Miron KŁOSOWSKI, Bogdan PANKIEWICZ, Marek WÓJCIKOWSKI

Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

Cyfrowy akcelerator wybranych modułów standardu kompresji wideo H.264

Streszczenie. *W artykule przedstawiono konfigurowalny cyfrowy akcelerator estymacji ruchu przeznaczony dla enkodera wideo standardu H.264. Akcelerator został zaimplementowany w technologii FPGA oraz w układzie ASIC w technologii UMC 90 nm. Obie implementacje zostały zweryfikowane, a szczegółowe wyniki pomiarów akceleratora ASIC zostały porównane z innymi dostępnymi w literaturze propozycjami. System został zoptymalizowany do współpracy z oprogramowaniem x.264 i jest przeznaczony do sprzętowego wspierania kompresji wideo.*

Abstract. *In the paper a configurable digital motion estimation accelerator for H.264 video compression standard has been described. The accelerator has been implemented in the FPGA and then in the ASIC using the 90 nm UMC technology. These two implementations were successfully verified. Detailed measurement results have been compared with results presented in some papers in the topic of video compression. The system has been optimized for easy integration with x.264 encoder software and is devoted to accelerate video compression. (**Digital accelerator of selected H.264 video compression modules**).*

Słowa kluczowe: kompresja wideo, H.264, estymacja ruchu, ASIC.
Keywords: video compression, H.264, motion estimation, ASIC.

doi:10.12915/pe.2014.09.15

Wstęp

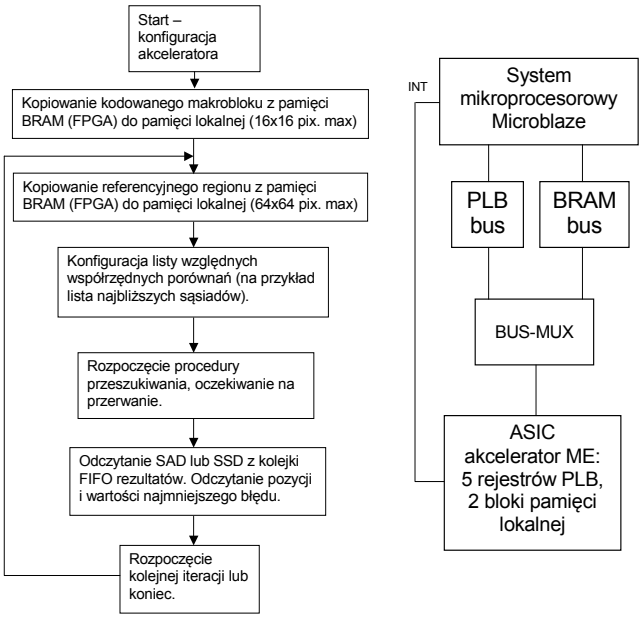
Estymacja ruchu (ang. motion estimation) jest jednym z najbardziej złożonych obliczeniowo modułów enkodera wideo standardu H.264. Wzrost złożoności mechanizmów estymacji ruchu w stosunku do poprzednich standardów spowodował, że obecnie od 50 do 80 procent mocy obliczeniowej całego enkodera jest przeznaczone na obsługę algorytmów związanych z ME [1]. W związku z tym coraz częściej do implementacji algorytmów ME stosuje się dedykowane sprzętowe moduły cyfrowe – akceleratory [1, 2].

Poniżej przedstawiono konfigurowalny akcelerator estymacji ruchu przeznaczony dla enkodera wideo standardu H.264. Akcelerator został wstępnie zaimplementowany w układzie FPGA VIRTEX6-VLX365T, a następnie w układzie ASIC w technologii 90 nm (UMC). Do realizacji wersji FPGA oraz do weryfikacji i testowania prototypu ASIC została wykorzystana zaawansowana platforma prototypowania [3]. Dzięki swojej konstrukcji platforma ta jest szczególnie przydatna do implementacji algorytmów kompresji dźwięku i obrazu HD, oraz systemów widzenia maszynowego. Na platformie został zaimplementowany również system mikroprocesorowy oparty na mikroprocesorze Microblaze, pracujący pod kontrolą systemu operacyjnego Linux.

Weryfikację prototypu przeprowadzono poprzez porównanie wyników jego pracy z wynikami generowanymi przez popularny program enkodera x.264 (który może być uruchomiony na platformie) [4]. Platforma została wyposażona również w złącza pozwalające na podłączenie i testowanie modułu z układem ASIC zawierającym zaimplementowany akcelerator estymacji ruchu.

