Sebastian KORYCIAK, Agnieszka DĄBROWSKA-BORUCH, Kazimierz WIATR
AGH AKADEMIA GÓRNICZO-HUTNICZA W KRAKOWIE, ACK CYFRONET AGH, ul. Nawojki 11, 30-950 Kraków
AGH AKADEMIA GÓRNICZO-HUTNICZA W KRAKOWIE, KATEDRA ELEKTRONIKI, Al. Mickiewicza 30, 30-059 Kraków

Sprzętowa akceleracja wybranych algorytmów kompresji obrazu nieruchomego w standardzie JPEG

Mgr inż. Sebastian KORYCIAK

Ukończył studia na AGH (2011), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest doktorantem na tym samym wydziale, oraz studiuje w Międzywydziałowej Szkole Inżynierii Biomedycznej AGH na studiach inżynierskich z inżynierii biomedycznej. Jego zainteresowania to implementacja algorytmów kompresji obrazów i sieci neuronowych przy pomocy układów programowalnych.

e-mail: koryciak@agh.edu.pl

Dr inż. Agnieszka DĄBROWSKA - BORUCH

Absolwentka kierunku Elektronika i Telekomunikacja na Wydziale EAIiE AGH (2002), dr nauk technicznych (2007). Obecnie jest adiunktem w Katedrze Elektroniki AGH oraz członkiem Zespołu Akceleracji Obliczeń ACK Cyfronet AGH. Jej zainteresowania naukowe to kompresja obrazu, systemy czasu rzeczywistego, układy programowalne oraz rekonfigurowalne.

e-mail: adabrow@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR

Studia AGH Kraków (1980), doktor nauk technicznych (1987), doktor habilitowany (1999) i profesor (2002). Profesor zwyczajny w Akademii Górniczo - Hutniczej oraz dyrektor ACK Cyfronet AGH. Prowadzone prace badawcze dotyczą komputerowego sterowania procesami, systemów wizyjnych, systemów wieloprocesorowych, układów programowalnych, rekonfigurowalnych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń.

e-mail: wiatr@agh.edu.pl

Streszczenie

Artykuł opisuje opracowanie akceleratora dla wybranych algorytmów kompresji obrazu nieruchomego. Do jego sprzętowej realizacji został wykorzystany język opisu sprzętu VHDL. Wynikiem pracy była skuteczna implementacja na układ programowalny dekompresora obrazów nieruchomych zapisanych w standardzie JPEG ISO/IEC 10918-1(1993), trybie Baseline będącym podstawowym i obowiązkowym trybem dla tego standardu. Szczególną uwagę poświęcono wyborowi i implementacji dwóch najważniejszych zdaniem autora algorytmów występujących w omawianym standardzie.

Słowa kluczowe: Akceleracja sprzętowa, JPEG, FPGA, IDCT, Huffman.

Hardware acceleration of image compression algorithms in JPEG standard

Abstract

Image compression is one of the most important topics in the industry, commerce and scientific research. Image compression algorithms need to perform a large number of operations on a large number of data. In the case of compression and decompression of still images the time needed to process a single image is not critical. However, the assumption of this project was to build a solution which would be fully parallel, sequential and synchronous. The paper describes the development of an accelerator for selected still image compression algorithms. In its hardware implementation there was used the hardware description language VHDL. The result of this work was a successful implementation on a programmable system decompressor of still images saved in JPEG standard ISO / IEC 10918-1 (1993), Baseline mode, which is a primary, fundamental, and mandatory mode for this standard. The modular system and method of connection allows the continuous input data stream. Particular attention was paid to selection and implementation of two major, in the authors opinion, algorithms occuring in this standard. Executing the IDCT module uses an algorithm transformation IDCT-SQ modified by the authors of this paper. It provides a full pipelining by applying the same kind of arithmetic operations between each stage. The module used to decode Huffman's code proved to be a bottleneck for the whole project.

Keywords: Custom Computing, JPEG, FPGA, IDCT, Huffman.

1. Wstęp

Istnieją tysiące medycznych, przemysłowych, wojskowych i naukowych zastosowań cyfrowej analizy i przetwarzania obrazów.

