Верификация проектов и построение тестов контроля СБИС на уровне RTL

Золоторевич Л.А.
Факультет прикладной математики и информатики, каф. КТС
Белорусский государственный университет
г. Минск, Беларусь
e-mail: zolotorevichla@bsu.by

***Аннотация* — Рассматриваются особенности верификации проектов и построения тестов контроля цифровых устройств и систем, описанных на языке VHDL. Предлагается метод верификации проектов и направленного построения тестов контроля сверхбольших сложно- функциональных интегральных схем, представленных на уровне межрегистровых передач на языке VHDL. Метод основан на описании операций объекта функциями разрешения и сведении задачи к решению КНФ – выполнимости полученной системы булевых функций. Решается также задача верификации проектов на уровне RTL путем моделирования системы на заданном тесте.**

Ключевые слова: цифровая СБИС; построение теста; уровень межрегистровых передач; верификация проекта, КНФ - выполнимость

I. Введение

Быстрое развитие информационных технологий, наблюдаемое в настоящее время во всех сферах человеческой деятельности, стало возможным вследствие создания технической базы и, в первую очередь, развития микроэлектроники. Появление сверхбольших интегральных схем с миллиардом транзисторов или сотнями миллионов логических элементов на кристалле, безусловно, является успехом, в первую очередь, в области производства объектов микроэлектроники. Открытый в 1965 году закон Мура относительно увеличения степени интеграции интегральных схем в полтора раза за два года при незначительном увеличении стоимости претерпевает изменения в плане существенного возрастания стоимости и производства, и проектирования современных сложно-функциональных СБИС. Высокая стоимость современных фабрик по производству СБИС позволяет только немногим странам создавать собственное производство изделий в субмикронном диапазоне. Вместе с тем, у многих стран, обладающих научно-техническим потенциалом (Индия, Китай, Россия и др.) появилась возможность включиться в решение сложных наукоемких задач проектирования сложно-функциональных цифровых систем. Изменившиеся подходы к проектированию, появление языков и методологий проектирования привело к тому, что объект, спроектированный и описанный на основе стандартного подхода, включающий средства контроля, является товаром, и его изготовление может быть размещено на немногочисленных фабриках производства, обеспечение загруженности которых также является актуальной задачей.

В республике Беларусь в настоящее время в области развития информационных технологий большое внимание уделяется изучению и развитию Web – технологий, компьютерной графики, автоматизации процессов управления и обслуживания населения в разных отраслях. Эта тенденция заметна по открываемым специализациям в вузах, по спектру задач, выполняемых по заказам зарубежных инвесторов. Имеющаяся техническая база по производству интегральных схем недостаточно обеспечивается квалифицированными кадрами и научными исследованиями, что связано с невысокой оплатой труда в отрасли, спецификой заказов, поступающих из-за рубежа, не приводящих к крупным внешним инвестициям и не способствующих развитию научно-технического потенциала в области проектирования объектов современной микроэлектроники.

В то же время следует отметить, что успехи в областях микроэлектроники и электроники, которые привели к созданию таких уникальных цифровых систем как современные персональные компьютеры, ноутбуки, смартфоны, навигационные системы, системы безопасности автомобилей, принципиально изменившие условия созидания и быт человека, в определяющей степени обусловлены применением современных систем автоматизированного проектирования (САПР). История развития процессов автоматизации проектирования в электронике началась в начале 60-х годов после создания в 1958 г. интегральных схем [1]. Первая конференция по автоматизации проектирования была проведена в 1964 г., но реальные практические результаты были получены только в средине 70- годов фирмами Bell Labs, Hewlett Packard, IBM, Intel, Tektronix. В настоящее время лидирующие позиции в разработке САПР микроэлектроники занимают фирмы Cadence, Mentor Graphics, Xilinx и др., поставляющие на рынок средства проектирования, начиная с системного уровня и до получения топологического проекта. Все фирменные САПР характеризуются высокой стоимостью, сложностью их адаптации и эффективного применения. Особенностью проектирования сложно-функциональных цифровых систем является то, что ряд задач в данной области до настоящего времени не имеют эффективного теоретического, ни практического решения. Это объясняется большой размерностью и функциональной сложностью проектируемых систем, сложностью задач верификации проектов на разных этапах проектирования, отсутствием эффективных методов построения тестов, постоянным уточнением и изменением постановок указанных задач.

