ЧАСТИНА 3. ЯКІСТЬ ІНФОРМАЦІЙНИХ СИСТЕМ

Г.В. Табунщик,Т.І. Каплієнко

ВСТУП ДО ЧАСТИНИ 3

На всіх етапах життєвого циклу складних технічних об'єктів питання верифікації є з найважливіших.

Сучасні інформаційні системи (ІС) це складні програмно-апаратні комплекси, що вимагають всебічну верифікацію, але існуючи підходи не орієнтуються на одночасну верифікацію як програмного так і апаратного забезпечення складної вбудованої системи.

Для автоматизації процесу верифікації інформаційних систем необхідно мати формалізовану модель, тому розділ 7 містить запропоновану авторами формалізовану модель верифікації інформаційних систем.

Ефективним інструментом верифікації інформаційних систем є тестування, що дозволяє провести всебічну верифікацію всіх частин системи де були внесені зміни. Розділ 8 містить методи тестування програмного та апаратного забезпечення вбудованих систем, зокрема авторський модифікований метод регресійного тестування, що дозволяє автоматизувати процес формування взаємозалежних компонент при повторному тестуванні інформаційної системи.

Дев'ятий розділ присвячений прикладам використання віддаленої лабораторії GOLDi при вивченні методів тестування вбудованих систем.

For all stages of lifecycle of compound technical objects the questions of verification are highly important.

Concurrent informational systems are compound systems consistes from software and hardware but existed approaches don't concider complex verification of hardware and software of the embedded systems.

For complex automatization of the informational system its important to have formalized model for objects included in the system and description of the processes of data transmition, so the chapter 7 consists developed by the authors the model of verification.

One of the most effective instruments of verification is testing, which allow to provide verification of all modificated parts of the developing system. Chapter 8 depicts software and hardware testing techniques, including modificated method for regression testing, which allow to form automatevly list of connected compotents of the system within the repeated testing of the informational system.

Chapter 9 devoted for examples of online lab GOLDi developed by Integrated Communication Systems Group at the Ilmenau University of Technology usage for the tasks of teaching of the embedded systems testing teqniques.

7 ОСНОВИ ВЕРИФІКАЦІЇ ВБУДОВАНИХ СИСТЕМ

Верифікація системи в найзагальнішому сенсі – це перевірка відповідності між вимогами до системи і властивостями працюючої системи [13].

7.1 Методи верифікації вбудованих систем

Розрізняють такі методи верифікації як експертиза, статичний аналіз, формальні методи, динамічні методи, синтетичні методи (рис. 9.1) [13]. Зазвичай серед експертиз виділяють такі: організаційні експертизи (management review), технічні експертизи (technical review), наскрізний контроль (walkthrough), інспекції (inspection) та аудити (audit).

З середини 1990-х активно розвиваються методи оцінки архітектури програмного забезпечення (ПЗ) на основі сценаріїв (scenario based software architecture evaluation), які зазвичай не співвідносяться з «традиційними» експертизами. Від інших методів верифікації експертизу відрізняє можливість виконувати її, використовуючи тільки самі артефакти життєвого циклу, а не їх моделі (як у формальних методах) або результати роботи (як в динамічних). Експертиза застосовується до будь-яких характеристик ПЗ і до будь-яких артефактів життєвого циклу на всіх етапах життєвого циклу, хоча для різних цілей можуть використовуватися різні її види. Вона дозволяє виявляти широкий спектр помилок, причому робити це на етапі проектування відповідного артефакту, тим самим мінімізуючи час існування дефекту і його наслідки для якості похідних артефактів.

У той же час експертиза не може бути автоматизована і вимагає активної участі людей.

Рисунок 9.1. Методи верифікації

Ефективність експертизи істотно залежить від досвіду та мотивації її учасників, організації процесу, а також від забезпечення коректної взаємодії між різними учасниками. Це накладає додаткові обмеження на розподіл ресурсів у проекті і може призводити до конфліктів між розробниками, якщо керівництво проекту звертає мало уваги на комунікативні аспекти проведення експертиз.

Кожний вид експертизи має свої властивості:

- технічна експертиза (*technical review*) – це систематичний аналіз артефактів проекту кваліфікованими фахівцями для оцінки їх внутрішньої узгодженості, точності, повноти, відповідності стандартам і прийнятим в організації процесів, а також відповідності один одному і загальним завданням проекту;
- наскрізний контроль (*walkthrough*) – метод експертизи, в рамках якого один з членів команди перевірки надає її учасникам послідовно всі характеристики артефакту, що перевіряється, і вони аналізують цей артефакт, ставлячи питання, вносячи зауваження, відзначаючи можливі помилки, порушення стандартів і інші дефекти;

- інспекція (*software inspection*) – це послідовне вивчення характеристик артефакту, зазвичай слідує деякому плану, з метою виявлення в ньому помилок і дефектів;
- аудит (*audit*) – аналіз артефактів і процесів життєвого циклу, що виконується людьми, які не входять до команду проекту, для оцінки відповідності цих артефактів і процесів завданням проекту, укладеним контрактом, загальним стандартам, один одному і ін.

Статичний аналіз використовується для перевірки формалізованих правил коректної побудови артефактів ІС і пошуку дефектів за визначеними шаблонами.

Такий аналіз добре автоматизується і може бути практично повністю покладений на програмний інструментарій, хоча іноді необхідно вручну визначити, наприклад, прийняті в проекті стандарти кодування. Однак він застосовується лише до коду або до певних форматів уявлення проектних артефактів ІС, і здатний виявляти тільки обмежену кількість типів помилок. Проблемою цих методів є або велика надмірність, при використанні методів строго аналізу, або вірогідність пропустити помилки, при використані методів що ще генерують повідомлення про помилку. Інструменти автоматичної верифікації на основі статичного аналізу застосовуються досить широко, оскільки не вимагають спеціальної підготовки і досить зручні у використанні. Більшість технік статичної перевірки коректності програм, які довели свою ефективність на практиці, рано чи пізно стають частиною компіляторів або навіть перетворюються в семантичні правила мов програмування.

На практиці, формальні методи верифікації набагато частіше застосовуються до апаратного забезпечення ніж до програмного. Це обумовлено більш високою вартістю помилок, більшою однорідністю його структури, більш широким багаторазовим використанням проектної інформації, а також більшою звичністю строгих обмежень і точних описів для інженерів.

Логіко-алгебраїчні моделі (*property-based models*), вони ж – логічні або алгебраїчні обчислення. При моделюванні ПЗ ІС модель такого типу описує деякий набір властивостей системи, що змінюється з часом, але не дає точного уявлення про те, за рахунок чого змінюються ці властивості.

Здійснимі моделі (або операційні, *executable models*) характери-

зуються тим, що їх можна визначеним способом виконати, щоб простежити зміну властивостей модельованого ПЗ. Кожна модель, що виконується, є, по суті, програмою для деякої досить строго визначеної віртуальної машини.

Всі види здійсненних моделей можна вважати розширенням і узагальненням моделей на базі кінцевих автоматів.

Моделі проміжного типу мають риси як логіко-алгебраїчних, так і здійсненних моделей. Частина наведених вище прикладів може по ряду властивостей бути віднесена до обох цих класів.

Методи та інструменти перевірки моделей. Перевірка моделей (*model checking*) використовується для перевірки виконання набору властивостей, записаних у вигляді тверджень визначеного логіко-алгебраіческого обчислення за здійсненною моделлю що моделює певні проектні рішення. Найчастіше для перевірки опису властивостей використовується деяка тимчасова логіка або – числення, а в якості моделі, властивості якої перевіряються, виступає кінцевий автомат, стани якого відповідають наборам значень елементарних формул у властивостях, що перевіряється, зазвичай він називається моделлю Кріпке. Перевірку моделі виконує спеціалізований інструмент, який або підтверджує, що модель дійсно володіє заданими властивостями, або видає сценарій її роботи, в кінці якого ці властивості порушуються, або не може прийти до певного вердикту, оскільки аналіз моделі вимагає занадто великих ресурсів. Властивості, що перевіряються, зазвичай поділяють на властивості безпеки (*safety properties*), які перевіряють умову, що щось небажане при будь-якому варіанті роботи системи ніколи не трапиться, і властивості живучості (*liveness properties*), які, навпаки, перевіряють умову, що щось бажане рано чи пізно відбудеться. Іноді додатково виділяють властивості стабільності (або збереження, *persistence properties*) – при будь-якому сценарії роботи системи задане твердження в деякий момент стає істинним і з тих пір залишається виконаним, і властивості відповідності (*fairness properties*) – деяке твердження при будь-якому сценарії роботи буде виконано в нескінченній множині моментів часу. Ті чи інші властивості відповідності, часто є вихідними припущеннями, при виконанні яких потрібно перевірити властивості безпеки або живучості.

Методи та інструменти перевірки узгодженості. При перевірці узгодженості аналізується відповідність між двома здійснимими моделями, одна з яких моделює артефакт, що перевіряється, зазвичай проект або реальну роботу системи (її компонента), а друга – перевіряються властивості. Перевіряються властивості в цьому випадку – це вимоги до поведінки системи або її компонента, представлені у вигляді узагальненого автомата (системи переходів, мережі Петрі та ін.), Всі сценарії роботи якого оголошуються правильними. В цьому випадку зазвичай перевіряється умова, що всі можливі сценарії поведінки реалізації можливі також і в специфікації. Іноді встановлюється їх еквівалентність, тобто додатково перевіряється, що всі сценарії поведінки специфікації є і у реалізації. Більшість методів та інструментів перевірки узгодженості використовують для цього тестування, і тому відносяться до синтетичних методів верифікації – до тестування на основі моделей.

Динамічні методи верифікації, в рамках яких аналіз і оцінка властивостей програмної системи виконуються за результатами її реальної роботи або роботи деяких її моделей і прототипів. Прикладами такого роду методів є звичайне тестування або імітаційне тестування, моніторинг. Для застосування динамічних методів необхідно мати працюючу систему, або хоча б деякі її компоненти, чи їхні прототипи, тому зазвичай вони не використовуються на перших стадіях розроблення. З допомогою цих методів можна контролювати характеристики роботи ІС в її реальному оточенні, коли інші підходи використати неможливо. Методи динамічної верифікації дозволяють виявляти тільки помилки, що проявляються при її роботі. Застосування динамічних методів верифікації зазвичай вимагає додаткової кваліфікації для створення тестів, розробки тестової системи, що дозволяє їх виконувати, або системи моніторингу, яка дозволяє контролювати певні характеристики поведінки ІС. Але системи тестування або моніторингу можуть бути зроблені один раз і використовуватися багаторазово для широких класів ІС, лише самі тести необхідно готувати заново для кожної тестуємої ІС. У той же час підготовка тестів на ранніх етапах створення ІС дозволяє виявити безліч дефектів в описі вимог і проектних документах – фактично, розробники тестів змушені в ході своєї діяльності виконувати експертизу артефактів, які є основою для тестів. Створення тестових наборів, які дозволять отримати адекватну оцінку якості складної системи, є досить трудомістким завданням. Оскільки тестування це досить трудомісткий процес, тому зазвичай використовуються

не надто надійні, але досить дешеві техніки, такі як (нестроге) вірогідне тестування, при якому тестові дані генеруються випадковим чином, або ж тестування на основі найпростіших сценаріїв використання, що перевіряють лише найбільш прості ситуації.

Відокремились динамічні методи, що використовують елементи формальних, – до них відносяться тестування на основі моделей (*model-based testing, model driven testing*) і моніторинг формальних властивостей (*runtime verification, passive testing*). Ряд інструментів побудови тестів істотно використовує як формалізацію деяких властивостей ПЗ, так і статичний аналіз коду. Загальна ідея таких методів міститься в наступному – поєднувати переваги основних підходів до верифікації для балансування їх недоліків.

Тестування на основі моделей (*model based testing*) використовує для побудови тестів формальні моделі вимог до ІС. Як критерії повноти тестування, так інші критерії будуються на основі інформації, що міститься в цих моделях. Отримані в результаті тести зазвичай слабко пов'язані зі специфічними особливостями коду тестованої системи, але містять репрезентативну вибірку ситуацій з точки зору вихідної моделі.

В даний час методи тестування на основі моделей використовують такі типи моделей і технік як методи перевірки узгодженості автоматів і систем переходів. Такі методи відносяться до одного з трьох типів, залежно від моделей, що використовуються.

Методи побудови тестів на основі кінцевих автоматів найбільш глибоко розроблені, відомі їхні точні обмеження і гарантії повноти виконуваних перевірок. Методи, що використовують розширені автомати, зводять їх до звичайних, але застосовують більш детальні критерії покриття, засновані на використанні даних в розширених автоматах.

Системи переходів. Такі методи частіше використовуються при тестуванні розподілених систем, оскільки моделювання таких систем за допомогою кінцевих автоматів є дуже трудомістким. Більшість цих методів не визначають практично застосовних критеріїв повноти і не дають кінцевих тестових наборів для реальних систем, тому інструменти, що їх використовують, спираються на певні евристики для забезпечення завершеності набору тестів.

Методи побудови тестів на основі формального аналізу властивостей ПЗ використовують формальний аналіз для класифікації тестових ситуацій і націленої генерації тестів.

В рамках методів на основі перевірки моделей тестові ситуації вибираються як представники класів еквівалентності, що задаються критерієм покриття. Відповідна ситуації тестова послідовність будується як контр приклад при перевірці моделі (model checking) на виконання властивості, що є запереченням умови досягнення цієї ситуації.

Методи на основі дедуктивного аналізу. Обирають тестові ситуації, що відповідають особливим випадкам в дедуктивному аналізі властивостей тестованої системи.

7.2 Формалізована модель верифікації інформаційних систем

Для автоматизації комбінованих методів верифікації ІС авторами запропонована формалізована модель верифікації [19], що дозволяє визначити основні концепти системи.

Модель домену в запропонованій моделі визначає концептуальну основу та семантику змісту інформаційної системи. Модель домену – це набір

$$DM = (C, D, RC, RD),$$

де $C = CC \cup CA$ – набір понять (композиційних та атомарних), D – набір екземплярів концепції домену, RC – набір відносин між концепціями, RD – набір відносин між екземплярами концепції.

Елемент ІС *ed* – це будь-який ресурс, доступний по мережі і ідентифікований унікальним URL. ed визначається функцією:

$$rd: CA \rightarrow Ed | \forall c \in CA: ed = rd(c), ed \in Ed,$$

де CA – набір атомарних концепцій, Ed – набір ресурсів програмного комплексу (ПК) ІС, тобто сукупність інтегрованих програмно-апаратних та технічних засобів, а також інформації, призначеної для публікації в мережі Інтернет, та яка відображається в певній текстовій, графічній або звуковій формах.

Модель вимог користувача має спеціальну архітектуру, яка була складена з вимог до типів збереженої інформації та необхідних методів її обробки. Розділимо дані користувача на дві частини – профіль користувача і неявні вимоги користувача.

Явні вимоги (профіль) користувача *u* – набір *UP (u) = (A, V, rp)*

$$rp: A \to V \mid \forall a \in A: rp\ (a) \in Va,$$

де *A* – набір атрибутів, визначених в якості словника характеристик користувача, $V = \bigcup_{a \in A} V_a$ – набір значень атрибута і *Va* – діапазон атрибуту a.

Неявні вимоги користувача в даній термінології позначають модель, що містить неявні характеристики користувача. У той час як явні характеристики встановлюються самим користувачем, неявні характеристики визначаються адаптивною системою. Система збирає різні значення, які стосуються конкретного об'єкта домена.

Неявні вимоги користувача *u* – це набір *UM (u) = (D, A, V, rm)*

$$rm: D \times A \to V \mid \forall a \in D \times A: rm\ (a) \in Va,$$

де *D* – набір екземплярів концепції домена;

A – набір атрибутів, визначених в якості словника неявних користувальницьких характеристик;

$V = \bigcup_{a \in A} V_a$ – набір значень атрибута;

Va – діапазон атрибуту a.

Сукупність явних *UP (u)* та неявних *UM (u)* вимог користувача, виражених у вигляді множини елементів прототипу інтерфейсу, являють собою початкову версію ІС, яка використовується для демонстрації концепцій, закладених в системі, перевірки варіантів вимог, а також пошуку проблем:

$$interprototype = (A, D, V, rp, rm). \qquad (7.1)$$

Адаптивна функція *fa* є перетворенням між елементами ІС та вимог користувача, вираженими у вигляді прототипу. Елемент ІС, наприклад може вважатися частиною коду, який в свою чергу є частиною підсистеми:

$$fa: ea \to e,$$

де *fa AF*,

ea – елементарна одиниця interprototype ПК ІС, *ea {rp (a), rm (a)};*

e= {ed} – підмножина ресурсів ПК ІС, $e \subset E_d$, *ed* – елемент ІС, *Ed* – набір web-ресурсів.

Ефективність функціонування ІС може бути оцінена такими найбільш важливими критеріями як відповідність вимогам користувача та достовірність інформації, тому в формальному вигляді постановка задачі оцінки ефективності може бути представлена наступним чином:

$$\begin{cases} interprototype \to E_d \\ DB \subset (DB_1 \cap DB_2 \cap ... \cap DB_N), \end{cases} \qquad (7.2)$$

де interprototype – об'єднана множина явних та неявних вимог користувача до функціональності ПК ІС;

E_d = *{ed}* – існуюча функціональність ПК ІС, виражена множиною елементів ІС;

$DB_1, DB_2, ..., DB_N$ – сховища даних ІС у разі використання поширення даних або федеративного підходу до інтеграції даних (дані підходи частіше використовуються у зв'язку з можливістю отримувати найбільш актуальну інформацію, без затримок в оновленні, як у випадку з консолідацією даних). Кожна *DB* містить в собі множину D.

Розроблену модель будемо використовувати для наступних завдань:

- враховуючи те, що вимоги користувача найнаочніше можуть бути представлені у вигляді прототипу користувача інтерфейсу, виникає необхідність у засобах порівняння розроблених функціональних елементів ІС та елементів прототипу:

$$interprototype \equiv E_d.$$

- для обліку невизначеності розроблюваної ІС та висунутих до неї вимог необхідні методи визначення ризиків розроблення ІС, а також методів спрямованих на їх зниження з урахуванням специфіки розробляється ІС та моделі її розроблення;

- для контролю відповідності даних, що зберігаються в кількох БД, необхідний організований процес тестування інтегрованих БД, що дозволяє забезпечити достовірність та актуальність даних при мінімальних тимчасових витратах.

Будемо розглядати $f = \bigcup_{d \in D} \{e_d, p\}$, $e_d \in E_d$, $f \in F$, p – варіант тестування для f, набір з одного або декількох елементів ІС, як «функціональну одиницю системи» – елементарну структурну складову ІС, що реалізує закінчений функціональний блок, для перевірки якого може бути розроблений один або більше автоматизованих або автоматичних перевірочних тестів:

$$F = (E_d, P, r)\big|_{\forall\ e_d},\ e_d \in Ed, \qquad (7.2)$$

$$\text{rf: ed} \to \text{p}.$$

Функціональну одиницю (ФО) розробки необхідно вибирати залежно від виду ІС. Наприклад, якщо розробляється ІС вимагає зв'язку з БД або є об'єднанням різних апаратних платформ, то ФЕ це функція обміну інформацією між вказаними компонентами.

На відміну від функціональної точки використання при оцінюванні поняття ФО дає можливість використовувати більш узагальнений функціональний блок, характерний для конкретного типу ІС, що дозволяє спростити процедуру планування та тестування цих елементів.

Враховуючи зазначені переваги використання ФО і відсутність її в існуючих моделях, модель верифікації MV отримає вигляд:

$$MV = (interprototype, Ed, DB, rmv)$$

$$rmv: interprototype \rightarrow Ed \times DB \mid \forall\ ea, ea \in \{rp\ (a), rm\ (a)\}, ea \rightarrow e,$$

$$e = \{ed\},$$

$$e_d \in E_d,\ E_d \in W_{res},\ DB \in W_{res}$$

8 МЕТОДИ ТЕСТУВАННЯ ВБУДОВАНИХ СИСТЕМ

Вбудовані системи це складні системи, що можуть об'єднувати різноманітні архітектури. Крім того більшість вбудованих систем – це системи реального часу тому коректність функціонування залежить не тільки від логічної достовірності але й від часу коли отриманий результат.

Для більшості вбудованих систем проектування апаратних платформ та програмного забезпечення це відокремлені процеси. Але для діагностування систем існує істотний зв'язок між програмно-апаратним забезпеченням вбудованих систем. Відмови системи можуть виникнути у зв'язку з дефектами як програмного так и апаратного забезпечення. Але більшість підходів не розглядає тестування на системному рівні і рівні реалізації як єдине ціле. Тому цей розділ присвячений як методам тестування апаратного так і програмного забезпечення.

8.1 Загальні поняття та визначення

Тест (test) являє собою набір операцій, призначених для отримання одного або більшої кількості очікуваних результатів в певній програмній системі. Якщо отримані всі очікувані результати, вважається, що тест пройдено (тобто тест виконано успішно). Якщо фактичний результат відрізняється від очікуваного, вважається, що тест не пройдено (тобто тест завершився невдало).

Перше, що слід відзначити в наведеному визначенні, так це те, що кожен тест складається з двох компонентів: сукупність виконуваних тестувальником дій та послідовність подій, що повинні відбутися в результаті цих дій.

Тестові дії в сукупності утворюють методику тестування. Послідовність подій, що відбуваються в результаті цих дій, називаються очікуваними результатами. Щоб тест був ефективним, повинні бути чітко й однозначно визначені як методика, так і очікувані результати.

По-друге, якщо методика тестування та очікувані результати визначені правильно, тест повинен давати результат, за яким можна зробити однозначний висновок щодо успіху або невдачі випробування. При введенні в програму двох чисел з метою отримання їх суми тест вважається пройденим, якщо на виході програми буде отриманий коректний результат, інакше тест розглядається як не пройдений.

