Michał WIECZOREK, Grzegorz PASTUSZAK
POLITECHNIKA WARSZAWSKA, INSTYTUT RADIOELEKTRONIKI
ul. Nowowiejska 15/19, 00-665 Warszawa

Sprzętowa implementacja dekodera nagłówków i dekodera CAVLC w standardzie kompresji wideo H.264/AVC

Mgr inż. Michał WIECZOREK

Uzyskał stopień magistra inżyniera o specjalności telekomunikacja we wrześniu 2009 roku na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie pracuje w Instytucie Radioelektroniki Politechniki Warszawskiej na stanowisku asystenta naukowego. Od 2008 roku zajmuje się realizacją układów w technologii FPGA związanych z kompresją wideo.

e-mail: M.Wieczorek@ire.pw.edu.pl

Dr inż. Grzegorz PASTUSZAK

Uzyskał stopień magistra inżyniera o specjalności mikroelektronika w czerwcu 2001 roku a stopień doktora inżyniera o specjalności techniki multimedialne w czerwcu 2006 roku. Obecnie jest pracownikiem naukowym Instytutu Radioelektroniki Politechniki Warszawskiej. Jego obszar zainteresowania obejmuje: architektury i algorytmy VLSI, przetwarzanie obrazów/wideo/audio oraz kompresję, wydajne cyfrowe układy scalone.

e-mail: g.pastuszak@ire.pw.edu.pl

Streszczenie

Poniższy artykuł zawiera opis sprzętowej realizacji dekodera nagłówków strumienia oraz kontekstowo-adaptacyjnego dekodera kodów zmiennej długości zgodnych ze standardem kompresji wideo H.264/AVC. Przedstawiony układ jest w stanie odczytać i zdekodować parametry strumienia oraz dane sterujące poszczególnych elementów składni jak również odtworzyć bloki współczynników zapisanych przy użyciu kodera VLC. Zaprojektowany moduł został poddany syntezie zarówno dla technologii FPGA jak i ASIC a poprawność jego działania została zweryfikowana zgodnie z modelem referencyjnym JM w wersji 16. Wyniki syntezy proponowanego dekodera pokazują, iż może pracować on z częstotliwością taktowania 100MHz na układach FPGA z rodziny Stratix II, co pozwala na obsłużenie sekwencji w wysokiej rozdzielczości HDTV.

Słowa kluczowe: kompresja wideo, CAVLC, H.264/AVC.

Stream header decoder and context-adaptive variable-length decoder hardware module for H.264/AVC codec

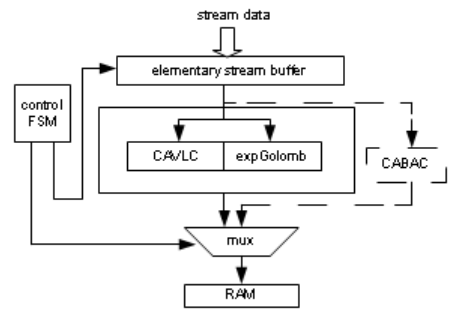
Abstract

This paper describes the implementation of a stream header decoder and a context-adaptive variable-length decoder in conformity with the H.264/AVC standard. This module is able to decode headers of syntax elements and to decode blocks of transform coefficients coded using context-adaptive variable-length coder. The designed module is synthesized based on FPGA and ASIC technologies and verified with the reference model JM in version 16. The implementation results show that the architecture can work at 100 MHz for FPGA Stratix II devices and can support HDTV in real time. There are two main methods of improving the CAVLC decoding process. The most common is a multi-symbol decoding architecture shown in [1], [4], and [5]. In [2] a Hierarchical logic for Look-up tables (HLLT) algorithm is proposed. It simplifies calculation of the coefficient-token parameter but generates a five-element long cascade which may reduce the speed of decoding process. In [5] also a way of grouping the coefficient-token codewords is proposed. All the publications concentrate mainly on the CAVLC design and do not describe decoding of control data in detail (e.g., headers, macroblock/block types, coded block pattern, and motion vectors). The proposed binary decoder supports all the functionality of H.264/AVC High Profile, except of MBAFF mode and SEI elements. Although the architecture needs more logic gates than other analyzed designs, it enables also decoding of all syntax elements and provides much more functionality. The throughput is sufficient to support HDTV applications in real time.

