AKADEMIA GÓRNICZO-HUTNICZA

WEAIiIB, Katedra Automatyki i Robotyki

Laboratorium Biocybernetyki

Przedmiot:	**Akceleracja Algorytmów Wizyjnych w GPU i OpenCL**			
			pr24aaw05	
Temat projektu:				
	Detekcja cieni			
Wykonali: Brzeziński Jan Kocwa Michał Kopacz Kamila				
	Automatyka i Robotyka		studia:	magisterskie
Rok akademicki: 2023/2024		Semestr letni		
Prowadzący:		dr inż. M. Jabłoński		

wersja 1.2

Kraków, maj, 2024r.

Spis treści:

1. Streszczenie

Niniejszy projekt ma na celu opracowanie efektywnej metody detekcji cieni na obrazach liści przy wykorzystaniu algorytmu Mean Shift oraz operacji morfologicznych i implementacji algorytmu w środowisku OpenCL. Głównym celem projektu jest dostarczenie narzędzia umożliwiającego automatyczną detekcję obszarów cieni na obrazach liści z wysoką dokładnością i wydajnością, wykorzystując środowisko OpenCL do zrównoleglenia obliczeń i przyspieszenia procesu detekcji. Założenia projektowe obejmują wykorzystanie istniejącego zestawu danych do testowania i walidacji algorytmu detekcji cieni, implementację algorytmu Mean Shift i operacji morfologicznych w języku OpenCL, ocenę skuteczności i wydajności algorytmu detekcji cieni oraz udostępnienie finalnego narzędzia detekcji cieni jako aplikacji w środowisku OpenCL. Propozycja rozwiązania obejmuje przeprowadzenie kompleksowego procesu, począwszy od implementacji algorytmu w OpenCL, poprzez analizę danych i wyników, aż po wdrożenie gotowego narzędzia detekcji cieni na liściach. Alternatywne rozwiązania, takie jak wykorzystanie uczenia maszynowego czy metody superpiksel, zostały również rozważone, lecz uwzględniono ograniczenia związane z dostępnością danych treningowych, złożonością implementacji oraz wydajnością detekcji. Ostateczny wybór algorytmu Mean Shift opiera się na jego potencjale w detekcji obszarów o podobnych właściwościach, co może być korzystne w identyfikacji obszarów cieni na obrazach liści.

Słowa kluczowe: detekcja cieni, algorytm Mean Shift, środowisko OpenCL, segmentacja obrazów, operacje morfologiczne.

2. Wstęp

a) Cele i założenia projektu.

Celem projektu jest opracowanie efektywnej metody detekcji cieni na obrazach liści przy wykorzystaniu algorytmu Mean Shift oraz operacji morfologicznych i implementacja algorytmu w środowisku OpenCL. Projekt skupia się na dostarczeniu narzędzia umożliwiającego automatyczną detekcję obszarów cieni na obrazach liści z wysoką dokładnością i wydajnością, wykorzystując środowisko OpenCL do zrównoleglenia obliczeń i przyspieszenia procesu detekcji.

Założenia projektowe obejmują:

1. Wykorzystanie istniejącego zestawu danych zawierającego zdjęcia liści w stałych warunkach oświetleniowych do testowania i walidacji algorytmu detekcji cieni.

2. Implementację algorytmu Mean Shift i operacji morfologicznych w języku OpenCL, aby umożliwić przetwarzanie równoległe na różnych urządzeniach, takich jak karty graficzne i procesory wielordzeniowe.
3. Ocena skuteczności i wydajności algorytmu detekcji cieni w środowisku OpenCL na różnych urządzeniach.
4. Udostępnienie finalnego narzędzia detekcji cieni na liściach jako aplikacji w środowisku OpenCL, co umożliwi praktyczne wykorzystanie opracowanej metody.

