ИНФОРМАТИКА

2005 июль-сентябрь № 3

УДК 519.873:519.718.7

Л.А. Золоторевич

МОДЕЛИРОВАНИЕ НЕИСПРАВНОСТЕЙ СБИС НА ПОВЕДЕНЧЕСКОМ УРОВНЕ НА ЯЗЫКЕ VHDL

Рассматривается задача моделирования неисправностей блоков комбинационного типа СБИС, представленных на поведенческом уровне на языке VHDL. Предлагается формальный подход к решению задачи, в основе которого лежит построение тестов для разных реализаций операторов языка VHDL на базе применения системы генерации тестов и моделирования неисправностей. Особенность получаемых функциональных моделей неисправностей состоит в их полном соответствии физическим неисправностям.

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 решения задачи моделирования неисправностей во время всего процесса проектирования на разных уровнях представления объекта, в том числе на уровне поведенческого описания объекта, когда структура устройства неизвестна. 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 настоящей работе рассматривается задача построения моделей неисправностей цифровых устройств, описанных на поведенческом уровне.

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

Моделирование неисправностей и тестовое диагностирование цифровых систем исторически выполнялись 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 по-

веденческие 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).

Первая категория предполагает замену одних операций другими, например операцию логического сложения на операцию умножения, 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:

– ошибочные переходы в конструкции IF (переход осуществляется только в одном направлении независимо от значения управляющего сигнала (*stuck THEN, stuck ELSE*));

– невыполняемую последовательность предложений в операторе CASE (*Dead Clause*);

– невыполняемый оператор PROCESS (*Dead Process*);

– неправильно выполняемый 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 «B-алгоритм» [10, 11], эффективность которого напрямую связана с корректностью применяемых моделей неисправностей.

2. 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 часто применяемых конструкций языка, к примеру if-then-else (рис.1, *а*), и рассмотрим ее ДНФ- и КНФ-аппаратные реализации на основе мультиплексора (рис. 2, 3).

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

а)

```
if    Y = '00' then
      Z <=X1;
elsif Y = '10' then
      Z <=X2;
elsif Y = '01' then
      Z <=X3;
elsif Y = '11' then
      Z <=X4;
end if
```

б)

Рис. 1. Конструкция языка VHDL <if-then-else>: *а)* для выбора одного из двух входных сигналов; *б)* для выбора одного из четырех

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»). Наборы полученных тестов и покрываемые ими неисправности приведены в табл. 1. Заметим, что в системе VLSI_SIM имеются возможности

интерактивной генерации оптимального по длине и контролирующей способности теста для цифровых устройств комбинационного и последовательностного типов с числом компонентов порядка десятков тысяч функциональных блоков.

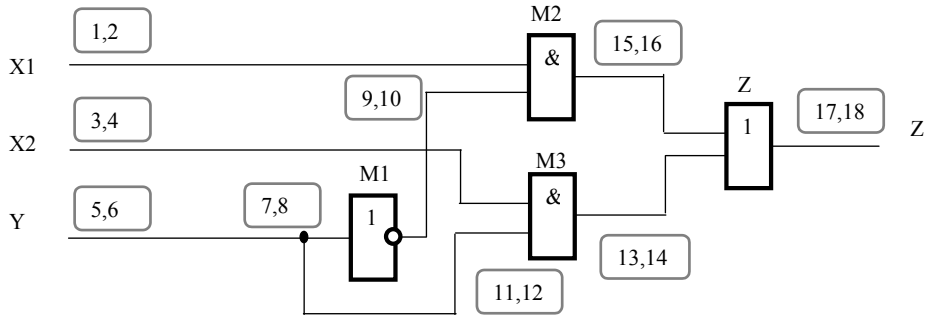


Рис. 2. Реализация конструкции <if-then-else> на основе ДНФ

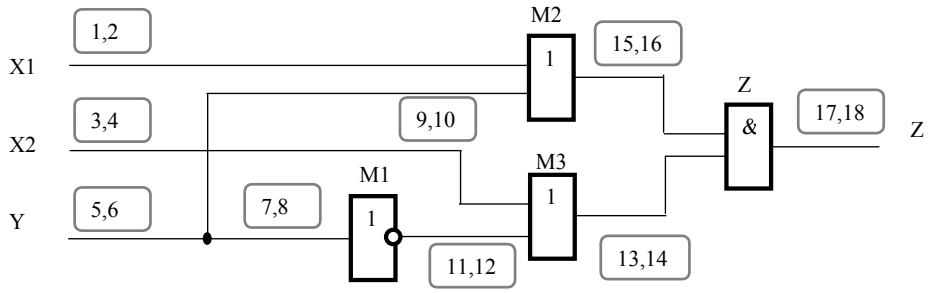


Рис. 3. Реализация конструкции <if-then-else> на основе КНФ

Очевидно, что 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 покрыты при обнаружении неисправностей из групп 1–4. Получим следующие функции неисправностей.

