Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

Факультет электроники и вычислительной техники

Кафедра «Электронно-вычислительные машины и системы»

Контрольная работа

по дисциплине «Архитектура ЭВМ»

вариант

39 (100111)

7 (00111)

Выполнил:

студент гр. ПрИн-267

Смирнов Дмитрий Андреевич

Проверил:

к. т. н. доцент

Виталий Алексеевич Егунов

Волгоград 2023

**Содержание**

[Цель и задачи работы](#_jkoo9eiz3sij) 3

Характеристика процессора 3

Разработка формата команд и кодировки команд 5

Структурная схема 9

Схема операционного устройства 10

1)Блок регистров общего назначения(GPR) 10

2)Адресное АЛУ 11

3)Общее АЛУ 12

4) Общая схема операционного устройства 14

Контролирующие устройств 14

1) ПЗУ с микрокомандами 14

2) Сигналы Jump/Decode в зависимости от условия 15

3) Устройство формирования адреса 16

4) Общая схема контролирующего устройство 17

5) MuxA 17

6) MuxB 17

7) MuxC 18

Общая схема процессора 19

Микрокод 20

Тесты 22

## **Цель и задачи работы**

Синтезировать по варианту структуру простого магистрального процессора с одним или двумя АЛУ (возможно наличие отдельного адресного АЛУ), выполняющего от 8-и заданных команд. Разработать форматы команд, кодировку команд. Разработать структурную схему процессора, функциональные схемы всех блоков процессора, функциональную схему процессора в целом с указанием всех шин и управляющих сигналов. Разработать формат микрокоманд, организацию управления всеми устройствами процессора, микрокод для каждой из заданных команд. Привести примеры выполнения каждой команды с указанием значения всех основных сигналов и содержимого основных регистров на каждом такте. Привести 2 примера небольших программ с указанием значения основных сигналов и содержимого основных регистров на каждом такте.

Определить максимальную тактовую частоту процессора. Определить производительность процессора в операциях в секунду (IPS), а также выраженную в числе выполняемых тестовых программ в секунду. Указать способы повышения производительности процессора. Характеристика процессора: простой процессор магистрального типа с одноблочным универсальным АЛУ (и возможно – с дополнительным адресным АЛУ по вариантам). Разрядность регистров РОН и АЛУ процессора – 8 бит. Число РОН – не менее 4. Адресуемая память - от 256 слов. Устройство управления – микропрограммное с памятью микропрограмм. Способ выполнения команд – последовательное выполнение.

**Характеристики процессора**

Простой процессор магистрального типа с одноблочным универсальным АЛУ (и возможно – с дополнительным адресным АЛУ по вариантам).

Разрядность регистров РОН и АЛУ процессора – 8 бит.

Число РОН – не менее 4.

Адресуемая память – от 256 слов.

Устройство управления – микропрограммное с памятью микропрограмм.

Способ выполнения команд – последовательное выполнение.

*Индивидуальные особенности процессора:*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Пункт(Особенности АЛУ)** | **Размер** | *1* | *0* | *0* | *1* | *1* | *1* | **Вариант** |
| Адресность команд | 1 бит |  |  |  |  |  | 1 | 2 |
| Адресная арифметика | 1 бит |  |  |  |  | 1 |  | 2 |
| Операции АЛУ | 2 бита |  |  | 0 | 1 |  |  | 2 |
| Тип сдвигателя АЛУ | 2 бита | 1 | 0 |  |  |  |  | 3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Пункт** | **Размер** | *0* | *0* | *1* | *1* | *1* | **Вариант** |
| Логика обработки команд | 1 бит |  |  |  |  | 1 | 2 |
| Состав команд | 2 бита |  |  | 1 | 1 |  | 4 |
| Варианты адресации | 2 бита | 0 | 0 |  |  |  | 1 |

|  |  |
| --- | --- |
| Используемые варианты адресации | Прямая, регистровая, косвенная регистровая, непосредственная |
| Адресность команд | Трехадресные (типа add r1, r2, r3) |
| Операции АЛУ | ADD, SHRA(вправо арифм.), NAND, SET |
| Тип сдвигателя в АЛУ | Монтажный сдвиг на 1 разряд в одну сторону(вправо) |
| Адресная арифметика | На отдельном АЛУ(Вместо NOP исп SET) |
| Состав команд | LD, ST, ADD, SHRA, JE, **NAND**, JMP, XCH |
| Логика обработки разных форматов одних и тех же команд | Комбинационная |

## **Разработка формата команд и кодировки команд**

Исходные данные:

- перечень команд: LD, ST, ADD, SHRA, JE, NAND, JMP, XCH.

- команды трехадресные

- перечень регистров: 4 регистра (R0 – R3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| LD |  | Регистровая - 00 |  |  | R0 | 00 |
| ST |  | Непосредственная - 01 |  | R1 | 01 |
| ADD |  | Прямая - 10 |  | R2 | 10 |
| SHRA |  | Косвенно-регистровая - 11 |  | R3 | 11 |
| NAND |  |  |  |  |  |  |
| JE |  |  |  |  |  |  |
| JMP |  |  |  |  |  |  |
| XCH |  |  |  |  |  |  |

1. LD – все варианты адресации; двухместные, так как трехместные здесь не имеют смысла.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регистровая  *LD R0, R1* | КОП + с/а | Rdst | Rsrc |  |  |  |
| 0000 | xx | xx | *не исп.* | 8 бит | 1 байт |
|  | | | | | | | |
| Непосредственная  *LD R0, #10* | КОП + с/а | Rdst |  | const |  |  |
| 0001 | xx | *не исп.* | xxxxxxxx | 14 бит | 2 байт |
|  | | | | | | | |
| Косвенно-регистровая  *LD R0, [R1]* | КОП + с/а | Rdst | [Rsrc] |  |  |  |
| 0010 | xx | xx | *не исп.* | 8 бит | 1 байт |
|  | | | | | | | |
| Прямая  *LD R0, 10* | КОП + с/а | Rdst |  | addr8 |  |  |
| 0011 | xx | *не исп.* | xxxxxxxx | 14 бит | 2 байт |
|  |  |  |  |  |  |

1. ST – Аналогично LD, однако нет смысла в использовании регистровой и непосредственной адресации, т.к. получаем полные аналоги команды LD. Используется только прямая и косвенно - регистровая, имеет смысл использовать только варианты с памятью. Трехадресные команды не имею смыла, используем лишь двухадресные.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Косвенно-регистровая  *ST R0, [R2]* | КОП + с/а | Rsrc | [Rdst] |  |  |  |
| 0100 | xx | xx | *не исп.* | 8 бит | 1 байт |
|  | | | | | | | |
| Прямая  *ST R0, addr* | КОП + с/а | Rsrc |  | addr8 |  |  |
| 0101 | xx | *не исп.* | xxxxxxxx | 14 бит | 2 байт |

Команды арифметико-логической обработки только внутри процессора, к памяти не обращаемся, поэтому используется только регистровая и непосредственная адресация.

1. ADD – регистровая и непосредственная адресация

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регистровая  *ADD R0, R1, R2* | КОП + с/а | Rdst | Rsrc1 | Rsrc2 |  |  |
| 0110 | xx | xx | \_\_\_\_\_\_xx | 10 бит | 2 байт |
|  |  |  |  |  |  |  |
| Непосредственная  *ADD R0, R1, #10* | КОП + с/а | Rdst | Rsrc1 | const |  |  |
| 0111 | xx | xx | xxxxxxxx | 16 бит | 2 байт |

1. SHRA – регистровая и непосредственная адресация(только двухместная)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регистровая  *SHRA R0, R1* | КОП + с/а | Rdst | Rsrc1 | Rsrc2 |  |  |
| 1000 | xx | xx | *не исп.* | 8 бит | 1 байт |
|  |  |  |  |  |  |  |
| Непосредственная  *SHRA R0, #10* | КОП + с/а | Rdst | Rsrc1 | imm8 |  |  |
| 1001 | xx | *не исп.* | xxxxxxxx | 14 бит | 2 байт |

1. NAND – регистровая и непосредственная адресация

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регистровая  *NAND R0, R1, R2* | КОП + с/а | Rdst | Rsrc1 | Rsrc2 |  |  |
| 1010 | xx | xx | \_\_\_\_\_\_xx*.* | 10 бит | 2 байт |
|  |  |  |  |  |  |  |
| Непосредственная  *NAND R0, R1, #10* | КОП + с/а | Rdst | Rsrc1 | const |  |  |
| 1011 | xx | xx | xxxxxxxx | 16 бит | 2 байт |

1. JE – прямая адресация (можно оставить один вариант адресации)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Прямая  *JE 10* | КОП +с/а | Rdst | Rsrc1 | addr8 |  |  |
| 1100 | *не исп.* | *не исп.* | xxxxxxxx | 12 бит | 2 байт |
|  | | | | | | | |

1. JMP – прямая адресация (можно оставить один вариант адресации)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Прямая  *JMP 10* | КОП + с/а | Rdst | Rsrc1 | addr8 |  |  |
| 1101 | *не исп.* | *не исп.* | xxxxxxxx | 12 бит | 2 байт |
|  | | | | | | | |

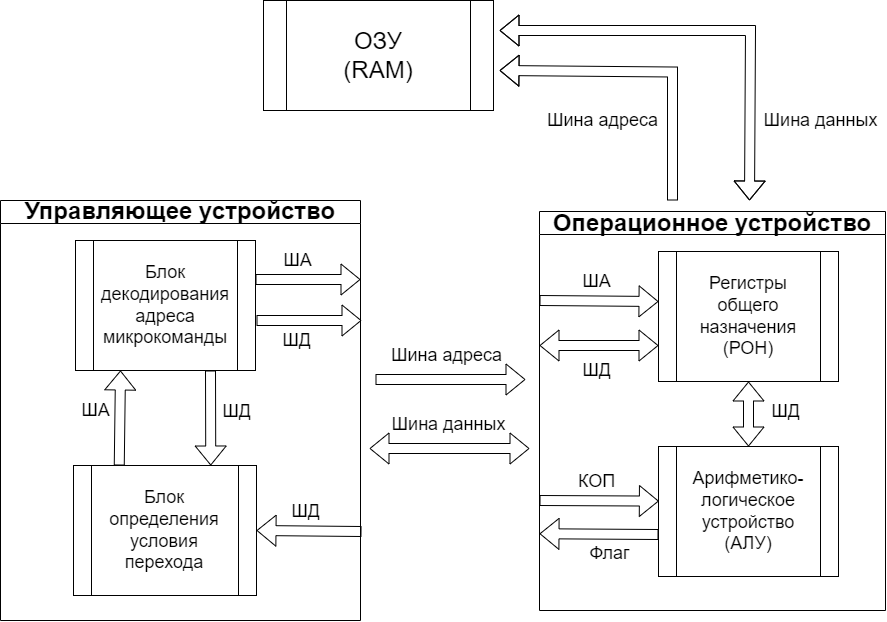
1. XCH – обмен данных в регистре, двухместная.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регистровая  *XCH R0, R1* | КОП +с/а | Rsrc1 | Rsrc2 |  |  |  |
| 1110 | xx | xx | *не исп.* | 8 бит | 1 байт |

**Структурная схема**

В операционном устройстве осуществляется хранение данных, а также выполняются операции над ними. Оно может считывать данные из оперативного запоминающего устройства (ОЗУ) или записывать их в него.

