Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Systemów Elektronicznych

Praca dyplomowa magisterska

na kierunku Elektronika
w specjalności Mikrosystemy i Systemy Elektroniczne

Implementacja sztucznych sieci neuronowych w systemach SoC i MPSoC z wykorzystaniem akceleratorów, realizowanych w technice HLS

Krzysztof Wasilewski
Numer albumu 265956

promotor
dr inż. Wojciech Zabołotny

WARSZAWA 2020

Implementacja sztucznych sieci neuronowych w systemach SoC i MPSoC z wykorzystaniem akceleratorów, realizowanych w technice HLS

Streszczenie

Sztuczne Sieci Neuronowe są w dzisiejszych czasach wykorzystywane w wielu zastosowaniach. Duża złożoność algorytmów Sztucznej Inteligencji wymaga dużej mocy obliczeniowej oraz odpowiednich metod optymalizacji i właściwego wyboru sprzętu. W przypadku obliczeń, które można zrównoleglić, rozsądnym wyborem są układy FPGA (ang. *Field Programmable Gate Array*).

Głównym celem pracy było zaprojektowanie i implementacja sztucznej sieci neuronowej przy wykorzystaniu systemu SoC (ang. *System on Chip*) z układem FPGA i techniki HLS (ang. *High Level Synthesis*). Użycie techniki HLS pozwala na projektowanie przy wykorzystaniu języka C, C++ lub System C i umożliwia korzystanie z wielu bibliotek zaimplementowanych w języku C i C++, co znacznie przyspiesza pracę nad projektem.

Założeniem pracy było stworzenie akceleratora, umożliwiającego osiągnięcie wzrostu wydajności algorytmu detekcji obiektów znajdujących się na obrazie w czasie rzeczywistym. W ostatnich latach można zaobserwować wielki postęp w dziedzinie komputerowego rozpoznawania obrazów (ang. *Computer Vision*), jednak większość dostępnych implementacji jest przeznaczona do uruchomienia na komputerze PC.

Słowa kluczowe: Sztuczne Sieci Neuronowe, HLS, komputerowe rozpoznawanie obrazów, FPGA

Artificial Neural Networks implementation in SoC and MPSoC systems using accelerators synthesized by HLS method

Abstract

Nowadays, Artificial Neural Networks (ANN) are used in many applications. High complexity of Artificial Intelligence algorithms requires high computing power, appropriate optimization methods and efficient hardware. In the case of computation that is easy to parallelize it is reasonable to use FPGA systems.

The main aim of the thesis was to design and implement an Artificial Neural Network algorithm using SoC (System on Chip) with FPGA system and HLS (High-Level Synthesis) method. HLS method allows to design a project using C, C++ or System C language and use lots of C libraries, which makes working on the project faster.

Assumption was made that the created accelerator will allow to achieve efficiency improvement in the real-time object detection algorithm. Recently, it is seen that huge improvement was made in the field of Computer Vision, but most of the available implementations are made to run on PC.

Keywords: Artificial Neural Networks, HLS, Computer Vision, FPGA

załącznik nr 3 do zarządzenia
nr 28 /2016 Rektora PW

Politechnika Warszawska

……..
miejscowość i data

……..
imię i nazwisko studenta

……..
numer albumu

…….…….
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta

Spis treści

1. Wstęp

1.1. Wprowadzenie

W ostatnich latach można zaobserwować gwałtowny rozwój w dziedzinie Uczenia Maszynowego (ang. *Machnie Learning*) i Sztucznej Inteligencji (ang. *Artificial Intelligence*). Jednym z algorytmów, który powstał już dość dawno są Sztuczne Sieci Neuronowe (ang. *Artificial Neural Networks*). Większość algorytmów wykorzysztujących Sztuczną Inteligencję wymaga dużej mocy obliczeniowej i wyboru odpowiedniego sprzętu. Często powtarzaną operacją matematyczną w przypadku algorytmu Sztucznej Sieci Neuronowej jest mnożenie macierzy. Działanie to można w łatwy sposób zrównoleglić, implementując sieć w układzie FPGA i tym samym zwiększyć efektywność algorytmu.

Praca nad implementacją algorytmów Sztucznej Inteligencji w większości przypadków zaczyna się od stworzenia i uruchomienia modelu. Do tego zadania często wykorzystywane są biblioteki takie jak *Keras* lub *Theano*, które w znacznym stopniu przyspieszają proces tworzenia oprogramowania oraz ułatwiają wprowadzanie zmian w modelu sieci. Rozwój i dopracowywanie algorytmu Sztucznej Inteligencji wymaga wielu iteracji uruchamiania kodu z różnymi parametrami i właściwościami sieci.

