МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Хмельницький національний університет
Військовий інститут Київського національного університету
ім.Тараса Шевченка
ПВНЗ “Університет економіки і підприємництва”

Інтелектуальний потенціал - 2016

МАТЕРІАЛИ

Всеукраїнської науково-практичної конференції молодих науковців та студентів

1-3 грудня 2016р.

Частина 2

м. Хмельницький
2016

ББК 74.480.278
С.88

Інтелектуальний потенціал - 2016 (Ч.2): Матеріали Всеукраїнської науково-практичної конференції молодих науковців та студентів. 1-3 грудня 2016р., м. Хмельницький/Кол. авт. – Хмельницький: УЕП, 2016. – 100 с.

ЗМІСТ

Аналіз проблем дослідження в області програмної надійності

Бабій І.М.
Науковий керівник – к.т.н., доц. Огнєвий О.В.
Хмельницький національний університет

Теорія надійності як наука отримала розвиток стосовно складних технічних систем. Необхідність і корисність контролю технічних компонент систем і систем в цілому, з метою перевірки відповідності їх поточних характеристик заданим, доведені практикою. У цьому плані виконана значна кількість робіт по надійності стосовно технічних систем, розроблено безліч моделей забезпечення розумними методами надійності складних систем і їх технічної готовності.

Ці моделі у ряді випадків дозволяють не лише оцінювати показники надійності і готовності технічних систем і їх компонентів, але і дають можливість передбачати значення цих показників на основі накопиченого досвіду. Крім того, ряд моделей дозволяє на основі накопичених даних висловлювати припущення відносно режимів роботи, при яких найчастіше проявляються відхилення від нормального функціонування, а також про вживаний підхід до відновлення (ремонту) системи або її компонентів після збою.

Під системою в теорії надійності прийнято розуміти сукупність підсистем або елементів, функціонально об'єднаних відповідно до деякого алгоритму взаємодії при виконанні заданого завдання в процесі застосування за призначенням. Під це визначення системи повністю підходить програмне забезпечення. У роботі [2] вказується, що дослідження в області програмної надійності знаходяться на початковій стадії свого розвитку.

До основних проблем досліджень надійності ПЗ відносяться:

1) розробка методів оцінки і прогнозування надійності ПЗ;
2) визначення основних чинників, що впливають на надійність ПЗ;
3) розробка методів, що забезпечують досягнення заданого рівня надійності ПЗ;
4) вдосконалення методів підвищення надійності ПЗ в процесі проектування і експлуатації.

Головна причина помилок в ПЗ - це його складність. Для боротьби із складністю виділяються дві концепції: незалежність та ієрархічна структура.

У роботі [1] наводиться правило "n ± 1": Перевірка правильності фази n проекту повинна здійснюватися проектувальниками (виконавцями) фаз (n+1) і (n - 1). Крім того, в [1] наводиться обґрунтування необхідності як можна більше раннього виявлення помилок проектування ПЗ. Воно полягає в тому, що вартість виправлення помилки з часом зростає (рис. 1а), а вірогідність правильно виправити помилку - падає (рис. 1б).

При цьому вірогідність правильно виправити помилку знаходиться в протиріччі з вірогідністю виявити помилку. Вірогідність виявити помилку зростає з часом при уточненні вимог замовника і під час дослідної

експлуатації. В зв'язку з цим важливо вирішити завдання оптимізації часу виявлення помилки при мінімальних витратах на її виправлення.

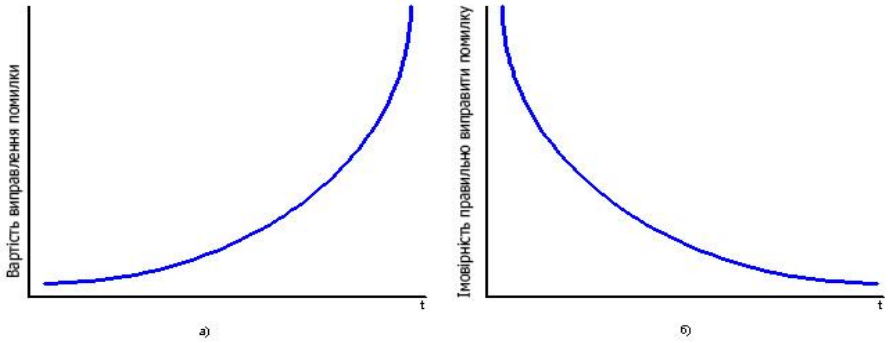


Рисунок 1 - Обґрунтування необхідності раннього виявлення помилки

Процентні частоти появи помилок в ПЗ по типах помилок представлені в таблиці1.

Таблиця 1 - Процентні частоти появи помилок в ПЗ

Тип помилки	Частота появи %
Не повна або помилкова специфікація	28
Відхилення від специфікації	12
Нехтування правилами програмування	10
Помилкова вибірка даних	10
Помилкова логіка або послідовність операцій	12
Помилкові арифметичні операції	9
Брак часу для вирішення	4
Помилка обробки переривань	4
Помилка у вихідних даних	3
Неточний запис	8

Як видно з таблиці основна кількість помилок робиться із-за невірної специфікації або ТЗ.

Література

1. Майерс Г. Надежность программного обеспечения. / Г. Майерс – М.: Мир, 1980. – 360 с.
2. Гаспер Б.С. Надежность функционирования автоматизированных систем./ Б.С Гаспер. – Пермь: ПГТУ, 1999. – 70 с.

Аналіз методів підвищення ефективності роботи функції системи стеження за об'єктом

Білик О.М., Козак С.
Науковий керівник – д.т.н., проф Мясіщев О.А.
Хмельницький національний університет

Метою роботи є аналіз та вдосконалення існуючих пристроїв, так званих «безпілотників», в яких є можливість слідкування за об'єктом, але мають проблеми в реалізації цієї функції. Завдання підвищення ефективності розпізнавання і пошуку об'єктів. Тому для вирішення цього завдання будем використовувати наступні методи.

Розпізнавання з урахуванням яскравості.

При такому підході за основу береться підмножина пікселів зображення, яка відповідає образу який ми розпізнаєм, і визначаються дані про характеристики як дані про самі вихідні значення яскравості пікселів. Ще один варіант цього методу полягає в тому, що спочатку може бути виконана згортка зображення з різними лінійними фільтрами, після чого значення пікселів в результуючих зображеннях розглядаються як характеристики. Такий підхід виявився дуже успішним при вирішенні таких завдань, як розпізнавання рукописних цифр.

Для створення детекторів обличь, що дозволяють розпізнавати обличчя за допомогою баз даних з прикладами, використовувався цілий ряд статистичних методів, включаючи методи на основі нейронних мереж з необробленими вхідними даними, представленими у вигляді характеристик пікселів; дерева рішень з характеристиками, обумовленими за допомогою різних фільтрів смуг і країв. Деякі результати застосування останнього методу показані на рис 1.

Рисунок 1 - Приклад розпізнавання обличь з урахуванням яскравості.

Одним з недоліків методу, в якому в якості векторів характеристик використовуються необроблені дані про пікселі, є велика надмірність,

властива цьому способу представлення. Припустимо, що розглядаються два пікселя, розташовані поруч на зображенні, щоки лиця якоїсь людини; між ними, швидше за все, буде досить висока кореляція, оскільки для них властиві аналогічне геометричне розташування, освітленість і т.д. Для скорочення кількості розмірностей вектора характеристик можна з успіхом застосовувати методи зменшення обсягу даних, такі як аналіз найбільш важливих компонентів; використання таких методів забезпечує розпізнавання образів, подібних обличчю, з більшою швидкодією в порівнянні з тим, що може бути досягнуто з використанням простору більшої розмірності.

Розпізнавання з урахуванням характеристик.

Замість застосування в якості характеристик необроблених даних про яскравість пікселів можна використовувати методи виявлення та розмітки просторово локалізованих характеристик, таких як поля і краї. Застосування країв є доцільним завдяки двом описаним нижче важливих причин. Однією з них є зменшення обсягу даних, пов'язане з тим, що кількість країв набагато менше в порівнянні з кількістю пікселів зображення. Друга причина обумовлена можливістю домогтися інваріантності освітленості, оскільки краї (при наявності відповідного діапазону контрастів) виявляються приблизно в одних і тих же місцях, незалежно від точної конфігурації освітленості. Краї представляють собою одномірні характеристики; були також зроблені спроби використовувати двомірні характеристики (ділянки) і безмірні характеристики (точки). Зверніть увагу на те, як відрізняються трактування просторового розташування в підходах з урахуванням яскравості і з урахуванням характеристик. У підходах з урахуванням яскравості ці дані кодуються неявно, як індекси компонентів вектора характеристик, а в підходах з урахуванням характеристик, характеристикою є саме місцезнаходження (x, y).

Невід'ємною властивістю будь-якого об'єкта є інваріантне розташування країв; саме з цієї причини люди можуть легко інтерпретувати контурні малюнки, навіть незважаючи на те, що подібні зображення не зустрічаються в природі. Найпростіший спосіб використання цих знань заснований на класифікаторі по найближчих сусідніх точках. При цьому попередньо обчислюються і зберігаються дані про зміни країв, всім відомим образам. А після отримання конфігурації країв, яка відповідає невідомому об'єкту на зображенні, що є предметом запиту, можна визначити "відстань" цього об'єкта від кожного елемента бібліотеки де зберігаються образи. Після цього класифікатор по найближчих сусідніх точках вибирає найбільш похожий образ.

Оцінка пози.

Інтерес представляє не тільки завдання визначення того, яким є певний об'єкт, але і завдання визначення його пози, тобто його позиції і орієнтації по відношенню до спостерігача. Наприклад, при вирішенні проблеми маніпулювання об'єктами на виробництві доводиться враховувати, що захоплення робота не може взяти об'єкт до тих пір, поки не буде відома його

поза. У разі жорстких об'єктів, як тривимірних, так і двомірних, ця проблема має просте і цілком визначене рішення, засноване на методі вирівнювання, який описаний нижче.

У цьому методі об'єкт представляється за допомогою характеристик, або помітних точок m1, m2,…mN в тривимірному просторі (як такі можуть, припустимо, розглядатися вершини багатогранного об'єкта). Координати цих точок вимірюються в деякій системі координат, найбільш підходящою для даного об'єкта. Після цього точки піддаються операції тривимірного обертання R з невідомими параметрами, за якою слідує операція перенесення на невідому величину t, а потім виконується операція проекції, яка призводить до появи точок характеристик зображення p1, p2,… pN на площині зображення. Оскільки деякі точки моделі можуть закривати один одного, а детектор характеристик може пропустити деякі характеристики (або виявити помилкові характеристики, поява яких обумовлена наявністю шуму). Таке перетворення для тривимірної моделі точок mi і відповідних точок зображення pi можна представити таким чином: де R - матриця обертання, t - вектор переносу; Чистим результатом стає трансформація Q, яка призводить точки моделі mi у відповідність з точками зображення pi. При цьому, хоча спочатку трансформація Q не визначена, відомо, що (застосовується до жорстких об’єктів) Q повинна бути однаковою для всіх точок моделі.

OpenCV

OpenCV - розроблена Intel на C/C++ бібліотека з відкритим вихідним кодом, здатна значно спростити програмування комп'ютерного зору. Вона надає багато можливостей: детектування, відстеження та розпізнавання осіб, фільтрація Кальмана і різні готові до використання методи штучного інтелекту (II). Крім того через низькорівневий API надається велика кількість базових алгоритмів комп'ютерного зору.

Розуміння роботи методів дозволить отримати хороші результати від використання OpenCV. Тут буде описано, як, використовуючи OpenCV, реалізувати детектування, відстеження та розпізнавання осіб. Крім цього буде показано, як працює кожен метод і представлено кілька трюків для досягнення найкращих результатів.

Перша версія OpenCV була випущена Intel в 1999р. Спочатку для її використання потрібна Image Processing Library. Але ця залежність була видалена і тепер OpenCV може використовуватися незалежно.

OpenCV є багатоплатформною бібліотекою, вона підтримує Windows, Linux і MacOSX. З одним лише исключеним (модуль CVCAM, який буде описаний нижче), її інтерфейси платформо-незалежні.

Особливості

OpenCV має дуже багато можливостей і спочатку може здатися складною, але щоб почати знайомство потрібно зовсім не багато. Тут короткий опис найважливіших функціональних категорій в бібліотеці OpenCV версії 1.0:

Введення/виведення зображень і відео. Ці функції дозволяють читати

зображення з файлів або з відео-потоків, а так само створювати зображення і відео.

Основні алгоритми комп'ютерного зору і обробки зображень (середньо- і низькорівневі API). Використовуючи цей інтерфейс можна експериментувати з більшістю стандартних алгоритмів комп'ютерного зору, без необхідності їх кодувати. Це включає пошуку перепадів, ліній і кутів, апроксимація еліпсом, піраміди зображень для різномасштабної обробки, порівняння з еталоном (pattern matching), різні варіанти перетворень (Фур'є, дискретне косинус-перетворення) і багато іншого.

Модуль високорівневого комп'ютерного зору OpenCV включає кілька високорівневих можливостей, наприклад оптичний потік (optic flow) (використання руху камери для визначення тривимірної структури), калібрування камери і стереозору.

Методи II і машинного навчання. Для додатків, що використовують комп'ютерний зір, часто потрібні методи машинного навчання або інші методи II. Деякі з них доступні в пакеті OpenCV Mashine Learning.

Складання вибірки зображення і перетворення виду. Часто буває корисно обробляти групу пікселів, як блок. OpenCV містить функції для виділення областей зображення, випадкової вибірки, зміни розміру, обгортка, повороту і застосування ефектів перспективи.

Методи для створення і аналізу бінарних (двозначних) зображень. Бінарні зображення часто використовуються в перевірочних системах, які сканують на предмет дефекту форми або кількості частин. Бінарне представлення також зручно, коли об'єкт не рухомий.

Методи для обчислення тривимірної інформації. Ці функції корисні для складання карт і виявлення з використанням стерео оснащення або декількох зображень, отриманих однією камерою з різних точок огляду.

Математичні функції для обробки зображень, комп'ютерного зору і інтерпретації зображень. OpenCV включає часто використовувані алгоритми лінійної алгебри, статистики та обчислювальної геометрії.

Графіка. Цей інтерфейс дозволяє писати текст і малювати на зображеннях. Крім того ці функції корисні для для маркування та створення різних знаків. Наприклад, якщо ви пишете програму для детектування об'єкта, корисно написати на зображенні розмір і положення об’єкта.

Методи GUI. OpenCV містить власні функції для виведення вікон. Вони представляють просту мультиплатформену API для виведення зображень, допускають обробку подій миші та клавіатури і реалізують керуючий елемент повзунок.

Структури даних і алгоритми. З цими функціями ви можете ефективно створювати і маніпулювати великими списками, послідовностями, графами і деревами.

Отже, такі методи покликані для вирішення поставлених задач в яких використовуються методи теорії, методи обробки цифрових зображень, теорії нейронних мереж. Програма, яка використовуватиме алгоритм, буде написана на мовах C# и C++ з використанням бібліотеки OpenCv.

Література
1. Бойко И. А. Распознавание объектов на основе видеосигнала, полученного с камеры, установленной на подвижной платформе / Бойко И. А. –М.: Молодой ученый, 2013.
2. Юревич Е. И. Основы робототехники. — М.: БХВ-Петербург, 2007.
3. Степанченко И.В. Нейронные сети для распознавание образов / Волгоград. гос. техн. ун-т. –М.: Волгоград, 2004.

Сигнальні процесори – перспективні тенденції

Величко Я.О.
Науковий керівник – к.т.н. Гурман І.В.
Хмельницький національний університет

Процесори цифрової обробки сигналів представляють собою клас спеціалізованих мікропроцесорів, призначених для вирішення завдань цифрової обробки сигналів (ЦОС), до яких відносяться обробка звукових сигналів, обробка зображень, розпізнавання мови, розпізнавання образів, цифрова фільтрація, спектральний аналіз та ін

Часто в літературі такі мікропроцесори називаються цифровими сигнальними процесорами (ЦСП), або DSP (Digital Signal Processors).

Перші процесори цього класу з'явилися наприкінці 1970-х років. Вимоги практики, пов'язані з широким розвитком мобільного бездротового зв'язку, стаціонарних систем широкосмугового зв'язку, використанням цифрової обробки сигналів в побутової аудіо-і відеотехніки і пристроях комп'ютерної периферії, з одного боку і колосальний прогрес електронної промисловості з іншого привели до того, що до теперішнього часу продуктивність ЦСП зросла до сотень мільйонів операцій в секунду, а ціна впала більш ніж на 90% і навіть для самих потужних процесорів складає в даний час менш $ 20. Низька споживана потужність (менше 1 Вт на максимальній частоті роботи процесора) забезпечує їх широке використання в різних вбудованих пристроях від побутової електроніки до бортових систем спеціального призначення.

Цифрова обробка сигналів - це арифметична обробка послідовності значень амплітуд сигналу, одержуваних через рівні проміжки часу. Головне, що відрізняє цю інформацію, - вона необов'язково заноситься в пам'ять і тому може виявитися недоступною в майбутньому. Отже, обробляти її потрібно в реальному масштабі часу (РМЧ).

До основних особливостей цифрової обробки сигналів , які багато в чому визначають архітектуру процесорів DSP , відносяться:

- потоковий характер обробки великих обсягів даних в РМЧ;
- забезпечення можливості інтенсивного обміну з зовнішніми пристроями.

Для ефективної реалізації алгоритмів цифрової обробки сигналів необхідна апаратна підтримка базових операцій ЦГЗ. Тому розглянемо

спочатку принципи цифрової обробки сигналів, оказ вающие особливий вплив на архітектуру ЦСП.

Будь аналоговий сигнал можна представити у вигляді характеристик-або амплітуда-час, або частота-амплітуда. Для переходу від однієї форми подання до іншої використовується перетворення Фур'є . Операції, які виконують це перетворення, є базовими в цифровій обробці сигналів .

Перетворення Фур'є є в загальному випадку роботою з деякою безперервної функцією. З безперервним перетворенням Фур'є зручно працювати в теорії, але на практиці ми зазвичай маємо справу з дискретними даними. Для обробки звукових і відеосигналів, в комп'ютері вони спочатку перетворюються в цифрову форму і подаються у вигляді деякого набору відліків частот і амплітуд, вироблених через певні проміжки часу (період дискретизації). Тому тут варто говорити не про інтегральне, а про дискретні перетворення Фур'є (ДПФ).

Нехай дана кінцева послідовність чисел x_0 , x_1 , x_2 , ..., X_{N-1} (в загальному випадку комплексних). Дискретне перетворення Фур'є полягає в пошуку іншої послідовності X_0 , X_1 , X_2 , ..., X_{N-1} , елементи якої обчислюються за формулою:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\Pi * kn}{N}}$$

Нехай дана кінцева послідовність чисел X_0 , X_1 , X_2 , ..., X_{N-1} (в загальному випадку комплексна). Зворотне дискретне перетворення Фур'є полягає в пошуку іншої послідовності x_0 , x_1 , x_2 , ... , X_{N-1} , еле менти якої обчислюються за формулою:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{-j2\Pi * kn}{N}}$$

У загальному випадку гармонійне коливання подане на рисунку 1, описується наступним виразом:

$$x = A \sin\left(\frac{2\pi * t}{T} + \varphi\right)$$

де A 0 - амплітуда сигналу, T - його період, ϕ - зсув фази сигналу.

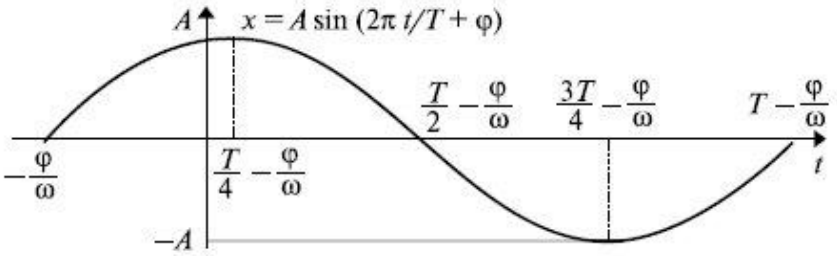


Рисунок 1 - Загальний вид гармонійного коливання

Це коливання можна описати також виразом:

$$x = A\cos(\omega t + \varphi)$$

де $\omega = 2\pi/T$ - кругова частота сигналу. Цей вираз будемо називати гармонікою.

Розглянемо тепер функцію x = f (t) , що представляє собою деякий звукове або якесь інше коливання. Нехай це коливання описано графіком на тимчасовому інтервалі [0, T] (рисунок 2).

Для обробки цього сигналу в комп'ютері потрібно виконати його дискретизацію. З цією метою часовий інтервал ділиться на N-1 частин і зберігаються значення функції x_0 , x_1 , x_2 , ..., X_{N-1} для N точок на кордонах інтервалів.

В результаті прямого дискретного перетворення Фур'є можуть бути отримані N значень для X_K.

Параметри кожної гармоніки обчислюються прямим перетворенням Фур'є, а сума гармонік - зворотним.

Тепер, наприклад, операція "фільтр нижніх частот", яка "вирізає" з сигналу всі частоти вище деякої заданої, може просто обнулити коефіцієнти, що відповідають частотам, які необхідно видалити. Потім, після обробки, виконується зворотне перетворення.

Особливості цифрової обробки сигналів розглянемо на прикладі алгоритму нерекурсивною фільтрації. Структура пристрою, реалізую щего даний алгоритм, показана на рис. 3.

Обробка полягає у формуванні вихідного сигналу Y [k] за значеннями N останніх вхідних відліків x [k] , які надходять на вхід пристрою через певний інтервал часу T . Прийняті відліки зберігаються в комірках циклічного буфера. При прийомі чергового відліку вміст всіх комірки буфера переписується в сусідню позицію, найстаріший відлік залишає буфер, а новий записується в його молодшу клітинку.

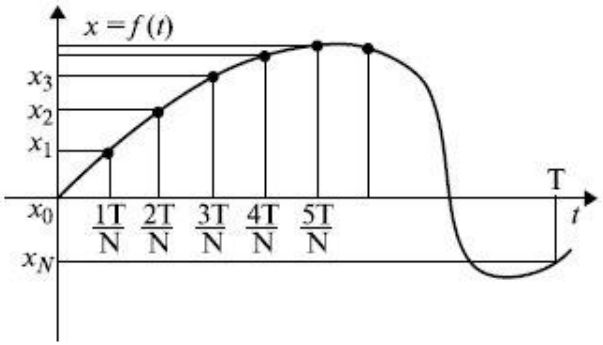


Рисунок 2 - Дискретизація гармонійної функції

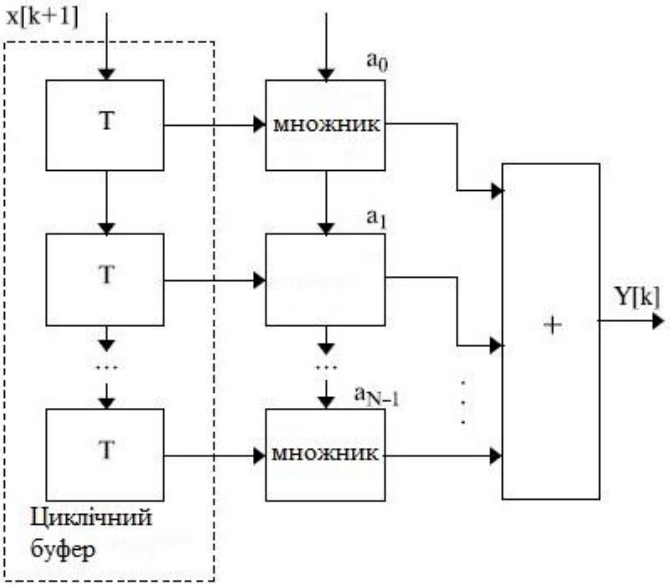


Рисунок 3 - Схема роботи нерекурсивного фільтра

Аналітично алгоритм роботи нерекурсивні фільтри записуючи ється як:

$$Y[k] = \sum_{i=1}^{N-1} x[k-1] * a_i$$

де A_I - коефіцієнти, які визначаються типом фільтру.

Відлік з виходів елементів буфера надходять на помножувачі, на другі входи яких поступають коефіцієнти A_I . Результати творів складаються і формують відлік вихідного сигналу Y [k] , після чого вміст буфера зсувається на 1 позицію і цикл роботи фільтра повторюється. Вихідний сигнал Y [k] повинен бути обчислений до надходження наступного вхідного сигналу, тобто за інтервал T. У цьому полягає суть роботи пристрою в реальному масштабі часу. Інтервал часу T задається частотою дискретизації, яка визначається областьюпріменення фільтра. За слідству з теореми Котельникова в дискретному сигналі період, що відповідає найвищій представимо частоті, відповідає двом періодам дискретизації. При обробці звукового сигналу частоту дискретизації можна прийняти в 40 кГц. В цьому випадку якщо необхідно реалізувати цифровий нерекурсівние фільтр 50-го порядку, то за час в 1/40 кГц = 25 мкс повинно бути виконано 50 множень і 50 накопичень результатів множення. Для обробки відеосигналу інтервал часу, за який мають бути виконані ці дії, буде на кілька порядків менше.

Якщо виконувати ДПФ вхідний послідовності безпосередньо, строго по вихідній формулі, то буде потрібно багато часу. Порахувавши за визначенням (N раз підсумовувати N доданків), отримуємо величину порядку N_2 .

Тим не менш, можна обійтися істотно меншим числом операцій.

Найбільш популярним з алгоритмів прискореного обчислення ДПФ є метод Кулі-Тьюкі (Cooley-Tukey), що дозволяє обчислити ДПФ для числа відліків N = 2k за час порядку N * $\log_2$ N (звідси і назва - швидке перетворення Фур'є , ШПФ, або в англійському варіанті FFT - Fast Fourier TRansformation). Основна ідея цього методу полягає в рекурсивному розбитті масиву чисел на два підмасива і зведенні обчислення ДПФ від цілого масиву до обчислення ДПФ від підмасива окремо. При цьому процес розбиття вихідного масиву на подмассіва проводиться за методом побітової зворотного сортування (bit-reversal sortINg).

Спочатку вхідний масив ділиться на дві подмассіва - з парними і непарними номерами. Кожен з подмассіва перенумеровуються і знову ділиться на два подмассіва - з парними і непарними номерами. Ця сортування продовжується до тих пір, поки розмір кожного подмассіва не досягне 2 елементів. В результаті (що можна показати математично) номер кожного вихідного елементу в двійковій системі перевертається. Тобто, наприклад, для однобайтних номерів двійковий номер 00000011 стане номером 110000000, номер 01010101 - номером 10101010.

Існують алгоритми ШПФ для випадків, коли N є ступенем довільного простого числа (а не тільки двійки), а також у разі, коли число N є твором ступенів простих чисел будь-якого числа відліків. Однак ШПФ, реалізоване за методом Кулі-Тьюкі для випадку N = 2k, отримало найбільш широке поширення. Причина цього в тому, що алгоритм, побудований за цим методом, має низку дуже хороших технологічних властивостей:

- структура алгоритму і його базові операції не залежать від числа відліків (змінюється тільки число прогонів базової операції);

• алгоритм легко розпаралелюється з використанням базової операції і конвейерізуется;

• алгоритм простий і компактний, допускає обробку даних "на місці" і не вимагає додаткової оперативної пам'яті.

Однокристальний мікроконтролери і навіть універсальні мікропроцесори виявляються повільними при виконанні операцій, характерних для цифрової фільтрації. До того ж вимоги до якості перетворення аналогових сигналів постійно зростають. У сигнальних мікропроцесорах такі операції підтримуються на апаратному рівні і виконуються, відповідно, досить швидко. Робота в реальному масштабі часу вимагає від процесора також підтримки на апаратному рівні таких дій, як обробка переривань, програмних циклів.

Він повинен управляти роботою різних компонентів апаратного забезпечення, таких як дисководи, графічні дисплеї, мережевий інтерфейс, з тим щоб забезпечити їх злагоджену роботу. Це призводить до досить складної архітектурі, оскільки вона повинна підтримувати поряд з цілочисленою арифметикою та операціями з плаваючою комою такі базові функції, як захист пам'яті, мультипрограмування, обробка векторної графіки і т. п. В результаті типовий універсальний мікропроцесор з CISC-, а часто і RISC-архітектурою має систему з кількох сотень команд, які забезпечують виконання всіх цих функцій, і відповідну апаратну підтримку. Це веде до необхідності мати в складі такого МП десятки мільйонів транзисторів.

У той же час DSP-процесор є вузькоспеціалізованим пристроєм. Його єдине завдання - швидко обробляти потік цифрових сигналів. Він складається головним чином з високошвидкісних апаратних схем, які виконують арифметичні функції і маніпулюють битами, оптимізованих таким чином, щоб швидко обробляти великі обсяги даних. В силу цього набір команд у DSP куди менше, ніж у універсального мікропроцесора: їх число зазвичай не перевищує 80. Це означає, що для DSP потрібно полегшений декодер команд і набагато менше число виконавчих пристроїв. Крім того, всі виконавчі пристрої в кінцевому підсумку повинні підтримувати високопродуктивні арифметичні операції. Таким чином, типовий DSP-процесор складається не більше ніж з декількох сот тисяч (а не десятків мільйонів, як в сучасних CISC-МП) транзисторів. В силу цього такі МП споживають менше енергії, що дозволяє використовувати їх в продуктах, що працюють від батарей. Вкрай спрощується їх виробництво, тому вони знаходять собі застосування в недорогих пристроях. Поєднання низького енергоспоживання і невисокій вартості дозволяє використовувати їх не тільки у високій сфері телекомунікацій, але і в стільникових телефонах і роботах-іграшках.

Особливості архітектури процесорів цифрової обробки сигналів

Відзначимо основні особливості архітектури процесорів цифрової обробки сигналів :

1. Гарвардська архітектура, основу якої становить фізичне і логічний поділ пам'яті команд і пам'яті даних. Ключові команди DSP-процесора є багатоопераннимі, і прискорення їх роботи вимагає одночасного читання

декількох комірок пам'яті. Відповідно на кристалі є роздільні шини адреси і даних (у деяких типах процесорів - кілька шин адреси і даних). Це дозволяє поєднувати в часі вибірку операндів і виконання команд. Використання модифікованої гарвардської архітектури припускає, що операнди можуть зберігатися не тільки в пам'яті даних, але і в пам'яті команд разом з програмами. Наприклад, у разі реалізації цифрових фільтрів коефіцієнти можуть зберігатися в пам'яті програм, а значення даних - в пам'яті даних. Тому коефіцієнт і дані можуть вибиратися в одному машинному циклі. Для забезпечення вибірки команди в тому ж машинному циклі використовується або кеш-пам'ять програм, або дворазове звернення до пам'яті програм за час машинного циклу.

2. Для скорочення часу виконання однієї з основних операцій цифрової обробки сигналу - множення - застосовується апаратний множувач. У процесорах загального призначення ця операція реалізується за кілька тактів зсуву і додавання і займає багато часу, а в DSP-процесорах завдяки спеціалізованому множувачу потрібен всього один цикл. Вбудована схема апаратного множення дозволяє виконати за 1 такт основну операцію ЦГЗ - множення з накопиченням (MultIPly-Accumulate - MAC) для 16 - та / або 32-розрядних операндів.

