

КОМПЬЮТЕРНАЯ ИНЖЕНЕРИЯ И ТЕХНИЧЕСКАЯ ДИАГНОСТИКА



УДК 681.326:519.613

ИНФРАСТРУКТУРА ВСТРОЕННОГО ВОССТАНОВЛЕНИЯ ЛОГИЧЕСКИХ PLD-СХЕМ

MURAD ALI ABBAS, ХАХАНОВ В.И.,
ЛИТВИНОВА Е.И., ХАХАНОВА И.В.

Предлагается инфраструктура моделирования комбинационных схем, ориентированная на решение практических задач встроенного восстановления работоспособности компонентов логических устройств. Логическая схема дополняется операционным и управляющим автоматами моделирования цифровых устройств, что увеличивает время обработки и аппаратные затраты для создания оболочки адресуемых элементов. Структуры также можно использовать для аппаратного моделирования функциональностей цифровых проектов на основе использования PLD, что дает возможность существенно повысить быстродействие верификации программных моделей. Предложенное решение задачи встроенного ремонта логических элементов комбинационных схем дает возможность комплексно решать проблему автономного восстановления работоспособности цифровых систем на кристаллах за счет временной и аппаратной избыточности проекта.

1. Введение

Понятие адресного выполнения логических операций, реализованных на элементах памяти LUT в программируемых логических устройствах (PLD), дает потенциальную возможность создавать на кристалле только адресное пространство для встроенного восстановления работоспособности всех компонентов, участвующих в формировании функциональности [1-18]. Актуальность создания адресного пространства для всех компонентов подтверждается следующим распределением логики и памяти на кристалле, представленным на рис. 1.

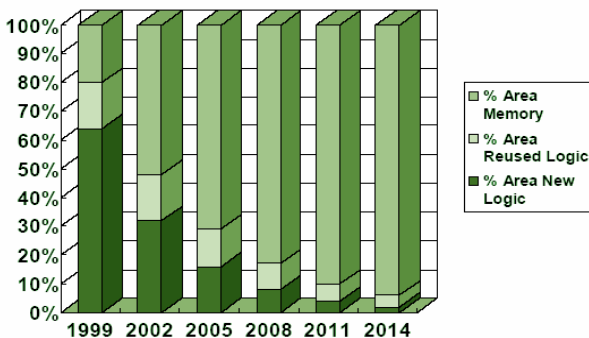


Рис. 1. Соотношение памяти и логики на кристалле

Тенденция к увеличению памяти влечет возможность встроенного восстановления работоспособности отказавших ячеек за счет выделенных дополнительных ресурсов для их ремонта (spare logic cells). Проблема автономного устранения дефектов (самовосстановления работоспособности) логических элементов связана с отсутствием у них адресов. Но решить ее можно, если связи между элементами логики сделать гибкими с помощью программы описания структуры, помещенной в память, которая соединит логические компоненты в схему. Кроме структуры взаимодействия элементов, память должна содержать порядок их обработки. В случае возникновения дефекта в одном из адресуемых логических элементов система встроенного тестирования восстановит его работоспособность путем переадресации на заведомо исправный аналог из ремонтного запаса.

Цель – повышение качества цифровых систем на кристаллах путем создания инфраструктуры встроенного тестирования, диагностирования, оптимизации, восстановления работоспособности за счет аппаратной избыточности и уменьшения быстродействия выполнения функциональности [1-18].

Задачи и источники. 1) Разработка математической модели встроенного ремонта логических элементов, входящих в комбинационную структуру функциональности в виде цифровой системы на кристалле [7-18]. 2) Создание операционного и управляющего автоматов для эмулирования или моделирования функциональности комбинационной схемы в кристалле PLD [1-6].

2. Модель комбинационной структуры

Одна из немногочисленных работ, посвященных восстановлению работоспособности логических схем, представлена в [9]. Здесь основная идея заключается в реконфигурации структуры логических элементов в режиме off-line, которая обеспечивает возможность замены каждого из неисправных примитивов. Далее предлагается в качестве примера для рассмотрения теории и практики встроенного ремонта функциональных нарушений логических элементов использовать описание простейшей схемной структуры (рис. 2).

