設 計 自 動 化 66－6
（1993. 3. 12）

LSI 協調設計支援システムの概要

細田泰弘

NTT LSI 研究所

LSIの設計は複数の設計者の" 協調"を必要とするグループワークである。本報告では設計の特質を考慮したチームによるASICの機能/論理設計過程のモデルと設計の規範としての"協調設計"を提示 する。これは要約すると設計の"プロセスとプロダクト"をあらかじめ定義し、それに従って進める設計である。最後にこのような設計スタイルを支援するために市販フレームワーク上に構築したシステムの概要について述べる。

A new LSI Design Environment based on the Collaboration Model of Engineer's Team

Yasuhiro HOSODA

NTT LSI Laboratories

3-1 Morinosato-wakamiya,Atsugi-shi,Kanagawa-ken,Japan

LSI design is a collaborative work process of engineer's team. In this paper, we first present a model of ASIC function and logic design team. "Collaborative design" is introduced for this team. At the center of this design is the definition of the process and the product of design. This definition facilitates communication between the designers and computer support for design.We conclude this paper by presenting the collaborative design support system built on a commercial framework.

1. はじめに

近年LSIの設計環境はEWSのコストパーフォーマンスの急激な向上、CADソフトの充実により長足の進歩をとげている。にもかかわらず設計の効率そのものはかならずしも期待したほど改善されてはいない。報告者の最近の設計経験[1][2]などから考えるとから原因は次の 2 つに帰着される。①設計者は多数の設計ツールを使いこなしながら相互に関連づけられていない多量の設計データを管理する必要があり、負担が大きくまた誤りも入りやすい。②設計情報の理解、管理、共有のために現在十分な計算機支援のない多くの業務(ドキュメント作成、打合せなど)をこなす必要がある。

①の問題を解決するためにCADフレームワーク[3][4]が提唱された。ただし現状ではCADフレームワークそのもの、あるいはその上に構築されたシステムも発展途上にあり、まだ十分な問題解決をみたとはいい難い。

②の問題はLSIの設計過程は複数の設計者の**協調**を必要とするグループワークであることに起因する点が大である。**協調**については現在種々の立場から研究がなされ、支援ツールとしてのグループウエア[5][6]が関心を集めている。

グループウエア導入の前提としては個人レベルの情報の作成支援、電子化が必要である。それを支援する道具としてDTP、表計算等は徐々に充実し、かつEWSでも使用可能になってきている。パソコンのほうがこれらのソフトの種類が多く、安価であることは否定できないが、CAD用としてはやはりEWSは必須であり、EWSでなるべく閉じて仕事ができればそれに越したことはない。

また、協調という立場から設計をモデル化し支援する研究も各所で行なわれている。[7][8]LSI設計に対してもこのような立場は有効であるが、LSI設計の特質や種々の設計ツール、設計データとの関係を十分踏まえたものである必要がある。

本報告では"協調"に基づくLSI、特に ASICの設計過程のモデルと CADフレームワーク(Power-Frame)上に構築した支援システムの概要について述べる。

まず、2.では協調と設計問題との関係を論じる。

2. 設計問題と協調

なぜ設計がチームにより行なわれるか考えてみると主な原因は次の2つである。

1. 問題が複雑すぎて一人で行なうと許容されるタイムリミットまでに終らない。
2. 問題が種々の分野から構成されている。一人で解こうとすると各分野について専門家にならなければならない。それよりも分業をはかった方が効率がよい。

分割には一般にかならず分割損が伴う。分割損を最小にするためには次のことが必要である。

1. 対象を機能的になるべく独立した部分に分け、設計途中でのやりとりを最小にする。ブレインストーミング等の場合を除き、複数人で一緒に考えるのは生産性を低下させる。
2. 各担当に担当部分の責任を持たせる。複数人の共同責任にすると手抜きがおき、品質が低下する。
3. 各自が勝手な問題解決に走ると部分解を結合して全体の解を得ることができない。プロジェクトとしての標準作成や個人の解に対してのグループとしての承認が必要である。
4. 各自が考えたノウハウや解を他人が容易に利用できるようにしておかないと多くの無駄を生じる。

