

Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 «Программная инженерия»

О Т Ч Е Т по лабораторной работе N 2

Название Изучение принципов работы	микропроцессорного ядра	RISC-V	
Дисциплина Архитектура элекронно-вы	ычислительных машин		
Студент:		Ковель А.Д.	
Отудент.	подпись, дата	Фамилия, И.О.	
Преподаватель:	полиись пата	Попов А. Ю. Фамилия И О	

Цель работы

Основной целью работы является ознакомление с принципами функционирования, построения и особенностями архитектуры суперскалярных конвейерных микропроцессоров. Дополнительной целью работы является знакомство с принципами проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС.

Основные теоретические сведения

RISC-V является открытым современным набором команд, который может использоваться для построения как микроконтроллеров, так и высокопроизводительных микропроцессоров.

В данной работе исследуется набор команд RV32I, который включает в себя основные команды 32-битной целочисленной арифметики кроме умножения и деления.

Набор команд RV32I предполагает использование 32 регистров общего назначения x0-x31 размером в 32 бита каждый и регистр рс, хранящего адрес следующей команды. Все регистры общего назначения равноправны, в любой команде могут использоваться любые из регистров. Регистр рс не может использоваться в командах.

Архитектура RV32I предполагает плоское линейное 32-х битное адресное пространство. Минимальной адресуемой единицей информации является 1 байт. Используется порядок байтов от младшего к старшему (Little Endian), то есть, младший байт 32-х битного слова находится по младшему адресу (по смещению 0). Отсутствует разделение на адресные пространства команд, данных и ввода-вывода. Распределение областей памяти между различными устройствами (ОЗУ, ПЗУ, устройства ввода-вывода) определяется реализацией.

Большая часть команд RV32I является трехадресными, выполняющими операции над двумя заданными явно операндами, и сохраняющими результат в регистре. Операндами могут являться регистры или константы, явно заданные в коде команды. Операнды всех команд (кроме команды ацірс) задаются явно.

Архитектура RV32I, как и большая часть RISC-архитектур, предполагает разделение команд на команды доступа к памяти (чтение данных из памяти в регистр или запись данных из регистра в память) и команды обработки данных в регистрах.

Общая для всех вариантов программа

Исследуемая программа

Исходный текст исследуемой программы представлен на листинге 1.

Листинг 1 – Исходный текст общей программы

```
1 section text (1)
 2 globl start; (2)
 3 \mid \text{len} = 8 \# \text{size of array } (3)
 |4| enroll = 4 #amount of processed elements in one iteration
 5 elem sz = 4 \#size of one element in array
 6 start: (4)
 7 addi \times 20, \times 0, len/enroll (5)
8 | 1a \times 1, \times (6) |
 9 loop:
10 | \text{Iw } \times 2, 0(\times 1) (7)
11 add x31, x31, x2 (8)
12 | \text{Iw } \times 2, 4(\times 1) |
13 add x31, x31, x2
14 lw x2, 8(x1)
15 add x31, x31, x2
16 | \text{Iw } \times 2, 12(\times 1) |
17 add x31, x31, x2
18 addi x1, x1, elem sz*enroll (9)
19 addi \times 20, \times 20, -1 (10)
20 bne \times 20, \times 0, loop (11)
21 addi x31, x31, 1
22 forever: j forever (12)
23
24 section data (13)
            .4 byte 0x1 (14)
25 x:
26 .4 byte 0x2
27 .4 byte 0x3
28 .4 byte 0x4
29 .4 byte 0x5
30 .4 byte 0x6
31 .4 byte 0x7
32 .4 byte 0x8
```

Дизассемблерный листинг исследуемой программы представлен на листинге 2.

