C-002

上位ハードウェア設計言語 Melasy+ による VHDL コード生成と動作検証

VHDL Code Generation and Verification by A Meta Hardware Description Language "Melasy+"

白鳥 航亮 †　　和崎 克己 ††
Kousuke Shirotori　Katsumi Wasaki

1 はじめに

IT 技術の発展により，様々な環境で電子機器が動作しており，年々規模・数ともに増加傾向にある．また，様々な社会システムがこれらの技術と信頼性に依存しており，とりわけディジタルシステムにおける技術と信頼性は社会システムの維持に必要不可欠である．仕様通りに確実に動作する信頼性の高いシステムを，より安価に効率よく設計できる環境が望まれている [1].

ハードウェア設計の正当性を形式的にチェックするツールとしては，NuSMV[2] 等のモデル検査ツール (Model Checker)[3] が存在する．これらのツールを用いることで設計の正当性の評価を自動で行うことができる．しかし，NuSMV によって実ハードウェアの設計を全て記述するには，極めて低級な言語を利用しなければならない．また，VHDL[4] 等の言語で設計したシステムを，検証目的のために，改めて NuSMV 等の言語で再記述する工程が発生する．同じ設計を異なる言語で複数回行うことは，実際の開発現場の状況に鑑みるとコスト・納期などの制約において困難である．

上記問題を解決する手段として，一つのコードから様々な処理系に対応するように設計された上位ハードウェア設計言語 Melasy+[8] がある．Melasy+ は C++ のライブラリとして実装されている．Melasy+ のコードは，Melasy+ 用の特殊な型や構文を用いて C++ のコードとして記述する．

本稿では，組み込み自己テスト機能付きバスアービタについて記述した Melasy+ コードから VHDL コードを自動生成し，Xilinx ISE シミュレータ上で動作検証を行った事例について述べる．得られた VHDL コードに一切の変更を加えることなくシミュレータに与え，正常に調停動作が行われることを確認した．

2 研究背景

2.1 モデル検査

設計されたシステムが仕様を満足するのかを確かめる設計段階での検査手法．対象をオートマトンと対応付けられた，状態機械として記述する．検査は，モデル検査器によってモデルの取りうるすべての状態について，自動的且つ網羅的に行われる．

† 信州大学大学院工学系研究科, Graduate School of Science and Technology, Shinshu University.
†† 信州大学工学部, Faculty of Engineering, Shinshu University.

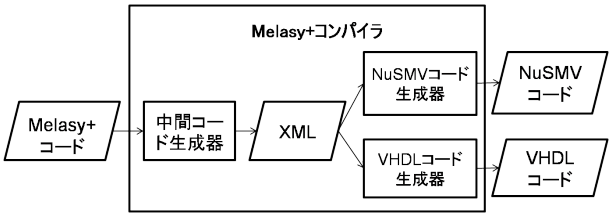

図 1　Melasy+ コンパイラ処理系の位置づけ

2.2 ハードウェア上位設計系

同じ設計を異なる言語で複数回記述することは一貫性を保つ観点から困難である．VHDL，Velilog，NuSMV などの異なる言語の上位に，設計用の共通言語を設置し，このコードから各々のコードを生成するハードウェア上位設計という手法が存在する．ハードウェア上位設計言語には，本報告で用いた Melasy+ の他に，HDCaml[5]，Confluence[6] などが存在するが，すでに開発は終息している．

2.3 組み込み自己テスト回路

外部にテスト回路を新たに設けるのではなく，被テスト回路と同じチップ上に埋め込んでしまう技術である．テスタコストの削減や，テストの容易化などの利点がある反面，テスト時の電力増加による誤動作が懸念されるなどの注意点も存在する [7].

3 上位ハードウェア設計言語 Melasy+

3.1 概要

Melasy+ は NuSMV や VHDL など，様々な処理系向けのコードを生成することを目的とした，上位ハードウェア記述言語である．Melasy+ と他の言語の関係について，図 1 に示す．Melasy+ コンパイラは，構文パーサ・中間コード生成器と，各対象言語向けコード生成器から構成される．中間コード生成器は C++ のクラスライブラリとして実装されている．Melasy+ コードを構文パーサ・中間コード生成器に与えると，中間コードとして XML コードが出力される．得られた XML 中間表現コードを各言語向けコード生成器に与え，各言語のコードを得る．