Budowa i działanie akceleratora

Opisywany akcelerator przyspiesza procedury estymacji ruchu związane z kompresją wideo. Aby prawidłowo działał należy najpierw załadować do pamięci cache akceleratora kodowany makroblok (pamięć p_fenc), następnie należy załadować pamięć obszaru przeszukiwania (p_fref). Akcelerator będzie realizował obliczenie błędu pomiędzy makroblokiem w pamięci p_fenc, a identycznym pod względem rozmiaru fragmentem pamięci p_fref. Pamięć p_fref ma pojemność 64x64 pikseli. Nie ma jednak konieczności umieszczenia w tej pamięci od razu całego obszaru przeszukiwania. Można wprowadzić tylko bezpośrednie sąsiedztwo przewidywanego obszaru przeszukiwania, a następnie uzupełniać je w przypadku gdy schemat przeszukiwania podąży dalej. Akcelerator może współpracować z dowolnymi schematami przeszukiwań typu nearest-neighbor. Za jednym razem można zaprogramować do szesnastu współrzędnych w pamięci p_fref, dla których wynikowe koszty są umieszczane w wyjściowej kolejce FIFO. Na rysunku 1 przedstawiono schemat blokowy obsługi opisywanego akceleratora oraz schemat blokowy systemu prototypowego.



Rys.1. Schematy blokowe algorytmu obsługi akceleratora oraz systemu prototypowego

Konfiguracja akceleratora

Przed uruchomieniem należy skonfigurować akcelerator zapisując rejestr konfiguracji. W bitach b0-b7 znajduje się mniej znaczący bajt parametru „stride" będącego dopełnieniem dodawanym do adresu na końcu każdej kopiowanej linii. Dwa bardziej znaczące bity „stride" znajdują się w bitach b16-17. Parametr „stride" dotyczy adresu w pamięci BRAM układu FPGA podczas kopiowania do pamięci cache akceleratora ASIC i podawany jest w słowach 32-bitowych (minimalny rozmiar poziomy – 4 piksele). W bitach b8-b11 należy wpisać pionowy rozmiar porównywanego makrobloku pomniejszony o 1. W bitach b14-b15 należy wpisać poziomy rozmiar porównywanego makrobloku (w słowach 4-pikselowych) pomniejszony o 1.

Bit b18 wybiera rodzaj testowanej pamięci cache (b18 = 1 – p_fref cache test, b18 = 0 – p_fenc cache test) i ma znaczenie tylko przy uruchomieniu funkcji testu wewnętrznej pamięci cache akceleratora. Bit b19 określa sposób obliczania błędu przy porównywaniu makrobloku z obszarem: b19 = 1 - SSD (sum of squared differences), b19 = 0 - SAD (sum of absolute differences). Bity b22-b27 określają rozmiar pionowy obszaru kopiowanego do pamięci cache pomniejszony o 1. Bity b28-b31 określają rozmiar poziomy obszaru kopiowanego do pamięci cache (w słowach 4-pikselowych) pomniejszony o 1.

Ładowanie pamięci cache

Należy zapisać do pamięci BRAM (lub przetwarzać makrobloki bezpośrednio w tej pamięci po odpowiednim ustawieniu wskaźników np. w programie x.264) obszar podlegający kodowaniu (p_fenc, maksymalny rozmiar 16x16 pikseli) oraz obszar przeszukiwań (p_fref, maksymalny rozmiar 64x64 piksele).

Następnie należy zapisać do rejestru komunikacji rozkaz załadowania obszaru do pamięci cache. Bit b0 tego rozkazu zawiera informację jaka pamięć będzie ładowana: 0 - p_fenc, 1 - p_fref.

Adres źródłowy znajduje się w bitach b16-b29 i dotyczy pamięci BRAM (najmłodsze 2 bity są ignorowane ponieważ transfer odbywa się całymi 32-bitowymi słowami). Adres docelowy znajduje się w bitach b6-b15 i dotyczy pamięci cache w akceleratorze (w przypadku p_fenc tylko 6 bitów jest znaczących – ta pamięć ma pojemność 16x16 = 256 pikseli czyli 64 32-bitowe słowa).

Bit b1 pozwala na rozpoczęcie pojedynczej operacji obliczania błędu już podczas ładowania pamięci p_fref (pamięć p_fenc musi być wcześniej załadowana). W tym przypadku komenda przeszukiwania musi być wprowadzona jako pierwsza do rejestru schematu przeszukiwania z wyłączonym bitem „valid”.