3. Апаратна підтримка циклічних буферів. Наприклад, для фільтру, представленого на рисунку 2.3 , при кожному обчисленні відліку вихідного сигналу використовується новий відлік вхідного сигналу, який зберігається в пам'яті на місці самого старого. Для такого циркулюючого буфера може використовуватися фіксована область ОЗП. При цьому під час обчислень генеруються лише послідовні значення адрес ОЗП незалежно від того, яка операція - запис або читання - виконується в даний момент. Апаратна реалізація циклічних буферів дозволяє встановити параметри буфера (адреса початку, довжина) в програмі поза тілом циклу фільтрації, що дозволяє скоротити час виконання циклічного ділянки програми.

4. Скорочення тривалості командного такту. Це багато в чому забезпечується прийомами, характерними для RISC-процесорів. Головними з них є розміщення операндів більшості команд в регістрах, а також конвеєризація на рівні команд і мікрокоманд. Конвеєр має від 2 до 10 ступенів, що дозволяє на різних стадіях виконання одночасно обробляти до 10 команд. При цьому використовується генерація адрес регістрів паралельно з виконанням арифметичних операцій, а також багатопортовий доступ до пам'яті. Сюди ж можна віднести і такий прийом, характерний для універсальних мікропроцесорів з EPIC-архітектурою , як застосування команд зі надвеликої довжиною слова (VLIW), що генеруються на стадії компіляції програми. Цьому ж служить і розглянута вище Гарвардська архітектура процесора, характерна для однокристальних мікроконтролерів.

5. Наявність на кристалі процесора внутрішньої пам'яті, що ріднить ЦСП з однокристальними МК. Вбудована в процесор пам'ять зазвичай має значно більшу швидкодію, ніж зовнішня. Наявність вбудованої пам'яті дозволяє значно спростити систему в цілому, зменшити її розміри,

енергоспоживання і вартість. Ємність внутрішньої пам'яті є результатом певного компромісу. Її збільшення веде до подорожчання процесора і збільшує енергоспоживання, а обмежена ємність пам'яті програм не дозволяє зберігати складні алгоритми. Більшість DS P-процесорів з фіксованою точкою мають малу ємність внутрішньої пам'яті, звичайно від 4 до 256 Кбайт, і невисоку розрядність зовнішніх шин даних, що пов'язують процесор з зовнішньою пам'яттю. У той же час ЦСП із плаваючою точкою зазвичай передбачають роботу з великими масивами даних і складними алгоритмами і мають або вбудовану пам'ять великої ємності, або велику розрядність адресних шин для підключення зовнішньої пам'яті (а іноді і те, і інше).

6. Широкі можливості по апаратному взаємодії із зовнішніми пристроями, які включають:

- велика різноманітність інтерфейсів, у тому числі контролери локальної промислової мережі CAN, вбудовані комунікаційні (SCI) і периферійні (SPI) інтерфейси, I2C, UART;
- кілька входів для аналогових сигналів і, відповідно, вбудований АЦП;
- вихідні канали широтно-імпульсної модуляції (ШІМ);
- розвинену систему зовнішніх переривань;
- контролери прямого доступу на згадку.

7. В деяких DSP-сімействах передбачені спеціальні апаратні засоби, що полегшують створення мультипроцесорних систем з паралельною обробкою даних для нарощування продуктивності.

8. DSP-процесори широко використовуються в мобільних пристроях, де споживана потужність є основною характеристикою. Для зниження енергоспоживання в сигнальних процесорах застосовується безліч методів, в тому числі зменшення напруги живлення і введення функцій управління споживанням, наприклад, динамічна зміна тактової частоти, сплячий або черговий режим або відключення не використовуваної в даний момент периферії. Слід зазначити, що ці заходи роблять значний вплив на швидкість роботи процесора і при некоректному використанні можуть привести до непрацездатності проектованого пристрою (як приклад можна згадати деякі мобільні телефони, які в результаті помилок в програмах управління енергоспоживанням іноді переставали включатися) або до погіршення його експлуатаційних характеристик (наприклад, значного часу відновлення працездатності при виході із сплячого режиму).

Напрямки розвитку систем обробки надвеликих об’ємів даних

к.т.н., доц. Джулій Л.В., к.т.н. Ємчук Л.В.
Хмельницький національний університет

В даний час наукова і практична діяльність людини висуває все нові масштабні завдання, що вимагають обробки надвеликих об’ємів даних. Згідно з прогнозами аналітичної компанії IDC до 2020 кількість даних у світі

досягне 40 зеттабайт. У зв'язку з появою задач, що вимагають обробки надвеликих баз даних, необхідні нові ефективні методи паралельної обробки та аналізу таких обсягів даних на багатопроцесорних обчислювальних системах. Фактично єдиним ефективним вирішенням проблеми зберігання і обробки надвеликих об'ємів даних є використання паралельних систем баз даних, що забезпечують паралельну обробку запитів на багатопроцесорних обчислювальних системах.

Багато інновацій та покращень в продуктивності потребують в своєчасному і економічно ефективному аналізі великих обсягів даних. За останнє десятиліття ця потреба привела до появи нових значних інновацій в системах аналітики великомасштабних даних. Паралельні бази даних додали такі методи, як колонкове зберігання і обробка даних. Одночасно з цим були розроблені нові розподілені масиви даних та обчислювальні системи, такі як MapReduce і Bigtable.

Паралельні бази даних, які можна віднести до категорії класичних систем, вони були першими системами для паралельної обробки даних, доступних для широкого класу користувачів за допомогою інтуїтивно зрозумілої моделі програмування високого рівня. Паралельні бази даних були засновані переважно на реляційній моделі даних. Декларативний SQL-був використаний в якості мови запитів для вираження завдань обробки даних, що зберігаються у вигляді записів таблиці.

Паралельні бази даних досягли високої продуктивності і масштабованості секціонування таблиць по вузлах в "shared-nothing cluster" (розподілена обчислювальна архітектура, в якій кожен вузол є незалежним і самодостатнім, і немає жодної точки неузгодження у всій системі. Зокрема, жоден з вузлів не розділяє пам'ять або дисковий простір). Така горизонтальна схема розбиття включає реляційні операції, такі як фільтри з'єднання і агрегатування для запуску паралельно з різних розділів якоїсь таблиці збереженої на різних вузлах.

Три тенденції почали ставати помітними на початку 2000-х років, що викликали питання про перевагу класичних паралельних баз даних: все більше компаній почали зберігати стільки інформації, скільки могли зібрати. Класичні паралельні бази даних почали створювати серйозні перешкоди в плані масштабованості і сукупної вартості володіння, як і виникнення необхідності обробки цих постійно зростаючих обсягів даних; дані що збираються і зберігаються в компанії були різноманітні по своїй структурі. Наприклад, стало звичайною практикою для збору добре структурованих даних, таких як дані про продажі і користувацьку демографію поряд з менш структурованими даними, таких як журнали пошукових запитів і веб-вміст сторінок. Було важко підігнати такі різноманітні дані до жорсткої моделі даних, підтримуваної класичними паралельними базами даних; потреби бізнесу почали вимагати більш короткі інтервали між часом, коли будуть зібрані дані (як правило, в OLTP – онлайнова обробка транзакцій) і час, коли результати аналізу даних були б доступні для ручного або алгоритмічного прийняття рішень.

Колонкові системи були піонерами нової концепції зберігання даних у таблицях, розташовуючи всі стовпці разом, замість розташування по рядкам, як це зроблено в класичних паралельних базах даних. Системи з колонкоподібним зберіганням та обробкою даних, такі як Vertica, показали використання процесора, пам'яті і ресурсів введення/виведення більш ефективно у великомасштабному аналізі даних в порівнянні з рядково-орієнтованими системами. Деякі з основних переваг отримуються від зменшення операцій введення/виводу в колонкових системах, читаючи тільки необхідні стовпці при обробці запиту.

MapReduce – це програмна модель та програмний каркас, що її реалізує, розроблені компанією Google для обробки величезних масивів даних шляхом використання дуже великого скупчення кластерів зі звичайних вузлів. Системи в класичній категорії традиційно намагалися масштабуватись до таких рівнів. MapReduce системи стали піонером концепції множино автономної побудови масштабованих розподілених систем, і можливості компонувати дві або більше з цих систем разом для того, щоб запустити аналітичні завдання по вибірці з великих наборів даних. Популярні системи у цій категорії, такі як Hadoop, зберігають дані в автономну блок-орієнтовану розподілену файлову-систему і запускають обчислювальні завдання в іншій розподіленій система, яка підтримує програмну модель MapReduce.

Деякі недоліки в MapReduce були визначені тим, як ці системи використовувались для завдань аналізу великого числа даних. Програмна модель MapReduce є занадто обмеженою щоб легко виразити певні завдання аналізу даних, наприклад, з'єднання двох наборів даних разом. Що ще більш важливо, методика виконання яка використовується в MapReduce системах неоптимальна для багатьох поширених типів завдань аналізу даних, таких як реляційних операцій, ітераційного машинного навчання та опрацювання графа. Більшість з цих проблем можуть бути вирішені шляхом заміни MapReduce на більш гнучку модель виконання на основі потоків даних, яка може виражати широкий спектр доступу до даних і комунікаційних моделей. Різні моделі виконання, основані на потоках даних, були використані в системах цієї категорії, в тому числі спрямовані ациклічні графи в «Dryad», дерева обслуговування в «Dremel», і масова синхронно паралельна обробка в «Pregel».

Стало зрозуміло, що з плином часу нові системи можуть бути побудовані шляхом об'єднання принципів проектування з різних категорій систем. Наприклад, методи, використовувані для високоефективної обробки в класичних паралельних базах даних можуть бути використані разом з методами що використовуються для дрібномодульної відмовостійкості в MapReduce системах. Кожна система в цій об’єднаній категорії надає єдиний інтерфейс системи, що забезпечує загальний набір функцій, які традиційно асоціюються з різними категоріями системи.

Необхідність скоротити розрив між генерацією даних і генерацією результатів аналітики цих даних потребувало системних розробників постійно підвищувати планку в великомасштабних аналізах даних. З одного

боку, ця потреба спонукала появу масштабованих розподілених систем зберігання, які забезпечують різні ступені транзакційних можливостей. Підтримка для транзакцій дозволяє цим системам служити, як сервіс сховища даних онлайн, а ці дані робить доступними одночасно в тих самих системах для аналітики. Ця потреба призвела до появи паралельних систем управління базами даних, які підтримують як OLTP і OLAP в одній системи. Ми ставимо обидва типи систем в категорію названу змішаними системами через їх здатності виконувати ефективно змішані робочі навантаження, такі які містять транзакційні завдання, а також завдання аналітики.

Можна виділити вісім ключових особливостей систем, які описують основні характеристики систем для аналітики великомасштабних даних:

1. Модель даних і інтерфейси. Модель даних забезпечує чіткість і логічну структуру даних, і визначає в якому форматі дані можуть бути збережені, організовані і використовуватись системою. Найбільш популярний приклад моделі даних є реляційна модель (яка використовує формат таблиці в своїй основі), в той час як більшість систем в MapReduce і потоках даних дозволяють щоб дані бути в будь-якій довільній формі, що зберігається в одноманітних файлах. Модель даних, використовувана кожною з систем тісно пов'язана з інтерфейсом запиту, доступного в системі, що дозволяє користувачам управляти і маніпулювати збереженими даними.

2. Storage Layer (рівні збереження). На високому рівні, Storage Layer просто відповідає за збереження даних, а також надання методів для доступу й зміни даних. Тим не менш, розробка, впровадження та особливості умов використання рівнів збереження, в різних категоріях систем сильно розрізняються, особливо коли ми починаємо порівняння систем з різних категорій. Наприклад, класичні паралельні бази даних використовують інтегровані і спеціалізовані сховища даних, які тісно взаємопов'язані з їх механізмом виконання, в той час як системи MapReduce зазвичай використовують незалежні розподілені файлові системи для доступу до даних.

3. Механізм виконання. Коли система отримує запит на виконання, вона, як правило, перетворює його в схему виконання для доступу й обробки вхідних даних запиту. Механізм виконання несе відповідальність за те як фактично працює дана схема виконання в системі і генеруються результати запиту. У системах, які ми обговорюємо, виконавчий механізм також несе відповідальність за розпаралелювання обчислень по великомасштабних кластерах машин, обробки відмов машини, і створення міжмашинного зв'язку для ефективного використання дискової пропускної здатності та мережі.

4. Оптимізація запитів. В цілому, оптимізація запитів процес який система використовує для визначення найбільш ефективнішого способу виконання даного запиту з урахуванням кількох альтернативних еквівалентних, схем виконання. Методи, використовувані для оптимізації запитів в системах, ми розглядаємо з різних точок зору: (I) простір можливих схем виконання (наприклад, реляційні оператори в базах даних у порівнянні з настройками параметрів конфігурації в MapReduce систем), (II) тип

оптимізації запитів (наприклад, на основі витрат у порівнянні з основаних на правилах), (III) типу технік моделювання витрат (наприклад, аналітичні моделі в порівнянні з моделями навченими за допомогою методів машинного навчання) і (IV) зрілість методів оптимізації (наприклад, повністю автоматизована проти ручного налаштування).

5. Планування. Враховуючи розподілений характер більшості систем аналізу даних, планування виконання схеми запиту є важливою частиною системи. В даний час системи повинні приймати кілька рішень планування, в тому числі планування, де запустити кожне обчислення, планування передачі даних між вузлами, а також планування поновлення виконання та роботи з технічного обслуговування.

6. Управління ресурсами. Управління ресурсами в першу чергу відноситься до ефективного і результативного використання ресурсів кластера на основі вимог до ресурсів запитів додатків, запущених в системі. Крім того, багато систем сьогодні пропонують гнучкі властивості, які дозволяють користувачам динамічно додавати або видаляти ресурси, як необхідно відповідно до вимог робочого навантаження.

7. Відмовостійкість. Апаратні збої є досить поширеним явищем у великих кластерах. Таким чином, більшість систем мають вбудовані функціональні можливості відмовостійкості, що дозволить їм продовжувати надання послуг, можливо, з поступовим погіршенням, на фоні небажаних подій, таких як апаратних збоїв, помилки в програмному забезпеченні, а також псування даних. Приклади типових особливостей допустимих збоїв, включають перезапуск невдалих завдань, як через прикладні або апаратні збої, відновлення даних в результаті збою машини або псування, і використання спекулятивного виконання, щоб уникнути відсталості.

8. Системне адміністрування. Системне адміністрування відноситься до діяльності де додаткові зусилля людини можуть бути необхідні для підтримки системи працюючою, система обслуговує потреби різних користувачів і додатків. Загальні заходи по системному адмініструванню включають моніторинг та налаштування, діагностика причин поганого виконання або невдачі, планування потужностей, і відновлення системи при безповоротних збоях (наприклад, відмови диска) або стихійного лиха.

Основна маса проблем в аналізі даних сьогодні виявляється з чистого обсягу доступних даних для обробки. Обсяги даних, які багато компаній хочуть обробити вчасно і економічно ефективними способами, постійно росли від діапазону мультігігабайта до терабайт і тепер до багатьох петабайт. Цю проблему контакту з дуже великими наборами даних назвали проблемою обсягу. Існує дві інших пов'язаних проблеми, а саме, проблеми зі швидкістю обробки та різноманітністю даних.

До проблеми швидкості відноситься вимоги до часу відгуку для збору, зберігання і обробки даних. Більшість систем, є пакетними системами. Для чутливих до затримок додатків, таких як ідентифікація потенційного шахрайства та рекомендація персоналізованого контенту, пакетної обробки

даних недостатньо. Дані, повинні бути оброблені, під час передачі їх потоком в систему для вилучення максимальної користі з даних. Наразі існує потреба в збільшення швидкості отримання результатів запиту.

Проблема різноманітності відображається на зростаючому списку типів даних - реляційного, часового ряду, тексту, графіків, аудіо, відео, зображень, генетичних кодів, а також зростаючого списку аналітичних методів до таких даних. Нові ідеї знаходяться при аналізі більше ніж одного з цих типів даних разом. Методи зберігання і обробки переважно націлені на обробку даних, які можуть бути представлені за допомогою реляційної моделі (рядки і стовпці) і оброблені операторами типу запитів як фільтри, з'єднання і агрегації. Однак нові типи даних, що з'являються, не можуть бути легко отримані в реляційній моделі даних або легко проаналізовані програмним забезпеченням, що залежать від роботи операторів як фільтрів, з'єднання і агрегації. Замість цього для нових типів даних потрібно безліч аналітичних методів, таких як лінійна алгебра, статистичне машинне навчання, текстовий пошук, обробка сигналів, обробка природної мови та ітеративна обробка графіків.

Ці проблеми формують нові тенденції досліджень в обробці даних з масовим паралелізмом.

Потреба зменшити розрив між генерацією даних і генерацією результатів аналітики за цими даними привела до систем, які можуть підтримувати і OLTP і робочі навантаження OLAP в єдиній системі. Масштабовані розподілені системи зберігання, що забезпечують різні ступені транзакційних можливостей, вже розробляються. Підтримка транзакцій дозволяє цим системам служити сховищем даних для онлайнових служб при створенні доступних даних одночасно в тій же системі для аналітики. Найбільш видатним прикладом тут є система Google Bigtable, яка є розподіленою, що має версію, яка орієнтована на колонкову систему, що зберігає багатовимірні і сортовані набори даних. Bigtable забезпечує атомарність на рівні окремих кортежів.

Традиційно, паралельні бази даних використовували різні системи для підтримки OLTP і OLAP. Робочі навантаження OLTP характеризуються з'єднанням запитів на читання і записів до кількох кортежів за один раз, зазвичай через індексні структури як B-дерева. Робочі навантаження OLAP характеризуються масовими оновленнями і великими послідовними скануваннями однак тільки для читання декількох стовпців за один раз. Однак більш нові робочі навантаження баз даних все більш і більш є поєднанням традиційного OLTP і робочих навантажень OLAP. Наприклад, “available-to-promise” (доступні від наявності) додатки, вимагають запитів стилю OLTP при агрегації рівнів запасів в режимі реального часу за допомогою запитів стилю OLAP, щоб визначити, чи може замовлення бути виконано.

Системи такі як HYRISE, HyPer і SAP HANA мають за мету підтримувати OLTP і OLAP в єдиній системі. Одна з проблем для цих систем те, що формати даних, які гарно підходять для OLTP, можуть не підходити

для OLAP, і навпаки. Наприклад, таблиці розділів HYRISE у вертикальних групах різної ширини залежать від того, як до стовпців таблиць отримують доступ. Менші групи стовпців віддають перевагу для доступу до даних за стилем OLAP, в той час як більш широкі групи стовпців віддають перевагу для доступу до даних за схемою OLTP (для скорочення невдалих звернень в кеш при виконанні єдиних витягів рядка). Будучи в системній пам'яті, HYRISE ідентифікує кращий стовпець, який групуються на основі докладної моделі вартості продуктивності кеша в змішаних налаштуваннях OLAP / OLTP.

Проведений огляд принципів побудови та особливостей систем аналізу надвеликих об'ємів даних. Описані деякі з основних технологічних інновацій, кожна з яких породила окрему категорію систем для аналізу даних. Зроблена спроба класифікувати ряд аспектів по яких можна оцінити та порівняти кожну з категорій систем для великомасштабного аналізу даних.

Виявлений ряд проблем, які породжені сучасними вимогами до аналізу даних та стрімким збільшенням об'ємів самих даних.

Література

1. Круг П.Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу «Микропроцессоры». /П.Г.Круг//– М.: Издательство МЭИ, 2002.–176 с.
2. Логовский А. Технология ПЛИС и ее применение для создания нейрочипов / А. Логовский //«Открытые системы», № 10, 2000.

Види пасивних оптичних мереж - технологія PON

Заморока О.І.
Науковий керівник – к.т.н.,доц. Красильников С.Р.
Хмельницький національний університет

В даний час все більшого поширення поряд з мережами прокладеними за допомогою звитої пари отримують оптичні мережі. Це й не дивно, оптичні мережі дозволяють передавати набагато більші обсяги інформації за короткий час, а значить можна збільшити якість звуку чи відео переданих по мережі. PON - це абревіатура від passive optical network, що означає пасивні оптичні мережі. При використанні технології PON мережу організують оптичним кабелем, починаючи від комп'ютера.

Мережа PON заснована на використанні всього одного приймально-передавального пристрою, від якого інформація надходить до абонентських пристроїв і приймається від них. Кількість абонентських пристроїв залежить від потужності і максимальної швидкості приймально-передавального пристрою. Для побудови PON використовується топологія «точка - багатоточка» і сама мережа має деревоподібну структуру. Кожен волоконно-оптичний сегмент підключається до одного приемопередатчик в центральному вузлі, на відміну від топології «точка - точка», що також дає значну економію у вартості обладнання. Один волоконно-оптичний сегмент

мережі PON може охоплювати до 32 абонентських вузлів у радіусі до 20 км для технологій EPON / BPON і до 128 вузлів в радіусі до 60 км для технології GPON. Кожен абонентський вузол розрахований на звичайний житловий будинок або офісний будинок і в свою чергу може охоплювати сотні абонентів. Всі абонентські вузли є термінальними, і відключення або вихід з ладу одного або декількох абонентських вузлів ніяк не впливає на роботу інших.

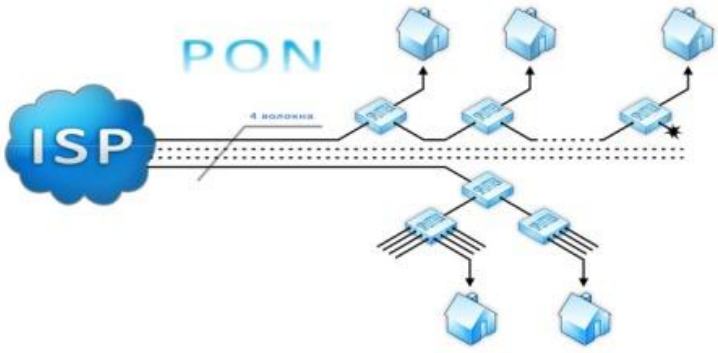


Рисунок 1 – Архітектура PON мережі

Таблиця 1 - Порівняльний аналіз технологій APON/BPON, EPON, GPON

	APON	BPON	EPON (GEPON)	GPON
Стандарт	G.983	ITU G.983	IEEE 802.3ah	ITU G.984.6
Смуга пропускання для низхідного потоку	155 Мбіт / с	622 Мбіт / с	1,244 Гбіт / с	2,488 Гбіт / с
Смуга пропускання для висхідного потоку	155 Мбіт / с	155 Мбіт / с	1,244 Гбіт / с	1,244 Гбіт / с
Ємність		32	32	64
Максимальна довжина передачі, км		20	20	60
Загасання лінії PON			26 дБ	22 дБ

Основні переваги технології PON:

- економія волокон (до 128 абонентів на одне волокно, протяжність мережі до 60 км);
- ефективне використання смуги пропускання оптичного волокна;
- швидкість до 2,488 Гбіт/с по низхідному потоку і 1,244 Гбіт/с по висхідному потоку;
- надійність (у проміжних вузлах дерева знаходяться тільки пасивні оптичні розгалужувачі, не потребують обслуговування);

- масштабованість (деревоподібна структура мережі доступу дає можливість підключати нових абонентів найбільш економічним способом);
- можливість резервування як всіх, так і окремих абонентів;
- гнучкість (використання АТМ, як транспорту дозволяє надавати абонентам саме той рівень сервісу, який їм потрібен).

При тестуванні мережі PON є важливими наступні питання:
- реальне загасання в оптичній лінії між центральним вузлом і абонентським пристроєм (діючого або того, що готується до підключення).
- місцезнаходження проблемної ділянки, якщо реальне загасання в лінії виявилося вище очікуваного (розрахункового або опорного).

Література

1. Андрушко Л.М., В ознесенкий В.А., Каток В.Б. и др.Справочник по волоконно¬оптическим линиям связи / Під ред. Свечникова С.М. и Андрушко Л.М. - К.: Техніка, 1988. - 239 с.
2. Бейли Д., Райт Э. В олоконная оптика: теория и практика. - М.: ОБРАЗ, 2006. - 320 с.
3. Берлин Б.З. и др.Волоконно-оптические системы связи на ГТС: Справочник - М.: Радиосвязь, 1994. - 160 с.

Технології конфігурування ПЛІС

Карпович Д.В.
Науковий керівник - к.т.н. Гурман І.В.
Хмельницький національний університет

ПЛІС кожної фірми вимагають застосування своїх програмних пакетів, ступінь доступності яких також різна. Повні версії програмних продуктів усіх без винятку фірм є комерційними продуктами з вартістю від декількох сотень до декількох тисяч у.е. Однак деякі фірми надають безкоштовні версії своїх програмних продуктів з деякими обмеженнями можливостей.

Програмні засоби WebPACK ISE являють собою систему наскрізного проектування, що реалізує всі етапи створення цифрового пристрою на базі ПЛІС, включаючи програмування кристала: розробка проекту, синтез, моделювання, трасування і завантаження в кристал. САПР WebPACK ISE призначена для проектування цифрових пристроїв на базі ПЛІС виробництва Xilinx, що відносяться як сімействам CPLD: XC9500, XC9500XL, XC9500XV, XCR22V10, XCR3000 (XPLA1_3, XPLA2), XCR3000XL (XPLA3), XCR5000 (XPLA1_5), так і FPGA: Spartan™-II, Virtex™-E (тільки кристал XCV300E), Virtex-II (кристали 2V40, 2V80 і 2V250).

Можливості цього пакета:
- підтримка різних методів опису проектованих пристроїв (графічних і текстових);
- можливість використання проектів, підготовлених в інших системах проектування, у тому числі в середовищі пакета Altera MAX+PlusII™;

- наявність схемотехнічного редактора, укомплектованого набором великих бібліотек;

- інтелектуальні засоби створення HDL (Hardware Description Language) – описів, що формують шаблони на підставі інформації, наданої користувачем, для мов опису апаратури VHDL, Verilog™ і ABEL™ HDL;

- високоефективні засоби синтезу HDL-проектів, що підтримують мови VHDL, Verilog і ABEL HDL, з можливістю оптимізації;

- автоматичні засоби трасування проекту в кристали різних сімейств ПЛІС Xilinx з урахуванням оптимізації проекту по різних параметрах;

- засоби програмування кристалів сімейств ПЛІС Xilinx, виконаних за різною технологією (CPLD і FPGA), що підтримують кілька типів завантажувальних кабелів JTAG-інтерфейсу;

- наявність інтегрованого з пакетом САПР набору інструментів і утиліт інших фірм, що надають додаткові зручності в процесі проектування, що включає утиліту генерації тестових сигналів HDL Bencher™, програму моделювання ModelSim XE Starter™ і редактор діаграм станів StateCAD™.

- Для одержання програмного забезпечення WebPACK ISE необхідно зареєструватися на web-сайті www.xilinx.com. Після реєстрації варто виконати процедуру копіювання модулів пакета на ПК, що буде використаний для розгортання САПР. Дистрибутив пакета виконаний у виді набору модулів, кожний з яких являє собою архів, що саморозпаковується. Для цього необхідно запустити програму Licensing Wizard, що збере необхідну для одержання ліцензійного коду інформацію про використовуваний ПК. Файл із цією інформацією повинний бути відправлений по електронній пошті.

Пакет Quartus II, при більш високій продуктивності, не вимагає таких довгих настроювань для запуску.

Для розробки цифрових пристроїв на ПЛІС фірми «Altera» зараз використовується інтегроване середовище розробки цифрових пристроїв, комерційна версія якої називається Quartus II.

Типи ПЛІС, підтримувані безкоштовними версіями пакета Quartus 2 можна подивитися в таблиці 1. Програма має можливості змінити набрану інформацію і підбудуватися практично під кожну з даних ПЛІС.

Таблиця 1 – ПЛІС, підтримуваних середовищем програмування Quartus

Сімейство	Підтримувані ПЛІС
Max II, MAX 3000 A, MAX 7000B(S), Arria GX, Cyclone I–IV, FLEX 6000, ACEX	Усі
STRATIX III	EP3SE50, EP3SL70
STRATIX II	EP2S15
APEX II	EP2A15
APEX 20KE	EP20K30E, EP20K60E, EP20K100E, EP20K200E
FLEX 10 KE	EPF10K50S, EPF10K200S

По кількості доступних ПЛІС Quartus значно випереджає інші системи програмування ПЛІС, у той час, як по іншим не менш чим відповідає їх рівневі.

Номенклатура мікросхем, підтримуваних безкоштовними версіями системи Quartus II, охоплює практично всю лінійку ПЛІС.

Усі пакети інтегрованого середовища розробки цифрових пристроїв на ПЛІС фірми «Altera» мають наступні загальносистемні властивості:

1.Забезпечують повний виробничий цикл випуску готових цифрових пристроїв на ПЛІС, що включає:

- розробку проекту пристрою (завдання необхідної логіки функціонування пристрою);
- перевірку коректності проекту і локалізацію помилок;
- синтез внутрішньої структури пристрою з мінімізацією необхідних ресурсів;
- компіляцію проекту (створення файлу для програмування або конфігурування ПЛІС);
- моделювання процесу функціонування пристрою, часовий аналіз і, нарешті, програмування (конфігурування) ПЛІС.

2.Мають в користуванні засоби розробки проектів, до складу яких входять:

- редактор схем (Graphic Editor);
- редактор часових діаграм (Waveform Editor);
- текстовий редактор проектів мовою AHDL (Text Editor);
- (Усі редактори можуть використовуватися для створення різних частин основного проекту)

3.Мають велику бібліотеку елементів різного виду.

4.Дозволяють розробляти проекти у виді багаторівневої ієрархії вкладених функціональних модулів і створювати за допомогою різних редакторів власні бібліотеки модулів, що можуть використовуватися в різних проектах.

5.Забезпечують оптимальний синтез і мінімізацію використовуваних для реалізації проекту ресурсів мікросхем.

6.Забезпечують перевірку і локалізацію помилок при створенні вихідного проекту і при компіляції з обліком формальних і емпіричних правил проектування цифрових пристроїв і достатності наявних ресурсів, гарантуючи працездатність успішно скомпільованого проекту (але не гарантують від помилок при завданні розроблювачем алгоритму функціонування пристрою).

7.Мають можливість автоматичного вибору найбільш підходящої мікросхеми необхідного обсягу або розподілу проекту між декількома мікросхемами малого обсягу.

8.Забезпечують можливість закріплення призначених компілятором висновків мікросхем для постійної прив'язки до зовнішніх компонентів цільового пристрою або перепризначення висновків.

9.Мають убудовані засоби функціонального і тимчасового

моделювання, що забезпечують швидку верифікацію і налагодження проектів.

10. Забезпечують програмування і перепрограмування мікросхем, що мають убудовану систему програмування, безпосередньо в складі кінцевого виробу через спеціальний кабель, що підключається або до LPT-порту (Byte Blaster), або до COM-порту (Bit Blaster) комп'ютера і технологічного 10-контактного з'єднувача, встановлюваного на платі виробу. Для програмування інших мікросхем необхідно додатково використовувати зовнішній програматор, що також може підключатися до COM- або LPT-порту.

Усі пакети мають однаковий, стандартний для Windows-додатків інтерфейс користувача.

Література

1. Рябенький В.М., Ушкаренко О.О. MAX+PLUS II. Основи проектування цифрових пристроїв на ПЛІС. – К.: « Корнійчук», 2004. – 253 с.

