4S-2

スコープルールの破壊とデータ依存関係の抽出に特徴をもつ自動方式・機能設計システム：A²DL－DA

高橋隆一　村岡泰釈　林 孝雄
日本電気（株）　複合交換開発本部

当社C&Cシステム研究所で開発した（マイクロ）アーキテクチャ評価システム(1)を母体に、自動方式・機能設計システムの開発を行ってきたので報告する。

1．概要

A²DL－DA（ｴｲｽｸｴｱ･ﾃﾞｨｰｴﾙ･ﾃﾞｨｰｴｲ）は、①核言語A²DL（ｴｲｽｸｴｱ･ﾃﾞｨｰｴﾙ）におけるスコープルールの破壊と②そのモデル：オートマトンによって陽に表現された並列性からデータ依存関係のみを抽出することに特徴を有する自動設計システムである。A²DLによって表現された仕様はＡＤＣＤＳと名付けた内部表現に翻訳され、複数の応用プログラムに共有される。

シミュレーションには、同期信号の変化点で記憶素子の値を入力に遡って評価する「要求駆動型ＲＴ（ﾚｼﾞｽﾀ･ﾄﾗﾝｽﾌｧ）レベルシミュレーション法」を用い、機能仕様の自動合成にあたっては内部表現：ＡＤＣＤＳから、さらにデータフローグラフを生成し、局所的なデータ依存解析による仕様の圧縮や資源の削減と、大域的なデータ依存関係の解析による、システムレベルの、自動パイプライン化を試みる。

2．新言語A²DL

A²DLは６５個の予約語と７２個の非終端記号を用いたＬＡＬＲ（１）で定義された応用向き言語である。記述全体はオートマトンとしてモデル化されたモジュールの階層をなしており、モジュール単位に、①主たる同期信号と②添え字が昇順（第０ビットがＭＳＢ）であるか降順（この逆）であるかの流儀を指定できる。

モジュール間通信の表現には、他のモジュールの制御／データレジスタに直接アクセスする記法があれば、十分である(3)。A²DLは、階層のどのロケーションのファシリティ（レジスタや組合せ論理回路など）にアクセスするかを宣言するための記法

ｕｓｅ ﾌｧｼﾘﾃｨ型：ﾛｹｰｼｮﾝ／ＩＤ［添え字］；

を有しており、接続関係を用いない柔軟な表現を行うことができる。「スコープルールの破壊」とはこの意味である。これによって引き起こされる可能性のある同時動作は

ｅｓｃａｐｅ　：　条件　；

の形の、アクセスされる側のモジュールにおける、「このような条件が成立しないときの動作を表現している」という宣言で回避できる。リセット条件や割り込み条件をｅｓｃａｐｅすることが考えられる。外部からの強制的な、リセット／割り込みシーケンスへの状態遷移やレジスタの初期化なども自由に表現できる。

モジュール間の通信は、これ以外にも、「論理端子（input, output, inout）」とその接続関係（ネット）

```
register : ITLC1, ITLC2;   " ITLC1 for PC, ITLC2 for RAM "
module M1 : : :
  output : MAR[7:8];
  use register : ../PC[7:8];
  use memory : ../RAM[11:12];
  use register : ../PFQ0[11:12], ../PFQ1[11:12],
                 ../PFQ2[11:12], ../PFQ3[11:12];
  use register : ../ITLC1, ../ITLC2;
  use logic : ../ACK_M1;
  register : IR[11:12];
  virtual : OP[2:3]=IR[11-9], ADR[7:8]=IR[7-0];
  register : PFQWPT[1:2]; " PreFetch Queue Write PoinTer "
  logic : NJMP=OP ne b(101),  " Not JMP "
          WPT0=NJMP * (PFQWPT eq b(00)),
          WPT1=NJMP * (PFQWPT eq b(01)),
          WPT2=NJMP * (PFQWPT eq b(10)),
          WPT3=NJMP * (PFQWPT eq b(11)),
          IRW[11:12]=b(1) comb IR[11:9] comb IR[7:0];
  operator : ADD1<AG1[7:8],AG2[7:8]>[7:8]
             < ADD1 = (AG1 add AG2) size0 8 >;
  use input : ADD2/PTR4[1:2];
  use output : ADD2/NEXT[1:2];
  module ADD2 : : :   " increment WPTR modulo 4 "
    input : PTR2[1:2];
    output : NEXT[1:2];
    PTR4 eq b(00) -> NEXT=b(01);
    PTR4 eq b(01) -> NEXT=b(10);
    PTR4 eq b(10) -> NEXT=b(11);
    PTR4 eq b(11) -> NEXT=b(00);
  ; " end of 2 bit adder "
  P0 : wait(ACK_M1) IR:=RAM, MAR=PC,
       PC:=ADD1<PC,h(01)>, state:=P1;
  P1 : OP eq b(010) -> ITLC1:=0, ITLC2:=0, state:=P2!
       OP eq b(011) -> ITLC2:=0, state:=P3!
       (OP ne b(010)) * (OP ne b(011)) -> state:=P0!
       NJMP' -> PC:=ADR!        " Jump instruction "
       WPT0  -> PFQ0:=IRW!
       WPT1  -> PFQ1:=IRW!
```

