設計自動化　５０－５
（1989.12.14）

機能レベルシミュレーションの一手法

水野　雅信＊　高井　裕司＊＊　村岡　道明＊＊

松下電器産業株式会社
＊＊半導体研究センター　＊映像音響研究センター

あらまし　現在、開発を進めている機能、論理、スイッチのミックスレベルシミュレータＭｅｌｏｎ２における機能シミュレーション手法について述べる。本システムでは、入力言語として機能記述核言語ＭＨＤＬを開発した。ＭＨＤＬは、意味が明瞭で簡潔な構文と十分な記述能力を持つレジスタ転送レベルの言語である。標準化言語ＵＤＬ／ＩはＭＨＤＬに変換され、Ｍｅｌｏｎ２でシミュレーションされる。ＭＨＤＬを入力とするシミュレーション方式として、ＣＴ方式（Compiled & Table driven method）を提案する。ＣＴ方式はＭＨＤＬの機能記述単位でコンパイルを行い、表駆動方式で制御することによって、非同期回路や遅延を伴う回路を扱い、かつ従来のコンパイル方式並の高速処理を可能とする。

A FUNCTIONAL SIMULATION METHOD FOR VLSI DESIGN

Masanobu MIZUNO*, Yuji TAKAI**, Michiaki MURAOKA**

**Semiconductor Research Center　*Audio Video Research Center
Matsushita Electric Industrial Co.,Ltd.Osaka,Japan

Abstract　This paper represents a functional level simulation method used in the mixed-level simulator Melon2.　A simple hardware description language MHDL has been developed as a register transfer level core language. The language has a well-defined semantics, so that the semantics is clear for circuit desiners and the simulator.
A standard language UDL/I is translated into MHDL to be simulated by Melon2.
A new simulation technique CT(Compiled and table driven) method is proposed. The CT method, which generates compiled-codes for each statement of MHDL source and evaluates those codes by table driven, enables an asynchronous circuit and delay description to be simulated at high-speed equal to the compiled-code simulator.

1. はじめに

大規模、複雑化する集積回路を効率的に開発するために階層化設計手法が用いられている。階層化設計手法では、各モジュールの設計進度に合わせて異なった設計レベルが混在する回路モデルを検証するミックスレベルシミュレータが不可欠である。現在、機能、論理、スイッチ[1]の各設計レベルが混在した回路モデルを検証するミックスレベルシミュレータＭｅｌｏｎ２の開発を進めている。本報告では、Ｍｅｌｏｎ２における機能レベルシミュレーション手法について述べる。

機能記述された回路モデルのシミュレーションは、機能設計初期での検証とミックスレベルで行うシステムシミュレーションに用いられる。これを行なう機能シミュレータには、以下のような点が要求される。(1)標準言語にインターフェースすること。(2)入力言語は回路設計者からもシミュレータからも意味が明確に定義された言語であること。(3)詳細化された論理レベル以降のシミュレーションに比較して充分高速であること。(4)設計の詳細化に入る段階で機能記述レベルのタイミング（遅延）を考慮したシミュレーションや、ミックスレベルのシミュレーションにおいて機能記述に非同期動作の記述を含むシミュレーションが可能であること。

標準言語としては、現在、標準化が進みつつある設計記述言語ＵＤＬ／Ｉ[2]のレジスタ転送レベル言語を考えている。ＵＤＬ／Ｉは、オートマトン動作、タスク転移など制御系を簡潔に記述でき、設計者にとって記述が容易で汎用性のある言語である。本システムでは、設計者が記述するＵＤＬ／Ｉ記述を意味が明確なＭＨＤＬ[3]記述に変換する。これにより、入力記述の意味の明確化とシミュレータ実現の容易化を図っている。ＭＨＤＬは機能記述核言語として開発され、意味が明瞭で簡潔な構文と充分な記述能力を持つレジスタ転送レベルの言語である。

