УДК 681.3.06

А.І. Роговенко, ст. викладач
С.О. Сєнько, магістрант
Чернігівський національний технологічний університет, м. Чернігів, Україна

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ГЕНЕРАТОРІВ ВИХІДНИХ КОДІВ НА МОВІ ОПИСУ АПАРАТУРИ

А.И. Роговенко, ст. преподаватель
С.А. Сенько, магистрант
Черниговский национальный технологический университет, г. Чернигов, Украина

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ГЕНЕРАТОРОВ ИСХОДНЫХ КОДОВ НА ЯЗЫКЕ ОПИСАНИЯ АППАРАТУРЫ

Andrii Rohovenko, senior teacher
Serhii Sienko, Master's Degree student
Chernihiv National Technological University, Chernihiv, Ukraine

FEATURES OF USING GENERATOR OF SOURCE CODES ON HARDWARE DESCRIPTION LANGUAGE

Проведено аналіз сучасних методів та підходів до автоматичної генерації вихідних кодів, виділено обмеження та особливості їх використання для мов опису апаратури. Акцентовано увагу на методах, які у разі їх використання нададуть змогу зменшити час розроблення пристрою на основі ПЛІС, при відповідній надійності та якості вихідного коду на мові опису апаратури.

***Ключові слова**: генерація, вихідний код, мови опису апаратури, VHDL, ПЛІС.*

Проведен анализ современных методов и подходов к автоматической генерации исходных кодов, выделены ограничения и особенности их использования для языков описания аппаратуры. Акцентировано внимание на методах, которые при их использовании дадут возможность уменьшить время разработки устройства на основе ПЛИС, при соответствующей надежности и качества исходного кода на языке описания аппаратуры.

***Ключевые слова:** генерация, исходный код, языки описания аппаратуры, VHDL, ПЛИС*

The article analyzes modern methods and approaches to automatically generate source code are highlighted limitations and peculiarities of their use for hardware description languages. Attention is focused on methods which when used, will help reduce development time of the device based PLD, with appropriate reliability and quality of source code in hardware description language.

***Key words:** generation, source code, hardware description languages, VHDL, PLD.*

Постановка проблеми. Останнім часом вбудовані системи на основі ПЛІС набули значних структурних ускладнень. Це характеризується, по-перше, ускладненням периферії, по-друге, підвищеними вимогами щодо функціональних можливостей та швидкодії. Ці особливості обумовлюють підвищення складності створення програм, що призводить до удорожчання процесу їх написання, зниження їх надійності та відмовостійкості, збільшення кількості дефектів та помилок.

Використання ручного налагодження складної периферії, в неадаптованих до сприймання людиною форматах, ускладнює процес відлагодження програми, її діагностику та пошук помилок. Використання застарілих методів та підходів до розроблення програм не може забезпечити їх належну якість. Одним з варіантів вирішення цієї проблеми є використання генераторів вихідних кодів. Є декілька підходів до реалізації генераторів вихідних кодів, але у разі їх використання для генерації кодів на мові опису апаратури виникають специфічні особливості, через які виникає необхідність розглянути питання генерації кодів з додатковими обмеженнями. Ці обмеження є базовими для вибору відповідного методу генерації вихідних кодів.

Аналіз останніх досліджень і публікацій. У результаті аналізу підходів до створення програм визначено певні недоліки в ручному створенні програм для вбудованих систем. У табл. 1 зображена статистична інформація про залежність кількості помилок від розміру проекту.

Як видно з табл. 1, зростання кількості помилок безпосередньо залежить від розміру проекту. Саме тому розроблення великих проектів вимагає значних витрат часу на про-

ектування структури вихідного коду. Детальність спроектованої структури є одним з вирішальних критеріїв, які впливають на якість проекту загалом і зменшення кількості помилок. При великих розмірах проекту зрозуміти функціональне призначення чи алгоритми функціонування на основі написаного вручну коду є дуже складною процедурою, яка вимагає високої кваліфікації фахівця, який читає код.

