Vol. 33 No. 11　　情　報　処　理　　Nov. 1992

解　説

ハードウェア記述言語

2. 主要なハードウェア記述言語の特徴と標準化状況

2.1 UDL/I†

星野民夫††　唐津　修††

1. はじめに

ASICが発展する中で，設計言語を標準化して，自由な設計データの流通，CADの自由な組み合わせ使用を実現することは，今やメーカ・ユーザを問わずLSI産業に係わるものに共通の課題となってきている．

設計言語の標準化活動には，日本を中心として標準化作業が進められている設計言語UDL/I (Unified Design Language for Integrated Circuit)[6),4)]，米国DOD（国防総省）の開発したVHDL (VHSIC Hardware Description Language)[2)]，有力市販シミュレータの入力言語であるVerilog HDL[1)]を基にしたOVI (Open Verilog International)を中心とした動き，米国民間委員会がまとめたEDIF (Electronic Design Interchange Format)，などがある．

UDL/Iは，1987年以来日本電子工業振興協会の中に設置されたLSI設計用記述言語標準化委員会（委員長：相磯秀夫慶應義塾大学教授）に，内外30以上の大学・会社・関係機関からの委員の参加を得て検討を進めてきた論理合成向きハードウェア設計記述言語である．

当初3年間の第1期計画により UDL/I 第1版の言語仕様が固まり，その仕様をもとにコンパイラおよびシミュレータを参加メンバの負担金でプログラム開発を行った．開発ソフト会社は国際入札で選定した．1991年3月には，UDL/I の中間フォーマットを確定して公表し，1992年6月にコンパイラ，シミュレータなどが完成した．

引き続き論理合成処理系の開発と言語の拡張作業を進めている．言語の拡張計画については，第1版で盛り込みきれなかった機能の取り込み，より使いやすい言語をめざして検討を進めている．

本稿では，**2.** において UDL/I の設計の基本思想を，**3.** で特徴を述べる．さらに処理系の開発状況と標準化への取り組みについて **4.**，**5.** で述べる．

2. 言語設計の基本思想

標準化にあたり，以下に述べるような解決すべきさまざまな問題があった．これらの問題の検討結果についても手短かに述べる．詳しくは，日本電子振興協会からの報告書を参照されたい[5)]．

1)　記述の対象領域をどこに置くか：ASICの設計プロセスは，(1)企画，(2)基本設計，(3)機能設計，(4)論理設計，(5)テスト設計，(6)回路設計，(7)レイアウト設計，に大別できる．設計言語の種類は，設計プロセスや，使用するCADプログラムに応じて多岐にわたる．

言語の標準化を進めるにあたり，まずどの範囲を記述対象にするか議論した．その結果，標準化言語の対象として，主にRTL (Register Transfer Level)と構造記述（ネットリスト）に焦点を絞ることとし，サポート対象を機能設計，論理設計，テスト設計の設計プロセスとする言語として設計した．

2)　設計文化の多様性：ハードウェア設計においては，組織ごとに独自の設計手法，ツール，慣習，設計ルールなどがあり，いわゆる設計文化が形成されている．標準言語を検討するにあたり，この設計文化／スタイルの違いが問題になる．たとえば，順序回路の設計スタイルには a)単相同期式順序回路のみ許す，b)多相同期式順序回路まで許す，c)Dラッチなど非同期回路などまで含めて許す，など幅広いスタイルがあり，設計言語としてどの範囲をサポートするか議論にな

† UDL/I by Tamio HOSHINO and Osamu KARATSU (NTT LSI Laboratories).
†† NTT LSI 研究所

った．

結論として，『標準言語としてできるかぎり固有の設計文化にとらわれない記述形式とする』との方針で検討を進めることとした．その結果，たとえば順序回路の記述能力としては，単相同期回路から非同期回路まで幅広い記述範囲をサポートすることとした．そのため，特定の設計文化のユーザからみると，許されない設計様式も記述できることになり，設計ルールチェックの導入など運用上の工夫を必要とする．