Kompresory i dekompresory obrazu są implementowane głównie w oprogramowaniu, ponieważ współczesne procesory dysponują pełną paletą przydatnych instrukcji. Innym rozwiązaniem jest implementacja kodeka obrazu (kompresor plus dekompresor) bezpośrednio w sprzęcie. Pozwala to całkowicie uniezależnić operację obróbki obrazu od wykonywanych przez procesor instrukcji, pozwalając na pracę równoległą, a tym samym przyspieszenie działania danego urządzenia. W chwili obecnej większość procesorów przeznaczonych na rynek rozwiązań mobilnych posiada wbudowane tego typu kodeki. Implementacja z wykorzystaniem układu programowalnego FPGA pozwala nam na maksymalne zrównoleglenie wykonywanych operacji przy równoczesnej możliwości ciągłego modyfikowania i udoskonalania.

Sama w sobie kompresja danych to proces redukcji ilości informacji do poziomu pozwalającego odtworzyć oryginalną informację. Sposoby kompresji obrazu możemy podzielić na kompresje stratną i bezstratną, a ich użycie w dużej mierze zależy od wymagań stawianych przez daną aplikację. Kompresja JPEG może być użyta zarówno jako stratna jak i bezstratna, w obu przypadkach oferując redukcję zbędnej ilości danych.

2. Platforma obliczeniowa Pico Computing

Cały projekt był realizowany z myślą o implementacji na platformie PICO EX-500 stanowiącej interfejs do komunikacji pomiędzy komputerem klasy PC a układami programowalnymi FPGA [4]. Firma Pico Computing dostarcza rozwiązania mające na celu wykorzystaniu wielu modułów FPGA do wysokowydajnego przetwarzania danych. Płyty EX-Series oferują wysoką wydajność dla najbardziej wymagających zastosowań układów programowalnych.

Możliwości platformy PICO EX-500 to:

- jednoczesna obsługa do 6 układów FPGA firmy Xilinx
- kompatybilność z modułami PICO serii M
- komunikacja z komputerem za pomocą złącza PCI Express x16 drugiej generacji

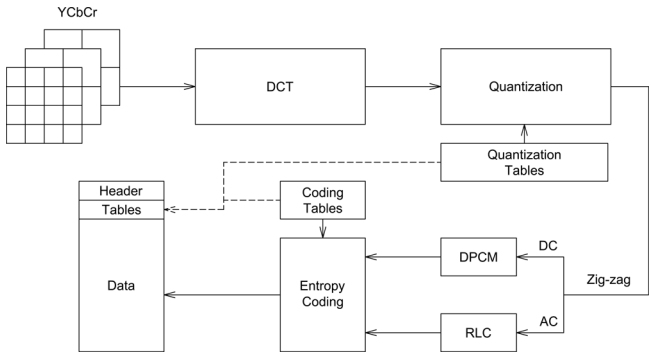
Serce układu stanowią moduły z serii PICO M-503.

Ich parametry to:

- Virtex-6 LX240T (xc6vlx240t-2) [5]
- PCI Express x8 drugiej generacji
- dwie kości 4GB DDR3 SODIMM
- trzy kości 9MB QDRII SRAM

3. Standard JPEG

Tryb sekwencyjny standardu JPEG, a dokładniej JPEG Baseline ISO/IEC 10918-1, jest trybem obowiązkowym dla każdego dekodera wykorzystującego kompresję stratną i działającego na podstawie dyskretnej transformaty kosinusowej. Korzystając z kwantyzacji, kodowania entropijnego oraz transformaty DCT na blokach zapisanych z 8 bitową precyzją kompresor potrafi osiągnąć dosyć wysoki stopień kompresji, który okupiony jest przez spadek jakości obrazu odtworzonego. Tryb podstawowy zakłada, że obraz jest zapisany z 8 bitową precyzją, ale inne tryby zapewniają znacznie większe głębie kolorów. Standard JPEG przyjmuje jako podstawowe bloki do przetwarzania matrycę wielkości 8 x 8 pikseli, które kolejno są wprowadzane do kompresora.