Konfiguracja współrzędnych schematu przeszukiwania

Przed uruchomieniem akceleratora należy wykonać programowanie współrzędnych dla operacji obliczania kosztów w obszarze p_fref. Koszt jest obliczany jako suma bezwzględnej wartości różnic odpowiadających sobie pikseli (ewentualnie możliwe jest włączenie opcji obliczania sumy kwadratów różnic pikseli). Operację tę można zaprogramować na maksimum 16 pozycjach w obrębie pamięci p_fref w celu poszukiwania najmniejszego błędu. W celu zaprogramowania pozycji w pamięci p_fref która ma być porównywana z makroblokiem wprowadzonym do p_fenc należy zaprogramować rejestr schematu przeszukiwania.

Każdy zapis do tego rejestru programuje maksymalnie 2 współrzędne (pierwsza w bardziej znaczącej 16-bitowej połowie, druga w mniej znaczącej połowie). Komenda 16-bitowa ma następujący kształt: VXXXXXXXIYYYYYYY. Bit V (valid) – oznacza że komenda jest ważna – napotkanie nieważnej komendy kończy wprowadzanie (następne nie są sprawdzane). Konieczne jest zapisanie nieważnej komendy jako ostatniej zawsze jeżeli nie wypełniono wszystkich 8 słów FIFO (16 komend). Bity X,Y – oznaczają względne przesunięcie makrobloku porównywanego względem pozycji bazowej (24,24). Bit I jest ignorowany.

Pozycja bazowa to pozycja, w której makroblok p_fenc o rozmiarze 16x16 znajduje się dokładnie w środku obszaru p_fref (64x64). Oznacza to że makroblok 16x16 może być „przesuwany” w każdą stronę obszaru p_fref o taką samą odległość w pikselach. Dla makrobloków o innych wymiarach stosowana jest taka sama pozycja bazowa (chociaż nie jest już w środku obszaru pamięci p_fref).

Uruchomienie operacji obliczania kosztów

Operację uruchamia się poprzez zapis do rejestru rozkazów. Poszczególne bity mają następujące znaczenie: b0 = 1 - uruchamia procedurę odczytu pamięci cache (test pamięci), a jeżeli była ona już uruchomiona to inkrementuje adres odczytu; b0 = 0 - kończy procedurę odczytu pamięci cache – lub nic nie robi jeżeli nie była uruchomiona; b17-23 - w tych bitach zapisany jest dodatkowy offset dodawany do współrzędnej X wszystkich komend z rejestru FIFO (7-bit signed); b25-31 - w tych bitach zapisany jest dodatkowy offset dodawany do współrzędnej Y wszystkich komend z rejestru FIFO (7-bit signed).

Po uruchomieniu należy poczekać na wyniki. Gotowość danych do odczytu można określić badając bit b4 (inter_accel_busy) rejestru statusu lub linię przerwań. Jeżeli jest zerowy można odczytać wyniki z rejestru schematu przeszukiwania – każdy odczyt zawiera wyniki kolejnej komendy obliczenia kosztów (24 młodsze bity) – można odczytać maksymalnie 16 wyników.

Wyniki można także odczytywać podczas działania akceleratora kiedy są sukcesywnie wprowadzane do FIFO, należy jednak najpierw sprawdzić status FIFO w rejestrze rozkazów (bity 28-31) – zawiera on liczbę wyników dostępnych w FIFO czyli liczbę odczytów z rejestru schematu przeszukiwania którą można bezpiecznie wykonać. Uwaga, jeżeli status zawiera 0 może to oznaczać 16 – jeżeli tyle było zaprogramowane, a akcelerator zakończył już działanie (należy sprawdzić rejestr statusu).

Wyniki zbiorcze można odczytać z rejestru komunikacji (wartość minimalnego błędu spośród wszystkich obliczonych) oraz rejestru rozkazu (bity b19-b23) – pozycja minimalnego błędu spośród wszystkich obliczonych (16 oznacza, że jeszcze nie obliczono wszystkich zadanych współrzędnych schematu poszukiwania).

Implementacja akceleratora

Przeprowadzono analizę możliwych do wykorzystania technologii produkcji biorąc pod uwagę: dostępność samej technologii jak i odpowiednich bibliotek, parametry czasowe i elektryczne, czas realizacji oraz cenę. Finalnie zdecydowano się na wykorzystanie procesu CMOS 90nm firmy UMC w wersji L90N Mixed-Mode/RF – 1P9M2T1F wraz z bibliotekami firmy Faraday.

