Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki Informatyki i Elektroniki
Katedra Elektroniki

Agnieszka Dąbrowska

Implementacja w układach FPGA kodeka obrazów w standardzie MPEG-2 spełniającego wymogi czasu rzeczywistego

Rozprawa doktorska

Promotor:
prof. dr hab. inż. Kazimierz Wiatr

Kraków 2007

Moim Rodzicom

Serdeczne podziękowania dla
Profesora Kazimierza Wiatra,
Którego uśmiech i Dobre Słowo
Były najlepszą motywacją.

Serdeczne podziękowania wszystkim,
Którzy swoją cierpliwością i życzliwością
Przyczynili się do powstania tej rozprawy.

Spis treści

Wykaz skrótów

AC - współczynnik DCT dla składowych zmiennych (tzn. o współrzędnych niezerowych)

ALU - jednostka arytmetczno-logiczna (ang. *Arithmetric-Logic Unit*)

ASIC - układy scalone projektowane do ściśle określonych zastosowań, zgodnie ze specyfikacją użytkownika (ang. *Application Specific Integrated Circuits*)

BPDC - suma różnicy pikseli (ang. *Difference Pixel Count*)

BPM - dopasowanie płaszczyzny bitowej (ang. *Bit-Plane Matching*)

BPROP - kryterium binarnego poziomu dopasowania (ang. *Binary Level Matching Criterion*)

CCIR 656 - międzynarodowy standard definiujący interfejs telewizji cyfrowej pod względem elektrycznym i mechanicznym. CCIR 656 jest praktyczną implementacją standardu ITU-R Rec. 601

CCM - dedykowane dla użytkownika struktury obliczeniowe (ang. *Custom Computing Machines*)

CPU - jednostka centralna procesora (ang. *Central Processor Unit*)

CSD - kanoniczna reprezentacja cyfry ze znakiem (ang. *Canonic Sign Digit*)

DC - współczynnik DCT dla składowej stałej (tzn. o współrzędnej 0, 0)

DCT - dyskretna transformata kosinusowa (ang. *Discrete Cosine Transform*)

DFT - dyskretna transformata Fouriera (ang. *Discrete Fourier Transform*)

DPC - suma różnicy pikseli (ang. *Difference Pixel Count*)

DSP - cyfrowy procesor sygnałowy (ang. *Digital Signal Processor*)

E3SS - efektywny algorytm trójkrokowego przeszukiwania (ang. *Efficient Three-Step Search*)

FDCT - szybki algorytm dyskretnej transformacji kosinusowej (ang. *Fast DCT*)

FF - przerzutnik Flip-Flop

FPGA - układy programowalne o architekturze matrycowej (ang. *Field Programmable Gate Array*)

FS - algorytm pełnego przeszukiwania (ang. *Full Search*)

GOP - grupa obrazów (ang. *Group Of Pictures*).

GPP - procesor ogólnego przeznaczenia (ang. *General Purpose Processor*)

HDTV - telewizja wysokiej rozdzielczości (ang. *High Definition TV*)

HP - profil wysoki (ang. *High Profile*)

IDCT - odwrotna dyskretna transformata kosinusowa (ang. *Inverse Discrete Cosine Transform*)

IP-Core - elementy biblioteczne o określonej funkcjonalności (ang. *Intellectual Property Core*)

IQUANT - dekwantyzacja (ang. *Dequantisation*)

JPEG - standard kompresji obrazów statycznych (ang. *Joint Photographic Experts Group*)

KLT - transformata Karhunena-Loévego (ang. *Karhunen-Loéve Transform*)

LUT - tablica spełniająca rolę tablicy wartości funkcji logicznej (ang *Look-Up-Table*)

MAD - średnia różnica bezwzględna (ang. *Mean Absolute Difference*)

MAE - średni błąd bezwzględny (ang. *Mean Absolute Error*)

MC - kompensacja ruchu (ang. *Motion Compensation*)

MD - maksymalna różnica (ang. *Maximal Difference*)

ME - estymacja ruchu (ang. *Motion Estimation*)

MM - bezmnożny układ mnożenia (ang. *Multiplierless Multiplication*)

MP - profil główny (ang. *Main Profile*)

MPEG - standard kompresji obrazów ruchomych (ang. *Moving Picture Experts Group*)

MSD - różnica średniokwadratowa (ang. *Mean-Squared Difference*)

MVP - profil wielowidokowy (ang. *Multi-View Profile*)

NTSC - amerykański system kodowania obrazu telewizyjnego (ang. *National Television System Committee*)

NTSS - nowy algorytm trójkrokowego przeszukiwania (ang. *New Three-Step Search*)

PAL - europejski system kodowania obrazu telewizyjnego (ang. *Phase Alternating Line*)

PDC - klasyfikacji różnicy pikseli (ang. *Pixel Difference Classification*)

PHODS - algorytm równoległego jednowymiarowego hierarchicznego przeszukiwania

(ang. *Parallel Hierarchical One-Dimensional Search*)

PSNR - szczytowy stosunek sygnału do szumu (ang. *Peak Signal-to-Noise-Ratio*)

QUANT - kwantyzacja (ang. *Quantisation*)

RAM - pamięć o swobodnym dostępie (ang. *Random Access Memory*)

RLC - kodowanie długości ciągu (ang. *Run Length Coding*)

RLD - dekodowanie długości ciągu (ang. *Run Length Decoding*)

ROM - pamięć tylko do odczytu (ang. *Read Only Memory*)

SDTV - telewizja standardowej rozdzielczości (ang. *Standard Definition TV*)

SNR - stosunek sygnału do szumu (ang. *Signal-to-Noise Ratio*)

SP - profil prosty (ang. *Simple Profile*)

SS - układ mnożący ze wspólną podstrukturą (ang. *Sub-structure Sharing*)

TAP - kontroler standardu IEEE 1149.1 (ang. *Test Access Port*)

TDL - algorytm dwuwymiarowego przeszukiwania logarytmicznego (ang. *Two-Dimentional Logarithmic Search*)

TSS - algorytm trójkrokowego przeszukiwania (ang. *Three-Step Search*)

VLC - kodowanie ze zmienną długością słowa (ang. *Variable Length Coding*)

VLD - dekodowanie ze zmienną długością słowa (ang. *Variable Length Decoding*)

WHT - transformata Walsha-Hadamarda (ang. *Walsh-Hadamard Transform*)

SIPO - rejestr z szeregowym wejściem i równoległym wyjściem (ang. *Serial In Parallel Out*)

PISO - rejestr z równoległym wejściem i szeregowym wyjściem (ang. *Parallel In Serial Out*)

1. Wstęp

Przekaz wizyjny jest obecnie jednym z najważniejszych środków komunikowania się i przekazu wiedzy. W erze rozkwitu radiofonii nośnikiem informacji był dźwięk. Obecnie już od wielu lat nastała era telewizji, a najpowszechniejszym sposobem przekazywania informacji stał się obraz. W erze internetu, przy takim sposobie przekazywania informacji i wiedzy, pojawia się wiele nowych problemów. Pierwszym z nich jest przepustowość bitowa medium transmisyjnego. Kolejnymi problemami jest poprawność otrzymanej informacji oraz wyjątkowo duża objętość przesyłanych danych. Próbą rozwiązania tych problemów jest kompresja obrazów. Wprowadzenie kompresji obrazów było koniecznie zarówno ze względów technicznych jak również ze względów ekonomicznych.

Standard MPEG-2 jest obecnie jednym z najpowszechniej używanych standardów kompresji sekwencji obrazów wizyjnych. Najczęściej wykorzystywany jest on w telewizji cyfrowej. Standard ten jest także stosowany w kamerach cyfrowych, na płytach DVD, w tunerach cyfrowych itp.

Algorytmy kompresji obrazów wymagają wykonywania dużej liczby operacji na dużej liczbie danych. W przypadku kompresji obrazów nieruchomych (przykładowo w kompresji JPEG) czas potrzebny na kompresję pojedynczego obrazu nie jest parametrem krytycznym. Dla kompresji obrazów ruchomych czas potrzebny na kompresję pojedynczej klatki zaczyna odgrywać coraz większą rolę, zwłaszcza, jeżeli układ musi pracować w trybie czasu rzeczywistego. W przypadku rejestracji sekwencji obrazów najważniejszym zagadnieniem jest możliwość zapisu danych bez zauważalnych opóźnień. Jeśli dane przed zapisem są kompresowane to wymagane jest, aby kompresja odbywała się w czasie rzeczywistym. Oznacza to, że jednosekundowa sekwencja obrazów byłaby kompresowana w ciągu jednej sekundy z niewielkim opóźnieniem między danymi wejściowymi i danymi wyjściowymi.

Platformą technologiczną, której zastosowanie może sprostać wymaganiom pracy w czasie rzeczywistym są układy programowalne FPGA (ang. *Field Programmable Gate Array*). Są to układy elastyczne, podobnie jak procesory ogólnego przeznaczenia GPP (ang.

General Purpose Processor) i procesory sygnałowe DSP (ang. *Digital Signal Processor*). W porównaniu z procesorami GPP układy FPGA nie wymagają jednostki, która przyjmowałaby rozkazy z zewnątrz oraz jednostki dekodującej te rozkazy. Patrząc z perspektywy rozwiązań software'owych, układ FPGA jest rozwiązaniem czysto sprzętowym, a do prawidłowego jego działania wymagany jest strumień danych oraz sygnały sterujące operacjami wykonywanymi na tym strumieniu. Funkcje oferowane przez układ FPGA są określone w logice programowalnej układu. W procesorach GPP i DSP oprócz wejściowego strumienia danych wymagane jest także dostarczenie programu z instrukcjami, jakie operacje należy wykonać na danych wejściowych. Zatem rozwiązania oparte na tych procesorach są rozwiązaniami software'owymi. Innymi układami, które oferują sprzętowe realizacje algorytmów są układy ASIC (ang. *Application Specific Integrated Circuits*). Jednak wadą układów ASIC jest ich wąska specjalizacja, ustalona na etapie ich produkcji. Układy te są projektowane dla konkretnych zastosowań, natomiast układy FPGA po rekonfiguracji w dowolnym momencie mogą wykonywać nowe operacje na strumieniu danych wejściowych. Właśnie częściowa lub całkowita rekonfiguracja, bez potrzeby zaprzestania pracy całego układu, jest atutem obecnych układów FPGA. Światowym liderem wśród producentów układów FPGA jest firma Xilinx. W trakcie badań zostały wykorzystane układy programowalne z rodzin: Virtex, Virtex-II PRO oraz Virtex-4. Wszystkie te układy, poza zasobami programowalnymi, zawierają także wbudowane bloki pamięci RAM. Dodatkowo układy Virtex-II PRO posiadają wbudowane układy mnożące dwie liczby 18-bitowe ze znakiem, bloki zarządzające zegarem oraz do czterech procesorów PowerPC. Podobnie jak Virtex-II PRO, układy Virtex-4 są wyposażone w bloki zarządzania zegarem i procesory PowerPC. Ponadto układy Virtex-4 mają wbudowane bloki XtremeDSP. Taki pojedynczy blok XtremeDSP zbudowany jest z 18-bitowej mnożarki, układu dodającego oraz akumulatora. Układy Virtex pracują z częstotliwością do 200MHz, a układy z rodzin Virtex-II PRO i Virtex-4 pracują z jeszcze większym zegarem systemowym.

W trakcie badań wykorzystane zostały pakiety narzędziowe ISE w wersji 6.3i oraz 8.1i. Pakiety te, opracowane przez firmę Xilinx, zawierają narzędzia programowe wspomagające projektowanie, testowanie oraz uruchamianie aplikacji dla układów programowalnych.

Celem pracy jest skuteczna implementacja systemu kompresji i dekompresji sekwencji obrazów wizyjnych w standardzie MPEG-2 pracującego w trybie czasu rzeczywistego, z wykorzystaniem technologii układów programowalnych FPGA.

W wyniku prowadzonych badań, autor zamierza udowodnić następującą tezę pracy:
Odpowiednia modyfikacja algorytmów pozwala na zaimplementowanie w układach FPGA kodeka standardu MPEG-2, wykorzystującego w kompresji obrazy typu I, P oraz B, spełniającego wymogi czasu rzeczywistego i kodującego obrazy o rozdzielczości HDTV.

W rozdziale 2 zostały opisane podstawowe algorytmy składowe stosowane w kompresji i dekompresji obrazów, stanowiące części składowe kodera i dekodera standardu MPEG-2. W szczególności są to algorytmy: dyskretna transformacja kosinusowa, odwrotna dyskretna transformacja kosinusowa i kodowanie Huffmana. Ponadto w rozdziale tym zawarte zostały podstawowe wiadomości o estymacji i kompensacji ruchu oraz istniejące funkcje celu, będące podstawowym elementem decydującym w procesie poszukiwania wektorów ruchu.

W rozdziale 3 zawarty jest opis funkcjonalności oraz parametrów standardu ISO/IEC 13818-2. Opisane zostały również struktury obowiązujące przy kompresji i dekompresji zgodnej ze standardem MPEG-2. Rozdział ten zawiera opis tworzenia obrazów typu I, P i B.

Rozdział 4 zawiera krótki opis przykładowych dotychczasowych realizacji standardu MPEG-2. Opisane zostały przykłady koderów, dekoderów oraz kodeków standardu ISO/IEC 13818-2. Są to rozwiązania sprzętowe, zrealizowane w postaci specjalizowanych układów scalonych ASIC oraz oparte na układach z logiką programowalną. Rozdział ten zawiera również opis rozwiązań software'owych wykorzystujących procesory ogólnego przeznaczenia GPP i DSP.

Rozdział 5 zawiera opis algorytmów dyskretnej transformacji kosinusowej. Ponadto rozdział ten prezentuje różne aspekty sprzętowej implementacji tych bloków oraz otrzymane parametry. W rozdziale tym omówiono także zaproponowane przez autora możliwe modyfikacje istniejących już algorytmów. Modyfikacje te mają na celu przystosowanie tych algorytmów do ich sprzętowej implementacji w układach programowalnych FPGA.

W rozdziale 6 zawarty jest szczegółowy opis operacji kwantyzacji i dekwantyzacji w standardzie MPEG-2 z uwzględnieniem parametrów sterujących tymi operacjami. Ponadto rozdział prezentuje możliwe sposoby implementacji tych operacji.

W rozdziale 7 opisane są podstawowe algorytmy estymacji ruchu. Dodatkowo zaprezentowane są implementacje dwóch wybranych algorytmów estymacji ruchu. Dopełnieniem tego rozdziału jest opis, proponowanych przez autora, modyfikacji algorytmu efektywnego trójkrokowego przeszukiwania.

Rozdział 8 prezentuje algorytmy bezstratnej kompresji obrazów wizyjnych zastosowane w koderach i dekoderach standardu MPEG-2. Ponadto rozdział ten zawiera parametry implementacji kodera oraz dekodera entropijnego w układzie FPGA.

Rozdział 9 zawiera opis kolejnych wersji koderów i kodeków standardu MPEG-2 zaimplementowanych w układzie FPGA, jak również porównanie otrzymanych wyników z istniejącymi już rozwiązaniami zaprezentowanymi w rozdziale 3.

W rozdziale 10 dokonano podsumowania wyników, ze szczególnym uwzględnieniem osiągnięć własnych autora pracy.

Zamiarem autora pracy było, aby realizując przedstawione w rozprawie prace badawcze z jednej strony zrealizować kompletny system kodowania i dekodowania obrazów ruchomych. Z drugiej strony dążono, aby realizacja tego zadania przy pomocy układów programowalnych FPGA pozwalała na pełną rekonfigurację systemu, umożliwiając w ten sposób zmianę parametrów przetwarzania obrazów, a nawet implementację w zasobach programowalnych FPGA całkowicie innych zadań. Należy się spodziewać, że już w nieodległej przyszłości, nawet komputer osobisty będzie wyposażony w rekonfigurowalny koprocesor sprzętowy, programowany za pomocą bibliotek zapisanych na twardym dysku. Koprocesor taki będzie mógł realizować zadania kompresji i dekompresji obrazów, na przykład przy współpracy z internetem, lub całkowicie inne złożone zadania obliczeniowe. Z tego względu praca ta ma być istotnym przyczynkiem zarówno do dedykowanych sprzętowych maszyn obliczeniowych CCM (ang. *Custom Computing Machines*) jak również do praktyki codziennego użytkowania komputera.

2. Podstawowe funkcje wykorzystywane w systemach kompresji obrazów ruchomych

Nieskompresowany cyfrowy sygnał wizyjny wymaga niezmiernie dużej szerokości pasma transmisyjnego. Przykładowo obraz cyfrowy w standardzie NTSC wymaga szybkości transmisji około 100 Mb/s. Cyfrowe sekwencje obrazów wymagają kompresji, w celu redukcji szybkości transmisji, aby można je było użyć w większości aplikacji. Wymagany stopień kompresji osiągany jest dzięki przestrzennej i czasowej redundancji w sygnale wizyjnym. Zastosowany proces kompresji jest najczęściej procesem stratnym, a zatem sygnał zrekonstruowany, otrzymany z zakodowanego strumienia bitowego, będzie się różnił nieznacznie od sygnału wejściowego.

Tab. 2.1. Zależności pomiędzy różnymi standardami reprezentacji obrazu, a rozdzielczością obrazu oraz przepływnościami bitowymi przed i po kompresji [32]

	Rozdzielczość obrazu [liczba pikseli×linii×ramek/s]	Przepływność bitowa obrazu (RGB) nieskompresowanego	Przepływność bitowa obrazu skompresowanego
NTSC	480×480×29,97	168 Mb/s	4 do 8 Mb/s
PAL	576×576×25	199 Mb/s	4 do 9 Mb/s
HDTV	1920×1080×30	1493 Mb/s	18 do 30 Mb/s
HDTV	1280×720×60	1327 Mb/s	18 do 30 Mb/s
CIF (ISDN)	352×288×29,97	73 Mb/s	64 do 1920 kb/s
QCIF (PTSN)	176×144×29,97	18 Mb/s	10 do 30 kb/s

2.1. Dyskretna transformacja kosinusowa

2.1.1. Kodowanie transformatowe

Obraz o wymiarach $L_x \times L_y$ pikseli, kodowany transformatowo, jest dzielony na nienakładające się na siebie bloki o wymiarach $N \times N$ pikseli. Każdy taki blok jest opisywany przez odwracalną transformatę, której jądro opisuje zbiór ortonormalnych funkcji bazowych. Celem zastosowania liniowej transformaty jest dekorelacja oryginalnego sygnału, co powoduje rozdzielenie energii sygnału pomiędzy małą liczbę składowych (mały zbiór współczynników). W ten sposób, wiele współczynników zostaje odrzuconych po kwantyzacji. Dzięki temu działaniu otrzymujemy zbiór danych o małej entropii. Tak przetransformowany sygnał można poddać dalszej kompresji bezstratnej, otrzymując jeszcze lepszy współczynnik kompresji.

Wśród technik kodowania transformatowego najczęściej w praktyce stosowane są metody: dyskretna transformata Fouriera DFT (ang. *Discrete Fourier Transform*), dyskretna transformata kosinusowa DCT (ang. *Discrete Cosine Transform*), transformata Karhunena-Loévego KLT (ang. *Karhunen-Loéve Transform*) oraz transformata Walsha-Hadamarda WHT (ang. *Walsh-Hadamard Transform*).

2.1.2. Dyskretna ortonormalna transformata liniowa

Jeśli spróbkowany obraz zostanie podzielony na nie zachodzące na siebie bloki o rozmiarach $1 \times N$ pikseli (1 piksel pionowo, N pikseli poziomo) i wartości pikseli takiego pojedynczego bloku zapisane zostaną w postaci wektora $\boldsymbol{b}$ o długości N to ortonormalna transformata liniowa będzie miała postać (2.1), a odwrotna transformata będzie reprezentowana przez zależność (2.2).

$$\boldsymbol{c} = \boldsymbol{T}\boldsymbol{b} \tag{2.1}$$

$$\boldsymbol{b} = \boldsymbol{T}'\boldsymbol{c} \tag{2.2}$$

gdzie: $\boldsymbol{T}$ - macierz transformaty $N \times N$,

$\boldsymbol{T}'$ - macierz transformaty odwrotnej $N \times N$,

$\boldsymbol{c}$ - wektor współczynników transformaty.

Jeżeli *m*-ta kolumna macierzy $\boldsymbol{T}'$ oznaczona zostanie jako $\boldsymbol{t}_m$ i spełniony jest warunek (2.3), to wtedy wektor $\boldsymbol{t}_m$ jest ortonormalnym wektorem bazowym macierzy transformaty $\boldsymbol{T}$.

$$\boldsymbol{t}'_m \boldsymbol{t}_n = \delta_{mn} \tag{2.3}$$

gdzie: $\delta_{mn} = 1$ jeśli $m=n$,

w przeciwnym razie $\delta_{mn} = 0$.

Zatem wektor $\boldsymbol{b}$ można wyrazić zależnością (2.4).

$$\boldsymbol{b} = \sum_{m=1}^{N} c_m \boldsymbol{t}_m \tag{2.4}$$

gdzie c_m jest *m*-tym elementem wektora $\boldsymbol{c}$.

W przypadku, gdy spróbkowany obraz zostanie podzielony na bloki o rozmiarach $N \times N$ pikseli, pojedynczy blok będzie reprezentowany przez macierz $\boldsymbol{B} = [b_{ij}]$, transformacja będzie realizowana w dwóch etapach. Pierwszy etap stanowić będzie transformacja wykorzystująca wiersze macierzy $\boldsymbol{B}$ o długości *N*. Celem takiego działania jest wykorzystanie horyzontalnej korelacji w przetwarzanym obrazie. W drugim etapie udział wezmą kolumny macierzy $\boldsymbol{B}$ w celu wykorzystania wertykalnej korelacji obrazu. Taka kombinacja operacji może zostać zapisana w postaci (2.5).

$$c_{mn} = \sum_{i=1}^{N} \sum_{j=1}^{N} b_{ij} t_{nj} t_{mi} \tag{2.5}$$

gdzie: t_{nj}, t_{mi} są elementami macierzy $\boldsymbol{T}$ transformaty,

$\boldsymbol{C} = [c_{mn}]$ jest macierzą współczynników transformaty o rozmiarze $N \times N$.

2.1.3. Dyskretna transformata kosinusowa (DCT)

Jedną z metod kodowania transformatowego jest metoda wykorzystująca dyskretną transformatę kosinusową DCT. Najważniejszą cechą tej transformaty jest zdolność skupiania energii, co oznacza , że większość energii zawarta jest w kilkunastu początkowych

współczynnikach. W związku z takim rozkładem energii pozostałe współczynniki można zastąpić zerami, co wpływa na poprawę efektywności kodowania.

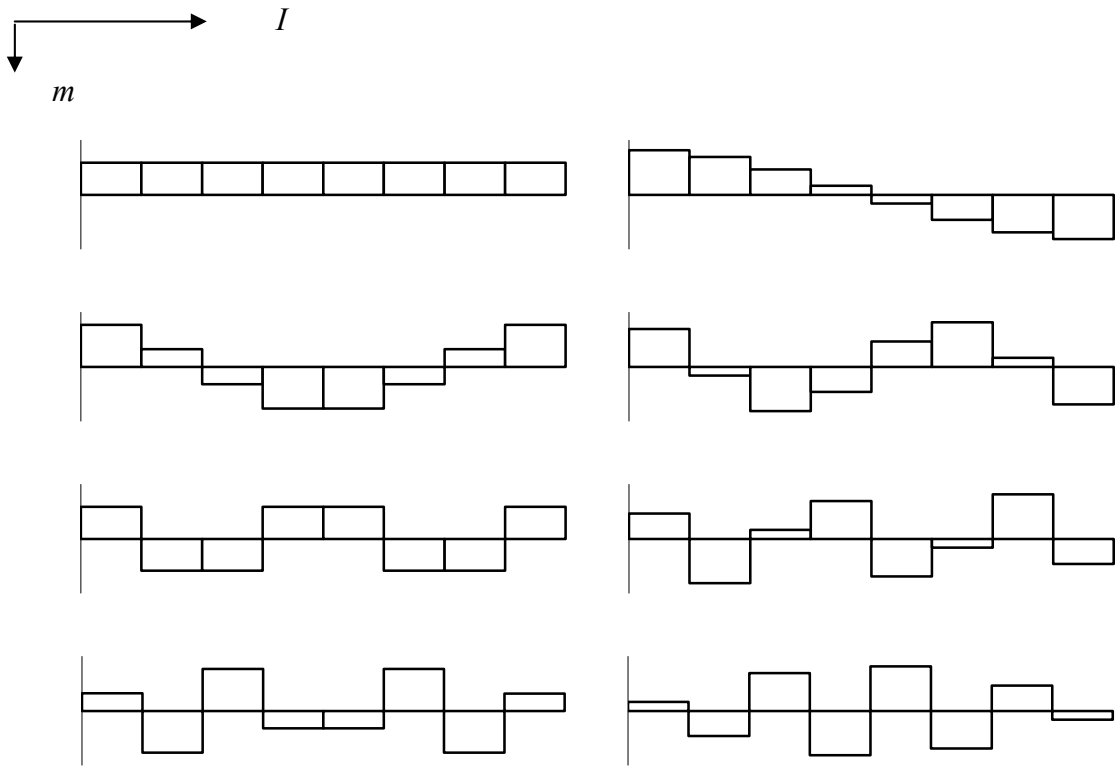
Elementy macierzy $\boldsymbol{T}$ transformaty DCT są określane przez zależność (2.6).

$$t_{mi} = \sqrt{\frac{2-\delta_{m-q}}{N}} \cos\left\{\frac{\pi}{N}\left(i-\frac{1}{2}\right)(m-1)\right\} \qquad i,m = 1\ldots N \tag{2.6}$$

gdzie: $\delta_0 = 1$,

$\delta_p = 0$ dla $p > 0$.

W zależności (2.6) można zauważyć, że wektory bazowe t_m są sinusoidami, których częstotliwości są reprezentowane przez indeks m. Przykładowo dla N=8 wektory bazowe pokazane są na rysunku 2.1.



Rys. 2.1. Wektory bazowe dyskretnej transformaty kosinusowej dla N=8

2.2. Kodowanie Huffmana

Kodowanie metodą Huffmana jest zaliczane do metod entropijnych, inaczej nazywanych także metodami ze zmienną długością słowa.

W kodowaniu entropijnym symbolom często występującym przypisuje się krótkie słowa kodowe, natomiast symbolom rzadko występującym długie słowa kodowe.

Tab. 2.2. Przykładowy proces redukcji źródła przy kodowaniu Huffmana

Oryginalne źródło		Zredukowane źródło krok 1		Zredukowane źródło krok 2		Zredukowane źródło krok 3	
s_i	$p(s_i)$	s_i'	$p(s_i')$	s_i''	$p(s_i'')$	s_i'''	$p(s_i''')$
s_1	0,40	$s_1'(s_1)$	0,40	$s_1''(s_1')$	0,40	$s_1'''(s_2'', s_3'')$	0,60
s_2	0,20	$s_2'(s_4, s_5)$ lub $s_2'(s_3, s_5)$	0,25	$s_2''(s_3', s_4')$	0,35	$s_2'''(s_1'')$	0,40
s_3	0,15	$s_3'(s_2)$	0,20	$s_3''(s_2')$	0,25		
s_4	0,15	$s_4'(s_3)$ lub $s_4'(s_4)$	0,15				
s_5	0,10						

Kodowanie Huffmana jest oparte na rozkładzie prawdopodobieństwa występowania symboli w ciągu kodowanych danych. Pierwszym krokiem tego algorytmu kodowania jest wyznaczenie dla poszczególnych symboli ($s_1,...,s_N$) prawdopodobieństwa ich występowania ($p_1,...,p_N$). W kolejnym etapie, tzw. procesie redukcji źródła, krokiem podstawowym jest zastępowanie dwóch symboli o najmniejszych prawdopodobieństwach przez nowy symbol o prawdopodobieństwie równym sumie prawdopodobieństw symboli składowych. Krok ten jest powtarzany do momentu, gdy pozostaną tylko dwa symbole (tab. 2.2).

Kolejnym etapem algorytmu jest proces konstrukcji słów kodowych. Jest on rozpoczynany od dwóch symboli, będących wynikiem procesu redukcji źródła. Symbolowi o mniejszym prawdopodobieństwie przyporządkowujemy "1", natomiast symbolowi o większym prawdopodobieństwie przyporządkowujemy "0". Przy procesie konstrukcji słów kodowych

pokonujemy tę samą "drogę" jak w procesie redukcji źródła tylko w odwrotnym kierunku, rozkładając symbole skonstruowane na symbole, z których one powstały (tab. 2.3).

Tab. 2.3. Przykładowy proces konstrukcji słów kodowych w kodowaniu Huffmana

Oryginalne źródło krok 3		Zredukowane źródło krok 2		Zredukowane źródło krok 1		Zredukowane źródło	
s_i	Słowo kodowe	s_i'	Słowo kodowe	s_i''	Słowo kodowe	s_i'''	Słowo kodowe
s_1	1	$s_1'(s_1)$	1	$s_1''(s_1')$	1	$s_1'''(s_2'', s_3'')$	0
s_2	000	$s_2'(s_4, s_5)$ lub $s_2'(s_3, s_5)$	01	$s_2''(s_3', s_4')$	00	$s_2'''(s_1'')$	1
s_3	001 lub 010	$s_3'(s_2)$	000	$s_3''(s_2')$	01		
s_4	010 lub 001	$s_4'(s_3)$ lub $s_4'(s_4)$	001				
s_5	011						

Kodowanie Huffmana jest kodowaniem jednoznacznym pod warunkiem, że podczas przetwarzania nie wystąpią równocześnie symbole o tym samym prawdopodobieństwie.

Na efektywność kodowania omawianej metody ma wpływ częstotliwość występowania symboli. Efektywność można oszacować poprzez obliczenie średniej długości słowa kodowego, określanego zależnością (2.7).

$$\overline{L} = \sum_{i=1}^{N} p(s_i) l(s_i) \tag{2.7}$$

gdzie : $p(s_i)$ - prawdopodobieństwo wystąpienia symbolu s_i,

$l(s_i)$ - długość binarnej reprezentacji symbolu s_i,

$\overline{L}$ - średnia długość słowa kodowego.

Im bardziej średnia długość słowa kodowego jest zbliżona do entropii źródła, obliczanej z zależności (2.8), tym efektywniejsze jest kodowanie metodą Huffmana.

$$H = -\sum_{i=1}^{N} p(s_i) \log_2 p(s_i) \tag{2.8}$$

gdzie: H - entropia źródła.

Zdarza się, że słowa kodowe osiągają znaczne długości, co zmniejsza efektywność tej metody. Dlatego stosuje się pewne modyfikacje kodowania Huffmana. Jedną z takich modyfikacji jest kod Huffmana z ograniczeniem długości. Niestety ograniczenie długości słowa kodowego powoduje pogorszenie stopnia kompresji.

2.3. Estymacja i kompensacja ruchu

2.3.1. Wprowadzenie

W sekwecji obrazów oprócz redundancji przestrzennej, występuje również redundancja czasowa. Do redukcji redundancji czasowej używany jest proces kompensacji ruchu. Najprościej proces ten można zdefiniować jako kompensację przesunięcia poruszającego się obiektu, czyli różnicy między położeniem obiektu w ramce obecnej a położeniem tego samego obiektu w ramce następnej. Proces kompensacji ruchu jest poprzedzany przez estymację ruchu – proces, poprzez który znajdowane są odpowiadające sobie piksele z sąsiednich ramek. Dane wyjściowe otrzymane w wyniku procesu kompensacji ruchu można wyrazić zależnością (2.9).

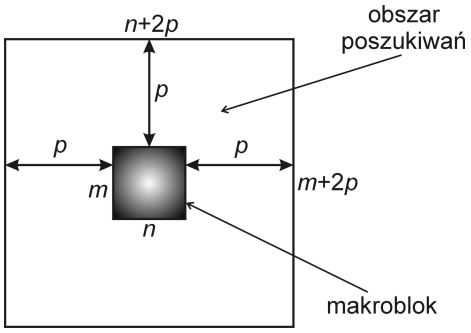
$$e(x, y, t) = I(x, y, t) - I(x - u, y - v, t - 1) \tag{2.9}$$

gdzie: $I(x,y,t)$ – wartość piksela w punkcie o współrzędnych (x,y) w ramce t,
$I(x - u,y - v,t - 1)$ – wartość najbardziej pasującego piksela w punkcie o współrzędnych $(x – u,y – v)$ w ramce $t – 1$.

Estymacja ruchu jest to proces wyznaczania wektorów przesunięcia obszarów między obrazem bieżącym a obrazem referencyjnym. Można wyróżnić cztery rodzaje metod estymacji ruchu: gradientowe, rekurencyjne na poziomie pikseli, dopasowywania blokowego oraz metody w dziedzinie częstotliwości. W analizie sekwencji obrazów najczęściej stosowane są metody gradientowe. Podzbiorem metod gradientowych są metody rekurencyjne na poziomie pikseli. Metody te używane są, jeżeli wektory przesunięcia zmieniają się od jednego piksela do drugiego. Metody te w dziedzinie częstotliwości oparte są na powiązaniu

między transformowanymi współczynnikami przesuniętych obrazów. Najpowszechniej używanymi metodami estymacji ruchu są metody dopasowywania blokowego.

Danymi wyjściowymi procesu estymacji ruchu są współrzędne (*dx*,*dy*), definiujące względny ruch bloku między ramką obecną a ramką poprzednią. Otrzymane współrzędne określają wektor ruchu bloku mieszczącego się pod współrzędnymi przestrzennymi (*x*,*y*) – najczęściej są to współrzędne określające górny lewy róg danego bloku.



Rys. 2.2. Obszar przeszukiwań w algorytmach estymacji ruchu

W idealnym przypadku zakres poszukiwania takiego bloku jest nieograniczony. Jednak w rzeczywistości byłoby to niepraktyczne, dlatego zakres przeszukiwań (Rys. 2.2) został ograniczony do obszaru wokół danego bloku w zakresie [-*p*, *p*]. Zatem współrzędne wektora ruchu będą ograniczone warunkami (2.10) i (2.11).

$$-p \le dx \le p \tag{2.10}$$

$$-p \le dy \le p \tag{2.11}$$

Najczęściej w standardach kodowania MPEG, H.261 i H.263 stosowane są wartości współczynników: m=n=16 oraz p=6.

Jedną z najczęściej używanych funkcji celu jest różnica średniokwadratowa *MSD* (*ang. Mean-Squared Difference*). Funkcję tę opisuje wyrażenie (2.12).

$$MSD(dx, dy) = \frac{1}{mn} \sum_{i=-n/2}^{n/2} \sum_{j=-m/2}^{m/2} \left[F(i, j) - G(i + dx, j + dy)\right]^2 \tag{2.12}$$

gdzie: $F(i, j)$ - kompresowany makroblok,

$G(i, j)$ - referencyjny makroblok,

(dx, dy) - wektor przesunięcia,

$dx = \{-p, p\}, dy = \{-p, p\}$.

MSD jest funkcją dającą najdokładniejsze wyniki, ale równocześnie najbardziej złożoną pod względem obliczeniowym. Realizacja funkcji *MSD* wymaga wykorzystania mnożenia każdego piksela makrobloku.

Inną popularną funkcją celu jest średnia różnica bezwzględna *MAD* (*ang. Mean Absolute Diffrence*) reprezentowana przez wyrażenie (2.13). W literaturze [30] funkcję tę nazywa się także jako średni błąd bezwzględny *MAE* (ang. *Mean Absolute Error*)

$$MAD(dx, dy) = \frac{1}{mn} \sum_{i=-n/2}^{n/2} \sum_{j=-m/2}^{m/2} \left| F(i, j) - G(i + dx, j + dy) \right| \tag{2.13}$$

Funkcja *MAD* charakteryzuje się mniejszą złożonością obliczeniową niż *MSD* lecz również jest mniej dokładna.

2.3.2. Funkcje celu o małej złożoności obliczeniowej

Kolejną funkcją celu jest funkcja klasyfikacji różnicy pikseli *PDC* (*ang. Pixel Difference Classification*). *PDC* (2.14) charakteryzuje się mniejszą złożonością obliczeniową niż *MSD* oraz *MAD*.

$$PDC(dx, dy) = \sum_i \sum_j T(dx, dy, i, j) \;\; dla \;\; (dx, dy) = \{-p, p\} \tag{2.14}$$

Funkcja $T(dx, dy, i, j)$ jest binarną reprezentacją różnicy pomiędzy pikselami, definiowaną za pomocą wyrażenia (2.15)

$$T(dx, dy, i, j) = \begin{cases} 1 \longrightarrow \left| F(i, j) - G(i + dx, j + dy) \right| \le t \\ 0 \longrightarrow w \quad przeciwnym \quad przypadku \end{cases} \tag{2.15}$$

gdzie: t – predefiniowana wartość progowa.

Przy tak zdefiniowanej funkcji celu, blokiem najbardziej dopasowanym jest blok z maksymalną wartością funkcji *PDC*.

Przy założeniu, że $t = 2^{t_1}$ otrzymuje się binarną reprezentację *PDC* wyrażoną zależnością (2.16).

$$BPDC(dx,dy) = \sum_i \sum_j and\{xnor(F_{t_1}(i,j), G_{t_1}(i+dx, j+dy))\} \tag{2.16}$$

gdzie: F_{t_1} oraz G_{t_1} są 8-t_1 najbardziej znaczącymi bitami *F* oraz *G*.

Inną funkcją celu o małej złożoności obliczeniowej jest kryterium binarnego poziomu dopasowania *BPROP* (ang. *Binary Level Matching Criterion*) wyrażone zależnością (2.17).

$$BPROP(dx,dy) = \sum_i \sum_j xor(F_{t_1}(i,j), G_{t_1}(i+dx, j+dy)) \tag{2.17}$$

Jeśli funkcja *BPROP* osiągnie minimum w punkcie (*dx*,*dy*) to wektor ruchu będzie określany przez wartości współrzędnych tego punktu. Przy $t = 2^{t_1} = 16$ otrzymywany jest najkorzystniejszy stosunek sygnału do szumu, o 0,5-1dB mniejszy niż w przypadku funkcji *MAE*. Kolejną funkcją oparta na *BPDC* jest suma różnicy pikseli *DPC* (ang. *Difference Pixel Count*) wyrażona zależnością (2.18).

$$DPC(dx,dy) = \sum_i \sum_j and\{xnor(F_q(i,j), G_q(i+dx, j+dy))\} \tag{2.18}$$

gdzie: F_q oraz G_q są skwantowanymi wartościami odpowiednio *F* oraz *G*.

Najlepszy kompromis pomiędzy dokładnością a złożonością obliczeniową funkcja *DPC* osiąga w wyniku kwantyzacji i wówczas pojedynczy piksel będzie reprezentowany przez 2 bity zamiast przez 8 bitów. Jeżeli piksel będzie reprezentowany przez *p*(*dx*,*dy*) to skwantowaną wartość piksela można wyznaczyć na podstawie zależności (2.19).

$$p_q(dx,dy) = \begin{cases} 01 & \Leftarrow \quad p(dx,dy) \geq t \\ 00 & \Leftarrow \quad t > p(dx,dy) \geq 0 \\ 11 & \Leftarrow \quad 0 > p(dx,dy) \geq -t \\ 10 & \Leftarrow \quad p(dx,dy) < -t \end{cases} \tag{2.19}$$

przy czym wartości *t* są określane za pomocą wyrażenia (2.20)

$$t = \frac{3}{2mn} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \left| p(dx, dy) - l \right| \tag{2.20}$$

gdzie: *l* oznacza środek bloku.

Inną funkcją celu stosowaną w procesie estymacji ruchu jest dopasowanie płaszczyzny bitowej *BPM* (ang. *Bit-plane Matching*) określone zależnością (2.21).

$$BPM(dx, dy) = \frac{1}{mn} \sum_{i} \sum_{j} xor(\hat{F}(i, j), \hat{G}(i + dx, j + dy)) \tag{2.21}$$

gdzie: $\hat{F}$ oraz $\hat{G}$ reprezentują wartości pikseli po transformacji do jednobitowych ramek.

Transformacja taka przebiega w następujący sposób. Jeżeli jako *C* zostanie oznaczona ramka oraz *R* będzie ramką *C* po konwolucji to wartość $\hat{C}$ można wyznaczyć za pomocą (2.22).

$$\hat{C}(dx, dy) = \begin{cases} 1 \Leftarrow & C(dx, dy) \geq R(dx, dy) \\ 0 \Leftarrow & w \ \ przeciwnym \ \ przypadku \end{cases} \tag{2.22}$$

Osiągany stosunek sygnału do szumu w przypadku funkcji jest porównywalny z PSNR otrzymanym przy pomocy funkcji *BPROP* oraz *BPDC*.

