УДК 512.624.95(045)

РЕАЛІЗАЦІЯ АЛГОРИТМІВ ФОРМУВАННЯ ПСЕВДОВИПАДКОВИХ СЛІВ НА БАЗІ FPGA

Володимир Опанасенко, Станіслав Зав'ялов, Олександр Софіюк

У даній статті запропоновано апаратну реалізацію генераторів ПВС на базі кристалів FPGA, які використають принцип реконфигурируемости, що забезпечує можливість модернізації їх алгоритмів та оперативну зміну внутрішній структури (реконфігурацію) у процесі функціонування. Наявність в структурі кристалу FPGA вбудованих блоків DSP дозволяє ефективну апаратну реалізацію генераторів псевдовипадкової послідовності за рахунок реалізації основних операцій множення із накопиченням на вентильному рівні. За допомогою САПР ISE 14.02 Foundation шляхом описання мовою VHDL виконано проектування і реалізація трьох типів генераторів ПВС на базі кристалу серії Spartan6 (6SLX4CSG225-3), для яких наведено часові та апаратні витрати. Наведено часові діаграми моделювання цих структур, які отримано за допомогою системи моделювання ModelSim SE 10.1c.

Ключові слова: генератор псевдовипадкових послідовностей, САПР, DSP, FPGA.

ВСТУП

Сьогодні, у зв'язку з інтенсивним розвитком систем зв'язку з кодовим розподілом каналів, актуальною стає задача технічної модернізації пристроїв, які реалізують алгоритми генерації псевдовипадкових слів (ПВС) [1, 3], за допомогою сучасної елементної бази – програмовних логічних інтегральних схем (ПЛІС). ПЛІС або Programmable Logic Devices (PLD), які знаходять усе більше широке застосування в світі для створення сучасних систем управління, високопродуктивної обробки даних, цифрової обробки сигналів, підтримки телекомунікацій та інших [4, 5, 8].

Генерування ПВС (вибірка) виконується датчиками псевдовипадкових чисел. Кількість псевдовипадкових чисел при цьому знаходиться в досить широких межах: від десятків тисяч для простих задач, до сотень тисяч і більше для складних систем. Тому важливою задачею є забезпечення високої швидкодії.

Датчики з заданим законом розподілу (наприклад, нормальним, експоненціальним та іншими) реалізуються, як правило, програмно, їхня робота основана на перетворенні послідовності псевдовипадкових чисел із рівномірним розподілом в інтервалі [0, 1] в ПВС із заданим законом розподілу. Тому якість та ефективність процедур формування значною мірою залежать від властивостей датчика рівномірно розподілених псевдовипадкових чисел [2, 3].

Сьогодні існує велика кількість алгоритмів формування псевдовипадкових слів, які мають власні переваги і недоліки та використовуються в різних застосуваннях.

Найбільше поширення на практиці отримали лінійні конгруентні методи [2, 3, 9] генерації псевдовипадкових чисел із рівномірним розподілом та формуванням на їх основі ПВС заданої довжини, які мають задані властивості. В загальному вигляді алгоритм таких датчиків реалізується за допомогою рекурентного співвідношення:

$$x_{n+1} = \sum_{i=0}^{j} a_i x_{n-1} + c(\mathrm{mod}\, M), \qquad (1)$$

де: $a_0, a_1, ..., a_j$, $c > 1, M > 1$, а також отримані числа x_1, x_2, ... x_j є цілими числами. Модуль M означає: число $A = \sum_{i=0}^{j} a_i x_{n-1} + c$ ділиться на M; отримане ціле число q та цілочисельний залишок x_{n-1}, подається у вигляді:

$$A = qM + x_{n+1}; 0 \le x_{n-1} \le M - 1.$$

Так як x_{n+1} – число, яке знаходиться між 0 та M, то його ще необхідно поділити на M, щоб отримати число, яке знаходиться між 0 та 1:

$$R_{n+1} = \frac{x_{n+1}}{M}.$$

Послідовності, які отримано за допомогою лінійних конгруентних методів, періодично повторюються. Це пов'язано з тим, що числа x можуть приймати тільки значення $0, 1, 2, ..., (M-1)$. Максимальна довжина періоду послідовності не може перевищувати $M = 2m$, тому, як правило, приймають $m = N$, де N – число значущих розрядів для представлення цілих чисел.

Із співвідношення (1) можна отримати різні модифікації лінійних алгоритмів датчиків псевдовипадкових чисел.

Змішаний конгруентний метод генерування псевдовипадкових чисел, який запропонував Лемер, отримуємо із (1) при $a_1 = a_2 = ... = a_j = 0$ та приймаючи $a_0 > 0, c > 0$. Тоді:

$$x_{n+1} = a x_n + c(\mathrm{mod}\, M). \qquad (2)$$

Можна удосконалити алгоритм, який реалізує мультиплікативний конгруентний метод.

Для цього в (1) підставимо $c = a_1 = a_2 = ... = a_j = 0$ та приймаючи $a_0 > 0$. В цьому випадку:

$$x_{n+1} = ax_n + c .$$

Якість чисел, які обчислюються за цим алгоритмом, гірша, ніж в алгоритмі (2), але програма, яка його реалізує, простіше і дозволяє формувати числа із більш високою швидкодією. Це має значення при проведенні експериментів із імітаційними моделями, тому що зменшується час прогону.

Числа $c, M, a_0, a_1, ..., a_j, x_0$ називають *параметрами датчика.* x_0 – це початкове значення числа, з якого починається генерування вибірки. Якість генерування вибірки залежить від параметрів датчика, тому вони не можуть бути підібрані випадковим чином. Правила вибору параметрів лінійних датчиків розглянуто в [2].

АЛГОРИТМИ ФОРМУВАННЯ ПВС

Алгоритм 1.

