Architektura systemów komputerowych

Mariusz Wiśniewski

Politechnika Świętokrzyska w Kielcach
Katedra Informatyki

Ścieżka danych

Plan wykładu

1. Ścieżka danych – wstęp.
2. Ścieżki danych dla instrukcji różnych typów.
3. Projekt μP – ścieżka danych.
4. Synteza ścieżki danych.

Cele

Znajomość architektury oraz technik projektowania mikroprocesorów. Poznanie pojęć: ścieżki danych i kontrolera.

Ścieżka danych – wstęp

Ścieżka danych

podstawowe wiadomości na temat ścieżki danych

Procesor składa się z wielu autonomicznych układów. Jednymi z nich są moduły biorące bezpośredni udział w przetwarzaniu danych.

Pojęcia:
- ścieżka danych
- jednostki funkcjonalne
- szyna systemowa

Ścieżka danych

podstawowe wiadomości na temat ścieżki danych

- wstęp
- moduły

Procesor składa się z wielu autonomicznych układów. Jednymi z nich są moduły biorące bezpośredni udział w przetwarzaniu danych.

Pojęcia:
- ścieżka danych
- jednostki funkcjonalne
- szyna systemowa

Przetwarzanie danych wymaga zastosowania jednostek obliczeniowych i pamiętających, takich jak:
- jednostki arytmetyczne i logiczne,
- mnożarki / dzielarki,
- rejestry i bufory danych,

które łącznie stanowią ścieżkę danych. Przepływ danych w ścieżce organizuje kontroler.

Ścieżka danych

podstawowe wiadomości na temat ścieżki danych

– wstęp

– moduły

Procesor składa się z wielu autonomicznych układów. Jednymi z nich są moduły biorące bezpośredni udział w przetwarzaniu danych.

Pojęcia:
- ścieżka danych
- jednostki funkcjonalne
- szyna systemowa

Przetwarzanie danych wymaga zastosowania jednostek obliczeniowych i pamiętających, takich jak:
- jednostki arytmetyczne i logiczne,
- mnożarki / dzielarki,
- rejestry i bufory danych,

które łącznie stanowią ścieżkę danych. Przepływ danych w ścieżce organizuje kontroler.

W procesorze do ścieżki danych zalicza się również następujące bloki funkcjonalne:
- rejestr rozkazów (IR),
- licznik programu (PC),
- rejestr adresowy pamięci (MAR),
- rejestr danych pamięci (MBR).

Zależnie od stopnia złożoności architektury CPU konstrukcyjnie powyższe jednostki mogą zostać wykonane w postaci pojedynczego rejestru lub bloku funkcjonalnego, posiadającego znacznie bardziej rozbudowaną mechanikę pracy.

M.Wiśniewski

Ścieżka danych

podstawowe wiadomości na temat ścieżki danych

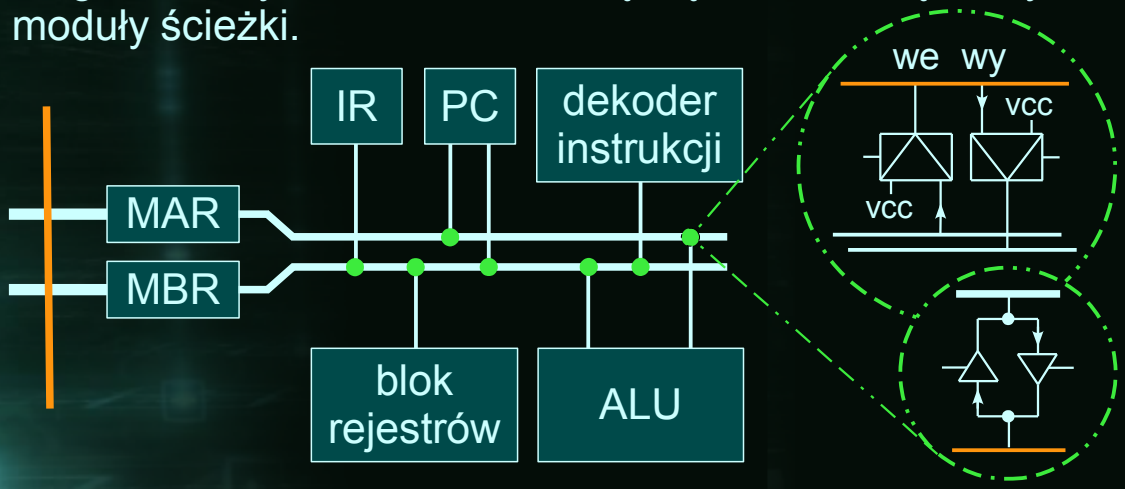
– wstęp

– moduły

Procesor składa się z wielu autonomicznych układów. Jednymi z nich są moduły biorące bezpośredni udział w przetwarzaniu danych.

Pojęcia:
- ścieżka danych
- jednostki funkcjonalne
- szyna systemowa

Elementarną jednostką konstrukcyjną ścieżki danych jest magistrala systemowa CPU, łącząca ze sobą wszystkie moduły ścieżki.



Magistrala systemowa wykorzystywana jest:
- do transferów danych między rejestrami CPU,
- w operacjach arytmetycznych i logicznych,
- podczas operacji odczytu danych z pamięci RAM,
- w czasie zapisu danych do pamięci RAM.

Przetwarzanie danych wymaga zastosowania jednostek obliczeniowych i pamiętających, takich jak:
- jednostki arytmetyczne i logiczne,
- mnożarki / dzielarki,
- rejestry i bufory danych,

które łącznie stanowią ścieżkę danych. Przepływ danych w ścieżce organizuje kontroler.

W procesorze do ścieżki danych zalicza się również następujące bloki funkcjonalne:
- rejestr rozkazów (IR),
- licznik programu (PC),
- rejestr adresowy pamięci (MAR),
- rejestr danych pamięci (MBR).

Zależnie od stopnia złożoności architektury CPU konstrukcyjnie powyższe jednostki mogą zostać wykonane w postaci pojedynczego rejestru lub bloku funkcjonalnego, posiadającego znacznie bardziej rozbudowaną mechanikę pracy.

Ścieżka danych

podstawowe wiadomości na temat ścieżki danych

- wstęp
- moduły

W przypadku procesora projekt ścieżki danych rozpoczyna się od jej najbardziej podstawowego elementu. W miarę postępowania prac projektowych w ścieżce danych znajdzie się coraz więcej modułów.

Moduły:
- rejestru PC
- rejestru IR

Ścieżka danych

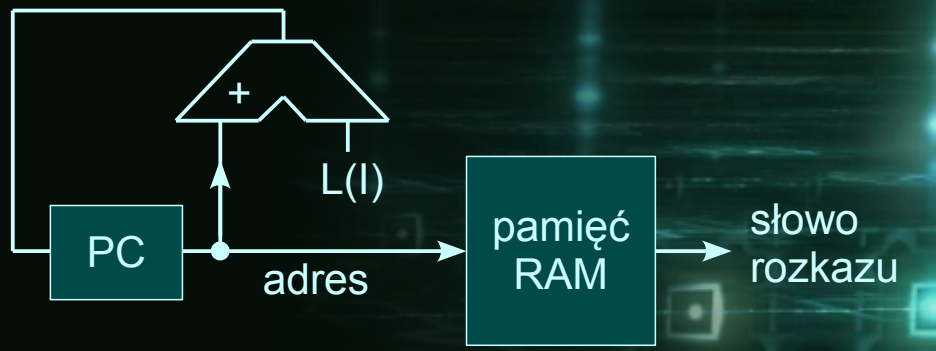
podstawowe wiadomości na temat ścieżki danych

- wstęp
- moduły

W przypadku procesora projekt ścieżki danych rozpoczyna się od jej najbardziej podstawowego elementu. W miarę postępowania prac projektowych w ścieżce danych znajdzie się coraz więcej modułów.

Moduły:
- rejestru PC
- rejestru IR

Rejestr PC wskazuje miejsce w pamięci, z którego zostanie pobrany następny rozkaz po wykonaniu bieżącego. Rejestr będzie podlegał zmianom w następujących przypadkach:
- po wykonaniu „zwykłej” instrukcji,
- po wykonaniu instrukcji modyfikującej kolejność wykonania rozkazów.



Wartości L(I) oznaczają liczbę bajtów (słów) jakie należy dodać do bieżącej wartości PC, aby po wykonaniu instrukcji możliwe było pobranie następnego rozkazu.

M.Wiśniewski

Ścieżka danych

podstawowe wiadomości na temat ścieżki danych

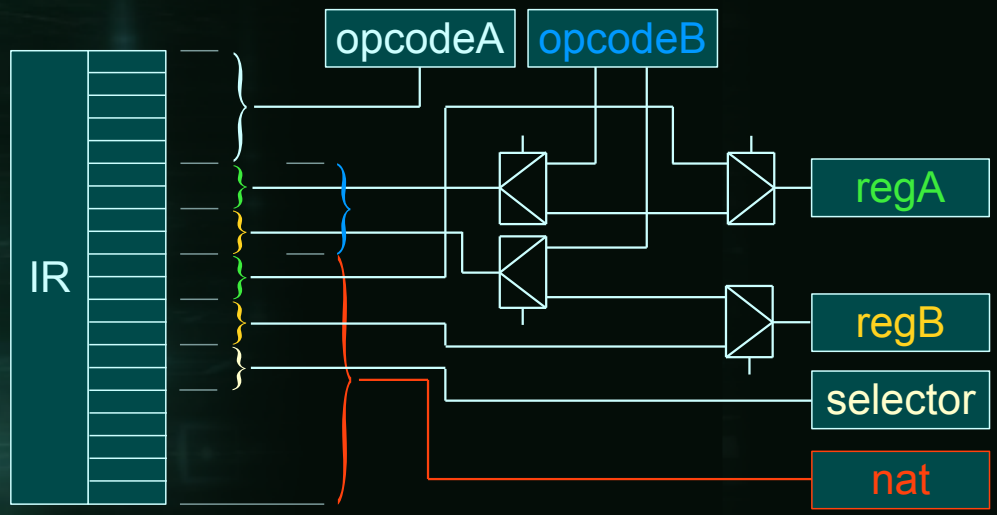
– wstęp

– moduły

W przypadku procesora projekt ścieżki danych rozpoczyna się od jej najbardziej podstawowego elementu. W miarę postępowania prac projektowych w ścieżce danych znajdzie się coraz więcej modułów.

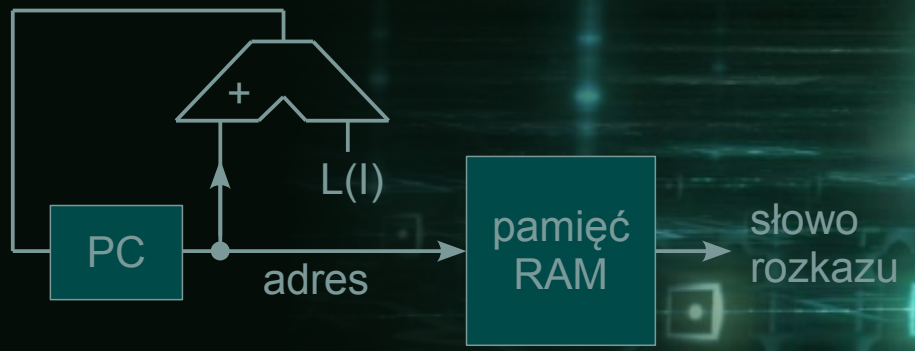
Moduły:
- rejestru PC
- rejestru IR

Rejestr IR przechowuje pobrany rozkaz. W praktyce stanowi zbiór rejestrów adresowanych bitowo.



