УДК 004.383

Сергієнко А.М.

ЗЛОНАМІРЕНО СТВОРЕНЕ АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

Анотація:

Дається огляд видів злонамірено створеного апаратного забезпечення (ЗСАЗ), їх прихованих каналів передачі інформації, можливостей реалізації ЗСАЗ в інтегральних схемах, включаючи програмовані логічні інтегральні схеми (ПЛІС), способів їх виявлення та перешкоджання їх впровадженню. Робиться висновок про те, що ПЛІС найбільш захищені від впровадження в них ЗСАЗ.

Аннотация:

Дается обзор видов злоумышленно созданного аппаратного обеспечения (ЗСАО), их скрытых каналов передачи информации, возможностей реализации ЗСАО в интегральных схемах, включая программированные логические интегральные схемы (ПЛИС), способов их выявления и препятствия их внедрению. Делается вывод о том, что ПЛИС наиболее защищены от внедрения в них ЗСАО.

Abstract:

A survey of malicious hardware, its hidden channels, possibilities of its loading in the integral circuits including FPGA is considered. The methods for malicious hardware searching and preventing to its loading onto the device are highlighted as well. A conclusion is made that FPGA is the most safe device against malicious hardware loading.

Вступ

Злонамірено створеним програмним забезпеченням (malicious software) вважається програми, розроблені з метою порушувати режим захищеного та надійного функціонування обчислювальних систем (ОС). Створювати таке програмне забезпечення доволі просто та заманливо. Це обумовлюється поширенням однакових архітектур комп′ютерів, великою кількістю можливостей ураження обчислювальних систем з боку матзабезпечення, а також у певних випадках – значною вигодою від такої діяльності. Тому досі найбільшу увагу наукових досліджень та організаційних і технічних заходів щодо захисту ОС присвячено саме такому програмному забезпеченню.

Злонамірено створене апаратне забезпечення (ЗСАЗ, malicious hardware) можна визначити аналогічно. Це апаратне забезпечення ОС, розроблене та впроваджене задля порушення режиму захищеного та надійного функціонування ОС.

ЗСАЗ починає свою історію з часів холодної війни, коли серйозні доповнення до ОС, виготовлених за тогочасною провідною технологією мікросхемотехніки, оплачувались значним військовим бюджетом. Так, під керівництвом ЦРУ були розроблені мікросхеми з вбудованим ЗСАЗ у вигляді знищувального перемикача (kill switch). Ці мікросхеми певним шляхом попали в СРСР, де вони були впроваджені у систему контролю магістральним газопроводом і в результаті спрацювання ЗСАЗ привели до його аварії. З іншого боку служби КДБ забезпечили друкарські машинки для посольства США у Москві електронікою, яка була спроможна копіювати документи, що друкувались на них.

Зі збільшенням кількості транзисторів на кристалі надвеликих інтегральних схем (НВІС) можливості створення та схову ЗСАЗ зростають. Є думка, що успішні дії ізраїльської авіації у Сирії були забезпечені спрацюванням ЗСАЗ у мікропроцесорах сирійських радарів. Зовнішні НЖМД, які було продано фірмою Seagate у Тайвані, були

оснащені ЗСАЗ, що призначались для пересилки на відстань персональних даних користувача [1,2].

Останнім часом все більше уваги приділяється ЗСАЗ [3,4]. Це пов′язано насамперед з поширенням програмованих логічних інтегральних схем (ПЛІС). Через високу продуктивність, зменшене енергоспоживання, економічну ефективність, можливість забезпечити надійне та захищене функціонування системи, широку доступність засобів проектування ПЛІС все частіше застосовуються як елементна база сучасних ОС. Відомо, що розробка та впровадження проектів для ПЛІС проходить такі самі стадії, які має створення матзабезпечення: складання, компіляція та відлагодження відповідних програм, написаних високорівневою мовою програмування, завантаження скомпільованих кодів у апаратуру. Тому ЗСАЗ та шкідливе програмне забезпечення мають багато спільного.