Процесс проектирования цифровых систем представляет собой многоуровневый итерационный процесс «синтез-анализ» (рис. 1). Средствами проектирования необходимо обеспечить быструю и стабильную сходимость процесса и низкую вероятность возврата на более высокий уровень в процессе проектирования функционально-сложного объекта. К настоящему времени достаточно разработаны методы проектирования и верификации проектов на уровнях топологического и функционально - логического представлений объекта, методы построения тестов контроля комбинационных схем. Последнее десятилетие усилия направлены на повышение эффективности решения задач проектирования на системном уровне и на уровне межрегистровых передач и, в первую очередь, на

генерацию тестов на начальных этапах проектирования.

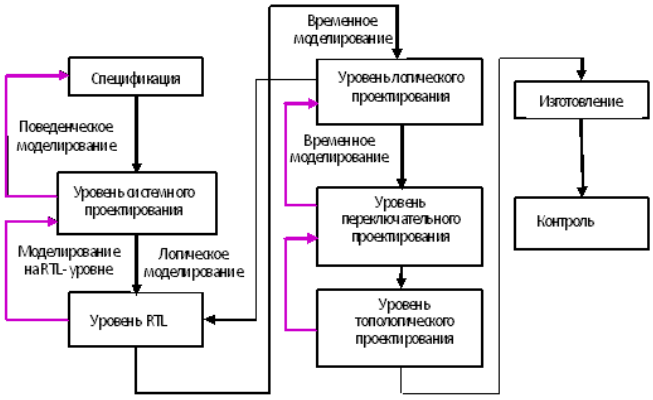

Рис. 1. Уровни проектирования и методы моделирования СБИС

II. Состояние проблем верификации и построения тестов

Проблемы верификации проектов и построения тестов контроля сложно-функциональных электронных систем остаются наиболее наукоемкими во всем спектре проблем проектирования современных СБИС. Современные интегральные схемы имеют порядка миллиарда транзисторов на кристалле, и разработка тестов для объектов такого размера на уровне структурного представления оказалась практически неразрешимой задачей. В то же время, острая потребность в тестах имеет место с самого начала проектирования, тесты необходимы также на этапе производства для отбраковки готовых изделий и при эксплуатации для оценки работоспособности. Сложность задач контроля подтверждает тот факт, что в соответствии с Международными технологическими стандартами количество диагностов при проектировании сложно - функциональных объектов превосходит количество разработчиков в 2-3 раза, что свидетельствует об актуальности задачи разработки формальных методов построения тестов.

Подойти к эффективному решению задачи построения тестов можно на основе идентификации объекта на начальных этапах проектирования, когда имеется некоторое поведенческое описание или описание объекта на уровне межрегистровых передач, которое содержит существенно меньшее число базовых примитивов, чем на структурном уровне. В то же время, известные в литературе методы построения тестов ориентированы на структурные модели объекта и оказались не пригодными для работы в условиях отсутствия структуры объекта, при переходе на высокоуровневые модели. Разработка тестов контроля на системном уровне или на уровне RTL позволяет не только снизить размерность задачи построения тестов, но и более эффективно решать задачи верификации проектов, начиная с самых ранних этапов проектирования, позволяет оценить степень контролепригодности проекта, сравнивать проекты, реализуемые в разных технологических библиотеках проектирования. Попытка переноса задачи генерации тестов контроля с функционально-логического уровня на начальные этапы проектирования возникла из естественного желания эффективнее решать сложные задачи в процессе итерационного проектирования и сократить его сроки.

Вопросы разработки тестов контроля СБИС на системном уровне проектирования и уровне RTL, когда отсутствуют сведения относительно структурной реализации объекта, рассматриваются в работах [2-9] . Предлагаются методы построения тестов, основанные на внесении в описание объекта функциональных неисправностей и моделировании на псевдослучайных входных последовательностях. При этом рассматривается достаточно широкий спектр предлагаемых моделей неисправностей. Это и константные неисправности сигналов и переменных, вносимые в программный код описания объекта, ошибки программного кода (замена условных переходов безусловными и др.). В работе [10] предлагается методика построения функциональных неисправностей, аргументировано соответствующих неисправностям структурной реализации соответствующего механизма. Известны работы, рассматривающие задачи направленного построения тестов на верхних уровнях проектирования [2, 3, 5]. Достоинством предлагаемых решений является то, что описание объекта в этом случае содержит существенно меньшее число примитивов и снижает соответственно размерность задачи генерации тестов. В то же время требуется разработка новых подходов к построению моделей объектов и методов построения тестов, методов описания неисправностей на функциональном уровне.