Листинг 2 – Дизассемблерный листинг общей программы

```
80000000 < start >:
1
2
       80000000:
                    00200a13
                                           addi
                                                    x20,x0,2
3
                    00000097
                                                    x1,0x0(1)
       80000004:
                                           auipc
                                                    \times 1, \times 1, 60 # 80000040 <_x>
4
       80000008:
                    03c08093
                                           addi
5
6
       8000000c <loop >:
7
                                               x2,0(x1)
       8000000c:
                    0000a103
                                           lw
8
       80000010:
                    002f8fb3
                                           add x31,x31,x2
9
                                           lw x2, 4(x1)
       80000014:
                    0040a103
                                           add x31,x31,x2
10
       80000018:
                    002f8fb3
11
       8000001c:
                    0080a103
                                           lw x2,8(x1)
                                           add x31,x31,x2
12
       80000020:
                    002f8fb3
                                           lw x2,12(x1)
13
       80000024:
                    00c0a103
14
       80000028:
                    002f8fb3
                                           add x31,x31,x2
                                                    \times 1 , \times 1 , 16
15
       8000002c:
                    01008093
                                           addi
                                                    x20, x20, -1
16
       80000030:
                    fffa0a13
                                           addi
       80000034:
                    fc0a1ce3
                                           bne x20, x0, 8000000c < loop >
17
18
       80000038:
                    001f8f93
                                           addi
                                                    x31,x31,1
19
20
       8000003c <forever >:
21
       8000003c:
                    0000006 f
                                           jal x0,8000003c <forever>
```

Можно сказать, что данная программа эквивалентна псевдокоду на языке С, представленному на листинге 3.

Листинг 3 – Псевдокод общей программы

```
1 #define len 8
2 #define enroll 4
3 #define elem sz 4
4 int _x[]=\{1,2,3,4,5,6,7,8\};
  void _start() {
6
       int x20 = len/enroll;
7
       int *x1 = _x;
8
      do {
9
10
           int x2 = x1[0];
11
           x31 += x2;
           x2 = x1[1];
12
           x31 += x2;
13
14
           x2 = x1[2];
           x31 += x2;
15
           x2 = x1[3];
16
17
           x31 += x2;
18
           x1 += enroll;
           x20--;
19
       } while(x20 != 0);
20
21
       x31++;
22
       while(1)\{\}
23 }
```

Результаты исследования программы

Все задания выполнялись по индивидуальному варианту (6).

Скриншот, полученный в ходе выполнения задания №2 (получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с адресом 8000020 на первой итерации) представлен на рисунке 1.

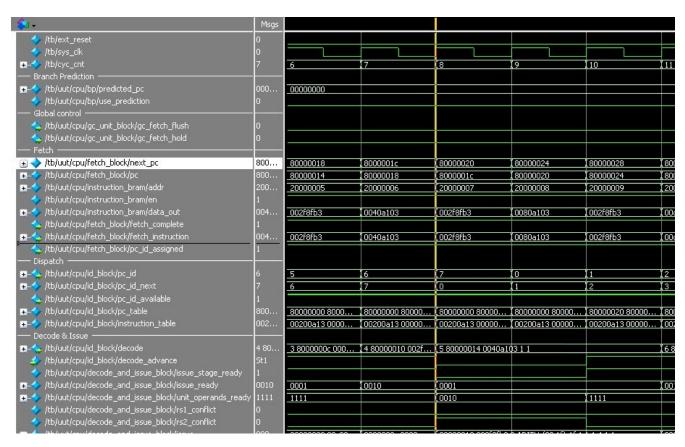


Рисунок 1 — Временная диаграмма выполнения стадий выборки и диспетчеризации

Скриншот, полученный в ходе выполнения задания №3 (получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 8000002с на первой итерации) представлен на рисунке 2.

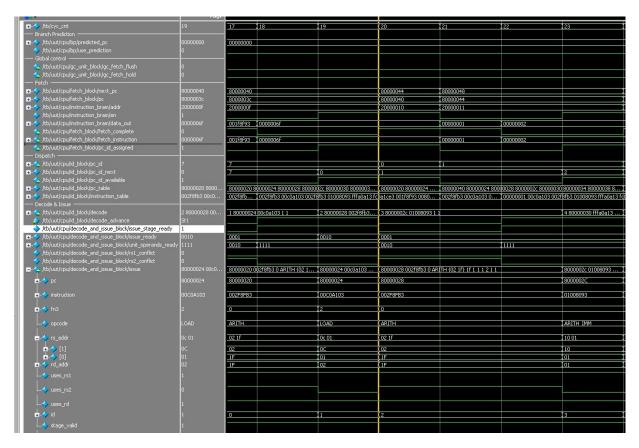


Рисунок 2 — Временная диаграмма выполнения стадии декодирования и планирования на выполнение

Скриншот, полученный в ходе выполнения задания \mathbb{N}^4 (получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000014 на первой итерации) представлен на рисунке 3.

