機能設計支援システム “SUPERVENCH”

“SUPERVENCH” Functional Design CAD

岩脇　邦夫　K. Iwawaki
黒澤　雄一　Y. Kurosawa
西　宏晃　H. Nishi

ディジタル情報機器はその小型・多機能化に伴い，内部処理も多様化，複雑化している。また，製品仕様は標準規格に沿ったものが要求されている。このような状況に対応するためには，世界標準のハードウェア記述言語 VHDL(注1) (Very high speed IC Hardware Description Language) による設計とそれを支援する CAD が必要であると判断し “SUPERVENCH” を開発した。VHDL に不慣れでも設計できるよう機能ブロック図，真理値表，状態遷移図による設計データ入力ツールを用意した。VHDL シミュレータは高速であることを最優先に開発し業界最高速を実現した。また，VHDL と機能ブロック図の二つを入力とする論理合成も開発した。このシステムを活用することにより，LSI 設計工期が半減した。

It is becoming increasingly difficult to design digital information equipment because of the need to implement more complex and diverse functions in smaller and lighter bodies, while at the same time conforming to standards when deciding the specifications.

To solve these problems we have developed “SUPERVENCH,” an integrated design environment for digital LSIs. It supports a design methodology using VHDL, the global standard hardware description language. The features of “SUPERVENCH”, including various entry methods, the fastest VHDL simulator in the world, and a circuit-oriented logic synthesizer, make it possible to reduce LSI design times by half.

1 まえがき

ディジタル機器は，年を追うごとに小型・多機能化しており，それに伴い機器内部で行う処理の多様化，複雑化も進んでいる。また，パソコンやワープロに代表されるような情報機器は，流通アプリケーションソフトウェアの活用や公衆回線を使用しての通信を行うために，世界標準規格に沿って開発されている。これらの機器に搭載する LSI を効率よく短期間で開発するためには，CAD による設計支援が欠かせない。この CAD に要求されることは，設計者が頭の中で描く設計イメージを容易に CAD データ化できること，およびそれが仕様どおりかどうかを容易に検証できることである。また，抽象的な設計イメージを具体化するための支援ツールも必要である。

以上のような条件を満たすには，世界標準のハードウェア記述言語 VHDL による機器開発が最適であると判断した。それは，IEEE (米国電子電気技術者協会) が VHDL を規格化して以来，LSI 設計に必要な情報が VHDL で提供されるようになり，これらを活用することにより LSI 設計工期の短縮が可能だからである。そこで VHDL に対応した機能設計 CAD “SUPERVENCH” を開発した (**図 1**)。

以下，システムの設計データ入力ツール，VHDL シミュレータ，論理合成などについて紹介する。

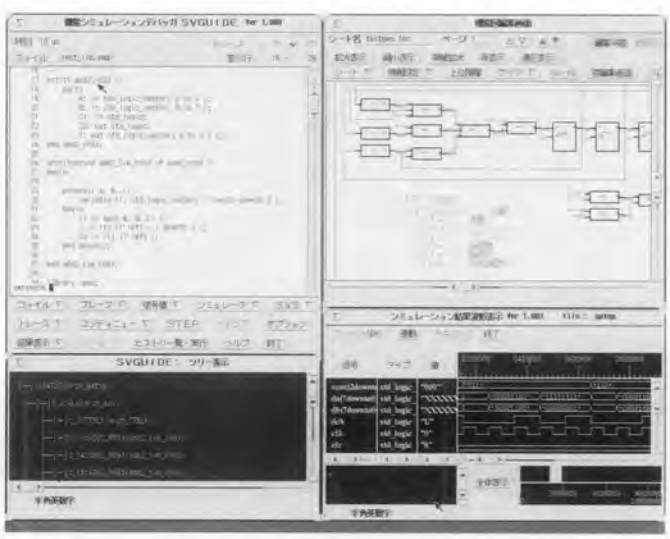

図 1．機能設計支援システム “SUPERVENCH”　VHDL 対応の機能設計 CAD。LSI 設計工期を半減できる。
Functional design CAD, “SUPERVENCH”

