ДОДАТОК А Графічна частина атестаційної роботи

Харківський національний університет радіоелектроніки

Кафедра АПОТ

Кваліфікаційна робота магістра

Моделі та методи проектування апаратного біт-потокового обчислювача дробово-раціональних функцій

Магістранта групи СКСм-20-1
Ющенко Сергія
Валерійовича

Керівник:
доц. каф. АПОТ
Ларченко Л.В.

Kharkov National University of Radio Electronics

Біт-потокові функціональні обчислювачі в розподілених системах управління

Система управління реального часу

У розподілених системах управління реального часу основою комплексу систем є обчислювальна система, що здатна:

- прийняти та провести аналіз інформаційних даних про стан об'єкту управління;
- провести обробку даних, порівняти отриману інформацію з задачами та цілями управління;
- сформувати за результатами порівняння відповідні керуючі впливи на об'єкт управління.

Система управління реального часу містить компоненти:

- **об'єкт управління**;
- **датчики, сенсори та перетворювачі**;
- **виконавчі пристрої (актюатори);**
- **підсистема управління реального часу.**

В системах управління сигнали отримують від сенсорів фізичних величин для прийняття рішень про результати вимірювань з метою реалізації задач управління.

Об'єкт управління, **сенсори** та **перетворювачі** створюють інтерфейс зв'язку.

Інтерфейси зв'язку систем управління мають у своєму складі спеціалізовані пристрої – **функціональні перетворювачі** та **обчислювачі**, що працюють з потоковими формами даних.

Kharkov National University of Radio Electronics 2

Галузі застосування біт-потокових обчислювачів математичних функцій

- в розподілених системах управління реального часу в якості функціональних перетворювачів та обчислювачів частотних та час-імпульсних сигналів, отриманих від сенсорів фізичних величин;
- в якості зовнішніх апаратних модулів потокової обробки даних в архітектурах потокових процесорів;
- при вирішенні завдань узгодження сенсорів з цифровими системами збору і обробки даних для удосконалення інтерфейсів зв'язку;
- в сучасних системах управління в якості спеціальної апаратури їх спряження з виконавчими органами об'єкту керування;
- при відтворенні траєкторій рухомих об'єктів в двомірному і тривимірному просторі;

МЕТА і ПОСТАНОВКА ЗАВДАННЯ

Метою кваліфікаційної роботи є розробка моделей та методів автоматизованого проектування біт-потокового обчислювача дробово-раціональних функцій на основі ПЛІС.

Завдання дослідження:

- аналіз особливостей функціонального перетворення бітових потоків в апаратних обчислювачах математичних функцій;
- аналіз способу обчислення дробово-раціональних функцій, аргумент яких представлений бітовим потоком;
- розробка математичної моделі біт-потокового обчислювача дробово-раціональної функції;
- аналіз способу побудови конвеєрних архітектур біт-потокових обчислювачів на основі алгоритму обчислення поліноміальних функцій;
- розробка архітектурної моделі обчислювача заданої функції;
- розробка апаратної реалізації пристрою на основі кінцевого автомата;
- розробка HDL-моделі обчислювача на основі автоматного опису;
- верифікація, тестування та імплементація отриманої моделі обчислювача в платформу ПЛІС.

Особливості функціонального перетворення бітових потоків даних

Інформативним параметром в біт-потоковому кодуванні є **фіксоване значення імпульсів (біт) за часовий інтервал.**

Потокові способи передачі та обробки даних характеризуються:

- можливістю реалізації перетворення за рахунок використання методів формування приростів і послідовної обробки потоків у міру надходження одиничних імпульсів потоку;
- високою завадостійкістю, обумовленою непозиційністю і ваговою рівнозначністю біт в імпульсному потоці.

Особливістю функціонального перетворення бітових потоків даних в online-обчислювачах елементарних математичних функцій є **реалізація потокового способу обчислень.**

- **Потоковий спосіб обчислень** полягає в розгортці кодової інформації в часі з одночасним паралельно-послідовним виконанням перетворень над одиничними бітами вхідного потоку у відповідності до заданої функції.
- При цьому здійснюється **послідовне обчислення значень функції**, що виконуються для сусідніх значень аргументу.
- **Кожне наступне значення** функції визначається на підставі попереднього результату обчислень.
- **Перше обчислення** виконується з урахуванням вводу початкових умов (ініціалізації компонентів пристрою).

