ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

Кваліфікаційна робота

«Метод апаратного прискорення згорткових нейронних мереж на FPGA з використанням VHDL»

Виконал:
ст. гр. КСМм-20-1
Шевченко Д.Ю.

Керівник:
доц. Лебедєв О.Г.

Мета та завдання кваліфікаційної роботи

Мета кваліфікаційної роботи: розробка методу апаратного прискорення згорткових нейронних мереж на FPGA з використанням VHDL.

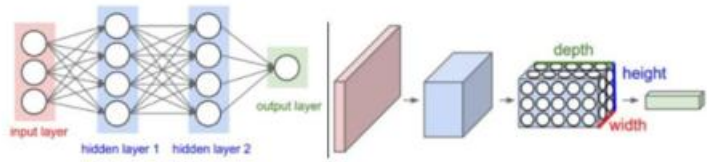
Об'єкт досліджень: згорткові нейронні мережі.

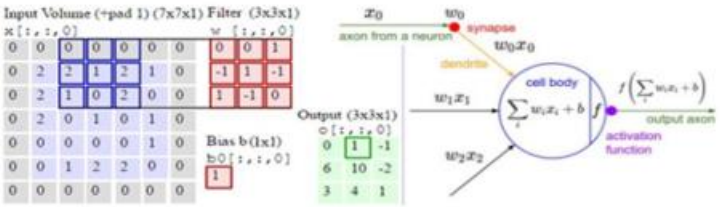
Завдання:

- аналіз топологій згорткових нейронних мереж;
- аналіз існуючих алгоритмів навчання згорткових нейронних мереж;
- дослідження існуючих рішень апаратного прискорення;
- розробка інструменту генерації VHDL.