3) 意味論の必要性：従来多くの言語が，1言語1シミュレータであったのに対し，標準言語では複数のシミュレータが作られる．このため，シミュレータの細かな動作までを言語の意味論として設計する必要がある．さらには，論理合成系を始め，シミュレータ以外のツールも言語の厳密な意味論を必要とするが，ハードウェア記述言語の意味論の研究は十分に行われていない．

そこで，あらたに意味定義方法の検討を言語設計と並行して進め，言語仕様書に反映することとした．

UDL/I の設計方針は，1)論理合成時代の言語としての特長をもつ，2)言語の意味論を明確化する，3)簡潔で実用的な言語とする，の3点に集約される．

1) 論理合成向き言語：論理合成時代の言語とするため，論理合成系を意識した構文を採用している．加算におけるキャリ入力，リダクション演算子，Don't care の表現など，設計者が合成系への入力を記述しやすくする構文を用意している．また，同期回路のクロック系の設計などにおいて重要となるクロックの極性などが陽に指定でき，設計者および論理合成系の作成者ともに重要なタイミングに関する情報を制御できるようになる．さらに，バスなどの接続を意識したオフ状態値の指定により，ワイヤード論理も陽に表すことができる．順序回路の合成については，有限状態機械をオートマトン記述によって書き表すことができ，ハードウェアタスクの概念によってパイプライン処理のような複雑な制御も簡潔に記述できる．

2) 意味論：UDL/I では，言語の意味論を明確化するために，新しい手法を導入した．言語の単純で本質的な部分だけを核言語として定義し，言語の各文に対して，核言語への変換規則を明示することで，言語の意味を明確化している．これは，ハードウェア設計言語の意味論に対する新しい試みであり，言語の意味論をシミュレータと分離したことにも意義がある．UDL/I の意味論の詳細は **3.** に述べる．

3) 実用的言語：UDL/I では，実用的な言語とするために，言語の全体構造を簡潔にし，しかも種々の設計文化に適応できるような文法を用意している．階層記述，遅延の記述，基本素子記述，制約記述など階層設計や細かなタイミングまでも考慮した設計法，ライブラリの構築，検証などに利用できる構文要素を使うことができる．

3. UDL/I 言語の特徴

3.1 言語文法からみた特徴

並列動作を表現する言語

ハードウェア記述言語とソフトウェア・プログラミング言語の大きな違いはコンパイル後のオブジェクトの表現するモデルが，基本的にはすべて並列動作を行うハードウェアの構成要素を表すのか，ノイマン型計算機上で実行されるシーケンシャルなプログラムを表現するかである．

UDL/I では，ハードウェアの動作を忠実に表現するために，すべての実行可能な文は並列に評価・実行される．この点がソフトウェア・プログラミング言語と大きくことなる点である．

論理合成向き言語

ハードウェア記述言語でも，機能シミュレータの入力言語として開発された言語においては，そのまま論理合成に用いるとさまざまな問題点が生じる場合がある．これは，シミュレータのモデルを効率良く記述することだけを考慮して言語設計されているため，ハードウェアとソフトウェアのモデルのギャップがあることに注意が払われていないためである．

UDL/I では，論理合成のための枠組として，ハードウェアモデルとシミュレータモデルの対応関係に注意深く設計されているため，モデル間のギャップが少ない．これは，ハードウェア構成の要素である，レジスタ，配線，組合せ回路，クロックなどに対応した記述の枠組が用意されているからである．以下に幾つかの代表的例を示す．

合成に柔軟に対応できるクロックの表現

順序回路の設計ではクロックをどう設計するか

が重要な問題である．レジスタは大別してクロックのエッジで動作するタイプとレベルで動作するものがある．またクロックの極性（立ち上がりエッジ／立ち下がりエッジ，ハイ・イネーブル／ロー・イネーブル）などをうまく使い分けることが高性能な設計には欠かせない．UDL/I ではクロック式として，rise(), fall(), high(), low() などをもっており，これらを陽に指定できる．