ＭＨＤＬ記述を入力とする機能シミュレーション方式として、新たにＣＴ方式（Compiled & Table driven method）を提案する。従来の方式としては、コンパイル方式[4]、表駆動方式[5,6,7]等がある。コンパイル方式は、対象を同期回路に限定し高速処理（論理シミュレーションに対して１桁から２桁の速度比をもつ）が可能であるが、遅延を伴う回路や非同期回路のシミュレーションが困難である。また、非同期動作のシミュレーションが可能な表駆動方式（通常の論理シミュレータに用いられる）では、論理シミュレーションに比較して充分な処理速度が得られない。ＣＴ方式は、コンパイル方式と表駆動方式を混合した方式であり、機能記述核言語の機能記述単位毎に生成した評価関数をイベントにより駆動する。これによって、遅延を伴う回路や非同期回路の扱いを可能とし、同時にコンパイル方式並の高速処理の実現を図っている。また、本方式は基本的にイベントで制御することから論理レベルやスイッチレベルのシミュレーションとも実現上整合性が取り易い方式である。

本稿では、まずシステム構成について述べ、機能記述核言語ＭＨＤＬと、ＣＴ方式およびその評価結果について述べる。

2. システムの構成

ミックスレベルシミュレータＭｅｌｏｎのシステム構成を図2.1に示す。本システムは、ＵＤＬ／Ｉトランスレータ、ＭＨＤＬコンパイラ、リンカ、シミュレーション実行部から構成される。

(1) ＵＤＬ／Ｉトランスレータ

ＵＤＬ／Ｉによる高度な制御機能の記述を構文解析し、基本的なレジスタ転送記述文から構成されたＭＨＤＬ記述に変換する。

(2) ＭＨＤＬコンパイラ

モジュール単位で処理を行い、機能記述を評価するＣ言語のソースファイルと「モジュールオブジェクト」ファイルを生成する。Ｃ言語ソースファイルは、ファシリティの状態値変化を

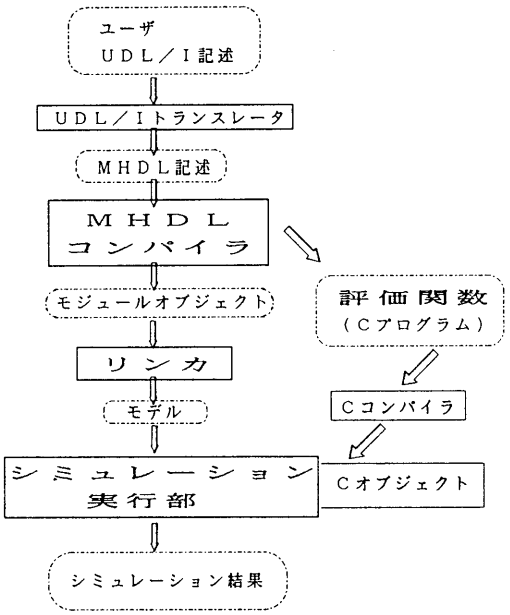

図2.1 Melonのシステム構成

求める評価関数群から成る。

「モジュールオブジェクト」ファイルの内容は、表駆動方式によるシミュレーションを行うためのネットリスト、機能記述中で使用されるファシリティに関する情報（名前、ビット幅、遅延などの情報）等である。

(3) リンカ

リンカは、複数の「モジュールオブジェクト」ファイルを入力して階層展開を行ない、対象回路全体のデータをもつ「モデル」ファイルを生成する。

(4) シミュレーション実行部

シミュレーション実行部は、表駆動方式によりミックスレベルシミュレーションを実行するコアプログラムである。おもに時間管理、イベント処理等を行なう。MHDLコンパイラが生成した評価関数は、コンパイル後、シミュレーション実行部にリンクされる。

機能レベルシミュレーションは、「モデル」ファイルのデータと上記の評価関数を用いて実行される。機能レベルでは、信号値として0、1、X（不定値）、Z（ハイインピーダンス）の4値を扱う。シミュレーションでは、ファシリティ単位（ビット幅を持つ）で信号値の参照、更新を行なう。遅延は、ファシリティへの信号転送毎に評価される。

3. 機能記述核言語MHDL

3.1 MHDL開発の目的

MHDLは基本的なレジスタ転送レベル動作に1対1に対応した単純な構文を持つ言語である。核言語としてMHDLを開発した目的を以下に述べる。

(1) 高度なUDL／I機能記述を単純なMHDL記述に翻訳することによって設計者に機能設計記述の明確な意味解釈を示す。

(2) 意味が明確な核言語によって入力言語処理系を階層化し、実現の容易化、処理系の汎用化を図る。

(3) CT方式の前処理として同期動作記述と非同期動作記述を文単位で分割する。

3.2 MHDLの言語構成

MHDLは、構文が基本的なレジスタ転送レベル動作に1対1に対応すること（一意性）、元のUDL／I記述と対応付けが容易であること（対応性）、簡潔な構文則から構成することを考慮して設計されている。