Проблеми синхронізації в структурах баз даних

Кизима К.А.
Науковий керівник – к.т.н., доц. Чорненький В.І.
Хмельницький національний університет

Об’єм накопиченої в усьому світі інформації представляється числом фантастичної величини. Кількість інформації, виробленої світовою спільнотою, подвоюється кожні чотири роки, при цьому темпи її отримання непреривно зростають. Інформація накопичується в базах даних та інформаційних сховищах різних типів. У той же час тільки її незначна частка використовується для вирішення прикладних і наукових проблем, що може бути пояснено декількома факторами. Процес отримання інформації пов'язаний з предметними областями, для опису яких не існує як формалізованого, так і загальновизнаного неформалізованого підходу. Прийнято вважати, що формалізація поняття "предметна область" неможлива в силу того, що воно є первинним. Не існує навіть загальновизнаних його описів. Така ситуація обумовлює безліч невизначеностей, які виникають при проектуванні баз даних, накопиченні в них інформації, отриманні нових знань з накопиченої інформації, побудові процедур прийняття рішень в системах управління і методів їх інформаційної підтримки.

Розробка ефективних процедур прийняття рішень і систем їх інформаційної підтримки в значній мірі залежить від ступеня відповідності структури баз даних предметним областям. Бази даних повинні відображати процес інформаційного обміну між предметними областями та зовнішнім середовищем, а також між їхніми об’єктами.

Неадекватне представлення інформаційного обміну обумовлює різноманіття невизначеності при побудові процедур прийняття рішень.

Невизначеності побудови баз даних і процедур прийняття рішень призводять до некоректної інформаційної підтримки процесів управління предметними областями. Перераховані факти обумовлюють необхідність створення методів побудови процедур контролю за процесами накопичення інформації в базах даних з послідовним зменшенням або усуненням всіх форм неповноти та невизначеностей, що виникають як на етапі проектування структури баз даних, так і в процесі їх формування. Необхідна розробка математичних методів перевірки адекватності структури баз даних предметним областям як їх інформаційної моделі, коректування структур опису баз даних і інструментальних засобів їх наповнення вибірковими даними.

Відстеження змін в базах даних СУБД вимагає, як правило, значних адміністративних зусиль, незалежно від того, чи це початківець розробник БД або DBA-експерт. Представимо ситуацію, коли зміни, внесені в локальну базу даних PostgreSQL, повинні бути в точності продубльовані в БД віддаленого сервера. Пошук відмінностей між останньою і попередньою версією структури бази даних, в даному випадку, є досить актуальним завданням. Великі розподілені конфігурації часто містять численні виробничі копії однотипних баз даних PostgreSQL, які необхідні для підтримки діяльності організації, однак наявність множини копій може значно збільшити час, необхідний для синхронізації інформації, і зажадати залучення додаткових коштів для здійснення контролю за змінами структури БД.

Дуже часто розробка додатків для баз даних PostgreSQL є географічно розподіленим процесом. Команди розробників можуть перебувати в різних організаційних підрозділах, розташованих не тільки в різних частинах країни, а й на різних континентах, при цьому, щоб уникнути втрати даних, важлива інформація періодично реплікується на кілька PostgreSQL серверів. Отже, на кінець або проміжному етапах розробки, дуже важливо синхронізувати існуючі версії баз даних для остаточної фіксації внесених змін, які відносяться до даних або схеми БД. Спеціальне програмне забезпечення для синхронізації баз даних, призначене для PostgreSQL, здатне значно спростити процес порівняння і, якщо потрібно, клонування БД шляхом створення та виконання SQL сценаріїв, необхідних для переміщення бази даних в актуальний стан. Зазвичай процедура клонування бази даних PostgreSQL складається з двох фаз: синхронізація схеми і синхронізація даних, тому розглянемо програмне забезпечення для кожного з цих етапів синхронізації окремо.

Синхронізація схеми БД PostgreSQL EMS DB Comparer for PostgreSQL - це потужна програма для порівняння та синхронізації баз даних. Утиліта допоможе знайти існуючі відмінності в порівнюваних об'єктах схем обраних баз даних PostgreSQL і виконати автоматично створений сценарій, необхідний для усунення всіх або вказаних користувачем невідповідностей в схемах. Процес порівняння схеми, розділений на кілька окремих етапів, а також зручний інтерфейс користувача, дозволяє синхронізувати бази даних

PostgreSQL, всього за кілька клацань миші. Програма наочно відображає невідповідності між схемами вихідної і цільової баз даних і надає можливість синхронізації БД вручну, крок за кроком, або автоматично. Існує можливість генерувати звіти, що містять інформацію про знайдені відмінності, порівнювати і синхронізувати схеми баз даних, які розташовані на різних серверах або ж на одному і тому ж сервері і багато іншого. Дає можливість керувати будь-яким з етапів процесу синхронізації, визначивши необхідні параметри і вибравши об'єкти схеми, які повинні бути клоновані. Утиліта підтримує останні версії СУБД PostgreSQL, має потужний вбудований редактор скриптів SQL з підсвіткою синтаксису і безліч інших корисних функцій. Таким чином, EMS DB Comparer for PostgreSQL дозволяє адміністраторам і розробникам автоматизувати і спростити складні, звичайно що виконуються вручну і вимагають значних витрат часу, процеси, пов'язані з синхронізацією схем баз даних PostgreSQL.

Література

1. Вентцель Е.С. Исследование операций. Задачи, принципы, методология. Учебное пособие для студентов втузов. – 2-е изд., стер./ Е.С. Вентцель – М.: Высшая школа, 2001. – 208 с.
2. Гусева Т. И. Проектирование баз данных в примерах и задачах / Т. И. Гусева, Ю. Б Башин. – М. : Радио и связь, 1992. – 480 с.
3. Ивахненко А.Г. Долгосрочное прогнозирование и управление сложными системами./ А.Г. Ивахненко – К.: Техника, 1978. – 356 .
4. Корнеев В.В. Базы данных. Интеллектуальная обработка информации./ В.В. Корнеев, А.Ф. Гареев, С.В. Васютин, В.В. Райх — М.: "Нолидж", 2000. - 352 с.

Підходи до протидії прихованим загрозам у середовищі хмарних обчислень

Козак І.В.
Науковий керівник – к.т.н.,доц. Огнєвий О.В.
Хмельницький національний університет

В результаті проведеного аналізу [1] доводиться актуальність розробки технології протидії спробам змінити стан захищеності інформаційних ресурсів у середовищі хмарних обчислень. В результаті досліджень, проведених авторами, зроблено висновок про те, що перспективним напрямком вдосконалення технологій кіберзахисту в середовищі хмарних обчислень є розробка методів протидії загрозам, які реалізуються з допомогою процесів, в яких суб'єкти і об'єкти інформаційної взаємодії можуть використовувати різні канали передачі даних, у тому числі і приховані. Показано, як реалізація прихованих загроз дозволяє шкідливому коду маскуватися під системний процес, завдаючи шкоди безпеці середовища хмарних обчислень за допомогою блокування, розкрадання, знищення або

несанкціонованої передачі інформації.

Доктрина інформаційної безпеки визначає поняття інформаційної сфери як сукупність інформації, інформаційної інфраструктури, суб'єктів, що здійснюють збір, формування, розповсюдження та користування інформації, а також системи регулювання виникаючих при цьому громадських відносин. Інформаційна безпека в широкому розумінні являє собою такий стан об'єкта захисту, який виключає можливість нанесення шкоди властивостями об'єкта, обумовлена його взаємодією з інформаційної сферою [2]. Загроза безпеки визначається як сукупність умов і факторів, що створюють потенційну або реально існуючу небезпеку, пов'язану з витоком інформації та/або несанкціонованими та/або ненавмисними діями на неї.

Принцип «невидимості» заснований на тому, що існують недокументовані стани в системі, які ніяк не помітні для монітора безпеки і які дозволяють шкідливому коду маскуватися під штатний процес.

Під визначенням «руткіти» розуміється набір утиліт або спеціальний модуль ядра, які зловмисник використовує для прихованого вбудовування в операційні системи користувачів шкідливого програмного забезпечення.

Під шкідливістю розуміється здатність програм завдати шкоди комп'ютерній системі за допомогою блокування, розкрадання, знищення і несанкціонованої передачі інформації.

Особливу увагу приділено аналізу недоліків сучасних технологій захисту інформації в середовищі хмарних обчислень, які не враховують динамічний характер наданих прикладних і системних програмних сервісів. Показано, як хмарні системи класу «інфраструктура як сервіс» можуть стати джерелом загрози порушення безпеки програмного забезпечення, що пов'язано з активним характером взаємодії суб'єктів та об'єктів доступу до інформаційних ресурсів, що призводить до ризиків порушення цілісності та доступності програмних сервісів, що надаються в режимі віддаленого доступу. Відзначено, що особливу небезпеку надають загрози, які реалізуються всередині периметра безпеки комп'ютерної мережі, так як їх локалізація із застосуванням сучасних засобів захисту інформації, наприклад, антивірусів та сканерів безпеки, зустрічає істотні труднощі.

Середовище хмарних обчислень - це сукупність обчислювальних ресурсів у вигляді віртуальних машин, наданих користувачу за допомогою загальних сервісів доступу. Фізичний рівень хмарної системи складається з апаратних ресурсів, які необхідні для забезпечення сервісів, що надаються, і, як правило, включає сервери, системи зберігання і мережеві компоненти [3]. Розглянуті хмарні системи відносяться до типу «інфраструктура як сервіс», і для них характерно наявність гіпервізора для управління обчислювальними ресурсами, який розглядається як додаткове джерело вразливостей, список яких з кожним роком збільшується. Застосування технологій хмарних обчислень визначає необхідність розгляду можливих способів дестабілізуючих впливів, що призводять до порушення функціонування компонентів інформаційного середовища.

Характерною особливістю сучасного середовища хмарних обчислень

є активний характер суб'єктів та об'єктів інформаційної взаємодії. Це дозволяє розглядати цільову функцію системи безпеки як збереження конфіденційності, цілісності і доступності програмних та інфраструктурних сервісів, що надаються в режимі віддаленого доступу в умовах динамічного зміни стану обчислювальних ресурсів. В сучасних антивірусах, обманних системах захисту і сканерах безпеки не враховуються загрози, які реалізуються всередині периметра безпеки, Розробники програмно-технічних засобів захисту керуються власними уявленнями про створення прототипу продукту, використовуючи традиційні шаблони реалізації механізмів безпеки, саме тому найчастіше представлені на ринку засобів захисту інформації володіють безліччю загальновідомих вразливостей навіть в умовах застосування новітніх технологій. Побудова перспективних механізмів забезпечення безпеки в середовищі хмарних обчислень пов'язується не з захистом від виявлених вразливостей, а полягає в можливості запобігання нових невідомих методів проведення атак, в розробці нових моделей загроз і методів запобігання або відображення комп'ютерних атак на інформаційні ресурси, які використовують можливості предикативній ідентифікації прихованих каналів і потенційно небезпечних процесів інформаційної взаємодії.

В умовах розвитку ринкової економіки фахівцями в різних країнах все більше уваги приділяється питанням розробки засобів захисту, що дозволяють протидіяти загрозам інформаційної безпеки з боку зловмисників, на основі єдиного концептуального підходу, що поєднує в собі переваги різних методів захисту інформації. Розвиток засобів, методів і форм автоматизації процесів обробки інформації і масове застосування персональних комп'ютерів, що обслуговуються непідготовленими в спеціальному відношенні користувачами, роблять інформаційний процес вразливим по ряду показників .

Причини, що зумовлюють виникнення вразливостей в середовищі хмарних обчислень, наступні:

- обсяг оброблюваної інформації постійно збільшується з урахуванням розширення інформаційного простору мереж загального і спеціального призначення;
- у сучасних обчислювальних комплексах використовуються програмно-технічні засоби, різних по своїй архітектурі, функціональним можливостям та цільового призначення;
- доступ до ресурсів обчислювальних комплексів одержує все більше число користувачів, операторів у зв'язку з застосуванням Internet-технологій;
- за рахунок використання нових, не пройшли тривалу апробацію в різних соціальних структурах технологій збільшується ймовірність виникнення нових класів вразливостей;
- низький рівень комп'ютерної грамотності користувачів, недостатня кваліфікація системних адміністраторів;
- використання передачі інформації з використанням Wi-Fi мереж безпровідного доступу, що значно спрощує зловмисника процес

несанкціонованого знімання інформації, поширюваної за межі контрольованої зони.

Для середовища хмарних обчислень однією з головних проблем управління є нерівномірність запиту ресурсів з боку клієнтів. Для згладжування нерівномірності надання сервісів, т.д. розподілу ресурсів між реальним апаратним забезпеченням і хмарним програмним забезпеченням, використовують проміжний шар серверної віртуалізації.

Захист самих віртуальних машин. На відміну від фізичної машини, коли віртуальна машина вимкнена, залишається можливість її компрометації або «зараження». Досить доступу до сховища образів віртуальних машин через мережу. Тим же, хто використовує сервіси хмарних обчислень, слід переконатися в тому, що провайдер має СЗІ, які встановлені на серверах віртуалізації. Одночасно проводиться оцінка небезпеки реалізації загрози, що визначається трьома значеннями:

- низька небезпека – якщо реалізація може призвести до загрози незначним негативних наслідків для суб'єктів персональних даних;
- середня небезпека – якщо реалізація може призвести до загрози негативних наслідків для суб'єктів персональних даних;
- висока небезпека – якщо реалізація може призвести до загрози значних негативних наслідків для суб'єктів персональних даних.

Засоби захисту середовища хмарних обчислень розподілені на різних рівнях ієрархії обчислювальної середовища. В якості базового рівня ієрархії виділимо ядро операційної системи сервера віртуалізації.

Вивчати поведінку компонентів гіпервізора при генерації подій в середовищі хмарних обчислень можна двома способами: перший спосіб полягає в аналізі вихідного коду, що украй скрутно, оскільки більшість розробленого ПЗ для гіпервізора є пропрієтарним. Другий спосіб є більш кращим і полягає у підрахунку кількості подій, при якому стан компонентів гіпервізора змінюється, при коректно завершується виконання операцій на кожному рівні ієрархії.

Для боротьби з такими загрозами актуальною є розробка нових засобів захисту інформації, заснованих на методах оперативної ідентифікації потенційних вразливостей, що виникають як на рівні процесів контролю доступу до ресурсів гостьових ОС, так і на рівні системних викликів гіпервізора, які за певних умов самі можуть ставати джерелами різних видів руйнівних впливів.

Відповідно, виникає необхідність в розробці математичної моделі прихованих загроз інформаційної безпеки, в якій враховується поведінковий характер об'єктів інформаційної взаємодії. Також розроблено модель інформаційного забезпечення для хмарних обчисленнях, що дозволяє представити формалізований опис інформаційних процесів у середовищі хмарних обчислень у вигляді мультиграфа транзакцій. На основі цих моделей розроблено метод протидії прихованим загрозам, заснований на контролі транзакцій, що відповідають вимогам політики безпеки.

Література
1. Atkinson, D.C. Effective whole-program analysis in the presence of pointers. / D. C. Atkinson, W. G. Griswold // ACM SIGSOFT 1998 Symposium on the Foundations of Software Engineering. - 1998. - pp. 13-42.
2. Олифер В.Г. Компьютерные сети / В.Г. Олифер. - СПб.: Изд- во Питер, 2004. – с. 198-199 .
3. Larochelle , D. Statically detecting likely buffer overflow vulnerabilities / D,Larochelle and D. Evans // In USENIX Security Symposium-2001.-p.177-190.

Тенденції розвитку технологій виготовлення цифрових інтегральних компонентів

Кравченко А.О.
Науковий керівник – к.т.н., доц. Чешун В.М.
Хмельницький національний університет

Інтегральна мікроелектроніка - найбільш значне і, як багато хто вважає, найважливіше науково-технічне досягнення сучасності. За даними опитувань сайту CNN.com, в якому взяли участь понад 119 тис. чоловік, з безлічі технологічних новинок і винаходів, які прийшли в наше життя за останні півстоліття, найважливішою є інтегральна мікросхема, і лише на другому місці, з невеликим відривом - інтернет, на третьому - персональний комп'ютер.

Роком народження напівпровідникової мікросхеми прийнято вважати 1959 рік. Авторами винаходу, радикально змінив спосіб життя людства, стали американські інженери Джек Кілбі (Jack Kilby), який працював на той час в компанії Texas Instruments, і майбутній засновник корпорації Intel Роберт Нойс (Robert Noyce).

Ключовою подією стала пропозиція американського інженера Дж. Кілбі з фірми "Texas Instruments" отримувати еквівалентні елементи схеми (резистори, конденсатори, транзистори і діоди) в монолітному шматку чистого кремнію. Першу інтегральну напівпровідникову схему Кілбі створив влітку 1958 року. А вже в 1961 році фірма "Fairchild Semiconductor Corporation" випустила перші серійні мікросхеми для ЕОМ: схему збігів, напівзсувний регістр і тригер. У тому ж році виробництво напівпровідникових інтегральних логічних схем освоїла фірма "Texas". Наступного року з'явилися інтегральні схеми інших фірм. У стислі терміни в інтегральному виконанні були створені різні типи підсилювачів. У 1962 році фірма RCA розробила інтегральні мікросхеми матриць пам'яті для запам'ятовуючих пристроїв ЕОМ. Поступово випуск мікросхем був налагоджений у всіх країнах - ера мікроелектроніки почалася.

Одночасно з появою перших інтегральних компонентів починає розвиток новий напрямок науки – технічна діагностика інтегральних компонентів та виробів на їх основі. Подальший розвиток зазначеного напрямку технічної діагностики безпосередньо зв’язаний з розвитком

технологій виготовлення інтегральних компонентів.

Таким чином, якісне вирішення питання розробки методу для забезпечення ефективної організації діагностування цифрових пристроїв з комплементарними інтегральними компонентами потребує детального дослідження тенденцій розвитку технологій виготовлення інтегральних компонентів та їх впливу на розвиток технічної діагностики.

Масовий випуск інтегральних компонентів зумовив не лише конкуренцію виробників мікросхем, але і конкуренію застосовуваних ними технологій.

На старті розвитку мікроелектроніки основними були дві технології:

- резисторно-транзисторна логіка (РТЛ);
- діодно-транзисторна логіка (ДТЛ).

При формуванні назви типу логіки було застосовано спосіб реалізації двох каскадів в структурі базових логічних елементів – вхідного і вихідного.

Вхідний каскад логічних елементів відповідає за опрацювання вхідних сигналів, а вихідний – за формування (підсилення) вихідних сигналів.

Резисторно-транзисторна логіка (Resistor-Transistor Logic - RTL), відповідно, як технологія свою назву отримала завдяки реалізації логічних функцій складанням струмів на резисторах, а посилення і інверсії вихідних сигналів - за допомогою біполярних транзисторів.

Модифікацією РТЛ вважають резисторно-ємнісно-транзисторну логіка (РЄТЛ), елементи якої відрізняються від елементів РТЛ тільки наявністю конденсаторів, які шунтують базові резистори[Літ004]. Така модифікація пояснюється тим, що вхідний резистор разом з ємністю переходу база-емітер транзистора утворюють RC-ланцюг, який призводить до затягування фронту вхідного сигналу і зменшує швидкодію логічного елемента. Для уникнення впливу RC-ланцюга в РТЛ паралельно вхідному резистору включають невелику шунтуючу ємність, яка зменшує постійну часу RC-ланцюга і підвищує швидкодію логічного елемента (час перемикання РТЛ близько 30 нС, а для РЄТЛ 15-20 нС). Через незначні відмінності та застарілість технологій РЄТЛ не відділяють з РТЛ і розглядають як її модифікацію.

Серед переваг РТЛ відзначають конструктивну простоту і низьку вартість (серед технологій того часу).

Недоліки РТЛ більш суттєві:

- відносно висока розсіювана потужність (енергоспоживання) як на включеному транзисторному ключі, так і на резисторах;
- нечіткий рівень сигналів (рівень одиниці від ~ 0,9В до напруги живлення);
- вкрай низька швидкодія;
- висока чутливість до шуму в сигналі (низька стійкість);
- складність виготовлення;
- низька навантажувальна здатність виходів (зазвичай не більше трьох входів інших елементів).

Під час переходу на інтегральні мікросхеми РТЛ-логіка практично

зникла і застосовується тільки в спеціальних цілях.

Діодно-транзисторна логіка (Diode-Transistor Logic - DTL) - технологія побудови цифрових схем на основі біполярних транзисторів, діодів і резисторів. Свою назву технологія отримала завдяки реалізації логічних функцій за допомогою діодних ланцюгів, а посилення і інверсії вихідних сигналів - за допомогою біполярних транзисторів.

Основна перевага ДТЛ перед технологією РТЛ - можливість організації значного числа входів логічного елемента.

Основний недолік - значна затримка перемикання елемента через повільність перезарядження ємності переходу база-емітер (близько 30 нС), яка є наслідком роботи транзистора в режимі насичення. Для зменшення часу розряду включено резистор R2, для зменшення часу заряду - встановлено резистор R1 з відносно малим опором, але це ускладнює схему елемента. За енергоспоживанням та чутливістю ДТЛ трохи краща за РТЛ, але виграш є не дуже суттєвим.

Винахід багатоемітерного транзистора, який дозволив замінити сукупність діодів ДТЛ, зумовив початок епохи транзисторно-транзисторної логіки (ТТЛ) та її модифікацій.

Транзисторно-транзисторна логіка (Transistor-Transistor Logic - TTL) - перша широко поширена технологія виготовлення напівпровідникових інтегральних схем. Свою назву технологія отримала через те, що транзистори використовуються як для виконання логічних функцій (наприклад, І, АБО), так і для інвертування та посилення вихідного сигналу (на відміну від резисторно-транзисторної і діод-транзисторної логіки). Основна ознака ТТЛ - використання біполярних транзисторів, причому структури тільки n-p-n.

Мікросхеми ТТЛ почали випускати в середині 1960-х років.

ТТЛ стала популярною серед розробників електронних систем після того, як в 1965 році фірма Texas Instruments представила серію інтегральних мікросхем 74xx (1963-1965 роки - час від виготовлення перших дослідних зразків до початку серійного виробництва найбільш популярної серії 54/74 фірми Texas Instruments), хоча фірма Texas Instruments і не була першою, хто почав випуск ТТЛ мікросхем. Першою була серія SUHL фірми Sylvania (1963 р.), потім HLTTL фірми Transitron, ці серії були практично ідентичні за параметрами, номенклатурі ІС і збігалися за цоколівкою, тобто були повністю взаємозамінні. Лише третьою була серія 54/74 фірми Texas Instruments. Вона принципово відрізнялася від перших двох за параметрами (трохи менше швидкодія і споживана потужність), але мала зовсім іншу цоколівку і відрізнялася складом мікросхем.Тим не менш промисловим стандартом стала саме серія 74xx фірми Texas Instruments, що значною мірою пояснюється великими виробничими потужностями фірми Texas Instruments, широкою номенклатурою мікросхем, а також її зусиллями по просуванню серії 74xx.

Саме різна цоколівка означала існування двох невзаємозамінних ліній ТТЛ. Завжди в таких випадках "повинен залишитися тільки один" і, через не такий вже й великий час (початок 1970-х), залишилася тільки серія 54/74 і

взаємозамінні з нею (цифрами 74 позначаються мікросхеми "комерційного" класу з температурним діапазоном 0-+70^{0}С; 54 - "військового" класу, їх робота гарантована при температурах від -55 до +125 ^{0}С). Оскільки біполярні інтегральні ІМС серії 74хх фірми Texas Instruments стали найбільш поширеними, їх функціонально і параметрично повторює продукція інших фірм (Advanced Micro Devices, серія 90/9N/9L/9H/9S Fairchild, Harris, Intel, Intersil, Motorola, National, частково К155 виробництва СРСР тощо). З випусків інших фірм слід згадати серію 9000 (схеми малого ступеня інтеграції), 9300 (підвищений ступінь інтеграції), 9600 (генератори і одновібратори) фірми Fairchild. Ці схеми були сумісні за цоколівкою з 54/74, стали промисловим стандартом, але саме в цій серії був вперше випущений цілий ряд цікавих схем підвищеного ступеня інтеграції. Зокрема, популярні 4-розрядні синхронні лічильники 74160 (у нас - К155ІЕ9), 74161 вперше були випущені як 9310 і 9316 ("говорять" назви - 9310 лічильник до 10, 9316 лічильник до 16).

ТТЛ схеми на рівні технології 1960-х - 1970-х років були кращими з усіх, які можна було зробити. Вони перевершували РТЛ та ДТЛ за швидкодією і були дешевші у виробництві, були набагато дешевші і швидші КМОН, в порівнянні з ЕЗЛ (найшвидшими) мали меншу споживану потужність, вартість, були набагато зручніші в застосуванні, а швидкодії їх було досить для більшості завдань. Тому ТТЛ схеми набули найширшого розповсюдження і випускалися дуже багатьма фірмами в усіх країнах, де взагалі вироблялися мікросхеми.

Першою з ТТЛ серій фірми Texas Instruments в 1965 році була випущена "стандартна" серія 74 (і відповідна їй 54). Типові параметри серії 74: час затримки 10 нс, споживана потужність 10 міліват для одного 2-входового елемента мікросхеми 7400 (далі наводяться типові параметри також для 2-входових елементів І-НЕ, для інших вони можуть дещо відрізнятися). Коефіцієнт розгалуження по виходу (тобто, число стандартних логічних елементів тієї ж серії, якими може керувати один логічний елемент; також цей параметр називають здатністю навантаження) дорівнює 10. Всі ці мікросхеми були в 14-вивідних корпусах, скляному плоскому мініатюрному для 54 і пластмасовому ДІП (Dual in Line Plastic, DIP) для 74.

Кількість входів і логічних елементів у 7400-7430, 7450, 7460 була обмежена числом виводів 14-вивідного корпусу, а також тим, що вони робилися на одному і тому ж "базовому кристалі". Провідники, створювані в шарі металізації, для кожного типу елементів були різні, вони і забезпечували створення на типовому кристалі різних мікросхем.

Розвиток ТТЛ мікросхем спочатку йшов в бік збільшення як ступеня інтеграції, так і збільшення номенклатури. Наступним кроком став розвиток ТТЛ в напрямку покращення робочих параметрів.

У 1967 році серія 74 була доповнена серією "підвищеної швидкодії" 74Н і "малопотужною" 74L. Для 74Н типова затримка 6 нс, типова потужність 22 мВт. Для 74L типова затримка 30 нс, споживання - 1 мВт.

Оскільки 14 виводів для більшості типів мікросхем підвищеного

ступеня інтеграції не вистачало, були створені корпуси з 16 і 24 виводами. Корпус з 24 виводами дозволяє розмістити регістр на 8 розрядів з входами і виходами кожного розряду, а також виводами управління (або 16-входовий мультиплексор, дешифратор на 16 виходів, арифметично-логічний пристрій на 4 розряди). У 16-вивідному корпусі можна розмістити 4-розрядний регістр зсуву або лічильник з паралельним завантаженням з усіма керуючими виводами, мультиплексор на 8 входів зі стробом, прямим і інверсним виходами (14 виводів не вистачає).

Проблемою для серії 74L при створенні мікросхем підвищеного ступеня інтеграції була "інтегральна реалізація резисторів", тобто, сумарний опір всіх резисторів в мікросхемі. Оскільки у 74L номінали резисторів в 10 разів вище, ніж у 74, ці резистори займають майже все місце на кристалі (резистори - смужки, ширину яких зменшувати не можна, оскільки вона визначається можливостями технології; можна тільки збільшувати довжину, а тим самим і місце, займане резистором, збільшується). Тому найперші мікросхеми підвищеного ступеня інтеграції сери 74L (дешифратор 4вх в 10вих 74L42, 5-розрядний регістр 74L96 і ін.) мали резисторів не в 10, а всього в 2 рази більше ніж у 74. Це підвищувало вхідні струми і споживану потужність в порівнянні з іншими мікросхемами 74L, але дозволяло розмістити все на кристалі мікросхеми при тодішньому, ще досить низькому рівні технології.

Слід зазначити, що саме поява в широкій номенклатурі середніх інтегральних схем (СІС) дозволила створити міні-ЕОМ. СІС, кожна з яких заміняла до десятка і більше малих інтегральних схем (МІС), дозволила зробити центральний процесор ЕОМ у вигляді небагатьох друкованих плат, а згодом нескладні процесора і на одній платі. На МІС процесор виходив настільки громіздкий і дорогий, що мав сенс тільки в складі "великої ЕОМ".

Майже всі ТТЛ СІС 1970-х років були орієнтовані на застосування в складі процесорів і подібних до них пристроїв обробки інформації.

Щоб збільшити швидкодію ТТЛ логічних елементів, під час виготовлення мікросхеми додатково створюють p-n переходи, або так звані діоди Шоткі, які запобігають режиму насичення транзисторів і тим самим зменшують час переходу транзистора з активного режиму в режим відсікання. Транзисторно-транзисторні логічні елементи з додатковими діодами Шоткі називають ТТЛШ логічними елементами або ТТЛШ логікою.

На рубежі 60-х - 70-х років XX століття була випущена перша серія ТТЛ з діодами Шотткі - 74S (1969 - перші дослідні зразки, 1971 - більш-менш широка номенклатура і масовий випуск).

ТТЛШ-логіка 74S відрізняється від ТТЛ наявністю діодів Шоттки в колах база-колектор, що виключає насичення транзистора, а також наявністю демпферних діодів Шотткі на входах (рідко на виходах) для придушення імпульсних завад, що утворюються через відбиття в довгих лініях зв'язку (довгою вважається лінія, час поширення сигналу в якій більше тривалості його фронту, для найшвидших ТТЛШ мікросхем лінія стає довгою починаючи з довжини в кілька сантиметрів).

Діоди Шотткі, шунтуючи перехід база-колектор транзисторів, дозволили практично виключити насичення транзисторів і зменшити час їх закривання. Прогрес технології дозволив скоротити також паразитні ємності і, тим самим, час відкривання (визначається зарядом цих ємностей).

Типовий час затримки 74S 3 наносекунди, споживана потужність елемента 19 міліват, тобто, 74S вдвічі швидша 74H при трохи меншому споживанні. І при цьому тривалість фронтів сигналів така ж, як у 74H, тобто проектування пристроїв не ускладнюється, вимоги до монтажу не зростають.

Після випуску 74S серія 74H припинила розвиватися, а незабаром і взагалі перестала застосовуватися. Тому в 74H практично не було мікросхем підвищеного ступеня інтеграції, на відміну від 74S, де їх було багато. Проблема більшого енергоспоживання в порівнянні з 74 серією (і взагалі великого споживання ІС підвищеної складності, що доходило до 0,5 ват і більше) вирішувалася за рахунок інтенсивного повітряного охолодження. Це ускладнення систем охолодження цілком окупалося підвищенням швидкодії пристроїв на серії 74S, чому вона і отримала досить велике поширення і розвиток. Різновидів елементів цієї серії було багато. В цей же час набули поширення елементи з трьома станами на виходів. Перші два стани - 0 і 1, третій - відключено (стан високого імпедансу). Це дозволяло організувати шинну архітектуру, при якій на одну і ту ж шину (сукупність ліній з логічними сигналами) по черзі працюють різні вихідні елементи.

