Wykorzystanie mikroprogramowanych układów sterujących do implementacji przykładowego algorytmu przetwarzania danych wizyjnych

Małgorzata Kołopieńczyk, Sebastian Pawlak, Wojciech Zając

Streszczenie: W artykule przedstawiono propozycję zastosowania mikroprogramowanych układów sterujących do implementacji algorytmu przetwarzania danych wizyjnych. Omówiono strukturę implementowanego fragmentu algorytmu HECA, zaproponowano strukturę mikroprogramowanego układu sterującego oraz przedstawiono sprzętową implementację algorytmu.

Słowa kluczowe: mikroprogramowany układ sterujący, przetwarzanie danych wizyjnych

e- WPROWADZENIE

Algorytmy przetwarzania danych wizyjnych bardzo często znajdują realizację w implementacji sprzętowej. Sprzyja temu zarówno wzrost mocy obliczeniowej struktur programowalnych, jak również coraz bardziej przystępna cena tych układów.

Do licznych przykładów wykorzystania rosnących możliwości implementacji złożonych układów cyfrowych można zaliczyć inż. stosowanie coraz bardziej złożonych algorytmów przetwarzania, transmisji oraz korekcji zakłóceń danych wizyjnych [5].

W artykule omówiono przykład implementacji sprzętowej części algorytmu przetwarzania danych wizyjnych HECA [5, 6] z wykorzystaniem mikroprogramowanego układu sterującego [2, 9]. Do syntezy i implementacji struktur wykorzystano oprogramowanie Xilinx ISE 9.1i. Platformę docelową stanowił układ FPGA Xilinx Virtex–II Pro (xc2vp30-7ff896c) [4].

Do implementacji algorytmu przetwarzania danych wizyjnych wykorzystano mikroprogramowany układ sterujący (ang. Compositional Microprogram Control Unit, CMCU) ze współdzieleniem kodów i konwerterem adresu.

Metoda współdzielenia kodów polega na przekształceniu adresu, reprezentowanego jako pary <kod łańcucha, kod elementu łańcucha>, w adres mikroinstrukcji. Szczegółowy opis metody współdzielenia kodów można znaleźć w pracach [1, 3, 8].

Klasyczna metoda implementacji jednostki sterującej jako skończonego automatu stanów pochłania zasoby układów programowalnych, które mogą być w efektywniejszy sposób wykorzystane przez inne bloki projektowanego systemu.

Zaproponowane w artykule wykorzystanie mikroprogramowanego układu sterującego ze współdzieleniem kodów i konwerterem adresu umożliwia zachowanie minimalnego rozmiaru pamięci (dzięki zastosowaniu konwertera adresu) w porównaniu z dobrze znanymi strukturami opisanymi w literaturze [7, 10] oraz minimalizuje ilość elementów LUT w układzie generującym funkcje wzbudzeń automatu (dzięki współdzieleniu kodów).

2. ALGORYTM HECA

Hybrydowy algorytm maskowania błędów transmisji obrazu stałego HECA (Hybrid Error Concealment Algorithm) [5, 6] przeznaczony jest do zwalczania zakłóceń transmisji, powstających przy przesyłaniu założonej klasy sygnału wizyjnego. Algorytm realizuje technikę maskowania błędów, w odróżnieniu od innej klasy algorytmów, przeznaczonych do korygowania błędów.

Technika maskowania błędów [11, 12] ma charakter nieinwazyjnej korekcji uszkodzonego sygnału. Jej stosowanie jest możliwe w praktycznie każdym systemie transmisyjnym, w szczególności w systemach już istniejących. Metoda ta może jako jedyna być także zastosowana w systemach archiwizacji danych, w których dane już istnieją, a proces ich składowania nie przewidywał stosowania technik korekcyjnych. Proces maskowania realizuje określone kroki w celu minimalizacji wpływu zakłóceń na jakość sygnału, zazwyczaj wyzyskując szczątkową redundancję sygnału, pozostałą po niedoskonałej dekorelacji danych. Podjęte zabiegi korekcyjne z założenia nie mają odtworzyć oryginalnej postaci sygnału, jedynie mają zbliżyć wartości uszkodzonych danych do wartości oryginalnych. Co bardzo istotne, stosowanie metody maskowania zakłóceń jest wysoce celowe w odniesieniu do danych wizyjnych, ponieważ z natury rzeczy systemy tego rodzaju tolerują lub nawet celowo wprowadzają pewne niedokładności w treści sygnału.

