НАЦИОНАЛЬНАЯ АКАДЕМИЯ НАУК БЕЛАРУСИ
ОБЪЕДИНЕННЫЙ ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ДИСКРЕТНЫХ СИСТЕМ

Материалы
Пятой международной конференции
16 – 17 ноября 2004 г., Минск

В 2 томах

Том 2

COMPUTER-AIDED DESIGN OF DISCRETE DEVICES CAD DD'04

Proceedings
of the Fifth International Conference
16 – 17 November 2004, Minsk

Volume 2

Минск 2004

УДК 519.873:519.718.7

МОДЕЛИРОВАНИЕ НЕИСПРАВНОСТЕЙ СБИС НА ПОВЕДЕНЧЕСКОМ УРОВНЕ НА ЯЗЫКЕ VHDL

Л.А. Золоторевич
Беларусь, Минск, Белорусский государственный университет

Рассматривается задача моделирования неисправностей СБИС и устройств цифровой электроники, представленных на поведенческом уровне на VHDL. Решение задачи базируется на построении тестов для разных реализаций цифровых блоков на основе применения системы логико-динамического моделирования и генерации тестов VLSI_SIM, разработанной в Белгосуниверситете, анализа полученных тестов и разработке VHDL- моделей неисправностей.

1. Введение

Развитие интегральной схемотехники постоянно сопровождается особым вниманием разработчиков к обеспечению надежности объектов проектирования. В последнее время актуальность проблемы разработки тестов и анализа их полноты продолжает повышаться, а поиск эффективных решений задачи анализа функционирования устройств при наличии неисправностей при нисходящем проектировании на основе VHDL идет по двум различным направлениям. Одно из направлений исследований основано на применении аппаратных прототипов проектируемых устройств [1, 2], другое – на моделировании неисправностей [3 - 5]. Для сокращения времени моделирования и связанных с ним сроков проектирования в некоторых случаях предпочтение отдается созданию аппаратного прототипа. В работе [6] рассматривается создание прототипа на основе применения FPGA. При этом задача решается не на основе выполнения повторного синтеза каждого неисправного прототипа, а путем частичного изменения ресурсов устройства на основе их реконфигурации в процессе функционирования, что сокращает время проведения эксперимента.

В настоящее время ощущается потребность в новых методах и поддерживающих их инструментальных средствах проектирования «толерантных к неисправностям» электронных систем. Необходимы методы и средства построения тестов, моделирования неисправностей, а также методы оценки уровня надежности, которые также предполагают моделирование неисправностей. При этом существует необходимость реше-

ния задачи моделирования неисправностей во время всего процесса проектирования на разных уровнях представления объекта, в том числе на уровне поведенческого описания объекта, когда структура устройства не известна. Такой подход позволяет определять и решать ряд потенциальных проблем на раннем этапе проектирования, что, в свою очередь, упрощает процесс проектирования и сокращает его длительность.

В работе рассматривается задача построения моделей неисправностей цифровых устройств, описанных на поведенческом уровне.

2. Анализ известных методов моделирования неисправностей применительно к объекту, описанному на уровне поведения

Моделирование неисправностей и тестовое диагностирование цифровых систем исторически выполнялось на вентильном уровне. Однако размерность данной задачи применительно к проектам современных СБИС в целом на уровне вентильного представления ограничивает возможность ее эффективного решения. В литературе имеются сообщения о некоторых подходах к моделированию неисправностей на поведенческом уровне, в том числе основанных на применении специальных инструментариев для введения неисправностей в исходное описание объекта, грамматического анализа VHDL-описаний и моделирования. Имеются инструменты и для моделирования СБИС на структурном уровне на языке VHDL [4, 5]. В то же время отсутствуют эффективные методы и системы для решения задачи в целом. В работах [7, 8] предлагаются относительно сложные модели неисправностей, в которых делается попытка связать поведенческие модели с реальными аппаратными неисправностями, однако аргументация приводимого соответствия отсутствует. В работе [9] поведенческие неисправности подразделяются на две категории: неисправности микроопераций (micro-operation faults) и неисправности управления (control faults). Одна из них предполагает замену одних операций другими, например, операцию логического сложения на операцию умножения и наоборот, операцию арифметического сложения на операцию вычитания и т.д. Остается открытым вопрос, насколько подобная замена адекватна реальным неисправностям в соответствующих аппаратных средствах.

