

ГЪВКАВИ ИНСТРУМЕНТИ ЗА САМОПРОВЕРКА НА ЗНАНИЯТА: ОПРЕДЕЛЯНЕ НА ТЕСТАБИЛНОСТТА НА КОМБИНАЦИОННИ И ПОСЛЕДОВАТЕЛНИ СХЕМИ

Павлинка Радойска
Технически университет, София

Резюме

Съгласно IEEE P1484 и IMSQTI са разработени стандарти за изграждане на тестове за проверка на знанията в системите за е-обучение. На базата на тях са създадени множество добре работещи тестващи системи с общо предназначение. Тези системи са насочени повече към оценяване на знанията и складиране на получените резултати и по-малко към самоконтрол и самоусъвършенстване на обучаемите. Общото им предназначение не позволява да се използват характеристиките на конкретната предметна област, позволяващи автоматично изграждане на гъвкави и многообразни задачи, с които да се подпомогнат обучаемите в овладяването на знанията. В доклада се представя приложение, отчитащо специфичните характеристики при определянето на тестабилността на комбинационни и последователни интегрални схеми. На базата на тези характеристики приложението генерира няколко типа задачи. Входните данни се генерират на случаен принцип, а проверката се прави веднага след въвеждането на отговора. При грешка проверката показва верния отговор и основни инструкции, свързани с неговото получаване. Това прави приложението подходящо за провеждане на упражнения и самопроверка на знанията. Тъй като верните отговори са вградени в приложението като логика, а не като стойност, зазубряне, преписване или крадене на отговори не е възможно. Това прави приложението подходящо за проверка и оценяване на знания в областта на тестабилността на комбинационни и последователни схеми. Този доклад се финансира по договор № 102ни013-10 / 2010-2011, ТУ-София.

Ключови думи: е-обучение, самопроверка, гъвкави инструменти, тестабилност, контролируемост, наблюдаемост, комбинационни схеми, последователни схеми.

Key words: e-learning, self assessment, flexible tool, testability, controllability, observability, combination circuit, serial circuit.

Увод

Съвременните тенденции в развитието на електронното обучение налагат създаване на възможности за самостоятелно провеждане на упражнения от страна на студентите в удобно за тях време и непосредствено оценяване на техните резултати с възможност за посочване на грешките. IMS-QTI [1, 2] е базовата спецификация, която се използва за създаване на преносими тестови въпроси. Тя е с общо предназначение. В нея се описва широка гама от типове въпроси. Създадени са възможности за генериране на гъвкави тестове. Тестовите се състоят от секции. Всяка секция обединява група въпроси. Възможно е да се зададе избор на определен брой въпроси от секция или да се зададе ред на извеждане на въпросите в една секция. В спецификацията е залегнала логика, с помощта на която може да се създават адаптивни тестове [3]. Въпреки постигнатата гъвкавост, даден въпрос, като дефиниция и отговор е един и същ при всяко негово извеждане в теста. Това създава предпоставка за преписване и наизустяване. Така създадени тестовите са подходящи за проверка на знанията, но не и за провеждане на упражнения и самообучение.

Някои симулационни среди (Cisco Packet Tracer [4, 5]) дават възможност да се заложат сложни цялостни упражнения за усъвършенстване на практическите умения на студентите. Тези упражнения могат се провеждат контролирано от средата и автоматично да се показва коя част от действията е изпълнена правилно и коя – не. Този тип упражнения са изключително полезни за студентите, но отново дефиницията на проблема и възможните стъпки за неговото решаване са едни и същи, макар и в произволна последователност.

Моята идея е да се създаде среда за автоматично генериране на еднотипни, но с различни входни параметри задачи, които респективно имат различни решения. Тези задачи дават ниска степен на повтаряемост на въпросите, при което се игнорира възможността от преписване и наизустяване и се стимулира логическото мислене и усвояване на принципите на вземане на решение у студентите.

1. Характеристики и динамичните задачи

Предимства на динамичните задачи:

1. Подходящи са за проверка на знанията. Вероятността от преписване и наизустяване е почти нулева.

2. Подходящи са за самообучение.