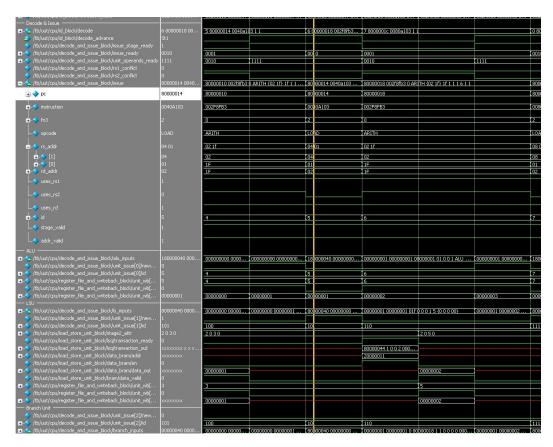


Рисунок 3 – Временная диаграмма выполнения стадии выполнения

Программа по варианту

Все задания выполнялись по индивидуальному варианту (6).

Исследуемая программа

Исходный текст исследуемой программы представлен на листинге 4.

Листинг 4 – Исходный текст исследуемой программы

```
.section .text
 1
 2
        .globl _start;
 3
        len = 8 \# size of array
 4
        enroll = 2 #amount of processed elements in one iteration
 5
        elem sz = 4 \# size of one element in array
 6
 7
        start:
 8
        addi x20, x0, len/enroll
        la x1, _x
 9
10
        lp:
       lw \times 2, 0(\times 1)
11
       lw x3, 4(x1) #!
12
13
        addi x1, x1, elem sz*enroll
        addi \times 20, \times 20, -1
14
       add x31, x31, x2
15
       add x31, x31, x3
16
17
       bne x20, x0, lp
        addi x31, x31, 1
18
19
       lp2: j lp2
20
21
        .section .data
22
       x:
                 .4 byte 0x1
23
       .4 byte 0x2
24
       .4 byte 0x3
25
        .4 byte 0x4
26
       .4 byte 0x5
27
       .4 byte 0x6
28
        .4 \, \text{byte} \, 0 \times 7
29
        .4 byte 0x8
```

Дизассемблерный листинг исследуемой программы представлен на листинге 5.

Листинг 5 – Дизассемблерный листинг исследуемой программы

1	80000000 <_st	:art >:		
2	80000000:	00400a13	addi	×20 , ×0 , 4
3	8000004:	00000097	auipc	×1,0×0
4	8000008:	02c08093	addi	×1,×1,44 # 80000030
	<_x>			
5				
6	8000000c <lp></lp>	> :		
7	8000000c:	0000a103	lw	x2,0(x1)
8	8000010:	0040a183	lw	x3,4(x1)
9	80000014:	00808093	addi	×1 , ×1 , 8
10	80000018:	fffa0a13	addi	×20 , ×20 , —1
11	8000001c:	002 f8fb3	add	x31 , x31 , x2
12	80000020:	003f8fb3	add	x31 , x31 , x3
13	80000024:	fe0a14e3	bne	x20,x0,8000000c <lp></lp>
14	80000028:	001 f8f93	addi	×31 , ×31 ,1
15				
16	8000002c < lp2	>:		
17	8000002c:	0000006 f	jal	x0,8000002c <lp2></lp2>

Можно сказать, что данная программа эквивалентна псевдокоду на языке С, представленному на листинге 6.

Листинг 6 – Псевдокод исследуемой программы

```
#define len 8
 1
 2
       #define enroll 2
 3
       #define elem sz 4
       int _{x}[]=\{1,2,3,4,5,6,7,8\};
4
 5
       void _start() {
6
           int x20 = len/enroll;
 7
           int *x1 = _x;
8
9
           do {
10
                int x2 = x1[0];
11
                int x3 = x1[1];
12
                x1 += enroll;
13
                ×20 ——;
14
                x31 += x2;
15
                x31 += x3;
16
           } while (x20 != 0);
           x31++;
17
18
           while (1) {}
19
       }
```

Трасса работы программы

Трасса работы программы представлена на рисунке 4.