Вторая категория включает

- неисправные переходы в конструкции IF: переход осуществляется только в одном направлении, независимо от значения управляющего сигнала (*stuck THEN, stuck ELSE*);
- не выполняемая последовательность предложений в операторе CASE *(Dead Clause)*;
- не выполняется оператор PROCESS *(Dead Process)*;
- не правильно выполняется оператор назначения.

Недостатком предлагаемых решений является то, что отсутствует обоснование соответствия поведенческих моделей и реальных неисправностей аппаратуры.

Ряд исследователей предлагают рассматривать поведенческие неисправности объектов как программные ошибки и соответственно применять методы мутирования исходных кодов, разработанные для тестирования программного обеспечения [12]. На основе предложенных поведенческих моделей разработан алгоритм генерации тестов «В-алгоритм» [10,11], эффективность которого напрямую связана с корректностью применяемых моделей неисправностей.

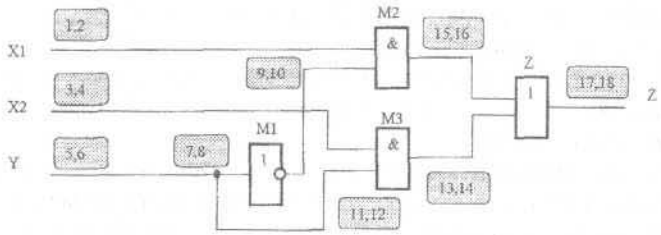
3. Переход от функционально-логического к поведенческому уровню моделирования неисправностей

Известно, что средства синтеза объектов, описанных на языке VHDL, ориентированы на некоторое подмножество операторов языка, так называемое синтезируемое подмножество [13]. К тому же они ориентированы на применение соответствующих библиотек компонентов, так что каждая конструкция операторов языка из его синтезируемого подмножества реализуется определенной структурой.

Поставим задачу построения и моделирования неисправностей цифровых объектов комбинационного типа на поведенческом уровне их описания таким образом, чтобы они *соответствовали неисправностям их физических реализаций*. Для иллюстрации подхода выберем одну из простых конструкций языка, к примеру, if-then-else (рис. 1), и рассмотрим ее некоторые аппаратные реализации на основе мультиплексора. На рис. 2 приведена реализация функции на основе ДНФ, на рис. 3 та же функция реализована на основе конъюнктивной нормальной формы.

```
if  Y = '0' then
    Z <=X1;
else
    Z <=X2;
end if
```

Рис. 1. Конструкция языка VHDL <if-then-else> для выбора одного из двух входных сигналов



а)

X1 \ Y X2	00	01	11	10
0	0	0	1	0
1	1	1	1	0

б)

Рис. 2. Реализация конструкции на основе ДНФ:
а) логическая структура, б) карта Карно

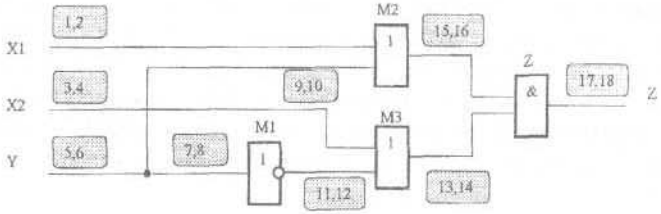


Рис. 3. Реализация конструкции на основе
конъюнктивной нормальной формы

Средствами программной системы VLSI_SIM [14] сгенерируем тесты (рис. 4) контроля неисправностей константного типа для обеих реализаций схем (моделируемые неисправности пронумерованы на рисунках: нечетные номера обозначают неисправности типа "const 0" на соответствующей линии, четные - "const 1"). Результаты сведем в табл. 1. (Заметим, что в системе VLSI_SIM имеются возможности генерации оптимального по длине и контролирующей способности теста для цифровых устройств комбинационного и последовательностного типов с числом компонентов порядка десятков тысяч функциональных блоков).