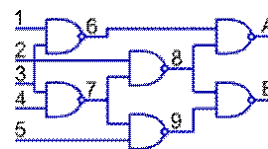


Рис. 2. Пример схемной структуры из неадресуемых элементов

Она содержит шесть однотипных логических элементов, которые можно представить в адресном пространстве следующим списком (двумерным массивом):

$$S = \begin{matrix} \text{No} = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \text{P} = & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \text{L}_1 = & 1 & 3 & 2 & 7 & 6 & 8 & X & X & X \\ \text{L}_2 = & 3 & 4 & 7 & 5 & 8 & 9 & X & X & X \\ \text{L}_3 = & 6 & 7 & 8 & 9 & A & B & Y & Y & Y \end{matrix} \quad \begin{matrix} \text{No} = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & A & B \\ \text{M} = & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix} \quad \begin{matrix} X_1 & X_2 & Y \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix} \quad F(i) =$$

Каждый столбец соответствует логическому элементу схемы, а примитивы с номерами 7, 8,9 являются запасными, которые используются для замены любых трех из шести элементов при диагностировании в последних каких-либо функциональных нарушений. В строке Р указаны типы примитивов, ниже – номера входных и выходных переменных, вектор моделирования М содержит результат моделирования входного слова 11111 на схемной структуре, представленной рис. 3.

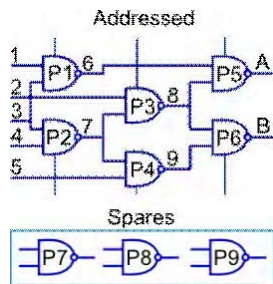


Рис. 3. Пример схемной структуры из адресуемых и запасных элементов

Процесс-модель формирования выходных значений схемы в зависимости от конкатенированных состояний входов, формирующих адрес ячейки состояния выхода, имеет следующий вид:

$$\begin{aligned} Y_6 &= P_1(X_1 * X_3); \\ Y_7 &= P_2(X_3 * X_4); \\ Y_8 &= P_3(X_2 * X_7); \\ Y_9 &= P_4(X_7 * X_5); \\ Y_A &= P_5(X_6 * X_8); \\ Y_B &= P_6(X_8 * X_9). \end{aligned}$$

Учитывая, что все значения переменных сведены в один вектор состояния М, можно получить процесс-модель:

$$\begin{aligned} M_6 &= P_1(M_1 * M_3); \\ M_7 &= P_2(M_3 * M_4); \\ M_8 &= P_3(M_2 * M_7); \\ M_9 &= P_4(M_7 * M_5); \\ M_A &= P_5(M_6 * M_8); \\ M_B &= P_6(M_8 * M_9). \end{aligned}$$

Здесь \underline{M} – вектор состояния линий схемы; $F = M_i \wedge M_j$ – логическая функция И-НЕ, имеющая два входа, реализованная в виде элемента памяти LUT. Поскольку все шесть примитивных элементов реализуют одну логическую функцию И-НЕ, то предыдущее выражение можно упростить:

$$\begin{aligned} M_6 &= F(M_1 * M_3); \\ M_7 &= F(M_3 * M_4); \\ M_8 &= F(M_2 * M_7); \\ M_9 &= F(M_7 * M_5); \\ M_A &= F(M_6 * M_8); \\ M_B &= F(M_8 * M_9). \end{aligned}$$

Имея в виду наличие двумерного массива линий связи (L) между входами и выходами логических элементов, предыдущее выражение можно свести к виду:

$$\begin{aligned} M_6 &= F[M(L_{11}) * M(L_{12})]; \\ M_7 &= F[M(L_{21}) * M(L_{22})]; \\ M_8 &= F[M(L_{31}) * M(L_{32})]; \\ M_9 &= F[M(L_{41}) * M(L_{42})]; \\ M_A &= F[M(L_{51}) * M(L_{52})]; \\ M_B &= F[M(L_{61}) * M(L_{62})]. \end{aligned}$$

Таким образом, можно синтезировать структуру для реализации процесс-модели схемы, имеющей двух-входные функциональные примитивы, в следующем виде: $M(L_{is_p}) = F[M(L_{ij}) * M(L_{ir})] = F[M(L)]$.