Варіант тестування (тестовий сценарій) – це опис початкових умов, вхідних даних, дій користувача і очікуваного результату (рис. 8.1). Гарною практикою вважається використовувати прецеденти в якості варіанта тестування.

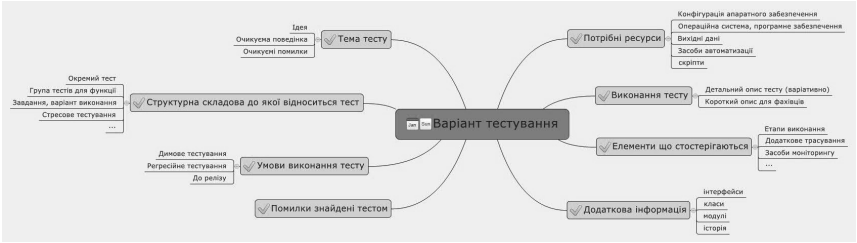

Рисунок 8.1. Варіант тестування

З варіантом тестування повинні бути пов'язані такі ознаки:

- він повинен володіти високою імовірністю виявлення дефекту;
- він повинен бути відтворюваним;
- він повинен володіти чітко визначеними очікуваними результатами і критеріями успішного або невдалого виконання тесту;

- він не повинен бути надмірним.

Мета тестування полягає в знаходженні дефектів.

Дефект – це певна невідповідність продукції вимогам, встановленим нормативно-технічною документацією;

Брак – це дефектна одиниця продукції, тобто продукція, що має хоча б один дефект.

Необхідно розрізняти дефекти апаратного забезпечення та помилки програмного забезпечення.

Дефект апаратного забезпечення [11] це різниця між реалізованим обладнанням та його проектом. Це може бути дефекти процесу виробництва, дефекти матеріалу, вікові дефекти або дефекти пакування.

Помилкою називається неправильний вихідний сигнал дефектної системи. Помилка це «ефект», що призводиться визначеним «дефектом». Помилки викликають відмови системи, тобто відхилення від відповідної поведінки. Якщо відмова може призвести до аварійної ситуації, вона являє собою ризик.

Під відмовою (несправністю) будемо вважати уявлення про «дефект» на рівні абстракції, будемо називати його несправністю. Несправності, це фізичні або логічні дефекти в конструкції або реалізації пристрою.

8.2 Моделі відмов апаратного забезпечення

З розвитком технологій складність та потужність апаратних засобів збільшується. Модель повинна допомогти визначити цільову функцію для тестування та аналізу відмов. Крім того, ефективність моделі відносно фактичних відмов визначаються експериментально. В більшості випадків несправності цифрових систем можна розділити на три групи [11]: помилки проектної документації, помилки виробництва та експлуатаційні відмови. Помилки проектування виникають завдяки людському фактору або помилкам програмного забезпечення САПР (симулятори, генератори та генераторів шарів) і відбуваються в процесі проектування. Ці відмови не пов’язані безпосередньо з процесом тестування. Відмови, що обумовлені недосконалістю процесів виробництва, призводить до дефектів самого обладнання (пропущені при металізації контактні вікна або ділянки оксиду, паразитичні транзистори, пробій оксиду (в МОП-структурах) і т.п.). Фізичні відмови також називаються дефект-орієнтованими відмовами (*defect-oriented faults*). Експлуатаційні або логічні відмови відбуваються через відхилення в умовах експлуатації вбудованих систем. Прикладом таких відхилень виступають електромагнитні перешкоди, помилки оператора, вібрації.

Несправність є моделлю, яка є ефектом фізичного дефекту на логічному або функціональному рівні. Зазначимо, що кілька різних дефектів можуть представлятися однією і тією ж несправністю (має місце відношення «багато до одного»). З іншого боку, одному фізичному дефекту іноді може відповідати кілька несправностей (відношення «один до багатьох»).

Апаратні несправності класифікуються як константні відмови, наприклад, несправності типу замикання, дефекти обриву ланцюга, збоїв пам’яті та інші.

Крім того, їх можна класифікувати як структурні та функціональні. Ті несправності, що визначаються на структурній моделі системи, називаються структурними несправностями. Їх ефект, як правило, зводиться до зміни з’єднань компонент. Функціональні несправності визначаються на функціональної моделі системи. Наприклад, ефект функціональної несправності може проявлятися у зміні функції, реалізованої компоненти системи або оператором мови опису апаратури.

Типовими несправностями сполучень компонентів системи є обрив (open) і замикання (short). Обрив відповідає порушенню з’єднання компонентів схеми. Причиною порушення з’єднання може бути нестача або відсутність провідного матеріалу, наприклад, в металевому провіднику. З іншого боку, відсутність з’єднання може виникнути внаслідок наявності зайвих частинок діелектрика, наприклад, між провідними шарами. Замикання утворюється в результаті з’єднання ліній схеми, які в справній системі ізольовані один від одного. Воно може бути викликано наявністю зайвих провідних частинок між провідниками, пробоєм оксиду в МОП-структурах, який утворює з’єднання з деяким невеликим, але необов’язково нульовим опором і т.п.

Константні відмови (Stuck-at fault). Для визначення константних несправностей використовується структурна модель у вигляді

логічної схеми. Вважається, що одиночна константна несправність (single stack-at fault) діє тільки на з'єднання між вентилями (при цьому логічні елементи функціонують правильно). Кожна лінія схеми може мати два типи цих несправностей: константа 0 і константа 1 (sa-0, sa-1). Отже, константна несправність фіксує на даній лінії постійне значення сигналу 0 або 1 (sa-0, sa-1), незалежно від значення сигналу, що подається на неї. Часто такі несправності моделюють замикання лінії схеми на землю (sa-0) або джерело живлення (sa-1).

Ця модель може використовуватися для генерації тестів незалежно від її адекватності реальних фізичних дефектів. Відзначимо, що для деяких технологій адекватність цієї моделі досить висока, для інших нижче. Але в цілому, дана модель надзвичайно корисна в силу своєї виняткової простоти і задовільної адекватності, і тому використовується в якості базової для багатьох методів моделювання несправностей і генерації тестів.

Несправності типу замикання мають місце в тому випадку, коли відбувається з'єднання двох або більше ліній схеми і утворюється «дротова логіка» (wired logic) в місці виниклої електричного зв'язку. Кратні замикання (з'єднання більше двох ліній) виникають звичайно на зовнішніх входах ІС. В певний час кількість дефектів, провідних до замикань, збільшується внаслідок зменшення розмірів схем і збільшення щільності вентилів у кристалі. Очевидно, що кількість простих замикань (між двома лініями) в схемі, що має m ліній одно. Однак, звичайно, не всі лінії схеми можуть замкнутися між собою. Тому реальна кількість можливих замикань істотно менше і залежить від фізичного сусідства провідників.

Поведінка логічної схеми при замиканні залежить від технології виготовлення цієї схеми.

Слід зазначити, що дефекти замикання можуть викликати функціональні зміни в логічній схемі, які не можна уявити традиційними моделями – несправностями.

Деякі фізичні дефекти не можуть бути представлені константними несправностями. Основна причина полягає в тому, що МОП комбінаційні схеми не завжди залишаються комбінаційними при деяких фізичних дефектах. Найбільш поширеними є такі види відмов у МОН технології: 1) обрив і замикання транзисторів; 2) обриви між стоком, витоком і затвором; 3) короткі замикання: витік – стік, затвор – стік, затвор – витік. Дефекти третьої групи зазвичай обумовлені пробоєм оксиду.

У сучасних цифрових системах можливі ситуації, коли схема структурно і логічно коректна, але час поширення сигналів по деяким її шляхах перевищує допустиме для правильного функціонування значення. У таких випадках говорять про наявність несправності типу «затримка» (поширення сигналів). Такі несправності не можуть бути виявлені на низькій частоті роботи схеми. Метою тестування несправностей «затримка» є визначення правильного функціонування схеми на високих тактових робочих частотах. При цьому виявляється, чи містить схема шляху поширення сигналів, які є занадто повільними або швидкими при зміні вхідних наборів. Для цих цілей використовуються дві основні моделі: затримка вентиля та затримка шляху.

Перша модель передбачає, що затримка обумовлена несправним логічним елементом. Слід зазначити, що час перемикання елемента, може залежати від напрямку зміни сигналу – його підйому або спаду. Це є недоліком даної моделі, оскільки не дозволяє у затримці одного елемента врахувати затримки інших елементів шляху. Очевидно, тут також повністю ігноруються затримки з'єднань між елементами.

Друга модель бере до уваги загальну затримку поширення сигналу від зовнішнього входу до зовнішнього виходу схеми. Хоча дана модель вимагає розгляду занадто багатьох можливих шляхів у схемі, вона більш реалістична, особливо для сучасних технологій, де затримки поширення сигналів мають місце насамперед за рахунок ліній з'єднань. Як правило, тестування затримок проводиться шляхом подачі на схему пари вхідних наборів на бажаної швидкості та спостереженні для кожного зміненого виходу швидкості його перемикання.

Часові несправності. При даних несправностях відбувається тимчасово поява неправильних сигналів в схемі. Вони зустрічаються в різних цифрових елементах, але найчастіше в мікросхемах пам'яті і мікропроцесорів. Серед цих несправностей розрізняють «короткочасні» (transient) «відмови» intermittent. Короткочасні несправності відбуваються, коли сигнали змінюють своє значення внаслідок, наприклад, шумів. Такі несправності важко виявити та виправити. Тут важливо мінімізувати шуми і підвищити перешкодозахищеність

схеми. Дані несправності можуть бути викликані, наприклад, флуктуаціями напруги або космічним випромінюванням.

Аварії є однією з причин відмов при експлуатації комп'ютерних систем. Серед них можна виділити несправності залежні від коду, які зустрічаються в мікросхемах пам'яті і мікропроцесорах.

Несправності рівня кристалу. В даний час нові технології дозволяють проектувати складні цифрові системи (ЦС) на одному кристалі (System-on-Chip – SOC). При цьому проектування ЦС виконується за допомогою досить складних мов високого рівня опису апаратури (HDL), таких як VHDL, Verilog і SpecC. Тому актуальною є проблема верифікації та тестування складних ЦС, описаних за допомогою цих мов.

На логічному рівні моделювання, де ЦС представляється у вигляді логічної схеми, основною моделлю фізичних дефектів є константні несправності, які еквівалентні постійним сигналам 0 або 1 на лініях схеми. На відміну від логічного рівня моделей несправностей, де зазвичай можна встановити відповідність між фізичним дефектом провідників у кремнії і з'єднаннями в логічній схемі, на поведінковому рівні, як правило, важко встановити відповідність між описом ЦС на HDL і структурним описом. Один оператор HDL може відповідати сотням логічних вентилів, з'єднаних між собою. Тому необхідно розглядати функціональні моделі несправностей безпосередньо на мовних конструкціях HDL. При цьому якість (або адекватність) функціональних моделей несправностей, як правило, перевіряється за допомогою логічного моделювання ЦС, яке визначає повноту тесту щодо одиночних константних несправностей схеми, реалізує ЦС. Тому даний підхід орієнтований швидше на досягнення високої повноти тесту для константних несправностей, а не виявлення помилок в мовних конструкціях HDL. Більш того, при цьому ефективність тестової послідовності не може бути визначена безпосередньо на функціональному рівні. Тому в даний час для верифікації та тестування ЦС, описаних на HDL, застосовуються методи, запозичені з тестування програмного забезпечення.

8.3 Функціональне тестування апаратного забезпечення

Функціональне тестування з'явилося найпершим. Постійне збільшення складності виробів роблять процес підготовки функціонального тесту нескінченно довгим. Діагностування несправностей, виявлених у процесі функціонального тестування, може бути досить складним, що вимагає залучення кваліфікованих фахівців. Тому перед тестом системи в цілому часто здійснюється тестування на рівні окремих плат. Тестування плат може бути здійснено і на функціональному рівні, але даний поділ робить діагностування несправностей і підготовку тестів більш гнучкими. Швидко зростаюча складність інтегральних мікросхем викликає схожі проблеми із функціональним тестуванням на рівні плат, так само як і в системному тестуванні – довгий час підготовки тестів, неточне тестове покриття, слабка діагностика.

Наступний широко поширений метод тестування – це внутрішнє-схемне тестування (In-Circuit Test, ICT). Цей метод дозволяє знаходити дефекти і помилки монтажу шляхом забезпечення прямого електричного доступу до компонентів на платі через адаптер. Внутрісхемне тестування ідеально підходило для DIP-компонентів і технології штирьового монтажу. Але у зв'язку з появою багатошарових друкованих плат і більш складних корпусів мікросхемтестовий доступ став сильно обмежений. Технологія внутрисхемного тестування не може розвиватися також швидко, як мініатюризація розмірів компонентів і виробів.

Електронна індустрія передбачала ці проблеми заздалегідь, тому був розроблений метод периферійного сканування, закріплений стандартом IEEE 1149.1, який описує порт тестового доступу (TAP – Test Access Port) і архітектуру периферійного сканування [8, 9]. Метою створення даного стандарту було подолання недоліків інших методів тестування.

Дивлячись на еволюцію тестових методів можна зробити наступні спостереження:

(1) розроблення тестопригодності виробів (Design-For-Test, DFT) стає все більш і більш необхідним доповненням функціонального тестування, дозволяючи зробити контроль більш повним і інформативним, (2) для того щоб виробляти і тестувати сучасні передові розробки, тестопригодність необхідна.

Спочатку, тестування було похідною процесу налагодження нової розробки і пошуку дефектів монтажу. Через зростання складності схем пристроїв керованість цими процесами могла бути підвищена тільки при роздільному їх проведенні. Виявлення і виправлення виробничих дефектів на стадії налагодження дослідних зразків стало необхідністю.

З ростом складності продукції багато виробників почали застосовувати багатоступеневу стратегію тестування, метою застосування якої є якомога більш раннє виявлення і виправлення помилок виробничого процесу.

Перша версія стандарту Boundary-Scan [8], з'явилася на початку 1990 року, і отримала ім'я, яке зберіглося і сьогодні – IEEE 1149.1. Стандарт цієї технології також називається Test Access Port and Boundary-Scan Architecture (порт тестового доступу та архітектура граничного сканування). Проект був розроблений міжнародною групою експертів, яка носила назву JTAG (Joint Test Action Group – об'єднана група розробки методів тестування).

Сама архітектура цифрового стандарт Boundary-Scan не відрізняється особливою складністю, на відміну від своїх можливостей. Відповідно до стандарту IEEE 1149.1, так звана Boundary-Scan-IC, повинна бути оснащена чотирма обов'язковими елементами:

- TAP-портом, який складають чотири обов'язкових сигналів, і п'ятий за рішення розробників безпосередньо самої плати (TCK – контакт синхронізації роботи механізму Boundary-Scan; TMS – контакт вибору тестового режиму; TDI – контакт введення тестових даних; TDO – контакт виведення тестових даних (знаходиться в постійно в третьому стані, окрім режиму зсуву); RST – контакт асинхронного скидання стану TAP-контролера (може зовсім не бути присутньою)). TAP-контролер виступає одним з важливих елементів управління всією роботою технології Boundary-Scan;
- IR (Instruktion Register від анг. регістр команд) – перша група регістрів, в якій обов'язковим за стандартом повинен бути присутнім хоча б один Регістр Команд (РК);
- DR (Data Registers від анг.– Регістри Даних) – друга група регістрів, відповідно до стандарту зобов'язана в себе містити як мінімум два регістри: Регістр Обходу (РО, також його іноді називають Шунт-Регістр), Регістр Boundary-Scan.

Такий мінімальний комплект елементів вимагає стандарт IEEE 1149.1. Інші регістри, які можуть доповнити групу, як IR, так DR на розсуд розробників плат, також допускаються створеним стандартом.

Для того, щоб отримати хороше тестове покриття, немає необхідності в тому, щоб всі компоненти на платі мали JTAG-інтерфейс. Наприклад, багато блоків, складається з нескановних компонентів (кластера), можуть тестуватися, незважаючи на відсутність прямого доступу для периферійного сканування. У дійсності, існують практичні приклади, коли здійснюється контроль і детальне тестування абсолютно всієї плати (включаючи пам'ять) за допомогою одного або двох компонентів, що підтримують периферійне сканування.

На рис. 8.2 зображена архітектура Boundary-Scan. На TAP-контролер подаються 2 (3) сигналу, за допомогою яких контролер встановлює відповідний режим роботи схеми. Сам TAP-контролер є автомат з кінцевою кількістю вершин.

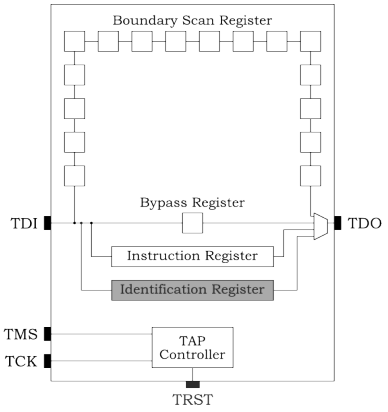

Рисунок 8.2. Архітектура JTAG

Сьогодні на світовому ринку в цій області лідирують чотири представники США і Європи, які поставляють програмно-апаратні комплекси (ASSET InterTech Inc. і CORELIS Inc – США; GOEPEL Electronic – Німеччина; JTAG Technologies – Нідерланди). Такі розробки називаються BS-тестери.

Використання JTAG і технології граничного сканування в мікросхемі, на платі або в пристрої додає вартість і збільшує час розро-

блення проекту. Але, все ж ці витрати легко окупаються при проведенні тестування, яке забезпечується на кожній стадії циклу життя виробу. Крім безпосередньо граничного тестування, проектувальники використовують технологію JTAG для того, щоб виробляти самотестування (BIST) (у тих компонентах, де воно реалізовано) і завантажувати внутрішні значення в регістри пристрою або програмувати мікросхеми ПЗУ. Тести, які були розроблені та використані на етапі проектування, можуть бути передані виробництву, для того щоб забезпечити додаткове зниження вартості і часу на перевірку виробів при вихідному контролі.

8.4 Тестування програмного забезпечення вбудованих систем

Тестування програмного забезпечення (*software testing*) – це планова, впорядкована діяльність, процес аналізу або експлуатації програмного забезпечення з метою виявлення дефектів.

Всі види тестування програмного забезпечення вбудованих систем можливо класифікувати: за рівнем елементів, що перевіряються, за характеристикам якості, за джерелами даних, по ролі команди.

За масштабом перевіряємих елементів розрізняють тестування модулів, комплексні випробування, системне тестування, приймальне тестування.

За характеристиками якості можливо тестування функціональності, тестування надійності, тестування переносимості, тестування зручності використання.

За джерелами даних виділяють тестування чорного ящика, сірого ящика та білого ящика.

Функціональні тести базуються на функціях і особливостях, а також взаємодії з іншими системами, і розглядають зовнішню поведінку системи. Далі перераховані одні з найпоширеніших видів функціональних тестів: функціональне тестування (Functional testing); тестування безпеки (Security and Access Control Testing); тестування взаємодії (Interoperability Testing).

Нефункціональне тестування описує тести, необхідні для визначення характеристик програмного забезпечення, які можуть бути виміряні різними величинами. В цілому, це тестування того, «як» система працює. Це всі види тестування продуктивності такі як тестування навантаження (Performance and Load Testing); стресове тестування (Stress Testing); тестування стабільності або надійності (Stability / Reliability Testing); об'ємне тестування (Volume Testing); тестування інсталяції (Installation testing); тестування зручності користування (Usability Testing); тестування на відмову і відновлення (Failover and Recovery Testing); конфігураційне тестування (Configuration Testing).

Після проведення необхідних змін, таких як виправлення бага/дефекту, програмне забезпечення повинне бути протестовано для підтвердження того факту, що проблема була дійсно вирішена. Нижче перераховані види тестування, що необхідно проводити після установки програмного забезпечення, для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

1. Димове тестування (Smoke Testing);
2. Регресійне тестування (Regression Testing);
3. Тестування збірки (Build Verification Test);
4. Санітарне тестування або перевірка (Sanity Testing).

Тестування на різних рівнях проводиться протягом усього життєвого циклу розробки і супроводу програмного забезпечення. Рівень тестування визначає те, над чим виробляються тести: над окремим модулем, групою модулів або системою, в цілому. Проведення тестування на всіх рівнях системи – це запорука успішного завершення проекту.

8.5 Метод регресійного тестування інформаційних систем

Розглянемо метод регресійного тестування ІС, зокрема ІС з web-орієнтованим інтерфейсом (BOIC) [16, 13]. Даний клас ІС будуються за принципом еволюційного прототипування, тобто послідовно створюється макет системи, який буде з кожною ітерацією ближче до реального продукту. Такий підхід має ту перевагу, що на кожному кроці розробники манипулюють працюючою системою, яка має часткову функціональність, яку можна тестувати і покращувати з кожною ітерацією.

Для тестування будемо виділяти ФО, що є атомарним об'єктом, пов'язаним з вимогами користувача (прототипом ІС) і функціональністю системи за допомогою адаптивних правил, залежно від логіки додатка. Таким чином, якщо прийняти за *F* – безліч виділених в ІС ФО, а за *Fk* – безліч реалізованих і пройшли тестування в k-ої версії ФО, то прогрес розробки *IC* в першому наближенні можна оцінити за допомогою критерію:

$$M_{progress1} = \frac{|F_k|}{|F|} \times 100\%. \quad (8.1)$$

Більш точну інформацію про стан розроблення ІС дасть критерій, який враховує трудомісткість інтеграції ФО і відсоток завершеності розробки ФО і показує прогрес виконання ІС за поточним кале

$$M_{progress2} = \frac{\sum_{i=1}^{N}(tp_i \times p_i + tp_i^{(int)} \times p_i^{(\text{int})})}{100 \times \sum_{i=1}^{N}(tp_i + tp_i^{(int)})}, \quad (8.2)$$

де *tp* – планова трудомісткість ФО; $tp^{(int)}$ – Планові трудовитрати на інтеграцію ФО у ІС; *N* – загальна кількість ФО, $N = |F| - |F_{cansel}|$, | F | – загальна кількість виділених в ІС ФО; $|F_{cansel}|$ – Кількість ФО, знятих з розробки; p_i – відсоток завершеності розробки ФЕ; $p_i^{(int)}$ – відсоток завершеності інтеграції ФО.