Досліджувана функція

Апаратний обчислювач відтворює дробово-раціональну функцію $y = [\frac{\sum_{i=0}^{2} a_i x^i}{m} + |\delta_{max}|]$, (1)

що апроксимує неперервну функцію $y^* = \frac{\sum_{i=0}^{2} a_i x^{*i}}{m}$, (2)

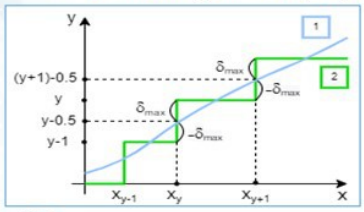
де x – аргумент функції, що представляє собою бітовий потік даних;
$|\delta_{max}|$ – граничне значення абсолютної похибки обчислення функції.

Абсолютна похибка обчислення дробово-раціональної функції виникає при поділі полінома на число m і може бути забезпечена 0,5 одиниці молодшого біту числа x, що є раціональним, $|\delta_{max}| = 0,5$.

Отже, обчислювач реалізує функцію $y = [\frac{\sum_{i=0}^{2} a_i x^i}{m} + 0,5]$. (3)

Обчислення функції в пристрою здійснюється виконанням двох операцій над бітовим потоком x:

- обчислення значення полінома другого порядку $\sum_{i=0}^{2} a_i x^i$;
- ділення отриманого результату на число m з похибкою $|\delta_{max}| = 0,5$.



На графіку:

1 - неперервна дробово-раціональна функція $y^* = \frac{\sum_{i=0}^{2} a_i x^{*i}}{m}$

2 - апроксимуюча дробово-раціональна функція що відтворює неперервну на виході пристрою. $y = [\frac{\sum_{i=0}^{2} a_i x^i}{m} + 0,5]$,

Математична модель біт-потокового обчислювача дробово-раціональних функцій

Для дробово-раціональної функції $y = [\frac{\sum_{i=0}^{2} a_i x^i}{m} + 0,5]$ була визначена формула загального члена числової послідовності x_y, що відповідає вузлам апроксимації заданої функції і має вигляд:

$$2\sum_{i=0}^{2} a_i x_y^i \geq m\,(2y_k - 1) \qquad (4)$$

На основі нерівності (4) було отримано математичну модель обчислювача дробово-раціональних функцій, яку можна представити системою різницевих нерівностей:

$$\begin{aligned}
&2\sum_{i=0}^{2} a_i x_1^i \geq m \\
&2(\sum_{i=0}^{2} a_i x_2^i - \sum_{i=0}^{2} a_i x_1^i) + \Delta_1 \geq 2m \\
&2(\sum_{i=0}^{2} a_i x_3^i - \sum_{i=0}^{2} a_i x_2^i) + \Delta_2 \geq 2m \qquad (5)\\
&\cdots\cdots\cdots\cdots \\
&2(\sum_{i=0}^{2} a_i x_y^i - \sum_{i=0}^{2} a_i x_{y-1}^i) + \Delta_{y-1} \geq 2m,
\end{aligned}$$

де $\Delta_{y-1} = 2(\sum_{i=0}^{2} a_i x_y^i - \sum_{i=0}^{2} a_i x_{y-1}^i) + \Delta_{y-2} - 2m.$

Реалізацію системи нерівностей (5) можна здійснити обчислювачем, шляхом одночасного обчислення приростів ґратчастих функцій $2\sum_{i=0}^{2} a_i x_y^i$ і $m(2y_k - 1)$, порівняння їх поточних значень з урахуванням різниці, отриманої на попередньому кроці обчислень Δ_{y-1}.

При надходженні на вхід пристрою певного біта x_y бітового потоку x на його виході буде сформований вихідний біт y_k при виконанні кожної нерівності системи (5).

Алгоритм конвеєрних обчислень в біт-потокових обчислювачах дробово-раціональних функцій

Поліноміальна функція має вигляд $y = \sum_{i=0}^{n} a_i x^i$. (1)

Послідовність цілочисельних значень $y_0, y_1, y_2, y_3, \ldots, y_i$, що відповідають значенням аргументу $x = 0, 1, 2, 3, \ldots, i$ є арифметичним рядом n-го порядку.

Задача синтезу біт-потокових обчислювачів поліномів вирішується шляхом зниження порядку різниць.