3. Opis kompresji obrazów w standardzie MPEG-2

Nazwa MPEG pochodzi od nazwy komitetu *Moving Picture Experts Group* powołanego do życia przez ISO i IEC w 1988 roku w celu opracowania standardu kodowania obrazów ruchomych. Pierwszym, opracowanym w 1991 roku standardem był MPEG-1. Standard ten zawarty jest w normie ISO/IEC 11172 i jest stosowany do aplikacji, w których transmisja ma prędkość bitową około 1,5 Mb/s (z czego na transmisję danych wizyjnych przeznaczono 1,15 Mb/s). Aby sprostać takiej transmisji w standardzie MPEG-1 rozdzielczość obrazu wynosiła 352×240 pikseli dla systemu NTSC lub 352×288 pikseli dla systemu PAL. Kolejnym standardem opracowanym w 1995 roku przez wspomniany komitet był MPEG-2 zawarty w normie ISO/IEC 13818. Podjęto również próby stworzenia standardu MPEG-3 do zastosowania w HDTV, jednak do tego celu równie dobrze nadaje się MPEG-2. W związku z tym prace nad MPEG-3 zostały zawieszone.

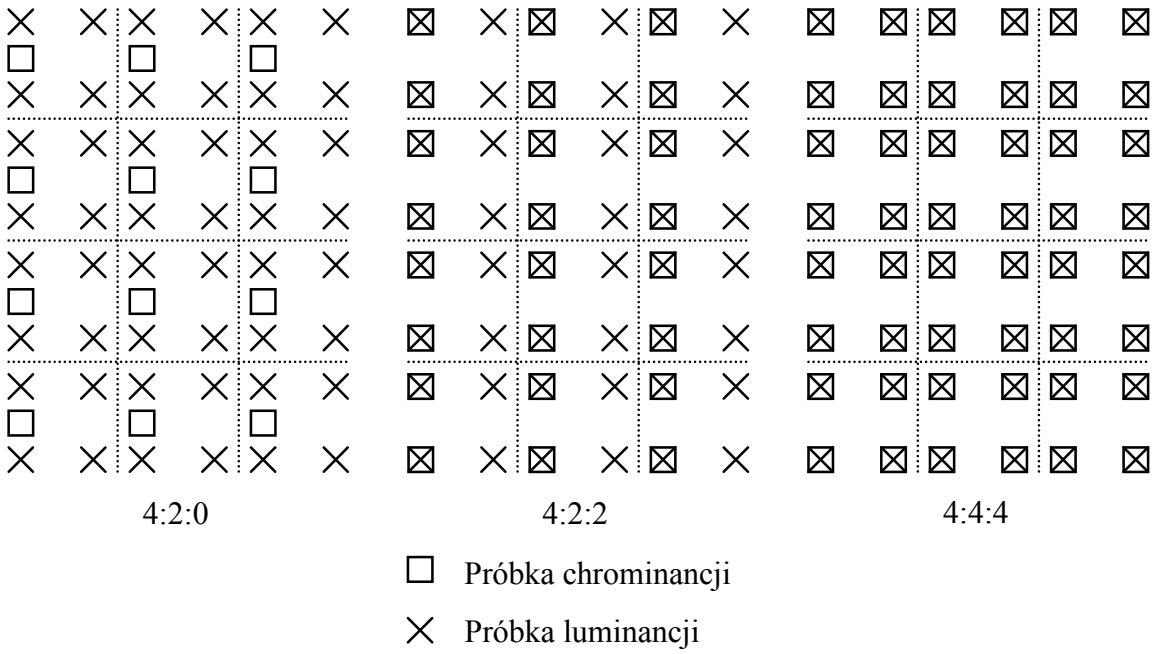
Specyfikacja standardu MPEG-2 składa się z następujących części: systemy (norma 13818-1), sygnały wizyjne (norma 13818-2), sygnały audio (norma 13818-3), testy zgodności (norma 13818-4), oprogramowanie symulacyjne (norma 13818-5), cyfrowe media zapisu - rozkazy i sterowanie – DSM-CC (norma 13818-6), zaawansowane kodowanie sygnałów audio - AAC (norma 13818-7), 10-bitowa reprezentacja obrazu (norma 13818-8), interfejs czasu rzeczywistego (norma 13818-9).

Standard MPEG-2, oznaczany czasem jako H.262, znajduje swoje zastosowanie przy przepływnościach bitowych powyżej 2 Mbitów/s. Górna granica przepływności bitowej, osiągalnej przez ten standard, wynosi aż 429 Gbitów/s. Jednak MPEG-2 jest zoptymalizowany dla przepływności bitowej równej 4 Mbity/s oraz dla obrazu z przeplotem.

Podstawowym wymaganiem postawionym przed standardem MPEG-2 jest osiągnięcie jak największego stopnia kompresji oraz możliwość osiągnięcia jak największej jakości dekodowanej sekwencji wizyjnej przy założonej szybkości transmisji. Dodatkowo aplikacje

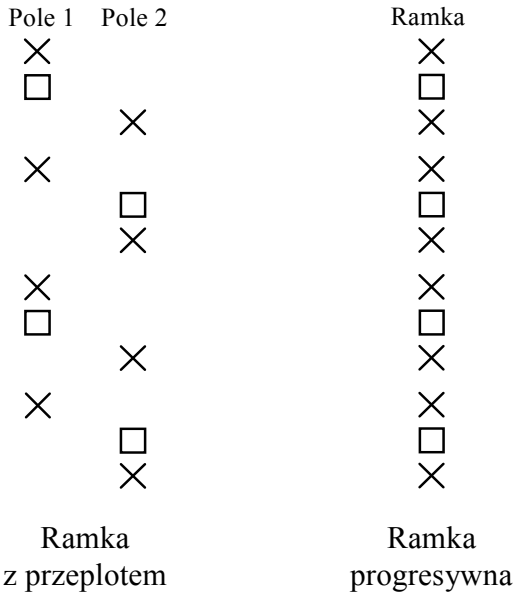
multimedialne powinny mieć zapewniony dostęp i możliwość zdekodowania pojedynczych obrazów ze strumienia bitowego. Również zdolność do wykonywania szybkich i bezpośrednich przeszukiwań strumienia bitowego, zarówno w przód jak i wstecz, jest niezmiernie wskazana, jeśli medium transmisyjne ma małe możliwości. Ponadto wymagana jest kompatybilność ze standardem MPEG-1, możliwość swobodnego dostępu do odpowiedniej sekwencji, skalowalność strumienia bitowego oraz małe opóźnienie dla komunikacji dupleksowej. MPEG-2 nie standaryzuje kodowania strumienia wizyjnego, standaryzowana jest natomiast składnia strumienia wizyjnego oraz semantyka dekodowania.

W MPEG-2 dopuszczone są trzy formaty próbkowania chrominancji i luminancji (Rys. 3.1).

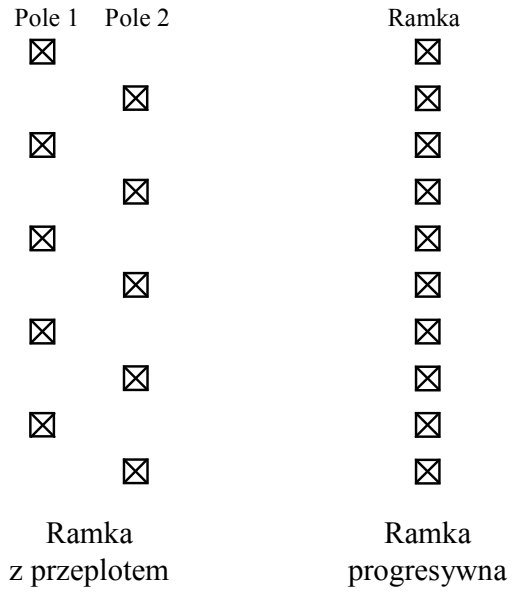


Rys. 3.1. Formaty próbkowania chrominancji i luminancji w MPEG-2

Pierwszy z tych formatów, czyli 4:2:0, był jako jedyny dopuszczalny w standardzie MPEG-1, oznaczany czasem w literaturze jako 4:1:1. Pozostałe formaty to 4:2:2 i 4:4:4. Format 4:2:2 jest używany w aplikacjach wymagających większej rozdzielczości chrominancji. W standardzie MPEG-2 istnieje kilka sposobów reprezentacji każdego z formatów w zależności od tego czy przychodzące ramki są ramkami progresywnymi czy też ramkami z przeplotem. Przykłady takich reprezentacji ramki w formatach 4:2:0 oraz 4:2:2 przedstawiają odpowiednio rysunki 3.2 i 3.3.

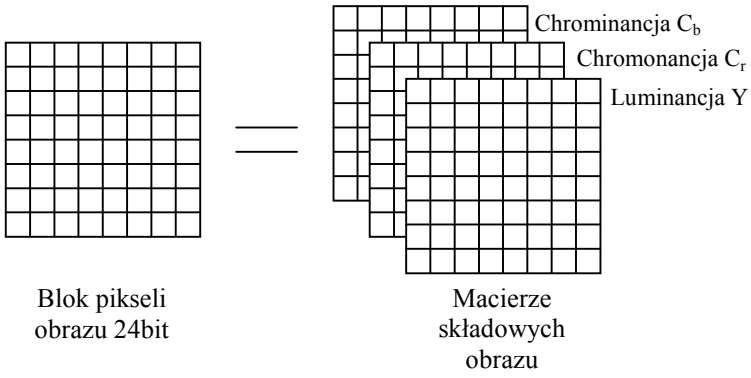


Rys. 3.2. Rozmieszczenie pikseli w formacie 4:2:0



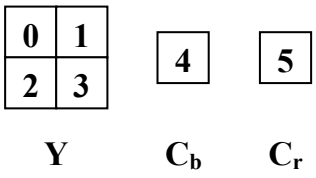
Rys. 3.3. Rozmieszczenie pikseli w formacie 4:2:2

Najmniejszą strukturą jaka jest przetwarzana w procesie kompresji jest blok złożony z 64 pikseli. Budowę bloku przy 24-bitowej reprezentacji pikseli przedstawia rysunek 3.4.

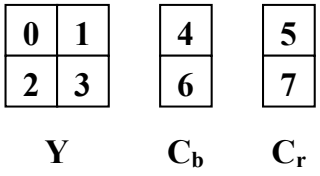


Rys. 3.4. Struktura bloku przy 24-bitowej reprezentacji pikseli

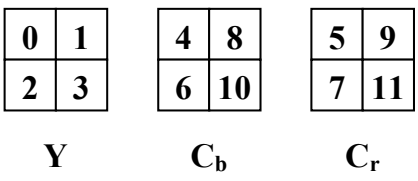
Większą strukturą, składającą się z kilku lub kilkunastu bloków o rozmiarach 8×8 pikseli, jest makroblok. Każdy makroblok można podzielić na dwie części. Jedną z nich stanowią 4 bloki zawierające wartości luminancji poszczególnych pikseli. Druga część makrobloku składa się z bloków zawierających wartości chrominancji, przestrzennie odpowiadających blokom luminancji z pierwszej części makrobloku. W zależności od formatu makrobloki zawierają różną liczbę bloków poszczególnych składowych chrominancji – od 4 do 8 bloków. Struktury makrobloku w zależności od zastosowanego formatu przedstawiają rysunki 3.5, 3.6 i 3.7.



Rys. 3.5. Struktura makrobloku w formacie 4:2:0

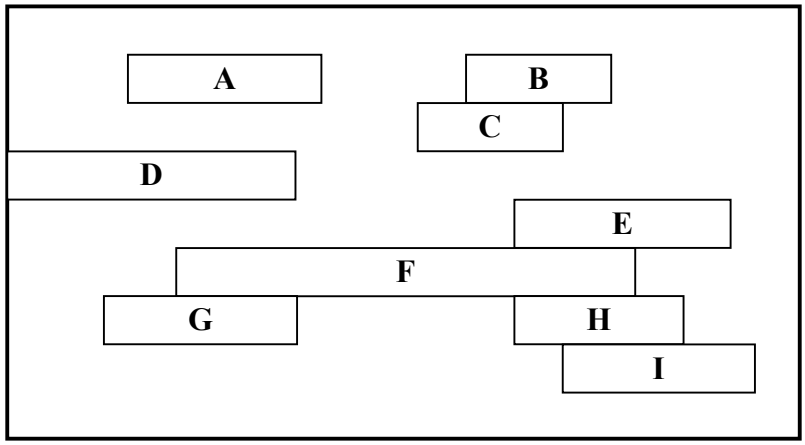


Rys. 3.6. Struktura makrobloku w formacie 4:2:2

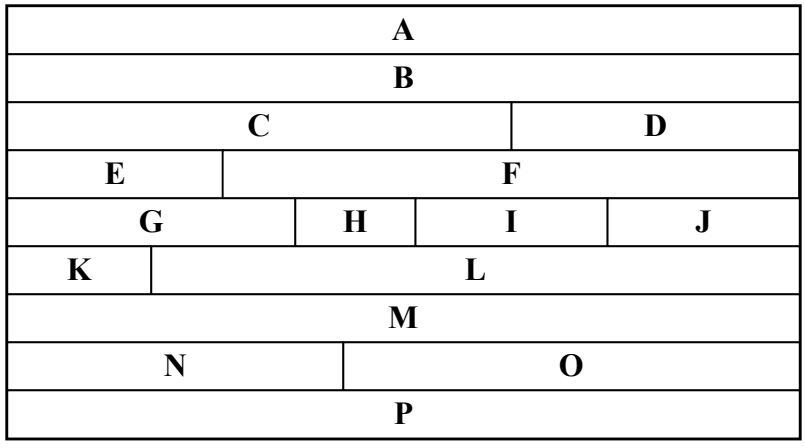


Rys. 3.7. Struktura makrobloku w formacie 4:4:4

Większą jednostką od makrobloku jest struktura zwana slice'm. Pojedynczy slice składa się z dowolnej liczby kolejnych makrobloków. W związku z tym, że w standardzie MPEG-2 istnieje możliwość pomijania makrobloków, slice powinien składać się z co najmniej jednego makrobloku. Zalecane jest jednak, aby w skład tej struktury wchodziły co najmniej dwa makrobloki: pierwszy i ostatni. Ponadto poszczególne struktury slice nie mogą na siebie nachodzić. Pojedynczy slice powinien zaczynać się i kończyć w tej samej linii poziomej makrobloków, lecz nie musi zawierać całej linii. Przykładowe struktury złożone ze struktur slice przedstawiają rysunki 3.8 i 3.9.



Rys. 3.8. Typowa konfiguracja struktur slice

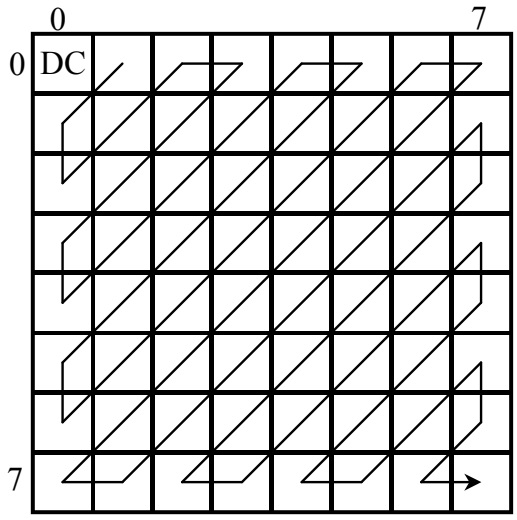


Rys. 3.9. Restrykcyjna konfiguracja struktur slice

Metoda kompresji zastosowana w standardzie MPEG-2 jest kombinacją wcześniej stosowanych standardów: JPEG oraz H.261. Ponieważ sygnał wizyjny może być traktowany jak sekwencja nieruchomych obrazów, możliwe jest zastosowanie technik kompresji podobnych jak w przypadku standardu JPEG. Takie techniki kompresji nazywane są

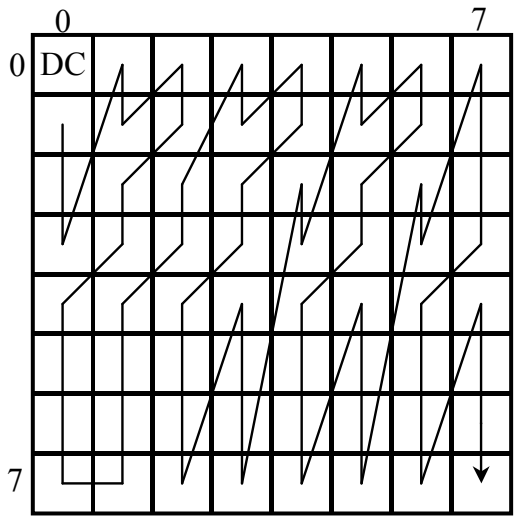
metodami kodowania wewnątrzobrazowego, gdzie każdy obraz jest niezależnie kompresowany i kodowany. Metody kodowania wewnątrzobrazowego wykorzystują przestrzenną redundancję istniejącą pomiędzy sąsiednimi pikselami w danym obrazie. Tak zakodowane obrazy są to tak zwane obrazy typu I (*Intraframe* lub *Intrapicture*). Obrazy tego typu zapewniają swobodny dostęp do sekwencji. Dodatkowo obrazy I można stosować przy zmianie sceny oraz w innych przypadkach, w których kompensacja obrazu jest nieefektywna.

W ramach podobieństwa MPEG-2 do JPEG i H.261, algorytm kompresji bazuje na dwuwymiarowej dyskretnej transformacie kosinusowej. Zatem w pierwszej fazie przetwarzany obraz zostanie podzielony na rozłączne bloki o rozmiarze 8×8 pikseli, a dwuwymiarowa transformata DCT jest stosowana niezależnie dla każdego takiego bloku. Efektem takiego postępowania będzie powstanie bloków współczynników transformacji DCT również o rozmiarze 8×8, w których większość energii jest reprezentowana przez zaledwie kilka współczynników. Obecna w standardzie JPEG, podobnie jak w MPEG-1, metoda Zig-Zag (Rys. 3.10) przestała być metodą optymalną, zatem w standardzie MPEG-2 zdefiniowano tak zwaną alternatywną metodę uporządkowania (Rys. 3.11), chociaż jest stosowana również metoda Zig-Zag.



Rys. 3.10. Działanie metody Zig-Zag dla bloku o rozmiarze 8×8 pikseli w trybie kodowania sekwencyjnego

Kolejnym krokiem jest kwantyzacja współczynników związanych ze składowymi AC transformacji według zależności (3.1). Współczynniki dotyczące składowej stałej DC są kwantowane ze stałym krokiem równym 8.



Rys. 3.11. Alternatywa metoda uporządkowania współczynników transformacji DCT w standardzie MPEG-2

$$FQ_{uv} = \frac{8c_{uv}}{qQ[u,v]} \qquad \text{dla } u,v \neq 0 \tag{3.1}$$

gdzie: c_{uv} - współczynniki transformacji DCT,

FQ_{uv} - skwantowane współczynniki transformacji DCT,

$Q[u,v]$ - macierz kwantyzacji,

q - współczynnik skalowania kwantowania.

Podobnie jak w standardzie JPEG również w MPEG-2 są określone typowe macierze kwantyzacji dla współczynników transformacji, uwzględniające własności systemu wzrokowego człowieka. Macierze kwantyzacji można adaptacyjnie skalować za pomocą współczynnika q podawanego indywidualnie dla obrazu, warstwy lub makrobloku.

W wyniku kwantowania wiele współczynników ma wartość zero a kompresja jest osiągana poprzez transmisję tylko niezerowych współczynników oraz kodowanie entropijne ich amplitudy i zajmowanej pozycji.

Aby umożliwić swobodny dostęp do dowolnego punktu sekwencji, obrazy I powinny pojawiać się stosunkowo często. Jednak w związku z tym, że obrazy tego typu nie wykorzystują korelacji czasowej ich stopień kompresji jest mały. Aby zwiększyć efektywność kompresji takiej sekwencji należy wstawić pomiędzy dwa kolejne obrazy typu I kilka obrazów innego typu. Zbyt duża liczba takich dodatkowych obrazów powoduje utratę swobodnego dostępu do dowolnego punktu sekwencji.

$$\begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 49 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix}$$

a) dla trybu wewnątrzobrazowego

$$\begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix}$$

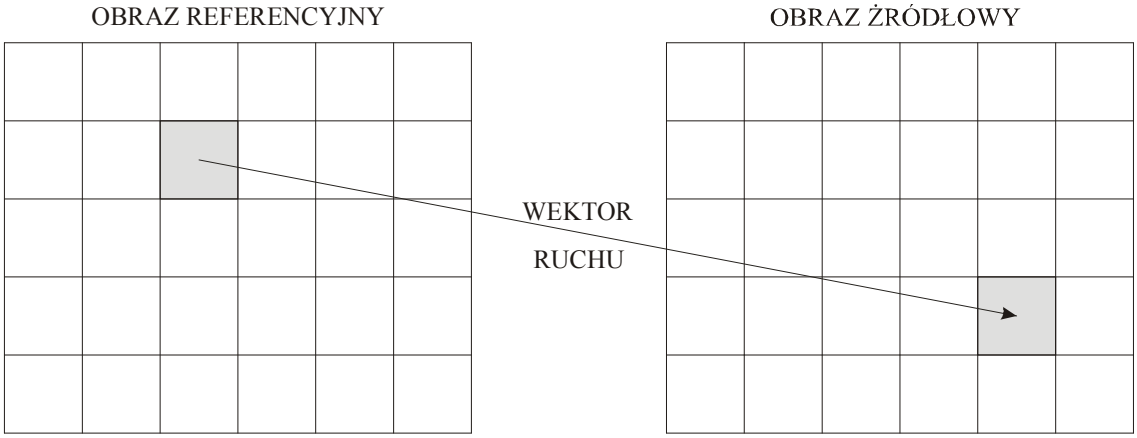
b) dla trybu międzyobrazowego

Rys. 3.12. Typowe macierze kwantyzacji dla współczynników transformacji kodowanych wewnątrzobrazowo (a) i międzyobrazowo (b) w standardzie MPEG-2

W standardzie MPEG-2 występują jeszcze dwa inne typy obrazów a mianowicie obrazy typu P - zakodowane predykcyjnie (ang. *Predictive coded pictures*) oraz obrazy typu B - zakodowane predykcyjnie dwukierunkowo (ang. *Bidirectinally-predictive coded pictures*).

Oprócz przestrzennej redundancji występuje również redundancja czasowa wynikająca z wysokiego stopnia korelacji pomiędzy kolejnymi obrazami. Algorytm standardu MPEG-2 wykorzystuje redundancję czasową do obliczenia błędu predykcji, czyli różnicy pomiędzy sąsiednimi obrazami. Przy wyznaczaniu błędu predykcji używana jest technika kompensacji ruchu oparta na przesunięciach makrobloków. Przy jednokierunkowej estymacji ruchu, nazywanej też predykcją wprzód, następuje przeszukanie obrazu poprzednio przetworzonego w celu odnalezienia zbioru bloków najbardziej podobnych do makrobloku z obrazu obecnie przetwarzanego. Jeśli taki zbiór zostanie odnaleziony to będzie on wykorzystany jako makroblok predykcyjny. Wtedy błąd predykcji będzie różnicą pomiędzy makroblokiem

z obecnego obrazu a makroblokiem predykcyjnym. Natomiast przesunięcie pomiędzy pozycjami tych makrobloków będzie określone za pomocą wektora ruchu (Rys. 3.13).



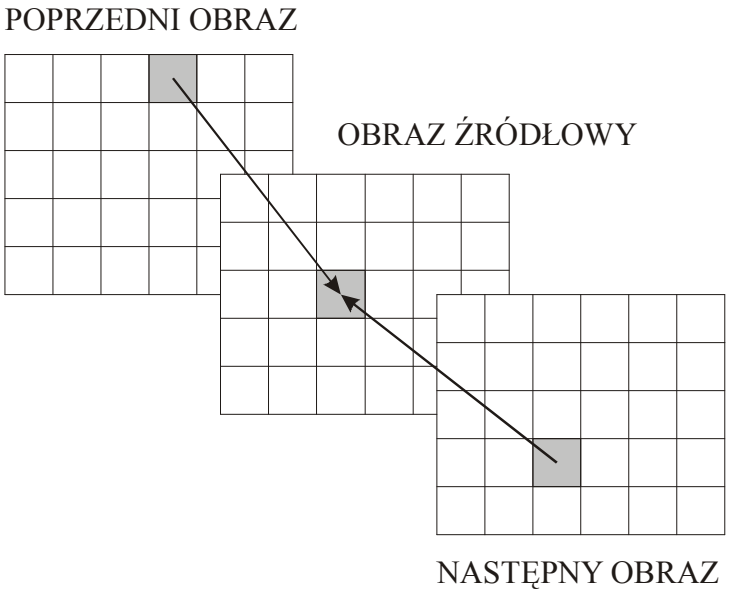
Rys. 3.13. Kompensacja ruchu przy użyciu predykcji

Zarówno błąd predykcji jak i informacje o wektorze ruchu są kodowane i transmitowane. Obrazy, które używają do kodowania takiej właśnie predykcji nazywane są obrazami typu P. Efektywność kompresji obrazów tego typu jest z reguły większa niż w przypadku obrazów typu I. Niestety nie może to zrekompensować spadku stopnia kompresji wynikającego z częstego użycia obrazów I. W związku z tym w standardzie MPEG istnieje trzeci rodzaj obrazów. Są to obrazy typu B oparte na predykcji dwukierunkowej, nazywanej inaczej interpolacją kompensacji ruchu. Do kodowania za pomocą predykcji dwukierunkowej potrzebne są dwa obrazy odniesienia, jeden poprzedni i jeden następny. Podobnie jak przy obrazach typu P również w tym przypadku następuje etap poszukiwania zbioru najbardziej podobnych bloków. Różnicą jest to, że przeszukiwany jest zarówno obraz poprzedni jak i następny. Zatem dla jednego makrobloku istnieć będą dwa wektory ruchu (Rys. 3.14).

Obrazy typu B nie mogą być wykorzystywane jako obrazy odniesienia dla innych obrazów. Natomiast dla obrazów tego typu jako obrazy referencyjne wykorzystywane są obrazy I oraz obrazy P. Główną zaletą zastosowania predykcji dwukierunkowej jest większy stopień kompresji niż przy zastosowaniu predykcji opartej tylko na poprzednim obrazie.

Standard MPEG-2 ze względu na możliwość zastosowania przeplotu określa dwie struktury obrazu: ramkową i polową. W obrazie ramkowym (ang. *Frame Picture*) ramka tworzona jest z pary pól jeszcze przed procesem kodowania. Obraz polowy (ang. *Field Picture*) składa się z dwóch pól, które są kodowane niezależnie. Jeżeli pierwsze z pól zostało zakodowane jako obraz typu P, to drugie pole również musi być zakodowane jako obraz typu

P. Podobnie jest w przypadku pól zakodowanych jako obrazy typu B. Jeżeli pierwsze z pól zostanie zakodowane jako obraz typu I to drugie może zostać zakodowane jako obraz typu I lub P.



Rys. 3.14. Kompensacja ruchu przy użyciu predykcji dwukierunkowej

Obrazy typu P i B są tworzone przy zastosowaniu metod redukcji redundancji czasowej, takich jak w MPEG-1, w połączeniu z metodami opartymi na przeplocie.

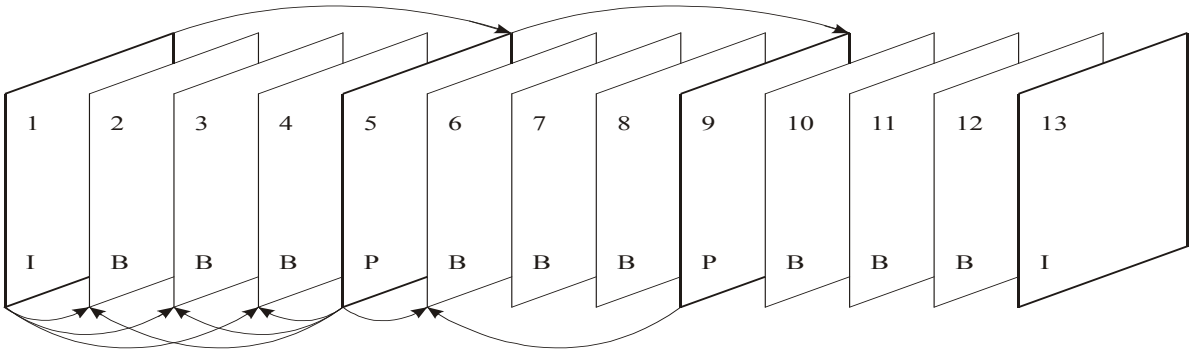
Pierwszą z takich metod jest predykcja ramkowa stosowana przy obrazach ramkowych. W przypadku predykcji wprzód tworzony jest jeden wektor ruchu dla danego makrobloku, a dla predykcji dwukierunkowej wyznaczane są dwa wektory dla makrobloku. Predykcja wykorzystuje poprzednio kodowane ramki referencyjne. Predykcja ramkowa jest stosowana wówczas, jeśli ruch jest wolny lub umiarkowany i nie może być zastosowana dla obrazów polowych. Oprócz predykcji ramkowej występuje również predykcja polowa. Może być ona używana zarówno dla obrazów polowych jak i dla obrazów ramkowych. Predykcja polowa dla obrazów polowych jest podobna do predykcji ramkowej dla obrazów ramkowych – z założeniem, że wszystkie piksele przetwarzanego makrobloku należą do tego samego pola. Makrobloki referencyjne mogą również należeć do tego samego pola co makroblok przetwarzany lub mogą należeć do drugiego pola ramki obrazu. Dodatkowo dla obrazów typu P makrobloki referencyjne pochodzą z ostatnio zakodowanych pól typu I lub typu P. W przypadku obrazów typu B makrobloki muszą pochodzić z ostatnio zakodowanych ramek typu I lub P.

Trzecim rodzajem predykcji jest predykcja polowa dla obrazów ramkowych. W tym trybie predykcji przetwarzany makroblok jest dzielony na dwa obszary: pole górne i pole dolne. Każde z tych pól jest następnie przetwarzane niezależnie, co oznacza, że przy predykcji wprzód będą tworzone dwa wektory ruchu (a nie jeden jak poprzednio), a dla predykcji dwukierunkowej będą już cztery wektory ruchu zamiast dwóch. Predykcja polowa dla obrazów ramkowych jest stosowana, jeśli ruch jest szybki.

Kolejnym rodzajem predykcji jest tak zwana predykcja *dual prime* dla obrazów typu P. W trybie tym transmitowany jest jeden wektor ruchu na makroblok. Na podstawie tego jednego wektora są wyliczane dwie wstępne predykcje, które składają się na końcową predykcję. Pierwsza z wstępnych predykcji jest niemalże identyczna z predykcją polową. Jedyna różnica polega na tym, że w przypadku predykcji *dual prime* wszystkie piksele pochodzące z poprzednio zakodowanego pola muszą być tej samej polaryzacji (muszą pochodzić z tego samego rodzaju pola: górnego lub dolnego), co piksele obecnie przetwarzane. Druga z wstępnych predykcji jest wyznaczana przy pomocy wyznaczonego wektora ruchu oraz dodatkowego małego wektora korekcji ruchu. W tym przypadku piksele używane w predykcji muszą pochodzić z pola o przeciwnej polaryzacji niż przy pierwszej wstępnej predykcji.

Piąty typ predykcji jest predykcja 16×8 MC dla obrazów polowych. Pierwszą czynnością w tym modzie jest podział makrobloków obrazu polowego na dwie połowy: górną i dolną. Następnie każda z tych części jest poddawana niezależnie predykcji polowej. W wyniku tego dla makrobloków w obrazie typu P transmitowane są dwa wektory ruchu (podobnie jak w przypadku predykcji polowej dla obrazów ramkowych), a dla makrobloków w obrazach typu B transmitowane są cztery wektory.

Obrazy są organizowane w grupy obrazów GOP (*ang. Group Of Pictures*). GOP jest najmniejszą strukturą umożliwiającą swobodny dostęp do sekwencji obrazów (Rys. 3.15).



Rys. 3.15. Przykładowy układ grupy obrazów GOP w kodowaniu MPEG-2

W strukturze GOP musi być co najmniej jeden obraz typu I. Może on zajmować pozycję pierwszą lub może być poprzedzony obrazami typu B, które używają predykcji bazującej na tym obrazie typu I. Typowa sekwencja obrazów uporządkowana w kolejności wyświetlenia obrazów jest przedstawiona w tabeli 3.1.

Tab. 3.1. Typowa sekwencja obrazów wyświetlanych w standardzie MPEG

Typ obrazu	I	B	B	P	B	B	P	B	B	P	B	B	I
Pozycja w sekwencji	1	2	3	4	5	6	7	8	9	10	11	12	13

Tab. 3.2. Typowa sekwencja obrazów w strumieniu bitowym w standardzie MPEG

Typ obrazu	I	P	B	B	P	B	B	P	B	B	I	B	B
Pozycja w sekwencji wyświetlanej	1	4	2	3	7	5	6	10	8	9	13	11	12

Kolejność obrazów w strumieniu bitowym (Tab. 3.2) jest inna niż kolejność obrazów wyświetlanych. Pierwszą pozycję, tak jak w sekwencji obrazów wyświetlanych, zajmuje obraz typu I, lecz jako drugi transmitowany jest obraz typu P, zajmujący pozycję 4 w sekwencji wyświetlanej. Trzecią pozycję w strumieniu bitowym zajmuje obraz B z pozycji 2 a na czwartym jest również obraz typu B, lecz z pozycji 3. Następnie do kompresji trafia obraz typu P z pozycji 7. Potem kolejno pojawiają się obrazy B odpowiednio z pozycji 5 i 6. Następnie transmitowany jest obraz P z pozycji 10, a po nim wystąpią obrazy typu B z pozycji 8 i 9. Ostatnią pozycję w sekwencji wyświetlanej zajmuje obraz I, natomiast w strumieniu bitowym będzie on na pozycji 11. Na ostatnich dwóch pozycjach w sekwencji strumienia bitowego zajmują obrazy typu B.

Różnica pomiędzy sekwencją obrazów w strumieniu bitowym a sekwencją obrazów wyświetlanych wynika z zależności pomiędzy obrazami typu B, a ich obrazami referencyjnymi. Obrazy w sekwencji wyświetlanej ułożone są w kolejności w jakiej odbiera je użytkownik. Natomiast porządek obrazów w strumieniu bitowym jest uwarunkowany zależnościami pomiędzy obrazami. Dlatego obrazy typu I oraz typu P transmitowane są przed utworzonymi na ich podstawie obrazami typu B.

Kolejną cechą standardu MPEG-2 jest podział strumienia wejściowego na warstwy: warstwę podstawową (ang. *base layer*) i jedną lub dwie warstwy rozszerzeń (ang. *enhancement layer*). Warstwa podstawowa zawiera dane pozwalające odtworzyć starszym urządzeniom obrazy o gorszej jakości. Warstwa rozszerzeń jest pomocna przy odtwarzaniu

obrazów o lepszych parametrach w nowszych urządzeniach. W zależności od zawartości warstwy podstawowej zostały wyróżnione cztery rodzaje skalowalności: przestrzenna (ang. *spatial scalability*), *SNR* (ang. *SNR scalability*), czasowa (ang. *temporal scalability*) oraz podział danych (ang. *data partitioning*). W przypadku skalowalności przestrzennej, w warstwie podstawowej znajdują się dane potrzebne do odtworzenia obrazów o najmniejszej (podstawowej) rozdzielczości przestrzennej. Natomiast warstwa rozszerzeń będzie zawierała informację pomocną przy odtworzeniu obrazów o większej rozdzielczości. Warstwy mogą zawierać ramki o różnych rozmiarach, różnej częstotliwości wyświetlania ramek oraz różny format chrominancji. Przy skalowalności według SNR obie warstwy zawierają informacje dla obrazów o takiej samej rozdzielczości, ale różnią się dokładnością kodowania tych informacji. Warstwa podstawowa niesie z sobą obraz kodowany z podstawową jakością. Przy skalowalności czasowej, warstwa podstawowa zawiera informacje o sekwencji z mniejszą liczbą obrazów na sekundę. W takim przypadku warstwa rozszerzeń niesie z sobą informacje o dodatkowych obrazach w sekwencji. Przy podziale danych w warstwie podstawowej znajdują się między innymi informacje o wektorach ruchu i współczynniki transformaty DCT odpowiadające małym częstotliwościom. Pozostała część współczynników jest zawarta w warstwie rozszerzeń.

Ponieważ nie wszystkie aplikacje standardu MPEG-2 wymagają pełnego dostępu do jego możliwości, dlatego zostało skonstruowanych kilka zbiorów o określonych parametrach. Dany zbiór jest określany na podstawie profilu i poziomu pracy kodeka.

Standard MPEG-2 określa cztery poziomy pracy kodeka. Przy pracy na poziomie niskim maksymalna możliwa rozdzielczość wyświetlanego obrazu wynosi 352×288 pikseli w systemie PAL oraz 352×240 pikseli w systemie NTSC - przy prędkości transmisji równej 4 Mb/s. Na poziomie głównym obraz osiąga maksymalnie rozdzielczość 720×576 lub 720×480 pikseli przy 15 Mb/s. Poziom wysoki 1440 charakteryzuje się rozdzielczością 1440×1152 lub 1440×960 pikseli przy 60 Mb/s. Największą prędkość transmisji równą 80 Mb/s można osiągnąć przy pracy kodeka na poziomie wysokim, dla którego rozdzielczość obrazu wynosi 1920×1152 pikseli lub 1920×1080 pikseli.

Oprócz poziomów zostało zdefiniowane siedem profili określających własności procesu kodowania: prosty SP (ang. *Simple Profile*), główny MP (ang. *Main Profile*), skalowalny według SNR (ang. *SNR Scalable*), skalowalny przestrzennie Spt (ang. *Spatially scalable*), wysoki HP (ang. *High Profile*), 4:2:2, wielowidokowy MVP (ang. *Multi-View Profile*). Profil prosty charakteryzuje się brakiem obrazów typu B, nieskalowalnością, formatem próbkowania

4:2:0 oraz małym opóźnieniem kodowania i dekodowania. Profil główny jest profilem najczęściej używanym. Podobnie jak profil prosty jest nieskalowalny i używa formatu 4:2:0, lecz w przeciwieństwie do SP w sekwencji występują obrazu typu B. Profil skalowalny według SNR oprócz skalowalności charakteryzuje się formatem 4:2:0 oraz możliwością wystąpienia obrazów typu B. Innym profilem skalowalnym jest profil przestrzenny. W tym przypadku występują trzy rodzaje skalowalności: według SNR, przestrzenna lub kombinacja skalowalności SNR i skalowalności przestrzennej. W profilu przestrzennym istnieje możliwość zastosowania dwóch warstw rozszerzeń. Tak jak w poprzednim profilu obowiązującym formatem jest 4:2:0 oraz można użyć obrazów B. Profil wysoki różni się od przestrzennego możliwością zastosowania formatu 4:2:2 oprócz 4:2:0. Profil 4:2:2 jest profilem nieskalowalnym, w którym dozwolone jest użycie obrazów typu B a formatem próbkowania może być 4:2:2 lub 4:2:0. Ostatnim z listy jest skalowalny profil wielowidokowy, w którym dopuszczono jedną warstwę rozszerzeń. Niestety w standardzie MPEG-2 nie wszystkie kombinacje profili i poziomów są dozwolone (Tab. 3.3).

Najważniejszą z kombinacji profil-poziom jest kombinacja profilu głównego i poziomu głównego oznaczana jako MP@ML (ang. *Main Profile at Main Level*). MP@ML dopuszcza tylko jeden format próbkowania chrominancji i luminancji a mianowicie 4:2:0. Ponadto MP@ML charakteryzuje się następującymi parametrami: rozdzielczość 720×576 pikseli, częstotliwość obrazów 30 Hz, szybkość transmisji 15 Mb/s.

Tab. 3.3. Kombinacje profili i poziomów dozwolone w standardzie MPEG-2

	Profil prosty	Profil główny	Profil skalowalny według SNR	Profil przestrzennie skalowalny	Profil wysoki	Profil 4:2:2	Profil wielowidokowy
Poziom niski	—	+	+	—	—	—	—
Poziom główny	+	+	+	—	+	+	+
Poziom wysoki 1440	—	+	—	+	+	—	+
Poziom wysoki	—	+	—	—	+	—	+

MPEG-2 został stworzony z myślą o telewizji cyfrowej. Początkowo nie przewidziano zastosowania w telewizji HDTV, na potrzeby której rozpoczęto prace nad standardem MPEG-3. Jednak okazało się, że standard MPEG-2 dobrze radzi sobie z wymogami telewizji HDTV. W związku z tym prace nad MPEG-3 zostały zawieszone.