Таблиця 1

Залежність кількості помилок у коді від розміру проекту

Розмір проекту (кількість рядків коду)	Типова густина помилок
Меньше 2К	0 – 25 помилок на 1000 рядків коду
2К-16К	0 – 40 помилок на 1000 рядків коду
16К-64К	0,5 – 55 помилок на 1000 рядків коду
64К-512К	2 – 70 помилок на 1000 рядків коду
512К – і більше	4 – 100 помилок на 1000 рядків коду

Вважається, що вирішенням цих проблем може бути використання високорівневих графічних редакторів проекту, або автоматизованих генераторів коду, які дозволять автоматизувати процес проектування, створювати наочні проекти, які буде легко супроводжувати.

Мета статті. Метою даної статті є проведення аналізу методів та підходів до автоматизованої генерації коду, аналіз їх переваг та недоліків, визначення рівня їх придатності до використання у генераторах вихідних кодів на мові опису апаратури, а також формування рекомендацій щодо доцільності їх подальшого використання під час проектування генератора коду компонентів обчислювальної системи з прискореним виконанням операцій у базисі ПЛІС.

Виклад основного матеріалу. Аналізуючи генератори вихідного коду на мовах опису апаратури, треба враховувати їх особливості. Здебільшого мови опису апаратури працюють не з логічними даними чи об’єктами, а з сигналами, що спричиняє додаткові помилки під час оброблення даних. Більшість лексем у цих мовах описують не алгоритм оброблення даних, а саме роботу з рівнями на виводах мікросхем чи перемикання станів тригерів та регістрів. Ці особливості змушують програміста не тільки знати алгоритм оброблення даних, але й мати знання та навички у схемотехніці. Таким чином, основними критеріями під час вибору методу генерації коду на мовах опису апаратури є можливість винесення рівнів абстракцій та виділення логічних сутностей, можливість вести роботу на рівні алгоритмів, а також можливість автоматизувати контроль за обробленням даних та поведінкою сутностей у системі.

Метод статичної генерації коду. Розглянемо статичний метод генерації коду. Цей метод генерації коду представлений у більшості сучасних САПР. Цей метод генерації коду в більшості випадків не генерує коду, який виконує алгоритми, а лише «скелет», в який у подальшому буде вписаний код, що безпосередньо виконує необхідні функції. Генерація коду виконується за його статичним описом. Під статичним описом розуміють незмінювані у процесі функціонування програми її характеристики чи параметри. Такими характеристиками можуть бути назви класів, сигнатури методів, розрядність шини, кількість входів або виходів мікросхеми і т. ін.

Особливістю статичних генераторів коду є те, що для генерації коду їм достатньо однієї ітерації (одного проходу по вихідному опису). Загальну схему їх роботи представлено на схемі, яка представлена на рис. 1.

Як видно з рис. 1, вихідний опис передається в аналізатор коду, де він розбивається на частини, які являють собою лексеми мови вихідного опису. Для кожної лексеми є правило, за яким вона однозначно передвоюється в результуючу форму. Правил для перетворення лексеми може бути декілька. У цьому випадку аналізатор вибирає прави-

ло, виходячи з його налаштувань, які задаються користувачем, тобто користувач має вказати на те, яке правило буде використано, бо однією з особливостей статичних генераторів є те, що вони не приймають рішень самостійно, а повністю керуються користувачем. Можливість вибрати між декількома правилами дозволяє керувати характеристиками результуючого коду. Прикладом використання декількох правил є оптимізація за розміром або за швидкістю роботи. В цьому випадку складається набір правил, які створюють швидкий код, і інший набір правил, які створюють код малий за розміром, та надається користувачу можливість перемикатися між цими правилами.

