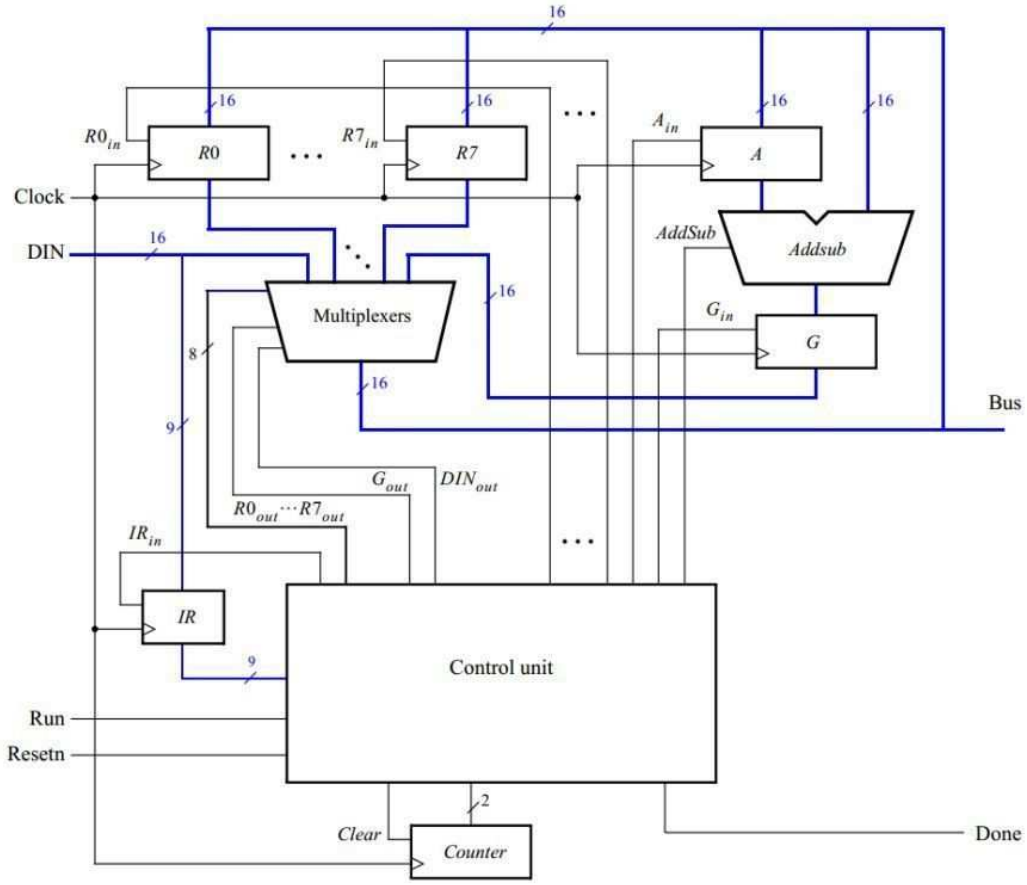
Projekt prostego procesora

Opracowany przez Rafała Walkowiaka dla zajęć z PTC 2012/2013 w oparciu o Laboratory Exercise 9 Altera Corporation

Rysunek 1 przedstawia schemat układu cyfrowego stanowiącego elementy struktury prostego procesora. W skład układu wchodzą: rejestry 16 bitowe, multiplekser, sumator, licznik i układ sterujący. Dane są wprowadzane do systemu za pomocą 16 bitowe magistrali wejściowej. Dane mogą być ładowane do rejestrów za pomocą multipleksera. Multiplekser umożliwia również przesyłanie danych między rejestrami.

Operacje arytmetyczne są realizowane w oparciu o zawartość rejestru akumulatora A. Drugi operand jest pobierany z rejestru R*, wynik działania jest zapisywany do rejestru G, a następnie przepisywany do dowolnego rejestru R.