3.2 言語仕様

Melasy+ コードは C++ のクラス拡張として記述し，文法は C++ の規則に従う．しかしながら，値の保持に int や char といった C++ の既存の型を利用することはできない．Melasy+ では Logic 型と Digit 型を値の保持に用いる．それぞれ，1 ビットと N ビットの値を表すのに用いる．Logic 型では char 型の ‘ 0 ’, ‘ 1 ’ の

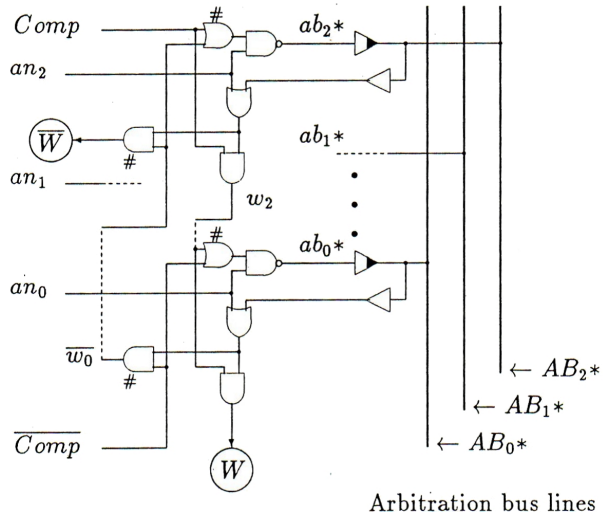

図 2　自己テスト機能付きアービタ回路図 (1 ノード分)

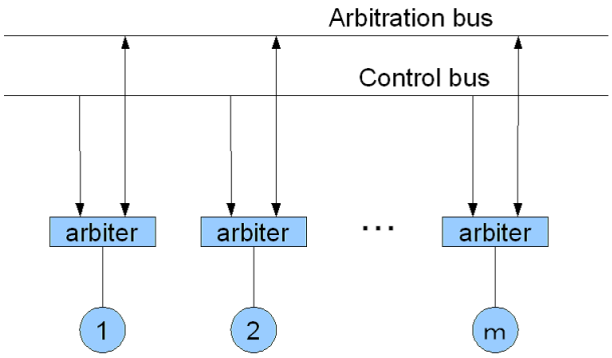

図 3　アービタ全体図（3 ノード）

いずれかの値を代入することが可能である．Digit 型ではテンプレート引数にビット幅を指定することで，任意のビット幅の型として扱うことが可能であり，値の代入には文字列定数か int 型の値を代入することで行う．また，Melasy+ ではコンポーネント単位で回路を記述する．各コンポーネントは入出力を持ち，出力の動作，入力の接続などを定義することで，ハードウェアの仕様を記述する．コンポーネントの記述に必要な機能は Component クラスが提供する．各コンポーネントの動作はコンストラクタ内で定義を行う．入出力は in，out，sync などの関数を呼び出すことで定義可能である．出力の定義には switch_，case_，default_といった特殊な条件分岐構文を利用することが可能である．for や if のような C++ の言語機能をプリプロセッサの様に使うことが可能であり，さらに，再利用可能なコードを関数として定義し，呼び出すことも可能となっている．

3.3　XML 中間生成系

Melasy+ コードを C++ コンパイラでコンパイルして得られる実行可能ファイルを駆動することでコード生成が行われ，XML による中間表現を得ることが可能である．コード生成は getXML 関数によって行われ，これの返り値を標準出力等に出力することで XML 中間コード生成器として動作する．

3.4　各言語向けコード生成器

現在 2 つの言語向けのコード生成器が実装されている．対象となる言語は NuSMV と VHDL である．両コード生成器共に python を用いて記述されており，Melasy+ の中間コードとして得られた.XML ファイルを入力とし，NuSMV のコードである.smv ファイル又は，VHDL のコードである.vhd ファイルを出力する．

4　組み込み自己テスト機能付きバスアービタの記述

4.1　概要

バスアービタは，ノード間でデータ転送するためのバスを複数のノードで共有するときに，データの衝突が起きないよう，各々の上位モジュールからのバス使用要求に基づいて，バスの使用権（バスマスタ）をどのノードに与えるか決定するための回路である．本稿で用いた高機能バスアービタ [9] の Melasy+ コードは，すでに，モデル検査器 NuSMV 向けの対象コード自働生成と生成コードに対するモデル検査に成功しており，最大の id を持つノード唯一つがバスの使用権を獲得する仕様を満たすことが証明されている [10].