ЗСАЗ і держава

ЗСАЗ, як, наприклад, знищувальний перемикач, являє собою приховану загрозу телекомунікаційній інфраструктурі, транспорту, засобам зброї, аерокосмічним об′єктам, медичним, атомним та енергогенеруючим установкам держави, якщо шкідливі мікросхеми певним чином попадуть у схеми цих об′єктів. Так, у США, де впровадження мікроелектроніки досягло найбільшого поширення, дуже стурбовані цією загрозою. Це значною мірою пов′язано з тим, що в останні десятиріччя сучасна мікроелектронна технологія успішно розвивається у східних країнах, які є економічними та геополітичними конкурентами США [5]. Президент США Б. Обама сказав стосовно захисту американської кібернетичної інфраструктури: “Та сама технологія, що озброює нас для створення та будівництва, також озброює тих, хто може підривати та руйнувати, і це є великою іронією інформаційної ери, причому цей парадокс є те, що явно чи неявно ми пізнаємо у нашому щоденному досвіді. ...Коротше, американське економічне процвітання у 21-му столітті залежатиме від кібербезпеки” [6].

Тому невипадково, що агенція DARPA міністерства оборони США створила програму, яка називається: “TRUST in Integrated Circuits – TIC”, що перекладається як довіра до інтегральних схем. Завданням програми TIC є розробка методів виявлення вбудованих ЗСАЗ у НВІС з використанням як втручання у мікросхему (шліфування та сканування), так і неруйнуючих методів. Ця програма опікувалася також проблемами захисту ПЛІС та впровадження потенційно небезпечних віртуальних модулів.

Програма TIC виконувалась кількома групами дослідників, які створили кооперацію промисловості та університетів. Це група розробки та виготовлення тестових мікросхем, група дослідження методів встроювання та пошуку ЗСАЗ та метрична група, яка приймає рішення про успішність експериментів. Ця трьохрічна програма переросла у програму IRIS – Integrity and Reliability in Integrated Circuits initiative [3,7].

Види ЗСАЗ

ЗСАЗ має багато спільного зі шкідливим програмним забезпеченням. Тому класифікація ЗСАЗ походить від класифікації останнього. ЗСАЗ типу *чорний хід* (*backdoor, trapdoor*) дозволяє доступ до ОС, який не обумовлюється його технічними умовами. Воно може бути впроваджене під час розробки ОС або під час її модернізації.

ЗСАЗ типу *знищувальний перемикач* (*kill switch*) – це шкідливий штучний об′єкт, який дає змогу атакуючим порушити коректне функціонування апаратури або матзабезпечення. Так само, як і чорний хід, знищувальний перемикач може бути впроваджений під час розробки ОС або під час її модернізації. Але замість можливості

несанкціонованого доступу, він викликає відмову у вигляді неспроможності виконувати необхідні функції. Він може бути встановлений як частина прошивки ПЛІС або у проект мікросхеми за допомогою утиліт САПР мікросхем, або під час виготовлення мікросхеми.

Вірус ПЛІС – це частина прошивки ПЛІС, яка може викликати закорочування виходів внутрішніх вентилів і як результат – перегрів та вихід з ладу мікросхеми [8]. Віруси ПЛІС можуть бути впроваджені за допомогою матзабезпечення або навіть шляхом корекції топології мікросхеми ПЛІС (що маловірогідне, але можливе). Хоча віруси ПЛІС – не є поширеним явищем, їх можливе застосування слід завжди мати на увазі.

Троянська апаратура – це широкий клас різноманітних ЗСАЗ. Як і троянські програми, ці ЗСАЗ впроваджуються сторонніми особами і не призначені для виводу з ладу ОС. Результатом дії може бути порушення нормального функціонування пристрою, як, наприклад, уповільнення дії, зменшення точності або передавання чи приймання важливої інформації через прихований канал. Вони розрізняються за фізичним принципом реалізації, способом активізації та цільовою функцією [9].

Впровадити ЗСАЗ у мікросхему можна під час її виготовлення або шляхом фізичного втручання в неї. В останньому випадку є можливість дістатись до поверхні кристала і засобами обробки сфокусованим іонним променем змінити геометрію шарів кристала, тобто видалити певні провідники, додати інших провідників чи скоригувати ємність, опір, затримку, максимально допустимий струм.

Функціональне ЗСАЗ виконується шляхом додавання, вилучення або закорочування транзисторів чи вентилів. *Параметричне ЗСАЗ* реалізується через модифікацію існуючих ліній зв′язку та інших дискретних компонентів кристала. Наприклад, утончення провідника або порушення транзистора чи резистора не спричиняють відмову, але суттєво збільшують вірогідність відмови з часом внаслідок електроміграції або призводять до суттєвого зменшення продуктивності чи точності.

