ДОДАТОК А

Графічний матеріал атестаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра АПОТ
Атестаційна робота магістра

Моделі та методи проектування біт-потокового обчислювача степеневих функцій на базі FPGA

Магістранта групи СКСм-19-1
Шапа Людмили
Сергіївни

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

Харьковский национальный университет радиоэлектроники,
кафедра АПВТ, тел. 7021 326, e-mail: ri@kture.kharkov.ua

Мета та завдання дослідження

Мета атестаційної роботи — дослідження та розробка моделей і методів автоматизованого проектування біт-потокового обчислювача степеневих функцій на основі ПЛІС.

Об'єктом дослідження є спеціалізовані обчислювачі степеневих функцій перетворення бітових потоків даних з зовнішніх сенсорів фізичних величин.

Предмет дослідження – математична та структурна моделі біт-потокового обчислювача степеневих функцій, структурні моделі обчислювачів на основі конвеєрних архітектур, апаратні моделі обчислювачів на основі цифрових автоматів.

Завдання дослідження:

- аналіз особливостей функціонального перетворення бітових потоків в апаратних обчислювачах математичних функцій;
- аналіз способу обчислення степеневих функцій, аргумент яких представлений бітовим потоком, на основі методу ступінчастої апроксимації відтворення неперервних висхідних функцій;
- розробка математичної моделі біт-потокового обчислювача степеневої функції;
- аналіз способу побудови конвеєрних архітектур апаратних обчислювачів поліноміальних функцій з біт-потоковою формою даних;
- розробка структурно-функціональної моделі обчислювача заданої функції;
- розробка апаратної реалізації пристрою на основі кінцевого автомата Мура;
- розробка HDL-моделі обчислювача;
- верифікація, тестування та імплементація отриманої апаратної моделі обчислювача в платформу ПЛІС.

Kharkov National University of Radio Electronics,
Design Automation Department, phone 7021 326

Галузі застосування біт-потокових обчислювачів математичних функцій:

- в системах, орієнтованих на технології інтелектуальних сенсорів в якості функціональних перетворювачів бітових потоків даних, отриманих з вимірювальних сенсорів фізичних величин;
- в сучасних системах управління та контролю в якості спеціальної апаратури їх спряження з датчиками і виконавчими органами об'єкту керування; при відтворенні траєкторій рухомих об'єктів в двомірному і тривимірному просторі;
- при відтворенні траєкторій рухомих об'єктів в двомірному і тривимірному просторі;
- при проведенні математичної обробки первинної вимірювальної інформації в інформаційно-вимірювальних системах;
- в якості обчислювальних вузлів в вимірювальних системах і приборах при здійсненні непрямих вимірювань;
- при виконанні різних функціональних перетворень частоти імпульсних послідовностей.

Особливості функціонального перетворення бітових потоків даних

При неперервному аналізі інформаційних процесів, що відбуваються в природі та технічних системах необхідне здійснення

-неперервного прийому та обробки потоку даних у міру їх надходження;

-неперервне формування результату в процесі обробки.

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 імпульсного потоку.

Реалізація потокового методу обчислень полягає в розгортці кодової інформації в часі з одночасним паралельно-послідовним виконанням перетворень над бітовими потоками час-імпульсних сигналів відповідно до необхідної функції.

Вхідний і вихідний інформаційні сигнали біт-потокових обчислювачів являють собою два імпульсних потоки, періодичність проходження імпульсів першого з яких визначається способом квантування відтворюваної функції по аргументу, а другого - алгоритмом функціонування пристрою.

Досліджувана функція

Апаратний обчислювач степеневих функцій обчислює функцію

$$y = \left[x^{\frac{2}{3}} + \lfloor \delta_{max} \rfloor \right]$$

де x – аргумент функції, що представляє собою бітовий потік даних;

$|\delta_{max}|$ – граничне значення абсолютної похибки при піднесенні аргументу до дробового степеню. Задана абсолютна похибка обчислень $|\delta_{max}| = 0,5$

Отже, обчислювач реалізує функцію

$$y = \left[x^{\frac{2}{3}} + 0,5 \right]$$

Біт-потоковий обчислювач степеневих функцій має містити 2 блоки, в яких операції піднесення до степеню і добування кореня суміщені і виконуються одночасно.