Згорткові нейронні мережі

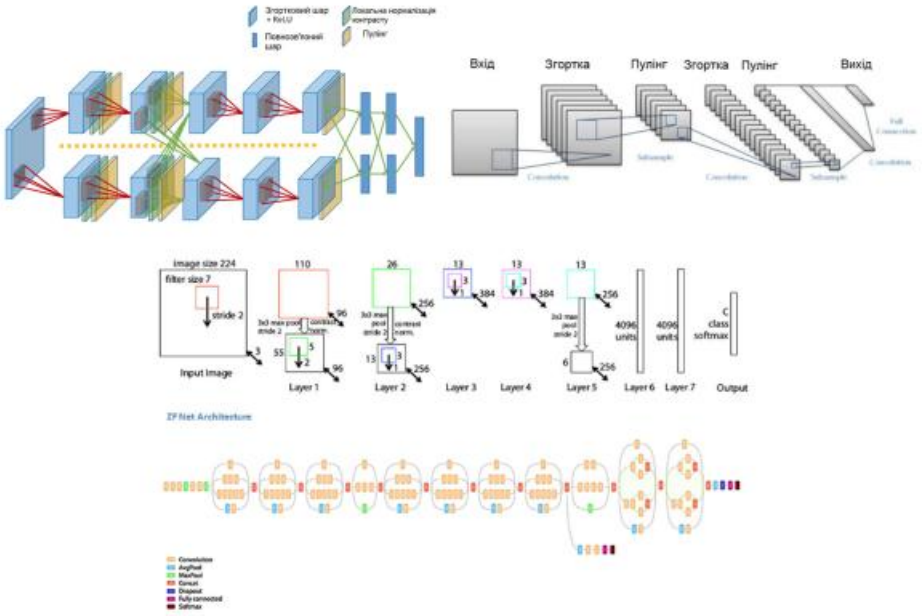
3





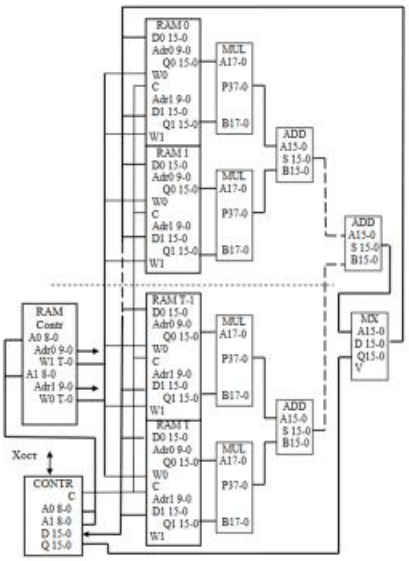
Топології згорткових нейронних мереж

4



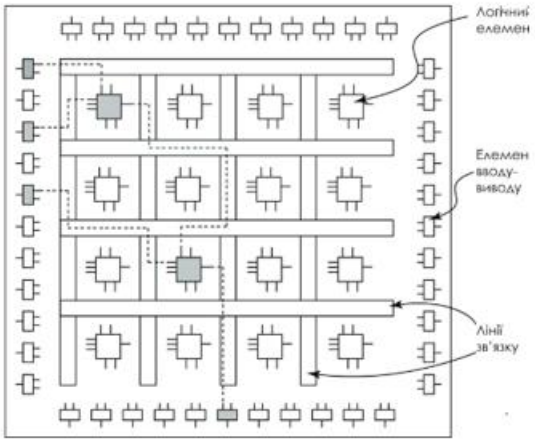
Структурна схема нейромережі на FPGA з паралельним обчисленням ваг

5



Спрощена архітектура FPGA

6



Методика апаратного прискорення

7

Пропонується впровадження інструменту генерації VHDL на основі FPGA для реалізації згорткових нейронних мереж. Для розробки використовується мова Java. Метод призначений для полегшення процесу апаратного прискорення моделей згорткових нейронних мереж з використанням FPGA шляхом параметризації реалізації цих моделей. Інструмент має містити графічний інтерфейс користувача, за допомогою якого можливе налаштування своєї цільової моделі згорткової нейронної мережі, надавши специфікації моделі. Використання методу значно зменшить час розробки, необхідний для впровадження згорткової нейронної мережі, також знащить недоліки, що виникають через складність розробки мов опису апаратних засобів, і має пом'якшити недостатню оптимізацію, викликану інструментами синтезу високого рівня. Інструмент повинен бути оптимізован для створення модульної, масштабованої, реконфігурованої та паралельної реалізації моделей згорткових нейронних мереж.

Інструмент генерації VHDL

8

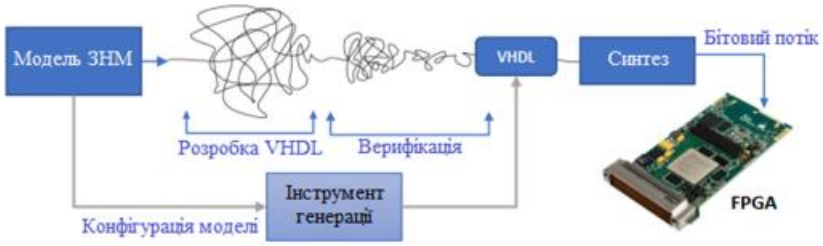
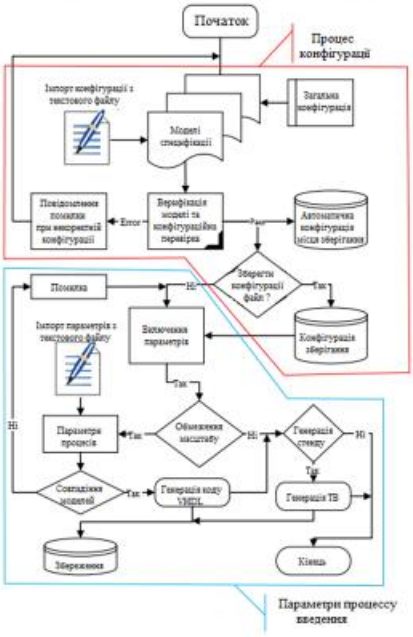