これらの目標を達成するために設計管理の必要性が生じる。この際上下の関係である"管理"という立場に終始すると管理される側の反発を買う、組織の階層における同一レベル間の意志疎通が十分行なわれないなどの問題が伴う。もう一段進んだ立場に立ちチームの生産性の極大化をめさすのが協調である。すなわち①設計チーム内での必要十分な意志疎通、相互理解がおこなわれる。②設計情報が適切に管理され、チームのメンバーが必要な情報を遅滞なく入手できる。ことを重視する。これに沿って設計のモデル化と支援をはかることにより"効率の良い、誤りの少ない、再現可能な"設計の実現が可能になる。

3.ではLSI設計の特徴と設計チームのひとつのモデルおよび彼らが従うべき規範としての"協調設計の姿"についてのべる。

3. 設計のモデル

3.1. LSI 設計の特徴

ソフト開発などど比較した論理系ASIC設計の特徴について述べる。

1. ASIC設計の1プロジェクトに関与する人数は2~3人から数十人位であり、大規模ソフトの開発より1桁以上少ない。その分モデルが作成しやすい。
2. 各人に設計管理者、機能設計者、論理設計者、レイアウト設計者、テスト設計者等、比較的内容のはっきりした役割がある。ただし、機能設計/論理設計の境はかならずしも明確ではない。
3. 設計の初期段階(チップのブロック構成決定まで)を除きCADツールによる支援が普及している。
4. 現在の設計手法はすべてチップの階層化によるブロック分割を基本にしている。
5. 設計フェーズのなかで特に機能/論理設計部分はチップの機能ブロックごとに複数人で並列に進む。
6. 設計ミスをした場合のコストがきわめて大きく、プロセスやり直しになることが多い。このことは"きちんとした、バグのない設計"に対する厳しい要求を意味する。

次に2.で述べた協調のあり方とこれらの特徴を踏まえて構造(structural)、機能(functional)、動的(dynamic)の3つの視点から設計チームの一つのモデルを提示し、彼らが守るべき規範としての協調設計の定義を示す。設計フェーズとしては通常設計工数に占める割合が最も高く、設計の並列度も高い機能/論理設計フェーズに限定する。

3.2. 設計チームの構造モデル

ソフト開発チームの3つのモデルが[10]で紹介されている。(図1)b)は企業内での設計のモデルとして適当でない。c)のように管理を階層化することはよく行なわれる。ただし数十人程度のチームではサブチームのリーダーが技術面での指導を行ない進捗管理等はチーフリーダーが直接行なうことも多い。設計者からは両者の責任分担がはっきりしており、一人の仮想的な"管理者"という形で一体化して見えることが望ましい。プロジェクトのなかで2重の命令系統が存在することは混乱を招く。ここではa)の構造(リーダーは一人だけ)を基本として考える。

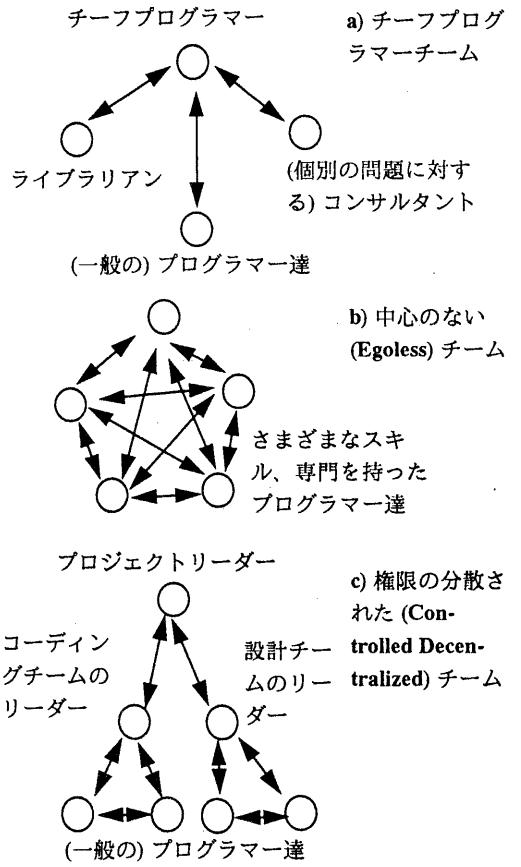


図 1 設計チームの構成のモデル

3.3. 設計チームの機能モデル

チーム内での担当者の役割と相互作用のモデルについて論じる。IStar[11]のモデルは契約者が仕事を一人でこなせないと判断した場合、任意にサブ契約者を雇って仕事の一部を分担させるものである。(図2の(a))仕事の一部を他の設計者に頼むことは開発の現場でしばしば起こるが実際は管理者の助言と承認に基づかなければプロジェクトが維持できない。ここでは次のようなモデルを考える。(図2の(b))

