計算機アーキテクチャ　98－10
設　計　自　動　化　65－10
（1993. 1. 21）

グラフィック入力による動作機能設計の一手法

松本 道弘　　高井 裕司　　岩崎 知恵　　村岡 道明

松下電器産業 株式会社
半導体研究センター　超LSIデバイス研究所

〒570 大阪府守口市八雲中町　3-1-1

あらまし　設計者が機能記述言語を用いて論理合成ツールや機能シミュレータを使用して機能設計検証をおこなう設計スタイルが普及してきた。しかしながら、機能記述言語の場合、(1)複数の標準言語が存在する、(2)同じ言語でもツールによって(例えば論理合成とシミュレータ)動作が一部異なる場合がある、(3)キャラクタベースであり他人が書いたものは理解しにくい、などの問題点が指摘されている。

我々は、このような問題点を解決する一つのアプローチとして、動作機能をグラフィック入力により設計する手法を検討した。設計者のドキュメンテーションスタイルにヒントを得てBchartと呼ばれるグラフィックエディタのプロトタイプを開発し、評価したので報告する。

和文キーワード　グラフィック、機能、設計、LSI、HDL、合成

A Functional Design Method based on Graphical Representation

Michihiro Matsumoto, Yuji Takai, Chie Iwasaki and Michiaki Muraoka

VLSI Devices Research Laboratory, Semiconductor Research Center,
Matsushita Electric Industrial Co., Ltd.

3-1-1, Yagumo-Nakamachi, Moriguchi, Osaka 570 JAPAN

Abstract　It becomes popular that the LSI designers verify the LSI functional design in the hardware design language(HDL) using the logic synthesis tools. As for the HDL, however, there are some issues as follows;

(1) many standardization of the HDL .
(2) result difference between tools (such as synthesis or simulator) is spite of using same HDL
(3) difficulty of understanding the HDL description written by others because the HDL is constructed by character.

We have been investigating the method of designing the LSI functional desig by the graphical representaion as a approach of solving the above issues. We are developing the Bchart system which is a functional design system on graphical representaion, as we get the idea from the designer's documentation style. In this paper we report the Bchart.

英文 key words　Graphic, function, design, LSI, HDL, Synthesis

1．はじめに

近年ますます大規模化するVLSIの開発において、その開発工数の増大、特に論理設計とその検証にかかる工数の増大が問題となっている。この解決方法の一つとして、近年論理合成ソフトウェアおよび機能シミュレータが用いられるようになってきた。

これらを用いる設計では、設計者は機能記述言語で機能設計を行ない、機能シミュレーションにより記述の動作を検証する。検証後、その記述を論理合成ソフトウェアにかけて、論理回路情報(ネットリスト)を得る。このような設計手法をとることにより、論理設計を直接人間がする方法にくらべて、短い期間で品質の良い設計が行なえるようになってきた。また、機能記述言語は半導体プロセスとは独立であり、論理合成によって複数のベンダーのASICライブラリにマッピングできるため、設計資産を機能記述言語で蓄積し次の設計でそれを流用して効率化を図る、いわゆる流用設計環境が構築されることも期待されている。

しかしながら、論理合成が使われるにしたがって、機能記述言語を用いた設計手法に関して、いくつかの問題点が表面化してきた。まず第一に、機能記述言語としては従来Verilog-HDLが使われてきたが、このほかに標準化作業が行なわれているものでは、VHDL、UDL/Iなどがあり、複数の言語が標準化される可能性が大きい。これは、設計資産の蓄積と流用には大きな障害となる。これらはそれぞれ言語の設計思想が異なり、カバーする領域も違うため、数年の間はそれぞれがそれなりに使われていくだろうと予想される。言語のカバーする領域が違うということは、ある言語で書いた記述を他の言語の記述に変換する場合に、単純な変換ではできないという問題を引き起こす。

第二に、同じ言語であっても、それを処理するツールによって動作が異なってくる場合があるという問題がある。例えば簡単な例で言うと、ある変数に対して複数の代入が起こりうる場合、機能シミュレーションでは、衝突が起こった場合にそこを不定値として表示することにより、設計上の問題点が見つかりやすいよう処理される。ところが論理合成ツールの場合では、このような記述の場所では、実際には信号衝突が起こらない設計になっていても、それを十分認識できずに無駄な優先順位処理回路を生成してしまうことがある。これに限らず一般に、機能シミュレーションの処理がシミュレーションの実行によって動的に行なわれるのに対し、論理合成の処理は言語記述を静的に解釈するという違いがあるので、その間に微妙な差が出てくるのは避けられない問題である。