Код команды Команда	id																							_			pτ			_			_	_	_	_	_		_		_	_
Команда			2 3		5 6	7	8	9	10 1	11 1	12 1	13 1	4 1	5 1	6 17	18	19	20	21	22 2	3 24	4 25	26	27						34	35 3	6 37	38 3	19 40	41	42	43 4	14 4	5 40	6 47	7 48	3
00400a13 addi x20,x0,4	0		D D	0.000																																						ı
00000097 *auip c x1,0x0	1		F ID																																						┸	
02c08093 addi x1,x1,44 # 80000030 <_x>	2		F	ID I																																						
0000a103 1w x2,0(x1)	3			FI	D D	M1	M2	М3																																		
0040a183]w x3,4(x1)	4				F II	D D	M1	M2	МЗ																																	
808093 addi x1,x1,8	5				F	ID	D	AL							Т	П			П			Т	П		\neg	Т		Т		П					П	П			Т	Т	Т	
fffa0a13 addi x20,x20,-1	6					F	ID	D	AL																																	
002f8fb3 add x31,x31,x2	7	Т			Т	Т	F	ID	D	AL	Т	Т	Т	Т	Т	П			П		Т	Т			П	Т	Т	Т		П		т		Т	П	П	П	Т	Т	Т	Т	
003f8fb3 add x31,x31,x3	0							F	ID	DA	AL																															
fe0a14e3	1	Т			Т	Т	П		F I	ID	D	В				П			\neg		Т	Т			\neg	Т				П		т		Т	П		\Box	Т	Т	т	Т	
001f8f93 addi x31,x31,1	2											D)	X																													
0000006f	3						П		т			[D D				П			т			Т	Т		Т	Т						т		т	П	П		т	т	т	т	
1	4											F)																														Ì
0000a103 lw x2,0(x1)	3	7					П		_	т				F I	D D	М1	M2	МЗ	т			т	П			т				П				т	П			т	т	т	т	
0040a183 Nw x3.4(x1)	4																	M2 I	МЗ																							j
808093 addi x1.x1.8	5								7							ID																			П				т	т	т	
fffa0a13 addi x20,x20,-1	6														i.			D	ΔI																							í
002f8fb3 add x31.x31.x2	7																	ID		۸۱																				-		
003f8fb3 add x31,x31,x3	0																			D A	1																					į
fe0a14e3 None x20,x0,8000000c fe0a14e3 None x20,x0,800000c fe0a14e3 None x20,x0,800000c fe0a14e3 None x20,x0,80000c fe0a14e3 None x20,x0,80000c fe0a14e3 None x20,x0,80000c fe0a14e3 None x20,x0,80000c fe0a14e3 None x20,x0,8000c fe0a14e3 None x20,x0,800c fe	1																			ID																		+	+	+		
0000a103 Nw x2,0(x1)	2																			FI			MO	MO																		į
~~	3																				FIL				мэ							-						-	+	+	-	j
300	4																			,			D D		M3							-		-							-	
30000	4																				1			D								-		-				-	+	+		į
	5																					F										_		+	\vdash		_	+	_	_	-	
002f8fb3	6							Н															F	ID										-					-	#		į
003f8fb3 add x31,x31,x3	- 1/1	-						ш						-					_						ID							_		+	\vdash		_	_	_	_	-	١
fe0a14e3 bne x20,x0,8000000c <lp></lp>	0																								F		D B												-	-		į
0000a103 x2,0(x1)	1	_						ш	_	_									_								D D					_		_	\vdash		_	_	_	_	_	
0040a183 \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	2																									-	F II				43											į
808093 addi x1,x1,8	3	_																	_						_	4	F	ID				_		_	ш		_	_	_	_	_	
fffa0a13 addi x20,x20,-1	4																											F		D.												l
002f8fb3 add x31,x31,x2	5							Ш																		_			F		D A			_	\perp	ш	\perp	_	_	┸	┸	
003f8fb3 add x31,x31,x3	6) AL										l