Rysunek 1 Układ cyfrowy prostego procesora

System może realizować różne operacje zarządzane za pomocą układu sterującego. Jest on odpowiedzialny za umieszczenie danych na wewnętrznej magistrali danych oraz podanie sygnałów sterujących zapisem danych do wybranego rejestru. Przykładowo wysterowanie sygnałów R0out i Ain powoduje umieszczenie informacji z R0 na magistrali, a następnie (w kolejnym kroku sterowania) załadowanie do rejestru A. Układ sterujący zarządza realizacją operacji zdefiniowanych jako instrukcje. Listę instrukcji zawiera Tabela 1. Notacja RX ← [RY] oznacza, że zawartość rejestru RY jest ładowana do

rejestru RX (instrukcja mv) . Wyrażenie RX ← D oznacza, że 16-bitowa stała jest ładowana do rejestru RX (instrukcja mvi).

Operation	Function performed
mv Rx,Ry	$Rx \leftarrow [Ry]$
mvi Rx,#D	$Rx \leftarrow D$
add Rx, Ry	$Rx \leftarrow [Rx] + [Ry]$
sub Rx, Ry	$Rx \leftarrow [Rx] - [Ry]$

Tabela 1 Instrukcje i realizowane funkcje

Instrukcja jest zapisywana w rejestrze instrukcji IR w 6 bitowym formacie IIXXYY, gdzie bity II koduj ą instrukcję, XX kodują rejestry RX a YY kodują rejestry RY. Instrukcje są pobierane z magistrali wejściowej układu. W przypadku instrukcji mvi pole YY nie ma znaczenia, natomiast kolejne (konieczna synchronizacja podawania danych wejściowych z działaniem układu sterującego) po instrukcji słowo podawane na magistralę wejściową zawiera wartość przeznaczoną do załadowania do rejestru.
Niektóre instrukcje wymagają dla swojej realizacji kilku stanów układu, przykładowo: konieczne jest realizowanie wielu przesłań danych dla sumowania. Układ sterujący korzysta z dwubitowego licznika dla określenia kolejnych kroków realizacji instrukcji.
Układ przechodzi do realizacji instrukcji podanej na magistrali wejściowej w odpowiedzi na wysterowanie sygnału wejściowego Run oraz wysteruje sygnał Done po ukończeniu realizacji rozkazu. Tablica 2 określa sygnały sterujące aktywne w poszczególnych krokach realizacji poszczególnych instrukcji. W kroku T0 jedynym aktywnym jest sygnał IRin.

	T_1	T_2	T_3
(mv): I_0	$RY_{out}, RX_{in},$ *Done*		
(mvi): I_1	$DIN_{out}, RX_{in},$ *Done*		
(add): I_2	RX_{out}, A_{in}	RY_{out}, G_{in}	$G_{out}, RX_{in},$ *Done*
(sub): I_3	RX_{out}, A_{in}	$RY_{out}, G_{in},$ *AddSub*	$G_{out}, RX_{in},$ *Done*