Standard MPEG-2 jest w stanie przetwarzać sekwencje obrazów o rozmiarach 16383×16383 pikseli (16k×16k). Jednak jako podstawowy format danych wejściowych uznaje się format określony jako ITU-T BT.601. Parametry tego formatu przedstawia tabela 3.4.

Dozwolone formaty próbkowania luminancji i chrominancji, oprócz 4:2:0 jak w MPEG-1, to 4:2:2 oraz 4:4:4. Liczba klatek na sekundę w standardzie MPEG-2 może być równa 23,976 (tryb 3-2 NTSC), 24 (film), 25 (PAL/SECAM), 29,97 (NTSC), 30 (525 linii przy 60 klatkach na sekundę), 50 (PAL o podwójnej rozdzielczości), 59,97 (NTSC o podwójnej rozdzielczości) oraz 60 (525 linii przy 60 klatkach na sekundę o podwojonej rozdzielczości).

Tab. 3.4. Parametry formatu danych wejściowych określonych przez rekomendację ITU-T BT.601

	System NTSC	System PAL
Rozdzielczość luminancji	720×480	720×576
Rozdzielczość chrominancji	360×480	360×576
Liczba klatek na sekundę	29,97	25
Strumień danych [Mb/s]	165,9	165,9

Dla formatu wejściowego określonego przez rekomendację ITU-T BT.601 przy 30 klatkach na sekundę i pod warunkiem, że ramka typu I występuje jeden raz na 15 ramek a ramka typu P jeden raz na 3 ramki - średni rozmiar ramek przy przepływności 4 Mbitów/s wynosi odpowiednio: dla ramki typu I – 400 kbitów, dla ramki typu P – 200 kbitów a dla ramki typu B – 80 kbitów. Zatem średni rozmiar ramek, przy typowej konfiguracji grupy obrazów, wynosi 130 kbitów. Podobnie jak w MPEG-1 wprowadzenie ramek typu B poprawia PSNR o kilka dB.

Jednym z wymogów postawionych przed standardem MPEG-2 była skalowalność strumienia bitowego. W związku z tym zostały wyróżnione cztery typy skalowalności: przestrzenna, SNR, czasowa i tak zwany podział danych. Z tego względu, że przy zastosowaniu skalowalności oprócz warstwy podstawowej występują również dwie warstwy rozszerzeń, które niosą ze sobą dodatkowe informacje a strumień bitowy może się zwiększyć nawet o 60 do 80 %.

Jak przedstawiono standard MPEG-2 oferuje wiele funkcji. Jednak nie każdy dekoder będzie je w pełni wykorzystywał. Dlatego też zostały zdefiniowane profile i poziomy. Zestawienie parametrów odpowiednich kombinacji profili i poziomów przedstawiają tabele 3.5 i 3.6.

Podstawowy koder jest określany przez kombinację profilu podstawowego i poziomu głównego MP@ML. W tym przypadku rozmiar bufora jest równy 224 kB, a liczba pikseli na sekundę wynosi 10 368 000. Ponadto dopuszczalna liczba klatek na sekundę dla poziomu głównego wynosi 23,976; 24; 25; 29,97 oraz 30. W trybie MP@ML standardu MPEG-2 typowa grupa obrazów GOP składa się z 15 ramek o konfiguracji: IBBPBBPBBPBBPBB.

Tab. 3.5. Rozdzielczość obrazu, liczba klatek na sekundę oraz przepływność bitowa dla profili nieskalowalnych w standardzie MPEG-2

		Profil prosty	Profil główny	Profil 4:2:2
Poziom niski	Maksymalna rozdzielczość	—	352×288	—
	Liczba klatek na sekundę		30	
	Przepływność bitowa (Mb/s)		4	
Poziom główny	Maksymalna rozdzielczość	720×576	720×576	720×608
	Liczba klatek na sekundę	30	30	30
	Przepływność bitowa (Mb/s)	15	15	50
Poziom wysoki 1440	Maksymalna rozdzielczość	—	1440×1152	—
	Liczba klatek na sekundę		60	
	Przepływność bitowa (Mb/s)		60	
Poziom wysoki	Maksymalna rozdzielczość	—	1920×1152	—
	Liczba klatek na sekundę		60	
	Przepływność bitowa (Mb/s)		80	

Przyjmuje się, że standard MPEG-2 jest optymalny dla przepływności bitowej 4 Mb/s, co przekłada się na obraz o rozmiarze 544×480 pikseli przy 24 klatkach na sekundę. Jakość sekwencji charakteryzującej się powyższymi parametrami jest porównywalna z jakością obrazu wyświetlanego w systemie PAL. Kolejną przepływnością uważaną za optymalną jest 2 Mb/s. Obraz otrzymany w tym przypadku ma rozmiar 352×480 pikseli przy 30 klatkach na sekundę i jest porównywalnej jakości z obrazem w systemie NTSC. Trzecią przepływnością optymalną jest 6 Mb/s. Odpowiada to obrazowi o rozmiarze 704×480 pikseli i 30 klatkach na sekundę.

Tab. 3.6. Rozdzielczość obrazu, liczba klatek na sekundę oraz przepływność bitowa dla profili skalowalnych w standardzie MPEG-2

		Profil skalowalny według SNR	Profil przestrzennie skalowalny	Profil wysoki	Profil wielowidokowy
Poziom niski	R	352×288 / 30	—	—	352×288 / 30
	P	—			352×288 / 30
	Przepływność bitowa (Mb/s)	4 (W) 3 (P)			8 (W) 4 (R) 4 (P)
Poziom główny	R	720×576 / 30	—	720×576 / 30	720×576 / 30
	P	—		352×288 / 30	720×576 / 30
	Przepływność bitowa (Mb/s)	15 (W) 10 (P)		20 (W) 15 (P+N) 4 (P)	25 (W) 10 (R) 15 (P)
Poziom wysoki 1440	R	—	1440×1152 / 60	1440×1152 / 60	1920×1152 / 60
	P		720×576 / 30	720×576 / 30	1920×1152 / 60
	Przepływność bitowa (Mb/s)		60 (W) 40 (P+N) 15 (P)	80 (W) 60 (P+N) 20 (P)	100 (W) 40 (R) 60 (P)
Poziom wysoki	R	—	—	1920×1152 / 60	1920×1152 / 60
	P			960×576 / 30	1920×1152 / 60
	Przepływność bitowa (Mb/s)			100 (W) 80 (P+N) 25 (P)	130 (W) 50 (R) 80 (P)

Oznaczenia zastosowane w tabeli 3.7 są następujące:

P - warstwa podstawowa,

R - warstwa rozszerzeń,

N - niższa warstwa rozszerzeń,

W - wszystkie warstwy (podstawowa + dwie warstwy rozszerzeń).

Standard MPEG-2 znalazł swoje zastosowanie w telewizji SDTV, gdzie przepływność bitowa wynosi od 4 Mb/s do 6 Mb/s. MPEG-2 jest też stosowany przy przesyłaniu sygnału telewizyjnego do stacji z prędkością od 8 Mb/s do 10 Mb/s. MPEG-2 pokrywa również zapotrzebowanie telewizji HDTV, dla której wymagana przepływność bitowa wynosi około 15 Mb/s. Ponadto standard dobrze radzi sobie przy przepływności bitowej od 30 Mb/s do 50 Mb/s, co odpowiada jakości produkcji telewizyjnej.

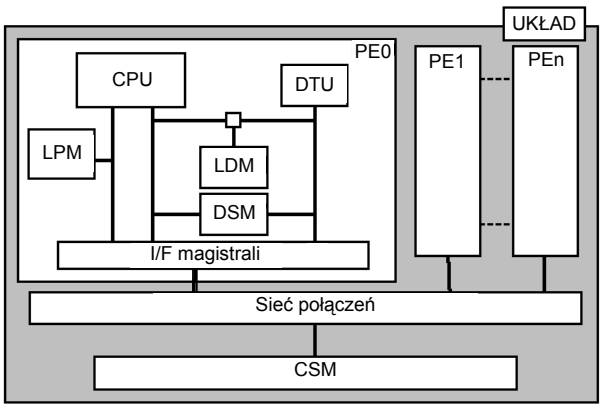
4. Dotychczasowe rozwiązania koderów, dekoderów oraz kodeków w standardzie MPEG-2

Obecnie dostępnych jest wiele software'owych i sprzętowych koderów i dekoderów standardu MPEG-2. Jednymi z pierwszych były: dekodery AViA-500 i AViA-502 firmy C-Cube, dekoder HDM8211M firmy Hyundai Electronics, kodery MPEGSE10, MPEGSE20, MPEGSE30, dekodery MPEGCD10, MPEGCD20 i MPEGCD21 firmy IBM Microelectronics, dekoder STi3500A firmy SGS-Thomson Microelectronics oraz dekodery TC81201F i TC81211F firmy Toshiba [2].

4.1. Rozwiązania z procesorami DSP i GPP

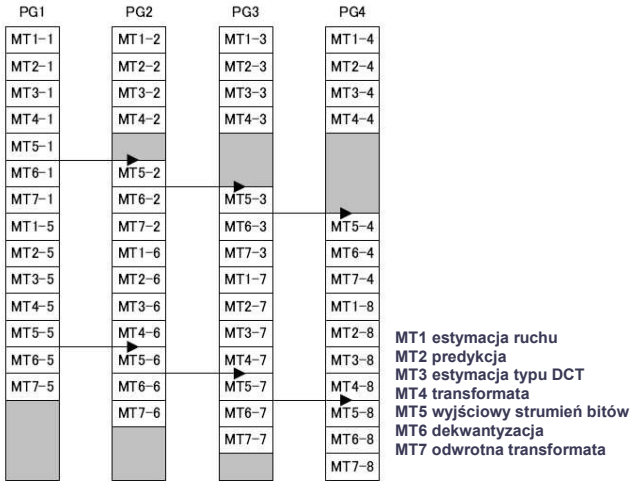
Multiprocesor OSCAR

Przykładem kodera standardu MPEG-2, wykorzystującego niekomercyjny układ OSCAR, jest rozwiązanie opracowane na Uniwersytecie Waseda [40]. Budowę układu specjalizowanego multiprocesora na przedstawia rysunek 4.1.



Rys. 4.1. Architektura układu multiprocesora OSCAR

Program kodera MPEG-2 zastosowany w zaproponowanym rozwiązaniu został napisany w Fortranie na podstawie software'u opracowanego przez firmę MediaBench. Wejściowy sygnał video, przy jakim zostało przetestowane rozwiązanie opracowane na Uniwersytecie Waseda, miał rozdzielczość 176×144. Biorąc pod uwagę pojemność pamięci, jaka jest zawarta w układzie OSCAR, projektanci stwierdzili, że układ nie będzie w stanie skompresować sygnału wejściowego o rozdzielczości QCIF lub QVGA. Z tego powodu został zastosowany algorytm dzielący każdy z etapów przetwarzania na częściowe pętle, biorąc pod uwagę zależności danych między poszczególnymi pętlami. Przykładowy zaproponowany przez twórców podział etapów kompresji sygnału wejściowego przedstawia rysunek 4.2.



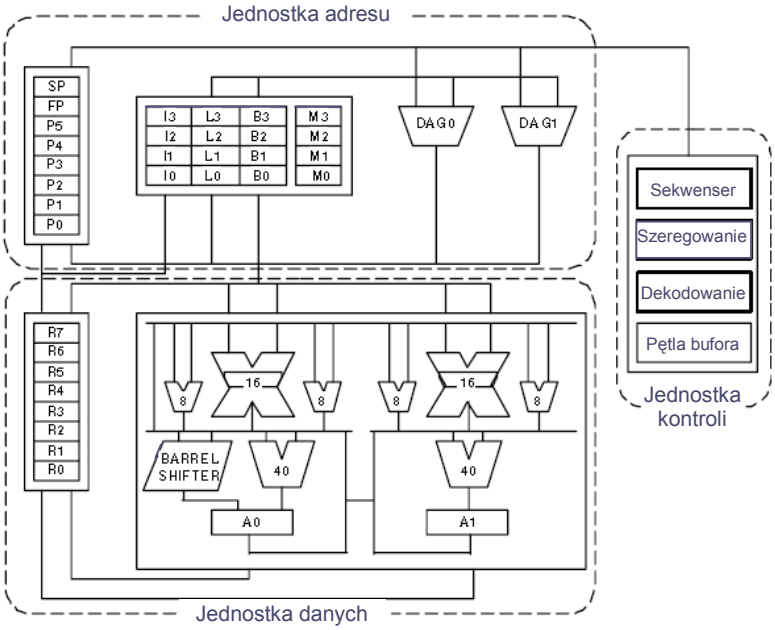
Rys. 4.2. Rezultaty podziału każdego z etapów na 8 częściowych pętli wykonywanych na 4 grupach procesorów

Dzięki takiemu dzieleniu etapów na częściowe pętle udało się uzyskać przyspieszenie przetwarzania w stosunku do przetwarzania sekwencyjnego, równe 2,12 przy zastosowaniu dwóch jednostek procesorowych. Przy zastosowaniu 4 jednostek uzyskane przyspieszenie wynosiło 4,06 a przy 8 jednostkach 6,82. Przyspieszenia takie udało się uzyskać dzięki temu, że podczas gdy jedna z jednostek procesorowych wykonuje kwantyzację oraz kodowanie VLC, inne jednostki mogą obliczać estymację ruchu kolejnego makrobloku przy użyciu dostępnej pamięci.

Procesor DSP z rodziny Blackfin

Jednym z przykładów zastosowania software'owego kodeka MPEG-2 jest implementacja tego standardu w 16-bitowym układzie DSP z rodziny Blackfin [50]. Układy te łączą w sobie

funkcje mikrokontrolera oraz procesora DSP. Schemat blokowy rdzenia procesora Blackfin przedstawia rysunek 4.3.



Rys. 4.3. Rdzeń procesora DSP z rodziny Blackfin

Rdzeń procesora, który powstał przy współpracy firm Analog Devices oraz Intela, zawiera dwie 40-bitowe jednostki arytmetyczno-logiczne, dwa 16-bitowe układy mnożące oraz cztery 8-bitowe jednostki arytmetyczno-logiczne odpowiedzialne za przetwarzanie obrazu. Kodek wykorzystuje instrukcje dedykowane dla jednostki arytmetyczno-logicznej przetwarzającej piksele.

Zaimplementowany kodek standardu 13818-2 ma możliwość kompresji oraz dekompresji obrazów z przeplotem oraz obrazów bez przeplotu. Dodatkowo kodek współpracuje z trzema formatami obrazu: 4:4:4, 4:2:2 oraz 4:2:0. Przy kompresji/dekompresji możliwe jest zastosowanie kombinacji złożonych z trzech typów obrazów: I, P oraz B.

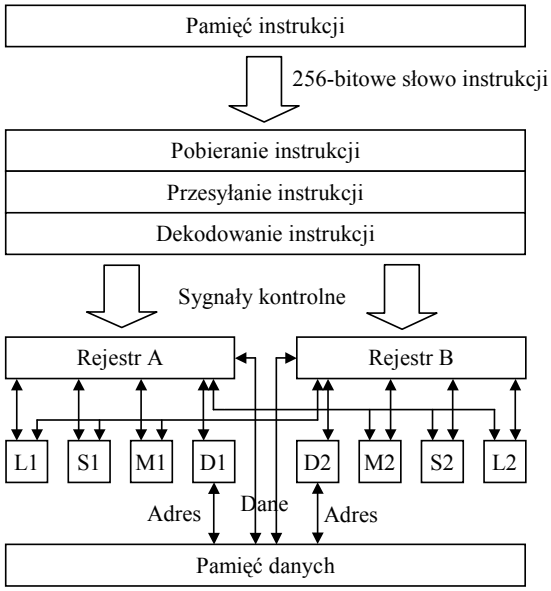
Tab. 4.1. Zależności czasowe poszczególnych algorytmów

	Funkcja	% wykorzystanie całkowitego czasu
enkoder	VLC	59,0
	Estymacja ruchu	20,0
	DCT	4,0
dekoder	Dekodowanie makrobloku	28,7
	Predykcja	26,8
	IDCT	25,0

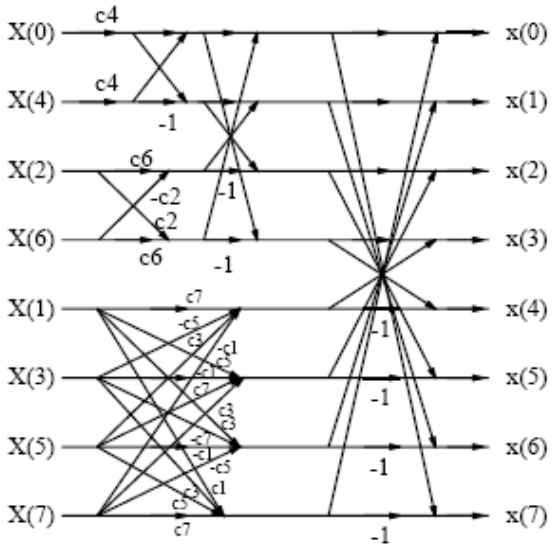
Zaimplementowany kodek w procesorze taktowanym zegarem 300 MHz może skompresować 0,4 klatki na sekundę obrazu o rozdzielczości 352×288 pikseli oraz 1,5 klatki na sekundę obrazu 176×144 pikseli. W przypadku dekompresji kodek jest w stanie przetworzyć 30 klatek obrazu 176×144 pikseli oraz 15 klatek o rozdzielczości 352×288 pikseli w ciągu sekundy. Takie rozwiązanie nie nadaje się do kompresji sekwencji obrazu w czasie rzeczywistym, ponieważ jednosekundowa sekwencja 24 klatkowa o rozdzielczości 176×144 pikseli byłaby kodowana przez 16 sekund.

Układ TMS320C6x firmy Texas Instruments

Kolejnym przykładem zastosowania standardu MPEG-2 jest software'owa implementacja dekodera w procesorze sygnałowym TMS320C6201 firmy Texas Instruments [72]. W procesorze z rodziny C6x znajduje się 8 jednostek funkcjonalnych, rozdzielonych na dwa odrębne zbiory zawierające po 4 jednostki funkcjonalne. Jedną z nich jest jednostka D sterująca odczytem i zapisem do pamięci danych oraz obsługująca operacje dodawania oraz odejmowania. Jednostki M odpowiadają za operacje mnożenia. Jednostki L oprócz dodawania i odejmowania wykonują operacje logiczne oraz porównania. Za operacje logiczne odpowiadają jednostki S, które dodatkowo obsługują przesunięcia. Każda czwórka jednostek ma własny rejestr co uwidoczniono na rysunku 4.4.



Rys. 4.4. Architektura układu TMS320C6x



Rys. 4.5. Graf przepływu sygnału dla 8-punktowej transformacji IDCT zastosowanej w implementacji w układzie TMS320C6X

Tab. 4.2. Parametry sekwencji wyjściowej w zależności od wielkości strumienia bitowego oraz zastosowanego układu firmy Teksas Instruments

Strumień bitowy	Format		Układ TMS320...		
			C6203-300	C6201-200	C6211-150
1,5 Mb/s	CIF PAL	Liczba ramek na sekundę	100	67	50
	352×288	ms/ramkę	10	15	20
2Mb/s	Full size	Liczba ramek na sekundę	29	20	15
	720×576	ms/ramkę	34	51	68
4Mb/s	Full size	Liczba ramek na sekundę	26	17	13
	720×576	ms/ramkę	38	58	77
8Mb/s	Full size	Liczba ramek na sekundę	25	17	13
	720×576	ms/ramkę	39	59	78
15Mb/s	Full size	Liczba ramek na sekundę	21	14	10
	720×576	ms/ramkę	448	72	96

Wszystkie jednostki są kontrolowane za pomocą 256-bitowego słowa instrukcji. Dekodowanie entropijne w zaproponowanym rozwiązaniu opiera się na 8 tablicach LUT (ang. *Look-Up-Table*), które są wybierane na podstawie 16 pierwszych bitów strumienia danych. Proces dekodowania entropijnego trwa maksymalnie 29 cykli. Dwuwymiarowa odwrotna dyskretna transformacja kosinusowa FDCT (ang. *Fast DCT*) dla bloku 8×8 pikseli, realizowana jest za pomocą 16 jednowymiarowych 8-punktowych IDCT. Implementacja 2D-IDCT składa się z 12-cyklowej pętli w jednym wymiarze oraz 13-cyklowej pętli w drugim wymiarze. Wyznaczenie 6 bloków o wielkości 8×8 pikseli zaimplementowanemu algorytmowi

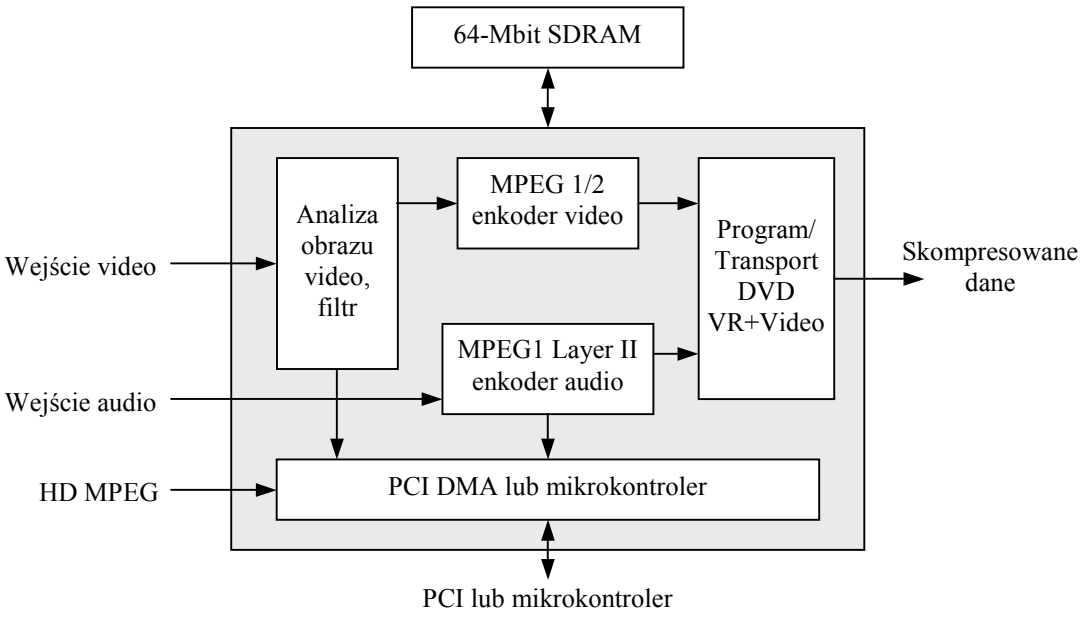
2D-IDCT zajmuje 1249 cykli. Estymacja ruchu w zaproponowanym rozwiązaniu opiera się na 2 blokach o składowych 17×17 dla luminancji i 9×9 dla chrominancji.

Z zastosowanego przez twórców testu wynika, że przy obrazie 720×480 pikseli i 30 klatkach na sekundę transformacja IDCT wykorzystuje 51 Mcykli/sekundę, VLD w połączeniu z dekwantyzacją zajmuje 39M cykli/sekundę a proces kompensacji ruchu 62M cykli/sekundę. Wyniki takie zostały otrzymane przy założeniu, że tylko 25% współczynników DCT jest niezerowych oraz że na każde 30 ramek 20 stanowią ramki B, 8 to ramki P a tylko 2 to ramki typu I. Wyjściowa sekwencja video jest przedstawiona w formacie YUV 4:2:0. Bufor wyjściowy układu mieści 4 ramki obrazu.

4.2. Rozwiązania z układmi ASIC

Układ CX23416 firmy Conaxant

Przykładem sprzętowego kodera standardu MPEG-2 jest układ CX23416 firmy Conexant [9]. Schemat blokowy tego układu przedstawia rysunek 4.6.



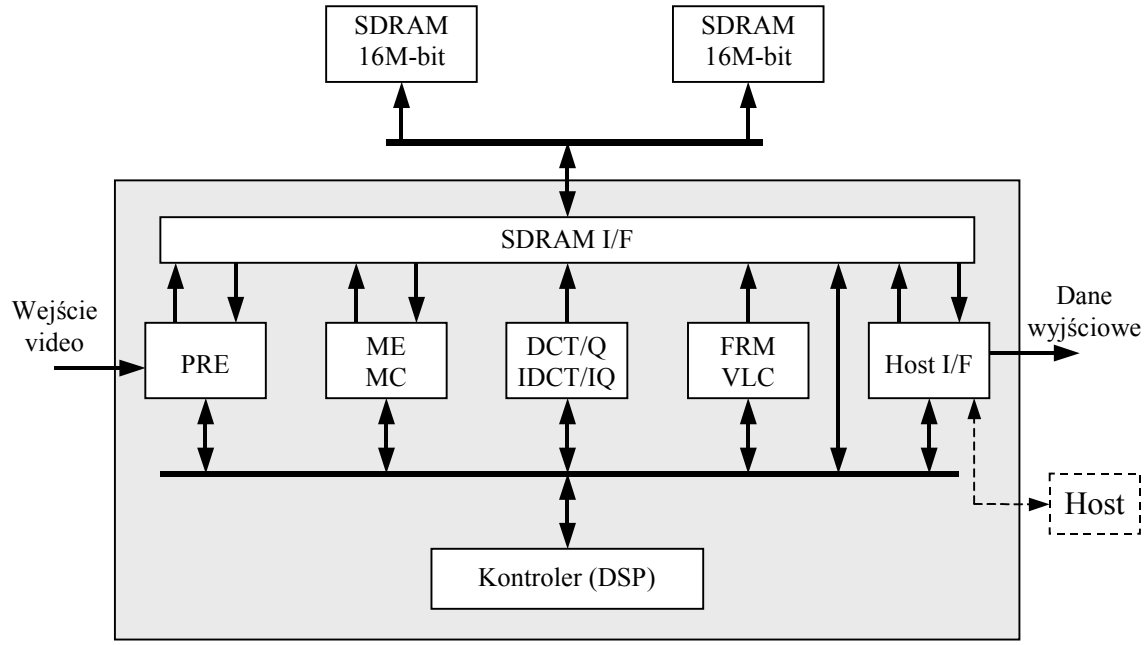
Rys. 4.6. Schemat blokowy układu CX23416 firmy Conexant

Koder jest w stanie skompresować obraz o rozdzielczości 720×480 przy 30 ramkach na sekundę oraz 720×576 przy 25 ramkach na sekundę. Przy kompresji używane są wszystkie typy obrazów: I, P oraz B. W układzie CX23416 długość grupy obrazów jest programowalna. Wielkości obszarów poszukiwań w estymacji ruchu są różne dla obrazów typu P i B.

W przypadku predykcji tylko na podstawie poprzedniego obrazu wielkość wyznaczonego wektora ruchu może się mieścić w zakresie ±326 w poziomie oraz ±202 w pionie. Przy wyznaczaniu wektorów ruchu na podstawie obrazu poprzedniego oraz następnego pojedynczy wektor ruchu może się wahać w zakresie ±296 w poziomie oraz ±184 w pionie. Wektory wyznaczane są z dokładnością półpikselową. Koder MPEG-2 może pracować w dwóch trybach MP@ML oraz SP@ML. Ponadto układ CX23416 oferuje konwersję z formatu 4:2:2 do formatu 4:2:0. Koder współpracuje z wejściowym sygnałem video w formacie 4:2:2 YUV CCIR-656.

Układ CXD1922Q firmy Sony

Podobnymi parametrami jak układ CX23416 firmy Conexant, charakteryzuje się układ CXD1922Q firmy Sony [76]. Schemat blokowy proponowanego przez Sony rozwiązania przedstawia rysunek 4.7.



Rys. 4.7. Schemat blokowy kodera CXD1922Q firmy Sony

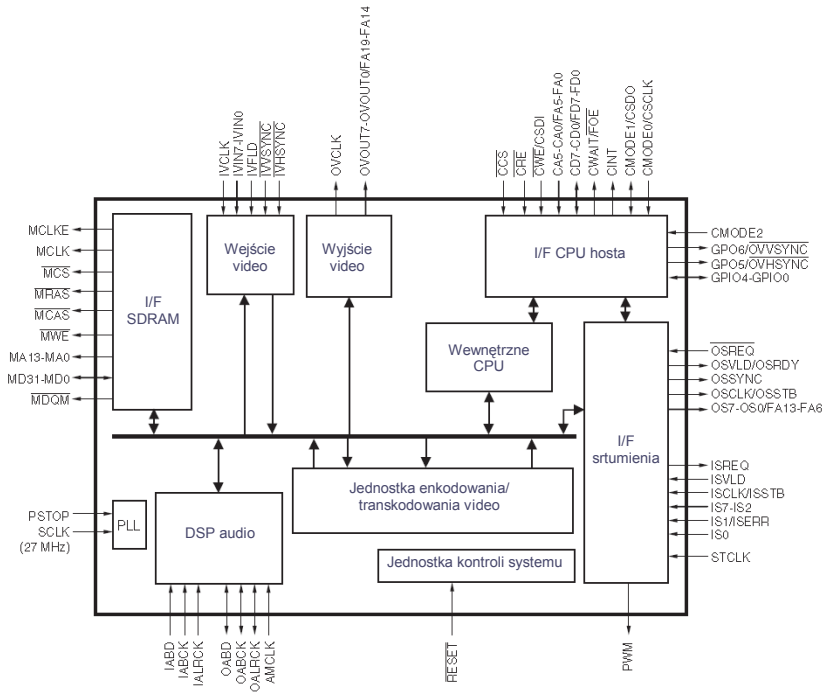
Koder pracuje w dwóch trybach MP@ML. Maksymalna rozdzielczość obrazu wejściowego, jaką jest w stanie skompresować, jest równa 720×480 przy 30 ramkach na sekundę dla systemu NTSC lub 720×576 przy 25 ramkach na sekundę dla systemu PAL. Ponadto CXD1922Q używa wszystkich typów obrazów (I, P oraz B) przy kompresji danych wejściowych. W przypadku, gdy używane są tylko obrazy typu I, to przepływność bitowa kodera wynosi maksymalnie 25Mbps. Jeżeli zastosowane zostaną dodatkowo obrazu typu P oraz B, przy

czym pomiędzy obrazami typu I oraz P mogą pojawić się maksymalnie 2 obrazy typu B, przepływność bitowa będzie wynosiła maksymalnie 15 Mbps. Układ CXD1922Q współpracuje z sygnałem wejściowym zgodnym z formatem 4:2:2 YUV dla standardu CCIR-656. Formatem wyjściowym jest jednak 4:2:0. W przeciwieństwie do układu CX23416, Sony w swoim koderze zdefiniowało taki sam zakres wielkości wektorów ruchu dla obrazów typu P oraz B. Zakres ten wynosi od –288 do +287,5 w poziomie oraz od –96 do +95,5 w pionie. Układ CXD1922Q został wykonany w technologii CMOS 0,4µm i zawiera 4,5 miliona tranzystorów.

Koder jest taktowany zegarem zewnętrznym wynoszącym 27 MHz. Wewnętrzne operacje wykonywane są z częstotliwościami: 67,5 MHz, 45 MHz, 27 MHz, 22,5 MHz oraz 13,5 MHz. Częstotliwość 67,5 MHz wykorzystywana jest przy komunikacji z pamięcią SRAM. Estymacja i kompensacja ruchu wykorzystuje zegar wewnętrzny 45 MHz. Blok odpowiedzialny za kodowanie entropijne współpracuje z zegarem 22,5 MHz. Częstotliwość 13,5 MHz używana jest przez filtry a 27 MHz przez rdzeń układu.

Układ µPD61051/2 firmy Nec

Kolejnym koderem standardu MPEG-2 jest układ µPD61051/2 firmy Nec [53]. Schemat blokowy tego rozwiązania sprzętowego przedstawia rysunek 4.8.



Rys. 4.8. Schemat blokowy układu µPD61051/2 firmy Nec

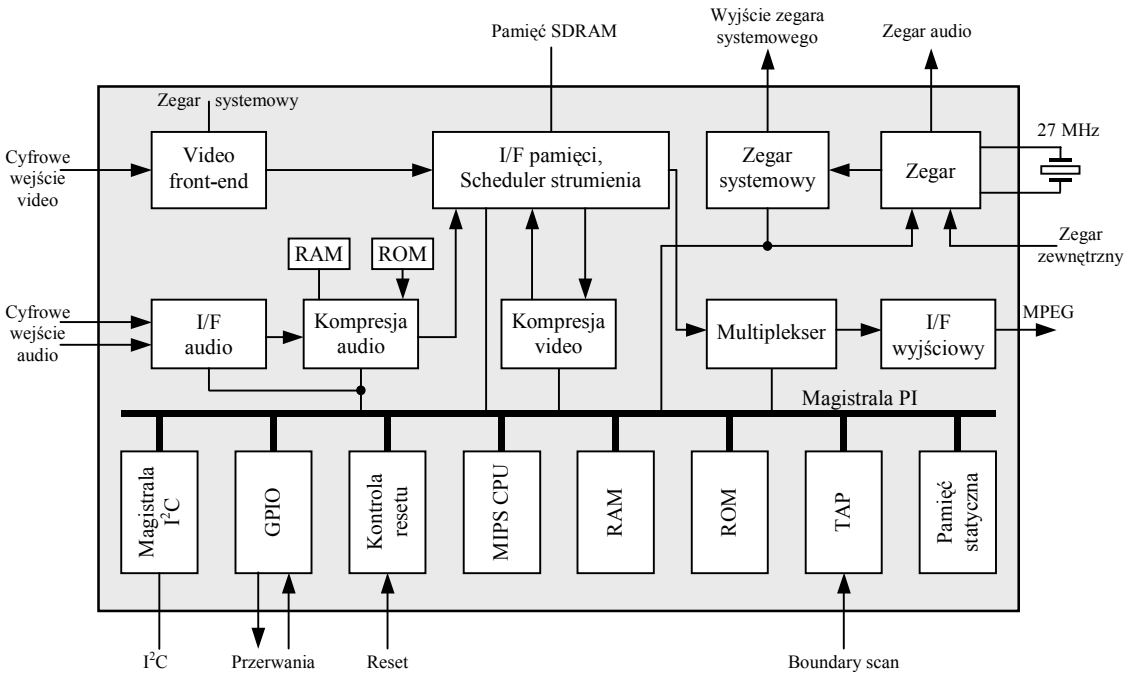
Koder µPD61051/2 obsługuje dwa tryby pracy zgodne z normą ISO/IEC 13818: MP@ML oraz SP@ML. Maksymalny obraz, jaki może być skompresowany, ma rozdzielczość 720×480 dla systemu NTSC oraz 720×576 dla systemu PAL. Dostępne jest również przetwarzanie obrazów o innych rozdzielczościach. Układ µPD61051/2 współpracuje z obrazami o rozdzielczościach poziomych równych: 720, 704, 640, 544, 480, 352 oraz 320 pikseli w linii. Rozdzielczości pionowe są następujące: 480, 240, 576 oraz 288 linii w ramce. Koder współpracuje z dwoma formatami wejściowego/wyjściowego sygnału video: 8-bitowym YC_bC_r 4:2:2 (zgodnym z ITU-R Rec. 656) oraz 8-bitowym YC_bC_r 4:2:0. Układ µPD61051/2 wykorzystuje wszystkie trzy typy obrazów: I, P oraz B. Podobnie jak w przypadku układu CX23416 Conexant'a, Nec zastosował różne zakresy wielkości wektorów ruchu w zależności od rodzaju obrazu. Dla obrazów typu P wektory ruchu zawierają się z zakresie ±128 w poziomie oraz ±64 w pionie. Dla obrazów typu B określone są dwa zakresy. Pierwszy z nich wynosi ±96 w poziomie oraz ±48 w pionie. Drugi to ±64 w poziomie i ±32 w pionie. Pomiędzy obrazami I i P mogą wystąpić maksymalnie 2 obrazy typu B. Skompresowanie obrazu o powyższych parametrach wymaga od kodera współpracy z pamięcią SDRAM o pojemności 128Mb. Firma Nec opracowała również inne układy wykorzystujące standard MPEG-2. Są wśród nich zarówno kodery (µPD61151/2, µPD61153B, µPD61154B), dekodery (µPD61110, µPD61111, µPD61120, µPD61122, µPD61123, µPD61126, µPD61128, µPD61130A, µPD61132A, µPD61133, µPD61135 oraz µPD61160) jak i kodeki (µPD61175, µPD61176, µPD61177, µPD61178, µPD61181). Z wyżej wymienionych jedynie dekoder µPD61160 ma możliwość pracy w trybie MP@HL co oznacza, że wyjściowy sygnał video może mieć maksymalnie 1080 linii przy obrazach z przeplotem (przy obrazach bez przeplotu sygnał video może mieć 720 linii).

Układ SAA6752HS firmy Philips Semiconductors

Również w ofercie firmy Philips Semiconductors można znaleźć koder wykorzystujący standard MPEG-2. Układem tym jest koder oznaczany symbolem SAA6752HS [60]. Schemat blokowy tego rozwiązania przedstawia rysunek 4.10.

Koder Philipsa pracuje w trybie MP@ML. Układ SAA6752HS kompresuje wejściowy sygnał video w formacie 4:2:0 oraz 4:2:2 zgodnie z normami ITU-R Rec.601 i ITU-R Rec.656. Układ przetwarza wejściowy sygnał video o rozdzielczości 720×576 dla systemu PAL oraz 720×480 dla systemu NTSC. W procesie kompresji SAA6752HS wykorzystuje wszystkie typy obrazów: I, P oraz B. Grupa obrazów może mieć postać typu: IPP..., IBPBP... lub

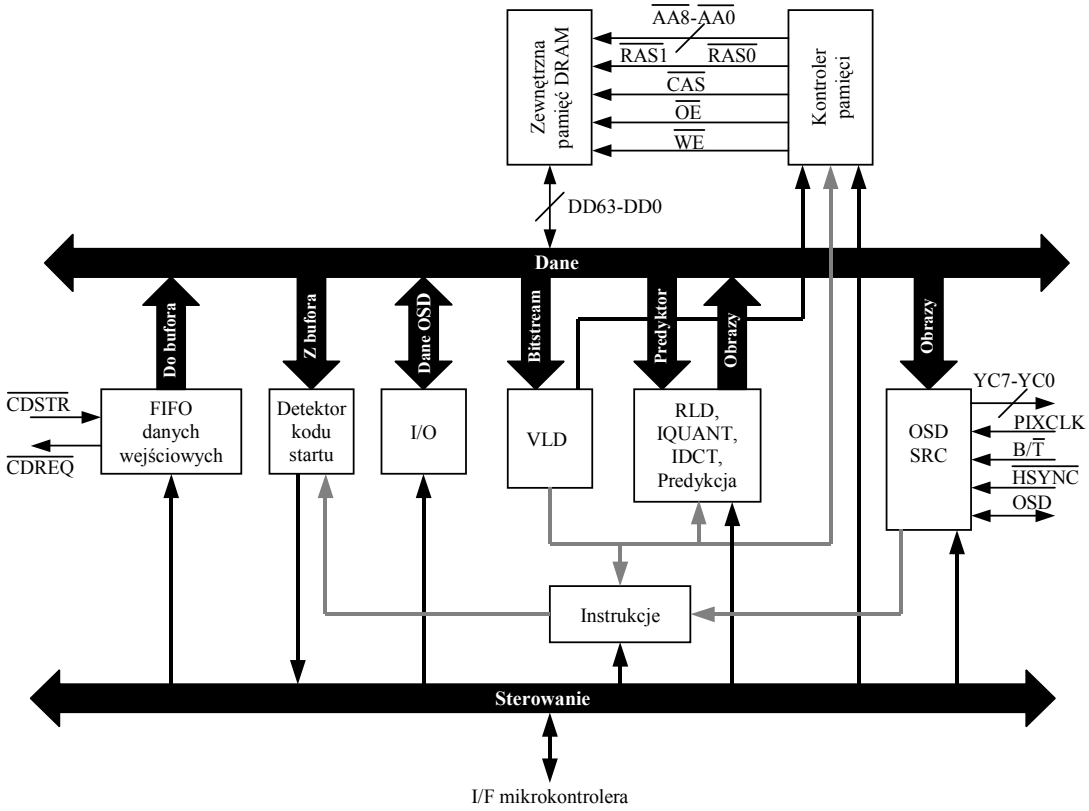
IBBP... . Przy wykorzystaniu ramek typu B niezbędne jest zastosowanie 64Mb pamięci zewnętrznej. Dodatkowo GOP może mieć długość od 2 do 19 ramek.