(注 1)　VHDL は，1983 年，米国国防省の指導のもとに言語仕様の開発が始まり，1987 年に規格化された。これにより事実上世界標準となった。

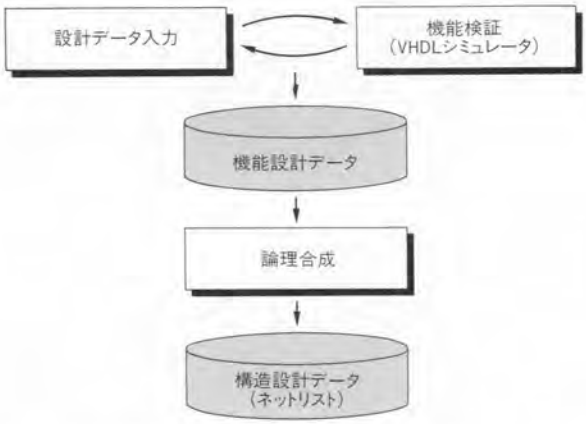

図2．“SUPERVENCH”の構成　“SUPERVENCH”は，設計データ入力，VHDLシミュレータ，論理合成から構成されている。
Configuration of “SUPERVENCH” system

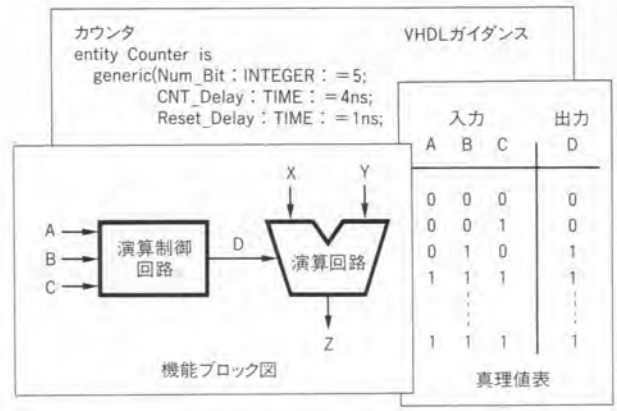

図3．設計データ入力の例　機能ブロック図，真理値表，VHDLガイダンスによる設計データの入力例
Example of schematic capture

2 全体構成

“SUPERVENCH”の全体構成を図2に示す。開発する製品の仕様に従って搭載するLSIの機能を考え，設計データ入力ツールと機能検証ツール（VHDLシミュレータ）とを使用して機能設計データを完全なものとする。次に論理合成ツールを使用し，機能が記述されたデータをLSIを製造するためのネットリストと呼ばれる構造データに変換する。

3 設計データ入力

LSIの設計は，必要とされる機能の設計から始まる。たとえば，VHDLを使用して機能を記述していく。一般的にハードウェア記述言語（以下，HDLと略記）は，抽象度の高い設計を可能にするため，大規模で複雑なLSI設計に適している。したがって情報機器の設計に向いているが，次のような短所もある。

(1) 従来，設計は論理回路図で行っていたが，HDLはテキスト言語であり，設計者にとってなじみにくい。
(2) HDLは文法が複雑であるため，使いこなすまでに数週間～数か月を要してしまう。
(3) LSI内部で処理するデータの流れかたは，回路図と比較してわかりにくい。

そこで図3に示すような設計データ入力ツールを開発した。

機能ブロック図は従来の論理回路図の延長線上にあり，図面上に配置する素子の抽象度を上げたものである。入力完了後，自動的にVHDLに変換できる。したがって，VHDLを意識せずに機能設計することが可能となり，設計者にとってなじみやすい。また，データの流れかたを容易に表現できるため，設計効率も向上する。

真理値表は，入力信号に対する出力信号の組合せを作成するものである。データの流れかたを制御する回路の設計に使用する。これも自動的にVHDLに変換できる。

VHDLガイダンスは，VHDLの文法や記述例を設計者の指示に応じて表示するものである。前述したとおり，VHDLを使用すると，機能ブロック図や真理値表以上に抽象度の高い設計が可能となるが，複雑な文法を習得しなければならない。これを軽減する目的でこのツールを作成した。

