ISSN 0002-306X. Изв. НАН РА и ГИУА. Сер. ТН. 2005. Т. LVIII, № 1.

УДК 517.587

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ИНФОРМАТИКА

Т.Р. АТОЯН

ГЕНЕРАЦИЯ И ИНТЕРПРЕТАЦИЯ СКАНИРУЮЩИХ ЦЕПОЧЕК СИСТЕМ НА ЧИПАХ С ПОМОЩЬЮ СИСТЕМЫ ОБРАБОТКИ ШАБЛОНОВ

Предлагается методология создания тестовой среды, основанной на технике обработки шаблонов путем ее представления на языке CTL и специальной системе обработки шаблонов (TPS). Рассмотрены средства автоматизации генерации и интерпретации контента сканирующих цепочек для Систем на Чипах (SoC) с целью тестирования и отладки в соответствии со спецификациями пользователей.

Ключевые слова: системы на чипах, тестирование, сканирующие цепочки, шаблоны, обработка.

Благодаря последним достижениям в области проектирования и изготовления сверхбольших интегральных схем (СБИС, VLSI) возникло новое направление в технологии их создания – так называемые **системы на чипах** СнЧ (Systems on Chip - SoC). В основе этой технологии лежит интеграция в одной большой схеме готовых к повторному использованию компонентов (ядер) интеллектуальной собственности (IP component, cores). При этом, наряду с прочими, возникает проблема отладки и тестирования SoC, решаемая с помощью **сканирующих цепочек** (scan chain), через которые передается вся необходимая информация, предназначенная для этих целей.

Для защиты прав собственника IP компонент информация, необходимая для их применения, изготовления и тестирования, кодируется и сжимается, в частности, для уменьшения времени и цены тестирования, что, в свою очередь, порождает проблему ее декодирования и интерпретации при отладке и тестировании SoC в целом [1]. В процессе тестирования SoC, включая составляющие компоненты, создаются различные режимы и условия их проверок, при этом сами проверки и их результаты получаются через сканирующие цепочки (СЦ), и их содержимое (content) сравнивается с ожидаемым, которое формируется в процессе проектирования SoC [2].

В работе рассматриваются три аспекта указанной проблемы:

– интерпретация контента, получаемого через СЦ;

– генерация ожидаемого контента СЦ;

– сравнение полученных и ожидаемых контентов СЦ.

Каждая SoC имеет свою **архитектуру тестовой инфраструктуры** (TIA), которая зависит от специфических особенностей SoC и входящих в него компонент, что и определяет структуру СЦ, их контент и отладочную информацию. Различные реализации TIA, очевидно, влекут за собой различные структуры СЦ, их форматы и контент [3,4].

Гибкая **система обработки шаблонов** (TPS), имеющая средства представления различных форматов СЦ с помощью шаблонов, позволит легко их описать при новых различных реализациях TIA. В случаях, когда СЦ имеют регулярную структуру и формат, они легко могут быть представлены с помощью шаблонов для TPS [5].

Проблема автоматической интерпретации и генерации/сравнения СЦ на базе спецификации пользователя может быть решена с помощью соответствующего расширения TPS. В этом случае применяется многоступенчатый подход [5,7], как это показано на рис.1, когда результат предыдущего шага подается на вход последующего и т.д., до получения конечного результата.

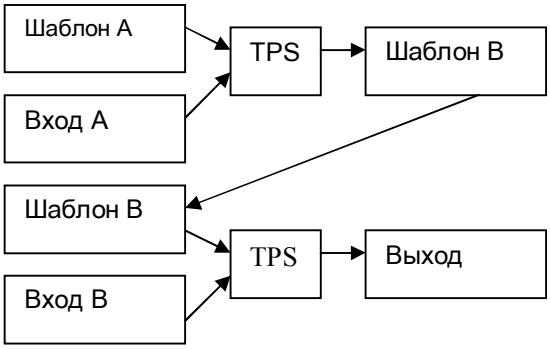

Рис. 1. TPS в режиме многоступенчатой обработки

В настоящей работе описываются представляемая пользователем спецификация, необходимая для интерпретации и генерации СЦ; шаги алгоритма их реализации, а также “тэги”, необходимые для реализации процедур генерации, интерпретации и сравнения шаблонов.