fe0a14e3 bne x20,x0,8000000c <lp>bne x20,x0,8000000c <lp>bne x20,x0,8000000c <lp>bne x20,x0,8000000c <lp>bne x20,x0,8000000c <lp>bne x20,x0,8000000c <lp>bne x20,x0,8000000c</lp></lp></lp></lp></lp></lp>	7							Ш																		_						D D	-								┸	
0000a103 × 1,w x2,0(x1)	0																														F		DG	~								
0040a183 → 1, ₩ x3,4(x1)	1																																ID								┸	
808093 addi x1,x1,8	2																																F	X								
001f8f93 ddi x31,x31,1	1																																	F		D						
0000006f jal x0,8000002c <lp2></lp2>	2																																		F	ID	DI	В				
1	3																																				ID I					
2	4																																				FI	D D	X			
3	5																																				1	Fχ	4			
0000006f jal x0,8000002c <lp2></lp2>	4																																						F	II	D	ı
Код команды	1,0	1	2 3	4 !	5 6	7	8	9	10	11 1	12 1	13 1	4 1	5 1	6 17	18	19	20	21	22 2	3 24	4 25	26	27	28 2	29 3	0 3	1 32	33	34	35 3	6 37	38 3	19 40	41	42	43/4	144	5 47	547	48	3
***	x0,8000002c <lp2> Команда</lp2>		1	1 2 3	1 2 3 4	1 2 3 4 5 6	1 2 3 4 5 6 7	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10 11	1 2 3 4 5 6 7 8 9 10 11 12	1 2 3 4 5 6 7 8 9 101112131	1 2 3 4 5 6 7 8 9 1011 1213 141	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	1 2 3 4 5 6 7 8 9 101112131415161718	1 2 3 4 5 6 7 8 9 1011 1213 14 15 16 17 18 19	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 101112131415161718192021	1 2 3 4 5 6 7 8 9 101112131415161718192021222	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 2	1 2 3 4 5 6 7 8 9 10111213141516171819202122232425	1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	1 2 3 4 5 6 7 8 9 10111213141516171819202122232425262728	1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526272829	1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526272829303	1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526272829303132	1 2 3 4 5 6 7 8 9 101112131415161718192021222324252627282930313233	1 2 3 4 5 6 7 8 9 101112131415161718192021222324252627282930313233334	1 2 3 4 5 6 7 8 9 10111213141516171819202122232425262728293031323334353	1 2 3 4 5 6 7 8 9 1011121314151617181920212212425262728293031323334353637	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 3	1 2 3 4 5 6 7 8 9 10111213141516171819202122232425262728293031322334353637383940	1 2 3 4 5 6 7 8 9 1011121314151617181920212223242526272829303132333435363738394041	1 2 3 4 5 6 7 8 9 101112131415161718192021222324252627282930313233343536373839404142	Команда id 1 2 3 4 5 6 7 8 9 1911 1213 14 15 16 17 18 19 2921 22 23 24 25 26 27 28 29 39 31 32 33 34 35 36 37 38 39 40 41 42 43 4	Команда dd 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 44	1 2 3 4 5 6 7 8 9 10111213141516171819202122232425[26]27[28]29[30]3132[33]34[35]36[37]38[39]40[41]42[43]44[45]47	Команда id 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 45	Команда 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 (46 5) 45 (45 45)