第三に、機能記述言語は文字列の集まりであり、他人が書いたものはわかりにくいという問題がある。かつて論理シミュレーションが初めて使われ出した頃には、文字ベースのネットリストを人間が手入力していた。しかしながら、規模が大きくなってくると結局見にくくなり、図面(回路図)を専用のエディタで書き、それからネットリストを生成するという現在のスタイルになった。機能記述言語でも、プロセッサなどの動作を書こうとすると、数1000行以上にも及ぶことが珍しくなくなってきた。こうなると、書いた本人以外には理解しにくい。一方で、大規模LSI(システム)は複数の設計者で協力して設計を進めなければならないため、このような意志の疎通の障害によって、機能設計の効率は規模が大きくなるほど急激に悪くなることが、大きな問題である。

我々は、機能設計の領域においても、ネットリストがたどったのと同様に、グラフィック化への道をたどると考えた。機能レベルの情報をグラフィックで表現する試みとしては、[1]、[2]などがあり、また現在市販されているものにi-Logix社のExpress-HDL[3][4]がある。これらは、状態遷移図を拡張したようなものに記述言語のテキストを混在させたような形式のグラフィック表現となっている。特にi-Logixのものはソフトウェア

のCASEツールの流れから出てきたものであり、状態遷移を階層に拡張し、機能記述言語の特徴である並列動作の表現を可能にしたものである。

しかしながら、これらのグラフィック入力では、動作の仕様すなわち記述言語で順次処理文に相当するレベルのものは表現しやすいが、ハードウェアモデルとの対応の面でわかりにくいと言われている。現在の論理合成の技術レベルでは、ある程度合成結果を予想して機能記述のスタイルや分割の仕方を工夫しなければ良い結果を得られないことが多いが、前述のツールでは合成結果が予想できないという所がネックとなり、まだまだ一般の設計者には浸透していない。

本稿では、機能レベルのグラフィック表現の方法として、設計者が従来からドキュメントの記法として使用してきた表現形式(ブロック図など) を取り入れた、新しい表現形式Bchart(Behavioral Chart) を提案する。このBchart を用いることで、機能記述言語を意識せずに機能設計/検証をすることができ、論理合成の結果との対応もとりやすくなる。

我々は、Bchart の有効性を確認するために、グラフィックエディタのプロトタイプを開発し、評価を行なった。プロトタイプの評価では、プロセッサの機能動作をBchart で入力し評価を行なったので、あわせて報告する。

2. Bchart の概要

Bchartは、複数のグラフィックウィンドウの集まりで、レジスタ・トランスファ・レベルの機能動作を表現する。ウィンドウとして、以下に挙げるものを備える。

(1) 状態遷移図
(2) レジスタトランスファ (RT)
(3) 真理値表
(4) 論理式

以下、それぞれのウィンドウについて説明する。

2－1. 状態遷移図 (Statemachine)

設計者が用いる状態遷移図にはいろいろなスタイルのものがあるが、Bchart では、単一のステートマシンを一つの状態遷移図で表現するように扱う。すなわち、状態遷移図の中でアクティブな状態は常に一つのみであり、状態の遷移はすべて単一のクロックに同期したタイミングで行なわれる。

状態遷移図の例を図 1 に示す。図中のState1、State2 などが状態であり、それらを結ぶ矢印に従って状態遷移動作が起こる。矢印の横には遷移条件が書かれる。図が見にくくなるのを避ける意味で、遷移条件に書けるのは単純変数のみとし、論理式を直接書けないようにした。遷移条件の定義は、後で出てくる真理値表あるいは論理式など、別のウィンドウで定義される。

2－2. レジスタトランスファ (RT) 表現

レジスタトランスファ (以下 RT) は、設計者が従来使用してきたブロック図あるいはデータフローに似た形式のグラフィック表現である。記憶素子 (メモリ、レジスタ) やバス (ターミナル) などのファシリティを置き、その間のデータの流れを矢印で表現する。転送条件は矢印の横に書かれるが、これは状態遷移の時と同様、単純変数のみとした。