Управляющее устройство определяет, какая микрокоманда выполняется следующей, а также регулирует все действия процессора. Доступ к следующему коду операции (КОП) и другим данным оно получает из операционного устройства.



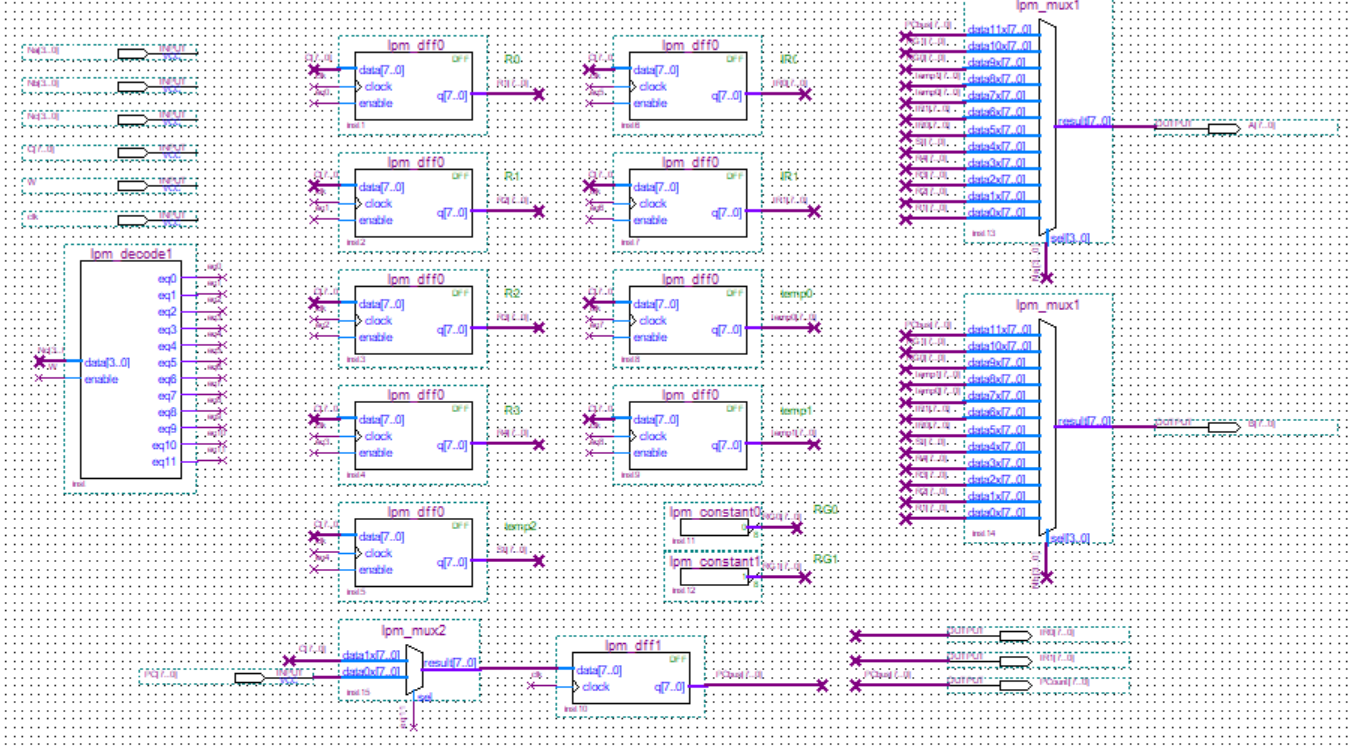
**Схема операционного устройства**

1. **Блок регистров общего назначения(GPR)**

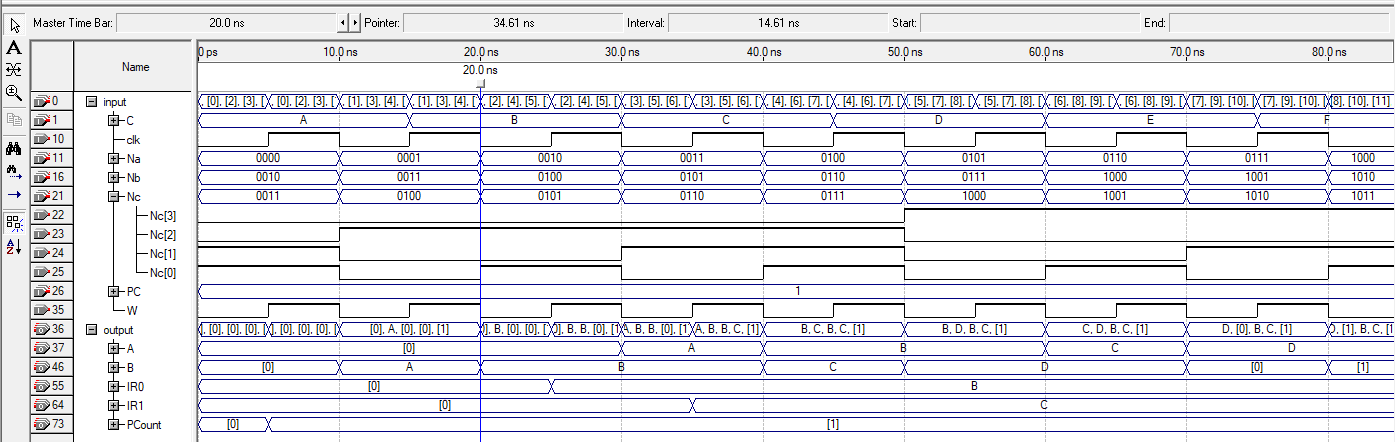
Блок регистров включают в себя 4 программно-доступных регистров (R0-R3), программно-недоступные регистры: программный счетчик (PC), регистры команд (IR0 и IR1), временный регистры temp0, temp1,temp2, а также константы rg0(хранит 0), rg1(хранит 1).

|  |  |  |  |
| --- | --- | --- | --- |
| Регистры | Название | Адрес | |
| В микрокоманде | В коде |
| R0 | Регистры общего назначения | 0000 | 00 |
| R1 | 0001 | 01 |
| R2 | 0010 | 10 |
| R3 | 0011 | 11 |
| temp2 | Буферный регистр | 0100 | - |
| IR0 | Регистр команд | 0101 | - |
| IR1 | 0110 | - |
| temp0 | Буферные регистры | 0111 | - |
| temp1 | 1000 | - |
| rg0 | Константные регистры | 1001 | - |
| rg1 | 1010 | - |
| PC | Регистр программного счетчика | 1011 | - |

Схема:



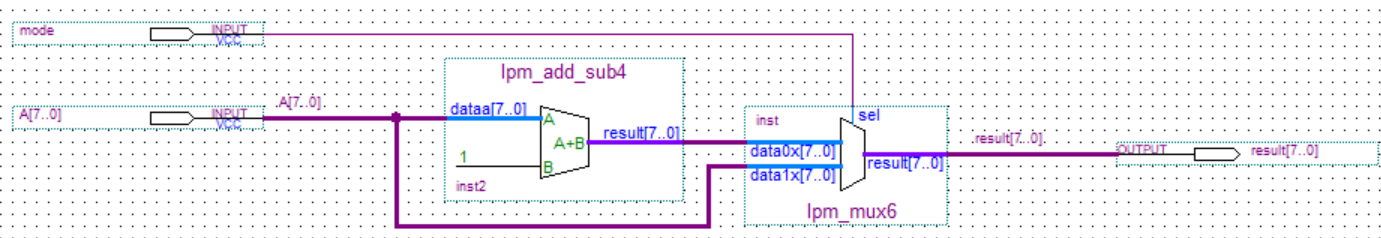
Функциональная диаграмма:



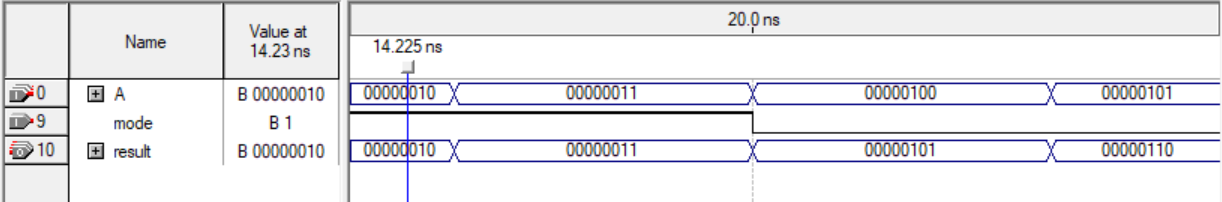
1. **Адресное АЛУ**

Так как по варианту имеется отдельное АЛУ для адресной арифметики, он в данном случае пригодится лишь для параллельного инкрементирования программного счетчика.