Учитывая факт, что все вычисления в схеме привязаны к структурным элементам, которые имеют идентификатор логической операции, предыдущую формулу можно трансформировать к виду:

$$M(L_{is_p}) = P_i[M(L_{ij}) * M(L_{ir})] = P[M(L)].$$

В общем случае структура модели функциональности, ориентированной на реализацию в кристалле PLD, содержит пять компонентов:

$$\begin{aligned} S &= \langle P, F, M, L, T \rangle, \\ P &= (P_1, P_2, \dots, P_i, \dots, P_n); \\ F &= (F_1, F_2, \dots, F_j, \dots, F_m); \\ M &= (M_1, M_2, \dots, M_r, \dots, M_k); \\ L &= [L_{pq}]; p = \overline{1, n}; q = \overline{1, s_p}; \\ T &= [T_{te}]; t = \overline{1, \eta}; e = \overline{1, \mu}; \\ M(L) &= P[M(L)]. \end{aligned}$$

Здесь представлены: 1) примитивы схемной структуры Р, определенные идентификаторами типа функциональности (номер или код команды); 2) типы функциональных элементов F – набор элементов памяти LUT, из которых реализуются примитивы, а также избыточные элементы для ремонта функциональностей; 3) вектор моделирования М (двоичный), определяющий состояния всех линий (входные, внутренние, выходные); 4) матрица эквивалентных линий связи L для объединения логических элементов в структуру; 5) матрица входных тестовых (рабочих) наборов Т. Обработка (processing) схемы в кристалле сводится к определению адреса, составленного двоичными битами вектора моделирования, по которому находится логическая функция. Каждый примитив имеет цикл обработки, содержащий три процедуры:

- 1) Адресное считывание номеров входных переменных из соответствующего столбца матрицы L для формирования адреса состояния входной переменной вектора моделирования: $A = L_{ij}$, $i = \overline{1, n}$; $j = \overline{1, s_p} - 1$.
- 2) Формирование адреса (двоичного кода) для вычисления логической функции путем конкатенации

соответствующих состояний входных переменных в векторе моделирования $A = M(L_{ij}) * M(L_{ir})$.

3) Запись результата выполнения логической функции как состояния выхода в соответствующий разряд вектора моделирования

$$M(L_{is_p}) = F[M(L_{ij}) * M(L_{ir})].$$

3. Операционное устройство для моделирования комбинационной структуры

Процесс обработки всех примитивов схемы в данном случае является строго последовательным, что представляет собой существенное замедление процедуры формирования состояний выходных переменных. Однако уменьшение быстродействия можно считать платой за сервис встроеного и автономного восстановления работоспособности цифровой структуры, который является одним из этапов функционирования инфраструктуры обслуживания SoC, представленной на рис. 4. Комбинационная схема становится операционным устройством, где присутствуют операционный и управляющий автоматы. Заменяемыми компонентами в операционном автомате являются типы примитивов – функциональные элементы (рис. 5).

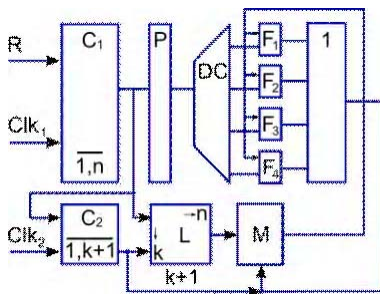


Рис. 4. Операционная структура комбинационной схемы

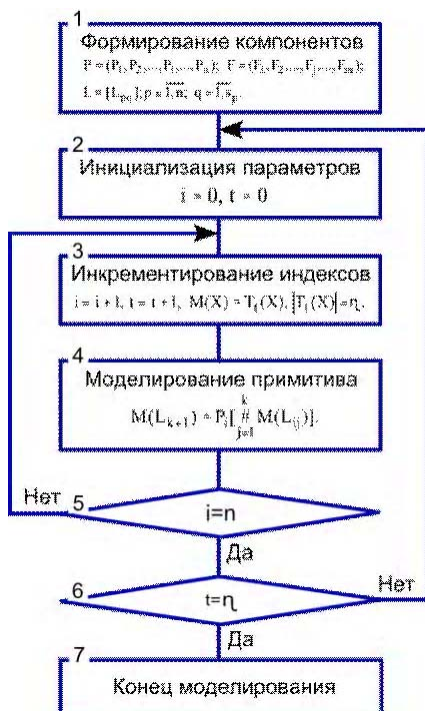


Рис. 5. Граф-схема алгоритма управления процессом моделирования

Операционное устройство реализации элементо-адресуемых комбинационных схем содержит: счетчик обработки текущего примитива C1; память для хранения типов примитивов, соответствующих структурным элементам P; счетчик считывания номеров входных и выходных переменных текущего примитива C2; дешифратор типов примитивов DC; память для хранения вектора моделирования M; матричную память для хранения номеров входов-выходов структурных примитивов L; линейку памяти, реализующих функциональные примитивы F; регистр формирования входного адресного слова для обрабатываемого примитива RG; логический элемент Og для коммутации результатов обработки функциональных примитивов.