Omawiany algorytm HECA przeznaczony jest do maskowania zakłóceń transmisji monochromatycznego obrazu stałego, przetwarzanego z wykorzystaniem dyskretnej transformaty kosinusowej DCT. Jego działanie podzielone jest na pięć etapów, które szczegółowo przedstawione są w pracach [5, 6]. Niniejsza praca

dotyczy implementacji pierwszego stopnia algorytmu – wejściowego filtra dolnoprzepustowego.

4. IMPLEMENTACJA

Algorytm HECA operuje na blokach obrazu o rozmiarze 8 × 8 pikseli. Każdemu blokowi odpowiada macierz sześćdziesięciu czterech 8-bitowych współczynników. Współczynniki te trafiają do wejściowego filtra dolnoprzepustowego.

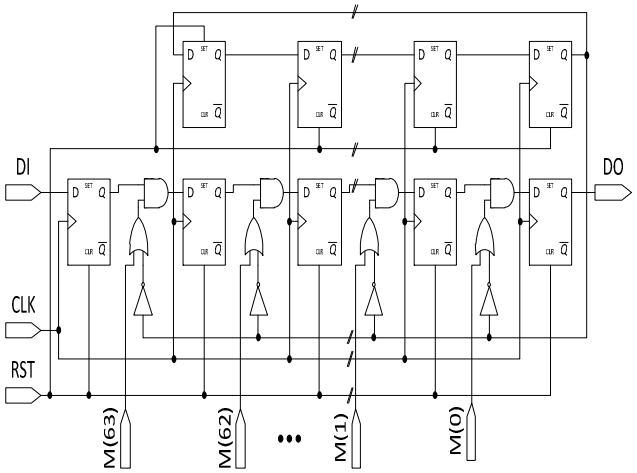
Wejściowy filtr dolnoprzepustowy został zaimplementowany jako układ synchroniczny zawierający w swej strukturze kolejkę FIFO. Taka realizacja filtra jest uzasadniona, ponieważ dane wizyjne transmitowane są w postaci strumienia – w sposób ciągły.

Pomiędzy kolejne rejestry kolejki wbudowane zostały bloki kombinacyjne realizujące filtrację danych jako maskowanie określonych bitów macierzy współczynników.

Do zerowania układu służy synchroniczne wejście *RST*. Układ po wyzerowaniu potrzebuje sześćdziesięciu czterech cykli zegarowych do zapełnienia kolejki danymi pobieranymi z wejścia *DI*. W sześćdziesiątym piątym cyklu zegarowym przetworzone dane zaczynają się pojawiać na wyjściu układu – *DO*. Od tego momentu możliwe jest dostarczanie współczynników do filtra w sposób ciągły, przy jednoczesnym pobieraniu przetworzonych współczynników z wyjścia układu. Początek kolejnego bloku obrazu sygnalizowany jest zboczem narastającym sygnału wyjściowego *BLOCK_SYNC*. Na rysunku 2 przedstawiono kod w języku VHDL opisujący omawiany filtr dolnoprzepustowy.

```
REG_N_INSTANCE_MSB:
REG_N
  generic map (N)
  port map (DI, CLK, RST, FIFO_B(DEPTH));
BL_S <= RND_FLAG(DEPTH-1);
BLOCK_SYNC <= not BL_S;
GEN_0:
for I in DEPTH-1 downto 0
generate
  MASK(I) <= (others=>((not BL_S) or MASK_ROM(G)(I)));
  AND_N_INSTANCE:
  AND_N
    generic map (N)
    port map (FIFO_B(I+1), MASK(I), FIFO_A(I));
  REG_N_INSTANCE:
  REG_N
    generic map (N)
    port map (FIFO_A(I), CLK, RST, FIFO_B(I));
  GEN_1:
  if I < DEPTH-1
  generate
    DFF_INSTANCE:
    DFF
      port map (RND_FLAG(I+1), CLK, RST, RND_FLAG(I));
  end generate;
  GEN_2:
  if I = DEPTH-1
  generate
    DFF_S_INSTANCE:
    DFF_S
      port map (RND_FLAG(0), CLK, RST, RND_FLAG(I));
  end generate;
end generate;
```