Следует заметить, что быстро развивающаяся электронная отрасль выставляет все новые требования и условия к задаче построения тестов. Разработка таких объектов как «системы на кристалле» требует развития методов и средств внутреннего самотестирования, эффективных методов построения тестов в разных системах идентификации, методов минимизации тестов, совершенствования методов временного моделирования, статического временного анализа и др..

В работе [2] приведен общий подход к иерархической генерации тестов СБИС на RTL-уровне. Каждая операция программного кода реализуется на аппаратном уровне некоторым набором аппаратных средств, тест контроля для которых строится известными методами и средствами на основе структурного представления устройства. В литературе известны некоторые модели функциональных неисправностей, которые предлагается использовать при иерархическом построении тестов. Тест вносится в описание объекта, устанавливаются ограничения на функционирование объекта, задача построения теста контроля всего объекта сводится к решению системы арифметических уравнений с внесенными ограничениями. В настоящей работе описывается механизм решения задачи направленного построения тестов, основанный на построении системы арифметических уравнений и итерационном решении задачи КНФ-выполнимости соответствующей системы булевых функций разрешения.

Проблема построения тестов непосредственно связана с проблемой верификации проектов сложно-функциональных цифровых систем. На сегодняшний день проблема верификации проектов на разных этапах проектирования решается, в основном, на основе

моделирования объекта, т. к. применяемые методы формальной верификации ориентированы на решение некоторых частных задач. Полнота верификации проектов обеспечивается полнотой применяемых при верификации тестов. Предлагаемый в данной статье метод направлен как на решение задачи направленного построения тестов контроля, так и на верификацию проекта путем моделирования на заданном тесте.

III. Математическая платформа

В указанной выше постановке задачи структура объекта отсутствует. Имеется описание поведения системы на уровне RTL, который является промежуточным между системным уровнем и уровнем структурного представления. С учетом сложившейся традиции цифровая система рассматривается на уровне RTL как две подсистемы – операционная, выполняющая преобразование данных в соответствии с заданными алгоритмами, и управляющая, реализующая управление операционной частью. Поэтому в качестве математической платформы для описания цифровой системы будем использовать описание в виде графов потоков данных и потока управления (в работе [4] DD потока данных и DD потока управления). Граф потока управления – это ориентированный граф с узлами, соответствующими операторам программного кода, и ребрами, указывающими порядок выполнения операторов. Граф потока данных - это ориентированный граф с узлами, соответствующими выполняемым операциям и ребрами, указывающими последовательность операций по преобразованию некоторой переменной или сигнала.

В теории тестового диагностирования известны методы направленного и случайного построения тестов. Если рассматривать возможность их реализации в указанной выше постановке, то следует отметить весьма существенную особенность. Если методы построения теста случайным образом можно пытаться реализовать на основе имеющихся фирменных компиляторов языка VHDL, то для направленного построения тестов необходимо знание внутреннего представления программного кода для построения структур графов, которое, к сожалению, закрыто для внешнего использования.

В работе [4] приведена структура представления графов. Построение графов осуществляется при статическом анализе программного кода, который выполняется при его синтаксическом анализе. При этом строится один граф потока управления и столько графов потока данных, сколько переменных описывают полное состояние моделируемой системы.

Ответственным моментом при построении тестов для контроля объекта, описанного на языке высокого уровня, является выбор моделей неисправностей объекта, рассматриваемых на уровне операторов программного кода. В настоящее время в литературе отсутствует доказательная база для выбора некоторой определенной модели используемой неисправности, что требует обращения к более широкому диапазону известных в литературе моделей. В данной работе кроме таких известных моделей, как модель выпавшего оператора, замены условного перехода безусловным или одной операции некоторой другой, неисправности константного типа переменной, сигнала, определенного разряда переменной или сигнала также будет использоваться и модель функциональной неисправности, которая аргументировано соответствует неисправности константного типа реального объекта, которая предложена в работе [10].

Научная гипотеза, используемая при разработке метода направленного построения тестов на RTL-уровне, формулируется так: Для того, чтобы найти входные данные, которые позволят определить по выходным данным наличие или отсутствие некоторой неисправности в системе, представленной в рамках любой системы идентификации, необходимо активизировать неисправность, то есть заставить ее проявиться на выходе некоторого элемента системы, затем заставить ее проявиться хотя бы на одном из выходов системы, после чего необходимо вычислить все входные данные, которые позволят сохранить условия, полученные на предыдущих этапах решения данной задачи. Данная гипотеза сформулирована на основе идеи Рота, реализованной при построении тестов контроля объекта на структурном уровне.