Значення функції визначаються за формулою: $y_i = f(i+1) - f(i)$, (2)

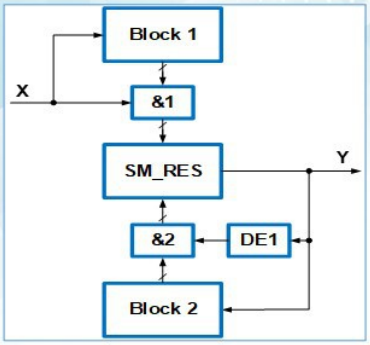
де $i = 0, 1, 2, 3, \ldots$

Алгоритм обчислення арифметичних рядів n-го прядку та його різниць представлено системою різницевих рівнянь:

$$\begin{aligned}
&y_i = f(i+1) - f(i) \\
&\Delta_i = y_{i+1} - y_i \\
&\Delta_i^2 = \Delta_{i+1} - \Delta_i \qquad (3)\\
&\cdots\cdots\cdots \\
&\Delta_i^n = \Delta_{i+1}^{n-1} - \Delta_i^{n-1}
\end{aligned}$$

Розглянута методика може бути використана при проектуванні конвеєрної архітектури біт-потокового обчислювача дробово-раціональної функції.

Узагальнена архітектура біт-потокового обчислювача



Block 1 – блок відтворення функції $2\sum_{i=0}^{n} a_i x_y^i$

Block 2 – блок відтворення функції $m(2y_k - 1)$

&1, &2 – група елементів &

DE1 – елемент затримки

Block1 і **Block2** формують прирости ґратчастих функцій лівої та правої частин нерівностей математичної моделі обчислювача.

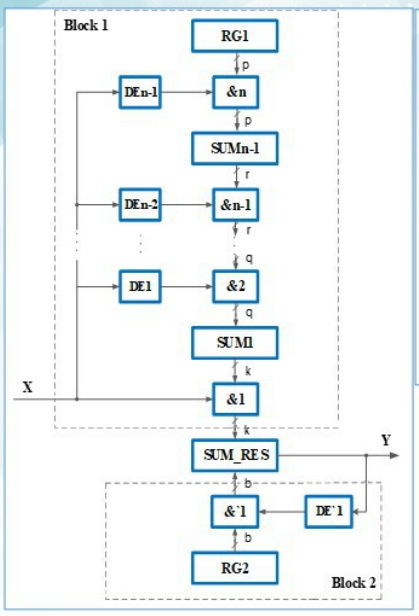
Нерівність, що реалізується в узагальненій архітектурі:

$$2\sum_{i=0}^{n} a_i x_y^i \geq m(2y_k - 1)$$

Основним обчислювальним ядром архітектури є **паралельний нагромаджуючий суматор SM_RES зі зворотним зв'язком**, що використовується в якості компонента порівняння паралельних кодів.

Формування сходинок відтворюваної функції, здійснюється на виході **суматора результату SM_RES**, в який з Block 1 надходять прямі коди чисел, а з Block 2 - додаткові коди чисел.

Деталізована архітектура дробово-раціонального обчислювача



Функція, що реалізується в дробово-раціональному обчислювачі

$$y = [\frac{\sum_{i=0}^{n} a_i x^i}{m} + 0{,}5]$$

Архітектура є синтезом двох архітектур:

Block1 представляє собою **конвеєрну архітектуру**, побудовану на основі базової структури біт-потокового **обчислювача поліноміальних функцій**, в якому реалізується алгоритм конвеєрних обчислень.

Block2 - **дільник чисел**, побудований на основі архітектури обчислювача лінійних функцій.

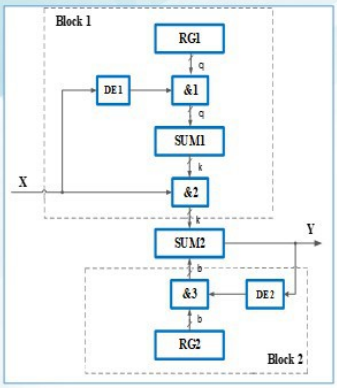
Ініціалізація компонентів в **Block1** здійснюється значеннями перших членів арифметичних рядів n-го порядку та його арифметичних рядів різниць відповідно.

Ініціалізація RG1 здійснена значенням константи арифметичного ряду різниць n-го порядку.

Суматор SUM_RES ініціалізується числом, що враховує числа ініціалізації з боку поліноміального обчислювача та з боку дільника чисел.

На підставі алгоритму конвеєрних обчислень у Block1, в суматорах SM1, …, SMm-1 формуються паралельні коди чисел арифметичного ряду m-го та арифметичних рядів різниць 1-го, …, m-1-го порядків відповідно.

Архітектура досліджуваного обчислювача дробово-раціональних функцій



Ініціалізація компонентів пристрою:
регістр **RG1:** 2a
суматор **SUM1:** $a_2 + a_1$
суматор **SUM2:** $-m \pm a_0$
регістр **RG2:** -2m

Біт-потокових пристрій відтворює функцію $y = [\frac{\sum_{i=0}^{2} a_i x^i}{m} + 0,5]$

В обчислювачі реалізується нерівність $2\sum_{i=0}^{2} a_i x_y^i \geq m(2y_k - 1)$

SUM2 використовується в якості схеми порівняння паралельних кодів приростів ґратчастої функції $2\sum_{i=0}^{2} a_i x_y^i$ з приростами ґратчастої функції $m(2y_k - 1)$ з урахуванням їх різниці Δ_{y-1}, отриманої на попередньому кроці обчислення.