"各設計者は対等の決定権を持つ。設計者間で意見の調整が必要な時は管理者がその任に当たる。設計の都合上、他人の担当ブロックを手伝うことはあるがあくまでも例外である。"

もちろん管理者はチーム内で十分な意思疎通が行なわれていること、そのために不必要なコストが払われていないかを常に注意する必要がある。

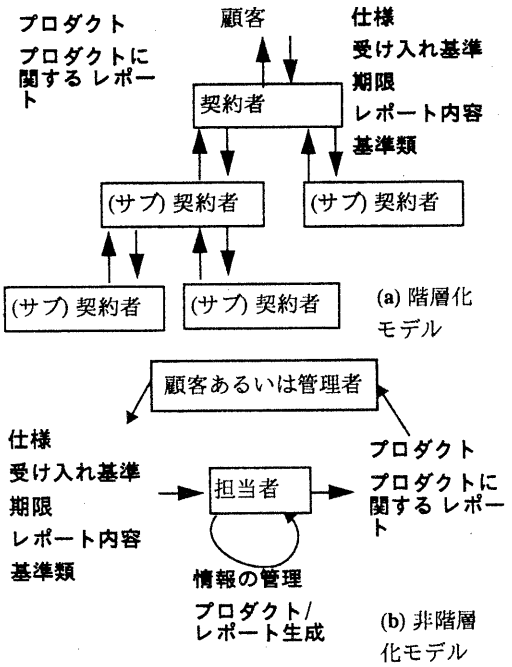


図 2 設計のチームの機能モデル

3.4. 設計チームの動的モデル

次に時間軸方向のモデルを考える。ソフト設計においてはwaterfall,spiralの2つの有名なモデルがある。[12]ここで提唱するモデルはspiralに近く、具体的な内容をASIC設計に即して考えたものである。次の4つのフェーズに分けて考える。

フェーズ1(基本設計)

担当ブロックの機能、設計手法を理解する。それをもとに担当ブロックのサブブロックへの分割や実現方法を検討する。場合によりこのフェーズに戻って設計変更を行なうこともある。

フェーズ2(詳細設計)

担当ブロックのネットリストを論理図入力、機能記述言語等を用いて入力し、ネットリストを作成する。このときテスト項目、機能概要等の設計ドキュメントも作成しながら進める。検証やレビューの結果によってはここに戻ってネットリストやドキュメントを修正する。バグ修正のためにはバグ票を発行する。検証が終了すると必要なドキュメントを揃え、かつファイル等を整理して管理者に提出可能な形にする。

フェーズ3(設計検証)

ネットリストに対して動的/静的な検証を加える。動的検証のほとんど唯一の手段はシミュレーションである。あらかじめ作成した検証項目に対応してテストパターンを作成し、シミュレーションを行なって確認する。静的な検証手段としてはネットリストの机上チェック及びチェックプログラムの実行がある。

フェーズ4(レビュー)

設計のマイルストーンごとにレビューを行なう。設計終了後、管理者は提出されたプロダクトをチェックし、OK(Clean File認定)がでればそのブロックの設計を終了できる。