RT で用意したファシリティには、以下のものがある。

・レジスタ (記憶機能をもつ変数)
・ターミナル (記憶機能のない変数)
・ラベル (1 ビット単純変数)
・I/O ピン
・コネクト (Page 間接続ポート)
・標準演算器
　(論理演算、算術演算、他)
・メモリ (ROM/RAM)
・サブモジュール

RT の例は、5 章で後述する。(図 5 参照)

2－3. 真理値表 (Truth Table)

設計者の使う真理値表には、クロックに同期した動作を表現できるように拡張されたものもあるが、Bchart では、組合せ論理の動作を表現するものだけに限定する。

図２が真理値表の例である。左端の列に書かれている変数を、それぞれ行毎に定義する。一マスの中に複数書かれているものはORがとられ、横方向にはANDがとられる。図2の中の変数SelLowの場合、定義を論理式に書き直すと図３の論理式となる。

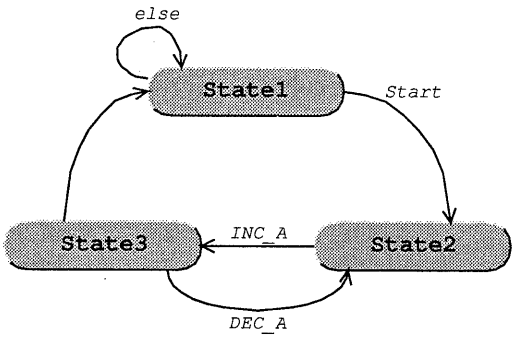

図１．状態遷移図の例

２－４．論理式 (Boolean Expression)

一般的な論理式の表現形式に準じた形式である。演算は、比較演算以外は１ビットに限定した。これは、Bchart 全体を通して、ビット幅のあるものは全て RT で扱うという思想で統一したためである。

図３は論理式の例である。

SelGen	EN_A	EN_B	Sel[1:0]
SelLow	0	1	0 1 2
SelHigh	1	1	3

図２．真理値表の例

```
SelLow = (EN_A==0) & (EN_B==1)
         & ((Sel[1:0]==0)or(Sel[1:0]==1)or(Sel[1:0]==2))
```

図３．論理式の例

３．Bchart 入力システムのソフトウェア構成

我々は、Bchartの有効性の検証のために、プロトタイプを開発した。これは、以下のソフトウェアで構成される。

(1) Bchart エディタ

Bchart の入力/ 修正を行なうグラフィックエディタである。Xウィンドウ上で動作する。

(2) Bchart/HDL トランスレータ

Bchartエディタで作られたデータを機能記述言語 (HDL) に変換する。

現在は Verilog-HDL が完成しており、VHDLやUDL/Iも検討中である。

(3) 対話型 Bchart シミュレータ

Bchartの図上で機能シミュレーションが行なえる対話型シミュレータである。現在開発中。

ソフトウェア構成と処理の流れを、図４にしたがい説明する。設計者はまず、Bchart エディタを用いて、機能動作をBchartで表現する。入力されたデータは、中間ファイルに出力される。その後、対話型シミュレータを用いてデバッグを行ない、修正すべき箇所があればエディタに戻って修正する。

このような作業を繰り返してデータが完成したら、Bchart/HDL トランスレータを起動し、中間フォーマットから機能記述言語に変換する。出力する機能記述言語は、Synospsys 社の HDL コンパイラの仕様に沿ったVerilog-HDL である。

４．実データによる評価

Bchartの評価をするために、プロセッサの実データを Bchart で入力した。入力したのは、kue-chip[5]のデータである。kue-chipの仕様に基づきデータを Bchart で入力し、Verilog-XL でシミュレーションを行なった。なお、今の段階では、論理合成ツール(Synopsys)への言語出力に関しては現在評価中のため、今回の報告では、論理合成までは行なわなかった。