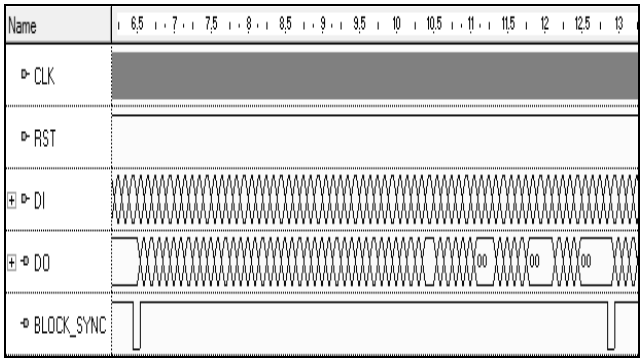
Rys. 2. Kod VHDL opisujący wejściowy filtr dolnoprzepustowy HECA



Rys. 3. Schemat logiczny filtra dolnoprzepustowego HECA dla pojedynczej składowej barw RGB

Na rysunku 3 przedstawiono schemat logiczny filtra dolnoprzepustowego, pracującego dla pojedynczej składowej barw RGB. Wektor *M(63..0)* jest maską bitową odpowiadającą progowi granicznemu *g* algorytmu HECA [5, 6]. Gdy bit *M(x) = 0*, *x*-ty współczynnik bloku obrazu jest zerowany, w przeciwnym przypadku pozostaje niezmieniony.

Do syntezy i implementacji wykorzystano oprogramowanie Xilinx ISE 9.1i. Na rysunku 4 przedstawiono wyniki symulacji jednostki filtra przeprowadzonej w oprogramowaniu Active-HDL v.8.1.



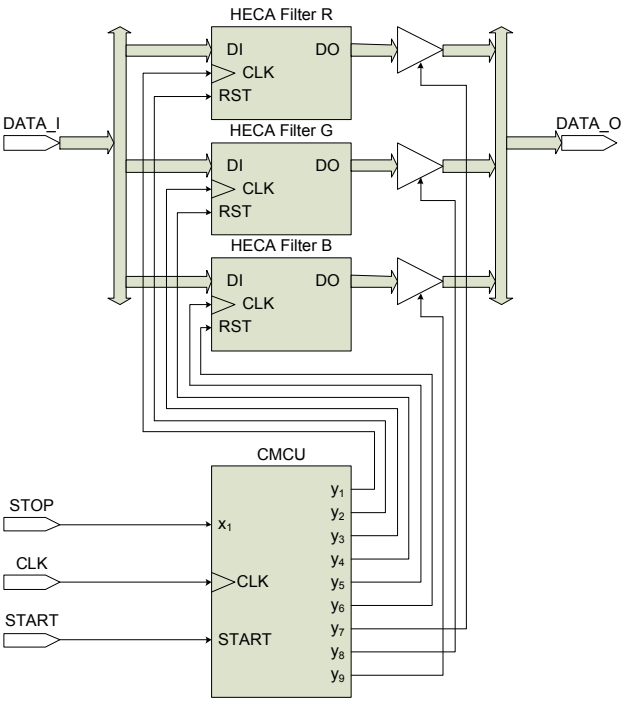
Rys. 4. Działanie filtra dla progu *g* = 11

W tabeli 1 przedstawiono zużycie zasobów sprzętowych dla filtra dolnoprzepustowego HECA.

Tabela 1.
Zasoby wykorzystane do implementacji filtra HECA

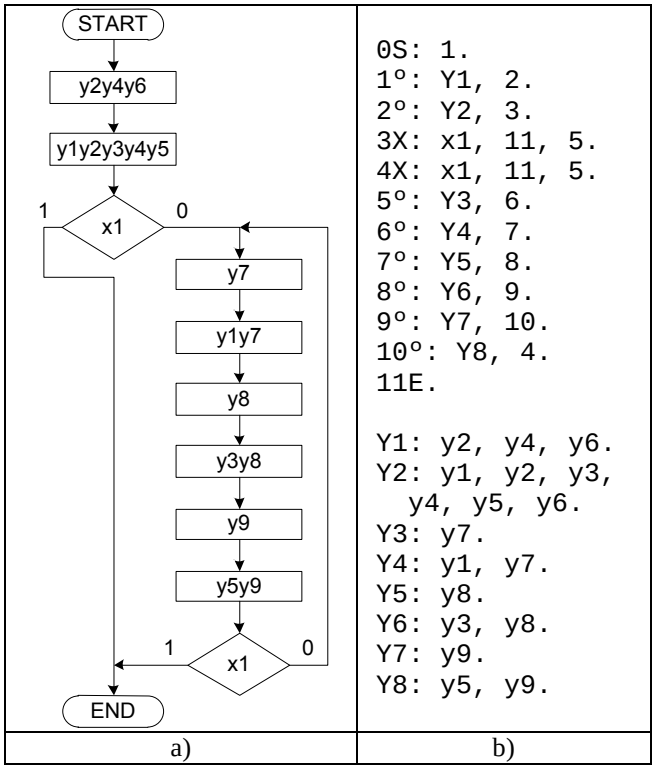
Nazwa zasobu	Liczba
LUT4	123
Przerzutniki	593