Implementacja metody detekcji cieni poprzez wykorzystanie algorytmu Mean Shift bazuje na przeprowadzonych w fazie wstępnej projektu badaniach literaturowych:

1. Salvador E, Andrea C, Ebrahimi T, *Cast shadow segmentation using invariant color features*, 2004 - koncentruje się na problemie segmentacji cieni rzucanych na obrazach, przy wykorzystaniu cech kolorów inwariantnych, czyli takich, które pozostają stałe niezależnie od zmian w oświetleniu, geometrii obiektów czy innych czynników wpływających na wygląd obrazu.
2. Bradski G. R, *Real Time Face and Object Tracking as a Component of a Perceptual User Interface*, 1998 - skupia się na problemie śledzenia twarzy i obiektów w czasie rzeczywistym jako części interfejsu percepcyjnego dla użytkowników. Systemy śledzenia obiektów często wykorzystują algorytmy przetwarzania obrazu, które mogą być również użyteczne w detekcji cieni. Techniki przetwarzania obrazu, takie jak segmentacja, analiza kolorów czy detekcja krawędzi, mogą być stosowane zarówno do śledzenia obiektów, jak i detekcji cieni na obrazach.
3. Benedek C, Sziranyi T, *Study on Color Space Selection for Detecting Cast Shadows in Video Surveillance*, 2007 - koncentruje się na badaniach dotyczących wyboru przestrzeni kolorów do detekcji rzutowanych cieni w monitoringu wideo.

Na podstawie przeprowadzonej analizy problemu, została podjęta decyzja o implementacji metody detekcji cieni na bazie segmentacji przy użyciu algorytmu Mean Shift. Algorytm ten jest używany do segmentacji obiektów na obrazach poprzez grupowanie pikseli o podobnych właściwościach, takich jak kolor czy tekstura. Dzięki temu można wyodrębnić obiekty na tle i przeprowadzić dalszą analizę. Wybór przestrzeni barw, w której implementowany by miał być algorytm, pomimo że na bazie przeprowadzonych badań początkowo został uznany za istotny, po dokładniejszej konsultacji z prowadzącym został uznany za mniej ważny i zostawiony do późniejszej dyskusji po właściwym zaimplementowaniu metody w środowisku OpenCL.

b) Zarys ogólny proponowanego rozwiązania.

1. Przygotowanie danych:
    - Zaimportowanie istniejącego zestawu danych zawierającego obrazy liści w formie wejściowej.
    - Przeprowadzenie wstępnej analizy danych w celu zrozumienia struktury obrazów i charakterystyki cieni.
2. Implementacja algorytmu Mean Shift w OpenCL:
    - Przetłumaczenie klasycznego algorytmu Mean Shift na język OpenCL w celu zrównoleglenia obliczeń.
    - Zdefiniowanie struktury danych i operacji niezbędnych do realizacji procesu detekcji cieni w środowisku OpenCL.
3. Inicjalizacja i aktualizacja okien Mean Shift:
    - Opracowanie strategii inicjalizacji okien Mean Shift w obszarach podejrzanych o występowanie cieni na liściach.
    - Zaimplementowanie iteracyjnej procedury aktualizacji okien w celu śledzenia obszarów o największej gęstości pikseli, co może wskazywać na obecność cieni.
4. Analiza wyników i progowanie:
    - Ocena wyników algorytmu Mean Shift pod kątem identyfikacji obszarów potencjalnych cieni na obrazach liści.
    - Implementacja operacji morfologicznych na obrazach
    - Zastosowanie progowania lub innych technik filtracji w celu klasyfikacji pikseli jako cienie lub nie.
5. Ocena skuteczności i wydajności:
    - Analiza wydajności implementacji algorytmu w środowisku OpenCL pod kątem szybkości działania i zużycia zasobów.
6. Wdrożenie i udostępnienie narzędzia:
    - Stworzenie parametryzowalnej aplikacji umożliwiającej łatwe użycie opracowanej metody detekcji cieni na obrazach liści.

Zaproponowane rozwiązanie obejmuje kompleksowy proces, począwszy od implementacji algorytmu w OpenCL, poprzez analizę danych i wyników, aż po wdrożenie gotowego narzędzia detekcji cieni na liściach. Dzięki zastosowaniu środowiska OpenCL możliwe będzie osiągnięcie szybkiego przetwarzania równoległego, co przyczyni się do efektywnej detekcji cieni na dużych zbiorach danych.

c) Dyskusja alternatywnych rozwiązań.