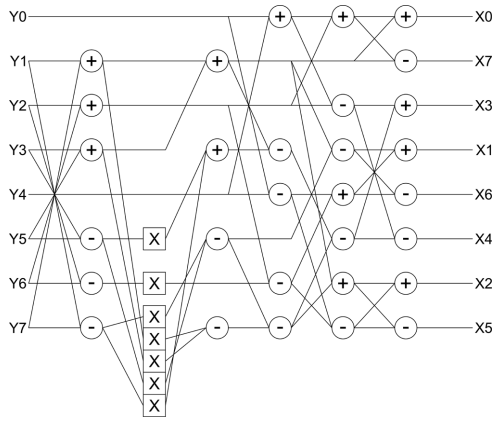


Rys. 1. Idea działania kodera JPEG pracującego w trybie sekwencyjnym DCT
Fig. 1. The main idea of the JPEG coder working in sequential DCT mode

Podstawowy tryb standardu JPEG posiada kilka konfigurowalnych zmiennych, w tym tablice kwantyzacyjne oraz tablice Huffmana, które mogą być dokładnie opisane w części nagłówkowej pliku. Analizując dokładnie obraz źródłowy poddawany kompresji algorytmy kwantyzacji i Huffmana mogą zostać zoptymalizowane tak aby osiągnąć jak najlepszy poziom kompresji bez zbyt dużej utraty jakości informacji. Ponad to istnieje możliwość zmniejszenia ilości próbek przy zapisie chrominancji. Pomimo tego, że są to jedyne stopnie swobody w tym trybie, pozwala on na osiągnięcie bardzo dobrych rezultatów.

4. Algorytm dyskretnej transformaty kosinusowej

Dyskretną transformatę DCT można przedstawić w postaci macierzowej [1]. Rozwiązanie tego typu niesie za sobą potrzebę wykonania 64 mnożeń i 54 dodawań do obliczenia pojedynczego wektora współczynników.



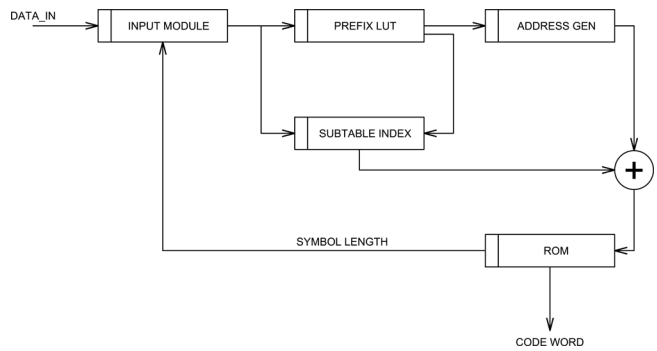
Rys. 2. Graficzny schemat przepływu danych dla transformaty IDCT w algorytmie Arai zmodyfikowanego na potrzeby wykonywanego projektu
Fig. 2. Data flow diagram for the modified IDCT transform algorithm of Arai

Po wykonaniu paru operacji na macierzy liczbę mnożeń można zredukować do 22. Jednak w przypadku realizacji sprzętowej rozwiązanie tego typu powoduje wystąpienie problemów związanych z synchronizacją zadań arytmetycznych, którego konsekwencją jest konieczność dobierania odpowiednich opóźnień.

Powszechnie znanym rozwiązaniem jest algorytm DCT-SQ (sekwencyjna kwantyzacja) zaproponowany przez Arai [2]. Wymaga on jedynie 5 mnożeń oraz 29 operacji dodawania-odejmowania.

5. Algorytm dekodowania Huffmana

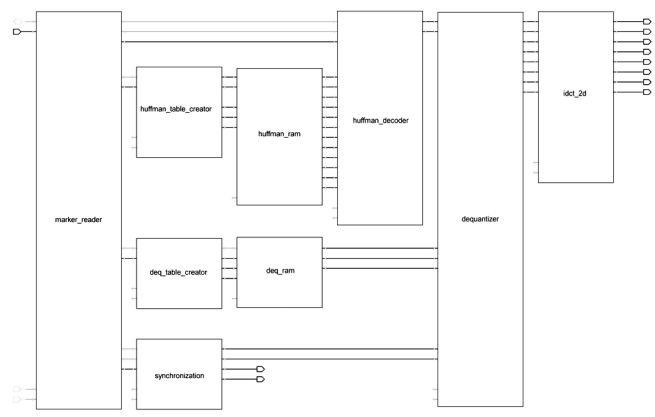
Algorytmem stosowanym do dekodowania Huffmana jest procedura dopasowywania prefiksów [3]. Polega ona na podzieleniu słów kodowych na dwie części: przedrostek i jego rozwinięcie. W ten sposób tworzy się tablice dla każdego z prefiksów. Podczas dekodowania najpierw porównywany jest przedrostek, a następnie kojarzona z nim tablica. Aby cały proces przyśpieszyć pierwsza część procedury wykonywana jest na jednostkach LUT, a sczytywanie ostatecznego słowa na zasadzie podania adresu w odpowiedniej tablicy. Krytycznym dla tego rodzaju algorytmów okazuje się być wybór i selekcja odpowiednich przedrostków. W przypadku dekodowania na podstawie tablic stworzonych dla standardów JPEG i MPEG-2 najskuteczniejszym sposobem jest sczytywanie wiodących jedynek słowa kodowego.