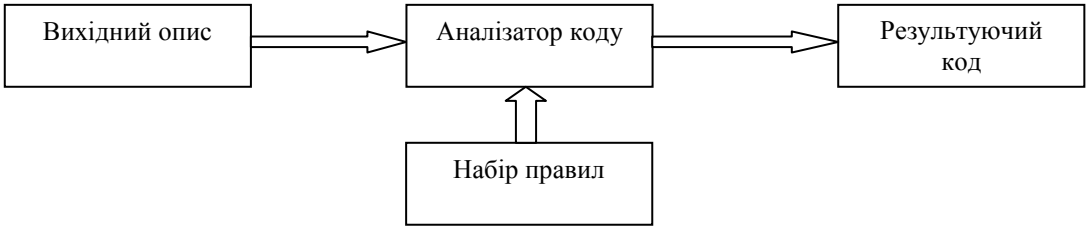

Рис. 1. Загальна схема статичної генерації

Такий тип генераторів широко використовується як допоміжний засіб під час розроблення програм на мовах опису апаратури. Прикладом використання цього підходу для генерації коду на мовах опису апаратури є САПР Quartus, де ми маємо можливість вказати кількість входів та виходів мікросхем, а також у графічному редакторі розташувати їх та з'єднати провідниками. САПР, керуючись створеною схемою, згенерує шаблон на мові опису апаратури, де розробнику потрібно буде вписати алгоритм роботи кожної мікросхеми. Це дозволяє наочно згрупувати код за блоками та зосередитись лише на алгоритмі роботи, бо позбавить від необхідності писати декларації входів та виходів мікросхем.

Перевагами статичних генераторів коду є простота їх реалізації, бо кожній лексемі на момент генерації відповідає одне вибране правило, а також передбачуваність результату та відносна швидкість роботи. Цей метод дозволяє виділити додаткові рівні абстракції та логічно розділити проект на сутності, що є дуже важливим, але він не відслідковує роботу з даними, що є одним з ключових параметрів під час вибору методу генерації кодів на мовах опису апаратури. Ще одним недоліком є те, що цей тип генераторів не аналізує поведінку об'єктів у програмі, і як наслідок контроль за виявленням помилок у поведінці програми покладається на плечі програміста, що є небажаним при генерації коду на мовах опису апаратури.

Метод динамічної генерації коду. Метод динамічної генерації коду передбачає використання не тільки статичної інформації, а й інформації про поведінку програми та її об'єктів, її можливих станах, про можливі переходи із стану в стан, а також про дані, які вона буде обробляти, що є важливим для генерації кодів на мовах опису апаратури.

Найбільш зручним та простим описом для такого типу генераторів є кінцевий автомат. Цей метод опису наочно відображає роботу пристрою, оскільки багато мікросхем є реалізаціями цифрового автомату. В автоматі вказують стан об'єктів системи, причини, через які вони переходять з одного стану в інший, а також дії, які вони при цьому виконують. Опис кінцевого автомату можна здійснювати на спеціальних метамовах, або графічно у вигляді графу.

Також зручним та наочним способом створення вихідного опису є графічний опис роботи програми у вигляді алгоритму чи діаграми поведінки. В таких випадках з деякого набору команд чи логічних блоків складають алгоритм чи модель поведінки, вказуючи необхідні параметри блоків, наприклад, назви змінних.

Для проведення аналізу, верифікації та процесу генерації графічний опис, чи кінцевий автомат переводять в опис на метамові. Метамова може бути як мовою програму-

вання, так і деяким, найчастіше текстовим, описом, який можливо легко зчитати. Прикладом мета опису є опис за допомогою XML.

Таким чином, загальну схему динамічної генерації коду представимо в такому вигляді (рис. 2).