Схема

****

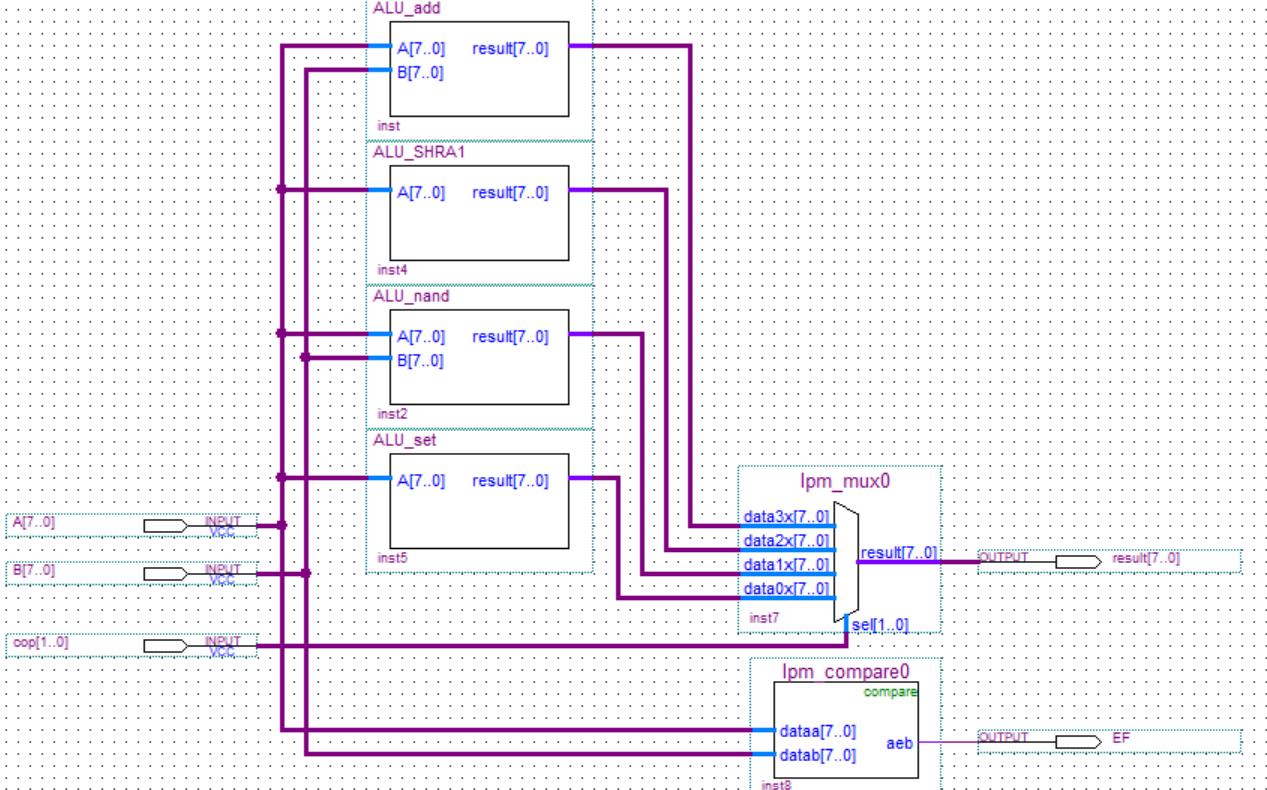
Фнкциональная диаграмма:

****

1. **Общее АЛУ**

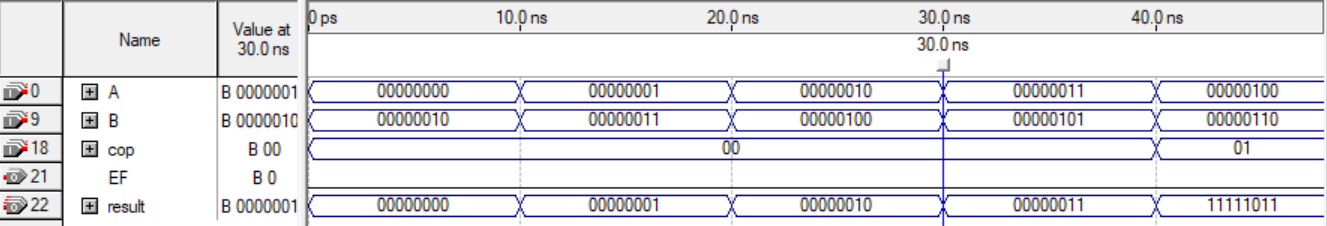
|  |  |  |  |
| --- | --- | --- | --- |
| Операция | На выходе | КОП | Комментарий |
| SET | A | 00 | Установление поданного значения (аналогично NOP) |
| NAND | ! (A && B) | 01 | И-НЕ |
| SHRA | A >> 1 | 10 | Сдвиг числа на 1 бит вправо |
| ADD | A + B | 11 | Сложение |

Схема

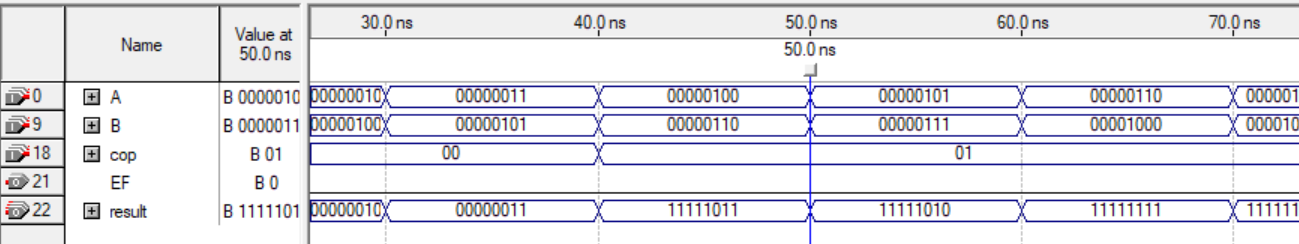


Функциональная диаграмма:

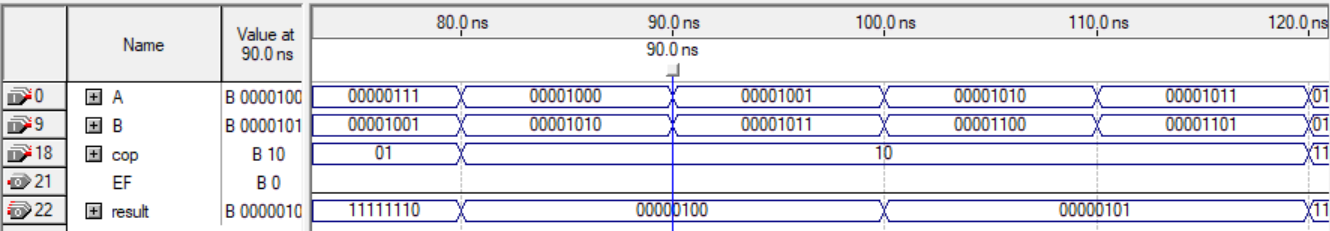
COP = 00 (SET)



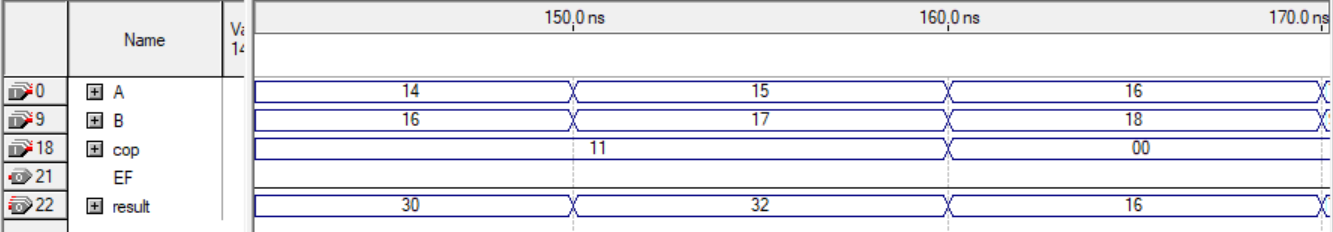
COP = 01(NAND)



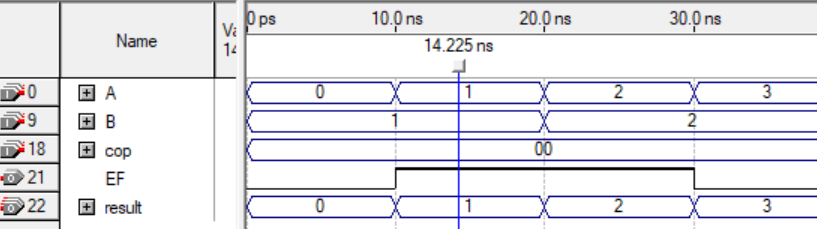
COP = 10(SHRA)