Rys. 4.9. Schemat blokowy układu SAA6752HS firmy Philips

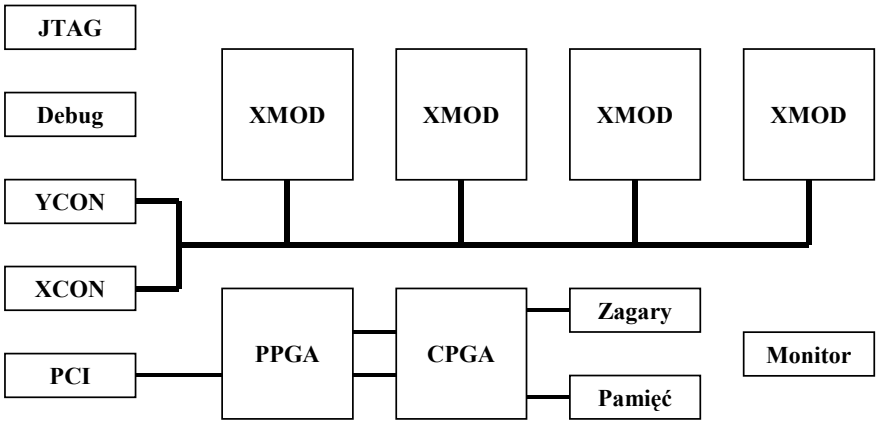
Układ STi3500A firmy SGS-Thomson Microelectronics

Jednym z dekoderów standardu MPEG-2 jest układ STi3500A firmy SGS-Thomson Microelectronics [69]. Dekoder oferuje przetwarzanie obrazów w czasie rzeczywistym. Wyjściowe obrazy mają rozdzielczość 720×480 lub 720×576 pikseli. Układ SGS-Thomson może przetworzyć 60 klatek na sekundę obrazu o pierwszej rozdzielczości. W przypadku drugiej rozdzielczości dekoder może przetworzyć 50 klatek w ciągu sekundy. Układ STi3500A współpracuje z wszystkimi typami obrazów I, P oraz B. Obsługiwane wektory ruchu mieszczą się w zakresie od –1024 do +1023,5 zarówno w pionie jak i w poziomie. Współczynniki dyskretnej transformacji kosinusowej muszą mieścić się w zakresie od –2048 do +2047, co oznacza, że mogą być to współczynniki maksymalnie 12-bitowe. Dekoder STi3500A obsługuje trzy konfiguracje profil-poziom: MP@ML, MP@LL oraz SP@ML. Wyjściowy sygnał video spełnia wymagania standardów CCIR 601 oraz 656. Schemat blokowy dekodera przedstawia rysunek 4.10.



Rys. 4.10. Schemat blokowy dekodera STi3500A

Do pracy dekodera wymagany jest mikrokontroler oraz zewnętrzna pamięć DRAM. Typową konfiguracją współpracującej pamięci są 4 układy DRAM o organizacji 256k×16. Zadaniem pamięci jest buforowanie wejściowego strumienia bitowego, przechowywanie zdekodowanych obrazów oraz buforowanie danych wyjściowych. Mikrokontroler używany jest przy przetwarzaniu warstwy systemowej oraz kontroluje pracę dekodera i umożliwia przesyłanie skompresowanych danych do bufora danych wejściowych. Dekoder pracuje z zegarem 56 MHz.



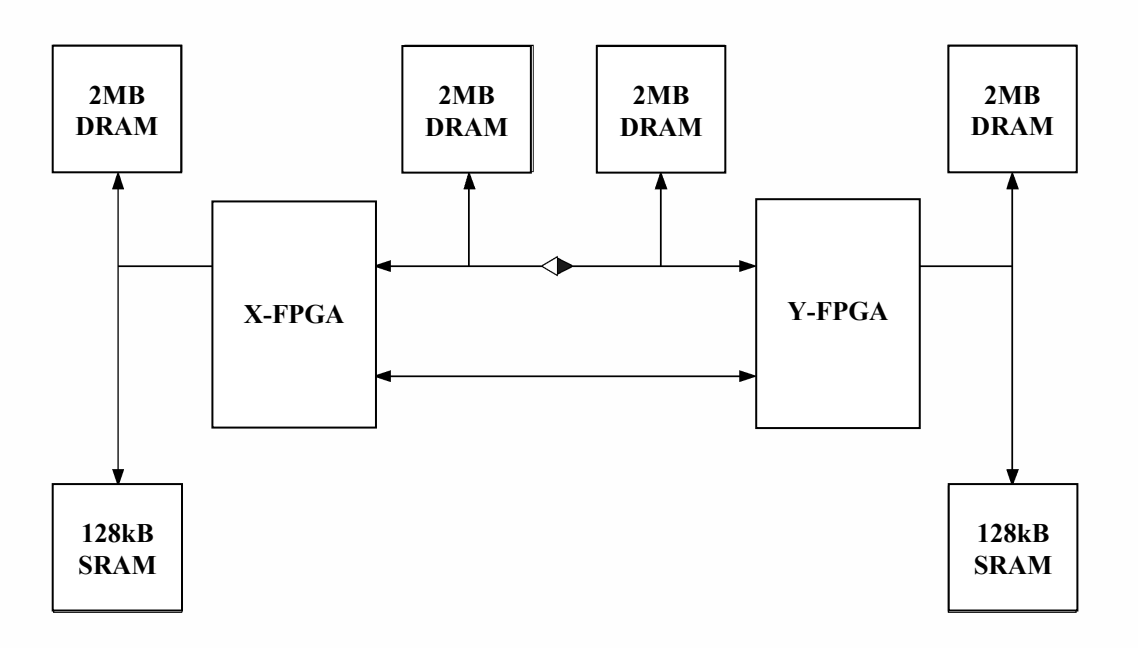
Rys. 4.11. Schemat blokowy platformy G900 firmy GigaOps

4.3. Rozwiązania z układami FPGA

Koder MPEG-2 opracowany na Wright State University

Rozwiązanie opracowane na Wright State University [10] wykorzystuje platformę rekonfigurowalną G900 firmy GigaOps. Schemat blokowy tej platformy przedstawia rysunek 4.11.

Bloki PPGA oraz CPGA są to układy programowalne FPGA firmy Xilinx, odpowiadające za sterowanie platformą G900. Blok PPGA, który wykorzystuje układ XC4013E-2, kontroluje komunikację pomiędzy hostem a modułami XMOD. Schemat pojedynczego modułu XMOD przedstawia rysunek 4.12. Drugi z bloków sterujących CPGA wykorzystuje układ XC5210-5. Blok ten odpowiada za generację zegara i konfigurację przetwarzania.



Rys. 4.12. Schemat blokowy modułu XMOD

Platforma G900 może obsłużyć maksymalnie 16 modułów XMOD, co oznacza maksymalnie 32 układy FPGA. XMOD składa się z dwóch układów FPGA serii XC4000E firmy Xilinx. W opracowanym rozwiązaniu wykorzystywane są układy XC4020E.

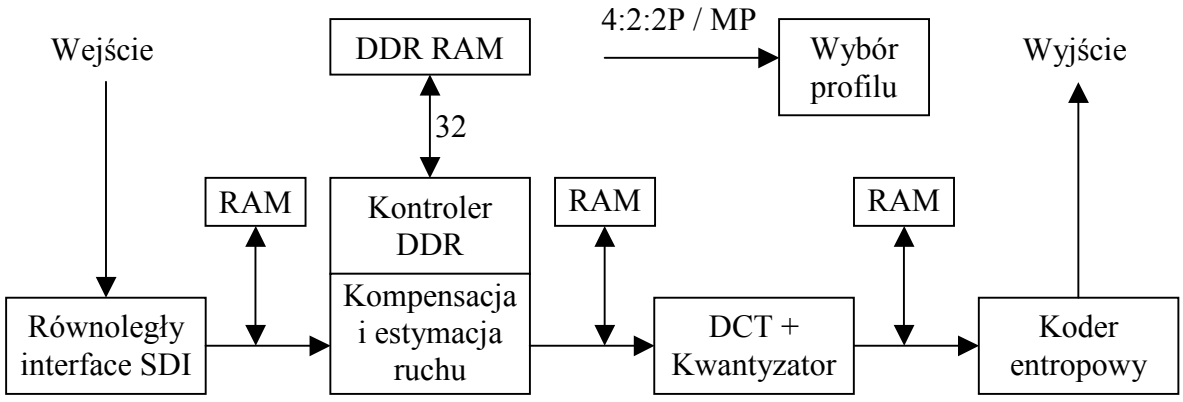
Koder MPEG-2 został opracowany przy wykorzystaniu języka C++. Było to możliwe przy wykorzystaniu narzędzia XLINK-OS dostarczonego przez firmę GigaOps. Koder MPEG-2 zastał zaimplementowany przy użyciu kodu źródłowego uzyskanego od MPEG Software Simulation Group. Koder został przetestowany za pomocą sekwencji 27 obrazów o rozdzielczości 320×240 pikseli. Skompresowanie takiej sekwencji w proponowanym rozwiązaniu zajmuje 46,5 sekundy z czego 49% czasu przetwarzania stanowi obliczanie przemieszczenia makrobloków. Po modyfikacji pierwotnego algorytmu, polegającej na

kopiowaniu następnego obrazu pamięci SRAM oraz wykorzystaniu obu układów FPGA modułu XMOD, koder MPEG-2 kompresował 30 obrazów o rozdzielczości 320×240 pikseli w ciągu 9 sekund przy wykorzystaniu predykcji w przód oraz w tył. Takie rozwiązanie nie spełnia wymogów pracy w czasie rzeczywistym.

Kodery MPEG-2 firmy Duma Video

Przykładem sprzętowej implementacji kodera MPEG-2 w układach FPGA mogą być rozwiązania oferowane przez firmę Duma Video [95] wykorzystujące układy XC2V1500-5 i XC2V3000-5 z rodziny Virtex-II firmy Xilinx. Rozwiązania oparte na układzie XC2V1500-5 są przeznaczone dla telewizji cyfrowej o rozdzielczości 720×480. Jedno z tych rozwiązań, oznaczone jako SDTVp, przetwarza obrazy bez przeplotu przy 59,94 ramkach na sekundę. Implementacja rozwiązania wykorzystuje 6851 bloków CLB i 113 bloków I/O. Rozwiązanie oznaczone jako SDTVi przetwarza obrazy z przeplotem, ponadto charakteryzuje się mniejszym upakowaniem niż SDTVp bo wykorzystuje 5883 CLB i 58 IOB. Kodery SDTV wykorzystują poziom wysoki oraz dwa profile: główny i 4:2:2.

Rozwiązania wykorzystujące układ XC2V3000-5 przeznaczone są dla telewizji o podwyższonej rozdzielczości HDTV. Jedno z nich nich (HDTVp) oferuje rozdzielczość obrazu 1280×720 pikseli bez przeplotu przy 59,94 klatkach na sekundę. HDTVi przetwarza obrazy z przeplotem a oferowana rozdzielczość obrazu wynosi 1920×1080 przy 29,97 ramkach na sekundę. HDTVp oparte jest na 6508 blokach CLB i 113 blokach I/O, natomiast HDTVi wykorzystuje 5540 CLB i 58 IOB. Kodery HDTVi i HDTVp wykorzystują profile główny i 4:2:2 oraz poziom wysoki.



Rys. 4.13. Ogólny schemat blokowy kodera standardu MPEG-2 firmy Duma Video

Schemat blokowy, obowiązujący dla każdego z powyższych rozwiązań, jest przedstawiony na rysunku 4.13. Schemat ten jest wykorzystywany przy implementacjach HDTVi, HDTVp, SDTVi oraz SDTVp. W przypadku implementacji wykorzystującej w procesie kompresji tylko ramki typu I nie jest wymagane dołączenie zewnętrznych układów pamięci RAM. Natomiast jeżeli oprócz ramek typu I występują również ramki typu P niezbędna jest zewnętrzna pamięć DDR RAM.

Kolejnym koderem standardu MPEG-2 firmy Duma Video [95] jest DV-X. Koder został zaimplementowany w układzie XC2VP70 firmy Xilinx. Użycie zasobów tego układu przedstawia tabela 4.3.

Maksymalne opóźnienie zakodowanej ramki wynosi 33ms dla obrazu bez przeplotu lub 66ms dla obrazu z przeplotem. DV-X obsługuje dwa formaty próbkowania luminancji i chrominancji: 4:2:0 oraz 4:2:2. Koder oferuje współpracę z maksymalnymi rozdzielczościami 1920×1080 dla obrazów z przeplotem oraz 1280×720 dla obrazów bez przeplotu.

Tab. 4.3. Wyniki implementacji koder DV-X firmy Duma Video w układzie XC2VP70

Rodzaj zasobów	Użyte zasoby	Dostępne zasoby	Procentowe wykorzystanie
Slice	31555	33088	95,6%
BRAM	297	328	90,5%
Mnożarki	144	328	43,9%
DCM	1	8	12,5%
BUFGMUX	7	16	43,7%
I/O	267	992	26,8%

4.4. Podsumowanie

Standard MPEG-2 znalazł swoje zastosowanie w transmisji telewizyjnej, archiwizacji sekwencji obrazów, zapisie danych na płytach DVD, kamerach cyfrowych itp.

Istniejące obecnie sprzętowe kodery, dekodery i kodeki standardu MPEG-2, oparte na dedykowanych układach ASIC, współpracują z rozdzielczością SDTV, czyli 720×480 pikseli. Rozwiązania te wykorzystują w procesie kompresji wszystkie typy obrazów z możliwością kompresji oraz dekompresji sekwencji obrazów w czasie rzeczywistym. Istniejące

rozwiązania koderów standardu MPEG-2, wykorzystujące układy FPGA, oferują kompresję w czasie rzeczywistym, z możliwością zastosowania jedynie obrazów typu I oraz P. Ponadto kodery te mogą współpracować z maksymalną rozdzielczością 1920×1080 pikseli dla obrazów z przeplotem oraz rozdzielczością 1280×720 dla obrazów bez przeplotu.

5. Implementacje algorytmów transformacji DCT i IDCT

5.1. Wprowadzenie

Dwuwymiarową dyskretną transformatę kosinusową 2D-DCT można rozłożyć na dwie transformaty jednowymiarowe 1D-DCT [54] - z tym, że jedna będzie przeprowadzana względem wierszy, a druga względem kolumn. Jeżeli x_n oznacza wektor wartości wejściowych to wektor współczynników transformaty y_n można wyznaczyć na podstawie zależności (5.1).

$$y_k = \frac{1}{2}\alpha_k \sum_{n=0}^{N-1} x_n \cos\left[\frac{2\pi}{4N}(2n+1)k\right] \tag{5.1}$$

gdzie: k=0, 1, ..., N-1,

$\alpha_0 = 1/\sqrt{N}$ dla k=0 lub $\alpha_k = \sqrt{2/N}$ w pozostałych przypadkach

Rys. 5.1. Schemat blokowy dwuwymiarowej transformacji DCT

Dyskretną transformatę DCT można również przedstawić w postaci macierzowej. Dla N=8 postać macierzowa 1D-DCT jest zobrazowana zależnością (5.2).

Jak widać z zależności (5.2) do obliczenia pojedynczego wektora współczynników transformaty trzeba wykonać 64 mnożenia i 56 dodawań.

Liczbę wykonywanych operacji mnożenia ze wzoru (5.2) można zredukować z 64 do 32 poprzez wprowadzenie zerowych elementów c_k macierzy współczynników, lecz wtedy wzrośnie liczba dodawań i odejmowań. W tym przypadku postać macierzowa będzie miała postać

(5.3). Warto zauważyć pewne symetrie w macierzy współczynników we wzorze (5.3): górna lewa ćwiartka macierzy ma pionową oś symetrii, a ćwiartka dolna prawa ma ukośną oś symetrii.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \alpha_k \begin{bmatrix} c_0 & c_0 & c_0 & c_0 & c_0 & c_0 & c_0 & c_0 \\ c_1 & c_3 & c_5 & c_7 & -c_7 & -c_5 & -c_3 & -c_1 \\ c_2 & c_6 & -c_6 & -c_2 & -c_2 & -c_6 & c_6 & c_2 \\ c_3 & -c_7 & -c_1 & -c_5 & c_5 & c_1 & c_7 & -c_3 \\ c_4 & -c_4 & -c_4 & c_4 & c_4 & -c_4 & -c_4 & c_4 \\ c_5 & -c_1 & c_7 & c_3 & -c_3 & -c_7 & c_1 & -c_5 \\ c_6 & -c_2 & c_2 & -c_6 & -c_6 & c_2 & -c_2 & c_6 \\ c_7 & -c_5 & c_3 & -c_1 & c_1 & -c_3 & c_5 & -c_7 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \tag{5.2}$$

gdzie współczynniki c_k wynoszą: $c_k = \cos\frac{k\pi}{16}$.

$$\begin{bmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \\ y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \alpha_k \left[\begin{array}{cccc:cccc} c_0 & c_0 & c_0 & c_0 & 0 & 0 & 0 & 0 \\ c_2 & c_6 & -c_6 & -c_2 & 0 & 0 & 0 & 0 \\ c_4 & -c_4 & -c_4 & c_4 & 0 & 0 & 0 & 0 \\ c_6 & -c_2 & c_2 & -c_6 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & c_1 & c_3 & c_5 & c_7 \\ 0 & 0 & 0 & 0 & c_3 & -c_7 & -c_1 & -c_5 \\ 0 & 0 & 0 & 0 & c_5 & -c_1 & c_7 & c_3 \\ 0 & 0 & 0 & 0 & c_7 & -c_5 & c_3 & -c_1 \end{array}\right] \cdot \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \\ x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \tag{5.3}$$

Postępując dalej w ten sam sposób można zredukować liczbę mnożeń do zaledwie 22, co zaowocuje zależnością (5.4).

$$\begin{bmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \\ y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \alpha_k \left[\begin{array}{cccc:cccc} c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -c_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_2 & c_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_6 & -c_2 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & c_1 & c_3 & c_5 & c_7 \\ 0 & 0 & 0 & 0 & c_3 & -c_7 & -c_1 & -c_5 \\ 0 & 0 & 0 & 0 & c_5 & -c_1 & c_7 & c_3 \\ 0 & 0 & 0 & 0 & c_7 & -c_5 & c_3 & -c_1 \end{array}\right] \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} \tag{5.4}$$

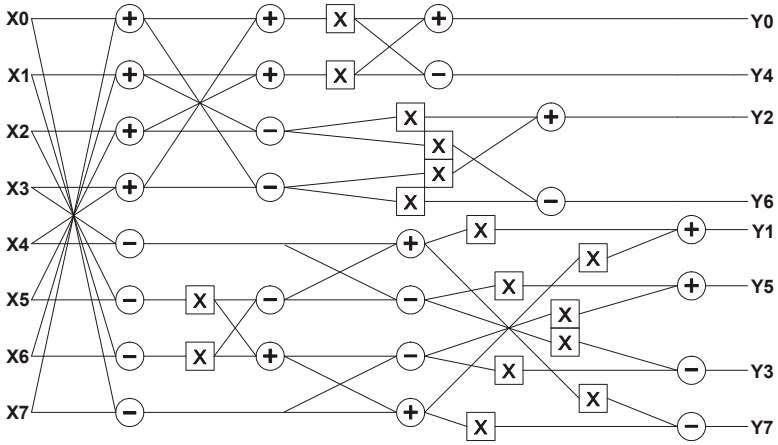
gdzie wektor w_n ma postać (5.5).

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \begin{bmatrix} x_0 + x_7 + x_3 + x_4 + x_1 + x_6 + x_2 + x_5 \\ x_0 + x_7 + x_3 + x_4 - x_1 - x_6 - x_2 - x_5 \\ x_0 + x_7 - x_3 - x_4 \\ x_1 + x_6 - x_2 - x_5 \\ x_0 - x_7 \\ x_1 - x_6 \\ x_3 - x_4 \\ x_2 - x_5 \end{bmatrix} \tag{5.5}$$

Tak otrzymana macierz współczynników, pomimo że wymaga mniejszej liczby mnożeń, to jednak w przypadku realizacji sprzętowej powoduje wystąpienie problemu synchronizacji zadań obliczeniowych, wynikających z asymetrii lokalizacji zer w macierzy współczynników c_k we wzorze (5.4). W związku z tym liczba potrzebnych mnożeń do wyliczenia poszczególnych składowych wektora y_n będzie różna. Konsekwencją tego jest konieczność dobrania odpowiednich opóźnień w przypadku sprzętowej implementacji takiej niesymetrycznej dyskretnej transformaty kosinusowej.

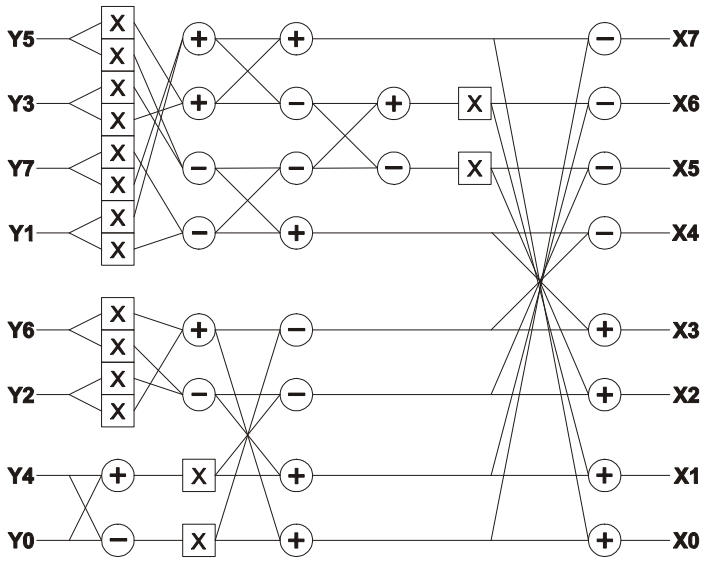
Dalsza redukcja wymaganej liczby mnożeń prowadzi do tak zwanych szybkich algorytmów dyskretnej transformacji kosinusowej FDCT (ang. *Fast DCT*).

Jednym z algorytmów FDCT jest algorytm Chena (Rys. 5.2 i 5.3) [48]. Liczba mnożeń została w nim zredukowana do 16, a liczba operacji dodawań-odejmowań do 26.

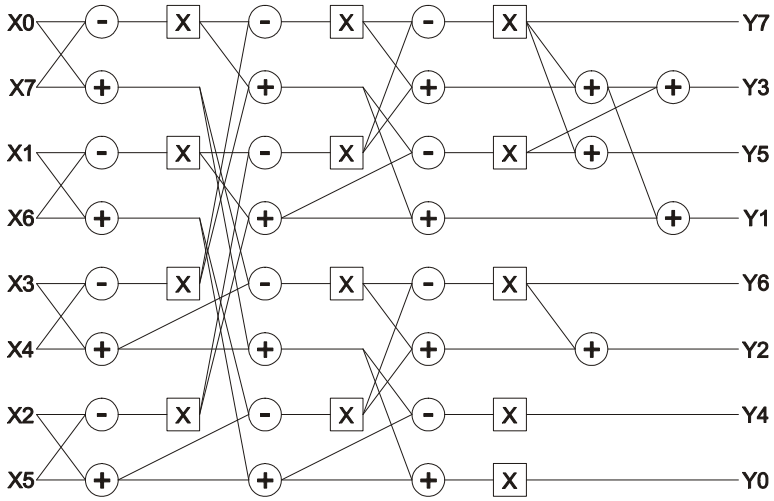


Rys. 5.2. Graficzny schemat przepływu danych dla transformaty DCT w algorytmie Chena

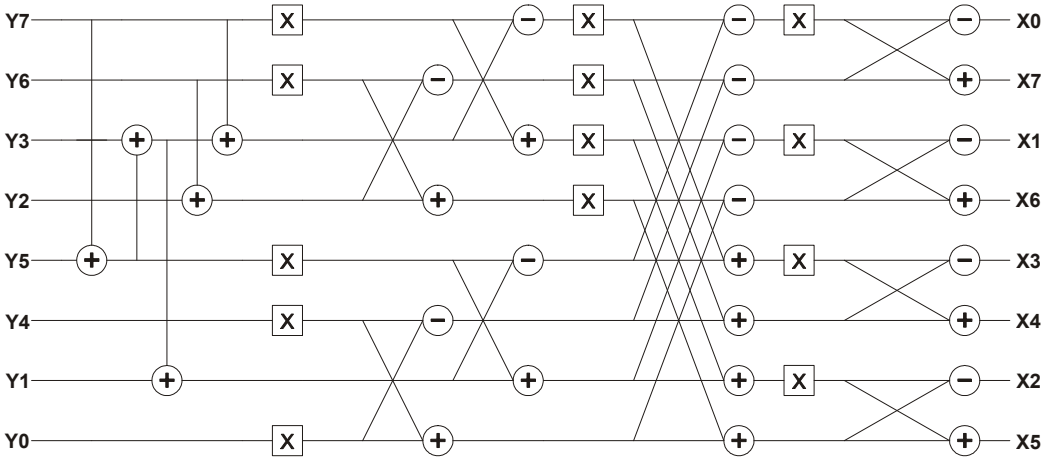
Jeszcze większą redukcję liczby potrzebnych operacji arytmetycznych otrzymano w algorytmie Lee [62] (Rys. 5.4 i 5.5), w którym liczba mnożeń została zredukowana do 13, a liczba dodawań-odejmowań do 29.



Rys. 5.3. Graficzny schemat przepływu danych dla transformaty IDCT w algorytmie Chena

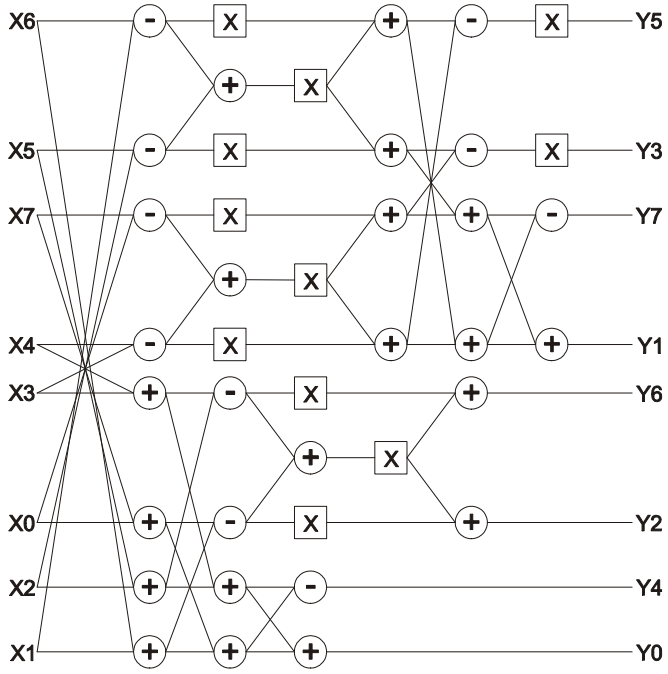


Rys. 5.4. Graficzny schemat przepływu danych dla transformaty DCT w algorytmie Lee



Rys. 5.5. Graficzny schemat przepływu danych dla transformaty IDCT w algorytmie Lee

Kolejnym algorytmem DCT jest algorytm Loefflera [2,3] (Rys. 5.6). W tym przypadku liczba wymaganych mnożeń wynosi 11, a liczba operacji dodawań-odejmowań jest taka sama jak dla algorytmu Lee.



Rys. 5.6. Graficzny schemat przepływu danych dla transformaty DCT w algorytmie Loefflera

5.2. Wyniki implementacji algorytmów FDCT

Podczas implementacji algorytmu 1D-DCT powstaje problem z jaką dokładnością należy stosować współczynniki mnożenia. Czy wystarczy dokładność rzędu 10^{-2}, czy też trzeba przyjąć współczynniki mnożenia z dokładnością rzędu wyższego. Szczególnie ważne jest, jak przyjęta dokładność przenosi się na wartości współczynników transformaty.

Dla dokładności odwzorowania współczynników mnożenia do drugiego miejsca po przecinku otrzymana maksymalna różnica pomiędzy współczynnikami transformacji obliczonymi analitycznie, a współczynnikami otrzymanymi z zaimplementowanego algorytmu Chena wynosi 12,75. Występuje ona przy maksymalnych wartościach wejściowych pikseli (x_0 do x_7 wynoszą maksymalnie 255). W tym przypadku wartość współczynnika transformacji y_0 wynosi 734, podczas gdy wartość tego współczynnika obliczona analitycznie wynosi 721,249. Teoretycznie maksymalna różnica przy dokładności odwzorowania współczynników mnożenia do drugiego miejsca po przecinku wynosi 20,4 czyli jest większa od wartości otrzymanej z zaimplementowanego algorytmu.

Jeżeli współczynniki mnożenia zostały zrealizowane z dokładnością do trzeciego miejsca po przecinku to maksymalna różnica pomiędzy współczynnikami transformacji obliczonymi analitycznie, a otrzymanymi z zaimplementowanego algorytmu wynosi 1,182. Przy tej dokładności maksymalna różnica wyznaczona teoretycznie wynosi 2,04.

Pomimo wcześniejszych rozważań pozostaje nadal problem odpowiedzi na pytanie o wystarczającą dokładność odwzorowania współczynników mnożenia. W związku z tym należy odpowiedzieć na pytanie jak otrzymane różnice pomiędzy poszczególnymi wartościami współczynników transformacji przenoszą się na wartości pikseli otrzymanych poprzez odwrotną dyskretną transformacje kosinusową IDCT.

W tabelach od 5.1 do 5.3 można zauważyć, że przy dokładności rzędu 10^{-2} maksymalna różnica *MD* (ang. *Maximal Difference*) pomiędzy wartością piksela wejściowego, a wartością piksela zrekonstruowanego przy pomocy zaimplementowanego algorytmu 1D-DCT Chena wynosi 5. Jeżeli dokładność odwzorowania współczynników mnożenia jest rzędu 10^{-3} to maksymalna różnica *MD* pomiędzy pikselami wynosi 2.

Tab. 5.1. Rozważania dla przykładowego wektora wejściowego

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Chena z dokładnością rzędu		Różnica pomiędzy y_n otrzymanym a y_n obliczonym		x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Chena	
		10^{-2}	10^{-3}	10^{-2}	10^{-3}	10^{-2}	10^{-3}
$x_0 = 252$	$y_0 = 645{,}5885$	$y_0 = 657$	$y_0 = 645$	-11,4115	0,58849	255	252
$x_1 = 250$	$y_1 = 55{,}33119$	$y_1 = 55$	$y_1 = 55$	0,33119	0,33119	255	250
$x_2 = 249$	$y_2 = 5{,}037243$	$y_2 = 5$	$y_2 = 5$	0,037243	0,03724	253	248
$x_3 = 247$	$y_3 = -27{,}3337$	$y_3 = -28$	$y_3 = -27$	0,66633	-0,33367	251	247
$x_4 = 207$	$y_4 = -22{,}6274$	$y_4 = -23$	$y_4 = -23$	0,372583	0,37258	211	206
$x_5 = 223$	$y_5 = -21{,}849$	$y_5 = -23$	$y_5 = -21$	1,151043	-0,848958	226	223
$x_6 = 159$	$y_6 = 36{,}18185$	$y_6 = 37$	$y_6 = 36$	-0,81815	0,181849	162	159
$x_7 = 239$	$y_7 = -32{,}817$	$y_7 = -33$	$y_7 = -33$	0,183041	0,183041	244	238

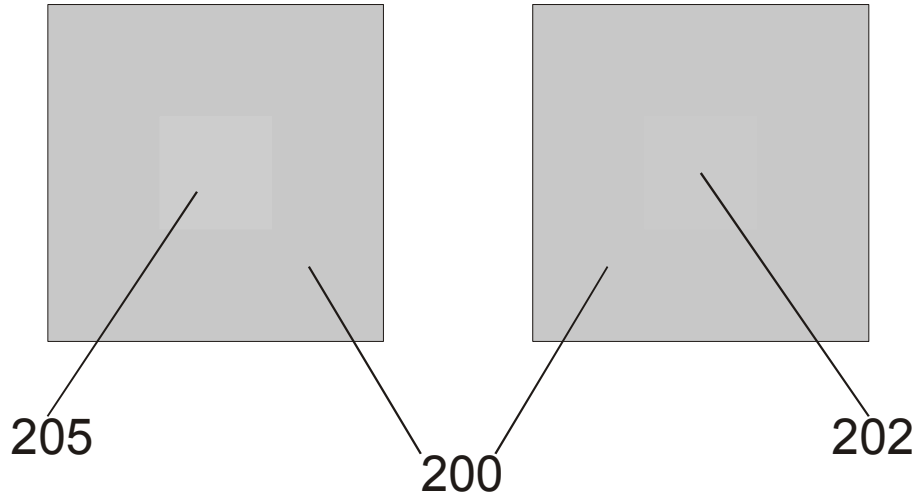
Tab. 5.2. Przypadek dla maksymalnej różnicy pomiędzy y_n otrzymanym a y_n obliczonym dla dokładności rzędu 10^{-3}

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Chena z dokładnością rzędu		Różnica pomiędzy y_n otrzymanym a y_n obliczonym		x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Chena	
		10^{-2}	10^{-3}	10^{-2}	10^{-3}	10^{-2}	10^{-3}
$x_0 = 253$	$y_0 = 393{,}15137$	$y_0 = 400$	$y_0 = 393$	-6,84862	0,15137	256	253
$x_1 = 33$	$y_1 = 83{,}52634$	$y_1 = 82$	$y_1 = 83$	1,5263376	0,526338	34	34
$x_2 = 247$	$y_2 = -25{,}132389$	$y_2 = -23$	$y_2 = -25$	-2,132389	-0,132389	250	247
$x_3 = 121$	$y_3 = 78{,}72797$	$y_3 = 80$	$y_3 = 78$	-1,2720269	0,72797	122	120
$x_4 = 87$	$y_4 = 46{,}669048$	$y_4 = 48$	$y_4 = 47$	-1,3309524	-0,330952	87	87
$x_5 = 219$	$y_5 = 123{,}22545$	$y_5 = 126$	$y_5 = 123$	-2,7745484	0,2254516	223	217
$x_6 = 123$	$y_6 = 157{,}36061$	$y_6 = 159$	$y_6 = 157$	-1,639385	0,3606145	127	123
$x_7 = 29$	$y_7 = 41{,}818001$	$y_7 = 42$	$y_7 = 43$	-0,1819986	-1,1819986	32	29

Tab. 5.3. Przykład dla małych różnic pomiędzy sąsiednimi pikselami

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Chena z dokładnością rzędu		Różnica pomiędzy y_n otrzymanym a y_n obliczonym		x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Chena	
		10^{-2}	10^{-3}	10^{-2}	10^{-3}	10^{-2}	10^{-3}
$x_0 = 191$	$y_0 = 530{,}3301$	$y_0 = 540$	$y_0 = 530$	-9,67	0,3301	194	190
$x_1 = 190$	$y_1 = 6{,}442323$	$y_1 = 6$	$y_1 = 6$	0,4423	0,4423	194	190
$x_2 = 189$	$y_2 = 0$	$y_2 = 0$	$y_2 = 0$	0	0	193	189
$x_3 = 188$	$y_3 = 0{,}673455$	$y_3 = 0$	$y_3 = 0$	0,67345	0,67345	191	187
$x_4 = 187$	$y_4 = 0$	$y_4 = 0$	$y_4 = 0$	0	0	190	186
$x_5 = 186$	$y_5 = 0{,}200903$	$y_5 = 0$	$y_5 = 0$	0,2009	0,2009	189	185
$x_6 = 185$	$y_6 = 0$	$y_6 = 0$	$y_6 = 0$	0	0	188	184
$x_7 = 184$	$y_7 = 0{,}050702$	$y_7 = 0$	$y_7 = 0$	0,0507	0,0507	188	184

Na rysunku 5.7 widać, że różnica poziomu szarości pikseli równa 2 jest prawie niezauważalna natomiast różnica równa 5 jest już różnicą widoczną.



a) dokładność 10^{-2} **b) dokładność 10^{-3}**

Rys. 5.7. Maksymalne różnice pomiędzy pikselami dla dokładności 10^{-2} i 10^{-3}

Implementacja DCT Chena z dokładnością rzędu 10^{-3} w układzie XCV200BG352-6 firmy Xilinx zajmuje 574 z 2352 bloków SLICE tego układu (co stanowi 24% tych zasobów) oraz 958 z 4704 bloków LUT (20% tych zasobów układu). Maksymalna osiągalna częstotliwość dla tej implementacji wynosi 76,115 MHz. Dla implementacji DCT Chena z dokładnością odwzorowania współczynników mnożenia rzędu 10^{-2} częstotliwość ta sięga 105,831 MHz (również dla układu XCV200BG352-6). W tym przypadku bloki SLICE są zajęte w 16% (382 z 2352) a bloki LUT w 13% (614 z 4704).

Istnieje jeszcze jedna możliwość, która będzie kompromisem pomiędzy szybkością działania zaimplementowanego algorytmu, a dokładnością rekonstrukcji pikseli. Wystarczy

zaimplementować algorytm DCT Chena z dokładnością odwzorowania współczynników mnożenia rzędu 10^{-2}, z wyjątkiem współczynnika mnożenia występującego przy wyznaczaniu współczynnika transformacji y_0.

Tab. 5.4. Rozważania dla przykładowego wektora wejściowego

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Chena z dokładnością rzędu 10^{-2} z wyjątkiem współczynnika dla y_0	Różnica pomiędzy y_n otrzymanym a y_n obliczonym	x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Chena
$x_0 = 253$	$y_0 = 393{,}15137$	$y_0 = 393$	0,15137	255
$x_1 = 33$	$y_1 = 83{,}52634$	$y_1 = 82$	1,526338	31
$x_2 = 247$	$y_2 = -25{,}132389$	$y_2 = -23$	-2,13239	247
$x_3 = 121$	$y_3 = 78{,}72797$	$y_3 = 80$	-1,27203	119
$x_4 = 87$	$y_4 = 46{,}669048$	$y_4 = 47$	-0,33095	84
$x_5 = 219$	$y_5 = 123{,}22545$	$y_5 = 126$	-2,77455	220
$x_6 = 123$	$y_6 = 157{,}36061$	$y_6 = 159$	-1,63939	124
$x_7 = 29$	$y_7 = 41{,}818001$	$y_7 = 42$	-0,182	30

W tabelach 5.4 i 5.5 widać, że przy dokładności odwzorowania współczynników rzędu 10^{-2} (z wyjątkiem współczynnika mnożenia występującego przy wyznaczaniu y_0) maksymalna różnica *MD* pomiędzy wartością piksela wejściowego, a wartością piksela zrekonstruowanego wynosi 3. Otrzymana maksymalna różnica pomiędzy współczynnikami transformacji obliczonymi analitycznie, a współczynnikami otrzymanymi z zaimplementowanego algorytmu Chena w tym przypadku wynosi 2,89. Implementacja DCT Chena z tak zmodyfikowaną dokładnością rzędu 10^{-2} w układzie XCV200BG352-6 firmy Xilinx zajmuje 402 z 2352 bloków SLICE (co stanowi 17% zasobów układu) oraz 648 z 4704 bloków LUT (13% zasobów układu). Maksymalna osiągnięta częstotliwość dla tej implementacji wynosi 93,932 MHz.

Tab. 5.5. Przykład dla małych różnic pomiędzy sąsiednimi pikselami

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Chena z dokładnością rzędu 10^{-2} z wyjątkiem współczynnika dla y_0	Różnica pomiędzy y_n otrzymanym a y_n obliczonym	x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Chena
$x_0 = 191$	$y_0 = 530{,}3301$	$y_0 = 530$	0,330086	190
$x_1 = 190$	$y_1 = 6{,}442323$	$y_1 = 6$	0,442323	190
$x_2 = 189$	$y_2 = 0$	$y_2 = 0$	0	189
$x_3 = 188$	$y_3 = 0{,}673455$	$y_3 = 0$	0,673455	187
$x_4 = 187$	$y_4 = 0$	$y_4 = 0$	0	186
$x_5 = 186$	$y_5 = 0{,}200903$	$y_5 = 0$	0,200903	185
$x_6 = 185$	$y_6 = 0$	$y_6 = 0$	0	184
$x_7 = 184$	$y_7 = 0{,}050702$	$y_7 = 0$	0,050702	184

Implementacja algorytmu 1D-DCT Lee z dokładnością rzędu 10^{-3} w układzie XCV200BG352 zajmuje 1277 z 2352 bloków SLICE (czyli 54% bloków SLICE) oraz 2237 z 4704 bloków LUT (czyli 47% LUT). Maksymalna osiągalna częstotliwość dla implementacji algorytmu 1D-DCT Lee wynosi 35,668 MHz.