Надалі розвиток ТТЛ йшов по лінії зменшення споживання і поліпшення електричних характеристик, в основному за рахунок використання переходів (діодів) Шотткі, на яких падіння напруги може становити 0,2-0,3В замість звичайних 0,6-0,7 В.

Наступною в середині 1970-х років була випущена серія 74LS (малопотужна Шотткі). Точної дати назвати не можна, тому що ця серія була випущена не миттєво як серія, а поступово, спершу окремі елементи, потім ще, все більше і більше. Можна лише вказати, що найперші згадки (і випуск дослідних зразків) відносяться до 1971 року.

74LS, за рахунок прогресу технології, мала малі розміри елементів, і, як наслідок, малі паразитні ємності всередині кристалу. Крім того, була значно (втричі) підвищена гранична частота транзисторів (до 1,5 гігагерц). Типовий час затримки у 74LS близько 10 нс, потужність споживання 2 мілівата, тобто потужність при тій же швидкодії зменшена в 5 разів.

74LS заміняла як 74 серію, так і 74L. 74L була практично витіснена серією 74LS, а 74 продовжувала існувати - в ряді застосувань дещо менша ціна і дещо менші вимоги до конструкції, характерні для 74, не давали 74LS вирішальної переваги.

У цей період (середина і друга половина 70-х років) тривав розвиток СІС, чому значно сприяла мала споживана потужність 74LS - в одній мікросхемі можна використовувати багато елементів без ризику перегріву. Великого поширення набули елементи з 3 станами і 20-вивідниq корпус для таких елементів з шиною 8-бітної ширини (16 входів/виходів, 2 виводи управління, земля і живлення).

Цілий ряд елементів був створений саме на 74LS, оскільки споживана потужність інших серій викликала б надзвичайно сильний перегрів через велику потужність розсіювання. Наприклад, односпрямований 8-розрядний буфер 74S244 має потужність до 900 міліват. Розмістити у такий же корпус двонаправлений буфер серії 74S з удвічі більшою потужністю нереально. А 74LS245 з потужністю близько 0,5 ват - можна без будь-яких проблем.

На технології 74LS були створені так звані "секціоновані мікропроцесори" - набори мікросхем, основою яких була "процесорна секція" - мікросхема розрядністю 2 або 4 біта, що включала арифметико-логічний пристрій, регістри, логіку формування прапорів, необхідні канали зв'язку між згаданими частинами. Це дозволяло, взявши відповідну кількість процесорних секцій, отримати повний пристрій обробки даних процесора, що має будь-яку бажану розрядність. У набір входили також мікросхеми для організації послідовності мікрокоманд, інтерфейсу процесора, контролери переривань і ін. Все це дозволяло на досить невеликій кількості мікросхем створити процесор міні-ЕОМ з будь-якою бажаною системою команд, досить швидкий (мільйони команд в секунду).

Найбільш відомі представники таких наборів мікросхем - Intel 3000 (2-розрядна процесорна секція), відомий у нас як 585 і К589 серії, і AM2900 (4-розрядна процесорна секція) - у нас серії 1804 КР1804.

Починаючи з серії 74LS відбулася відмова від багтоеміттерного транзистора в схемі збірки "І" ТТЛ серій. Замість нього застосовуються діоди Шотткі, звичайні діоди або PnP транзистори (звичайні діоди або PnP транзистори - в поєднанні з діодами Шотткі). Так що для 74LS і наступних серій назва "ТТЛ" має чисто історичне значення, в сенсі наступності. По факту це модифікована ДТЛ логіка, але під назвою ТТЛШ.

У 1979 році фірма Fairchild створює серію 74F. Використання технології Isoplanar-II (глибоке селективне окислення, що забезпечує бічну ізоляцію елементів замість P-N переходів), чергове зниження проектних норм (тобто зменшення розмірів елементів), підвищення в 3 рази в порівнянні з 74LS граничної частоти транзисторів (до 4.5 ГГц) забезпечили високі параметри цієї серії. Типові часи затримки краще ніж у 74S (близько півтора разу, тобто 2 нс при малій ємності навантаження, 3 нс при ємності навантаження 50 пФ) при різко зниженій типовій споживаній потужності – 5.4 мВт, тобто, в 3.5 разів менше ніж у 74S. Максимальний вихідний струм логічного нуля у них такий же, як у 74H і 74S (20 міліампер). Буферні схеми з підсиленим виходом мають вихідний струм до 64 мА (в основному застосовуються для роботи на шини, розташовані на задній панелі корпусу, і розраховані на підключення багатьох плат, наприклад, шина VME).

До складу серії 74F входить багато СІС, чому сприяє невелика розсіювана потужність.

У 1980-1982 роки відповідь дала Texas Instruments. Були випущені дві серії практично того ж технологічного рівня, що і 74F - 74ALS (початок виробництва 1980 рік) і 74AS (початок виробництва 1982 рік).

У 74ALS технологічні вдосконалення були використані для

максимального скорочення споживаної потужності при зростанні швидкодії щодо 74LS. Типова потужність споживання 74ALS дорівнює 1.2 мВт, тобто, майже стільки ж, скільки у 74L. Час затримки дорівнює 5 нс, що краще ніж у 74H, яка споживала майже в 10 разів більше, і приблизно в 2 рази краще 74LS.

74AS - спроба перевершити 74F за швидкодією. Споживання 8 мВт, середній час затримки трохи менше ніж у 74F. Оскільки потужність споживання була набагато більше, ніж у 74F, за незначної переваги в швидкості, тому 74AS отримала обмежене застосування.

Дійсно масовими стали 74ALS і 74F (останню виробляла і фірма TI, мабуть, за ліцензією фірми Fairchild).

До цього ж часу (1981 рік) відноситься зняття з виробництва безнадійно застарілих серій 74H і 74L. 74H у всьому поступалася серії 74S, а 74L були витіснені більш швидкою серією 74LS з усіх областей, де важлива швидкодію, а з областей, де головним є мінімум споживання енергії -, мікросхемами на польових транзисторах, які до цього часу неабияк подешевшали і (при напрузі живлення 10 вольт) зрівняти з 74L в швидкодії, при цьому мали перевагу в економічності. Загальна споживана потужність для пристрою, що виконує ті ж функції, на на польових транзисторах (у комплементарни елементів), зазвичай в 10 і більше разів менше, ніж на 74L.

Останньою спробою подальшого розвитку ТТЛ була створена фірмою Fairchild серія 74Fr (кінець 1980-х). Подальше вдосконалення технології, підвищення вдвічі граничної частоти транзисторів (до 9 ГГц) дало досить невеликий виграш у швидкодії, приблизно в 1.4 рази, при зростанні в 1.5 рази споживаної потужності, в порівнянні з 74F (це дані порівняння 74F74 і 74Fr74). Це не привернуло уваги споживачів, серія 74Fr не набула поширення, була знята з виробництва фірмою Fairchild, хоча 74F продовжувала вироблятися.

Окрім ТТЛ і її модифікацій було створено ряд альтернативних технологій виготовлення мікросхем на біполярних транзисторах. До їх числа слід віднести емітерно-зв'язану логіку (ЕЗЛ) та інтегрально-інжекційну логіку (ІІЛ або І2Л). Через специфічні особливості зазначених технологій в їх назвах відійшли від застосування ідентифікаторів застосовуваних радіоелементів (резисторів, діодів та транзисторів).

Емітерно-зв'язана логіка (Emitter Coupled Logic, ECL) - технологія побудови цифрових схем на основі біполярних транзисторів включених за схемою диференціального підсилювача [Літ010].

ЕЗЛ була винайдена в серпні 1956 року інженером IBM Хеноном Йорком (англ. Hannon S. Yourke). Спочатку мала назву «керована струмом логіка», застосовувалася в комп'ютерах Stretch, IBM 7090, і IBM 7094. Також використовувалася назва «схема токового режиму».

Перша серія мікросхем ЕЗЛ логіки, MECL I, була представлена фірмою Motorola в 1962 році. Motorola розробила поліпшені серії MECL II в 1966 році і MECL III в 1968 році. MECL III володіла затримкою розповсюдження сигналу в 1 наносекунду і частотою перемикання тригерів

до 500 МГц. У 1971 році випущено серію 10000 зі зниженим енергоспоживанням і швидкодією.

Основна деталь ЕЗЛ-логіки - схема потенційного порівняння, зібрана не на діодах (як у ДТЛ), а на транзисторах за схемою диференціального підсилювача. Включення транзисторів в вхідних каскадах інтрегральних мікросхем ЕЗЛ дозволяють їм працювати в ненасиченому режимі. Як правило, один транзистор у схемі порівняння підключений до опорного рівня, рівному напрузі логічного порогу, а інші транзистори є входами. Вихід схеми порівняння подається на підсилювальні транзистори, а з них - на вихідні емітерні повторювачі. Використання емітерних повторювачів на виході значно зменшує вихідний опір схеми, забезпечує можливість підключення значної кількості входів інших елементів та забезпечує шунтування збурень в ланцюгу навантаження.

Основною перевагою ЕЗЛ є підвищена швидкість (150 МГц вже в перших зразках 1960-х років і 0,5-2 ГГц у 1970-1980-х). ЕЗЛ є найшвидкодіючою з усіх типів логіки, побудованих на біполярних транзисторах. Це пояснюється тим, що транзистори в ЕЗЛ працюють у лінійному режимі, не переходячи у режим насичення, вихід з якого уповільнений.

Низькі значення логічних перепадів в ЕЗЛ-логіці сприяють зниженню впливу на швидкодію паразитних ємностей, а диференціальний підсилювач на вході дозволяє значно зменшити вплив синфазних збурень у вхідному сигналі.

Також, виходи кількох елементів ЕЗЛ можуть бути об'єднані на спільному резисторі навантаження.

Основні недоліки - відносно високе енергоспоживання, необхідність використання двох джерел живлення з від’ємною напругою (для вхідного та вихідного каскадів окремо), від’ємні значення рівнів напруги дискретних сигналів 0 та 1, необхідність встановлення додаткових резисторів навантаження. Ні за напругою живлення, ні по логічним рівням ЕЗЛ схеми не сумісні з ТТЛ.

Високе енергоспоживання ЕЗЛ та інші недоліки обмежили її розповсюдження застосуванням тільки в схемах, де було важливо мати максимальну швидкодію.

Інтегрально-інжекційна логіка (ІІЛ, I2L) - технологія побудови логічних елементів на біполярних транзисторах, що є розвитком технології логіки з безпосередніми зв'язками між транзисторами, яка в іноземній літературі називається MTL - Merged Transistor Logic або DCTL - Direct-Coupled Transistor Logic.

Інтегрально-інжекційна логіка з'явилася в 1971 році.

В основі логіки ІІЛ лежить використання «особливих» транзисторів з об'єднаною базою (рис.1.5). Ці транзистори не здатні проводити струм через брак носіїв зарядів в базі. Тому поряд з транзистором знаходиться «інжектор» - електрод, що «додає», або, як кажуть, «інжектує» заряд в базу. При цьому транзистор ніби включається і може виконувати корисну роботу.

При проектуванні мікросхем ІІЛ основну роль відводять саме інжекторам. Емітери, як правило, з'єднані ними, є підкладкою мікросхеми. На поверхні кристала знаходяться тільки бази, а на базах - колектори. Таким чином, ІІЛ-транзистор за розміром (якщо не зважати на інжектор) менше польового транзистора. Причому один інжектор може використовуватися для багатьох транзисторів.

Хоча логічні рівні ІІЛ дуже близькі («Нуль»: 0,2В, «Одиниця»: 0,7В) схеми ІІЛ мають високу стійкість до шуму, оскільки управляються струмом, а не напругою.

Переваги ІІЛ:

- висока економічність;
- висока щільність транзисторів на кристалі (іноді вище, ніж у структур на польових транзисторах), що дає можливість отримати високий ступінь інтеграції і меншу вартість, ніж у пристроїв, побудованих за принципами інших логік;
- мале споживання енергії на одне перемикання ~ $1 / 10^{12}$Дж;
- низька напруга живлення: 1-3 В.

До недоліків відносять невисокі робочі частоти – максимальні робочі частоти до 50 МГц.

Хоча технології ЕЗЛ та ІІЛ і мали суттєві переваги за окремими показниками перед ТТЛШ, наявні недоліки не дозволили їм стати технологіями масового серійного випуску мікросхем і перемогти в конкуренції ТТЛШ.

Спільною рисою всіх розглянутих технологій є їх реалізація на біполярних транзисторах.

Іншим напрямком розвитку технологій виготовлення інтегральних компонентів була їх реалізація із застосуванням польових транзисторів.

Розвиток комп'ютерної схемотехніки на основі польових транзисторів почався з появою в 1962 р. польового транзистора з індукованим каналом.

МОП-транзистор (метал-оксид-напівпровідник) - один з видів польового транзистора, в якому керуючий електрод (затвор) відділений від каналу шаром діелектрика, в найпростішому випадку, оксиду кремнію. Транзистори МОП-структури краще за інші активні напівпровідникові прилади підходили для створення логічних БІС і НВІС і ранній прогрес цифрової техніки обумовлений мікросхемами на транзисторах з МОП-структурою. На відміну від біполярного транзистора, вихідний струм якого управляється вхідним струмом, МОП-транзистор управляється напругою.

Оскільки польові (уніполярні, канальні) транзистори мають структуру метал-діелектрик-напівпровідник, то в загальному випадку називаються МДН-транзисторами. З іншої сторони, діелектрик найчастіше реалізувався на основі діоксиду кремнію SiO_2, тому застосовують частіше назву МОН-транзистори (метал-оксид-напівпровідник), хоча останнім часом діелектрики перестали бути різновидами оксидів і назва МДН-транзистори є правильнішою, хоча і менш звичною (поширеною). Для усунення неоднозначності як ідентифікатор назви транзистора часто використовують

загальні терміни – польовий або уніполярний.

Польові транзистори займають на кристалі мікросхеми в 6-9 разів меншу площу, ніж біполярні транзистори, які використовуються в ТТЛ, а також значно простіші за технологією виготовленння. Це дозволило добитися високого ступеня інтеграції і створити мікропроцесори (процесори зібрані в одній мікросхемі), а також забезпечити їх помірну вартість.

Загалом, виготовлення мікросхем на польових транзисторах має ряд переваг перед іншими технологіями:

- менший розмір польового транзистора дає змогу реалізовувати на кристалі схеми більшої складності;
- простіша технологія виготовлення польового транзистора зменшує вартість виготовлення мікросхем;
- польві транзистори можуть виконувати роль резисторів та конденсаторів (пасивних елементів), що забезпечує однорідність застосовуваних елементів і операцій при виготовленні мікросхем;
- польові транзистори можуть використовуватись як елементи пам'яті;
- на польових транзисторах можуть виготовлятися як цифрові, так і аналогові мікросхеми.

Перелічені переваги не могли не викликати зацікавленості у розробників інтегральних компонентів.

За типом застосовуваного напівпровідника і, відповідно, залежно від типу носіїв зарядів, польові транзистори можуть бути n-канальними або p-канальними, в перших в якості носіїв заряду використовуються електрони, в других - дірки. Обидва типи транзисторів знайшли застосування в інтегральних технологіях виготовлення мікросхем. Технологія p-МОП при майже повній відповідності N-МОП має ті ж характеристики, але швидкодія багато нижче з причини в 3 рази меншої рухливості носіїв струму типу «Р» - «дірок».

Ще одним рішенням у виробництві інтегральних компонентів на польових транзисторах стала комплементарна технологія (КМОН або КМДН).

Комплементарні схеми на польових транзисторах в 1963р. винайшов Френк Вонлас (Frank Wanlass) з компанії Fairchild Semiconductor, а перші мікросхеми за цією технологією були створені в 1968р. Термін комплементарні означає, що в схемі використовуються пари транзисторів, ідентичних за значеннями електричних параметрів, але з різною провідністю. Таким чином, відмінною особливістю структури КМОН в порівнянні з іншими МОН-структурами (N-МОН, Р-МОН) є наявність як n-канальних, так і p-канальних польових транзисторів на одній напівпровідниковій підкладці. Як наслідок, КМОН-схеми мають вищу швидкодію та менше енергоспоживання, проте, при цьому характеризуються складнішим технологічним процесом виготовлення і меншою щільністю упаковки.

Особливістю мікросхем на комплементарних польових транзисторах є те, що в цих мікросхемах в статичному режимі ток практично не споживається. Споживання струму відбувається тільки в момент її

перемикання з одиничного стану в нульове і навпаки. Цей струм викликаний двома причинами - одночасним переходом верхнього і нижнього транзисторів в активний режим роботи і перезарядом паразитного ємності навантаження.

В результаті цієї особливості комплементарних мікросхем, вони мають перевагу перед розглянутими раніше видами цифрових мікросхем - споживають струм в залежності від поданої на вхід тактової частоти.

В основі побудови інших логічних елементів комплементарної логіки лежить схема інвертора на комплементарних (взаємодоповнюючих) польових транзисторах з індукованим каналом різної провідності p і n типу, виконаних на загальній підкладці.

Як і в разі простого інвертора, особливістю двоходових елементів є наявність двох ярусів транзисторів щодо вихідного виводу. Логічна функція, виконувана схемою, визначається транзисторами нижнього ярусу. Для реалізації І-НЕ в позитивній логіці транзистори з n-каналом включаються послідовно один з одним, з p-каналом - паралельно, а для реалізації АБО-НЕ - навпаки. Логічні елементи з великим числом входів організовані подібним же чином.

Мікросхеми КМОН-структури близькі до ідеальних ключів: в статичному режимі вони практично не споживають потужності, мають великий вхідний і малий вихідний опір, високу перешкодозахищеність, велику навантажувальну здатність, гарну температурну стабільність, стійко працюють в широкому діапазоні живлячої напруги (від +2 до +18В). Вихідний сигнал практично дорівнює напрузі джерела живлення. При напрузі джерела живлення +5В забезпечується сумісність логічних рівнів зі стандартною ТТЛ/ТТЛШ-логікою. Гранична напруга при будь-якій напрузі живлення дорівнює половині напруги живлення $U_{пор} = 0,5\ E_{ж}$, що забезпечує високу стійкість перед перешкодами (допустима напруга статичної завади рівна половині напруги живлення).

Хоча комплементарна технологія мала великі переваги і навіть була застосована фірмою RCA при випуску процесора COSMAC, стрімкого розвитку вона не зазнала.

У середині 70х років вирішувалася проблема, за якою технологією робити чипи процесорів. Вона обговорювалася в колі фахівців і навіть у пресі. У той час випускалися процесори по різних технологіях, стояло питання, яка технологія може послужити основою для масового випуску дешевого процесора. Застосуванню ЕЗТЛ і ТТЛ заважала підвищена розсіювана потужність і складна технологія. Технологія p-МОП при майже повній відповідності N-МОП мала ті ж характеристики, але швидкодія багато нижче, з причини в 3 рази меншої рухливості носіїв струму типу «Р» - «дірок». ІІЛ на той момент вимагала другого джерела живлення і обов'язкового застосування схем перетворення рівня сигналів.

Основна боротьба йшла між КМОН і n-МОН технологіями.

При обговоренні говорилося, що КМОН складні в технології (на 1 етап маскування і 1 етап дифузії і легування більше ніж n-МОН), мають великі

розміри вентиля і переважніші в застосуванні там, де на перший план виходять вимоги зниженого споживання живлення. А n-МОН структури при малих розмірах вентиля, прийнятній затримці на вентиль, має більш простий і відпрацьований технологічний процес. Тоді, Intel зупинила свій вибір на n-МОН техпроцесі, про що було гучно оголошено.

У документації на процесори проглядається тенденція:

- технологія процесорів до 8086 описується як n-МОН;
- процесор 8086 - 3 мікрона n-канал, silicon gate technology (МОН);
- процесор 80186 - n-МОН;
- починаючи з процесора 80286 всякі згадки про технологію пропадають (як вважається, чере відхід від раніше обгрунтованого вибору n-МОН).

Вирішальною перепоною поширення «класичної» комплементарної технології був її основний недолік - низька в порівнянні з ТТЛ швидкодія. Це обумовлено тим, що ізольований затвор МОП-транзистора являє собою конденсатори досить великої ємності - в базовому елементі до 10-15 пФ [Літ015]. У сукупності з вихідним резистивним опором попередньої схеми такий конденсатор утворює фільтр низьких частот. Зазвичай, розглядають не просто частотні властивості, а час затримки поширення сигналу на один логічний елемент. Затримка виникає через те, що фронт сігналу не вертикальний, а похилий, і напруга на виході ще тільки почне наростати (або знижуватися), коли напруга на вході досягне вже значної величини (в ідеалі, половини напруги живлення). Час затримки міг досягати у ранніх серій КМОН величини 200-250 нс (у базовій серії ТТЛ всього 7,5 нс). На практиці, при напрузі живлення 5В максимальна робоча частота «класичного» КМОН не перевищує 1-3 МГц.

Іншим наслідком наявності високої вхідної ємності є те, що при перемиканні виникає імпульс струму перезарядки цієї ємності, тобто, чим вище робоча частота, тим більше споживає мікросхема, і вважалося, що при максимальних робочих частотах її споживання може зрівнятися зі споживанням ТТЛ (принаймні , ТТЛ серії 74LS). Справа ще ускладнюється тим, що через затягування фронтів імпульсів елемент досить тривалий час знаходиться в активному стані, коли обидва вихідних транзистора відкриті (тобто, виникає так званий ефект «наскрізного струму»).

Це ж затягування фронтів в поєднанні з високоомним входом призводить до зниження завадостійкості при перемиканні - якщо на фронті сигналу є високочастотна перешкода, то це може призводити до багаторазових перемикань виходу.

Довгий час КМОН розглядалася як енергоекономна, але повільна альтернатива ТТЛ. Номінальна швидкодія КМОН мікросхем характеризувалася частотою перемикання до 3МГц, тому основна область застосування мікросхем КМОН типу визначалася як цифрові пристрої невисокої швидкодії з обмеженим енергоресурсом. Мікросхеми КМОН знайшли застосування в електронних годинниках, калькуляторах і інших пристроях з батарейним живленням, де енергоспоживання було критичним.

Подібна тенденція спостерігалася майже до кінця 20-го сторіччя. Через мале розповсюдження технологій виготовлення мікросхем на польових транзисторах, основна увага весь цей час зверталася на розвиток технологій діагностування інтегральних компонентів на біполярних транзисторах, а особливостям комплементарних елементів як об'єктів діагностування уваги майже не приділялося уваги.

Ситуація почала змінюватися на межі 1990 року, коли, з підвищенням ступеня інтеграції мікросхем, гостро постала проблема розсіювання енергії на елементах ТТЛш.

Оскільки РТЛ, ДТЛ і ТТЛ не могли конкурувати з ТТЛш і були визнані застарілими, а інжекційна та емітерно зв'язана технології, через специфічні недоліки, були непридатними для масового застосування, з 90-років спостерігається тенденція до конкуренції і намагань вдосконалення тільки двох технологій виготовлення цифрових елементів:

- комплементарної логіки в напрямку збільшення швидкодії;
- транзисторно-транзисторної з діодами Шоткі в напрямку зменшення енергоспоживання.

Якщо наприкінці минулого сторіччя зазначалося, що існує дві основних перспективних технології виготовлення мікросхем – високошвидкісна ТТЛш та економна КМОН, а їх розвиток спрямовується на усунення основного недоліку кожної технології (на зменшення енергоспоживання ТТЛш і збільшення швидкодії КМОН), то вже з появою надшвидкісних серій мікросхем КМОН-логіки 74VHC і 74VHCT швидкодія ТТЛш-елементів стає досяжною для комплементарної логіки, а після випуску супер-надшвидкісної серії 74G робочі частоти переходять за межі 1ГГц і стають недосяжними для ТТЛш-елементів. Це зумовило в 2009р. відмову фірми Texas Instruments від продовження випуску серій мікросхем з модифікаціями ТТЛ-логіки 54/74 і остаточну перемогу комплементарних технологій.

Переважна більшість сучасних мікросхем, в тому числі процесорів, використовують саме комплементарну схемотехніку.

Визначення рівня захищеності інформаційних ресурсів на основі нечітких моделей

Крижанський Р.О.
Науковий керівник – к.т.н.,доц. Красильников С.Р.
Хмельницький національний університет

Володіння та подальше опрацювання інформації є, для цілого ряду підприємств, основним видом діяльності. Інформація стала об'єктом товарних відносин. Її, як і будь-який інший об'єкт економічних відносин, можна імпортувати, експортувати, купити, продати, вкрасти, сфальсифікувати, знищити, перетворити, спотворити, здійснити над нею інші дії, що матимуть несприятливі результати. Інформаційні системи являють

собою основний засіб керування та організації виробництва. Розвиток будь-якого сучасного підприємства залежить від ефективної та надійної підтримки масових та специфічних його зв'язків через глобальні та локальні мережі. Втрата інформації або ж порушення її цілісності та конфіденційності може призвести до значних негативних фінансово-економічних наслідків, в зв'язку з чим, галузь інформаційного захисту привертає до себе все більшої уваги.

Повністю забезпечити конфіденційність ресурсу неможливо лише апаратним або ж лише програмним способом, тому для більш ефективного захисту існує потреба об'єднання різних засобів захисту. Крім того, для побудови ефективних систем захисту необхідно дослідити всі можливі шляхи та способи порушення цілісності інформаційних ресурсів, види атак на ресурси комп'ютерних систем, що дозволить будувати системи захисту вже з урахуванням факторів ризику, за рахунок чого підвищиться ефективність роботи систем захисту.

Серед відомих на сьогодні методів організації систем захисту інформаційних ресурсів і досі немає таких, використання яких забезпечує повну конфіденційність інформації. Отже, на разі постає необхідність пошуку, розробки та реалізації нових методів захисту, а також удосконалення існуючих.

Для забезпечення ефективного вирішення проблеми оцінки захисту інформації в комп'ютерній системі (КС) необхідні спеціальні інтелектуальні засоби. Традиційними математичними методами не завжди можливо ефективно і коректно вирішити дане питання, тому тут доцільніше використовувати методи, основані на нечітких множинах (НМ) та лінгвістичних змінних (ЛЗ), неформальному оцінюванні та пошуку оптимальних рішень.

Стан безпеки в системі характеризується, зазвичай, лінгвістичними даними і для їх формалізації найкраще використовувати поняття ЛЗ. Лінгвістична змінна характеризується набором $(X, T(X), U, G, M)$, де X – назва змінної; $T(X)$ – терм-множина змінної X, тобто це множина назв лінгвістичних значень змінної A, причому кожне з таких значень являє собою нечітку змінну X із значеннями з універсальної множини U з базовою змінною u; G – синтаксичне правило (зазвичай має форму граматики), що породжує назву A значень змінної X, а M – семантичне правило, яке ставить у відповідність кожній нечіткій змінній A її зміст $M(X)$.

Логіко-лінгвістичний підхід можна застосувати для побудови моделі формування нечітких параметрів, які можна використовувати для підвищення ефективності технологій в системі виявлення атак. Така модель представляє собою сукупність дій, представлених у кілька етапів: визначення нечітких понять, формування нечітких еталонів, формування поточних нечітких параметрів та оцінка стану безпеки на основі порівняння еталонних та поточних параметрів.

На основі розглянутого підходу та нечітких арифметичних операцій розроблено нечітку модель з бальною шкалою для оцінки рівня захищеності КС на основі даних експертів (еталонних значень) та результатів опитування користувачів, які проводять оцінку комплексу певних мір захисту. Процес оцінювання стану захищеності КС складається з кількох етапів: формування еталонних значень; оцінка і формування поточного значення та порівняння отриманого значення з еталонними й на основі чого формування висновку про рівень захищеності оцінюваної КС. Суть оцінки згідно моделі з бальною шкалою полягає в тому, що користувач відповідає на попередньо ранжовані питання (компоненти експертного запиту) за складеною експертом N-бальною шкалою. На основі запропонованої нечіткої моделі з бальною шкалою розроблено метод визначення рівня безпеки в КС з варіативною бальною шкалою (рис. 1).

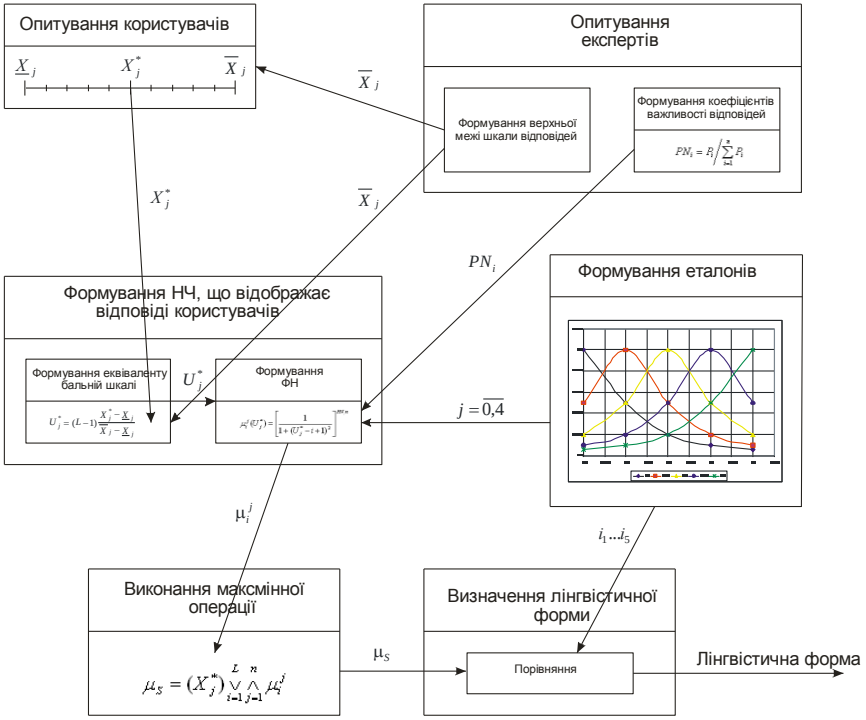


Рисунок 1 – Метод визначення рівня безпеки КС на моделі з варіативною бальною шкалою

Таким чином першим кроком буде формування нечітких еталонів для відображення ЛЗ “Рівень захисту”. Дану ЛЗ визначаємо у вигляді базової терм множини з п’ятьма нечіткими термами $T = \{T_1,\ T_2,\ T_3,\ T_4,\ T_5\} =$

{“низький”(Н), “нижче середнього” (НС), “середній” (С), “вижче середнього” (ВС), “високий” (В)}, які і будуть прикладом для порівняння НЧ.

Визначення стану безпеки КС реалізовується за результатами опитування користувачів системи відповідно до експертного запиту, компоненти якого попередньо ранжуються через визначення коефіцієнту важливості кожного з компонентів експертного запиту. Для ранжування компонентів складається матриця попарного порівняння $A = \left\| a_{ij} \right\|$.

На основі запропонованої моделі можна визначити оцінку рівня безпеки досліджуваної КС за допомогою даних, отриманих від користувачів у вигляді бальних оцінок на кожен компонент попередньо ранжованого експертного запиту.

Суть моделі з лінгвістичною шкалою полягає у тому, що група з N користувачів відповідає на n запитань, відповідно до складеної експертом нечіткої шкали. За відповідями користувачів формується НЧ $Z_t, t = \overline{1, N}$, якому ставиться у відповідність одне з еталонних значень. Значення НЧ, що відповідає оцінці відповідей всієї групи користувачів на певне питання визначається за формулою:

$$L_j = \sum_{t=1}^{N} Z_t \Big/ N, \qquad (1)$$

де $\sum_{t=1}^{N} Z_t$ – нечітка сума, визначена.