Rys. 3. Idea działania algorytmu dopasowywania prefiksów
Fig. 3. The idea of the prefix fit algorithm

6. Struktura zaimplementowanych modułów

Zadaniem układu jest zdekodowanie obrazu zapisanego w postaci binarnej w standardzie JPEG ISO/IEC 10918-1(1993) w trybie baseline. Po odebraniu danych od jednostki obsługującej pamięć następuje sekwencja czynności wykonywanych synchronicznie przez serię automatów zawartych w poszczególnych modułach. Cały dekoder składa się z dziewięciu bloków wyznaczonych w taki sposób, aby jednoznacznie pełnić określoną funkcję.

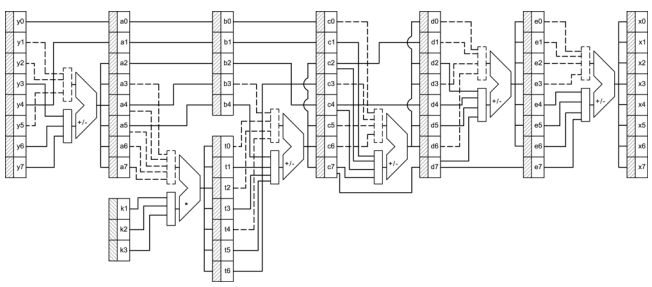


Rys. 4. Schemat blokowy dekodera JPEG
Fig. 4. Block diagram of the JPEG decoder

Moduł transformacji IDCT składa się z szeregu rejestrów i poprzedzających je układów wykonujących podstawowe operacje arytmetyczne, które w trybie potokowym przetwarzają dane wejściowe. Cały układ działa w oparciu o algorytm DCT-SQ zaproponowany przez Arai [2]. Został on odwrócony i zmodyfikowany w celu zapewnienia występowania tego samego rodzaju operacji arytmetycznej pomiędzy kolejnymi stopniami rejestrów. Mnożenie wykonywane jest przez stały współczynnik, co pozwoliło na zastąpienie go serią rejestrów przesuwnych i sumatorów. Pełna transformacja 8 danych wejściowych wymaga jedynie 6 taktów zegara. Potokowość pozwala na wprowadzanie nowych danych w każdym kolejnym takcie.

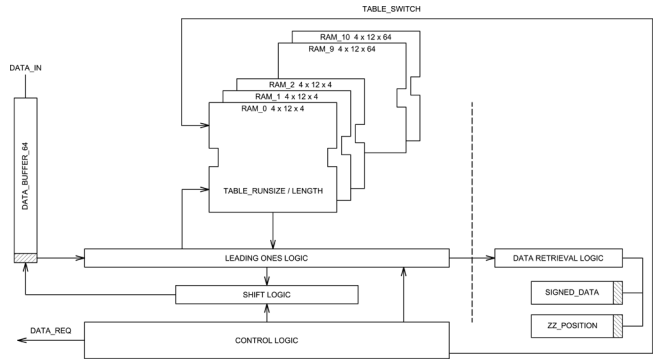
Tab. 1. Operacje arytmetyczne wykonywane w algorytmie ułożone w kolejności występowania, oraz wartości współczynników stałych, przez które odbywa się mnożenie
Tab. 1. Arithmetic operations performed in the algorithm arranged in order of appearance

	b0=a0 b1=a1				
a0=y0	b2=a2	c0=b0	d0=c0+c1	e0=d0+d4	x0=e0+e7
a1=y4	b3=a4	c1=b1	d1=c2	e1=d3+d5	x1=e1+e6
a2=y2+y6	b4=a5	c2=t1-t2	d2=c6-c2	e2=d0-d4	x2=e3+e4
a3=y2-y6	t0=a6*k1	c3=t4	d3=c0-c1	e3=d3-d5	x3=e2+e5
a4=y1+y7	t1=a7*k1	c4=b2	d4=c4	e4=d2+d7	x4=e2-e5
a5=y5+y3	t2=a6*k2	c5=t0+t3	d5=c3-c4	e5=d6-d2	x5=e3-e4
a6=y5-y3	t3=a7*k2	c6=t5-t6	d6=c5-c7	e6=d1-d7	x6=e1-e6
a7=y1-y7	t4=a3*k3	c7=b3+b4	d7=c7	e7=d7	x7=e0-e7
	t5=a4*k3				
	t6=a5*k3				
k1=2*cos(2*pi/16)		k2=2*cos(6*pi/16)		k3=2*cos(4*pi/16)	