IV. Построение и решение системы арифметических уравнений

Общая идея метода направленного построения тестов контроля цифровых систем, описанных на уровне RTL на языке VHDL заключается

- в переходе от системы арифметических и логических уравнений, описывающих поведение объекта, к построению системы КНФ булевых функций разрешения;
- конъюнктивном объединении функций разрешения;
- решении задачи выполнимости результирующей КНФ разрешения объекта.

Предположим, что все входные переменные являются целочисленными размерностью n бит (mod 2^n).

Для генерации тестов необходимо:

1) На основе программного кода объекта составить систему арифметических уравнений, описывающих функционирование объекта;

2) Выполнить корректировку системы с учетом внесения неисправностей соответствующего оператора;

3) Поставить в соответствие каждой целочисленной переменной размерностью n бит (mod 2^n) логический вектор длины n;

4) Итеративно выполнить решение системы. Вначале необходимо получить 1-й бит результата. Для этого арифметические выражения транслируются в КНФ булевых функций разрешения;

5) Все полученные КНФ-функций разрешения объединяются по правилу И. Решается задача КНФ-выполнимости полученной системы булевых функций.

Если система выполнима, то продолжается рекурсивное вычисление следующего бита. В противном случае, тест не может быть построен, так как внесенные ограничения не могут быть удовлетворены. В таком случае для проверки рассматриваемой неисправности необходимо изменить систему управления с целью повышения управляемости и наблюдаемости объекта проекта.

Пример: Рассмотрим фрагмент некоторого программного кода, приведенный на рисунке 2. Здесь

A, B, C, S – входные данные, L – переменная выхода. Положим, что переменные A, B, C являются целочисленными по модулю 2^n, а S – однобитовая переменная. На рисунке 3 приведена система арифметических уравнений, описывающих функционирование объекта, представленного программным кодом, приведенным на рисунке 2. На рис. 4 приведена ярусно-параллельная форма рассматриваемого программного кода. В объекте имеется мультиплексор, два сумматора, умножитель и схема сравнения. Построим тест, проверяющий правильность выполнения оператора целочисленного сложения

G = B + C.

```
if   S = '0' then
        D <= A;
     else  D <= B;
        G <= B + C;
        E <= D + C;
        F <= E * G;
        L <= D < G;
endif;
```

Рис. 2. Фрагмент программного кода

```
D = /S*A + S*B;
G = B + C;
E = D + C;
    F = E * G;
L = D < G;
```

Рис. 3. Система арифметических уравнений (mod 2^n)

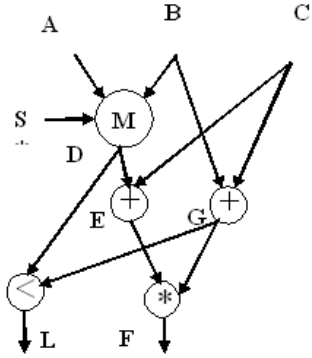

Рис. 4. Ярусно-параллельная форма представления кода

С данным оператором связаны аппаратные средства, обеспечивающие сложение целых чисел. Положим, что нам известен тест (последовательность входных наборов) для контроля сумматора по модулю 2^n, и нами выбран один набор теста, который задает B = 15, C = 1, G' = 17 по модулю 2^n где G' – неисправное значение переменной G. Для построения теста контроля рассматриваемого объекта система уравнений (рис. 3.) должна быть скорректирована, как показано на рис. 5, чтобы обеспечить распространение неисправности к выходам объекта и определение входных переменных. Здесь уравнения 1-5 описывают исправный исходный объект, операторы 6-8 описывают процесс внесения неисправности, операторы 9-13 задают ограничения, обеспечивающие распространение эффекта неисправности к выходу объекта.

Идея решения подобных систем арифметических уравнений основана на поразрядном подходе к вычислению значений, и состоит в том, что каждой целочисленной переменной m <= 2^n ставится в соответствие определенный логический вектор

№ п/п	Арифметические уравнения	№ п/п	Арифметические уравнения
1	D = /S*A + S*B; (mod 2^n)	9	G'≠ G; (mod 2^n)
2	**G = B + C;** (mod 2^n)	10	F' = E * G'; (mod 2^n)
3	E = D + C; (mod 2^n)	11	F ≠ F' (mod 2^n)
4	F = E * G; (mod 2^n)	12	L' = D < G'; (mod 2^n)
5	L = D < G; (mod 2^n)	13	L ≠ L' ; (mod 2^n)
6	B = 15; (mod 2^n)		
7	C = 1; (mod 2^n)		
8	G' = 17; (mod 2^n)		

Рис. 5. Система арифметических уравнений (mod 2^n) с внесенной неисправностью

размерности n. Для описания алгоритма вычисления системы арифметических уравнений применяется логическая функция разрешения, которая задает соотношения между исправными логическими состояниями выводов физических элементов, реализующих определенную логическую функцию.