Позволяват студента да се учи на принципа на пробите и грешките и да коригира знанията си в процеса на решаване на задачата. Позволява се показване на междинни и крайни резултати и логиката за тяхното получаване.

3. Подходящи са за генериране на адаптивни тестове.

При адаптивните тестове следващият въпрос се избира на базата на показаните до момента знания. Тук не е необходима калибровка, за да се определи тежестта на въпросите, а от там е тяхното градиране. Тук въпросите се подреждат на базата на заложената логика, а генерирането на различни, но еднотипни задачи може да продължи до правилното усвояване на проблема.

Недостатъци:

1. задачите не могат да бъдат универсални – обвързани са с определена предметна област;
2. задачите не могат да се описват текстово и да се използват от стандартни тествани системи. Всяка задача представлява приложение - изпълним код, който се изпълнява на клиентската машина или на сървъра, в зависимост от избраното решение.

Основни изисквания към задачите, подходящи за динамично генериране:

1. да са еднозначно определени (при всяка комбинация от входни данни да имат точно едно решение);
2. да са подходящи за формално описание на алгоритъма, водещ да правилното решение;
3. да имат безкрайно или поне достатъчно голямо множество от входни данни, което да осигури разнообразие от задачи.

2. Концептуален модел

Основните стъпки при създаването и експлоатирането на динамични задачи могат да се опишат по следния начин:

1. създаване на динамичната задача – 2 подхода:
 - а. статично вграден в сорса алгоритъм;
 - б. създаване на среда за формализирано въвеждане на алгоритъма;
2. съхраняване на динамичната задача в преносим вид и в достъпно хранилище;
3. стартиране, контролирано изпълнение и оценяване;

4. обратна връзка с отчетените резултати към системата за електронно обучение.

Разглежданията до тук определят необходимостта от два самостоятелни инструмента: авторски, за създаване на типови задачи и изпълнителен, за генериране на динамичните задачи и за контрол и обратна връзка към клиента и към системата. Архитектурният модел на изпълнителния инструмент е показан на фиг.1.

Управленският модул има 3 основни задачи:

1. генериране на теста (последователно извикване на модула Задача до достигане на предварително зададения брой задачи);
2. оценка на резултата (в зависимост от създадената скала на оценяване в Административния модул);
3. изготвяне и изпращане на рапорт за получения резултат към обучаващата система и/или e-mail към преподавателя. Рапортът съдържа: описание на въпроса, верен отговор, отговор на студента и оценяване).

Модулът Задачи отговаря за генериране на една задача чрез:

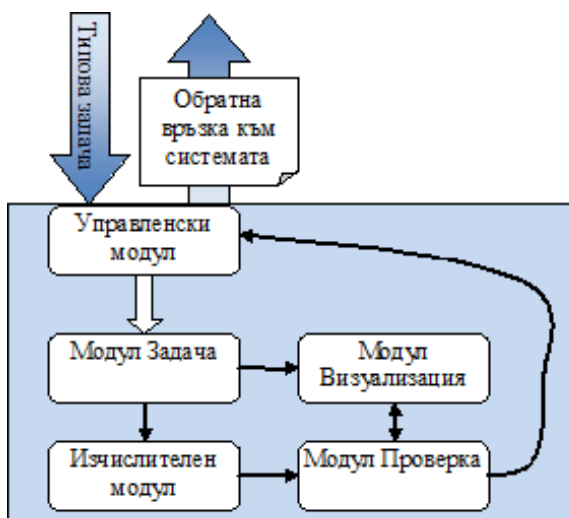
1. за всеки входен параметър генерира случайно число от диапазона на допустимите стойности;
2. асемблира условието на задачата.

Модулът Визуализация отговаря за визуализиране на задачата и осигурява интерфейс за взаимодействие със студента.

Изчислителният модул изчислява стойностите на търсените параметри на базата на заложената логика.

Модулът Проверка прави оценка на решението на задачата:

1. сравнява въведените от студента отговори с верните отговори и генерира оценка (в %);
2. взаимодейства с модула Визуализация и показва верните отговори при грешка, както и правилата за тяхното получаване;
3. изпраща на Управленския модул информация за задачата:
 - а. пореден номер;
 - б. условие;
 - в. верни отговори;
 - г. студентски отговори;
 - д. оценка в %.