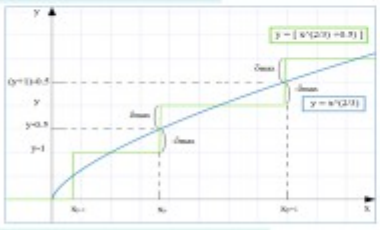
Метод ступінчастої апроксимації неперервних висхідних функцій

Безперервна функція $y^* = f(x^*)$ може бути відтворена на виході пристрою із заданою абсолютною похибкою відтворення $|\delta_{max}|$ ступінчастою функцією $y = [f(x) + |\delta_{max}|]$

де x=1,2,3,..., квадратні дужки позначають цілу частину числа при обмеженнях $y \le x$, функція $y^* = f(x^*)$ має зворотну $x^* = \psi(y^*)$.

З урахуванням обмежень для будь-якого рівня y–|δmax|, де y = 1, 2, 3, ..., можна вказати пару цілочисельних значень аргументу $x_y - 1$ та x_y, для яких має місце система нерівностей

$$\begin{cases} f(x_y - 1) < y - |\delta_{max}|, \\ f(x_y) \ge y - |\delta_{max}|. \end{cases}$$



Формула загального члена числової послідовності x1, x2, x3,.., відповідна вузлам апроксимації вихідної функції y

$$\Psi(y - |\delta_{max}|) \le x_y < \Psi(y - |\delta_{max}|) + 1$$

В окремому випадку при $|\delta_{max}| = 0,5$, буде забезпечена мінімальна похибка обчислення функції в цілочисельних точках аргументу, формула прийме вигляд

$$\Psi(y - 0,5) \le x_y < \Psi(y - 0,5) + 1$$

Математична модель апаратного обчислювача степеневих функцій

Для степеневої функції $y = \left[x^{\frac{2}{3}} + 0{,}5 \right]$ була визначена зворотна функція і формула загального члена числової послідовності, що відповідає вузлам апроксимації заданої функції має вигляд

$$(2y-1)^3 \le x_y^2 2^3 < (2y-1)^3 + 1$$

Було отримано математичну модель обчислювача степеневих функції, що представляє собою систему нерівностей в різницях.

Визначення x_y було зведено до обчислення приростів ґратчастої функції 2^3x^2 на кожному інтервалі $(x_{y-1};\ x_y]$ і їх порівнянні з приростами ґратчастої функції $(2y-1)^3$ з урахуванням їх різниці обчислення Δ_{y-1}, отриманої на попередньому кроці обчислень.

$$2^3 x_1^2 \ge (2y_1 - 1)^3$$
$$2^3(x_2^2 - x_1^2) + \Delta_1 \ge (2y_2 - 1)^3 - (2y_1 - 1)^3$$
$$2^3(x_3^2 - x_2^2) + \Delta_2 \ge (2y_3 - 1)^3 - (2y_2 - 1)^3$$
. . . . . .
$$2^3(x_y^2 - x_{y-1}^2) + \Delta_{y-1} \ge (2y_1 - 1)^3 - (2y_{1-1} - 1)^3,$$

де $\Delta_{y-1} = 2^3(x_y^2 - x_{y-1}^2) + \Delta_{y-2} - (2y_1 - 1)^3 - (2y_{1-1} - 1)^3$.

Алгоритм обчислення поліноміальних функцій

При обчисленні дробово-раціональної функції вигляду $y = \sum_{i=0}^{n} a_i x^i$

Забезпечено обчислення полінома n-го степеню. Характерна ознака поліномів з цілочисельними коефіцієнтами a_i це відповідність послідовності його цілочисельних значень функції $y_0, y_1, y_2, \dots, y_i$ значенням x_y=0, 1, 2, ... і. Отримано числову послідовність y_i. Значення функції y_i визначаються за виразом $y_i = f(i+1) - f(i)$

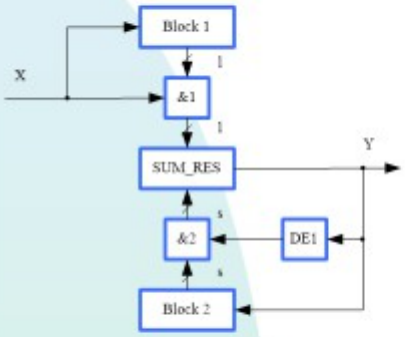
Математична модель обчислення поліноміальної функції представлена в різницевих рівняннях

$$\Delta_i = y_{i+1} - y_i$$
$$\Delta_i^2 = \Delta_{i+1} - \Delta_i$$
. . . . . .
$$\Delta_i^n = \Delta_{i+1}^{n-1} - \Delta_i^{n-1}$$

де $i = 0, 1, 2, 3, \dots$

Ця методика може бути використана при проектуванні конвеєрних архітектур біт-потокових спеціалізованих обчислювачів.