1.2. Wstęp teoretyczny

Temat Sieci Neuronowe ma długą historię rozwoju, sięgającą początku lat 40-tych XX wieku, jednak w ostatnich latach można zaobserwować znaczny postęp w tej dziedzinie[1]. Rozwój technologii umożliwił zastosowanie algorytmów AI w wielu aplikacjach.

1.2.1. Model neuronu

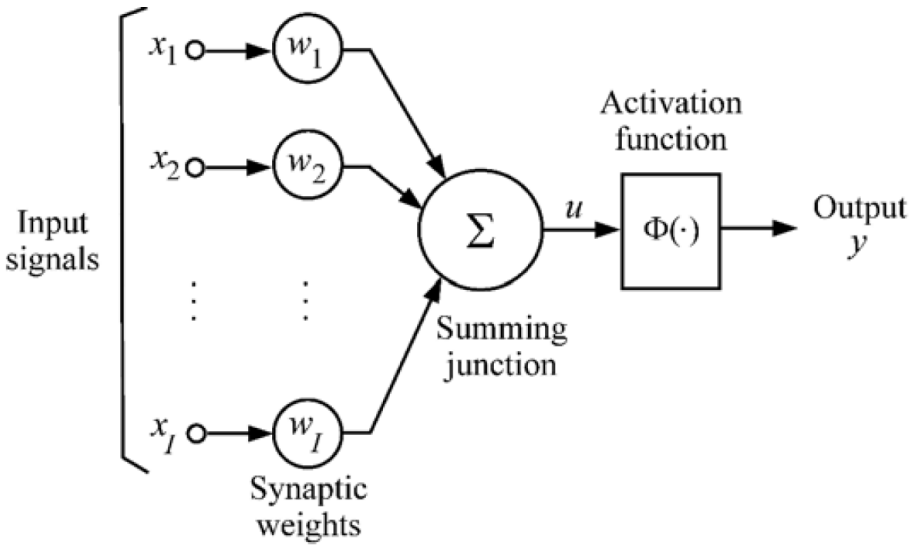
Sztuczne Sieci Neuronowe to algorytm wzorowany działaniem ludzkiego mózgu i znajdujących się w nim neuronów. Model matematyczny pojedynczego neuronu - Perceptron [2] składa się z wektora wejściowego $x = (x_1, x_2, ..., x_I)^T$, wektora wag $w = (w_1, w_2, ..., w_I)^T$ przypisanych do każdego z wejść i funkcji aktywacji $\phi(u)$ (Rys. 1.1). Wyjście neuronu można policzyć według wzoru

$x = (x_1, x_2, ..., x_I)^T$, wektora wag $w = (w_1, w_2, ..., w_I)^T$

1.2.2. Funkcja aktywacji

Na działanie algorytmu znaczny wpływ może mieć dobór odpowiedniej funkcji aktywacji. Wśród najczęściej stosowanych funkcji aktywacji wyróżnia się:

- funkcję ReLu: $\phi(u) = max(0, x)$
- funkcję sigmoid: $\phi(u) = \frac{a}{a+exp(-bu)}$
- funkcję softmax: $\phi(u) = \frac{a}{a+exp(-bu)}$



Rysunek 1.1. Model pojedynczego perceptronu

1.2.3. Perceptron wielowarstwowy

Jednym z pierwszych modeli Sztucznych Sieci Neuronowych był Perceptron Wielowarstwowy (MLP - ang. *Multi-Layer Perceptron*), składający się z warstwy neuronów wejściowej, ukrytych i wyjściowej (Rys.1.2). Wyjście neuronów w danej warstwie staje się wejściem neuronów warstwy następnej.