図5は、RTで再上位階層の図面である。図6がkue-chipのブロック図であり、この部分に対応する。図中にあるように、主要なバスラインにあたる所にターミナルを置き、各部分からの転送関係を明示した矢印と、それを制御する条件の文字列で表現される。図中には、インクリメンタ、RAMと、ユーザー定義のサブモジュールがある。サブモジュールの中身は、一つ下の階層として他のページで定義される。この 2 つの図を比較してわかるように、RT表現はブロック図と全く同じレベルである。

図７は、プロセッサの実行フェーズを制御するステートマシンの状態遷移図である。表1にkue-chipの命令実行フェーズ(仕様)の図を示すが、この図からわかるように、kue-chipでは命令の種類によって、実行に必要なクロック数が異なる。図 7 の状態遷移図では、実行フェーズのP0 ～ P6 に相当する状態を定義し、命令の種類に応じたクロック数で状態遷移が一回りするようにした。

kue-chip には、文献[6]にあるように、人間が作成した Verilog-HDL 記述が存在する。今回はそれを入手し、Bchart で入力してVerilog-HDL を出力させた場合との比較を行なった。両方の記述に同じテストパターンを与え、Verilog-XL でシミュレーションを実行し、両者のシミュレーション結果が一致することを確認した。

表 2 は、Verilog-HDL 記述同士を比較したものである。Bchart から生成された Verilog-HDL の方が行数が多くなっているが、これは function 文を多用していることに起因する。現在は、Bchart 上で演算器の部分が、全て個別の function 文となる。手書きのものは、assign文を多用しているが、Bchartのほうは assign 文を全く出さない。どちらが良いかということについては、いくつもの例を作

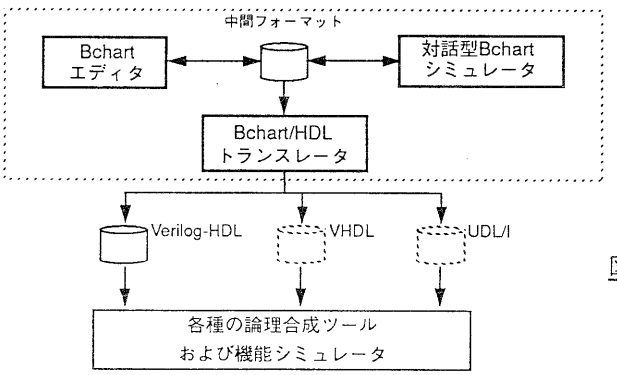

図４．Bchart入力システムのソフトウェア構成

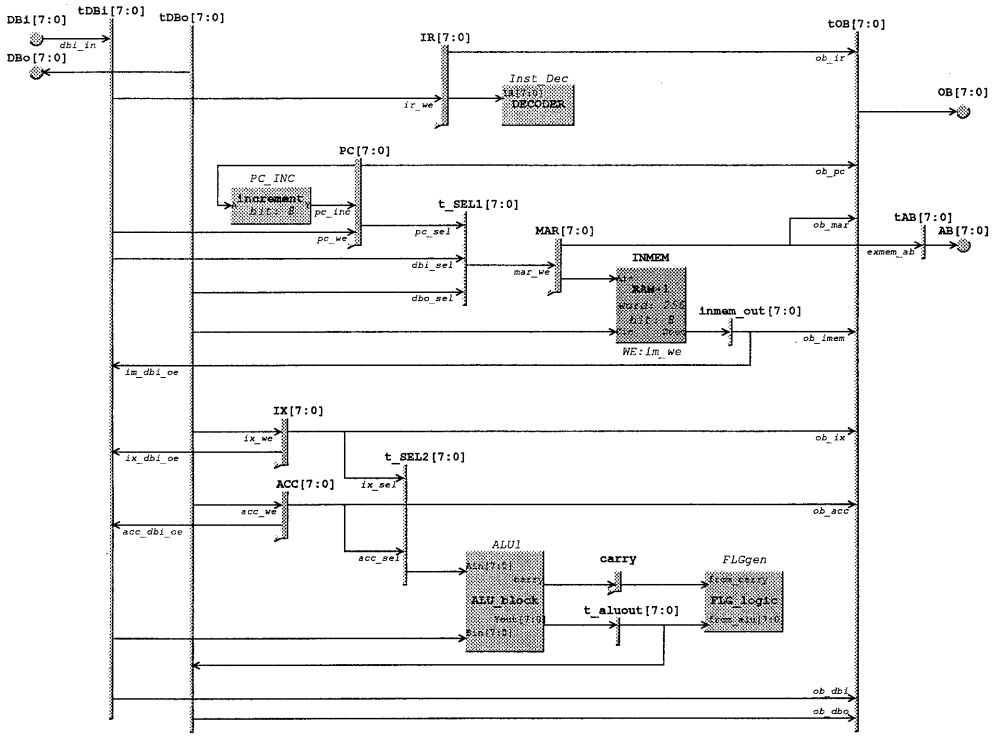

図5. kue-chip TOP 階層 (RT 表現)