V. Построение функции разрешения

Введем в рассмотрение функцию F^f, называемую функцией разрешения для логической функции f [11]. Функция F^f зависит не только от аргументов функции f, но и от самой f и принимает значение логической 1 при всех допустимых состояниях входных и выходной переменных. Функция F^f принимает значение 0 при всех недопустимых состояниях входных и выходной переменных. Рассмотрим получение функции разрешения F^f для функции конъюнкции $f = a*b$ (табл. 1).

Получим СКНФ функции F^f по таблице 1, выбирая конституенты 0. Для этого

1) выбираются наборы аргументов, на которых функция обращается в нуль;

2) выписываются дизъюнкции, соответствующие этим наборам, причем если аргумент x_i входит в набор как нуль, то в дизъюнкцию он вписывается без изменения. Если же аргумент x_i входит в данный набор как единица,

Табл.1. Функции разрешения и запрета

a	*b*	*f*	F^f	$\overline{F^f}$
0	**0**	**0**	1	0
0	**1**	**0**	1	0
1	**0**	**0**	1	0
1	**1**	**1**	1	0
0	0	1	0	1
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1

то в соответствующую дизъюнкцию вписывается его отрицание;

3) все выписанные дизъюнкции соединяют знаком конъюнкции:

$$F^f = (a \vee b \vee \overline{f})(a \vee \overline{b} \vee \overline{f})(\overline{a} \vee b \vee \overline{f})(\overline{a} \vee \overline{b} \vee f) =$$

$$F^f = (a \vee \overline{f})(b \vee \overline{f})(\overline{a} \vee \overline{b} \vee f).$$

Заметим, что функция $\overline{F^f}$, называемая функцией запрета, принимает значения, инверсные функции разрешения F^f.

Выражение функции разрешения для операции поразрядного сложения $f = a \oplus b$ выглядит так: $(a \vee b \vee \overline{f})(a \vee \overline{b} \vee f)(\overline{a} \vee b \vee f)(\overline{a} \vee \overline{b} \vee \overline{f})$.

В таблице 2 приведены однобитовые операции и соответствующие функции разрешения в виде КНФ.

Табл. 2. Функции разрешения

Однобитовые арифметические и логические уравнения	КНФ функций разрешения
$f = b \vee c$	$(\overline{b} \vee f)(\overline{c} \vee f)(b \vee c \vee \overline{f})$
$f = b * c$	$(b \vee \overline{f})(c \vee \overline{f})(\overline{b} \vee \overline{c} \vee f)$
$f = a \oplus b$	$(a \vee b \vee \overline{f})(a \vee \overline{b} \vee f)(\overline{a} \vee b \vee f)(\overline{a} \vee \overline{b} \vee \overline{f})$
$f = a \sim b$	$(a \vee b \vee f)(a \vee \overline{b} \vee \overline{f})(\overline{a} \vee b \vee \overline{f})(\overline{a} \vee \overline{b} \vee f)$
$f = \overline{s} * a + s * b$	$(s \vee a \vee \overline{f})(a \vee b \vee \overline{f})(\overline{s} \vee b \vee \overline{f}) * (s \vee \overline{a} \vee f)(\overline{s} \vee \overline{b} \vee f)(\overline{a} \vee \overline{b} \vee f)$
$f = a \geq b$	$(a \vee f)(\overline{a} \vee \overline{f})$
$f = b < c$	$(\overline{b} \vee \overline{f})(c \vee \overline{f})(b \vee \overline{c} \vee f)$
$f = 1$	f
$f = 0$	$\overline{f}$
$f \neq f'$	$(f \vee f' \vee \overline{f''})(f \vee \overline{f'} \vee f'')(\overline{f} \vee f' \vee f'') * (\overline{f} \vee \overline{f'} \vee \overline{f''})$

VI. Решение системы уравнений

Чтобы одновременно решить равенства по модулю 2 мы объединяем все КНФ разрешения логических функций вместе, используя логический элемент И. Чтобы найти второй бит решения этого уравнения, мы должны найти рекурсивные уравнения по модулю 2 аналогично, используя результаты предыдущей итерации.

Решение системы уравнений, приведенной на рис. 5, выполняется итеративно. Для вычисления очередного бита результата формируется система функций разрешения, соответствующих каждой арифметической функции, затем решается задача выполнимости конъюнкции всех функций разрешения.