Rys. 5. Schemat blokowy obrazujący sposób wykorzystania zasobów do realizacji zmodyfikowanego algorytmu IDCT
Fig. 5. Block diagram illustrating use of resources to implement the modified IDCT algorithm

Moduł dekodera Huffmana działa w oparciu o automat, którego zadaniem jest zapisanie informacji z nagłówka zawierających sposób dekodowania poszczególnych komponentów, a następnie sukcesywne wypełnianie bufora danymi do dekodowania. Proces dekodowania polega na odpowiedniej interpretacji pierwszych 16 bitów zawartych w buforze. Na podstawie ilości wiodących jedynek wybierana jest dana tablica, z której odczytywana jest szukana wartość. Dzięki odpowiedniej organizacji tablic nie jest wymagane nakładanie żadnej maski (w zależności od długości słowa). W trakcie jednego taktu zegara następuje zdekodowanie wartości run/size i długości słowa kodowego, a na ich podstawie przesunięcie danych w głównym buforze. W kolejnym takcie następuje kolejna iteracja dekodowania, a na podstawie wcześniej uzyskanych danych obliczana jest wartość końcowa przekazywana do modułu dekwantyzacji. Równocześnie uruchomiona jest seria liczników, których zadaniem jest wybieranie odpowiednich adresów z pamięci RAM, a także wyznaczanie współrzędnych kolejnych wartości wyjściowych. Logika kontrolująca cały moduł odpowiada za wykrywanie przepełnienia buforu wejściowego, a także wykrywanie kodów specjalnych, takich jak na przykład EOB (koniec danego bloku).

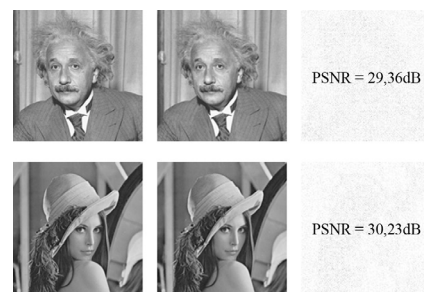


Rys. 6. Schemat przedstawiający sposób działania modułu dekodera Huffmana
Fig. 6. Diagram showing how the Huffman decoder module works

7. Wyniki implementacji

Tab. 2. Wyniki implementacji modułów dla kości xc6vlx240t-2
Tab. 2. Implementation results of the module for xc6vlx240t-2 system

Modul	Slice Registers	Slice LUT
IDCT	578	625
Huffman	247	2330
PICO Framework	15800	8800
Available	301440	150720



Rys. 7. Podgląd efektów działania dekompresora na podstawie obrazów Einstein i Lena (8bit). Kolejne kolumny przedstawiają: obraz wejściowy, wynik wyjściowy, bezwzględna wartość różnicy pomiędzy wejściem i wyjściem
Fig. 7. Decompressor work on the basis of Einstein and Lena pictures (8bit)

8. Wnioski

Przedstawiona w niniejszej pracy implementacja dekodera JPEG działa w pełni synchronicznie w układzie programowalnym FPGA. Sposób połączenia modułów umożliwia wprowadzanie strumienia danych w trybie ciągłym. Szczególną uwagę poświęcono wyborowi i implementacji dwóch najważniejszych algorytmów występujących w tym standardzie.

Praca finansowana ze środków na działalność statutową katedry Elektroniki.

9. Literatura

[1] Dąbrowska-Boruch A.: Implementacja w układach FPGA kodera obrazów w standardzie MPEG-2 spełniającego wymogi czasu rzeczywistego, Kraków 2007.
[2] Arai Y., Agui T., Nakajima M.: A Fast DCT-SQ Scheme for Images, Transactions of the IEICE.E 71(11):1095, Nov. 1988.
[3] Mansour M.F.: Efficient Huffman Decoding with Table Lookup, ICASSP, 2007.
[4] http://www.picocomputing.com
[5] http://www.xilinx.com

otrzymano / received: 02.04.2012
przyjęto do druku / accepted: 01.06.2012 *artykuł recenzowany / revised paper*