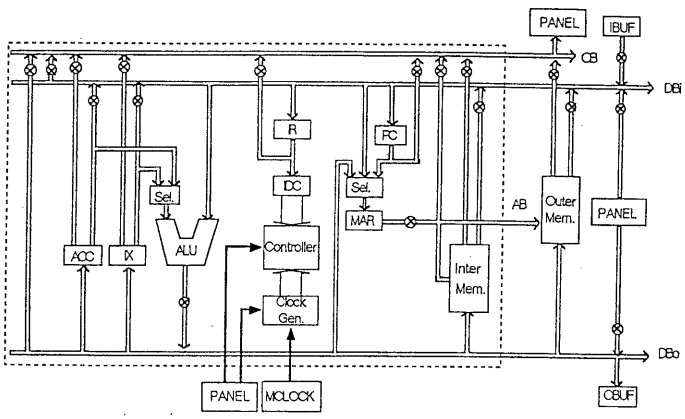

図6. kue-chip のブロック図

ACC	ACCumulator	MAR	Memory Address Register	OB	Observer Bus
IX	IndeX register	Clock Gen.	Clock Generator	DBi	Data Bus for input
Sel.	Selector	Inter Mem.	Internal Memory	DBo	Data Bus for output
ALU	Alithmetic logic unit	IBUF	Input BUFfer	AB	Address Bus
PC	Program Counter	OBUF	Output BUFfer	⊗	3 State Buffer
IR	Instruction Register	Outer Mem.	Outer Memory		
IDC	Instruction DeCoder	MCLOCK	Master CLOCK		

って論理合成にかけてみる必要があり、今後の検討課題である。状態遷移の部分については、どちらも Synopsys の論理合成に合わせた記述としているため、殆ど差はなかった。

表 3 は、Verilog でのシミュレーション時間の比較である。全てのパターンについて Bchart から変換した記述の方が若干時間がかかっているが、これも function 文が多い分のオーバーヘッドがあるためと思われる。

なお、今回 kue-chip を Bchart で入力した担当者は機能設計の経験が全くなかったため、設計工数などの定量的なデータの比較は出来なかったが、習得時間という点では、Bchart は機能記述言語の 1/4～1/5 の時間で使用できるようになることがわかった。

5．今後の課題

今後の課題として、まず、論理合成結果による調整がある。現在、Bchart から Ver-

		P0	P1	P2	P3	P4	P5
H		(PC)→MAR PC++	(Mem)→IR	HALT			
NOP				NOP			
IN				(IBUF)→ACC	FLAG CLEAR		
OUT				ACC→(OBUF)	STROBE OUT		
SHIFT				SHIFT	STATUS SET		
IS SA EOR OR AND	ACC, IX			(α)→ (α')→ ALU→α	STATUS SET		
	即値			(PC)→MAR PC++	(IX)→ (Mem)→ ALU→MAR	STATUS SET	
	直接				(Mem)→MAR	(α)→ (Mem)→ ALU→α	STATUS SET
	修飾				(IX)→ (Mem)→ ALU→MAR		
LD	ACC, IX			(α)→(α')			
	即値			(PC)→MAR PC++	(Mem)→(α)		
	直接				(Mem)→MAR	(Mem)→α	
	修飾				(IX)→ (Mem)→ ALU→MAR		
ST	直接			(PC)→MAR PC++	(Mem)→MAR	(α)→Mem	
	修飾				(IX)→ (Mem)→ ALU→MAR		
BRANCH				(PC)→MAR PC++	STATUS CHECK (Mem)→PC		