За розміром та кількістю задіяних входів-виходів ЗСАЗ розділяють на *малі* та *великі*. Великий блок ЗСАЗ значно легше активізувати та виявити, ніж малий. Зате він має функціональність, яка забезпечує вирішення складних завдань, що спричиняють великі втрати.

Розрізняють також *сконцентровані* та *розподілені* ЗСАЗ. Останні однорідно розподілені по площі кристала і тому є значно складнішими для виявлення через аналіз топології. Їх легше виготовити, якщо кількість доступних для модифікації вентилів кристала є обмеженою. З іншого боку, у сконцентрованого ЗСАЗ менша площа, коротші провідники і тому його важче виявити за непрямими ознаками – зміні енергоспоживання, максимальної тактової частоти у порівнянні з еталонним кристалом.

ЗСАЗ може бути *активованим ззовні* через прихований канал або *зсередини*. Останні ЗСАЗ розділяються на *завжди ввімкнуті* та такі, що *вмикаються за умовою*. Завжди ввімкнуте ЗСАЗ є постійно активованим і може порушити функціонування ОС у будь-який момент. Це, як правило, параметричне ЗСАЗ.

ЗСАЗ, що вмикається за умовою, є неактивним допоки не справджується певна умова. Це, наприклад, детектування конкретних параметрів, таких як температура, тиск, напруга, рівень електромагнітного поля тощо, або їх комбінації чи послідовності. Це може бути фіксація заданого внутрішнього логічного стану чи стану вхідних сигналів, або переповнення внутрішнього лічильника. Таке ЗСАЗ майже “невидиме”, тобто не може бути виявленим до своєї активізації.

За своєю цільовою функцією ЗСАЗ поділяються на такі, що змінюють функціональність мікросхеми, модифікують її параметри чи призначені для передачі інформації. Впровадження ЗСАЗ *змінює функціональність* через додавання нових логічних схем, або через вилучення чи закорочення вже існуючих. *Модифікація параметрів* – затримок, порогу спрацьовування, рівня підсилення, струму живлення тощо – досягається зміною геометрії провідників та резисторів.

ЗСАЗ *для передачі інформації* призначені для пересилки секретної інформації з захищеної області назовні. Можливе також поєднання передачі інформації зі зміною функціональності чи параметрів. Наприклад, певна зміна параметрів схеми блоку шифрування ОС дає змогу визначити ключ шифрування через аналіз вимірів енергоспоживання цієї ОС.

Можливості реалізації ЗСАЗ у ПЛІС

Теоретично ЗСАЗ може бути закладена у ПЛІС під час виготовлення кристала. Зловмисник може знаходитись у компанії, що виготовляє маски чи кристали, або вмонтовує їх у корпуси. Він має можливість копіювати, аналізувати та змінювати проект мікросхеми. Але через те, що на етапі виробництва немає думки про те, де і яким чином будуть використані ПЛІС, яка саме прошивка буде реалізована, впровадження ЗСАЗ, що має виконати певні функції, є занадто складним завданням. Потенційними місцями, до яких може бути під′єднано ЗСАЗ шляхом певної корекції масок, є апаратне мікропроцесорне ядро, блоки оперативної пам′яті, вузол вводу-виводу, блок вводу конфігураційної послідовності та блок криптообробки цієї послідовності [10].

Зловмисник потенційно спроможний вставити ЗСАЗ у матрицю ПЛІС на етапі її виготовлення, маючи надію, що ЗСАЗ спрацює як знищувальний перемикач для проекту, що в неї буде сконфігуровано. Розраховуючи на цей випадок, розробник проекту може вжити заходів, аби запобігти його ненадійній роботі. Наприклад, відповідальні блоки проекту можна виконати з потрійним резервуванням та розосередити їх по кристалу. Крім того, система завантаження та збереження прошивки має доволі надійний захист, щоб бути порушеною втручанням з боку ЗСАЗ. Щоб виконати таку атаку проти конкретного користувача ПЛІС, виробник повинен випустити значно більшу партію кристалів, ніж потрібно. При цьому ЗСАЗ спрацює у багатьох користувачів і через процедуру рекламації, урешті-решт, буде виявлено наявність ЗСАЗ. Тому такий підхід є ненадійним.