Интерпретация СЦ. Полное описание СЦ должно использоваться в качестве входной информации для этапов последующего синтаксического анализа (СА) и интерпретации их контента. Ниже рассматривается формат описания СЦ, который состоит из:

– архитектурной части, описывающей схемно реализованные элементы цепочек, такие как: сканирующие ячейки, соединительные входы и выходы, управляющие сигналы, синхронизаторы и др.;

– логической части, описывающей логическое применение информации, которую несут цепочки.

Формальное описание архитектурной части может производиться с помощью стандартных языков описания тестов, таких как BSDL [3] или CTL. Описание же логической части сильно зависит от конкретной реализации и может отличаться для различных SoC.

На рис.2 схематически представлен пример логического описания цепочки. Здесь последняя строка – algo_type/algo может определять тип

контента сегмента (двоичные числа, символы ASC II и др.) или спецификацию очередного алгоритма его декодирования.

```
Chain Description:
Chain_Length = n;
Number_of_Segments = m;
Segment_Names = a,b,c;

Segment Description:
Segment_Length = z;
Segment_algorithm = algo_type/algo;
```

Рис. 2. Формат описания СЦ

Для полного описания СЦ вводится некое расширение одного из принятых языков описания тестов, например CTL, который наиболее полно подходит для описания архитектуры СЦ. Необходимое расширение может представляться соответствующими средствами комментариев описания CTL. Назовем этот базовый язык вместе с целевым расширением – Chain Description Language (CDL). Описание на CDL представляет полную спецификацию СЦ, включающую: длину, число сегментов, имена сегментов и для каждого из них его длину и алгоритм интерпретации и др.

Таким образом, TPS должен быть выстроен так, чтобы распознавать вышеупомянутые описания и интерпретировать необходимые СЦ, что будет делаться путем СА контента и применением алгоритмов интерпретации, даваемых в описании цепочек.

Генерация СЦ. Эта задача может быть сведена к предыдущей путем описания алгоритма генерации контента СЦ на языке CDL и затем, посредством TPS, автоматической выработки требуемых результатов, т.е. генерирования ожидаемого контента.

Интерпретация СЦ / шаги алгоритма генерации. На основе представляемых пользователем SoC описания форматов СЦ TPS может синтезировать специальные шаблоны, включающие всю необходимую информацию для синтаксического анализа и интерпретации СЦ, или автоматически генерировать их ожидаемые (необходимые) контенты.

Алгоритмы интерпретации или генерации СЦ состоят из следующих шагов:

1. *Представление спецификации СЦ.* Пользователь SoC, исходя из своих специфических требований, создает полную спецификацию СЦ в описанном выше формате, включая алгоритмы интерпретации и генерации контента.
2. *Генерация шаблонов синтаксического анализа.* TPS генерирует специальные шаблоны, с помощью которых осуществляется СА

контента СЦ и генерируется их интерпретация или создается ожидаемый новый контент.

3. *Генерация декодированных или ожидаемых СЦ.* На этом шаге TPS обрабатывает шаблоны предыдущего шага и входные данные пользователя, в результате чего синтезируется интерпретированный или сгенерированный контент СЦ.

Графическое представление описанных выше шагов приведено на рис.3.

Указанные процедуры включают предварительный этап, обозначенный шагом 0, где представляются шаблоны для СА нового языка CDL, на котором производится описание цепочек. Далее пользователь дает описание цепочек и в качестве выхода на шаге 1 получает их внутреннее представление. Шаг 2 производится автоматически, т.е. не требует вмешательства потребителя, и на его выходе получаются два типа шаблонов – для интерпретации и генерации контента СЦ. На завершающем шаге 3 производится конечная интерпретация или генерация контента СЦ.