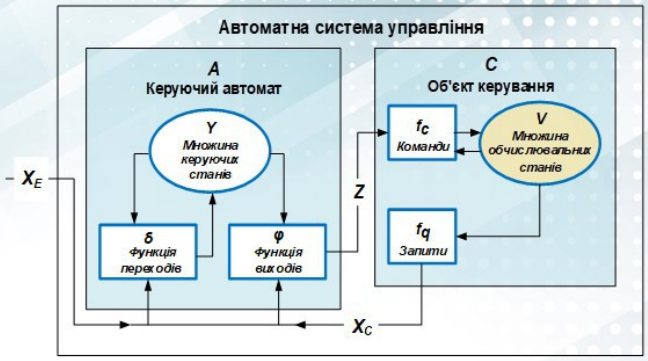
Архітектура обчислювача містить у складі **два блоки**:

I. **Квадратор** призначений для реалізації лівої частини нерівності, а саме, квадратичного полінома. Квадратор містить суматори SUM1, SUM2, групи логічних елементів &1,&2, елемент затримки DE1 і регістр RG1.

II. **Дільник чисел** призначений для реалізації правої частини нерівності, а саме, ділення полінома на число m. Дільник чисел містить суматор SUM2, групу логічних елементів &3, елемент затримки DE2, регістр RG2.

Формування сходинок відтворюваної функції у здійснюється на виході суматора SUM2, в який з Block 1 за допомогою вхідних біт x надходять прямі двійкові коди чисел зі зсувом вправо на один розряд,а з Block 2 наджодять додаткові коди чисел за допомогою вихідних біт у пристрою.

Апаратна модель досліджуваного обчислювача на основі кінцевого автомату



Біт-потоковий online-обчислювач представляє собою автоматну систему управління на основі **мікропрограмного автомату**, що є композицією керуючого і операційного автоматів.

Операційний автомат обчислювача має **обчислювальні стани**, в яких здійснюється перетворення бітових потоків у відповідності до необхідної функції в біт-потоковий і двійковий код.

Специфікація досліджуваного обчислювача

Для автоматизації досліджуваної моделі була створена експериментальна апаратна реалізація обчислювача дробово-раціональної функції з бітовим потоком даних.

$$y = [\frac{a_2 x^2 + a_1 x + a_0}{m} + 0{,}5]$$

Було обрано значення цілочисельних коефіцієнтів a_2, a_1, a_0, знаменника m функції y та довжину вхідного бітового потоку x_{max} :

$$a_2 = 1,\ a_1 = 2,\ a_0 = 5,\ m = 10,\quad x_{max} = 10.$$

Дробово-раціональна функція містить в чисельнику квадратичний поліном, який реалізований в блоку квадратора обчислювача.

Підставляючи у вираз $y = x^2 + 2x + 5$ значення x = 0,1, 2, …, 10, отримаємо послідовність значень функції y, яка є арифметичним рядом 2-го порядку

Y : 5, 8, 13, 20, 29, 40, 53, 68, 85, 104.

Арифметичні ряди різниць 1-го і 2-го порядків для цієї послідовності мають вигляд:

Δ 3, 5, 7, 9, 11, 13, 15, 17, 19;

Δ^2 2, 2, 2, 2, 2, 2, 2, 2.

Ініціалізація компонентів обчислювача:

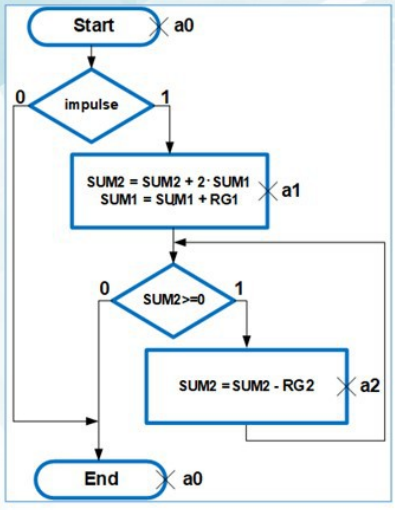
регістр RG1: $2 \cdot a_2 = 2 \cdot 1 = 2$

суматор SUM1: $a_2 + a_1 = 1 + 2 = 3$

суматор SUM2: $-m + a_0 = -10 + 5 = -5$

регістр RG2: $-2 \cdot m = -2 \cdot 10 = -20$

ГСА операційного автомата реалізації дробово-раціональної функції



На підставі математичної моделі і архітектури обчислювача розроблено **змістовну ГСА операційного автомату** реалізації дробово-раціональної функції.