Algorytm 1D-DCT Lee został zaimplementowany z dokładnością odwzorowania współczynników mnożenia rzędu 10^{-3}. Zatem teoretycznie maksymalna różnica pomiędzy współczynnikami transformacji obliczonymi analitycznie a współczynnikami otrzymanymi z zaimplementowanego algorytmu nie powinna być większa niż 2,04. W tym przypadku maksymalna różnica wynosi 2,594. Maksymalna różnica *MD* pomiędzy wartością piksela wejściowego a odtworzoną wartością piksela wynosi 2.

Tab. 5.6. Przykładowy wektor wejściowy

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Lee	Różnica pomiędzy y_n otrzymanym a y_n obliczonym	x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Lee
$x_0 = 252$	$y_0 = 645{,}5885$	$y_0 = 646$	-0,411509	$x_0' = 251$
$x_1 = 250$	$y_1 = 55{,}33119$	$y_1 = 54$	1,3311912	$x_1' = 249$
$x_2 = 249$	$y_2 = 5{,}037243$	$y_2 = 5$	0,03724323	$x_2' = 247$
$x_3 = 247$	$y_3 = -27{,}3337$	$y_3 = -27$	-0,3336705	$x_3' = 247$
$x_4 = 207$	$y_4 = -22{,}6274$	$y_4 = -23$	0,372583	$x_4' = 207$
$x_5 = 223$	$y_5 = -21{,}849$	$y_5 = -22$	0,1510427	$x_5' = 223$
$x_6 = 159$	$y_6 = 36{,}18185$	$y_6 = 36$	0,1818488	$x_6' = 159$
$x_7 = 239$	$y_7 = -32{,}817$	$y_7 = -22$	0,1830408	$x_7' = 239$

Tab. 5.7. Przykład dla małych różnic pomiędzy sąsiednimi pikselami dla zaimplementowanego algorytmu Lee

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Lee	Różnica pomiędzy y_n otrzymanym a y_n obliczonym	x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Lee
$x_0 = 191$	$y_0 = 530{,}3301$	$y_0 = 530$	0,330086	$x_0' = 189$
$x_1 = 190$	$y_1 = 6{,}442323$	$y_1 = 4$	2,442323	$x_1' = 189$
$x_2 = 189$	$y_2 = 0$	$y_2 = 0$	0	$x_2' = 188$
$x_3 = 188$	$y_3 = 0{,}673455$	$y_3 = 0$	0,6734548	$x_3' = 187$
$x_4 = 187$	$y_4 = 0$	$y_4 = 0$	0	$x_4' = 187$
$x_5 = 186$	$y_5 = 0{,}200903$	$y_5 = 0$	0,200903	$x_5' = 186$
$x_6 = 185$	$y_6 = 0$	$y_6 = 0$	0	$x_6' = 185$
$x_7 = 184$	$y_7 = 0{,}050702$	$y_7 = 0$	0,0507023	$x_7' = 185$

Kolejnym algorytmem, który został zaimplementowany z dokładnością odwzorowania współczynników mnożenia rzędu 10^{-3}, jest algorytm 1D-DCT Loefflera. W przypadku tej implementacji maksymalna różnica wynosi 2,40562. Jest to wartość większa niż teoretyczna maksymalna różnica dla tej dokładności odwzorowania współczynników mnożenia.

Maksymalna różnica *MD* pomiędzy wartością piksela wejściowego, a jego rekonstrukcją wynosi 2, co można zaobserwować w tabelach 5.8 i 5.9.

Tab. 5.8. Przykład dla małych różnic pomiędzy sąsiednimi pikselami

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Loefflera z dokładnością rzędu 10^{-3}	Różnica pomiędzy y_n otrzymanym a y_n obliczonym	x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Loefflera
$x_0 = 191$	$y_0 = 530{,}3301$	$y_0 = 530$	0,330086	190
$x_1 = 190$	$y_1 = 6{,}442323$	$y_1 = 6$	0,442323	189
$x_2 = 189$	$y_2 = 0$	$y_2 = 0$	0	190
$x_3 = 188$	$y_3 = 0{,}673455$	$y_3 = -1$	1,673455	189
$x_4 = 187$	$y_4 = 0$	$y_4 = 0$	0	186
$x_5 = 186$	$y_5 = 0{,}200903$	$y_5 = 1$	-0,7991	185
$x_6 = 185$	$y_6 = 0$	$y_6 = 0$	0	185
$x_7 = 184$	$y_7 = 0{,}050702$	$y_7 = 0$	0,050702	185

Tab. 5.9. Przypadek dla maksymalnej różnicy pomiędzy y_n otrzymanym a y_n obliczonym dla dokładności rzędu 10^{-3}

x_n	y_n obliczone analitycznie	y_n otrzymane z zaimplementowanego algorytmu 1D-DCT Loefflera z dokładnością rzędu 10^{-3}	Różnica pomiędzy y_n otrzymanym a y_n obliczonym	x_n otrzymane z zaimplementowanego algorytmu 1D-IDCT Loefflera
$x_0 = 8$	$y_0 = 101{,}82338$	$y_0 = 102$	-0,17662	8
$x_1 = 16$	$y_1 = -51{,}53858$	$y_1 = -52$	0,461416	14
$x_2 = 24$	$y_2 = 0$	$y_2 = 0$	0	24
$x_3 = 32$	$y_3 = -5{,}387638$	$y_3 = -4$	-1,38764	30
$x_4 = 40$	$y_4 = 0$	$y_4 = 0$	0	41
$x_5 = 48$	$y_5 = -1{,}307223$	$y_5 = -1$	-0,60722	48
$x_6 = 56$	$y_6 = 0$	$y_6 = 0$	0	57
$x_7 = 64$	$y_7 = -0{,}405619$	$y_7 = 2$	-2,40562	63

Implementacja algorytmu 1D-DCT Loefflera z dokładnością rzędu 10^{-3} w układzie XCV200BG352-6 firmy Xilinx zajmuje 508 z 2352 bloków SLICE (co stanowi 21% zasobów układu) oraz 877 z 4704 bloków LUT (18% zasobów układu). Maksymalna osiągnięta częstotliwość dla implementacji tego algorytmu wynosi 79,195 MHz.

Porównując parametry osiągane przez zaimplementowane algorytmy Loefflera, Chena oraz Lee najlepszym algorytmem z dokładnością odwzorowania współczynników mnożenia rzędu 10^{-3} jest algorytm Loefflera. Przetwarzanie jednego wektora wejściowego wymaga 6 cykli zegara zarówno dla algorytmu Loefflera jak i dla algorytmu Chena, natomiast algorytm Lee wymaga 8 cykli. Pod kątem osiąganych maksymalnych częstotliwości lepszym algorytmem jest algorytm Loefflera, ponieważ maksymalna częstotliwość dla dokładności odwzoro-

wania współczynników mnożenia rzędu 10^{-3} wynosi 79,195 MHz a przy tej samej dokładności dla algorytmu Chena wynosi ona 76,115MHz a dla algorytmu Lee 35,668 MHz. Również pod względem liczby wykorzystanych bramek algorytm Loefflera jest korzystniejszy. Implementacja tego algorytmu zajmuje 13723 bramek w przypadku dokładności rzędu 10^{-3}, natomiast dla algorytmu Chena potrzebne jest 14514 bramek oraz 28257 bramek dla algorytmu Lee. Fakt ten również przemawia na korzyść implementacji algorytmu Loefflera. Dla zaimplementowanych algorytmów maksymalna różnica *MD* pomiędzy wartością pikseli wejściowych a wartością ich rekonstrukcji wynosi 2. Argumentem przemawiającym na korzyść algorytmu Chena jest maksymalna różnica pomiędzy współczynnikiem obliczonym analitycznie a współczynnikiem otrzymanym z zaimplementowanego algorytmu. Różnica ta wynosi 1,182 dla algorytmu Chena i 2,40562 dla algorytmu Loefflera. W przypadku algorytmu Lee różnica ta jest większa i wynosi 2,594.

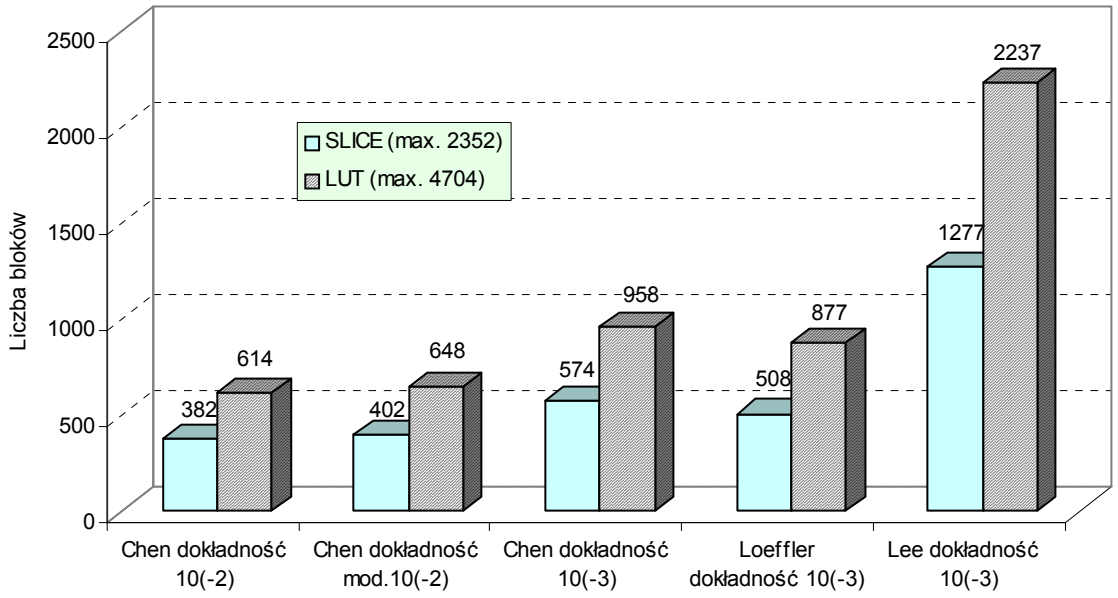
Jeżeli jednak porównać implementacje algorytmów Chena ze zmodyfikowaną dokładnością rzędu 10^{-2} i Loefflera z dokładnością rzędu 10^{-3} okazuje się, że przy zbliżonej różnicy pomiędzy y_n obliczonymi analitycznie a y_n otrzymanymi z zaimplementowanego algorytmu lepszy jest algorytm Chena. Maksymalna częstotliwość osiągnięta dla tej implementacji wynosi 93,932 MHz. Implementacja algorytmu Chena ze zmodyfikowaną dokładnością zajmuje mniej zasobów niż implementacja algorytmu Loefflera z dokładnością rzędu 10^{-3}. Maksymalna różnica pomiędzy wartością piksela wejściowego a jego rekonstrukcją wynosi 3 dla algorytmu Chena. Różnica ta jest prawie niezauważalna.

Tab. 5.10. Parametry implementacji algorytmów jednowymiarowej dyskretnej transformacji kosinusowej w układzie FPGA

XCV200BG352-6	Algorytm				
	Chena			Loefflra 10^{-3}	Lee 10^{-3}
	10^{-2}	Mod.10^{-2}	10^{-3}		
MD	5	3	2	2	2
max. różnica pomiędzy y_n obliczonymi analitycznie a y_n otrzymanymi	12,75	2,89	1,182	2,406	2,594
SLICE	16%	17%	24%	21%	54%
LUT	13%	13%	20%	18%	47%
Częstotliwość [MHz]	105,831	93,932	76,115	79,195	35,668

Podsumowując, najlepszym rozwiązaniem pod względem szybkości przetwarzania danych przy zachowaniu ich zadowalającej dokładności jest implementacja algorytmu Chena

z dokładnością odwzorowania współczynników mnożenia rzędu 10^{-2} (z wyjątkiem współczynnika mnożenia występującego przy wyznaczaniu y_0). Algorytm ten, opracowany przez autora pracy, został użyty do skonstruowania dwuwymiarowej DCT.



Rys. 5.8. Zajętość zasobów układu XCV200BG352 w zależności od implementowanego algorytmu

Maksymalna różnica pomiędzy współczynnikiem y_n obliczonym analitycznie a współczynnikiem y_n otrzymanym z zaimplementowanego algorytmu nie przekroczyła 5,939. Maksymalna różnica *MD* pomiędzy wartością piksela wejściowego o jego rekonstrukcją otrzymaną z zaimplementowanego złożenia 2D-DCT i 2D-IDCT wyniosła 3. Przykładowe otrzymane wyniki przedstawione są w tabelach 5.11, 5.12 i 5.13.

Tab. 5.11. Blok pikseli wejściowych dla 2D-DCT i jego rekonstrukcja otrzymana z zaimplementowanego algorytmu 2D-IDCT

191	190	189	188	187	186	185	184
190	189	188	187	186	185	184	183
189	188	187	186	185	184	183	182
188	187	186	185	184	183	182	181
187	186	185	184	183	182	181	180
186	185	184	183	182	181	180	179
185	184	183	182	181	180	179	178
184	183	182	181	180	179	178	177

$\Rightarrow$

189	189	187	187	185	184	183	182
188	189	186	186	184	183	183	181
188	188	186	186	183	183	182	181
187	187	185	185	182	182	181	180
186	186	184	184	182	181	181	179
185	185	183	183	180	180	179	178
184	184	182	182	180	179	179	177
184	184	182	182	180	179	178	177

Tab. 5.12. Wartości wyjściowe 2D-DCT obliczone analitycznie dla bloku wejściowego z tabeli 5.11

1472	15,8646	4,3296	1,9048	0	0,5682	0	0,1434
18,2216	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1,9048	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0,5682	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0,1434	0	0	0	0	0	0	0

Tab. 5.13. Wartości wyjściowe otrzymane z zaimplementowanego algorytmu 2D-DCT Chena ze zmodyfikowaną dokładnością rzędu 10^{-2} dla bloku wejściowego z tabeli 5.11

1471	16	0	0	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
−1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
−1	0	0	0	0	0	0	0
−1	0	0	0	0	0	0	0

Zaimplementowana dwuwymiarowa dyskretna transformata kosinusowa oparta na algorytmie Chena ze zmodyfikowaną dokładnością odwzorowania 10^{-2} charakteryzuje się taką samą maksymalną różnicą pomiędzy wartością piksela wejściowego a jego rekonstrukcją jak w przypadku implementacji 1D-DCT opartej na tym samym algorytmie. Maksymalna osiągnięta częstotliwość wynosi 91,158 MHz. Zaimplementowany algorytm 2D-DCT w układzie XCV200BG352-6 zajmuje 1595 bloków SLICE co stanowi 67% zasobów układu oraz 1628 bloków LUT (34% zasobów układu). Taka zajętość układu przekłada się na 35406 bramek. Różnica pomiędzy wartością współczynnika wyjściowego transformaty wyznaczoną analitycznie a wartością otrzymaną z zaimplementowanego algorytmu wynosi 5,939. Pojedynczy zaimplementowany blok dwuwymiarowej dyskretnej transformacji kosinusowej może przetwarzać w ciągu sekundy 41 obrazów o rozdzielczości 1920×1152 pikseli.

5.3. Podsumowanie

Podstawowy algorytm jednowymiarowej dyskretnej transformacji kosinusowej, który przetwarza wektor złożony z 8 pikseli, wykorzystuje 64 operacje mnożenia oraz 56 operacji dodawania. Z porównania wykorzystania zasobów sprzętowych układu programowalnego FPGA przez układ mnożący i przez układ dodający wynika, że mnożenie jest operacją wykorzystującą więcej zasobów układu programowalnego niż operacja dodawania. Z punktu widzenia sprzętowej implementacji korzystniejsze jest zatem zastosowanie algorytmu DCT o zredukowanej liczbie operacji mnożenia kosztem nieznacznego zwiększenia liczby operacji dodawania. Jednak nie zawsze algorytm wykorzystujący mniejszą liczbę operacji jest algorytmem najlepszym. Przy realizacji algorytmu dyskretnej transformacji kosinusowej, ważnym aspektem jest również dokładność otrzymywanych wyników, która w znaczący sposób wpływa na dokładność rekonstrukcji skompresowanego obrazu. W związku z tym, głównym kryterium decydującym o wyborze algorytmu dyskretnej transformacji kosinusowej, przy zbliżonych parametrach sprzętowej implementacji i przy tej samej dokładności odwzorowania współczynników mnożenia, była maksymalna różnica pomiędzy współczynnikami 1D-DCT obliczonymi analitycznie a współczynnikami otrzymanymi z zaimplementowanego w FPGA algorytmu. Z punktu widzenia działania systemu, w skład którego wchodzi blok dyskretnej transformacji kosinusowej, ważna jest również szybkość przetwarzania danych wejściowych. Biorąc pod uwagę powyższe zagadnienia najkorzystniejszym rozwiązaniem jest zastosowanie algorytmu z układami mnożącymi o różnej dokładności zastosowanych współczynników mnożenia, stanowiącego kompromis pomiędzy szybkością przetwarzania a dokładnością otrzymywanych wyników. O tym, które współczynniki zostały zaimplementowane z mniejszą dokładnością decydował ich wpływ na wartości otrzymywane w wyniku dyskretnej transformacji kosinusowej. Opracowana przez autora modyfikacja polega na implementacji współczynnika y_0 z dokładnością 10^{-3} i pozostałych współczynników z dokładnością 10^{-2}.

Działanie takie pozwoliło na znaczną redukcję zajętości zastosowanego układu programowalnego FPGA, przy nieznacznym pogorszeniu otrzymanych wartości wyjściowych, Przy czym różnica pomiędzy pikselem zrekonstruowanym a jego pierwowzorem pozostała niezauważalna. Ponadto należy zauważyć, że modyfikacja algorytmu Chena, zaproponowana przez autora i uwzględniająca te założenia, pozwala na sprzętową implementację zapewniając jednocześnie małą różnicę pomiędzy obrazem pierwotnym a obrazem zdekompresowanym.

6. Implementacje operacji kwantyzacji i dekwantyzacji

Kolejnym etapem algorytmów JPEG oraz MPEG jest kwantyzacja współczynników otrzymanych z transformacji 2D-DCT. Etap ten opiera się na dzieleniu współczynników wyjściowych 2D-DCT przez macierz kwantyzacji.

6.1. Kwantyzacja i dekwantyzacja w standardzie MPEG-2

W standardzie MPEG-2 wykorzystywana jest tzw. zmienna kwantyzacja skalarna [32, 44, 51], która opiera się na skalowaniu przy pomocy parametru *quantizer_scale*, który może być określany indywidualnie dla każdego obrazu, warstwy czy nawet makrobloku. Taka kwantyzacja pozwala na osiągnięcie większego stopnia kompresji, jak również pomaga utrzymać stałą prędkość bitową zarówno kodera jak i dekodera. Parametr *quantizer_scale* jest określany na podstawie dwóch parametrów transmitowanych w nagłówku obrazu, warstwy lub makrobloku: *q_scale_type* oraz *quantizer_scale_code*. Jeżeli *q_scale_code* = 0 to wtedy parametr *quantizer_scale* jest określany na podstawie wyrażenia (6.1).

$$quantizer\ \_\ scale = 2 \times quantizer\ \_\ scale\ \_\ code \tag{6.1}$$

W przypadku, gdy parametr *q_scale_type* = 1 *quantizer_scale* jest określany na podstawie tabeli 6.1.

W standardzie MPEG-2 rozróżniane są dwa tryby kodowania: kodowanie wewnątrzobrazowe oraz kodowanie międzyobrazowe. W obu przypadkach definicja procesu kwantyzacji jest inna. W trybie międzyobrazowym współczynniki dyskretnej transformaty kosinusowej [32] są poddane kwantyzacji opisanej wyrażeniem (6.2).

$$NonIntraq_{ij} = \frac{32 \times y_{ij}}{2 \times quantizer\_scale \times NonIntraQ_{ij}} \tag{6.2}$$

gdzie:

$NonIntraq_{ij}$ – współczynniki DCT po kwantyzacji w trybie międzyobrazowym,

$NonIntraQ_{ij}$ – elementy macierzy kwantyzacji trybu międzyobrazowego,

y_{ij} – współczynniki DCT,

quantizer_scale – parametr skalowania.

Dla trybu wewnątrzobrazowego proces kwantyzacji współczynników dotyczących składowych AC dyskretnej transformaty kosinusowej jest opisany wyrażeniem (6.3)

$$Intraq_{ij} = \frac{32 \times y_{ij}}{2 \times quantizer\_scale \times IntraQ_{ij}} \text{ dla } i, j \neq 0 \tag{6.3}$$

gdzie:

$Intraq_{ij}$ – współczynniki DCT po kwantyzacji w trybie wewnątrzobrazowym,

$IntraQ_{ij}$ – elementy macierzy kwantyzacji trybu wewnątrzobrazowego.

Tab. 6.1. Relacja pomiędzy parametrem *quantizer_scale* a parametrem q*uantizer_scale_code*

quantizer_scale_code	*quantizer_scale*	*quantizer_scale_code*	*quantizer_scale*
0	Zarezerwowane	16	24
1	1	17	28
2	2	18	32
3	3	19	36
4	4	20	40
5	5	21	44
6	6	22	48
7	7	23	52
8	8	24	56
9	10	25	64
10	12	26	72
11	14	27	80
12	16	28	88
13	18	29	96
14	20	30	104
15	22	31	112

Współczynnik y_{00} otrzymany w wyniku dwuwymiarowej dyskretnej transformacji kosinusowej DCT jest związany ze składową stałą i nazywany współczynnikiem DC. Analogicznie współczynniki dotyczące pozostałych składowych są krótko określane mianem współczynników AC. Krok kwantyzacji w przypadku współczynników DC otrzymanych z 2D-DCT [26] w trybie wewnątrzobrazowym jest uzależniony od precyzji. Precyzja DC oznacza liczbę bitów użytych do reprezentacji współczynników DC po procesie kwantyzacji. W standardzie MPEG-2 dopuszczone są 3 rodzaje precyzji: 8-bitowa, 9-bitowa oraz 10-bitowa. Współczynniki DC w trybie wewnątrzobrazowym są poddane procesowi kwantyzacji zgodnie z zależnością (6.4).

$$Intraq_{00} = \frac{y_{00}}{k} \tag{6.4}$$

gdzie:

y_{00} – współczynnik DC transformacji 2D-DCT,

$Intraq_{00}$ – współczynnik DC 2D-DCT po kwantyzacji w trybie wewnątrzobrazowym,

$k = 8$ – dla 8 bitowej precyzji,

$k = 4$ – dla 9 bitowej precyzji,

$k = 2$ – dla 10 bitowej precyzji.

W standardzie MPEG-2 są określone typowe macierze kwantyzacji dla współczynników transformacji, w zależności od trybu kodowania, uwzględniające własności systemu wizyjnego człowieka. Macierze te obowiązują zarówno dla współczynników DCT luminancji jak i współczynników DCT chrominancji.

Procesem, dzięki któremu można odtworzyć skwantowane wartości jest dekwantyzacja. W trybie międzyobrazowym dekwantyzację można opisać wyrażeniem (6.5).

$$NonIntra\hat{y}_{ij} = \frac{1}{32}\left(2 \times NonIntraq_{ij} + sign_{ij}\right) \times quantizer\_scale \times NonIntraQ_{ij} \tag{6.5}$$

gdzie:

$sign_{ij}$ = -1 dla $NonIntraq_{ij} < 0$,

0 dla $NonIntraq_{ij} = 0$,

1 dla $NonIntraq_{ij} > 0$,

$NonIntra\,\hat{y}_{ij}$ – rekonstrukcja współczynników DCT.

W trybie wewnątrzobrazowym współczynniki AC można zrekonstruować na podstawie zależności (6.6).

$$Intra\hat{y}_{ij} = \frac{2}{32} \times Intraq_{ij} \times IntraQ_{ij} \times quantizer\_scale \tag{6.6}$$

gdzie:

$Intra\hat{y}_{ij}$ – rekonstrukcja współczynników DCT w trybie wewnątrzobrazowym.

Współczynniki DC w tym trybie można zrekonstruować za pomocą prostego przemnożenia wartości otrzymanej z procesu kwantyzacji przez współczynnik *k* odpowiedni dla danej precyzji.

6.2. Implementacje kwantyzatora i dekwantyzatora

Procesy kwantyzacji i dekwantyzacji zostały zaimplementowane w układzie FPGA przy następujących założeniach dotyczących parametrów *q_scale_type*, *quantizer_scale_code* oraz *quantizer_scale*:

- Jeżeli *q_scale_type* = 0 oraz *quantizer_scale_code* = 8 to zgodnie z zależnością (29) *quantizer_scale* =16.
- Jeżeli *q_scale_type* = 1 oraz *quantizer_scale_code* = 12 to zgodnie z tabelą 6.1 *quantizer_scale* = 16.

Przy powyższych założeniach wyrażenia (6.2), (6.3), (6.4) oraz (6.6) można uprościć odpowiednio do postaci (6.7), (6.8), (6.9) oraz (6.10)

$$NonIntraq_{ij} = \frac{y_{ij}}{NonIntraQ_{ij}} \tag{6.7}$$

$$Intraq_{ij} = \frac{y_{ij}}{IntraQ_{ij}} \text{ dla } i, j \neq 0 \tag{6.8}$$

$$NonIntra\hat{y}_{ij} = NonIntraQ_{ij} \times (NonIntraq_{ij} + 0{,}5sign_{ij}) \tag{6.9}$$

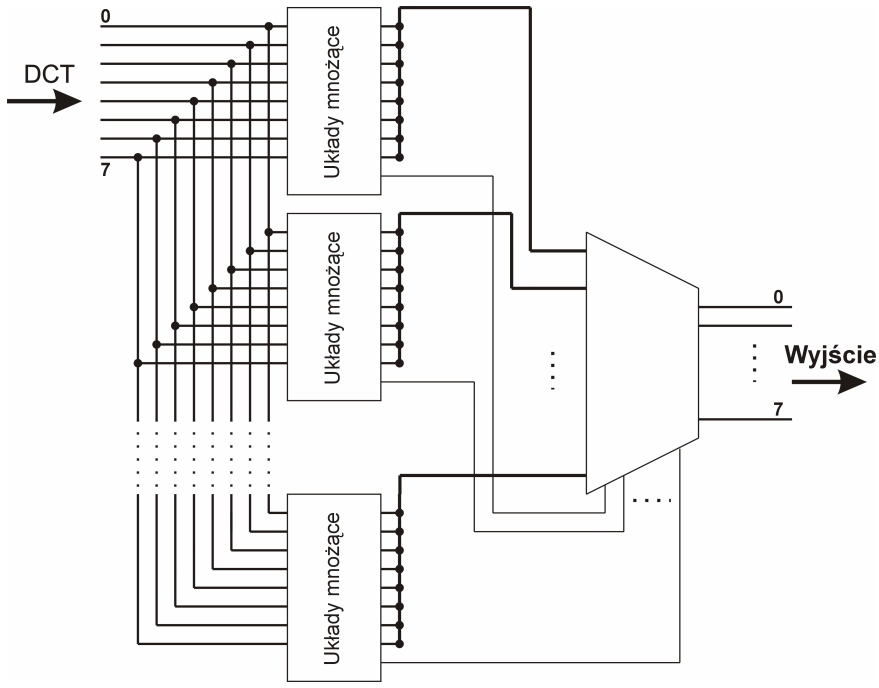
$$Intra\hat{y}_{ij} = IntraQ_{ij} \times Intraq_{ij} \tag{6.10}$$

Parametry *q_scale_type*, *quantizer_scale_code* jak również *quantizer_scale* nie mają wpływu na sposób kwantowania współczynnika DC w trybie wewnątrzobrazowym. Jedynym

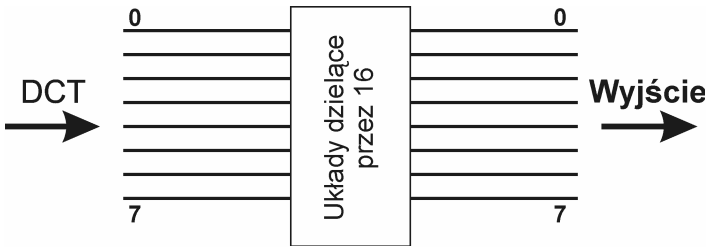
parametrem warunkującym kwantyzację DC w tym trybie, jest precyzja reprezentacji skwantowanego współczynnika. Dla potrzeb implementacji założono, że parametr k wynikający z precyzji, będzie równy 8. Jest to równoznaczne z przyjęciem 8-bitowej reprezentacji skwantowanego współczynnika DC otrzymanego z dwuwymiarowej dyskretnej transformacji kosinusowej. Wynikiem powyższego założenia jest przekształcenie wyrażenia (6.4) do wyrażenia (6.11).

$$Intraq_{00} = \frac{y_{00}}{8} \tag{6.11}$$

Schematy ideowe zaimplementowanych kwantyzatorów trybu wewnątrzobrazowego i trybu międzyobrazowego zostały przedstawione na rysunkach 6.1 oraz 6.2.



Rys. 6.1. Schemat ideowy kwantyzatora dla trybu wewnątrzobrazowego



Rys. 6.2. Schemat ideowy kwantyzatora dla trybu międzyobrazowego

Do realizacji operacji dzielenia zostały wykorzystane układy mnożące bezmnożne *MM* [87, 90] opierające się o algorytmy: CSD, SS oraz CSD-SS.

W proponowanym rozwiązaniu kwantyzacji poddawanych jest jednocześnie 8 współczynników 2D-DCT. Konsekwencją tego będzie pojawianie się na wyjściu kwantyzatora po każdym cyklu zegara 8 skwantowanych współczynników DCT.

Implementacja kwantyzatora w układzie XCV200BG532(-6) firmy Xilinx zajmuje 1735 bloków SLICE (73% zasobów układu). Na taką zajętość układu składa się 649 wykorzystanych przerzutników Flip-Flop oraz 3003 bloków LUT (63% dostępnych bloków LUT w układzie), z czego 2664 bloki zajmuje logika, natomiast 339 jest wykorzystanych jako zasoby połączeniowe. Maksymalna osiągnięta częstotliwość dla tej implementacji wynosi 94,616 MHz.

Jeśli proponowane rozwiązanie byłoby zaimplementowane w układzie firmy Xilinx XCV2P125FF1704, to zostałoby wykorzystanych 1755 bloków SLICE, co stanowi 3% zasobów danego układu. Na takie wykorzystanie składa się 648 przerzutników typu Flip-Flop oraz 3042 bloków LUT (2% dostępnych bloków układu), z czego 2665 bloków wykorzystywałaby logika, a 377 byłoby wykorzystanych jako zasoby połączeniowe. Maksymalna osiągnięta częstotliwość dla tej implementacji wynosiłaby 232,829 MHz.

Podobnie jak w przypadku kwantyzatora, procesowi dekwantyzacji poddawane jest równocześnie 8 skwantowanych współczynników. Implementacja dekwantyzatora w układzie XCV200BG352 zajmuje 658 bloków SLICE (co stanowi 27% zasobów tego układu). Na taką zajętość składa się 1013 bloków LUT (21% dostępnych bloków). Maksymalna częstotliwość osiągnięta dla tej implementacji w układzie XCV200BG352 wynosi 92,259 MHz.

Jeśli dekwantyzator zostałby zaimplementowany w układzie XCV2P125FF1704, to zostałoby wykorzystane zaledwie 1% dostępnych bloków SLICE (653) a wśród nich mniej niż 1% bloków LUT (1014). Maksymalna osiągnięta częstotliwość wynosiłaby 231,535 MHz.

Powyższe rozwiązania oparte były na mnożarkach o stałych współczynnikach, przy czym współczynniki te były zakodowane w strukturze mnożarek. Innym sposobem realizacji kwantyzatora i dekwantyzatora jest wykorzystanie mnożarek wbudowanych w układach z rodziny Virtex-II Pro firmy Xilinx. Implementacja kwantyzatora jak i dekwantyzatora wymaga 8 takich mnożarek 18-bitowych. Oprócz mnożarek kwantyzator wykorzystuje 102 bloki SLICE. Na taką liczbę bloków SLICE składa się 120 przerzutników Flip-Flop oraz 187 bloków LUT. Maksymalna osiągnięta częstotliwość dla tej implementacji wynosi 589,623 MHz.

Tab. 6.2. Parametry implementacji kwantyzatora w układach XCV200BG352 oraz XCV2P125FF1704

	Kwantyzacja	
Układ	XCV200BG352	XCV2P125FF1704
SLICE	1729 (73%)	1755 (3%)
LUT	3003 (63%)	3042 (2%)
Częstotliwość [MHz]	94,616 (10,569 ns)	232,829 (4,295 ns)

Tab. 6.3. Parametry implementacji dekwantyzatora w układach XCV200BG352 oraz XCV2P125FF1704

	Dekwantyzacja	
Układ	XCV200BG352	XCV2P125FF1704
SLICE	658 (27%)	653 (1%)
LUT	1013 (21%)	1014 (<1%)
Częstotliwość [MHz]	92,259 (10,839 ns)	231,535 (4,319 ns)

Implementacja dekwantyzatora wymaga, oprócz 8 mnożarek, 42 bloki SLICE, na co składają się 82 przerzutniki Flip-Flop oraz 81 bloków LUT. Maksymalna częstotliwość w przypadku tej implementacji wyniosła 617,284 MHz.

Zaletą zaimplementowanego kwantyzatora jest możliwość jednoczesnego kwantowania 8 współczynników otrzymanych z dwuwymiarowej dyskretnej transformacji kosinusowej DCT. Na wyjściu wyniki pojawiają się po każdym cyklu zegara, zatem przetworzenie wszystkich 64 współczynników otrzymanych z transformacji 2D-DCT bloku o rozmiarach 8×8 pikseli zajmuje 8 cykli zegara od momentu pojawienia się na wyjściu pierwszych 8 wartości. Tak samo jak kwantyzator również dekwantyzator przetwarza jednocześnie 8 współczynników wejściowych. Niestety, konsekwencją takiego rozwiązania jest większe wykorzystanie zasobów układu niż dla rozwiązania, w którym przetwarzany jest tylko pojedynczy współczynnik otrzymany z 2D-DCT.

Implementacja kwantyzatora zajmuje 39422 bramki, co przekłada się na zajętość zasobów układu FPGA, odpowiednio: dla XCV200BG352 jest to 73% wszystkich zasobów, a dla XCV2P125FF1704 jest to 2%. Pozostałe parametry implementacji zostały przedstawione

w tabeli 6.2. W przypadku implementacji dekwantyzatora wykorzystywane jest 27% wszystkich zasobów układu XCV200BG352 lub zaledwie 1% zasobów układu XCV2P125FF1704. Z punktu widzenia osiągniętej częstotliwości oraz wykorzystanych zasobów najlepiej prezentuje się implementacja kwantyzatora i dekwantyzatora z wykorzystaniem wbudowanych 18-bitowych mnożarek. Parametry tych implementacji przedstawiają tabele 6.2, 6.3 i 6.4.

Tab. 6.4. Parametry implementacji kwantyzatora i dekwantyzatora z wykorzystaniem wbudowanych mnożarek 18-bitowych

	XCV2P125FF1704	
	Kwantyzacja	Dekwantyzacja
SLICE	102	42
LUT	187	81
Mnożarki 18x18	8	8
Częstotliwość [MHz]	589,623 (1,696 ns)	617,284 (1,619 ns)

6.3. Podsumowanie

Głównym założeniem przy implementacji procesów kwantyzacji oraz dekwantyzacji było zapewnienie zgodności funkcjonalnej określonej przez specyfikację standardu MPEG-2.

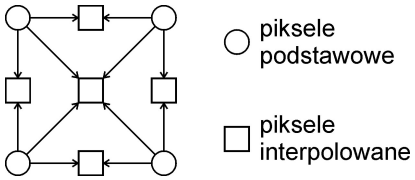
Z punktu widzenia wydajności przetwarzania danych wejściowych, ważnym aspektem było zrównoleglenie wykonywanych obliczeń. Zaimplementowane bloki kwantyzatora oraz dekwantyzatora przetwarzają wektory złożone z 8 danych wejściowych, a dane na wyjściu bloków pojawiają się co każdy cykl zegara. Takie zrównoleglenie obliczeń miało na celu przystosowanie bloków kwantyzacji/dekwantyzacji do przetwarzania danych dostarczanych z procesów 2D-DCT i 2D-IDCT. Dane te są wektorami złożonymi z 8 wartości. Taki sposób implementacji procesu kwantyzacji i dekwantyzacji zwiększa zajętość wykorzystywanych zasobów lecz równocześnie zwiększa szybkość przetwarzania. Zostały zaproponowane dwa sposoby realizacji procesów kwantyzacji i dekwantyzacji. Pierwszy z nich wykorzystuje bloki o stałym współczynniku mnożenia. Bloki te zostały tak zaimplementowane, żeby zajmowały jak najmniej zasobów. Drugim sposobem realizacji jest wykorzystanie bloków LUT oraz wbudowanych mnożarek o rozmiarach 18×18. Pod względem wydajności najlepsze jest rozwiązanie wykorzystujące wbudowane w układzie Virtex-II PRO firmy Xilinx mnożarki

18×18. Tak zaimplementowany blok kwantyzacji/dekwantyzacji jest w stanie przetworzyć znacznie więcej danych wejściowych niż rozwiązanie oparte na mnożarkach ze stałym współczynnikiem mnożenia. Jednak, jeśli docelowy układ, w którym zostaną zaimplementowane procesy kwantyzacji i dekwantyzacji, nie będzie miał możliwości wykorzystania wbudowanych mnożarek, układy mnożące będą realizowane w oparciu o oferowaną przez układ logikę programowalną. Taka realizacja bloków kwantyzacji i dekwantyzacji jest rozwiązaniem uniwersalnym dla wszystkich układów programowalnych FPGA. Pomimo, że jest to rozwiązanie mogące przetworzyć mniejszą liczbę danych wejściowych, to przy implementacji w układzie XCV2P125FF1704 firmy Xilinx, blok procesu kwantyzacji może przetwarzać w ciągu sekundy 842 obrazy o rozdzielczości 1920×1152 pikseli, a blok procesu dekwantyzacji może przetwarzać w ciągu sekundy 837 obrazów o rozdzielczości 1920×1152 pikseli. Zatem bloki te mogą być wykorzystywane w systemie pracującym w czasie rzeczywistym.

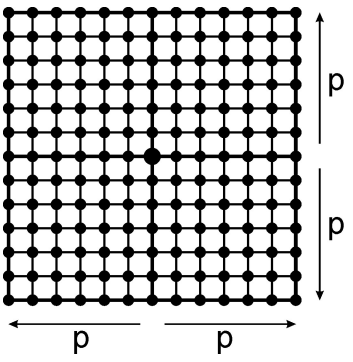
7. Implementacja procesu estymacji ruchu

7.1. Algorytmy estymacji ruchu

Algorytmy wyszukiwania najbardziej podobnego makrobloku w obrazie referencyjnym można podzielić na dwie grupy. Pierwszą z nich stanowią algorytmy przeszukiwania pełnego. Druga grupa to szybkie algorytmy przeszukiwania. Każdy algorytm estymacji ruchu może być zrealizowany na dwa sposoby: z dokładnością pikselową lub półpikselową. W przypadku przeszukiwania z dokładnością półpikselową zachodzi konieczność tworzenia dodatkowych interpolowanych bloków (Rys. 7.1). Niestety pociąga to za sobą większą złożoność obliczeniową.



Rys. 7.1. Zasada tworzenia interpolowanych bloków

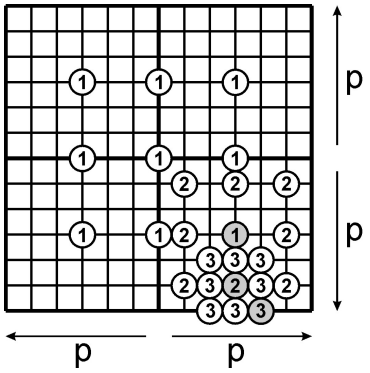


Rys. 7.2.Algorytm przeszukiwania pełnego

Algorytmem, który najlepiej znajduje najbardziej pasujące bloki jest algorytm pełnego przeszukiwania FS (ang. *Full Search*). W FS wszystkie możliwe przemieszczenia są używane do obliczenia funkcji celu. Zaletą tego algorytmu jest to, że zostanie znaleziony najlepiej pa-

sujący blok w obszarze przeszukiwania. Wadą algorytmu FS jest jego złożoność obliczeniowa wynosząca $O(p^2)$.

