Wiesław PAMUŁA
POLITECHNIKA ŚLĄSKA, WYDZIAŁ TRANSPORTU, KATEDRA SYSTEMÓW INFORMATYCZNYCH TRANSPORTU
Krasińskiego 13, 40-019 Katowice

Metoda dekompozycji algorytmów przetwarzania obrazów dla implementacji w układach FPGA

Dr inż. Wiesław PAMUŁA

Dr nauk technicznych (1998). Adiunkt w Katedrze Systemów Informatycznych Transportu na Wydziale Transportu Politechniki Śląskiej. Prowadzone prace badawcze dotyczą zastosowania przetwarzania obrazów do określania parametrów ruchu drogowego. Bierze udział w opracowaniu i wdrożeniu wideo detektorów pojazdów dla sterowników sygnalizacji świetlnej.

e-mail: wieslaw.pamula@polsl.pl

Streszczenie

Artykuł prezentuje metodę dekompozycji algorytmów przetwarzania obrazów na potok przetwarzania zrealizowany z użyciem sparametryzowanych modułów. Moduły realizują niskopoziomowe operacje na pikselach obrazu oraz śledzenie zmian w wyliczonym opisie klatki obrazu. Uznano taki zakres operacji za wystarczający dla opracowania wideo detektorów obiektów. Wykorzystywany jest szeregowy strumień wideo z kamery. Implementacje dowodzą skuteczności zastosowania metody. Uzyskano prędkość przetwarzania przewyższającą wymagania pracy w czasie rzeczywistym oraz znaczną zdolność do modyfikacji własności rozwiązań.

Słowa kluczowe: FPGA, potok przetwarzania, szeregowy strumień wideo, niskopoziomowe operacje pikselowe.

Method of decomposing image processing algorithms for implementation in FPGA

Abstract

Efficient decomposition of image processing algorithms is of paramount importance in designing FPGA based video detectors of objects for use, for instance, in surveillance systems or in road traffic control applications. Efficiency appraisal is done taking into account resource utilisation, capability of introducing new processing features and components costs. Real time processing adds additional constraints on this task. Available development tools do not facilitate the design process. The paper presents a method for decomposing the image processing algorithm into an efficient processing pipeline of parameterised components. The components perform low level image processing tasks and content tracking operations. Such a set of processing operations is adequate for designing video detectors of objects. Components for carrying out feature calculations using convolutions, morphology operators and corner detectors are presented. Their architecture is optimised for serial video streams, which provide the image contents using horizontal scanning. FPGA resource requirements are estimated for devices of leading manufacturers. The estimated processing speed exceeds the requirements of real time operation. Special attention is directed to pipelining calculations, streamlining multi operand computations, fast determination of minimum, median and maximum of values. An implementation of a video object detector, using a low cost FPGA, is presented proving the feasibility of this approach.

Keywords: FPGA, pipeline processing, serial video stream, low level image processing.

1. Wprowadzenie

Układy FPGA z powodzeniem wykorzystywane są do przetwarzania strumienia wideo w czasie rzeczywistym. Przede wszystkim znajdują zastosowanie w implementacji niskopoziomowych operacji na pikselach obrazów [1].

Znane rozwiązania wykorzystują potoki przetwarzania składające się z wielu kart z układami FPGA nadzorowanymi przez procesory [2]. Procesory organizują wymianę danych oraz dynamicznie rekonfigurują układy śledząc tok algorytmu przetwarzania [3, 4]. Rozwiązania można określić jako hybrydowe, ponieważ implementacja algorytmu przetwarzania dzielona jest na zadania realizowane sprzętowo i programowo. Sprawne wykorzystanie zasobów logicznych stanowi trudny element opracowania rozwiązania. Osiągane jest najczęściej przez wykorzystanie narzędzi HDL. Podejmowane były próby wykorzystania narzędzi wyższego poziomu takich jak Handel-C lub SystemC [5, 6].