ПВС формується генератором псевдовипадкової послідовності (ГПВП) за наступною формулою:

$$X_{i+1} = \left[A \times X_i + B_{i+1}\right], \qquad (3)$$

де: $B_{i+1} = B_i + 1$ (при переповненні B_i інформація в молодших розрядах не перекручується);

X_i – поточне n - розрядне слово ПВС;

N – кількість слів ПВС.

Оскільки результат X_{i+1} буде $2n$–розрядним, то беремо тільки n молодших розрядів результату.

Значення n– розрядних слів A, X_0 , B_0 є константою.

Блок-схема формування послідовності наведена на рис. 1.

Алгоритм 2.

ПВС формується генератором псевдовипадкової послідовності (ГПВП) за виразом (3).

Так як результат X_{i+1} буде $2n$– розрядним, то беремо тільки n розрядів (для непарних i – тільки n молодших розрядів, для парних i – тільки n середніх розрядів) результату $(i/2 = z + w)$, де: z – ціла, w – дробова частина результату ділення).

Блок-схема формування послідовності за алгоритмом 2 наведена на рис. 2.

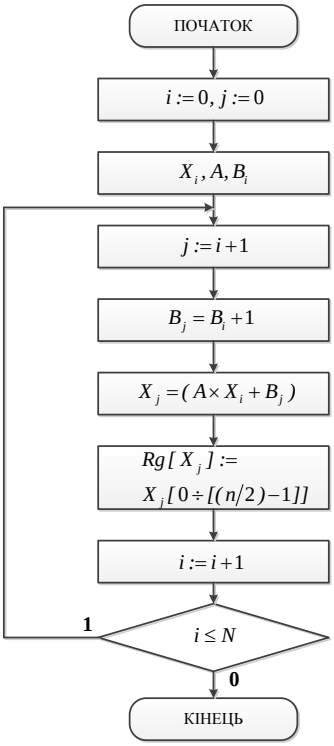

Рис. 1. Блок-схема обчислення псевдовипадкової послідовності за алгоритмом 1

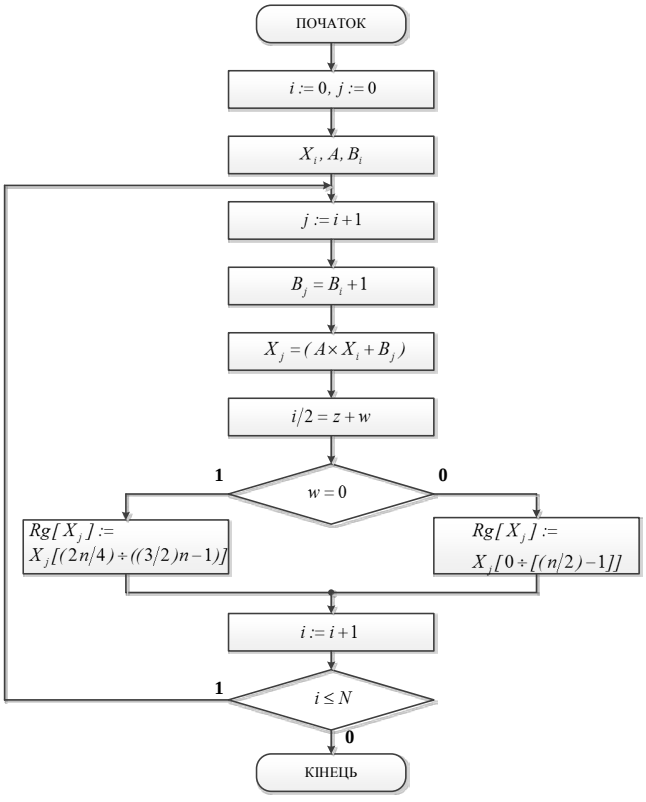

Рис. 2. Блок-схема обчислення псевдовипадкової послідовності за алгоритмом 2

Алгоритм 3.

ПВС формується генератором псевдовипадкової послідовності (ГПВП) за наступною формулою:

$$X_{i+1} = [A \times X_i (X_i + 1) + B_{i+1}], \qquad (4)$$

де: $B_{i+1} = B_i + 1$ (при переповненні B_i інформація в молодших розрядах не перекручується);

X_i – поточне 16-розрядне слово ПВС;

$(A \times X_i)$ – (береться тільки n молодших розрядів результату множення);

$(A \times X_i \times X_i)$ – (береться тільки n середніх розрядів результату множення).

Оскільки результат X_{i+1} буде $2n$–розрядним (при переповненні суми, інформація у молодших розрядах не перекручується), то беремо тільки n молодших розрядів результату. Блок-схема формування послідовності за алгоритмом 3 наведена на рис. 3.

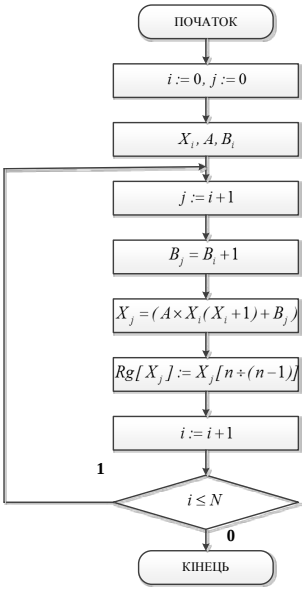

Рис. 3. Блок-схема обчислення псевдовипадкової послідовності за алгоритмом 3

РЕАЛІЗАЦІЯ АЛГОРИТМІВ ФОРМУВАННЯ ПВС

Вихідні дані для розробки алгоритму формування ПВС.

Розрядність псевдовипадкових слів – n біт. Кількість ПВС, яку необхідно отримувати – N. Шифрування кадру потребує N слів ПВС, то будемо визначати:

$$\langle X_1, B_1 \rangle, \langle X_2, B_2 \rangle, \ldots, \langle X_N, B_N \rangle.$$

Алгоритм 1.