Keywords: video coding, CAVLC decoder, H.264/AVC.

1. Realizacja

Funkcjonalny schemat realizowanego dekodera przedstawiony jest na rysunku 1. Składa się on z bufora wejściowego, automatu sterującego wraz z dekoderem nagłówków, dekodera eksponencjalnych kodów Golomba oraz dekodera CAVLC.



Rys. 1. Funkcjonalny schemat modułu dekodera
Fig. 1. Proposed architecture of H.264/AVC binary decoder

Wejściowy bufor strumienia bitowego (elementary stream buffer) odpowiada za komunikację dekodera z zewnętrznym źródłem danych. Te przyjmowane są w blokach po 32 bity. W każdym takcie zegara dekoder entropijny dekoduje słowa kodowe z najstarszych bitów bufora. Jednocześnie określa ile bitów jest do tego potrzebnych. Jeśli liczba ważnych bitów w buforze wejściowym jest niewystarczająca do zdekodowania danego słowa, praca dekodera jest wstrzymywana a układ bufora wysyła żądanie przesłania kolejnych 32 bitów danych.

Pracą poszczególnych elementów dekodera steruje odpowiedni automat (control FSM). Wymusza on właściwe tryby pracy i kolejność dekodowania poszczególnych elementów strumienia standardu H.264/AVC. Projektowany układ nie zawiera dekodera arytmetycznego CABAC, który jest zupełnie oddzielnym zagadnieniem. Niemniej automat sterujący przewiduje uruchomienie takiej jednostki w połączeniu z zaprojektowanym dekoderem. Ponadto automat sterujący odpowiada także za właściwe adresowanie pamięci zewnętrznych, do których zapisywane są między innymi wartości zdekodowanych współczynników, parametry predykcji czy wektory ruchu.

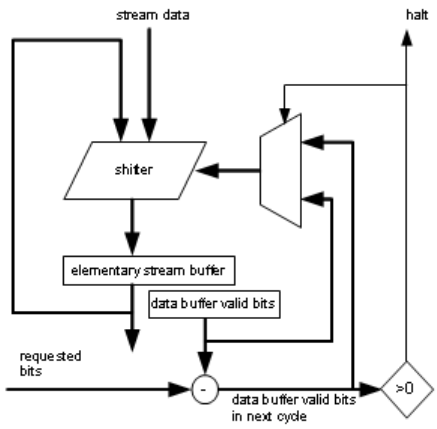
Podstawowym założeniem projektowym jest minimalna częstotliwość taktowania układu. Wynosi ona 100 MHz. Ponadto średnio na jeden takt zegara powinien być dekodowany jeden współczynnik. Takie założenia powinny pozwolić na zastosowanie projektowanego układu do dekodowania w czasie rzeczywistym sekwencji w wysokiej rozdzielczości HDTV.

1.1. Bufor wejściowy

Do bufora wejściowego (elementary stream buffer) podawane są odpowiednio przesunięte dane ze strumienia bitowego lub z wyjścia bufora. Procesem zapisu i przesuwania bitowego steruje oparty na logice kombinacyjnej prosty moduł sterujący. Na podstawie liczby ważnych bitów w buforze data buffer valid bits oraz liczby bitów potrzebnych do zdekodowania aktualnego symbolu

requested bits podejmowana jest decyzja o trybie pracy bufora. Schemat bufora wejściowego przedstawiono na rysunku 2.

Jeśli liczba bitów w buforze jest wystarczająca aby poprawnie zdekodować symbol następuje przesunięcie bitów z bufora o wartość requested bits. Odpowiednio dekrementowana jest też wartość z rejestru data buffer valid bits.