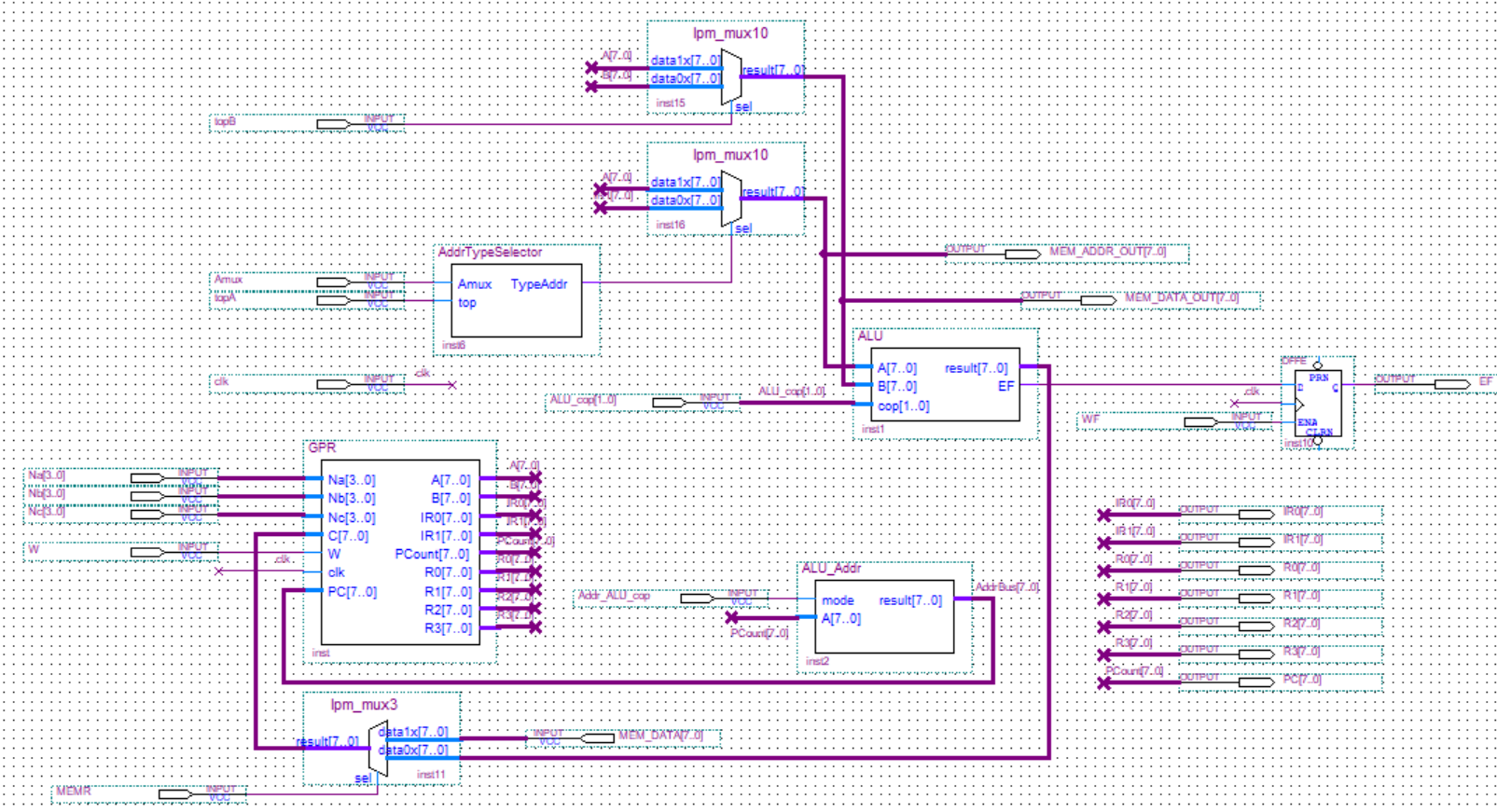
COP=11(ADD)



Equal Flag



1. **Операционное устройство**

Схема:

**Контролирующие устройство**

1. **ПЗУ с микрокомандами**

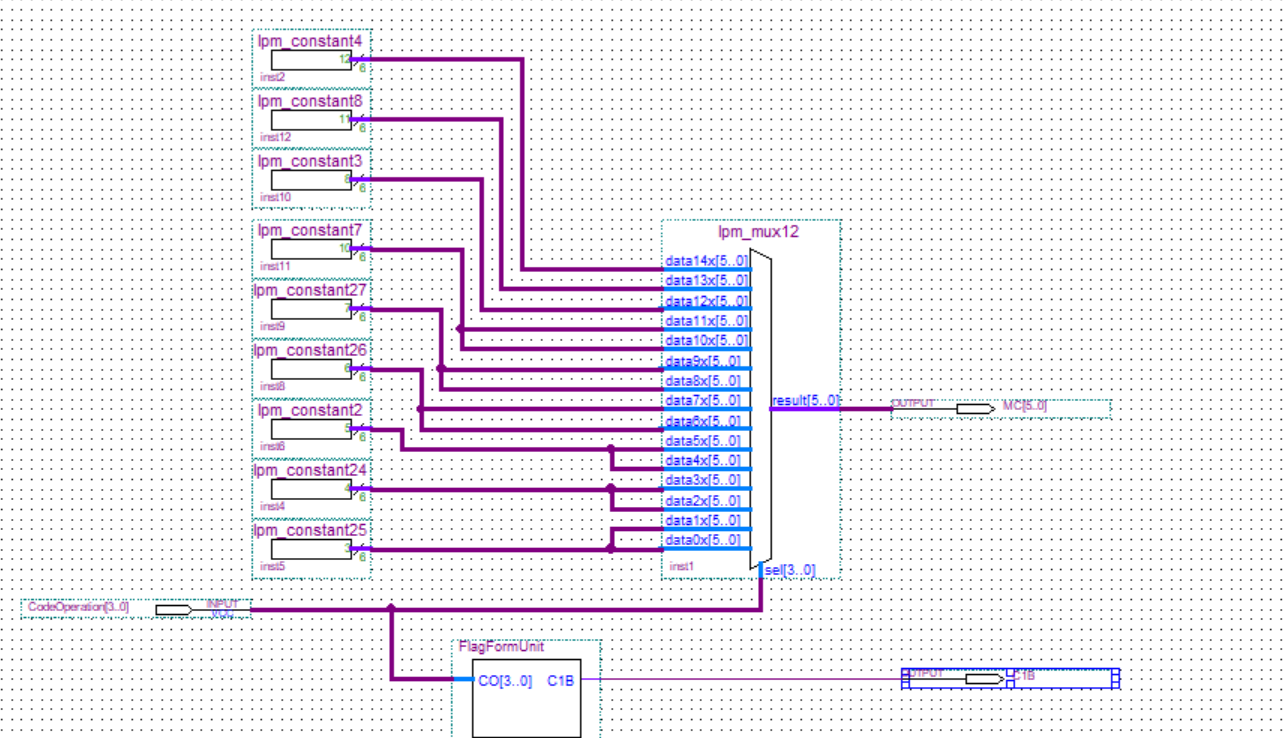
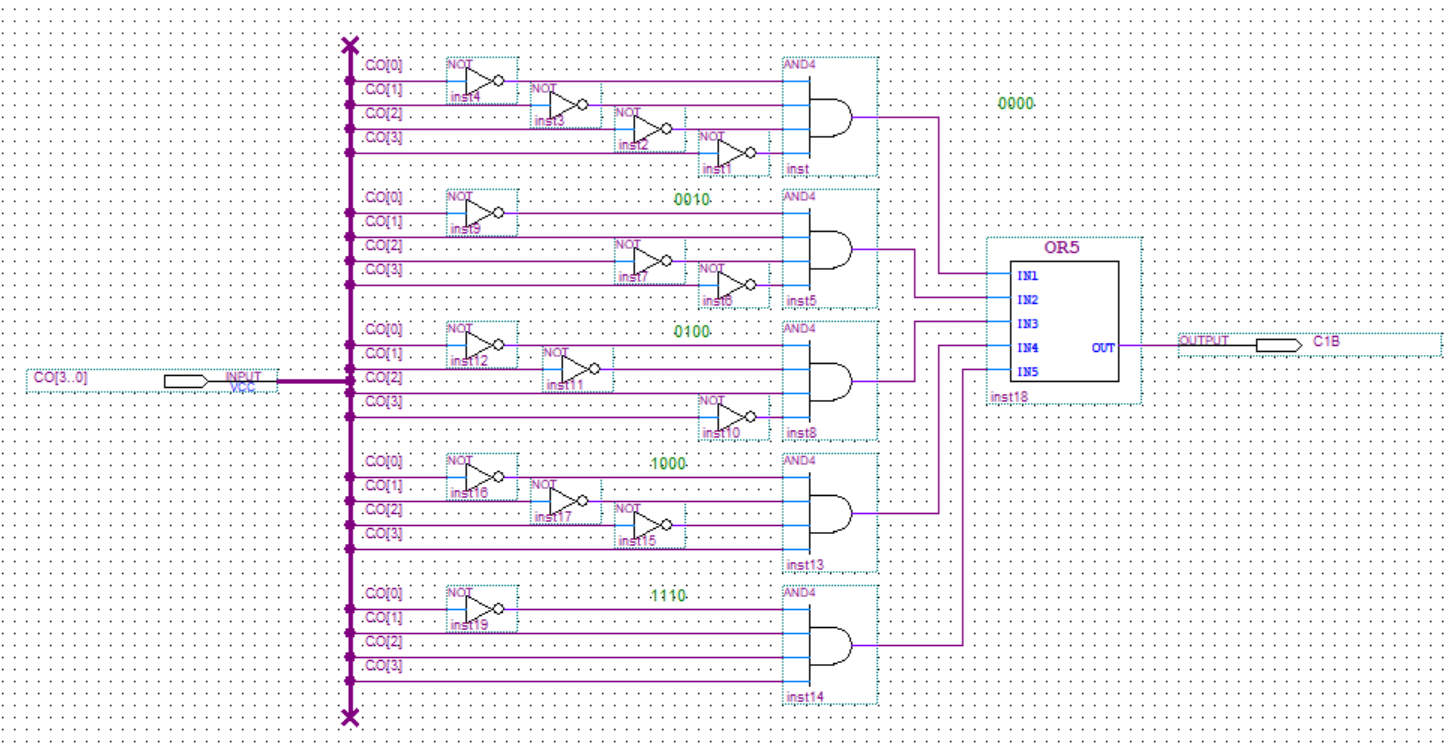
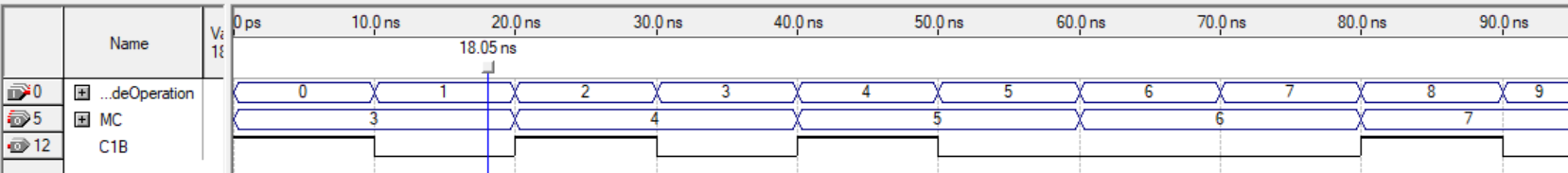
Схема:

Схема устройства формирование флага:

В устройсве перечислены однобайтовые команды, если таковая поступает навход для флаг C1B будет равен 1

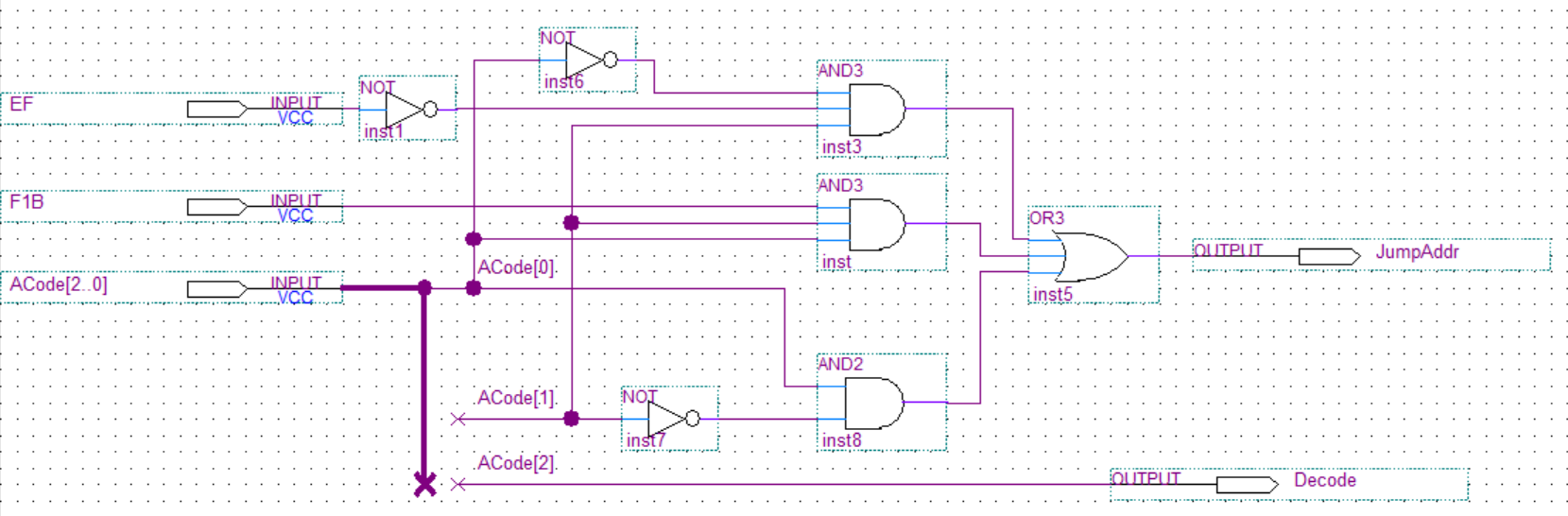


Функциональная схема:

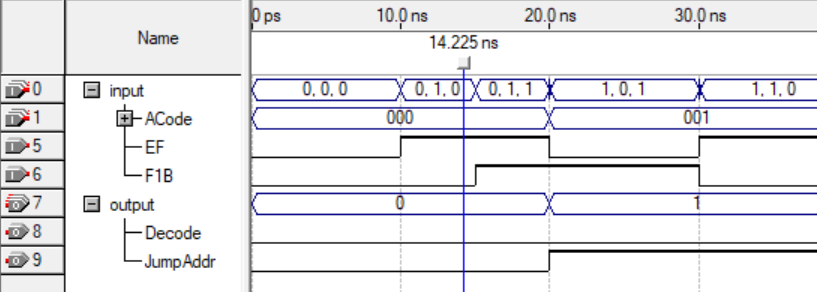
1. **Сигналы Jump/Decode в зависимости от условия**

|  |  |  |
| --- | --- | --- |
| Переходы | | |
| NEXT | 000 | Переход к следующей МК |
| JMP | 001 | Безусловный переход |
| JNE | 010 | Условный переход при EF = 0, иначе NEXT |
| J1B | 011 | Условный переход, если команда 1-байтная, иначе NEXT |
| DECODE | 100 | Декодирование КОП команды и определение адреса МК |

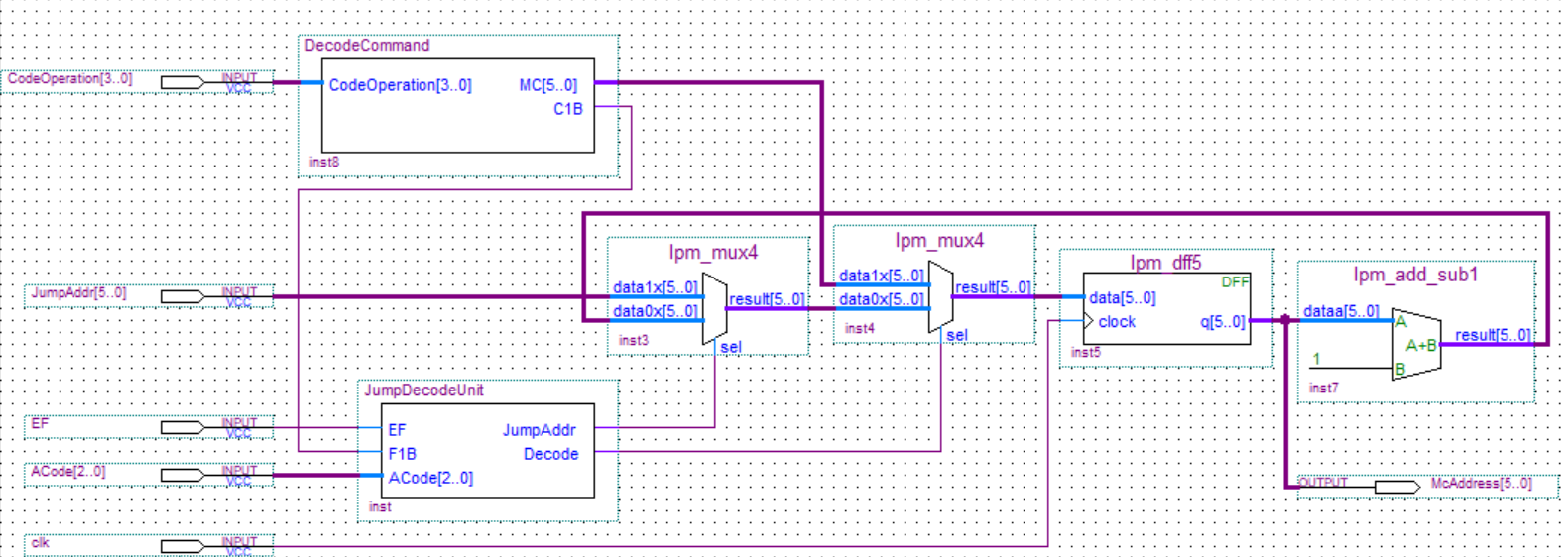
Cхема:



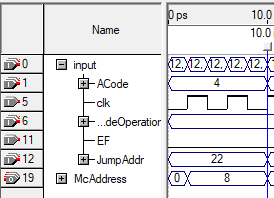
Функциональная диаграмма:



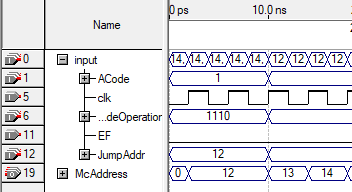
1. **Устройство формирования адреса**

Схема:

Функциональная диаграмма:

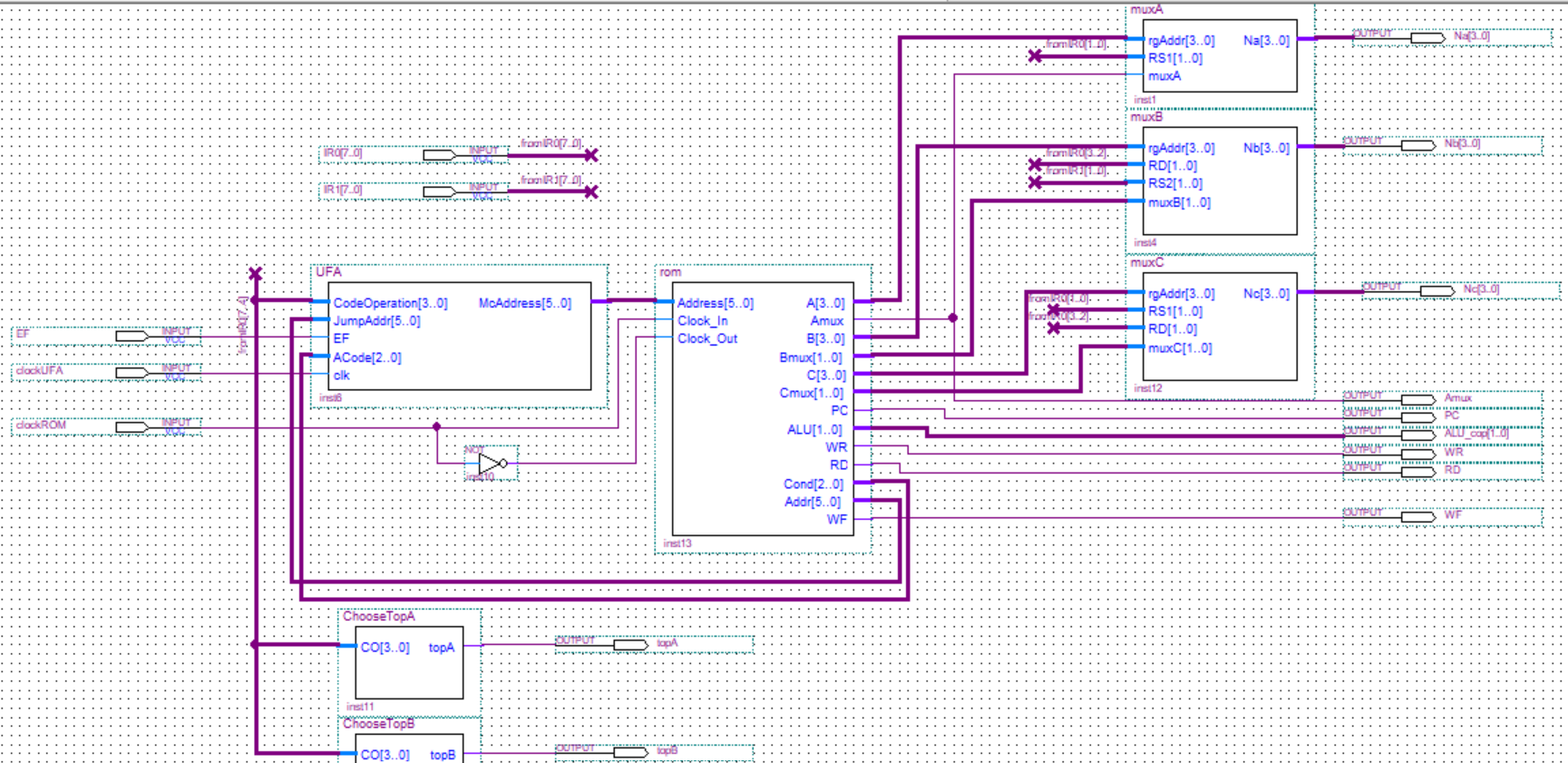
Определение кода команды JE:

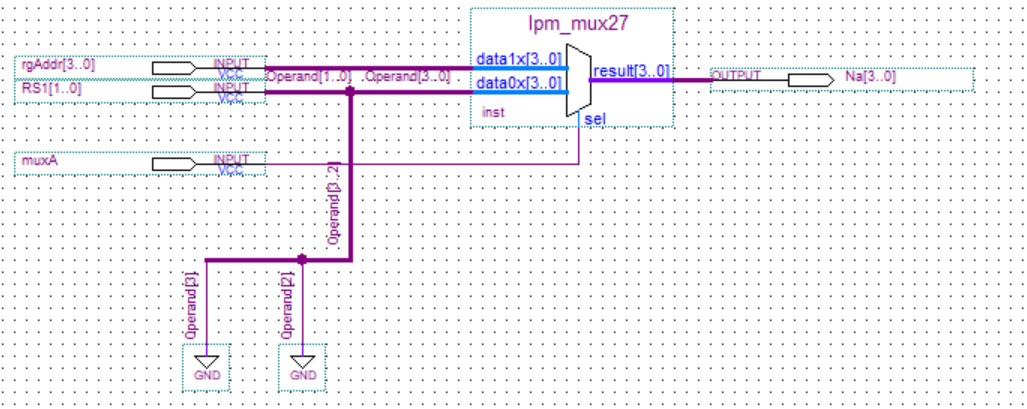
Абсолютный переход на команду XCH:



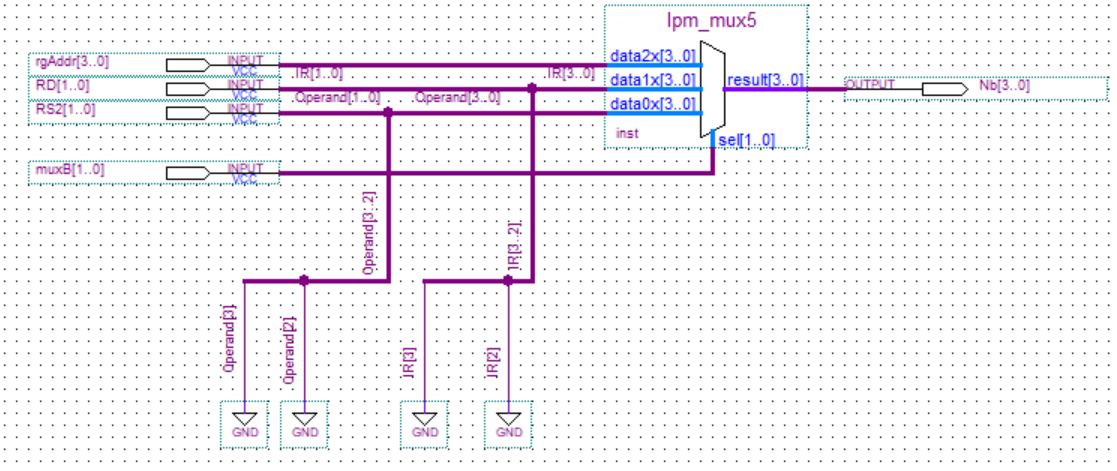
1. **Общая схема контролирующего устройство**

Схема:

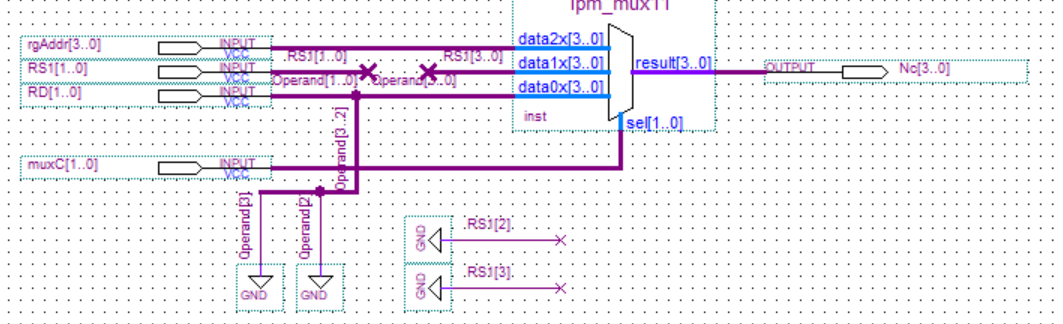


1. **Схема muxA:**

|  |  |
| --- | --- |
| Na | muxA |
| rgAddres(предустановленный из команды МК-кода) | 1 |
| RS1 | 0 |

1. **Схема muxB:**

|  |  |
| --- | --- |
| Nb | muxB |
| rgAddres(предустановленный из команды МК-кода) | 10 |
| RD | 01 |
| RS2 | 00 |

1. **Схема muxC:**

|  |  |
| --- | --- |
| Nc | MuxC |
| rgAddres(предустановленный из команды МК-кода) | 10 |
| RS1 | 01 |
| RD | 00 |

Схема topA:

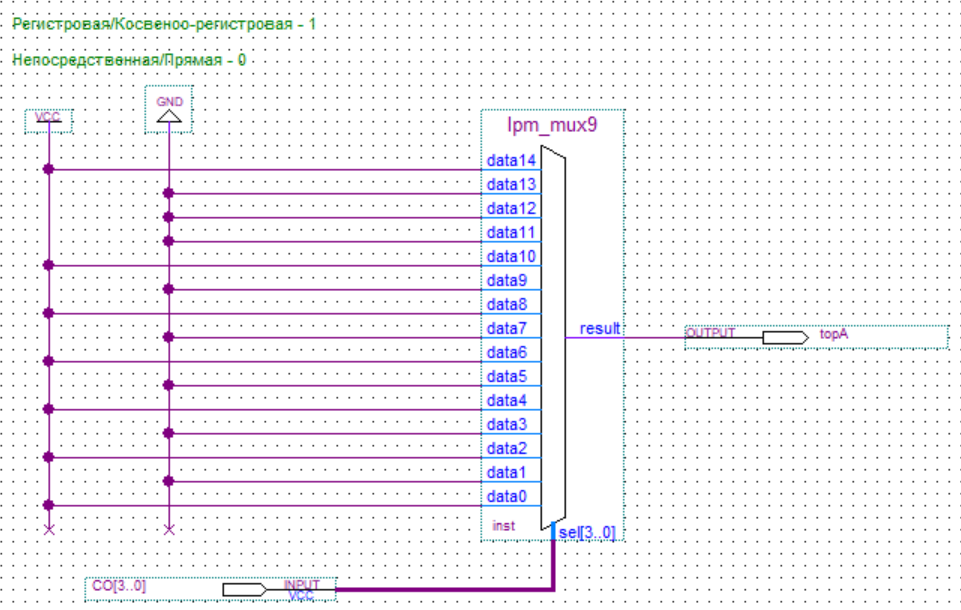
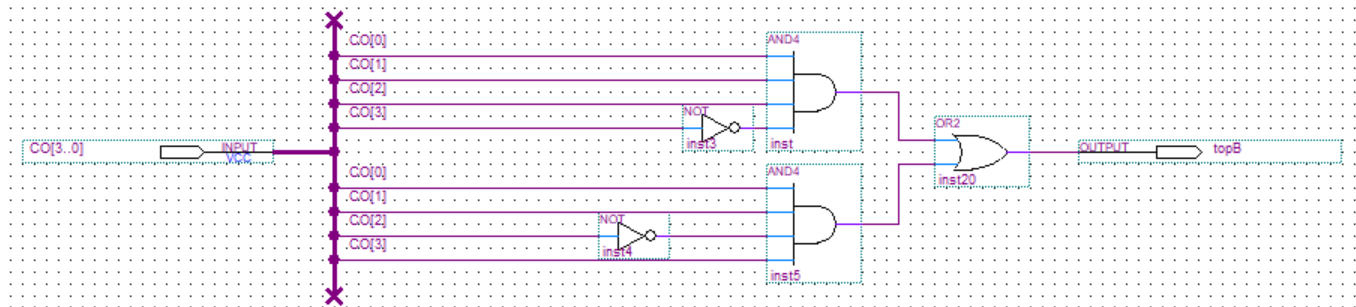
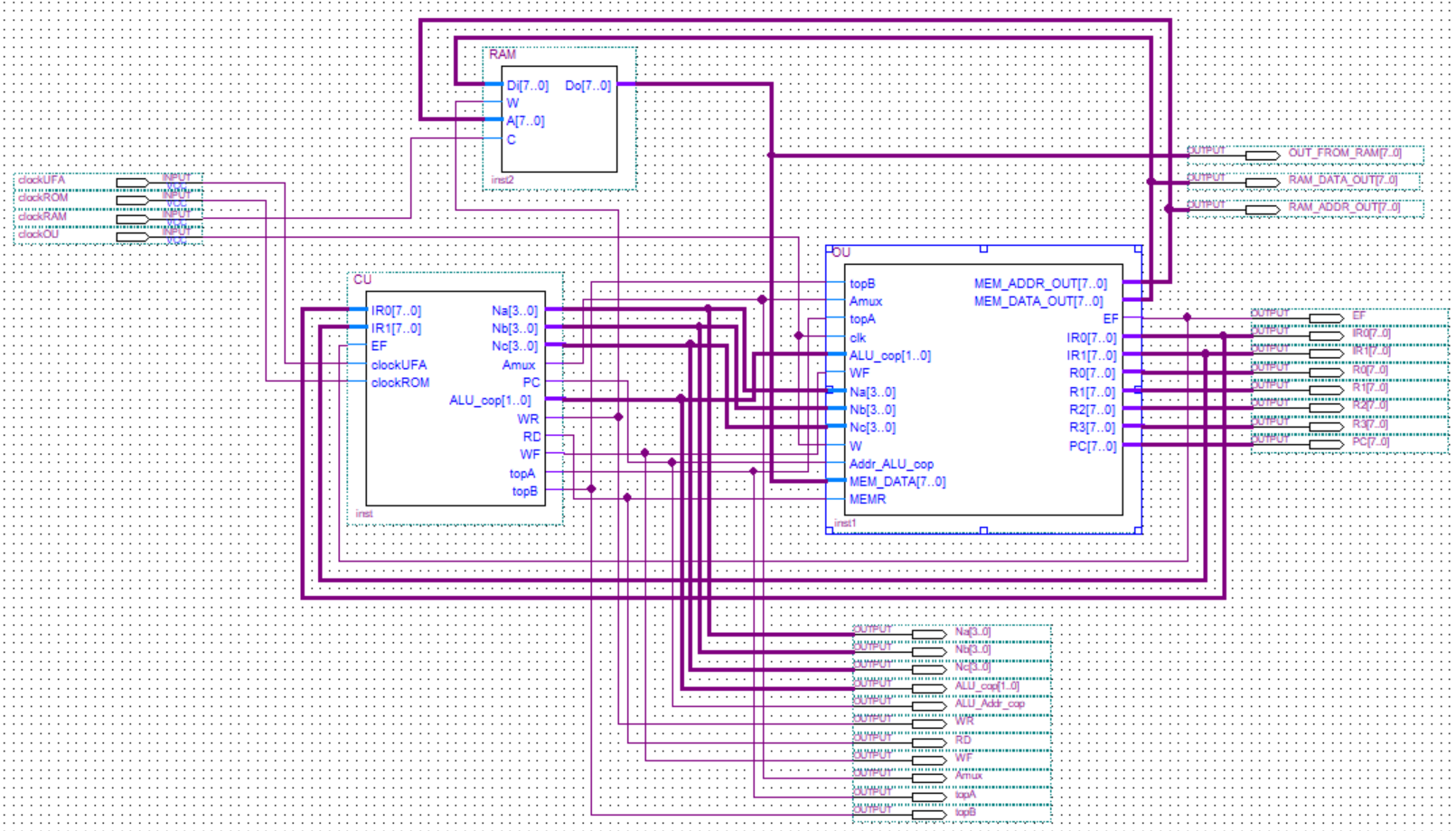


Схема topB:

Необходим для работы команд ADD rd, rs1, const и NAND rd, rs1, const.

Только шина A у ALU имеет доступ к константе, содержащеся во втором слове команды, необходимо чтобы на шине B ALU появился операнд rs1, который находится в первом слове , данное устройсво отвечает за перенаправление первого операнда на шину B.

******Общая схема процессора**

**Микрокод**

rs1 - muxA (A - шина адреса) (B - шина данных)

rd/rs2 - muxB

rd/rs1 – muxC (muxC расширен на rs1 для xch)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | Amux | B | Bmux | C | Cmux | PC | ALU | WR | RD | Cond | JumpAddr | WF |
| Кол-во бит | 4 | 1 | 4 | 2 | 4 | 2 | 1 | 2 | 1 | 1 | 3 | 6 | 1 |

Всего **32 бита** на одну микрокоманду:

*Команды:*

LD, ST, ADD, SHRA, JE, NAND, JMP, XCH

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Мнем. зап | A | Amux | B | Bmux | C | Cmux | PC | ALU | WR | RD | Cond  (ACode) | JAddr | WF |
| 0 | IR0<- mem(PC) | PC | 1 | \* | \* | IR0 | 10 | 1 | \* | 0 | 1 | j1b | 2 | 0 |
| 1 | IR1<-mem(PC) | PC | 1 | \* | \* | IR1 | 10 | 1 | \* | 0 | 1 | next | \* | 0 |
| 2 | decode(IR0) | \* | \* | \* | \* | tmp2 | 10 | 0 | set | 0 | 0 | decode | \* | 0 |
| **LD**  ld R0, R1 – регистровая  ld R0, const – непосредственная | | | | | | | | | | | | | | |
| 3 | Rd<-set(Rs1) | \* | 0 | \* | \* | \* | 00 | 0 | set | 0 | 0 | jmp | 0 | 0 |
| ld R0, addr - прямая  ld R0, [R1] – косвенно-регистровая | | | | | | | | | | | | | | |
| 4 | Rd<-mem(rs1) | \* | 0 | \* | 00 | \* | \* | 0 | \* | 0 | 1 | jmp | 0 | 0 |
| **ST**  st R0, addr - адресные  st R0, [R1] - косвенно-регистровая | | | | | | | | | | | | | | |
| 5 | mem(addr) <- R0 | \* | 0 | \* | 01 | tmp0 | 10 | 0 | set | 1 | 0 | jmp | 0 | 0 |
| **ADD**  add R0, R1, R2 – регистровая  add R0, R1, const – непосредственная | | | | | | | | | | | | | | |
| 6 | rd<-add(rs1, rs2/const) | \* | 0 | \* | 00 | \* | 00 | 0 | add | 0 | 0 | jmp | 0 | 1 |
| **SHRA**  shra R0, R1 – регистровая  shra R0, const – непостредственная | | | | | | | | | | | | | | |
| 7 | Rd<-shra(Rs1/const) | \* | 0 | \* | \* | \* | 00 | 0 | shra | 0 | 0 | jmp | 0 | 1 |
| **JE**  je addr | | | | | | | | | | | | | | |
| 8 | tmp0<-set(tmp0) | Tmp0 | 1 | \* | \* | Tmp0 | 10 | 0 | set | 0 | 0 | jne | 0 | 0 |
| 9 | PC<-set(addr) | \* | 0 | \* | \* | PC | 10 | 0 | set | 0 | 0 | jmp | 0 | 0 |
| **NAND**  nand R0, R1, R2 – регистровая  nand R0, R1, const – непосредственная | | | | | | | | | | | | | | |
| 10 | rd<-nand(rs1, rs2/const) | \* | 0 | \* | 00 | \* | 00 | 0 | nand | 0 | 0 | jmp | 0 | 1 |
| **JMP**  jmp addr | | | | | | | | | | | | | | |
| 11 | PC<-set(addr) | \* | 0 | \* | \* | PC | 10 | 0 | set | 0 | 0 | jmp | 0 | 0 |
| **XCH**  xch R0, R1 | | | | | | | | | | | | | | |
| 12 | Tmp0<-set(R1) | \* | 0 | \* | \* | Tmp0 | 10 | 0 | set | 0 | 0 | next | \* | 0 |
| 13 | R1<-add(R0, Rg0) | Rg0 | 1 | \* | 01 | \* | 01 | 0 | add | 0 | 0 | next | \* | 0 |
| 14 | R0<-set(tmp0) | Tmp0 | 1 | \* | \* | \* | 00 | 0 | set | 0 | 0 | jmp | 0 | 0 |

**Тесты**

WIDTH=8;

DEPTH=6;

ADDRESS\_RADIX=UNS;

DATA\_RADIX=BIN;

CONTENT BEGIN

0 : 00010000; -- ld r0, #22

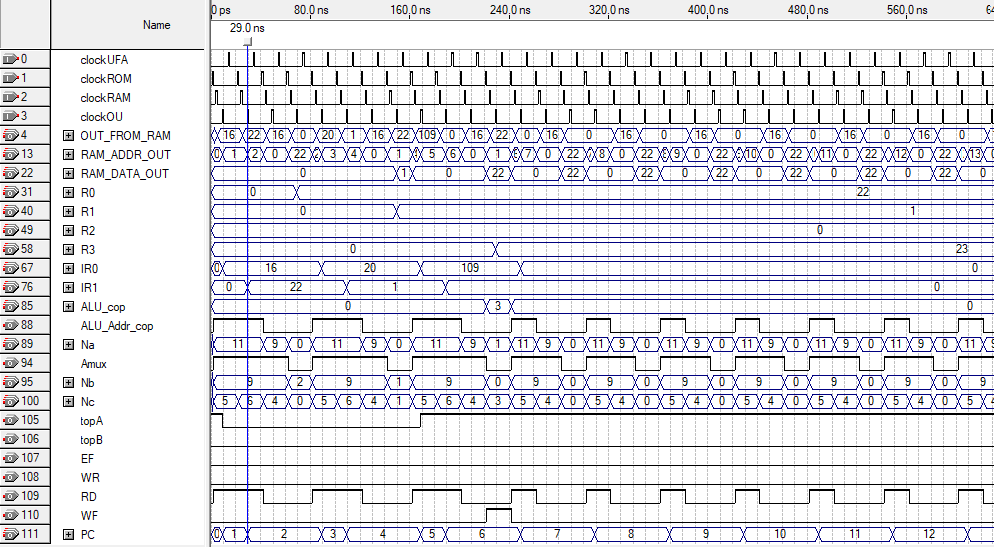
1 : 00010110;

