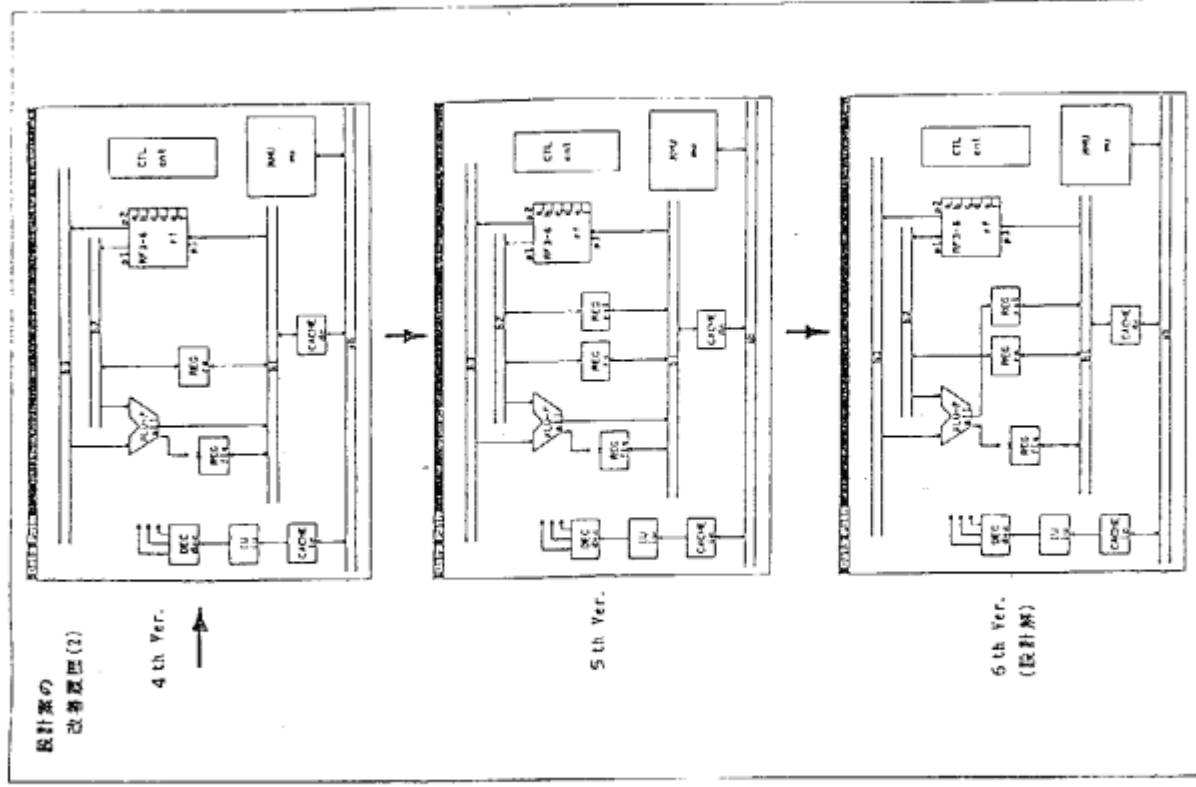
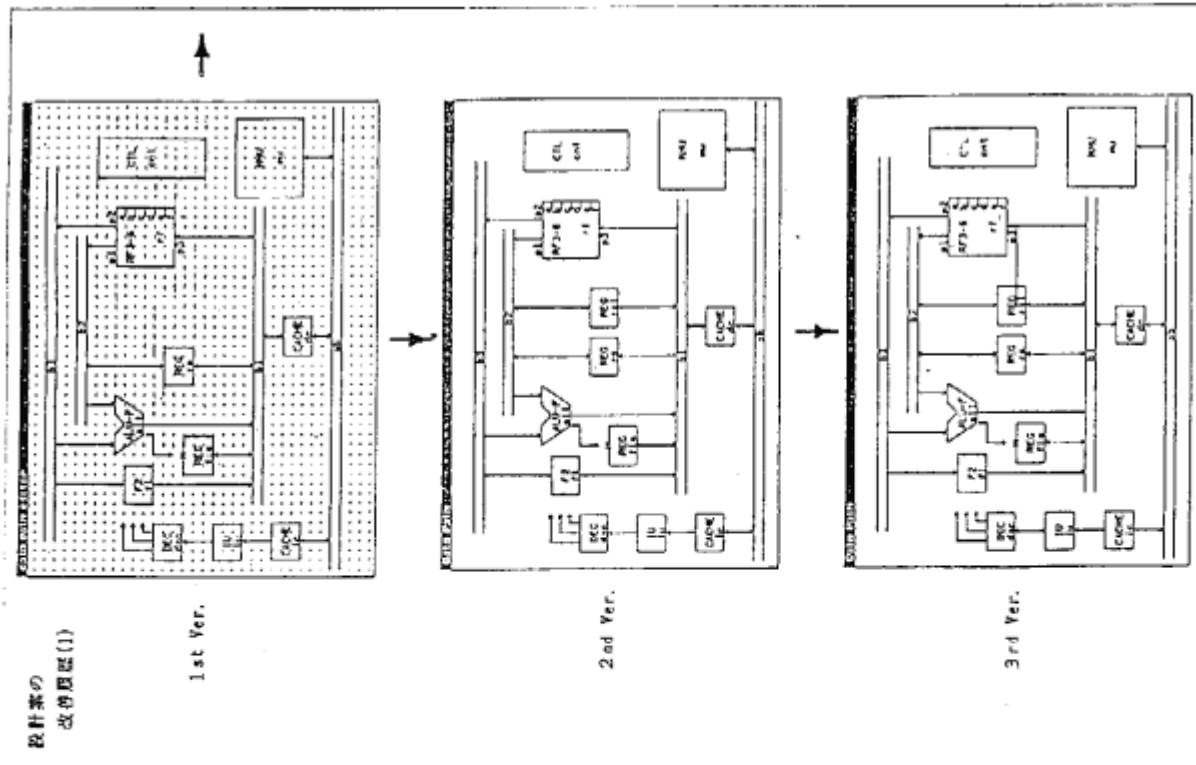