Узагальнена структурно-функціональна модель обчислювача степеневих функцій



Нерівність, що реалізується в узагальненій структурі має вигляд

$$2^n x_y^m \geq (2y-1)^n$$

де x, y = 1, 2, 3, …

Формування сходинок функції, що відтворюється, здійснюється на виході суматора результату SUM_RES, в який з Block 1 надходять прямі коди чисел, а з Block 2 - додаткові коди чисел.

Block 1 – модуль для відтворення функції $2^n x^m$,

Block 2 – модуль для відтворення функції $(2y-1)^n$,

&1, &2 – група елементів &,

DE1 – елемент затримки

Kharkov National University of Radio Electronics,
Design Automation Department, phone 7021 326

Структурно-функціональна модель поліноміального обчислювача

Модель поліноміального обчислювача являє собою структуру, в якій реалізовані конвеєрні обчислення.

Для функції $y = x^n$ при $x = 1, 2, 3, \ldots$ значення функції є арифметичним рядом n порядку, та має вигляд $1^n, 2^n, 3^n, 4^n, 5^n$

Арифметичні ряди різниць 1-го, 2-го та n-го порядку мають вигляд

$$\Delta \quad 2^n - 1^n,\ 3^n - 2^n,\ 4^n - 3^n, \ldots$$

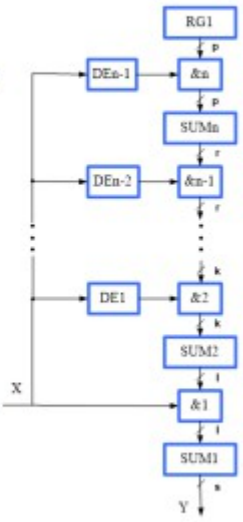
$$\Delta^2 \quad (3^n - 2^n) - (2^n - 1^n),\ (4^n - 3^n) - (3^n - 2^n), \ldots$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\Delta^n \quad n!,\ n!,\ n!,\ n!, \ldots,$$

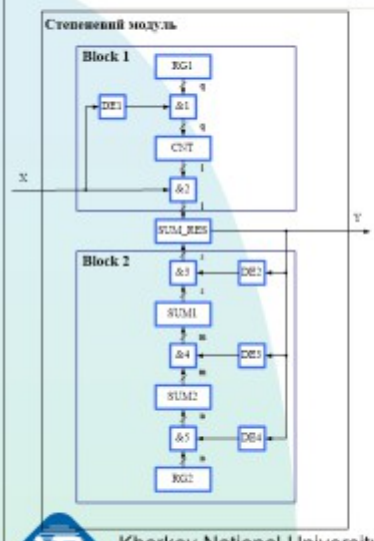
Ініціалізація компонентів модуля:

- SUM1 ініціалізується числом 0.
- Компоненти SUM2,….SUMn модуля ініціалізуються першими членами рядів різниць 1,2,…n-1 порядку.
- Регістр RG1 ініціалізується значенням константи n!
- В суматорі SUMn будуть формуватися члени арифметичного ряду різниць (n–1) – го порядку. В SUM2 – члени арифметичного ряду різниць 1-го порядку, а в SUM1 – члени арифметичного ряду n порядку: $1n$, $2n$, $3n$, $4n$ …



Kharkov National University of Radio Electronics,
Design Automation Department, phone 7021 326

Структурно-функціональна модель досліджуваного обчислювача



В степеневому модулі реалізується нерівність

$$2^3 x_y^{\ 2} \geq (2y-1)^3$$

Основним обчислювальним вузлом є SUM_RES, який використовується в якості схеми порівняння паралельних кодів приростів ґратчастої функції $2^n x^m$ з приростами ґратчастої функції $(2y-1)^k$ з урахуванням їх різниці Δ_{y-1} отриманої на попередньому кроці обчислень.