以上を図3に示す。

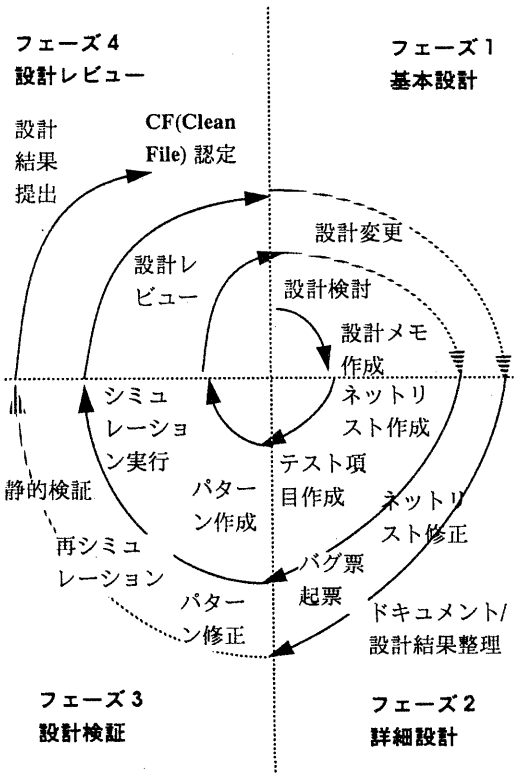


図 3 設計チームの動的モデル

一度CF認定を受けてもその後チップレベル等での追加シミュレーションでバグが見つかることもあり決して安心できない。また途中で外部要因による設計変更を行なったり設計途中での担当者の交替等さまざまな予期せぬ事態が生じうる。

3.5. 協調設計の定義

3.2.~3.4.に述べた設計チームがどのような設計をすれば"効率の良い、誤りの少ない、再現可能な"設計が可能か考える。3.4.で述べた4つのフェーズは次のような役割を持っている。

- 設計の計画 (plan)
- 設計の実行 (do)
- 設計の個人レベルでのチェック
- 設計のチームレベルでのチェック

後ろ3つのフェーズでよい設計をするためのはフェーズ1の計画で何を行なうかが鍵になる。ここでは協調設計を次のように定義する。

設計チームのメンバーが自分の設計(のプロダクトとプロセス)を定義し、それを他のメンバーおよび設計環境(計算機)に教えてやり、それにより相互の意思疎通を行ない、かつ必要な支援、チェックを受けながら進めていく設計

設計プロダクトは設計の終了までに生成される設計の階層構成、設計データ(ファイル、ドキュメント)とその属性、データ間の関係すべてである。設計プロセスはどのツールを使用し何を行なえば得たいプロダクトが生成されるかを示す粒度の細かいプロセスから設計者が一つの設計ライフサイクルのなかでどのような手順で設計を進めるかを示す粒度の粗いプロセスまでを含む。

これを実現するためにはもちろんマネージメント上の問題も重要である。同時に支援するシステムを構築することが必須であり、これを協調設計支援システムと名付ける。システム構築のためにはプロセス/プロダクトを定義するためのデータベースがまず必須である。

プロセス/プロダクトの定義を設計の開始時に完全に行なうのは無理があり、開始時に概略を定義し、それを詳細化して完全な定義にもっていくことが現実的である。図4に協調設計の(動的)モデルを示す。

図4の中見は新規設計と流用設計で当然違ってくる。流用設計の場合はフェーズ1において既設計のどの部分をそのまま使い、どの部分を変更し、どの部分を再設計するか計画するのが重要になってくる。

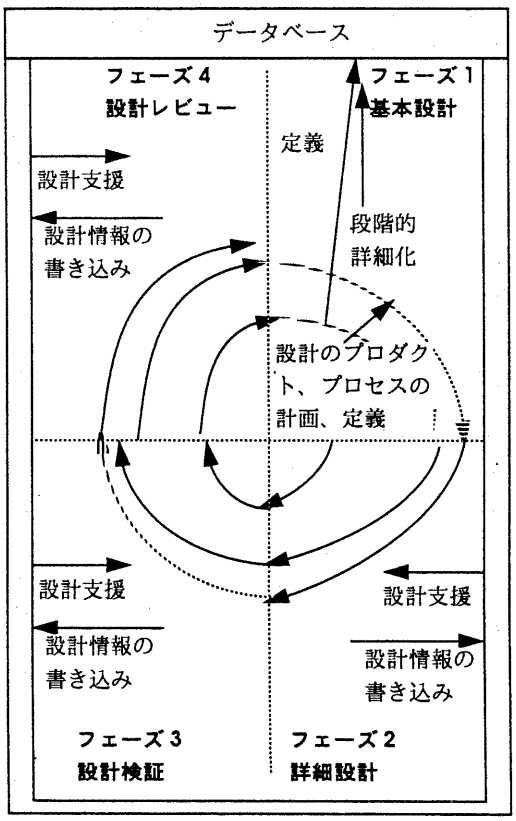


図 4 協調設計の動的モデル

4.では支援システムを実現するにあたって何を考え、いかに進めるべきかについて述べる。

4. 支援システムの実現

4.1. 協調に関する研究動向

現在、分散と協調は情報関連分野におけるひとつのキーワードとなっており、AI/グループウエア/ソフトウエア工学/データベース等さまざまな方向から研究が行なわれている。協調を扱うのに次の2つの代表的な立場がある。