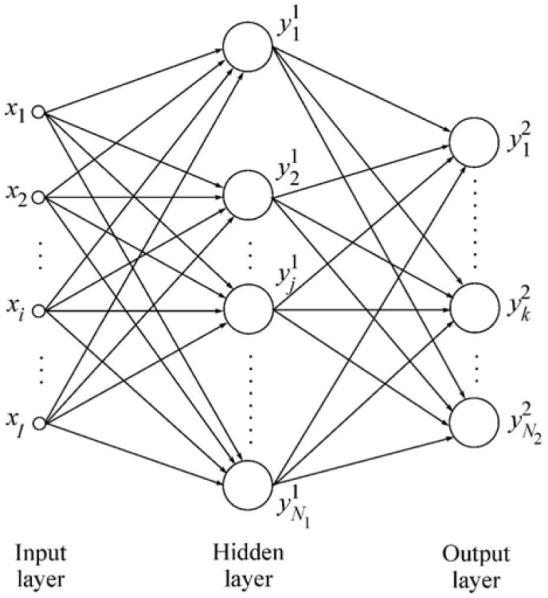
Często spotykaną wersją MLP jest model, z warstwami w pełni połączonymi (ang. *FC - Fully Connected*). W warstwie FC każde z wyjść jest podłączone do wszystkich wejść neuronów w warstwie następnej. Sieć posiadającą wszystkie warstwy w pełni połączone nazywana jest siecią w pełni połączoną (ang. *FC - Fully Connected Network*).

1.2.4. Uczenie Głębokie

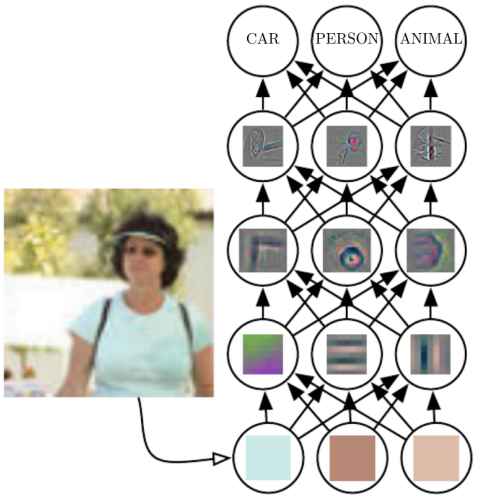
Rozwój algorytmów AI doprowadził do powstania Głębokich Sieci Neuronowych (ang. *DNN - Deep Neural Networks*), czyli takich, które posiadają wiele warstw ukrytych[3]. Algorytm Uczenia Głębokiego (ang. *Deep Learning*) umożliwia rozwiązywanie skomplikowanych problemów takich jak rozpoznawanie i klasyfikację obiektów na obrazie(Rys.1.3. Seria warstw ukrytych (ang. *hidden layers*) umożliwia ekstrakcję cech obiektów. Kolejne warstwy umożliwiają wykrywanie krawędzi, potem konturów, a na końcu całych kształtów i obiektów.

1.2.5. Wsteczna propagacja błędu

Architekturę sieci MLP często stosuje się wraz algorytmem wstecznej propagacji błędu (ang. Backpropagation), która umożliwia proces uczenia sieci. Poprzez obliczenie błędu w neuronach warstwy wyjściowej i przepropagowaniu wstecz błędu przez całą sieć pozwala



Rysunek 1.2. Model Perceptronu Wielowarstwowego



Rysunek 1.3. Model Głębokiego Uczenia sieci

dostosować wartość wagi każdej z krawędzi w taki sposób aby zminimalizować wartość funkcji kosztu.

- learning rule (w szczególności delta rule) - uczenie nadzorowane/nienadzorowane

1.3. Implementacja Sztucznych Sieci Neuronowych w układach FPGA

1.3.1. Reprezentacja liczb zmiennoprzecinkowych

1.3.2. Uczenie Sztucznej Sieci Neuronowej

1.4. Zastosowania Sztucznych Sieci Neuronowych

2. Cel i zakres pracy

Celem pracy było zaprojektowanie i implementacja sztucznej sieci neuronowej przy wykorzystaniu systemu SoC (ang. System on Chip) i techniki HLS. Użycie metody HLS pozwala na projektowanie przy wykorzystaniu języka C, C++ lub System C, co przyspiesza pracę nad projektem. Dodatkowo HLS umożliwia korzystanie z wielu bibliotek, które pozwalają wygodnie używać funkcji, które są wykorzystywane w implementacji Sztucznych Sieci Neuronowych. Efektem pracy powinno być stworzenie akceleratora, umożliwiającego osiągnięcie wzrostu wydajności w stosunku do rozwiązań software'owych.