Ініціалізація компонентів степеневого модуля:

- суматор SUM_RES ініціалізується числом 2i - 1 (i - розрядність суматора);
- лічильник CNT ініціалізується числом 8;
- суматор SUM1 ініціалізується числом 26;
- суматор SUM2 - ініціалізується числом 72;
- регістр RG1 ініціалізується числом 16;
- регістр RG2 ініціалізується константою, утвореною в ряду різниць 3-го порядку функції, тобто числом 48.

Специфікація досліджуваного обчислювача

Для автоматизації досліджуваної моделі була створена експериментальна апаратна реалізація спеціалізованого обчислювача степеневої функції з імпульсними потоками даних

Оберемо значення цілочисельних коефіцієнтів n, m підкореневого виразу функції y та x_{max} вхідного імпульсного потоку: m=2, n =3, x_{max} = 10.

$$y = \left[\sqrt[3]{x^2} + 0{,}5 \right]$$

Для функції $2^3 x_y^2$ при значеннях x = 1, 2, 3, ..., арифметичний ряд 2-го порядку і арифметичні ряди різниць 1-го 2-го порядку відповідно матимуть вигляд :

8, 32 , 72 , 128 , 200 , 288 , 392 ...

Δ 24 , 40 , 56 , 72, 88

Δ^2 16 , 16 , 16 , 16 ...

А для функції $(2y-1)^3$ при значеннях y = 1, 2, 3, ..., арифметичний ряд 3-го порядку і арифметичні ряди різниць1-го, 2-го і 3-го порядку відповідно матимуть вигляд :

1 , 27 , 125 , 343 , 729 , 1331 , 2197

Δ 26 , 98 , 218 , 386 , 602 , 866

Δ^2 72 , 120 , 168 , 216 , 264

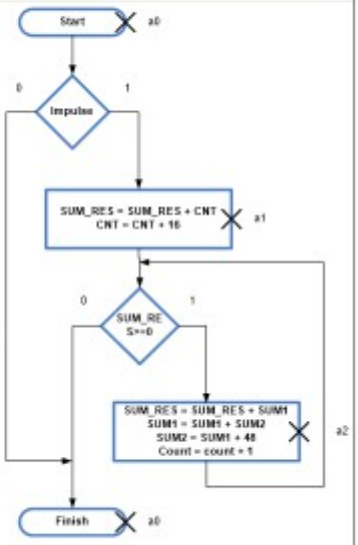
Δ^3 48 , 48 , 48 , 48 , 48

Граф-схема алгоритму роботи обчислювача

Пристрій, реалізований за типом автомата Мура, представлений керуючим і операційним автоматами

Пристрій працює за наступним алгоритмом:

1. при скиданні пристрою (сигнал reset = 1) регістри пристрою встановлюються в такі значення: CNT = 8, SUM_RES = -1, SUM1 = 26, SUM2 = 72, RG1 = 16, RG2 = 48, count = 0.
2. при надходженні чергового імпульсу значення суматора SUM_RES збільшується на значення суматора CNT, а значення суматора CNT збільшується на 16;
3. якщо значення регістра SUM_RES невід'ємне, то на виході пристрою генерується вихідний імпульс, значення лічильника імпульсів count збільшується на одиницю, з значення регістра SUM_RES віднімається значення регістра SUM1, до значення регістра SUM1 додається значення регістра SUM2, до значення регістра SUM2 додається 48;
4. п. 3 повторюється до тих пір, поки значення регістра SUM_RES невід'ємне.



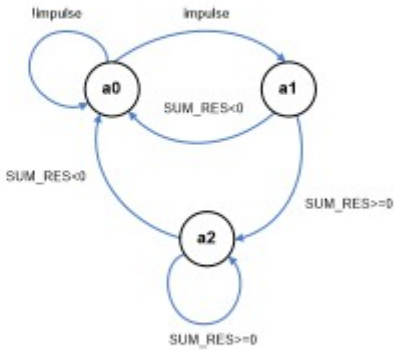
Граф переходів автомата Мура арифметичного блоку

Управляючий автомат описано отриманим в результаті аналізу ГСА для автомата Мура графом переходів. Він може приймати 3 стани: a0, a1, a2, a3