また，これら以外に状態遷移図による設計データ入力も可能である。制御回路の設計に適しており，これもVHDLへの自動変換が可能である。

4 VHDLシミュレータ

4.1 概要

設計データ入力ツールで設計したものが，仕様どおりかどうかを検証するためにVHDLシミュレータを開発した。

図4に示すように，機能設計データ（機能ブロック図，真理値表，VHDLによる機能記述など）をVHDLに変換した後のデータをシミュレートする本体と，シミュレーションの開始，中断，結果表示などの制御を行う部分から構成される。これ

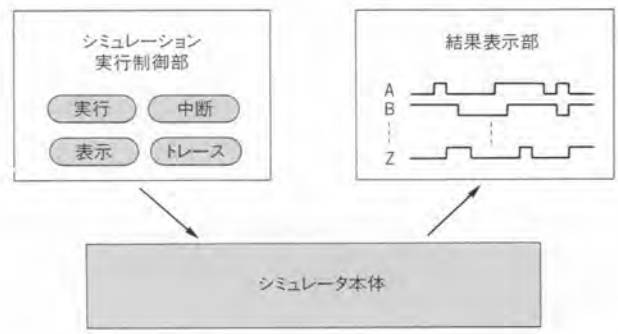

図4．シミュレーションの概念　シミュレータ本体とその制御部，結果表示部とでシミュレーションを行う。
Overview of simulator

を使用して，設計しているLSIの動作をシミュレーションし，そのときのレジスタや出力ピンの信号値を結果表示部に表示する。設計者は，その結果が仕様どおりでなければ，仕様どおりになるまで機能設計データを修正しシミュレーションするという作業を繰り返す（これをデバッグサイクルと称す）。特に，大規模化，複雑化するシステムの設計期間を短縮するためには，設計の上流工程におけるシステム全体シミュレーションによって，十分なデバッグを行っておくことが必要である。したがって，デバッグサイクルおよびシステム全体シミュレーションの時間を短縮することが製品の短期開発を実現するうえで非常に重要である。

そこで，高速であることを最優先として開発を行った。シミュレーション方式は，高速性，移植性，コンパイラの最適化能力の利用を考慮し，C言語を使用したコンパイル方式を採用した。また，シミュレーションアルゴリズムは，VHDL仕様にイベントを意識した仕様が現われているため，イベントドリブン法である。**図5**に構成を示す。

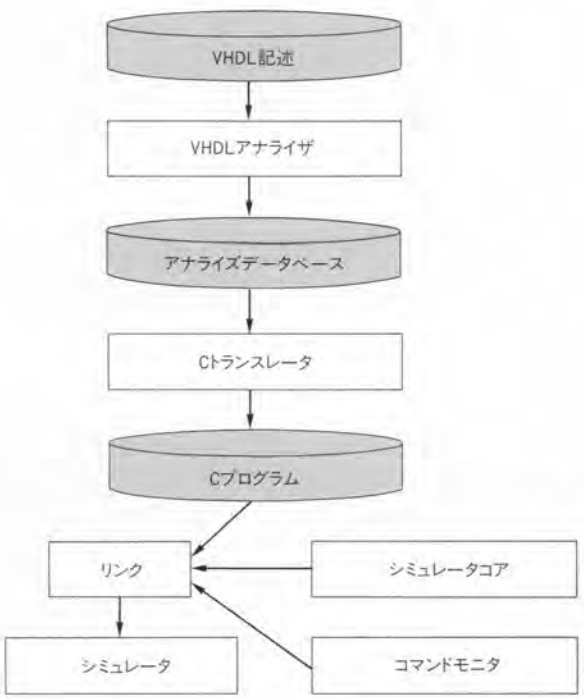

図5．シミュレータ本体の構成　VHDL記述をCプログラムに変換し，シミュレータコア，コマンドモニタと結合してシミュレータを生成する。
Block diagram of VHDL simulator

(1) VHDLアナライザ　VHDLを設計モジュール単位に構文解析し，アナライズデータベース（中間形式）への変換を行う。

(2) Cトランスレータ　アナライズデータベースからエラボレーションやプロセス実行のためのCプログラムを生成する。ここで，エラボレーションとはシミュレーションに必要な回路データ構造を作る処理である。

(3) シミュレータコア　エラボレーションにおける回路データのアロケーション，イベントドリブン処理の実行，シミュレーション時刻管理，VHDLオペレータの実行を行うライブラリ関数群である。