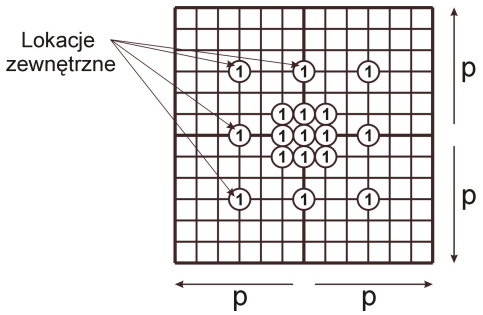
Algorytmami, które cechują się mniejszą złożonością są szybkie algorytmy estymacji. W przypadku tej grupy algorytmów złożoność obliczeniowa została zredukowana do $O(\log p)$. Jednym z takich algorytmów jest algorytm trójkrokowego przeszukiwania TSS (ang. *Three-Step Search*). W pierwszym etapie obliczana jest funkcja celu dla środka obszaru przeszukiwania, który jest traktowany jako najlepsze dopasowanie. Dalej wyznaczane jest 8 lokacji o następujących przesunięciach względem środka obszaru: $(0,p/2)$, $(p/2,p/2)$, $(p/2,0)$, $(p/2,-p/2)$, $(0,-p/2)$, $(-p/2,-p/2)$, $(-p/2,0)$, $(-p/2,p/2)$, dla których wyznaczane są funkcje celu w taki sam sposób jak to miało miejsce dla środka obszaru przeszukiwania. Spośród obliczonych wartości wybierana jest najlepsza z punktu widzenia zastosowanej funkcji celu. Lokacja, która została uznana za najlepszą jest uznana za najlepsze dopasowanie względem makrobloku. W drugim etapie wyznaczane jest kolejne 8 lokacji o przesunięciach względem nowego najlepszego dopasowania wynoszących odpowiednio: $(0,p/2^2)$, $(p/2^2,p/2^2)$, $(p/2^2,0)$, $(p/2^2,-p/2^2)$, $(0,-p/2^2)$, $(-p/2^2,-p/2^2)$, $(-p/2^2, 0)$, $(-p/2^2, p/2^2)$. Podobnie jak w pierwszym etapie wyznaczane są funkcje celu dla nowych lokacji. Następnie wybierana jest lokacja z najkorzystniejszą wartością funkcji celu. Wybrana lokacja staje się nowym najlepszym dopasowaniem. Taki proces wyszukiwania najlepszych lokacji trwa, aż krok przesunięcia będzie równy p/p oraz zostanie wyznaczone najlepsze dopasowanie z punktu widzenia zastosowanej funkcji celu przy tym kroku. W przypadku zastosowania algorytmu TSS istnieje możliwość, że lokacja uznana za najlepsze dopasowanie nie będzie równoznaczna z lokacją o absolutnie minimalnej funkcji celu w danym obszarze przeszukiwania. Zaletą TSS w porównaniu z algorytmem FS jest mniejsza złożoność wynosząca $O(8\log p)$.



Rys. 7.3. Algorytm TSS

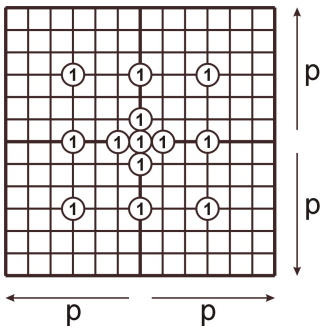
Algorytmem, który cechuje się większą wykrywalnością małego ruchu niż przy algorytmie TSS, jest nowy algorytm trójkrokowego przeszukiwania NTSS (ang. *New Three-Step Search*).

W pierwszym etapie algorytmu NTSS wyznaczane jest 8 lokacji na tych samych zasadach jak w pierwszym etapie algorytmu TSS. Dodatkowo wyznaczane jest 8 lokacji o następujących przesunięciach względem środka obszaru przeszukiwania: (0,1), (1,1), (1,0), (1,-1), (0,-1), (-1,-1), (-1,0), (-1,1). W kolejnym kroku obliczane są wartości funkcji celu dla 16 lokacji i porównywane z wartością funkcji celu wyznaczonej dla środka obszaru przeszukiwania. Jeżeli wartość funkcji jest najkorzystniejsza w przypadku punktu (0,0) to algorytm NTSS kończy działanie. Jeżeli najkorzystniejszą wartością cechuje się jedna z lokacji wokół środka obszaru przeszukiwania, wyznaczane jest wokół niej 8 nowych lokacji oraz wybierane jest nowe najlepsze dopasowanie będące wynikiem końcowym. Jeżeli najkorzystniejszą funkcją celu charakteryzuje się jedna z 8 lokacji zewnętrznych to kolejne etapy działania algorytmu NTSS są takie jak w przypadku algorytmu TSS.



Rys. 7.4. Pierwszy krok algorytmu NTSS

Kolejnym szybkim algorytmem estymacji, opartym na TSS, jest efektywny algorytm trójkrokowego przeszukiwania E3SS (ang. *Efficient Three-Step Search*).



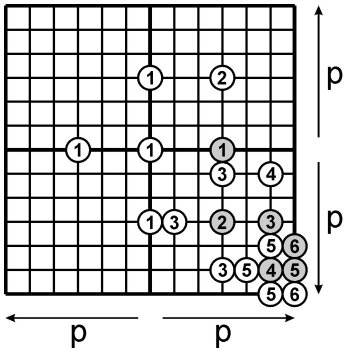
Rys. 7.5. Pierwszy etap algorytmu E3SS

Pierwszym etapem tego algorytmu jest wyznaczenie, podobnie jak w TSS, 8 punktów oraz 4 dodatkowych, leżących w centrum obszaru przeszukiwania, o współrzędnych: (0,1), (1,0), (0,-1) oraz (-1,0). Następnie dla wyżej z wymienionych lokacji obliczane są wartości funkcji celu, z których wybierana jest najkorzystniejsza. Dalsze działanie algorytmu jest uwarunko-

wane lokacją z najkorzystniejszą funkcją celu. Jeżeli wybranym punktem jest środek obszaru przeszukiwania to zakończone zostaje działanie algorytmu, a punkt (0,0) zostaje uznany za najlepsze dopasowanie. Jeżeli najkorzystniejszą funkcją celu charakteryzuje się jeden z 8 najbardziej zewnętrznych punktów to działanie algorytmu jest takie jak w przypadku algorytmu TSS. Jeżeli najkorzystniejszą funkcją celu cechuje się jedna z 4 lokacji wokół środka obszaru przeszukiwania, to punkt ten zostaje uznany za chwilowe najlepsze dopasowanie. Następnie zostają wyznaczone wokół niego 4 nowe lokacje o przesunięciach: (0,1), (1,0), (0,-1) oraz (-1,0), dla których obliczane są funkcje celu. Jeżeli najkorzystniejszą funkcją celu charakteryzuje się jedna z nowo wyznaczonych lokacji, to uznawana jest za chwilowe najlepsze dopasowanie i wyznaczane są wokół niej kolejne 4 lokacje na takiej samej zasadzie jak wcześniej. Działanie algorytmu trwa do momentu, gdy nowo wyznaczane lokacje nie będą miały korzystniejszej funkcji celu niż lokacja uznana za chwilowe najlepsze dopasowanie. W takim przypadku lokacja ta zostanie końcowym najlepszym dopasowaniem.

Kolejnym szybkim algorytmem estymacji ruchu jest algorytm dwuwymiarowego przeszukiwania logarytmicznego TDL (ang. *Two-Dimentional Logarithmic Search*). Podobnie jak dla TSS, dwuwymiarowe przeszukiwanie logarytmiczne nie gwarantuje znalezienia absolutnego minimum funkcji celu. W porównaniu z algorytmem TSS, algorytm TDL składa się z większej liczby etapów. Każdy z etapów TSS jest dzielony na dwa podetapy. Działanie takie zapewnia redukcję liczby wyznaczanych lokacji oraz obliczanych funkcji celu w trakcie pojedynczego etapu. Algorytm wykorzystuje funkcje *MSD* lub *MAD*. W TDL została wprowadzona wartość progowa. Jeżeli obliczona wartość funkcji celu jest mniejsza od określonej wartości progowej to proces przeszukiwania zostaje zakończony. Podobnie jak w TSS, pierwszy etap rozpoczyna się obliczeniem wartości funkcji celu dla środka obszaru przeszukiwania, który jest punktem startowym tego etapu. Wartość ta jest następnie porównywana z wartością progową. Jeśli wartość funkcji celu jest mniejsza niż wartość progowa to proces przeszukiwania zostaje przerwany a środek obszaru przeszukiwania stanowi najlepsze dopasowanie makrobloku. Jeśli wartość funkcji celu jest większa to zostają wyznaczone 4 nowe lokacje z krokiem równym $k=2^{(\log_2 p)-1}$: (0, k), (0,-k), (k, 0), (-k, 0). Dla powyższych punktów obliczane są funkcje celu i zostaje wybrana lokacja o minimalnej wartości funkcji. Następnie minimalna wartość jest porównywana z wartością funkcji celu dla środka obszaru. Jeżeli nowootrzymana wartość jest mniejsza, to zostaje porównana z wartością progową. Jeżeli wartość funkcji jest mniejsza od wartości progowej to proces przeszukiwania jest zatrzymywany, w przeciwnym przypadku proces jest kontynuowany. W kolejnym etapie wyznaczane są punkty leżące na

przekątnych względem środka obszaru będące najbliżej lokacji o najmniejszej wartości funkcji celu z poprzedniego etapu.



Rys. 7.6. Algorytm dwuwymiarowego przeszukiwania logarytmicznego

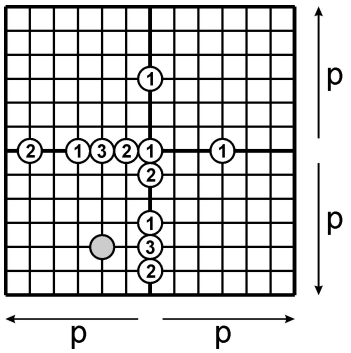
Proces przeszukiwania jest zatrzymywany, jeśli wartość funkcji celu dla jednego z nowo wyznaczonych punktów jest wartością mniejszą niż wartość otrzymana w pierwszym etapie i jednocześnie mniejsza niż wartość progowa. Jeżeli żaden z nowo wyznaczonych punktów w drugim etapie nie charakteryzuje się mniejszą wartością funkcji celu niż punkty z pierwszego etapu lub wartości funkcji celu 4 punktów z pierwszego etapu są mniejsze niż funkcja celu środka obszaru przeszukiwania rozpoczynany jest następny etap algorytmu. W etapie tym zostaje wyznaczony nowy krok będący połową kroku poprzedniego. Nowym punktem startowym zostaje lokacja, która w poprzednich etapach miała najmniejszą wartość funkcji celu. Proces wyszukiwania najlepszego dopasowania w tym etapie jest analogiczny jak w etapach poprzednich. Jeżeli nie będą spełnione warunki zatrzymania procesu wyszukiwania to będzie on kontynuowany aż do momentu osiągnięcia kroku równego 1. Algorytm dwuwymiarowego logarytmicznego przeszukiwania cechuje się mniejszą złożonością obliczeniową niż algorytm *TSS*, wynoszącą w najgorszym przypadku $O(6\log p)$.

Kolejnym algorytmem jest algorytm równoległego jednowymiarowego hierarchicznego przeszukiwania PHODS (ang. *Parallel Hierarchical One-Dimensional Search*). Podobnie jak w poprzednich algorytmach również i w PHODS pierwszy etap rozpoczyna się od obliczenia funkcji celu dla środka obszaru przeszukiwania, który staje się tymczasowym najlepszym dopasowaniem. Następnie wokół środka obszaru wyznaczane są dwa punkty w poziomie (0,p/2), (0,-p/2) oraz dwa punkty w pionie (p/2,0), (-p/2,0). Spośród tych punktów wybierane są dwa punkty, jeden na osi pionowej i jeden na osi poziomej, o najmniejszych wartościach funkcji celu. Przy pomocy wybranych punktów oraz wyrażenia (7.1) wyznaczana jest kolejna lokacja będąca nowym najlepszym dopasowaniem.

$$(x,0)+(0,y)=(x,y) \tag{7.1}$$

gdzie: $(x, 0)$ – punkt położony na linii poziomej,
$(0, y)$ – punkt położony na linii pionowej.

W drugim etapie wyznaczane są dwa punkty na osi pionowej w odległości $p/4$ od punktu $(0,y)$ o najmniejszej wartości funkcji celu z pierwszego etapu. Analogicznie wyznaczane są punkty na osi poziomej wokół punktu $(x,0)$ z poprzedniego etapu. W ten sposób na każdej osi są po 3 punkty, z których wybrane zostaną 2 - z najmniejszą wartością funkcji celu, po jednym na każdej osi. Proces wyznaczania punktów na obu osiach trwa do momentu, gdy krok z jakim wyznaczamy te punkty osiągnie minimum, czyli wartość p/p dla dokładności pikselowej lub $p/(2p)$ dla dokładności półpikselowej będzie najmniejsza. Przy pomocy punktów z najmniejszymi wartościami funkcji celu, otrzymanych przy tym kroku i wyrażenia (7.1) wyznaczona zostaje końcowa lokacja najlepszego dopasowania. Algorytm PHODS jest algorytmem o złożoności wynoszącej $O(4\log p)$.



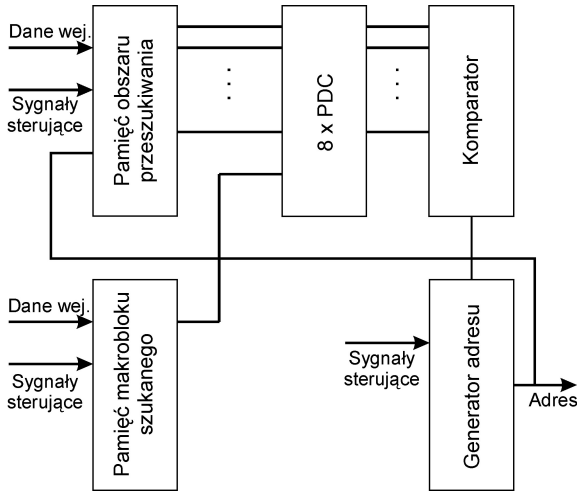
Rys. 7.7. Algorytm PHODS

7.2. Implementacja algorytmów TSS oraz E3SS

Algorytmem estymacji ruchu rekomendowanym do zastosowania w standardzie MPEG-2 jest algorytm trójkrokowego przeszukiwania. Algorytm ten w porównaniu z innymi szybkimi algorytmami przeszukiwania jest algorytmem o większej złożoności obliczeniowej, wynoszącej $O(\log p)$. Pod względem dokładności TSS ustępuje algorytmowi TDL, ale jednocześnie jest algorytmem prostszym w implementacji. Dodatkowo w TDL z góry narzucone jest jaką funkcję celu należy zastosować, natomiast algorytm trójkrokowego przeszukiwania pozostawia pod tym względem swobodę wyboru. Najmniejszą złożonością

obliczeniową wśród wymienionych algorytmów cechuje się PHODS. Jednak algorytm ten odznacza się największym prawdopodobieństwem, że lokacja wybrana jako najlepsze dopasowanie, nie będzie lokacją o absolutnie minimalnej wartości funkcji celu. Z powyższych powodów jako najlepszy algorytm dla celów implementacji w układach FPGA został wybrany algorytm trójkrokowego przeszukiwania TSS.

Jedną z decyzji, jakie należy podjąć przy implementacji algorytmów estymacji ruchu jest wybór funkcji celu. Z punktu widzenia dokładności otrzymywanych wyników, najlepsza jest różnica średniokwadratowa MSD. Wadą tej metody jest konieczność zastosowania operacji mnożenia o zmiennych współczynnikach. Funkcją, która nie wymaga mnożenia a jedynie końcowego skalowania jest średnia różnica bezwzględna MAD. Implementacja tej funkcji celu jest prostsza niż MSD. W związku z tym, że MAD sumuje różnice pomiędzy pikselem makrobloku i pikselem z obszaru przeszukiwania a pojedyncza różnica jest liczbą z zakresu od 0 do 255, wynik sumowania różnic może być liczbą 16-bitową. Funkcją, której wynik jest liczbą całkowitą z zakresu od 0 do 255 przy 8-bitowej reprezentacji piksela jest funkcja klasyfikacji różnicy pikseli PDC. Dodatkową zaletą tej funkcji jest zastosowanie wartości progowej dla różnicy pomiędzy pikselami makrobloku a pikselami obszaru referencyjnego. Wprowadzona wartość progowa jest wyznacznikiem dokładności wyszukiwania najlepszego dopasowania. Jeżeli wartość progowa zostanie potraktowana jako parametr, to będzie można sterować dokładnością wyszukiwania. Realizacja funkcji *PDC* bazuje na komparatorze, liczniku 8-bitowym oraz 8-bitowym układzie odejmującym.



Rys. 7.8. Schemat blokowy zaimplementowanego algorytmu TSS estymacji ruchu

Schemat blokowy zaimplementowanego algorytmu trójkrokowego przeszukiwania z funkcja celu *PDC* przedstawia rysunek 7.8.

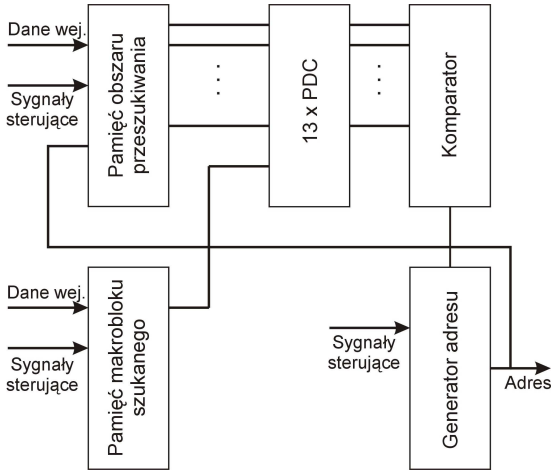
Implementacja algorytmu TSS w układzie XC2VP100-6FF1704 firmy Xilinx zajmuje 9 bloków pamięci BRAM (1% zasobów układu) oraz 526 bloków SLICE (1% zasobów układu). Na taką zajętość układu składa się 275 wykorzystanych przerzutników Flip-Flop oraz 902 bloków LUT (poniżej 1% dostępnych bloków LUT w układzie). Maksymalna osiągnięta częstotliwość pracy dla tej implementacji wynosi 95,120 MHz.

Zaimplementowany algorytm trójkrokowego przeszukiwania TSS jest w stanie przetwarzać 37 ramek o rozdzielczości 720×480 pikseli 8-bitowych w ciągu sekundy. Są to parametry przetwarzania przy implementacji używającej bloki pamięci wewnątrz układu XC2VP100-6FF1704. Zapis danych do pamięci obszaru przeszukiwania stanowi 58 % cykli całego procesu przetwarzania. Aby zminimalizować wpływ czasu zapisu danych do pamięci na opóźnienie pracy układu można użyć 2 bloki zaimplementowanego algorytmu TSS pracujące naprzemiennie. Takie działanie spowoduje dwukrotne zwiększenie szybkości przetwarzania. W przypadku rezygnacji z bloków pamięci zaimplementowanych wewnątrz układu FPGA większy wpływ na wartość opóźnienia będą miały transmisje sygnału poza układ FPGA.

Innym zaimplementowanym algorytmem estymacji ruchu jest efektywny algorytm trójkrokowego przeszukiwania E3SS. Algorytm ten charakteryzuje się większą wykrywalnością ruchu o niewielkich przemieszczeniach. Dodatkowym atutem algorytmu E3SS jest możliwość zakończenia przeszukiwania już w pierwszym etapie działania algorytmu z jednoczesnym wyznaczeniem najlepszego dopasowania. Taką samą zaletą cechuje się nowy algorytm trójkrokowego przeszukiwania. Jednak algorytm ten wymaga obliczania wartości funkcji celu dla większej liczby lokacji niż w przypadku algorytmu efektywnego trójkrokowego przeszukiwania. Schemat blokowy zaimplementowanego efektywnego algorytmu trójkrokowego przeszukiwania z funkcją celu PDC przedstawia rysunek 7.9.

Implementacja algorytmu E3SS w układzie XC2VP100FF1704 firmy Xilinx zajmuje 7 bloków pamięci BRAM oraz 623 bloków SLICE. Na taką zajętość układu składa się 364 wykorzystanych przerzutników Flip-Flop oraz 1029 bloków LUT (836 wykorzystywane przez logikę a 193 przez routing). Maksymalna osiągnięta częstotliwość pracy dla tej implementacji wynosi 106,883 MHz. Zaimplementowany algorytm efektywnego trójkrokowego przeszukiwania E3SS jest w stanie przetwarzać 44 ramki o rozdzielczości 720×480 pikseli 8-bitowych lub 29 ramki o rozdzielczości 800×600 pikseli 8-bitowych w ciągu sekundy. Są to parametry przetwarzania przy implementacji używającej bloków pamięci wewnątrz układu

XC2VP100-6FF1704. Zapis danych do pamięci obszaru przeszukiwania stanowi 58% cykli całego procesu przetwarzania. Aby zminimalizować wpływ opóźnienia wynikającego z zapisu danych do pamięci można użyć 2 bloków zaimplementowanego algorytmu E3SS pracujących naprzemiennie.



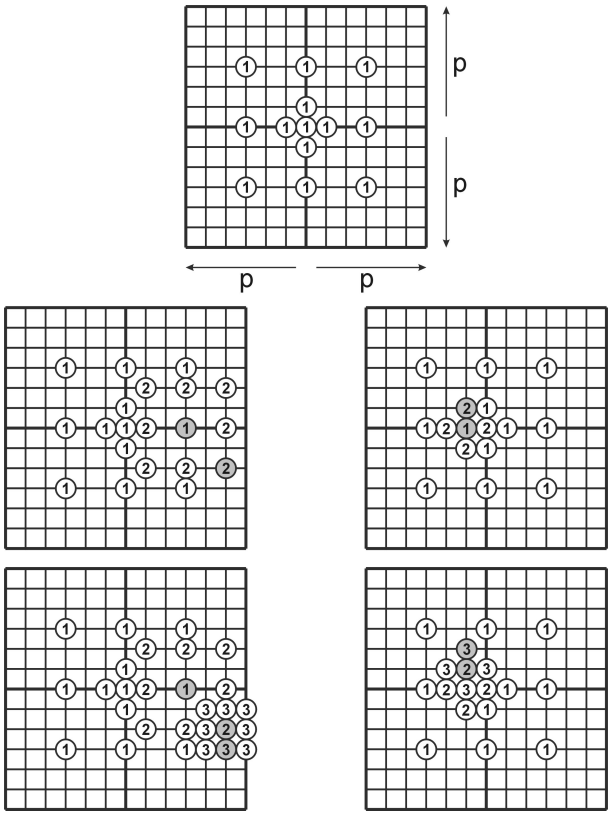
Rys. 7.9. Uproszczony schemat blokowy zaimplementowanego algorytmu E3SS estymacji ruchu

7.3. Modyfikacje algorytmu E3SS

Algorytm E3SS nie posiada zdefiniowanej liczby etapów, jakie wystarczają do wyznaczenia lokacji będącej najlepszym dopasowaniem z punktu widzenia zastosowanej funkcji celu. Algorytm efektywnego trójkrokowego przeszukiwania może zakończyć swoje działanie już w pierwszym etapie, jak również, koniec działania algorytmu może nastąpić dopiero po kilku lub nawet kilkunastu etapach. W przypadku sprzętowej implementacji kilku jednostek realizujących równolegle dany algorytm, znaczące są różnice między czasem realizacji przetwarzania danych przez poszczególne jednostki.

Jedną z możliwych modyfikacji algorytmu E3SS (Rys. 7.10), zmniejszającą czas potrzebny do przetworzenia danych przychodzących, jest ograniczenie liczby etapów związanych z lokacjami wokół środka obszaru przeszukiwania lub lokacji uznanej w poprzednim etapie jako najlepsze dopasowanie. Modyfikacja ta nosi nazwę „E3SS mod.1”. Pierwszy etap działania tak zmodyfikowanego algorytmu przebiega w identyczny sposób jak w algorytmie efektywnego trójkrokowego przeszukiwania. Część algorytmu, odpowiadająca za wyznaczanie 8 zewnętrznych lokacji na takich samych zasadach, jak w przypadku algorytmu trójkrokowego przeszukiwania, pozostała niezmieniona. Jeżeli po pierwszym etapie najkorzystniejszą funkcją charakteryzuje się jedna z 4 lokacji wyznaczonych wokół środka obszaru przeszu-

kiwania, to postępowanie jest takie samo jak dla E3SS. Drugi etap tej części algorytmu pozostał bez zmian. Modyfikacja dotyczy natomiast trzeciego etapu związanego z lokacjami wewnętrznymi, który jest jednocześnie ostatnim etapem działania tej części zmodyfikowanego algorytmu.

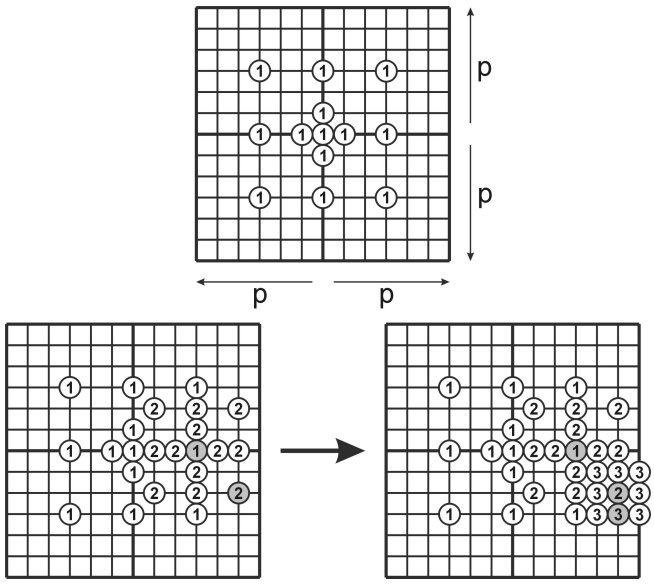


Rys. 7.10. Modyfikacja algorytmu E3SS z ograniczeniem do 3 etapów (E3SS mod.1)

Jeżeli działanie algorytmu nie zostało przerwane w poprzednich etapach to wyznaczane są 4 nowe lokacje wokół wyznaczonego chwilowego najlepszego dopasowania. Następnie obliczane są funkcje celu dla wyznaczonych lokacji i są one porównywane z funkcją celu chwilowego najlepszego dopasowania. Następnie wybierana jest lokacja z najkorzystniejszą funkcją celu i staje się ona ostatecznym najlepszym dopasowaniem. W przeciwieństwie do pełnego E3SS, algorytm kończy swoje działanie, nawet jeżeli lokacja z najkorzystniejszą funkcją celu nie jest jednocześnie lokacją uznaną w poprzednim etapie za chwilowe najlepsze dopasowanie.

Implementacja algorytmu „E3SS mod.1" w układzie XC2VP100-6FF1704 firmy Xilinx zajmuje 9 bloków pamięci BRAM oraz 627 bloków SLICE. Na taką zajętość układu składa się 415 wykorzystanych przerzutników Flip-Flop oraz 1209 bloków LUT (1178

wykorzystywane jest przez logikę a 31 przez routing). Maksymalna osiągnięta częstotliwość pracy dla tej implementacji wynosi 144,927 MHz.



Rys. 7.11. Modyfikacja algorytmu E3SS z ograniczeniem do 3 etapów oraz wykorzystaniem wszystkich jednostek obliczających funkcje celu (E3SS mod.2)

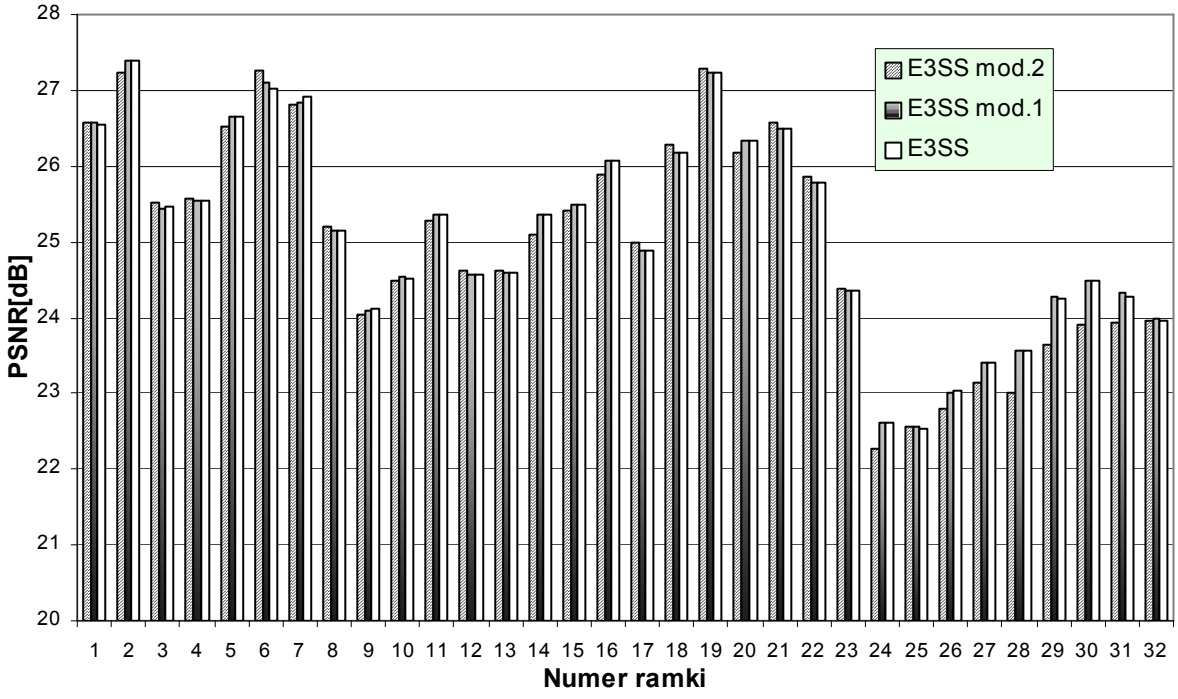
Działanie algorytmu efektywnego trójkrokowego przeszukiwania E3SS, oprócz ograniczenia liczby etapów, można również zmodyfikować pod względem wykorzystywania dostępnych zasobów (Rys. 7.11). Tak zmodyfikowany algorytm nosi nazwę „E3SS mod.2". Z punktu widzenia szybkości odpowiedzi na nadchodzące dane zaimplementowanego sprzętowo algorytmu, pożądane jest zrównoleglenie obliczeń. Dla algorytmu E3SS możliwe jest zrównoleglenie obliczania funkcji celu dla lokacji wyznaczanych w poszczególnych etapach działania algorytmu. W pierwszym etapie E3SS obliczanych jest równolegle 13 funkcji celu dla 13 różnych lokacji (bez konieczności wprowadzania etapów pośrednich odpowiadających za obliczenie części funkcji celu). W kolejnych etapach spośród 13 jednostek liczących funkcje celu wykorzystuje się 8 lub 4 jednostki, w zależności od decyzji podjętych w pierwszym etapie działania algorytmu.

Proponowana modyfikacja polega na rezygnacji z podjęcia decyzji, które lokacje będą wyznaczane w kolejnym etapie: zewnętrzne czy wewnętrzne. Prowadzi to do uproszczenia logiki sterującej działaniem algorytmu. W pierwszym etapie działania algorytmu wyznaczana jest lokacja z najkorzystniejszą funkcją celu. Jeżeli daną lokacją jest środek obszaru przeszukiwania to algorytm kończy działanie. Jeżeli najkorzystniejszą funkcją celu charakteryzuje się jedna z pozostałych lokacji to zostaje ona chwilowym najlepszym

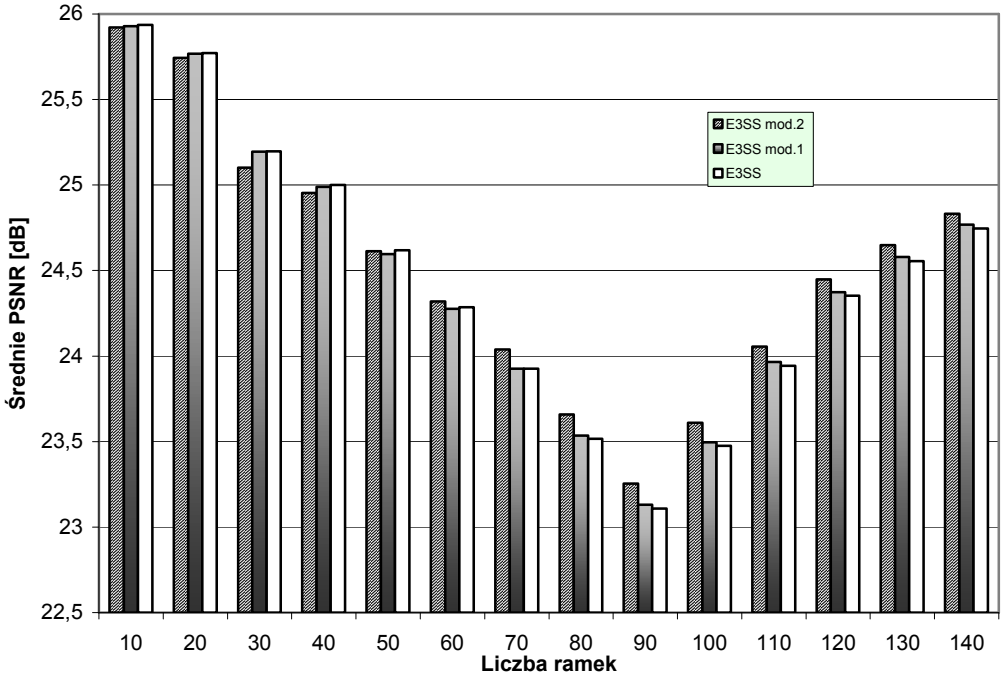
dopasowaniem i jednocześnie „środkiem" dla kolejnego etapu, w którym wyznaczane jest 8 lokacji zewnętrznych z krokiem *p*/4 oraz 4 lokacje wewnętrzne z krokiem równym 1. Podobnie jak w pierwszym etapie, wyznaczana jest lokacja z najkorzystniejszą funkcją celu z punktu widzenia zastosowanego kryterium. Jeżeli lokacją tą jest chwilowe najlepsze dopasowanie etapu poprzedniego to algorytm kończy działanie, w przeciwnym razie algorytm przechodzi do następnego etapu, w którym wyznaczane jest 8 lokacji zewnętrznych z krokiem *p*/8 oraz tak jak poprzednio 4 lokacje wewnętrzne z krokiem równym 1. Lokacja z najkorzystniejszą funkcją celu zostaje ostatecznym najlepszym dopasowaniem dla makrobloku z obrazu obecnie przetwarzanego.

Zaimplementowany, w układzie XC2VP100-6FF1704 firmy Xilinx, algorytm „E3SS mod.2" zajmuje 611 bloków SLICE (1%). Na taką zajętość składa się 415 przerzutników Flip-Flop oraz 1130 bloków LUT, z których 1108 wykorzystywanych jest przez logikę, a 22 przez routing. Ponadto wykorzystane jest 9 bloków pamięci BRAM. Maksymalna osiągnięta częstotliwość pracy dla tej implementacji wynosi 149,25 MHz.

Zmodyfikowane algorytmy zostały przetestowanie przy użyciu podstawowej sekwencji testowej „*tennis*" [3]. Uzyskane parametry zmodyfikowanych algorytmów w porównaniu z oryginalnym algorytmem efektywnego trójkrokowego przeszukiwania przedstawiają wykresy na rysunkach 7.12 i 7.13.



Rys. 7.12. Wykres zależności PSNR dla poszczególnych ramek zmodyfikowanych algorytmów E3SS (sekwencja testowa „*tennis*")



Rys. 7.13. Wykres zależności średniego szczytowego stosunku sygnału do szumu w zależności od liczby ramek zawartych w sekwencji testowej „*tennis*”

Rys. 7.14. Obrazy 63 i 64 sekwencji testowej „*tennis*”

Rys. 7.15. Rekonstrukcja obrazu 64 sekwencji „*tennis*” przy użyciu algorytmu E3SS

Rys. 7.16. Rekonstrukcja obrazu 64 sekwencji „*tennis*" przy użyciu algorytmu „E3SS mod.1"

Rys. 7.17. Rekonstrukcja obrazu 64 sekwencji „*tennis*" przy użyciu algorytmu „E3SS mod.2"

Zaletą zaproponowanej modyfikacji „E3SS mod.2" jest taki sam maksymalny czas wymagany do przetworzenia danych wejściowych jak w przypadku poprzedniej modyfikacji. Dodatkowym atutem tego algorytmu jest prostsza sprzętowa implementacja logiki sterującej działaniem algorytmu oraz wykorzystanie wszystkich dostępnych jednostek odpowiadających za obliczanie funkcji celu dla poszczególnych lokacji. Jednak algorytm „E3SS mod.2" w porównaniu z E3SS uzyskuje korzystniejszy szczytowy stosunek sygnału do szumu w przypadku sekwencji cechujących się ruchem na małym obszarze. Dla sekwencji o większym ruchu lepszymi algorytmami są E3SS oraz „E3SS mod.1". Wyniki otrzymane dla algorytmu E3SS oraz algorytmu „E3SS mod.1" są zbliżone. Zatem z punktu widzenia implementacji sprzętowej, lepszym algorytmem jest zmodyfikowany algorytm E3SS z ograniczoną liczbą etapów wymaganych do wyznaczenia najlepszego dopasowania. Maksymalny czas potrzebny do przetworzenia danych jest równy czasowi 3 etapów działania zmodyfikowanego algorytmu.

Algorytm z ograniczeniem etapów przetwarzania cechuje się zbliżoną wartością szczytowego stosunku sygnału do szumu PSNR do wartości PSNR oryginalnego algorytmu E3SS. Algorytm „E3SS mod.2" lepiej radzi sobie z wyznaczaniem najlepszego dopasowania w przypadku scen z małym ruchem, natomiast gorsze wartości PSNR otrzymywane są dla scen z większym ruchem. Jednak średni szczytowy stosunek sygnału do szumu dla tak zmodyfikowanego algorytmu zrównuje się ze średnim PSNR algorytmu E3SS przy sekwencjach zawierających większą liczbę ramek.

7.4. Podsumowanie

Estymacja ruchu wchodząca w skład toru kompresji sekwencji obrazów ruchomych, wykorzystującego standard MPEG-2, jest operacją najbardziej złożoną pod względem obliczeniowym, a zatem jest najbardziej czasochłonną. Najkorzystniejsze jest zatem jak największe zrównoleglenie tych obliczeń. Rozwiązaniem problemu ze zrównolegleniem obliczeń jest zastosowanie wewnętrznej pamięci BRAM układu programowalnego FPGA oraz zwielokrotnienie bloku odpowiedzialnego za obliczanie funkcji celu.

Z punktu widzenia dokładności otrzymywanych wyników najlepszym algorytmem jest algorytm pełnego przeszukiwania FS, jednak wymaga on wyznaczenia funkcji celu dla każdej lokacji w obszarze przeszukiwania. W przypadku zastosowania pamięci BRAM oznaczałoby to wykorzystanie bloków pamięci w liczbie przekraczającej możliwości istniejących układów programowalnych. W związku z tym zastosowane zostały algorytmy o mniejszej złożoności obliczeniowej. Algorytmem, który jest rekomendowany do wykorzystania przy sprzętowej implementacji kodera standardu MPEG-2, jest algorytm trójkrokowego przeszukiwania TSS. Na bazie tego algorytmu powstał algorytm efektywnego trójkrokowego przeszukiwania E3SS. Algorytm ten jest bardziej dokładny od swojego pierwowzoru. Pomimo większej zajętości zasobów układu programowalnego FPGA niż algorytm TSS, implementacja algorytmu efektywnego trójkrokowego przeszukiwania E3SS jest lepszym rozwiązaniem pod względem dokładności otrzymywanych wyników oraz zdolności przetwarzania danych wejściowych. Jednak wadą tego algorytmu jest duża liczba możliwych etapów. W związku z tym zaproponowane zostały modyfikacje polegające na redukcji liczby etapów działania algorytmu E3SS oraz uproszczenia logiki sterującej przy jednoczesnym wykorzystaniu wszystkich bloków wyznaczających funkcje celu. Dzięki tym modyfikacjom udało się osiągnąć mniejszą zajętość zasobów układu programowalnego FPGA oraz możliwość przetwarzania większej liczby da-

nych wejściowych niż miało to miejsce w przypadku zaimplementowanych standardowych algorytmów TSS oraz E3SS.