Фиг. 1. Архитектурен модел на генератор на динамични задачи

3. Една реализация на динамични задачи

Една реализация на динамични задачи е създадена в областта на Тестването и диагностиката на цифрови схеми и по-специално определяне на комбинационната и последователна тестабилност [6]. По темата могат да бъдат моделирани 4 типа задачи:

1. определяне на комбинационна тестабилност на базовите градивни елементи на цифровите схеми;
2. определяне на последователната тестабилност на базовите градивни елементи на цифровите схеми;
3. определяне на комбинационна тестабилност на първичните входове и изходи на комбинационни цифрови схеми;
4. определяне на комбинационна и последователна тестабилност на първичните входове и изходи на последователни цифрови схеми.

За реализирането на типовете задачи се налага моделиране на цифровата схема и нейните градивни елементи и вграждане в модела на алгоритмите за определяне на комбинационна и последователна тестабилност, дефиниране на областите от допустими стойности за входните параметри.

При създаването на програмния модел на цифровата схема в работата са използвани две линейни програмни структури – подредени колекции. Комбинационната схема C се описва чрез две множества: това на комбинационните елементи G и това на връзките между тях (връзки) L . Състоянията на сигналите се съхра-

няват в връзките, а логиката на поведение се описва във възлите:

$$C = \{G, L\} \quad (1)$$

Множеството на възлите се представя чрез наредената колекция $g_i \in G, i = 0, \dots, n-1$, където n е броят на възлите в схемата. Първичните входове, първичните изходи и разклоненията също се описват като възли. Възлите в колекцията са наредени така, че всеки възел се появява в колекцията след всички възли, които влияят върху формирането на неговите входни сигнали. Множеството на връзките се представя чрез наредена колекция $l_j \in L, j = 0, \dots, m-1$, където m е броят на връзките в схемата. Всеки един възел се описва чрез множеството на неговите входни връзки L_{inp} , множеството на неговите изходни връзки L_{out} и логическата му функция F_{log} (фиг. 2).

$$g = \{L_{inp}, L_{out}, F_{log}\}, \quad (2)$$

където

$L_{inp} = \{l_0, \dots, l_{n-1}\}, l_j \in L, j = 0 \div n$, а n е броят на входовете на възела g . Ако възела е първичен вход, то $n = 0$ и $L_{inp} = \emptyset$.

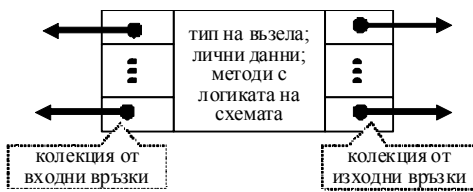
$L_{out} = \{l_0, \dots, l_{m-1}\}, l_j \in L, j = 0 \div m$ и m е броят на изходите на възела g . Ако възелът е първичен изход, то $m = 0$ и $L_{out} = \emptyset$. Само за разклоненията $m > 1$. За всички останали възли $m = 1$.

Всяка една връзка между възлите $l_i \in L$ се описва чрез адресите на предходния ($g_{before} \in G$) и следващия възел ($g_{after} \in G$), състоянието на сигнала $level$ и параметрите, носещи количествената оценка на тестабилността: CC0 (комбинационна контролируемост по отношение на 0), CC1 (комбинационна контролируемост по отношение на 1), CO (комбинационна наблюдаемост), SC0 (последователна контролируемост по отношение на 0), SC1 (последователна контролируемост по отношение на 1), SO (последователна наблюдаемост).

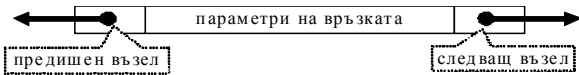
$$l_i = \{g_{before}, g_{after}, level, CC0, CC1, CO, SC0, SC1, SO\} \quad (3)$$

Моделите на един възел и на една връзка от колекциите, описващи схемата, са показани съответно на фиг. 2 и фиг. 3.

Двупосочните връзки между възлите, които се пазят в описанието на връзките, са предпоставка за извършване както на прави, така и на обратни обхождания.