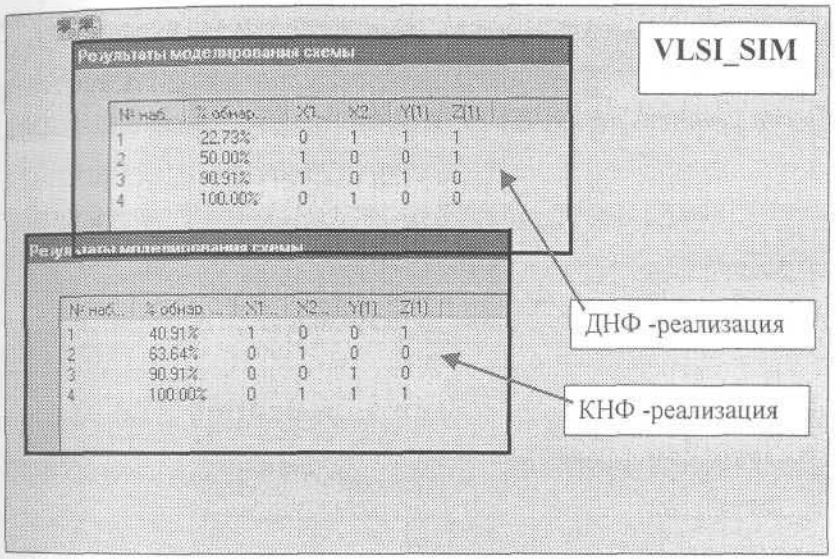


Рис. 4. Результаты генерации тестов в системе VLSI_SIM

Очевидно, табл. 1 можно значительно сократить за счет объединения одинаковых столбцов, соответствующих эквивалентным неисправностям. В табл. 2 каждый столбец соответствует группе эквивалентных неисправностей, покрываемых соответствующими входными наборами. Анализируя табл. 2, можно выделить группы 1-4 неисправностей, которые покрываются только единственным входным набором. В то же время эти наборы покрывают и неисправности из групп 5-8. Неисправности, входящие в группы 5-8, можно исключить из дальнейшего рассмотрения, так как они будут обнаружены при обнаружении неисправностей из групп 1-4. Анализируя неисправности групп 1-4, получаем множество соответствующих им моделей поведенческих неисправностей (табл. 3). Из полученного множества моделей неисправностей можно выделить

четыре поведенческие неисправности, сведенные в табл. 4. Обнаружение данных четырех поведенческих неисправностей обеспечивает обнаружение всех неисправностей константного типа, соответствующих реальному объекту, представленному на вентильном уровне.

В докладе приводятся результаты построения поведенческих тестов для некоторых арифметических, логических и других операторов языка VHDL.

Таблица 1

Автоматически сгенерированные тесты и покрываемые неисправности

Неисправности →	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Тест. век. ↓	ДНФ – реализация схемы																		
011	1			0		0						0		0				0	
010	0		1				1						1		1		1		1
101	0				1	1		1			1				1		1		1
100	1	0					0		0	0						0		0	
	КНФ – реализация схемы																		
100	1	0					0		0			0		0		0		0	
010	0		1				1				1						1		1
101	0				1			1					1		1				1
011	1			0		0				0				0		0		0	

Таблица 2

Группы эквивалентных неисправностей

Группы неисправностей →		1	2	3	4		5	6	7	8
Неисправности →	0	1,8,9,15	2,12	3,11,13	4,7,10		5	6	14,16,18	17
Тест-векторы↓	ДНФ – реализация схемы									
011	1			0			0			0
010	0		1					1	1	
101	0				1		1		1	
100	1	0						0		0
	КНФ – реализация схемы									
Группы неисправностей →		1	2	3	4		6	7	8	
Неисправности →		1,8,11	2,10,16	3,5,9	4,7,12,14		6	13,15,17	18	
100	1	0					0	0		
010	0		1				1		1	
001	0				1				1	
011	1			0				0		

Таблица 3

Множество неисправностей на поведенческом уровне

Группы ↓	ДНФ - реализация	КНФ реализация
1	if Y = '0' then Z <=0; else Z <=X2; end if	if Y = '0' then Z <=0; else Z <=X2; end if if Y = '0' then Z <=X1andX2; else Z <=X2; end if
2	if Y = '0' then Z <=1; else Z <=X2; end if if Y = '0' then Z <=X1 or X2; else Z <=X2; end if	if Y = '0' then Z <=1; else Z <=X2; end if
3	if Y = '0' then Z <=X1; else Z <=0; end if	if Y = '0' then Z <=X1; else Z <=0; end if if Y = '0' then Z <=X1; else Z <= X1andX2; end if
4	if Y = '0' then Z <=X1; else Z <=1; end if if Y = '0' then Z <=X1; else Z <= X1 or X2; end if	if Y = '0' then Z <=X1; else Z <=1; end if