Хоча кристал ПЛІС та ОС, де він впроваджений, є потенційно незахищеними від втручання, прошивка ПЛІС може бути розроблена в умовах секретності. Тому проект для ПЛІС, якщо з ним не буде ознайомлений зловмисник, є захищеним. Так як функціонування ПЛІС ґрунтується на інформації у файлі прошивки, до неї застосовуються стандартні методи інформаційного захисту. Атаки на апаратуру з реконструюванням її проекту, які використовуються для звичайних мікросхем, для ПЛІС є неприйнятними, тому що вони призводять до руйнації прошивки.

Щоб впровадити зловмиснику своє ЗСАЗ у ПЛІС, необхідно мати початковий проект. Інформація про проект закодована у файлі прошивки. Нехай зловмисник заволодів файлом прошивки. Незважаючи на те, що правила кодування прошивки відомі виробнику ПЛІС і закладені у відкритих засобах проектування ПЛІС, досі не існує доповідей про успішне реконструювання жодного проекту на основі такого файла прошивки. Незважаючи на це, у відповідальних застосуваннях приймаються додаткові заходи захисту.

Найпростіший захід – це однократне програмування, коли повністю відсутній доступ до прошивки за умови постійної підтримки живлення ПЛІС. Протягом останнього десятиріччя у ПЛІС застосовується шифрування файла прошивки, що дозволяє багатократне її завантаження у ПЛІС після вимикання та вмикання живлення без змоги несанкціонованого копіювання. Це додатково унеможливлює реконструювання проекту. Крім того, в режимі шифрованого файла прошивки в ПЛІС заборонена часткова конфігурація, яка дозволяє приєднувати додаткові частини проекту. Також умовою завантаження такого файла є обов′язкове вимкнення живлення для знищення попередньої конфігурації.

Ключ шифру файла прошивки зберігається у ПЛІС в оперативній пам′яті з батарейним живленням і автоматично стирається при його вимкненні. Відомі способи атаки на цей ключ призводять до вимкнення цього живлення та руйнації шифру й конфігурації, оскільки ключ зберігається у глибині кристала під кількома шарами діелектрика та металізації.

Таким чином, чи не єдиною можливістю вставити ЗСАЗ у ПЛІС є приєднання його під час проектування, тобто в умовах порушеної секретності. Наприклад, можна вставити в проект віртуальний модуль від стороннього виробника з прихованим у нього ЗСАЗ, вбудувати в програмні засоби автоматизованого проектування програму-генератор, яка непомітно припасує ЗСАЗ до проекту [10].

Приховані канали у ЗСАЗ

Прихований канал – це засіб передачі інформації між об′єктами способом, не передбаченим для пересилання інформації і не дозволеним правилами політики захисту [3]. Прихований канал може бути аналоговим чи цифровим. *Аналоговий канал* будують, використовуючи різноманітні фізичні явища. Це, наприклад, модульовані струм споживання, температура корпусу, яскравість контрольного світлодіоду, нарешті, радіохвилі, що випромінюються частинами схеми. *Цифровий канал* організують шляхом статистичної модуляції певних розрядів даних, які передаються через легальні канали або завдяки модуляції штатних подій, наприклад, сигналів переривання, помилки або неготовності блоку таким чином, щоб їх поведінка була правдоподібною.

Прихований канал конструюють з урахуванням того, щоб його було неможливо або важко виявити на фоні завад звичайними засобами протидії. Так, спеціально сконструйований ЗСАЗ, що реалізує прихований радіоканал, може мати радіус дії до кількох десятків метрів, причому сигнал каналу важко виявити через його шумоподібну модуляцію [12]. Для надійної пересилки даних слід синхронізувати як прийом, так і їх передачу. Оскільки можливості ЗСАЗ обмежені, передача даних найчастіше є несинхронізованою. Через це канал налаштований на повільну передачу певного типу даних, наприклад, лише криптографічного ключа.

Прихований канал часто конструюють двонаправленим. Зворотний напрямок використовується для синхронізації та активізації ЗСАЗ.

Складні ОС включають в себе блоки з різним ступенем захисту. Тоді прихований канал може зв′язувати блок з високим ступенем захисту з блоками з меншими ступенями захисту. Доволі ефективним прихованим каналом є ресурс, який розділяється між цими блоками. Одним з таких ресурсів є кеш-пам′ять системи. Організація прихованих каналів через такі розділені ресурси широко вживана при використанні шкідливого матзабезпечення. Їх застосування при використанні ЗСАЗ досліджене у [3].