Zależnie od modelu rozkazu, wartość natychmiastowa może być przekazana jako osobne słowo. W takim przypadku obecność dedykowanego rejestru na takie dane jest opcjonalna.

Rejestr PC wskazuje miejsce w pamięci, z którego zostanie pobrany następny rozkaz po wykonaniu bieżącego. Rejestr będzie podlegał zmianom w następujących przypadkach:
- po wykonaniu „zwykłej” instrukcji,
- po wykonaniu instrukcji modyfikującej kolejność wykonania rozkazów.



Wartości L(I) oznaczają liczbę bajtów (słów) jakie należy dodać do bieżącej wartości PC, aby po wykonaniu instrukcji możliwe było pobranie następnego rozkazu.

Ścieżki danych dla instrukcji różnych typów

Ścieżki danych instrukcji

przykładowa ścieżka danych dla R-instrukcji

- R-instrukcja
- I-instrukcja
- C-instrukcja

Operandami R-instrukcji są rejestry. Zazwyczaj niemal wszystkie tego typu rozkazy mają wspólną ścieżkę danych.

Pojęcia:
- blok rejestrów
- konstrukcja

M.Wiśniewski

Ścieżki danych instrukcji

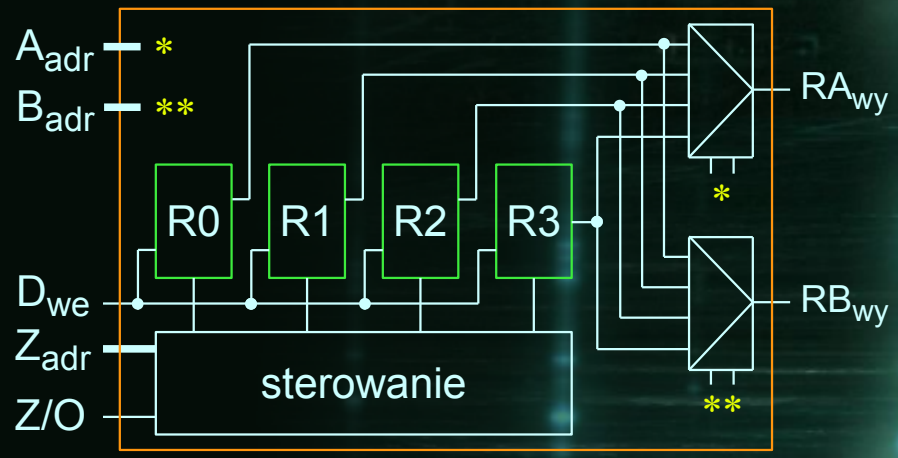
przykładowa ścieżka danych dla R-instrukcji

- R-instrukcja
- I-instrukcja
- C-instrukcja

Operandami R-instrukcji są rejestry. Zazwyczaj niemal wszystkie tego typu rozkazy mają wspólną ścieżkę danych.

Pojęcia:
- blok rejestrów
- konstrukcja

Jednym z podstawowych elementów ścieżki danych jest blok rejestrów (ang. register file).



Powyższy blok rejestrów wykonuje:
- zapis do jednego rejestr,
- odczyt jednego z rejestrów,
- odczyt z dwóch rejestrów.

Blok rejestrów może składać się z bardziej złożonych jednostek, które mogą wykonać inne operacje na swojej zawartości, np. funkcję logiczną lub arytmetyczną z danymi wejściowymi lub inkrementację / dekrementację.

M.Wiśniewski

Ścieżki danych instrukcji

przykładowa ścieżka danych dla R-instrukcji

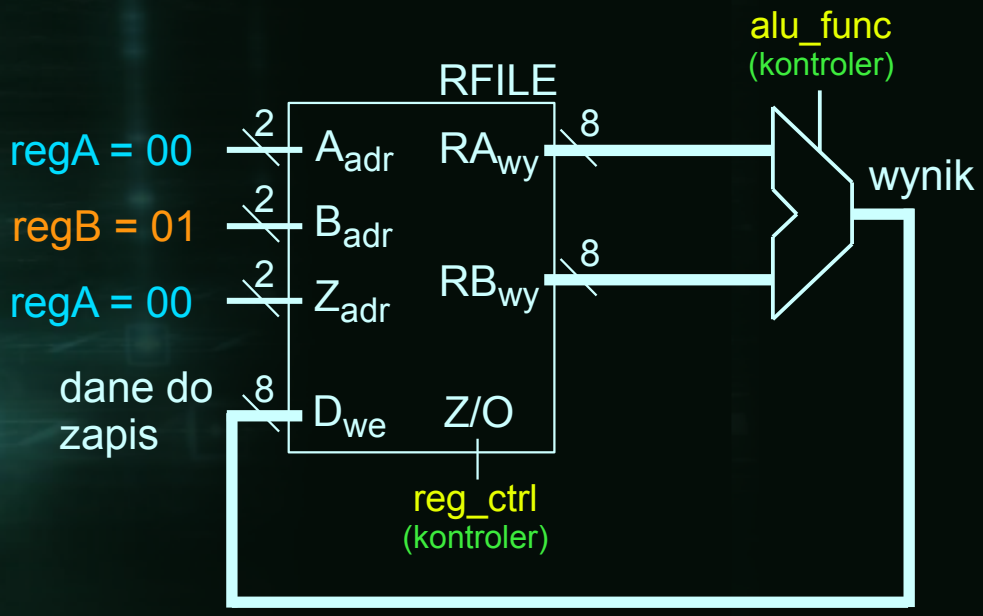
- R-instrukcja
- I-instrukcja
- C-instrukcja

Operandami R-instrukcji są rejestry. Zazwyczaj niemal wszystkie tego typu rozkazy mają wspólną ścieżkę danych.

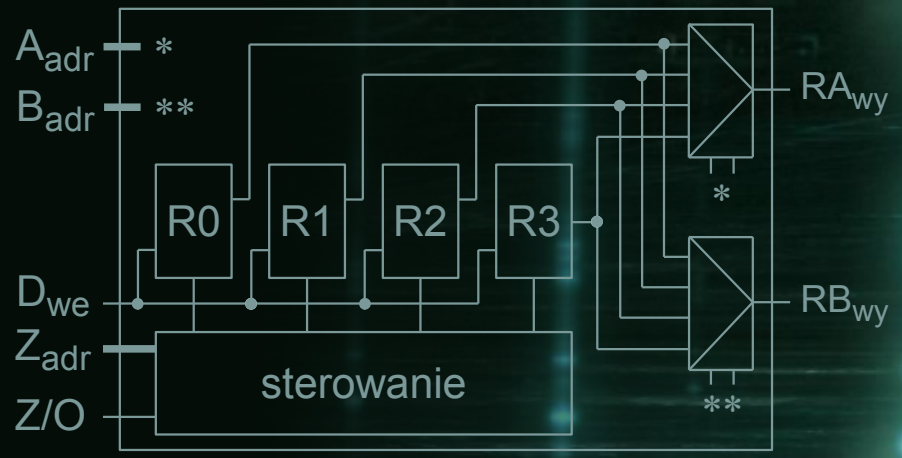
Pojęcia:
- blok rejestrów
- konstrukcja

Rozkaz:

```
ccc cccc- rr -- rr --
001 01010 00 11 01 00    - np. R=R+R
```



Jednym z podstawowych elementów ścieżki danych jest blok rejestrów (ang. register file).



Powyższy blok rejestrów wykonuje:
- zapis do jednego rejestr,
- odczyt jednego z rejestrów,
- odczyt z dwóch rejestrów.

Blok rejestrów może składać się z bardziej złożonych jednostek, które mogą wykonać inne operacje na swojej zawartości, np. funkcję logiczną lub arytmetyczną z danymi wejściowymi lub inkrementację / dekrementację.

Ścieżki danych instrukcji

przykładowa ścieżka danych dla I-instrukcji

- R-instrukcja
- I-instrukcja
- C-instrukcja

Wartość natychmiastowa w instrukcjach może posłużyć jako argument lub część adresu. Wtedy rejestry mogą brać udział w adresowaniu pamięci.

Instrukcje:
- odczyt danych
- zapis danych

M.Wiśniewski

Ścieżki danych instrukcji

przykładowa ścieżka danych dla I-instrukcji

- R-instrukcja
- I-instrukcja
- C-instrukcja

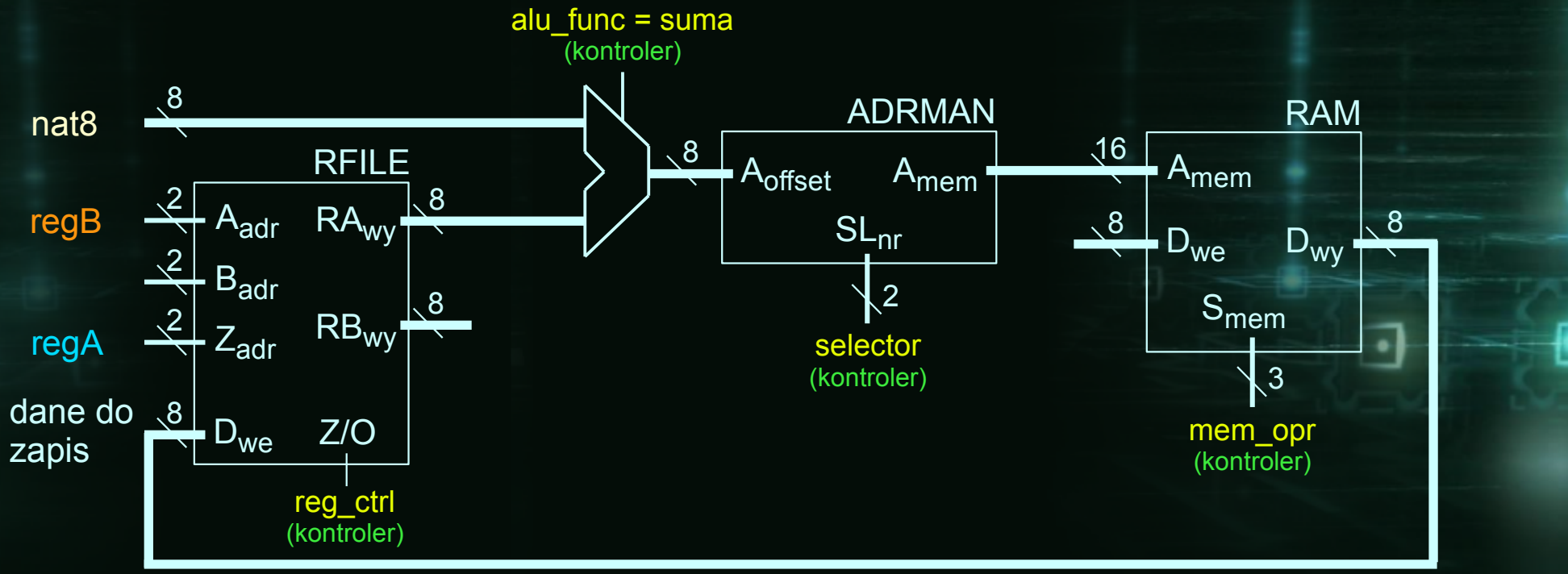
Wartość natychmiastowa w instrukcjach może posłużyć jako argument lub część adresu. Wtedy rejestry mogą brać udział w adresowaniu pamięci.