W związku z tym, że zaimplementowane algorytmy estymacji ruchu nie narzucają z jakich funkcji celu należy korzystać przy wyznaczaniu najlepszego dopasowania, powstaje problem, która funkcja celu jest najodpowiedniejsza przy sprzętowej realizacji wybranego algorytmu. W związku z powyższym, do celów sprzętowej realizacji została wybrana funkcja klasyfikacji różnic pomiędzy pikselami PDC. Funkcja ta jest kompromisem pomiędzy złożonością sprzętowej implementacji a dokładnością otrzymywanych wyników.

.

8. Implementacja algorytmów bezstratnej kompresji sekwencji obrazów wizyjnych

8.1. Kodowanie entropijne w standardzie MPEG-2

W standardzie MPEG-2 pierwszym etapem części bezstratnego kodowania jest proces porządkowania współczynników otrzymanych w wyniku dyskretnej transformacji kosinusowej DCT. Jedną z podstawowych metod uporządkowania jest metoda Zig-Zag przedstawiona na rysunku 3.10. Obecna w standardzie JPEG, podobnie jak w MPEG-1, metoda Zig-Zag nie jest metodą optymalną, zatem w standardzie MPEG-2 zdefiniowano tak zwaną alternatywną metodę uporządkowania (Rys. 3.11), chociaż jest stosowana również metoda Zig-Zag. W wyniku uporządkowania współczynników otrzymuje się ciąg 64-elementowy. W ciągu tym pomiędzy współczynnikami niezerowymi występują współczynniki o wartościach zerowych.

Drugim etapem części bezstratnego kodowania w MPEG-2 jest kodowanie długości ciągu RLC (ang. *Run Length Coding*). W ogólnym przypadku RLC wyznacza parę liczb. Pierwsza z nich określa ile razy pojawił się dany symbol, natomiast druga z liczb określa, jaki to symbol. Przykładowo AAAAAAAAAAAAAABBBBB $\Rightarrow$ (15,A), (5,B)

W MPEG-2 kodowanie RLC polega na wyznaczeniu pary liczb (*run*, *level*). Pierwsza liczba określa ile współczynników o wartościach równych zero poprzedza współczynnik o wartości różnej od zera, którego amplitudę określa parametr *level*.

Trzecim elementem bezstratnego kodowania w MPEG-2 jest kodowanie ze zmienną długością słowa VLC, oparte na kodowaniu Huffmana.

W standardzie ISO/IEC 13818 zostały ściśle określone trzy tablice – B14, B15 oraz B16 (tabele 8.1, 8.2, 8.4, 8.5), w których zawarte są słowa kodowe dla par *run-level* otrzymanych w wyniku kodowania RLC.

To, z której tabeli będzie korzystał koder/dekoder VLC (ang. *Variable Length Coding*), będzie zależało od kombinacji dwóch sygnałów sterujących: *intra_vlc_format* oraz

macroblock_intra. Zależności pomiędzy tymi dwoma sygnałami a wyborem odpowiedniej tablicy zawarte są w tabeli 8.3. Jednak nie wszystkie możliwe kombinacje są określone przy w tablicach B-14 i B-15. Jeśli dana para (*run, level*) nie występuje w tablicy zdeterminowanej przez *intra_vlc_format* i *macroblock_intra*, to sekwencja wyjściowa VLC składa się z 6-bitowej „ucieczki", oznaczanej w tabelach B-14 i B-15 jako Escape (*), 6-bitowego kodu reprezentującego *run* oraz 12-bitowego kodu określającego poziom. Kody dla nieokreślonych par (*run*, *level*) zawiera tablica B-16.

Tab. 8.1. Zależności pomiędzy parami run-level w zależności od tablicy B14 lub B15

Kod zmiennej długości		run	level
B14	B15		
10	0110	EOB	
1s	10s	0	1
11s		0	1
011s	010s	1	1
0100 s	110s	0	2
0101 s	0010 1s	2	1
0010 1s	0111 s	0	3
0011 0s	0001 10s	4	1
0001 10s	0011 0s	1	2
0001 01s	0000 110s	6	1
0001 00s	0000 100s	7	1
0000 110s	1110 0s	0	4
0000 100s	0000 111s	2	2
0000 111s	0000 101s	8	1
0000 101s	1111 000s	9	1
0010 0110 s	1110 1s	0	5
0010 0001 s	0001 01s	0	6
0010 0101 s	1111 001s	1	3
0010 0100 s	0010 0110 s	3	2
0010 0111 s	1111 010s	10	1
0010 0011 s	0010 0001 s	11	1
0010 0010 s	0010 0101 s	12	1
0010 0000 s	0010 0100 s	13	1
0000 0010 10s	0001 00s	0	7
0000 0011 00s	0010 0111 s	1	4

0000 0010 11s	1111 1100 s	2	3
0000 0011 11s	1111 1101 s	4	2
0000 0010 01s	0000 0010 0s	5	2
0000 0011 10s	0000 0010 1s	14	1
0000 0011 01s	0000 0011 1s	15	1
0000 0010 00s	0000 0011 01s	16	1
0000 0001 1101 s	1111 011s	0	8
0000 0001 1000 s	1111 100s	0	9
0000 0001 0011 s	0010 0011 s	0	10
0000 0001 0000 s	0010 0010 s	0	11
0000 0001 1011 s	0010 0000 s	1	5
0000 0001 0100 s	0000 0011 00s	2	4
0000 0000 1101 0s	1111 1010 s	0	12
0000 0000 1100 1s	1111 1011 s	0	13
0000 0000 1100 0s	1111 1110 s	0	14
0000 0000 1011 1s	1111 1111 s	0	15

Tab. 8.2. Pary run-level o takich samych kodach w tablicach B14 + B15

Kod zmiennej długości	run	level
0011 1s	3	1
0001 11s	5	1
0000 01	Escape* (s. 99)	
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1011 0s	1	6
0000 0000 1010 1s	1	7
0000 0000 1010 0s	2	5
0000 0000 1001 1s	3	4
0000 0000 1001 0s	5	3

0000 0000 1000 1s	9	2
0000 0000 1000 0s	10	2
0000 0000 1111 1s	22	1
0000 0000 1111 0s	23	1
0000 0000 1110 1s	24	1
0000 0000 1110 0s	25	1
0000 0000 1101 1s	26	1
0000 0000 0111 11s	0	16
0000 0000 0111 10s	0	17
0000 0000 0111 01s	0	18
0000 0000 0111 00s	0	19
0000 0000 0110 11s	0	20
0000 0000 0110 10s	0	21
0000 0000 0110 01s	0	22
0000 0000 0110 00s	0	23
0000 0000 0101 11s	0	24
0000 0000 0101 10s	0	25
0000 0000 0101 01s	0	26
0000 0000 0101 00s	0	27
0000 0000 0100 11s	0	28
0000 0000 0100 10s	0	29
0000 0000 0100 01s	0	30
0000 0000 0100 00s	0	31
0000 0000 0011 000s	0	32
0000 0000 0010 111s	0	33
0000 0000 0010 110s	0	34
0000 0000 0010 101s	0	35
0000 0000 0010 100s	0	36
0000 0000 0010 011s	0	37
0000 0000 0010 010s	0	38
0000 0000 0010 001s	0	39
0000 0000 0010 000s	0	40
0000 0000 0011 111s	1	8
0000 0000 0011 110s	1	9
0000 0000 0011 101s	1	10
0000 0000 0011 100s	1	11
0000 0000 0011 011s	1	12
0000 0000 0011 010s	1	13
0000 0000 0011 001s	1	14

0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1

Tab. 8.3. Zależności między sygnałami sterującymi a wyborem tablicy VLC

	intra_vlc_format	
	0	1
Bloki intra *macroblock_intra* = 1	B-14	B-15
Bloki inter *macroblock_intra* = 0	B-14	B-14

Tab. 8.4. Tablica B-16 – część dotycząca run

Kod	**run**
0000 00	0
0000 01	1
0000 10	2
...	...
1111 10	62
1111 11	63

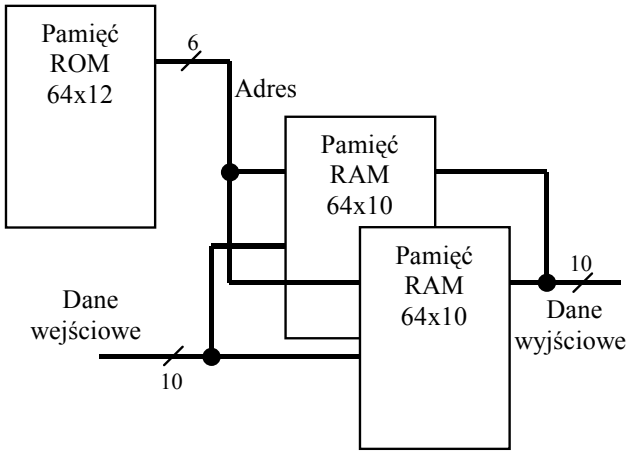
Tab. 8.5. Tablica B-16 – część dotycząca level

Kod	Level
1000 0000 0001	-2047
1000 0000 0010	-2046
...	...
1111 1111 1111	-1
0000 0000 0000	Zabronione
0000 0000 0001	+1
...	...
0111 1111 1110	+2046
0111 1111 1111	+2047

8.2. Implementacje algorytmów bezstratnej kompresji

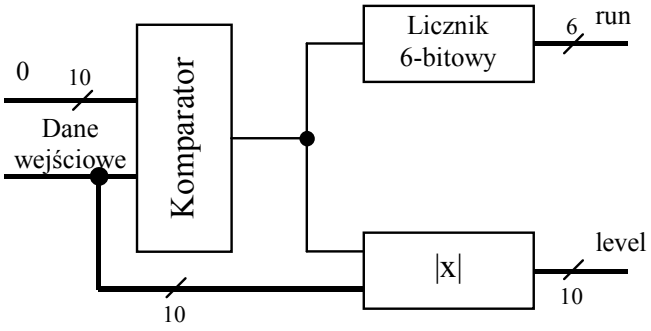
Pierwszym zaimplementowanym elementem kompresji bezstratnej standardu MPEG-2 jest układ porządkujący współczynniki otrzymane z dyskretnej transformacji kosinusowej. Schemat blokowy układu przedstawia rysunek 8.1. Układ składa się z 3 bloków pamięci. Pierwszy z bloków pamięci ROM o wielkości 64×12 bitów, zawiera 2 tablice, z których jedna odpowiada za uporządkowanie zgodne z metodą Zig-Zag, a druga jest zgodna z metodą alternatywnego uporządkowania. Tak zaimplementowana pamięć ROM generuje adresy dla cykli odczytu 2 bloków pamięci RAM 64x10 bitów. Pamięci te działają naprzemiennie. Jeżeli w danym momencie jeden z bloków jest w stanie odczytu to do drugiego z bloków zapisywane są sekwencyjnie współczynniki otrzymane z dwuwymiarowej dyskretnej transformacji kosinusowej. Takie działanie umożliwia ciągły odczyt danych, co byłoby niemożliwe w przypadku zastosowania pojedynczego bloku pamięci RAM. Przy takim rozwiązaniu odczyt byłby przerywany 1 raz na 64 takty zegara w celu zapisu nowych 64 współczynników, co oznaczałoby dwukrotnie mniejszą wydajność przetwarzania. Dane są wpisywane do pamięci pod kolejne adresy, natomiast kolejność odczytu jest uwarunkowana sekwencją adresów generowanych przez pamięć ROM. Implementacja takiego bloku porządkowania współczynników 2D-DCT w układzie XC2VP100-6FF1704 firmy Xilinx wymaga 23 bloków SLICE (z 44096 możliwych). Na taką zajętość składa się 26 przerzutników Flip-Flop (z 88192) oraz 40 bloków LUT (z 88192). Ponadto wykorzystane

zostają 3 bloki BRAM. Zaimplementowany układ może pracować z maksymalną częstotliwością wynoszącą 413,394 MHz.



Rys. 8.1. Schemat bloku uporządkowania współczynników 2D-DCT

Drugim zaimplementowanym blokiem toru kompresji bezstratnej standardu MPEG-2 jest koder RLC. Schemat blokowy kodera RLC przedstawia rysunek 8.2.



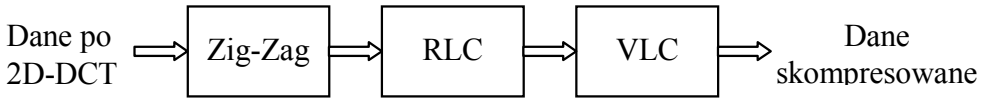
Rys. 8.2. Schemat blokowy kodera RLC

Układ kodera RLC składa się z trzech bloków. Pierwszy z nich odpowiada za wykrywanie czy dana wejściowa ma wartość zerową. Jeżeli na wejściu pojawi się współczynnik o zerowej wartości zostaje uruchomiony licznik liczący w zakresie od 0 do 63. Każdy pojawiający się kolejno współczynnik zerowy powoduje zwiększenie stanu licznika o 1. Jeśli jednak komparator wykryje współczynnik niezerowy to uruchamiany jest trzeci blok. Wyznacza on moduł wartości przetwarzanego współczynnika. Jako najstarszy bit danej wyjściowej przepisywany jest najstarszy bit danej wejściowej. Nowe dane wyjściowe w zaimplementowanym koderze pojawiają się dopiero po wykryciu współczynnika o niezerowej wartości. Implementacja kodera RLC w układzie XC2VP100 zajmuje 14 bloków

SLICE, na co składa się 12 przerzutników typu Flip-Flop oraz 24 bloki LUT. Koder może pracować z maksymalną częstotliwością 437,828 MHz.

Kolejnym zaimplementowanym blokiem toru kompresji bezstratnej jest koder o zmiennej długości słowa VLC. Układ kodera wykorzystującego określone przez specyfikację ISO/IEC 13818 tablice zajmuje 596 bloków SLICE, w tym 28 przerzutników Flip-Flop oraz 1076 bloków LUT. Koder zaimplementowany w układzie XC2VP100-6FF1704 firmy Xilinx może pracować z częstotliwością 193,949 MHz.

Schemat końcowy zaimplementowanego kodera entropijnego standardu MPEG-2 przedstawia rysunek 8.3.



Rys. 8.3. Uproszczony schemat blokowy kodera entropijnego standardu MPEG-2

Daną wejściową kodera entropijnego jest pojedynczy współczynnik dwuwymiarowej dyskretnej transformacji kosinusowej. Daną wyjściową jest kod o maksymalnej wielkości 24 bitów dla pary wyznaczonych liczb (*run*, *level*).

Tab. 8.6. Parametry implementacji implementacji algorytmów bezstratnej kompresji w układzie XC2VP100-6FF1704 firmy Xilinx

	Zig-Zag	**RLC**	**VLC**	**Koder entropijny**
SLICE	23	14	596	644
FF	26	12	29	162
LUT	40	24	1076	1235
BRAM	3	-	-	3
F(MHz)	413,394	437,828	193,949	139,005

Koder, zaimplementowany podobnie jak bloki składowe w układzie XC2VP100-6FF1704, zajmuje 717 bloków SLICE, co stanowi 1% zasobów układu. Na takie wykorzystanie składa się 274 przerzutników Flip-Flop oraz 1366 bloków LUT (1% zasobów). Dodatkowo wykorzystane są 3 bloki BRAM. Dane na wyjściu kodera mogą pojawiać się co każdy takt zegara, jeśli nie zostaną wykryte współczynniki o wartości zerowej i wtedy dana na wyjściu

pojawi się wraz z przyjściem współczynnika o niezerowej wartości. Koder może pracować z maksymalną częstotliwością 139,005 MHz.

Oprócz toru kodera entropijnego, został zaimplementowany dekoder. Implementacja dekodera wykorzystuje 193 bloków SLICE, na co składa się 67 przerzutników Flip-Flop oraz 343 bloków LUT. Dodatkowo w układzie XC2VP100-6FF1704 wykorzystywane są 4 bloki pamięci BRAM. Implementacja pojedynczego bloku dekodera długości ciągu RLD wymaga 35 bloków SLICE. Na taką liczbę bloków składa się 56 przerzutników Flip-Flop oraz 30 bloków LUT. Częstotliwość pracy dekodera ciągów wyniosła 233,236 MHz. W przypadku implementacji dekodera o zmiennej długości słowa kodowego wymagane jest 96 bloków SLICE (62 przerzutniki Flip-Flop oraz 169 bloki LUT). Ponadto wykorzystane zostały 323 bufory trójstanowe oraz jeden blok pamięci BRAM. Zaimplementowany dekoder VLD może pracować z częstotliwością 297,442 MHz.

Tab. 8.7. Parametry implementacji algorytmów bezstratnej dekompresji w układzie XC2VP100-6FF1704 firmy Xilinx

	RLD	**VLD**	**Dekoder entropijny**
SLICE	35	96	193
FF	56	62	67
LUT	30	169	343
BRAM	-	-	4
F(MHz)	233,236	297,442	233,236

Tab. 8.8. Wydajność zaimplementowanego kodera oraz dekodera entronijnego

Rozdzielczość obrazu [w pikselach]	**Liczba ramek na sekundę**	
	koder	**dekoder**
800x600	289	492
1024x768	176	300
1000x1000	139	236
1920x1152	62	107
2000x2000	34	59

8.3. Podsumowanie

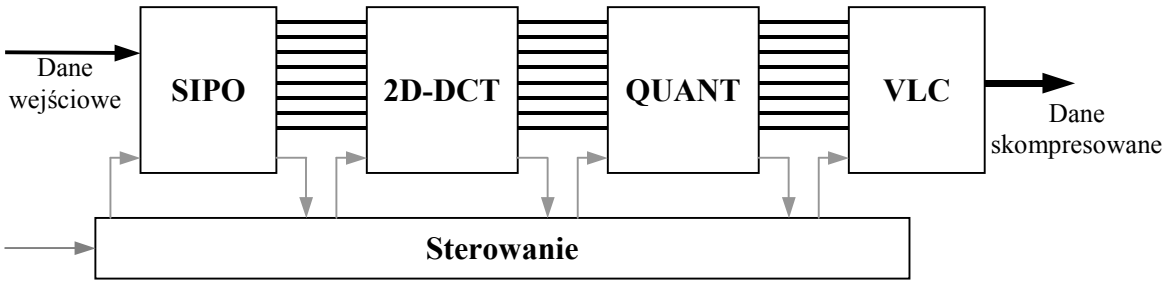
Z punktu widzenia wydajności przetwarzania systemu, w skład którego wchodzi koder entropijny, ważne jest, aby koder ten mógł przyjmować nowe dane w każdym cyklu zegara. W przypadku dekodera ważne jest, aby dane pojawiały się na jego wyjściu również w każdym cyklu zegara. Takie działanie bloków kodera i dekodera entropijnego jest wymagane w przypadku, gdy zostaną one wykorzystane w systemie wymagającym przetwarzania danych bez opóźnień związanych z wprowadzaniem lub wyprowadzaniem danych z poszczególnych bloków składowych systemu. Jednym z bloków wchodzących w skład kodera oraz dekodera jest blok uporządkowywania pikseli. Blok ten wnosi największe opóźnienia związane z przetwarzaniem danych wejściowych, jeśli zostanie zaimplementowany z pojedynczym blokiem pamięci przechowywującej wartości 64 skwantowanych współczynników otrzymanych z dwuwymiarowej dyskretnej transformacji kosinusowej. Rozwiązaniem tego problemu jest zastosowanie dwóch bloków pamięci, przechowywujących dane. Bloki te pracują naprzemiennie: jeśli jeden z bloków jest w trybie odczytu, to drugi blok jest w trybie zapisu danych.

Zaprojektowane przez autora realizacje kodera i dekodera entropijnego spełniają wymogi przetwarzania danych w czasie rzeczywistym.

9. Sprzętowa implementacja standardu MPEG-2

9.1. Kompresja MPEG-2 z wykorzystaniem obrazów typu I

Pierwszą zaimplementowaną wersją kodera standardu MPEG-2 jest koder, który w procesie kompresji wykorzystuje wyłącznie obrazy typu I. Schemat blokowy takiego kodera przedstawia rysunek 5.1.



Rys. 9.1. Schemat blokowy kodera standardu MPEG-2 wykorzystującego tylko obrazy typu I

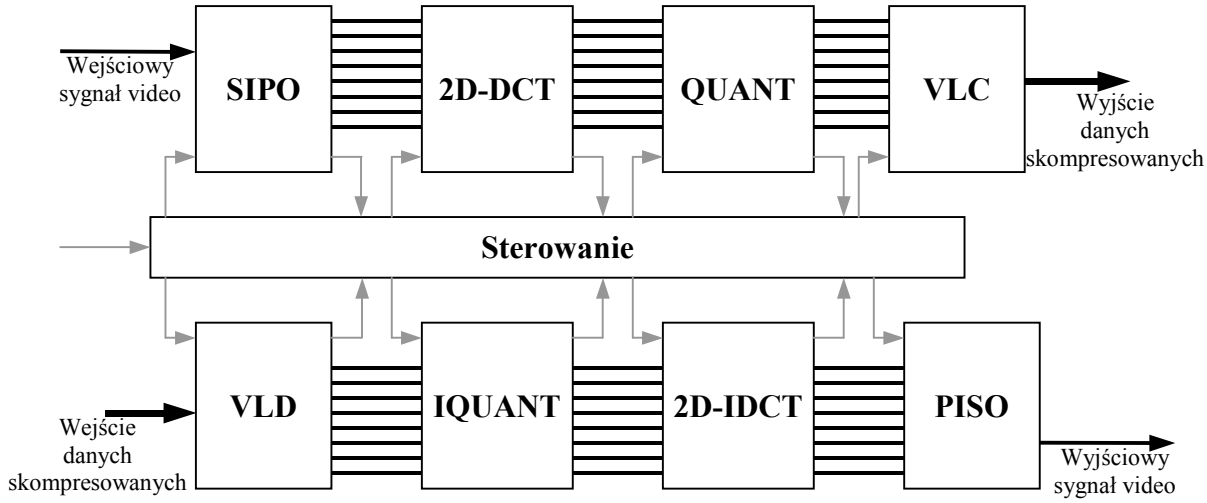
Pierwszym blokiem, przez jaki przechodzi wejściowy sygnał video, jest rejestr szeregowo-równoległy SIPO (ang. *Serial In Parallel Out*). Wyjście rejestru stanowi wektor złożony z 8 pikseli, który trafia na wejście bloku dwuwymiarowej dyskretnej transformacji kosinusowej 2D-DCT. Kolejnym blokiem w torze kompresji standardu MPEG-2 jest kwantyzacja. Blok QUANT został zaimplementowany w oparciu o dedykowane mnożarki układu Virtex-II Pro. Układ kodera entropijnego VLC, zastosowanego w koderze MPEG-2, zgodny jest ze schematem blokowym przedstawionym na rysunku 8.3.

Implementacja toru przetwarzania obrazu kodera standardu MPEG-2, wykorzystującego w procesie kompresji wyłącznie obrazy typu I, w układzie XC2VP100-6FF1704 zajmuje 4630 bloków SLICE, co stanowi 10% układu. Na taką zajętość składa się 6003 bloków LUT (5571

wykorzystanych przez logikę, 426 przez routing, 6 przez rejestry przesuwne) oraz 4470 przerzutników Flip-Flop. Ponadto wykorzystanych jest 19 bloków pamięci BRAM oraz 16 wbudowanych w układzie XC2VP100-6FF1704 mnożarek. Koder może przetwarzać 86 ramek o rozdzielczości 1920×1152 pikseli 8-bitowych w ciągu sekundy.

Implementacja kodera MPEG-2 wykorzystującego wyłącznie obrazy typu I, w układzie XC4VFX100-11FF11152 zajmuje 4678 bloków SLICE, co stanowi 11% układu. Na taką zajętość układu składa się 5901 bloków LUT (5627 wykorzystanych przez logikę, 261 przez routing, 13 przez rejestry przesuwne) oraz 4476 przerzutników Flip-Flop. Ponadto wykorzystanych jest 19 bloków pamięci BRAM oraz 16 wbudowanych w układzie XC4VFX100-11FF11152 mnożarek. Koder może przetwarzać 86 ramek o rozdzielczości 1920×1152 pikseli 8-bitowych w ciągu sekundy.

Schemat zaimplementowanego kodeka sygnału video standardu MPEG-2, który oferuje przetwarzanie tylko obrazów typu I, przedstawia rysunek 5.2.



Rys. 9.2. Schemat blokowy kodeka standardu MPEG-2 wykorzystującego tylko obrazy typu I

Implementacja toru przetwarzania obrazu kodeka standardu MPEG-2, wykorzystującego w procesie kompresji wyłącznie obrazy typu I, w układzie XC2VP100-6FF1704 zajmuje 4616 bloków SLICE (10%), na co składa się 6007 bloków LUT (5570 wykorzystanych przez logikę, 431 przez routing, 6 przez rejestry przesuwne) oraz 4472 przerzutników Flip-Flop. Ponadto wykorzystanych jest 19 bloków pamięci BRAM oraz 16 wbudowanych w układzie XC2VP100-6FF1704 mnożarek. Kodek może przetwarzać 88 ramek o rozdzielczości 1920×1152 pikseli 8-bitowych w ciągu sekundy. Oznacza to, że kodek taki może przetwarzać

w czasie rzeczywistym obraz o wysokiej rozdzielczości, jednak oferowany przez niego stopień kompresji będzie mały. Maksymalna różnica między pikselami sygnału wejściowego a ich rekonstrukcją wynosi 3, czyli taka, jaka została osiągnięta przy implementacji dwuwymiarowej dyskretnej transformacji kosinusowej. Na zwiększenie tej odchyłki nie wpływa niedokładność odwzorowania współczynników po procesie dekwantyzacji.

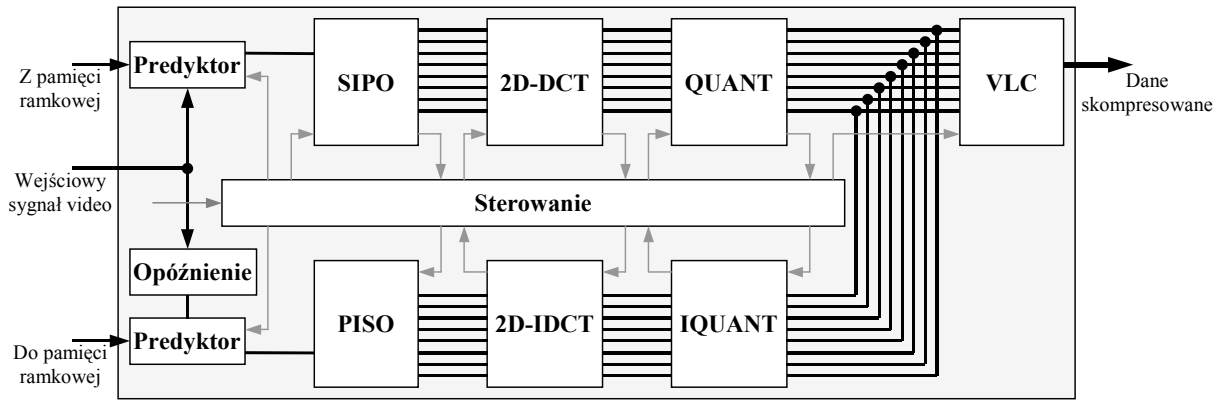
W przypadku implementacji kodeka w układzie XC4VFX100-11FF11152 wykorzystanych zostało 4672 bloki SLICE co stanowi 11% wszystkich tych bloków układu Virtex-4. Na taką zajętość zasobów składa się 4478 przerzutników Flip-Flop oraz 5897 bloków LUT. Większość LUT, bo aż 5626, zajmuje logika. Routing zajmuje 258 bloków LUT, a rejestr przesuwny 13. Ponadto kodek zajmuje 19 bloków pamięci BRAM oraz 16 bloków DSP48. Tak zaimplementowany kodek jest w stanie przetworzyć 89 ramek na sekundę obrazu o rozdzielczości 1920×1152 pikseli 8-bitowych.

9.2. Kompresja MPEG-2 z wykorzystaniem obrazów typu I i P

Stopień kompresji można zwiększyć dzięki zastosowaniu, oprócz obrazów typu I, obrazów P otrzymywanych w wyniku predykcji z obrazu poprzedniego.

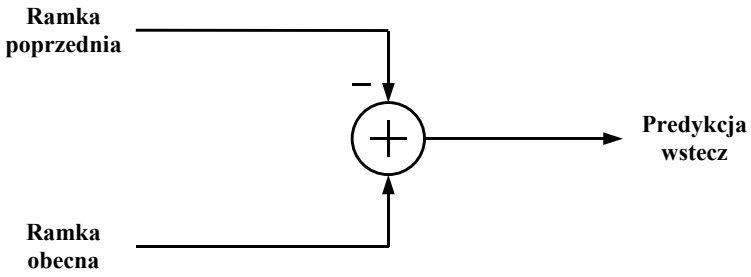
Obrazy typu P można uzyskać w dwojaki sposób. Pierwszy z nich, wymagający mniejszej złożoności obliczeniowej, jest oparty na zastosowaniu predykcji. W standardzie MPEG-2 jest to tryb z „zerowym" wektorem ruchu. Drugi sposób tworzenia obrazów typu P polega na wykorzystaniu procesu estymacji i kompensacji ruchu.

Schemat zaimplementowanego kodera wykorzystującego tryb z „zerowym" wektorem ruchu przedstawia rysunek 9.3.



Rys. 9.3. Schemat blokowy kodera standardu MPEG-2 wykorzystującego predykcję

Dodatkowym blokiem, który pojawia się w porównaniu z poprzednią wersją kodera jest predyktor (Rys. 9.4)



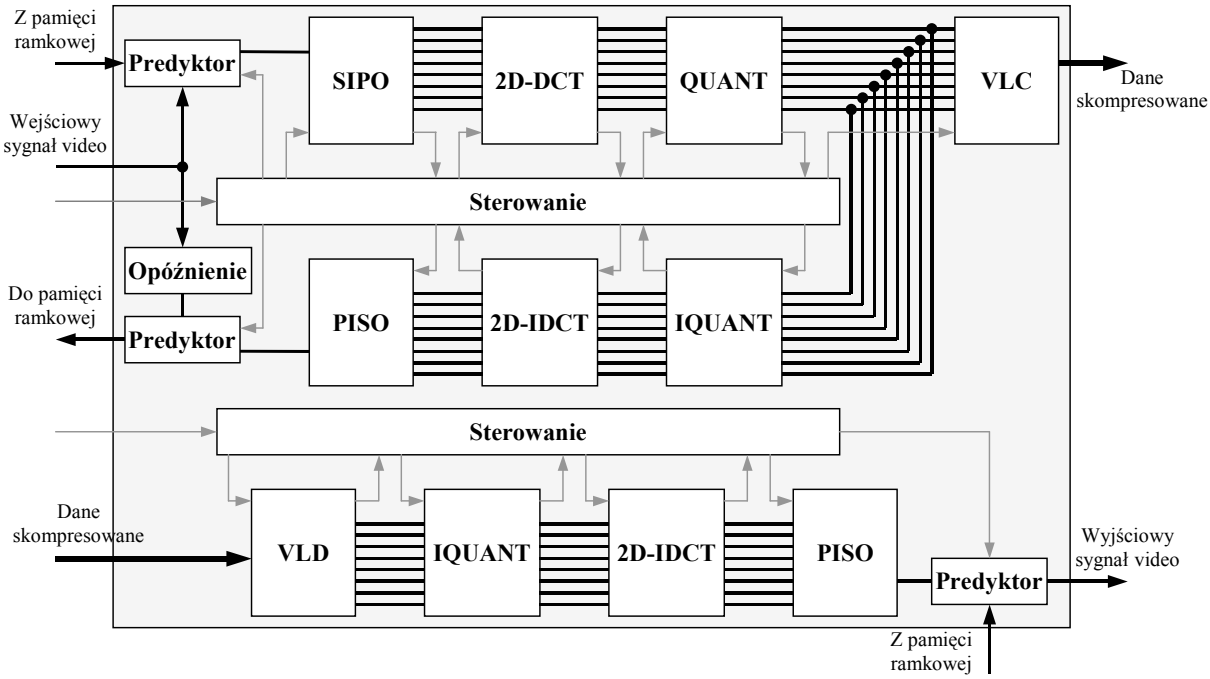
Rys. 9.4. Schemat blokowy predyktora dla obrazów typu P

Implementacja części odpowiadającej za kompresję obrazu kodera MPEG-2 wykorzystującego dodatkowo obrazy typu P otrzymane przy pomocy predyktora wykorzystuje 9370 bloków SLICE (21%) układu XC2VP100-6FF1704. Na taką zajętość składa się 12799 bloków LUT (11212 wykorzystanych przez logikę, 1579 przez routing, 8 przez rejestry przesuwne) oraz 8558 przerzutników Flip-Flop. Dodatkowo wykorzystano 35 bloków BRAM oraz 32 wbudowane mnożarki. Koder może przetworzyć 82 ramki na sekundę o rozdzielczości 1920×1152 pikseli.

Implementacja w układzie XC4VFX100-11FF11152 kodera MPEG-2 z obrazami I oraz obrazami P, tworzonymi na podstawie predykcji, wykorzystuje 9420 bloków SLICE, czyli 22% wszystkich zasobów tego układu. Na taką zajętość składa się 8564 przerzutniki Flip-Flop, 263 zatrzaski oraz 12167 bloków LUT. Wśród bloków LUT 11261 stanowi logika, 891 bloków zajmuje routing, a 15 rejestry przesuwne. Ponadto implementacja ta wymaga 35 bloków pamięci BRAM (9%) oraz 32 układów DSP48. Tak zaimplementowany koder może przetwarzać 85 ramek na sekundę o rozdzielczości 1920×1152 pikseli.

Implementacja części odpowiadającej za kompresję obrazu kodeka MPEG-2, wykorzystującego dodatkowo obrazy typu P otrzymane przy pomocy predyktora, wykorzystuje 14684 bloki SLICE (33%) układu XC2VP100-6FF1704, na co składa się 20119 bloków LUT (17358 wykorzystanych przez logikę, 2751 przez routing, 10 przez rejestry przesuwne) oraz 13670 przerzutników Flip-Flop. Dodatkowo wykorzystano 55 bloków BRAM oraz 48 wbudowanych mnożarek (10%).

Kodek może jednocześnie pracować w trybie kodera i dekodera. Podobnie jak koder również kodek może przetworzyć 82 ramki na sekundę o rozdzielczości 1920×1152 pikseli.



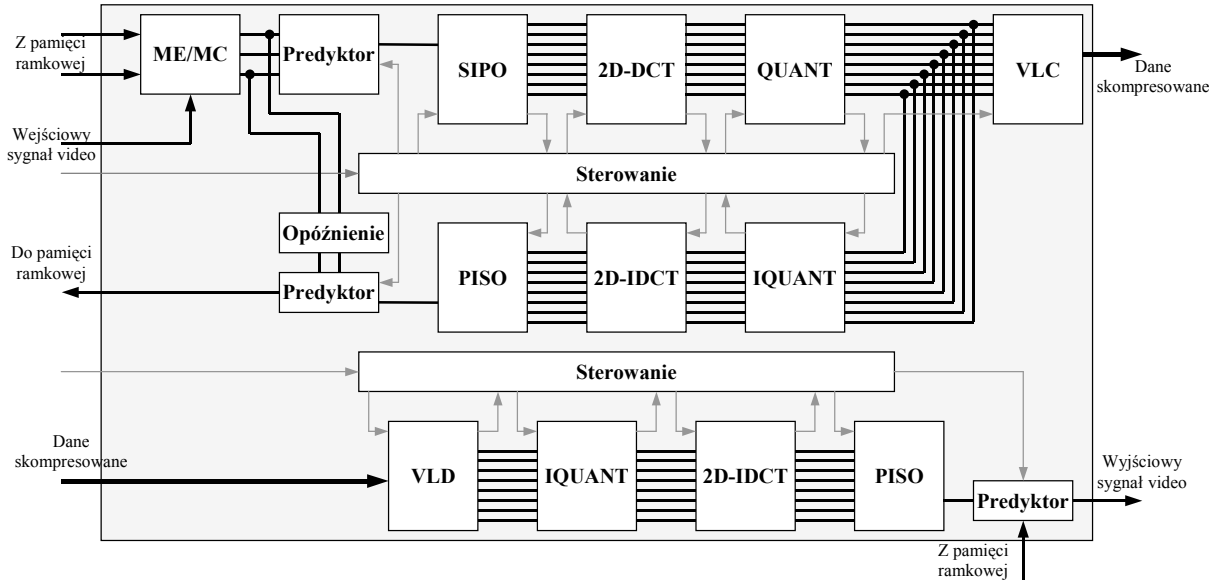
Rys. 9.5. Schemat blokowy kodeka standardu MPEG-2 wykorzystującego predykcję

W przypadku implementacji kodeka w układzie XC4VFX100-11FF11152 wykorzystanych zostaje 14705 bloków SLICE, co stanowi 34% wszystkich tych bloków układu Virtex-4FX100. Na taką zajętość zasobów składa się 13035 przerzutników Flip-Flop, 492 zatrzaski oraz 18955 bloków LUT (22%). Logika zajmuje 17404 bloki LUT, a routing 1534 bloki LUT. Rejestr przesuwny zajmuje 17 bloków LUT. Ponadto kodek używa 55 bloków pamięci BRAM (14%) oraz 48 bloków DSP48 (30%). Tak zaimplementowany kodek jest w stanie przetworzyć 80 ramek na sekundę obrazu o rozdzielczości 1920×1152 pikseli 8-bitowych.

9.3. Kompresja MPEG-2 z wykorzystaniem obrazów typu I, P i B

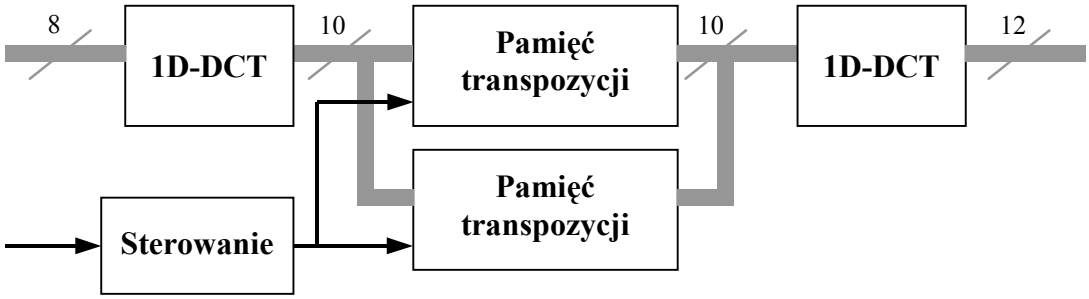
Ostateczną wersję zaimplementowanego kodeka standardu MPEG-2 przedstawia rysunek 9.6. Kodek ma możliwość różnego tworzenia obrazów typu P oraz B. Pierwszy ze sposobów jest analogiczny jak powyżej, a mianowicie obrazy P i B są tworzone na zasadzie predykcji przy obrazach P i interpolacji przy obrazach typu B, natomiast wektor ruchu jest „zerowy". Drugim sposobem tworzenia jest wykorzystanie estymacji ruchu. Wektory ruchu wyznaczone przez zaimplementowane bloki estymacji zawierają się w zakresie [±15, ±15].

Zaletą zaimplementowanego kodeka MPEG-2 jest możliwość przetwarzania obrazów o dowolnej rozdzielczości. Enkoder dopasowuje się do rozdzielczości w zależności od sygnałów: *picture_coding_type*, *EOB* oraz *start_linii*.



Rys. 9.6. Schemat blokowy kodeka standardu PMEG-2 wykorzystującego obrazy I, P oraz B

Blok 2D-DCT, który mógłby przetwarzać przychodzące dane w sposób ciągły, można zrealizować w dwojaki sposób. Pierwszym z nich jest zastosowanie dwóch bloków zgodnych ze schematem przedstawionym na rysunku 5.1. Bloki te przyjmowałyby naprzemiennie po 8 wektorów. Rozwiązanie takie składałoby się z 4 bloków jednowymiarowej DCT oraz z 2 bloków pamięci transpozycji. Innym sposobem, który zużywa mniej zasobów logicznych układu FPGA, jest rozwiązanie przedstawione na rysunku 9.7.