Do przeprowadzenia detekcji cieni na obrazach możliwe jest użycie alternatywnych metod, takich jak:

1. Wykorzystanie uczenia maszynowego: Metody uczenia maszynowego, takie jak klasyfikacja za pomocą klasyfikatorów lub głębokich sieci neuronowych, mogą być stosowane do detekcji cieni na obrazach poprzez trenowanie modeli na danych treningowych zawierających przykłady cieni i braku cieni. Niestety wadami tego rozwiązania jest potrzeba posiadania dużego i różnorodnego zbioru danych treningowych zawierających przykłady cieni i braku cieni, natomiast zbiór wykorzystywany w tym projekcie jest znacznie ograniczony. Zbieranie takich danych może być czasochłonne i kosztowne. Ponadto, zaawansowane modele klasyfikacyjne, szczególnie te oparte na głębokich sieciach neuronowych, mogą być złożone obliczeniowo i wymagają dużej mocy obliczeniowej do treningu i wykorzystania.
2. Superpiksel: Metoda superpiksel, znana również jako segmentacja na podstawie superpikseli, może być używana do detekcji cieni na obrazach. Użycie tej metody pozwala uwzględnić lokalne informacje o strukturze obrazu, co może przyczynić się do lepszej wydajności w porównaniu z globalnymi metodami analizy histogramu czy klasyfikacji pikseli. Jednakże, metoda superpiksel może mieć trudności w detekcji małych cieni lub cieni o niskiej kontraście względem otoczenia, zwłaszcza gdy są one zawarte w superpikselach o dużej rozpiętości cech. Problemem jest też złożoność implementacyjna. Implementacja metody superpiksel może wymagać zaawansowanych technik przetwarzania obrazów i obliczeń, co może być trudne dla osób bez odpowiedniego doświadczenia w dziedzinie wizji komputerowej.

d) Pokazanie logicznej konstrukcji pracy, czyli „co z czego będzie wynikać”. Przykładowo: *„W rozdziale 3 podane zostały informacje i uwagi dotyczące formy i treści informacji, które powinny znaleźć się w rozdziale 5”.*

3. Analiza złożoności i estymacja zapotrzebowania na zasoby

Implementację rozważanego rozwiązania należało przeanalizować pod względem kilku czynników, uwzględniając fakt, że obrazy wejściowe mają wymiary 6993 x 4564 pikseli, natomiast ich format może być dowolny, ponieważ nie wpływa na jakość wykonywania się algorytmu.

Na złożoność implementacji proponowanego rozwiązania składają się między innymi:

1. Złożoność obliczeniowa - Proces segmentacji wymaga wielokrotnego przetwarzania obrazu, a każda iteracja algorytmu Mean Shift obejmuje obliczenia dla każdego wycinka obrazu, co może prowadzić do znacznego obciążenia zasobów obliczeniowych.
2. Złożoność pamięciowa - Wymagane jest przechowywanie obrazu wejściowego oraz wyników pośrednich algorytmu, co może wymagać znacznej ilości pamięci RAM. W

przypadku implementacji w środowisku OpenCL, konieczne jest również przechowywanie danych na urządzeniu GPU, co może wpłynąć na dostępność pamięci.

3. Złożoność komunikacyjna - W przypadku wykorzystania GPU do przetwarzania, istnieje dodatkowa złożoność związana z transferem danych między pamięcią CPU a GPU. Wysyłanie dużych obrazów z pamięci CPU do pamięci GPU oraz przesyłanie wyników z powrotem do pamięci CPU może być czasochłonne i wymagać odpowiedniej optymalizacji.

Oszacowanie zapotrzebowania na zasoby obliczeniowe zostało oparte głównie na złożoności obliczeniowej i pamięciowej algorytmu oraz dostępnych zasobach sprzętowych:

1. Wysoka rozdzielczość obrazów oraz złożoność algorytmu Mean Shift mogą znacząco wpłynąć na czas przetwarzania. Aby spełnić rygory czasowe, konieczne będzie zoptymalizowanie algorytmu oraz wykorzystanie odpowiednich zasobów obliczeniowych.
2. Zasoby obliczeniowe CPU: Algorytm Mean Shift jest intensywny obliczeniowo, zwłaszcza dla dużych obrazów. Zasoby CPU, takie jak liczba rdzeni i częstotliwość taktowania, będą miały istotny wpływ na wydajność przetwarzania. W przypadku CPU, konieczne byłoby wykorzystanie wielowątkowości w celu równoległego przetwarzania danych.
3. Zasoby Obliczeniowe GPU: Środowisko OpenCL umożliwia wykorzystanie zasobów GPU do równoległego przetwarzania. Zasoby GPU, takie jak liczba jednostek obliczeniowych i ilość pamięci GPU, będą decydujące dla efektywności przetwarzania. Dla zadań intensywnych obliczeniowo, GPU jest bardziej efektywne niż CPU.
4. Duże obrazy wymagają znacznej ilości pamięci do przechowywania danych wejściowych oraz wyników pośrednich. Zarówno CPU, jak i GPU muszą dysponować odpowiednią ilością pamięci, aby obsłużyć te duże obrazy oraz wyniki pośrednie algorytmu.

Podsumowując, detekcja cieni na dużych obrazach w środowisku OpenCL będzie wymagać znacznych zasobów obliczeniowych i pamięciowych zarówno dla CPU, jak i GPU. Kluczowe będzie optymalizowanie algorytmu oraz wykorzystanie dostępnych zasobów sprzętowych w celu spełnienia rygorów czasowych i zapewnienia efektywnego przetwarzania.

4. Koncepcja proponowanego rozwiązania

Koncepcja proponowanego rozwiązania problemu detekcji cieni na obrazach liści metodą segmentacji Mean Shift w środowisku OpenCL opiera się na wykorzystaniu równoległego przetwarzania na GPU w celu przyspieszenia operacji na dużych rozmiarach danych wejściowych. Pierwszym krokiem jest wykonanie segmentacji na obrazach. Algorytm Mean Shift zostanie zaimplementowany w języku OpenCL, co umożliwi wykorzystanie zasobów GPU do równoległego przetwarzania. Obraz zostanie podzielony na wycinki, a każdy z nich będzie przetwarzany

niezależnie, co umożliwi wykorzystanie potencjału równoległego przetwarzania GPU. Następnie na wycinkach obrazów zostaną przeprowadzone operacje morfologiczne, takie jak erozja i dylatacja, w celu poprawienia jakości detekcji cieni. One również zostaną zaimplementowane w języku OpenCL, aby wykorzystać potencjał równoległego przetwarzania GPU.

Struktura topologiczna systemu:

- Na platformie sprzętowej wykorzystane zostaną głównie zasoby GPU do przetwarzania obrazów.
- Algorytm Mean Shift oraz operacje morfologiczne będą wykonywane przez kernele OpenCL uruchamiane na GPU.
- Platforma programowa będzie odpowiedzialna za koordynację procesu przetwarzania, zarządzanie pamięcią oraz ewentualne operacje wstępnej obróbki danych.

Rozpartycjonowanie zadań:

- Algorytm Mean Shift oraz operacje morfologiczne zostaną rozpartycjonowane na jednostki funkcjonalne, uruchamiane na GPU.
- CPU będzie odpowiedzialne głównie za zarządzanie procesem przetwarzania oraz ewentualne operacje wstępnej obróbki danych.

Protokoły komunikacyjne:

- Komunikacja między CPU a GPU będzie odbywać się za pomocą odpowiedniego API, OpenCL.
- Nie będzie potrzeby komunikacji między innymi jednostkami funkcjonalnymi, ponieważ przetwarzanie będzie realizowane w głównej mierze na GPU.

Wykorzystane narzędzia:

- Do implementacji algorytmu Mean Shift oraz operacji morfologicznych w OpenCL wykorzystane zostanie środowisko programistyczne dostarczane przez producentów.
- Do koordynacji procesu przetwarzania oraz zarządzania pamięcią wykorzystane zostaną języki programowania wysokiego poziomu, takie jak C++ lub Python, wraz z odpowiednimi bibliotekami do przetwarzania obrazów.

Poprzez wykorzystanie równoległego przetwarzania zarówno na GPU, proponowane rozwiązanie umożliwi efektywne przetwarzanie danych wejściowych o dużych rozmiarach w celu detekcji cieni

na obrazach liści. Umożliwi to osiągnięcie szybszych czasów przetwarzania oraz lepszej wydajności systemu.