図-5 データバス構造の改善履歴の例

設計案の変更に伴う評価データの変化の履歴

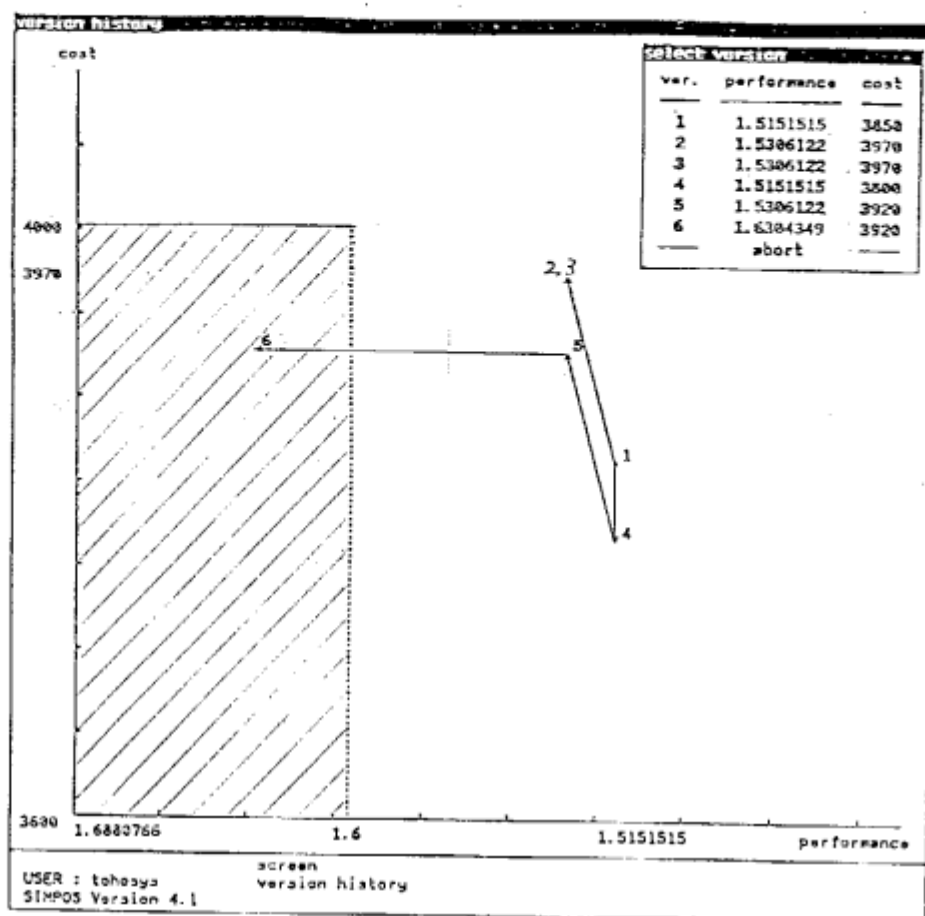


図-6 改善に伴う
評価結果の変化の履歴

以上