Величини, визначають за результатами тестування, виходячи з кількості пройдених тестів для даної ФО:

$$\frac{n\_test\_passed_i}{n\_test\_total_i} \times 100\%,$$

де $n\_test\_passed_i$ – кількість тестів, для яких виконаний критерій успішності, по і-тої ФО, $n\_test\_total_i$ – загальна кількість тестів з і-тої ФО.

Відхилення фактичних трудовитрат на розробку ІС від календарного графіка дозволяє визначити критерій:

$$M_{dev} = \sum_{i=1}^{N}\left[\left(\frac{tf_i \times p_i + tf_i^{(int)} \times p_i^{(\text{int})}}{100}\right) - (tp_i + tp_i^{(int)})\right], \quad (8.3)$$

де tf – поточні фактичні трудовитрати на ФО; $tf^{(int)}$ – поточні фактичні трудовитрати на інтеграцію ФО.

Всі запропоновані критерії оцінки прогресу прагнуть до 100% (відповідає повністю розробленої ІС, що задовольняє всім вимогам), а значення критерію відхилення календарного графіка дозволяє визначити неправильну оцінку трудомісткості ФО (від'ємне значення свідчить про завищену оцінку трудомісткості, позитивне визначає поточне перевищення календарного плану).

Дані критерії є заходами зусиль і вимірюються відносної шкалою по відношенню до початкових виділеним завданням і їх тривалості. Розраховані значення (8.1–8.3) дають на проміжних етапах розробки ІС менеджеру розробки кількісну оцінку виконання проекту та можливість визначити відповідність процесу розробки календарним планом. Використання критеріїв окремо може спричинити за собою невірні висновки, в той час як їх комплекс дозволяє точно визначити поточний стан розроблення ІС.

Таким чином, запропоновані критерії є об'єктивними обчислюваними динамічними критеріями, що служать заходом визначення прогресу розробки ІС і дозволяють на будь ітерації визначити відповідність фактичного стану процесу розроблення до плану. Отримані значення дозволяють визначити чи виконані на даний момент всі заплановані завдання, немає перевищення термінів виконання та незавершених завдань. Отримані значення критеріїв дозволяють переоцінити трудомісткість окремих модулів і ІС в цілому, щоб потім скорегувати процес розробки з метою зменшити відхилення від запланованої дати передання до експлуатації, таким чином, знизивши ризик порушень календарного планування.

Для ІС з веб-інтерфейсом (ВОІС) рівень критичності web-ресурсів різного рівня вкладеності значно відрізняється і тому необхідно розглядати їх окремо.

Рівень вкладеності – це параметр, який відповідає за стан вебсторінки в загальній структурі сайту. Рівень вкладеності визначається по мінімальній кількості кілків (переходів), які потрібно зробити, щоб досягти цієї сторінки з головної сторінки сайту. Головна сторінка сайту завжди має рівень вкладеності 1. Всі інші сторінки, на які можна потрапити з неї за один клік – рівень вкладеності 2. Сторінки з рівнем вкладеності 4 з точки зору оптимізації є неприпустимими при розробці структури сайтів, так як пошукові роботи проводять їх індексацію не дуже часто.

Таким чином, структура ПК ІС може бути представлена наступним чином:

$$WOIS = <structure0, S_p, structure, S>,$$

де structure0=$< Ed2_p, EdN3_p, EdN^p_{®\infty} >$ – первинна структура прототипу ВОІС, яка не містить інформації про рівень вкладеності web-ресурсів;

Ed2p – web-ресурси другого рівня вкладеності прототипу інтерфейсу ПК ВОІС;

Ed3p – web-ресурси третього рівня вкладеності прототипу інтерфейсу ПК ВОІС;

$Ed2_p, EdN3_p, EdN^p_{®\infty}$ – Web-ресурси глибше третього рівня вкладеності прототипу інтерфейсу ПК ВІС;

Sp – матриця зв'язків між web-ресурсами в прототипі інтерфейсу ПК ВОІС;

structure =$< Ed2, EdN3, EdN_{\to\infty} >$ – структура ПК ВОІС;

Ed2 – web-ресурси другого рівня вкладеності ПК ВОІС;

Ed3 – web-ресурси третього рівня вкладеності ПК ВОІС;

$EdN_{\to\infty}$ – Web-ресурси глибше третього рівня вкладеності ПК ВОІС;

S – матриця зв'язків між web-ресурсами в ПК ВОІС.

Відповідно, |*F*| – кількість виділених в ІС ФО – прийме вигляд:

где *N2p* – кількість елементів прототипу другого рівня вкладеності; *N3p* – кількість елементів прототипу третього рівня вкладеності; $N^p{}_{®\infty}$ - кількість елементів прототипу глибше третього рівня вкладеності;

а F_k – кількість реалізованих і пройшли тестування в k-ої версії ФО – прийме вигляд:

$$|F_k'| = N2 + N3 + N_{\to\infty},$$

де *N2* – кількість web-ресурсів версії ВОІС другого рівня вкладеності; *N3* – кількість web-ресурсів версії ВОІС третього рівня вкладеності; $N_{\to\infty}$ – кількість web-ресурсів версії ВОІС глибше третього рівня вкладеності.

Аналогічно критеріями для оцінки відповідності прототипу запропоновано тестування наступних критеріїв стану:

1.– критерій відповідності кількості web-ресурсів другого рівня вкладеності версії ВОІС прототипу:

$$M_{N_2} = \frac{N2}{N2_p} \times 100\%, \quad lim(M_{N_2}) \to 100\% \tag{8.4}$$

де *N2* = |*Ed2*| – кількість web-ресурсів версії ВОІС другого рівня вкладеності;

N2p = |*Ed2p*| – кількість елементів прототипу другого рівня вкладеності;

2.– критерій відповідності кількості web-ресурсів третього рівня вкладеності версії ВОІС прототипу:

$$M_{N3} = \frac{N3}{N3_p} \times 100\%, \; lim(M_{N_3}) \to 100\% \tag{8.5}$$

де *N3* – кількість web-ресурсів версії ВОІС третього рівня вкладеності;

N3p – кількість елементів прототипу третього рівня вкладеності;

3.– критерій кількості web-ресурсів в прототипі інтерфейсу глибше третього рівня вкладеності:

$$M_{N_{\to\infty}} = \frac{1}{N2_p + N3_p} \times \sum_{i=4}^{q} N_i \times 100\%, \quad lim(M_{N_{\to\infty}}) \to 0 \tag{8.6}$$

де *q* – максимальний рівень вкладеності версії ВОІС;

N2p і *N3p* – відповідно кількість елементів прототипу другого і третього рівня вкладеності;

для кращої індексації ВОІС $lim(M_{N_{\to\infty}}) \to 0$, тобто кількість web-ресурсів більш ніж третього рівня вкладеності має бути як можливо меншим порівняно з кількістю *N3* і *N2*.

4. *MS* – критерій відповідності структури версії ПК ВОІС і прототипу інтерфейсу (через симетричності відносно головної діагоналі розглядається тільки нижня трикутна матриця):

$$M_s = 2 \times \frac{\sum_{i=1}^{N} \sum_{j=1}^{i-1} a[i][j]}{N^2 - N}, lim(M_s) \to 1, \tag{8.7}$$

де *N* – кількість web-ресурсів ВОІС і прототипу (у разі кількість web-ресурсів не збігається, додаються відсутні рядок і стовпець і заповнюються нулями);

$$a[i][j] = \begin{cases} 1, если\ S_p[i][j] = S[i][j] \\ 0, если\ S_p[i][j] \neq S[i][j] \end{cases},$$

S – матриця зв'язків між web-ресурсами ВОІС;

Sp – матриця зв'язків між елементами прототипу.

У заголовках рядків і стовпців матриці Sp знаходяться назви всіх елементів прототипу ВОІС:

$$S_p[i][j] = \begin{cases} 0, е,е\ звязок\ між\ і\text{-}тойи\ \ j\text{-}т\text{-}той\ стор. відсутній; \\ 1, е,е\ \ існує\ перехід\ зі\text{-}т\text{-}той\ на\ую\ стор \\ 2, е,е\ існує\ \ перехід\ сj\text{-}т\text{-}той\ на\тую\ стор \\ 3, е,е\ існує\ двосторонній\ перехід. \end{cases}$$

Матриця *S* будується аналогічно.

5. Співвідношення трудомісткості розроблених та затверджених web-ресурсів версії ВОІС у загальної запланованої трудомісткості ВОІС розраховується аналогічно (8.6):

$$M_{progress} = \sum_{i=1}^{N} \left[\left(\frac{tp_i \times pct_i}{100} + tS_p[i] \right) - (tf_i + tS[i]) \right], lim(M_{progress}) \to 0,$$

де $i = 1..N$, $N = |F|$ – загальна кількість запланованих web-ресурсів;

tp_i – планові трудовитрати на ФО;

pct_i – відсоток завершеності і-ого web-ресурсу;

$tS_p[i]$ – планові трудовитрати на реалізацію зв'язків і-ого web-ресурсу;

tf_i – поточні фактичні трудовитрати на web-ресурс;

$tS[i]$ – поточні фактичні трудовитрати на реалізацію зв'язків і-ого web-ресурсу.

Дані критерії перевіряються на кожному етапі ЖЦ ВОІС за допомогою автоматизованого тестування посилань ВОІС. Це дозволяє перевіряти відповідність кількості web-ресурсів усіх рівнів необхідному замовником кількістю.

8.6 Метод регресійного тестування web-орієнтованих систем

В основу методу покладено прототип інтерфейсу ІС (7.1), на підставі якого обчислюються критерії оцінки якості (8.1–8.7), що зв'язують логічну модель ІС, що розробляється, з артефактами на різних етапах створення ВОІС.

Розглянемо обчислювальну схему методу:

Етап 1. Будується матриця атрибутів вимог – до атрибутів функціональних вимог *V* додається атрибут «Реалізувати на сторінці або для ФО», в якому вказується ідентифікатор web-ресурсу, в якому планується реалізувати дану функціональність.

Етап 2. З поля «Реалізувати на сторінці або для ФО» витягуються ідентифікатори web-ресурсів, прибираються повторювані і формується масив імен web-ресурсів майбутньої ІС structure0:

$$V_a^{e_d^j} == "Реалізувати\ \ для\ ФО" \Rightarrow structure0,$$

де $V_a^{e_d^j}$ – набір атрибутів web-ресурсу; *j = 1..N, N* – кількість web-ресурсів;

structure0- первинна структура ІС, яка не містить інформації про рівень вкладеності web-ресурсів.

Етап 3. Будується модифікована структура прототипу Sp, тобто структура взаємодії web-ресурсів, матриця можливих переходів між ними Sp [N+1] [N+1].

Етап 4. В результаті розробки на k-ой ітерації отримують модифіковану структуру ПК ІС structure і матрицю можливих переходів між реалізованими web-ресурсами S [N + 1] [N + 1].

Етап 5. Після етапу розробки, на кожній ітерації на етапі тестування розраховуються модифіковані критерії для оцінки стану процесу розробки ПК ІС, записані в формулах (8.1–8.6), тобто порівнюються модифікована і первинна структура прототипу і структури, отримані в результаті аналізу розробленої на поточній ітерації версії системи:

$$\begin{cases} S[N+1][N+1] == Sp[N+1][N+1] \\ structure == structure0 \end{cases}$$

Етап 6. Обираються методи тестування для змінених ФО ІС. Проводиться тестування і знайденим помилок залежно від результатів тестування присвоюються такі статуси:

$$status(bug_i) \in \{"open","fixed","verified\ fixed","verified\ closed"\},$$

де «open» – відкритий і вимагає тестування; «fixed» – виправлений; «verified fixed» – перевірений виправлений; «verified closed» – закритий виправлений.

Етап 7. Вибираються методи тестування для пов'язаних ФО ІС за рахунок аналізу структури зв'язків між ресурсами ІС.

Формально процес вибору методів тестування виглядає наступним чином:

1. Для дефектів (помилок) зі статусом « *fixed* », необхідно виконати тестування web-ресурсу, з яким пов'язано функціональне вимога, до якого відноситься виправлений дефект:

$$if\ (fixed(bug_i)\ \&\ bug_i \in V^{e^g_d}) \Rightarrow test(e^g_d),$$

де bug_i – будь-який з незакритих дефектів, i=1..Nb, Nb – кількість дефектів;

$fixed(bug_i)$ – привласнення дефекту статусу « *fixed* »;

$V^{e^g_d}$ – функціональна вимога, яка ставиться відповіно до ресурсу e^g_d, g = 1..N, N – кількість web-ресурсів;

$test(e^g_d)$ – запуск безлічі тестів для web-ресурсу e^g_d.

2. Визначаються ресурси, пов'язані зі зміненим, для тестування:

$$if\ (S_p[g][k] > 0) \Rightarrow test(e^k_d),$$

де k = 0..N – номер web-ресурсу.

3. Якщо тестований web-ресурс і все, пов'язані з ним, пройшли тестування, поточного дефекту присвоюється статус «перевірений виправлений» і потім «закритий виправлений»:

$$if\ (test(e^g_d) == passed\ \&\ test(e^k_d) == passed)$$

$$\Rightarrow verified\ fixed(bug_i)\ \&\ fixed\ closed(bug_i),$$

де $verified\ fixed(bug_i)$ – присвоєння bug_i статусу «перевірений виправлений»;

$fixed\ closed(bug_i)$ – присвоєння bug_i статусу «закритий виправлений».

Етап 8. Прийняття рішення про завершення розробки.

Етапи 1–7 виконуються на кожній ітерації до тих пір, поки результати тестування не зійдуться з прописаними в плані тестування критеріями закінчення тестування. В успішних розробках критерії $M_{N2,3} = 100\%$, $\lim M_S = 100\%$ і $\lim M_{N\infty} \to 0$ (граничний рівень може встановлюватися замовником).

Необхідно, щоб $\lim M_{N2,3} \to 100\%$, $\lim M_S \to 100\%$ и $\lim M_{N\to\infty} \to 0\%$, тому при інших значеннях необхідно визначити через що відбулося відхилення від норми і виправити невідповідність прототипу (етап 5) або неактуальне прототип (етап 2). У разі зміни або додавання вимог до системи – повернення на 1 етап.

Запропонований метод виходить за рамки стандартних цілей контролювати якість і дозволяє забезпечити якість на різних етапах ЖЦ. Це можливо за рахунок планомірного контролю та відстеження відповідності розроблюваної ІС вимогам замовника до функціональності і структурі ІС. Створені на кожному етапі артефакти тестуються на відповідність прототипу і, таким чином, даний метод надає кошти оцінки якості на різних етапах створення ІС за рахунок аналізу відповідності логічної моделі розроблюваної системи, прийнятої замовником.

9 ВИКОРИСТАННЯ ВІДДАЛЕНОЇ ЛАБОРАТОРІЇ GOLDI ДЛЯ ВИВЧЕННЯ ЗАСОБІВ ТЕСТУВАННЯ ВБУДОВАНИХ СИСТЕМ

9.1 Аналіз можливостей лабораторії GOLDi для цілей навчання

При сучасному навчанні неможливо не використовувати інформаційні комунікаційні технології (ІКТ).

Кожна з п'яти фаз, що присутні при викладанні матеріалу – вступ до матеріалу, мотиваційна фаза, фаза викладання матеріалу, фаза фіксації та діагностична фаза, містить свої ІКТ.

Однією з найважливіших фаз є фаза фіксації матеріалу, коли учень за допомогою лабораторного практикуму закріплює отриманий матеріал (рис. 9.1).

Рисунок 9.1. Піраміда ефективності методів навчання

Але в сучасних умовах дуже складно забезпечити постійний доступ до обладнання, особливо коли йде мова про короткосрочні курси для представників підприємств. На підтримку приходять віддалені лабораторії [1–7].

Для грунтовного використання існуючих віддалених лабораторій необхідно провести всебічний аналіз результатів навчання, що планується забезпечити та потужностей віддалених лабораторій. Ефективним інструментом для такого аналізу є метод стратегічного планування SWOT (Strengths, Weaknesses, Opportunities, Threats). Шаблон для використання віддалених лабораторій наведений у таблиці 9.1 [4].

Для вивчення дисципліни Якість інформаційних систем були проаналізовані цілі і завдання (рис. 9.2). Було проведено аналіз режимів функціонування лабораторії GOLDi [3–6] та проведено SWAT (таблиця 9.2).

Таблиця 9.1. Загальний шаблон SWOT аналізу для адаптації існуючих віддалених лабораторій

	Можливості Які можливості віддаленої лабораторії? Яка галузь використанні віддаленої лабораторії?	**Погрози** Які обмеження на вхідні та вихідні данні при використанні віддаленої лабораторії? Які передумови використання віддаленої лабораторії?
Сильні стороні Яка мета курсу? Які завдання навчання? Які методи повинен вивчити студент? Які ІКТ можуть бути використані в курсі? Для яких завдань можуть бути використання ІКТ?	Які можливості віддаленої лабораторії дозволять вирішити завдання навчання?	Які можливості використання матеріалу курсу, що дозволять запобігти передумовам віддаленої лабораторії?
Слабкі сторони Які існують передумови для вивчення дисципліни? Чи є в курсі матеріал для якого не можливо використовувати ІКТ?	Як потужності віддаленої лабораторії можуть бути використані для запобігання передумов курсу?	Як можливо мінімізувати внутрішні слабості для запобігання зовнішнім погрозам?

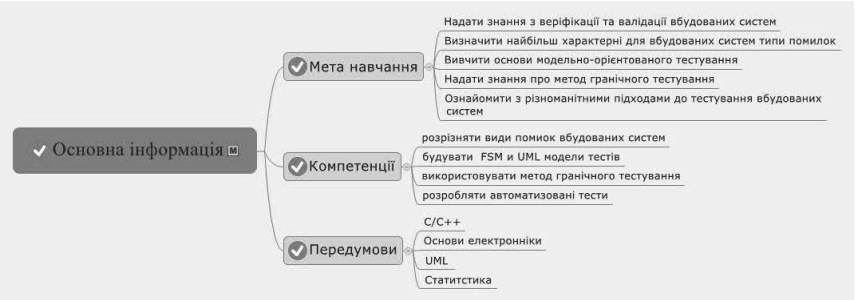

Рисунок 9.3. Основна інформація з дисципліни «Якість інформаційних систем»

Таблиця 9.2. Шаблон SWOT для адаптації лабораторії GOLDi для дисципліни «Якість інформаційних систем»

	Можливості	**Погрози**
Сильні стороні	Використання наступних режимів лабораторій “Stand alone mode” для навчанню генерації тестів з використанням FMS. “Remote control Mode” для виконання розроблених тестів “Rapid Prototype mode” для функціонального тестування	Необхідно розробити практикум для генерації тестів, що можуть використовуватись для плати швидкого прототипування
Слабкі сторони	Використання віртуального режиму “Virtual control Mode” перед використанням обладнання	Необхідно розробити додаткові тести з FMS, VHDL and C для вбудованих систем у системі LMS Moodle.

Так для викладання в рамках міжнародного проекту ICo-op для кожної фази викладання були використанні наступні ІКТ – вступ до матеріалу – презентація лабораторії GOLDi, мотиваційна фаза – демо режим лабораторії GOLDi, фаза викладання матеріалу – презентаційні матеріали, додаткові теоретичні матеріали були завантажені в університетську систему LMS Moodle, фаза фіксації – режими віддаленої лабораторії GOLDi для вивчення FSM для використання при генерації тестів та основи функціонального тестування, для діагностичної фази – система керування контентом Moodle

9.2 Використання лабораторії GOLDi для модельно-орієнтованого тестування

Тестування на основі моделі (англ. Model-based testing) – це тестування програмного забезпечення, в якому варіанти тестування частково або цілком виходять з моделі, яка описує деякі аспекти (частіше функціональні) тестованої системи (англ. System under test). Моделі можуть відображати бажану поведінку системи або використовуватися для створення тестових стратегій або середовища тестування. Тестування на основі моделі виконується в наступні етапи:

1. Побудова моделі (Модель кінчений автоматів, модель подій системи, і т.д.).
2. Генерування очікуваних вхідних даних.
3. Генерування очікуваних вихідних значень.
4. Порівняння з фактичними результатами.
5. Прийняття рішень

При побудові моделі, вона повинна відповідати наступним вимогам:

- простота – модель повинна бути настільки простою, щоб витрати на її побудову могли б окупитися. Хороша проста модель не вимагає великих вкладень на навчання персоналу. Модель повинна бути інтуїтивно зрозумілою кожному співробітнику;
- детальність – модель повинна бути настільки детальною, щоб з її допомогою можна було б описати всі стани і параметри програми для проведення повноцінного тестування;

- тестованість – модель повинна бути побудована таким чином, щоб при генерації тестів можна було б отримати тестові набори, придатні для ручного тестування (а згодом, і автоматизованого);
- автоматизованість – модель повинна підтримувати потенційну автоматизацію тестування. Тобто переходи між станами повинні бути настільки «низькорівневими», щоб їх можна було передавати на вхід інструменту автоматизації тестування.

Для побудови варіантів тестування заснованих на моделях ефективними інструментами виступають: діаграми станів UML, кінцеві автомати, автомати Мілі, автомати Мура, контекстно-незалежні граматики, марківські мережі, таблиці рішень.

При використанні кінцевих автоматів для тестування умови можуть бути обрані за наступними критеріями:

1. Які значення може приймати параметр? Перевірити граничні значення, неприпустимі значення, нормальні значення
2. Які сукупності значень мають особливий сенс? Якщо параметри залежать один від одного – простежити залежності
3. Які комбінації параметрів приведуть до коректній роботі програми / методу / модуля?
4. Які комбінації параметрів викличуть помилку, неправильний код повернення або аномальна поведінка?