Na rysunku 5 przedstawiono schemat blokowy układu filtra dolnoprzepustowego algorytmu HECA dla obrazu kolorowego. Współczynniki trafiają na wejścia *DI* poszczególnych filtrów HECA poprzez współdzieloną magistralę *DATA_I*. Przetworzone współczynniki poprzez bufory trójstanowe trafiają na magistralę *DATA_O*. Jednostka CMCU generuje sygnały zerujące (*RST*), sygnały zegarowe (*CLK*) oraz sygnały sterujące buforami trójstanowymi.



Rys. 5. Schemat blokowy układu HECA dla przetwarzania trzech składowych barw RGB

Do opisu algorytmu działania implementowanej jednostki sterującej wykorzystano sieć działań. Na rysunku 6 przedstawiono jej graficzną oraz tekstową reprezentację.



Rys. 6. Sieć działań: a) reprezentacja graficzna, b) reprezentacja tekstowa

Do edycji sieci działań oraz jej eksportu do postaci tekstowej wykorzystano oprogramowanie *FCA Diagrams Editor v.1.0*. Następnie powstały plik posłużył jako plik wejściowy dla oprogramowania *FCA2CMCU v.2.1*, które dokonało analizy reprezentacji tekstowej sieci działań i na jej podstawie wygenerowało kod – w syntezowalnym podzbiorze języka VHDL – opisujący działanie mikroprogramowanej jednostki sterującej.

W tabeli 2 przedstawiono zużycie zasobów sprzętowych dla jednostki CMCU.

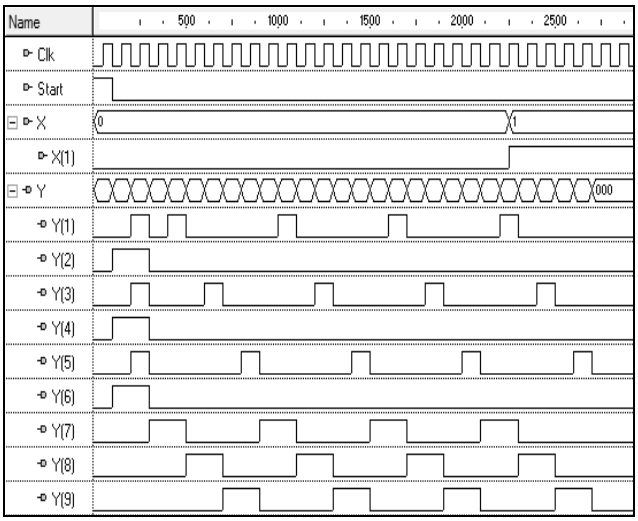
Tabela 2.
Zasoby wykorzystane do implementacji CMCU

Nazwa zasobu	Liczba
LUT4	12
Przerzutniki	6
Bloki pamięci BRAM	1

Na rysunku 7 przedstawiono wyniki symulacji jednostki sterującej CMCU.

Na rysunku 8 przedstawiono kod VHDL jednostki HECA RGB.

W tabeli 3 przedstawiono wykorzystanie zasobów dla kompletnego modułu HECA RGB.



Rys. 7. Działanie układu CMCU

Tabela 3.
Zasoby wykorzystane do implementacji HECA RGB

Nazwa zasobu	Liczba
LUT4	389
Przerzutniki	1785
Bloki pamięci BRAM	1

```
signal DO_R : STD_LOGIC_VECTOR(N - 1 downto 0);
signal DO_G : STD_LOGIC_VECTOR(N - 1 downto 0);
signal DO_B : STD_LOGIC_VECTOR(N - 1 downto 0);
signal Y : STD_LOGIC_VECTOR(1 to 9);
signal BL_S : STD_LOGIC_VECTOR(1 to 3);
signal RST : STD_LOGIC_VECTOR(1 to 3);
begin
   RST(1) <= not Y(2);
   RST(2) <= not Y(4);
   RST(3) <= not Y(6);
   HECA_R_INSTANCE:
   HECA
      generic map (N, 64, 10)
      port map (DATA_I, DO_R, Y(1), RST(1), BL_S(1));
   HECA_G_INSTANCE:
   HECA
      generic map (N, 64, 10)
      port map (DATA_I, DO_G, Y(3), RST(2), BL_S(2));
   HECA_B_INSTANCE:
   HECA
      generic map (N, 64, 10)
      port map (DATA_I, DO_B, Y(5), RST(3), BL_S(3));
   CMCU_INSTANCE:
   CMCU_OLC_U2 port map (CLK, START, STOP, Y);

   with Y(7 to 9) select
      DATA_O <= DO_R when "100",
                DO_G when "010",
                DO_B when "001",
                (others => 'Z') when others;
```