Dla komórek standardowych zastosowano biblioteki w wersji FSD0A_A_GENERIC_CORE_1D2V - 2011Q2v2.2 dla komórek wejścia-wyjścia zastosowano biblioteki FOD0A_B25_T25_GENERIC_IO - 2009Q2v3.0 oraz FSD0A_Memaker - 2010Q1v1.3 jako generator pamięci. Wybrana technologia wykorzystuje 9 warstw metalicznych. Ze względu na ograniczenia bibliotek firmy Faraday dla układów cyfrowych dostępnych jest 8 warstw metalicznych. Produkcja finalnego układu została przeprowadzona w ramach projektu Europractice przez organizację IMEC z Belgii. Synteza i implementacja układu została wykonana z wykorzystaniem oprogramowania Cadence.

Proces projektowania układu scalonego przeprowadzono zgodnie z zaleceniami firmy Cadence, wykonano optymalizację projektu, a także przeprowadzono wielokrotną weryfikację pod kątem poprawności projektu. Proces syntezy układu został wykonany syntezerem RTL Compiler w wersji Encounter(R) RTL Compiler RC10.1.302 - v10.10-s322_1. Do procesu syntezy wykorzystano kody RTL, a także przygotowano plik ograniczeń projektowych (constraints) zawierający wymagania czasowe oraz dyrektywy syntezy.

Podczas syntezy dodano układy testowania DFT. Układy DFT umożliwiają bezpośredni dostęp do wszystkich przerzutników układu. Bloki pamięci otoczono przerzutnikami DFT (tzw. DFT memory collar), przez co

możliwe jest testowanie wewnętrznych pamięci on-chip korzystając z mechanizmów DFT. Dzięki temu, w przypadku błędnego działania układu scalonego, zwiększona jest szansa na znalezienie przyczyny usterki, gdyż komórki DFT umożliwiają ustawienie i odczytanie wartości logicznych z wnętrza układu.

W czasie syntezy wykorzystano także funkcje redukcji poboru mocy poprzez bramkowanie sygnału zegara układami Clock Gating.

Założona powierzchnia układu scalonego wynosi 1874,76 x 1874,76 µm. Jest to związane z wymaganiami organizacji IMEC dotyczącymi maksymalnych wymiarów projektów typu Mini@Sic. W powiązaniu z możliwymi do wykorzystania bibliotekami I/O oraz możliwościami montażowymi IMEC maksymalna możliwa do zrealizowania liczba pól połączeniowych (ang. PAD) wynosi 84. Z tego względu wybrano obudowę układu typu CLCC84.

Biorąc pod uwagę ograniczenia związane z zakłóceniami SSO (Simultaneously Switching Outputs), ograniczenia związane z elektromigracją, zakładaną szybkość pracy i zużycie mocy wyliczona wymagana liczba wyprowadzeń zasilających wynosi łącznie 27.

Układ ASIC zawiera kilka projektów cyfrowych, które ze względu na ograniczoną liczbę wyprowadzeń muszą współdzielić jedną magistralę I/O. Opisywany akcelerator ME zajmuje około 20% rdzenia układu ASIC o powierzchni 1,239 mm^2 (całkowita powierzchnia układu ASIC wynosi 3,514 mm^2 zaś powierzchnia padów 1,362 mm^2). Wyniki pomiarów akceleratora zebrano w tabeli 1.

Tabela 1. Wyniki pomiarów akceleratora cyfrowego ASIC

Schemat przeszukiwania makrobloków 16x16	Liczba porównań makrobloków	Wydajność [makrobloki/s]	Pobór mocy [mW]
diamond search	4	270000	27
hexagon search	6	244000	31
blot search	8	222000	36
uneven-cross multi-hexagon search	16	164000	47

Dane do przeszukiwania są wprowadzane programowo do akceleratora. Przy zastosowaniu sprzętowego interfejsu danych można zlikwidować opóźnienie związane z wprowadzaniem danych, uruchomieniem i reakcją na zgłoszenie przerwania. W takim przypadku możliwe jest dalsze przyspieszenie operacji akceleratora, ponieważ teoretyczna maksymalna prędkość przetwarzania potokowego wynosi 80 cykli zegara / makroblok16x16 dla jednego przeszukania. Wobec tego dla schematu szesnastu przeszukiwań można by uzyskać maksymalną teoretyczną wydajność na poziomie: 1280 cykli / makroblok16x16. Tak więc stosując zegar 400 MHz (jest to maksymalna częstotliwość zegara w wersji ASIC) można osiągnąć przy schemacie UMH (uneven-cross multi-hexagon search) maksymalną wydajność 312500 makrobloków/s.