Instrukcje:
- odczyt danych
- zapis danych

Rozkaz:

```
ccc cccc- rr sl rr --  (nat8)
011 01110 10 01 00 00 10110001
```

- np. R=sl:[R+nat]



M.Wiśniewski

Ścieżki danych instrukcji

przykładowa ścieżka danych dla I-instrukcji

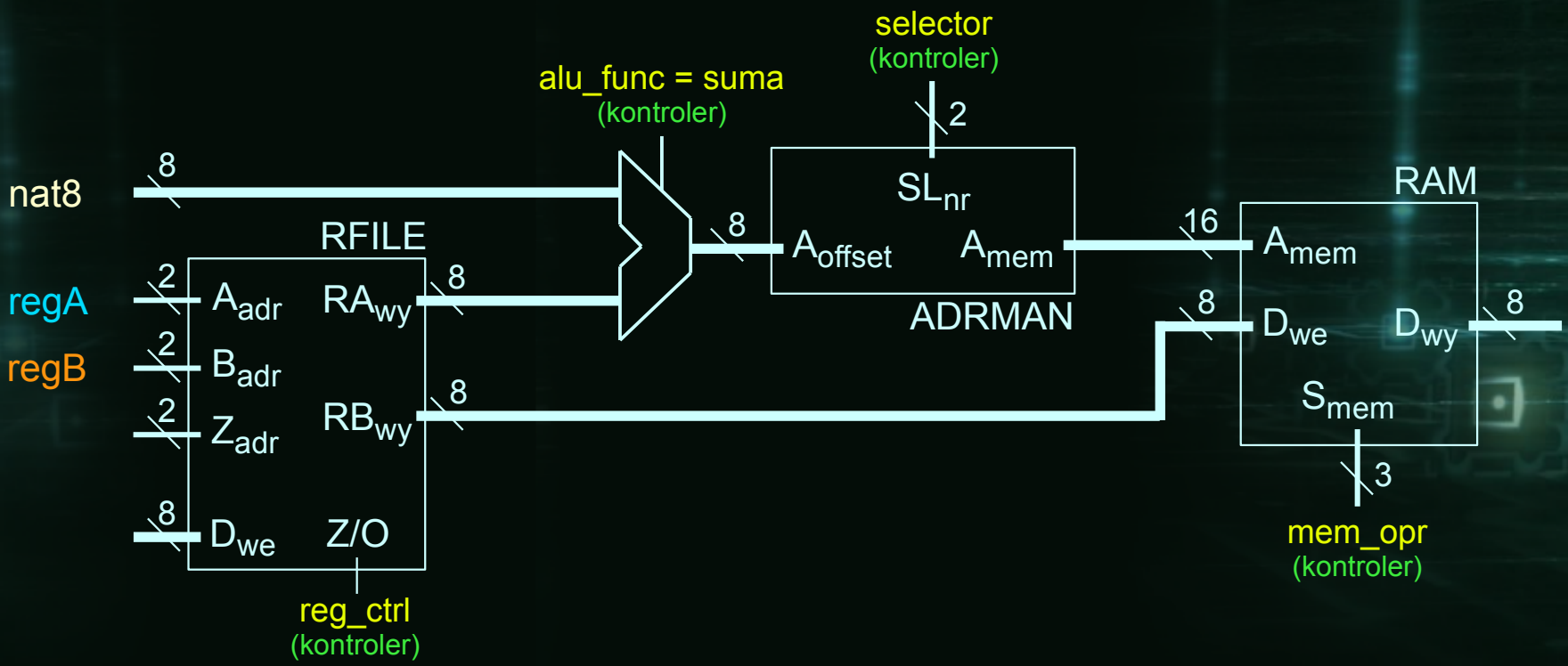
- R-instrukcja
- I-instrukcja
- C-instrukcja

Wartość natychmiastowa w instrukcjach może posłużyć jako argument lub część adresu. Wtedy rejestry mogą brać udział w adresowaniu pamięci.

Instrukcje:
- odczyt danych
- zapis danych

Rozkaz:

```
cc sl rr rr (nat8)
10 10 10 00 01110011   - np. sl:[R+nat]=R
```



Ścieżki danych instrukcji

przykładowa ścieżka danych dla C-instrukcji

- R-instrukcja
- I-instrukcja
- C-instrukcja

Rozkazy zmieniające porządek wykonania instrukcji występuję w dwóch wariantach: wewnątrz segmentowe (bliskie) oraz dalekie.

Instrukcje:
- skok bliski
- skok daleki

Ścieżki danych instrukcji

przykładowa ścieżka danych dla C-instrukcji

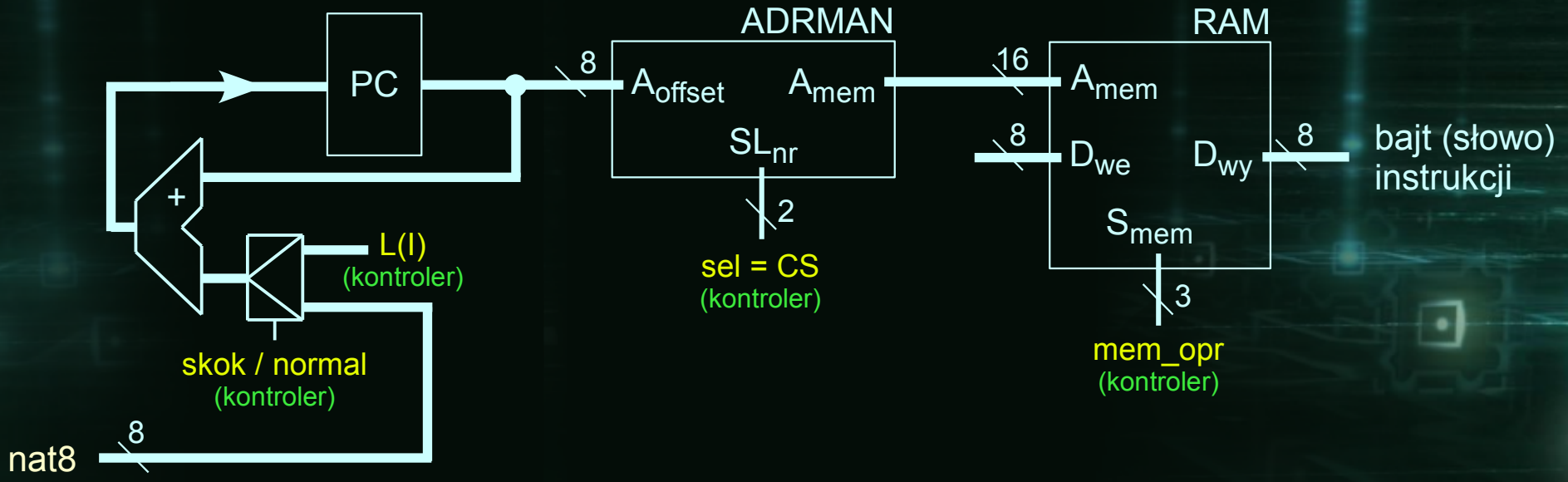
- R-instrukcja
- I-instrukcja
- C-instrukcja

Rozkazy zmieniające porządek wykonania instrukcji występuję w dwóch wariantach: wewnątrz segmentowe (bliskie) oraz dalekie.

Instrukcje:
- skok bliski
- skok daleki

Rozkaz:

```
cc -- cccc   (nat8)
11 00 1100 00001111      - jge PC+15
```



Ścieżki danych instrukcji

przykładowa ścieżka danych dla C-instrukcji

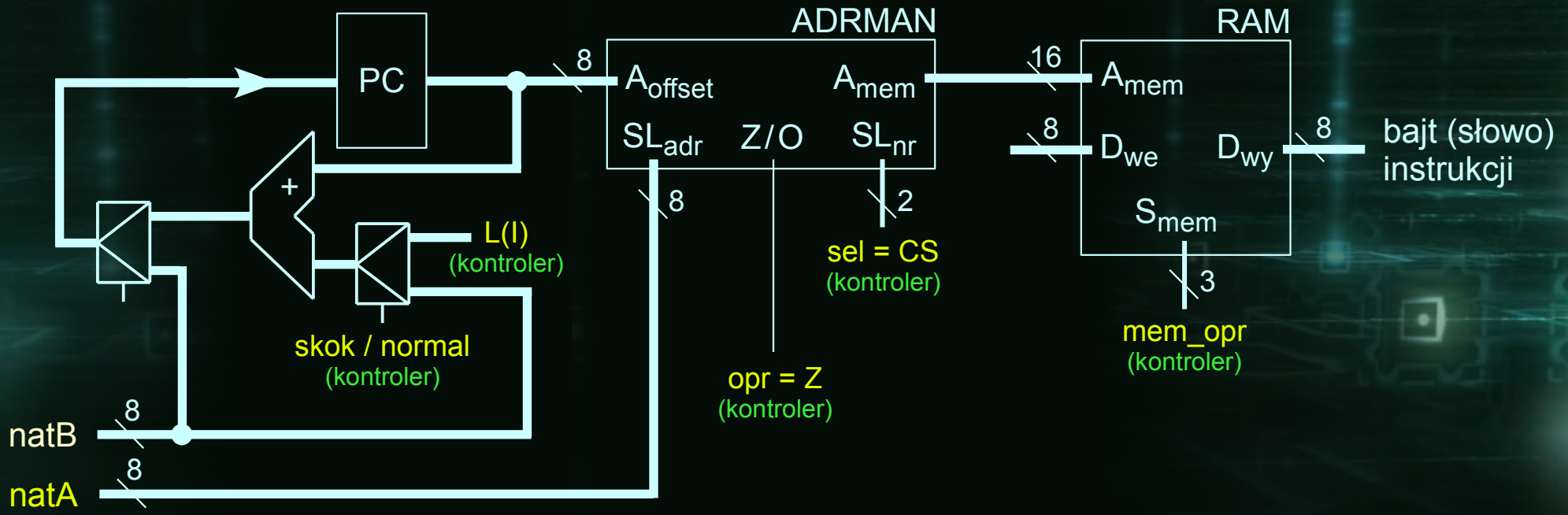
- R-instrukcja
- I-instrukcja
- C-instrukcja

Rozkazy zmieniające porządek wykonania instrukcji występuję w dwóch wariantach: wewnątrz segmentowe (bliskie) oraz dalekie.