記述中に使用できるファシリティ（レジスタ、バス、RAM等を総称して呼ぶ）の種類、遅延定義等は、UDL／I、あるいはHSL－FX［8］と同等である。

MHDLの機能記述文は、機能接続記述文と転送記述文の二種類の文により構成する。MHDLの構文の構成を図3.1に示す。

機能接続記述文では、信号値を記憶できないターミナル等への信号値伝搬を記述する。機能接続記述文は、制御条件、接続関係、接続する演算式の記述から構成される。一方、転送記述文では、レジスタ、ラッチ等の信号値を記憶できるものへの信号値転送を記述する。転送記述

文は、転送タイミング、制御条件、転送関係、転送する演算式の記述から構成される。エッジトリガ転送動作（同期動作）はＡＴ文、レベルトリガ転送動作（同期・非同期混合動作）はＷＨＩＬＥ文の各文で記述する。

ＭＨＤＬは、ユーザが直接記述するＵＤＬ／Ｉ（またはＨＳＬ－ＦＸ）と異なり、オートマトン文、リセット文等の制御機能の記述文を持たない。また、レジスタ、ラッチ、ターミナルの各々への転送毎に異なる記述文を持つ。

4. ＣＴ方式

4.1 特長

(1) モジュール単位の機能記述から同期動作・非同期動作の記述毎（文単位）に、各々を評価する評価関数を生成する。分割された各機能記述の評価は、コンパイル方式と同様高速に行なわれる。

(2) 上記の各評価関数の実行制御とファシリティ状態値の更新処理を表駆動（イベントとタイムホイールを用いる）によって行なう。これにより、実遅延の評価、非同期動作の扱いを可能としている。

本方式では、イベント・表駆動による制御のオーバヘッドを削減するため以下の２点を検討した。

(1) 同期動作記述をシミュレートする評価関数の実行はクロック信号の変化時にのみ行う。従来の方法［9］と異なり、評価関数内ではなく、評価関数の駆動時の制御により実現している。

(2) 評価関数内では、ファシリティの状態値更新（レジスタ転送記述）を、その制御条件（制御条件記述）が成立したときに無条件に、ファシリティの全ビットを一括して行う。これにより、多ビットを有するファシリティに対しては、１ビット毎にイベント判定を行うオーバヘッドが軽減される。

4.2 評価関数と伝搬先リスト

シミュレーションの前処理としてＭＨＤＬコンパイラが生成する評価関数と伝搬先リスト（ネットリストに相当する）について説明する。

(1) 評価関数

評価関数は、ＭＨＤＬ記述の機能記述単位であるＡＴ文、ＷＨＩＬＥ文、接続記述文の各々に対応して生成する。評価の単位となる各文は、複数の信号転送動作の記述を含む複文であり、複文内の転送は一つの評価関数で評価する。