Значение каждого бита результата вычисляется рекурсивно. К примеру, для уравнения E = D + C вначале вычисляется значение младшего бита результата $E_0 = D_0 + C_0$. После вычисления E_0 его результат используется при формировании результата для следующих битов. Для определения порядка вычисления битов высшего порядка из битов более низкого порядка необходимо рассмотреть различные формы уравнений. Для операции суммирования, начиная со 2-й итерации, очередной бит результата вычисляется следующим образом: $E_i = D_i + C_i + P_i$, где P_i - значение переноса из i+1 –го разряда.

На рис. 6 приведены функции разрешения для первого бита системы арифметических уравнений, приведенных на рис. 5.

№ п/п	Арифметические уравнения	Эквивалентные функции разрешения
1	$D_0 = /S_0 * A_0 + S_0 * B_0 \pmod 2$	$(S_0 \vee A_0 \vee \overline{D_0})(A_0 \vee B_0 \vee \overline{D_0})(\overline{S_0} \vee B_0 \vee \overline{D_0}) * (S_0 \vee \overline{A_0} \vee D_0)(\overline{S_0} \vee \overline{B_0} \vee D_0)(\overline{A_0} \vee \overline{B_0} \vee D_0)$
2	$G_0 = B_0 + C_0 \pmod 2$	$(C_0 \vee B_0 \vee \overline{G_0})(C_0 \vee \overline{B_0} \vee G_0)(\overline{C_0} \vee B_0 \vee G_0) * (\overline{C_0} \vee \overline{B_0} \vee \overline{G_0})$
3	$E_0 = D_0 + C_0 \pmod 2$	$(C_0 \vee D_0 \vee \overline{E_0})(C_0 \vee \overline{D_0} \vee E_0)(\overline{C_0} \vee D_0 \vee E_0) * (\overline{C_0} \vee \overline{D_0} \vee \overline{E_0})$
4	$F_0 = E_0 * G_0 \pmod 2$	$(E_0 \vee \overline{F_0})(G_0 \vee \overline{F_0})(\overline{E_0} \vee \overline{G_0} \vee F_0)$
5	$L_0 = D_0 < G_0 \pmod 2$	$(\overline{B_0} \vee \overline{L_0})(G_0 \vee \overline{L_0})(B_0 \vee \overline{G_0} \vee L_0)$
6	$B_0 = 1 \pmod 2$	B_0
7	$C_0 = 1 \pmod 2$	C_0
8	$G'_0 = 1 \pmod 2$	G'_0
9	$G'_0 \neq G \pmod 2$	$(G_0 \vee G'_0 \vee \overline{f''_0})(G_0 \vee \overline{G'_0} \vee f''_0)(\overline{G_0} \vee G'_0 \vee f''_0) * (\overline{G_0} \vee \overline{G'_0} \vee \overline{f''_0})$
10	$F'_0 = E_0 * G'_0 \pmod 2$	$(E_0 \vee \overline{F'_0})(G_0 \vee \overline{F'_0})(\overline{E_0} \vee \overline{G_0} \vee F'_0)$
11	$F_0 \neq F'_0 \pmod 2$	$(F_0 \vee F'_0 \vee \overline{f''_0})(F_0 \vee \overline{F'_0} \vee f''_0)(\overline{F_0} \vee F'_0 \vee f''_0) * (\overline{F_0} \vee \overline{F'_0} \vee \overline{f''_0})$
12	$L'_0 = D_0 < G'_0 \pmod 2$	$(\overline{D_0} \vee \overline{L'_0})(G_0 \vee \overline{L'_0})(D_0 \vee \overline{G_0} \vee L'_0)$
13	$L_0 \neq L'_0 \pmod 2$	$(L_0 \vee L'_0 \vee \overline{f''_0})(L_0 \vee \overline{L'_0} \vee f''_0)(\overline{L_0} \vee L'_0 \vee f''_0) * (\overline{L_0} \vee \overline{L'_0} \vee \overline{f''_0})$

Рис. 6. Функции разрешения для вычисления 1-го бита результата

Заметим, что для рекурсивного вычисления неравенства необходимо учитывать, что неравенство может быть определено только в старшем i-м бите (i от 0 до n-1).