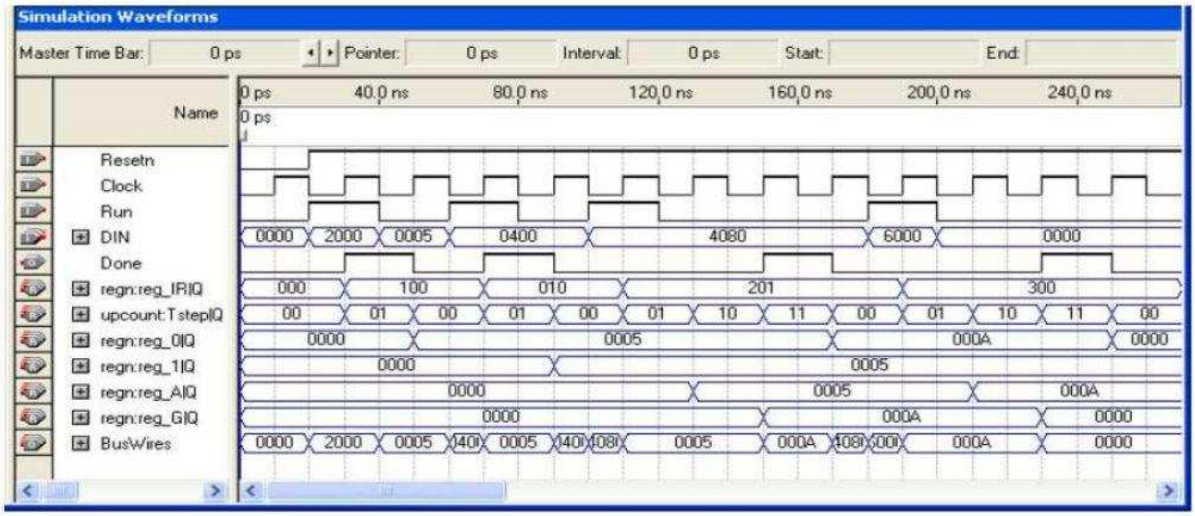
Tabela 2 Sygnały aktywne w poszczególnych krokach realizacji rozkazów (dla innych niż wersja 1 procesora konieczne modyfikacje).

Wersje projektów procesora:
Wersja 1 – według powyższej specyfikacji
Wersja 2 – dodatkowy rozkaz przepisania danych między rejestrem a akumulatora, operacje ADD i SUB dotyczą akumulatora jako pierwszego operandu, wynik jest zapisywany w akumulatorze.
Wersja 3 – wynik operacji arytmetycznych zostaje zapisany do rejestru REG3, brak rejestru G, jeden rozkaz arytmetyczny add-sub, niewykorzystane bity (brak konieczności kodowania rejestru docelowego) rozkazu arytmetycznego służą do określenia czy mamy do czynienia z dodawaniem czy odejmowaniem.

Wersja 4 – procesor posiada rozkaz skoku bezwarunkowego, w tym celu układ jest rozszerzony o licznik adresowy określający adres następnego do pobrania rozkazu, do licznika można przepisać wartość z rejestru. Dodatkowy rozkaz jest kodowany na 3 bicie kodującym rozkaz (bit ten nie jest wykorzystany ze względu na 4 rejestry). Do rejestru adresowego należy zapisać wartość o jeden cykl procesora wcześniej przed pobraniem kolejnego rozkazu.
Wersja 5 – procesor może realizować operację zapisu do pamięci wartości pobranej z rejestru pod adres pobrany z innego rejestru. Dodatkowy rozkaz jest kodowany na 3 bicie kodującym rozkaz (bit ten nie jest wykorzystany ze względu na 4 rejestry). W celu realizacji zapisu należy użyć dodatkowy sygnał /MW – memory write aktywny poziomem niskim przez jeden cykl procesora.

Zadania do realizacji w ramach ćwiczenia:

1. Przygotuj plik VHDL określający działanie opisanego powyżej układu. Kody pomocnicze do wykorzystania jako elementy składowe projektu przedstawiają rysunki 3a, 3b.
2. Przeprowadź symulację funkcjonalną dla sprawdzenia poprawności kodu. Zaprezentuj realizację każdego typu rozkazu. Przykładowe przebiegi umieszczono na Rysunku 2.
3. Utwórz projekt dla implementacji układu w płycie DE2. Użyj przełączników SW15-0 do wysterowania wejść danych. Sygnały Run, Resetn, Clock i Done podłącz odpowiednio do SW17, KEY0, KEY1, LEDR17. Stan magistrali wewnętrznej procesora obserwuj za pomocą LEDR15-0, za pomocą diód obserwuj inne kluczowe sygnały układu. Przetestuj krokowo działanie procesora określając sygnały wejściowe po załadowaniu do układu FPGA płyty DE2.