UDL/I では，非同期リセット／プリセットをもったレジスタの表現ももっている．

これらの記述法の良さが順序回路を論理合成するとき利点として現れる．

演　算　子

UDL/I ではハードウェア表現のためのデータタイプと演算子をもっている．たとえば加算演算は3入力関数で表現され，データ入力2，キャリ入力1をもっている．そのためキャリの概念を自然に陽に扱うことができる．また APL などにもみられるリダクション演算子をもっており，Exclusive or reduction などの演算子でパリティの発生，検証などが容易に表現できる．

真理値表などはハードウェア仕様書などによくみられる．この中で Don't care のビットはハイフン（－）などで表現され表の簡素化に役立っている．これに対応する信号表現としてクエスチョンマーク（？）の文字を UDL/I は割り当てている．

OFFSTATE

双方向のバスをドライブするときなどに有効になるOFF状態の信号値を決定できるOFFSTATEの概念がある．これはターミナル（ドライバ，双方向信号線バスなど）を定義するときにあわせて属性として定義する．このバスをだれもドライブしていないとき（代入の条件がどれも真にならないとき）の信号値を指定できる．これを基にドライバの生成時に tri-state 系のドライバを生成するか，wired or タイプのドライバを生成するかの制御に使える．たとえば，Aがターミナルとして宣言されていて，CASE 式の OFFSTATE が0の場合ワイアード OR ゲートを含む回路を意味する．

3.2 明確な意味づけの方法を導入

2. で指摘したように，ハードウェア設計用標準言語では，言語の厳密な意味が公開される必要がある．これは，言語のユーザやシミュレータ設計者に言語の細かな意味に関する共通の理解を与えるためである．さらに，論理合成系など他の CAD ツール設計者も，ツールの開発時に言語の意味論を必要とする．従来の多くの言語のように，言語の意味がシミュレータによって陰に与えられる状況は許されない．言語の形式的な意味論を明確にすることは，言語設計者自身にとっても，言語とマニュアルの無矛盾性や完全性の確認の上で重要となる．

しかし，これまでに，ハードウェア設計言語の意味論の研究はあまり行われてはいなかった．このため，UDL/I の設計においては，「どのような意味を与えるか」という問題と，「どのようにしてその意味を与えるか」という問題に議論が集中した．UDL/I においては，シミュレーションと意味論の分離を基本方針とし，シミュレーションだけでなく，論理合成をも考えた意味論の導入を図った．単に現在のシミュレーション技術に依存して意味を決めるのではなく，一旦シミュレーションから言語の意味を切り離して，論理合成時代の言語の意味論を探る手法を採った．最終的な UDL/I の意味は，高速なシミュレータとの整合性も考慮して決定されたが，この意味論に関する議論から幾つかの新しい知見が得られた[3),7)]．

意味の与え方については，特に，意味の誤解が生じやすい動作記述セクションについて，UDL/I の本質的な記述能力を保存した核言語を定義し，UDL/I の各構文要素の意味をこの核言語によって記述する手法を採った．

核言語は，本質的に2種の基本文からなる．一つは論理関数を計算する組み合わせ回路に対応する Terminal Statement で，もう一つは記憶素子に対応する Register Statement である．UDL/I の記述は，これら2種の基本要素からなる回路網の動作を記述していると解釈される．記憶素子は，任意の数の同期型／非同期型の入力端子組をもつことができる．UDL/I の各構文要素の意味は，等価な動作を表す核言語の記述で与えられる．ユーザは，この変換規則に則って UDL/I の記述の意味を理解することができる．

UDL/I の意味定義に採用した本手法は，言語の文法と比較的独立に意味を定義でき，標準化のような種々の要求のコンセンサスを作り上げる場においては一つの有効な手法といえる．ハードウェ

ア設計言語の意味論の最も微妙な部分は，核言語のその一部に集約されており，UDL/I の文法や核言語の構造を大幅に変えることなく言語の微妙な意味を変更することができる．

このような，UDL/I の意味論は，この言語を使って回路を設計するユーザの理解を助けるとともに，シミュレータ設計者や合成系を始めとするその他の CAD ツール設計者にとっても重要な情報となる．特に，標準言語として重要である他の既存の言語とのトランスレータの開発においては，きわめて利用価値が高いと考えられる．