Все функции разрешения объединяются знаком конъюнкции, решается задача выполнимости полученной системы булевых функций:

$$((S_0 \vee A_0 \vee \overline{D_0})(A_0 \vee B_0 \vee \overline{D_0})(\overline{S_0} \vee B_0 \vee \overline{D_0}) * (S_0 \vee \overline{A_0} \vee D_0)(\overline{S_0} \vee \overline{B_0} \vee D_0)(\overline{A_0} \vee \overline{B_0} \vee D_0)) *$$
$$*((C_0 \vee B_0 \vee \overline{G_0})(C_0 \vee \overline{B_0} \vee G_0)(\overline{C_0} \vee B_0 \vee G_0) * (\overline{C_0} \vee \overline{B_0} \vee \overline{G_0})) *$$
$$*((C_0 \vee D_0 \vee \overline{E_0})(C_0 \vee \overline{D_0} \vee E_0)(\overline{C_0} \vee D_0 \vee E_0) * (\overline{C_0} \vee \overline{D_0} \vee \overline{E_0})) *$$
$$((E_0 \vee \overline{F_0})(G_0 \vee \overline{F_0})(\overline{E_0} \vee \overline{G_0} \vee F_0)) * ((\overline{B_0} \vee \overline{L_0})(G_0 \vee \overline{L_0})(B_0 \vee \overline{G_0} \vee L_0)) * B_0 * C_0 * G'_0 *$$
$$*((G_0 \vee G'_0 \vee \overline{f''_0})(G_0 \vee \overline{G'_0} \vee f''_0)(\overline{G_0} \vee G'_0 \vee f''_0)(\overline{G_0} \vee \overline{G'_0} \vee \overline{f''_0})) *$$
$$*((E_0 \vee \overline{F'_0})(G_0 \vee \overline{F'_0})(\overline{E_0} \vee \overline{G_0} \vee F'_0)) *$$
$$*((F_0 \vee F'_0 \vee \overline{f''_0})(F_0 \vee \overline{F'_0} \vee f''_0)(\overline{F_0} \vee F'_0 \vee f''_0)(\overline{F_0} \vee \overline{F'_0} \vee \overline{f''_0})) *$$
$$*((\overline{D_0} \vee \overline{L'_0})(G_0 \vee \overline{L'_0})(D_0 \vee \overline{G_0} \vee L'_0)) *$$
$$*((L_0 \vee L'_0 \vee \overline{f''_0})(L_0 \vee \overline{L'_0} \vee f''_0)(\overline{L_0} \vee L'_0 \vee f''_0)(\overline{L_0} \vee \overline{L'_0} \vee \overline{f''_0})) = 1.$$

VII. Верификация проектов на RTL – уровне

Верификация проектов, как известно, занимает большую часть времени проектирования сложно-функциональной СБИС. На сегодняшний день основными методами практической верификации на всех этапах проектирования является моделирование. Такой подход, основанный на моделировании, требует наличия тестов. Для моделирования объекта, описанного на языке VHDL на уровне RTL, предлагается метод, который заключается в следующем:

1. Представить программный код в виде системы арифметических уравнений;
2. Добавить в полученную систему уравнений означивание входных и выходных переменных;
3. Итеративно решить систему. Для этого перейти от системы арифметических уравнений к

системе логических функций разрешения, учитывая особенность соответствующей итерации;

4. Получить КНФ-функцию разрешения и вычислить ее выполнимость. По результатам перейти к новому экземпляру теста или закончить моделирование и перейти к анализу ошибки.

Рассмотрим моделирование объекта, описанного программным кодом, приведенным на рис. 2. Предположим, нам известен тест, одним из шаблонов которого является A = 34, B = 45, C = 7, L = 1, F = 2132. Система арифметических уравнений дополняется тестовыми значениями входных и выходных переменных и переводится в систему функций разрешения для проведения первого этапа вычислений, затем формируется КНФ – разрешения конъюнктивным объединением полученных функций и решается задача выполнимости. На рис. 7 приведена система арифметических уравнений и система логических функций разрешения для выполнения первой итерации вычислений.

VIII. ЗАКЛЮЧЕНИЕ

В настоящее время становится очевидным, что проектирование объекта на уровне RTL постепенно становится все менее жизнеспособным [12], т.к. стоимость проекта при 65 нм технологии достигает 30 млн. долларов. Считается, что для сокращения стоимости проектирования необходимо увеличить базовый компонент проекта до 50 тыс. вентилей. Это уровень встроенного процессора. На уровне таких «вентилей» необходимо проектировать объект на системном уровне, чтобы удовлетворять требованиям рынка.