Rysunek 2 Symulacja pracy układu procesora.

```
LIBRARY ieee; USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;

    ENTITY proc IS
      PORT (  DIN                 : IN      STD_LOGIC_VECTOR(15 DOWNTO 0);
              Resetn, Clock, Run  : IN      STD_LOGIC;
              Done                : BUFFER  STD_LOGIC;
              BusWires            : BUFFER  STD_LOGIC_VECTOR(15 DOWNTO 0));
```

```
        END proc;

        ARCHITECTURE Behavior OF proc IS
          . . . declare components
          . . . declare signals
        BEGIN
          High <= '1';
          Clear <= . . .

          Tstep: upcount PORT MAP (Clear, Clock, Tstep_Q);
          I <= IR(1 TO 3);
          decX: dec3to8 PORT MAP (IR(4 TO 6), High, Xreg);
          decY: dec3to8 PORT MAP (IR(7 TO 9), High, Yreg);
controlsignals: PROCESS (Tstep_Q, I, Xreg, Yreg)
  BEGIN
    . . . specify initial values
    CASE Tstep_Q IS
      WHEN "00" => - - store DIN in IR as long as Tstep_Q = 0
        IRin <= '1';
      WHEN "01" => - - define signals in time step T1
        CASE I IS
          . . .
        END CASE;
      WHEN "10" => - - define signals in time step T2
        CASE I IS
          . . .
        END CASE;
      WHEN "11" => - - define signals in time step T3
        CASE I IS
          . . .
        END CASE;
    END CASE;
  END PROCESS;

  reg_0: regn PORT MAP (BusWires, Rin(0), Clock, R0);
  . . . instantiate other registers and the adder/subtracter unit
  . . . define the bus
END Behavior;
```

Rys 3a Podstawowe elementy opisu jednostki projektowej procesora.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;

ENTITY upcount IS
  PORT (  Clear, Clock       : IN     STD_LOGIC;
          Q                  : OUT    STD_LOGIC_VECTOR(1 DOWNTO 0));
```

```
END upcount;

ARCHITECTURE Behavior OF upcount IS
  SIGNAL Count : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
  PROCESS (Clock)
  BEGIN
    IF (Clock'EVENT AND Clock = '1') THEN
      IF Clear = '1' THEN
        Count <= "00";
      ELSE
        Count <= Count + 1;
      END IF;
    END IF;
  END PROCESS;
  Q <= Count;
END Behavior;
ARCHITECTURE Behavior OF dec3to8 IS
BEGIN
  PROCESS (W, En)
  BEGIN
    IF En = '1' THEN
      CASE W IS
        WHEN "000" => Y <= "10000000";
        WHEN "001" => Y <= "01000000";
        WHEN "010" => Y <= "00100000";
        WHEN "011" => Y <= "00010000";
        WHEN "100" => Y <= "00001000";
        WHEN "101" => Y <= "00000100";
        WHEN "110" => Y <= "00000010";
        WHEN "111" => Y <= "00000001";
      END CASE;
    ELSE
      Y <= "00000000";
    END IF;
  END PROCESS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY regn IS
  GENERIC (n : INTEGER := 16);
  PORT (  R          : IN       STD_LOGIC_VECTOR(n-1 DOWNTO 0);
        Rin, Clock   : IN       STD_LOGIC;
        Q          : BUFFER    STD_LOGIC_VECTOR(n-1 DOWNTO 0));
END regn;
```

```
ARCHITECTURE Behavior OF regn IS
BEGIN
  PROCESS (Clock)
  BEGIN
    IF Clock'EVENT AND Clock = '1' THEN
      IF Rin = '1' THEN
        Q <= R;
      END IF;
    END IF;
  END PROCESS;
END Behavior;
```

Rys. 3b Jednostki licznika, selektora i rejestru dla układu procesora.