На рисунку 9.4 відображені можливі ситуації при формуванні моделей з використанням FSM.

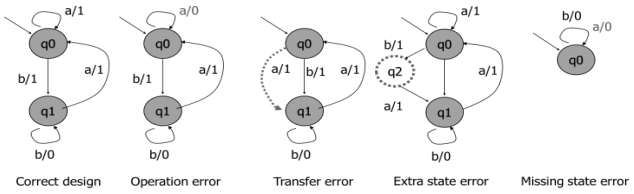

Рисунок 9.4. FSM моделі

Найбільш ефективним є використання FSM моделей для таких видів тестування як підтверджуюче тестування (conformance testing), тестування переходів (transition testing) та тестування на основі критерію текстового покриття (coverage testing).

Для навчання моделюванню з використанням FSM віддалена гібридна лабораторія GOLDi має віртуальний режим (рис. 9.5) та віддалений експеримент з використанням веб-клієнту (рис. 9.6) [5].

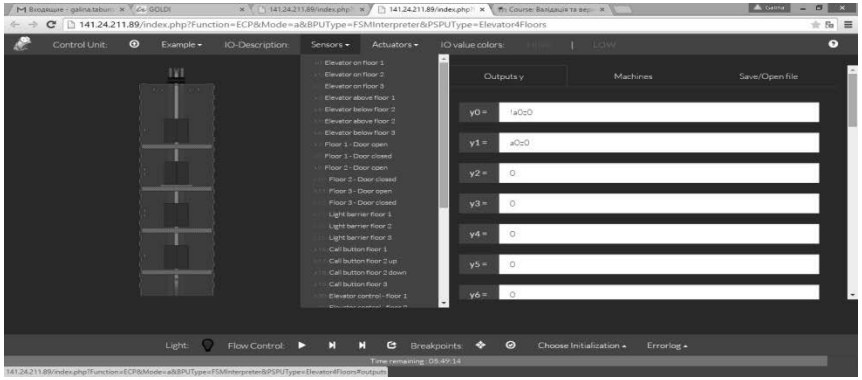

Рисунок 9.5. Віртуальний режим роботи ліфта

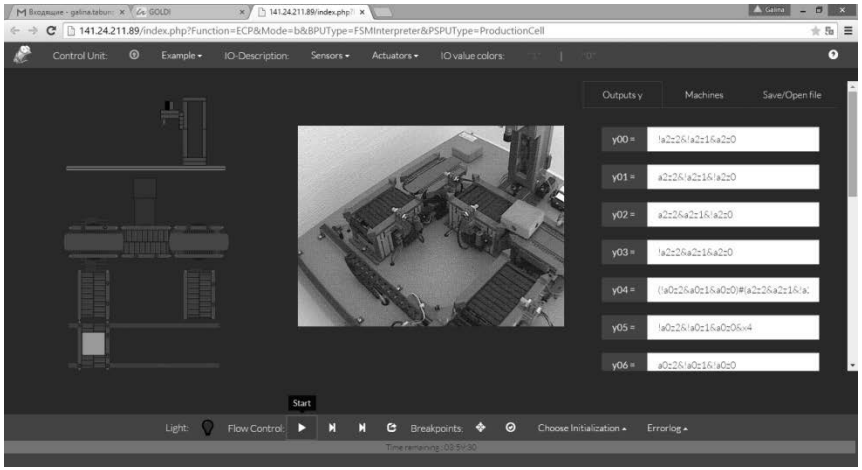

Рисунок 9.6. Віддалений експеримент для конвеєрної лінії

Завдання для студентів можуть бути наступними:

1. Побудуйте модель роботи 3-вісного порталу відповідно до графіка на рис. 9.7.

2. Розширити тестові послідовності і використовувати їх для конструювання покриття всіх переходів для моделі 3-вісний портал.
3. Розробити найпростіший критерій тестового покриття для 3-вісного порталу.

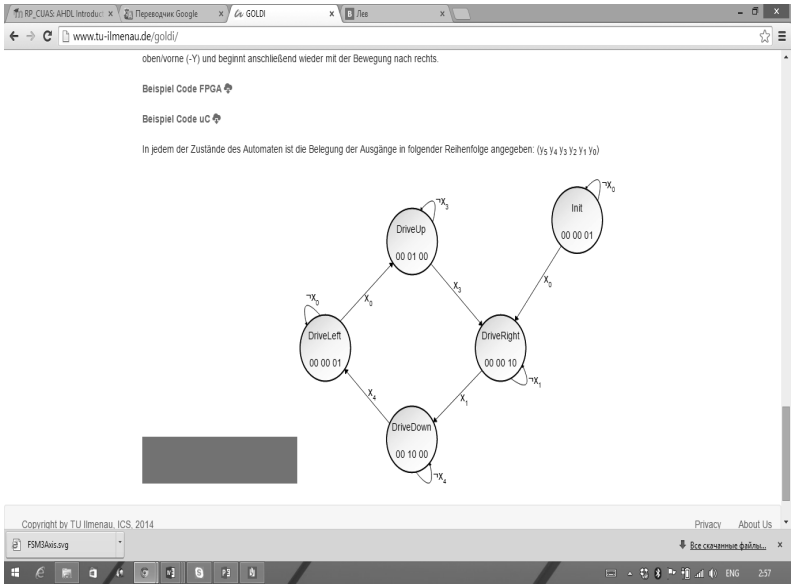

Рисунок 9.7. Модель роботи 3-вісного порталу

4. Розширити тестові послідовності і використовувати їх для покриття всіх переходів для моделі ліфт.
5. Розробити найпростіший критерій тестового покриття для моделі ліфт.
6. Розширити тестові послідовності і використовувати їх для покриття всіх переходів для моделі конвеєрна лінія.
7. Розробити найпростіший критерій повного тестового покриття для конвеєрної лінії.

9.3 Використання плати швидкого прототипування для навчання функціональному тестуванню вбудованих систем

Серцем віддаленої гібридної лабораторії GOLDi є плата швидкого прототипування. Робота з цією платою можлива як з використанням автономного програматора (рис. 9.9) так і з використанням віртуального та віддаленого режимів лабораторії (рис. 9.10–9.11) [4, 5].

Рисунок 9.9. Плата швидкого прототипування

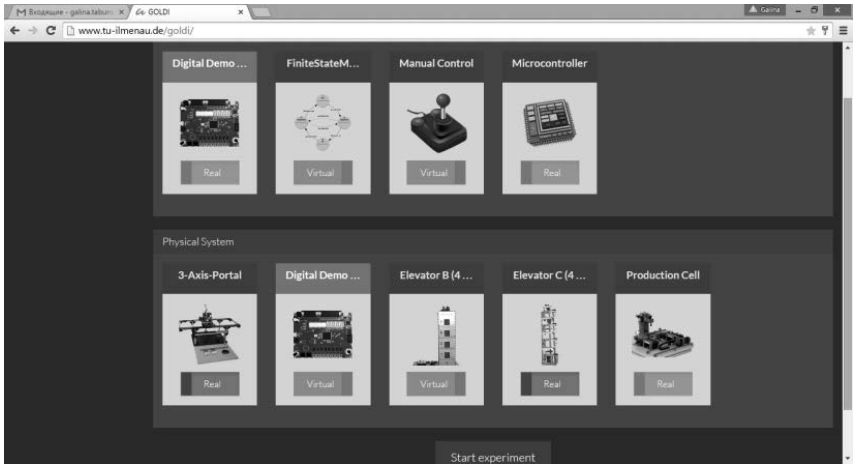

Рисунок 9.10. Режими роботи віддаленої лабораторії

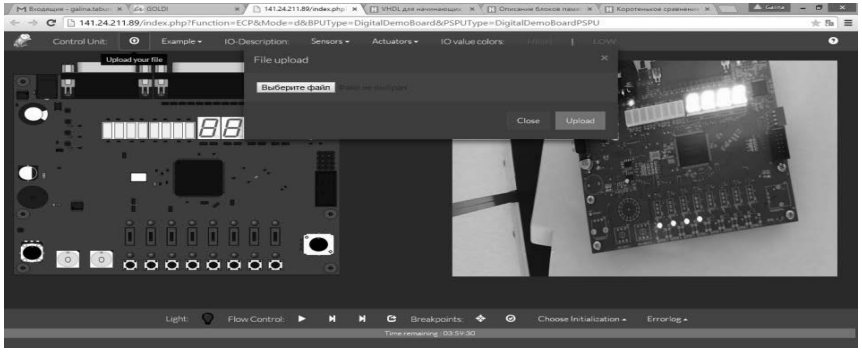

Рисунок 9.11. Робота з платою швидкого прототипування

Для роботи з платою вихідними даними є.pof файл, отриманий при компіляції проекту в Quartus II [5].

Для виконання функціонального тестування можливо використовувати програми розроблені з використанням VHDL у середовищі Quartus II Web-edition від Altera (рис. 9.12).

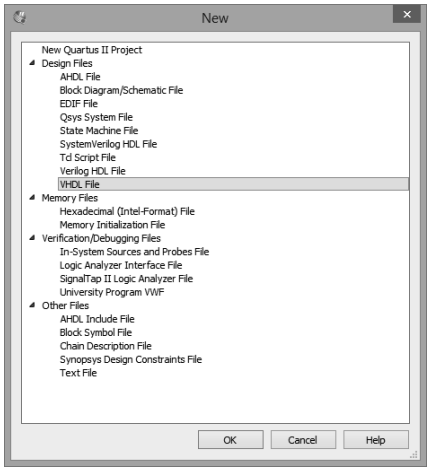

Рисунок 9.12. Створення VHDL проекту

Створення тесту складається з наступних кроків:

Крок 1. Введення проекту на мові vhdl.

Крок 2. Компіляція проекту (рис. 9.13).

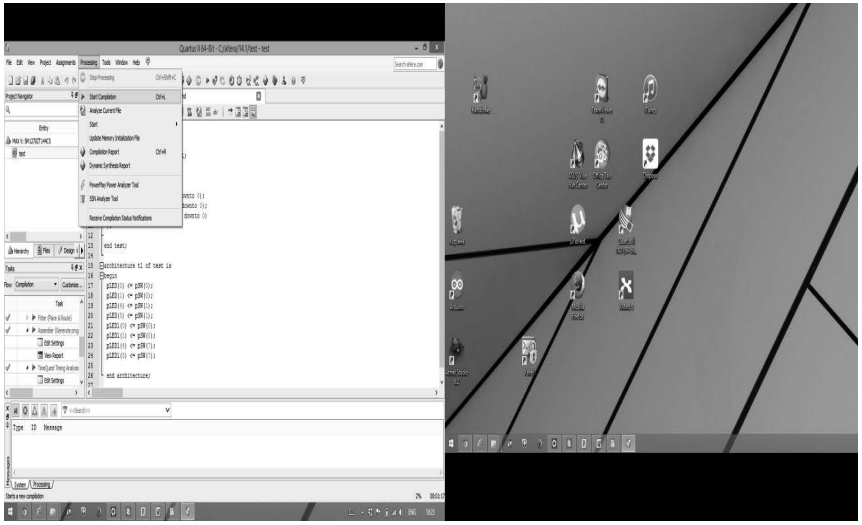

Рисунок 9.13. Компіляція проекту

Крок 3. Зв'язування контактів з використанням Pin Planner (рис. 9.14)

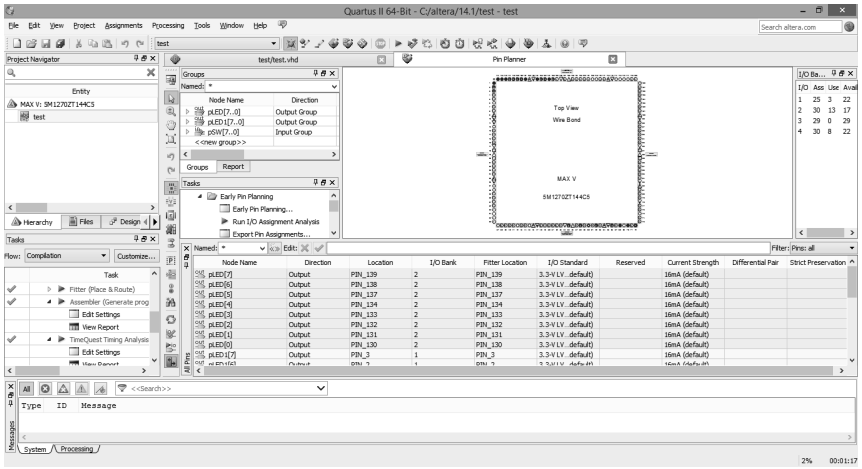

Рисунок 9.14. Pin Planner для Quartus II

На рис. 9.15 міститься відповідність виходів та контактів для MAX® V – 5M1270Z CPLD від Альтера.

Name	Input/Output	Pin
Hex0[2]	Output	PIN_85
Hex0[1]	Output	PIN_86
Hex0[0]	Output	PIN_87
Hex1[7]	Output	PIN_96
Hex1[6]	Output	PIN_102
Hex1[5]	Output	PIN_103
Hex1[4]	Output	PIN_104
Hex1[3]	Output	PIN_105
Hex1[2]	Output	PIN_97
Hex1[1]	Output	PIN_98
Hex1[0]	Output	PIN_101
Hex2[7]	Output	PIN_108
Hex2[6]	Output	PIN_112
Hex2[5]	Output	PIN_113
Hex2[4]	Output	PIN_114
Hex2[3]	Output	PIN_117
Hex2[2]	Output	PIN_109
Hex2[1]	Output	PIN_110
Hex2[0]	Output	PIN_111
Hex3[7]	Output	PIN_118
Hex3[6]	Output	PIN_122
Hex3[5]	Output	PIN_123
Hex3[4]	Output	PIN_124
Hex3[3]	Output	PIN_125
Hex3[2]	Output	PIN_119
Hex3[1]	Output	PIN_120
Hex3[0]	Output	PIN_121
HexCode0[3]	Input	PIN_52
HexCode0[2]	Input	PIN_50
HexCode0[1]	Input	PIN_51
HexCode0[0]	Input	PIN_49
HexCode1[3]	Input	PIN_48
HexCode1[2]	Input	PIN_44
HexCode1[1]	Input	PIN_45
HexCode1[0]	Input	PIN_43
Incremental[1]	Input	PIN_80
Incremental[0]	Input	PIN_81
Switch[7]	Input	PIN_55
Switch[6]	Input	PIN_58
Switch[5]	Input	PIN_60
Switch[4]	Input	PIN_62
Switch[3]	Input	PIN_66
Switch[2]	Input	PIN_68
Switch[1]	Input	PIN_70
Switch[0]	Input	PIN_72

Name	Input/Output	Pin
Bargraph0[7]	Output	PIN_139
Bargraph0[6]	Output	PIN_138
Bargraph0[5]	Output	PIN_137
Bargraph0[4]	Output	PIN_134
Bargraph0[3]	Output	PIN_133
Bargraph0[2]	Output	PIN_132
Bargraph0[1]	Output	PIN_131
Bargraph0[0]	Output	PIN_130
Bargraph1[7]	Output	PIN_3
Bargraph1[6]	Output	PIN_2
Bargraph1[5]	Output	PIN_1
Bargraph1[4]	Output	PIN_144
Bargraph1[3]	Output	PIN_143
Bargraph1[2]	Output	PIN_142
Bargraph1[1]	Output	PIN_141
Bargraph1[0]	Output	PIN_140
Button[7]	Input	PIN_53
Button[6]	Input	PIN_57
Button[5]	Input	PIN_59
Button[4]	Input	PIN_61
Button[3]	Input	PIN_63
Button[2]	Input	PIN_67
Button[1]	Input	PIN_69
Button[0]	Input	PIN_71
Buzzer	Output	PIN_4
Clock	Input	PIN_18
Frequency	Input	PIN_20
FTDI_nCTS	Output	PIN_5
FTDI_nRTS	Input	PIN_6
FTDI_RXD	Output	PIN_7
FTDI_TXD	Input	PIN_8
GPIO[11]	Output	PIN_127
GPIO[10]	Output	PIN_73
GPIO[9]	Output	PIN_74
GPIO[8]	Output	PIN_75
GPIO[7]	Output	PIN_76
GPIO[6]	Output	PIN_77
GPIO[5]	Output	PIN_79
GPIO[4]	Output	PIN_94
GPIO[3]	Output	PIN_95
GPIO[2]	Output	PIN_106
GPIO[1]	Output	PIN_107
GPIO[0]	Output	PIN_129
Hex0[7]	Output	PIN_84
Hex0[6]	Output	PIN_88
Hex0[5]	Output	PIN_89
Hex0[4]	Output	PIN_91
Hex0[3]	Output	PIN_93

Рис 9.15. Відповідність контактів для MAX® V – 5M1270Z CPLD

Крок 4. Повторна компіляція програми та отримання.pof файлу для подальшого програмування плати швидкого прототипування.

Командою, що відповідає за розробку та тестування програмного забезпечення для вбудованих систем (рис. 9.16) були розроблені навчальні та методичні матеріали для використання в міжнародному проекті ICo-op [16, 18, 20].

Рисунок 9.16. Команда з тестування вбудованих систем

Розглянемо приклади завдань для розробки варіантів тестування для функціонального тестування.

Завдання 1

Розробити тест-кейс і модифікувати проект test1 у відповідності до завдання: змінити номер функціонального перемикача і кількість світлодіодів, що світяться відповідно до номера варіанту.

Тест-кейс test1

Унікальний ідентифікатор варіанта тестування – test1.

Короткий опис варіанта тестування – крайній справа перемикач sw [0] буде вмикати/вимикати крайній справа світлодіод ld [0].

Порядок виконання – ввімкнути плату, ввімкнути правий перемикач.

Вимоги – тест завантажений на плату, плата підключена до комп'ютера.

Критерій завершеності – при ввімкненому правому перемикачі світиться правий світлодіод, при вимкненому – не світиться.

Категорія тесту – тестування системних компонентів плати.
Автор – Іванов І. І.
Автоматизований – так.

Приклад програми

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity test is
port (
pSW: in std_logic_vector (7 downto 0);
pLED: out std_logic_vector (7 downto 0);
pLED1: out std_logic_vector (7 downto 0)
);
end test;
architecture t1 of test is
begin
pLED (0) <= pSW (0);
pLED (1) <= pSW (0);
pLED (4) <= pSW (1);
pLED (5) <= pSW (1);
pLED1 (0) <= pSW (0);
pLED1 (1) <= pSW (0);
pLED1 (4) <= pSW (7);
pLED1 (5) <= pSW (7);
end architecture;
```

Завдання 2

Розробити тест-кейс і модифікувати проект test2 у відповідності до завдання: змінити номер функціонального перемикача і кількість світлодіодів, що світяться відповідно до номера варіанту.

Тест-кейс test2

Унікальний ідентифікатор варіанта тестування – test2.
Короткий опис варіанта тестування – вмикання крайнього справа перемикача sw [0] буде виводити на дісплей номер варіанту.
Порядок виконання – ввімкнути плату, ввмікнути правий перемикач.
Вимоги – тест завантажений на плату, плата підключена до комп'ютера.

Критерій завершеності – при ввімкненому правому перемикачі на 7 сегментному дисплеї світеться варіант користувача, при вимкненому – не світиться.
Категорія тесту – тестування системних компонентів плати.
Автор – Іванов І. І.
Автоматизований – так.

Приклад програми роботи з кнопками та дісплеєм

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity btn_test is
port (
pbtn: in STD_LOGIC_VECTOR (7 downto 0);
pHex0: out  STD_LOGIC_VECTOR (7 downto 0);
pHex1: out  STD_LOGIC_VECTOR (7 downto 0);
pHex2: out  STD_LOGIC_VECTOR (7 downto 0);
pHex3: out  STD_LOGIC_VECTOR (7 downto 0)
);
end btn_test;
architecture numbers of btn_test is
begin
if (pbtn = «000000001») then
pHex0<=»11110000»;
phex1 <=»00000000»;
phex2 <=»00000000»;
phex3 <=»00000000»;
elsif (pbtn=»00000010») then
pHex1<=»11110000»;
phex0 <=»00000000»;
phex2 <=»00000000»;
phex3 <=»00000000»;
end if;
end numbers;
```

ЛІТЕРАТУРА ДО ЧАСТИНИ 3

1. Arras, P. E-learning concept for the properties of materials remote study/ Arras, P.; Tabunshchyk, G.; Kozik, T.// IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing

Systems (IDAACS), 2013 Volume: 02, 2013, P3: 742–747. DOI: 10.1109/IDAACS.2013.6663024

2. Arras, P. E-learning environment for the remote study in material properties courses/ Arras, P., Kolot, Y., Galyna, T., Kozik, T // International Journal of Computing, 12 (3), 2014, 233–238.
3. Arras, P. Architectural Characteristics and Educational Possibilities of the Remote Laboratory in Materials Properties/ P. Arras, G. Tabunshchyk, Ye. Kolot, B. Tanghe// REV2014 Conference26–28 February 2014, Porto, Portugal, PP. 94–97
4. Arras, P. Iterative Pattern for the Embedding of Remote Laboratories in the Educational Process / P. Arras, K. Henke, G. Tabunshchyk, D. V. Merode // 12th International Conference on Remote Engineering and Virtual Instrumentation (REV 2015) 25–28 February 2015, Bangkok, Thailand, PP. 52–55 (DOI 978–1–4799–7838–0/15)
5. Henke, K. Using Interactive Hybrid Online Labs for Rapid Prototyping of Digital Systems/ G. Tabushchyk, H-D Wuttke, St. Ostendorff, T. Vietzke //REV2014 Conference26–28 February 2014, Porto, Portugal, PP.48–59
6. Henke, K. Using Interactive Hybrid Online Labs for Rapid Prototyping of Digital Systems/G. Tabushchyk, H-D Wuttke, St. Ostendorff, T. Vietzke // iJOE, Volume 10, Issue 5, 2014,– PP. 57–62
7. Kozik, T. Techniques and tools for virtual and remote experiments/ Kozik T., Arras P., Tabunshchyk G. // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: зб. тез доп. VII міжнар. наук.– практ. конф. (Запоріжжя, 17–19 вересня 2014 р.).– Запоріжжя: ЗНТУ, 2014.– С. 112–113.
8. «JTAG Boundary – Тестування сканування в MAX V пристроїв» http://www.altera.com/literature/hb/max-v/mv51008.pdf
9. JTAG Boundary – Тестування сканування для Cyclone IV пристроїв http://www.altera.com/literature/hb/cyclone-iv/cyiv-51010.pdf
10. Modelling, diagnostics and testing of digital systems [Available electronically] / INTUIT. Access mode: http://www.intuit.ru/studies/courses/3440/682/info [RU]

11. Yu. Skobtsov. Logical modelling and testing of digital systems / Yu. Skobtsov,. V. Skobtsov.– Donetsk: IPMM NASU, DonNTU, 2005.–436c [RU]
12. Брагина, Т. И. Информационная технология риск-ориентированного оценивания функциональности web-ориентированных систем /Т.И. Брагина, Г. В. Табунщик // Системи обробки інформації. Вип.2 (118).– 2014.– С. 245–252.
13. Брагина, Т. И. Риск-ориентированный метод тестирования интегрированных баз данных /Т.И. Брагина, Г. В. Табунщик // «Электротехнические и компьютерные системы».– Одесса, 2014.– № 13 (89). – С. 223–230.
14. Табунщик Г. В. Інженерія якості програмного забезпечення: навч. посіб. / Табунщик Г. В., Кудерметов Р. К., Брагіна Т. І.; Запоріз. нац. техн. ун-т.– Запоріжжя: Дике поле, 2013.– 173 с.
15. Табунщик, Г. В. Перспективи впровадження віртуальної інженерії у навчальний процес ЗНТУ/ Г. В. Табунщик, А. В. Пархоменко //Тиждень науки – 2014: зб. тез доп. щоріч. наук.– практ. конф. викладачів, науковців, молодих учених, аспірантів, студентів ЗНТУ (Запоріжжя, 20–25 квіт. 2014 р.). – Запоріжжя: ЗНТУ, 2014.
16. Притула А. В. Практично-орієнтовані методи викладання в галузі вбудованих систем/ Притула А. В., Пархоменко А. В., Табунщик Г.В // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: зб. тез доп. VII міжнар. наук.– практ. конф. (Запоріжжя, 17–19 вересня 2014 р.).– Запоріжжя: ЗНТУ, 2014.– С. 216–217.
17. Шитикова Е. В. Оптимизация процесса исследовательских испытаний сложных технических систем/ Шитикова Е. В., Табунщик Г. В. // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: зб. тез доп. VII міжнар. наук.– практ. конф. (Запоріжжя, 17–19 вересня 2014 р.) .– Запоріжжя: ЗНТУ, 2014.– С. 252–253.
18. Щербак Н. В. Обучающий эксперимент для интерактивной образовательной среды /Щербак Н.В., Табунщик Г. В. //Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: зб. тез доп. VII міжнар. наук.– практ.