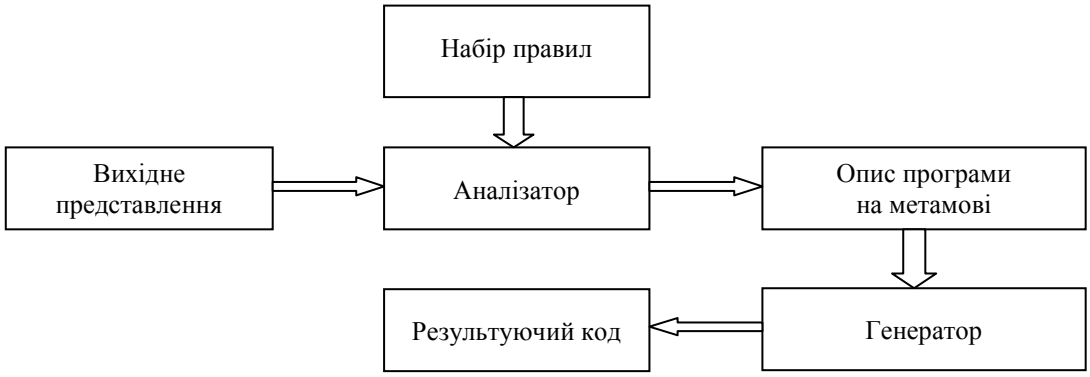

Рис. 2. Загальна схема динамічної генерації коду

Як видно з рис. 2, схема перетворення вихідного представлення в опис на метамові є абсолютно аналогічним схемі генерації коду статичним генератором коду. Тобто, які видно на рис. 2, із вихідного представлення шляхом статичної генерації отримуємо на метамові опис алгоритму, цифрового автомату, діаграми поведінки тощо. Метаопис аналізується генератором коду, на виході якого ми отримуємо вихідний код програми.

Динамічні генератори дуже часто використовують під час створення описів апаратури. Як приклад наведемо САПР Quartus, де є можливість у графічному редакторі створити цифровий автомат. Також у цьому САПР є редактор, в якому є можливість вказати, які вихідні сигнали повинні буди встановлені на задані вхідні сигнали. Обидва редактори генерують як результат опис на мові опису апаратури.

Цей метод дозволяє створювати опис у наочному форматі, що є ключовим фактором для успішної підтримки та обслуговування пристрою. Опис у форматі цифрового автомату дозволяє представити роботу системи у форматі, який є одночасно простим та наочним, а також близьким до дійсних процесів, що відбуваються у ній. Опис у вигляді алгоритмів чи діаграм поведінки дозволяє зосередитися на алгоритмах оброблення даних та позбавити необхідності програміста працювати з сигналами. Перевагою динамічних генераторів є те, що вони враховують потоки даних, з якими працює пристрій чи програма, а також поведінку об’єктів, що дозволяє отримати найбільш оптимальний код, який є ключовим для генерації кодів на мовах опису апаратури. Незважаючи на складність в їх реалізації, порівняно великий час генерації результату, метод динамічної генерації є найбільш придатним до генерації кодів на мовах опису апаратури.

Генератор на основі шаблонів та підстановок. Цей метод передбачає, що розробник буде створювати шаблон коду у спеціальному форматі. Також він повинен підготувати дані, або створити програму, яка буде отримувати дані та зводити їх до спеціального формату.

Набір шаблонів може бути як статичним, так і динамічним. При статичному шаблоні підстановка виконується під час генерації коду. Як приклад наведемо templates в мові java. Під час використання templates описується алгоритм оброблення даних без вказування типів даних. Типи даних вказуються пізніше і впливають на поведінку описаного алгоритму. Наприклад операція порівняння буде проводитися по-різному для числових і для строкових типів. Такий підхід є порівняно простим, але дуже обмеженим у використанні.

Під час використання динамічних шаблонів дані, що підставляються в нього, безпосередньо впливають на результуючий код. Гарним прикладом цього типу генерації є jsp

web-сторінка. В цьому прикладі HTML код результуючої сторінки буде залежати від даних, які вона буде відображати. Наприклад, залежно від даних буде змінюватись кількість рядків у табличках, чи будуть приховуватися, чи навпаки відображатися поля залежно від їх значення. Загальна схема генерації відображена на рис. 3.