Podział zadań w projekcie jest następujący:

Zadanie	Szczegóły zadania	Osoba odpowiedzialna
Uzupełnienie pierwszej wersji raportu	• Uzupełnienie strony tytułowej, • Uzupełnienie wymaganych rozdziałów (1,2,3,4,8,12)	K.K.
Testy sekwencyjnej wersji algorytmu	• Przeprowadzenie testów działania algorytmu z wykorzystaniem różnych przestani barw	M.K.
Implementacja konwersji na różne przestrzenie barw	• Implementacja równoległej konwersji na inne przestrzenie barwne	
Rozwijanie implementacji algorytmu w OpenCL	• Przeprowadzenie testów optymalizacji i zrównoleglenia algorytmu w OpenCL, implementacja operacji morfologicznych	J.B.

5. Symulacja i Testowanie

Modelowanie i symulacja

W tej części opisana jest metodologia weryfikacji poprawności implementacji przedstawionej koncepcji rozwiązania na poszczególnych etapach realizacji (symulacja funkcjonalna, *post synthesis*, *post map, post place&route*). Czytelnik może się dowiedzieć z tego paragrafu, jakie kryteria oceny jakościowej i ilościowej zostały zdefiniowane przez projektanta aby ocenić poprawność działania realizowanej aplikacji i jej zgodność ze specyfikacją. Opisana jest ogólna idea symulacji projektu całościowego lub komponentów składowych. Określone są zarówno narzędzia symulacyjne, jak i formaty danych wektorów testowych. Szczególną uwagę należy poświęcić doborowi wektorów testowych i referencyjnych aby uwzględnić zarówno typowe konfiguracje przetwarzanych danych jak i przypadki szczególne (np. przepełnienie liczników, akumulatorów, przekroczenie zakresów, itd. ....) dla realizowanego algorytmu. Przedstawiony jest sposób uzyskania wektorów referencyjnych np. poprzez utworzenie programowego modelu referencyjnego (ang. *Golden Reference*). Szczegóły procedury symulacji przedstawione są w rozdziale 10 ale tutaj pokazana jest ogólna idea konfrontacji wektorów wynikowych z wektorami referencyjnymi oraz sposób wyliczania wskaźnika jakości.

Testowanie o weryfikacja

Ten akapit przedstawia w jaki sposób, zaimplementowany na platformie docelowej, algorytm będzie uruchamiany i testowany. Opisane są mechanizmy komunikacji: sterowania i transmisji danych pomiędzy komputerem nadrzędnym a platformą docelową. Krótko opisane jest środowisko uruchamiania aplikacji sprzętowej oraz dodatkowe narzędzia programowe i sprzętowe, jakie zostały na tę potrzebę zaprojektowane i wykonane. Podrozdział stanowi niejako kontynuację poprzedniego i tutaj czytelnik również znajduje definicję wskaźnika jakości działającej aplikacji (ang. *run time simulation*) i sposób jego wyznaczania. Oprócz analizy funkcjonalnej, bieżący rozdział zawiera również opis eksperymentów praktycznych których celem jest wyznaczenie wydajności systemu – np. ilości danych przetwarzanych w jednostce czasu.

6. Rezultaty i wnioski

Na załączonym nośniku, oprócz niniejszego sprawozdania musi znaleźć się katalog z wektorami testowymi, referencyjnymi wynikowymi dla poszczególnych wersji implementacji projektu. Dane te stanowią podstawę do analiz i postawienia wniosków w tym rozdziale. Oprócz symulacji i weryfikacji należy przeprowadzić analizę zrealizowanego rozwiązania pod kątem wydajności (szybkość przetwarzania danych) i zużycia zasobów (obciążenie procesora, zużycie zasobów pamięciowych i obliczeniowych FPGA), maksymalnej dopuszczalnej częstotliwości pracy.

W przypadku niespełnienia wymagań jakościowych lub ilościowych określonych w specyfikacji należy wskazać potencjalne przyczyny i zaproponować dodatkowe procedury testowe i korekty w implementacji. Jeśli wybrana metoda realizacji zadania okazała się nieskuteczna to taką informację należy również podać – pozwala to na wybór innej drogi postępowania podczas kontynuacji projektu przez inne grupy lub prowadzącego.