конф. (Запоріжжя, 17–19 вересня 2014 р.).– Запоріжжя: ЗНТУ, 2014.– С.254–255.

19. Брагина, Т. И. Модель верификации web-ориентированных систем [Текст] / Т. И. Брагина, Г. В. Табунщик // ІНТЕРНЕТ-ОСВІТА-НАУКА-2014: тези доп. 19-ої міжнар. наук.– практ. конф., ІОН–2014, жовтень 14–17, 2014.– Вінниця: ВНТУ, 2014.– С. 23–25.
20. Табунщик Г. В. Нові технології в підготовці фахівців з вбудованих систем / Г. В. Табунщик, А. В. Пархоменко // ІНТЕРНЕТ-ОСВІТА-НАУКА-2014: тези доп. 19-ої міжнар. наук.– практ. конф., ІОН–2014, жовтень 14–17, 2014.– Вінниця: ВНТУ, 2014.– С.248–250.

CONTENTS

АВТОРЫ/AUTHORS

ПАРХОМЕНКО Анжеліка Володимирівна, кандидат технічних наук, доцент кафедри програмних засобів Запорізького національного технічного університету. Випускниця Запорізького машинобудівного інституту ім. В. Я. Чубаря. Захистила кандидатську дисертацію на тему «Розробка комплексних моделей елементної бази мікроелектронної апаратури для систем автоматизації проектування». Автор більш ніж 100 наукових публікацій та навчально-методичних робіт, у тому числі 1 навчального посібника з грифом Міністерства освіти та науки України, 2 авторських свідоцтв. Основні напрямки наукових досліджень: вбудовані системи управління рухомими об'єктами; технології та системи віртуальної та віддаленої інженерії; CAD/CAM/CAE-системи.

Anzhelika PARKHOMENKO, Ph.D., Associate professor of Software Tools Department of Zaporizhzhya National Technical University. A graduate of Zaporizhzhya Machine-Building Institute named V. Ya. Chubarya. Defended the PhD thesis on topic "Development of complex models of microelectronic equipment components for CAD". Author of over 100 scientific publications and educational works, including one textbook approved by Ministry of Education and Science of Ukraine, 2 Certificates for invention. Main research fields: Embedded Systems for moving objects control; technologies and systems of virtual and remote engineering; CAD/CAM/CAE-system.

ГЛАДКОВА Ольга Миколаївна, аспірантка кафедри програмних засобів Запорізького національного технічного університету. Закінчила магістратуру за спеціальністю «Програмне забезпечення систем». Автор 18 наукових публікацій, 1 авторського свідоцтва, 3 навчально-методичних робіт. Основні напрямки наукових досліджень: CAD/CAM/CAE системи; вбудовані системи управління рухомими об'єктами; Інтернет Речей; технології та системи віртуальної та віддаленої інженерії.

Olga GLADKOVA, a postgraduate student of Software Tools Department of Zaporizhzhya National Technical University. She has a master's degree in the specialty "Software systems". The author of 18 scientific publications, 1 Certificate for invention, three educational works. Main research fields: CAD/CAM/CAE systems; Embedded Systems for moving objects control; Internet of Things; technologies and systems of virtual and remote engineering;

ПОЛЯКОВ Михайло Олексійович, к.т. н., доцент кафедри електричних та електронних апаратів ЗНТУ. Після закінчення в 1974 р. ЗМІ (ЗНТУ) працював розробником систем управління в НДІ "Марс" в Росії. Науково-педагогічний стаж – близько 40 років в університетах Києва, Ульяновська, Запоріжжя, Дніпропетровська, автор понад 100 наукових праць. Сфера наукових інтересів: проектування систем управління з контролерами, моніторинг і прогнозування технічного стану трансформаторів.

Mykhailo POLIAKOV, PhD., associate professor at department of Electrical and electronic devices at ZNTU. After graduating in 1974 ZNTU worked as a developer of control systems at the Research Institute "Mars" in Russia. Scientific-teaching experience – 40 years in universities of Kiev, Ulyanovsk, Zaporozhe, Dnepropetrovsk, author of more than 100 scientific works. Research interests: design of control systems with controllers, monitoring and forecasting of technical states of transformers.

ЛАРІОНОВА Тетяна Юріївна, асистент кафедри «Електричні та електронні апарати», аспірант Запорізького національного технічного університету (ЗНТУ). Закінчила магістратуру ЗНТУ за спеціальністю «Електричні машини та апарати» в 2011-му році. Має 7 наукових праць. Коло наукових інтересів: проектування інтелектуальних систем керування з контролерами, розробка перетворювачів постійного струму, енергоефективність систем енергоживлення.

Tetiana LARIONOVA, teaching assistant at department of Electrical and electronic devices, postgraduate student at the Zaporozhye National Technical University (ZNTU). Graduated from ZNTU with a master degree in "Electrical machines and apparatuses" in 2011, has 7 scientific papers. Scientific interests: design of intelligent control systems with controllers, development of DC-DC converters, power efficiency of electricity supply systems.

ТАБУНЩИК Галина Володимирівна, к.т.н., доцент, професор Запорізького національного технічного університету. Закінчила Запорізький державний технічний унверситет за спеціальністю програмне забезпечення автоматизованих систем, захистила дисертацію на звання кандидата технічних наук за спецальністю 05.13.03 – системи і процеси керування. Автор понад 100 наукових праць. Наукові інтереси – інженерія програмного забезпечення, верифікація інформаційних систем, програмування вбудованих систем, керування ризиками.

Galyna TABUNSHCHYK, PhD, Prof of Software Tool Department of Zaporizhzhya National Technical Univerity. Graduated from Zaporizhzhya National Technical University with speciality Software Engineering, in 2004 finished PhD work in control systems and process. Have more than 100 scientific works. Scientific interests – Software Engineering, System Verification, Embeded Systems, Risk Management.

КАПЛІЄНКО Тетяна Ігорівна, старший викладач кафедри програмних засобів Запорізького національного технічного університету. Закінчила магістратуру за спеціальністю «Програмне забезпечення автоматизованих систем», аспірантуру за спеціальністю «Інформаційні технології». Автор 33 наукових публікацій, 2 авторських свідоцтв і одного патенту. Основні напрямки наукових досліджень: аналіз та верифікація якості програмного забезпечення; керування програмними проектами; керування ризиками.

Tetiana KAPLIIENKO, a senior lecturer of Software Tools Department at Zaporizhzhya National Technical University. She has a master's degree in "Automatic systems software" and has finished the postgraduate course in "Information technology". She is the author of 33 scientific publications, 2 Certificates for invention and 1 patent. Main research fields: the analysis and verification for the software quality; software design managements; risks management.

Appendix 1. Using Interactive Hybrid Online Labs for Rapid Prototyping of Digital Systems

K. Henke[1],G. Tabunshchyk[2],H.-D. Wuttke[3],T. Vietzke[4], St. Ostendorff[5]

iJOE Volume 10, Issue 5, 2014
(http://dx.doi.org/10.3991/ijoe.v10i5.3994)

Abstract

The Ilmenau Interactive Hybrid Online Lab offers several fields of application. In this article an enhancement of the existing Online Lab will be described to provide additional functionalities for a Web-based rapid prototyping of digital systems as well as the Web-based verification of such systems. For this new operation mode of the online lab a special rapid-prototyping board for digital systems was developed to fulfill the design tasks of digital systems. All the components of the rapid prototyping board will be described in detail. The implemented online lab infrastructure allows to interconnect online labs and to exchange remote lab experiments among different universities worldwide.

Index Terms

control engineering education, laboratories, Web-based education, virtual and remote labs, Web-based design tools, distance learning, rapid prototyping.

1 Ilmenau University of Technology, Ilmenau, Germany
2 Zaporizhzhya National Technical University, Zaporizhzhya, Ukraine
3 Ilmenau University of Technology, Ilmenau, Germany
4 Ilmenau University of Technology, Ilmenau, Germany
5 Ilmenau University of Technology, Ilmenau, Germany

INTRODUCTION

In our contribution we would like to present an enhancement of the Ilmenau Interactive Hybrid[6] Online Lab to provide additional functionalities for a Web-based rapid prototyping of digital systems as well as the Web-based verification of such systems. Facilities of Hybrid Online Labs provide permanent online access for students and supervisors, and give possibilities to check different parts of the designs most easily. This gives possibilities to realize correct designs, to organize self-study process of the student more efficiently, to control student's work and to broaden the ways of communication in research work with companies. This solution is intended for the use in teaching materials dealing with the design of digital control systems and embedded systems – from the basics up to complex design tasks as well as within the newly established Tempus project "ICo-op – Industrial Cooperation and Creative Engineering Education based on Remote Engineering and Virtual Instrumentation", founded by the European Commission of the program "Tempus", Grant No 530278TEMPUS-1-2012-1-DE-TEMPUS-JPHES [1].

The main topic of this Tempus project is to empower university-enterprise partnerships in Armenia, Georgia, and Ukraine by modernizing engineering education based on remote engineering and virtual instrumentation enhanced with transversal knowledge and competences at universities. It also offers new possibilities for bidirectional cooperation between universities and enterprises in education and research. The remote teaching of enterprise's staff or usage of universities remote labs for research purposes is one example for such cooperation. In order to achieve this, Ilmenau Interactive Hybrid Online Labs will be put in place at several project partners to develop common learning modules and to interchange these modules between all the partners based on EU best practices, partners' industry expertise, and knowledge of business demand of target countries.

Within the Tempus project a bilateral agreement between Ilmenau University of Technology and Zaporizhzhya National Technical University (ZNTU) was signed as well. The main purpose of this cooperation

[6] Hybrid online labs provide both remote experiments on real electromechanical models (physical systems) in the remote lab as well as simulation models of these physical systems in virtual labs [2].

is to engage in joint scientific work in the fields of Automation Systems, Computer and Software Engineering and Remote Engineering, using the Ilmenau Interactive Hybrid Online Lab for this purpose. The Institute of Computer Science and Radio Electronics of ZNTU is in close contact with the leading specialized enterprises of the Ukrainian region. But unfortunately it is tended to reduce the number of diplomas ordered by enterprises during last years. One of the factors is difficulties in communication between student, university supervisor and company supervisor. Further is considering how Hybrid Online Lab can solve this problem as well providing an effective remote tool for different level designs.

RAPID PROTOTYPING OF DIGITAL SYSTEMS

With the described enhancement of the functionality we want to provide exciting and challenging Web-based lab experiments in the field of digital system design. Course material starts with the basics of Boolean algebra, combinational logic and simple sequential circuits. This is followed by various minimization techniques for logical expressions, dynamic effects in combinational and sequential circuits and the design of digital control systems based on Finite State Machines (FSM). Finally we offer different methods and tool concepts to create, implement and validate digital systems to solve complex design tasks.

The goal is to introduce methods and technologies for a rapid prototyping of digital (embedded) control systems, especially the hardware oriented part of these systems. In connection with the mentioned Tempus project design engineers working in industry may also want to consider the offered learning scenarios of the project to handle modern CAE tools, logic simulation and logic synthesis using hardware description languages (e.g. VHDL), design hierarchy, and current generation of field programmable gate array (FPGA) technology – if they have not had previous experience with these rapidly evolving technologies. With modern logic synthesis tools and large FPGAs, more advanced designs are needed to present challenging laboratory projects.

ARCHITECTURE

The concept and the architecture as well as different fields of application of the Ilmenau Interactive Hybrid Online Lab were presented in various publications during the last years in detail, e.g. in [3, 4, 5]. A special rapid prototyping board for digital systems was developed for the new operation mode of the Ilmenau Interactive Hybrid Online Lab, presented in this paper to fulfill the mentioned design tasks. All the components of the rapid prototyping board will be described in detail in the following sections.

The Ilmenau Interactive Hybrid Online Lab

Figure 1 gives an overview about the Ilmenau Interactive Hybrid Online Lab. The infrastructure is based on a universal grid concept which guaranties a reliable, flexible as well as robust usage of this online lab. A more detailed description of this grid concept as well as the main components is presented in [5, 6].

The server side infrastructure (remote lab) consists of three parts:

an internal serial remote lab bus to interconnect all parts of the remote lab, realized as CAN bus, a bus protection unit (BPU) to interface the control units to the remote lab bus and to protect the bus from blockage, misuse and damage as well as a physical system protection unit (PSPU), which protects the physical systems (the electro-mechanical models in the remote lab) against deliberate damage or accidentally wrong control commands and which offers different access and control mechanisms.

The interconnection between the Web-control units and the selected physical systems during a remote lab work session (experiment) as well as the webcam handling is done by the lab server as part of the remote lab infrastructure.

During a running experiment, the client application will interact directly with the online lab grid infrastructure. Based on this infrastructure we offer several operation modes, described in detail in [3, 5]. In the following we would like to present an enhancement of the existing operation modes to provide additional functionalities for a Web-based rapid prototyping of digital systems as well as the Web-based verification of such systems.

The Rapid Prototyping Board

To fulfil all the mentioned design tasks for the design of digital systems, we have developed a special rapid prototyping board, shown in Figures 2 and 8. Over the last years, a number of interesting and challenging rapid prototyping boards were developed for educational purposes, the UP 3 from Altera [7] or development boards from Xilinx [8]. But most of them support very specific design tasks – not the whole spectrum, needed from the beginning (e.g. students in the first semester) to exciting and challenging design tasks in high-level courses.

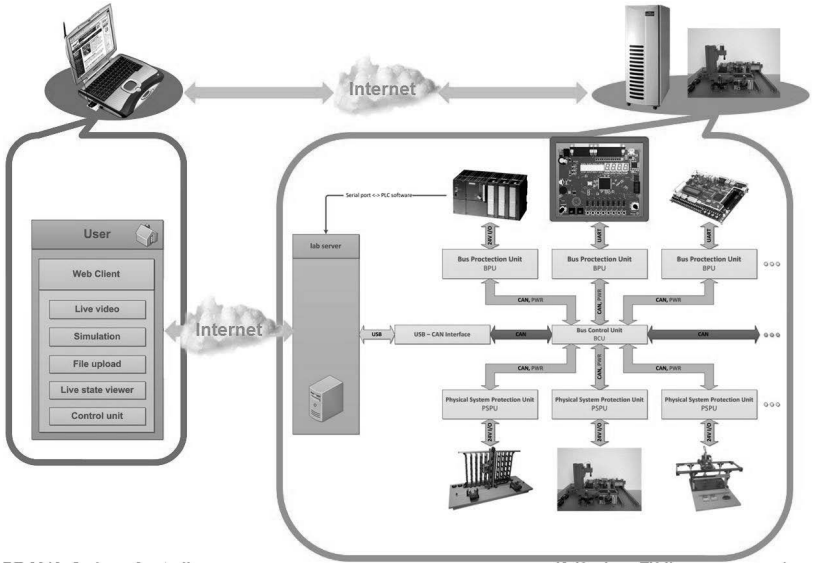

Figure 1. Overview of the Ilmenau Interactive Hybrid Online Lab infrastructure

Our rapid prototyping board is based on the MAX® V 5M1270Z CPLD from Altera [9]. With its mix of low price, low power, and new features, the MAX V CPLD family delivers the market's best value. Featuring a unique, non-volatile architecture and one of the industry's largest density CPLDs, MAX V devices provide robust new features at up to 50 percent lower total power compared to competitive CPLDs. The MAX V architecture integrates previously external functions, such as flash, RAM, oscillators, and phase-locked loops [9].

MAX V CPLDs are supported by the free Quartus® II Web Edition development software. With Quartus II software, students will get productivity enhancements resulting in faster simulation, faster board bring-up, and faster timing closure [10].

Finally, students can use the free Quartus II simulator tool QSim (with an integrated waveform editor). QSim is a graphical user interface (GUI) that is used to run simulations and launch the Waveform Editor. Also, QSim is used to set whether the simulation should be a functional or timing simulation. The Vector Waveform Editor is used to draw the test input signals for the simulation and select which signal should be shown in the simulation results [11]. In High-level courses students can use the more powerful simulation tool ModelSim® to validate the design. ModelSim supports behavioral and gate-level simulations, including VHDL testbenches [12].

Besides the MAX V CPLD the rapid prototyping board consists of the following components (see Figure 2):

- Input buttons:
- 8 push buttons:
- PB_7.. PB_4 (active low)
- PB_3.. PB_0 (active high)
- 2 rotary hexadecimal encoder
- 8 slide switches
- LED outputs:
- 4 7-segment displays (active low)
- 1 LED bar display with 8 LED (active low)

Other components:

- 10.000 MHz crystal oscillator
- Frequency synthesizer (200 Hz.. 10 kHz)
- Piezoelectric oscillator (for “sounds”)
- Incremental encoder
- UART (over USB)
- 25-pin SUB-D connector (to connect additional hardware boards, e.g. for a VGA adapter)

To program the MAX V CPLD on this stand-alone version of the rapid prototyping board, an additional FTDI chip [13] was placed on the board. The student has to connect the prototyping board via USB to his PC or laptop to upload the synthesized CPLD design to the FTDI chip at

the prototyping board. This chip will program the CPLD automatically via JTAG interface (see Figure 3).

The Online Lab Rapid Prototyping Mode

For a Web-based usage of the rapid prototyping board, an additional “interconnection” FPGA is placed on the bottom side of the PCB to realize the communication with the remote lab infrastructure (see Figure 4). All inputs of the board (buttons, synthesizer, incremental encoder and oscillator) have to be removed from the PCB and are replaced by a direct connection to the outputs of the “interconnection” FPGA, which will set all input signals according to the user’s input via the user interface at the student’s client PC. The generated outputs of the prototyping board can be directly read by the “interconnection” FPGA without removing any LED. The FPGA is interconnected to the BPU within the remote lab infrastructure.