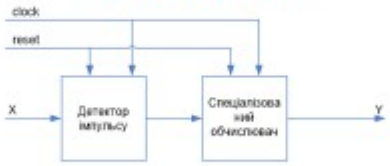
a0: По сигналу reset = 1 автомат переходить в початковий стан a0. Автомат знаходиться в стані a0 допоки немає сигналу impulse, який надходить з вхідного буфера.

a1: В стані a1 за допомогою автомата значення суматора SUM_RES збільшується на значення суматора CNT, а значення суматора CNT збільшується на 16. Автомат входить в стан a2, коли на виході апроксиматора з'являється сигнал наявності імпульса на виході результуючого суматора.

a2: В стані a2 автомат збільшує на одиницю значення Count, значення суматора SUM1 віднімається від значення суматора SUM_RES, до значення суматора SUM1 додається значення регістра SUM2, і з RG2 число 48 додається до значення суматора SUM2. В цьому стані також виводиться сигнал для генерації імпульсу на виході пристрою.



Структурно-блокова схема досліджуваного обчислювача



У пристрої використовується два основних блоки: детектор вхідного імпульсу і блок спеціалізованого обчислювача.

Блок Детектор імпульсу призначений для детектування число-імпульсної послідовності на вході пристрою x. Імпульс детектується за двома сусідніми тактами сигналу clock. Якщо на черговому такті значення сигналу x = 0, а на наступному такті значення x = 1, то вхідний буфер детектує імпульс, і на виході встановлює відповідний сигнал impulse = 1.

Цей сигнал буде отримано пристроєм обчислювача.

Блок Спеціалізований обчислювач виконує операцію піднесення аргументу x до дробового степеню з заданою похибкою, результат округляється до найближчих цілих чисел.

Обчислювальний процес в компонентах досліджуваного пристрою

В таблиці наведено деталі обчислювального процесу, що відбуваються в компонентах обчислювача при подачі на вхід 10 імпульсів.

При x = 1 $y=[x^{\frac{2}{3}}+0.5]=1;$

При x = 2 $y=[x^{\frac{2}{3}}+0.5]=[2^{\frac{2}{3}}+0.5]=2;$

При x = 3 $y=[x^{\frac{2}{3}}+0.5]=[3^{\frac{2}{3}}+0.5]=2;$

При x = 4 $y=[x^{\frac{2}{3}}+0.5]=[4^{\frac{2}{3}}+0.5]=3;$

При x = 5 $y=[x^{\frac{2}{3}}+0.5]=[5^{\frac{2}{3}}+0.5]=3;$

При x = 6 $y=[x^{\frac{2}{3}}+0.5]=[6^{\frac{2}{3}}+0.5]=3;$

При x = 7 $y=[x^{\frac{2}{3}}+0.5]=[7^{\frac{2}{3}}+0.5]=4;$

При x = 8 $y=[x^{\frac{2}{3}}+0.5]=[8^{\frac{2}{3}}+0.5]=4;$

При x = 9 $y=[x^{\frac{2}{3}}+0.5]=[9^{\frac{2}{3}}+0.5]=4;$

При x = 10 $y=[x^{\frac{2}{3}}+0.5]=[10^{\frac{2}{3}}+0.5]=5;$

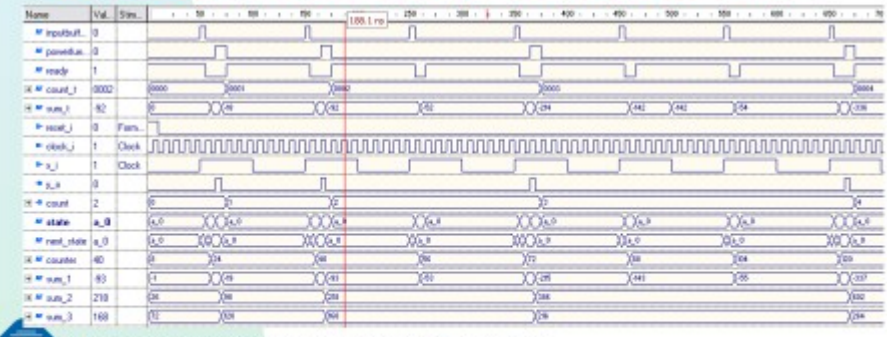
X	SUM_RES	I	CNT	SUM1	SUM2	CountRes
1	-1+8 = 7	1	8+16 = 24	26+72 = 98	72+48 = 120	1
2	7-26= - 19 -19+24=5	1	24+16 = 40	98+120 = 218	120+48 = 168	2
	5-98 = -93					
3	-93+40 = -53		40+16=56			
4	-53+56=3	1	56+16 = 72	218+168=386	168+48=216	3
	3-218 = -215					
5	-215+72 = -143		72+16 = 88			
6	-143+88 = -55		88+16 = 104			
7	-55+104 = 49	1	104+16 = 120	386+216 = 602	216+48 = 264	4
	49-386 = -337					
8	-337+120 = -217		120+16 = 136			
9	-217+136 = -81		136+16 = 152			
10	-81 +152 = 71	1	152+16 = 168	602+264 = 866	264+48 = 312	5
	71-602 = -531					