Будемо використати формулу (1).

Цей алгоритм для реалізації на кристалі ПЛІС має наступну функціональну схему, що наведена на рис. 4 (де: $СчB_i$ – лічильник, що реалізує інкремент; $Rg(X_i)$ – регістр для зберігання n–розрядних значень X_i ; $Rg(A)$ – регістр для зберігання n–розрядної константи $Rg(A)$).

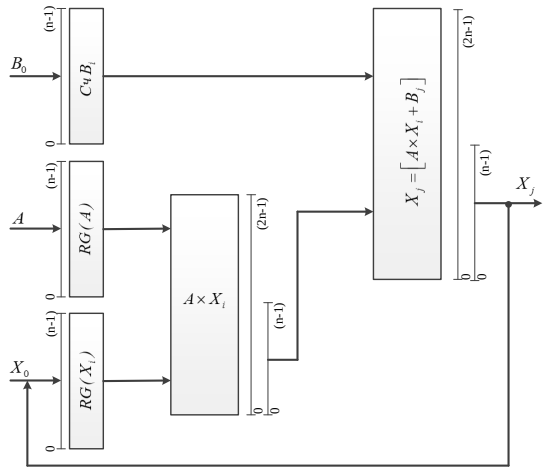

Рис. 4. Функціональна схема генератора псевдовипадкових слів, яка реалізує алгоритм 1

Алгоритм 2.

Будемо використати формулу (3).

Функціональна схема генератора псевдовипадкових слів на основі запропонованого алгоритму наведено на рис. 5 (де: МХ – мультиплексор, що передає n молодших розрядів результату X_j на вихід для непарних i , або n середніх розрядів для парних i).

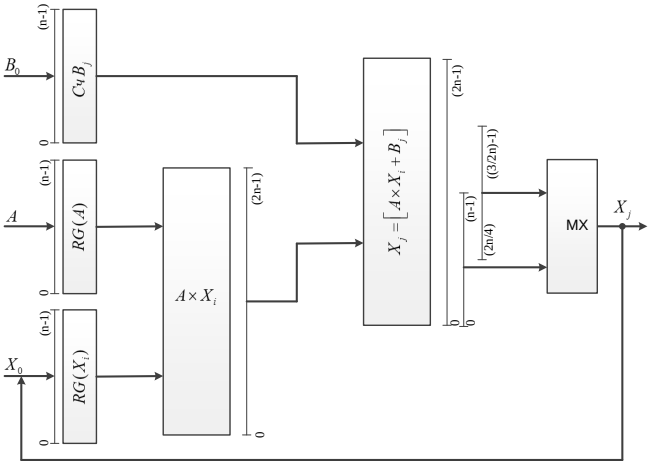

Рис. 5. Функціональна схема генератора псевдовипадкових слів, яка реалізує алгоритм 2

Алгоритм 3.

Будемо використати формулу (4).

Функціональна схема генератора псевдовипадкових слів на основі запропонованого алгоритму наведено на рис. 6.

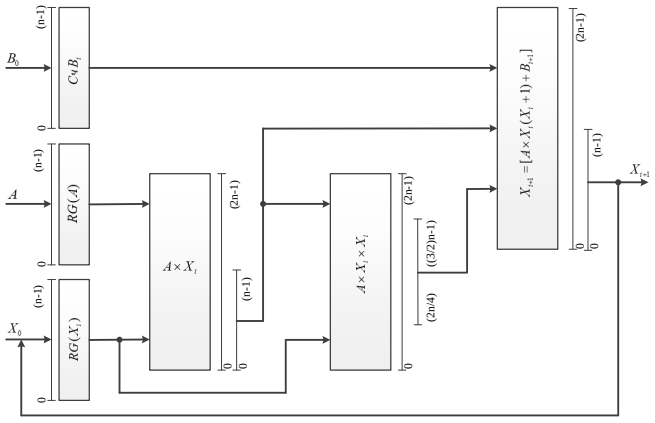

Рис. 6. Функціональна схема генератора псевдовипадкових слів, яка реалізує алгоритм 3

АНАЛІЗ СУЧАСНОЇ ЕЛЕМЕНТНОЇ БАЗИ ПЛІС

На відміну від традиційних засобів комп'ютерної техніки з програмною інтерпретацією алгоритмів, проблемно-орієнтовані засоби на основі кристалів ПЛІС реалізують повністю апаратну або змішану – програмно-апаратну інтерпретацію, також їх структура не є фіксованою й змінюється в залежності від виконуваної задачі (алгоритму). Підвищення продуктивності таких пристроїв забезпечується як за рахунок зазначеної реалізації алгоритмів, так і за рахунок високого ступеня паралелізму при виконанні завдання.

Впровадження кристалів ПЛІС типу FPGA (Field Programmable Gate Array – програмовні користувачем вентильні матриці) в реальні проекти стало можливим тільки з появою сучасних кристалів ПЛІС типу FPGA [4, 10].

Структура ПЛІС типу FPGA являє собою регулярне середовище, якому притаманна багаторазова реконфігуровність. Логічна ємність сучасних ПЛІС забезпечує можливість створення в одному кристалі великої кількості апаратних фрагментів, що реалізують задані алгоритми. Високий ступінь паралелізму досягається за рахунок одночасного виконання алгоритмів, а багаторазова (динамічна) реконфігурація всього кристала або його частини в процесі виконання завдання дозволяє реалізувати багатофункціональні або складні системи з істотною економією апаратних ресурсів.