ｴｲｽｸｴｱ･ﾃﾞｨｰｴﾙ

図１．A²DLの記述例

によって表現することもできる。

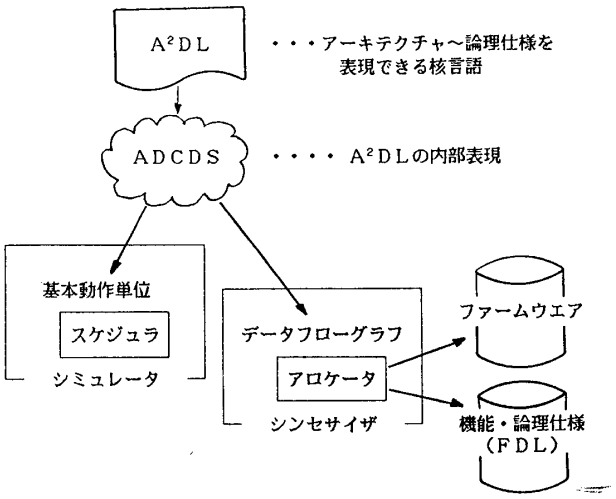
動作の表現は、条件毎の箇条書（CDL-likeな記法）

条件　－>　動作

を基本としており、記憶素子や入出力双方向論理端子を対象に、同期／非同期の動作を列挙できる。ひとつのモジュールをこの記法のみで表現することもできる。複数の状態毎に箇条書した場合（DDL-likeな記法）は、この状態にあり且つ条件が成立している場合の動作を表現することになる。A²DLではこれらの記法をひとつのモジュールに混在させることもできる。互いに関連する状態を「セクタ」としてまとめることもできる。遅延時間は組合せ論理回路単位に指定できる。

A²DLは単一の言語でありながら、テクノロジー独立な範囲のアーキテクチャ～論理仕様までの広汎な表現能力を有する。図１に記述例を示す。

Automatic Digital System Design System: A squared DL-DA breaks prior scoop rule & extracts data dependencies from Automaton
Ryuichi TAKAHASHI, Yasutoki MURAOKA, Takao HAYASHI　Abiko Plant, NEC Corporation



ｴｲｽｸｴｱ･ﾃﾞｨｰｴﾙ･ﾃﾞｨｰｴｲ

図２．A²DL－DAの
システム構成

３．ＡＤＣＤＳ

A²DLによって表現されたアーキテクチャ～論理仕様はＡＤＣＤＳ　(A squared DL-DA Core Data Structure) と名付けた内部表現に翻訳され 、シミュレーション、テストパターン生成、アーキテクチャ分割、機能仕様の自動合成などの、応用プログラム　(Application Program) に利用される。図２にA²DL－DAのシステム構成を示す。