2 : 00010100; -- ld r1, #1

3 : 00000001;

4 : 01101101; -- add r3, r1,r0

5 : 00000000;

END; -- exp: R3 == 23

WIDTH=8;

DEPTH=5;

ADDRESS\_RADIX=UNS;

DATA\_RADIX=BIN;

CONTENT BEGIN

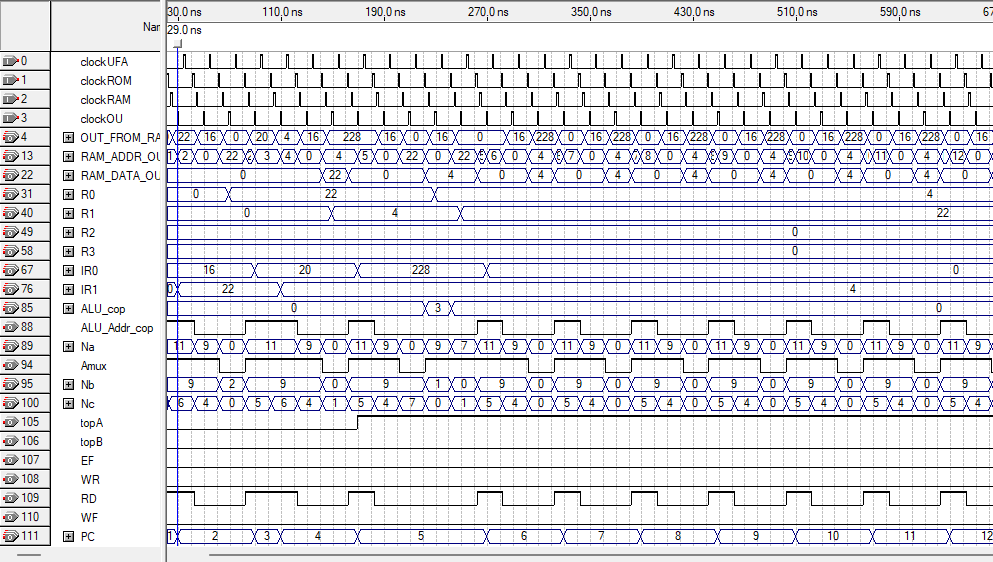
0 : 00010000; -- ld r0, #22

1 : 00010110;

2 : 00010100; -- ld r1, 4

3 : 00000100;

4 : 11100100; -- xch r1, r0

END; -- exp: R0 == 4 && R1 == 22

CONTENT BEGIN

0 : 00010000; -- ld r0, 7

1 : 00000111;

2 : 00010100; -- ld r1,2

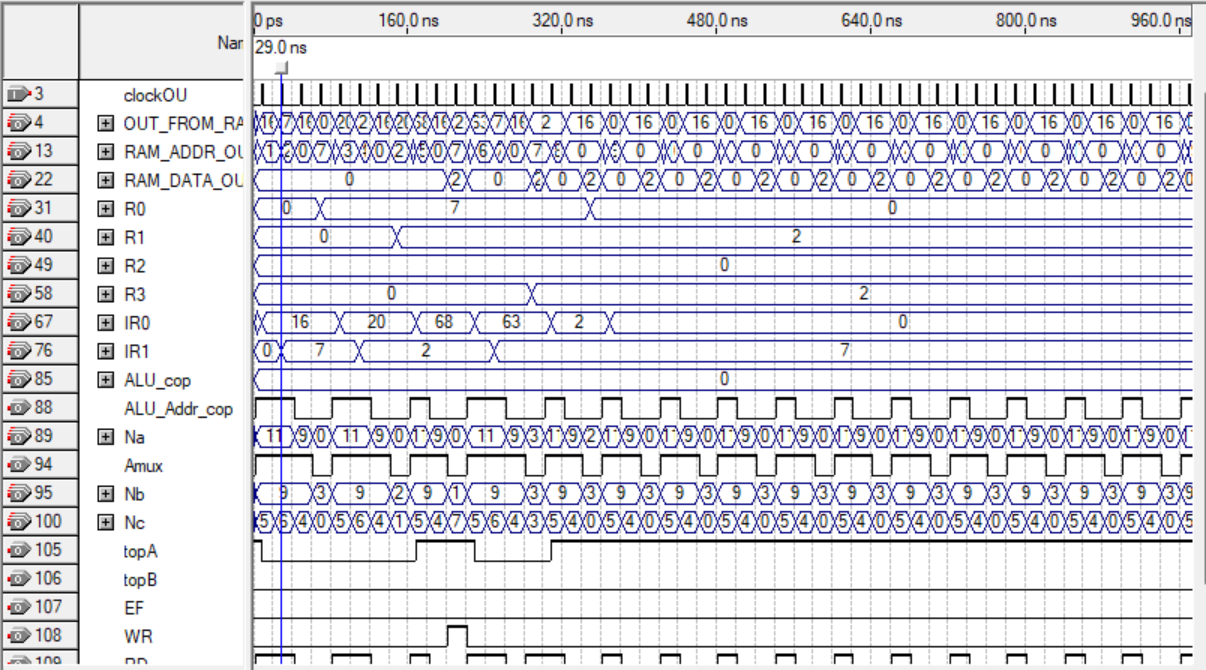
3 : 00000010;

4 : 01000100; -- st r1, [r0] // записываем значение из r1 по адресу содержащемся в r0, т.е по адрессу 111(7)

5 : 00111111; -- ld r3, 111 // записываем значение в r3 из адрема 111

6 : 00000111;

END; -- expected: R3 == 2



WIDTH=8;

DEPTH=7;

ADDRESS\_RADIX=UNS;

DATA\_RADIX=BIN;

CONTENT BEGIN

0 : 00010000; - ld r0, 2;

1 : 00000010;

2 : 11010000; - jmp 6;

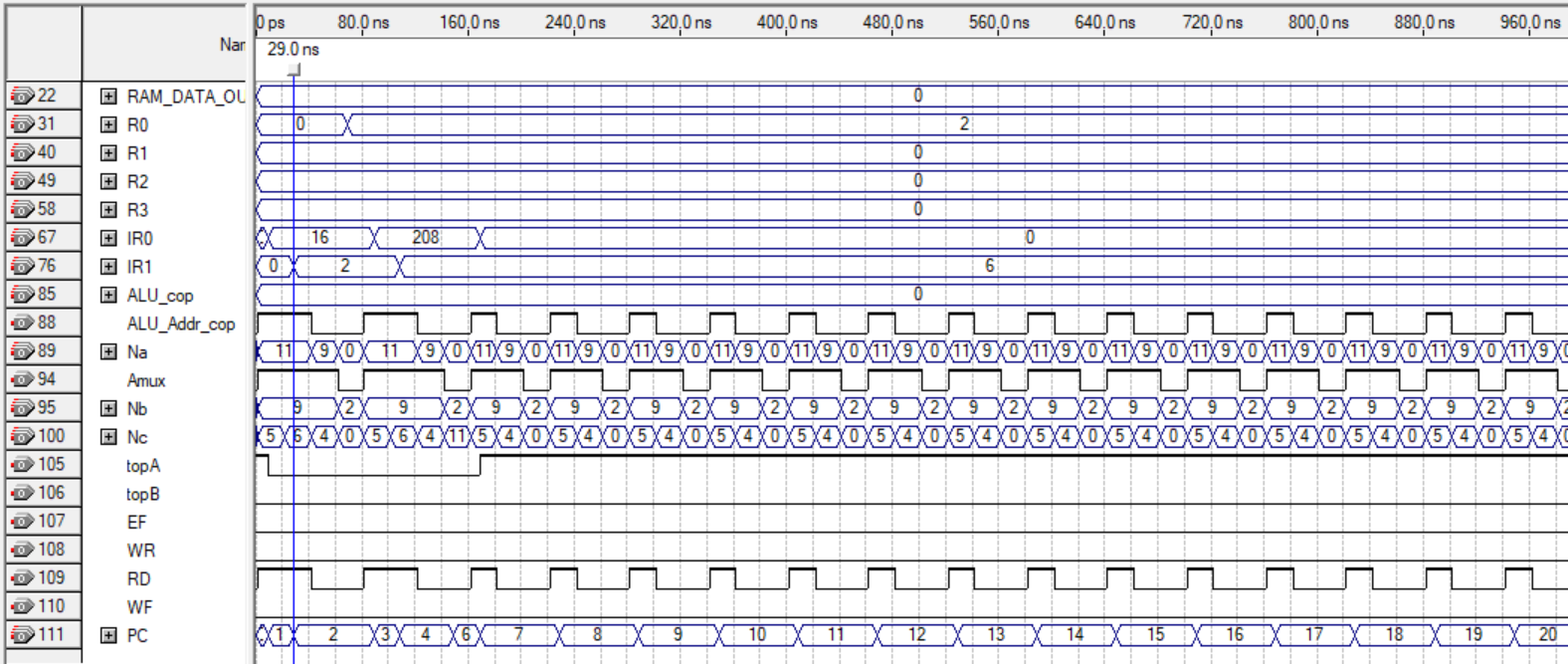
3 : 00000110;

4 : 01110000; add r0,r0,2

5 : 00000010;

6 : 00000000;

END; // значение r0 остается равным 2, значит сложение на строке 4 не просходит, т.к был выполнен прыжок сразу на 6 строку



WIDTH=8;

DEPTH=8;

ADDRESS\_RADIX=UNS;

DATA\_RADIX=BIN;

CONTENT BEGIN

0 : 00010000; -- ld r0, 1

1 : 00000001;

2 : 01111000; -- add r2, r0, 1

3 : 00000001;

4 : 11000000; -- je 7

5 : 00000111; -- 7

6 : 11100010; -- xch r0,r2

7 : 00000000;

END; -- exp: r0 и r2 не должны поменять значения, тк отработает ef флаг(из-за сложение r0, где находится 1, и 1 во 2 строке)