Граф-схема алгоритма управления процессом моделирования структуры комбинационной схемы представлен на рис. 5.

1. Инициализация (формирование) всех компонентов (номера и типы элементов, линии связей для входов и выходов логических элементов) схемной структуры:

$$P = (P_1, P_2, \dots, P_i, \dots, P_n); F = (F_1, F_2, \dots, F_j, \dots, F_m);$$

$$L = [L_{pq}]; p = \overline{1, n}; q = \overline{1, s_p}.$$

2. Инициализация параметра обрабатываемого примитива и номера входного набора $i = 0, t = 0$ для его моделирования в двоичном алфавите $M_T = \{0,1\}$.

3. Инкрементирование индекса примитива, номера теста и инициализация входного тестового (рабочего) набора:

$$i = i + 1, t = t + 1, M(X) = T_t(X), |T_t(X)| = \eta.$$

4. Конкатенация (#) разрядов слова для формирования входного воздействия $\#_{j=1}^k M(L_{ij})$ логического

элемента P_i и выполнение процедуры определения состояния его выхода с последующей записью в соответствующую координату вектора моделирования:

$$M(L_{k+1}) : M(L_{k+1}) = P_i \left[\#_{j=1}^k M(L_{ij}) \right].$$

5. Повторение пунктов 3 и 4 в целях получения состояний выходов всех логических элементов до выполнения условия: $i = n$.

6. Повторение пунктов 2–4 в целях моделирования всех входных тестовых (рабочих) наборов, до выполнения равенства: $t = \eta$, где η – длина теста.

7. Окончание процесса моделирования цифрового устройства.

4. Заключение

Научная новизна. Предложенные операционный и управляющий автоматы моделирования цифровых комбинационных схем ориентированы на решение

двух практически ориентированных задач: 1) Встроенное восстановление работоспособности компонентов комбинационных логических схем путем увеличения времени обработки цифрового устройства и дополнительных аппаратных затрат для создания инфраструктуры моделирования адресных элементов. 2) Аппаратное моделирование функциональностей цифровых проектов на основе использования PLD, что дает возможность существенно повысить быстродействие верификации программных моделей.

Практическая значимость. Положительное решение задачи встроенного ремонта логических элементов комбинационных схем дает возможность удовлетворительно решить проблему автономного восстановления работоспособности цифровых систем на кристаллах за счет временной и аппаратной избыточности проекта.

Направления дальнейших научных исследований в данной области связаны с решением задач: 1) Модели коммутирования примитивов, вышедших из строя. 2) Распараллеливание вычислений по уровням элементов комбинационной схемы. 3) Замена как типов, так и примитивов структуры. 4) Создание операционного устройства или инфраструктуры для моделирования последовательных элементов и структур. 5) Моделирование схем, составленных из функционально сложных примитивов. 6) Разработка инфраструктуры встроенного тестирования, диагностирования и ремонта элементов комбинационных и последовательных устройств. 7) Тестирование и ремонт инфраструктуры сервисного обслуживания комбинационной схемы – решение проблемы «сторож над сторожем». 8) Эффективность использования инфраструктуры встроенного ремонта для сервисного обслуживания комбинационных цифровых систем с различным уровнем сложности примитивов и структуры.