Ґрунтуючись на моделі з лінгвістичною шкалою запропонований метод оцінювання стану безпеки інформаційних ресурсів в КС.

Модель з лінгвістичною шкалою дає змогу опрацьовувати вхідні дані, отримані від користувачів у лінгвістичному вигляді, вона є більш точною, проте потребує потужнішого математичного механізму для опрацювання даних.

Застосування логіко-лінгвістичого підходу в моделях дозволило формалізувати (за допомогою використання лінгвістичних змінних) стан безпеки в комп’ютерній системі, який, зазвичай, визначається у лінгвістичній формі. Лінгвістичні змінні використовуються як механізм для опису складної системи з параметрами, які представлені не лише в кількісному, але і в якісному вигляді. Процес оцінювання стану захищеності комп’ютерної системи згідно побудованих моделей складається з наступних етапів: формування еталонних значень, оцінка і формування поточного значення, порівняння отриманого значення з еталонними і формування висновку про стан безпеки інформації. На базі побудованих нечітких моделей розроблено методи для визначення стану безпеки досліджуваної комп’ютерної системи. На основі розроблених нечітких моделей та на запропонованій Л.Хоффманом загальній послідовності дій по визначенню рівня безпеки, в результаті проведеного дослідження розроблено структуру

системи оцінки стану безпеки інформаційних ресурсів на лінгвістичних та варіативних бальних шкалах.

Реалізація та використання системи оцінки рівня безпеки даних в комп'ютерній системі має вагоме практичне значення, так як з її допомогою можливо «використання» вчасно виявляти (і відповідно при необхідності ліквідовувати) загрози на інформацію за рахунок чого підвищується надійність та захищеність комп'ютерних систем.

Література

1. Корченко О.Г. Построение систем защиты информации на нечетких множествах. Теория и практические решения./ О.Г. Корченко – К.: "МК-Пресс", 2006. – 320 с., ил.
2. Анин Б. Ю. Защита компьютерной информации. /Б. Ю. Анин –СПб.: БХВ-Петербург, 2000. –384 с.: ил.
3. Герасименко В.А. Защита информации в автоматизированных системах обработки данных. /В.А. Герасименко - М: Энергоатомиздат, 1994. – 400с.
4. Хорошко В.А. Методы и средства защиты информации. / В.А. Хорошко, А.А.Чекатов – К.: Юниор, 2003. –. 478 с. 3.
5. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений. / Л.А. Заде -М.:Мир, 1976.-165 с.

Моделі комплементарних інтегральних компонентів як об'єктів діагностування

Лаврінчук В.В.
Науковий керівник – к.т.н., доц. Чешун В.М.
Хмельницький національний університет

Аналіз моделей цифрових пристроїв з компонентами, які побудовані за комплементарною технологією, дозволяє стверджувати, що жодна з розглянутих математичних моделей не придатна для моделювання процесу діагностування з урахуванням особливостей прояву характерних для комплементарних інтегральних компонентів несправностей. Найбільш близькими є перемикальні моделі, але їх недоліком є відсутність можливості відображення динамічних несправностей, тому наявні перемикальні моделі потребують уточнення для надання їм необхідних властивостей.

Першочергово відзначимо, що модель має бути орієнтована на діагностування цифрових пристроїв з комплементарними інтегральними компонентами з локалізацією несправностей динамічного типу, що є некласичними для інших технологій виготовлення мікросхем. Некласичні несправності цифрових пристроїв з компонентами, побудованими за комплементарною технологією, можуть бути адекватно відображені тільки на транзисторному рівні [1].

Для розробки уточненої математичної моделі за базову візьмем перемикальну модель виду:

$$M = < X, Y, Q, C, S_n, S_p, W, f_C, f_{Sn}, f_{Sp}, f_W, G >$$

де

$X = \{x_1, x_2, ..., x_l\}$ - множина вхідних значень;
$Y = \{y_1, y_2, ..., y_m\}$ - множина внутрішніх значень;
$Q = \{q_1, q_2, ..., q_n\}$ - множина вихідних значень;
$C = \{c_1, c_2, ..., c_p\}$ - множина з'єднувачів;
$S_n = \{s_{n1}, s_{n2}, ..., s_{ns}\}$ - множина додатніх перемикачів;
$S_p = \{s_{p1}, s_{p2}, ..., s_{pr}\}$ - множина від'ємних перемикачів;
$W = \{w_1, w_2, ..., w_q\}$ - множина накопичувачів;
f_C - функція з'єднувачів;
f_{Sn} - функція додатніх перемикачів;
f_{Sp} - функція від'ємних перемикачів;
f_W - функція накопичувачів;
G - структурна схема моделі;
$l, m, n, p, s, r, q \in N$.

Наведена математична модель може бути ефективно використана для моделювання роботи цифрових пристроїв з комплементарними інтегральними компонентами в процесі тестового діагностування з пошуком статичних несправностей, але не є придатною для застосування при пошуку несправностей динамічного типу, оскільки не має інструментарію для відображення характеру зміни сигналів в часі.

Для надання математичній моделі здатності реєструвати характер зміни сигналів в часі введемо до її складу множину моментів часу $T = \{t_1, t_2, ..., t_k\}$, де $k \in N$.

Таким чином, математична модель прийме наступний вигляд:

$$M = < X, Y, Q, C, S_n, S_p, W, f_C, f_{Sn}, f_{Sp}, f_W, G, T >.$$

Надамо пояснення уточненій моделі відносно задач діагностування цифрових пристроїв з комплементарними інтегральними компонентами.

Елементами цифрових пристроїв з комплементарними інтегральними компонентами є:

1) *n*-канальні транзистори;
2) *p*-канальні транзистори;
3) провідники з'єднань.

Оскільки значну роль у функціонуванні цих пристроїв відіграють паразитні ємності, то необхідно їх також явно відобразити в моделі. Звідси ми приходимо до наступних компонент моделі:

1) з'єднувач C, який моделює безпосереднє з'єднання провідників типу «монтажне АБО»;
2) додатній перемикач S_n, який відображає *n*-канальний польовий транзистор;
3) від'ємний перемикач S_p, який описує *p*-канальний польовий

транзистор;

4) накопичувач *W*, який описує паразитну ємність.

Роботу кожного з перелічених компонентів описують вхідні і вихідні сигнали, а накопичувач характеризується нетиповим для комбінаційних елементів інших логік параметром стану, що відображується значенням збережуваного «внутрішнього сигналу». В моделі кожен вид сигналів відображено окремою множиною: $X=\{x_1, x_2, ..., x_l\}$ - множина вхідних значень, $Y=\{y_1, y_2, ..., y_m\}$ - множина внутрішніх значень, $Q=\{q_1, q_2, ..., q_n\}$ - множина вихідних значень моделі.

Сигнали цифрових пристроїв, які розглядаються, відносяться до цифрових сигналів. До останніх віднесемо: 1) логічні нуль та одиницю; 2) стан високого опору. В комплементарних схемах можлива так звана «боротьба» сигналів (наприклад, при під'єднанні до заряду паразитної ємності тощо). При цьому фактичне значення сигналу в певний момент часу важко визначити. Тому в модель, як правило, вводять поняття невизначеного стану, яке і описує такі явища. Виходячи з цього, задамо опис сигналів пристроїв, що розглядаються, у вигляді мінімальної множини значень сигналів V_4:

$$V_4 = \{0, 1, U, Z\},$$

де

0 - логічний нуль, який описує сигнали, що належать області низьких значень напруги V_L;

1 - логічна одиниця, яка описує сигнали, що належать області високих значень напруги V_H;

U - невизначене значення, яке відповідає невідомим, невизначеним станам схеми, «боротьбі» сигналів;

Z - значення, яке відповідає стану схеми, в якому вона має високий вихідний опір або розриву з'єднання.

З урахуванням прийнятої системи позначень сигналів дамо опис законів функціонування елементів моделі, що належать кожній з перелічених множин *C, S_n , S_p і W*.

З'єднувач *C* (рис.1) характеризується:

- вхідними змінними x_1, x_2;
- вихідною змінною q_C;
- функцією з'єднувачів f_C, яка реалізує логіку «монтажне АБО».

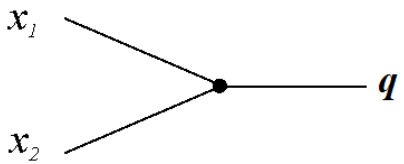


Рисунок 1 – Схематичне зображення з'єднувача $c_i \in C$

$$q_{Ci} = f_C(x_1, x_2) = [0]((x_1 = 0) \wedge (x_2 = 0)) \vee [1]((x_1 = 1) \wedge (x_2 = 1)) \vee [x_1](x_2 = Z) \vee$$
$$\vee [x_2](x_1 = Z) \vee [U]((x_1 = U) \vee (x_2 = U)) \vee [U]((x_1 = 1) \wedge (x_2 = 0)) \vee \quad ,$$
$$\vee [U]((x_1 = 0) \wedge (x_2 = 1));$$

де значення у квадратних дужках вказує логічне значення, а у круглих - означає умови його встановлення.

Вихідний сигнал на стоці польового транзистора залежить від характеру розповсюдження сигналів на його складових елементах (затвор, підкладка, витік). Введемо наступні позначення елементів польового транзистора: *G* - затвор, *BS* - підкладка, *D* - стік, *S* – витік.

Додатній перемикач $s_{ni} \in S_n$ (рис.2) характеризується:

- вхідними змінними *G, S, BS,* де $G \in X \cup Y$, $S \in X \cup Y$, $BS \in X \cup Y$;
- вихідною змінною *D,* де $D \in Y \cup Z$;
- функцією додатніх перемикачів $f_{Sn}(G, S, BS)$.

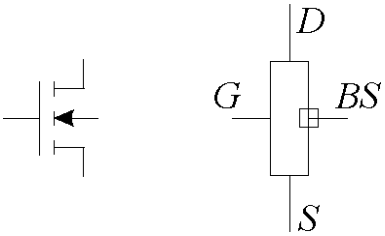


Рисунок 2 – Схематичне зображення додатнього перемикача $s_{ni} \in S_n$

Отримаємо опис додатнього перемикача $s_{ni} \in S_n$:

$$D_{Sni} = f_{S_n}(G, S, BS) = [Z](G = BS) \vee [S]((G = 1) \wedge (BS = 0)),$$

де позначення [S] означає будь-яке значення витоку ([S] дублюється на вихід D) .

Від'ємний перемикач $s_{pi} \in S_p$ (рис. 3) характеризується:

- вхідними змінними *G, S, BS,* де $G \in X \cup Y$, $S \in X \cup Y$, $BS \in X \cup Y$;
- вихідною змінною *D,* де $D \in Y \cup Z$;
- функцією від'ємних перемикачів $f_{Sp}(G, S, BS)$.

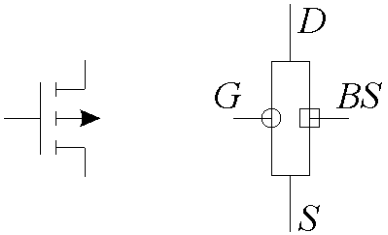


Рисунок 3 – Схематичне зображення від'ємного перемикача $s_{pi} \in S_p$

Отримаємо опис додатнього перемикача $s_{ni} \in S_n$:

$$D_{Spi} = f_{S_p}(G, S, BS) = [Z](G = BS) \vee [S]((G = 0) \wedge (BS = 1)).$$

Накопичувач $w_i \in W$ (рис.4) характеризується:
- вхідною змінною x, де $x \in X \cup Y$;
- вихідними змінними q_w, залежними від часу t, де $z_w \in Y \cup Z$, $t \in T$;
- внутрішніми змінними y, де $y \in V_4$;
- функцією накопичувачів f_W.

Особливістю накопичувача є те, що вихідний сигнал y приймає значення вхідного сигналу x через певний час затримки розповсюдження сигналу Δt:

$$y(t + \Delta t) = x(t).$$

Переходи між рівнями цифрових сигналів характеризуються двома варіантами затримки розповсюдження сигналу Δt: τ_{01} - час затримки переходу між станами з 0 в 1; τ_{0Z} - час затримки переходу між станами з 0 в Z; τ_{10} - час затримки переходу між станами з 1 в 0; τ_{1Z} - час затримки переходу між станами з 1 в Z; τ_{Z0} - час затримки переходу між станами з Z в 0; τ_{Z1} - час затримки переходу між станами з Z в 1; $\tau_{01} \in T$, $\tau_{0Z} \in T$, $\tau_{10} \in T$, $\tau_{1Z} \in T$, $\tau_{Z0} \in T$, $\tau_{Z1} \in T$.

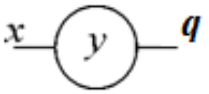


Рисунок 4 – Схематичне зображення накопичувача $w_i \in W$

$$q_{Wi}(t) = f_W(x,y) = [0][t=0](x=0 \wedge y=0) \vee [0][t=\tau_{10}](x=0 \wedge y=1) \vee$$
$$\vee [0][t=\tau_{Z0}](x=0 \wedge y=Z) \vee [0][t=\tau_{10}](x=0 \wedge y=U) \vee$$
$$\vee [1][t=0](x=1 \wedge y=1) \vee [1][t=\tau_{01}](x=1 \wedge y=0) \vee$$
$$\vee [1][t=\tau_{Z1}](x=1 \wedge y=Z) \vee [1][t=\tau_{01}](x=1 \wedge y=U) \vee$$
$$\vee [Z][t=0](x=Z \wedge y=Z) \vee [Z][t=\tau_R](x=Z \wedge y=0) \vee$$
$$\vee [Z][t=\tau_{1Z}](x=Z \wedge y=1) \vee [Z][t=\tau_{1Z}](x=Z \wedge y=U) \vee$$
$$\vee [U][t=\tau_{01} \vee t=\tau_{10}](x=U \wedge y=U) \vee [U][t=\tau_{01}](x=U \wedge y=0) \vee$$
$$\vee [U][t=\tau_{10}](x=U \wedge y=1) \vee [U][t=\tau_{Z1}](x=U \wedge y=Z).$$

Динамічні характеристики комплементарної схеми відображається накопичувачем через часові параметри t, τ_R, τ_F, які поставлені у відповідність його переходам як логічного автомата з одного стану в інший.

Структурна схема G моделі М - це граф, ребра якого - з'єднувачі, а вершини - перемикачі і накопичувачі, причому граф ізоморфний топології пристрою, що моделюється.

Уточнена модель М цифрових пристроїв з комплементарними інтегральними компонентами як об'єктів діагностування має такий запис:

$$M = \begin{cases} q_C = f_C(x_1,x_2) = [0]((x_1=0) \wedge (x_2=0)) \vee [1]((x_1=1) \wedge (x_2=1)) \vee [x_1](x_2=Z) \vee \\ \vee [x_2](x_1=Z) \vee [U]((x_1=U) \vee (x_2=U)) \vee [U]((x_1=1) \wedge (x_2=0)) \vee \\ \vee [U]((x_1=0) \wedge (x_2=1)); \\ \\ D_{Sn} = f_{S_n}(G,S,BS) = [Z](G=BS) \vee [S]((G=1) \wedge (BS=0)); \\ \\ D_{Sp} = f_{S_p}(G,S,BS) = [Z](G=BS) \vee [S]((G=0) \wedge (BS=1)); \\ \\ q_W(t) = f_W(x,y) = [0][t=0](x=0 \wedge y=0) \vee [0][t=\tau_{10}](x=0 \wedge y=1) \vee \\ \vee [0][t=\tau_{Z0}](x=0 \wedge y=Z) \vee [0][t=\tau_{10}](x=0 \wedge y=U) \vee \\ \vee [1][t=0](x=1 \wedge y=1) \vee [1][t=\tau_{01}](x=1 \wedge y=0) \vee \\ \vee [1][t=\tau_{Z1}](x=1 \wedge y=Z) \vee [1][t=\tau_{01}](x=1 \wedge y=U) \vee \\ \vee [Z][t=0](x=Z \wedge y=Z) \vee [Z][t=\tau_R](x=Z \wedge y=0) \vee \\ \vee [Z][t=\tau_{1Z}](x=Z \wedge y=1) \vee [Z][t=\tau_{1Z}](x=Z \wedge y=U) \vee \\ \vee [U][t=\tau_{01} \vee t=\tau_{10}](x=U \wedge y=U) \vee [U][t=\tau_{01}](x=U \wedge y=0) \vee \\ \vee [U][t=\tau_{10}](x=U \wedge y=1) \vee [U][t=\tau_{Z1}](x=U \wedge y=Z). \end{cases}$$

Таким чином, отримано уточнену математичну модель цифрових пристроїв з комплементарними інтегральними компонентами як об'єктів діагностування, орієнтовану на виявлення несправностей динамічного типу з урахуванням часових параметрів.

Література

1. Жиров Г.Б. Узагальнена діагностична модель цифрової ВІС для енергостатичного методу діагностування /Г.Б. Жиров //Вісник КНУ ім. Т. Шевченка. – К.: Київ. ун-т, 2005. – Сер. Військово-спеціальні науки, Вип. 11. – С. 54-60.
2. Глущак С.В. Метод і засоби тестового діагностування цифрових та мікропроцесорних пристроїв з компонентами, побудованими за КМДН-технологією: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 05.13.05 "Елементи та пристрої обчислювальної техніки та систем керування" /С.В. Глущак – Вінниця, 2002. – 19с.
3. Жердєв М.К. Концептуальні засади методу діагностування сучасних цифрових типових елементів заміни по форматним перехідного процесу в шині живлення / М.К. Жердєв, В.О. Савран //Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2016. – Вип. №52. – С. 20-31.
4. Гаврилов С.В. Статический временной анализ КМОП-схем с учетом дестабилизирующих факторов / С.В. Гаврилов, Г.А. Пирютина, А.Н. Щелоков // Известия ЮФУ. Технические науки. – 2013. – №7 – С.65-70
5. Андрюхин А.И. Метод параллельной генерации тестов на переключательном уровне для МОП-схем / А.И. Андрюхин // Электронное моделирование. – 2011 – Т. 33, № 1. – С. 91-98.
6. Діагностичні моделі трансляторів n-МОН та КМОН структур виготовлення / М.К. Жердєв, В.В. Вишнівський, Г.Б. Жиров, С.І. Глухов //Збірник наукових праць ВІТІ НТУУ "КПІ". – К., 2006. – №3 – С.9-12.

Формування бездротових ad hoc мереж з використанням теорії ігор

Михалечко Р.М.
Науковий керівник – к.т.н.,доц. Муляр І.В.
Хмельницький національний університет

Теорія ігор займається описом і аналізом конфліктних ситуацій будь-якої природи. Під конфліктною ситуацією, або грою, розуміється взаємодію незалежних учасників, що володіють свободою волі і діючих відповідно до своїх власних інтересів. Гра не обов'язково являє собою антагоністичне протистояння, в якому завжди є переможець і переможений. Інтереси гравців можуть частково збігатися, але конфлікт полягає в тому, що кожен гравець прагне максимально збільшити свій виграш, не турбуючись про загальне благо або справедливості. Теорія ігор вивчає, якої поведінки слід чекати від раціональних гравців, до яких наслідків воно призведе, чи можна отримати

результати, влаштовують кожного з гравців, і як забезпечити загальне благо при егоїстичних установках учасників.

Один з підходів до розробки децентралізованих алгоритмів формування мережі полягає в застосуванні методів теорії ігор [1]. Вузли мережі розглядаються як раціональні агенти, а загальний критерій якості мережі визначається через індивідуальні функції корисності агентів. Замість глобального або локального оптимуму розглядається рівновага Неша - стан, при якому жоден агент не може поліпшити свою корисність, поодинці змінивши свої дію. Аналогічний підхід використовується при дослідженнях процесів формування соціально-економічних мереж. В грі формування мережі розглядаються різні концепції стійких мереж, а також динамічні процедури побудови мережі.

В теоретико-ігрових алгоритмах важливу роль відіграє правило, за яким агент приймає рішення що до вибору тієї чи іншого дії. Здебільшого використовуються модифікації правила найкращої відповіді. Агент, який використовує це правило, представляє обстановку в грі статичною і вибирає оптимальну дію, виходячи з цього припущення. Відомі класи ігор, для яких послідовність найкращих відповідей сходиться до рівноваги Неша. Недолік такої «наївної» найкращої відповіді в тому, що отримана з її допомогою рівновага може сильно відрізнятися від оптимального рішення.

У даній роботі пропонується нове правило прийняття рішення, назване подвійною найкращою відповіддю double best response. Воно засноване на методах рефлексивних ігор. Основна ідея полягає в тому, що агент може прогнозувати реакцію інших агентів на свої дії і враховувати цю інформацію при прийнятті рішення. Раніше рефлексивні ігри застосовувалися для управління групою агентів, які повинні проникнути через систему датчиків. З позицій теорії рефлексивних ігор розглядаються дуополія Курно, пошук консенсусу в багатоагентній системі, активна експертиза та інші завдання соціально-економічних систем. Також досліджуються безперервні динамічні процедури пошуку рівноваги в активних системах.

Використовується запропонована форма завдання формування топології у вигляді некооперативної гри. Вузли мережі є агентами, а вимоги зв'язності мережі і мінімізації потужності задані в функціях корисності. На основі правила подвійної найкращої відповіді розроблено два алгоритму формування топології бездротової ad hoc мережі: з постійним і змінним правилом прийняття рішення.

Перший алгоритм отримав назву IDBR (Iterated Double Best Response), в ньому вузли постійно використовують подвійну найкращу відповідь. цей алгоритм побудований за прямої аналогії з базовим алгоритмом «наївної» найкращої відповіді. Другий алгоритм називається VDBR (Variable Double Best Response). У ньому вузол використовує нове правило тільки якщо не може збільшити свою корисність за допомогою стандартної «наївної» найкращої відповіді. Обидва алгоритма менш залежні від порядку дій агентів, ніж базовий алгоритм найкращої відповіді.

Розглянемо детальніше другий алгоритм, заснований на динаміці подвійних найкращих відповідей [2].

Цей алгоритм побудований за прямої аналогії з динамікою найкращих відповідей, але агенти вибирають дії за правилом подвійної найкращої відповіді (2).

Подвійною найкращою відповіддю агента i на обстановку a_{-i} називається дія:

$$BR_i^2(a_{-i}) = \arg\max u_i(x, BR_{-i}(x, a_{-i})),$$

(1)

де $BR_{-i}(x, a_{-i}) = (BR_1(x, a_{-i}),..., BR_{i-1}(x, a_{-i}))$,

$(BR_{i+1}(x, a_{-i}),..., BR_n(x, a_{-i}))$ - вектор одночасних найкращих відповідей інших агентів на вибір агентом i дії x.

Алгоритм послідовних подвійних найкращих відповідей.

1. (ініціалізація). Кожен вузол встановлює початкову потужність свого передавача $p_i^0 = p^0$.

   Формується граф $g^0 = g(p^0)$.

2. (Адаптація). Черговий вузол i змінює потужність за правилом подвійної найкращої відповіді (1) або $p_i^{t+1} = BR_i^2(p_{-i}^t)$.

Обмеженою подвійною найкращою відповіддю агента i на обстановку a_{-i} називається дія:

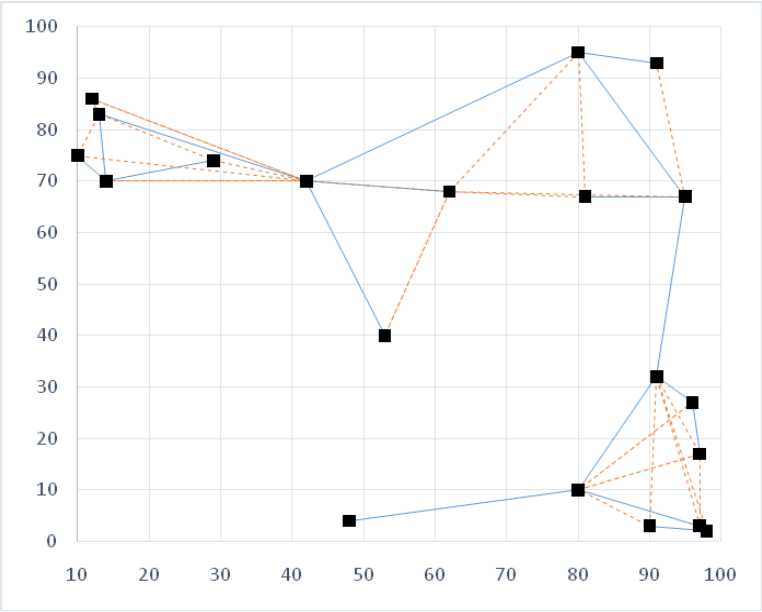
$$BR_{i,R_i}^2(a_{-i}) = \arg\max u_i\left(x, a_{N/R_i} BR_{R_i}(x, a_{-i})\right), \quad (2)$$

де a_{N/R_i} - дії агентів, що не входять в R_i, $BR_{R_i}(x, a_{-i})$ - найкращі відповіді агентів, що входять в R_i.

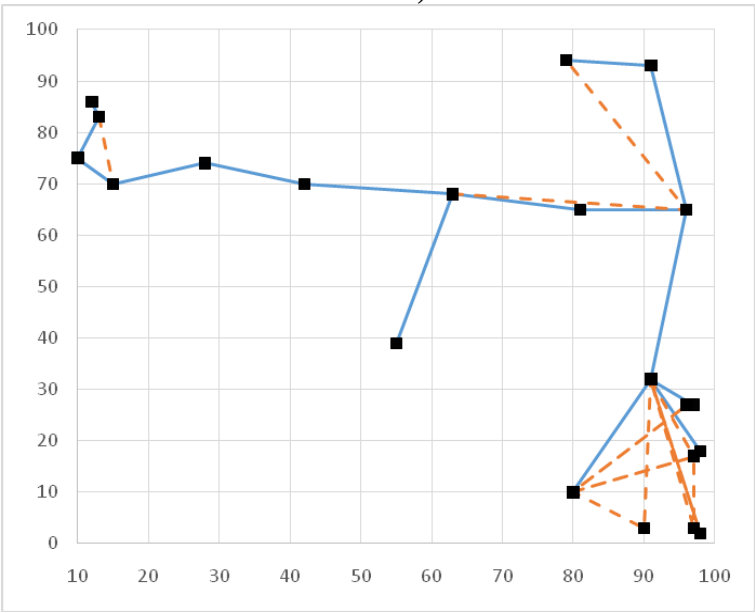
або

$p_i^{t+1} = BR_i^2(p_{-i}^t)$, де R_i - рефлексивна множина.

3. (Оновлення мережі). Формується новий граф $g^{t+1} = g(p_i^{t+1}, p_{-i}^t)$.
4. (Зупинка). Кроки 2 і 3 повторюються до тих пір, поки хоча б один вузол продовжує змінювати свою потужність.
5. (Завершення). Якщо граф g^t не зв'язний, всі вузли переходять на правило найкращого відповіді (1). Перехід до кроку 2.



а)



б)

Рис. 1 - Приклад мережі з 20 вузлів, сформованої алгоритмами а) послідовних найкращих відповідей; б) послідовних подвійних найкращих відповідей

На кроці 2 замість правила (2) може використовуватися правило з обмеженням на рефлексивну множину агентів (3). В експериментах досліджуються обидві модифікації алгоритму.

Після зупинки на кроці 4 не завжди утворюється зв'язкова мережа. Але після запуску однієї ітерації динаміки найкращих відповідей зв'язкова мережу формується завжди. В процесі виконання кроків 2 і 3 можуть утворюватися односторонні зв'язку, для перетворення яких в двосторонні необхідний крок 5.

Подвійна найкраща відповідь в чистому вигляді припускає, що агент може прогнозувати дії всіх інших агентів. Для систем з великим числом просторово віддалених один від одного агентів, наприклад, як сенсорні мережі, ці припущення не реалістичні. Запропоновано локальну версія подвійної найкращої відповіді, в якій агент обчислює тільки реакцію свого найближчого оточення. Надалі динаміка подвійної найкращої відповіді буде досліджена аналітично. В ідеалі необхідно строго довести збіжність і ефективність у порівнянні з динамікою звичайної найкращого відповіді. А також сформулювати ті особливості гри формування топології, з якими пов’язані дані властивості динаміки подвійної найкращої відповіді. Також представляє інтерес дослідження ігор формування мереж з іншими функціями корисності агентів або іншими механізмами формування мережі.

Література

1. Чхартишвили А.Г. Рефлексивные игры. - М.: СИНТЕГ, 2003. - 149 с.
2. Лєнков, Є.С. Формування топології бездротової ad hoc мережі спеціального призначення на основі динаміки подвійних найкращих відповідей/ Д.В. Зайцев, І.В. Муляр, Р.М. Михалечко // Системи обробки інформації. — 2016. — № 9(146). — С. 172-176

Впровадження інформації в звукові файли

Москалюк О. В.
СЗОШ №12, м.Хмельницький

Захист інформації може бути забезпечений криптографією або стеганографією, або одночасно за допомогою криптографії та стеганографії. Найпоширенішим методом впровадження інформації в звукові файли є метод заміни найменшого значущого біта (LSB - Least Significant Bit). В даний час більшість програм, які використовують в якості контейнерів дискретні звукові сигнали, впроваджують інформацію тільки методом LSB, на відміну від програм, що використовують текстові та графічні контейнери. Це пояснюється складністю реалізації альтернативних методів впровадження інформації в звукові сигнали (метод фазової варіації, метод розширення спектра, метод впровадження за допомогою луна-сигналу) і малим об'ємом секретної інформації, що пересилається по таємного каналу зв'язку, організованому на основі зазначених методів.

В даний час нерідко спостерігаються випадки несанкціонованого використання мультимедійної продукції (фотографій, аудіо- та відеофайлів). Одним із прийомів захисту авторських прав є приховане впровадження міток (маркерів, водяних знаків) в мультимедійні файли, що захищаються. Виявлення цих міток дозволяє порушнику видалити водяні знаки з контейнера. Очевидно, що впровадження прихованої інформації в мультимедійні файли слід здійснювати таким чином, щоб порушник не зміг виявити і видалити зроблені зміни в контейнері. Файл формату WAV містить в собі квантовані цифрові значення амплітуди сигналу, виміряні в дискретні моменти часу (так звані відліки). Для файлу формату WAV найбільш відомим і поширеним методом приховування секретної інформації є метод заміни найменшого значущого біта. При впровадженні інформації в звукові файли формату WAV методом найменшого значущого біта доводиться вирішувати завдання вибору номера розряду відліку, в який можна помістити приховувану інформацію, з урахуванням двох конфліктуючих вимог. З одного боку, необхідно збільшувати обсяг прихованої інформації в одному файлі (збільшувати пропускну здатність каналу), а з іншого боку, потрібно забезпечити високий ступінь скритності вкладеної інформації.