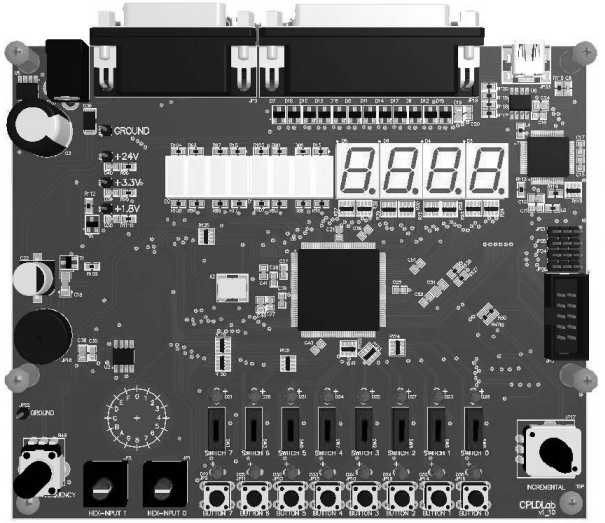

Figure 2. Rapid prototyping board (schematic view)

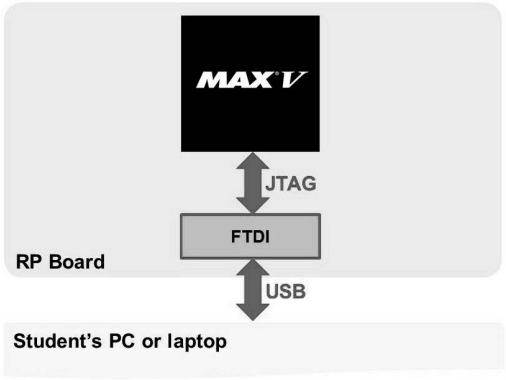

Figure 3. Programming of the stand-alone rapid prototyping board

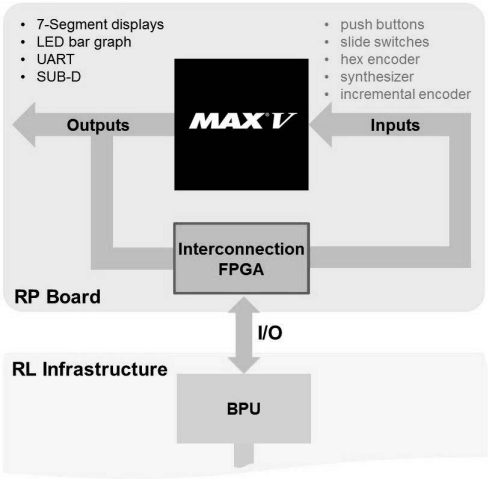

Figure 4. Interfacing the rapid prototyping board to the remote lab

To program the MAX V CPLD on the prototyping board in the remote lab via the Internet, the existing remote lab infrastructure can be used, as shown in Figure 5. The student has to upload his synthesized design (FPGA programming file) via the RIA (Rich Internet Application – a consistent enhancement of previous Java applets – see next section) on his client PC at home to the remote lab. Then the remote lab server will forward the data to the corresponding BPU (compare Figure 1). The BPU in turn will program the connected CPLD automatically via JTAG.

The Web-based User Interface

The increasing capacity of wireless communication and the growing number of mobile devices (e.g. smartphones and tablets) on the one hand as well as modern Internet technologies like JavaScript, HTML5 and Web Sockets on the other hand provides new possibilities and challenges in the area of mobile learning. Therefore a realization as HTML5 RIA was chosen for the Web-interface. Figure 6 gives an impression of this Web-interface.

By using the client‘s Web-interface, the student is able to upload the synthesized CPLD code of the design to program the CPLD on the rapid prototyping board (automatically in the remote lab – see section III.C) and to handle the whole lab procedure. This Web-interface allows the student to manipulate all the inputs of the rapid prototyping board virtually (slide switches, hex coding switches, pushbuttons and incremental encoder).

For the look-and-feel of the RIA, we use a visual model of the prototyping board (on the upper left side). All the inputs are realized as HTML5 control elements and can be activated via a mouse interactively. Changes are immediately sent to the rapid prototyping board in the remote lab and the corresponding results are displayed again inside the visual model. Furthermore a webcam will be used to observe the rapid prototyping board (on the upper right side) to watch the results of the user’s actions directly as reaction in the remote lab.

FIELDS OF APPLICATION

In this section possible fields of application of the developed rapid prototyping platform will be discussed.

Following the design flow for digital systems, students have to use common design tools to implement their control tasks. As mentioned in Section III.B they have to use Altera’s development system (Quartus II, simulation tools).

After synthesizing the bit file as shown in Figure 7, students can use the prototyping board as stand-alone system (see Section IV.A) or Web-based rapid prototyping system (see Section IV.B and IV.C).

Stand-alone Rapid Prototyping System

The simplest way to use the designed hardware platform is as a stand-alone solution for the rapid prototyping of digital systems (without Internet connectivity), shown in Figure 8. This application is not the focus of this article and therefore not discussed in detail.

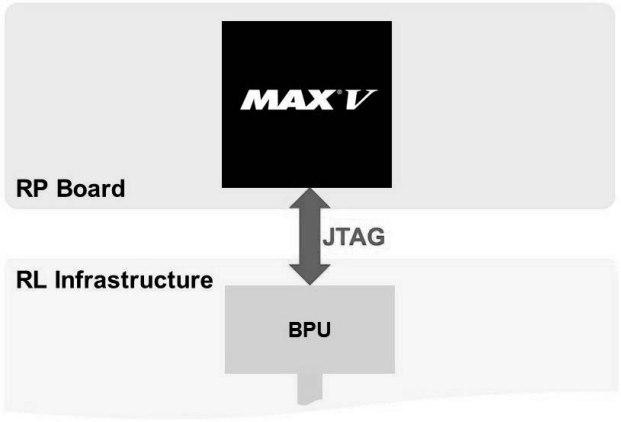

Figure 5. Web-based programming of the rapid prototyping board via the remote lab

Figure 6. Web-interface of the rapid prototyping board (RIA)

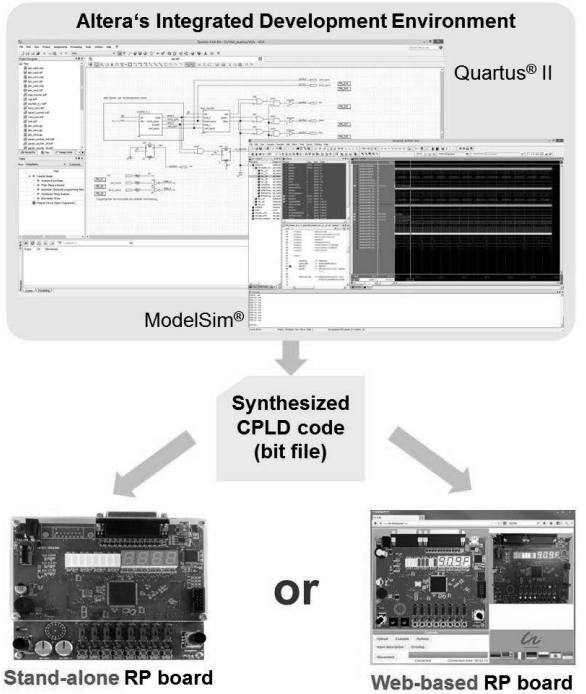

Figure 7. Using Altera's IDE and the rapid prototyping board for the design of digital control tasks

Web-based Prototyping of Digital systems

By using the mentioned Altera's development system, students are able to specify their design via:

- Text based design methods,

Here the student can enter his design by means of logical equations, truth tables or hardware description languages (AHDL, VHDL or Verilog), as shown in Figure 9.

- Graphically based design methods,

The student can use block or schematic diagrams to input his design, as shown in Figure 10.

- Integrated FSM editors.

In case of sequential design tasks he can directly enter the derived automaton graph (or graphs of parallel automata) with the built-in FSM editor, as shown in Figure 11.

The editor itself generates VHDL code for the further design steps.

Furthermore, students can specify, describe, implement and verify different digital systems using “self-made” IP core libraries (e.g. digital control systems, serial communication modules, robot sensors control, models of RISC processors, etc.).

After specification of the given task, students have to simulate their design. They can choose between different simulation tools within Altera’s development system, e.g. ModelSim for challenging designs in high-level courses.

Once the design is completed and error free, the student can upload the synthesized design to the remote lab, program the CPLD on the prototyping board (as described in Section III.C) and can test his solution using the rapid prototyping platform.

Web-based Validation of Digital Systems

Another use case of the rapid prototyping board is to identify the function of a given design (black box) or to find malfunctions of a given well-known design.

The CPLD will be pre-programmed by the teacher and the student has no ability to reprogram the device.

By manipulating the inputs of the unknown black box (e.g. to enter all the input sets of a truth table or special input sequences), the student attempts to analyze the response (the real output signals) of the board to find out the function of the given design. The student has to enter his test vectors based on truth tables or input sequences one by one using the provided controls and observe the results.

It is planned to support the upload of a whole truth table or input sequence as a text file via the Web-interface of the prototyping board. In this case the student will be able to record the responses corresponding to the specified inputs creating a waveform file. This file can then be used for further investigation of the current design.

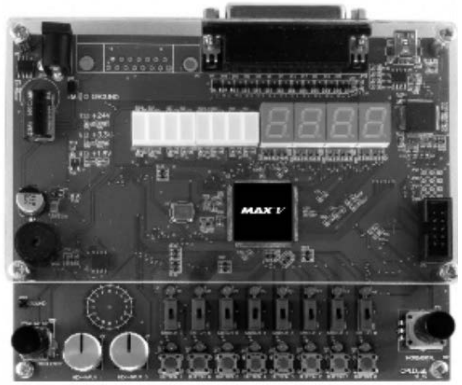

Figure 8. Rapid prototyping board

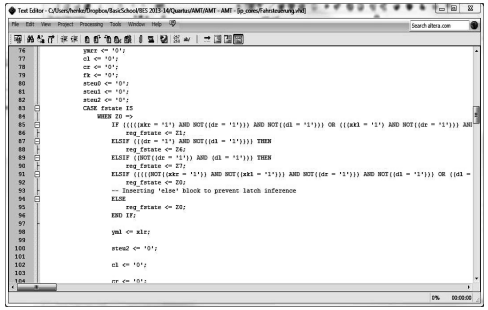

Figure 9. VHDL text-based specification

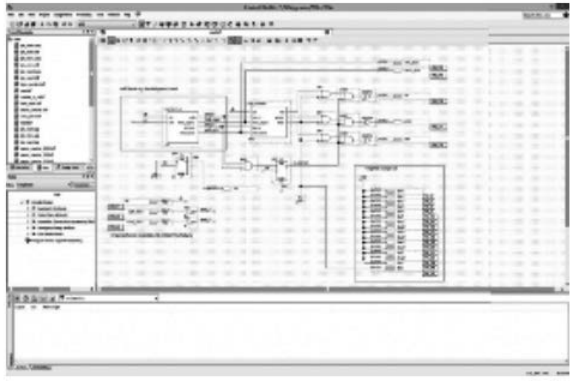

Figure 10. Block diagram-based specification

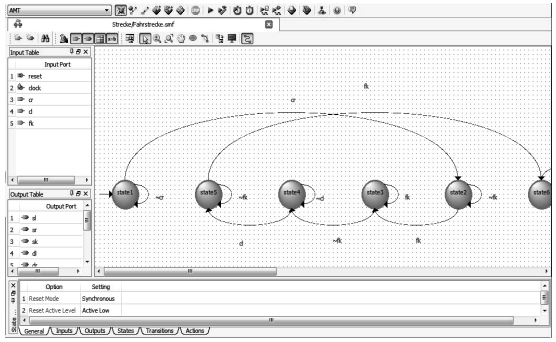

Figure 11. FSM-based specification

CONCLUSION

A universal hardware platform has been discussed, which can be used for rapid prototyping of digital systems. It is used for teaching purposes, from the basics to complex control design tasks including the ability for both local and web-based remote access.

Within several new European projects in the area of "Remote Engineering" e.g. ICo-op [1], eScience [14] and DesIRE [15] it is increasingly necessary to allow and organize a shared use of equipment. Therefor the main focus is a Web-wide usage of design tools and remote labs for the design of digital systems.

Using both, online tool support and laboratories, has the potential of removing the obstacles of cost, timeinefficient use of facilities, inadequate technical support and limited access to design and laboratory resources. This would also benefit students and researchers with special needs and students/researchers working from home, so they do not have to travel to their companies' facilities to perform their work. Even students/researchers working at their university's facilities can use remote specialized equipment at another university without travelling.

Students will learn more about the possibilities and limits of remote control and observation via Internet on practical examples.

The student – besides knowledge consolidation by executing design and practical experiments using these new Internet technologies – is forced to estimate the technologies critically.

ACKNOWLEDGMENT

The authors would like to acknowledge the work of Alexander Härtel and Tobias Fäth in the development of the Web-based programmer as well as the Web-based user interface (RIA).

REFERENCES

[1] ICo-op project Website: http://www.ico-op.eu.

[2] K. Henke, St. Ostendorff, H.-D. Wuttke, Th. Vietzke, Ch. Lutze, "Fields of Applications for Hybrid Online Labs", International Journal of Online Engineering (iJOE), Vol 9 (2013) Special Issue REV2013; pp. 20–30 , Vienna, May 2013.

[3] K. Henke, St. Ostendorff, St. Vogel, H.-D. Wuttke, "A Grid Concept for Reliable, Flexible and Robust Remote Engineering Laboratories", International Journal of Online Engineering (iJOE), Vol 8 (2012), pp. 42–49, Vienna, December 2012.

[4] K. Henke, St. Ostendorff, H.-D. Wuttke, Th. Vietzke, Ch. Lutze, "Fields of Applications for Hybrid Online Labs", Remote Engineering & Virtual Instrumentation, REV2013, Sydney, Australia, 06–08 February 2013.

[5] K. Henke, St. Ostendorff, H.-D. Wuttke, "A Flexible and Scalable Infrastructure for Remote Laboratories Robustness in Remote Engineering Laboratories", The Impact of Virtual, Remote and Real Logistics Labs ImViReLL2012 in: CCIS 282 pp. 13–24, Springer Verlag, DOI: 10.1007/978-3-642-28816-6_2, Bremen, Berlin, Heidelberg, February 2012. http://dx.doi.org/10.1007/9783-642-28816-6_2

[6] K. Henke, St. Ostendorff, St. Vogel, H.-D. Wuttke, "A Grid Concept for Reliable, Flexible and Robust Remote Engineering Laboratories", International Conference on Remote Engineering and Virtual Instrumentation, REV2012, Bilbao, Spain, July 04–06, 2012.

[7] Altera Corporation, http://www.altera.com. [8] Xilinx, Inc., http://www.xilinx.com.

[9] MAX V CPLD, http://www.altera.com/devices/cpld/max-v/mxvindex.jsp

[10] Quartus II Web Edition Software, http://www.altera.com/products/software/quartus-ii/subscription-edition/qts-se-index.html.

[11] Quartus II Simulator Tools for Education, http://www.altera.com/education/univ/software/qsim/unvqsim.html.

[12] ModelSim-Altera Software, http://www.altera.com/products/software/quartus-ii/modelsim/qts-modelsim-index.html

[13] http://www.ftdichip.com/Products/ICs/FT2232H.htm

[14] eScience Website, http://www.esience.org

[15] DesIRE Website, http://tempus-desire.thomasmore.be

AUTHORS

Karsten Henke is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: karsten.henke@tu-ilmenau.de).

Galina Tabunshchyk is with the Zaporizhzhya National Technical University, Software Tools Department, 69063 Zaporizhzhya, Ukraine, Zhukovskogo 64, (e-mail: galina.tabunshchik@gmail.com).

Heinz-Dietrich Wuttke is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: dieter.wuttke@tu-ilmenau.de).

Tobias Vietzke is with the Ilmenau University of Technology, Master Student in Computer Engineering at the Faculty of Computer Science and Automation, 98684 Ilmenau, Germany, POB 10 05 65. He obtained his BSc. in Computer Engineering in 2011 (e-mail: tobias.vietzke@tu-ilmenau.de).

Steffen Ostendorff is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: steffen.ostendorff@tu-ilmenau.de).

This work was supported in part by the European Commission within the program “Tempus”, “ICo-op – Industrial Cooperation and Creative Engineering Education based on Remote Engineering and Virtual Instrumentation”, Grant No 530278-TEMPUS-1-2012-1-DE-TEMPUSJPHES. Submitted 01 July 2014. Published as resubmitted by the authors 13 September 2014.

Appendix 2. Fields of Applications for Hybrid Online Labs

K. Henke,St. Ostendorff,H.-D. Wuttke,T. Vietzke and Ch. Lutze (Ilmenau University of Technology,Ilmenau,Germany)

iJOE – Volume 9, Special Issue 3, 2013
(http://dx.doi.org/10.3991/ijoe.v9iS3.2542)

Abstract

Based on a grid concept of an interactive hybrid online laboratory we will describe different fields of applications in different learning scenarios. The infrastructure is based on a universal grid concept which guaranties a reliable, flexible as well as robust usage of this online lab. By using the online lab, students are able to design control algorithms with different specification techniques to control electromechanical models in the online lab. Additionally, the reconfigurable rapid prototyping platform of the REAL system can be used to test all the taught topics of a given lectures in the field of digital system design. Finally, a special demonstration platform (a ball in a labyrinth on a balance plate) can be used to give the students a better feeling about the possibilities and limitations of remote control and observation via Internet and to evaluate these technologies critically. The implemented online lab infrastructure is based on the iLab architecture of the MIT, which allows to interconnect online labs and to exchange remote lab experiments among different universities worldwide.

Index Terms

control engineering education, laboratories, Web-based education, virtual and remote labs, Web-based design tools, distance learning.

INTRODUCTION

Our Integrated Communication Systems Group at the Ilmenau University of Technology has many years of experience in integrated hardand software systems and over 10 years of experience in dealing with Internetsupported teaching in the field of digital system design ([1] and [2]).

We have developed a new teaching concept, called “Living Pictures” [3] that we use in several phases of the learning process. Living Pictures are highly interactive Java applets that can be used for demonstrations as well as for experimental purposes, and also serve as tools in certain steps of the design process of digital systems. To complete the learning outcomes by own experiences, the students have to pass hands on examination in a lab. A task during this examination is to design an algorithm for a control system that controls one of various physical systems, for instance an electromechanical model of an elevator or a production cell.

For all students, hands-on experiences are important to deepen their knowledge about topics they learned during lectures (see Figure 1). At our university we offer an online laboratory, which gives the students the possibility to work on real physical systems without the need to stand in line at a lab or the need to take care of opening hours.

With our hybrid[7] online lab we want to offer the students a working environment that is as close as possible to a real world laboratory. Under real laboratory conditions disturbances can appear and lead to failures of the control algorithm that cannot be detected under virtual lab conditions.

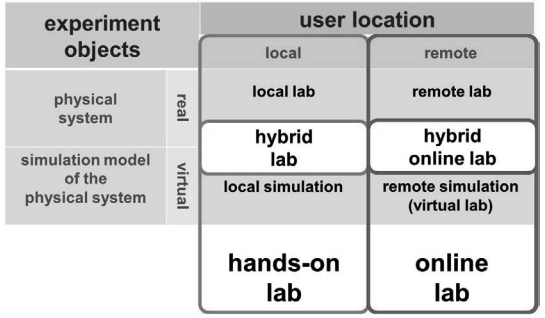

Figure 1. Classification of lab experiments

Therefore, it is important to include such real disruptive factors for a closer relation to practical conditions. Furthermore, the online lab should offer the students stimulus with regards to the design of safety critical control systems. By the conception of our lab (described in the next section), the virtual worlds were embedded within a real laboratory.

[7] Hybrid online labs provide both remote experiments on real electromechanical models (physical systems) in the remote lab as well as simulation models of these physical systems in virtual labs.

In principle, we would like to concentrate on giving students the chance to check their prepared control algorithms in real environmental conditions, which they can interactively influence themselves, and correct or modify the received results

- via Web-based simulation,
- via Web-based remote control of the existing physical system (that means before the tutorial course).

Currently, there are no common standards for the architecture of remote labs. This is the reason why universities develop their own remote lab solutions – suitable for their specific requirements, for example [4], [5], [6] and [7]. This handicaps a networking of different online labs among each other and also a usage by different institutions. Missing uniform interfaces prevent the programming of universal plug-ins and other software modules.

ARCHITECTURE OF THE REAL SYSTEM

In the following, we will describe a hybrid interactive online lab, supporting all the design steps for complex control tasks to control various electromechanical models the REAL system. Goal of the REAL (Remote and Applications Laboratory) system is to show new ways and chances of remote controlling and remote observation of real processes (e.g., in the fields of control engineering, robotics, tele-control engineering), dealing with the integrated and interactive usage of modern Internet and intranet technologies, like HTML5, jQuery, JavaScript etc. It was developed at the Department of Integrated Communication Systems at the Ilmenau University of Technology [8] – see Figure 2.

The implemented REAL infrastructure is based on the iLab Shared Architecture of the MIT (see Figure 3) which meanwhile is established as a standardized implementation for online laboratories and will be implemented in more and more locations all over the world. Furthermore, it allows to interconnect online labs and to exchange remote lab experiments among different universities worldwide.

As mentioned in many papers (e.g., [9], [10] and [11]), interactive online labs can open opportunities which allow an experimental approach for a wider audience and are also independent of opening hours of the laboratory rooms and their staff.

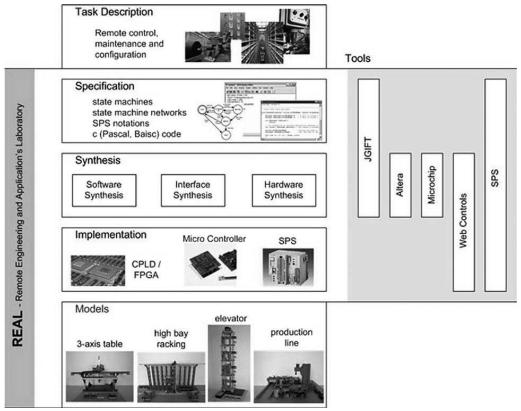

Figure 2. Overview of the REAL system

Interactive labs will be used, when:

- an experiment requires real-time processing or
- the user wants to observe the whole physical system (the electro-mechanical models) during the experiment or
- some parameters (e.g., input variables to a control system) need to be changed interactively during the experiment.

They offer various features like visualization and animation, which allows to observe and to test all the properties of the design. In connection with formal design techniques, simulation and prototyping are used to establish a foundation for the development of a reliable system design. To check the functionality of the whole design, some special simulation and validation features are included as integral part of the REAL system. This offers various possibilities for the execution of simulations, such as:

- usage of simulation models of the physical system for visual prototyping,
- step by step and parallel execution of these prototypes,
- visualization of the simulation process with the tools also used for specification,
- features for test pattern generation and
- code generation for hardware and software synthesis.

REAL offers a Web-based environment supporting the above mentioned features to generate and execute a design by using simulation models. An example will be given in Section III.A.

At any time the students have the chance to adjust their algorithms in case of faults. Therefore, they are able to achieve a fault free solution (a validated control algorithm) step by step. For more details see the previous publications [2], [12] and [13].