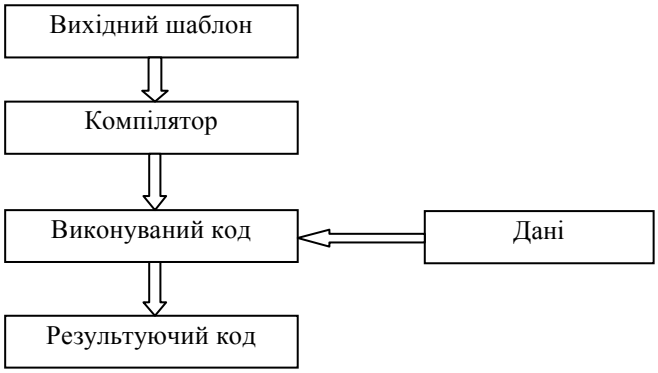

Рис. 3. Загальна схема генерації коду по методу динамічних шаблонів

Як видно на рис. 3, вихідний шаблон компілюється у виконуваний код. Дані обробляються виконуваним кодом, у результаті роботи якого ми отримуємо результуючий код. Для прикладу, наведеному вище, виконуваним кодом буде java bytecode, який генерує HTML код сторінки, який і є результуючим кодом.

Крім даних, у шаблони може підставлятися інший код. Прикладом таких підстановок є jsp web-сторінка, в яку, крім даних, підставляється JavaScript код, який буде виконуватися браузером. Загальна схема генерації наведена на рис. 4.

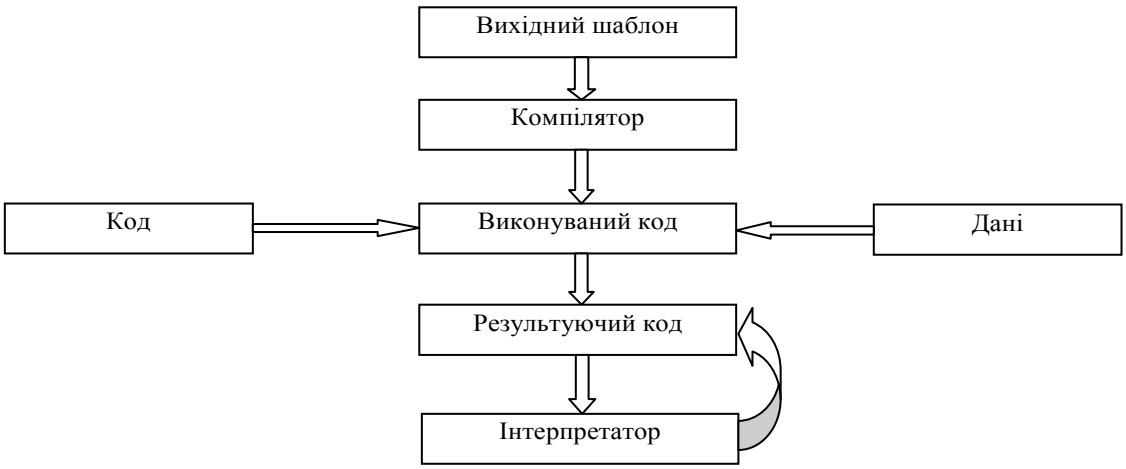

Рис. 4. Схема генерації коду по методу динамічних шаблонів з підстановкою коду

На відміну від попереднього способу використання шаблонів, у цьому методі ми отримуємо результуючий код, який може змінюватися у процесі роботи. На рис. 4 це відображено зворотним зв'язком від результуючого коду до інтерпретатора. Тобто код, який ми підставляємо у шаблон у процесі генерації, може змінювати код заповненого даними шаблону в процесі своєї роботи.

Важливими перевагами цього методу з огляду на генерацію кодів на мовах опису апаратури є те, що вони дозволяють виділяти рівні абстракції та сутності, що є необхідним для побудови моделей системи. Цей метод спроможний проконтролювати коректність роботи з даними, але не може виявити помилок у поведінці об'єктів, що є неприйнятним для генерації кодів на мовах опису апаратури, за яким будуть генеруватися реальні пристрої. Тому цей метод у сфері вбудованих систем використовується здебільшого для генерації моделей пристрою, та не набув розповсюдження для генерації описів, з яких будуть генеруватися реальні пристрої.