2.1. Motywacja

Sztuczne Sieci Neuronowe są związane z dużą ilością obliczeń, które mogą być wykonywane równolegle. Pozwala to osiągnąć krótszy czas wykonania programu, co ma duże znaczenie dla zastosowań w systemach działających w czasie rzeczywistym np. w branży Automotive. Aby osiągnąć przyspieszenie obliczeń stosuje się różne metody. Jednym z najpopularniejszych obecnie sposobów na zwiększenie wydajności algorytmów AI jest wykorzystanie kart graficznych GPU (ang. *Graphics Processing Unit*). Metodą najbardziej przyspieszającą obliczenia, lecz wymagającą najdłuższego czasu projektowania i najbardziej kosztowną, jest zastosowanie specjalizowanych układów ASIC (ang. *Application-Specific Integrated Circuit*). Opcją pośrednią pomiędzy powyższymi dwoma rozwiązaniami jest zastosowanie układów FPGA. To podejście umożliwia osiągnięcie znacznego przyspieszenia wykonywania obliczeń i nie powoduje wielkiego wzrostu kosztów. Dodatkowo zastosowanie metody HLS ułatwia i minimalizuje czas tworzenia sprzętowej implementacji modelu ANN oraz wprowadzanie zmian w projekcie.

3. Wybór sprzętu

Dlaczego Sztuczne Sieci Neuronowe odpala się na GPU? Dlaczego nie CPU i GPU tylko FPGA? Typically, neural networks are designed, trained, and executed on a conventional processor, often with GPU acceleration. But for embedded devices which may need to process data at multiple-MHz sample rates, the computational requirements can be overwhelming for an embedded processor where no GPU is available, creating a tempting opportunity for FPGA acceleration. (https://github.com/Xilinx/RFNoC-HLS-NeuralNet) Co z ASIC, dlaczego rzadko się je stosuje, jak wygląda proces tworzenia? Dlaczego PC ma swoje ograniczenia? Jakie możliwości mają nowe algorytmy uruchamiane na PC? Jaką przewagę dają układy FPGA, skąd się to bierze? porównanie zużycia mocy itp..

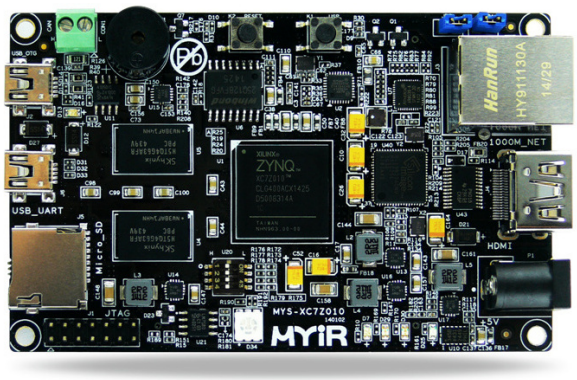
Trzeba tu tylko uważać, żeby nie powielać tekstu z rozdziału cel i zakres pracy.

Tabela 3.1. Porównanie cen płytek z układami Zynq firmy Xilinx

Nazwa płytki	Układ SoC	Cena
Z-turn Board MYS-7Z010-C-S	XC7Z010-1CLG400C	99$[1]
Z-turn Board MYS-7Z020-C-S	XC7Z020-1CLG400C	119$[1]
Zybo Z7-10 Development Board	XC7Z010-1CLG400C	199$[2]
Zybo Z7-20 Development Board	XC7Z020-1CLG400C	299$[2]
ZedBoard Zynq-7000	XC7Z020-CLG484-1	449$[3]

3.1. Z-turn Board

Z-turn Board (Rys. 3.1 jest komputerem jednopłytkowym (ang. SBC – *Single Board Computer*), opartym o układ SoC Xilinx Zynq-7020 (XC7Z020-1CLG400C), zawierający dwurdzeniowy procesor ARM Cortex-A9 i układ FPGA Artix 7.



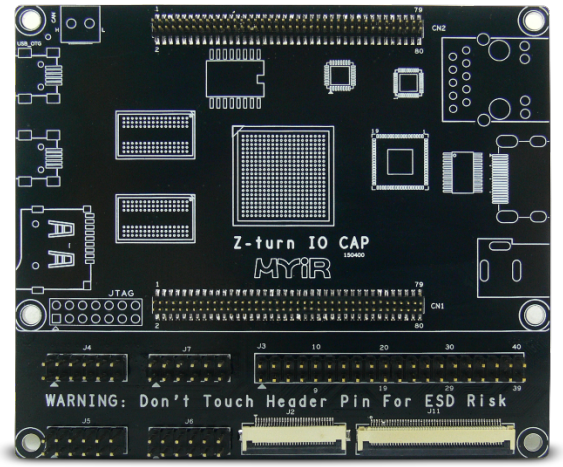
Rysunek 3.1. Płytka Z-turn-Board 7020

[1] http://www.myirtech.com/list.asp?id=502
[2] https://store.digilentinc.com/zybo-z7-zynq-7000-arm-fpga-soc-development-board/
[3] https://store.digilentinc.com/zedboard-zynq-7000-arm-fpga-soc-development-board/

Biorąc pod uwagę parametry, płytka charakteryzuje się wysokim stodunkiem ceny do jakości, podstawowa wersja kosztuje 99$. Dla porównania płytka Zybo Z7-20 kosztuje 199$. Zestawienie cen płytek zawierających układ Zynq XC7Z010 oraz XC7Z020 znajduje się w Tabeli 3.1.