Postęp w technologii scalania pozwala zaproponować implementację złożonych algorytmów przetwarzania w pojedynczych układach FPGA. Złożoność układów wiodących producentów takich jak Xilinx, układy serii 7, lub Altera, układy Stratix V, osiąga poziom milionów bramek logicznych [7, 8]. Specyfika organizacji wewnętrznej układów uniemożliwia jednak prostą konwersję algorytmów, opartych przede wszystkim na masowym korzystaniu z pamięci, w struktury logiczne.

Układy FPGA podzielone są na konfigurowalne bloki logiczne CLB zawierające kilka elementów LUT, przerzutników oraz łącznice pozwalające ustawić wewnętrzne i zewnętrzne połączenia. Układy posiadają niewielkie zasoby pamięciowe zorganizowane w wydzielone podzespoły pamięciowe o rozmiarach kilku kB. Układy konfigurowane z użyciem SRAM pozwalają dodatkowo na wykorzystanie pamięci konfiguracji jako pamięci rozproszonej. Zasoby pamięci wewnątrz FPGA nie pozwalają jednak zapamiętać pełnych klatek przetwarzanych obrazów.

Optymalne wykorzystanie zasobów FPGA wymaga opracowania systematycznego podejścia do dekompozycji algorytmu przetwarzania wykorzystującego cechy architektury układów. Dodatkowym utrudnieniem może być konieczność korzystania z firmowych generatorów konfiguracji. Proces może być wspomagany przez biblioteki podzespołów dostarczane przez producentów w wielu przypadkach jest to jednak bardzo kosztowne rozwiązanie.

Proponowana metoda dekompozycji algorytmów opiera się na konwersji zadania do potoku przetwarzania zrealizowanego z użyciem sparametryzowanych modułów przetwarzających [9]. Moduły działają na szeregowej reprezentacji klatki obrazu. Realizują niskopoziomowe operacje na pikselach obrazu oraz śledzenie zmian w wyliczonym opisie zawartości klatki. Uznano taki zakres operacji za wystarczający dla klasy algorytmów przetwarzania stosowanych w opracowaniu wideo detektorów obiektów [10].

Nie wyklucza to zastosowania metody np. do zadań wyznaczania trajektorii ruchu obiektów.

Dostępne na rynku opracowania wideo detektorów nie korzystają z przetwarzania z użyciem układów FPGA [18].

2. Dekompozycja

Założenie wykorzystania szeregowej postaci obrazów przyjęto dla zachowania zgodności z praktyką stosowania typowych kamer CCTV do monitorowania ruchu drogowego. Uwzględniając znaczny rozrzut parametrów takich kamer jako układ standaryzujący strumień zastosowano procesor wideo. Procesor przetwarza sygnał z kamery na strumień danych cyfrowych zgodny z standardem ITU-R BT 656.

Obraz O z czujnika kamery składający się z m wierszy i n kolumn:

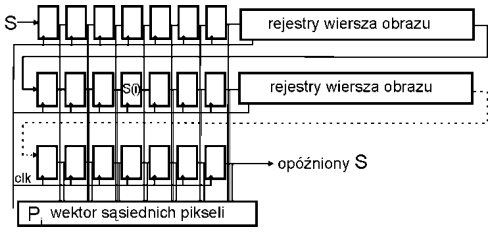
$$O(i,j) = \{i,j : i = 1,...,n,\ j = 1,...,m\} \tag{1}$$

zostaje zamieniony na strumień S o elementach:

$$S(i) = O((i \bmod n),(i \operatorname{div} m)) \quad i = 1,...,mn \tag{2}$$

który wprowadzany jest do układu FPGA.

Operacje niskopoziomowe wymagają dla obliczeń dostępu do zbioru sąsiednich pikseli [11]. Wykorzystując rejestry opóźniające uzyskuje się podstawowy moduł (rys. 1), na którym opiera się potok przetwarzania.



Rys. 1. Moduł sąsiedztwa
Fig. 1. Neighbourhood module

Wynikiem działania modułu jest wektor wartości pikseli P_i z sąsiedztwa przetwarzanego piksela *S(i)* obrazu. Moduł wprowadza opóźnienie *D* w przetwarzaniu, maksymalna wartość opóźnienia wynosi:

$$D_{\max} = (wn + s)/f \tag{3}$$

gdzie: *w,s* – wysokość, szerokość okna otoczenia, *f* – częstotliwość taktowania strumienia danych.