4.2　ハードウェア構成と組み込み自己テスト機能

自己テスト機能付きバスアービタの 1 ノード分の回路ならびにアービタ全体の構成図（ノード数 3 ）を，それぞれ図 2，図 3 に示す．アービタの 1 ノードは，優先度を示す ID を持つ．調停は 2 フェーズに分けて行われ，ID の比較を行うための調停用バス線 (AB_{i*}) で共有接続されている．

2 フェーズ調停動作について以下に説明する．各ノードは，アービタに対してバス要求信号 $Comp$ をアサートし，調停動作を開始する．バスに接続されたアービタ同士で上位ビットから順次調停（レベル H 優位）が行われる．バス要求を出しているノード中で，ID の値が最も大きい高々 1 ノードに限り，勝者となり，バスマスタとなったことを示す信号線 w がアサートされる．次に ID を交番し，$CompB$ をアサートし，2 回目の調停を行う．2 回目の調停は下位ビットから，順次調停が行われる．2 回目の調停で勝者となれば，信号線 wB がアサートされる．回路が正常動作していれば，同一のノードの w ・wB がそれぞれ 1 回目，2 回目でアサートされる．一方，回路が故障している場合には 1 回目あるいは 2 回目のどちらか一方で「勝者」である結果となるので，w ・wB の値を監視すれば，故障の検知が可能である．

4.3　Melasy+ による記述

Melasy+ はコンポーネント技術を使ったオブジェクト指向プログラミングによる抽象化が可能である．この高い記述性を用いて，任意のノード数と調停バス線を有する，組み込み自己テスト機能付きバスアービタの上位設計を行った．図 4 にアービタの Melasy+ コード記述例を示す．図 4 は各アービタを記述した部分である．一ビット分の調停回路（ArbiterElement）はすべて同様の構造をしており，これを任意のビット数分接続することにより，一つのアービタを実現する．最上位，最下位の ArbiterElement には特有の入出力信号線が存在するが，その他の ArbiterElement の接続は一様である．アービ

```
template<int N>
class Arbiter : public Component
{
public :
  Logic an[N]; //ID
  Logic comp;
        …
  ArbiterElement BA[N];
        …
  Arbiter()
  {
    in(an);
    in(comp);
    …
    //PortMap For BusArbiter
    for(int i = 0; i < N; i++)
    {
      if(i == 0){
        PortMap pmba[] = {
          BA[i].comp  <= BA[i+1].w,
          BA[i].compB <= compB,
          BA[i].an    <= an[i],
          BA[i].ab_wb <= ab_wb[i]
        };
        instance(BA[i], pmba);
      }
    …
```

図 4　Arbiter の Melasy+ 記述（抜粋）

タのビット数を N で抽象し，上位クラスにおいて，インスタンスを作成する際にテンプレート引数に N の値を渡すことで，任意のビット数に対応するアービタを作成できる．

5　コードの自動生成と動作検証

Melasy+ コードをコンパイルし，得た XML 中間コードから VHDL コードを生成する．Xilinx ISE シミュレータを用いて，生成されたコードがバスアアービタの求められる動作をするか検証を行う．

5.1　NuSMV コード生成と仕様検証

NuSMV コード生成器は python 言語を用いて記述されている．XML 中間コードを入力として，NuSMV モデル検査器で駆動可能なコードを生成できる．本稿で対象とした組み込み自己テスト機能付きバスアービタについて，Melasy+ コードからの NuSMV コードの生成と NuSMV モデル検査器によるモデル検査に成功した事例がある．すでに，組み込み自己テスト回路の動作の正当性，ならびに網羅的検査によってバス優先権獲得に関する仕様を満たすことが証明されている [10].

5.2　VHDL コード生成

VHDL コード生成器は python 言語を用いて記述されている．VHDL モジュールを python のクラスとして実装した．引数として与えられた XML 中間コードファイルから DOM ツリーを生成し，generate_start 関数に渡すことで，VHDL コードの作成が開始する．VHDL モジュールクラスは，中間コードを元に，ポートの記述などに必要な信号線や式の追加を行い，最後に VHDL コードの出力を行う．図 5 に生成された VHDL コード例を示す．

5.3　テスト項目

バスアービタの基本的な仕様として同時に二つ以上のノードがバスマスタに決してならない，というものがある．本報告で用いたバスアービタの仕様としてバス使用

```
 architecture melasy_generated of Arbiter_3 is
component ArbiterElement
port (
  compB : in std_logic;
  ab : out std_logic;
       …
);
end component;
  signal ArbiterElement_0_w : std_logic;
  signal ArbiterElement_0_ab : std_logic;
  signal ArbiterElement_0_wB : std_logic;
       …
begin
  ab_0 <= ArbiterElement_0_ab;
  ArbiterElement_0 : ArbiterElement port map (
    compB => compB,
    ab => ArbiterElement_0_ab,
          wB => ArbiterElement_0_wB,
               …
  );
       …
end melasy_generated;
```