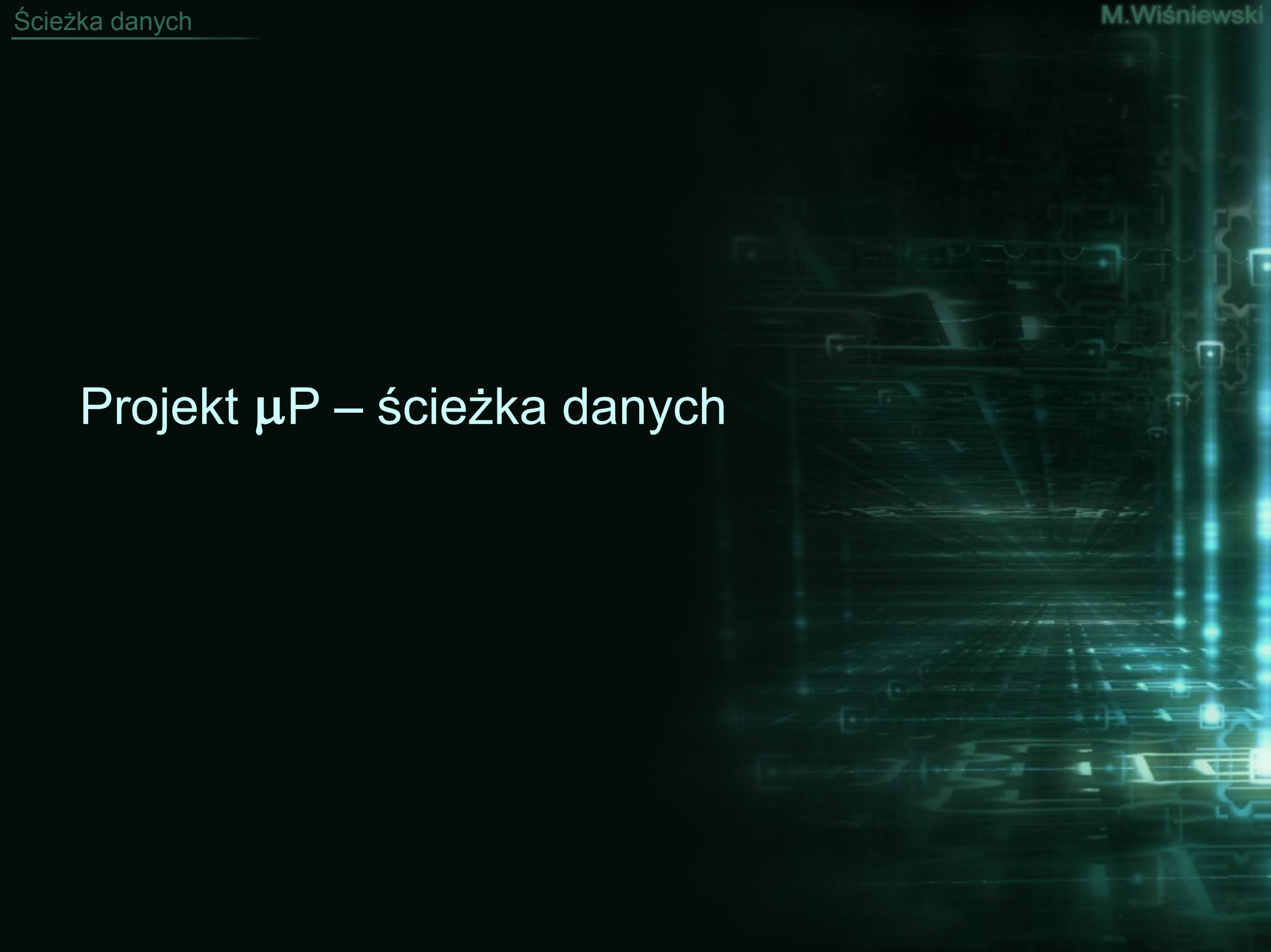
Instrukcje:
- skok bliski
- skok daleki

Rozkaz:

```
    ccc mcc cc   (natA)    (natB)
    000 111 11 00001111 11011100     - jmp far sel:ofs
```