Decydujący wpływ na opóźnienie ma długość wiersza obrazu. Dla okien o wysokości 3 do 7, moduł wprowadza opóźnienie od około 0,2 do 0,5ms.

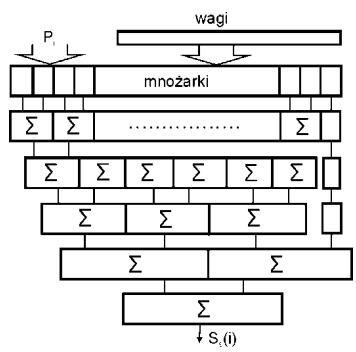
Uzupełniając moduł o struktury realizujące procedury obliczeniowe, uzyskuje się moduły przetwarzania stanowiące elementy konstrukcyjne potoku przetwarzania. Wyróżnić należy moduły wyliczania splotów, operacji morfologicznych, detektorów cech punktowych (narożników).

Śledzenie zmian w opisie zawartości klatki obrazu wykorzystuje liczniki zmian i nie wymaga włączania do potoku przetwarzania.

Moduły wyliczania splotów. Wykorzystywane są przede wszystkim do przeprowadzania filtracji obrazu. Operacja wyliczania splotu dyskretnego polega na zsumowaniu ważonych wartości pikseli sąsiedztwa:

$$S_s(i) = w * P_i = \sum_{j=1}^{ws} w(j) P_i(j) \tag{4}$$

Wektor wag *w* definiuje własności filtru. Istotnym zagadnieniem z punktu widzenia złożoności struktury rozwiązania modułu są układy mnożenia oraz normalizacji wyniku. W przypadku wag ułamkowych wprowadza się skalowanie i zwiększa długość zapisu wartości. Unika się korzystania z wbudowanych układów mnożących stosując tablicowanie lub aproksymację z użyciem dodawań i przesunięć arytmetycznych.



Rys. 2. Moduł wyliczania splotu – filtr Gaussa 5x5
Fig. 2. Convolution module – Gaussian filter 5x5

Sumowanie wykonuje się parami dla uniknięcia dużych opóźnień. Liczba poziomów sumowania jest nie mniejsza od $log_2 ws$. Zatem opóźnienie wprowadzane przez strukturę wyliczania będzie wynosiło kilka taktów zegara co w porównaniu do D_{max} jest niewiele znaczącą wielkością.

Rysunek 2 ilustruje rozwiązanie struktury filtru Gaussa na oknie 5x5 pikseli. Zawiera 25 bajtowy rejestr wag, blok mnożenia i pięciopoziomowy sumator.

Operacje morfologiczne. Operacje morfologiczne należą do najchętniej używanych w przetwarzaniu obrazów ze względu na wysoką skuteczność [12]. Erozje, dylacje zwiększają widoczność obiektów i ułatwiają zadania segmentacji obrazu. Złożone operacje szczególnie HMT pozwalają na wyodrębnianie cech obiektów. Operacje morfologiczne dla obrazów w skali szarości wymagają wyznaczania ekstremów wartości w zbiorach pikseli pokrywanych przez elementy strukturyzujące. Może to stanowić pewną trudność implementacyjną.

Moduł sąsiedztwa dostarcza wektor pikseli pokrywający się z kształtem elementu strukturyzującego SE. Erozja wyliczana jest:

$$S_e(i) \ominus SE = \min_{j \in SE}(S(i) + SE(j)) \tag{5}$$

Dylacja:

$$S_d(i) \oplus SE = \max_{j \in SE}(S(i) + SE(j)) \tag{6}$$

Ekstrema wyznaczane są z użyciem sieci sortującej [13]. Zastosowano sieć sortującą parzyście-nieparzyście ze scalaniem, która wymaga minimalnej liczby elementów porównujących [14]. Sieć została uzupełniona o rejestry buforujące dane na każdym poziomie sortowania, dla realizacji przetwarzania potokowego. Sygnał synchronizujący przepisuje dane między poziomami. Piksele nie biorące udziału w danym poziomie sortowania przenoszone są bez zmian aby zachować jednakowe opóźnienia danych w potoku przetwarzania. Sieć sortująca zawiera $k(k+1)/2$ poziomów i $(k^2 - k+4)2^{p-2} - 1$ komparatorów, gdzie $k \geq log_2 ws$.