До інших переваг ПЛІС варто віднести:

– універсальність, тобто можливість створення практично будь-якого цифрового пристрою в кристалі при наявності персонального комп'ютера та відповідних інструментальних засобів (САПР). Для виконання проектів на базі ПЛІС фірми Xilinx використовують систему проектування – ISE (Integrated Synthesis Environment) Foundation, яка орієнтована на використання HDL–технології;

– високу швидкодію, малу споживану потужність і високу надійність, що забезпечуються технологією виготовлення кристалів;

– низьку в порівнянні з замовленими та напівзамовленими НВІС вартість реалізації проектів за рахунок масового виробництва кристалів з регулярною структурою і невеликим часом, що витрачається на розробку проектів та їх верифікацію.

Сімейство кристалів ПЛІС фірми Xilinx (Virtex і Spartan) представляє собою архітектуру FPGA, являє собою найбільш сучасну технологічну платформу, яка оптимізована для використання як hard, так й soft cores.

Hard cores (типу DSP) – фіксовані логічні проекти, які вбудовані в архітектуру кристала FPGA. Архітектура Virtex і Spartan дозволяє легко інтегрувати обидва типи cores у проекти, надаючи максимальну гнучкість [10, 11]. Надалі hard cores будуть усе більш інтенсивно інтегруватися в FPGA платформу для збільшення характеристик і легкості у використанні. Прикладами є центральні процесори, блоки пам'яті, керування синхронізацією, блоки DSP та високошвидкісні системи введення–виведення.

У внутрішній області кристалів FPGA розташовано: матриця регулярно розташованих ідентичних конфігуровних логічних блоків – Configurable logic blocks (CLB) між якими проходять канали трасувань, блоків статичної пам'яті (Block RAM), модулів цифрової обробки сигналів Digital Signal Processing (DSP), швидкодіючих приймачів-передавачів (Rocket IO transceivers GTP/GTX/ GTH/ GTZ) а також аналого-цифрові перетворювачі. На границях кристала розташовано матрицю блоків вводу/виводу – Input/output Blocks (IOB) та засоби синхронізації Clocks & Delay-Locked Loop (DLL). Загальна схема FPGA фірми Xilinx [4] приведена на рис. 7.

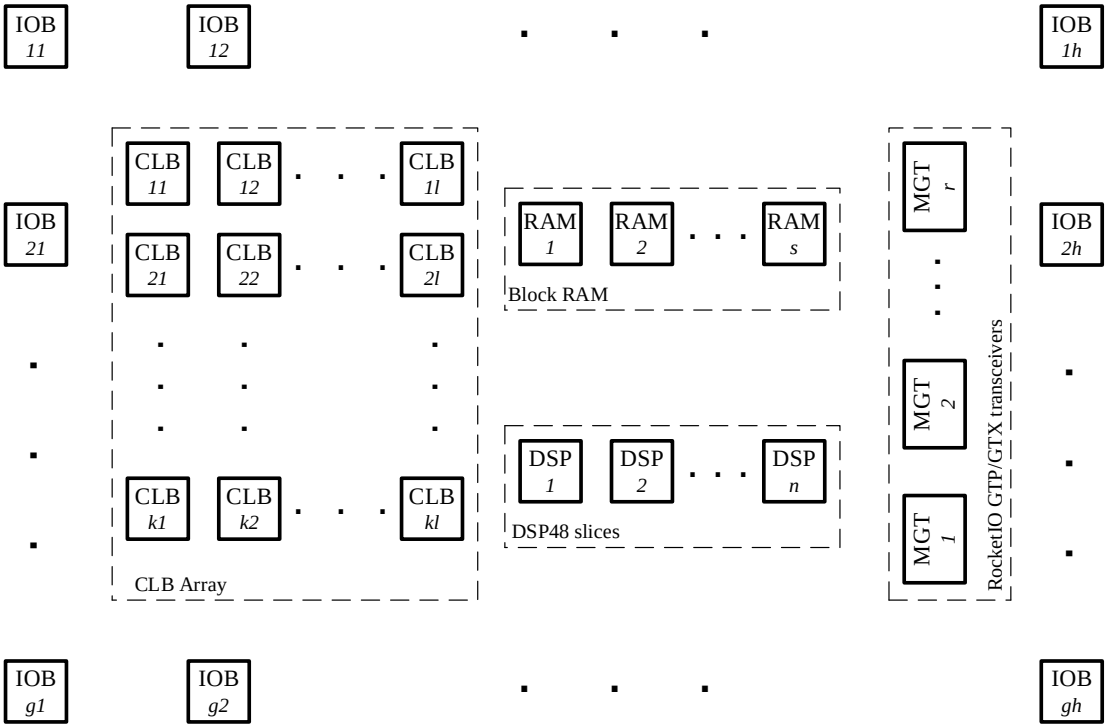

Рис. 7. Загальна структура FPGA фірми Xilinx

Основні логічні ресурси кристалу розташовано в конфігуровних логічних блоках CLB.

Елементарні програмовні логічні блоки в FPGA фірми Xilinx називаються секціями (slices). Ця архітектура (з незначними модифікаціями) використовуються в усіх сімействах Virtex FPGA до сімейства Virtex-7 та в усіх сімействах Spartan.

Розглянемо для прикладу секції Virtex-4, які складаються з:

– двох 4-входових переглядових таблиць Look-Up Tables (LUTs), тобто логічних модулів на основі ПЗП, які можуть реалізувати будь-яку логічну функцію (4 змінних), які використовуються як генератори комбінаційних функції;

– двох мультиплексорів для комбінаційної логіки, які контролюються користувачем. Один з мультиплексорів може бути використаний для об'єднання виходів блоків LUT щоб реалізувати функцій від 5 й більше змінних. Другий мультиплексор використовується для об'єднання виходів першого мультиплексора й мультиплексорів інших секцій;

– арифметичної логіки (два 1-розрядних з'єднувача, ланцюг суматорів та два виділених шлюзи для реалізації швидкого і ефективного множення);

– двох 1-розрядних регістра, які можуть конфігуруватися як тригери, вхід до цих тригерів вибираються за допомогою мультиплексорів.