3.1.1. Interfejsy komunikacji

Płytka Z-turn posiada interfejsy UART oraz Ethernet, które zostały wykorzystane do komunikacji komputera PC z systemem przy użyciu portu szeregowego i protokołu SSH (ang. *Secure Shell*). Istnieje również możliwość podłączenia wyświetlacza bezpośrednio do płytki przy użyciu portu HDMI. Dodatkowo producent oferuje płytkę rozszerzeniową Z-turn IO-Cape (Rys. 3.2), która zawiera porty do podłączenia kamery przez protokół DVP (ang. *Digital Video Port*) oraz wyświetlacza LCD.



Rysunek 3.2. Płytka rozszerzeniowa Z-turn IO Cape

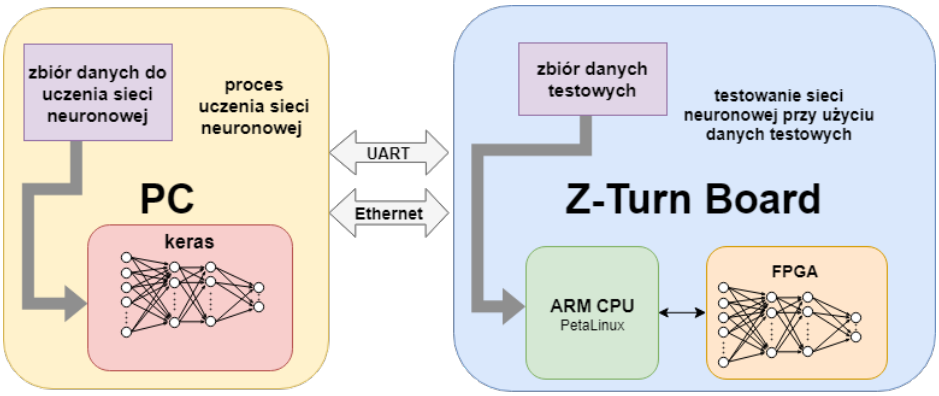
3.2. Kamera

4. Implementacja

4.1. Schemat blokowy systemu

System składa się z dwóch głównych części (Rys. 4.1):

- aplikacja wykorzystująca pakiet keras, uruchamiana na komputerze PC
- część uruchamiana na płytce Z-Turn Board