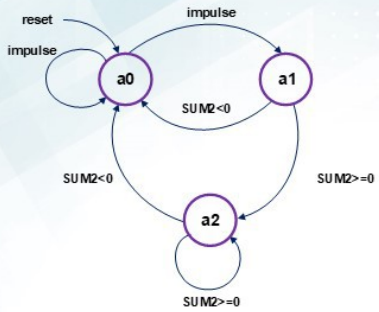
ГСА розмічені для синтезу **автомата Мура** та мають керуючі стани: a_0, a_1 і a_2 .

Фрагмент HDL-моделі операційного автомату обчислювача, в якому приведено ініціалізацію компонентів арифметичного блоку обчислювача та конвеєрні обчислення в компонентах пристрою.

```
begin
if (reset_i = '1') then
        sum  <=  CONV_STD_LOGIC_VECTOR(c  -  m,
width);
        counter  <=  CONV_STD_LOGIC_VECTOR(a+b,
width);
else
        if (falling_edge(clock_i)) then
                if (sum_plus_a_i = '1') then
                        sum <= sum + counter + counter;
                        counter <= counter + 2*a;
                else
                        if (sum_minus_b_i = '1') then
                                sum <= sum - 2*m;
                        end if;
                end if;
        end if;
end if;
```

Граф переходів керуючого автомату обчислювача

Керуючий автомат арифметичного блоку обчислювача описується графом переходів, що був отриманий в результаті синтезу граф – схеми алгоритму для автомата моделі Мура.

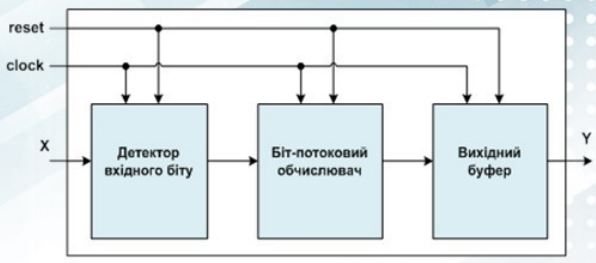


Граф переходів керуючого автомату арифметичного блоку дробово-раціонального обчислювача, що отриманий в результаті розмітки ГСА, має три стани **a0, a1, a2.**
Переходи з одного стану в інший відбуваються на основі умов переповнення (так/ні) суматорів SUM2.

Фрагмент **HDL-моделі керуючого автомату** дробово-раціонального обчислювача, в формі автоматного шаблону з використанням оператору case.

```
begin
        case (state) is
                when a_0 =>
                        if xi = '1' then
                                next state <= a_1;
                        else
                                next state <= a_0;
                        end if;
                when a_1 =>
                        if sum less zero i = '1' then
                                next state <= a_0;
                        else
                                next state <= a_2;
                        end if;
                when a_2 =>
                        if sum less zero i = '1' then
                                next state <= a_0
                        else
                                next state <= a_2;
                        end if;
                when others =>
                                next state <= a_0;
        end case;
```


Структурно-блокова схема пристрою



Пристрій містить три основних блоки: детектор вхідного біту, блок біт-потокового обчислювача і блок вихідного буфера.

1. Детектор вхідного біту призначений для детектування бітів вхідної послідовності x і на виході встановлює відповідний сигнал impulse = 1. Цей сигнал буде отримано арифметичним блоком обчислювача.

2. Блок біт-потокового обчислювача містить «Aproximator», в якому поєднано 2 модулі: квадратор та дільник чисел. Даний блок виконує операції відтворення квадратичного полінома та ділення отриманого результату на число m=10 з похибкою $|\delta_{max}| = 0{,}5$.

3. Блок вихідного буфера призначений для формування вихідної бітової послідовності y.

Обчислювальний процес в компонентах пристрою

У таблиці наведено обчислювальний процес, що відбувається **в компонентах обчислювача** дробово-раціональних функцій при подачі на вхід бітового потоку довжиною 10 імпульсів (x_{max}=10). Результати обчислення функції та поява бітів переповнення на виході SUM2 співпадають.