Способи виявлення ЗСАЗ

Можна виділити три основних підходи до виявлення ЗСАЗ у мікросхемах. Це фізичний аналіз топології кристала, тестування з автоматичною генерацією тестів та аналіз сигналів непрямих каналів передачі інформації [9]. Топологію кристала аналізують за допомогою автоматизованих засобів, що застосовуються при організації виробництва мікросхем, наприклад, скануючого електронного мікроскопу. При цьому тестована мікросхема порівнюється з еталонною. Такий аналіз є дорогим та надзвичайно повільним. Крім того, мікросхему слід обережно вилучити з корпусу та препарувати, що також є складним завданням. При зменшенні проектних норм мікросхем зменшується вірогідність виявлення ЗСАЗ та зростає ціна обладнання.

Згідно з другим підходом, використовуються стандартні засоби тестування мікросхем на основі автоматизованої генерації тестових послідовностей. При цьому тестування має бути налаштоване на реєстрацію параметричних змін у схемі у разі пошуку параметричного ЗСАЗ, або на виявлення модифікації логічної схеми при пошуку функціональних ЗСАЗ.

На жаль, у другому випадку тестування є неефективним через певні причини. Відомо, що задача виявлення шкідливого програмного забезпечення зводиться до задачі зупинки машини Т'юрінга і тому не може бути вирішеною у загальному випадку [3,12]. Те саме стосується виявлення ЗСАЗ. Про складність цієї задачі свідчить приклад розробки троянського ЗСАЗ, призначеного для створення прихованого каналу мікропроцесора. Два варіанти ЗСАЗ мають, відповідно, 959 та 1341 логічних вентилів та складають лише 0.05% та 0.08% від загальних апаратних витрат ОС [4]. Без знань про логічну схему цих ЗСАЗ, місця їх підключення, способу і деталей активізації практично неможливо знайти тестову послідовність, яка виявляє наявність цих ЗСАЗ.

Аналіз непрямих каналів витоку інформації (side channels) дає змогу виявити присутність ЗСАЗ. Наприклад, можна ввімкнути мікросхему та виміряти аналогові сигнали відгуку, такі як струм споживання, інфрачервоне та радіовипромінювання. Через аналіз споживаного струму потенційно можливо виявити навіть неактивоване ЗСАЗ, тому що схема слідкування за умовою активізації весь час споживає деякий струм та виконує певні перемикання, які відмінні від струму та перемикань оригінальної мікросхеми.

Ефективність аналізу непрямих каналів зростає, якщо в мікросхему було впроваджено спеціальні модулі, які сприяють цьому аналізу. Це можуть бути датчики напруги, температури у різних точках кристала, модулі вимірювання затримки (додавання провідників ЗСАЗ збільшує затримку), а також схеми вбудованого самотестування [2,3].

На відміну від складного аналізу НВІС, виявлення ЗСАЗ у ПЛІС є тривіальним. Будь-яка корекція файла прошивки змінює його контрольний код. Крім того, існуючі програмні засоби дають змогу порівнювати між собою проекти на різних стадіях проектування і виявляти між ними відмінності, включаючи небажані логічні схеми [10].

Попередження впровадження ЗСАЗ

Шкідливу дію ЗСАЗ можна унеможливити або компенсувати завдяки спеціальним заходам структурного проектування ОС. Вище було згадано про спосіб резервування, що протидіє знищувальному перемикачу, а також про впровадження інфраструктурних модулів – параметричних датчиків, які допомагають виявити чи протидіяти ЗСАЗ. Далі перелічено деякі інші заходи.

Найбільш ефективним способом є просторова та логічна ізоляція відповідальних блоків від решти схеми ОС (спосіб *moats and drawbridges* – буквально – замковий рів та ланцюговий міст). Така ізоляція істотно ускладнює налаштування прихованого каналу та спрощує його виявлення.

Для усунення прихованого каналу через блок з роздільним доступом слід використовувати механізм підтвердження звертання до такого блоку. Тоді схема моніторингу звертання буде відмовляти у доступі іншим блокам, включаючи ЗСАЗ, які не мають на це прав [3].

Для мінімізації витоку інформації через побічні канали слід використовувати логічні схеми, в яких або мінімізовано шум струму перемикань, або цей шум маскується додатковим генератором шуму. При таких умовах організація прихованого аналогового каналу значно ускладнюється [13].