Головною відмінністю у кристалах Virtex−5 від попередньої архітектури є збільшення логічної потужності секції за рахунок збільшення кількості входів (з чотирьох до п'яти) двох блоків LUT, а також збільшення кількості секцій в CLB − від двох до чотирьох.

У сучасних кристалах серії Virtex−6,7 для побудови секції використана технологія повноцінних 6-входових переглядових таблиць LUT, кожна секція має два тригера, у відмінності від попередніх сімейств (тільки один тригер), а взагалі CLB має вісім тригерів.

Починаючи з сімейства Virtex-4 фірма Xilinx представила блок DSP48 [11] для високошвидкісної цифрової обробки сигналів. Взагалі це ядро – помножувач-накопичувач з багатьма іншими особливостями. Також ці блоки DSP є складовою частиною кристалів серій Spartan3A (DSP48A), Spartan3A (DSP48A1), Virtex-5 (блок DSP48E) та Virtex-6,7 (блок DSP48E1). Основні характеристики блоків DSP48 зведено в табл. 1.

Як загальну тенденцію можна відзначити конвеєризацію і каскадування, які полегшують побудову високочастотних блоків великої розрядності. Для сімейств Virtex-5,6,7 важливим є наслідком переходу до формату 18x25 (замість 18x18 у Virtex-4) стала можливість виконувати множення мантис чисел з плаваючою точкою одинарної точності при використанні лише двох секцій (замість чотирьох в Virtex-4).

Таблиця 1

Параметри блоків цифрової обробки сигналів DSP48

Параметри	*Типи блоків*				
	DSP48 (Virtex-4)	DSP48A (Spartan3A)	DSP48A1 (Spartan6)	DSP48E (Virtex-5)	DSP48E1 (Virtex-6,7)
	Кількість слайсів у кристалі				
	32–192	84–126	8–180	32–1.056	288–3.960
Робоча частота, МГц	500	250		550	600
Помножувач	18x18	18x18	18x18	25x18	25x18
Попередній суматор (Pre-Adder)	–	–	–	–	+
Каскадовані входи	1	1	1	2	2
Підтримка SIMD	–	–	–	+	+

Кожен блок DSP48 має спеціальний перемножувач та акумулятор і реалізує наступні функції, які широко використаються в задачах цифрової обробки сигналів:

$$Adder\ Out = (Z \pm (X + CIN));$$

$$Adder\ Out = C \pm (A \times B + CIN);$$

$$Adder\ Out = C \pm (A \times (D \pm B) + CIN).$$

Для розробки пристроїв на ПЛІС використовуються інструментальні пакети САПР фірми Xilinx [6]. Набір інструментальних засобів розробки пристроїв на ПЛІС, починаючи з версії 10.1 називаються ISE Design Suite. Це комерційний набір програм й утиліт, об'єднаних загальним середовищем, що дозволяє розробляти проекти для всіх кристалів фірми Xilinx.

Програма PROMGEN (Генератор файлів ППЗП) перетворить BIT–файл у файл формату PROM. Файл ППЗП містить дані конфігурації FPGA, що зберігаються в ППЗП. PROMGEN перетворює BIT–файл в один з чотирьох форматів PROM.

АПАРАТНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ФОРМУВАННЯ ПСЕВДОВИПАДКОВИХ СЛІВ

Розглянемо приклад розробки генератора псевдовипадкових чисел шляхом опису мовою VHDL за допомогою пакета ISE Foundation, його моделювання за допомогою системи ModelSim [7] на базі кристалу 6SLX4CSG225-3 серії Spartan6 [10] для розрядності псевдовипадкових слів – n біт та кількості ПВС, які необхідно отримувати – N.

Етапи розробки передбачають верифікацію проекту методом моделювання, у процесі якої на входи логічної моделі проектованого пристрою подаються вхідні впливи у вигляді віртуальних сигналів (test–bench), сформованих розроблювачем, тобто за допомогою стенду, опис якого наведено нижче.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY PVS_1_TB IS
END PVS_1_TB;

ARCHITECTURE behavior OF PVS_1_TB IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT PVS_1
    PORT(
         CLK    : IN  std_logic;
         A           : IN  std_logic_vector(15 downto 0);
         X0     : IN  std_logic_vector(15 downto 0);
         B0     : IN  std_logic_vector(15 downto 0);
         X1     : OUT  std_logic_vector(15 downto 0);
         X2     : OUT  std_logic_vector(15 downto 0);
         X3     : OUT  std_logic_vector(15 downto 0);
```

```
         X4     : OUT  std_logic_vector(15 downto 0);
         X5     : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal CLK   : std_logic := '0';
   signal A     : std_logic_vector(15 downto 0) := (others => '0');
   signal X0    : std_logic_vector(15 downto 0) := (others => '0');
   signal B0    : std_logic_vector(15 downto 0) := (others => '0');

     --Outputs
   signal X1    : std_logic_vector(15 downto 0);
   signal X2    : std_logic_vector(15 downto 0);
   signal X3    : std_logic_vector(15 downto 0);
   signal X4    : std_logic_vector(15 downto 0);
   signal X5    : std_logic_vector(15 downto 0);