Рисунок 4 – Трасса работы программы

Временные диаграммы

Временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #! (80000010: 0040a183 lw x3,4(x1)), представлены на рисунке 5.

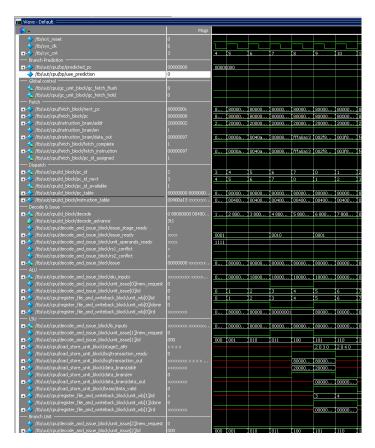


Рисунок 5 – Временные диаграммы сигналов

Вывод об эффективности работы программы

Как видно на трассе работы программы, представленой на рисунке 4, конфликта по регистрам не возникает ни разу, и все команды принимаются на исполнение сразу же по готовности. Это обеспечивается продуманным чередованием команд обращения к памяти и арифметических команд.

Можно также заметить, что трижды возникает ситуация ошибочной выборки, которая крайне негативно сказывается на производительности, так как приводит к необходимости очистки конвеера, и ожидания прохождения новой команы по всем стадиям работы. Однако, учитывая большое количество ветвлений в данной программе (только сам цикл повторяется 4 раза), можно сказать, что предсказаель ветвлений ошибается не слишком часто.

Оптимизировать же программу можно, уменьшив количество циклов в ней. В исходном коде цикл повторяется 4 раза, и некоторые команды, не относящиеся напрямую к вычислению суммы элементов массива (во-первых, команда условного перехода bne x20, x0, lp; во-вторых, декремент счетчика цикла addi x20, x20,

-1; в-третьих, смещение указателя на начало массива: addi x1, x1, elem_sz*enroll) повторяются 4 раза.

Если же уменьшить количество циклов до двух (за счет увеличения количества обрабатываемых элементов массива за одну итерацию до 4), то эти команды выполнятся всего 2 раза. При правильной и продуманной последовательности операций обращения к памяти и выполнения арифметических вычисслений, все еще удастся избежать конфликтов по регистрам.

В итоге можно обеспечить выигрыш в 3*2=6 тактов, и оптимизированная программа будет работать на 6/49=13% бысрее.

Оптимизированная программа

Исходный текст оптимизированной программы представлен на листинге 7.

Листинг 7 – Исходный текст оптимизированной программы

```
.section .text
 1
 2
       .globl start;
 3
       len = 8
       enroll = 4
 4
 5
       elem sz = 4
 6
 7
       start:
8
       addi x20, x0, len/enroll
9
       la x1, _x
10
       lp:
       lw x2, 0(x1)
11
12
       lw x3, 4(x1) #!
13
       addi \times 20, \times 20, -1
       add x31, x31, x2
14
15
       add x31, x31, x3
16
       lw x2, 8(x1)
17
       lw x3, 12(x1)
18
19
       addi x1, x1, elem_sz*enroll
       add x31, x31, x2
20
       add x31, x31, x3
21
22
23
       bne x20, x0, lp
```

```
24
       addi\ x31\,,\ x31\,,\ 1
25
       lp2: j lp2
26
27
       .section .data
28
                .4 byte 0x1
       .4 byte 0x2
29
30
       .4 byte 0x3
       .4 byte 0x4
31
       .4 byte 0x5
32
33
       .4 byte 0x6
34
       .4 byte 0x7
35
       .4 byte 0x8
```

Дизассемблерный листинг оптимизированной программы представлен на листинге 8.

Листинг 8 – Дизассемблерный листинг оптимизированной программы

	1 1	T T	рванной программы
80000000 <_s	tart>:		
80000000:	00200a13	addi	x20 , x0 , 2
80000004:	00000097	auipc	×1,0×0
80000008:	03c08093	addi	x1,x1,60 # 80000040
<_x>			
8000000c <lp>3</lp>	>:		
8000000c:	0000a103	lw	x2,0(x1)
80000010:	0040a183	lw	×3 ,4(×1)
80000014:	fffa0a13	addi	×20 , ×20 , —1
80000018:	002f8fb3	add	x31 , x31 , x2
8000001c:	003f8fb3	add	x31 , x31 , x3
80000020:	0080a103	lw	×2,8(×1)
80000024:	00c0a183	lw	x3,12(x1)
80000028:	01008093	addi	×1,×1,16
8000002c:	002f8fb3	add	x31 , x31 , x2
80000030:	003f8fb3	add	x31 , x31 , x3
80000034:	fc0a1ce3	bne	x20,x0,8000000c <lp></lp>
80000038:	001f8f93	addi	x31 , x31 , 1
8000003c < lp2	:>:		
8000003c:	0000006 f	jal	x0,8000003c <lp2></lp2>
	80000000: 80000004: 80000008: <x> 8000000c < lp : 8000000c: 80000010: 80000014: 8000001c: 8000001c: 80000020: 80000020: 8000002c: 80000030: 80000030: 80000030: 8000003c < lp 2</x>	80000004: 00000097 80000008: 03c08093 <_x> 8000000c < p>: 8000000c: 0000a103 80000010: 0040a183 80000014: fffa0a13 8000001c: 003f8fb3 80000020: 0080a103 80000024: 00c0a183 80000024: 00c0a183 8000002c: 002f8fb3 8000002c: 002f8fb3 8000003c < p2>:	80000000: 00200a13 addi 80000008: 03c08093 addi <_x> 8000000c < lp >: 8000000c: 0000a103 lw 80000010: 0040a183 lw 80000014: fffa0a13 addi 80000018: 002f8fb3 add 8000001c: 003f8fb3 add 80000020: 0080a103 lw 80000024: 00c0a183 lw 80000028: 01008093 addi 8000002c: 002f8fb3 add 80000030: 003f8fb3 add 800003c add 80000034: fc0a1ce3 bne 8000003c 001f8f93 addi