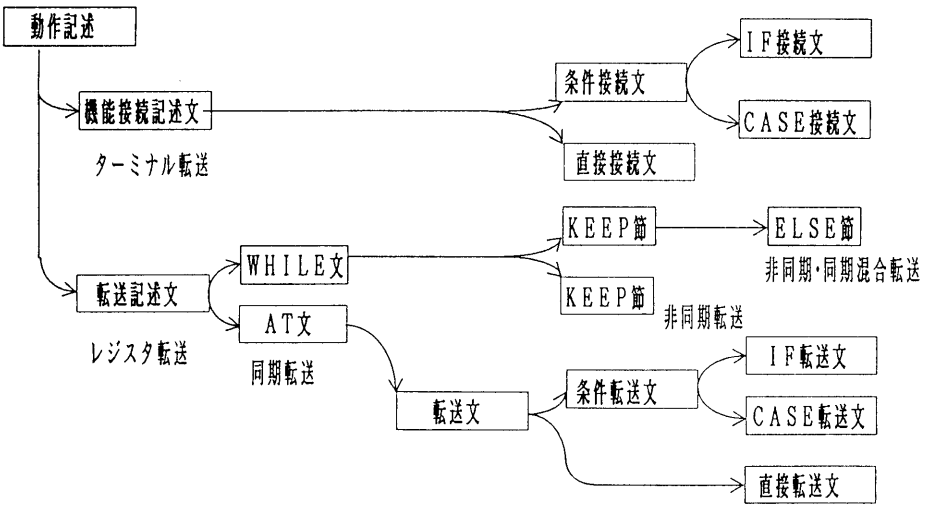

図3.1 ＭＨＤＬ構文の構成

具体的例を図4.1に示す。図4.1のＭＨＤＬ記述は、ＣＬＫ１とＣＬＫ２による２つの同期動作記述と１つの非同期動作記述から構成され、図中で同期動作１とした文は複文である。この機能記述からＭＨＤＬコンパイラは３つの評価関数プログラムを生成する。（図4.1のREGA、REGB、REGC、REGD、REGEはレジスタ、BUSはターミナル、SELは制御信号である。）

```
at (CLK1) {
  REGA<31:0> = REGC<31:0>+ REGD<31:0>;   同期動作１
  REGD<31:0> = BUS<31:0>;                （ＡＴ文）
}

at (CLK2) {
  REGB<31:0> = REGE<31:0>+ BUS<31:0>;    同期動作２
}                                        （ＡＴ文）

if (SEL) {
  BUS<31:0> = REGA<31:0>;                非同期
} else {                                   動作１
  BUS<31:0> = REGB<31:0>;                （接続文）
}
```

図4.1　機能記述と評価関数の対応

(2)　伝搬先リスト

伝搬先リストは、評価関数の実行を制御するためのデータである。ファシリティの状態値が更新された時に実行される評価関数がリストされる。

本方式では、同期動作（ＡＴ文）の評価は同期信号（クロック信号）となるファシリティにイベント（状態値の変化）が起きた時のみ評価関数を実行し、それ以外のファシリティの状態値が更新されても評価関数を実行しない（状態値を参照するのみである）。非同期動作（接続記述文・ＷＨＩＬＥ文）の評価は、参照するファシリティの状態値が更新される毎に評価関数を実行する。これにより非同期な動作のシミュレーションを行なう。以上の制御を行うために伝搬先リストには、クロック信号である場合を除き非同期動作を評価する関数だけがリストされる。図4.1の記述例に対しては、図4.2に示す伝搬先リストが生成される。同期動作１と同期動作２の評価関数は各々ＣＬＫ１とＣＬＫ２にのみリストされている。

ファシリティ名	評価関数
ＣＬＫ１	同期動作１
ＣＬＫ２	同期動作２
ＲＥＧＡ	非同期動作１
ＲＥＧＢ	非同期動作１
ＲＥＧＣ	
ＲＥＧＤ	
ＲＥＧＥ	
ＳＥＬ	非同期動作１
ＢＵＳ	