AI(とくに分散AI)の立場の人は利害の対立やリソースの制約、情報の欠如といった困難な問題を解決してゴールに達するのを目標にした研究が多く[9]、また協調する主体がロボットやプログラムであることが多い。これに対してグループウエアではこのような問題の解を与えるよりグループとしての作業効率の向上を指向したシステムが多

く、また協調する主体は人が基本である。

4.2. 協調設計支援システムの立場

協調設計支援システムでの協調の主体は人であり、また設計途中にブロック間インターフェイスなどで発生する可能性のある利害の対立調整までは踏み込まず解決を間接的に支援するだけである。その点ではグループウエアの立場に近く、ASIC設計を対象にしたグループウエアということもできる。システムの核は3.5.に述べたようにデータベースを用いた協調である。多くのグループウエアが対象とする電子会議、グループ著作支援、プレゼンテーション支援、グループ意志決定といったハイパーテキスト、ネットワーク、マルチメディア、共有ウインドウ等の技術をベースにした協調はシステムの拡張の方向として考える。

実現方法としてはOODBなどを用いることも考えられるが適用対象をASICに限定する以上まずCADフレームワークが持つデータベース機能を活用し、足りない機能を作り込むアプローチが最も少ない手間で目標を達成できる。

4.3. システムへの要求条件

協調設計支援システムの実現のための要求条件は以下のようにまとめられる。

1. まず設計プロダクト、プロセスの計算機上での定義とそれによる共通化。
2. それを達成するため全ての設計プロダクトの電子化
3. 階層変更への対応などプロジェクト運営に必要なflexibilityの確保

さらに次のような付加機能がのぞましい。

4. 設計進捗、ゲート数などの管理 指標の簡便な採取、集計
5. プロジェクトレベルでのファイル管理(一括版数付与、バックアップ等)
6. ツールのバッチ実行
7. 計算機上でチェック可能な誤り(図面修正時のHDL未出力など)の僕滅。
8. 設計レビュー支援(必要なデータの一括出力など)

4.4. 実現のプロセス

いずれのフレームワーク上で実現するにせよおおむね次のような過程を踏む。

1. 設計に使用するデータ、データに付与すべき属性やデータ間の関係、ツールおよびツールを組み合わせて作るプロセスを洗いだし(図5)、それらをデータベース化するためのスキーマを作成する。プロジェクト途中でのデータの追加、削除によるスキーマ変更は当然あるものとして対処を考えておく必要がある。

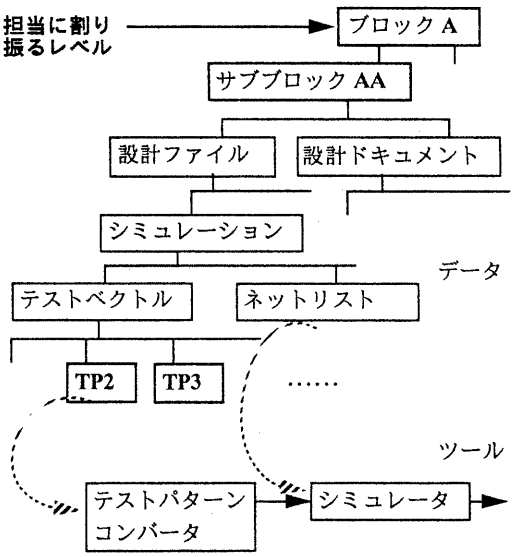
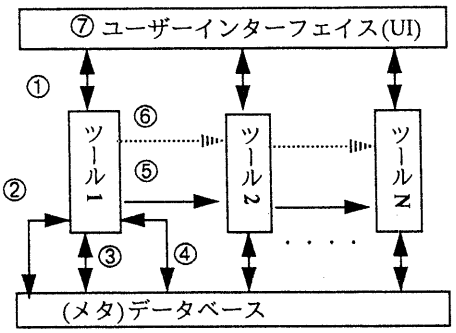