Висновки. Таким чином результати аналізу представимо у вигляді табл. 2. Як видно за табл. 2, статичний метод генерації кодів виграє у швидкості, оскільки час генерації лінійно залежить від кількості лексем вихідного опису. Він простий у реалізації, його використовують під час проектування вбудованих систем та генерації кодів на мовах опису апаратури для автоматизації однотипних операцій, але його великим недоліком є те, що він покладає більшість на плечі програміста.

Таблиця 2

Зведена таблиця характеристик методів генерації кодів

Метод генерації	Статичний	Метод шаблонів	Динамічний
Простота реалізації	Простий	Простий	Складний
Чи враховує потоки даних?	Не враховує	Враховує	Враховує
Чи враховує поведінку об'єктів?	Не враховує	Не враховує	Враховує
Виявлення синтаксичних помилок	Виявляє	Виявляє	Виявляє
Виявлення помилок даних	Не виявляє	Виявляє	Виявляє
Виявлення помилок у логіці виконання	Не виявляє	Не виявляє	Виявляє
Швидкість генерації	$t=k\cdot N$ t – час генерації; k – вартість однієї заміни; N – кількість підстановок	$t=p\cdot D\cdot N$ t – час генерації; p – вартість однієї заміни; N – кількість підстановок; D – кількість записів даних	$t=s^N$ t – час генерації; s – вартість однієї заміни; N – кількість підстановок

Швидкість роботи методу шаблонів важко оцінити, адже вона залежить від кількості даних, але здебільшого він програє статистичному методу генерації. Метод шаблонів відслідковує потоки даних, а це дозволяє знаходити помилки, пов'язані з цим, а також генерувати більш оптимальний код. Він досить простий у реалізації, але використання для генерації коду на мовах опису апаратури обмежується побудовами математичних моделей пристроїв.

Швидкість динамічного методу генерації експоненціально залежить від кількості лексем у метаописі, це призводить до досить значного часу генерації при великих розмірах проекту. Цей метод відслідковує потоки даних і поведінку об'єктів, а також виявляє помилки, що пов'язані з цим, бере на себе вирішення багатьох проблем та знімає з програміста необхідність замислюватися над низькорівневими процесами, а дозволяє зосередитись лише на алгоритмі.

На рис. 5 наведено порівняльний графік швидкості генерації коду від розміру проекту для усіх трьох методів генерації.

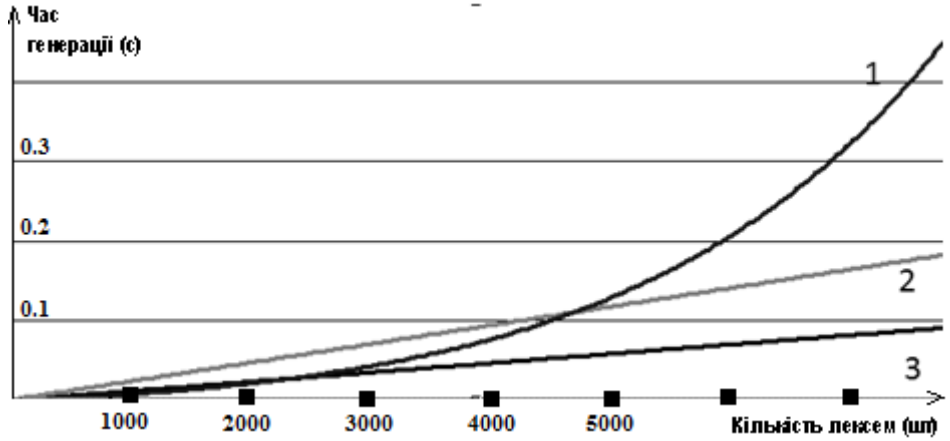

Рис. 5. Порівняльній графік швидкості генерації коду від складності проекту

На рис. 5 під номером *1* зображено динамічний метод генерації, під номером *2* метод шаблонів, під номером *3* статичній метод генерації. З рис. 5 видно, що динамічний метод генерації значно програє по швидкості іншим методам при великих розмірах проекту, але це спричинено тим, що методу необхідно виконувати додаткові проходи по коду та виконувати над ним складні аналізи для виявлення додаткових помилок. Незважаючи на більший час генерації коду, результуючий код є більш швидким та надійним, як наслідок ми зекономимо час на пошук та виправлення помилок, а також отримаємо кращий час виконання.