図 5　生成された VHDL コード（抜粋）

要求を出しているノードの内，最も大きい id を持つノードがバスの使用権を得る，というものがある．NuSMV を用いたモデル検査において，二つの仕様が満たされることが証明されている．そこで，実際に VHDL コードでのシミュレーションにおいて仕様を満たす動作が行われるのかを検証する．

ノード数 3，調停バス線 3 本でテストを行う．二段階の調停動作を行う上で，id の反転を行うことから，id の衝突や優先度の入れ替わりを防ぐため，id 同士の間隔を 2bit 以上離す必要がある．ノード数 3 で使用可能な id 値は 000~111 であるが，優先度入れ替わりの防止のため，まず 000 と 111 を除く．001~110 の中から，それぞれの id の間隔が 2bit 以上離れるように id を選択し，001,010,100 を使用する．バスの使用権を要求する信号のアサートを，各バスアービタに対して適切な間隔で指示する．最後に，バスの使用権を要求するノードの内，最大の id を持つノード唯一つのみが常にバスの使用権を得るかを確かめる．

5.4　動作検証と考察

Xilinx ISE シミュレータを用いて，生成されたコードがテスト項目を満たすか否かの動作の検証を行った．得られた VHDL コードには生成後に一切の変更を加えていない．図 6 にシミュレート結果を示す．

まず，各ノードに id を与える．各ノードの id はノード 0 (001), ノード 1 (010), ノード 2 (100) とし, バス使用権の優先度はノード 0 ＜ノード 1 ＜ノード 2 である．次にバスの使用権を要求する信号 $comp\_0, \cdots, comp\_2$ を準にアサートする．このとき，バスの使用権を獲得したこと示す信号 $w\_0, \cdots, w\_1$ のうち，常に高々一つのみが H レベルになっている．また，バスの使用権を要求しているノードのうち，最大の id を保持するノードが必ず勝者になっている．よって本ケースでは調停の第一段階は正常に行われている．

次に，それぞれの id を bit 反転し，調停の第二段階に移る．第二段階ではバスの使用権を要求するノードは $compB$ をアサートすることで，調停に参加する．$compB\_0, \cdots, compB\_2$ を準にアサートし，第一段階と同様に結果を観察する．第二段階の調停は第一段階とはことなり，最下位ビットより行われる．よって，反転後