7. Podsumowanie

Rozdział ten zawiera przedstawione w skrócie najważniejsze osiągnięcia i wnioski z całej pracy oraz opisuje jak odpowiadające wyniki są powiązane z oczekiwaniami i badaniami literaturowymi cytowanymi we wstępie. Opisany jest również stopień realizacji projektu (rozwiązania problemu): Na jakim etapie zakończono prace, jakie funkcjonalności udało się zrealizować a jakie nie? Na zakończenie konieczne jest krytyczne przyjrzenie się swojej pracy i zaproponowanie kierunków kolejnych modyfikacji lub wskazanie innych metod, możliwych do zastosowania w przyszłości.

8. Literatura

[1] Salvador E, Andrea C, Ebrahimi T, *Cast shadow segmentation using invariant color features*, 2004.

[2] Bradski G. R, *Real Time Face and Object Tracking as a Component of a Perceptual User Interface*, 1998.

[3] Benedek C, Sziranyi T, *Study on Color Space Selection for Detecting Cast Shadows in Video Surveillance*, 2007.

[4] OpenCV: Meanshift and Camshift, https://docs.opencv.org/4.x/d7/d00/tutorial_meanshift.html, ostatni dostęp do źródła: 13-05-2024.

[5] Huang, F., Chen, Y., Li, L., Zhou, J., Tao, J., Tan, X., & Fan, G. Implementation of the parallel mean shift-based image segmentation algorithm on a GPU cluster, 2019.

9. DODATEK A: Szczegółowy opis zadania

Specyfikacja projektu

Rozdział ten zawiera opis zadania projektowego postawiony przez osobę prowadzącą projekt.

Szczegółowy opis realizowanych algorytmów przetwarzania danych.

W kolejnych podrozdziałach opisane są aspekty algorytmów uwzględniające istotne dla technologii implementacji i docelowej platformy realizacji (program/sprzęt, język implementacji) i sposób przetwarzania (sekwencyjnie, współbieżnie, strumieniowo, potokowo). Opisy algorytmów przetwarzania danych i komunikacji wytypowanych do implementacji wzbogacone są pseudokodem, wykresami czasowymi lub graficznym językiem opisu algorytmów (ang. *flowchart*). Istotne parametry implementowanych algorytmów przedstawione są w postaci wzorów, macierzy współczynników filtrów. Zamieszczone w tekście odnośniki wskazują na literaturę źródłową szczegółowo opisującą analizowane algorytmy. Jeśli w czasie realizacji zadania, okazało się, że z racji na ograniczenia platformy lub zastosowanych narzędzi konieczne było wprowadzenie modyfikacji, uproszczeń lub aproksymacji do oryginalnego algorytmu, informacja ta wraz z uzasadnieniem jest umieszczona w tym miejscu.

10. DODATEK B: Dokumentacja techniczna

Rozdział ten ma charakter ściśle techniczny. Rozpoczynają go informacje o środowisku programowania, ew. modularyzacji i opcjach kompilacji, plikach, które muszą być dołączone oraz użytych „obcych" bibliotekach. W rozdziale należ zamieścić informacje o wszystkich procedurach programowych i modułach sprzętowych które zostały zrealizowane lub zmodyfikowane dla potrzeb realizacji projektu. W dokumentacji technicznej nie zamieszczamy kodu źródłowego lecz jedynie deklaracje funkcji, encji (w przypadku kodu VHDL), opisy modułów funkcjonalnych. Jeśli skorzystano z gotowych rozwiązań należy odwołać się do istniejącej dokumentacji technicznej jak do pozycji literaturowej, po uprzednim umieszczeniu wpisu w bibliografii.

Dokumentacja oprogramowania

Procedury programowe (funkcje, klasy, zmienne globalne) implementowane w języku C, C++, SystemC, w pliku źródłowym i raporcie opisywane są zgodnie ze składnią pakietu Doxygen - patrz [4]. Dotyczy to również języka kodu procesora Microblaze, oraz modułów HandelC.

Opis funkcji obejmuje:

- autor, data utworzenia i ostatniej zmiany,

- przeznaczenie,
- funkcjonalny opis działania, ograniczenia,
- argumenty wywołania,
- wartość zwracaną,
- wykorzystywane zmienne globalne,
- wywoływane procedury i ew. biblioteki z których procedury te są importowane.