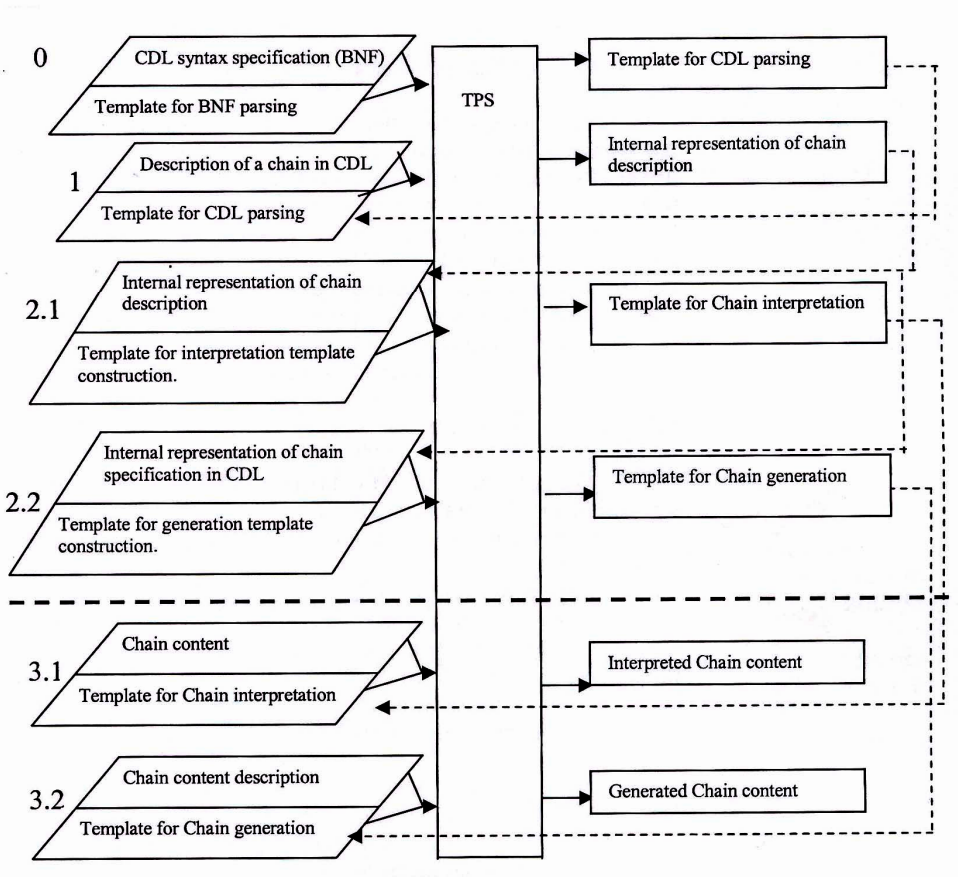

Рис. 3. Алгоритм интерпретации и генерации контента СЦ

TCXL коды, шаблоны. TPS оперирует шаблонами, описанными на языке TCXL [5], который является расширением языка XML. XML имеет иерархическую структуру и удобен для описания иерархических объектов. В нашем случае СЦ благодаря TIA имеют регулярную структуру, следовательно, легко описываются через TCXL. TCXL имеет также все необходимые средства для обработки входных и представления выходных текстов.

Приведем краткое описание языка.

TCXL разработан для представления логики преобразования, СА входных текстов, их преобразования в выходные и состоит из функциональных "тэгов" и функций, используемых в выражениях.

Типы данных: "str" – строки символов, "int" – целые числа, "real" – числа с плавающей запятой. Заметим, что все величины хранятся как строки символов. Это значит, что все величины преобразовываются в соответствующее текстовое представление, а все операции производятся над элементами текста. Целые числа и вещественные величины должны быть преобразованы в текстовые представления, а булевые выражения – в символьные строки "да", "нет".

Объектные "тэги"

<text> - определяет порцию текстов, которая должна соответствовать исходному тексту. Может определять также порцию текста, которая должна преобразовываться в выходной текст.

<var> - специфицирует переменные с замещением и без него.

<list> - специфицирует наборы величин.

"Тэги" процесса СА специфицируют:

<multiple> - блок текста, который может включать исходный код несколько раз;

<optional> - блок текста, в котором исходный код может быть опущен;

<case> and <or> - блоки, один из которых должен содержать исходный код;

<if>, <else> - "теги" описания условий;

<notordered>, <item> - блоки, в которых последовательность элементов не имеет значения.