図4.2　伝搬先リスト例

4.3　シミュレーション処理手順

シミュレーション実行部は、ファシリティの状態値を一括管理するファシリティ表、伝搬先リスト、タイムホイールを用いる。各シミュレーション時刻の処理手順を以下に示す。

step1：現時刻にスケジュールされた状態値にファシリティ表の状態値を更新する。

step2：伝搬先リストから実行する評価関数を選択する。

step3：選択された評価関数を実行する。

step4：更新するファシリティ状態値をタイムホイールの遅延時間経過時にスケジュールする。

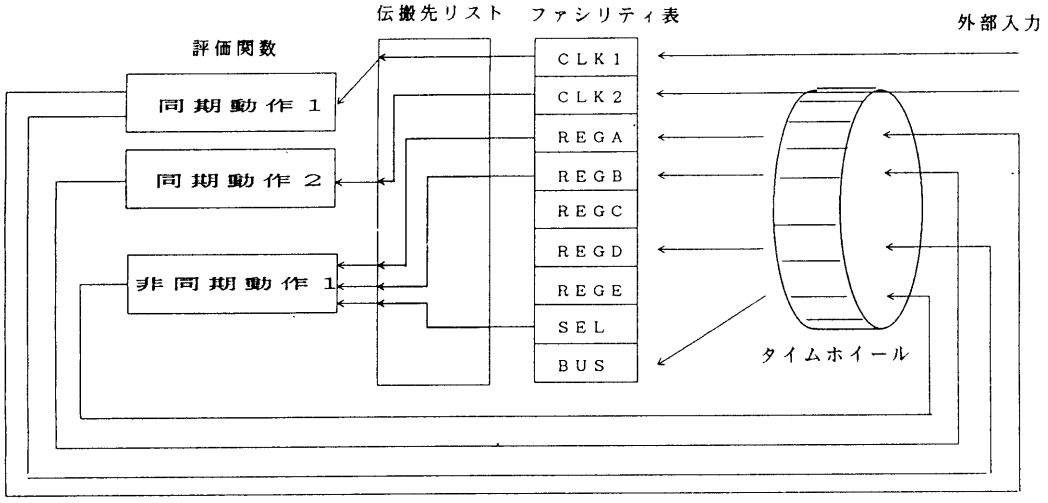

図4.3 シミュレーション処理の概略

以上のようにファシリティ状態値の更新をイベントとしてファシリティ表をから評価関数を駆動することによりシミュレーションが行われる。なお、零遅延であってもファシリティ状態値をタイムホイールにスタックすることにより順序的に矛盾なく評価を行う。

図4.1のＭＨＤＬ機能記述例に対するシミュレーション処理の概略を図4.3に示す。図では、伝搬先リストを矢印で示し、さらに各評価関数が実行された場合にタイムホイールにスケジュール後更新するファシリティを矢印で示している。

5．ＣＴ方式の評価

(1) 従来方式との比較

表駆動方式、コンパイル方式と本報告のＣＴ方式の主な違いを表5.1に示す（処理速度についてはＣＴ方式の位置付けを示したものである）。ＣＴ方式の高速性は、コンパイル方式と同等の原理に基づいている。しかし、表駆動方式を混合するためコンパイル方式に比較して処理速度が劣ると考えられる。非同期動作を扱うことが可能な表駆動方式機能シミュレータのオーバヘッドは、分析例[4]によるとモデル評価を除くイベント処理等が処理時間の３分の２を占る。ＣＴ方式では、表5.1の２項目に示す処理により表駆動方式のオーバヘッド（イベント処理時間）の削減を試みている。

(2) 処理速度の評価

処理速度の評価実験を32ビット幅のデータを２相クロックで転送するシフトレジスタ回路を用いて行った。各レジスタは、クロックのエッジから1、2nsの転送遅延を持つ。

100段、200段、300段、400段、500段の各シフトレジスタに対して、50000ns（クロック周期20ns、分解能0.1ns）のシミュレーションを行った結果を図5.1に示す。図は、回路規模（等価なフリップフロップレベル論理回路の素子数）に対する処理速度（Ｍｅｌｏｎ２の論理シミュレーションとの処理速度比）を示している。図中実線は完全な同期回路のシミュレーション結果を、破線はシフトレジスタの10段毎に非同期に信号が伝搬するターミナルを挿入した回路の結果を示している。表5.2には各シフトレジスタの段数、論理回路換算素子数、論理回路換算評価素子数、機能シミュレーションと論理シミュレーションの処理時間（CPU）を示す。なお、以上の評価は16MB実メモリを持つSolbourne series 4/600（SUN4/260とほぼ同等の性能）上で行った。

図5.1は、本方式が同期動作、非同期動作の両者とも機能シミュレーションが論理シミュレー

ションに比較して10倍以上高速であることを示している。これから本方式は、非同期動作をシミュレーションする場合にも、従来のコンパイル方式に匹敵する性能を持つといえる。

表5.1　シミュレーション方式の比較

	CT方式		表駆動方式	コンパイル方式
評価関数の単位	複文単位 同期動作	非同期動作	レジスタ転送 または機能素子	同期回路 モジュール
評価関数駆動の制御方法	同期動作 クロック イベント	非同期動作 参照する状態値の変化	イベント （状態値の変化）	外部クロック
非同期回路	可能		可能	不可
遅延	可能		可能	不可
処理速度	高速		低速	極めて高速