W tabeli 2 przedstawiono porównanie prezentowanego rozwiązania z innymi dostępnymi w literaturze.

W celu weryfikacji prawidłowości działania akceleratora ASIC zaprojektowano płytkę prototypową współpracującą z wcześniej opracowanym systemem prototypowym opartym na układzie FPGA Virtex6 produkcji Xilinx. System ten został wyposażony w złącze, do którego podłączana jest płytka z układem ASIC. Na płytce testowej z układem ASIC znajduje się dodatkowo regulator napięcia dla rdzenia ASIC (1 V). Zasilanie komórek I/O ASICa odbywa się bezpośrednio z płyty prototypowej FPGA napięciem 2,5 V.

Tabela 2. Porównanie akceleratora ASIC z rozwiązaniami [1] i [2]

Schemat przesz. mbloków 16x16	Liczba porów. mblok.	Prezentow. ASIC (interfejs program.) [mbloki/s]	Prezentow. ASIC (teor. potok sprzętowy) [mbloki/s]	Wyniki z publik. [1] [mbloki/s]	Wyniki z publik. [2]
diamond search	4	270000	1250000	236842	
hexagon search	6	244000	833000	161483	
blot search	8	222000 [1920x1080 @ 27 fps] (36 mW)	625000 [1920x1080 @ 76 fps]		1920x1080 @ 55 fps (453 mW)
UMH	16	164000	312500	52837	

Oprócz tego płytkę wyposażono w dodatkowe złącza pomiarowe oraz możliwość podłączenia sygnałów sterujących funkcjonalnością design-for-test (DFT) układu ASIC.

Na rysunku 2 przedstawiono zdjęcie płytki prototypowej z zamontowanym układem ASIC akceleratora.

Rys.2. Fotografia płytki prototypowej z zamontowanym układem ASIC akceleratora estymacji ruchu

Podsumowanie

Przedstawiona sprzętowa implementacja algorytmu ME (motion estimation) za pomocą cyfrowego układu ASIC została zweryfikowana pozytywnie. Wydajność obliczeniowa oraz pobór mocy są porównywalne lub lepsze

od innych opublikowanych propozycji implementacji algorytmu ME. Dzięki łatwości integracji z oprogramowaniem x.264 opracowany ASIC może zostać wykorzystany do natychmiastowego wdrożenia, może także zostać dodany do większego układu ASIC i uzyskać jeszcze większą wydajność dzięki zredukowaniu ograniczeń komunikacyjnych (mała liczba końcówek w prototypowym układzie). Opisany akcelerator może także zostać wykorzystany w algorytmach predykcji ruchu (motion prediction), a także z niewielkimi modyfikacjami może posłużyć do implementacji algorytmów intra-predykcji. Opisany akcelerator może także zostać zastosowany jako wstępny prototyp do dalszych badań rozwojowych nad implementacją elementów inter-predykcji i intra-predykcji nowego standardu H.265/HEVC (High Efficiency Video Coding), który obecnie upowszechnia się ze względu na zwiększające się zainteresowanie standardami telewizji UHDTV (Ultra High Definition Television).

Praca była częściowo finansowana z grantu NCBiR nr O R00 0046 09.

LITERATURA

[1] Nunez-Yanez J.L., Hung E., Chouliaras V., A configurable and programmable motion estimation processor for the H.264 video codec, *International Conference on Field Programmable Logic and Applications*, (2008), 149-154

[2] Chandrasetty V.A., Laddha S.R., A novel dual processing architecture for implementation of motion estimation unit of H.264 AVC on FPGA, *IEEE Symposium on Industrial Electronics & Applications*, (2009), Vol.1, 62-67

[3] Kłosowski M., Wireless intelligent audio-video surveillance prototyping system, *Przegląd Elektrotechniczny*, 89 (2013), nr 10, 97-99

[4] x.264 encoder, http://www.videolan.org/x264.html

***Autorzy**: dr inż. Miron Kłosowski, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Katedra Systemów Mikroelektronicznych, ul. Gabriela Narutowicza 11/12, 80-233 Gdańsk, E-mail: klosowsk@ue.eti.pg.gda.pl;*
dr inż. Bogdan Pankiewicz, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Katedra Systemów Mikroelektronicznych, ul. Gabriela Narutowicza 11/12, 80-233 Gdańsk, E-mail: bpa@ue.eti.pg.gda.pl;
dr inż. Marek Wójcikowski, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Katedra Systemów Mikroelektronicznych, ul. Gabriela Narutowicza 11/12, 80-233 Gdańsk, E-mail: wujek@ue.eti.pg.gda.pl.