Фиг. 2. Модел на възел



Фиг. 3. Модел на връзка

Изводи и препоръки

В настоящия вариант инструментът е създаден като самостоятелно приложение. Първите две типове задачи са разработени напълно. Дефинирани са 9 типа възли: базови комбинационни елементи (NOT, AND, NAND, OR, NOR, XOR, NXOR), D-тригер и разклонение. За многоходовите възли (AND, NAND, OR, NOR) броят на входовете е ограничен до 10. Броят на изходите при разклоненията също са ограничени до 10. Максималните стойности на контролируемостите и наблюдаемостите са ограничени до 20. Това прави повече от 56 милиона различни реални задачи за всяка от типовете задачи. В представеното приложение типовете задачи от първите 2 типа са разделени на по 9 самостоятелни типове задачи за всеки възел по една, за да се гарантира усвояване на принципите за определяне на тестобилността при всеки тип възли.

За нуждите на третия и четвъртия тип задачи е създаден парсер, който чете текстово описание на схемата от ISCAS-85 формат и го преобразува до програмния модел на схемата.

Заради разработката на инструмента като самостоятелно приложение, се работи само с твърдо зададени и вградени в кода описания на цифрови схеми. Текста на задачите се генерира по време на изпълнение на задачата, но графичните изображения, необходими за онаг-

ледяване на задачата също са твърдо вградени в кода.

Предстои инструментът да се развива в две посоки.

1. Създаване на отворено приложение чрез взаимодействие с база от данни, в която да се съхраняват описанията на цифровите схеми и съответните графични изображения, с цел да се създаде отворен, свободен за разширяване инструмент.
2. Унифициране на инструментът. Създаване на авторски модул, формален език за описание на типовата задача и интерпретатор, така че системата да позволи създаване на нови типове задачи, които не са вградени в кода на приложението, а се съхраняват в база от данни и могат да се пренасят в интернет пространството.

Литература

1. *IMS Question & Test Interoperability Specification*. IMS QTIv2.1. <http://www.imslobal.org/question/>.
2. Constandache, I., Zamfir, C., Udrea, O. *Implementing the IMS QTI specification: an architectural overview*. E-Business, E-Government, E-Learning, 2004. <http://jupiter.ksi.edu/~changsk/chronobot/ref-scorm.pdf>.
3. Радойска, П., Атанасова Г. *Мястото и ролята на адаптивните тестове в дистанционното обучение по чужди езици*, Четвърта Международна научно-практическа конференция „Съвременни измерения във дистанционното обучение“, Плевен, 2007.
4. *Cisco Packet Tracer*. http://www.cisco.com/web/learning/netacad/course_catalog/docs/Cisco_PacketTracer_DS.pdf.
5. *New Tools to Teach CCNA: Packet Tracer v4.0*. ftp://64.107.8.176/rsl/AC_PacketTracer4_2.pdf.
6. *Bushnell, M., Agrawal, V., Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. 2005.

FLEXIBLE TOOLS FOR SELF ASSESSMENTS: OBSERVATION THE TESTABILITY FOR THE COMBINATION AND SERIAL CIRCUITS

Pavlinka Radoyska
Technical University, Sofia, Bulgaria

Abstract

In the IEEE P1484 and IMSQTI are described specifications and standards for e-learning assessments and tests questions. Many assessment tools are developed based on these specification and standards. But these

systems are focused mainly on assessing and collecting the assessment result. They are not focused on self test and improvement the knowledge level. Moreover, their aims are to be used in wide area of learning, because of that these tools are enough general. They do not allow using the characteristic in the specific area of learning to perform more flexible and attractive assessments and task and in this case facilitate the students in their learning. In this paper I offer an application, designed especially for area of combination and serial circuit testing. The application is consistent with the nature of a specific group of tasks: controllability and observability calculations. Owing to this the application has flexibility to generate non repeated tasks in different order. The input data is generated randomly. The number of the tasks of the same type is dependent of the student responses. The revision makes immediately after inserting the answers. The true answer is presented if the answer is wrong. This mechanism makes the application suitable for the self control and for the exercises. The true answers are built in the application as logic, not as a value. Therefore the swopping, cheating or stealing the answers is impossible, which is not true for the conventional (standardized) question types. So the application is suitable also for assessment the students' grade.