4. 処理系の開発状況

UDL/I 言語標準化の議論の中で，処理系をどう実現・提供していくかは大きなポイントであった．他の言語標準化，たとえば IEEE の VHDL，EIA の EDIF などでは標準化はあくまでも言語仕様書の確定までであって，処理系の開発は活動の範囲外である．このことは言語の普及の遅れ，開発された処理系の認定方法の問題など，標準化とその実用導入との間に多くの問題を引き起こす原因となっていた．これらを克服するために，UDL/I 委員会は処理系を開発してプロトタイプとして頒布する方針を決定した．

主たる目的は，1)言語の標準化後遅滞なく処理系を提供することにより現場での普及促進を計る，2)現場での試用経験を言語に反映することにより早期に言語の改良を計っていく，3)プロトタイプ標準処理系により認定の基準を提供し第3者による開発を容易にする，4)処理系を教育機関などに安価に配布することにより普及を加速する，などである．

基本的な考え方

A．処理系開発の第1期（平成2年—3年度の2年間）

本開発第1期においては，UDL/I コンパイラ，シミュレータ，c. テストベクタインタフェース，を開発した．また，UDL/I 標準処理系が既存の他言語（VHDL，Verilog など）と容易にインタフェース可能なように，中間フォーマットを公開している．

プロトタイプの機能・性能の評価のため委員会と開発機関が協力してテストデータを作成し試験を実行した．テストデータは言語仕様書から8000にのぼるチェックポイントを抽出し，これらを漏れなくカバーする2500の UDL/I 記述および，その入力信号，出力期待値を用意し，さらに，このテストデータを自動で実行評価する環境を開発した．これらは，今後標準処理系以外の処理系に対する認定試験の基準データとして利用していく考えである．

処理系と中間フォーマット・アクセス・パッケージは大学など公共研究機関，一般企業などに実費頒布する計画である．これらの入手の詳細は，日本電子工業振興協会にて知ることができる．

B．処理系開発の第2期（平成4年以降）

言語標準化の機能拡充を受けて，処理系の改良・機能拡張を進める予定である．

図-1 に詳細開発スケジュールを示す．

5. 標準化の状況

UDL/I は日本から発案して，標準化活動を行っている．しかし，LSI 設計が国際的広がりの中で発展している現状に鑑み，UDL/I においては，委員会構成メンバに海外の LSI メーカ，CAD ベンダを加えるなど，国際的に開かれた組織作りを念頭において活動を進めており，米国支部も開設した．近い将来，欧州にもその支部を開設し，標準化活動の拠点にする計画である．

現在 UDL/I の言語仕様書は日本語版と英語版が同時に管理されており，プリプロセッサ CPP の助けをかり，必要に応じてどちらの版でも印刷できる工夫がされている．英語版のマニュアルは，海外の研究者から多く引合いがあり，電子メール・ネットワーク上でそのポストスクリプト・コードが配布された．Nifty などの BBS 上にもポストされているので，興味のある読者は比較的容易に仕様書を入手できよう．

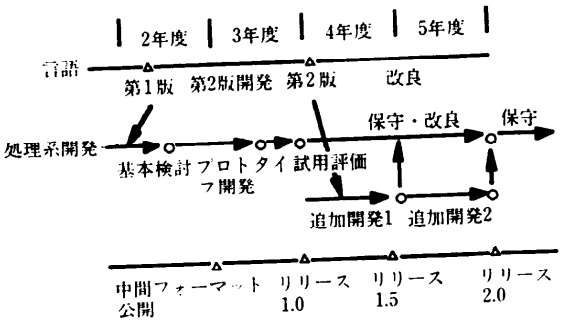

図-1 UDL/I 処理系開発スケジュール

UDL/I を JIS にする前に，ISO や IEC などの国際標準化の機構の中で標準化手続きを経たのちに JIS 化を行うことが考えられている．

最近，IEC の中に Design Automation 関連の標準化を検討する TC 93 (Technical Committee) が設置された．この中で，設計言語の標準化を検討する SC (Sub Committee) が設置された場合，ここにおいて，UDL/I を標準設計言語として提案することを検討している．