Для неисправностей ДНФ-реализации:

гр. 1 – $Z = X2 \wedge Y$;

гр. 2 – $Z = \neg Y \vee X2 \wedge Y$; $Z = X1 \wedge \neg Y \vee X2$;

гр. 3 – $Z = X1 \wedge \neg Y$;

гр. 4 – $Z = Y \vee X1 \wedge \neg Y$; $Z = X1 \vee X2 \wedge Y$.

Для неисправностей КНФ-реализации:

гр. 1 – $Z = X2 \wedge Y$, $Z = X1 \wedge X2 \vee X2 \wedge Y$;

гр. 2 – $Z = X2 \vee \neg Y$;

гр. 3 – $Z = X1 \wedge \neg Y$, $Z = X1 \wedge X2 \vee X1 \wedge \neg Y$

гр. 4 – $Z = X1 \vee Y$.

Анализируя данные функции, получаем множество моделей поведенческих неисправностей (табл. 3). Из множества моделей неисправностей можно выделить четыре поведенческие неисправности (табл. 4). Покрытие этих четырех поведенческих неисправностей обеспечивает обнаружение всех неисправностей константного типа, соответствующих реальному объекту, представленному на вентильном уровне в виде ДНФ- или КНФ-реализаций.

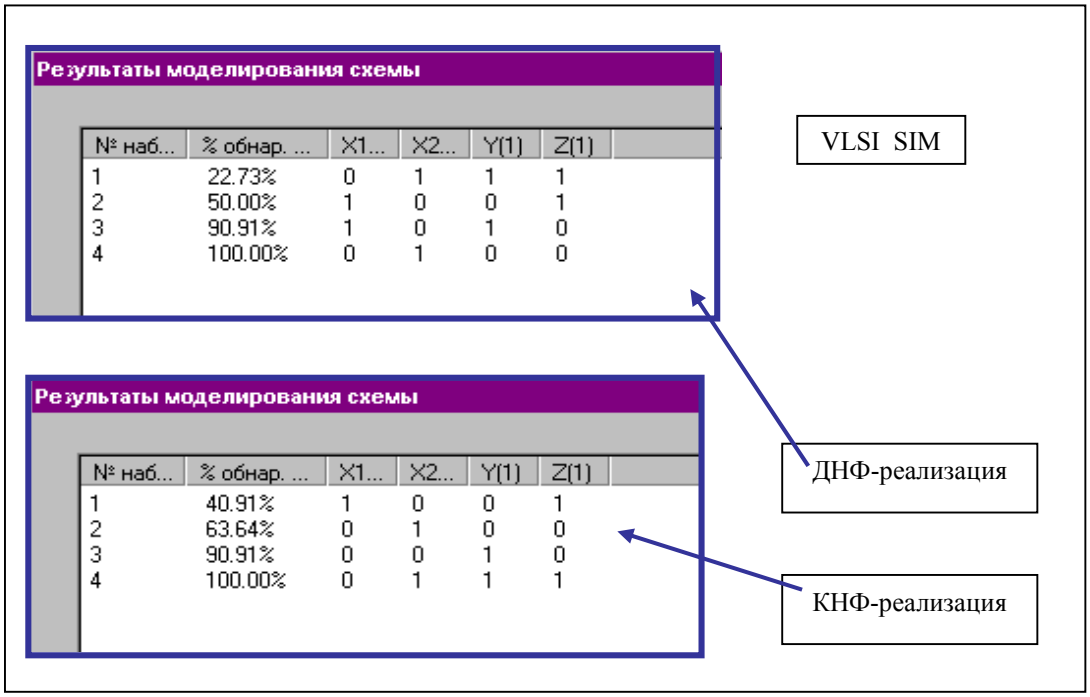


Рис. 4. Результаты генерации тестов в системе VLSI_SIM

Таблица 1

Автоматически сгенерированные тесты и покрываемые неисправности

Неиспр. / Тест-векторы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
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**
001	0				**1**	**1**		**1**					**1**		**1**				**1**
011	1			**0**		**0**				**0**				**0**		**0**		**0**	

Построим аналогичным образом поведенческие тесты для конструкции операторов if-then-else (см. рис. 1, *б*), реализованной в виде четырехканального мультиплексора (рис. 5). На рис. 5 показаны результаты генерации тестов контроля в системе VLSI_SIM; в табл. 5 приведены неисправности, обнаруживаемые на каждом тестовом векторе, а в табл. 6 обнаруживаемые неисправности объединены в группы эквивалентных неисправностей. Рассматривая первые восемь групп неисправностей, покрываемых только одним набором теста, и выписывая функции, соответствующие этим неисправностям, можно для данной вентильной реализации построить соответствующие функциональные неисправности, которые содержат 12 конструкций операторов if-then-else, приведенных в табл. 7.