Таблица 4

Неисправные модификации оператора if-then-else

Неисправности типа «И»	Неисправности типа «ИЛИ»

if Y = '0' then
Z <=X1andX2;
else
Z <=X2;
end if
	if Y = '0' then
	Z <=X1 or X2;
	else
	Z <=X2;
	end if

if Y = '0' then
Z <=X1;
else
Z <= X1andX2;
end if
	if Y = '0' then
	Z <=X1;
	else
	Z <= X1 or X2;
	end if

4. Заключение

Предложена методика построения функциональных моделей неис правностей конструкций операторов if-then-else описаний СБИС на по веденческом уровне их представления. Для решения задачи рассматри ваются возможные физические реализации каждой конструкции. Мето дика может использоваться для других операторов, физически реали зуемых блоками комбинационного типа. Для каждой известной физиче ской реализации на вентильном уровне в системе VLSI_SIM в режим интерактивного поиска генерируются тесты, покрывающие все возмож ные неисправности константного типа. На основе неисправных функ ций, соответствующих определенному подмножеству рассматриваемы неисправностей, строятся поведенческие неисправности. При их обна ружении обеспечивается обнаружение всех неисправностей любой и физических реализаций рассматриваемого оператора VHDL.

Литература

1. J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.C. Fabre, J.-C. Laprie, E Martins, D. Powell. Fault Injection for Dependability Validation: A Methodology and some Applications// IEEE Transactions on Software Engineering.- Vol 16.- No. 2.- 1990.

2. J. Karlsson, P. Liden, P. Dahlgren, R. Johansson, U. Gunneflo. Using Heavy-Ion Radiation to Validate Fault- Handling Mechanisms// IEEE Micro.- Vol. 14.- No. 1-1994.- P. 8-32.

3. T. A. Delong, B. W. Johnson, and J. A. Profeta. A fault injection technique for VHDL behavioral-level models// IEEE Design Test Comput.- Vol. 13.- 1996.- P. 24–33.

4. E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson. Fault injection into VHDL models: The MEFISTO tool // In 24th Int. Symp. Fault-Tolerant Comput. – June 1994. – P. 66–75.

5. J. Boué, P. Pétilton, and Y. Crouzet. MEFISTO-L: A VHDL-based fault injection tool for the experimental assessment of fault tolerance // In 28th FTCS.- June 1998.- P. 168–173.

6. Lörinc Antoni, Régis Leveugle, and Béla Fehér. Using Run-Time Reconfiguration for Fault Injection Applications // IEEE Ttransactions on instrumentation and Measurement.- Vol. 52.- No. 5.- 2003.

7. Chakraborty, T. and S. Ghosh. On Behavior Fault Modeling for Combitional Digital Designs // Proceedings International Test Conference, September 1988.- P.593-600.

8. Ghosh, S. and T.J. Chakraborty. On Behavior Fault Modeling for Digital Designs // Journal of Electronic Testing.- Vol. 2.- Kluwer Academic Publishers.- 1991.

9. Armstrong, J.R., F.S. Lam, and P.C. Ward. Test Generation and Fault Simulation for Behavioral Models // Performance and Fault Modeling with VHDL.-J.M. Schoen, ed., Prentice Hall, Englewood Cliffs, NJ.- 1992.-P. 240.

10. Cho, C.H. A Formal Model for Behavioral Test Generation // Doctoral dissertation.-Virginia Polytechnic Institute and State University.- Department of Electrical Engineering.- Blacksburg, VA.- 1994.

11. Cho, C.H. and J.R. Armstrong. B-algorithm: A Behavioral Test Generation Algorithm // Proceedings International Test Conference.- 1994.-P. 968-979.

12. Al Hayek, G. and C. Robach . On the Adequacy of Deriving Hardware Test Data from the Behavioral Specification // Proceedings EUROMICRO 96, 22nd Euromicro Conference.- September 1996.- P. 337-342.

13. IEEE P1076.6/D1.12, Draft Standard for VHDL Register Transfer Level Synthesis /VHDL Synthesis Interoperability Working Group.- IEEE.- Piscataway, NJ.- March 1998.

14. Золоторевич Л.А., Сидоренко О.М., Юхневич Д.И. Основные возможности базовых средств САПР микроэлектроники VLSI_SIM // Материалы 3-ей Международной научно-практической конференции «Вузовская наука, промышленность, международное сотрудничество». -Т.2.- Мн: БГУ. -2000. -С. 164-170.