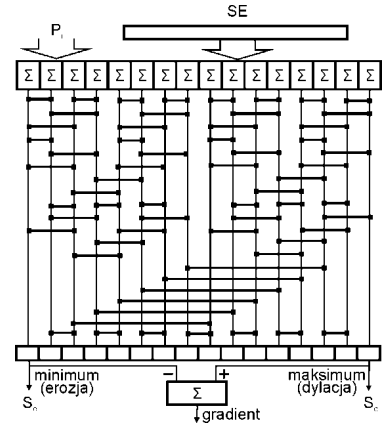
Stosowanie dużych elementów strukturyzujących np. podczas dopasowania wzorców korzystnie jest rozłożyć na operacje z użyciem składowych elementów. Wynik uzyskuje się przez szeregowe złożenie działania modułów przetwarzających. Dla złożonego elementu strukturyzującego SE:

$$SE = SE_1 \oplus ... \oplus SE_n \tag{7}$$

dylacja wyznaczana jest:

$$S_d(i) \oplus SE = (S(i) \oplus SE_1) \oplus ...) \oplus SE_n \tag{8}$$

Całkowite opóźnienie wyznaczone jest praktycznie przez wysokość SE.



Rys. 3. Moduł operacji morfologicznych
Fig. 3. Morphology operations assembly

Wartość środkowa wektora uporządkowanych pikseli sąsiedztwa może być traktowana jako wynik działania filtru medianowego bardzo przydatnego do czyszczenia obrazów. Wyznaczenie gradientu morfologicznego można wykonać z użyciem jednego modułu ponieważ może wyznaczać jednocześnie erozję i dylację.

Rys. 3 ilustruje moduł operacji morfologicznych uzupełniony o obliczanie gradientu. Wykorzystuje 16 pikselowy płaski kołowy SE, składa się z sumatorów i 10 poziomowej sieci sortującej.

Detektory cech punktowych. Cechy punktowe wyznacza się analizując zmiany wartości pikseli lub wyliczając statystyki wartości w kołowym obszarze wokół danego piksela. Przekroczenie zadanego progu detekcji wskazuje na wystąpienie cechy punktowej. Ważnymi w zastosowaniach są detektory „narożników" Harrisa, SUSAN, FAST [15,16].

Każdy z detektorów ma swoją specyficzną strukturę odwzorowującą algorytm obliczania. Detektor Harrisa wylicza cechy rozkładu gradientów w sąsiedztwie danego piksela. Na podstawie współczynników macierzy autokorelacji wartości pikseli M_i

$$M_i = w_\sigma * \begin{bmatrix} A & C \\ C & B \end{bmatrix} \qquad (9)$$
$$A = (w_{x\sigma} * P_i)^2 \quad B = (w_{y\sigma} * P_i)^2 \quad C = (w_{x\sigma} * P_i)(w_{y\sigma} * P_i)$$

gdzie: w_σ, $w_{x\sigma}$,$w_{y\sigma}$,- wektory wag odpowiednio: dla wyliczania gausowskiego filtru wygładzającego, gradientu w kierunku poziomym, gradientu w kierunku pionowym,

określona zostaje miara obecności „narożników":

$$R_H = \det(M_i) - k \cdot \text{trace}(M_i)^2$$
$$R_H = (M_{i11}M_{i22} - M_{i12}M_{i21}) - k \cdot (M_{i11} + M_{i22})^2 \qquad (10)$$

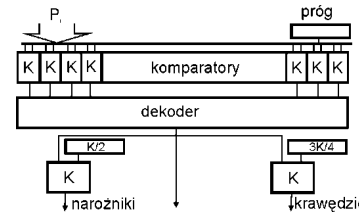
Moduł będzie zawierał dwa równolegle działające moduły wyliczania splotów ($w_{x\sigma}$,$w_{y\sigma}$) wyniki z modułów po wymnożeniu podane zostaną do kolejnego modułu wyliczania splotu (w_σ). Moduł wyznacza elementy macierzy M_i, uzupełniony o strukturę wyliczania R_H będzie generował strumień miar obecności „narożników".