Верифікація та тестування роботи пристрою

Для верифікації даної апаратної реалізації, мовою VHDL було розроблено тестове оточення (test bench).

Взаємодія і ієрархічне відношення тест-бенча і модуля, що тестується показано на рисунку праворуч

Верифікацію режимів роботи пристрою представлено на часовій діаграмі



Висновки

В атестаційній роботі розроблено і досліджено біт-потоковий апаратний обчислювач степеневих функцій, аргумент якого представлений бітовим потоком даних.

- Було розроблено математичну модель біт-потокового пристрою обчислення степеневих функцій, що являє собою синтез операції піднесення до степеню і взяття кореня в одному пристрої
- Для отримання математичної моделі пристрою був застосований метод ступінчастої апроксимації безперервних степеневих функцій
- Використано переваги принципу побудови біт-потокової конвеєрної архітектури обчислювача поліноміальних функцій, яка реалізує функціональне перетворення розгортуючого типу на основі обчислення приростів відтворюваної функції.

Висновки

- Структурно-функціональну модель біт-потокового обчислювача степеневих функцій реалізовано синтезом двох послідовно з'єднаних блоків: блоком для реалізації піднесення до степеню і блоком взяття кореню, що поєднані між собою основним обчислювальним вузлом.
- В результаті розробки та аналізу структурно-функціональної моделі біт-потокового обчислювача, було здійснено опис проекту для введення в САПР.
- Апаратна модель спроектованого обчислювача сформована на основі цифрового кінцевого автомата за типом Мура.
- Була розроблена змістовна граф-схема алгоритму його роботи, і, на підставі ГСА, отриманий граф переходів.
- За графами переходів з використанням стандартних шаблонів кода розроблено модель пристрою на мові опису апаратури VHDL.
- Для верифікації апаратної реалізації, було розроблено тестове оточення (test bench).
- Модель синтезована в програмовану логічну інтегральну схему кристал сімейства Xilinx Spartan 3E.

Kharkov National University of Radio Electronics,
Design Automation Department, phone 7021 326

ДОДАТОК Б

Лістинг коду програм

Powerfunc.vhdl

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

entity powerfunc is
	generic(
		width1: natural := 16;
		width2: natural := 24);
	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(width2-1 downto 0));
end entity powerfunc;

architecture struct of powerfunc is

	-- Component declaration of the "aproximator_ua(beh)" unit defined in
	-- file: "./src/aproximator_ua.vhd"
	component powerfunc_ua
		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 component;
```

```
	-- Component declaration of the "aproximator_oa(beh)" unit defined in
	-- file: "./src/aproximator_oa.vhd"
	component powerfunc_oa
		generic(
			width1: natural := 16;
			width2: natural := 24);
		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(width2-1 downto 0));
	end component;

	signal sum_less_zero, sum_plus_a, sum_minus_b: std_logic;
	signal t: std_logic_vector(1 downto 0);

begin

	CONTROL_UNIT : powerfunc_ua
	port map(
		clock_i => clock_i,
		reset_i => reset_i,
		x_i => x_i,
		y_o => y_o,
		ready_o => ready_o,
		sum_less_zero_i => sum_less_zero,
		sum_plus_a_o => sum_plus_a,
		sum_minus_b_o => sum_minus_b,
		states => t
		);
	DATA_FLOW : powerfunc_oa
	generic map(
		width1 => width1,
		width2 => width2)
	port map(
		clock_i => clock_i,
		reset_i => reset_i,
		sum_plus_a_i => sum_plus_a,
		sum_minus_b_i => sum_minus_b,
		sum_less_zero_o => sum_less_zero,
```