Як результат, цей метод найбільш прийнятний для генерації вихідних кодів на мовах опису апаратури, тому він взятий за базовий для реалізації генератору коду компонентів обчислювальної системи в базисі ПЛІС.

Список використаних джерел

1. *Новиков Ф. А.* Визуальное конструирование программ / Ф. А. Новиков // Информационно-управляющие системы. – 2005. – № 6. – С. 9-22.

2. *Селлз К.* Современные способы автоматизации повторяющихся задач программирования / К. Селлз // MSDN Magazine. – 2002. – № 6.

3. *Шалыто А. А.* SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем / А. А. Шалыто, Н. И. Туккель // Программирование. – 2001. – № 5. – С. 45-62.

УДК 621.397

К.Н. Григорьев, аспирант
Черниговский национальный технологический университет, г. Чернигов, Украина

МЕТОДЫ НАДЕЖНОЙ ДОСТАВКИ ВИДЕОСИГНАЛОВ ВЫСОКОГО КАЧЕСТВА

К.М. Григор'єв, аспірант
Чернігівський національний технологічний університет, м. Чернігів, Україна

МЕТОДИ НАДІЙНОЇ ДОСТАВКИ ВІДЕОСИГНАЛІВ ВИСОКОЇ ЯКОСТІ

Konstantin Grigoryev, PhD student
Chernigov National Technological University, Chernigov, Ukraine

METHODS OF GUARANTEED DELIVERY OF HIGH QUALITY VIDEO SIGNALS

На пути применения техники качественного видеопроизводства стоят ограничения по оперативной передачи сигнала в ТВ студии и Интернет медиа-сервера. Очевидной альтернативой системе спутниковой передачи видеосигнала является Интернет, однако недостаточная надежность параметров каналов Интернет вынуждает искать способы в области оптимизации протоколов передачи данных и алгоритмов кодирования видеосигналов. В ситуациях с «прямым эфиром» надежность системы производства и доставки видеосигнала является определяющей. Именно поэтому при использовании каналов Интернет важной является задача определения граничных параметров кодирования видеосигнала, а также выбор протоколов передачи данных и их параметров в условиях неопределенности и варьируемости параметров каналов передачи данных.

***Ключевые слова**: доставка видео, надежная передача потоковых данных, видео через Интернет, сетевые видеопротоколы, кодирование видеоданных.*

На шляху використання техніки створення якісного відео є обмеження щодо передавання сигналу до студії ТБ та Інтернет медіа-серверів. Очевидною альтернативою супутникової передачі відеосигналу є мережа Інтернет, але недостатня надійність параметрів каналів Інтернету змушує шукати способи оптимізації протоколів передавання даних та алгоритмів кодування відеосигналів. У випадках «прямого ефіру» надійність системи створення відео є визначальною. Саме тому під час використання каналів Інтернет важливим є завдання визначення граничних параметрів кодування відеосигналу, а також вибір протоколів передавання даних та їх параметрів в умовах невизначеності та варіативності параметрів каналів передавання даних.

***Ключові слова**: доставка відео, надійна передача потокових даних, відео через Інтернет, мережеві відеопротоколи, кодування відеоданих.*

Wide application of high definition video production is limited due to complications in live transmission to TV studio or to Internet media delivery networks. Internet is an obvious alternative to commonly used satellite transmission. However, insufficient reliability of Internet channels require optimized data transmission protocols and specialized video coding systems. Reliability of video production cycle and video delivery system are keys to cases of live video streaming. That is why