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

     -- Instantiate the Unit Under Test (UUT)
   uut: PVS_1 PORT MAP (
          CLK   => CLK,
          A     => A,
          X0    => X0,
          B0    => B0,
          X1    => X1,
          X2    => X2,
          X3    => X3,
          X4    => X4,
          X5    => X5
        );

    tb : PROCESS
   BEGIN
      CLK <= '1'; wait for 12.5 ns;
      CLK <= '0'; wait for 12.5 ns;
   END PROCESS;

    tb1 : PROCESS
   BEGIN
      A <= X"1357"; wait;
   END PROCESS;

    tb2 : PROCESS
   BEGIN
      X0 <= X"2468"; wait;
   END PROCESS;

    tb3 : PROCESS
   BEGIN
      B0 <= X"ABCD"; wait;
   END PROCESS;

END;
```

Результати моделювання (часові діаграми) запропонованих генераторів випадкових слів для відповідних алгоритмів наведено на рис. 8-10.

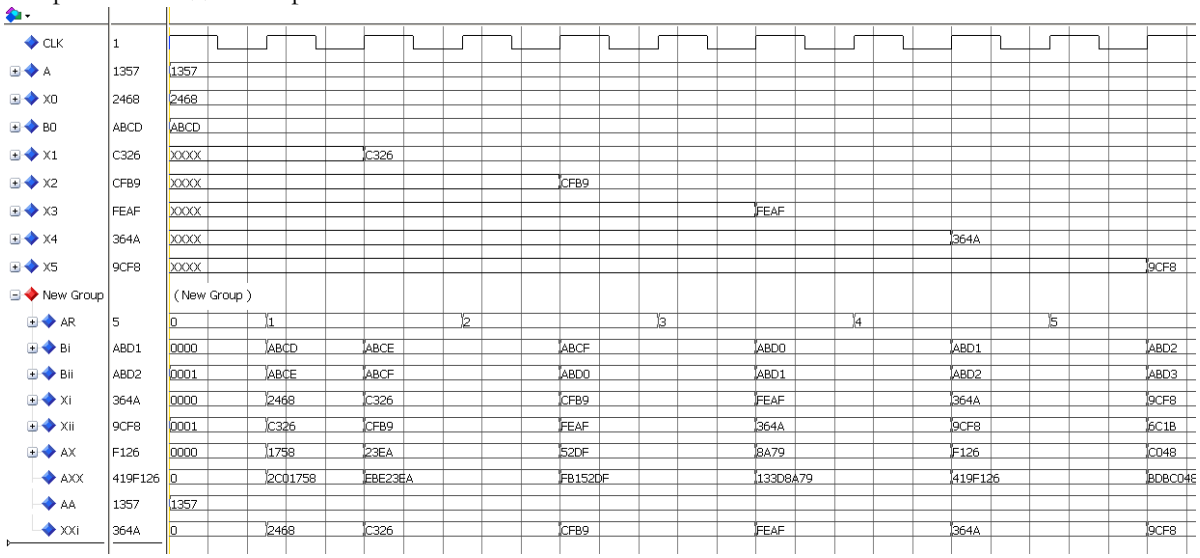

Рис. 8. Часова діаграма работи генератора ПВС за алгоритмом 1

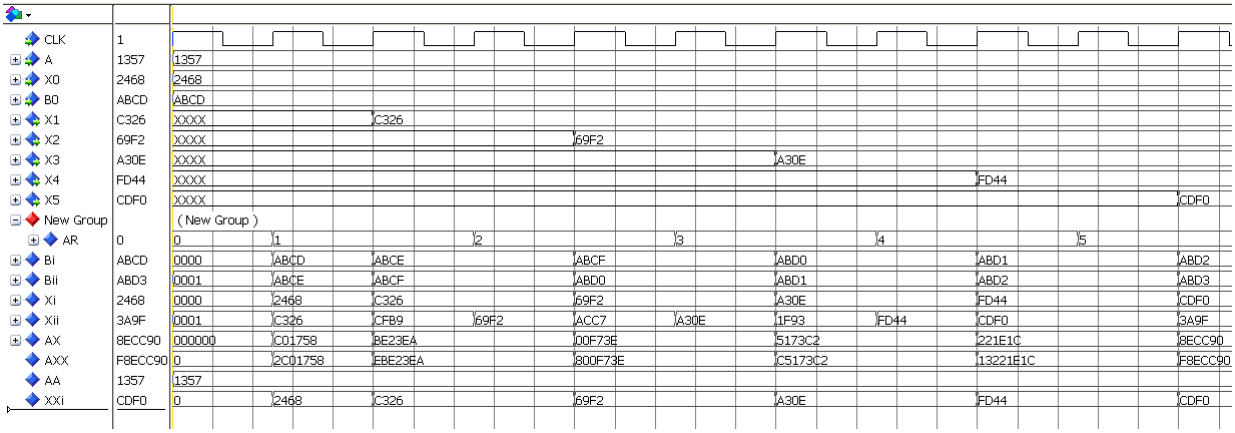

Рис. 9. Часова діаграма работи генератора ПВС за алгоритмом 2

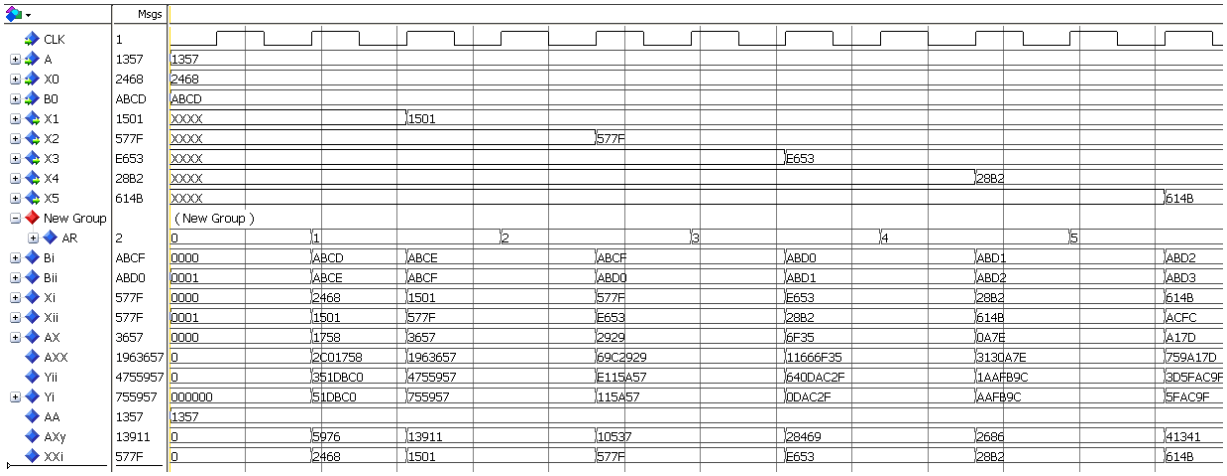

Рис. 10. Часова діаграма работи генератора ПВС за алгоритмом 3

Отримані дані буде використано в процесі шифрування слів, для передавання по радіолокаційної лінії, а також в процесі дешифрування слів після прийому посилок на об'єкті.

У результаті реалізації генераторів ПВС за трьома алгоритмами отримано наступні характеристики, наведені в табл. 2.

Таблиця 2

Оцінки апаратних і часових витрат при реалізації генераторів ПВС

	T_{CLK}, нс	DSP		Кількість Tg			Кількість LUTs		
		Викор.	Дост.	Викор.	Дост.	%	Викор.	Дост.	%
Алгоритм 1	5,773	1	8	103	4800	2	108	2400	4
Алгоритм 2	10,750	2	8	103	4800	2	92	2400	3
Алгоритм 3	6,224	2	8	87	4800	1	45	2400	1

ВИСНОВКИ. Виконано аналіз архітектурно-структурної організації кристалів ПЛІС типу FPGA, зокрема сімейств Virtex, які дозволяють виконувати велике різноманіття алгоритмів завдяки своїй гнучкій системі налаштування, що в свою чергу дозволяє реалізовувати алгоритм майже будь-якої складності. Основні переваги використання ПЛІС для побудови проблемно-орієнтованих процесорів:

– принцип реконфігуровності – можливість модифікації процесорів (зміна архітектури й структури пристроїв, тобто зміна алгоритмів функціонування системи) на будь-яких стадіях розробки та у процесі експлуатації;

– високу швидкодію, малу споживану потужність і високу надійність, що забезпечуються технологією виготовлення кристалів;

– значне скорочення часу впровадження нових перспективних виробів на основі ПЛІС (час проектування, моделювання, верифікації та реалізації) при незначних затратах, а застосування ПЛІС для виробництва одиничних екземплярів – це єдино прийнятний варіант.

Розроблено генератор на основі ПЛІС для реалізації алгоритмів генерації псевдовипадкових слів за допомогою системи автоматизованого проектування Xilinx ISE (САПР) ISE 14.02 Foundation фірми Xilinx и виконано його моделювання за допомогою системи ModelSim SE 10.1c. Перевага розроблених пристроїв над існуючими аналогами полягає у використанні принципу реконфігуровності для побудови високопродуктивних комп'ютерних засобів, що забезпечує можливості модернізації алгоритмів та оперативну заміну їх структури (реконфігурацію) в процесі функціонування.

ЛІТЕРАТУРА

[1]. Кнут Д.Э. Искусство программирования: В 7 т.; [пер. с англ. В. Тертышный]. – [3-е изд.]. – М.: «Вильямс», 2007. – Т.2: Получисленные алгоритмы. – 832 с.