Фрагмент описания шаблона СА СЦ приведен на рис.4.

```
//Scan Chains
 <text src="scanChain"/>
 <text dest="  &lt;ScanChains&gt;\n"/>
 <multiple>
  <case>
   <or>
    <text src="ch_"/>
    <var name="chain_name" stop="["/>
    <text src="["/>
    <var name="signal_name" stop=","/>
    <var name="irev" stop=";"/>
    <text src=";"/>
    <text dest="    &lt;chain name=&quot;@chain_name@&quot;&gt;@signal_name@&lt;/chain&gt;\n"/>
   </or>
   <or>
    <text src="#"/>
    <var name="irev" stop="#"/>
    <text src="#"/>
   </or>
  </case>
 </multiple>
 <text src ="end" dest="  &lt;/ScanChains&gt;\n"/>
```

Рис. 4. Шаблон для СА СЦ

Поскольку шаблоны описаны на одном и том же языке (TCXL), как это показано в [5], и необходимы для решения указанной выше проблемы, их верификация может быть произведена автоматически [6].

СПИСОК ЛИТЕРАТУРЫ

1. **Douglas Kay, Samiha Mourad**. Compression Techniques for Interactive BIST Application //Proceedings of 19-th VLSI Test Symposium (VTS), April-May, 2001. - P. 9-15.
2. **Karim Arabi.** Logic BIST and Scan Test Techniques for Multiple Identical Blocks //Proceedings of 20-th VLSI Test Symposium (VTS), April-May, 2002. - P. 60-65.
3. Supplement to IEEE Std. 1149.1-1990. IEEE Standard Test Access Port and Boundary-Scan Architecture //IEEE Standards Electronic Library.
4. **Rohit Kapur.** CTL for Test Information of Digital ICs /Kluwer Academic Publishers. - 2003.
5. **Baghdasaryan S., Shoukourian S.** An Approach for CTL Implementation //Digest of Papers of IEEE Workshop on Testing Embedded Core-Based Systems (TECS), Hyatt Regency Hotel Monterey, CA, May 2002. - P. 43-49.
6. **Atoyan T., Bagdasaryan S.** Verification of templates in CTL implementation //Proceedings of Computer Science and Information Technologies (CSIT), Conference, Armenia Sept,. 2003. - P. 395-399.
7. **Atoyan T.** Using a Template Processing System for Easy CTL Coding and Debugging. A Case of Study //Digest of Papers of IEEE Test Resource Partitioning Workshop held in conjunction with VLSI Test Symposium (VTS) Conference, Napa, California April, 28-29. – 2004.

ИПИА НАН РА. Материал поступил в редакцию 04.01.2005.

Տ.Ռ.ԱԹՈՅԱՆ

ՉԻՊԵՐԻ ՎՐԱ ՀԻՄՆՎԱԾ ՀԱՄԱԿԱՐԳԵՐԻ ՍԿԱՆԱՎՈՐՈՂ ՇՂԹԱՆԵՐԻ ԳԵՆԵՐԱՑԻԱ ԵՎ ԻՆՏԵՐՊՐԵՏԱՑԻԱ՝ ՇԱԲԼՈՆՆԵՐԻ ՄՇԱԿՄԱՆ ՀԱՄԱԿԱՐԳԵՐԻ ՕԳՆՈՒԹՅԱՄԲ

Առաջարկվում է տեստային միջավայրի ստեղծման՝ շաբլոնների մշակման տեխնիկայի վրա հիմնված մեթոդոլոգիա՝ CTL լեզվով նրա ներկայացման և շաբլոնների մշակման հատուկ համակարգի (TPS) միջոցով: Դիտարկված են չիպերի վրա հիմնված համակարգի (SoC) սկանավորող շղթաների կոնտենտի գեներացիայի և ինտերպրետացիայի ավտոմատացման միջոցները՝ նրանց տեստավորման և կարգաբերման համար՝ օգտագործողի դասակարգմանը համապատասխան:

T. R. ATOYAN

SoC SCAN CHAIN GENERATION AND INTERPRETATION VIA TPS

A methodology of a testing environment design based on template processing technique was proposed for implementation of CTL and a special Template Processing System based on the methodology was developed. The current paper focuses on an automatic generation and interpretation of custom SoC debug scan chains via TPS. A special set of templates, an extension to TPS, is built to facilitate the solution of the problem.