Досліджуючи структуру та виконувані алгоритми існуючої ОС, можна передбачити місця та принцип дії потенційних ЗСАЗ і на основі цього організувати відповідні попереджувальні заходи. Таке дослідження проводиться, як правило, шляхом моделювання [2,14].

Висновки

Останнім часом злонамірено створеному апаратному забезпеченню (ЗСАЗ) у світі приділяється багато уваги через те, що воно становить серйозну загрозу безпеці держави та людей. Виявлення чи знешкодження ЗСАЗ, що вбудовані в НВІС, є дуже складним завданням, яке ускладнюється зі зростанням ступеня інтеграції НВІС.

Обчислювальні системи на базі ПЛІС мають суттєві переваги в тому, що значно ускладнене впровадження ЗСАЗ у ПЛІС та суттєво спрощене його виявлення у порівнянні з НВІС. Найдешевший і найнадійніший спосіб унеможливити впровадження ЗСАЗ у ПЛІС – це виконання відповідних заходів безпеки під час розробки файла конфігурації проекту.

Структурними та логічними заходами, що попереджають впровадження ЗСАЗ, є проектування з резервуванням, просторова та логічна ізоляція відповідальних блоків, керування доступом до розділених ресурсів, мінімізація шуму перемикань, впровадження інфраструктурних модулів протидії ЗСАЗ.

Розробка, виявлення, протидія та попередження ЗСАЗ – це коло завдань, які потребують подальших багатосторонніх науково-технічних досліджень.

Література:

1. *Adee S.* The Hunt for the Kill Switch // IEEE Spectrum, 2008. – N5. – P. 34-39.

2. *King S.T.*,Tucek J., Cozzie A., Grier C., Jiang W., Zhou Y. Designing and implementing malicious hardware // Proceedings of the 1-st Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'08. USENIX Association Berkeley, CA, USA, 2008. –8 p.

3. *Huffmire T.*, Irvine C., Nguyen T.D., Levin T., Kastner R., Sherwood T. Handbook of FPGA Design Security. –Springer, 2010. – 177 p.

4. *Security* Trends for FPGAs. From Secured to Secure Reconfigurable Systems / Editors*:* B.Badrignans, G.Gogniat, J.L.Danger, L.Torres, V.Fischer. –Springer, 2011. – 196 p.

5. *Lieberman J.I.* National security aspects of the global migration of the US semiconductor industry // White Paper, June 2003. Congressional Record: June 5, 2003 (Senate) – P. S7468-S7471.

6. *Obama B.H.* Remarks by the President on Securing Our Nation's Cyber Infrastructure (Washington DC, May 29, 2009), режим доступу: http://www.whitehouse.gov/the_press_office/Remarks-by-the-President-on-Securing-Our-Nations-Cyber-Infrastructure/

7. *Secure* Semiconductors: Sensible, or Sisyphean? From TIC to IRIS at DARPA. // Defense Industry Daily, 2011. – 16 Feb., режим доступу: http://www.defenseindustrydaily.com/Secure-Semiconductors-Sensible-or-Sisiphyean -04928/

8. *Hadzic I.*, Udani S., Smith J., FPGA viruses // Proceedings of the 9-th Int. Workshop on Field-Programmable Logic and Applications (FPL'99), Glasgow, UK, August, 1999.

9. *Wang X.*, Tehranipoor M., Plusquellic J. Detecting malicious inclusions in secure hardware: challenges and solutions // Proc. IEEE Workshop on Hardware Oriented Security and Trust (HOST), Anaheim, CA, June, 2008, 2008. – P. 15-19.

10. *Trimberger S.* Trusted design in FPGAs // Proceedings of the 44th Design Automation Conference, San Diego, CA, USA, DAC 2007, June 4–8, 2007, – P. 1.5-1.8.

11. *Sarma S.E.,* Weis S.A., Engels D.W. Radio-Frequency Identification: Security Risks and Challenges // RSA Laboratories Cryptobytes, Spring 2003. –V.6, N1. – P. 2-9.

12. *Котов В.Е.* Введение в теорию схем программ. – Новосибирск-: Наука, 1978. – 256 с.

13. *Tiri K.,* Verbauwhede I. Synthesis of Secure FPGA Implementations // Proceedings of International Workshop on Logic and Synthesis, 2004. – P. 224-231.

14. *Smith S.C.*, Di J. Detecting Malicious Logic Through Structural Checking // Proc. IEEE Region 5 Techn. Conf., 20-22 April 2007, 2007. – P. 217-222.