表5.2　シミュレーション時間

シフトレジスタ段数	回路規模＊（素子数）	評価素子数＊	機能レベル CPU時間（秒）	論理レベル CPU時間（秒）
１００	３２００	２３．９Ｍ	９３	１２０９
１００（非）	３２００	２３．９Ｍ	１００	１２０９
２００	６４００	４７．７Ｍ	１７４	２４４９
２００（非）	６４００	４７．７Ｍ	１８７	２４４９
３００	９６００	７１．３Ｍ	２６２	３６８７
３００（非）	９６００	７１．３Ｍ	２７０	３６８７
４００	１２８００	９４．７Ｍ	３３６	４９５７
４００（非）	１２８００	９４．７Ｍ	３６６	４９５７
５００	１６０００	１１８Ｍ	４２０	６２７４
５００（非）	１６０００	１１８Ｍ	４３９	６２７４

＊　素子数：フリップフロップレベル論理回路で換した値を示している。

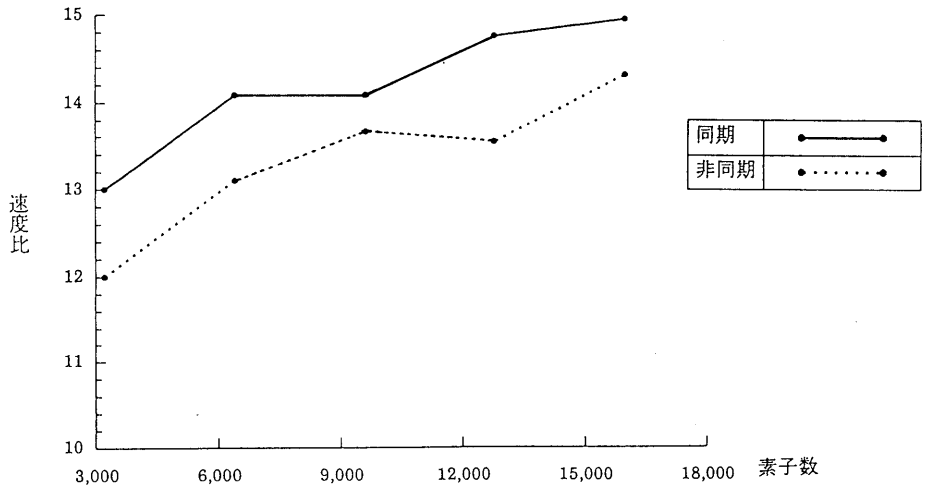

図5.1　シミュレーション処理速度

6. おわりに

本報告では、機能記述核言語ＭＨＤＬについて述べ、ＭＨＤＬを入力とする新たなシミュレーション方式としてＣＴ方式を提案した。評価結果では、同期動作、非同期動作の両者とも論理シミュレーションに比較して10倍以上の処理速度が得られた。

今後の課題として、より詳細な性能分析を行い処理速度の改善を図ることとミックスレベルシミュレータとしての観点から提案したＣＴ方式の有効性の確認をすることがある。また、実用化にあたって、ユーザインターフェースの良いシミュレーション環境を構築する必要がある。特にソースレベルデバッグ機能の開発が重要である。

参考文献

[1] 吉田、村岡、秋濃：スイッチレベルシミュレーションの一手法、情報処理学会設計自動化研究会資料48-4、pp.1-8,1989.

[2] ＬＳＩ設計用記述言語の標準化に関する調査研究、日本電子工業振興協会（平成元年３月）.

[3] 高井、水野、村岡：機能レベルシミュレーションの一手法、　DAシンポ'89資料、1989.

[4] C.Hansen:Hardware Logic Simulation by Compilation,25th DAC,pp.712-715,1988.

[5] L.Soule,T.Blank:Statistics for Parallelism and Abstraction Level in Digital Simulation,24th DAC,pp.588-591,1987.

[6] 水野、宮阪、井川他：高速化イベント駆動方式によるＲＴＬ機能シミュレーション、信学技報、Vol.88,No.80,pp.15-21,1988.

[7] 桶浦、星野、上田他：テーブルドリブン方式論理機能シミュレータ、情報処理学会設計自動化研究会資料16-2、pp.1-10,1983

[8] 星野、唐津、中島：ＨＳＬ－ＦＸと言語標準化、信学技報、Vol.87,No.126,pp.1-8,1987.

[9] A.Miczo,D.Mohapatra,S.Perkis, K.Kaufmf,K.Huang:The Effects of Modeing on simulator Performance, IEEE Design & Test, Dec.,pp.46-54, 1987.