$D_0 = /S_0 * A_0 + S_0 * B_0 \pmod 2$	$(S_0 \vee A_0 \vee \overline{D_0})(A_0 \vee B_0 \vee \overline{D_0})(\overline{S_0} \vee B_0 \vee \overline{D_0}) * *(S_0 \vee \overline{A_0} \vee D_0)(\overline{S_0} \vee \overline{B_0} \vee D_0)(\overline{A_0} \vee \overline{B_0} \vee D_0)$
$G_0 = B_0 + C_0 \pmod 2$	$(C_0 \vee B_0 \vee \overline{G_0})(C_0 \vee \overline{B_0} \vee G_0)(\overline{C_0} \vee B_0 \vee G_0) * *(\overline{C_0} \vee \overline{B_0} \vee \overline{G_0})$
$E_0 = D_0 + C_0 \pmod 2$	$(C_0 \vee D_0 \vee \overline{E_0})(C_0 \vee \overline{D_0} \vee E_0)(\overline{C_0} \vee D_0 \vee E_0) * *(\overline{C_0} \vee \overline{D_0} \vee \overline{E_0})$
$F_0 = E_0 * G_0 \pmod 2$	$(E_0 \vee \overline{F_0})(G_0 \vee \overline{F_0})(\overline{E_0} \vee \overline{G_0} \vee F_0)$
$L_0 = D_0 < G_0 \pmod 2$	$(\overline{B_0} \vee \overline{L_0})(G_0 \vee \overline{L_0})(B_0 \vee \overline{G_0} \vee L_0)$
$A_0 = 0 \pmod 2$	$\overline{A_0}$
$B_0 = 1 \pmod 2$	B_0
$C_0 = 1 \pmod 2$	C_0
$L_0 = 1 \pmod 2$	L_0
$F_0 = 0 \pmod 2$	$\overline{F_0}$

Рис. 7. Функции разрешения для получения 1-го бита результата верификации

Предложенный метод направленного построения тестов может применяться также и на системном уровне проектирования.

[1] Kilby, J. Integrated circuits invented by Jack Kilby. - Texas Instruments, Dallas TX. http: //www.ti.com/corp/ docs/ company/ history/ timeline/ emicon/ 1950/ docs/ 58ic_ kilby.htm. 1958.

[2] Золоторевич, Л.А. Разработка тестов для анализа контролепригодности СБИС на верхних уровнях проектирования / Л.А. Золоторевич, А.В. Ильинкова // Автоматика и телемеханика. – 2010. – №9. – С.162-173.

[3] Zolotorevich, L. A. Development of tests for VLSI circuit testability at the upper design levels / L. A Zolotorevich, A. V. Il'inkova // Automation and Remote Control.- USA, NY, Plenum Press. -Vol. 71.- Issue 9.- 2010.- P. 1888-1898.

[4] Золоторевич, Л.А. Построение моделей цифровых систем для направленного построения тестов контроля / Л.А. Золоторевич, А.В. Ильинкова // The International Conference Computer-Aided Design of Diskrete Devices (CAD-DD'10) . Minsk. 2010. P. 279-286.

[5] Murray, B. T. Hierarchical Test Generation Using Precomputed Tests for Modules/ B. T. Murray, J. P. Hayes // International Test Conference. 1988. P. 221-229.

[6] Gharebaghi, A. M., Navabi, Z. High-Level Test Generation from VHDL Behavioral Descriptions/ A. M. Gharebaghi, Z. Navabi// Proceedings of VHDL International Users Forum Fall Workshop. 2000. P. 123-126.

[7] Inoue, M. Test synthesis for datapath using datapath-controller functions / M. Inoue, K. Suzuki, H. Okamoto, H. Fujiwara // Proceeding of the 12 th Asian Test Symposium (ATS'03). 2003. P. 294-299.

[8] Goloubeva, O. High-level test generation for hardware testing and software validation / O. Goloubeva, M. Sonza Reorda, M. Violante // Workshop of High-Level Design Validation and Test. 2003. P. 143-148.

[9] Jervan, G. High level and hierarchical test sequence generation / G. Jervan, Z. Peng, O. Goloubeva, M. S. Reorda // Workshop of High-Level Design Validation and Test. 2002. P. 196-174.

[10] Золоторевич Л.А. Моделирование неисправностей СБИС на поведенческом уровне на языке VHDL // Информатика. 2005. №3.

[11] Larrabee, T. Test pattern generation using Boolean satisfiability // IEEE Trans. Computer-Aided Design. Vol. 11. No. 1. 1992. P. 4–15.

[12] Electronic Design Automation: Synthesis, Verification, and Test. Edited by L.-T. Wang, Y.-W. Chang, K.-T. Cheng // Elsevier. 2009. 971 P.