Our online lab is used for teaching practical lessons as well as giving hands-on experiences for the development of embedded electronics. This is not done using on-site lessons, but remote via the Internet. This has the advantage that courses can be offered internationally world-wide and gives students from different countries, speaking different languages, the same access and equal possibilities in the lab.

Figure 3. iLab Shared Architecture of the MIT (4)

Additionally, even for local students, the lab offers extended opening hours (twenty-four-seven) when compared to a regular lab. Besides the advantages for students, this also reduces the costs for academic teaching and improves the quality by offering more practical training possibilities.

One implementation challenge is to protect the physical systems in the lab against wrong control algorithms of students without defining too many design constraints. Students should be free in their decisions and develop own creative solutions. They can implement their own design strategies, therefore a reference design and a method to check the students' design against this reference is needed [14] to protect the physical system in the lab (see Figure 4).

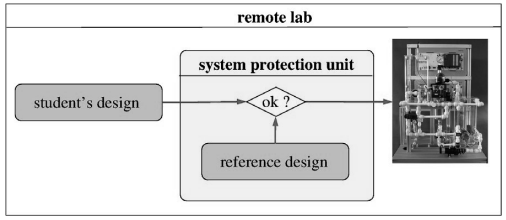

Figure 4. Observation of the student's design by the physical system protection unit

The reference design should be independent of the used control unit and the development tools. This is done by the protection unit of the physical system.

Figure 5 illustrates the grid architecture of the REAL system. The server side infrastructure (remote lab) consists of three parts:

- an internal serial remote lab bus to interconnect all parts of the remote lab,
- a bus protection unit to interface the control units to the remote lab bus and to protect the bus from blockage, misuse and damage as well as
- a physical system protection unit, which protects the physical systems (the electro-mechanical models in the remote lab) against deliberate damage or accidentally wrong control commands and which offers different access and control mechanisms.

For a more detailed description of this grid concept as well as of the main components see papers [15, 16].

The interconnection between the Web-control units and the selected physical systems during a remote lab work session (experiment) as well as the webcam handling is done by the lab server as part of the iLab architecture (see Figure 3).

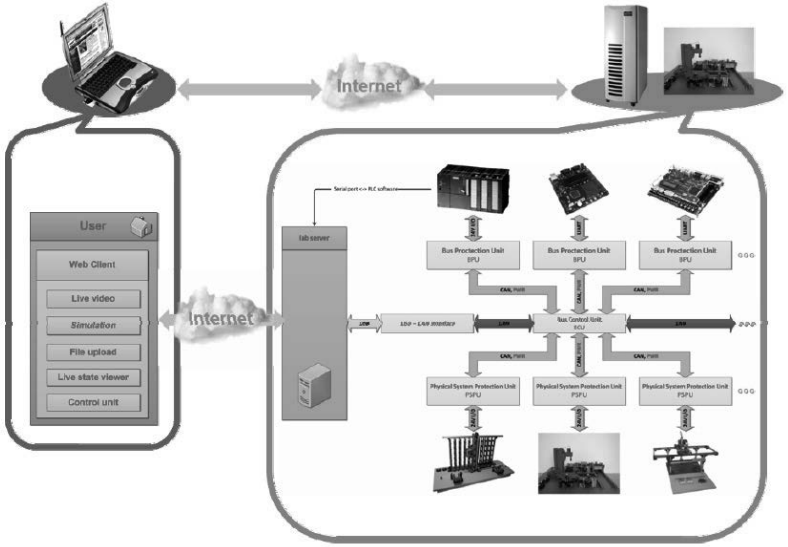

Figure 5. Grid architecture of the REAL system (server side) with Web-client

The iLab Service Broker is mainly responsible for the user management and time scheduling. All experiments are made available by this service and integrated into the iLab Cloud.

During a running experiment, the client application will interact directly with the online lab infrastructure as depicted in Figure 5 without any more access to the Service Broker. Details about the iLab architecture are beyond the scope of this paper. Please refer to [17] for further information.

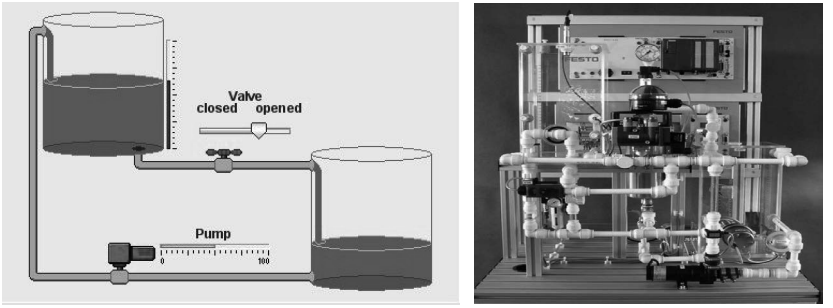

Figure 6. Simulation model and physical system of a water level control

FIELDS OF APPLICATIONS

Based on this flexible online lab structure we offer different operation modes to test the developed control task. Simulation and visual prototyping help to find functional errors. Before starting practical work on real systems, simulations and animations in “virtual worlds” are often used to verify the developed solutions. The behaviour of the physical system that should be controlled, as well as its environment, will be emulated as a simulation model. The student can influence this “virtual world” and analyze the caused reaction of his control algorithm. Figure 6 shows an example of such a simulation model.

These steps have to be executed until no more errors are detected. But there is an essential disadvantage in this method. Real disruptive factors (e.g., failure of single components, mechanical problems or process variations) cannot be recognized by the underlying virtual environmental model.

Generally, only a simulation of predetermined malfunctions is possible. After some time, all these effects are well known in the student’s

community. Unconsidered sources of errors lead to undetected failures of the control because the corresponding environmental situation was not simulated before [13]. That is why a fault free design algorithm finally should be tested on real physical systems (e.g., the water level control, shown in Figure 6) in the online laboratory as well. In the following we will describe possible operation modes of the REAL system based on the schematic view in Figure 5:

- Stand-alone Mode (visual prototyping)
- Remote Control Mode (via Web-client)
- Remote Control Mode (via control unit)
- Virtual Control Mode (visual prototyping)
- Virtual Control Mode (test mode)
- Local Control Mode (via control unit)
- Local Control Mode (manually)
- Rapid Prototyping Mode
- Visitor Mode

The complete schematic view (client and server side) of the online lab architecture is shown in Figure 7.

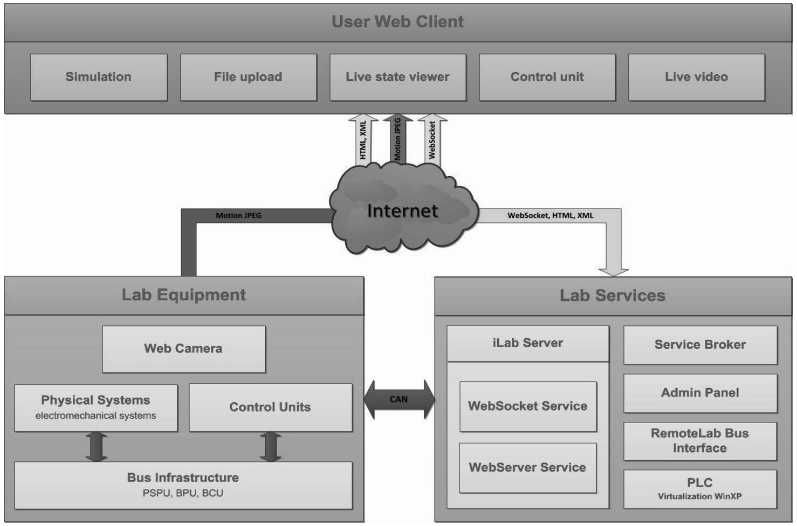

Figure 7. Schematic view of the remote lab components

Via the Web-client running on the student's PC at home the physical system (e.g., the electromechanical model of the water level control) can be controlled by using different control units (e.g., microcontroller, FPGA, PLC).

For the Web-clients modern Internet technologies like HTML5, jQuery and JavaScript can be used. These technologies make rich internet applications (RIAs) possible, which run in nearly any browser and on many terminal devices without needing any plugins.

These RIAs carry out many functions like animation, simulation, interpretation and control and therefore release the server from these tasks. The interaction with the user is speedup and can react faster and more direct.

By using this Web-interface, the student is able to:

- handle the experiment (e.g., start, stop, reset),
- change environmental variables if necessary and
- watch the experiment by manipulating environmental variables inside an I/O monitor or by observing the control of the physical system directly via a webcam.

Via an optional local control panel the student is able to manipulate the lab environment (e.g., the water level in the tank) or the actuators (e.g., the pump) when working on-site. Heart of this architecture is the physical system protection unit, which is described in detail in [15, 16].

Stand-alone Mode – Visual Prototyping

In case of using finite state machines (FSM) for specification, based upon an automaton graph, a student can use the JGIFT design environment [20] of the REAL system.

Figure 8 gives an impression of the verification and simulation features of the Web-based environment of the REAL system. The simulation model will be driven directly through the I/O signals by the control algorithm running on the embedded interpreter within the applet.

Assuming the student achieved a validated design, he gets the required next state and the output equations. By accessing the Web-browser interface of the REAL system, he is able to enter his algorithm (the received equations), handle the laboratory experiment (e.g., start, stop, reset) and change environmental variables if necessary. The control al-

gorithm is executed by an interpreter running inside the student's client PC (e.g., implemented as a HTML 5 RIA). No Internet connectivity to the laboratory and the physical systems in lab is necessary for this mode (see Figure 9).

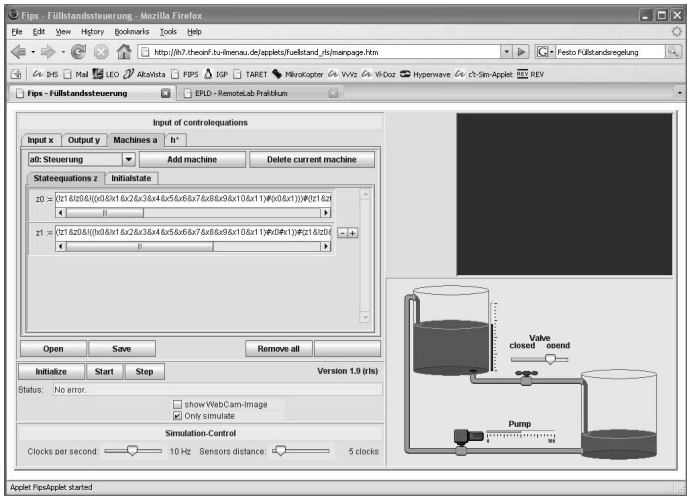

Figure 8. Offline regulation of the water level control (without Internet)

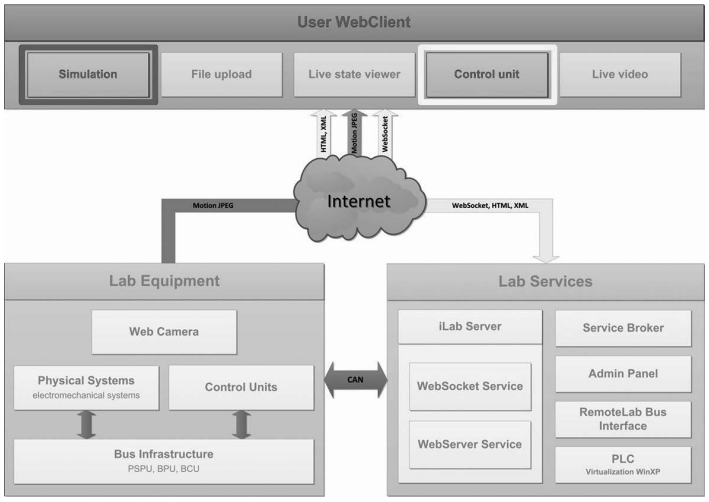

Figure 9. Stand-alone mode for visual prototyping

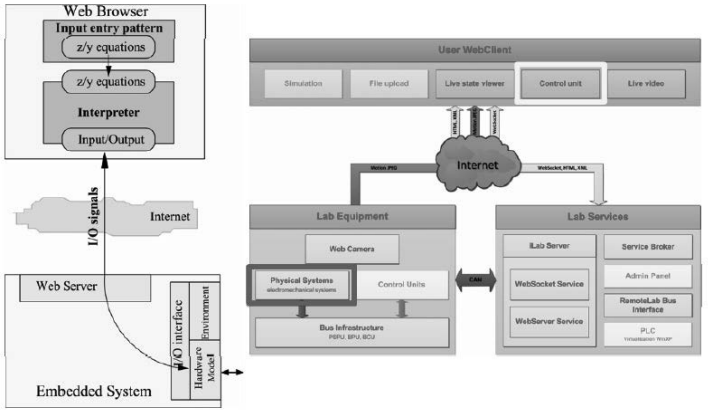

Figure 10. Remote control mode via a Web-client

Remote Control Mode – via Web-client

This operation mode is also for the FSM based specification based on equations. In this case, the physical system will be controlled via the Internet "from a distance" through the interpreter running inside the student's client PC. No additional control units in the remote lab are necessary (see Figure 10). In this case, only the input and output signals of the physical system will be transferred via Internet.

An example of such a Web-client is shown in Figure 11.

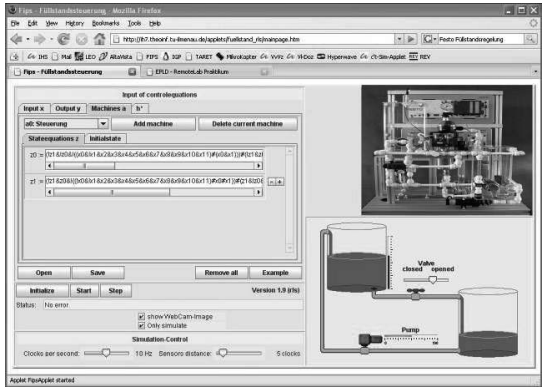

Figure 11. Online regulation of the water level control (with Internet)

Remote Control Mode – via Control Unit

This mode can be used to realize software or hardware oriented control tasks via connected microcontrollers or FPGAs as control units (see Figure 13).

Students can implement their control algorithm directly into a microcontroller for a software-oriented implementation. Therefore, they use common (non-commercial) development tools, for example MPLAB IDE and/or C18 C-compiler by Microchip [21], or AVR Studio by Atmel [22], to develop Assemblerand/or C-coded software projects. After compilation, the generated software control algorithm is transferred via REAL Web-interface to the remote lab, where the hex code is programmed into the microcontroller (see Figure 12).

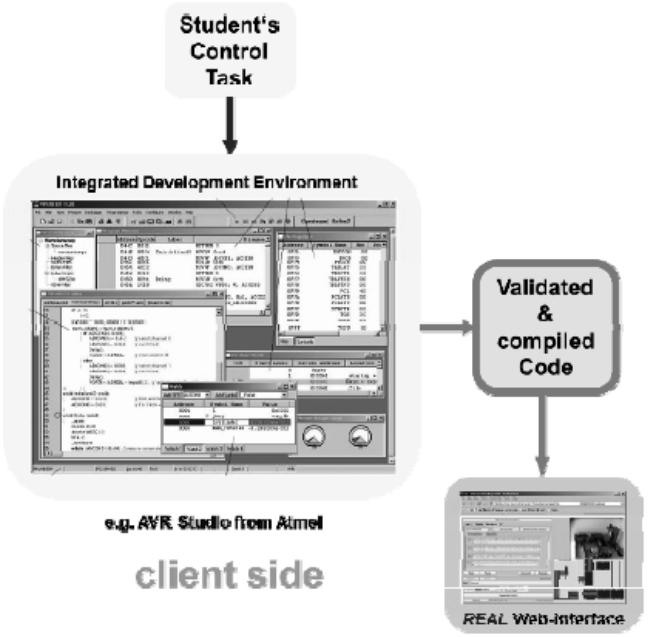

Figure 12. Software-oriented design of the control task

Now, the student can begin with his experiment, to check if his algorithm fulfills the requirements of the given control task.

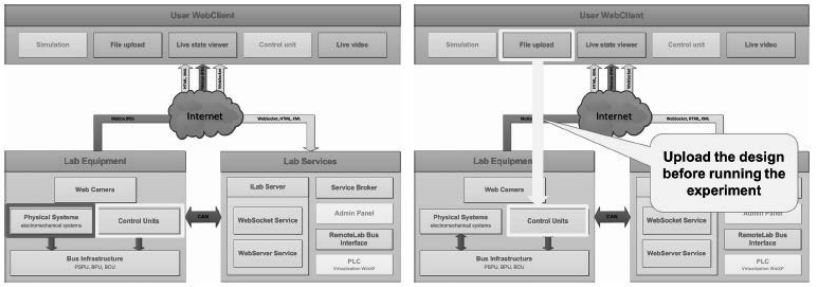

Figure 13. Remote control mode via a control unit

If a student prefers an exclusive hardware-oriented design using an FPGA and applying a hardware description language like VHDL as specification technique, he can prepare his design with common development tools, for example ISE, Quartus II, Diamond or others. The generated bit file is uploaded via the REAL Web-interface to the remote lab, where the FPGA will be programmed (see Figure 14). After programming the connected FPGA, the FPGA board operates as control unit for the designed control algorithm, and the student can start his experiment.

The student has the possibility to debug his design, by defining breakpoints depending on input/output signals from the physical system protection unit and stop the electromechanical model at certain sensor readings to validate the correct actor settings. It is also possible to do single step processing, by pausing the execution on every sensor/actor change.

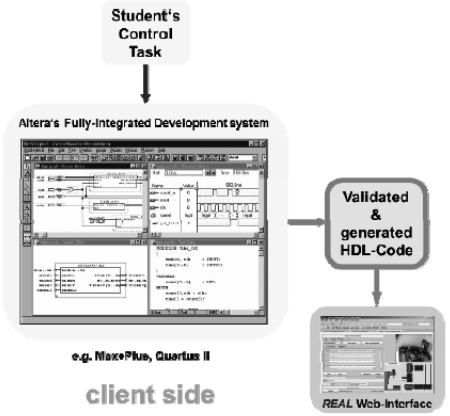

Figure 14. Hardware-oriented design of the control task

Virtual Control Mode – Visual Prototyping

This mode is comparable to the operation mode A (for visual prototyping). In this case, the simulation model is not connected to the interpreter, running inside the client, but via the Internet to the real control units which are running inside the remote lab. In this case, the student can test his prepared software or hardware oriented design on the corresponding control unit (microcontroller or FPGA) without the need for a real physical system – before he will use operation mode C (remote control mode – via control unit) to test his design task on the physical system in the remote lab (see Figure 15).

Furthermore it is possible to have many students working on control algorithms for the same electromechanical model without disturbing each other. They can test and validate their design before connecting it to real hardware. This can eliminate the need to install multiple instances of the same electromechanical model in one lab and therefore reduce the costs of this lab.

Besides this, it is also possible to emulate electromechanical models that are not available in the lab.

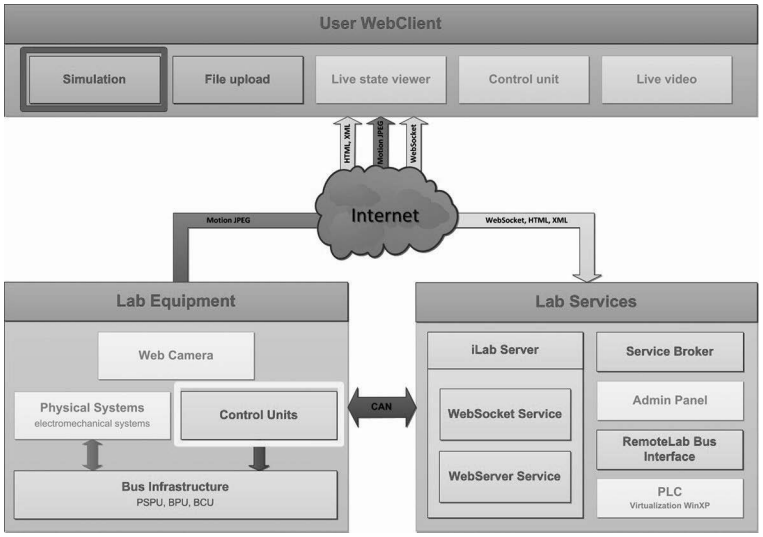

Figure 15. Virtual control mode for visual prototyping

Virtual Control Mode – Test Mode

This operation mode (see Figure 16) is mainly for debugging, testing and maintenance of the physical system protection unit.

By using this operation mode, it is possible to check the implemented reference design (as seen in Figure 4) in the physical system protection unit without the need to use a control unit or a physical system. This will be done by transmitting input and output pattern from the Web-client to the protection unit and by analysing its response.

Because this mode is not preferential to execute lab experiments, it is not explained in detail.

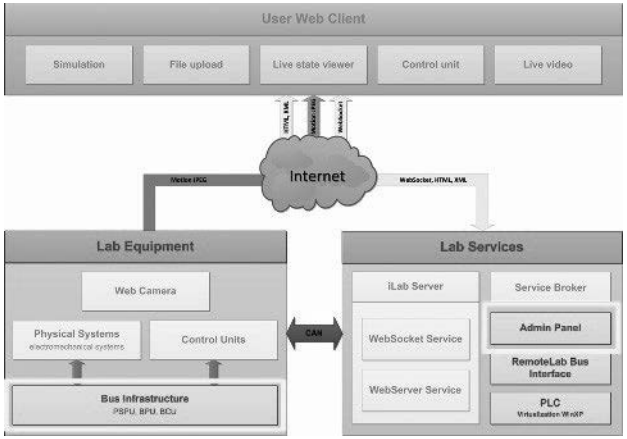

Figure 16. Virtual control mode to test the protection unit

Local Control Mode – via a Control Unit

Besides the possibility to work in the lab remotely, the following two operation modes explain the usage of the REAL system for on-site experiments or demonstrations.