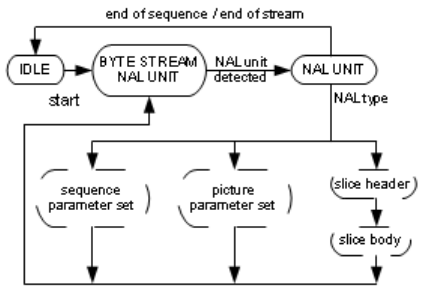
Rys. 2. Schemat modułu bufora wejściowego
Fig. 2. Proposed architecture of elementary stream buffer

Jeśli liczba ważnych bitów w buforze jest wówczas mniejsza niż 32, wystawiane jest żądanie otrzymania kolejnych 32 bitów danych. Komunikacja odbywa się na zasadzie dwu-sygnałowego handshake'u. Kiedy na wejściu są ważne dane, odpowiednia flaga sygnalizacyjna przyjmuje wartość '1'. W przypadku, gdy w buforze wejściowym nie ma wystarczającej liczby ważnych bitów, aby poprawnie zdekodować dany symbol, wstrzymywana jest praca układu dekodera – sygnał halt przyjmuje wartość '1'.

1.2. Automat sterujący

Zasada działania automatu sterującego analizą strumienia opiera się na charakterze i cechach strumienia standardu H.264/AVC.

Jest on podzielony na jednostki sieciowe NAL (Network Abstract Layer) i wewnątrz tych jednostek w sposób hierarchiczny przesyłane są poszczególne elementy składniowe. Praca dekodera opiera się na rozpoznawaniu, identyfikacji i odpowiednim dekodowaniu poszczególnych jednostek NAL. Zasadę pracy dekodera ilustruje rysunek 3.



Rys. 3. Schemat pracy automatu sterującego
Fig. 3. Control FSM data flow

Jednostki NAL rozpoczynają się z wyrównaniem do pełnego bajtu. Wystąpienie sekwencji rozpoczynającej NAL powoduje przejście automatu do trybu identyfikacji poszczególnych jednostek NAL. Ich typ zapisany jest w strumieniu jako liczba pięciobitowa. W zależności od zdekodowanej wartości, automat przechodzi do stanów odpowiadających za dekodowanie danych elementów składniowych. Koniec jednostki NAL także występuje z wyrównaniem do pełnego bajtu. Praca automatu sterującego polega na sekwencyjnym dekodowaniu poszczególnych elementów strumienia standardu H.264/AVC. W każdym ze stanów na bieżąco sprawdzane są warunki kolejnych przejść.

1.3. Dekoder kodów Golomba

W standardzie H.264/AVC wykorzystywana jest pewna modyfikacja podstawowej wersji kodu Golomba. Kodowany alfabet dzielony jest na podprzedziały, których długość rośnie wykładniczo. Stąd też nazwa tej modyfikacji – eksponencjalny kod Golomba. Słowa kodowe tworzone są według zależności:

$$code\_number = 2^{leading\_zeros} - 1 + B(n)_{leading\_zeros}, \qquad (1)$$

gdzie:
- leading zeros – liczba zer wiodących w strumieniu bitowym
- $B(x)_n$ – liczba w kodzie binarnym zapisana na n-bitach

Po przeprowadzeniu analizy eksponencjalnego kodu Golomba można zastosować pewną modyfikację definicyjnego sposobu dekodowania symboli. Modyfikacja dotyczy wyznaczania podstawę kodu Golomba. Jej wartość to:

$$2^{leading\ zeros} - 1, \qquad (2)$$

Jeśli jednak w układzie wczytywanie przyrostka kodu dodamy odpowiednią potęgę dwójki, co wiąże się jedynie z ze zmianą bitu poprzedzającego wartość wczytywaną ze strumienia z '0' na '1', wystarczy od tej wartości odjąć jeden i otrzymamy wartość eksponencjalnego kodu Golomba. Takie rozwiązanie zdecydowanie upraszcza proces dekodowania. Architektury proponowane w literaturze są w zasadzie odwzorowaniem teorii odczytywania eksponencjalnych kodów Golomba, nie wprowadzają żadnych optymalizacji sprzętowych.