ＡＤＣＤＳは「応用プログラム独立」であることを特徴としている。

母体としたシステムではシミュレーションにあたり、仕様をＣ言語の関数に分解していたため、Ｃコンパイルが大きな負荷となり、他方合成には別途仕様を読み込む必要があった。ＡＤＣＤＳの設計は高速に生成可能且つ修正が容易であり、しかもすべての応用プログラムが共通に利用可能な内部形式を目標とした。生成と修正の容易さは、母体としたシステムの合成プログラムと同様に、内部表現を構造体で行えば達成できると考えた。しかし新しい応用プログラムはいつでも考え出され得る。そこでＡＤＣＤＳには、単に仕様を反映するのみのデータ構造を与え、応用プログラムのための機構はいっさい置かないこととした。これが「応用プログラム独立」という考え方である。ＡＤＣＤＳは、新たな応用プログラムがいくら追加されても、影響をうけない。各応用プログラムはＡＤＣＤＳから、自身に必要な枠を適宜生成して動作する。

４．シミュレーション手法

論理（ゲート）レベルのシミュレーションでは通常「イベント駆動・タイムマッピング」方式によって、イベントすなわち各論理素子の入出力信号変化を追跡する。しかし「オートマトン」をモデルとしている場合は各モジュールがどの状態にあるかは常に分かっている。そこであえて面倒なイベント追跡をせずに、現在どの状態にあり、どのレジスタの内容を更新するのかに注目し、新しい値は何になるかを、それが必要になった時点(on demand)で逆向きに遡って評価するというアイデアが生まれる[1]。これが母体としたアーキテクチャ評価システムで開発し、培ってきた「要求駆動型ＲＴレベルシミュレーション法」である。

シミュレータは仕様の内部表現：ＡＤＣＤＳにアクセスして、各記憶素子がどのような信号に同期しているかを調べ、これらがいつ変化するかを決定するための枠をつくる。この手法により、条件が不成立になった時点で一度評価するというだけで、いわゆる「スルーラッチ」も容易に実現できる。

５．機能仕様の自動合成

さて、ＡＤＣＤＳは、オートマトンの各状態に列挙することで設計者が表現していた並列性を反映している。この並列性は、設計者が苦心して定めたものであり、機能仕様合成にとっても重要な仕様の一部であることが多い。この点で、母体としたシステムを含む、「評価」を目的としたツールに多く見られるペトリネットなどの特殊なモデルも実用には必ずしも適さない。他方、仕様が「オートマトン」によってモデル化されている場合も、データ依存関係のみに注目してこれを抽出することにより、設計空間の探索に十分な自由度は得られる[3]。オートマトンをモデルとしたのはこのためである。ちなみに仕様の表現に汎用言語を用いることは、合成のための処理系作りが極端に困難だという以前に、そもそもこのような記法は設計者にはなじまないと考え、不適切だと判断した。

シンセサイザ（機能仕様自動合成プログラム）は、ＡＤＣＤＳからデータ依存関係を抽出して、いわゆるデータフローグラフを生成することができる。記述の圧縮による高速化や資源の削減のためには、局所的な解析で十分だが、本システムでは将来、システムレベルの自動パイプライン化戦略[2]を遂行するために、大域的なデータ依存関係の解析も行うよう配慮している。この戦略ではハザード回避のためのインタロック機構も自動挿入される。

合成した機能仕様は、当社での使用実績を考慮し、母体としたシステムの合成ツールと同様、ハードウエア記述言語ＦＤＬで表現するものとした。

６．今後の課題

自動方式・機能設計システム：A²DL－DAはＯＳ：ＵＮＩＸ上で開発しており、この上で稼働する。

性能の評価、テクノロジーマッピングツールとのリンクなどが今後の課題として残されている。

文献

（１）Takahashi,R., Yoshimura,T., Goto,S.：
"A VLSI Architecture Evaluation System"
Proc.ICCD'86 pp.60-63 （１９８６）

（２）高橋：「データ依存関係に注目した自動パイプライン化」Ｓ６３年信学全大 C-250 p.2-211
（１９８８）

（３）今井、杉山編：超ＬＳＩ設計シリコンコンパイレーション、サイエンスフォーラム社
（１９８８）