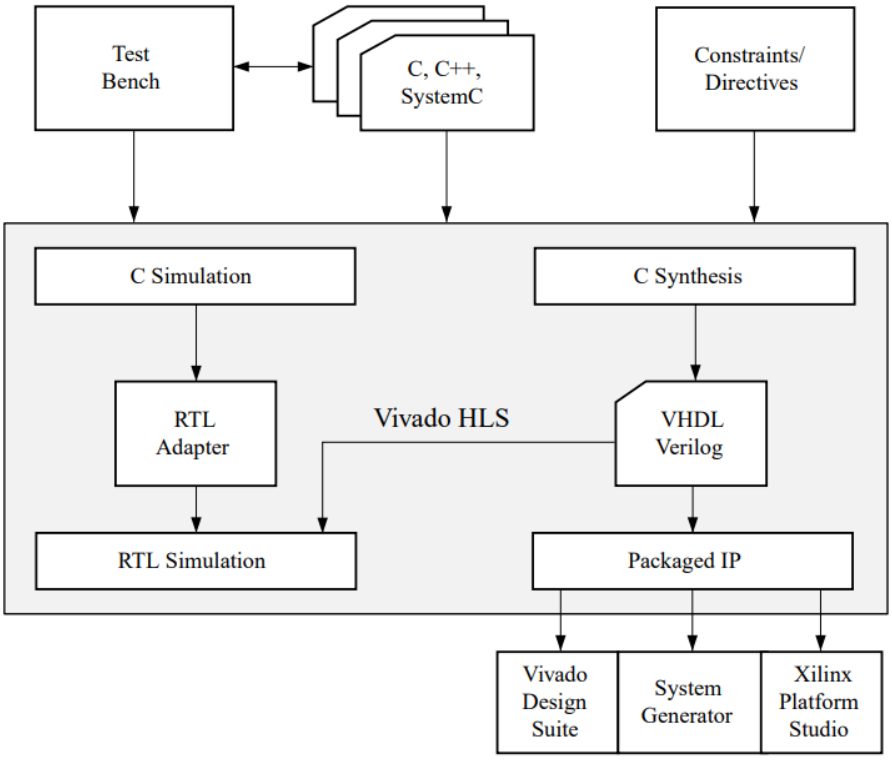
Rysunek 4.1. Schemat blokowy systemu

4.2. Wykorzystanie metody HLS

Przy użyciu metody HLS możliwe jest stworzenie własnego bloku IP (ang. Intellectual Property), który następnie jest umieszczany w katalogu IP i można go wielokrotnie wykorzystać w projekcie RTL (ang. Register Transfer Level). Do projektu z użyciem HLS potrzebny jest (Rys.4.2) plik z algorytmem w języku C/C++ lub System C, plik testowy napisany w języku C (ang. testbench) oraz plik z opisem ograniczeń sprzętowych (ang. constraints). Kolejne etapy projektu z wykorzystaniem metody HLS [4]:

1. Kompilacja, wykonanie (symulacja) i debugowanie algorytmu napisanego w języku C
2. Synteza algorytmu w języku C w implementację RTL
3. Wygenerowanie raportu i analiza projektu
4. Zweryfikowanie implementacji RTL
5. Spakowanie implementacji RTL w blok IP

Zastosowanie syntezy wysokiego poziomu umożliwia przeniesienie algorytmu napisanego w języku C/C++ lub System C na implementację w układzie FPGA. Dodatkową zaletą metody HLS jest dostępność bibliotek do przetwarzania obrazów oraz ułatwiających implementację operacji matematycznych.



Rysunek 4.2. Proces projektowania przy użyciu metody HLS

4.3. Petalinux

4.4. Uczenie Sztucznej Sieci Neuronowej

4.5. Zbiór danych wejściowych

W procesie uczenia oraz testowania poprawności działania modelu sztucznej sieci neuronowej wykorzystano zbiór odręcznie pisanych cyfr MNIST (ang. THE MNIST DATABASE of handwritten digits) [5]

4.6. Testowanie systemu

Algorytmy rozpoznawania obiektów mogą być wywoływane na różne sposoby. Jedną z metod jest rejestrowanie obrazu z możliwie maksymalną ilością klatek na sekundę, analizowanie każdej ramki, wyszukiwanie obiektów i klasyfikacja za pomocą algorytmu ANN. Drugim, prostszym w implementacji sposobem, jest wywoływanie zarejestrowania obrazu w momencie, gdy użytkownik, chce dokonać klasyfikacji obiektu, który znajduje się w zasięgu obiektywu kamery, a na zarejestrowanym obrazie nie ma innych obiektów. Z powodu ograniczeń zasobów systemu, na którym aplikacja była testowana oraz ograniczonego czasu wykonania projektu, podjęto decyzję o zastosowaniu drugiej metody.

5. Wyniki i wnioski

6. Posumowanie

Bibliografia

[1] D. Kriesel, *A Brief Introduction to Neural Networks*. 2007. adr.: `available%20at%20http://www.dkriesel.com`.

[2] A. Omondi i J. Rajapakse, "FPGA Implementations of Neural Networks", 2006.

[3] I. Goodfellow, Y. Bengio i A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[4] *Vivado Design Suite User Guide High-Level Synthesis (UG871 (v2019.2))*, 2020.

[5] Y. LeCun i C. Cortes, "MNIST handwritten digit database", 2010. adr.: `http://yann.lecun.com/exdb/mnist/`.

Wykaz symboli i skrótów

ANN – ang. *Artificial Neural Network*
ASIC – ang. *Application-Specific Integrated Circuit*
DVP – ang. *Digital Video Port*
FPGA – ang. *Field Programmable Gate Array*
FPGA – ang. *Graphics Processing Unit*
HLS – ang. *High Level Synthesis*
MPSoC – ang. *Multi-Processor System-on-Chip*
SBC – ang. *Single Board Computer*
SoC – ang. *System on Chip*
SSH – ang. *Secure Shell*

Spis rysunków

Spis tabel