図5 設計データとツールのモデリングの例

2. そのスキーマに基づき設計階層の定義/既存データのフレームワークへのインポートを行なうための機構を実現する。
3. 必要なツール、プロセスを統合する。統合する範囲としてはデータ(図6の②③④⑥)、制御(①⑤)UI(⑦)があげられる。[13]このうちどこまで統合するかはフレームワークの機能、ユーザーに提供するサービス、統合の手間を考えて決める必要がある。ツールの使用するデータベースそのものも統合する深い結合(③)よりまずそれを除く浅い結合(①②④)を考えた方が統合の手間、ツール実行時のオーバーヘッドなどで多くの場合有利である。また対話型ツールやバッチ型でも使用方法の複雑なものは⑦の完全な統一は不可能であることを前提に進める必要がある。



①:ツールの起動

②:ツール起動時の前処理

③:ツール実行中のデータアクセス

④:ツール終了後の後処理

⑤:ツール間の制御情報の受渡し(終了通知等)

⑥:ツール間のデータの受渡し(RPC、byte stream等による)

図6 ツール統合のイメージ

4. 4.3.で述べた付加機能のうちで優先度、実現容易性を検討し、実現する。

5. PowerFrame上での支援システムの実装

5.1. なぜPowerFrameを選んだか

本システムのベースとしてなぜPowerFrame[14]を選んだかを以下にまとめる。

- 4.4.で推奨した浅い統合のみサポート。
- ツール独立な階層型データベース。階層設計をモデル化するのに自然な構成である。
- ツール統合化にC言語と専用の関数群(ツールキット)を用いるため自由度が高い。
- サーバー/クライアント形式で動作するので共有データベースによる管理に向いている。

反面次のような問題点がある。

- 現バージョンはUIの完成度やDBアクセスの速度などに問題がある。
- 階層型データベースはRDBのような数学的モデルに基づいていない。さらにツールキットの機能が十分整理されていないこともあり、"きたない、力ずくの"統合に陥りやすい。
- PFで記述できる設計プロセスは粒度の細かい、ツール間のデータ/制御の受渡しを記述したものに過ぎない。ループやオフラインの手続きを含むプロセスや粒度の粗いプロセスなどプロセス記述のための種々の要求には対応できない。
- PowerFrameの設計データベースとしての機能はデータのメタ情報(存在するディレクトリーやデータの属性、データ間の関係)に限定されている。それ以外の情報、例えば設計途中で作成される文書そのものを保管するデータベースは別途考える必要がある。

5.2. 統合したツール群

これまでに図7に記したツールを統合した。設計担当者はフレームワーク上でまず設計階層および使用するデータ(ライブラリー/既設計の流用データ)をバッチもしくは対話的に定義し(i)、設計を開始する。(ii)、(iii)、(iv)はほぼ並行に進める。あるブロックの設計エントリーが終った段階で機能シミュレーション、論理合成、合成後の論理シミュレーション(タイミング検証)を行なう。階層変更を伴う修正の際は(i)に戻る。

これらの設計ツール実行のログやネットリストから(viii)のプログラムを経由して抽出された情報や設計文書から抽出された情報は(ix)の表計算プログラムに読み込まれ、担当者/管理者による管理に役立てられる。

5.3. 統合により実現した機能

これまでに4.4.の3.にあげた①~⑦の統合要素のうちで①②④については統合を実現した。⑦についてはツールの起動時のデータ/オプションをボタンで設定できることを主眼とした。"複雑な"ツール(図7の(v),(vii))は使い方が極めて多様なため、所内での標準的な使い方をサポートし、それが最小の設定でできることを目指した。⑤(ツールのプロセスへの統合)はパターンの入力(iii)→コンバート(vi)→シミュレーション(vii)→結果の確認(iii)というはっきりとしたフローが存在するものに限定した。これに(ii)や(v)も加えてプロセスにしようとするとループやツールの実行順序の変更など複雑な制御も必要となるため今回は見送った。③⑥についてははじめから対象外である。

