XXXI Неделя науки СПбГПУ. Материалы межвузовской научной конференции. Ч. VI: С.31-32, 2003.

УДК 658.512.011.56: 681.3.06

В.В. Ильин (асп., каф. ИУС), А.Н. Волков (асп., каф. ИУС),
М.А. Никитин (4 курс, каф. ИУС), В.П. Котляров, к.т.н., проф.

ФОРМАЛЬНЫЕ И ПОЛУФОРМАЛЬНЫЕ ПОДХОДЫ К ВЕРИФИКАЦИИ МОДЕЛЕЙ ПОЛУПРОВОДНИКОВЫХ УСТРОЙСТВ

Задача верификации моделей полупроводниковых устройств состоит в том, чтобы удостовериться, что разрабатываемая модель соответствует спецификации. В настоящее время традиционные подходы, основанные на динамической симуляции, зачастую, оказываются недостаточно эффективны. Существуют два пути повышения эффективности – использование автоматической генерации верификационных сценариев и применение методов формальной верификации. Как и динамическая, формальная верификация применяется, начиная с ранних стадий разработки устройств, чтобы как можно больше дефектов было найдено до перехода на поздние стадии (как то верификация модели уровня логических элементов – “gate-level”, или верификация готового полупроводникового устройства – “post-silicon”), на которых исправление дефектов становится весьма дорогостоящим.

Формальные подходы к верификации предполагают использование математических методов. Статическая формальная верификация (Static Formal Verification - SFV), в отличие от динамических подходов, связанных с симуляцией, не нуждается в симуляции и поэтому не требует создания тестового окружения (testbench). Вместо этого используется особый инструмент (formal checker), как правило, специализированный для проверки моделей определенного уровня (чаще всего, это RTL или gate-level) и языка моделирования (Verilog, VHDL и др.).

Данный инструмент производит перебор всех возможных комбинаций входных значений для проверки всех состояний модели, в которые она может переходить. При этом в каждом состоянии проверяется, что не нарушаются некоторые свойства модели (данный процесс называется property checking или model checking). Как правило, инструмент имеет набор встроенных проверок, характерных для моделей данного уровня (таких как переполнение очередей, одновременная установка сигнала из нескольких источников и т.д.), а также предоставляет возможность определения пользовательских проверок и входных ограничений для модели. Вызов такой проверки именуется термином assertion.

Инструменты разных компаний используют различные способы для описания assertion. В настоящее время существует три направления:

- библиотека assertion-конструкций на существующем языке моделирования;
- специализированный язык описания assertion-конструкций;
- язык моделирования, поддерживающий assertion-конструкции;

Примером первого направления является OVL (Open Verification Library), второго -

FPL (Formal Property Language), за основу которого взят язык Sugar компании IBM. В качестве третьего направления разрабатывается SystemVerilog – расширение языка Verilog.

Предполагается, что через два-три года FPL станет общепринятым. До тех пор рекомендуется пользоваться OVL, которая содержит в себе набор модулей-мониторов (assertion monitors). OVL существует в двух вариантах (Verilog и VHDL) и является свободно распространяемым продуктом. Хотя OVL предоставляет менее гибкие возможности, чем планируемые в FPL, она имеет преимущество в том, что является общедоступной в настоящий момент и может быть непосредственно использована и в динамической симуляции. В то время как FPL для этой цели будет требовать предварительной стадии генерации assertion-проверок в код на целевом языке моделирования. Наиболее легко будет осуществить такую генерацию в язык SystemVerilog, так как он поддерживает assertion-конструкции на уровне языка, для Verilog или VHDL необходимо будет писать дополнительные конструкции реализации assertion.

Основной плюс применения формализованных assertion по сравнению с обычными модулями-мониторами, применяемыми в симуляционных подходах, в том, что один и тот же синтаксис воспринимается как инструментами, предназначенными для формальной проверки, так и языками моделирования. Таким образом, assertion используются как для формального анализа, так и для динамических проверок в процессе симуляции.

Поскольку инструменты для формальной проверки производят полный перебор состояний модели, они эффективно применимы только к модулям небольшого размера, то есть дизайн системы нужно разбивать на множество более мелких частей и описывать множество допустимых значений их на границах. Построение множества допустимых значений также является довольно сложной задачей, так, например, не включение в него некоторых допустимых значений приведет к неполному перебору, а включение запрещенные значений приведет к нахождению “ложных” ошибок. В том случае, если множество допустимых значений составлено правильно, положительный вердикт, вынесенный на основе формальной проверки, означает отсутствие ошибок в верифицируемом элементе.

Ограниченная применимость формальных подходов для проверки больших модулей привела к появлению полуформальных методов верификации. Полуформальный подход (semiformal или dynamic formal verification) основан на совмещении динамического и формального методов.

Модель подвергается динамической верификации, в процессе которой собирается и сохраняется информация о состоянии модели. Затем для состояний, достигнутых в процессе динамической верификации, выполняется локальная формальная верификация, то есть верификация на ограниченном наборе состояний, ближайших к тем, которые были достигнуты в процессе симуляции. Таким образом, перебираются только доступные состояния модели, ограничения на объем значительно менее жесткие, но отсутствие найденных ошибок, не дает гарантии, что их нет совсем. Эффективность полуформальных методов также зависит от количества и полноты тестов и качества тестового окружения.