(4) コマンドモニタ　デバッグ機能，シミュレータ制御を行う。

(5) シミュレータ　シミュレーション対象となるすべての設計モジュールに対するCトランスレータ出力をシミュレータコア，コマンドモニタとリンクすることによって生成する。

4.2 高速化手法

特にレジスタ転送レベルの設計における使用を考慮し，以下のような高速化手法を取り入れた。

(1) 信号の特徴に基づいた信号代入処理の最適化　信号値更新段階では一般的に，①信号の現在値の更新，②信号の1時刻前の値，信号にイベントが起こった時刻などのVHDL定義済み属性情報の更新，③VHDL記述中のセンシティビティリストに従ってプロセスを実行キューに登録する処理，を行うことが必要である。しかし，②，③はそれぞれ，VHDL記述で定義済み属性を参照されない信号，センシティビティリストに現れない信号については不要な処理となる。そこで，あらかじめCトランスレータの段階においてすべての信号を，①だけが必要，①と②が必要，①と③が必要，①，②，③すべてが必要という四つのクラスに分類し，シミュレーション時には各クラスに必要最小限の処理だけを行うものとした。

(2) レゾリューションの省略　バス構造を含む回路をVHDLで記述する場合，通常はレゾリューション型の信号を使用する。このような回路を含んだモジュールのすべての信号をレゾリューション型として記述すると，代入源が1個の信号に対しても，レゾリューション関数が呼び出され，シミュレーションにおけるオーバヘッドとなる。そこで，代入源を1個しかもたないレゾリューション型信号については，レゾリューション関数がその唯一の代入源の値を返す場合に限り，レゾリューションをせずに代入を行うコードを生成して高速化を図った。

(3) システム組込みデータ型，関数の使用　4値，9値（IEEE1164パッケージ）データ型の演算と頻繁に使用される論理，加減算，シフト演算のコードはC関数をシミュレータコアのライブラリに組み込んで使用した。

4.3 性能

表1にこのシミュレータと業界最高速VHDLシミュレータによるシミュレーション実行CPU時間を示す。このシミュレータは上述の高速化手法によって，当該シミュレータの1.4～2.8倍の処理速度を達成した。

4.4 デバッグ機能

表1．シミュレーション実行 CPU 時間
Simulation CPU time

単位：s

モチーフ	VHDL 記述 行数	データ型	本シミュレータ	業界最高速シミュレータ
暗号復号回路	762	2値	45.5	63.7
実製品回路 1	15,053	4値	61.5	130.8
実製品回路 2	7,193	4値	32.5	91.2

デバッグサイクルを短くするためには，期待どおりでない結果から，その原因となる設計ミスを捜し出す時間を短縮することも重要である。そこで，次の機能を付加した。

(1) シミュレーション結果を設計者にとってわかりやすいタイミングチャート形式で表示する機能。

(2) シミュレーションを任意の時刻，任意の VHDL 行で中断し，信号値が観測できる機能。

(3) 回路が発振してしまった場合，一般的にシミュレーションは永久ループに陥ってしまうが，その箇所を自動検出したうえで中断する機能。

5 論理合成

5.1 概要

機能設計データを構造設計データに自動的に変換するための論理合成ツールとして，VHDL からの論理合成と設計データ入力ツールで作成した機能ブロック図からの論理合成との2種類を開発した（図 6）。

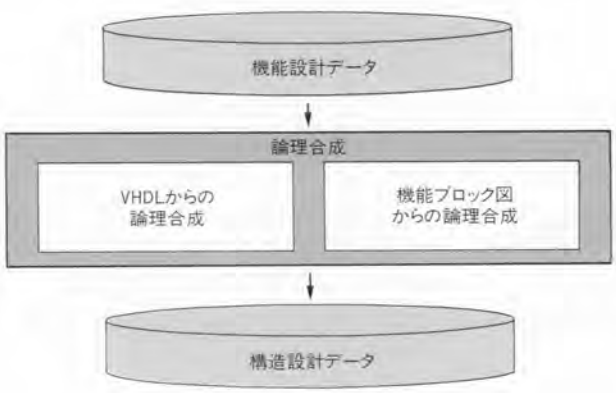

図6．論理合成の概念　VHDL と機能ブロック図のそれぞれを論理合成することができる。
Overview of logic synthesis

5.2 VHDL からの論理合成

このシステムは，VHDL による機能記述データを取り込んで，言語に依存しない初期回路に変換し，論理合成エンジン[1),(2)]を使用して最適化した論理回路を生成する。