Таблица 2

Группы эквивалентных неисправностей

Группы неиспр.		**1**	**2**	**3**	**4**		**5**	**6**	**7**	**8**
Неиспр. / Тест-векторы	0	1,8,9,15	2,12	3,11,13	4,7,10		5	6	14,16,18	17
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										
H		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

Поведенческие тесты

Группы неиспр.	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

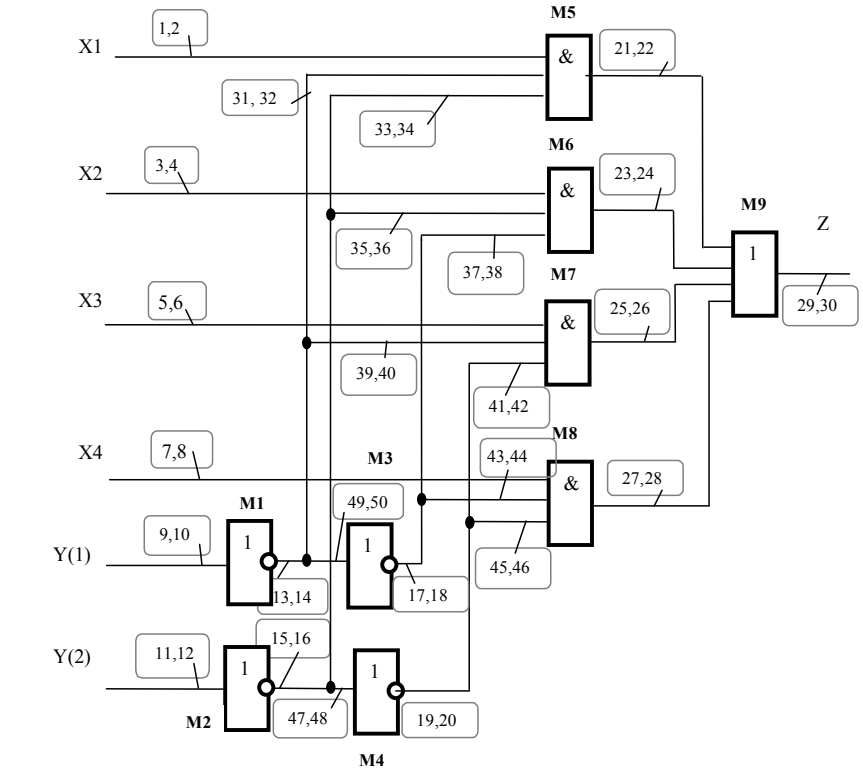
4	if Y = ‘0’
	then
	Z <=X1;
	else
	Z <=1;
	end if
	
	if Y = ‘0’
	then
	Z <=X1;
	else
	Z <= X1 or X2;
	end if
		if Y = ‘0’
		then
		Z <=X1;
		else
		Z <=1;
		end if

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



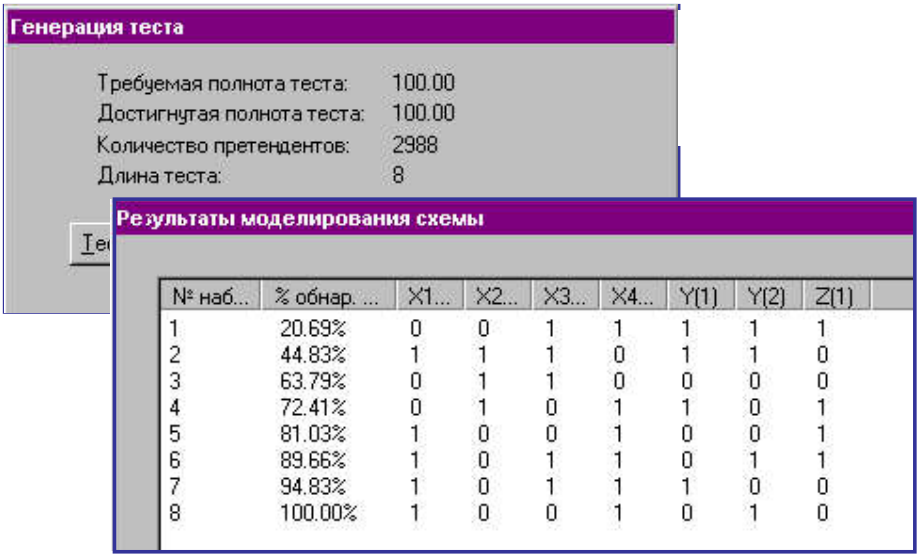


Рис. 5. Вентильное представление четырехканального дешифратора и результаты генерации теста в системе VLSI_SIM