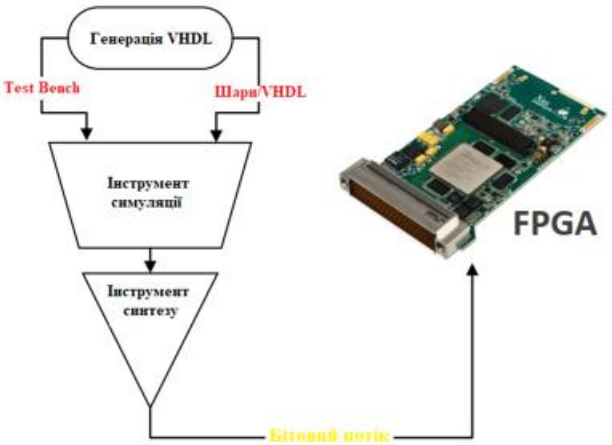
Схема розробленого інструменту

9

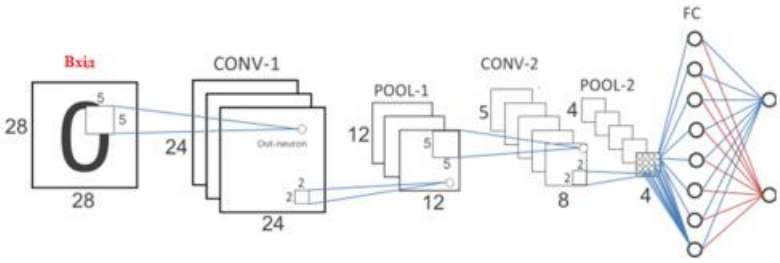


Процес впровадження моделі згорткової нейронної мережі

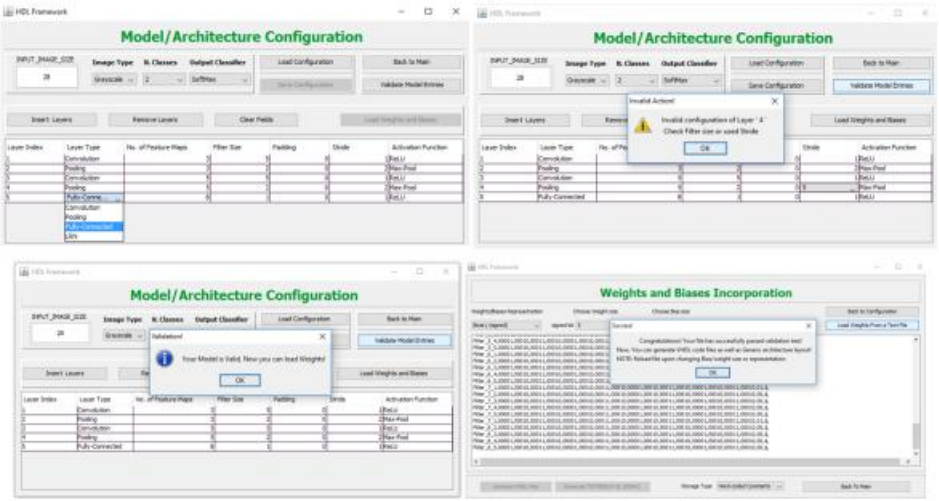
10



Тестова модель згорткової нейронної мережі 11

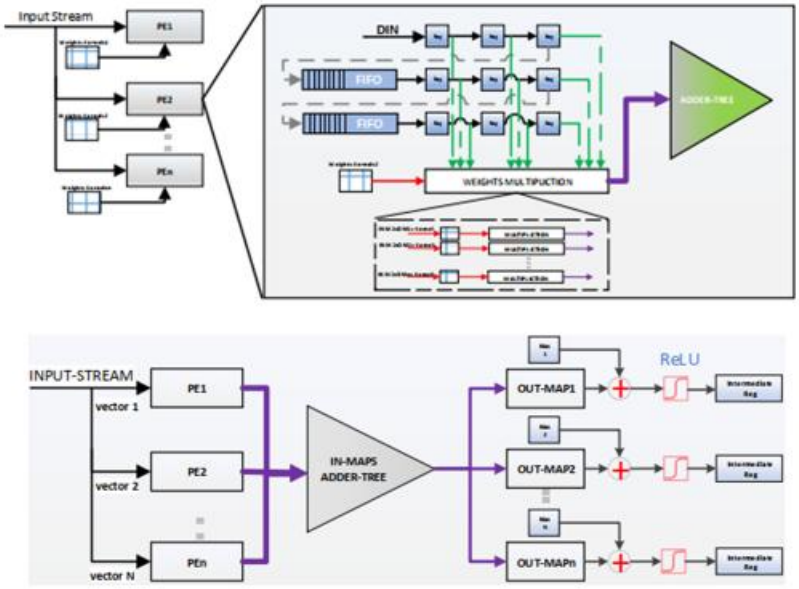


Реалізація моделі 12



Результати роботи

13



Висновки

14

Розроблено метод апаратного прискорення згорткових нейронних мереж на FPGA з використанням VHDL. Проведено аналіз топологій згорткових нейронних мереж; аналіз існуючих алгоритмів навчання згорткових нейронних мереж. Досліджено існуючі рішення апаратного прискорення. Розроблений інструмент генерації VHDL. Використання методу значно зменшить час розробки, необхідний для впровадження згорткової нейронної мережі, також знащить недоліки, що виникають через складність розробки мов опису апаратних засобів.