[2]. Корчинский В.В., Филькин К.М. О выборе первичного датчика для задач имитационного моделирования // Моделювання та інф. технології. Збірн. наук. праць ІПМЕ. – 2007. – Вип. 42. – С. 81-90.

[3]. Лавданский А.А. Оценка качества генераторов псевдослучайных чисел по величине ошибки воспроизведения закона распределения / А.А. Лавданский // Вісник Хмельницького національного університету. – 2014. – №1. – С. 113–116.

[4]. Палагин А.В. Реконфигурируемые вычислительные системы/ А.В. Палагин, В.Н. Опанасенко. – Киев: Просвіта. – 2006. – 295 с.

[5]. Палагин А.В. Проектирование реконфигурируемых систем на ПЛИС / А.В. Палагин, В.Н. Опанасенко, А.Н. Лисовый // Технология и конструирование в электронной аппаратуре. – 2007. – № 3. – С. 15-20.

[6]. Available at http://www.xilinx.com/products/design-tools/ise-design-suite.html.

[7]. ModelSim. ASIC and FPGA design / Available at http: // www.mentor.com/ products/ fv/ modelsim/

[8]. Palagin A. The structure of FPGA-based cyclic-code converters / Alexander Palagin, Vladimir Opanasenko, Sergey Krivoi // Optical Memory & Neural Networks (Information Optics). Springer. – 2013, Vol. 22, N.4. – PP. 207–216.

[9]. Random Number Generator Results [Електронний ресурс]. – Режим доступу: http://www.cacert.at/cgi-bin/rngresults.

[10]. Spartan-6 Family Overview / Product Specification // DS160 (v2.0) October 25, 2011. – Xilinx, Inc. – 11 p.

[11]. Spartan-6 FPGA DSP48A1 Slice / User Guide // UG389 (v1.2) May 29, 2014. – Xilinx, Inc. – 46 p.

REFERENCES

[1]. Knut D.E. *Art of Computer Programming. 3th ed. Vol. 2 Seminumerical algorithms*: (Russ. ed.: Tertyshnyi, V. *Iskusstvo programmirovaniya***.** *Tom 2. Poluchislennye algoritmy* [per. s angl.], Moscow, Vil'yams Publ., 2007. 832 p.).

[2]. Korchinskij V.V., Fil'kin K.M. O vybore pervichnogo datchika dlja zadach imitacionnogo modelirovanija [On the choice of the primary sensor for the simulation tasks]. *Modeljuvannja ta inf. tehnologij. Zbirn. nauk. prac' IPME*, 2007,no. 7, pp. 81-90. (In Russian)

[3]. Lavandsky A.A. Otsenka kachestva generatorov psevdosluchainykh chisel po velichine oshibki vosproizvedeniya zakona raspredeleniya [Quality assessment of pseudo-random number generators by argest reproduction error distribution law]. *Visnyk Khmel'nyts'koho natsional'noho universytetu,* 2014, no. 1, pp. 113-116. (In Russian)

[4]. Palagin A.V., Opanasenko V.N. *Rekonfiguriruemye vychislitel'nye sistemy* [Reconfigurable computing systems]. Kiev, Prosvita Publ., 2006. 295 p. (In Russian)
[5]. Palagin A., Opanasenko V. (2013), "The structure of FPGA-based cyclic-code converters", *Optical Memory & Neural Networks (Information Optics)*, Vol. 22, No.4, pp. 207–216.
[6]. Available at http://www.xilinx.com/products/design-tools/ise-design-suite.html/ (Accessed 12 May 2016).
[7]. ModelSim. ASIC and FPGA design. Available at http: // www.mentor.com/products/fv/modelsim/ (Accessed 28 April 2016).
[8]. Palagin A.V., Opanasenko V.N., Lisovyi A.N. Proektirovanie rekonfiguriruemykh sistem na PLIS [Design of the reconfigurable FPGA-based systems]. *Tekhnologiya i konstruirovanie v elektronnoi apparature*, 2007, no. 3, pp. 15-20. (In Russian)
[9]. Random Number Generator Results / Available at: http://www.cacert.at/cgi-bin/rngresults/ (Accessed 14 April 2016).
[10]. Spartan-6 Family Overview. Product Specification. DS160 (v.2.0), Xilinx, Inc., October 25, 2011, 11 p. (Accessed 17 April 2016).
[11]. Spartan-6 FPGA DSP48A1 Slice. User Guide. UG389 (v1.2), Xilinx, Inc., May 29, 2014, 46 p. (Accessed 17 April 2016).