特長として次のことが挙げられる。

(1) クロックトリガの記述から，値保持の必要な信号や変数を解析して記憶素子を自動生成する。

(2) 有限状態機械を認識し，状態を表現するのに必要なビット幅の状態コードを生成する。

(3) バスを合成する。

(4) ビットシフト演算や縮小演算などの組込み関数群を VHDL の関数として受け付けて合成する。

(5) 過去に合成した資産を VHDL のコンポーネントやサブプログラム記述を用いて再利用できる。

5.3 VHDL からの合成の処理フロー

図 7 は VHDL からの合成の構成図である。コンパイラは VHDL の構文意味解釈を行って中間形式を生成した後，これから設計データベースを生成する。設計データベースは，VHDL のシーケンシャル文やコンカレント文の“文”のデータや，ファシリティ（ポートや信号や演算子などの回路の構成要素をさす）とそれらの間の接続情報を格納している。

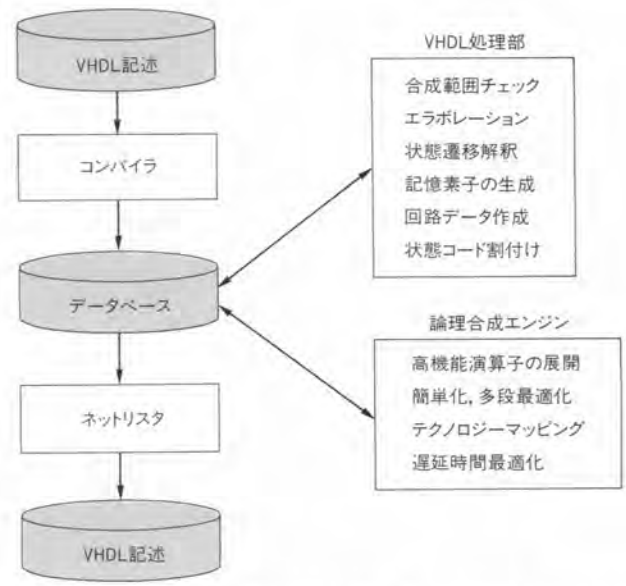

図7．VHDL からの論理合成システムの構成　VHDL コンパイラと VHDL から初期回路を合成する VHDL 処理部と論理合成エンジンから成っている。
Block diagram of logic synthesis system from VHDL

VHDL 処理部では合成できない浮動小数などのデータ型の定義やオブジェクトの宣言，剰余算などの演算子がないかどうかをまず調べる。エラボレーションで generate 文の展開や generic 値の伝搬を行う。さらに状態遷移解釈部は状態モデルの表現形式をチェックしてからプロセス文のタイプが状態遷移モデルか，状態レジスタモデルかを判別し，それ以外は通常動作モデルとして初期回路を生成する。

状態遷移の記述例を図 8 に示す。状態遷移は列挙型の現状態と次状態信号（変数）を用いて記述し，状態遷移と状態レジ

```
type ST_TYPE is (S0, S1);      ——>列挙型の定義
signal C_ST, N_ST: ST_TYPE;    ——>列挙型の現状態と次状態
                                  を表す信号の宣言
  P1: process(C_ST)
  begin
    case C_ST is
      when S0 =>               ——>現状態がS0の場合
        01<=A;
        N_ST<=S1;              ——>次状態を表す信号に状態名S1をセット
      when S1=>                ——>現状態がS1の場合
        01<=B;
        N_ST<=S0;
    end case;                        (a) 状態遷移を表すプロセス文
  end process P1;

  P2: process(CLK, C, D)
  begin
    if (C='1') then
      C_ST<= S0;               ——>非同期割込み遷移
    elsif (CLK'event and CLK='1') then——>CLKが状態遷移
      if (D='1') then                    のクロック
        C_ST<=S1;              ——>同期割込み遷移
      else
        C_ST<=N_ST;
      end if;
    end if;                          (b) 状態レジスタを記述したプロセス文
  end process P2;
```