Істотно ускладнити зловмисникові відновлення повідомлення можна шляхом розміщення біт повідомлення не в одному, а. в кількох файл-контейнерах (здійснити передачу інформації, по декількох каналах). Порядок розподілу інформації по контейнерах визначається секретним ключем; У цьому випадку, навіть якщо знати про наявність в файлах прихованого повідомлення, відновити його стає складніше, ніж при простому методі LSB. Алгоритм просторового розподілу інформації; полягає в. наступному: (рис. 1): перший біт приховуваного повідомлення записується в перший файл-контейнер, другий - в другий файл-контейнер, третій - в третій файл-контейнер; четвертий - в четвертий файл-контейнер; пятий - в перший, і так далі поки не закінчаться біти повідомлення. При використанні звукового потокового мовлення розпорошення ведеться між різними каналами зв'язку. В файл-контейнер біти записуються методом LSB.. Інформацію перед впровадженням в файл-контейнер, доцільно зашифрувати симетричними шифрами, застосовуваними в якості стандартів в США і в Україні – AES. Зазначений шифр при використанні, ключа довжиною 256 біт практично неможливо розкрити методом «грубої сили» (повного перебору).

Розподіл інформації по декількох контейнерів дозволяє істотно підвищити стійкість інформації до розкриттю зловмисниками. Захищеність впровадженої інформації залежить від кількості використовуваних файл-контейнерів, тобто від довжини ключа. При використанні чотирьох файл-контейнерів кількість додаткових ключів $k = n! = 4! = 24$ де n - довжина ключа в бітах. При використанні десяти файл-контейнерів кількість можливих ключів досягає $k = n! = 10! = 3628800$. Слід мати на увазі, що це додаткові ключі. Поле ключів, використовуване при шифруванні (воно визначено відповідним стандартом), залишається незмінним. Вважаючи, що комп'ютер може перебрати мільйон ключів в секунду, повний перебір 128¬бітного

криптографічного ключа займе 10^{25} років. Зауважимо, що планета Земля існує $5 \cdot 10^9$ років. З урахуванням часу, що витрачається на перебір 128-бітного криптографічного ключа для дешифрування повідомлення, розкриття повідомлення методом повного перебору всіх можливих комбінацій десяти файл-контейнерів потребує 10^{37} років. Ще однією перевагою розподілу інформації по декільком контейнерам (просторовий розподіл) є можливість передавати файл-контейнери по декількох відкритих каналах зв'язку. Це істотно знижує ймовірність перехоплення зловмисником всіх частин повідомлення.

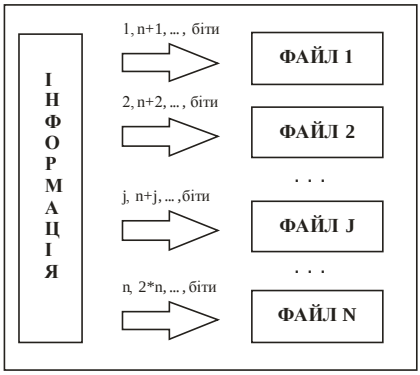


Рисунок 1 - Схема просторового розподілу інформації

Прихований канал зв'язку, заснований на використанні даного методу, має пропускну здатність, рівну пропускної здатності прихованого каналу зв'язку, організованого методом LSB. Пропускна здатність методу LSB дорівнює 1000 біт / с на 1000 Гц частоти дискретизації звукового сигналу. Отже, якщо врахувати розподіл повідомлення по декільком файл-контейнерів, інформацію можна передавати паралельно по декільком прихованим каналах зв'язку. У такому випадку пропускна здатність прихованого каналу зв'язку складе:

$$P = \sum_{i=0}^{n} P_i \qquad (1)$$

де n - кількість файл-контейнерів, P - пропускна здатність i - ого файл-контейнера. При частоті дискретизації звукового,, сигналу 44100 Гц, пропускна здатність прихованого каналу зв'язку на основі метода- LSB. складає 44100 біт / с. При організації прихованого каналу, зв'язку методом просторового розподілу інформації на основі десяти файл-контейнерів, пропускна здатність прихованого: каналу зв'язку у відповідності з формулою (1) буде становити $P = n \cdot P = 10 \cdot 44100 = 441000$ біт / с.

Іншим методом, що дозволяє підвищити стійкість, є метод часового розподілу інформації. В даному методі біти повідомлення розподіляються рівними частинами по файл-контейнеру, не змінюючи відліків, що містять

«тишу». На рис. 2 показані відліки аудіо-файлу, що містять звуковий сигнал і «тишу», а також впровадження інформації тільки в ті з них, в яких присутній сигнал.

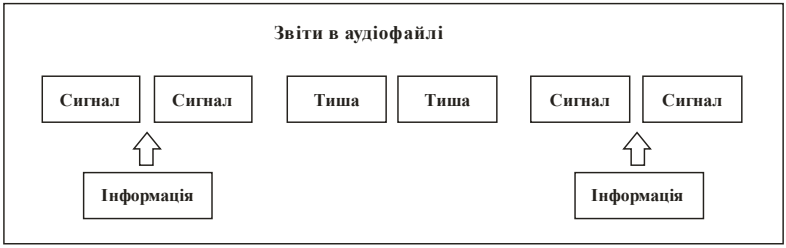


Рисунок 2 - Схема часового розподілу інформації

Суть методу полягає в тому, що в області даних аудіофайлу в кожному відліку замінюється кілька молодших біт в межах, від 1 до 8 значеннями біт інформації користувача. Також область даних аналізується на наявність «тиші», і, якщо вона виявлена, то поточний відлік пропускається. Такий метод розподілу інформації по файл-контейнеру дозволяє захистити впроваджену інформацію від виявлення вкладення методом спектрального стегоаналіза. Відсутність вкладення в звітах, що містять «тишу», не дозволяє виявити це вкладення, аналізуючи характеристики амплітудного спектра звукового сигналу, що міститься в файл-контейнері. Пропускна спроможність прихованого каналу зв'язку, організованого на основі даного методу, складе:

$$P = \frac{(100 - k) \cdot P_L}{100}, \qquad (2)$$

де k - тривалість ділянок «тиші» у відсотках, P_L – пропускна здатність методу LSB для розрахункового файл-контейнера. Для мовних звукових сигналів в середньому припадає 913 ділянок «тиші» на годину (0,2536 ділянок «тиші» в секунду), для музичних – 200 ділянок «тиші» на годину (0,0556 ділянок «тиші» в секунду). Тривалість ділянок «тиші» становить 5% часу звукового сигналу. Таким чином, пропускна здатність даного методу для файл-контейнера з частотою дискретизації 44100 Гц і пропускною здатністю методу LSB - 44100 біт/с відповідно до формули (2) становитиме $P = \frac{(100 - 5) \cdot 44100}{100} = 41895 біт/с$. Це становить 95% від максимальної пропускної здатності каналу.

Для впровадження інформації в MIDI-файли можна використовувати наступні події і параметри: портаменто; text; lyric; номер ноти; гучність звучання ноти; тривалість звучання ноти. Портаменто - це ефект плавного переходу по висоті тону від однієї ноти до наступної ноти. Даний ефект полягає в тому, що нота починає звучати на висоті попередньої ноти, а закінчує звучання на своїй нормальної висоті. Задіяти режим портаменто

можна за допомогою перемикача Portamento, що визначається повідомленям зміни режиму управління з параметром 65 зі значенням від 64 до 127. Щоб відключити режим портаменто, необхідно використовувати перемикач Portamento значенням від 0 до 63. Параметром ефекту портаменто, що впливає на відтворення ноти, є час портаменто, тобто час, за який відбувається зміна висоти тону. Часом портаменто керує подія Portamento Time - повідомлення зміни режиму керування з параметром 5, значення цієї події визначає старшу частину часу портаменто. Повідомлення зміни режиму керування з параметром 37 визначає молодшу частину часу портаменто. Text - це мета-подія, яка визначає текстовий рядок, що складається зASCII-символів. Дана подія ніяк не впливає на відтворення звуку, її можна побачити тільки в секвенсорі або музичному редакторі. Ця подія допомагає розробнику орієнтуватися у списку подій і полегшує спільну роботу над одним твором. Текстові події можуть розташовуватися в будь-якому місці треку в якості слів пісні або коментаря. Lyric - це мета-подія, що визначає слова пісні, які повинні бути виконані у визначений час. Кожен склад повинен бути представлений окремою мета-подією Lyric з заданим часом виконання, склад також складається з ASGII-символів.

Впровадження інформації, в MIDI-файли можна робити зміною часу портаменто. Необхідною умовою роботи цього методу є відключений режим портаменто, щоб впроваджена інформацію не впливала на звучання музичного твору (не спотворювала його). Для впровадження інформації необхідно застосовувати подію зміни режиму керування з номером 5 і подію зміни режиму управління з номером 37, які визначають старшу і молодшу частину часу портаменто. У разі застосування цих подій для впровадження інформації, в їх значеннях будуть кодуватися старший і молодший півбайти символа.

Наявність текстової інформації в MIDI-файлі дозволяє скористатися прийомами текстової стеганографії для впровадження секретних повідомлень. З огляду на те, що дані текстові події не підтримують яке-небудь форматування, крім зміни шрифту всього тексту, а подія Lyric також не може містити більше пропусків після слів, найбільш прийнятним способом впровадження інформації є заміна українських букв схожими за зображенням буквами латинського алфавіту. Якщо в слові присутні букви, схожі з латинськими, і перша з них замінена на латинську, то в даному слові закодована логічна одиниця. Якщо в слові присутні букви, схожі з латинськими, і перша з них не замінена на латинську, то в даному слові приховано логічний нуль.

Для файлів формату MIDI, що містять послідовні записи подій, найбільш відповідним є ключ, який показує, наскільки подій вліво або вправо від поточного положення необхідно перемістити для зчитування півбайта (чотири біта) впровадженої інформації. Алгоритм добування інформації з події визначається типом самої події і її параметрами. Довжина ключа залежить від кількості подій в MIDI-файлі. Для того щоб ключ не мав обмежень по кількості подій, що визначають інтервал, між напівбайтами

впровадженої інформації, необхідно використовувати: запис числа, у вигляді величини змінної довжини. Такий підхід застосовується в SMF для кодування інтервалу часу між записаними подіями. Початкове число записується в кілька байт, по сім біт. У перший байт записується сім біт, тому що старший розряд визначає напрямок переміщенн по файлу (0- вперед, 1 - назад). В другому і подальших байтах перший біт позначає перший і останній байти в послідовності. Другий і наступні байти повинні містити в першому розряді одиницю, останній байт – нуль.

Література

1. Грибунин В.Г. Цифровая стеганография. /В.Г.Грибунин, И.Н.,Оков И.В., Туринцев // М.: СОЛОН-Пресс; 2002. - 261 с.
2. Конахович Т.Ф. Компьютерная стеганография / Т.Ф Конахович, А.Ю Пузыренко // Теория и практика. Киев: МК-Пресс, 2006. -288с.
3. Мамаев М. Технологии защиты информации:в Интернете/ Мамаев М., Петренко С. //: Специальный;справочник..СИб::Иитер 2002. -848 с.
4. Хайкин С. Нейронные сети. / Хайкин С. //Полный курс: пер. с англ. / 2-е изд. М.: Издательский дом «Вильяме», 2006. -1104 с.

Інтелектуальні технології автомобільного транспорту

Мурава В.М.
Науковий керівник – к.т.н., доц. Чорненький В.І.
Хмельницький національний університет

Ми живемо в століття високих технологій, і бум інтелектуалізації всього, чим користується людина, дістався і до авто. Сьогодні автомобільна промисловість розробляє не тільки кращі запчастини та комфортний дизайн, але і системи, що дозволяють машинам спілкуватися, самостійно планувати маршрут і берегти екологію.

Без автомобіля дійсно як без рук, особливо в мегаполісі. Людям, які тільки купують машини, варто відразу подумати про страхування, тим більше що страхових компаній дуже багато. Приміром, КАСКО в Москві пропонує маса агентств та фірм, а розрахувати вартість страхування КАСКО допоможе калькулятор, представлений на багатьох сайтах. Хоча звичні «Мерседеси», «Тойоти» і «BMW», як виявилося - минулий день. Машини майбутнього будуть працювати на сонячній та електроенергії, вже створюються корпусу машин з тканини і бамбука, і навіть з'являться екологічні «яйцемобілі». Схоже, що головний недолік новітніх авторазработок - сам водій, але і його виробники авто примудрилися усунути. Втім, про все по порядку ...

Почнемо, мабуть, з більш звичних суспільній свідомості технологій. Повним ходом йде розробка систем, що дозволяють автомобілям «спілкуватися» між собою. Передача даних між авто покликана убезпечити ситуації на дорозі. Однак попередження про аварійну ситуацію - тільки

невелика частина з того, на що здатні системи передачі даних, іменовані «Vehicle to vehicle», або V2V. Через кілька років автогіганти прогнозують ці системи всім авто, включаючи, в першу чергу, громадський транспорт. Вже сьогодні шкільні автобуси Америки використовують технологію GPS для більш безпечного водіння. Розробку веде «Car-2-Car» - консорціум Європи, учасниками стали наукові університети і такі виробники, як «FIAT», «Renault», «General Motors», «BMW» та інші. Цілю розробки є велике поле зору власника авто та інформованість про ситуацію на дорозі, незалежно від відстані і видимості. Можливо, незабаром, ми будемо знати, що чекає нас як за поворотом, так і за сотні кілометрів. Виробники прагнуть до єдиної системи для всього світу і будь-яких машин. Поки задіяні 3 технології - GPS, бездротові мережі та ПК для обробки інформації. Машини зможуть «впізнавати» один про одного все: місце, напрямок, швидкість, стан запчастин і багато іншого. Є навіть плани забезпечити аналогічною системою світлофори і знаки на дорогах, щоб ті «повідомляли» авто останні дорожні новини. Над системою працюють і психологи, щоб забезпечити найбільш ефективну форму «спілкування» між людиною і машиною. Поки рано говорити про точний час випуску V2V в маси, але відомо, що за ціною система буде доступна всім.

Серед останніх оригінальних розробок варто виділити компанію «Peugeot» і її «яйцемобіль». Ідея створити машину - крихітку виникла через величезну чисельності людей і дефіциту вільного місця. Це індивідуальне авто у формі яйця, на великих колесах. Управляється «яйцемобіль» джойстиком, мотори машини - у великих колесах, працюють на сонячній та електроенергії. Сам автомобіль нагадує дитячу машинку для атракціонів, але можливо, це новий крок в майбутнє.

В Німеччині відзначилися оригінальністю. Спроектований автомобіль, здатний обходитися без водія (рис.1). Машина «Люкс» скоро відправиться в Америку, де десятки «самостійних» авто влаштують рейд по міських дорогах. Лідером буде визнана найшвидша машина, не вчинили аварію, наїзди на тротуари і не збилися з курсу. Ідея такого конкурсу належить Пентагону, розробники призера отримають 2 000 000 доларів. Нові розробки підуть не тільки у воєнний автомобілебудування, а й у масове виробництво. Наприклад, лазерні «очі», що приймають самостійні рішення, і автоматична парковка. Лазерні датчики під фарами авто сканують шлях на сотні метрів вперед і ззаду. Управляє машиною комп'ютерний чіп, включаючи газ, гальмо, передачі швидкостей і управління кермом. Реакція чіпа - менше 100 мл. сек. Для рейду машинам необхідна швидкість від 30 миль / год. Перед гонкою в систему вносять карту місцевості і точки проїзду. Все інше машина зробить сама. Враження - немов за кермом водій-невидимка.

В Китаї створено «солнцемобиль». Компанія «Zhejiang 001 Group» випустила машину, повністю працює від енергії сонця. У даху машини перебувають сонячні панелі, акумулюючі енергію, завдяки чому авто їде. Правда, поки машина може витримати лише 150 км і не зрозуміло, що з нею робити, якщо вона застрягне в тунелі. Зате приваблива ціна - всього близько

п'яти з половиною тисяч доларів. Втім, є у світі і більш потужні «солнцемобили». Канадський «Xof1» - лідер по дальності, проїжджає більше 15 000 км. за 140 днів. Назва в перекладі на російську звучить, як «один в полі теж воїн». Дизайн авто - як у тарілки НЛО, на корпусі - сонячна батарея в 7 кв. м. Швидкість машини тільки від сонячної енергії - 70 км / год, з акумулятором - до 120 км / ч.

Високі технології торкнулися не тільки способу отримання енергії та інтелектуалізації авто, але й матеріалів для корпусу. Серед всіх «концепт-карів» відзначилася компанія «BMW», створивши автомобіль, з тканини, здатний змінювати форму корпусу. Автомобіль «GINA» зібраний з гнучкої водонепроникної тканини, натягнутої на залізний каркас. За рахунок чого кузов змінює форму під час поїздки, це створює вражаючий ефект. Звичайно, машина не для серійного випуску, але місце в музеї «BMW» вона заслужила. А в Японії створено автомобіль з бамбуковим корпусом «Bamgoo». Він, як і всі електромобілі, маленького розміру, і важить всього 60 кг. На одній зарядці машина «тягне» 50км .

Екологічні, незвичайні, «розумні» машини входять в моду. Автогіганти всерйоз зайняті оснащенням авто високими технологіями, адже попит на екологічно чисті машини зростає. Наприклад, в Лос - Анджелесі відкрився автосалон «еко-машин». Компанія «Ford» представила седани «Ford Fusion» і «Mercury Milan» з гібридними двигунами. Серійний випуск гібридних машин запланований компанією на 2010 рік, в кількості 50 000 на рік. Дебютував електромобіль «BMW Mini», серійний випуск якого також припаде на 2010 рік. Місце для салону вибрано неспроста - в Каліфорнії дуже популярні «екомобілі». Можливо, нові безпечні тенденції розвинуться по всьому світу, і незабаром автомобілі перестануть псувати міське повітря, а ризик на дорогах буде не більше, ніж при катанні на атракціонах.

Перспективи розвитку сучасних інформаційних технологій

к.т.н., доц. Огнева А.М.
Хмельницький національний університет

Сучасні інформаційні технології (ІТ) міцно увійшли в наше життя. Вони принесли в життя сучасного суспільства безліч корисних і цікавих речей. Програмні і апаратні засоби, технології і сервіси дозволяють щодня підвищувати ефективність роботи з інформацією,але тільки використання новітніх інноваційних технологій управління підприємством дозволить підвищити ефективність його діяльності, своєчасність прийняття управлінських рішень; дозволить здійснювати ефективний контроль за заборгованістю та ефективне управління коштами підприємства тощо.

На підприємств з великим обсягом інформації і різноманітними задачами діяльності у бізнес-процесах приймає участь велика кількість компонентів, які підтримують різні апаратні засоби і різне програмне забезпечення, що може фізично розміщуватися на віддалених територіях.

Тому надзвичайно важливою є задача єдиного комплексного управління ресурсами інтегрованої систем [2].

Стрімкий розвиток комп'ютерних, комунікаційних, мобільних та інформаційних систем спричинив виникнення хмарних технологій, які останнім часом впроваджуються на підприємствах України. Використання хмарних технологій у бізнесі є предметом дискусії серед науковців, ІТ - спеціалістів та підприємців.

Хмарні обчислення **(**cloud computing) - це технологія обробки інформації, при використанні якої забезпечується мережевий доступ на вимогу до загальних конфігурованим мережевим і обчислювальних ресурсів (мереж передачі даних, серверів, сховищ даних, додаткам і сервісам).

Основними елементами такої технології обробки інформації в даному контексті є віртуалізація, управління ІТ-інфраструктурою та послугами (замовлення, підтримка, білінг).

Поняття віртуалізація умовно можна розділити на дві фундаментальні категорії [1]:

– віртуалізація платформ. Продуктом цього виду є віртуальні машини - тобто програмні абстракції, що запускаються на платформі реальних апаратно-програмних систем;

– віртуалізація ресурсів. Даний вид має за мету комбінування або спрощення подання апаратних ресурсів для користувача й одержання якихось користувальницьких абстракцій устаткування, просторів імен, мереж тощо.

Якщо розглядати віртуалізацію в широкому змісті, можна прийти до поняття віртуалізації ресурсів, що узагальнюють у собі підходи до створення віртуальних систем та дозволяють концентрувати, абстрагувати й спрощувати керування групами ресурсів, таких як мережі, сховища даних і простору імен. Таким чином віртуалізація, це процес створення системи, що надає користувачеві зручний інтерфейс для роботи з нею та приховує складність цієї реалізації.

Отже хмарні технології дозволяють застосовувати розташовані поза підприємством обчислювальні потужності, обладнання, дисковий простір, додатки та інформаційні мережі. При цьому користувачі не повинні піклуватися про інфраструктуру, в тому числі про операційну систему, власне ПЗ, з якими вони працюють, а підприємству не потрібно створювати фізичну ІТ-структуру.

Хмарні технології забезпечують роботу користувачів з прикладними програмами на будь-яких клієнтських пристроях з різними операційними системами незалежно від їх територіального розташування. Більшість досліджень у сфері хмарних технологій стосуються їх сутності та технічної організації. Проблеми обробки облікових даних з використанням інформаційних систем, що працюють в умовах хмарних технологій вивчені недостатньо.

Ринок «хмарних» технологій стрімко росте. На ньому активно пропонуються послуги як для фізичних осіб, так і корпорацій. За прогнозами

компанії IDC середньорічний темп приросту світового ринку хмарних сервісів з 2012 року по 2016 рік складе 26,4%, що в п'ять разів перевищує темпи росту IT індустрії в цілому [2]. Поряд з цим, 90% українських компаній ще не користуються цією технологією, що пов'язано із недостатньою обізнаністю підприємців та ризиками, що можуть виникнути при впровадженні новітніх систем.

Використання хмарних технологій може працювати за сценаріями:

1) «хмара» всередині організації – сервер з інформаційною базою розташовується на підприємстві;

2) «хмара» всередині холдингу – сервер з інформаційною базою розташовується у центральному офісі холдингу;

3) «хмара» для клієнтів – сервер з інформаційною базою розташовується у постачальника;

4) модель сервісу – сервер з інформаційною базою розташовується у постачальника послуг з використання «хмарних» технологій [3].

При використанні підприємством інформаційних систем із застосуванням хмарних обчислень воно отримує наступні переваги: можливість доступу облікового персоналу до інформаційних ресурсів у будь-який час, з будь-якого пристрою, що має підключення до Інтернет, не залежно від територіального розташування; скорочення витрат на обслуговування інформаційної технології, за рахунок скорочення витрат на технічне та програмне забезпечення, утримання IT-спеціалістів, зменшення паперового та запровадження електронного документообігу; високий рівень безпеки та надійності збереження інформаційних даних за умови їх професійної організації; необмеженість обчислювальних ресурсів та збільшення їх потужності. [1]

Поряд з цим слід зазначити ризики, що виникають при впровадженні на підприємстві інформаційних систем на основі хмарних технологій: обмеженість програмного забезпечення обумовлена тим, що користувач має доступ лише до інформаційних систем, які розташовані в «хмарі», можливості налагодження ним цих систем обмежені; відсутність абсолютної конфіденційності; складність відновлення втрачених у «хмарі» інформаційних ресурсів; необхідність забезпечення постійного підключення до Інтернет з достатньою пропускною здатністю, що збільшує відповідні витрати; відсутність абсолютної безпеки та збереження інформаційних ресурсів.

Саме впровадження новітніх інформаційних систем і технологій на підприємствах може у результаті не лише призвести до зниження витрат на здійснення інформаційного обміну даними, підвищення оперативності інформаційної системи та запобігання втратам від помилок у звітності, а й стати наслідком, що спрямований на вдосконалення організації управління.

Література

1. Гаврилов, Л. П. Инновационные технологии в коммерции и бизнесе: учебник для бакалавров / Л. П. Гаврилов. - М. : Издательство Юрайт, 2016. - 372с

2. Кузьмінський Ю. Оцінка ефективності впровадження інформаційних технологій у бухгалтерський облік / Ю. Кузьмінський /// Бухгалтерський облік і аудит: Всеукраїнський щомісячний науково-практичний журнал. – 2011. – №7. – С. 27-31.
3. Яковицький І.Л. Технологія «хмарних обчислень» як інструмент створення інфраструктури управління / І.Л. Яковицький // Комунальне господарство міст, 2012.

Моделі взаємодії технології «Клієнт-сервер»

к.т.н., доц. Огнєвий О.В.
Хмельницький національний університет

У своєму розвитку технології «клієнт-сервер» пройшли кілька етапів, тому є різні моделі технології. Їх реалізація заснована на поділі структури СКБД на три компоненти:

– введення і відображення даних (інтерфейс з користувачем);

– прикладний компонент (запити, події, правила, процедури та функції, які характерні для даної предметної області);

– функції керування ресурсами (файловою системою, базою даних тощо).

Тому, в будь-якому додатку виділяються наступні компоненти:

– компонент подання даних;

– прикладний компонент;

– компонент керування ресурсом.

Зв'язок між компонентами здійснюється за певними правилами, які називають «протокол взаємодії».

Існують різні класифікації, але однією з найпоширеніших є використання чотирьох моделей технології «Клієнт-сервер»:

1. Модель файлового серверу (File Server - FS) (рис. 1а).
2. Модель віддаленого доступу до даних (Remote Data Access - RDA) (рис. 1 б).
3. Модель сервера БД (Data Base Server - DBS) (рис. 1 в).
4. Модель сервера додатків (Application Server - AS) (рис. 1г).

Модель файлового серверу**.** В задачах обробки інформації, заснованих на системах баз даних, існують два варіанти розташування даних: локальний і віддалений. У першому випадку говорять про доступ до локальних даних, у другому - про доступ до віддалених даних. Локальні дані, як правило, розташовуються на жорсткому диску комп'ютера, на якому працює користувач, і знаходяться в монопольному керуванні цього користувача. Користувач при цьому працює автономно, не залежачи від інших користувачів та жодним чином не впливаючи на їх роботу. Дистанційні дані розташовуються поза комп'ютера користувача (користувачів) - на файловому сервері мережі або на спеціально виділеному для цих цілей комп'ютері.

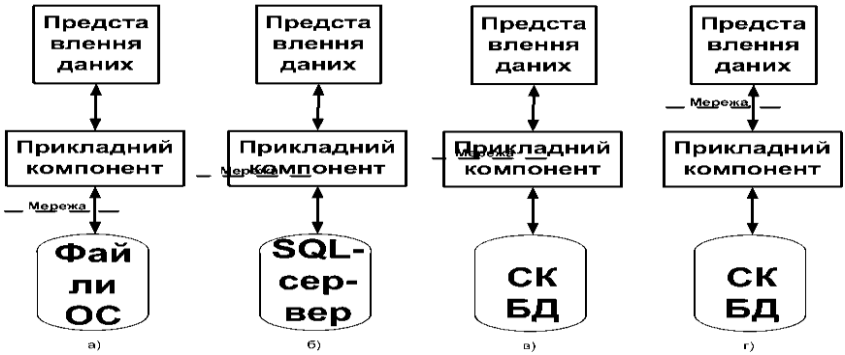


Рисунок 1 - Моделі технології «Клієнт-сервер»: а) FS; б) RDA; в) DBS; г) AS

Модель файлового серверу (File Server - FS*)* – це природне розширення персональних СКБД для підтримки багатокористувацького режиму/

Особливості FS-моделі:

– всі основні компоненти розміщуються на клієнтському комп'ютері;

– модель характеризує не стільки спосіб створення ІС, скільки загальний спосіб взаємодії комп'ютерів в локальній мережі;

– один з комп'ютерів виділяється і визначається файловим сервером, тобто загальним сховищем будь-яких даних;

– сервер виконує чисто пасивну функцію;

– дуже проста та зрозуміла модель.

На основі моделі файлового сервера функціонують такі популярні СКБД як FoxPro (Microsoft), dBase (Borland), CF-Clipper (Computer Associates International), Paradox (Borland) тощо. СКБД розглянутого класу коштують недорого, прості в установці та освоєнні. Також відсутні високі вимоги до продуктивності сервера та програмні компоненти СКБД не розподілені.

Модель віддаленого доступу до даних**.** Модель віддаленого доступу до даних (Remote Data Access – RDA) – архітектура, яка заснована на обліку специфіки розміщення і фізичного маніпулювання даними у зовнішній пам'яті для реляційних СКБД

У RDA-моделі для обробки даних виділяється спеціальне ядро - так званий SQL-сервер, який приймає на себе функції обробки запитів користувачів, іменованих тепер клієнтами. Сервер баз даних являє собою програму, яка виконується, як правило, на потужному комп'ютері. Додатки-клієнти посилають з робочих станцій запити на вибірку (вставку, оновлення, видалення) даних. При цьому сервер виконує всю «брудну» роботу з відбору даних, відправляючи клієнтові тільки необхідну «вичавлювання». Якщо наведений вище приклад перебудувати з урахуванням клієнт-серверної архітектури, то додаток-клієнт «отримає» від сервера в якості результату список тільки тих працівників, які беруть участь у заданому проекті, і не більше того!Такий підхід забезпечує вирішення трьох важливих завдань:

– зменшення навантаження на мережу;

– зменшення вимог до комп'ютерів-клієнтам;

– підвищення надійності та збереження логічної цілісності бази даних.

Тут додатки також виконуються, в основному, на робочих станціях. Додаток включає модулі для організації діалогу з користувачем і бізнес-правила (транзакції). Ядро СКБД є загальним для всіх робочих станцій і функціонує на сервері. Оператори звернення до СКБД (SQL-оператори), закодовані в транзакції, не виконуються на робочій станції, а пересилаються для обробки на сервер. Ядро СКБД транслює запит і виконує його, звертаючись для цього до індексів та інших проміжних даних. Назад на робочу станцію передаються тільки результати обробки оператора.

У файлах БД на сервері знаходиться також і системний каталог БД. У числі іншого, в каталог БД поміщаються:

– відомості про зареєстрованих користувачів;

– привілеї користувачів;

На клієнтських комп'ютерах встановлюються частини СКБД, які реалізують інтерфейсні функції та прикладні функції.

Прикладний компонент включає бібліотеки запитів та процедури обробки даних. Він повністю розміщується і виконується на клієнтській частині - формує SQL-інструкції, що направляються SQL-серверу.

Модель сервера бази даних**.** Для поліпшення попередньої RDA-моделі в сучасних СКБД використовується модель сервера баз даних (DataBase Server - DBS*)*, в якій на сервері можуть запускатися так звані збережені процедури і тригери, які разом з ядром СКБД утворюють сервер бази даних

До збережених процедур можна звертатися з додатків на робочих станціях. Це дозволяє скоротити розмір коду прикладної програми і зменшити потік SQL-операторів з робочої станції, так як групу необхідних SQL-запитів можна закодувати в збереженій процедурі. Тригери - це програми, які виконуються ядром СКБД перед або після оновлення (UPDATE, INSERT, DELETE) таблиці бази даних. Вони дозволяють автоматично підтримувати цілісність бази даних.

Модель сервера бази даних (БД) підтримують наступні СКБД: Oracle, DB2 (IBM), MS SQL Server (Microsoft), MySQL (Oracle), FireBird, PostgreSQL, Sybase (SAP), тощо. Причому на перші чотири СКБД припадають понад 85% ринку.

Модель сервера додатків**.** Щоб рознести вимоги до обчислювальних ресурсів сервера у відношенні швидкодії і пам'яті за різними машинам, використовується модель сервера додатків (Application Server AS*).*). AS-модель зберігає сильні сторони DBS-моделі.

Особливості AS-моделі:

– перенесення прикладного компонента АІС на спеціалізований сервер;

– на клієнтських машинах - тільки інтерфейсна частина системи;

– виклики функцій обробки даних спрямовуються на сервер додатків;

– низькорівневі операції з даними виконує SQL-сервер.