SUSAN – pozwala sklasyfikować zawartość kołowego otoczenia piksela. Używa się aproksymacji koła o średnicy do kilkunastu pikseli najczęściej 3, 7, 11. Wartości pikseli w sąsiedztwie porównuje się z wartością piksela centralnego zliczając różniące się. Różnica liczby pikseli sąsiedztwa i różniących się określa R_S - miarę obecności narożników.

$$e(j) = \begin{cases} 1 & gdy \quad |P_i(j) - P_i(c)| \le próg \\ 0 & gdy \quad |P_i(j) - P_i(c)| > próg \end{cases}$$
$$E_i = \sum_{j=1}^{K} e(j) \qquad (11)$$
$$R_S = \begin{cases} L - E_i & gdy \quad E_i < L \\ 0 \end{cases}$$

Wartość progu określa zdolność do eliminacji szumów i czułość współczynnika. Dla $L=K/2$ współczynnik wskazuje na występowanie „narożników", dla $L=3K/4$ krawędzi, gdzie: K - liczba pikseli w sąsiedztwie - w kole - otaczającym piksel centralny.

Rys. 4 przedstawia strukturę detektora cech punktowych SUSAN. Składa się z komparatorów i dekodera liczby wykrytych przekroczeń wartości progów. Konstrukcje zachowują potokowy charakter przetwarzania dostarczając strumień wyliczonych wartości, który może zostać zachowany w pamięci zewnętrznej lub poddany dalszej obróbce.



Rys. 4. Detektor cech punktowych SUSAN
Fig. 4. Interest points detector SUSAN

Rozwiązania przygotowuje się w postaci podzespołów w języku VHDL lub Verilog. Dla zmniejszenia problemów z implementacją dodatkowo należy przygotować w środowisku danej rodziny układów FPGA pliki lokalizujące struktury tj ograniczające pole rozmieszczenia zasobów logicznych modułu w układzie FPGA.

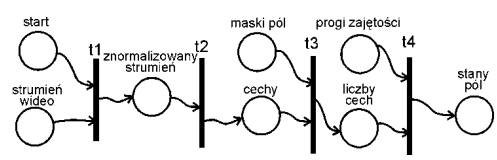
Nowe wersje środowisk konfiguracji układów FPGA udostępniają narzędzia do częściowej rekonfiguracji. Częściowa rekonfiguracja umożliwia modyfikację parametrów wyznaczania połączeń w skali układu FPGA co pozwala zachować połączenia wewnątrz opracowanych modułów. Optymalną strukturę połączeń wewnątrz modułów uzyskuje się zwykle w toku żmudnego strojenia działania [17].

Potok przetwarzania. W celu efektywnego wykorzystania dostępnych zasobów FPGA przyjęto koncepcję przetwarzania w postaci potoku. Etapy przetwarzania treści klatki obrazu identyfikowane są przez rodzaj operacji i wymagane struktury danych we/wy. Wygodnym narzędziem reprezentacji algorytmu przetwarzania jest sieć Petriego. Tranzycje odwzorowywane są przez moduły wyliczania a moduły sąsiedztwa pełnią rolę układów przygotowania struktur danych.

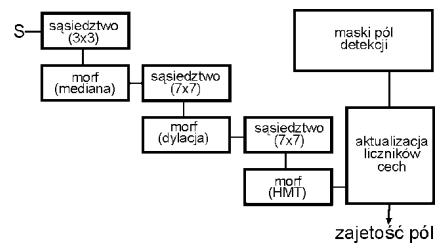
Dla przygotowanego algorytmu przetwarzania przedstawionego np. w postaci sieci Petriego rys. 5 konstruuje się potok przetwarzania rys. 6. Potok łączy szeregowo moduły przetwarzania odwzorowując przebieg algorytmu.