COMPUTING PROCESS IN DEVICE COMPONENTS

x	SUM2	Біт переповнення SUM2	SUM1
1	-5 + 6 = 1 1 – 20 = -19	1	3+2=5
2	-19 + 10 = -9		5+2 = 7
3	-9 + 14 = 5 5 – 20 = -15	1	7+2 = 9
4	-15 + 18 = 3 3 – 20 = -17	1	9+2=11
5	-17 + 22 = 5 5 – 20 = -15	1	11+2=13
6	-15 + 26 = 11 11 – 20 = -9	1	13+2 =15
7	-9 + 30 = 21 21 – 20 = 1 1 – 20 = -19	1 1	15+2=17
8	-19 + 34 = 15 15 – 20 = -5	1	17+2=19
9	-5 + 38 = 33 33 – 20 = 13 13 – 20 = -7	1 1	19+2=21
10	-7 +42 = 35 35 – 20 = 15 15 – 20 = -5	1 1	21+2=23

RESULTS OF CALCULATING THE Fractional-Rational FUNCTION

$x = 1 \quad y = [\frac{1^2 + 2 \cdot 1 + 5}{10} + 0,5] = [1,3] = 1$

$x = 2 \quad y = [\frac{2^2 + 2 \cdot 2 + 5}{10} + 0,5] = [1,8] = 1$

$x = 3 \quad y = [\frac{3^2 + 2 \cdot 3 + 5}{10} + 0,5] = [2,5] = 2$

$x = 4 \quad y = [\frac{4^2 + 2 \cdot 4 + 5}{10} + 0,5] = [3,4] = 3$

$x = 5 \quad y = [\frac{5^2 + 2 \cdot 5 + 5}{10} + 0,5] = [4,5] = 4$

$x = 6 \quad y = [\frac{6^2 + 2 \cdot 6 + 5}{10} + 0,5] = [5,8] = 5$

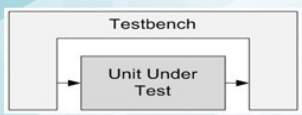
$x = 7 \quad y = [\frac{7^2 + 2 \cdot 7 + 5}{10} + 0,5] = [7,3] = 7$

$x = 8 \quad y = [\frac{8^2 + 2 \cdot 8 + 5}{10} + 0,5] = [9] = 9$

$x = 9 \quad y = [\frac{9^2 + 2 \cdot 9 + 5}{10} + 0,5] = [10,9] = 10$

$x = 10 \quad y = [\frac{10^2 + 2 \cdot 10 + 5}{10} + 0,5] = [12,5] = 12$

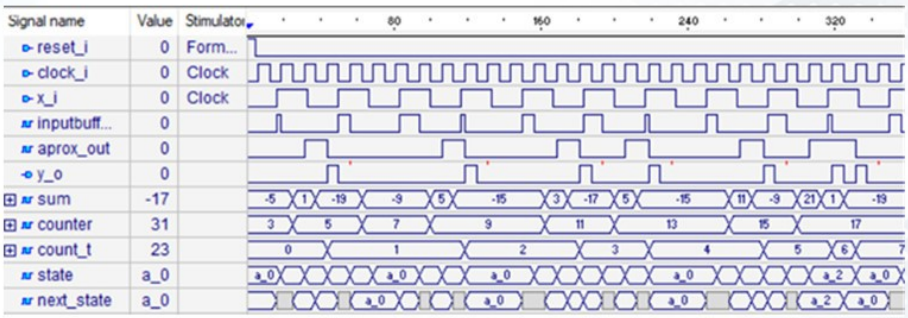
Верифікація та тестування роботи пристрою



Верифікація поведінкової моделі досліджуваного обчислювача виконувалась з використанням САПР Active-HDL.

На часовій діаграмі представлені результати моделювання поведінкової моделі біт-потокового online-обчислювача дробово-раціональної функції.

Значення в регістрах компонентів збігаються з розрахунковими даними обчислювального процесу і поява вихідних біт пристрою Y відповідає номерам вхідних біт Xy.



Висновки

В кваліфікаційній роботі розроблено і досліджено біт-потоковий апаратний обчислювач дробово-раціональних функцій, аргумент якого представлений бітовим потоком даних.

- Проаналізовано особливості функціонального перетворення бітових потоків даних в обчислювачах математичних функцій.
- Розроблено математичну модель біт-потокового обчислювача дробово-раціональних функцій. Для отримання математичної моделі був використаний спосіб формування приростів висхідних безперервних функцій на основі різницевих нерівностей.
- Використано переваги принципу побудови біт-потокової конвеєрної архітектури обчислювача поліноміальних функцій, яка реалізує функціональне перетворення розгортуючого типу на основі обчислення приростів відтворюваної функції.
- Розроблено архітектуру досліджуваного обчислювача, яка є синтезом двох блоків: блоку реалізації поліноміальної функції і блоку дільника чисел, що поєднані між собою основним обчислювальним вузлом – суматором зі зворотним зв'язком.