Предложенный подход к разработке моделей неисправных модификаций проекта, описанного на поведенческом уровне на языке VHDL, можно непосредственно применить для операторов VHDL-кода, которые реализуются на этапе синтеза структурами комбинационного типа. Что касается операторов, реализуемых блоками с элементами памяти, та данная задача требует дополнительного исследования. Полученные в работе модели неисправностей для конструкций операторов типа if-then-else могут быть непосредственно использованы в системе генерации тестов и анализа контролепригодности проектов на системном уровне и уровне межрегистровых передач для оценки полноты тестов и разработки системы тестового диагностирования. Заметим, что задача разработки методов и программных систем генерации тестов, анализа их контролирующих свойств и оценки контролепригодности проектов на высоких уровнях проектирования вытекает непосредственно из потребностей современной интегральной схемотехники, но ввиду ее сложности в литературе до настоящего времени отсутствуют эффективные, пригодные для практического применения решения.

Заключение

Предложена методика построения функциональных моделей неисправностей конструкций операторов if-then-else описаний СБИС на поведенческом уровне их представления. Для решения задачи рассматриваются возможные физические реализации каждой конструкции. Методика может использоваться для других операторов, физически реализуемых блоками комбинационного типа. Для каждой известной физической реализации на вентильном уровне в системе VLSI_SIM в режиме интерактивного поиска генерируются тесты, покрывающие все возможные неисправности константного типа. На основе функций неисправностей, соответствующих определенному подмножеству рассматриваемых неисправностей, строятся поведенческие неисправности. При их обнаружении обеспечивается нахождение всех неисправностей любой из физических реализаций рассматриваемого оператора VHDL.

Список литературы

1. Arlat J., Aguera M., Amat L. Fault injection for dependability validation: a methodology and some applications // IEEE transactions on software engineering. – V. 16. – № 2. – 1990.

2. Karlsson J., Liden P., Dahlgren P. Using heavy-ion radiation to validate fault-handling mechanisms // IEEE Micro. – V. 14. – № 1. – 1994. – P. 8–32.

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

4. Jenn E., Arlat J., Rimen M. Fault injection into VHDL models: The MEFISTO tool // 24th Int. symp. fault-tolerant comput. – June 1994. – P. 66–75.

5. Boué J., Pétilton P., Crouzet Y. 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, Béla Fehér. Using run-time reconfiguration for fault injection applications // IEEE Ttransactions on instrumentation and Measurement. – V. 52. – № 5. – 2003.

7. Chakraborty T., Ghosh S. On behavior fault modeling for combinational digital designs // Proc. International test conference, September 1988. – P. 593–600.

8. Ghosh S., Chakraborty T.J. On behavior fault modeling for digital designs // Journal of Electronic Testing. V. 2. – Kluwer Academic Publishers, 1991.

9. Armstrong J.R., Lam F.S., Ward P.C. Test generation and fault simulation for behavioral models // Performance and Fault Modeling with VHDL. Prentice Hall, Englewood Cliffs, NJ, 1992. – P. 240–303.

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., Armstrong J.R. B-algorithm: A behavioral test generation algorithm // Proc. International test conference, 1994. – P. 968–979.

12. Al Hayek G., Robach C. On the adequacy of deriving hardware test data from the behavioral specification // Proc. 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 // Мат. 3-й Междунар. науч.-практ. конф. «Вузовская наука, промышленность, международное сотрудничество». Т.2. – Мн.: Университетское, 2000. – С. 164–170.

Поступила 25.11.04

Белорусский государственный университет
Минск, пр. Независимости, 4
e-mail: zolotorevichLA@bsu.by

L.A. Zolotorevich

VLSI FAULTS SIMULATION ON BEHAVIORAL LEVEL BY VHDL LANGUAGE

The problem of simulation of faults combinative units VLSI on a behavioral level by VHDL is considered. The formal approach to the solution of the problem based on the construction of the tests for different realizations of VHDL language operators by use of test generation system and simulation of faults is offered. The developed functional models completely correspond with physical faults.