Projekt μP – ścieżka danych



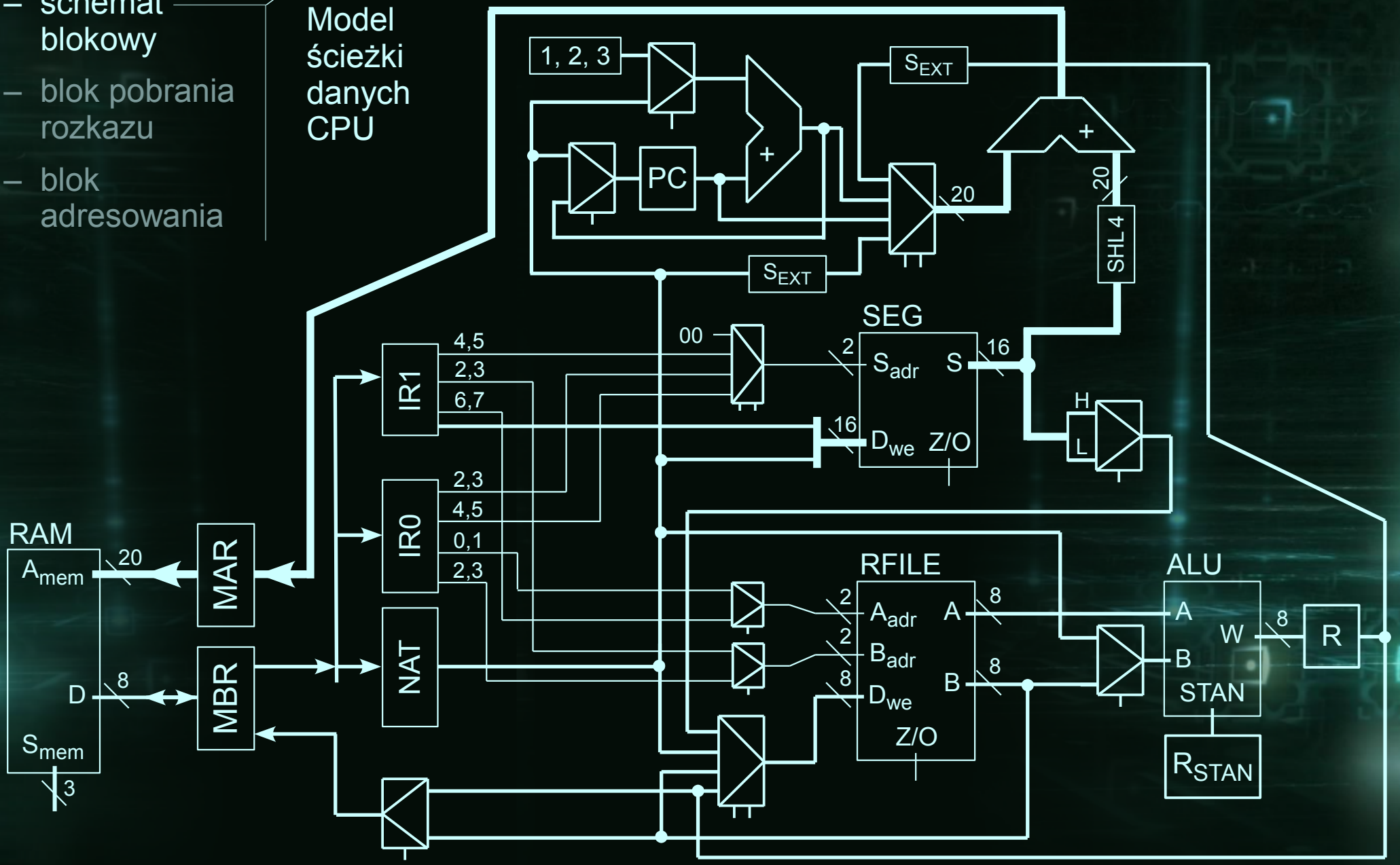
M.Wiśniewski

Projekt μP – ścieżka danych

projekt ścieżki danych CPU dla założeń z wykładu nr 4

- schemat blokowy
- blok pobrania rozkazu
- blok adresowania

Model ścieżki danych CPU



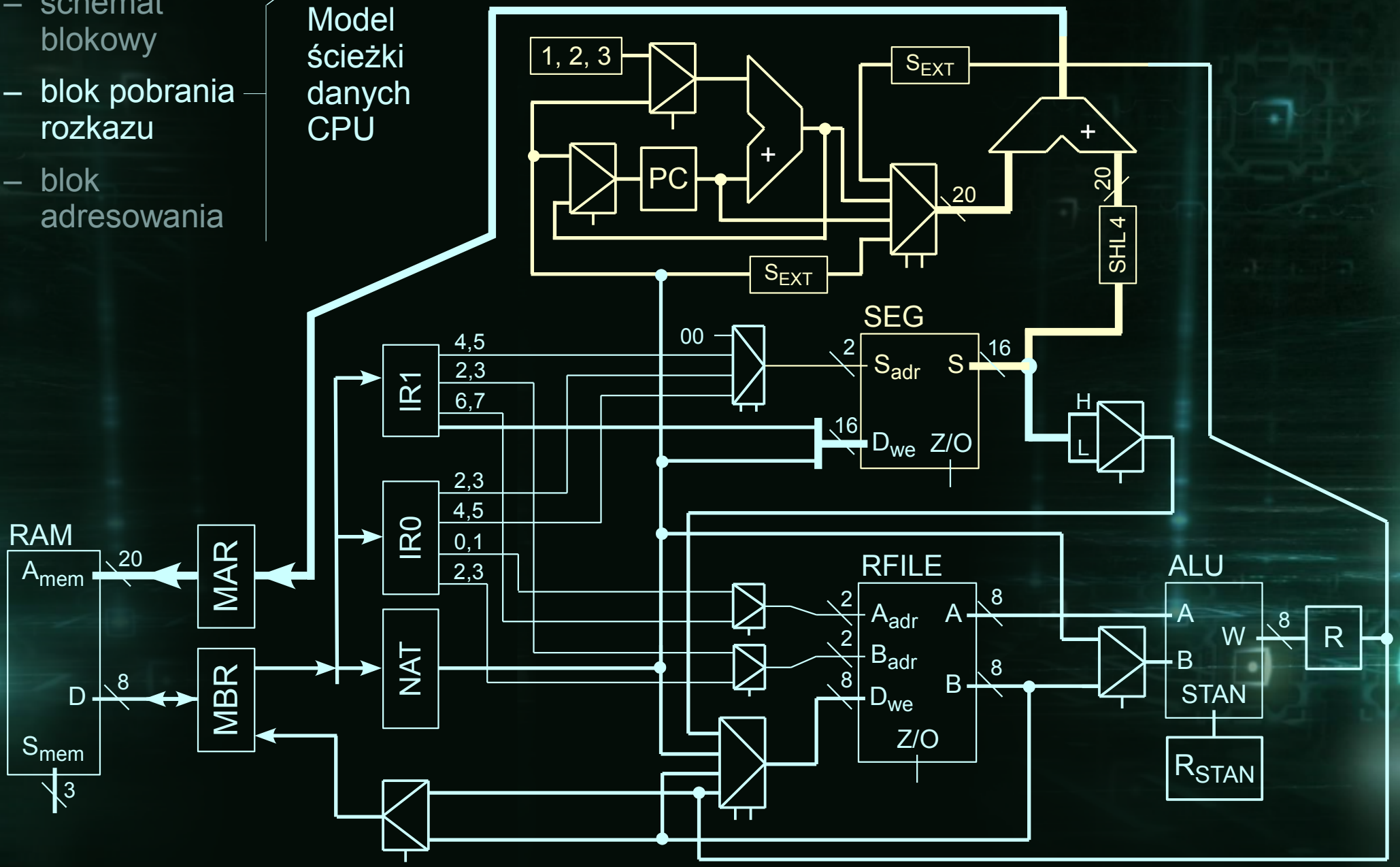
M.Wiśniewski

Projekt μP – ścieżka danych

projekt ścieżki danych CPU dla założeń z wykładu nr 4

- schemat blokowy
- blok pobrania rozkazu
- blok adresowania

Model ścieżki danych CPU



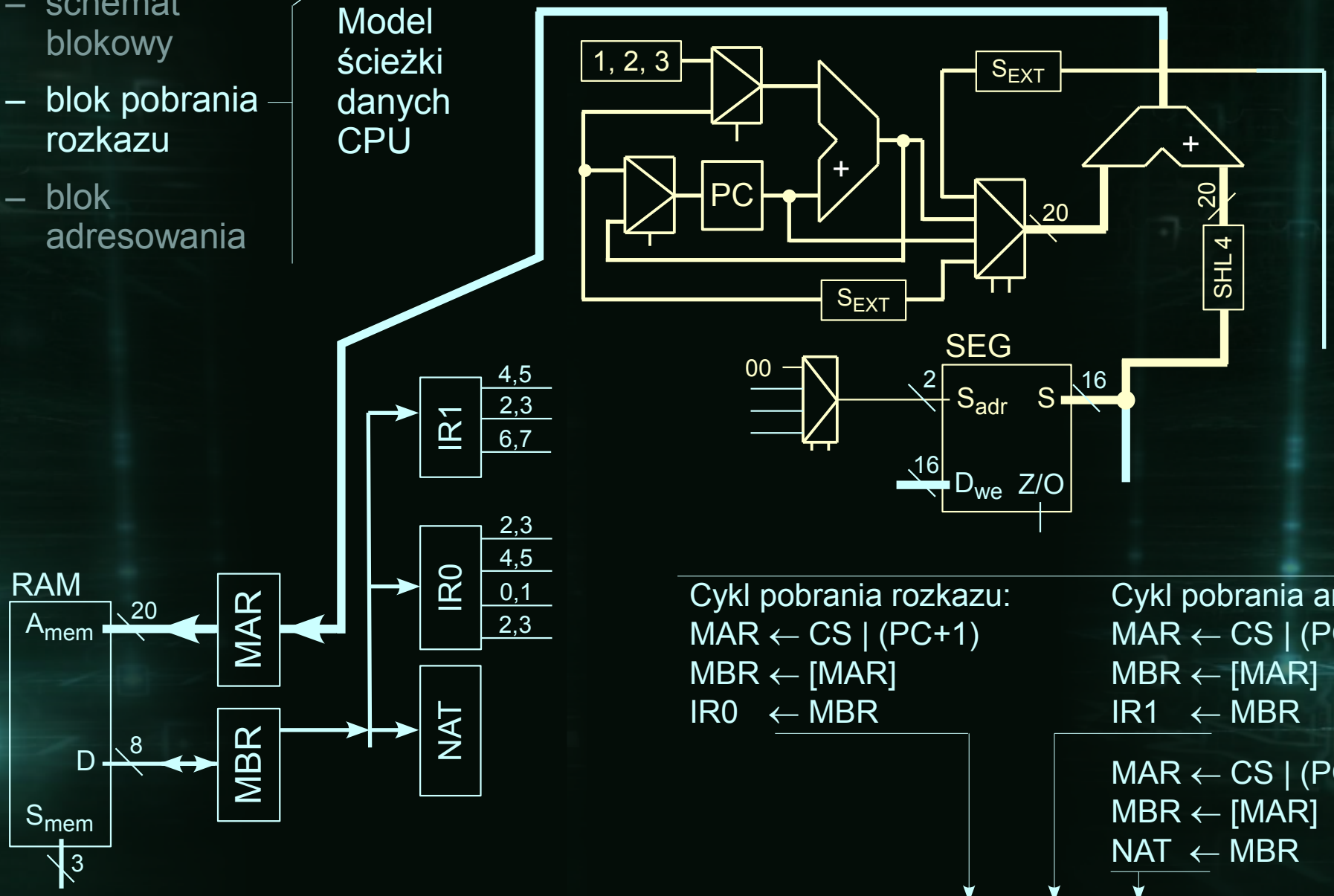
M.Wiśniewski

Projekt μP – ścieżka danych

projekt ścieżki danych CPU dla założeń z wykładu nr 4

- schemat blokowy
- blok pobrania rozkazu
- blok adresowania

Model ścieżki danych CPU



Cykl pobrania rozkazu:
MAR ← CS | (PC+1)
MBR ← [MAR]
IR0 ← MBR

Cykl pobrania argumentu:
MAR ← CS | (PC+2)
MBR ← [MAR]
IR1 ← MBR

MAR ← CS | (PC+3)
MBR ← [MAR]
NAT ← MBR

PC ← PC + (1, 2 lub 3)

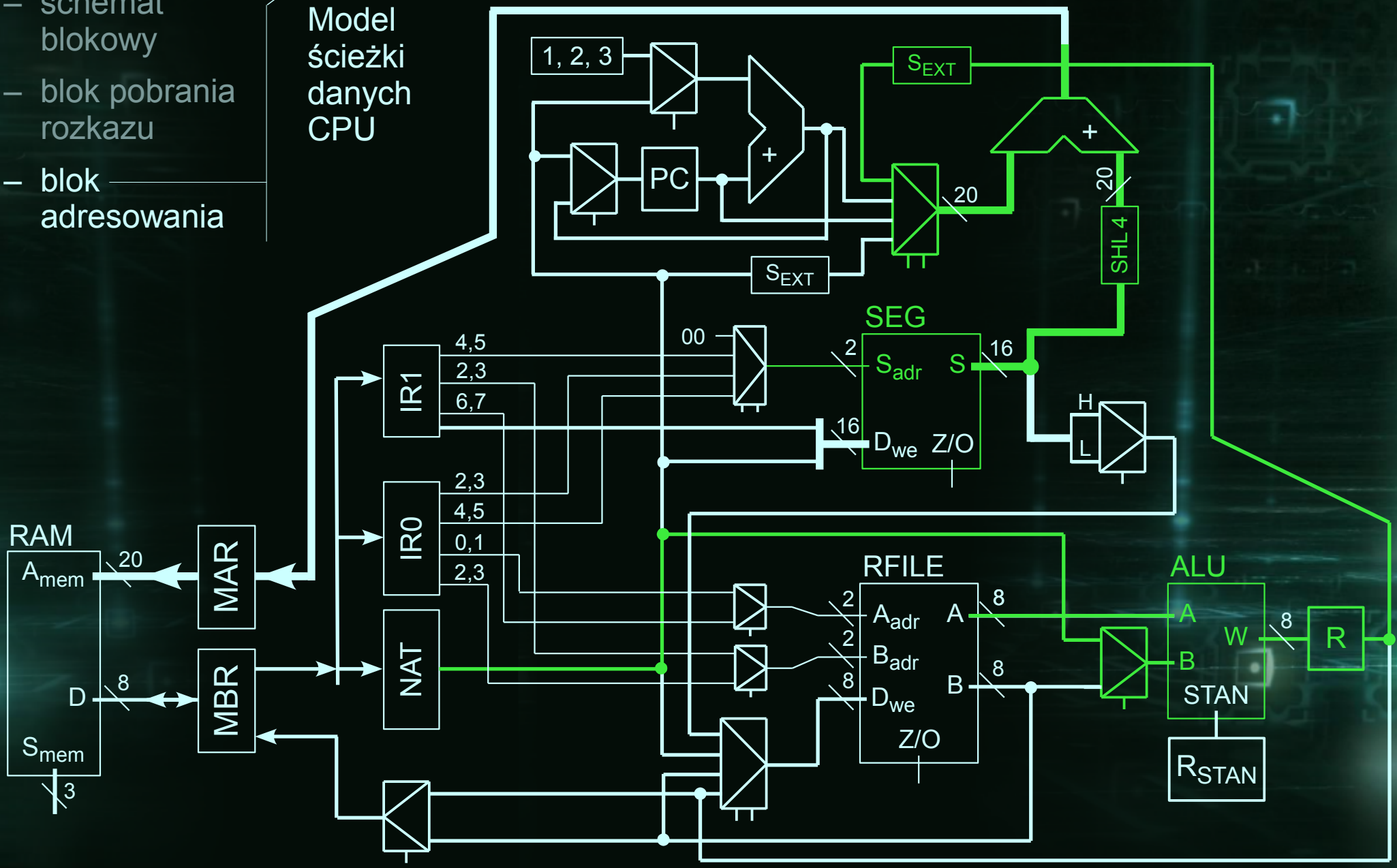
M.Wiśniewski

Projekt μP – ścieżka danych

projekt ścieżki danych CPU dla założeń z wykładu nr 4

- schemat blokowy
- blok pobrania rozkazu
- blok adresowania

Model ścieżki danych CPU



Synteza ścieżki danych

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły procesora są układami logicznymi o dużej złożoność. Z tego powodu do ich syntezy wykorzystuje się programy CAD oraz języki opisu sprzętu.

Pojęcia:
- programy CAD
- jakość syntezy
- języki HDL

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły procesora są układami logicznymi o dużej złożoność. Z tego powodu do ich syntezy wykorzystuje się programy CAD oraz języki opisu sprzętu.

Pojęcia:
- programy CAD
- jakość syntezy
- języki HDL

Aplikacje CAD (Computer Aid Design) stanowią oprogramowanie wspomagające projektowanie metodami komputerowymi – w przypadku syntezy układów cyfrowych generują strukturę logiczną na podstawie opisu graficznego lub słownego. Przykładami programów CAD są:
- Altera MAX+PLUS II, Altera Quartus II,
- Aldec Active-HDL,
- produkty firmy Mentor Graphics,
- produkty firmy XILINX.

Obecnie większość systemów EDA (Electornic Design Automation) umożliwia korzystanie z modów IP.

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły procesora są układami logicznymi o dużej złożoność. Z tego powodu do ich syntezy wykorzystuje się programy CAD oraz języki opisu sprzętu.

Pojęcia:
- programy CAD
- jakość syntezy
- języki HDL

Aplikacje CAD (Computer Aid Design) stanowią oprogramowanie wspomagające projektowanie metodami komputerowymi – w przypadku syntezy układów cyfrowych generują strukturę logiczną na podstawie opisu graficznego lub słownego. Przykładami programów CAD są:
- Altera MAX+PLUS II, Altera Quartus II,
- Aldec Active-HDL,
- produkty firmy Mentor Graphics,
- produkty firmy XILINX.

Obecnie większość systemów EDA (Electornic Design Automation) umożliwia korzystanie z modów IP.

Obecnie programy EDA stosują wyrafinowane metody syntezy układów logicznych. W przypadku translacji języków HDL stosują jednak metody uniwersalne, które nie zawsze generują najlepsze wyniki, ale są akceptowalne dla krótkich serii aplikacji lub wykonywanych na potrzeby szybkiego prototypowania.

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

– wstęp

– język VHDL

– przykłady modułów

Moduły procesora są układami logicznymi o dużej złożoność. Z tego powodu do ich syntezy wykorzystuje się programy CAD oraz języki opisu sprzętu.

Pojęcia:
- programy CAD
- jakość syntezy
- języki HDL

W syntezie złożonych układów cyfrowych wykorzystuje się języki opisu sprzętu HDL. Powszechnie stosuje się:
- VHDL (VHSIC HDL),
- Verilog,
- AHDL (Altera HDL),
- SystemC.