Dokumentacja sprzętowa

Zaprojektowane i zrealizowane funkcjonalne moduły sprzętowe opisane są w następujący sposób:

- autor, data utworzenia i ostatniej zmiany,
- przeznaczenie,
- funkcjonalny opis działania, ograniczenia,
- sygnały wejściowe z opisem,
- sygnały wyjściowe(i dwukierunkowe) z opisem,
- wykorzystywane komponenty i biblioteki z których komponenty te pochodzą.

Jeśli opisywany komponent posiada reprezentację graficzną należy ją tu zamieścić. Rozdział ten zawiera również schemat blokowy „top level” lub RTL aplikacji sprzętowej i ewentualnie schemat blokowy systemu mikroprocesorowego Microblaze pobrany ze środowiska PDK.

Procedura symulacji i testowania:

Wyliczone są programy narzędziowe i biblioteki wraz z numerami wersji które umożliwiają uruchomienie symulacji i weryfikację programową zrealizowanego projektu. Opisane są czynności jakie należy wykonać aby uruchomić symulację projektu w oparciu o dołączone na nośniku wektory testowe oraz interpretację wyników symulacji na podstawie załączone na nośniku wektory referencyjne.

Procedura uruchomieniowa i weryfikacji sprzętowej:

Wyliczone są programy narzędziowe, biblioteki i karty sprzętowe z numerami wersji które umożliwiają uruchomienie i weryfikację sprzętową zrealizowanego projektu sprzętowego. Opisane są czynności jakie należy wykonać aby uruchomić aplikację sprzętową i przeprowadzić procedurę sprzętowej weryfikacji.

11. DODATEK D: Spis zawartości dołączonego nośnika (płyta CD ROM)

W poszczególnych folderach nośnika, w zależności od rodzaju projektu, znajdują się:

- **SRC** - Kompletne foldery robocze z plikami źródłowymi programów i aplikacji sprzętowych zorganizowane w strukturę właściwą dla pakietów projektowych (np. ISE, EDK, VisualStudio, ModelSim, HandelC –DK4, SystemC itp.), oraz ewentualne pliki *makefile* skrypty symulacyjne, instalacyjne itp., kluczowe wersje projektów źródłowych umieszczone są w oddzielnych katalogach,
- **EXE** - postać programu gotową do uruchomienia wraz z ew. plikami konfiguracyjnymi lub innymi niezbędnymi komponentami (EXE, ELF, BIT, LIB, DLL),
- **BIT** - pliki konfiguracyjne FPGA odpowiednio opisane, jeśli występują w różnych wersjach,
- **TEST** - wejściowe wektory testowe (np. obrazy) oraz wektory referencyjne służące do testowania, symulacji i weryfikacji aplikacji oraz wyniki kluczowych symulacji, *testbench'*e oraz skrypty – jeśli nie zostały zawarte w folderach pakietu projektowego.
- **DOC** - aktualna wersja niniejszego raportu w postaci elektronicznej (MS WORD lub PDF) i ewentualnie dodatkowa dokumentacja elektroniczna pobrana z sieci WWW.
- **INST** - jeśli wymagane, dodatkowe pakiety oprogramowania wymagane do symulacji, weryfikacji, testowania i uruchomienia aplikacji.
- **ADDON** – pakiety instalacyjne nietypowego oprogramowania niezbędnego do zbudowania plików wynikowych, uruchomienia lub symulacji.

Struktura folderów nośnika powinna być dokładnie opisana w tym dodatku:

- Co znajduje się w poszczególnych pod-folderach? ze szczegółowym opisem kolejnych kluczowych wersji plików,
- opisy foldery robocze projektów źródłowych, zarówno sprzętowych jak programowych zawierają nazwę i numer wersji pakietu rozwojowego którym można otworzyć projekt np.: Visual Studio, Vivado, inne … .

Z racji na ewentualne poprawki w projekcie nie należy zamykać sesji na nośniku CD.

12. DODATEK E: Historia zmian

Tabela 1 Historia zmian.

Autor	Data	Opis zmian	Wersja
K.K.	2024-05-12	Utworzono dokument, uzupełniono stronę tytułową, dodano rozdziały 1,2,3.	1.1
K.K.	2024-05-13	Dodano rozdziały 4,8,12.	1.2

pr24aaw05			**Karta oceny projektu**
Data	Ocena	Podpis	Uwagi