図8．状態遷移を記述したVHDL　有限状態機械は列挙型の信号で状態を表し，状態の遷移と状態レジスタからなる二つのプロセス文で記述する。

VHDL descriptions of finite-state machine

スタをそれぞれプロセス文で記述する。状態遷移は同期／非同期割込みにも対応している。記憶素子の生成部は，プロセス文内に記述されたif文やcase文のネストを解析して条件分岐解析木（代入文を制御するすべての条件がフローで表現され，これから代入文の成立条件が求められる。これをコントロールフローグラフと称す）を作成する。同時に，クロックの記述に違反がないかどうか調べる。コントロールフローグラフの中に値の変更されない信号や変数がある場合は，それらの値を保持する記憶素子を生成する。回路データ作成部は，代入文やコンポーネントインスタンス文から回路を作成する。

5.4　VHDLからの合成用サブセット

現在サポートしているVHDLのサブセットは，データ型はbit，std_logic (IEEE1164) とそれらの配列と列挙型である。演算子は乗算や除算，剰余算，累乗以外のもので，文はloop文以外のシーケンシャル文と，コンカレント文である。さらに二重配列の信号を用いてRAMを，定数でROMのモデルを，std_logicの“Z”を用いてバスの記述をサポートしている。

5.5　性能

VHDLからの合成は，記述の演算子数にほぼ比例した時間で初期回路を合成でき，論理合成エンジンは局所最適化手法を用いているため，商用ツールよりも大規模な回路が短時間に合成できる。例えば当社のワークステーションを用いて20Kゲートの回路を1時間で合成できる。さらに質のよい回路を合成するため代数的な除算を用いた多段化[3]や，求められるすべての解から最適なものを選択する動的計画法によるテクノロジーマッピング手法を取り入れている。今後は回路の特徴に応じた合成方式や，最適化の繰り返し回数を少なくする手法を取り入れていく予定である。

5.6　機能ブロック図からの論理合成

抽象度の高い機能ブロック図から，LSI製造のための具体的な構造設計データを生成する。機能ブロック図とVHDLとを比較すると，前者のほうが構造設計データのイメージに近い。その分だけ設計者の知恵が機能ブロック図に入っているわけであり，そこから論理合成したほうが得られる結果もよいものとなる。そこで，機能ブロック図の構造を保持したまま，設計を具体化するようにした。これにより，あらかじめ大まかな構造がわかっているデータパス回路では，従来型のHDLからの合成と比較し，短時間で質のよい回路が得られ，かつ10倍以上大規模な回路が合成できるようになった。

6　あとがき

世界標準ハードウェア記述言語VHDLを使用したCAD“SUPERVENCH”を開発した。従来のCADと異なり，設計の初期段階から実機に近い形で検証が行えるようになった。これにより設計のやり直し回数が格段に減少し，設計工期を半減させることができた。今後は，マルチメディアのような情報機器を対象として，音声，文字，映像なども含めた検証が可能となるよう研究，開発を進める。

文　献

(1) 関根優年，他：機能設計CADシステム，東芝レビュー，**45**，4，pp.360-363 (1990)

(2) 西尾誠一，他：高性能LSI設計用CADシステム，東芝レビュー，**48**，7，pp.515-518 (1993)

(3) R. K. Brayton, et al: MIS: a multiple-level logic optimization system, IEEE Trans. on CAD, pp.1062-1081 (1987)

岩脇　邦夫　Kunio Iwawaki

1982年入社。機能・論理設計CADの開発設計に従事。現在，青梅工場情報処理・機器LSI技術開発部主任。
Ome Works

黒澤　雄一　Yûichi Kurosawa

1984年入社。機能・論理設計CADの研究・開発に従事。現在，研究開発センター 情報・通信システム研究所研究主務。
Communication & Information Systems Labs.

西　宏晃　Hiroaki Nishi

1984年入社。機能・論理設計CADの研究・開発に従事。現在，研究開発センター ULSI研究所研究主務。
ULSI Research Labs.