Opis układu w języku HDL przypomina składniowo programowanie. Różnice dotyczą interpretacji kodu, który w języku HDL odpowiada często równolegle działającym modułom rzeczywistego układu.

Poza syntezą układów, systemy EDA umożliwiają stosowanie różnych metod optymalizacji, np. minimalizację funkcji logicznych, kodowanie stanów, upraszczanie układów, stosowanie modułów IP (prefabrykatów), testowanie gotowych układów, itp..

Dużą popularnością wśród języków HDL cieszy się VHDL.

Aplikacje CAD (Computer Aid Design) stanowią oprogramowanie wspomagające projektowanie metodami komputerowymi – w przypadku syntezy układów cyfrowych generują strukturę logiczną na podstawie opisu graficznego lub słownego. Przykładami programów CAD są:
- Altera MAX+PLUS II, Altera Quartus II,
- Aldec Active-HDL,
- produkty firmy Mentor Graphics,
- produkty firmy XILINX.

Obecnie większość systemów EDA (Electornic Design Automation) umożliwia korzystanie z modów IP.

Obecnie programy EDA stosują wyrafinowane metody syntezy układów logicznych. W przypadku translacji języków HDL stosują jednak metody uniwersalne, które nie zawsze generują najlepsze wyniki, ale są akceptowalne dla krótkich serii aplikacji lub wykonywanych na potrzeby szybkiego prototypowania.

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Plik układu VHDL'u składa się z:
- nagłówka modułu:

```
entity naz_modułu is
    generic …
    port ...
end naz_modułu;
```

- opisu architektury:

```
architecture naz_arch of naz_modułu is
begin
    treść programu
end naz_arch;
```

W przypadku programu MAX+PLUS II nazwa modułu musi być identyczna z nazwą zamieszczoną po słowie **entity** (z pominięciem rozszerzenia pliku).

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

Do typów podstawowych należą typy liczbowe, kwantyfikatory fizyczne i typy wyliczeniowe. Do typów złożonych należą tablice i rekordy.

Ciągi bitowe:
- B“1001010“ - ciąg binarny,
- O“126“ - ciąg bitowy, odpowiada ciągowi B“001_010_110“,
- X“56“ - równoważny ciągowi bitowemu B“0101_0110“.

Typ całkowity:
- definiuje się jako podzbiór liczb z zakresu: -2147483647..+2147483647,
- przykłady:
  - **type** byte **is range** 0 **to** 255;
  - **type** shortint **is range** –128 **to** 127;
  - **type** bit_indeks **is range** 31 **downto** 0;

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

W wielu systemach projektowych głównie wykorzystywane są dwie biblioteki: STD oraz IEEE. W bibliotece IEEE występuje pakiet std_logic_1164, zawierający typy wielowartościowe: std_logic – odpowiednik typu bit, oraz std_logic_vector – odpowiednik typu bit_vector z biblioteki STD. Typ std_logic może przyjąć między innymi wartości: Z – wysoka impedancja, 0 – stan niski, 1 – stan wysoki, - – stan dowolny.

W celu użycia wymienionych typów wielowartościowych należy w pliku projektu umieścić następujące wiersze:

```
library ieee;
use ieee.std_logic_1164.all;
```

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

VHDL posiada elementy mające nazwę i wartość: stałe, zmienne i sygnały.

Stałe:
- **constant** e: int4 := 271828;

Zmienne:
- **variable** licznik: integer := 0;

Sygnały:
- **signal** nazwa: typ;

W VHDL zmienne służą do obliczeń wartości tymczasowych i nie są zamieniane na fizyczne elementy układu. Sygnały po syntezie przyjmują najczęściej w układzie postać fizyczną – połączenia, wejścia/wyjścia modułów, itp.

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

Podobnie jak inne języki programowania, także VHDL zawiera liczne wyrażenia i operatory. W poniższej tabeli zostały zdefiniowane operatory w kolejności ich priorytet.

**	abs	not	najwyższy
*	/	mod	
+ (znak)	- (znak)		
+	-	&	
=	/=	< <= > >=	
and	or	nand nor xor	najniższy

priorytet

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

Instrukcja IF ma postać:

```
if warunek0 then
   instrukcje
{ elsif warunke1 then
     instrukcje }
[ else
     instrukcje ]
end if;
```

Instrukcja CASE ma postać:

```
case wyrażenie is
  when warunek_wyboru0 => instrukcje;
{ when warunek_wyboru1 => instrukcje;}
[ when others => instrukcje; ]
end case;
```

Przykład:

```
case kod_instrukcji is
  when x"00"          => dodawanie;
  when x"01" | x"10"  => czekanie;
  when others         => nieznany_kod;
end case;
```

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

Instrukcja pętli ma postać:
- [nazwa_pętli**:**]
  [**while** warunke] |
  [**for** parametry] **loop**
  instrukcje;
  end loop;

W pętli można stosować:
- przejście do następnej iteracji:
  next [nazwa_pętli] **when** warunek;
- wyjście z pętli:
  exit [nazwa_pętli] **when** warunek;

Przykłady pętli:
- **loop**
  instrukcje;
  end loop;
- **while** i < dlugosc **loop**
  i := i+1;
  end loop;
- **for** i **in** tab'low **to** tab'high **loop**
  tab(i) := 0;
  end loop;

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

Podstawową jednostką opisu behawioralnego jest proces. W obrębie procesu definiuje się instrukcje, które wykonywane są sekwencyjnie w odpowiedzi na pojawiające się zdarzenia (zmiany stanu). Jeżeli więcej niż jeden proces zostanie uaktywniony w tym samym czasie, to instrukcje tych procesów są wykonywane równolegle. Deklaracja bloku procesu ma postać:

```
[nazwa_procesu:] process [(lista_sygnałów)] is
 część_deklaracyjna
 begin
   instrukcje
 end process [nazwa_procesu];
```

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

W ogólności proces jest blokiem wykonywanym współbieżnie z innymi procesami. Wstrzymanie pracy procesu jest możliwe w dwóch przypadkach:
- gdy warunek wznowienia nie jest spełniony: w takim przypadku zatrzymanie procesu zostanie wykonane po wykonaniu jego ostatniej instrukcji,
- gdy zostanie wykonana instrukcja wait: zatrzymanie procesu nastąpi po instrukcji wait.

Postać instrukcji wait jest następująca:
- **wait** [**on** sygnał {, sygnał}]

Procesy można synchronizować z poziomami logicznymi sygnałów:
- przez wykrywanie zmiany sygnałów wymienionych w deklaracji,
- przez warunek, np.: (clk'event and clk = '0') …
- makrodefinicją: rising_edge(clk) … .

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

VHDL jest językiem HDL, którego składnia została oparta na języku Ada. Obecnie każdy system EDA potrafi interpretować VHDL.

Pojęcia:
- metody projektowania
- struktura pliku vhd
- składnia języka

W VHDL'u układ można opisać:
- behawioralnie – opis zachowania układu,
- przepływowo – odpowiada modelowi RTL,
- strukturalnie – odzwierciedla strukturę układu.

Składnia języka VHDL odpowiada składni języka Ada. Język posiada również konstrukcje mające zastosowanie w syntezie układów cyfrowych.

Plik układu VHDL'u składa się z nagłówka modułu i opisu architektury ...

Elementy języka VHDL:
- typy podstawowe
- typy std_logic_1164
- zmienne i sygnały
- wyrażenia i operatory
- instrukcje if i case
- instrukcja loop
- proces i synchronizacja
- przypisania z listą wyboru

Przypisanie z listą wyboru jest podobne w implementacji do przypisania warunkowego. Deklaracja tego przypisania ma postać:

```
with wyrażenie select
   sygnał <= wartość when warunek {, wartość when warunek};
```

Przykład:

```
with funkcje_alu select
   wynik_alu  <=  op1 + op2     when alu_plus | alu_plusplus,
                  op1 - op2     when alu_minus | alu_minusminus,
                  op1 and op2 when alu_and,
                  op1 or op2    when alu_or;
```

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

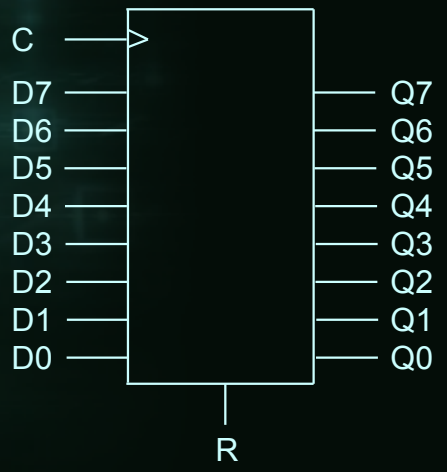
- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Wykorzystując język VHDL można w łatwy sposób modelować złożone układy cyfrowe, np. równoległy rejestr n-bitowy.

W pliku projektu należy umieścić dyrektywy biblioteki ieee i pakietu std_logic_1164.



Plik regn.vhd:

```
entity regn is
   generic (n:  positive := 8);
   port (C, R:  in std_logic;
         D:     in std_logic_vector(n-1 downto 0);
         Q:     out std_logic_vector(n-1 downto 0)
   );
end regn;

architecture regn_arch of regn is
begin
   process (R, C)
   begin
      if R = '1' then Q <= (others => '0');
      elsif rising_edge(C) then Q <= D;
      end if;
   end process;
end regn_arch;
```

M.Wiśniewski

Synteza ścieżki danych

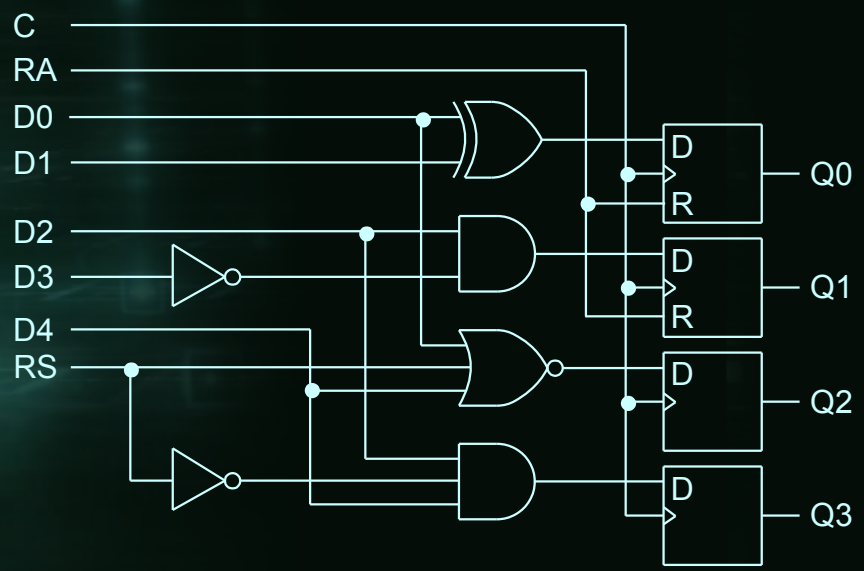
wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Poniższy układ zostanie przedstawiony w postaci behawioralnej i przepływowej. W obu stylach nagłówek entity jest taki sam.