Kharkov National University of Radio Electronics 19

Висновки

- В результаті розробки та аналізу математичної і архітектурної моделей обчислювача було здійснено опис проекту для введення в САПР.
- Апаратна модель обчислювача сформована на основі кінцевого автомата моделі Мура. Розроблена змістовна граф-схема алгоритму операційного автомату реалізації заданої функції і на підставі ГСА, отриманий граф переходів керуючого автомату арифметичного блоку обчислювача.
- За графами переходів з використанням стандартних шаблонів коду розроблено модель пристрою на мові опису апаратури VHDL.
- Працездатність апаратної моделі обчислювача підтверджено перевіркою результатів за допомогою верифікації поведінкової моделі з використанням САПР Active-HDL.
- Модель синтезована в програмовану логічну інтегральну схему Xilinx Spartan 3E.

Kharkov National University of Radio Electronics 20

ДОДАТОК Б Лістинг коду програм

Лістинг Б.1 – Програма верхнього рівня Aprox_top.vhdl

```
Aprox_top.vhdl

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;

entity aprox_toplevel is
	generic(
		width: natural := 8;
		a: integer := 1;
		b: integer := 2;
		c: integer := 5;
		m: integer := 10);
	port(
		reset_i: in std_logic;
		clock_i: in std_logic;
		x_i: in std_logic;
		y_o: out std_logic;
		count: out std_logic_vector(width-1 downto 0));
end aprox_toplevel;

architecture struct of aprox_toplevel is
	-- Component declaration of the "aproximator(struct)" unit defined in
	-- file: "./src/aproximator.vhd"
	component aproximator
		generic(
			width : NATURAL := 8;
			a: integer := 1;
			b: integer := 2;
			c: integer := 5;
			m: integer := 10);
		port(
			x_i : in std_logic;
			ready_o : out std_logic;
			clock_i : in std_logic;
			reset_i : in std_logic;
			y_o : out std_logic;
			sum_o: out std_logic_vector(width-1 downto 0));
	end component;

	-- Component declaration of the "inputbuffer(beh)" unit defined in
	-- file: "./src/inputbuffer.vhd"
	component aprox_inputbuffer
		generic(
			width : NATURAL := 8);
		port(
```

```
                x_i : in std_logic;
                clock_i : in std_logic;
                reset_i : in std_logic;
                ready_i : in std_logic;
                y_o : out std_logic);
        end component;

        -- Component declaration of the "outputbuffer(beh)" unit defined in
        -- file: "./src/outputbuffer.vhd"
        component aprox_outputbuffer
                port(
                        clock_i : in std_logic;
                        reset_i : in std_logic;
                        en_i : in std_logic;
                        y_o : out std_logic);
        end component;

        signal inputbuffer_out, aprox_out, ready, y_t: std_logic;
        signal count_t: std_logic_vector(width-1 downto 0);
        signal sum_t: std_logic_vector(width-1 downto 0);

begin
        y_o <= y_t;

        InputBuffer_1 : aprox_inputbuffer
        generic map(
                width => width
                )
        port map(
                x_i => x_i,
                clock_i => clock_i,
                reset_i => reset_i,
                ready_i => ready,
                y_o => inputbuffer_out
                );

        Aproximator_1 : aproximator
        generic map(
                width => width,
                a => a,
                b => b,
                c => c,
                m => m
                )
        port map(
                x_i => inputbuffer_out,
                ready_o => ready,
                clock_i => clock_i,
                reset_i => reset_i,
                y_o => aprox_out,
                sum_o => sum_t
                );

        OutputBuffer_1 : aprox_outputbuffer
        port map(
                clock_i => clock_i,
                reset_i => reset_i,
```

```
                en_i => aprox_out,
                y_o => y_t
                );
        process(clock_i, reset_i)
        begin
                if reset_i = '1' then
                        count_t <= (others => '0');
                else
                        if rising_edge(clock_i) then
                                if aprox_out = '1' then
                                        count_t <= count_t + 1;
                                end if;
                        end if;
                end if;
        end process;
        count <= count_t;
end struct;
```