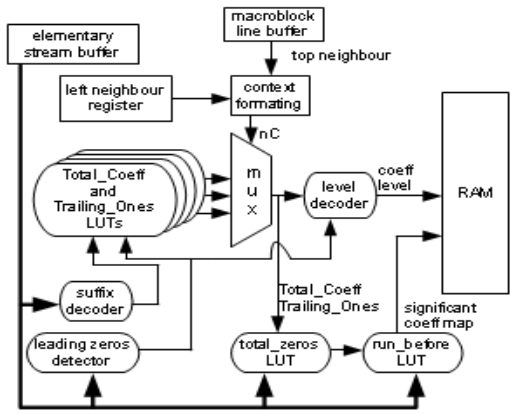
1.4. Dekoder CAVLC

Sekwencyjny sposób dekodowania kodów CAVLC wymusza standardową architekturę, która jest dość obszernie opisana w literaturze. Niemniej, spotykane są dwie metody optymalizacji procesu dekodowania – dekodowanie wielo-symbolowe zwiększające przepustowość dekodera oraz optymalizacja i uproszczenie dekodowania parametru coeff_token z tablic kontekstowych. W przedstawionym projekcie skupiono się na drugiej z metod celem redukcji zasobów sprzętowych modułu dekodera.

Do wyznaczenia wartości parametru coeff_token wykorzystywanych jest 6 tablic kontekstowych (4 dla luminancji i dwie dla chrominancji). Wybór odpowiedniej dokonywany jest zależnie od ilości niezerowych współczynników w sąsiednich, zdekodowanych już blokach. Jedynie jedną z tablic można uprościć do kilku elementarnych operacji, pozostałe zaś wymagają od 10 do 16 bitów aby zaadresować wprost całą ich przestrzeń danych. Zważywszy, że znajdują się w nich jedynie 62 elementy widać, iż występuje tu bardzo duża nadmiarowość. W [2] proponowana jest hierarchiczna struktura tabel pomocniczych (HLLT) polegająca na grupowaniu elementów o podobnym prawdopodobieństwie wystąpienia, a co za tym idzie i długości słowa kodowego, w mniejszych tabelach i ich kaskadowym połączeniu. Pozwala to zredukować ogólny rozmiar użytych tabel, niemniej powstała w ten sposób pięcioelementowa kaskada generuje znaczące opóźnienie w układzie.

Analiza słów kodowych parametru coeff_token pozwala zauważyć, że są one ciągiem zer zakończonym maksymalnie czterobitowym suffixem. Ponieważ pierwszym jego bitem będzie jedynka (kończąca ciąg zer) jego długość redukuje się de facto do 3 bitów. Poprzedzić go może ciąg maksymalnie 14 zer, taką zaś informację można zapisać na 4 bitach. Reasumując, do reprezentacji każdego słowa kodowego potrzeba jedynie 7 bitów, co jest znacząco mniejszą wielkością niż w przypadku bezpośredniej

implementacji tablic przedstawionych w normie standardu H.264/AVC [1]. Podobne podejście opisano w [5], ale wykorzystuje się tam 11 bitów, więc więcej niż w niniejszym projekcie. Warto nadmienić, że detektor zer wiodących wykorzystywany jest także w dekoderze eksponencjalnych kodów Golomba. Moduły te nie pracują nigdy w tym samym czasie, więc można bez przeszkód współdzielić dostęp do detektora. Na rysunku 4 przedstawiono proponowaną architekturę dekodera CAVLC.



Rys. 4. Proponowana architektura dekodera CAVLC
Fig. 4. Proposed architecture of CAVLC decoder

2. Wyniki syntezy

Przedstawiona architektura została zaimplementowana przy użyciu języka VHDL. Pozwoliło to na przeprowadzenie symulacji funkcjonalnej i weryfikację poprawności pracy zaprojektowanego dekodera w oparciu o wektory testowe otrzymane z oprogramowanie referencyjnego JM 16. Synteza logiczna została wykonana zarówno dla układu FPGA z rodziny Stratix II (EP2S15F484C3) jak również ASIC w technologii AMS 0.35 µm. Wyniki zamieszczono w tabelach 1 i 2.