Nagłówek pliku reg_rars.vhd:

```
library ieee;
use ieee.std_logic_1164.all;

entity reg_rars is
   port (C, RA, RS: in std_logic;
         D:         in std_logic_vector(4 downto 0);
         Q:         out std_logic_vector(3 downto 0)
   );
end reg_rars;
```

W ogólności sygnały mogą mieć charakter kombinacyjny lub rejestrowy. Zadeklarowane wyjścia będą sygnałami, które po syntezie będą wyjściami przerzutników (rejestrów).

Synteza ścieżki danych

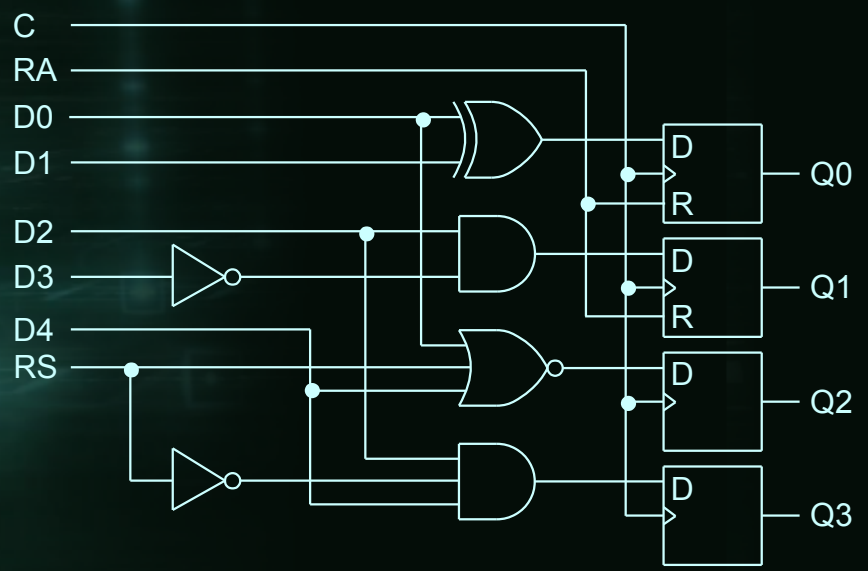
wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Poniższy układ zostanie przedstawiony w postaci behawioralnej i przepływowej. W obu stylach nagłówek entity jest taki sam.



Opis behawioralny układu reg_rars:

```
architecture regrars_arch of reg_rars is
begin
   p0: process
       begin
          if RA = '1' then Q(0) <= '0';
          else
              wait until rising_edge(C);
              Q(0) <= D(0) xor D(1);
          end if;
       end process p0;
   p1: process (C, RA)
       begin
          if RA = '1' then Q(1) <= '0';
          elseif rising_edge(C) then
              Q(1) <= D(2) and not D(3);
          end if;
       end process p1;
```

Synteza ścieżki danych

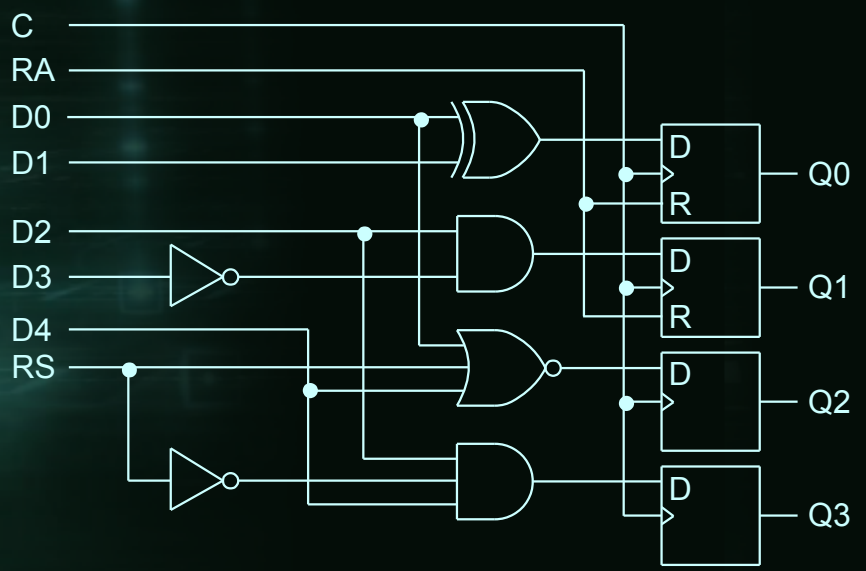
wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Poniższy układ zostanie przedstawiony w postaci behawioralnej i przepływowej. W obu stylach nagłówek entity jest taki sam.



Opis behawioralny układu reg_rars:

```
   . . .
   p2: process
       begin
          wait until rising_edge(C);
          Q(2) <= not (D(0) or D(4) or RS));
       end process p2;
   p3: process (C)
       begin
          if rising_edge(C) then
             if RS = '1' then Q(3) <= '0';
             else Q(3) <= D(2) and D(4);
          end if;
       end process p3;
end regrars_arch;
```

Synteza ścieżki danych

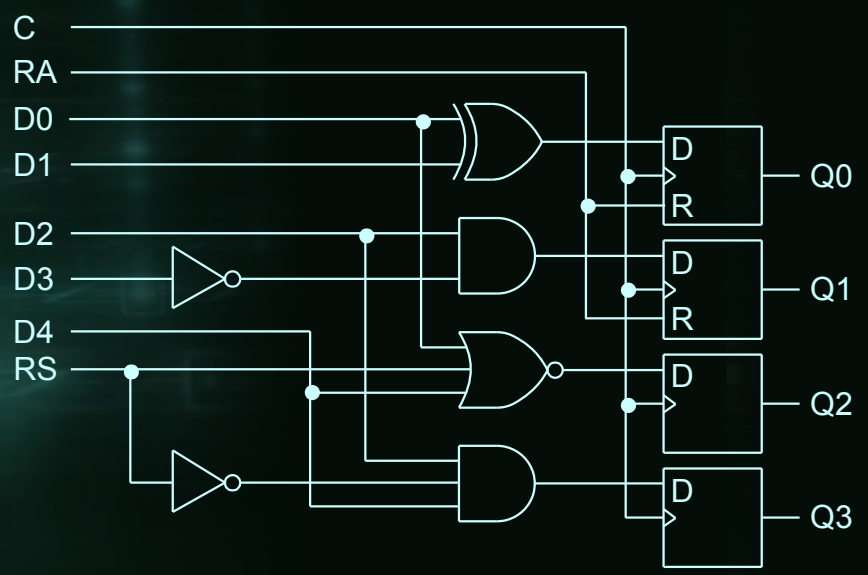
wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Poniższy układ zostanie przedstawiony w postaci behawioralnej i przepływowej. W obu stylach nagłówek entity jest taki sam.



Opis przepływowy układu reg_rars:

```
architecture regrars_arch of reg_rars is
   procedure regn (signal C: in std_logic;
                   signal R, D: in std_logic_vector;
                   signal Q: out std_logic_vector) is
      variable rr: boolean;
   begin
      rr := rising_edge(C);
      for  i  in R'range  loop
         if R(i) = '1' then Q(i) <= '0';
         elsif rr then Q(i) <= D(i);
         end if;
      end loop;
   end regn;

   signal w, z: std_logic_vector(3 downto 0);
   . . .
```

Synteza ścieżki danych

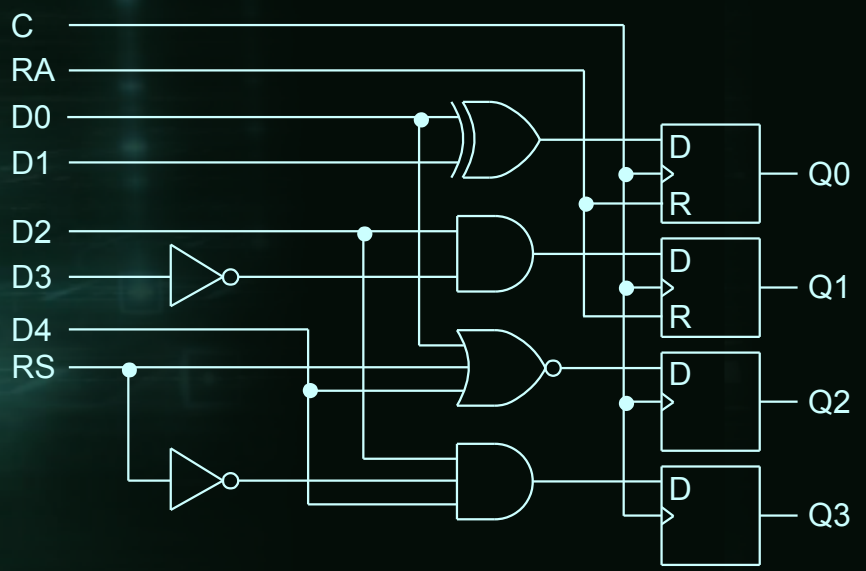
wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Poniższy układ zostanie przedstawiony w postaci behawioralnej i przepływowej. W obu stylach nagłówek entity jest taki sam.



Opis przepływowy układu reg_rars:

```
. . .
begin -- architecture
   w(0) <= D(0) xor D(1);
   w(1) <= D(2) and not D(3);
   w(2) <= not (D(0) or D(4) or RS);
   w(3) <= (not RS) and D(2) and D(4);
   z <= ('0', '0', RA, RA);
   regn(C, z, w, Q);
end regrars_arch;
```

M.Wiśniewski

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

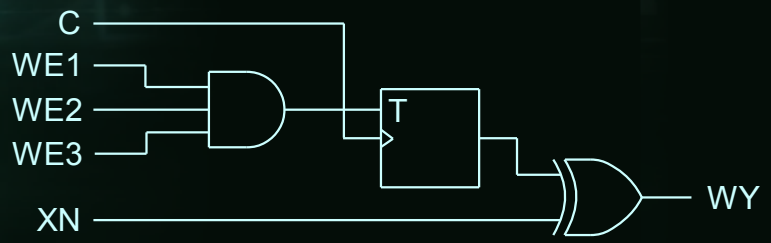
Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

W opisie strukturalnym opisuję się architekturę połączeń układów o prostszej strukturze (gotowych modułów), które deklaruje się słowem component.

Połączenia między wejściami i wyjściami modułów wymagają deklaracji sygnałów pomocniczych. Do wejść modułów mogą być przypisane wartości stałe lub wyrażenia.

Połączenia modułów tworzy się wykorzystując instrukcję łączenia komponentów port map.

Poniższy przykład składa się z trzech komponentów: bramek and i xor oraz przerzutnika typu T.



Moduł bramki AND3:

```
entity and3 is
   port (A, B, C: in std_logic; Y: out std_logic);
end and3;

architecture and3_arch of and3 is
begin
   Y <= A and B and C;
end and3;
```

Moduł bramki XOR2:

```
entity xor2 is
   port (A, B: in std_logic; Y: out std_logic);
end xor2;

architecture xor2_arch of xor2 is
begin
   Y <= A xor B;
end xor2;
```

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

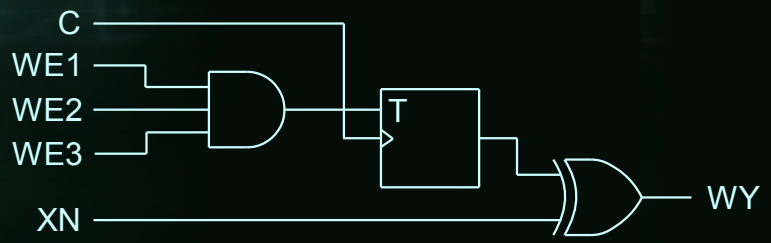
Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

W opisie strukturalnym opisuję się architekturę połączeń układów o prostszej strukturze (gotowych modułów), które deklaruje się słowem component.