Rys. 8. Kod VHDL opisujący jednostkę HECA dla przetwarzania trzech składowych barw RGB

5. PODSUMOWANIE

Przeprowadzone prace pozwalają na sformułowanie szeregu wniosków. Wykazano, że możliwe jest wydajne implementowanie złożonych rzeczywistych algorytmów przetwarzania danych wizyjnych w układach mikroprogramowanych.

Opis dany w formie algorytmicznej może w nieskomplikowany sposób być przekształcony do postaci opisu tekstowego, z którego można uzyskać opis w języku VHDL i następnie przeprowadzić syntezę projektowanego układu.

Dalsze prace zostaną ukierunkowane na realizację sprzętową omawianego fragmentu algorytmu HECA oraz na opracowanie sprzętowe jego kolejnych elementów.

LITERATURA

[1] Barkalov A., Titarenko L., Kołopieńczyk M., „Optymalizacja jednostki sterującej poprzez zastosowanie metody współdzielenia kodów”, *Pomiary Automatyka Kontrola*, nr 7, ss. 29-31, 2006.

[2] Bomar B.W., „Implementation of microprogrammed control in FPGAs”, *IEEE Transactions on Industrial Electronics*, Vol. 49, ss. 415-422, 2002.

[3] Kołopieńczyk M., *Application of Address Converter for Decreasing Memory Size of Compositional Mikroprogram Control Unit with Code Sharing*, Zielona Góra, University of Zielona Góra Press, ss. 88, 2008.

[4] Xilinx, Products and Services. www, Xilinx Corporation, http://www.xilinx.com/products.

[5] Sielicki A., Zając W. *Nowa koncepcja korekcji zakłóceń transmisji cyfrowych danych wizualnych z wykorzystaniem techniki maskowania błędów*. Pomiary, Automatyka, Kontrola nr 2-3/2003, s. 26-28

[6] Zając W. “An Error Concealment Algorithm for Digital Image Transmission.” W: *Proceedings of XIV International Symposium on Computer and Information Sciences*, Kusadasi Turcja, October 1999

[7] Barkalov A., Titarenko L. *Logic synthesis for FSM-based control units.* Berlin : Springer-Verlag, 2009

[8] Barkalov A., Titarenko L. *Logic synthesis for compositional microprogram control units.* Berlin : Springer-Verlag, 2008

[9] Adamski M, Barkalov A. *Architectural and sequential synthesis of digital devices*. Zielona Góra : University of Zielona Góra, 2006

[10] Barkalov A. Węgrzyn M. *Design of control units with programmable logic.* Zielona Góra : University of Zielona Góra Press, 2006

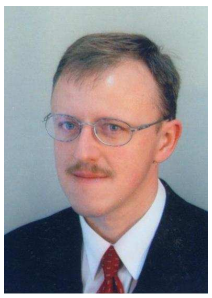
[11] Baroumand, A.; Avanaki, A.N. *Sequential best range matching: An error concealment technique with application in high packet loss image transmission*. IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.

[12] Wei-Ying Kung; Chang-Su Kim; Kuo, C.C.J. A *spatial-domain error concealment method with edge recovery and selective directional interpolation.* Multimedia and Expo, 2003. ICME ‘03. Proceedings. 2003 International Conference on

dr inż. Małgorzata Kołopieńczyk
Uniwersytet Zielonogórski
Wydział Elektrotechniki Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Licealna 9
65-417 Zielona Góra

el.: 68 328 25 99
e-mail: m.kolopienczyk@iie.uz.zgora.pl



dr inż. Wojciech Zając
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Licealna 9
65-417 Zielona Góra

el.: 68 328 25 99
e-mail: w.zajac@iie.uz.zgora.pl

mgr inż. Sebastian Pawlak
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Licealna 9
65-417 Zielona Góra

el.: 68 328 25 99
e-mail: s.pawlak@iie.uz.zgora.pl