During an on-site lab experiment students can observe the whole hardware setup as well as all the physical system and environmental variables directly in the lab room. In this case they can check their control task (running on a connected control unit). They have access to the whole lab setup via a connected control panel. Figure 17 illustrates this operation mode.

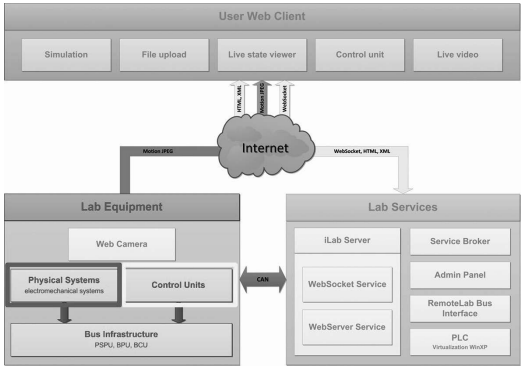

Figure 17. Local control mode via a control unit

In addition to its use for student experiments, this mode can also be used to demonstrate the remote lab on guided tours during open house presentations to inspire new students to study engineering courses.

Local Control Mode – manually

This operation mode (see Figure 18) can be used for demonstrations and maintenance of the connected electromechanical models.

By activating actuators (e.g., a water pump) manually without any control algorithm the sensor signals can be observed via the connected control panel.

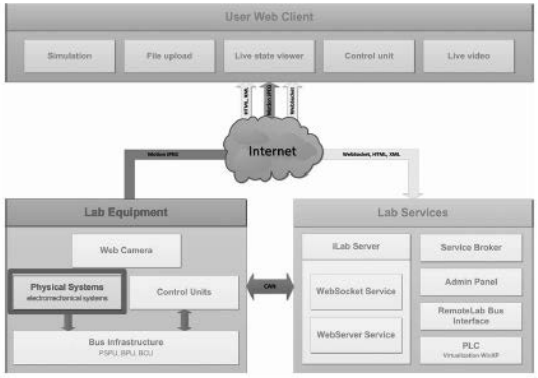

Figure 18. Manual local control

This mode will also be used for hands-on usage of the demonstration system “labyrinth on ball balance plate” (see Figure 19) or can be used for testing and debugging the electromechanical parts of the physical systems for service and inspection.

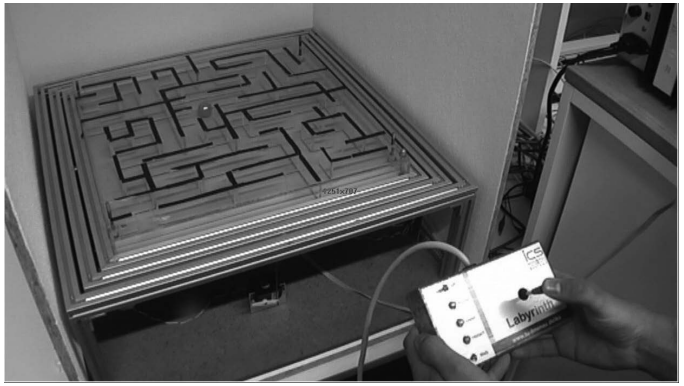

Figure 19. Labyrinth model

We would like to provide this game-like experiment especially for promotional means for open house presentations at the university to attract new students in engineering disciplines.

Rapid Prototyping Mode

For this special operation mode a rapid-prototyping board for digital systems was developed, which is directly connected to the serial remote lab bus. In this case, no physical system protection unit is needed.

By using the REAL Web-interface, the student is able to

- upload his/her design,
- program the FPGA and
- handle the lab procedure.

The applet allows the student to manipulate all the inputs of the rapid prototyping board (slide switches, hex coding switches, and pushbuttons). He can observe the outputs of the board (7-segment displays, row of LEDs) virtually inside the Java applet. Figure 21 gives an impression of the applet’s Web-interface including a “photo” (image) of the rapid prototyping board.

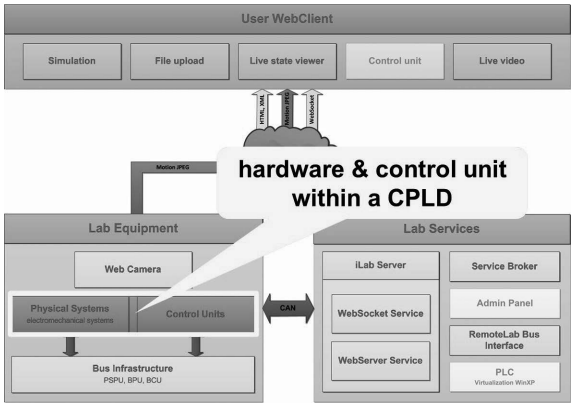

Figure 20: Rapid prototyping mode

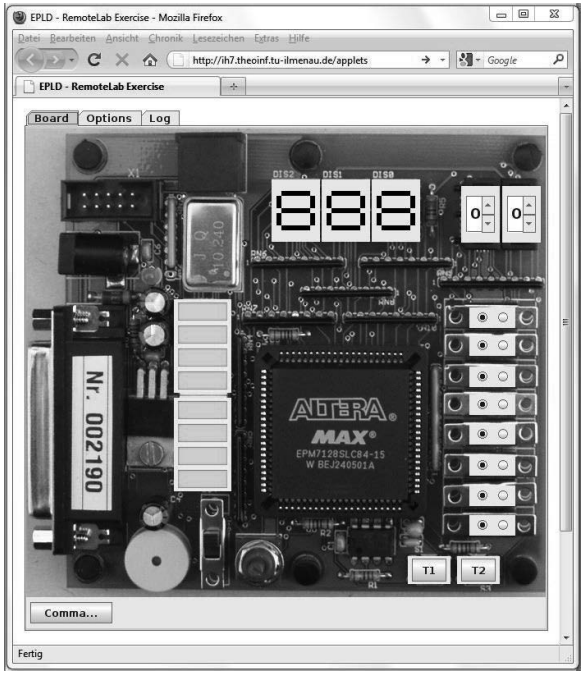

Figure 21. Web-interface of the Rapid Prototyping board

The manipulation of the input and output signals of the board are done virtually in the following way: For the look-and-feel of the applet, we use a "photo" of the board as background. All the inputs are realized as Java control elements and can be manipulated via the mouse interactively.

Changes are immediately sent to the rapid prototyping board and the corresponding results are displayed inside the applet. There are two general options to display the output results within the Web-interface:

- Representation by Java components

The graphical outputs are represented directly inside the "photo" using read-only Java components. Updates of the display only require information about the current state of the corresponding output signal. This enables students without fast Internet access to use the applet.

- Webcam based feedback

Another option is the use of a webcam to monitor the rapid prototyping board inside the lab room. The webcam image is replacing the background "photo" of the applet. This allows the user to watch the results of his/her actions directly as if present in the lab. The overlaying Java display components are not used in this option and therefore invisible.

Both options can be configured using custom user settings in the corresponding tab of the applet. All actions are documented in a separate Log Tab.

By using this Web-based rapid prototyping board the following application fields are possible:

Web-based Rapid Prototyping of Digital System

By using common design tools (e.g. MaxPlus+, Quartus II from Altera [23]), the student is able to specify his design via

- Text based design methods,

Here the student can enter his design by means of logical equations, truth tables or hardware description languages (like VHDL or Verilog).

- Graphically based design methods,

The student can use block or schematic diagrams to input his design.

- Integrated FSM editors.

In case of sequential design tasks he can directly enter the derived automaton graphs with the built in FSM editor.

The editor itself generates VHDL code for the further design steps.

Finally, students will specify, describe, implement and verify different digital systems using “selfmade” IP core libraries, e.g.

- Digital control systems
- Serial communication modules,
- Robot sensors,
- Model of a RISC processor.

Once the design is completed and error free, the student can test his solution using the described platform.

Web-based Verification of Digital Systems

Another use case of the rapid prototyping board is to identify the function of a given design (black box) or to find malfunctions of a well-known design. The CPLD will be programmed by the teacher, but the student has no ability to reprogram the device.

By manipulating the inputs of the unknown black box in order to find the malfunction (e.g. to enter all the input sets of a truth table or special input sequences), the student attempts to analyze the response (the real output signals of the board) to find out the function of the given design. The student has to enter his test vectors based on truth tables or input sequences one by one using the provided controls and observe the results.

It is planned to support the upload of a whole truth table or input sequence as a text file to the Webinterface of the prototyping board. In this case the student will be able to record the responses corresponding to the specified inputs creating a waveform file. This file can then be used for further investigation of the current design.

Visitor Mode

This mode of operation is intended for visiting the lab and passively taking part in experiments. It supplies a live video feed as well as a display of sensor and actor values from the electromechanical system of the selected experiment (see Figure 22).

Using the visitor mode does not require any login or booked time slot, but gives any interested person the possibility to view the lab and any

running experiment. This mode is also useful for tele-teaching, where a teacher wants to present an experiment to a student via Internet.

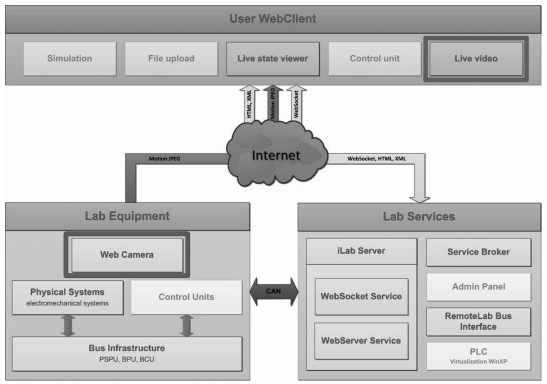

Figure 22. Visitor mode

CONCLUSION AND FORESIGHT

We have discussed different operation modes of the REAL system for various fields of applications – based on a new flexible grid-based online lab structure. In addition to simulation-based experiments, on-side and off-side experiments with real physical systems can be offered for students as well.

Besides the features already mentioned in this article, even more functionality can be added using the new concept of having a Web-based protection unit that checks the user input against a reference model. This protection unit can be connected to a learning management system like “moodle” to forward any experimental results of the user. For an effective usage of the REAL system within learning management systems, the reference design and a method to check the student’s design against this reference design step by step will be traced by the LMS. Figure 23 shows this idea.

The increasing capacity of wireless communication and the growing number of mobile devices (e.g. smartphones and tablets) on the one hand as well as modern Internet technologies like JavaScript, HTML5 and Web Sockets on the other hand provide new possibilities and challenges in the area of mobile learning (see Figure 24). In [24] a first Android client application for the iLab Shared Architecture is described.

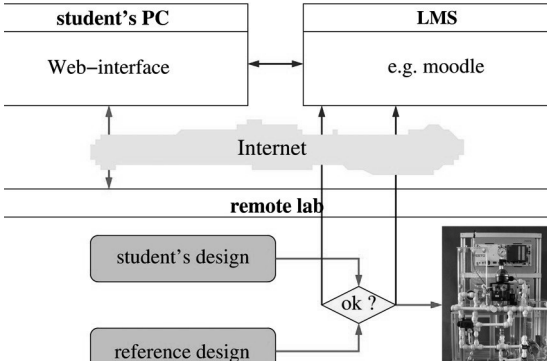

Figure 23. Observation of the student's design under LMS control

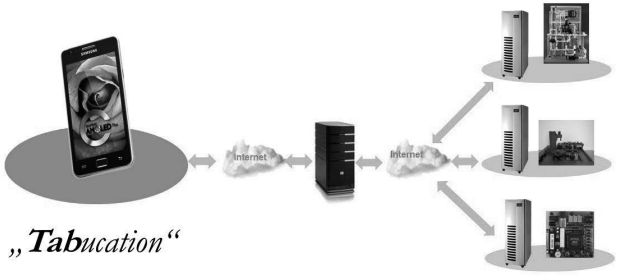

Figure 24. Mobile learning by using modern wireless technologies

Our Integrated Communication Systems Group at the Ilmenau University of Technology is involved in different national and international e-Learning projects (e.g., [25]) in which it is increasingly necessary to allow and organize a shared use of equipment. That is why, the main focus of the REAL system is

- a Web-wide usage of different design tools and control units to control different physical systems in the lab room,
- a robust, fault-protected access to any connected physical system,
- an LMS-coupling for all control units and physical systems used in the remote lab as well as
- a worldwide interchange of online experiments with other universities by interconnecting the iLab Service Broker to an "iLab cloud" (see Figure 25), as proposed by the iLab Europe consortium, where we are involved, as well [26], [27].

All these requirements can be fulfilled using the concept and the infrastructure presented in this paper.

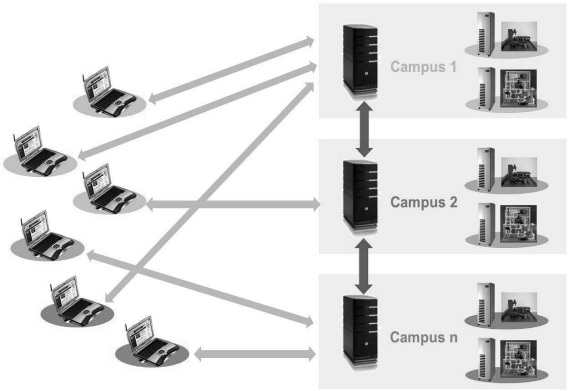

Figure 25. Interconnection of various iLab Broker to an "iLab cloud"

Future work will be concentrated on analyzing the student's behavior during the experiments and developing an adaptive feedback and assessment system to support the students in a better way.

ACKNOWLEDGMENT

The authors would like to thank Silvia Krug, Stephan Simon and Stefan Vogel for their work within the REAL framework.

Parts of the REAL project are supported by the society "Friends of the Faculty of Computer Science and Automation". This work is also supported by the project "ICo-op – Industrial Cooperation and Creative Engineering Education based on Remote Engineering and Virtual Instrumentation" by the European Commission within the program "Tempus", Grant No 530278TEMPUS-1-2012-1-DE-TEMPUS-JPHES [26].

REFERENCES

[1] S. Sire, F. Geoffroy and D. Gillet: "A Virtual Assistant for Sending Hints and Perturbations to Students based on an Electronic Laboratory Journal (eJournal)", Proceedings of the ITHET'03, Marrakech, Morocco, July 7-9, 2003.

[2] K. Henke and H.-D. Wuttke: “Web-based educational tool access”, IASTED International Conference Computers and Advanced Technology in Education – CATE 2003, Rhodes, Greece, June 30 July 2, 2003.

[3] H.-D. Wuttke and K. Henke: “Living Pictures – tool-oriented learning modules and laboratory for teaching digital via Internet”, Proceedings of the ICEE-2002 International Conference on Engineering Education UMIST, Manchester, Great Britain, August 18-22, 2002.

[4] The iLab Project, from: https://wikis.mit.edu/confluence/ display/ ILAB2/Home

[5] WebLab Deusto, from: https://www.weblab.deusto.es/web. [6] VISIR, from: http://openlabs.bth.se/index.php.

[7] LabShare Sahara, from: http://www.labshare.edu.au/project/ index.php.

[8] REAL: “Remote Engineering and Applications Laboratory”, http://lanai.theoinf.tu-ilmenau.de/applets/index.htm.

[9] Y. Torroja, et al.: “A Modular Environment for Learning Digital Control Applications”, Microelectronics Education, Marcombo, S.A, 2002.

[10] T. A. Fjeldly, J. O. Strandman, R. Berntzen and M. S. Shur: “Advanced Solutions for Laboratory Experiments over the Internet”, Engineering Education and Research-2001, Begell House Publishing.

[11] H.-D. Wuttke, K. Henke and N. Ludwig: “remote labs versus Virtual Labs for Teaching Digital System Design”, Proceedings of the Int. Conf. On Computer Systems and Technologies CompSysTech’05, Varna, 2005.

[12] K. Henke, H.-D. Wuttke and T. Braune: “Virtual and remote labs in the Educational Process”, International Conference on Remote Engineering and Virtual Instrumentation, REV2007, Porto, Portugal, June 25-27, 2007.

[13] K. Henke, H.-D. Wuttke and S. Hellbach: “Laboratory via Internet – new ways in education and research”, Int. Journal of Computers and Applications, vol. 25, ACTA press, 2002.

[14] K. Henke, H.-D. Wuttke and T. Braune: “Rapid Prototyping Modules for Remote Engineering Applications”, International Conference on Remote Engineering and Virtual Instrumentation, REV2008, Düsseldorf, Germany, June 23-25, 2008.

[15] K. Henke, St. Ostendorff and H.-D. Wuttke: “A Flexible and Scalable Infrastructure for Remote Laboratories Robustness in Remote Engineering Laboratories”, The Impact of Virtual, Remote and Real Logistics Labs ImViReLL2012, Bremen, February 28 – March 02, 2012.

[16] K. Henke, St. Ostendorff, H.-D. Wuttke and St. Vogel: “A Grid Concept for Reliable, Flexible and Robust Remote Engineering Laboratories”, International Conference on Remote Engineering and Virtual Instrumentation, REV2012, Bilbao, Spain, July 4-6, 2012.

[17] MIT iLab Service Broker, Project homepage, from: http://ilab.mit.edu/iLabServiceBroker.

[18] H.-D. Wuttke, R. Ubar, K. Henke and A. Jutman: “Assessment of Student’s Design Results in E-Learning-Scenarios”, 8th Conference on Information Technology Based Higher Education and Training (ITHET2007), Kumamoto City, Japan, July 10-13, 2007.

[19] K. Henke: “Reusable Assessment Objects for Learning Management Systems”, Computers and Advanced Technology in Education”(CATE 2007), Beijing, China, October 8–10, 2007.

[20] JGIFT: “Java-based Graphical Interactive FSM Tools”, http://wcms1.rz.tu-ilmenau.de/fakia/index.php?id=780, TU Ilmenau.

[21] Microchip Corporation, http://www.microchip.com. [22] Atmel Corporation, http://www.atmel.com.

[23] Altera Corporation, http://www.altera.com.

[24] B. Deaky, D.G. Zutin and P.H. Bailey: “The First Android Client Application for the iLab Shared Architecture”, International Journal of Online Engineering (iJOE), Vol 8, No 1 (2012).

[25] TRE – International Summer School in Technologies for Remote Engineering, www.ingenieria.deusto.es/summerschool2012.

[26] ICo-op project Website: http://www.ICo-op.eu.

[27] The iLab Europe Network Service Broker, from: http://ilabeurope.net/iLabServiceBroker/.

AUTHORS

Karsten Henke is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: karsten.henke@tu-ilmenau.de).

Steffen Ostendorff is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: steffen.ostendorff@tu-ilmenau.de).

Heinz-Dietrich Wuttke is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: dieter.wuttke@tu-ilmenau.de).

Tobias Vietzke is with the Ilmenau University of Technology, Master Student in Computer Engineering at the Faculty of Computer Science and Automation, 98684 Ilmenau, Germany, POB 10 05 65. He obtained his BSc. in Computer Engineering in 2012 (e-mail: tobias.vietzke@tu-ilmenau.de).

Christian Lutze is with the Ilmenau University of Technology, Master Student in Computer Engineering at the Faculty of Computer Science and Automation, 98684 Ilmenau, Germany, POB 10 05 65. He obtained his BSc. in Computer Engineering in 2012 (e-mail: christian.lutze@tu-ilmenau.de).

This work was supported in part by the European Commission within the program “Tempus”, Grant No 530278-TEMPUS-1-2012-1-DETEMPUS-JPHES and by the “Friends of the Faculty of Computer Science and Automation” of the TU Ilmenau. It is an extended and modified version of a paper presented at the International Conference on Remote Engineering & Virtual Instrumentation (REV2012), held at University of Deusto, Bilbao, Spain, July 4-6, 2012. Received 01 March 2013. Published as resubmitted by the authors 20 March 2013.

Наукове видання

ПАРХОМЕНКО Анжеліка Володимирівна,
ГЛАДКОВА Ольга Миколаївна,
ПОЛЯКОВ Михайло Олексійович,
ЛАРІОНОВА Тетяна Юріївна,
ТАБУНЩИК Галина Володимирівна,
КАПЛІЄНКО Тетяна Ігорівна

ВІДДАЛЕНИЙ ТА ВІРТУАЛЬНИЙ ІНСТРУМЕНТАРІЙ В ІНЖИНІРИНГУ

Монографія

Комп’ютерний набір	*Г. В. Табунщик*
Дизайн обкладинки	*Д. М. Головань*
Технічний редактор	*Л. А. Рябоконь*
Коректор	*Н. В. Чечеко*
Художник	*А. П. Кондаков*

Формат 60x84/16.
Папір офсетний. Гарнітура *Times*. Друк офсетний.
Підписано до друку 30.09.2015. Ум. друк. арк. 14,53. Наклад 300 прим.

Видавництво «Дике Поле»
Україна, 69063, м. Запоріжжя, вул. Чекістів, 31-А.
Тел.: (061) 213–75–95; 213–75–05.
Свідоцтво суб’єкта видавничої справи ЗЗ № 004 від 23.08.2001 р.

В-42 **Віддалений** та віртуальний інструментарій в інжинірингу: монографія /за заг. ред. Карстена Хенке. – Запоріжжя: Дике Поле, 2015. – 250 с.
ISBN 978–966–2752–74–8

Книга містить основні концепції використання віртуальних, керованих дистанційно пристроїв, а також розподілених віддалених лабораторій. Розглянуті можливості їх використання при проектуванні вбудованих систем, для розробки систем керування складними електротехнічними установками і комплексами та для оцінювання якості вбудованих систем. Видання може бути корисним для фахівців в галузі проектування вбудованих систем, спеціалістів з електромеханіки, студентів та аспірантів.

УДК 004.41

Видання здійснено за підтримки міжнародного проекту ICo-op «Промислове співробітництво та креативна інженерна освіта на основі дистанційного інженерного та віртуального інструментарію» (530278-TEMPUS-1-2012-1-DE-TEMPUS-JPHES) за програмою TEMPUS Європейської комісії.
Зміст даного матеріалу відображає думку авторів та Європейська комісія не несе відповідальності за використання інформації, що міститься в монографії.