Лістинг Б.2 – Програма операційного автомата Aprox_oa.vhdl

Aprox_oa.vhdl

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;

entity aproximator_oa is
        generic(
                width: natural := 8;
                a: integer := 1;
                b: integer := 2;
                c: integer := 5;
                m: integer := 10
                );
        port(
                clock_i: in std_logic;
                reset_i: in std_logic;
                sum_plus_a_i: in std_logic;
                sum_minus_b_i: in std_logic;
                sum_less_zero_o: out std_logic;
                sum_o: out std_logic_vector(width-1 downto 0));
end aproximator_oa;

architecture beh of aproximator_oa is
        signal sum: std_logic_vector(width-1 downto 0);
        signal counter: std_logic_vector(width-1 downto 0);
begin
        sum_o <= sum;

        sum_less_zero_o <= '1' when (sum < 0) else '0';

        process (clock_i, reset_i)
```

```
	begin
		if (reset_i = '1') then
			sum <= CONV_STD_LOGIC_VECTOR(c - m, width);
	counter <= CONV_STD_LOGIC_VECTOR(a+b, width);
		else
			if (falling_edge(clock_i)) then
				if (sum_plus_a_i = '1') then
					sum <= sum + counter + counter;
					counter <= counter + 2*a;
	else
					if (sum_minus_b_i = '1') then
						sum <= sum - 2*m;
					end if;
				end if;
			end if;
		end if;
	end process;
end beh;
```

Лістинг Б.3 – Програма керуючого автомата Aprox_ua.vhdl

Aprox_ua.vhdl

```
library ieee;
use ieee.std_logic_1164.all;

entity aproximator_ua is
	port(
		clock_i: in std_logic;
		reset_i: in std_logic;
		x_i: in std_logic;
		y_o: out std_logic;
		ready_o: out std_logic;
		sum_less_zero_i: in std_logic;
		sum_plus_a_o: out std_logic;
		sum_minus_b_o: out std_logic;
		states: out std_logic_vector(1 downto 0));
end aproximator_ua;

architecture beh of aproximator_ua is

	type state_type is (a_0, a_1, a_2);
	signal state, next_state: state_type;
	signal control: std_logic_vector(3 downto 0);
begin
	process(clock_i, reset_i)
	begin
		if (reset_i = '1') then
			state <= a_0;
		else
			if (rising_edge(clock_i)) then
				state <= next_state;
			end if;
```

```
		end if;
	end process;

	process(state, x_i, sum_less_zero_i)
	begin
		case (state) is
			when a_0 =>
				if x_i = '1' then
					next_state <= a_1;
				else
					next_state <= a_0;
			end if;
			when a_1 =>
				if sum_less_zero_i = '1' then
					next_state <= a_0;
				else
					next_state <= a_2;
			end if;
			when a_2 =>
				if sum_less_zero_i = '1' then
					next_state <= a_0;
				else
					next_state <= a_2;
			end if;
			when others =>
			next_state <= a_0;
		end case;
	end process;

	(ready_o, sum_plus_a_o, sum_minus_b_o, y_o) <= control;

	with state select
	control <= "1000" when a_0,
	"0100" when a_1,
	"0011" when a_2,
	"0000" when others;

	with state select
	states <= "11" when a_0,
	"10" when a_1,
	"01" when a_2,
	"00" when others;
end beh;
```

Лістинг Б.3 – Програма модуля testbench

```
testbench


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
```

```
entity testbench is
	generic(
		width: natural := 8);
end entity;

architecture testbench_1 of testbench is
component aprox_toplevel
port(reset_i: in std_logic;
		clock_i: in std_logic;
		x_i: in std_logic;
		y_o: out std_logic;
		count: out std_logic_vector(width-1 downto 0));
end component;


signal reset_i, clock_i, x_i, y_o : std_logic;
signal count: std_logic_vector(width-1 downto 0);

begin
UUT : aprox_toplevel
port map (reset_i=>reset_i, clock_i=>clock_i, x_i=>x_i, y_o=>y_o, count=>count);


process
begin
	clock_i<='0', '1' after 5 ns;
	wait for 10 ns;
end process;

process
begin
	reset_i<='1';
	wait for 10ns;
	reset_i<='0';
	wait for 1000ns;
end process;

process
begin
	x_i<='0', '1' after 20 ns; wait for 40 ns;
	end process;

end testbench_1;
```

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮІК 46741Х.009 ПЗ	Пояснювальна записка	77 с.
		Графічні документи	
2		Слайд-презентація	20 слайдів
		Носії інформації	
3		Компакт-диск (CD)	1 шт.

Змін.	Арк	Номер докум.	Підп.	Дата	46741Х.009 ВД			
Розроб.		Ющенко С. В.			Моделі та методи проектування апаратного біт-потокового обчислювача дробово-раціональних функцій Відомість атестаційної роботи	Літ	Аркуш	Аркушів
Перевір.		Ларченко Л.В				У		
Н.контр.		Шкіль О.С.				ХНУРЕ Кафедра АПОТ		
Затв.		Чумаченко С.В.						