РЕАЛИЗАЦИЯ АЛГОРИТМОВ ФОРМИРОВАНИЯ ПСЕВДОСЛУЧАЙНЫХ СЛОВ НА БАЗЕ FPGA

В данной статье предлагается аппаратная реализация генераторов ПСС на базе кристаллов FPGA, которые используют принцип реконфигурируемости, что обеспечивает возможность модернизации их алгоритмов и оперативную замену внутренней структуры (реконфигурацию) в процессе функционирования. Наличие в структуре кристалла FPGA встроенных блоков DSP позволяет эффективную аппаратную реализацию генераторов псевдослучайных последовательностей за счет реализации основных операций умножения с накоплением на вентильном уровне. С помощью САПР ISE 14.02 Foundation путем описания на языке VHDL выполнены проектирование и реализация трех типов генераторов ПСС на базе кристалла серии Spartan6 (6SLX4CSG225-3), для которых приведены временные и аппаратурные затраты. Приведены временные диаграммы моделирования этих структур, полученные с помощью системы моделирования ModelSim SE 10.1c.

Ключевые слова: генератор псевдослучайных последовательностей, моделирование, САПР, DSP, FPGA.

FPGA-BASED REALIZATION OF GENERATION ALGORITHMS PSEUDORANDOM WORDS

A hardware implementation of PSS generator based on FPGA crystals, which use the principle of reconfigurability that allows the modernization of their algorithms and online replacement of the internal structure (reconfiguration) in the process of functioning in this paper are proposed. Available embedded DSP blocks into structure of crystal FPGA allows efficient hardwarily implement the pseudorandom bit generator through the implementation of the basic operations of multiplication with accumulation on the gate level. With using of CAD ISE 14.02 Foundation by describing by means VHDL language implemented design and implementation on Spartan-based series crystal (6SLX4CSG225-3) for three types of the pseudorandom bit generators, for which time and hardware expenses are represented. With using the simulating system ModelSim SE 10.1c are obtained timing diagrams of simulation for these structures.

Key words: pseudorandom bit generator, simulation, CAD, DSP, FPGA.

Опанасенко Володимир Миколайович, доктор технічних наук, професор, провідний науковий співробітник відділу мікропроцесорної техніки Інституту кібернетики ім. В.М. Глушкова НАН України.
E-mail: opanasenko@incyb.kiev.ua
Опанасенко Владимир Николаевич, доктор технических наук, профессор, ведущий научный сотрудник отдела микропроцессорной техники Института кибернетики им. В.М. Глушкова НАН Украины.
Opanasenko Volodymyr, Doctor of Science, Professor, Leading Researcher of the Department of Microprocessor Devices of Institute of Cybernetics of the National Academy of Science of Ukraine.

Зав'ялов Станіслав Борисович, кандидат технічних наук, директор ООО «Радіонікс».
E-mail: radionix13@gmail.com
Завьялов Станислав Борисович, кандидат технических наук, директор ООО «Радионикс».
Zavyalov Stanislaw, PhD, director of "Radionix" Limited Liability Company.

Софіюк Олександр Танасович, науковий співробітник відділу мікропроцесорної техніки Інституту кібернетики ім. В.М. Глушкова НАН України.
E-mail: otsof@yandex.ua
Софиюк Александр Танасович, научный сотрудник отдела микропроцессорной техники Института кибернетики им. В.М. Глушкова НАН Украины.
Sofiyuk Alexander, junior scientist of the Department of Microprocessor Devices of Institute of Cybernetics of the National Academy of Science of Ukraine.