Połączenia między wejściami i wyjściami modułów wymagają deklaracji sygnałów pomocniczych. Do wejść modułów mogą być przypisane wartości stałe lub wyrażenia.

Połączenia modułów tworzy się wykorzystując instrukcję łączenia komponentów port map.

Poniższy przykład składa się z trzech komponentów: bramek and i xor oraz przerzutnika typu T.



Moduł przerzutnika typu T:

```
entity fft is
   port (T, C, R: in std_logic;
         Q: out std_logic);
end fft;

architecture fft_arch of fft is
   signal tt: std_logic;
begin
   process (C, R)
   begin
      if R = '1' then tt <= '0';
      elsif rising_edge(C) then tt <= tt xor T;
      end if;
   end process;
   Q <= tt;
end fft_arch;
```

Synteza ścieżki danych

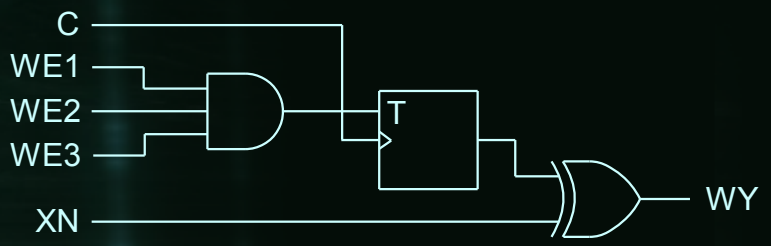
wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Opis strukturalny układu:




```
entity uklad is
   port (C, WE1, WE2, WE3, XN: in std_logic;
         WY: out std_logic
   );
end uklad;

architecture uklad_arch of uklad is
   component and3
      port (A, B, C: in std_logic := '1';
            Y: out std_logic);
   end component;
   component xor2
      port (A, B: in std_logic;
            Y: out std_logic);
   end component;
   component fft
      port (T, C, R: in std_logic;
            Q: out std_logic);
   end component;
   signal T1, QW, B_WY: std_logic;
begin
   B0: and3 port map (WE1, WE2, WE3, T1);
   T0: fft port map (T => T1, C => C, R => '0',  Q => QW);
   B1: xor2 port map (A => QW, B => XN, Y => B_WY);
   WY <= B_WY;
end uklad_arch;
```

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

W tym przykładzie układ został wykonany w stylu behawioralnym. Moduł wykonuje operacje:
- funkcje logiczne: not, and, nand, or, nor, xor, xnor,
- operacje arytmetyczne (U2): +, - , ++, --, >>>, <<<.

Układ posiada dodatkowe sygnały wyjściowe:
SGN – bit znaku, OV – bit przepełnienia, CO – wyjście przepełnienia, ZERO – bit zerowego wyniku operacji.

Operację wybiera się przez magistralę 4-bitową M.

$M_3M_2M_1M_0$	operacja	$M_3M_2M_1M_0$	operacja
0 0 0 0	P	1 0 0 0	P + Q
0 0 0 1	~P	1 0 0 1	P – Q
0 0 1 0	PQ	1 0 1 0	P + 1
0 0 1 1	~(PQ)	1 0 1 1	P – 1
0 1 0 0	P v Q	1 1 0 0	SHL P
0 1 0 1	~(P v Q)	1 1 0 1	SHR P
0 1 1 0	P ⊕ Q	1 1 1 0	00...0
0 1 1 1	~(P ⊕ Q)	1 1 1 1	11...1

Plik alu.vhd:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity alu is
   generic (N: positive := 8);
   port (P, Q: in std_logic_vector(N-1 downto 0);
         M: in std_ logic_vector(3 downto 0);
         F:  out std_logic_vector(N-1 downto 0);
         SGN, OV, CO, ZERO: out std_logic);
end alu;

architecture alu_arch of alu is
   signal tF: std_logic_vector(N-1 downto 0);
   signal tCO: std_logic;
. . .
```

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Plik alu.vhd – ciąg dalszy:

```
begin
   ALU: process (M, P, Q)
   begin
      case M is
         when ”0000” => tF <= P;
         when ”0001” => tF <= not P;
         when ”0010” => tF <= P and Q;
         when ”0011” => tF <= P nand Q;
         when ”0100” => tF <= P or Q;
         when ”0101” => tF <= P nor Q;
         when ”0110” => tF <= P xor Q;
         when ”0111” => tF <= not (P xor Q);
         when ”1000” => tF <= P + Q;
         when ”1001” => tF <= P – Q;
         when ”1010” => tF <= P + 1;
         when ”1011” => tF <= P – 1;
         . . .
         when ”1100” => tF <= P(N-2 downto 0) & ’0’;
         when ”1101” => tF <= ’0’ & P(N-1 downto 1);
         when ”1110” => tF <= (others => ’0’);
         when ”1111” => tF <= (others => ’1’);
         when others => tF <=  (others => ’0’);
      end case;
      F <= tF;
      . . .
```

Wewnątrz procesu ALU należy również obliczyć wartości bitów stanu i przypisać je do odpowiednich sygnałów.

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Plik alu.vhd – ciąg dalszy:

```
. . .
if  M > ”0111” and M < ”1100” then
    -- przeniesienie
    tCO <= (P(N-1) and Q(N-1)) or
           (not tF(N-1) and
           (P(N-1) xor Q(N-1)));
    -- bit znaku
    SGN <= tF(N-1);
    -- przepełnienie
    OV <= tCO xor (tF(N-1) xor
          P(N-1) xor Q(N-1));
else
    tCO <= ’0’;
    SGN <= ’0’;
    OV <= ’0’;
end if;
. . .
```

```
. . .
CO <= tCO;
if tF = 0 then
    ZERO <= ’1’;
else
    ZERO <= ’0’;
end if;
end process ALU;
end alu_arch;
```

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Architektura układów FLEX, STRATIX i innych jest złożona, a same układy oferują bogate możliwości, których nie można wprost wykorzystać podczas tworzenia opisów układów w języku VHDL. Aby umożliwić korzystanie z tych szczególnych własności, Altera udostępnia parametryzowane moduły, które zostały zadeklarowane jako komponenty w pakiecie lpm_components.

Dostępne moduły mają rozbudowaną listę parametrów wymienionych w sekcjach generic w nagłówkach komponentów. Podczas projektowania zmianę parametrów wykonuje się przez użycie polecenia generic map.

Moduł LPM	Opis działania
- podstawowe:	
lpm_and	- wielobitowa bramka AND,
lpm_bustri	- wielobitowy bufor trójstanowy,
lpm_clshift	- uniwersalny rejestr przesuwny,
lpm_decode	- dekoder 1 z n,
lpm_or	- wielobitowa bramka OR,
lpm_xor	- wielobitowa bramka XOR,
lpm_max	- multiplekser grupowy n na m.
- arytmetyczne:	
lpm_abs	- wartość bezwzględna,
lpm_add_sub	- wielobitowy sumator/subtraktor,
lpm_compare	- komparator dwóch liczb n-bitowych,
lpm_counter	- wielobitowy licznik dwójkowy,
lpm_mult	- wielobitowy multiplikator.

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Architektura układów FLEX, STRATIX i innych jest złożona, a same układy oferują bogate możliwości, których nie można wprost wykorzystać podczas tworzenia opisów układów w języku VHDL. Aby umożliwić korzystanie z tych szczególnych własności, Altera udostępnia parametryzowane moduły, które zostały zadeklarowane jako komponenty w pakiecie lpm_components.

Dostępne moduły mają rozbudowaną listę parametrów wymienionych w sekcjach generic w nagłówkach komponentów. Podczas projektowania zmianę parametrów wykonuje się przez użycie polecenia generic map.

Moduł LPM	Opis działania
- pamięciowe:	
lpm_ram_dq	- pamięć RAM z oddzielonymi portami wejściowym i wyjściowym,
lpm_ram_io	- pamięć RAM z jednym portem dwukierunkowym I/O,
lpm_rom	- pamięć ROM
csdpram	- pamięć RAM z oddzielonymi portami wejściowym i wyjściowym współdzielona w jednym cyklu zegara,
csfifo	- pamięć FIFO.

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Przykład użycia komponentu z biblioteki LPM: moduł pamięci RAM 256x8 z użyciem pakietu lpm_components i modułu lpm_ram_dq, który implementuje pamięć z rozdzielonymi wejściami i wyjściami.

W przykładzie określono liczbę portów wejściowych (lpm_width = 8) oraz liczbę wejść adresowych (lpm_widthad = 8). Pozostałe parametry przyjmują wartości zdefiniowane w klauzuli generic nagłówka modułu:
- parametry lpm_indata, lpm_address_control i lpm_outdata określają typy portów - rejestrowego (REGISTRED) lub nierejestrowe UNREGISTERED,
- sygnały zegarowe INCLOCK i OUTCLOCK realizują odpowiednio operację synchronicznego zapisu (gdy WE = '1') i odczytu danych z pamięci.

Moduł RAM 256x8:

```
library ieee;
use ieee.std_logic_1164.all;
library lpm;
use lpm.lpm_components.all;

entity ram256x8 is
   port(DATA: in std_logic_vector(7 downto 0);
        ADDR: in std_logic_vector(7 downto 0);
        WE, INCLOCK, OUTCLOCK: in std_logic;
        Q: out std_logic_vector(7 downto 0));
end ram256x8;

architecture ram256x8_arch of ram256x8 is
   component lpm_ram_dq
      generic (
        lpm_width: positive;
        lpm_type: String := "LPM_RAM_DQ";
        . . .
```

Synteza ścieżki danych

wysokopoziomowa metoda syntezy układów logicznych

- wstęp
- język VHDL
- przykłady modułów

Moduły VHDL wykonuje się stosując opis behawioralny, przepływowy lub strukturalny. Poniższe przykłady pokazują każdą z tych metod.

Przykłady:
- rejestr n-bitowy
- style projektowania
- moduły Altery w VHDL

Moduł RAM 256x8 ciąg dalszy:

```
. . .
    lpm_widthad: positive;
    lpm_numwords: natural := 8;
    lpm_file: String := ”UNUSED”;
    lpm_indata: String := ”REGISTERED”;
    lpm_address_control: String := ”REGISTERED”;
    lpm_outdata: String := ”REGISTERED”;
    lpm_hint: String := ”UNUSED”);
  port (
    data: in std_logic_vector (lpm_width-1 downto 0);
    address: in std_logic_vector (lpm_widthad-1 downto 0);
    we: in std_logic;
    inclock: in std_logic := ’0’;
    outclock: in std_logic := ’0’;
    q: out std_logic_vector (lpm_width-1 downto 0));
end component;
. . .
```

```
. . .
begin
   U1: lpm_ram_dq
          generic map (lpm_width =>8,
                       lpm_widthad => 8)
          port map (data => DATA,
                    address => ADDR,
                    we => WE,
                    inclock => INCLOCK,
                    outclock => OUTCLOCK,
                    q => Q);
end ram256x8_arch;
```

Koniec wykładu