Tab. 1. Wyniki syntezy proponowanego projektu dekodera dla układów FPGA z rodziny Stratix II
Tab. 1. Implementation results of the proposed decoder design for FPGA Stratix II devices

jednostki ALUT	2 961
rejestry	829
pamięć RAM [bity]	2 048
maksymalna częstotliwość taktowania [MHz]	111.07

Tab. 2. Porównanie z innymi realizacjami dekodera CAVLC
Tab. 2. Comparison with other CAVLD architectures

	[2]	[3]	[4]	proponowana architektura
technologia [µm]	0.18	0.18	TSMC 0.18	AMS 0.35
liczba bramek	9 943	17 202	13 189	19 789
pamięć RAM [bity]	1 152	5 120	-	2 048
zegar [MHz]	125	-	-	100

Proponowana realizacja dekodera wykorzystuje więcej zasobów sprzętowych niż inne spotykane w literaturze. Niemniej, poza samym dekoderem CAVLC umożliwia także dekodowanie elementów składni strumienia H.264/AVC oraz nagłówków z danymi sterującymi. Ponadto zawiera interfejsy do zewnętrznego źródła danych, modułów pamięci i pozostałych elementów dekodera wideo. Układ posiada również moduł generacji danych kontekstowych niezbędnych podczas dekodowania kodów CAVLC. W tabeli 3 zamieszczono zestawienie szacowanej prędkości pracy dekodera dla kilku standardowych sekwencji wideo przy różnych parametrach kwantyzacji przy użyciu wewnątrzramkowego trybu predykcji INTRA.

Tab. 3. Średnia liczba cykli zegara potrzebnych do dekodowania makrobloku w trybie INTRA
Tab. 3. Average number of clock cycles per macroblock for INTRA mode

sekwencja	QP	tylko CAVLD	cały dekoder	[2]
news	12	434,5	564,1	404
	20	273,3	367,6	282
	28	146,7	219,54	196
mobile	12	703,6	895,8	704
	20	528,4	655,3	570
	28	318,7	412,8	395
football	12	497,8	619,4	-
	20	268	365,5	-
	28	125,4	203,1	-
średnia	12	547,5	691,8	554
	20	343,2	446,7	426
	28	180,4	259,2	296

3. Wnioski

Zaprojektowany moduł dekodera binarnego zapewnia funkcjonalność profilu wysokiego standardu H.264/AVC z wyjątkiem trybu MBAFF oraz elementów SEI (supplemental enhancement information). W porównaniu z innymi realizacjami dekodera CAVLC, proponowana architektura zużywa więcej zasobów sprzętowych, niemniej poza rekonstrukcją bloków współczynników pozwala także na odczytanie ze strumienia wszystkich elementów składni i danych sterujących procesem dekodowania strumienia wideo. Przepustowość projektowanego układu pozwala na zastosowanie go do dekodowania w czasie rzeczywistym sekwencji w wysokiej rozdzielczości HDTV.

Praca jest elementem projektu "Zintegrowany mobilny system wspomagający działania antyterrorystyczne i antykryzysowe - PROTEUS", realizowanego dzięki wsparciu finansowemu Europejskiego Funduszu Rozwoju Regionalnego w ramach 1. osi priorytetowej Programu Operacyjnego Innowacyjna Gospodarka, 2007 - 2013, Poddziałanie 1.1.2, projekt numer PO-IG.01.02.01-00-014/08-00.

4. Literatura

[1] Recommendation ITU-T H.264(2007) — Corrigendum 1. Joint Video Team of ITU-T VCEG and ISO/IEC MPEG.
[2] Hsiu-Cheng Chang, Chien-Chang Lin, Jiun-In Guo (2005). A Novel Low-Cost High-Performance VLSI Architecture for MPEG-4 AVC/H.264 CAVLC Decoding. ISCAS 2006. Proceedings. IEEE International Symposium on Circuits and Systems.
[3] Guo-Shiuan Yu and Tian-Sheuan Chang.(2006). A zero-skipping multi-symbol CAVLC decoder for MPEG-4 AVC/H.264. ISCAS 2006. Proceedings. IEEE International Symposium on Circuits and Systems.
[4] Tsung-Han Tsa, De-Lung Fang, Yu-Nan Pan(2007). A Hybrid CAVLD Architecture Design with Low Complexity and Low Power Considerations. IEEE International Conference on Multimedia and Expo.
[5] Lee G. G., Lo C. C., Chen Y. C., Lin H. Y., Wang M. J. (2010). High-throughput low-cost VLSI architecture for AVC/H.264 CAVLC decoding. Image Processing, IET.

otrzymano / received: 15.10.2010
przyjęto do druku / accepted: 01.12.2010 *artykuł recenzowany*