記 述 例

記述例を図-2 に示す．

大規模な回路を効率良く設計する手法として，階層設計の手法がある．UDL/I では階層設計の手法を適用して，論理システムを記述できる．階層設計の記述の単位を UDL/I ではモジュールと呼んでいる．モジュールの記述は，記述の本体を中心に，名前を与えること，管理の情報を付加すること，外部ピンを宣言すること，下位の部品を呼びだし記述すること，シミュレータでの効率の良い検証，効率的な合成などのための制約を記述すること，から成り立つ．モジュールの宣言は

name : regex ;

の文でモジュール名が "regex" であることを宣言している．

```
ident          : example;
name           : regex;
purpose        : funcsim, synthesis;
level          : block;
inputs         : data 1⟨0 : 15⟩, data 2⟨0 : 15⟩,
                 data 3⟨0 : 15⟩;
inputs         : sel⟨0 : 1⟩, oe, rst;
outputs        : out⟨0 : 15⟩;
clock          : clk;
behavior_section;
    register : reg 1⟨0 : 15⟩;
    begin
        at rise(.clk) do
            reg 1 := case .sel of
                     #2b00 .data 1
                     #2b01 .data 2
                     #2b1? .data 3
            end_case;
        end_do;
        if .rst then reset(reg 1); end_if;
        if .oe then
            .out := reg 1;
        offstate z end_if;
    end;
end_section;
end;
cend;
```

図-2 モジュールの記述例

つぎに，モジュールの利用目的を記述するための PURPOSE 文，および，モジュールの記述が階層設計のどのネスティングレベルにあたるかを指定する LEVEL 文を記述する．そのための階層上の名称をレベル名と呼ぶ．PURPOSE 文の情報を用いることで，同一のモジュール名で利用目的により記述の内容が異なる場合，利用目的にあった記述を選択することが可能になる．

モジュールが外界と信号の授受を行うためのピンを宣言する．これらは，その入出力属性などに応じて，inputs, outputs など種類ごとにその名称とビット幅を宣言する．ビット幅はたとえば，⟨0 : 15⟩ のように記述し，最上位ビットのビットアドレスが 0，最下位 (LSB : Least Significant Bit) のビットアドレスが 15 である 16 ビット幅の素子であることを宣言する．

UDL/I では，動作記述で信号を授受できる資源をファシリティと呼んでいる．あらかじめその種類とビット幅，別名などを宣言する必要がある．

register : reg 1 ⟨0 : 15⟩ ;

により，16 ビットの "reg 1" という名称のクロックに同期した動作を行うレジスタを宣言する．非同期のリセット，プリセット，代入なども RESET 文，PRESET 文，HIGH()/LOW() のレベル指定のクロック式により可能である．

このほかに TERMINAL, LATCH, ROM, RAM などの宣言文が用意されており，おのおのの資源の属性に応じて宣言する．

動作記述では，IF 文，CASE 文などプログラム言語で用いられるような制御を行う文が用意されている．ただ，プログラム言語と異なり，全ての文は条件が整えばいつでも実行できる，完全に並列な実行文である．

また，ハードウェア記述言語特有な機能として，クロックの指定を行う AT 文がある．レジ

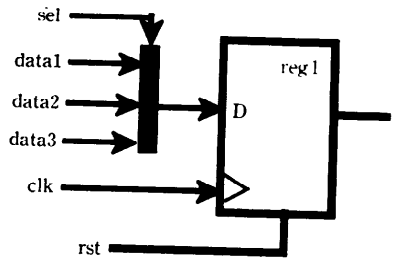

図-3 レジスタの回路例

スタにどのデータを取り込むかを指定するためには，どの条件でどのデータを取り込むかを IF 文(式)，CASE 文(式)などで指定する．たとえば，図-2 に示すように記述することにより，clk の立ち上がりエッジに同期して sel 信号によりデータを選択して取り込む回路を記述できる．