Використання цієї моделі дозволяє розвантажити робочі станції, тобто перейти до «тонких» клієнтів. Звичайно, сервер додатків можна організувати і за допомогою збережених процедур. Але для реалізації збережених процедур використовують мови високого рівня (наприклад, в Oracle - мову PL/SQL), тому програми виходять ресурсоємними. Причому можливості цих мов дуже широкі: від циклів до обробки даних на рівні бітів. Збережені процедури також не підтримують розподілені додатки, тобто вони не забезпечують автоматичний запуск необхідної програми на іншому сервері.

Сучасні СКБД використовують наступні процедурні мови:

– Transact-SQL (MS SQL Server, Sybase);

– PL/SQL (Oracle);

– PSQL (FireBird);

– SQL PL (IBM DB2);

– JetSQL (MS Access);

– PL/pgSQL (PostgreSQL).

У тому випадку, коли інформаційна система об'єднує досить велику кількість різних інформаційних ресурсів і серверів додатків, постає питання про оптимальне управління всіма її компонентами. Для цього використовують програмні засоби, які часто називають менеджерами транзакцій, моніторами транзакцій (Transaction Processing Monitor - TPM), і які розміщуються на сервері додатків. Також менеджер транзакцій і додатки можуть запускатися на одному комп'ютері (хоча це не є обов'язковим), щоб зменшити потік SQL-запитів по мережі.

Транзакція - це послідовна сукупність операцій над даними (SQL-інструкцій), що має окреме смислове значення. Але часто в моніторах транзакцій поняття транзакції розширюється - в даному випадку це не атомарна дія над базою даних, а будь-яка дія в системі – видача повідомлення, запис в індексний файл, друк звіту тощо.

Для спілкування прикладної програми з монітором транзакцій використовується спеціалізований API (Application Program Interface - інтерфейс прикладного програмування), який реалізується у вигляді бібліотеки, що містить виклики основних функцій (встановити з'єднання, викликати певний сервіс тощо). Сервери додатків (сервіси) також створюються за допомогою цього API, кожному сервісу присвоюється унікальна назва. Монітор транзакцій, отримавши запит від прикладної програми, передає її виклик відповідному сервісу (якщо той не запущений, породжується необхідний процес), після обробки запиту сервером додатків повертає результати клієнту. Для взаємодії моніторів транзакцій з серверами баз даних розроблений протокол XA. Наявність такого уніфікованого інтерфейсу дозволяє використовувати в рамках однієї програми кілька різних СКБД.

Монітор транзакцій усуває такі витрати спільної обробки:

– втрачені зміни - дві транзакції одночасно змінюють один об'єкт БД;

– брудні дані - одна транзакція змінює об'єкт друга читає дані з нього;

– неповторювані читання - одна читає об'єкт, а друга транзакція змінює його.

Для ізоляції транзакцій і подолання ситуацій неузгодженої обробки даних використовують серіалізацію транзакцій - виконання транзакцій таким чином, щоб результат їхньої спільного виконання був еквівалентний результату їх послідовного виконання.

Використання моніторів транзакцій у великих системах дає наступні переваги:

– концентрація всіх прикладних функцій на сервері додатків забезпечує значну незалежність як від реалізації інтерфейсу з користувачем, так і від конкретного способу керування ресурсами. При цьому також забезпечується централізоване адміністрування додатків, оскільки всі додатки знаходяться в одному місці, а не «розмазані» по мережі на клієнтських робочих місцях;

– монітор транзакцій в змозі сам запускати і зупиняти сервери додатків. В залежності від завантаження мережі та обчислювальних ресурсів він може перенести або скопіювати частину серверних процесів на інші вузли. Це забезпечує балансування навантаженням серверів;

– забезпечується динамічна конфігурація системи, тобто без її зупинки може бути доданий новий сервер ресурсів або сервер додатків;

– підвищується надійність системи, тому в разі збоїв сервер додатків може бути переміщений на резервний комп'ютер;

– з'являється можливість керування розподіленими базами даних.

Література

1. Андон Ф., Методы инженерии распределенных компьютерных систем./ Ф. Андон., М К Лаврищева. – К.: Техника, 1997. – 168 с.
2. Куроуз Дж., Росс К. Компьютерные сети./ Дж.Куроуз ,К. Росс. - СПб.: Питер, 2004.

Метод оптимізації інформаційного трафіку в телекомунікаційних системах з врахуванням якості послуг передачі

Рибалка А.В.
Науковий керівник – к.т.н., доц. Хмельницький Ю. В.
Хмельницький національний університет

Відповідно до рекомендації , якість послуг, це сукупність специфічних параметрів, що відносяться до внутрішньої структури мережі та обумовлюють якість її роботи і характеризують споживчі властивості послуги в термінах, зрозумілих користувачам. Уся сукупність характеристик якості обслуговування і функціонування інформаційного трафіку в телекомунікаційних системах поділяється на дві категорії:

– первинні - визначені шляхом прямого спостереження в точці доступу до послуги і відносяться до певного моменту часу;

– похідні - визначенні на підставі одного або декількох первинних атрибутів або усереднені за деякий інтервал часу.

Керуючі впливи системи керування по забезпеченню Q o S зводяться до структурування інформаційних потоків і визначенню алгоритмів обробки кожного виду трафіку у вузлах комутації. Розподіл ресурсів вузлів комутації виробляється за допомогою різних алгоритмів використання буферу пам'яті і призначення пріоритетів для організації черг пакетів і визначення порядку їхньої обробки. У середовищі IP можна реалізувати два механізми Q o S :

– абсолютний механізм Q o S, що резервує ресурси вузлів мережевої інфраструктури і гарантує відповідність параметрів трафіку заданій якості;

– релевантний (порівняльний) механізм Q o S**,** заснований на структуруванні інформаційних потоків за допомогою пріоритетів, що дозволяють складати інформаційні потоки з близькими вимогами по якості в обмежений набір класів, що ділять мережеві ресурси між собою відповідно до призначених їм пріоритетів.

Для IP середовища існують дві основні моделі організації Q o S при маршрутизації трафіку в телекомунікаційній мережі: Integrated Services (Int-Serv) та Differentiated Services (Diff-Serv). Int-Serv використовує протокол Resource Reservation Protocol (RSVP), який є протоколом сигналізації, що може використовуватися в режимі інкапсуляції в заголовок - UDP або TCP для резервування мережевих ресурсів між RSVP вузлами. Diff-Serv визначає метод структурування трафіка телекомунікаційної мережі по класах сервісу C o S з відповідними пріоритетами. Він використовує поле типу сервісу в заголовку IP-пакета для визначення алгоритму його транспортування від вузла до вузла мережі. Така схема, на відміну від Int-Serv, не вимагає витрат часу на попереднє визначення параметрів транспортування трафіка по маршруту проходження інформаційного потоку. Механізми Int-Serv забезпечують необхідну якість обслуговування для інформаційного трафіку в телекомунікаційних системах всіх додатків реального часу, проте вимагають більшого обсягу буфера і смуги пропускання, ніж механізми диференціального сервісу Diff-Serv. Таким чином, більш ефективним методом транспортування даних є механізми диференціального сервісу зі структуруванням трафіка по класах сервісу C o S і відповідними пріоритетами. Задача отримання заданої якості послуг та оптимальних параметрів телекомунікаційної мережі до кінця ще не вирішена оскільки динамічний характер мультимедійних послуг ускладнює можливість прогнозування необхідних мережевих ресурсів для забезпечення Q o S.

При досліджені моделей оптимізації інформаційного трафіку на базі якості послуг в телекомунікаційних мережах було визначено три основні стадії надання послуги, якість виконання яких дає сумарну якість послуги: доступ до передачі даних (з'єднання), передача даних і завершення з'єднання. Кожна із цих частин характеризується трьома основними показниками, до яких відносяться: швидкість (встановлення з'єднання, передачі даних, завершення з'єднання) $V = (V_1,...,V_k)$; точність встановлення з'єднання із зазначеними параметрами Q; гарантованість встановлення з'єднання P, тобто:

$$< V, Q, P > \rightarrow (QoS) \cdot \qquad (1)$$

Відомо визначення характеристики оцінки надання послуг з врахуванням функцій семи рівній моделі. Якість послуги фізичного рівня характеризується наступними факторами: доступністю послуги p_i; коефіцієнтом виникнення помилок *BER*; пропускною здатністю *C*; затримками передачі $T_{зат}$:

$$QoS_{фіз} =< p_i, BER, C, T_{зат} > \cdot \qquad (2)$$

Для канального рівня телекомунікаційних мереж:

$$QoS_{кан} =< V, BER, T_{транз}, A, R_{пріор} >, \qquad (3)$$

де *V* – швидкість передачі даних; *BER* – коефіцієнт помилок; $T_{транз}$ – транзитні затримки; *A* – коефіцієнт захищеності, $R_{пріор}$ – пріоритетність якості обслуговування.

Параметри мережевого рівня виражають узагальнені мережні показники:

$$QoS_{мер} =< T, P, C, E_з >, \qquad (4)$$

де $T = T_{зат} + T_{транз} + T_{розр}$, $P =< P_{н.встан}, P_{втрат}, P_{н.розр} >$, $T_{зат}$ – затримка встановлення мережевого з'єднання; $P_{н.встан}$ – імовірність невдалого встановлення мережевого з'єднання; *C* – пропускна здатність; $T_{транз}$ – транзитні затримки; $P_{втрат}$ – імовірність невдалої передачі інформації; $E_з$ – коефіцієнт залишкової помилки; $T_{розр}$ – затримка завершення мережевого з'єднання; $P_{н.розр}$ – імовірність відмови завершення транспортного з'єднання.

Послугу транспортного рівня характеризуємо:

$$QoS_{тр} =< T, P, C, BER, A, R_{пріор} >, \qquad (5)$$

де $T = T_{зат} + T_{транз} + T_{розр}$, $P =< P_{н.встан}, P_{втрат}, P_{н.розр} >$, $T_{зат}$ – затримка встановлення транспортного з'єднання; $P_{н.встан}$ – імовірність невдалого завершення процедури встановлення транспортного з'єднання; *C* – пропускна здатність; $T_{транз}$ – транзитні затримки; *BER* – коефіцієнт помилок; $P_{втрат}$ – імовірність невдалої передачі інформації; $T_{розр}$ – затримка завершення транспортного з'єднання; $P_{н.розр}$ – імовірність відмови завершення транспортного з'єднання; *A* – коефіцієнт захищеності; $R_{пріор}$ – пріоритетність якості обслуговування.

Для телекомунікаційних мереж передачі даних узагальнено:

$$QoS_{мід} =< T, P, V_{аб} >, \qquad (6)$$

де $T = T_{дост} + T_{зат} + T_{розр} + T_{відс}$

$P =< P_{н.дост}, P_{відм.дост}, P_{помил}, P_{д.дост}, P_{втрат}, P_{н.розр} > \cdot$

Таким чином, сформовані мережеві показники і метод оптимізації досягнення якості послуг телекомунікаційних мережах в залежності від:

а) виду інформаційного трафіку;

б) застосовуваних протоколів передачі в мережах;

в) виду телекомунікаційних мереж.

Методи, алгоритми та принципи передачі даних по каналах телекомунікації

Філіпчук О.А.
Науковий керівник – к.т.н., доц. Хмельницький Ю. В.
Хмельницький національний університет

Дослідження та аналіз моделей передачі даних в телекомунікаційних мережах показує, що вони залежать від багатьох чинників, тому в роботі використана класифікація потрібних для реалізації системи управління інфраструктурою моделей і алгоритмів з урахуванням таких чинників як ознак класифікації. Потрібні моделі визначаються комбінаціями зазначених параметрів. Перша ознака передбачає відмінність моделей в залежності від цілей управління інфраструктурою телекомунікаційною мережею для підтримки процесів чи надання послуг зовнішнім відносно даної організації клієнтам мережі. В першу чергу такий поділ впливає на вид критерію, який використовується у відповідній моделі. Другою ознакою є технологічні особливості інфраструктури телекомунікаційної мережі, які обумовлені архітектурою її побудови. Ці чинники впливають на всі елементи математичної моделі: критерій, обмеження, тип змінної. В залежності від етапу життєвого циклу – це третя ознака, на якому знаходиться сервіс виникають ті чи інші задачі. При цьому на етапі планування крім технологічних і ресурсних обмежень можуть використовуватись також і інші види обмежень, наприклад обмеження вартості чи обмеження по надійності. Рівень абстракції ресурсів – це четверта ознака, впливає на складність математичної моделі задачі, що вирішується. Задачі великої розмірності пропонується вирішувати у два етапи: на першому етапі здійснювати розподіл абстрактних або узагальнених ресурсів кожного типу без прив'язки до їх конкретного місцезнаходження, з уточненням отриманих результатів на другому етапі. Також суттєво впливає на вид моделей остання ознака – забезпечення ресурсами, тобто, чи дозволяється часткова підтримка сервісів, чи вони мають бути підтримані у повному обсязі або не підтримані зовсім. Згідно з класифікацією сформулювало 128 класів моделей.

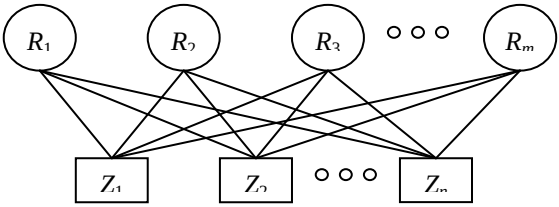


Рисунок 1 - Модель розподілу обмежених ресурсів інфраструктури телекомунікаційної мережі

У загальному вигляді кожна із моделей складається з критерію, який потрібно мінімізувати чи максимізувати, ресурсних обмежень, технологічних та інших обмежень. У найпростішому випадку дворівневої клієнт - серверної архітектури модель розподілу обмежених ресурсів може бути представлена у вигляді декількох процесів, що безпосередньо використовують частину одного чи декількох незалежних один від одного ресурсів (рис. 1.1).

Для побудови моделей використовуються наступні позначення:

1) $Z_1, ..., Z_n$ – процеси, підтримку яких забезпечує функціонування інфраструктури телекомунікаційної мережі;

2) $W = (w_1, ..., w_n)$ – вектор коефіцієнтів важливості процесів $Z_1,..., Z_n$ відповідно;

3) $R_1, ..., R_m$ – інтегровані ресурси інфраструктури телекомунікаційної мережі, необхідні для підтримки функціонування процесів;

4) $P = \|p_{ij}\|$ – матриця потреб процесів у ресурсах інфраструктури телекомунікаційної мережі, де p_{ij} відповідає приведеній кількості потрібного для процесу Z_i ресурсу R_j чи 0, якщо ресурс не потрібен;

5) $D = \|d_{ij}\|$ – матриця наявності потреб процесів у ресурсах інфраструктури телекомунікаційної мережі:

$$\begin{cases} d_{ij} = 1, \text{ якщо } p_{ij} > 0, \\ d_{ij} = 0, \text{ якщо } p_{ij} = 0. \end{cases} \quad (1)$$

6) $R = (r_1,..., r_m)$ – вектор існуючих обмежень на ресурси.

Розглянемо дискретний випадок, коли процес або підтримується в повному обсязі, або повністю блокується.

Введемо змінну $X = (x_1,..., x_n)$:

$$x_i = \begin{cases} 1, \text{ якщо процес } Z_i \text{ обслуговується}, \\ 0, \text{ в протилежному випадку}. \end{cases} \quad (2)$$

Тоді критерій оптимального розподілу ресурсів інфраструктури телекомунікаційної мережі можна представити у вигляді:

$$\max \sum_{i=1}^{n} x_i \cdot w_i \quad (3)$$

при виконанні ресурсних обмежень:

$$\sum_{i=1}^{n} x_i \cdot p_{ij} \le r_j, \quad j = 1, ..., m. \quad (4)$$

Наведені вище моделі, а також інші моделі, які розглядаються належать до лінійних або нелінійних, неперервних та змішаних задач математичного програмування, які до того ж мають стохастичні аналоги. Для детермінованих задач можна використовувати евристичні алгоритми, методи м'яких обчислень (штучного інтелекту) та точні методи, використання яких обмежено з огляду на розміри задач. Так для задач лінійного програмування можна використовувати відомі методи (метод внутрішньої точки). Задачі

програмування можна вирішувати методами часткового перебору, наприклад послідовного аналізу варіантів.

Використання точних методів дає можливість знайти найкраще рішення, але їх використання можливе лише для задач обмеженої розмірності, оскільки час пошуку рішень суттєво зростає при збільшенні складності задачі управління інфраструктурою телекомунікаційної мережі. На відміну від них евристичні методи та методи штучного інтелекту, генетичні алгоритми, дозволяють знайти задовільне рішення за досить короткий час навіть для задач великої розмірності. До того ж їх ефективність можна значно поліпшити за рахунок врахування особливості задачі. В роботі також були проаналізовані різні варіанти алгоритмів, і на їх основі був більш розглянутий керований генетичний алгоритм, який дозволяє ефективно керувати збіжністю в процесі його роботи.

Метод встановлення відповідності між окремими об'єктами на основі системи правил і ваг

Хлистун І.А.
Науковий керівник – к.т.н., доц. Джулій В.М.
Хмельницький національний університет

Основним чинником, що стимулює розвиток технологій пошуку, є поява великої кількості електронних бібліотек і архівів, що містять значні обсяги актуальних знань. Продуктивність і ефективність будь-якої системи зберігання інформації безпосередньо залежить від ефективності та продуктивності пошукових систем. Саме пошукова система визначає, чи перетворяться в знання численні розрізнені дані, що надходять по різних каналах зв'язку і накопичуються в різноманітних базах даних та електронних архівах.

При організації пошуку в базах персональних даних клієнтів виникають характерні проблеми, пов'язані з наявністю в запитах орфографічних і фонетичних помилок, помилок введення інформації, а також відсутністю єдиних стандартів транскрипції з іноземних мов. Внаслідок цього задача пошуку в базах персональних даних не може бути повною мірою вирішена тільки методами перевірки на точну відповідність. Стає актуальною задача розробки спеціальних методів і технологій текстового пошуку з використанням нетривіальних рішень, в тому числі на основі операцій не суворої відповідності. Проведений аналіз напрямків розвитку сучасних баз даних показує, що склалися і формуються за останні роки тенденції розвитку інформаційних технологій істотно впливають, у тому числі і на функціональні можливості автоматизованих систем. Задача встановлення відповідності між окремими об'єктами - побудова процедур ототожнення в даний час не має задовільного рішення. Існуючі роботи, присвячені інтеграції БД, дозволяють здійснити тільки інтеграцію схем БД, але не пропонують способів побудови процедур ототожнення. Побудова

процедур ототожнення ускладнюється відсутністю серед загальних атрибутів відповідних один одному таблиць різних БД первинних ключів і наявністю помилок операторського введення.

З урахуванням специфіки роботи з персональними даними пропонується вирішення наступних прикладних задач: повна ідентифікація клієнта при наявності спотворень інформації в базі даних або в пошукових запитах; усунення дублікатів записів при надходженні до БД з множинних джерел зі слабоструктурованою інформацією; пошук і коректування помилок в персональних даних клієнтів (фізичних і юридичних осіб).

Однією з задач при обміні інформацією про клієнта, є його однозначна ідентифікація. Одним, з таких рішень є ідентифікація фізичних осіб шляхом порівняння їх основних реквізитів. Таке рішення не завжди буде прийнятне при використанні, простого порівняння реквізитів, тому що по ряду причин, реквізити однієї і тієї ж особи, взяті з двох різних БД можуть не співпадати, тому що вони не завжди доступні (наприклад, через помилки в джерелі даних); присвоєні не всім категоріям громадян (ідентифікаційний код, страхове свідоцтво пенсійного фонду); реквізити документа змінені. внаслідок втрати/псування/за бажанням громадянина; реквізити документа відрізняються внаслідок помилки при занесенні в БД. Готової методики по такому виду ідентифікації на даний час не існує. На основі проведених досліджень і експериментів було знайдено рішення, що дозволяє проводити ідентифікацію фізичних осіб в БД з максимальною точністю. В результаті була розроблена технологія, із застосуванням якої може бути організований більш ефективний інформаційний обмін. Укрупнений алгоритм даного підходу складається з трьох основних блоків: формування масиву «подібних» людей; використання не суворої відповідності серед масиву «подібних» людей; відпрацювання виняткових ситуацій.

Розглянемо основні поняття що використовуються даним методом.

Вага - умовний коефіцієнт реквізиту. Він залежить від повноти, достовірності, і актуальності реквізиту. Вага визначає значимість реквізиту для ідентифікації

Правило - поєднання реквізитів клієнта, за якими здійснюється пошук. Механізм пошуку за правилами такий, що при пошуку клієнта порівнюються тільки ті реквізити, які вказані в правилах. Наприклад, при використанні правила 1 (табл.. 1) порівнюються тільки «Прізвище», «Ім'я» і «Дата народження», при цьому не враховуються інші реквізити. Для кожного правила визначається його сумарна «вага», яка складається з суми «ваг» реквізитів, що входять у дане правило (табл. 1).

Розглянемо перший блок алгоритму – формування масиву «подібних» людей, який наповнюється з використанням правил. Для вибору єдиної вірної людини з масиву «подібних» людей, встановлюється поріг ідентифікації. Поріг ідентифікації необхідний для того, щоб виключити людину, яка не задовольняє умовам. Поріг ідентифікації встановлюється, як показано в табл.1.

Таблиця 1 - Ваги правил

№ правила	Реквізити	Сумарна вага реквізитів	Поріг ідентифікації по правилу
Правило 1	Прізвище (5) + Ім'я (4) + По батькові (3)	12	11
Правило 2	Прізвище (5) + Ім'я (4) + Дата народження (4)	13	12
Правило 3	Ім'я (4) + По батькові (3)+ Дата народження (4)	11	10
Правило 4	Дата народження (4) + По батькові (3) + Місце народження (3)	10	9

Якщо поріг подолали більше однієї особи, то автоматизовано ідентифікувати особу неможливо. Така ситуація відпрацьовується оператором. Наступним кроком технології є вибір людини із застосуванням функції релевантності. Якщо поріг пройшли декілька записів, то автоматизовано ідентифікувати особу неможливо. Така ситуація відпрацьовується оператором. Дані записи надходять на розгляд аналітику, в іншому випадку записи вважаються різними і порівняння триває. Для визначення кількості помилок, що усуваються із застосуванням не суворої відповідності проведено розрахунок:

$$P(A) = \frac{m}{n} \qquad (1)$$

де m - кількість реквізитів з помилками, n - загальне число реквізитів, $P(A)$ - частота появи помилки в реквізиті.

Отриманий експериментально відсоток помилок може варіюватися в обидві сторони і в широких межах, але на практиці не було відмічено випадків, коли в імпортованій з різних баз даних - джерел інформації відсутні помилки. Це пов'язано з наявністю людського фактора при обробці великих масивів даних по фізичним особам: друкарські помилки, помилки введення, нечіткі копії первинних документів та інші випадки.

Можливі результати роботи алгоритму: людина знайдена (це точно вона - співпали всі правила); людини не знайдено (її точно немає); знайдено кілька схожих людей, автоматизовано визначити не можливо, вимагає відпрацювання оператором. Ситуації, які вимагають доопрацювання оператором, потрапляють в лог прийняття рішень. Відпрацювання логу здійснюється шляхом повторного звернення до джерела даних, первинних документів, особової справи і т.д.

Задачу пошуку за окремими атрибутами необхідно розбити на 3 підзадачі: пошук по рядках довжиною більше 50 символів; пошук по відносно коротким полях з середньою довжиною менше 50 символів; пошук

за прізвищами при ручному введенні реквізитів клієнтів або отриманням «на льоту» списку відповідних записів про клієнтів кредитним інспектором.

Спеціально розроблений алгоритм наближеного пошуку застосовується до першої задачі. Алгоритм пошуку за окремими атрибутами (рис.1) полягає в наступному. Якщо дані про клієнта вводяться безпосередньо співробітником і включений блок не суворого пошуку, то поки вводяться інші дані, блок Metaphone аналізує прізвище, введене в спеціальне поле, і формується попередній масив подібних прізвищ клієнтів існуючих в базі даних. По мірі заповнення форми інформації про клієнта, даний масив аналізується з використанням функції релевантності та урізається шляхом відкидання записів, що не пройшли поріг ідентифікації. Запропонований метод встановлення відповідності між окремими об'єктами на основі системи правил і ваг і усунення дублювання інформації, відрізняється побудовою функції релевантності, дозволяє знаходити рішення в найбільш загальному вигляді.

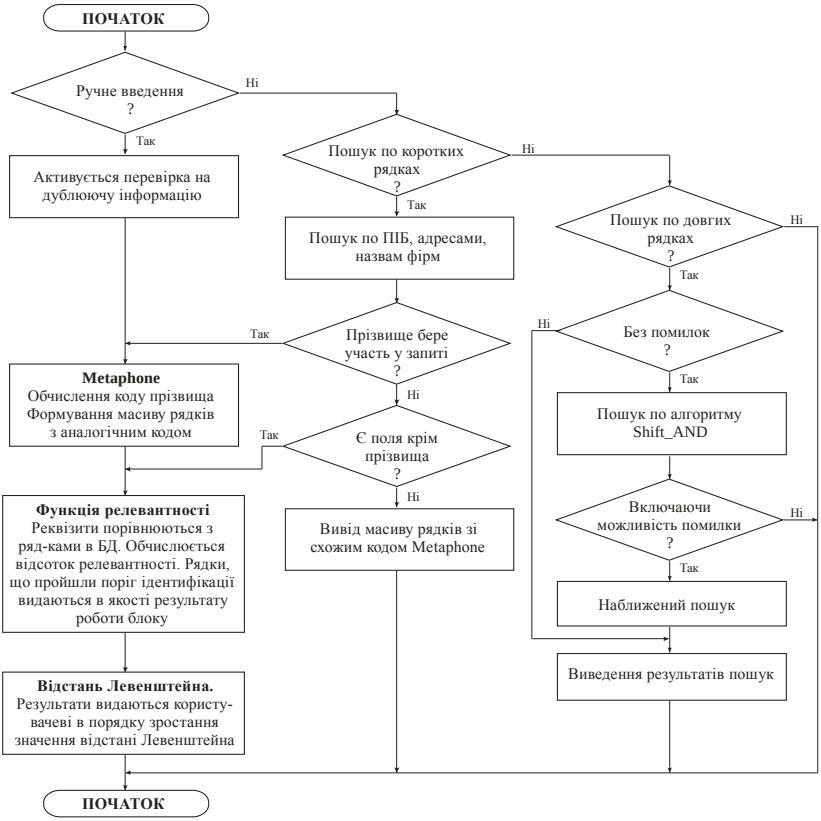


Рисунок 1 - Алгоритм пошуку за окремими атрибутами

Література
1. Ахо А. Структуры данных и алгоритмы./ А. Ахо, Д. Хопкрофт, Д..Ульман //—М.: Вильяме, 2009.- 400 с.
2.Вирт Н. Алгоритмы и структуры даннях / Н. Вирт// - М.: ДМК Пресс, 2010, - 272 с.
3. Гагарина Л.Г./ Разработка и эксплуатация автоматизированных информационных систем/ Л.Г. Гагарина, Д.В.,Киселев Е.Л.Федотова //: учеб. пособие. М.: ИД «Форум»: Инфа-М, 2007. 384 с.

Дослідження проблемних аспектів в роботі засобів обробки кодів Хеммінга

к.т.н., доц. Чешун В.М.
Хмельницький національний університет

Найбільш простою реалізацією завадостійкого кодування є паритетні коди. Прикладом паритетних кодів можуть служити коди з кодуванням за парністю, коди з кодуванням за непарністю, коди Хеммінга.

При використанні кодів з кодуванням за парністю контрольний розряд додається до коду таким чином, щоб загальна кількість одиниць в кодовій комбінації була парною.

Кодування за непарністю є зворотнім відносно кодування за парністю. При використанні кодів з кодуванням за непарністю контрольний розряд до коду додається таким чином, щоб загальна кількість одиниць в кодовій комбінації була непарною.

Недоліком методів кодування за парністю та за непарністю окремих повідомлень є низька перевіряюча здатність цих методів кодування – вони здатні виявляти лише непарну кількість помилок (1,3,5...) і не орієнтовані на виправлення помсилок.

Подальшим розвитком паритетних кодів став код Хеммінга, призначений для корегування одиночних помилок. Найсуттєвішою перевагою методики Хеммінга є те, що, у випадку виникнення помилки в закодованому повідомленні, отримана в процесі декодування контрольна кодова комбінація безпосередньо вказує номер позиції помилкового розряду (незалежно від того, інформаційний це розряд чи доданий контрольний). Цей фактор в сукупності з простотою реалізації засобів кодування та декодування коду Хеммінга дозволяє розглядати запропоновану Хеммінгом методику як визначне досягнення в області завадостійкого кодування двійкових повідомлень.

Як і при кодуванні за парністю, значення контрольних розрядів для коду Хеммінга визначається таким чином, щоб сума одиниць в інформаційних розрядах та доданому контрольному розряді була парною, але при цьому код Хеммінга має ряд суттєвих відмінностей, а саме:

- кількість контрольних розрядів коду Хеммінга залежить від розрядності інформаційного повідомлення, що кодується;

- контрольні розряди коду Хеммінга розташовуються на фіксованих позиціях серед інформаційних розрядів;

- при визначенні значення певного контрольного розряду аналізуються на парність не всі інформаційні розряди, а лише окремі розряди, які займають визначені позиції в кодовому слові.

Першим питанням, яке постає при формуванні коду Хеммінга, є питання визначення необхідної кількості контрольних розрядів та їх позицій в повідомленні, що кодується. Для визначення необхідної кількості контрольних розрядів можна скористатися формулами оцінки перевіряючої та корегуючої здатності, але їх кількість може бути також визначена за наведеною нижче методикою, яка враховує особливості побудови коду Хеммінга і дозволяє сумістити процеси визначення кількості контрольних розрядів та їх позицій.

Розглянемо зазначену методику.

Номера позицій, які призначаються для розміщення контрольних розрядів коду Хеммінга, визначаються значеннями 2^n, де n – цілі невід'ємні числа.

Таким чином, ми отримуємо, що під контрольні розряди згідно з методикою Хеммінга відводяться позиції $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$ і т.д.

У визначених таким чином позиціях коду Хеммінга інформаційні розряди знаходитись не можуть. Ті інформаційні розряди, що знаходились в зазначених позиціях до кодування, зсуваються в сторону старших розрядів (разом з старшими розрядами) на кількість позицій, необхідну для звільнення місця під контрольні розряди.

Кількість контрольних розрядів вважається достатньою в тому випадку, коли всі розряди на позиціях з номерами 2^n (n=0,1,2,3...) є контрольними.

Пояснимо сказане на прикладі.

Нехай потрібно визначити позиції контрольних розрядів та їх кількість в коді Хеммінга, якщо повідомлення, що кодується, складається з трьох інформаційних розрядів.

Для наочності домовимось позначати інформаційні розряди літерою X, контрольні – літерою K, а номер позиції, яку займає певний розряд, зазначати у нижньому індексі літер. Представлене таким чином вихідне повідомлення буде мати вигляд:

$$X_3\, X_2\, X_1.$$

Згідно з методикою Хеммінга номери позицій контрольних розрядів визначаються як 2^n (n=0,1,2,3...) і дорівнюють 1, 2, 4, 8,...