α は ACC 又は IX

表 1．kue-chip の命令実行フェーズ仕様

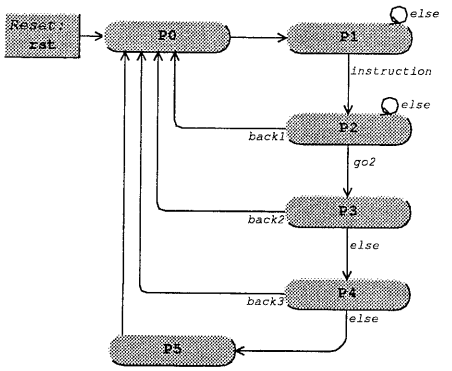

図 7．実行フェーズの状態遷移図

	Bchart	人手作成
全体行数	512 行	467 行
if 文	74	85
asisign 文	0	9
case 文	15	20
function 文	33	5
task 文	0	10

表 2．Verilog-HDL 記述の比較

ilog-HDLを出力してSynopsysの論理合成にかけ評価中である。現在の Synopsys に合わせた記述では、シミュレーション上正しく動作しないケースがある。両方を正しく動作させるために、同じ Verilog-HDL であっても、シミュレータ用/論理合成用というように種類を分けて出力する必要が出てくるかもしれない。

また、現在開発中の対話型シミュレータは、図のイメージの上でシミュレーションを行なうことによって、カットアンドトライの多い初期段階での設計検証の効率を向上させようというものである。これもまた、論理合成との動作合わせが必要である。

将来的には、Verilog-HDL だけではなく、VHDL、UDL/I などへのインターフェースも必要になってくると考えられる。この場合にも、言語やツールによる動作の違いという部分を、言語出力のマッピングルールの変更のみで対応していく予定である。

6．むすび

我々は、Bchartという機能動作の表現方法について、グラフィックエディタと機能記述言語変換のプロトタイプを開発し、プロセッサ (kue-chip) のデータを入力して評価した。その結果、設計者が直接機能記述言語を書く場合と比べて習得時間やドキュメント性の面で大きな向上があり、記述行数/シミュレーション時間ではほぼ同等であることがわかった。Bchart 入力システムを使えば、本稿で報告したkue-chipの例にみられるように、ブロック図と同じレベルで入力ができ、かつそのまま論理合成につながるため、機能設計効率の大きな向上が期待できる。

テストパターン	Bchart	人手作成
pattern 1	2.7	2.4
pattern 2	2.6	2.1
pattern 3	4.0	3.4
pattern 5	4.6	3.8
pattern 6	3.0	2.6
pattern 7	1.4	1.2

CPU 時間 単位: sec Solbourne Series4/600: Memory 32MB

表3. シミュレーション時間の比較

また、Bchart入力システムを使うと、データパス系/コントロール系などのような回路の性質によりあらかじめ分けて入力できるため、データパス合成やステートマシン合成などのツールにも、インターフェースが容易であると思われる。

謝辞

本報告では評価データとして kue-chip を使用した。評価に際しては、ASTEM の神原氏に有益な助言をいただいた。また、kue--chip の Verilog 記述の入手にご協力いただいた三菱電機の野地氏に深く感謝します。

参考文献

1) S.Narayan, F.Vahid, and D. Gajski, "System Specification and Synthesis with the SpecCharts Language", Proc. of ICCAD-91, pp.266-269, IEEE, 1991.
2) J. Lahti, M.Sipola, and J.Kivela, "SADE: A Graphical Tool for VHDL-based System Analysis", Proc. of ICCAD-91, pp.262-265, IEEE, 1991
3) P. Clemente, P.Runstadler, L. Specter, and K.Walsh, "From StateCharts to Hardware FPGA and ASIC Synthesis", VHDL International User's Forum Conference, 1992.
4) 山本:デザインオートメーションコンフェランス(DAC)に見るエレクトロニックデザインオートメーションの最新動向, コンピュータデザイン, Vol.9, pp.24-27(1990)
5) 神原, 教育用マイクロプロセッサ:KUE-CHIP, 情報処理学会研究報告, 90-ARC-82-6(1990)
6) 神原, 安浦,P.Kukkal,H.Kobayasi, 野地, 小栗: ハードウェア記述言語の比較, 情報処理, Vol.33, No.11, pp.1269-1283(1992)