Sieć przedstawia rdzeń algorytmu detekcji obiektów. Detekcja polega na określeniu przekroczeń progów zajętości, zadanych z użyciem masek, pól na obrazie. Miarą zajętości jest liczba cech obiektów. Jako cechę wybrano odpowiedź filtru morfologicznego.

Miejsca w sieci reprezentują struktury danych biorące udział w przetwarzaniu natomiast tranzycie operacje realizowane przez moduły. Wyróżnienie wyłącznie struktur i operacji umożliwia klarowne zdefiniowanie wymagań funkcjonalnych dla modułów przetwarzania. Rozmiary danych wyznaczają parametry działania, rodzaj operacji określa element konstrukcyjny potoku przetwarzania.



Rys. 5. Sieć Petriego algorytmu detekcji obiektów
Fig. 5. Petri net of the object detection algorithm



Rys. 6. Przekonwertowany potok
Fig. 6. Converted processing pipeline

Tranzycja t1 „czyści" strumień z użyciem filtra medianowego o rozmiarze okna 3x3 i sprawdza zakresy wartości pikseli aby uniknąć przetwarzania uszkodzonych klatek. Stąd uzyskuje się znormalizowany strumień. Na rysunku potoku operacja ta jest pominięta – wykonywana jest równolegle z filtrowaniem.

Kolejna tranzycja t2 reprezentuje wyznaczanie cech obiektów. Realizowana jest z użyciem dwóch modułów morfologicznych składających się na operację HMT. Uzyskany strumień cech skorelowany z strumieniem masek pół odczytywanym z pamięci podręcznej poddany jest zliczaniu w t3. Zawartości zaktualizowanych liczników cech porównywane są z progami i wyznaczane są stany pól w t4.

3. Wykorzystaniem zasobów układów FPGA

Opracowane rozwiązania zaimplementowano w języku VHDL w środowiskach projektowych wiodących producentów układów FPGA Xilinxa i Altery. Zastosowano zaawansowane i tanie układy aby porównać użycie zasobów. Tabela 1 zestawia uzyskane wyniki. Zwraca uwagę wykorzystanie pamięci blokowej do realizacji rejestrów opóźniających w układach Altery. Różnice w liczbie wymaganych podzespołów logicznych są pochodną złożoności tych elementów. Virtex posiada 6 argumentowe układy LUT, Spartan3E 4 argumentowe.

Prędkość działania ze względu na przyjęte potokowe działanie oraz optymalizację wyliczania sum, ekstremów zależy wyłącznie od ograniczeń narzucanych przez technologię wykonania układu. Wynosi od ponad 150 do 480MHz dla najszybszych układów. Oznacza to, że przeliczenie jednej klatki można wykonać w czasie poniżej 1 ms. Rzeczywisty układ taktowany jest zegarem 27MHz. Przygotowany prototyp wideo detektora pojazdów wykorzystuje detektor cech oparty na HMT i pracuje w czasie rzeczywistym z typową kamerą CCTV [18].

Tab. 1. Zasoby FPGA dla realizacji modułów
Tab. 1. FPGA resources for module implementations

	Rodzina układów FPGA			
	dużej wydajności		„low cost"	
moduł	Virtex6	StratixIII	Spartan3E	Cyclone
sąsiedztwa	379 slice 1216 LUT	515 LE 161 ALUT 34kb bl. mem	1274 slice 2168 LUT	517 LE 61 ALUT 34kb bl. mem
splotu	53 slice 195 LUT	608 LE 435 ALUT	308 slice 514 LUT	682 LE 435 ALUT
morfologiczny	1748 slice 5495 LUT	4108 LE 2925 ALUT	3017 slice 3855 LUT	2493 LE 2221 ALUT

Przeprowadzone implementacje wskazują na dużą elastyczność zaproponowanej metody i znaczną poprawę efektywności opracowywania modyfikacji rozwiązań sprzętowych układów przetwarzania obrazów.

4. Wnioski

Opracowana metoda dekompozycji zadania przetwarzania, oparta na potoku wykorzystującym specjalizowane moduły, pozwala sprawnie przygotować rozwiązanie sprzętowe. Kilkukrotne wykorzystanie modułów sąsiedztwa dla przetworzenia treści klatki obrazu, opóźnia wyznaczenie wyniku pozwalając jednak na zachowanie działania w czasie rzeczywistym.