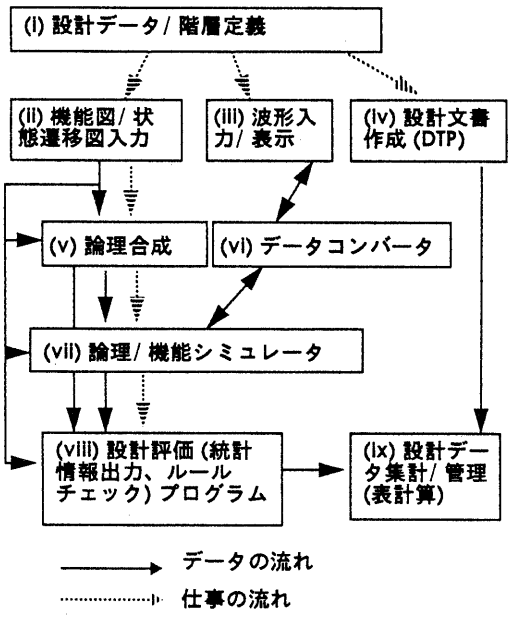


図 7 協調設計支援システム (第一版)

4.3.の要求条件の1.~4.については第一版で機能を提供した。1.についてはプロセスの定義手段は見送り、データ(プロダクト)とその階層関係の定義機構に限定して実現している。また、3.の設計階層変更は原則的に(i)のフェーズに戻って行なう必要があり、手間がかかる。この2つの問題は現行のPowerFrameを使用する限りは完全に解決するのは困難である。

5.~8.については技術的にはフレームワークのDB情報の読みだし/更新を行なうアプリケーションを作成することにより実現可能であることを確認した。ただし設計者にどのようなサービス仕様で提供するかはより検討が必要なため第一版では機能としては外してある。

6. 結び

以上協調設計のモデルとそれを支援するためにフレームワーク上に構築したシステムについて述べた。システムの実装を行なった結果、フレームワーク上にツールを統合し、さらに付加機能を実装することにより協調設計支援システムへの要求条件のかなりの部分を達成できることが確認できた。

今後、このシステムを設計者に試用してもらい評価、改良を進める。また、設計モデル自体の精密化や4.2.のなかで述べたグループウエアとしての機能強化についても検討したい。

参考文献

[1]細田他"大規模 ASIC 設計の一手法""、 1991 年電子情報通信学会全国大会 C-588.

[2]市森他"STM-0/1 インターフェイス LSI 構成法の検討"、 1991 年電子情報通信学会全国大会 B-769.

[3]"EDA フレームワーク - 設計チーム全体の作業を支援"、日経エレクトロニクス 1992.9.14(no. 563).

[4]T.J.Barnes et. al."Electronic CAD Frameworks",Kluwer Academic Pulishers(1992).

[5]松下温 (編著)"図解 グループウエア入門"、オーム社 (1991).

[6]D.Marca and G Bock(ed.)"Groupware: Software for Computer-Supported Cooperative Work",IEEE Computer Press(1992).

[7]小宮誠一 "ソフトウエア分散協調開発の為の協調型設計過程支援アルゴリズム" 情報処理学会グループウエア研究会資料 92-GW-2(Sept. 1992).

[8]C.Hübel, W.Käfer and B.Sutter"Controlling Cooperation Through Design - Object Specification - a Database-oriented Approach", Proc. EDAC'92.

[9]E. H. Durfee "The Distributed Artificial Intelligence Melting Pot" IEEE Trans. SMC Vol.21, No6.(Nov/Dec 1991).

[10]S.Sarkar and V.Venugopal "A Language-Based Approach to Building CSCW Systems", Proc. 24-th Hawaii Int. Conf. on System Sciences(1991).

[11]M.Dowson "Integrated Project Support with IStar", IEEE Software (Nov. 1987).

[12]B.W.Boehm "A Spiral Model of Software Development and Enhancement", Tutorial on:Software Engineering Project Management, IEEE(1988).

[13] M.Chen and R.J.Norman "A Framework for Integrated CASE", IEEE Software(Mar. 1992).

[14]''Design-Data-Management Frameworks help solve complex product design-team problems'',Electronic Design, Vol. 39, No.11(Jun. 1991).