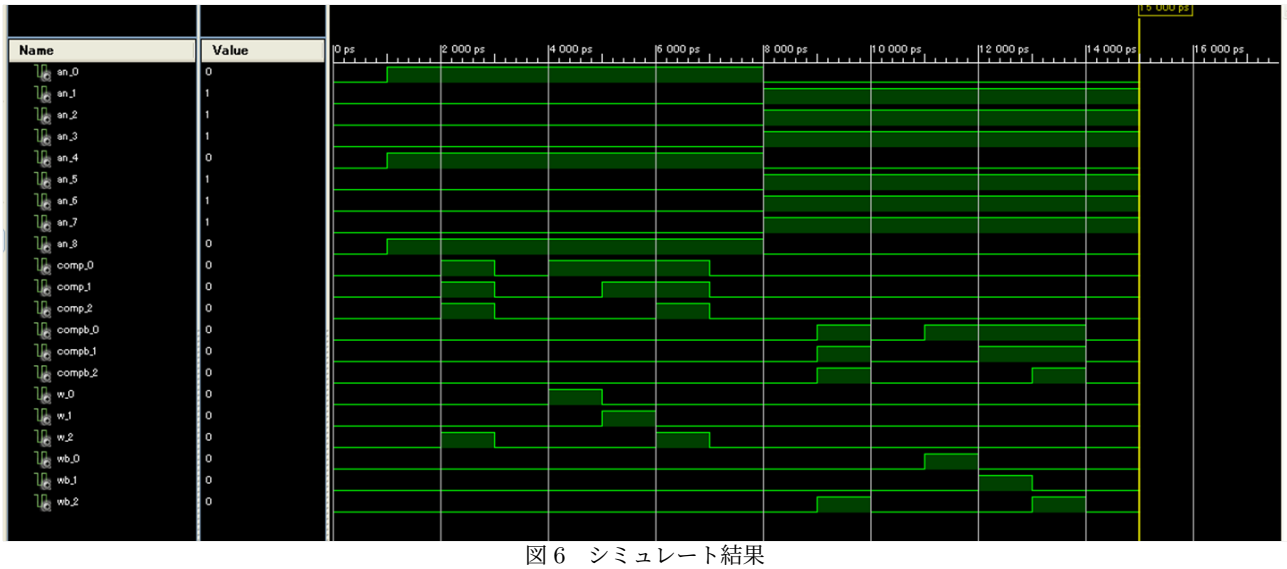

図 6　シミュレート結果

の id がもっとも小さなノードが調停に勝利する．第二段階では調停に勝利すると，信号線 wB がアサートされる．それぞれのノードの $CompB$ のアサートにともない，常に調停に参加するノードの内，最小の id を保持するもの唯一つが勝者となっている．よって，第二段階の調停動作も正常に行われている．

第一段階，第二段階ともに，最大の id を持つ，同一のノードが勝者となれば，正常な調停動作である．本ケースでは，第一段階，第二段階共に最大の id を持つノード 2 が勝者となっており，調停は正常に行われた．よって，本テストケースでは，仕様を満たす動作の確認ができたことになる．

6　まとめと今後の課題

組み込み自己テスト機能付きバスアービタの Melasy+ コード記述から VHDL コードの自動生成を行った．自動生成されたコードに一切の変更を加えることなく，駆動することを確認した．同時に二つ以上のノードがバスマスタにならない・バス使用要求を出しているノードの内もっとも大きい id を持つノードがバスの使用権を得る，という二つの仕様を満たす動作をするのかテストを行った．テストの結果，生成された VHDL コードが想定した入力パターンにおいて，期待された出力を得ることに成功し，二つの仕様を満たす動作をすることを確認した．

現在，Melasy+ は NuSMV ・VHDL の二つのコードが出力可能である．また,Melasy+ から自動生成された各コードは，それぞれ NuSMV モデル検査器・VHDL シミュレーターを用いて実際に駆動可能である．記述したモデルは，モデル検査によって仕様を満たす動作を行うのか，すべての状態について網羅的に検査が行われ，得られた VHDL コードは仕様を満たすことが保障される．従来は別々に記述を行っていたが，Melasy+ によって検証用・実装用のコードを一つのコードから自動的に得ることが可能となった．変更を加える際にも Melasy+ コードに変更を加えることで，すべてのコードに変更を反映できる．すべてのコードそれぞれに変更を書き加える必要がなく，時間コストを削減できる．

本稿の結果によって，当初想定した Melasy+ の基本機能の実装が完了した．今後の展望として，Melasy+ によって生成された VHDL コードを用いて，FPGA へ実装を行うことを考えている．また，論理ゲートのライブラリ化などの新機能を追加し，記述性やコードの見通しの向上を図る．さらに，対応言語の追加，Melasy+（C++）レベルでのシミュレート機能などの新機能の実装を行うと共に，より多くの応用事例の増加に努めたい．

参考文献

[1] E.M.Clarke, O.Grumberg, D.Peled : “Model Checking” ; MIT Press, 2000.
[2] NuSMV: a new symbolic model checker, http://nusmv.irst.itc.it/.
[3] B.Berard, M.Bidoit, A.Finkel, F.Laroussinie, A.Petit, L.Petrucci, Ph.Schnoebelen, P.McKenzie : “Systems and Software Verification Model-Checking Techniques and tools” ; Springer, 2001.
[4] VHDL : VHSIC Hardware Description Language, http://vhdl.org/.
[5] T.Hawkins : “The Caml Hump: HDCaml” ; INRIA /VASY, http://www.confluent.org/wiki/doku.php/.
[6] T.Hawkins : “Confluence System Design Language” ; INRIA/VASY, http://linux.softpedia.com/developer/Tom-Hawkins-24593.html.
[7] 米田, 梶原, 土屋 : “ディペンダブルシステム－高信頼システム実現のための耐故障・検証・テスト技術” ; 共立出版, 2005.
[8] N.Iwasaki, K.Wasaki : “A Meta Hardware Description Language Melasy for Model Checking Systems” ; Proc. of the 5th Int’l Conf. on Information Technology : New Generations (ITNG2008), 273-278, 2008.
[9] 時藤, 黒川, 古賀, : “セルフテスティングバスアービタの一実現法について” ; 信学論 D-I, 75(1), 30-40, 1992.
[10] 花里, 白鳥, 和崎 : “上位言語 Melasy+ による自己テスト機能付バスアービタの設計と NuSMV を用いた検証” ; 情報処理学会第 72 回全国大会講演論文集, 1, (1M-4), 151-152, 2010.