Integracja z środowiskiem projektowym danej rodziny układów FPGA poprzez włączenie modułów jako elementów bibliotecznych pozwala oszczędzić czas udoskonalania opracowywanych konstrukcji.

Przewiduje się opracowanie dalszych modułów w szczególności specjalizowanych do realizacji operacji śledzenia obiektów.

5. Literatura

[1] Benitez D.: Performance of Recongurable Architectures for Image-Procesing Applications, Journal of Systems Architecture 49, s. 193-210, 2003.

[2] Wiatr K.: Akceleracja obliczeń w systemach wizyjnych,WNT Warszawa 2003.

[3] Porter R., Frigo J., Conti A., Harvey N., Kenyon G., Gokhale M.: A Reconfigurable Computing Framework for Multiscale Cellular Image Processing, Microprocessors and Microsystems 31, s. 546-563, 2007.

[4] Kessal L., Abel N., Karabernou S.M., Demigny D.: Recongurable Computing: Design Methodology and Hardware Tasks Scheduling for Real-Time Image Processing, J Real-Time Image Proc 3, s. 131-147, 2008.

[5] Muthukumar V., Rao D.V.: Image Processing Algorithms on Recongurable Architecture Using HandelC, Proceedings of the 7th Euromicro Conference on Digital Systems Design, IEEE Computer Society, s. 362-370, 2004.

[6] Beun R., Karkowski I., Ditzel M.: C++ Based Design Flow for Recongurable Image Processing Systems International Conference on Field Programmable Logic and Applications, IEEE Computer Society, s. 571-575, 2007.

[7] Virtex-6, Spartan-6 Family Overview. Xilinx Inc. San Jose, CA USA, 2010.

[8] Altera Product Catalog. Altera Co. San Jose, CA USA, 2009.

[9] Pamuła W.: Vehicle Detection Algorithm for FPGA Based Implementation, Computer Recognition Systems Eds. M. Kurzyński, M. Woźniak, Springer Verlag, Berlin, s. 586-592, 2009.

[10] Pamuła W.: Object Classification Methods for Aplication in FPGA Based Vehicle Video Detector, Transport Problems, Wyd. Pol. Śląskiej s. 5-14, 2009.

[11] Porter R., Frigo J., Gokhale M., Wolinski C., Charot F., Wagner C.: A Run-Time Recongurable Parametric Architecture for Local Neighbourhood Image Processing, Proceedings of the 9th Euromicro Conference on Digital Systems Design, IEEE Computer Society, s. 362-370, 2006.

[12] Baumann D., Tinembart J.: Designing Mathematical Morphology Algorithms on FPGA: An Application to Image Processing LNCS 3691, s. 562-569, 2005.

[13] Batcher K. E.: Sorting Networks and Their Applications, Spring Joint Computer Conf., AFIPS Proc. 32, s. 307-314, 1968.

[14] Liszka K. J., Batcher K. E.: A Generalized Bitonic Sorting Network, Proceedings of the International Conference on Parallel Processing, s. 105-108, 1993.

[15] Torres-Huitzil C., Arias-Estrada M.: An FPGA Architecture for High Speed Edge and Corner Detection, Proceedings of the 5th IEEE International Workshop on Computer Architectures for Machine Perception, IEEE Computer Society, s. 112-116, 2000.

[16] Claus C., Huitl R., Rausch J., Stechele W.: Optimizing The SUSAN Corner Detection Algorithm for a High Speed FPGA Implementation, International Conference on Field Programmable Logic and Applications, FPL 2009. IEEE Computer Society, s. 138-145, 2009.

[17] ISE 13.1. Xilinx Inc. San Jose, CA USA, 2011.

[18] Raporty projektu: Moduły wideodetektorów ZIR WD dla sterowania i monitorowania ruchu drogowego. WKP-1/1.4.1/1/2005/14/14/231/2005, vol. 1-6 Katowice, 2005-2007.

otrzymano / received: 03.11.2010
przyjęto do druku / accepted: 04.05.2011 *artykuł recenzowany*