Литература: 1. *Aliferis P., Brito F., DiVincenzo D. P., Preskill J., Steffen M., Terhal B. M.* Fault-tolerant computing with biased-noise superconducting qubits // *New Journal of Physics*. January 30. 2009. 19 p. 2. *Mark Gregory Whitney.* Practical Fault Tolerance for Quantum Circuits. PhD Dissertation in Computer Science. Berkeley: University of California. 2009. 206p. 3. *Хаханов В. И., Литвинова Е. И., Чумаченко С. В., Гузь О.А.* Логический ассоциативный вычислитель. Электронное моделирование. 2011. № 1. С. 73-90. 4. *Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S.* Information analysis infrastructure for diagnosis. *Information. An international interdisciplinary journal*. 2011. Japan. Vol. 14, No 7. P. 2419-2433. 5. *Хаханов В.И.* Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. Харьков: ХНУРЭ, 2009. 484с. 6. *Hahanov V.I. and others.* Infrastructure of intellectual property for SoC simulation and diagnosis service. Springer, Germany, 2011. P. 289-330. 7. *Chung J., Park J., Abraham J. A.* Built-In Repair Analyzer With Optimal Repair Rate for Word-Oriented Memories // *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, 2012. Iss. 99. P. 1–11. 8. *Mincent Lee.* A Memory Built-In Self-Repair Scheme Based on Configurable Spares / Lee Mincent, Denq Li-Ming, Wu Cheng-Wen // *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Vol. 30, Iss. 6. 2011. P. 919 – 929. 9. *Koal T. A*

comprehensive scheme for logic self repair / T. Koal, D. Scheit, H.T. Vierhaus // *Conference Proc. on Signal Processing Algorithms, Architectures, Arrangements, and Applications*. 2009. P. 13 – 18. 10. *Rab M.T.* Improving Memory Repair by Selective Row Partitioning / M.T. Rab, A.A. Bawa, N.A. Touba // *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*. 2009. P. 211 – 219. 11. *Pekmetzi K.* A BISR Architecture for Embedded Memories / K. Pekmetzi, N. Axelos, I. Sideris, N. Moshopoulos // *14th IEEE International On-Line Testing Symposium*. 2008. P. 149 – 154. 12. *Tsu-Wei Tseng.* ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SOCs / Tsu-Wei Tseng; Jin-Fu Li; Chih-Chiang Hsu // *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. Vol. 18, Iss. 6. 2010. P. 921 – 932. 13. *Sharma R.K.* Modeling and Simulation of Multi-operation Microcode-Based Built-In Self Test for Memory Fault Detection and Repair / R.K. Sharma, A. Sood // *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2010. P. 381 – 386. 14. *Oehler P.* A Modular Memory BIST for Optimized Memory Repair / P. Oehler, A. Bosio; G. Di Natale, S. Hellebrand // *14th IEEE International On-Line Testing Symposium*. 2008. P. 171 – 172. 15. *Nicolaidis M.* Design for test and reliability in ultimate CMOS / Michael Nicolaidis, Lorena Anghel, Nacer-Eddine Zergainoh, Yervant Zorian, Tanay Kamik, Keith Bowman, James Tschanz, Shih-Lien Lu, Carlos Tokunaga, Arijit Raychowdhury, Muhammad Khellah, Jaydeep Kulkarni, Vivek De, Dimiter Avresky // *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2012. P. 677 – 682. 16. *Darbinyan K.* A Robust Solution for Embedded Memory Test and Repair / K. Darbinyan, G. Harutyunyan, S. Shoukourian, V. Vardanian, Y. Zorian // *20th Asian Test Symposium (ATS)*. 2011. P. 461–462. 17. *Grigoryan H.* Generic BIST architecture for testing of content addressable memories / H. Grigoryan, G. Harutyunyan, S. Shoukourian, V. Vardanian, Y. Zorian // *IEEE 17th International On-Line Testing Symposium (IOLTS)*. 2011. P. 86 – 91. 18. *Zorian Y.* Test and reliability concerns for 3D-ICs / Y. Zorian // *IEEE 16th International On-Line Testing Symposium (IOLTS)*. 2010. P. 219 – 219.

Поступила в редколлегию 22.04.2012

Рецензент: д-р техн. наук, проф. Баркалов А.А.

Murad Ali Abbas, аспирант кафедры автоматизации проектирования вычислительной техники ХНУРЭ. Научные интересы: техническая диагностика цифровых систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Литвинова Евгения Ивановна, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: автоматизация диагностирования и встроенный ремонт компонентов цифровых систем в пакете кристаллов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-421. E-mail: kiu@kture.kharkov.ua.

Хаханова Ирина Витальевна, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и тестирование цифровых систем и сетей на кристаллах. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.