ДОДАТОК Б

Фрагменти програмного коду

```
GENERIC (
constant PERCISION : positive := 5;
constant DOUT_WIDTH : positive := 5;
constant BIAS_SIZE : positive := 5;
constant MULT_SIZE : positive := 13;
constant MULT_SUM_SIZE : positive := 6;
constant DIN_WIDTH : positive := 8;
constant IMAGE_WIDTH : positive := 13;
constant F_SIZE : positive := 2;
constant WEIGHT_SIZE : positive := 5;
constant BIASES_SIZE : positive := 2;
constant STRIDE : positive := 1;
constant FEATURE_MAPS : positive := 3;
constant VALID_CYCLES : positive := 144;
constant STRIDE_CYCLES : positive := 12;
constant VALID_LOCAL_PIX: positive := 12;
constant ADD_TREE_DEPTH : positive := 2;
constant INPUT_DEPTH : positive := 1;
constant FIFO_DEPTH : positive := 12;
constant USED_FIFOS : positive := 1;
constant ADD_1 : positive := 2;
constant ADD_2 : positive := 1;
constant LOCAL_OUTPUT : positive := 5 );
---------------- ARCHITECTURE DECLARATION - START---------------
-------
architecture Behavioral of CONV_LAYER_1 is
------- INTERNAL FIXED CONSTANT & SIGNALS DECLARATION - START---
--------
type FILTER_TYPE is array (0 to F_SIZE-1, 0 to F_SIZE-1) of
signed(WEIGHT_SIZE- 1 downto 0);
type FIFO_Memory is array (0 to FIFO_DEPTH - 1) of
STD_LOGIC_VECTOR(DIN_WIDTH - 1 downto 0);
type SLIDING_WINDOW is array (0 to F_SIZE-1, 0 to F_SIZE-1) of
STD_LOGIC_VECTOR(DIN_WIDTH- 1 downto 0);
signal VALID_NXTLYR_PIX :integer range 0 to STRIDE_CYCLES;
signal PIXEL_COUNT :integer range 0 to VALID_CYCLES;
signal OUT_PIXEL_COUNT :integer range 0 to VALID_CYCLES;
signal EN_NXT_LYR_1 :std_logic;
signal FRST_TIM_EN_1 :std_logic;
signal Enable_MULT :std_logic;
signal Enable_ADDER :std_logic;
signal Enable_ReLU :std_logic;
signal Enable_BIAS :std_logic;
signal SIG_STRIDE :integer range 0 to IMAGE_SIZE;
signal PADDING_count :integer range 0 to IMAGE_SIZE; --
------------- FILTER HARDCODED CONSTANTS -WEIGHTS START---------
```

```
-------
constant FMAP_1: FILTER_TYPE:=
(("00001","00010"),("00011","00010"));
constant FMAP_2: FILTER_TYPE:= (("00001","00010"),
("00011","00010"));
constant FMAP_3: FILTER_TYPE:= (("00001","00010"),
("00011","00010"));
constant BIAS_VAL_1: signed (BIASES_SIZE-1 downto 0):="01";
constant BIAS_VAL_2: signed (BIASES_SIZE-1 downto 0):="01";
constant BIAS_VAL_3: signed (BIASES_SIZE-1 downto 0):="01";
------------------ MAP NEXT LAYER - COMPONENTS START------------
--------
COMPONENT POOL_LAYER_2
port( CLK,RST :IN std_logic;
DIN_1_2,DIN_2_2, DIN_3_2:IN std_logic_vector(LOCAL_OUTPUT-1
downto 0);
VALID_OUT_2, EN_STREAM_OUT_2 :OUT std_logic;
DOUT_1_2, DOUT_2_2 :OUT std_logic_vector(DOUT_WIDTH-1 downto 0);
EN_STREAM ,EN_LOC_STREAM_2 :IN std_logic );
END COMPONENT POOL_LAYER_2;
begin
POOL_LYR_2 : POOL_LAYER_2
port map(
CLK => CLK,
RST => RST,
DIN_1_2 => DOUT_BUF_1_1,
DIN_2_2 => DOUT_BUF_2_1,
DIN_3_2 => DOUT_BUF_3_1,
DOUT_1_2 => DOUT_1_2,
DOUT_2_2 => DOUT_2_2,
VALID_OUT_2 => VALID_OUT_2,
EN_STREAM_OUT_2 => EN_STREAM_OUT_2,
EN_LOC_STREAM_2 => EN_NXT_LYR_1,
EN_STREAM => EN_STREAM );
```