Трасса работы программы

Трасса работы оптимизированной программы представлена на рисунке 6.

Адрес	Код команды	Команда	1d HOMED TAKTA 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40	41 42
300000000< start>	00400a13	addi x20,x0,4	0 F ID D AL	41 42
30000004	00000097	auipc x1,0x0	1 FIDDAL	\Box
30000008	02c08093	addi x1,x1,60 # 80000040 < x>	2 F ID D AL	
3000000c <lp></lp>	0000a103	lw x2,0(x1)	3 F ID D M1 M2 M3	\Box
30000010	0040a183	lw x3,4(x1)	4 F ID D M1 M2 M3	
30000014	fffa0a13	addi x20,x20,-1	5 FIDDAL	
30000018	002f8fb3	add x31,x31,x2	6 FIDDAL	
3000001c	002f8fb3	add x31,x31,x2	7 FIDDAL	
30000020	0080a103	lw x2,8(x1)	0 F ID D M1 M2 M3	
30000024	00c0a183	lw x3,12(x1)	1 F ID D M1 M2 M3	
30000028	1008093	addi x1,x1,16	2 F ID D AL	
3000002c	002f8fb3	add x31,x31,x2	3 F ID D AL	
30000030	003f8fb3	add x31,x31,x3	4 F ID D AL	
30000034	fc0alce3	bne x20,x0,8000000c <lp></lp>	5 FID D B	
30000038	001f8f93	addi x31,x31,1	6 FIDDX	
3000003c <lp2></lp2>	0000006f	ial x0,8000003c <lp2></lp2>	7 F ID DX	
'80000040 (ошиб.)			0 F X	
3000000c <lp></lp>	0000a103	lw x2,0(x1)	7 F ID D M1 M2 M3	
30000010	0040a183	lw x3,4(x1)	6 F ID D M1 M2 M3	
30000014	fffa0a13	addi x20,x20,-1	1 F ID D AL	
30000018	002f8fb3	add x31,x31,x2	2 F ID D AL	
3000001c	002f8fb3	add x31,x31,x2	3 F ID D AL	
30000020	0080a103	lw x2,8(x1)	4 F ID D M1 M2 M3	
30000024	00c0a183	w x3,12(x1)	5 F ID D M1 M2 M3	
30000028		addi x1,x1,16	6 FIDDAL	
3000002c	002f8fb3	add x31,x31,x2	7 F ID D AL	
30000030	003f8fb3	add x31,x31,x3	6 FIDDAL	
30000034	fc0a1ce3	bne x20,x0,8000000c <lp></lp>	1 F ID D B	
3000000c< l p>	0000a103	lw x2,0(x1)	2 FIDDX	
30000010	0040a183	w x3,4(x1)	3 FIDDX	
30000014	fffa0a13	addi x20,x20,-1	4 FX	
30000038	001f8f93	addi x31,x31,1	3 F ID D AL	
3000003c <lp2></lp2>	0000006f	jal x0,8000003c <lp2></lp2>	4 F ID D AL	
0000040 (дшиб.)			5	
10000044 (ошиб.)			6	
30000048 (дшиб.)			7	
8000003c <lp2></lp2>	0000006f	jal x0,8000003c <lp2></lp2>		ID D
Адрес	Код команды	Команда	1d 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40	41 42
имрес		Команда	Hower Takie	

Рисунок 6 – Трасса работы оптимизированной программы

Вывод

В результате выполнения лабораторной работы были изучены принципы функционирования, построения и особенности архитектуры суперскалярных конвейерных микропроцессоров.

Также были рассмотрены принципы проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС.

На основе изученных материалов был найден способ оптимизации программы.