これを自動論理合成プログラムに入力すれば**図-3** に示す回路が得られる．

6. まとめ

電子協を中心にして検討が進められてきたハードウェア設計用標準言語 UDL/I について概説してきた．UDL/I が従来の記述言語に対して主張している特徴は，論理合成技術を指向した言語構成になっているという点である．それは言語文法開発上の工夫と，意味論による言語定義の一貫性の確保によって具現されている．これらの特徴により，UDL/I は今後実際の設計現場において，ハードウェア開発生産性の向上に大きな役割を果たすことが期待されている．一方 LSI 産業全体からみて見逃せないのが，設計データ権利保護における UDL/I の利用である．設計データの権利保護を図るには，2 組の設計データの異同を厳密に識別する必要がある．意味論で裏付けされた設計言語 UDL/I は，保護対象とする設計データの表現手段として最適であり，今後の権利関係の議論と実務において必要不可欠なものとなるであろう．

従来記述言語の標準化において日本から発案し貢献したケースは比較的少なかった経緯もあり，LSI の大量生産国としての日本としては，この分野の標準化に議長国など役割も引き受けながら積極的かかわりをもっていきたいという意向もある．委員会のメンバの協力を得ながら，UDL/I 標準化活動が国際貢献の一助となるべく，努力を続けていく覚悟である．

現在，UDL/I の処理系を試作して，その有効性の検証と普及に努める企画が UDL/I 開発委員会を中心に進んでいる．これらの処理系が，大学，企業などに普及することで，日本の CAD 研究の進展に寄与できよう．また，これを足掛りに CAD ベンダからの本格的なサポートと ASIC 設計現場への普及が望まれる．

最後に，LSI 設計用記述言語標準化委員会委員長相磯秀夫慶應義塾大学教授，UDL/I 標準化専門委員会委員長清水嗣雄氏をはじめとする，委員各位に感謝の意を表します．

参考文献

1) CADENCE: *VERILOG-XL Reference Manual*, CADENCE (1990).
2) IEEE: *IEEE Standard 1076-1988, VHDL Language Manual* (1988).
3) Ishiura, N., Yasuura, H. and Yajima, S.: NES: The Behavioral Model for the Formal Semantics of a Hardware Design Language UDL/I, in *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp. 8-13 (1990).
4) Karatsu, O.: VLSI Design Language Standardization Effort in Japan, in *Proc. 26th ACM/IEEE Design Automation Conference*, pp. 50-55 (1989).
5) LSI 設計用記述言語標準化委員会: LSI 設計用記述言語に関する調査研究，(社)日本電子工業振興協会 (昭和63年3月).
6) UDL/I Committee: *UDL/I Language Reference*, JEIDA (1990).
7) Yasuura, H. and Ishiura, N.: Semantics of a Hardware Design Language Japanese Standardization, in *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pp. 836-839 (1989).

(平成4年7月10日受付)

星野 民夫 (正会員)

1976 年群馬大学大学院工学研究科電子工学専攻修士課程修了．同年日本電信電話公社 (現在 NTT) に入社．以来，NTT 研究所において設計言語，設計データベース，論理合成システムなどの研究を行う．現在 NTT LSI 研究所設計システム研究部主幹研究員．IEEE，電子情報通信学会各会員．(社)日本電子工業振興協会の LSI 設計用記述言語標準化委員会委員．1985年計算機ハードウェア記述言語国際シンポジウム (CHDL 85) において論文賞受賞．

唐津 修

NTT 研究所において，LSI/VLSI のアーキテクチャ，設計手法，設計記述言語，CAD システム，CAD 用データベース，試験システムなどの研究開発を手がける．1987年以来，(社)日本電子工業振興協会の LSI 設計用記述言語標準化委員会において専門委員会委員長を勤める．東京大学工学部物理工学科卒業 (1970年)，同博士課程修了 (1975年)．現在，NTT LSI 研究所研究企画部長．IEEE，アメリカ物理学会，電子情報通信学会，応用物理学会，電気学会各会員．