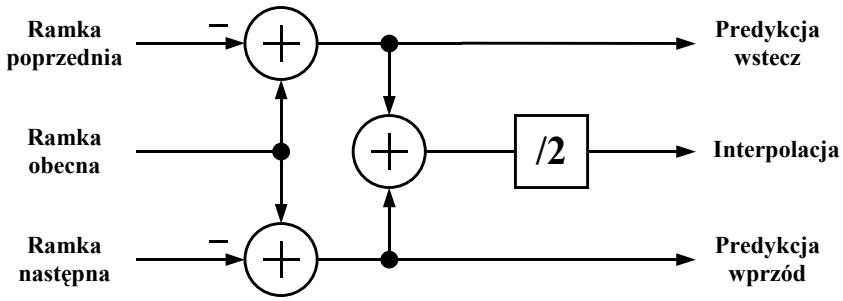


Rys. 9.7. Schemat blokowy modułu dwuwymiarowej dyskretnej transformacji kosinusowej

Na wyjściu układu 1D-DCT dane mogą pojawiać się co cykl zegara. Jednak pojedynczy blok zaimplementowanej pamięci transpozycji wymaga do pełnego przetworzenia bloku 8×8

pikseli 16 cykli zegara, w tym 8 cykli na zapis i 8 cykli na odczyt. Zastosowanie równoległe dwóch bloków pamięci transpozycji działających naprzemiennie umożliwia ciągłe zapisywanie danych z pierwszego bloku 1D-DCT do pamięci oraz dostarczanie danych do drugiego bloku 1D-DCT co każdy cykl zegara. Takie rozwiązanie 2D-DCT umożliwia zredukowanie potrzebnych zasobów sprzętowych, co oznacza również redukcję liczby używanych mnożarek sprzętowych układu XC4VFX100-11FF11152. Układ ten dysponuje 160 blokami DSP48, zawierającymi sprzętowe mnożarki 18×18 (układ V2P100 zawiera 444 mnożarki 18×18). Ze względu na małą liczbę dedykowanych mnożarek w układzie Virtex-4 uzasadnione jest zatem stosowanie mnożarek ze stałym współczynnikiem mnożenia, implementowanych za pomocą bloków SLICE układu FPGA.

Zastosowany predyktor, w kodeku wykorzystującym wszystkie typy obrazów, przedstawia rysunek 9.8.



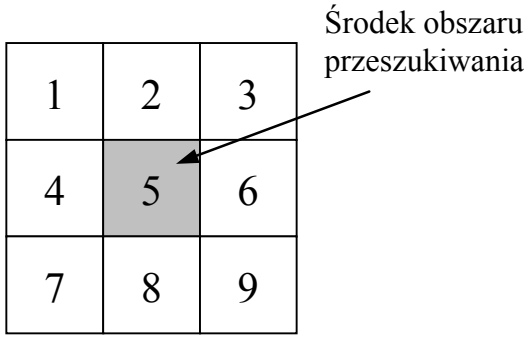
Rys. 9.8. Schemat blokowy predyktora dla obrazów typu P oraz B

Sygnałem, który określa jakiego typu będzie przetwarzany obraz, jest sygnał *picture_coding_type*. Możliwe wartości tego sygnału zawiera tablica 9.1.

Tab. 9.1. Możliwe wartości sygnału *picture_coding_type*

picture_coding_type	**Rodzaj obrazu**
000	Zabronione
001	I
010	P
011	B
100	D w standardzie ISO/IEC 11172-2
101	Zarezerwowane
110	Zarezerwowane
111	Zarezerwowane

W zależności od sygnału sterującego *mc_pred* obrazy typu P oraz B mogą być zakodowane przy pomocy estymacji ruchu (*mc_pred*=’1’) lub z tak zwanym „zerowym” wektorem ruchu (*mc_pred*=’0’). W przypadku zastosowania tylko predykcji dane wchodzącą do kodera entropijnego w każdym cyklu zegara. Jeśli tworzenie obrazów będzie wykorzystywało estymację ruchu dane wchodzące do kodera entropijnego będą się pojawiały co 780 cykli zegara. Wynika to z czasu potrzebnego na wyznaczenie wektorów ruchu. W przypadku proponowanego rozwiązania, przedstawionego na rysunku 7.9, 58% czasu działania układu zajmował zapis danych do pamięci obszaru zapisywania przy zakresie wektorów ruchu [±8, ±8]. W tym przypadku została zastosowana pamięć obszaru przeszukiwania o wielkości 1024×8 bitów. Przy zwiększeniu obszaru przeszukiwania do [±15, ±15] konieczne jest zastosowanie pamięci o wielkości 2304×8 bitów. Czas potrzebny do zapisu takiej pamięci stanowiłby 75% czasu pracy estymatora. Rozwiązaniem tego problemu jest zastosowanie 5 bloków pamięci obszaru przeszukiwania oraz 5 bloków pamięci zawierającej makrobloki z ramki obecnie przetwarzanej.



Rys. 9.9. Oznaczenia makrobloków stanowiących obszar przeszukiwania

Sposób działania układów pamięci przedstawiają tabele 9.2 (dla pamięci obszaru przeszukiwania) oraz 9.3 (dla pamięci makrobloku z ramki obecnie przetwarzanej).

Tab. 9.2. Kolejności cykli zapisu i odczytu poszczególnych pamięci obszaru przeszukiwania

Pamięć	1	2	3	4	5
Cykle	Zapis 1	Odczyt 2	Odczyt 1	Zapis 3	Zapis 2
	Zapis 2	Zapis 1	Odczyt 2	Odczyt 1	Zapis 3
	Zapis 3	Zapis 2	Zapis 1	Odczyt 2	Odczyt 1
	Odczyt1	Zapis 3	Zapis 2	Zapis 1	Odczyt 2
	Odczyt 2	Odczyt 1	Zapis 3	Zapis 2	Zapis 1

Oznaczenia w tabeli 9.2 przedstawiają:

Zapis 1 – zapis makrobloków 1, 4, 7,

Zapis 2 – zapis makrobloków 2, 5, 8 (makrobloki są jednocześnie zapisywane do pamięci n+1 jako makrobloki 1, 4, 7),

Zapis 3 – zapis makrobloków 3, 6, 9 (makrobloki są jednocześnie zapisywane do pamięci n+1 jako makrobloki 2, 5, 8 i do pamięci n+2 jako makrobloki 1, 4, 7),

Odczyt 1 – odczyt danych z lokacji dostarczonej z generatora adresu,

Odczyt 2 – odczyt danych z lokacji determinowanej wyznaczonym wektorem ruchu.

Cykl Odczyt 2 odpowiada za proces kompresji ruchu w układzie kodera. Dane podczas tego cyklu są przesyłane do predyktora (256 cykli).

Tab. 9.3. Cykle odczytu i zapisu poszczególnych pamięci zawierających makrobloki obrazu obecnie przetwarzanego

Pamięć	1	2	3	4	5
Cykle	Zapis 1	-	-	Odczyt 2	Odczyt 1
	Odczyt 1	Zapis 1	-	-	Odczyt 2
	Odczyt 2	Odczyt 1	Zapis 1	-	-
	-	Odczyt 2	Odczyt 1	Zapis 1	-
	-	-	Odczyt 2	Odczyt 1	Zapis 1

Oznaczenia w tabeli 9.3 przedstawiają:

Zapis 1 – zapis do pamięci makrobloku dla którego będzie wyznaczany wektor ruchu,

Odczyt 1 – cykl, w którym dane z pamięci są dostarczane do bloku odpowiedzialnego za wyznaczanie funkcji celu,

Odczyt 2 – zapewnia dostarczenie danych na wejście predyktora bez przesunięcia czasowego względem wyznaczonego najlepszego dopasowania.

Zaimplementowany w układzie XC4VFX100-11FF11152 kodek standardu MPEG-2, wykorzystujący wszystkie typy obrazów, wykorzystuje 19977 bloków SLICE - co stanowi 47% zasobów tego układu. Na taką zajętość składa się 13245 przerzutników Flip-Flop, 932 zatrzaski oraz 27785 bloków LUT (32% z 84352), z czego 24313 bloków wykorzystuje logika, 887 bloków zajmuje routing, 2560 zajmuje rozproszona dwuportowa pamięć RAM. Dodatkowo wykorzystanych jest 178 ze 376 dostępnych bloków BRAM oraz jeden układ

DCM - zarządzający zegarem układu. Przy typowej budowie grupy obrazów IBBPBBPBBIBB przy zastosowaniu procesu estymacji ruchu, kodek jest w stanie przetwarzać obraz bez przeplotu o rozdzielczości 1920×1152 pikseli 8-bitowych pikseli i 26 ramek na sekundę. Jeśli ważnym parametrem będzie liczka ramek na sekundę kodek może pracować w trybie z „zerowym" wektorem ruchu. Można wtedy osiągnąć szybkość 67 ramek na sekundę przy rozdzielczości HDTV, jednak płacąc za to mniejszym współczynnikiem kompresji.

9.4. Podsumowanie

Pierwszym zagadnieniem, które należało rozwiązać w przypadku implementacji kodera oraz kodeka standardu MPEG-2, była taka implementacja bloków dyskretnej transformacji kosinusowej DCT oraz odwrotnej transformacji kosinusowej IDCT, aby bloki te mogły przetwarzać dane wejściowe w sposób ciągły. Rozwiązaniem było zastosowanie dwóch oddzielnych bloków 2D-DCT pracujących naprzemiennie. Jednak takie rozwiązanie wymagało dwóch bloków pamięci transpozycji oraz 4 bloków odpowiedzialnych za 1D-DCT, z których w danym momencie tylko 2 bloki były efektywnie wykorzystywane. Lepszym sposobem okazała się realizacja wykorzystująca tylko 2 bloki odpowiedzialne za 1D-DCT pracujące w sposób ciągły oraz pamięć transpozycji wykorzystującą, tak jak w poprzednim rozwiązaniu, dwa bloki pamięci rozproszonej pracujących naprzemiennie.

Kolejnym aspektem, który należało rozpatrzyć było zapewnienie ciągłości pracy bloku estymacji ruchu, gdyż pierwotnie czas przetwarzania pojedynczego makrobloku był równy sumie czasów potrzebnych do zapisania pamięci przechowującej obszar przeszukiwania oraz działania samego procesu wyznaczania najlepszego dopasowania. Rozwiązaniem tego problemu było zastosowanie 5 bloków pamięci, z których każdy odpowiedzialny jest za inny obszar przeszukiwania. W danym momencie, tylko jeden z tak zaimplementowanych bloków pamięci jest źródłem danych dla działającego bloku algorytmu estymacji ruchu.

Wszystkie działania związane ze zrównolegleniem obliczeń oraz związane z samą optymalizacją kodu VHDL miały na celu zmniejszenie zajmowanych zasobów sprzętowych układu FPGA, wykorzystywanych przez zaimplementowany koder/kodek przy jednoczesnym dużym priorytecie na szybkość przetwarzania danych wejściowych.

Istniejące rozwiązania implementacji standardu MPEG-2, opisane w rozdziale 4, pracują w trybie MP@ML co oznacza, że przetwarzają obrazy o rozdzielczości 720×480 pikseli z szybkością 30 ramek na sekundę. Wśród implementacji, które mają możliwość pracy

w trybie MP@HL, jest dekoder μPD61160 firmy NEC, w którym wyjściowy sygnał video może mieć maksymalnie 1080 linii przy obrazach z przeplotem. Oba powyższe rozwiązania oferują większe zakresy obszarów przeszukiwania. Koder stworzony przez firmę Duma Video operuje na obrazach o rozdzielczości 1920×1080 z przeplotem przy szybkości 29,97 ramek na sekundę, a implementacja ta wykorzystuje tylko obrazy typu I oraz P. Zaproponowane w tym rozdziale rozwiązanie ma zbliżone parametry, ale wykorzystuje wszystkie typy obrazów, co umożliwia osiągnięcie większego stopnia kompresji niż przy rozwiązaniu oferowanym przez firmę Duma Video. Ponadto proponowany kodek przetwarza obrazy bez przeplotu o rozdzielczości 1920×1152 pikseli. W przypadku kodera DV-X firmy Duma Video maksymalna rozdzielczość przetwarzanego obrazu bez przeplotu wynosi 1280×720 pikseli. Dodatkowym atutem opisanego w tym rozdziale rozwiązania jest to, że łączy w sobie zarówno koder jak i dekoder, które mogą pracować niezależnie.

10. Podsumowanie

Celem pracy była implementacja systemu kompresji i dekompresji sekwencji obrazów ruchomych, wykorzystującego standard MPEG-2. System ten miał przetwarzać obrazy o wysokiej rozdzielczości w czasie rzeczywistym. Jednocześnie miał on wykorzystywać obrazy typu I, P oraz B.

Pierwszym elementem, który został przystosowany do implementacji sprzętowej w układach FPGA pod kątem zajętości zasobów oraz szybkości działania algorytmu, była dyskretna transformacja kosinusowa DCT. Spośród kilku możliwych algorytmów DCT w wyniku badań jako algorytm najlepiej nadający się do sprzętowej implementacji został wytypowany algorytm Chena. Modyfikacja tego algorytmu, dokonana przez autora pracy, polegała na ograniczeniu dokładności odwzorowania współczynników mnożenia, przy czym nie wszystkie współczynniki zostały zaimplementowane z taką samą dokładnością. Współczynniki mnożenia odpowiedzialne za wyznaczanie współczynników DCT niosących ze sobą największą energię, zostały zaimplementowane z większą dokładnością. Natomiast współczynniki mnożenia odpowiedzialne za wyznaczanie pozostałych współczynników zostały odwzorowane z mniejszą dokładnością. Działanie takie powodowało powstawanie różnicy pomiędzy pikselami wejściowymi a ich rekonstrukcją otrzymaną poprzez złożenie transformacji 2D-DCT i 2D-IDCT. Jednak maksymalna otrzymana różnica jest prawie niezauważalna dla oka ludzkiego. Ponadto wykazane zostało, że maksymalna różnica pomiędzy współczynnikami otrzymanymi z zaimplementowanego zmodyfikowanego algorytmu Chena a współczynnikami wyznaczonymi z podstawowej zależności opisującej 2D-DCT jest mniejsza niż najmniejszy możliwy współczynnik macierzy kwantyzacji. Sytuacja taka skutkuje wystąpieniem możliwego błędu na najmłodszym bicie skwantowanej wartości współczynnika dotyczącego składowej stałej. Oznacza to, że po procesie dekwantyzacji otrzymany współczynnik dotyczący składowej stałej może różnić się o maksymalnie ±8. Mimo takiej różnicy pomiędzy wyjściem z 2D-DCT a wejściem 2D-IDCT zrekonstruowane piksele nie różnią się od swoich pierwowzorów o więcej niż 3.

W przypadku implementacji funkcji celu, decydującej o wyznaczaniu najlepszego dopasowania dla makrobloków, z obrazu obecnie przetwarzanego wśród makrobloków obrazów referencyjnych została uwzględniona wartość maksymalnej różnicy między pikselem a jego rekonstrukcją. Znalazło to swoje odzwierciedlenie przy wyborze funkcji celu decydującej o wyznaczaniu wektorów ruchu. Jako funkcja, która najlepiej nadaje się do sprzętowej implementacji, została wybrana funkcja klasyfikacji różnicy pikseli. Zaletą tej funkcji jest możliwość sterowania dokładnością wyznaczania najlepszego dopasowania poprzez wartość progową. W przypadku takiej implementacji 2D-DCT wartość progowa nie powinna być mniejsza niż |±3|.

W wyniku badań została opracowana implementacja kodeka standardu MPEG-2, który wykorzystuje wszystkie typy obrazu: I, P oraz B. Kodek taki jest w stanie kompresować w czasie rzeczywistym obraz o rozdzielczości HDTV (np. 1920×1152) przy użyciu procesu estymacji i kompensacji ruchu. Dodatkowo możliwy jest tryb kodowania z „zerowym" wektorem ruchu.

Celem pracy była skuteczna implementacja systemu kompresji i dekompresji sekwencji obrazów wizyjnych w standardzie MPEG-2 pracującego w trybie czasu rzeczywistego z wykorzystaniem technologii układów programowalnych FPGA. Jednocześnie implementacja ta była ograniczona pod względem możliwości wykorzystania zasobów układu FPGA.

Zaproponowana przez autora teza: **Odpowiednia modyfikacja algorytmów pozwala na zaimplementowanie w układach FPGA kodeka standardu MPEG-2, wykorzystującego w kompresji obrazy typu I, P oraz B, spełniającego wymogi czasu rzeczywistego i kodującego obrazy o rozdzielczości HDTV**, została udowodniona poprzez realizację następujących zadań:

- ustalono minimalną dokładność współczynników mnożenia dyskretnej transformaty kosinusowej,
- przeanalizowano wpływ niedokładności realizacji współczynników mnożenia transformaty DCT na dokładność rekonstrukcji pikseli,
- zaimplementowano podstawowe algorytmy jednowymiarowej dyskretnej transformacji kosinusowej,
- dokonano wyboru najlepszego algorytmu 1D-DCT pod względem dokładności oraz wykorzystanych zasobów układu programowalnego FPGA,
- zrealizowano dwuwymiarową dyskretną transformację kosinusową w układzie FPGA,

- zaimplementowano proces kwantyzacji i dekwantyzacji według wymagań standardu MPEG-2,
- zrealizowano układ uporządkowania kolejności przetwarzania współczynników transformacji 2D-DCT zgodnego ze specyfikacją standardu ISO/IEC 13818-2,
- zrealizowano koder i dekoder entropijny,
- zaimplementowano algorytmy estymacji ruchu,
- dokonano modyfikacji algorytmu E3SS pod kątem implementacji w układach FPGA.

Do oryginalnych osiągnięć autora, które powstały w wyniku prowadzonych prac badawczych związanych z implementacją kodeka standardu MPEG-2, należy zaliczyć:

- modyfikację algorytmu Chena dwuwymiarowej dyskretnej transformacji kosinusowej, uwzględniającą wpływ dokładności współczynników mnożenia na wartości wyjściowe otrzymane w wyniku transformacji 2D-DCT,
- opracowanie kodu VHDL oraz implementację w układzie programowalnym algorytmu efektywnego trójkrokowego przeszukiwania, wcześniej nie zrealizowanego w układach FPGA ani też w jakikolwiek inny sposób sprzętowy,
- modyfikację algorytmu efektywnego trójkrokowego przeszukiwania mającą na celu uproszczenie sterowania sprzętowej realizacji algorytmu *E3SS*,
- opracowanie kodeka, który przy tworzeniu obrazów typu P oraz B wykorzystuje proces estymacji oraz kompensacji ruchu ale równocześnie może skompresować obraz o 26 klatkach na sekundę i rozdzielczości 1920×1152 przy typowej grupie obrazów GOP,
- opracowanie i przetestowanie szeregu modułów, jako wkładu do światowej bazy elementów IP-Cores (ang. *Intellectual Property Core*), w tym: dwuwymiarowej dyskretnej transformacji kosinusowej 2D-DCT, kwantyzatora, dekwantyzatora, kodera entropijnego, dekodera entropijnego, estymatora ruchu,
- opracowanie kompletnego systemu kodowania i dekodowania obrazów ruchomych w standardzie MPEG-2.

Należy zaznaczyć, że ważnym aspektem tych prac jest ich umiejscowienie w światowych badaniach nad budową systemów rekonfigurowanych dedykowanych do konkretnych obliczeń, osiągających niezwykłe szybkości obliczeń i zachowujące dużą elastyczność.

Pomimo iż referowane w rozprawie efekty prac badawczych stanowią opracowanie kompletnego systemu kodowania i dekodowania obrazów ruchomych w standardzie MPEG-2, należy zwrócić uwagę na dalsze możliwości rozwoju opracowanego kodeka. W szczególności przyszłe prace badawcze winny dotyczyć następujących zagadnień:

- opracowanie realizacji układu sterującego parametrami kwantyzacji uwarunkowanego wymaganym stopniem oraz jakością kompresji, informacją dostarczaną z bufora kodera MPEG-2 oraz parametrami medium transmisyjnego,
- realizacja w układach programowalnych FPGA kodera/dekodera standardu MPEG-2 spełniającego wymogi czasu rzeczywistego i wykorzystującego profile skalowalne przy rozdzielczości obrazów odpowiadającej systemowi HDTV,
- opracowanie algorytmu estymacji ruchu, który charakteryzowałby się dużą dokładnością odwzorowania ruchu przy jednoczesnej minimalizacji liczby wykonywanych operacji oraz wykorzystywanych zasobów układu FPGA.

Prace te pozwolą na osiągnięcie dalszej poprawy parametrów kompresji obrazów ruchomych, a w szczególności szybkości przetwarzania, jakości kompresji mierzonej zarówno większym upakowaniem jak również mniejszymi wnoszonymi zniekształceniami. Prace takie są szczególnie ważne ze względu na ciągle zwiększające się stosowane rozdzielczości obrazów oraz oczekiwanie wyższej ich jakości.

11. Literatura

[1] Agha S., Dwayer V.M. : *Algorithms and VLSI Architectures for MPEG-4 Motion Estimation*. Electronic Systems and Control Division Research 2003, pp. 24-27

[2] Bhaskaran V., Konstantinides K.: *Image and Video compression standards. Algorithms Architectures*, Boston/Dordrecht/London, Kluwer Academic Publishers 1997, Second Edition

[3] bmrc.berkeley.edu/ftp/pub/multimedia/mpeg/EncodeData/tennis.tar.gz

[4] Broadcom*, BCM3560 Digital TV System-on-Chip*, www.broadcom.com

[5] Bukhari K., Kuzmanov G., Vassiliadis S.: *DCT and IDCT Implementations on Different FPGA Technologies*, Proceedings of the 13th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC '02), Veldhoven, The Netherlands 2002, pp. 232–235

[6] Cavalli F., Cucchiara R., Piccardi M., Prati A.: *Performance analysis of MPEG-4 decoder and encoder*, Proceedings of International Symposium on Video/Image Processing and Multimedia Communications (VIPromCom-2002), Zadar, Croatia 2002, pp. 227-231

[7] Chotin R., Dumonteix Y., Mehrez H.: *Use of Redundant Arithmetic on Architecture and Design of a High Performance DCT Macro-block Generator*, Proceedings of the 15th International Conference on Design of Circuits and Integrated Systems (DCIS'00), Montpellier, France 2000, pp. 428-433

[8] Chunag Y.-J., Wu J.-L.: *An Efficient Matrix-Based DCT Splitter/Merger for MPEG-2-to-AVC/H.264 Transform Kernel Conversion,* IEEE Transactions on Circuits And Systems For Video Technology, Vol.17, No.1, 2007, pp.120-125

[9] Conexant: *MPEG-2 encoder CX23416*, www.conexant.com

[10] Cook R., Jean J., Chen J.: *Accelerating an MPEG-2 Encoder Utilizing Reconfigurable Computing*, www.cs.wright.edu/~jjean/research/cerc.ps

[11] Dąbrowska A., Wiatr K.: *Chen and Lee Fast DCT Modified Algorithms Implemented in FPGA Chips for Real-Time Image Compression*, Journal of Applied Computer Science 2005, Vol. 13, No 1, pp. 7-16

[12] Dąbrowska A., Wiatr K.: *Chen and Loeffler Fast DCT Modified Algorithms Implemented in FPGA Chips for Real-time Image Compression*, Proc. of the Int. Conf. On Computer Vision and Graphics, Warszawa 2004, Springer 2006, p.768-773

[13] Dąbrowska A., Wiatr K.: *Fast Discrete Cosine Transform Algorithms Implemented in FPGA Structures for Real-Time Image Compression*, Proceedings of 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Slovakia Stará Lesná 2004, pp. 179-182

[14] Dąbrowska A., Wiatr K.: *Fast-DCT Modified Algorithms Implemented in FPGA Chips for Real-Time Image Compression*, Proc. of IFAC Workshop Programmable Devices and Systems, Cracow 2004, pp. 479-483

[15] Dąbrowska A., Wiatr K.: *Implementacja algorytmów Chena i Lee transformacji DCT w układach FPGA*, Mat. VII Konferencji Rekonfigurowalne Układy Cyfrowe, Szczecin 2004, s. 195-202

[16] Dąbrowska A., Wiatr K.: *Implementacja algorytmów dyskretnej transformacji kosinusowej w strukturach FPGA*, Mat. IV Konferencji Metody i Systemy Komputerowe, Kraków, ONT 2003, s. 393-398

[17] Dąbrowska A., Wiatr K.: *Implementacja algorytmów transformacji FDCT w strukturach FPGA*, Mat. VIII Konferencji Rekonfigurowalne Układy Cyfrowe, Szczecin 2005, s. 95-102

[18] Dąbrowska A., Wiatr K.: *Implementacja algorytmu E3SS estymacji ruch w układach FPGA*, IX Konferencja Rekonfigurowalne Układy Cyfrowe, Szczecin 2006, wyd. PAK 2006 nr.7bis, s. 68-71

[19] Dąbrowska A., Wiatr K.: *Implementacja elementów bezstratnej kompresji standardu MPEG-2 w układach FPGA*, IV Konferencja KNWS, Szklarska Poręba 2007, wyd. PAK 2007 nr.5, s. 36-38

[20] Dąbrowska A., Wiatr K.: *Implementacja kwantyzacji i dekwantyzacji w układach FPGA dla potrzeb kompresji obrazu,* Mat. V Konferencji Metody i Systemy Komputerowe, Kraków, ONT 2005, t.II s. 65-69

[21] Dąbrowska A., Wiatr K.: *Implementacja procesu estymacji ruchu w strukturach FPGA*, Mat. XIII Konferencji Sieci i systemy informatyczne, Łódź 2005, t. 2, s. 413-421

[22] Dąbrowska A., Wiatr K.: *Implementacja procesu kwantyzacji w strukturach FPGA dla potrzeb kompresji obrazu*, Zeszyty Naukowe AGH - Automatyka, Tom 9, Zeszyt 3, Kraków 2005, s. 317-324

[23] Dąbrowska A., Wiatr K.: *Modyfikacja algorytmów Chena i Loefflera transformacji FDCT implementowanej w układach FPGA*, Zeszyty Naukowe AGH, Automatyka, Tom 8, Zeszyt 3, Kraków 2004, s. 179-188

[24] Dąbrowska A., Wiatr K.: *Modyfikacja algorytmów dyskretnej transformacji kosinusowej dla potrzeb kompresji obrazów implementowanej w strukturach FPGA*, Mat. XI Konferencji Sieci i Systemy Informatyczne, Łódź 2003, s. 375-382

[25] Dąbrowska A., Wiatr K.: *Modyfikacja algorytmów transformacji FDCT dla potrzeb kompresji obrazów implementowanej w układach FPGA*, Kwartalnik Elektroniki i Telekomunikacji PAN, t.52, z.4, Warszawa 2006, s. 629-641

[26] Domański M.: *Zaawansowane techniki kompresji obrazów i sekwencji wizyjnych*, Poznań, Wydawnictwo Politechniki Poznańskiej, 2000, wyd. 2

[27] Drozdek A.: *Wprowadzenie do kompresji danych*, Warszawa, WNT, 1999

[28] Ferguson T.: *CSE5302 – Digital Video Coding and Compression.* www.csse.monash.edu.au/~timf/cse5302/motion.pdf

[29] ftp3.itu.ch/av-arch/video-site/9706_Por/q15a18.ps

[30] Furht B., Greenberg J., Westwater R. : *Motion estimation algorithms for video compression.* Kluwer Academic Publishers, London 1997

[31] Ge J., Mirchandani G.: *A New Hybrid Block-Matching Motion Estimation Algorithm*, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing 2002, vol. 4, pp. 4190

[32] Haskell B.G., Puri A., Netravali A.N.: *Digital video: An introduction to MPEG-2*, New York, Chapman & Hall, 1997

[33] Heim K.: *Metody kompresji danych*, Warszawa, Wyd. MIKOM, 2000

[34] Heron J., Trainor D., Woods R.: *Image Compression Algorithms Using Reconfigurable Logic*, Proceedings of the 31st Asilomar Conference on Signals, Systems and Computers, Asilomar, USA 1997, vol. 1, pp. 399-403

[35] http://bmrc.berkeley.edu/frame/research/mpeg/mpeg2faq.html

[36] ISO/IEC JTC1/SC29/WG11 N4668

[37] Jamro E., Wiatr K.: *Dynamic Constant Coefficient Convolvers Implemented in FPGA*, Lecture Notes in Computer Science – no 2438, Springer Verlag 2002, pp. 1110-11

[38] Kato H., Takishima Y., Nakajima Y.: *A Fast DV to MPEG-4 Transcoder Integrated With Resolution Conversion and Quantisation,* IEEE Transactions on Circuits And Systems For Video Technology, Vol.17, No.1, 2007, pp.111-119

[39] Klimesh M., Stanton V., Watola D.: *Hardware Implementation of a Lossless Image Compression Algorithm Using a Field Programmable Gate Array*, The Telecommunications and Mission Operations Progress Report, TMO PR 42-144, NASA 2001, pp. 1-11

[40] Kodaka T., Nakano H., Kimura K., Kasahara H.: *Parallel Processing using Data Localization for MPEG2 Encoding on OSCAR Chip Multiprocessor*, Proceedings of the Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04) - Volume 00, pp.119-127

[41] Kuo T.-Y., Chan C.-H.: *Fast Variable Block Size Motion Estimation for H.264 Using Likelihood and Correlation of Motion Field*, IEEE Transactions on Circuits And Systems For Video Technology, Vol.16, No.10, 2006, pp.1285-1195

[42] Lai-Man Po, Wing-Chung Ma: *A Novel Four-Step Search Algorithm for Fast Block Motion Estimation*, IEEE Transactions on Circuits and Systems for Video Technology, vol.6, no.3, 1996, pp. 313-317

[43] Lap-Pui Chau, Xuan Jing: *Efficient three-step search algorithm for block motion estimation in video coding*, ICASSP '03, vol.3, pp. 421-424

[44] Latha Pillai 2003: *Quantization*, www.xilinx.com/bvdocs/appnotes/xapp615.pdf

[45] Lee J., Shin I., Park H.: *Adaptive Intra-Frame Assignment and Bit-Rate Estimation for Variable GOP Length in H.264,* IEEE Transactions on Circuits And Systems For Video Technology, Vol.16, No.10, 2006, pp. 1271-1279

[46] Lee S., Kim J., Chae S.: *New Motion Estimation Algorithm Using Adaptively Quantized Low Bit-Resolution Image and Its VLSI Architecture for MPEG-2 Video Encoding.* IEEE Transactions on Circuits and Systems for Video Technology, vol.8, no. 6, 1998, pp. 734-744

[47] Lee S., Yun K., Choi K., Hong S., Moon S., Lee J.: *Java-Based Programmable Networked Embedded System Architecture with Multiple Application Support*, Proceedings of Conference on Chip Design Automation (ICDA 2000), Beijing, China 2000, pp. 448-451

[48] Liang J., Tran T.D.: *Fast Multiplierless Approximtioms of the DCT With the Lifting Scheme*, IEEE Transactions on Signal Processing, Vol.49, No.12, p. 3032-3044

[49] Makris P., Vincent N.: *A new measurement of compressed image quality*, Machine Graphics and Vision vol.9, no. 1/2, 2000, pp. 369-378

[50] Materni S., Fischer S.: *Implementation of an MPEG-2 Codec on the ADI Blackfin DSP*, SHARC 2001 International DSP Conference, Boston 2001, www.fastcom.ch/pages/news_events/archives/Sharc2001.doc

[51] *Multimedia. Algorytmy i standardy kompresji* – pod redakcja W.Skarbka, Warszawa, Akademicka Oficyna Wydawnicza PLJ, 1998

[52] Naito Y., Kuroda I.: *H.263 mobile video codec based on a low power consumption digital signal processor*, Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98), Seattle, USA 1998, vol.5, pp. 3041-3044

[53] Nec, www.am.necel.com

[54] Netravali A.N., Haskell B.G.: *Digital pictures. Representation, Compression and Standards*, New York/London, Plenum Press, 1995

[55] Nikara J., Vassiliadis S., Takala J., Liuha P.: *FPGA-Based Variable Length Decoders*, International Conference on Very Large Scale Integration of System-on-Chip, Darmstadt, Germany 2003, pp. 437-441

[56] *NVIDIA Video Processing Engine. The Ultimate Digital Entertainment Experience. Technical Brief.* www.nvidia.com

[57] Ouyang Y.C., Lu G.T.: *A Fast Objected-Base Efficient Three-Step Search Algorithm for Block Motion Estimation*, Proceedings of IEEE ICSS2005 International Conference On Systems & Signals, Kaohsiung, Taiwan 2005 pp. 404-408

[58] Park S.R., Burleson W.: *Reconfiguration for power saving in real-time motion estimation*, Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98), Seattle, USA 1998, vol.5, pp. 3037-3040

[59] Pasierbiński J., Zbysiński P.: *Układy programowalne w praktyce*, wydanie 2, Warszawa, Wydawnictwa Komunikacji i Łączności 2002

[60] Philips: *SAA6752HS MPEG-2 video and MPEG-audio/AC-3 audio encoder with multiplexer*,

[61] Pourreza H.R., Rahmati M., Behazin F. *Weighted Multiple Bit-plane Matching, a Simple and Efficient Matching Criterion for Electronic Digital Image Stabilizer Application*, Proc. of the 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002, University of West Bohemia, Campus Bory, Plzen-Bory, Czech Republic 2002, WSCG (Posters) 2002: pp. 37-40

[62] Rabbani M., Jones P.W.: *Digital Image Compression Techniques*, Bellingham, SPIE Optical Enggineering Press, 1991

[63] Richmond II R.S., Ha D.S.: *A Low-Power Motion Estimation Block for Low Bit-Rate Wireless Video*, Proceedings of International Symposium on Low Power Electronics and Design (ISLPED), Huntington Beach, USA 2001, pp. 60-63

[64] Roma N., Sousa L.: *Efficient and Configurable Full-Search Block-Matching Processors*, IEEE Transactions on Circuits and Systems for Video Technology, vol.12, no.12, 2002, pp. 1160-1167

[65] Rudberg M.K., Wanhammar L.: *Implementation of a fast MPEG-2 compliant Huffman decoder*, Proceedings of EUSIPCO ´96, Trieste, Italy 1996, vol. 3, pp. 1467-1470

[66] Sankowski D., Pląskowski A., Mosorov V., Strzecha K., Jeżewski S.: *Image segmentation algorithms for industrial process tomography,* 2nd World Congress on Industrial Process Tomography, Hanover 2001, s. 736-741

[67] Sankowski D., Strzecha K., Jeżewski S.: *Image processsing in physical parameters measurement,* 16th IMEKO World Congress, Vienna 2000, s. 277-283

[68] Sayood K.: *Kompresja danych wprowadzenie*, Warszawa, RM, 2002, wyd.1

[69] SGS-Thomson Microelectronics: *Sti3500A. MPEG-2/CCIR 601 Video Decoder*

[70] Sima M., Cotofana S.D., van Eijndhoven J.T.J., Vassiliadis S., Vissers K.A.: *An 8x8 IDCT Implementation on an FPGA-augmented TriMedia*, Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2001), Rohnert Park, California, USA 2001, pp. 160-169

[71] *Single-Chip MPEG2 Video Encoder LSI,* www.sony.net

[72] Siram S., Ching-Yu Hung: *MPEG-2 Video Decoding on the TMS320C6X DSP Architecture*, Proceedings of IEEE Asilomar Conference on Signals, Systems & Computers, Pacific Grove, USA 1998, vol. 2. pp. 1735-1739

[73] Skarbek W.: *Methods of digital image archivization. Part three: Compressing images*, Machine Graphics and Vision vol.2, no. 1, 1993, pp. 53-86

[74] Skarbek W.: *Methods of digital image archivization. Part two: Binary coding and binary image compression*, Machine Graphics and Vision vol.1, no. 4, 1992, pp. 571-604

[75] Sonja Grgic S., Marta Mrak M., Mislav Grgic M.: *Comparison of JPEG Image Coders*, Proceedings of the 3rd International Symposium on Video Processing and Multimedia Communications, VIPromCom-2001, Zadar, Croatia 2001, pp. 79-85

[76] Sony: *CXD1922Q MPEG-2 Video Encoder* www.sony.com/semi

[77] Sorwar G., Mushed M., Dooley L.: *Fast Block-Based True Motion Estimation Using Distance Dependent Thresholds.* Journal of Research and Practice in Information Technology. Vol. 36, No. 3, 2004, pp.157-169

[78] Stiller C., Konrad J.: *Estimating Motion in Image Sequences. A Tutorial on Modeling and Computation of 2D Motion*, IEEE Signal Processing Magazine, July 1999, pp.70-91

[79] Sudharsan S., Sinnathamby M.: *Support for Variable Length Decode on Embedded Processors*, Proceedings of Workshop on Media and Signal Processors for Embedded Systems and SoCs (MASES04), Washington D.C., USA 2004, pp.33-40

[80] Symes P.D. : *Video compression: Fundamental compression techniques and an overview of the JPEG and MPEG compression systems.* McGraw-Hill, New York 1998

[81] Tadeusiewicz R., Flasiński M.: *Rozpoznawanie obrazów*, Warszawa, PWN, 1991

[82] Tadeusiewicz R., Korohoda P.: *Komputerowa analiza i przetwarzanie obrazów*, Kraków, Wydawnictwo Fundacji Postępu Telekomunikacji, 1997

[83] Tadeusiewicz R.: *Systemy wizyjne robotów przemysłowych*, Warszawa, Wydawnictwo Naukowo-Techniczne, 1992

[84] Tatas K., Siozios K., Soudris D., Thanailakis A.: *Power-Efficient Implementation of Multimedia Applications on Reconfigurable Platforms*, Proceedings of FPL 2003 Conference, Lisbon, Portugal 2003, pp. 1032-1035

[85] Trainor D., Heron J., Woods R.: *Implementation of the 2D DCT Using a Xilinx XC6264 FPGA*, IEEE Proceedings on the Workshop on Signal Processing Systems, SiPS' 97, IEEE Press, pp. 541-550

[86] Wayner P.: *Compression Algorithms for Real Programmers*, San Diego, Morgan Kaufman, Academic Press, 2000

[87] Wiatr K., Jamro E.: *Implementation of Multipliers in FPGA Structure*, Proc. of the IEEE Int. Symp. on Quality Electronic Design, Los Alamitos CA, IEEE Computer Society 2001, pp. 415- 420

[88] Wiatr K.: *Dedicated Hardware Processors for a Real-Time Image Data Pre-Processing Implemented in FPGA Structure*. Lecture Notes in Computer Science - no 1311, Springer-Verlag 1997, vol. II, pp. 69-75

[89] Wiatr K.: *Dedicated System Architecture for Parallel Image Computation used Specialised Hardware Processors Implemented in FPGA Structures*. International Journal of Parallel and Distributed Systems and Networks, vol. 1, No. 4, Pittsburgh 1998, pp. 161-168

[90] Wiatr K.: *Sprzętowe implementacje algorytmów przetwarzania obrazów w systemach wizyjnych czasu rzeczywistego*, Kraków, wyd. Naukowo-Dydaktyczne AGH 2002

[91] Woods R., Cassidy A., Gray J.: *VLSI Architectures for Field Programmable Gate Arrays: A Case Study*, Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley, USA 1996, pp.2-9

[92] www.alma-tech.com

[93] www.amphion.com/products.html

[94] www.barco-silex.com

[95] www.dumavideo.com

[96] www.eng.uci.edu/morphosys/docs/JVSP.ps

[97] www.hpl.hp.com

[98] www.mathstar.com

[99] www.xilinx.com

[100] Xilinx: xapp616, www.xilinx.com

[101] Xilinx: xapp621, www.xilinx.com

[102] Yamg W., Lu Y., Rajan D., Chia L.-T., Ngan K.N.: *Dynamic Programming-Based Reverse Frame Selection for VBR Video Delivery Under Constrained Resources*, IEEE Transactions on Circuits And Systems For Video Technology, Vol.16, No.11, 2006, pp. 1362-1375

[103] Yang S., Wolf W., Narayanan V.: *Power Modeling of Motion Estimation VLSI Architecture*, Proceedings of 5th Workshop on Media and Streaming Processors, IEEE, San Diego, USA 2003

[104] Zoran, *Activa 100 MPEG AV Encoder for DVD Recorders*, www.zoran.com

12. Dodatek

Spis zawartości załączonej płyty CD

Katalog **DCT**
Pliki VHDL zawierające opisy bloków algorytmów dyskretnej transformacji kosinusowej i odwrotnej dyskretnej transformacji kosinusowej

Katalog **KWANTYZACJA**
Pliki VHDL zawierające opisy bloków kwantyzacji i dekwantyzacji

Katalog **ESTYMACJA**
Pliki VHDL zawierające opisy bloków algorytmów estymacji ruchu

Katalog **VLC**
Pliki VHDL zawierające opisy bloków kodera oraz dekodera entropijnego

Katalog **MPEG-2**
Pliki VHDL zawierające opisy poszczególnych koderów i kodeków wykorzystujących standard MPEG-2