```
            sum_o => sum_o
            );
end struct;
```

Powerfunc. oa.vhdl

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

entity powerfunc_oa is
      generic(
      width1: natural := 16;
      width2: natural := 24);
      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(width2-1 downto 0));
end powerfunc_oa;

architecture beh of powerfunc_oa is

      signal counter, sum_3, count : std_logic_vector(width1-1 downto 0);
      signal sum_1, sum_2 : std_logic_vector(width2-1 downto 0);

begin

      sum_o <= sum_1;
      sum_less_zero_o <= '1' when (sum_1 < 0) else '0';

      process (clock_i, reset_i)
      begin
            if (reset_i = '1') then
                  counter <= CONV_STD_LOGIC_VECTOR(8, width1);
                  sum_1 <= CONV_STD_LOGIC_VECTOR(-1, width2);
                  sum_2 <= CONV_STD_LOGIC_VECTOR(26, width2);
                  sum_3 <= CONV_STD_LOGIC_VECTOR(72, width1);
            else
                  if (falling_edge(clock_i)) then
```

```
                        if (sum_plus_a_i = '1') then
                              sum_1 <= sum_1 + counter;
                              counter <= counter + 16;        -- add our
constant
                        else
                              if (sum_minus_b_i = '1') then
                                    count <= count + 1;
                                    sum_1 <= sum_1 - sum_2;
                                    sum_2 <= sum_2 + sum_3;
                                    sum_3 <= sum_3 + 48;
                              end if;
                        end if;
                  end if;
            end if;
      end process;
end beh;
```

Powerfunc. ua.vhdl

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

entity powerfunc_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 powerfunc_ua;

architecture beh of powerfunc_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;
```

Wrapper. vhdl

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

entity wrapper is
generic(
      width1 : natural := 16;
      width2 : natural := 24);
      port(
            rst, St, clk: in STD_LOGIC;
            LED: out STD_LOGIC_VECTOR(width1-1 downto 0)
            );
end wrapper;

architecture behav of wrapper is
component powerfunc_toplevel is
generic(
      width1: natural := 16;
      width2: natural := 24);
      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(width1-1 downto 0));
end component;

constant N : INTEGER := 17;
signal COUNT_INT : STD_LOGIC_VECTOR(N-1 downto 0);
signal s1, s2, s3, St_in, clk_out: std_logic;
signal y_out : std_logic;
```

```
begin

KVAD1: powerfunc_toplevel
generic map(
      width1 => width1,
      width2 => width2
)
port map(
      reset_i => rst,
      clock_i => clk,
      x_i => St_in,
      y_o => y_out,
      count => LED);
process(rst, clk)
begin
if rst = '1' then
      COUNT_INT <= (others => '0');
      elsif clk'event and clk = '1' then
            COUNT_INT <= COUNT_INT + 1;
end if;
end process;
clk_out <= '1' when COUNT_INT(N-1) = '1' else '0';
process (clk_out)
begin
if (clk_out'event and clk_out = '1') then
      s1<=St;
      s2<=s1;
      s3<=s2;
end if;
end process;
St_in <= s1 and s2 and not s3;
end behav;
```

№	Позначення	Найменування	Дод. Відомості
		Текстові документи	
1	ГЮІК. ХХХХХХ.021 ПЗ	Пояснювальна записка	77 с.
		Графічні документи	
2		Слайд-презентація	19 слайдів
		Носії інформації	
3		Компакт-диск (CD)	1 шт.

Змін.	Арк.	Номер докум.	Підп.	Дата	ХХХХХХ.021 ПЗ			
Розроб.		Шапа Л.С.			Моделі та методи проектування біт-потокового обчислювача степеневих функцій на базі FPGA Відомість атестаційної роботи	Літ	Аркуш	Аркушів
Перевір.		Ларченко Л.В.				У		
Н.контр.		Рожнова Т.Г.		17.12 2020		ХНУРЕ Кафедра АПОТ		
Затв.		Чумаченко С.В.						