Як видно, у вихідному повідомленні із зазначених позицій зайняті інформаційними розрядами позиції 1 та 2. Для їх звільнення необхідно

зсунути всі інформаційні розряди повідомлення на дві позиції вліво. В результаті виконання операції зсуву отримуємо:

$$X_5\ X_4\ X_3\ K_2\ K_1.$$

Аналіз отриманого формату повідомлення дозволяє побачити, що в ньому інформаційним розрядом зайнята позиція 4 (2^2). Це означає, що розряди X_4 та X_5 необхідно зсунути вліво на одну позицію. Отримуємо:

$$X_6\ X_5\ K_4\ X_3\ K_2\ K_1.$$

Як видно, наведений формат повідомлення відповідає умові знаходження контрольних розрядів на позиціях 2^n (n=0,1,2,3,...). Робимо висновок, що для кодування кодом Хеммінга повідомлення з трьох інформаційних розрядів до нього необхідно додати три контрольних розряди, при цьому контрольні розряди повинні розташовуватись на позиціях 1, 2, 4.

Наступним постає питання визначення позицій інформаційних розрядів, які аналізуються при розрахунку (методом доповнення до парності) значень кожного з контрольних розрядів.

Інформаційний розряд на позиції J враховується при розрахунку значення контрольного розряду на позиції L, якщо в двійковому коді значення J на позиції L стоїть 1.

Нехай потрібно закодувати кодом Хеммінга повідомлення 1001.

Розв'язок.

За аналогією з попереднім прикладом визначаємо формат повідомлення:

$$X_7\ X_6\ X_5\ K_4\ X_3\ K_2\ K_1$$

де X_7=1, X_6=0, X_5=0, X_3=1 ($X_7X_6X_5X_3$=1001).

Переведемо номери наявних в форматі повідомлення позицій в двійкову систему (таблиця 1.1).

В позиції 1 двійкового коду (молодший розряд) значення розряду дорівнює 1 в комбінаціях 11<u>1</u>, 10<u>1</u>, 01<u>1</u>, 00<u>1</u> (001 – позиція контрольного розряду, що розраховується). Тобто, значення контрольного розряду на позиції 1 обчислюється в нашому прикладі таким чином, щоб кількість одиниць на позиціях 1, 3, 5 та 7 була парною. Цю умову можна записати виразом:

$$X_7 \oplus X_5 \oplus X_3 \oplus K_1 = 0,$$

звідки отримуємо:

$$K_1 = X_7 \oplus X_5 \oplus X_3 = 1 \oplus 0 \oplus 1 = 0.$$

Таблиця 1.1 – Дані для розрахунку контрольних розрядів

Номер позиції в десятковому коді	Номер позиції в двійковому коді
1	001
2	010
3	011
4	100
5	101
6	110
7	111

В позиції 2 значення розряду дорівнює 1 в комбінаціях 1$\underline{1}$1, 1$\underline{1}$0, 0$\underline{1}$1, 0$\underline{1}$0 (010 – позиція контрольного розряду), тому для другого контрольного розряду отримуємо:

$$K_2 = X_7 \oplus X_6 \oplus X_3 = 1 \oplus 0 \oplus 1 = 0$$

Аналогічно для третього (четвертого в розрядній сітці) розряду:

$$K_4 = X_7 \oplus X_6 \oplus X_5 = 1 \oplus 0 \oplus 0 = 1$$

Таким чином, закодоване повідомлення буде мати вигляд: 100$\underline{1}$1$\underline{00}$ (контрольні розряди підкреслено).

Приклад схемного рішення апаратної реалізації кодера Хеммінга для кодування чотирьохрорядних вхідних повідомлень з отриманням семирозрядних закодованих повідомлень, наведено на рисунку 1.

Як видно з наведеної схеми, вона відрізняється простотою і складається з простих комбінаційних елементів, що передбачає високу швидкодію пристрою.

Перейдем до дослідження питань декодування кодів Хеммінга.

Перевірка повідомлень на наявність помилки та визначення місця її виникнення виконується повторним розрахунком значень контрольних розрядів, за результатами порівняння яких з отриманими фактично формується слово коду помилки. Розрядність слова коду помилки визначається кількістю контрольних розрядів в закодованому повідомлені, а значення розрядів цього слова повинні доповнювати відповідні інформаційні розряди та кодовий розряд отриманого повідомлення до парності (молодший розряд слова формується перевіркою молодшого контрольного розряду і т.д.)

Поясними сказане ще одним прикладом.

Отже, повідомлення 1001100 отримане в результаті кодування кодом Хеммінга. Необхідно перевірити, чи відповідає слово коду помилки номеру позиції викривленого розряду для розрядів на позиціях 6 (інформаційний розряд) та 4 (контрольний розряд).

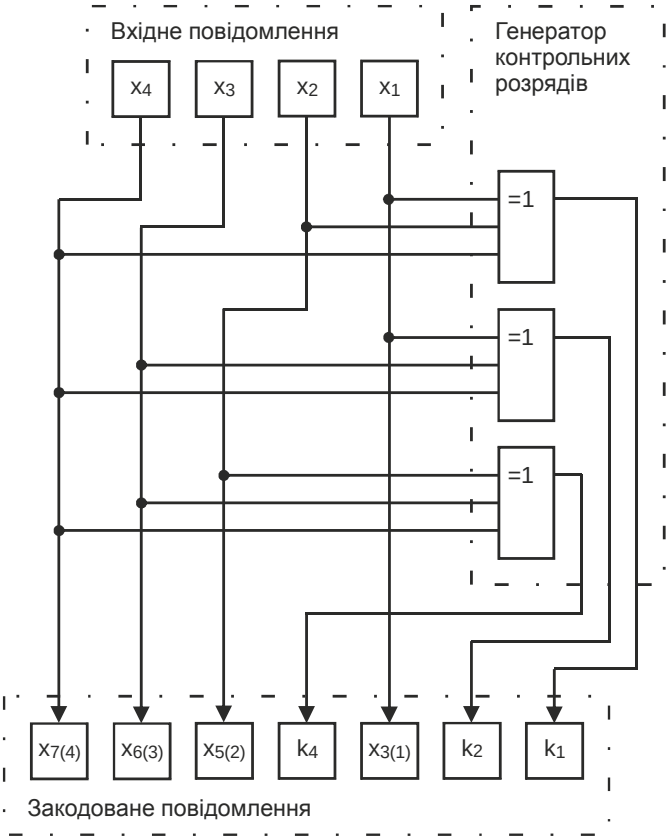


Рисунок 1 – Схема кодера кодів Хеммінга

Перед тим, як перейти до перевірки повідомлень визначимо позиції контрольних розрядів в повідомленні. Номери позицій контрольних розрядів визначаються як 2^n (n=0,1,2,3...) і дорівнюють 1, 2, 4, 8,... У нашому випадку в повідомлені присутні розряди на позиціях 1, 2 та 4, які і є контрольними.

З наведеного аналізу можна зробити висновок, що слово коду помилки повинне складатися з трьох розрядів, при чому перший розряд цього слова (позначимо його E_1) визначається відповідно до контрольного розряду на позиції 1; другий (E_2) – до контрольного розряду на позиції 2; третій (E_3) – до контрольного розряду на позиції 4.

Значення контрольного розряду на позиції 1 визначається згідно з правилом контролю на парність інформаційних розрядів X_7, X_5, X_3 та, відповідно, самого контрольного розряду K_1: $X_7 \oplus X_5 \oplus X_3 \oplus K_1 = 0$.

Оскільки розряд слова коду помилки E_1 повинен доповнювати значення перелічених розрядів до парності, його значення можна розрахувати за формулою: $E_1 = X_7 \oplus X_5 \oplus X_3 \oplus K_1$.

Аналогічним чином для розрядів E_2 та E_3 отримуємо:

$$E_2 = X_7 \oplus X_6 \oplus X_3 \oplus K_2$$
$$E_3 = X_7 \oplus X_6 \oplus X_5 \oplus K_4$$

У випадку викривлення розряду на позиції 6 повідомлення 1001100 прийме вигляд 1101100. За отриманими формулами розрахуємо значення розрядів слова коду помилки:

$$E_1 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$
$$E_2 = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$E_3 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

Упорядкувавши розряди слова коду помилки (у вигляді $E_3E_2E_1$), отримуємо значення 110, що дорівнює 6 – номеру викривленого розряду.

У випадку викривлення розряду на позиції 4 повідомлення 1001100 прийме вигляд 1000100. Розраховуємо значення розрядів слова коду помилки:

$$E_1 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$
$$E_2 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$
$$E_3 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

Отримуємо код помилки 100, що дорівнює номеру викривленого розряду.

Для виявлення факту виникнення помилки достатньо обчислити значення ознаки через логічне додавання всіх розрядів слова коду помилки: $E = E_1 \vee E_2 \vee E_3$.

Якщо ознака E=0, помилка в повідомлені відсутня, а якщо ознака E=1, то це свідчить про надходження ушкодженого повідомлення, до якого мають бути застосовані процедури (тобто, тезнічні засоби) корекції помилок.

Приклад схемного рішення апаратної реалізації декодера Хеммінга для декодування семирозрядних закодованих повідомлень (відповідно, з отриманням чотирьохрорядних вхідних повідомлень), що не зазнали ушкоджень, наведено на рисунку 2.

Як видно з наведеної на рисунку 2 схеми, вона також відрізняється простотою і складається з простих комбінаційних елементів, що передбачає високу швидкодію пристрою.

Якщо наведена схема генерує ознаку помилок E=1 , результат роботи декодера визнається недійсним і відбувається запуск процесу корекції помилок.

Типовий приклад схемного рішення апаратної реалізації коректора помилок кодів Хеммінга для семирозрядних закодованих повідомлень наведено на рисунку 3.

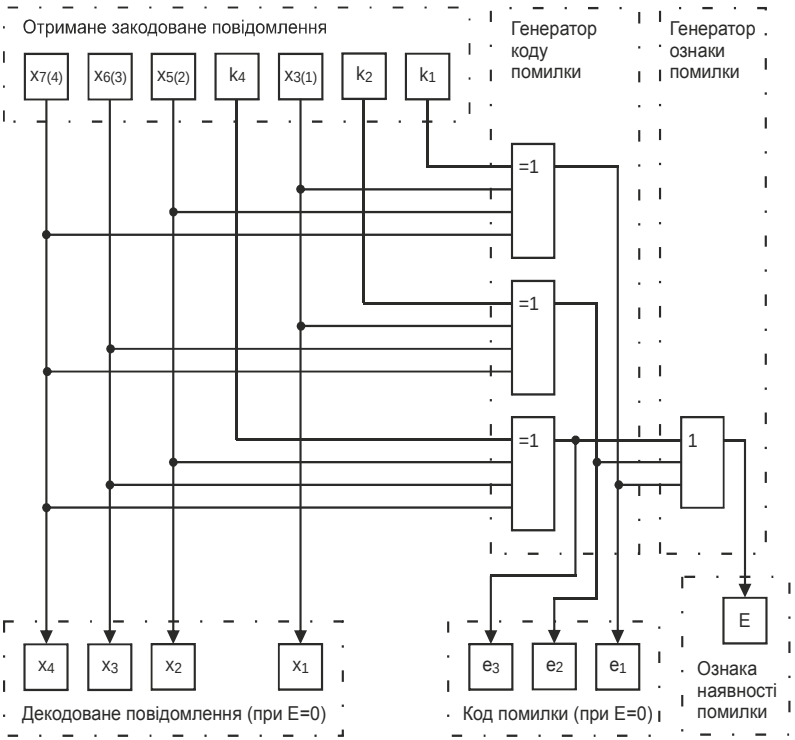


Рисунок 2 – Схема декодера кодів Хеммінга

Як видно з наведеної на рисунку 3 схеми, основною її складовою є вузли послідовнісного типу, що передбачає багатотактову ітераційну роботу і зумовлює найбільшу складність і найменшу швидкодію пристрою порівняно з розглянутими раніше. Наведені на рисунках 2-3 схемні рішення є класичними варіантами, орієнтованими на реалізацію «жорсткого» декодування кодів Хеммінга.

Розглянемо роботу коректора помилок більш детально.

На початку ушкоджене повідомлення попадає до регістра зсуву. Одночасно до лічильника 1 заноситься значення, що відповідає розрядності отриманого повідомлення, а в лічильнику 2 фіксується код помилки.

В процесі виконання процедури корекції помилки реалізується синхронний циклічний порозрядний зсув отриманого повідомлення в регістрі зсуву з одночасним декрементом (зменшенням на 1) значень в лічильниках.

В процесі декременту лічильника 2 очікується зменшення значення в ньому до одиниці, що свідчить про переміщення у молодший розряд регістра зсуву ушкодженого розряду повідомлення. Момент зменшення значення в лічильнику 2 до одиниці фіксує підключений до його виходу дешифратор на елементі множення, який, за отримання очікуваної ситуації, генерує сигнал дозволу корекції.

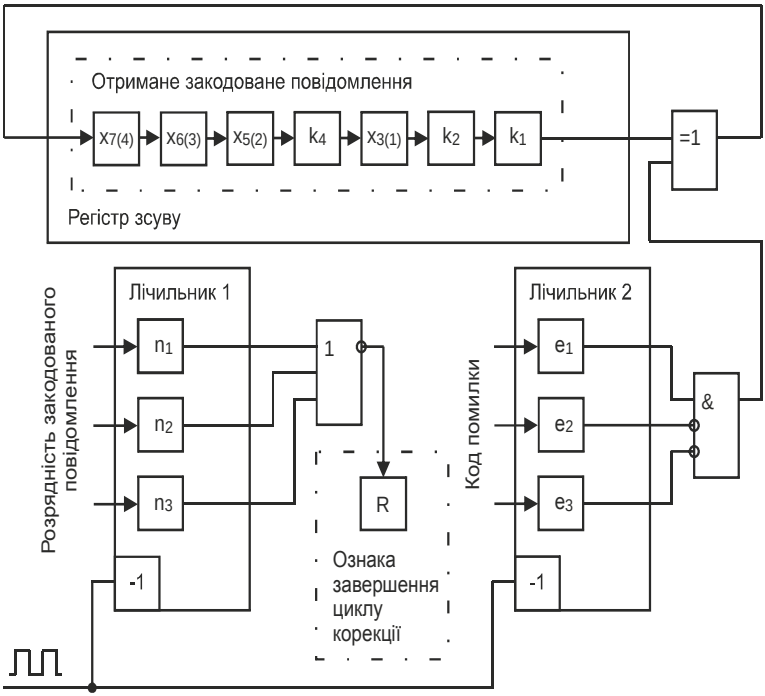


Рисунок 3 – Схема коректора помилок кодів Хеммінга

Сигнал дозволу корекції надходить на елемент додавання за модулем 2, який, в процесі циклічного зсуву, інвертує значення ушкодженого розряду.

Фактично, два зазначених логічних елементи виконують стосовно ушкодженого розряду однотипну операцію, незалежно від того, контрольний це розряд чи інформаційний:

$$k'_i = k_i \oplus (e'_1 \Lambda \text{ not}(e'_2) \Lambda \text{ not}(e'_3)), \quad x'_i = x_i \oplus (e'_1 \Lambda \text{ not}(e'_2) \Lambda \text{ not}(e'_3))$$

де k'_i та x'_i – значення відкорегованого розряду (k'_i - контрольний розряд, x'_i - інформаційний розряд) , яке заноситься в регістр в процесі циклічного зсуву замість ушкодженого.

Після корекції помилки процес зсувів продовжується до онулення лічильника 1, що свідчить про завершення повного циклу зсувів розрядів отриманого повідомлення, тобто, про те, що зазначені розряди стали на свої місця в розрядній сітці регістра зсуву. Момент зменшення значення в лічильнику 1 до нуля фіксує підключений до його виходу дешифратор на елементі додавання, який, за отримання очікуваної ситуації, генерує сигнал завершення корекції *R*.

В процесі корекції може виникати тупикова для «жорсткого»

декодування ситуація першого роду, коли регістр 1 онулюється раніше (відповідно, і генерується сигнал завершення корекції *R*), ніж значення в лічильнику 2 зменшиться до 1. При цьому цикл завершується без корекції.

Після завершення корекції запускається процедура декодування відкорегованого повідомлення класиним методом, для чого може застосовуватись схема з рисунку 1.2. В процесі повторного декодування може виникати тупикова для «жорсткого» декодування ситуація другого роду, коли декодер формує ознаку помилки і після корегованння повідомлення.

Тупикові ситуації першого і другого роду є ознаками однієї проблеми – ушкодження повідомлення в кількох розрядах. Виправлення подібних ушкоджень є не вирішуваною задачею для «жорсткого» декодування кодів Хеммінга.

Наявність тупикових ситуації першого і другого роду робить системи «жорсткого» декодування кодів Хеммінга нестійкими щодо кратних помилок і зумовлюють актуальність застосування систем «м'якого» декодування кодів Хеммінга.

Проблема систем «м'якого» декодування – відсутність однозначно визначених правил, які дозволяють визначити структуру системи завадостійкого кодування на етапі декодування, оскільки процес декодування потребує урахування великої кількості нестабільних характеристик імовірнісного характеру, що придає йому рису інтелектуальності.

Одним з основних завдань у даній роботі є визначення сучасного методу декодування завадостійких кодів Хеммінга з метою виявлення можливостей його модифікації для підвищення стійкості системи завадостійкого кодування. Одним із шляхів вирішення даної проблеми є використання паралельних методів декодування. Одним з таких напрямків є використання апарату теорії штучних нейронних мереж, що мають повністю паралельну архітектуру.

Дослідження властивостей методів декодування завадостійких кодів

к.т.н., доц. Чорненький В.І.
Хмельницький національний університет

Завдання завадостійкого кодування полягає в тому, щоб за прийнятим словом визначити (із заданим ступенем достовірності) і видати одержувачу саме переданий інформаційний вектор. Вирішення цього завдання називається декодуванням.

Автоматичний пристрій, що реалізує декодування, будемо називати декодером.

За необхідності застосовувати кодування і декодування до послідовності слів випливає, що кодер і декодер повинні закінчувати обробку одного слова за час, пропорційний довжині кодового слова, тобто за час передачі кодового слова через канал зв'язку. Такий режим роботи декодера

назвемо роботою в реальному масштабі часу.

Надалі при дослідженні складності задач декодування в якості основного режиму роботи будемо розглядати роботу саме в реальному масштабі часу.

З технічної точки зору процес декодування характеризується складністю і швидкодією декодера.

Таким чином, завдання декодера полягає в побудові відображення множини прийнятих кодових комбінацій (КК) A_*^N на множину інформаційних слів A^K [1]:

$$D: A_*^N \to A^K, \tag{1.1}$$

де D - оператор знаходження кодового слова (визначається вибором того чи іншого критерію декодування), N - довжина коду, K - довжина інформаційного слова, A_* - алфавіт з {0,1, *}, * - символ стирання.

Розглянемо відомі в науково-технічній літературі критерії декодування.

При декодуванні за максимумом співпадіння (подібності або правдоподібності) [2,3] прийнятому слову r з A_*^N ставиться у відповідність таке кодове слово $d_i \in A^{(N,K)}$ ($A^{(N,K)}$ – множина всіх дозволених кодових комбінацій коду), для якого ймовірність $P(d_i|r)$ того, що передано кодове слово d_i , за умови, що було прийнято слово r, максимальна з усіх можливих відносно слів d, що належать коду:

$$P(d_i \mid r) \equiv \max_{d \in коду} P(d \mid r). \tag{1.2}$$

Це означає, що всі слова $r \in A_*^N$, для яких $P(d_i \mid r) > P(d_j \mid r)$, для всіх $j \neq i$, повинні ототожнюватися з кодовим словом d_i.

Згідно з інформаційним критерієм декодування [1,2,3] прийнятому слову r ставиться у відповідність таке слово d_i, для якого взаємна відповідність між d_i і r є максимальною [4], що доводить справедливість умови:

$$P(r \mid d_i) \geq P(r \mid d_j) \text{ для всіх } j \neq i. \tag{1.3}$$

Тому, знаходження кодового слова *d* за прийнятим словом *r* на основі критерію максимуму правдоподібності і на основі інформаційного критерію призводить до одного й того ж оператора *D*, що визначається співвідношенням (1.1).

У зв'язку з цим в роботі будемо використовувати критерій максимуму правдоподібності, а також досліджувати й удосконалювати методи декодування, реалізовуючи саме цей критерій декодування.

При декодуванні двійкових блокових кодів по максимуму правдоподібності вся множина A_*^N прийнятих слів розбивається на підмножини, що не перетинаються $A_j, i = \overline{1,2^K}$, число яких дорівнює числу всіх дозволених кодових комбінацій (ДКК), причому кожна з цих підмножин ставиться у відповідність до однієї кодової комбінації d_i. У зв'язку з цим (1.1) може бути записано у вигляді:

$$D: A_*^N \to A^{(N,K)} \to A^K, \qquad (1.4)$$

де $A^{(N,K)}$ - множина усіх дозволених кодових комбінацій коду.

Слід зазначити, що при реалізації будь-якого типу декодування знаходження кодового слова *d* за прийнятим словом *r* є найбільш складною і трудомісткою частиною. Нею, по суті, визначається складність практичного застосування того чи іншого методу декодування.

Однак, перш ніж створювати принципово нові методи декодування, необхідно розглянути можливості розвитку існуючих сучасних методів декодування.

В даний час популярними стали алгоритми декодування з «м'якими» рішеннями.

Алгоритми декодування з «м'якими» рішеннями базуються на підході, коли на вхід декодера подається не тільки інформація, але і правдоподібність цієї інформації. Найкращі результати при такому підході дозволяють отримати ітеративне декодування, засноване на використанні інформації з виходу про правдоподібність декодера для декодування чергової ітерації.

В даний час існують два методи декодування завадостійких кодів, заснованих на прийнятій послідовності дійсних чисел [5]:

- декодування з «жорстким» рішенням (hard-decision detector - HDD), мета якого полягає у виправленні двійкових помилок, що виникли в процесі вибору «жорстких» рішень в декодері;

- декодування з «м'яким» рішенням (soft-decision detector - SDD), при якому прийняті з каналу величини вводяться безпосередньо в декодер для формування оцінок кодової послідовності.

У загальному випадку SDD більш трудомістке, ніж HDD, так як

вимагає виконання операцій з дійсними числами і пов'язана з необхідністю обчислення апостеріорних статистик для кодових символів. Проте, збільшення трудомісткості окупається потенційним підвищенням ефективності СЗК.

Розглянемо двійкове джерело, яке керує множиною кодованих сигналів d_i^j, де i - передана кодова комбінація або ДКК, j - відповідний біт усередині даної КК. В процесі передачі по каналу з завадами на дані накладається шум так, що на виході демодулятора утворюються безперервні стохастичні змінні r_i, що визначаються виразом $r_i=d_i+n_i$, де d_i - бажаний компонент сигналу, n_i – випадкова величина з нульовим середнім і дисперсією δ^2, яка характеризує рівень шуму в каналі зв'язку. Тому r_i - випадкова змінна із середнім відхиленням d_i^j -1 або d_i^j +1 (розряд змінює значення на протилежне), залежно від того, передавався двійковий нуль або двійкова одиниця.

При використанні HDD перед декодуванням вихід r_i - квантується на два рівні, що не дозволяє якісно поліпшити характеристики СЗК. Велике поліпшення досягається шляхом використання декодера SDD, який бере до уваги оцінку всієї інформації, що міститься в неквантованих вихідних змінних r_i. Як було зазначено раніше, такий оптимальний декодер, що мінімізує ймовірність помилки (коли всі передані послідовності різновірогідні), відомий як декодер максимальної правдоподібності (ДМП).

Функція правдоподібності (рішення задачі декодування ДМП) задається або обчислюється, виходячи з специфікації каналу. У разі використання каналу, в якому шум впливає на кожен символ коду незалежно від інших символів, обчислення правдоподібності $P(d_i|r)$ в (1.2) зводиться до виведення умовних ймовірностей окремих біт.

Як показано в [5] квадрат Евклідової відстані $(r_i - d_i^j)^2$ являється метрикою для декодування по максимуму правдоподібності.

При використанні HDD максимізація L_j еквівалентна максимізації:

$$L_i = P\left(d_i \middle| r\right) = \sum_{j=1}^{N} r_i d_i^j \qquad (1.5)$$

Таким чином, рішення задачі декодування по максимуму правдоподібності полягає в максимізації правдоподібності прийнятих з каналу кодових комбінацій відповідно до функції (1.5).

Оскільки методи «м'якого» декодування блокових кодів більш ефективні, проведемо їх огляд.

Для декодування блокових кодів в даний час використовуються наступні методи «м'якого» декодування.

Відомий алгоритм декодування згортальних кодів Вітербі в [2,3,5] застосовується і для декодування двійкових блокових кодів, для яких з перевірочної матриці коду будується мінімальна синдромна решітка коду [5]. У загальному випадку, решітка блокового коду має нерегулярну структуру. В результаті помітно ускладнюється реалізація алгоритмів декодування блокових кодів за решіткою в порівнянні зі згортальними кодами. Однак, для деяких класових кодів, таких як розширені коди БЧХ та коди Ріда-Маллера, решітка може бути розділена на секції. В результаті виходить більш регулярна і симетрична ґратчаста структура з великим числом паралельних підрешіток, які можуть бути використані для побудови швидких декодерів, наприклад, декодерів Вітербі для двійкових блокових кодів [5].

Застосування мінімальних кодових решіток дозволяє використовувати для декодування двійкових блокових кодів ітеративні методи декодування згортальних кодів такі, як алгоритм Вітербі з «м'яким» виходом [6], алгоритм декодування по максимуму апостеріорної ймовірності [6,7] і його модифікації [5]. Проте, дані алгоритми при нерегулярності решітки блокового коду вимагають великих обчислювальних витрат.

Існує безліч методів «м'якого» декодування завадостійких кодів, які формують на виході найбільш ймовірну кодову послідовність або кодове слово (або список кодових слів).

Такі алгоритми відомі в літературі, як алгоритми з «м'яким» входом і «жорстким» виходом. До них, наприклад, відносяться: алгоритм Чейза зі своїми модифікаціями [8], алгоритм декодування по впорядкованим статистикам [9], декодування по мінімуму узагальненої відстані, декодування по списку [3].

З появою в 1993 році турбо-кодів виникла потреба в алгоритмах, які обчислюють не тільки найбільш ймовірне кодове слово (або список кодових слів), але і оцінюють надійність їх інформаційних символів для подальшої обробки.

Технологія ітеративного (турбо) декодування з'явилася в 1954 році в роботі П. Елайс [3], присвяченій ітеративним кодами. Пізніше ітеративні алгоритми з «м'яким» виходом також були введені в 1962 році в роботі Р. Галлагера за кодами з низькою щільністю перевірок на парність і, дещо пізніше, в роботі Л. Бала [7] і Дж. Л. Мессі [8].

В обох випадках алгоритми виконують проходи як у прямому, так і в зворотному напрямках для обчислення надійності кодових символів. Головна ідея була тією ж самою, що і сьогодні – максимізувати ймовірність символу, який міг бути переданий, за умови, що відома спотворена версія кодової послідовності.

Як зазначалося в [7] алгоритм декодування, відповідний ІВР, обчислює точні ймовірності (досягає максимуму правдоподібності) після деякої кількості ітерацій, якщо граф Таннера [4] для даного коду не містить циклів. Так як для більшості використовуваних на практиці кодів граф Таннера містить короткі цикли, то збіжність ІВР алгоритму декодування не гарантовано [5].

У публікаціях [6] наводиться поліпшена реалізація ітеративного алгоритму Чейза для декодування даних з «жорстким» виходом.

У публікаціях наводиться опис та рекомендації до реалізації ітеративного алгоритму декодування лінійних блокових кодів, який використовує адаптацію перевірочної матриці коду від ітерації до ітерації, засновану на зв'язках бітів всередині використовуваного коду.

У публікаціях патентів наводиться алгоритм ітеративного декодування лінійних блокових кодів з використанням перевірочної матриці коду. Однак, число перевірочних виразів обмежене числом рядків перевірочної матриці.

Одним з основних завдань наукових досліджень в аналізованій галузі є визначення сучасних методів декодування завадостійких кодів з метою виявлення можливостей їй модифікації для підвищення стійкості системи завадостійкого кодування.

Стосовно лінійних блокових кодів дана задача може бути вирішена шляхом використання розширених перевірочних рівнянь, що в процесі ітеративного декодування дає більше інформації щодо інформаційних елементів КК.

Оскільки будь-яка лінійна комбінація рядків перевірочної матриці може служити перевірочним рівнянням, то використання подібних виразів може підвищити надійність прийнятих інформаційних елементів КК. У той же час, при збільшенні кількості перевірочних рівнянь пропорційно збільшується обчислювальна складність реалізації алгоритму.

Складнощі та обмеження, пов'язані з використанням існуючих сучасних методів декодування блокових кодів, дозволяють зробити висновок про необхідність пошуку їх нових альтернативних модифікацій, що дозволяють поліпшити окремі характеристики (наприклад, завадостійкість) декодера.

Література

1. Блох Э.Л. Обобщённые каскадные коды. Алгебраическая теория и сложность реализации / Э.Л. Блох, В.В. Зяблое - М.: Связь, 1976. - 462 с.
2. Кларк Д. Кодирование с исправлением ошибок в системах цифровой связи / Д. Кларк, Д. Кейн- М.: Радио и связь, 1987. - 195 с.
3. Питерсон У. Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон - М.: Мир, 1976. - 593 с.
4. Форни Д. Каскадные коды / Д. Форни - Пер. с англ.; под ред. Самойленко С.И. М.: Мир, 1970.- 208 с.
5. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса - Пер. с англ. В.Б.Афанасьева. - М.: Техносфера. 2005 .- 320 с.
6. Скляр Б. Цифровая связь. Теоретические основы и практическое применение / Б. Скляр - пер. с англ. - изд. 2-е., испр. - М.: Вильяме, 2004.- 156 с.
7. Bahl L.R. Optimal decoding of linear codes for minimizing symbol error rate / L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv // IEEE Transactions on Information

Theory. - 1974. - Vol. 20. - n. 2. - P. 284-287.
8. Chase D. A Class of Algorithms for Decoding Block Codes with Channel Measurement Information / D. Chase //IEEE Transactions on Information Theory. - 1972. -Vol. IT-18. - P. 170-182.
9. Fossorier M. Soft-Decision Decoding on Linear Block Codes Based on Ordered Statistics / M. Fossorier, S. Lin // Transactions on Information Theory. - 1995. - Vol. 41. - n. 5. -P. 1379-1396.

Наукове видання

Інтелектуальний потенціал - 2016 (Ч.1): Матеріали Всеукраїнської науково-практичної конференції молодих науковців та студентів. 1-3 грудня 2016р., м. Хмельницький/Кол. авт. – Хмельницький: УЕП, 2016. – 100 с.

Відповідальність за зміст текстів і якість редагування матеріалів покладена на авторів і наукових керівників.

Комп'ютерна верстка: Чешун В.М.
Дизайн: Муляр С.В.

Здано до складання 5.12.16. Підписано до друку 6.05.16. Формат 60х84/16. Папір друкарський. Тираж 50 прим. Умовних друкованих аркушів – 8,0. Обліково-видавничих аркушів –1,8.

Редакційний відділ УЕП 29016, м. Хмельницький, вул. Львівське шосе, 51/2.

ББК 74.480.278
С.88