

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»

ОТЧЕТ по лабораторной работе № 2

Название _ [/	Ізучение принципов работы микропро	оцессорного ядра	RISC-V
Дисциплина	а Архитектура элекронно-вычислител	ьных машин	
Студент:			Зайцева А. А.
Преподавате	ль:	подпись, дата ———————————————————————————————————	Фамилия, И.О. 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 auipc) 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

```
section text (1)
2 . globl _start; (2)
 3 \mid \mathsf{len} = 8 \# \mathsf{size} \mathsf{of} \mathsf{array} (3)
4 enroll = 4 #amount of processed elements in one iteration
 |\mathbf{5}| elem sz = 4 #size of one element in array
6 start: (4)
 7 \mid addi \times 20, \times 0, len/enroll (5)
8 | 1a x1, x (6) 
9 loop:
10 | \text{lw } \times 2, 0(\times 1) (7)
11 add x31, x31, x2 (8)
12 | \text{Iw } x2, 4(x1) |
13 add x31, x31, x2
14 | \text{Iw } x2, 8(x1) |
15 add x31, x31, x2
16 | \text{lw x2}, 12(\text{x1}) |
17 add x31, x31, x2
18 addi x1, x1, elem sz*enroll (9)
|19| \text{ addi } \times 20, \times 20, -1 (10)
20 bne x20, x0, 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
                                                   x1, x1, 60 \# 80000040 <_x>
4
       80000008:
                                           a d d i
                    03c08093
5
6
       8000000c <loop >:
7
                                               x2,0(x1)
       8000000c:
                    0000a103
                                           lw
8
       80000010:
                    002 f8fb3
                                           add x31,x31,x2
9
                                           lw x2, 4(x1)
       80000014:
                    0040a103
10
                    002 f8 fb3
                                           add x31,x31,x2
       80000018:
11
       8000001c:
                    0080a103
                                           lw x2,8(x1)
                                           add x31,x31,x2
12
       80000020:
                    002 f8 fb3
                                           lw x2, 12(x1)
13
       80000024:
                    00c0a103
14
       80000028:
                    002 f8 fb3
                                           add x31,x31,x2
                                                   x1,x1,16
15
       8000002c:
                    01008093
                                           addi
16
       80000030:
                    fffa0a13
                                           a d d i
                                                   x20, x20, -1
                    fc0a1ce3
                                           bne x20, x0, 8000000c < loop >
17
       80000034:
18
       80000038:
                    001f8f93
                                           addi
                                                   x31,x31,1
19
       8000003c <forever >:
20
21
       8000003c:
                    0000006f
                                           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 80000020 на первой итерации) представлен на рисунке 1.

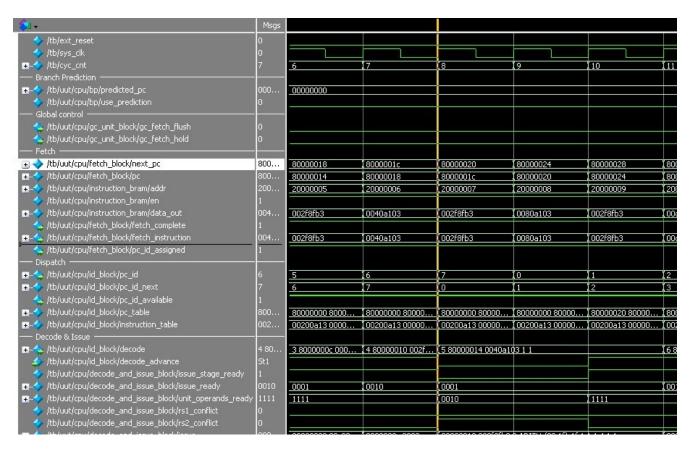


Рисунок 1 – Временная диаграмма выполнения стадий выборки и диспетчеризации

Скриншот, полученный в ходе выполнения задания \mathbb{N}_{2} (получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 8000002с на первой итерации) представлен на рисунке 2.

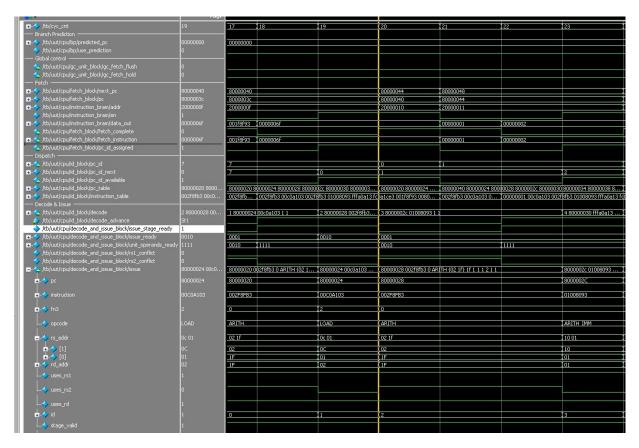


Рисунок 2 – Временная диаграмма выполнения стадии декодирования и планирования на выполнение

Скриншот, полученный в ходе выполнения задания №4 (получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000014 на первой итерации) представлен на рисунке 3.

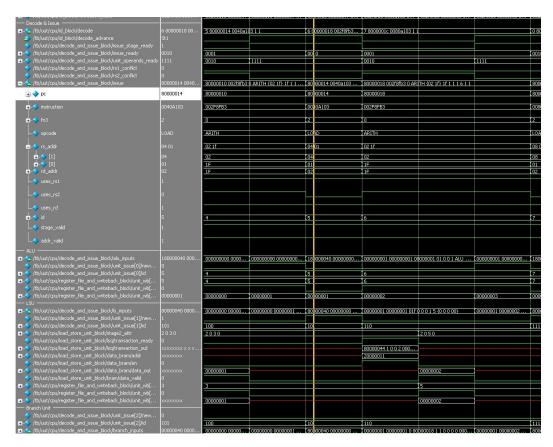


Рисунок 3 – Временная диаграмма выполнения стадии выполнения

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
       enroll = 2 \#amount of processed elements in one iteration
4
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 x2, 0(x1)
11
12
       |w x3, 4(x1) \#!
13
       addi x1, x1, elem sz*enroll
       addi x20, x20, -1
14
       add x31, x31, x2
15
       add x31, x31, x3
16
       bne x20, x0, lp
17
       addi x31, x31, 1
18
19
       lp2: j lp2
20
21
       section data
22
                4 byte 0x1
       X
23
       .4 byte 0x2
24
       .4 byte 0x3
25
       .4 byte 0x4
26
       .4 byte 0x5
27
       .4 byte 0x6
28
       4 byte 0x7
       4 byte 0x8
```

Дизассемблерный листинг исследуемой программы представлен на листинге 5.

Листинг 5 – Дизассемблерный листинг исследуемой программы

1	80000000 <_st	:art >:		
2	8000000:	00400a13	a d d i	x20 , x0 , 4
3	80000004:	00000097	auipc	×1 ,0 ×0
4	8000008:	02c08093	a d d i	x1,x1,44 # 80000030
	<_x>			
5				
6	8000000c <lp></lp>	> :		
7	8000000c:	0000a103	lw	x2 ,0 (x1)
8	8000010:	0040a183	lw	x3 ,4(x1)
9	8000014:	00808093	a d d i	×1 , ×1 , 8
10	8000018:	fffa0a13	a d d i	×20 , ×20 , —1
11	800001c:	002 f8 fb3	add	x31 , x31 , x2
12	80000020:	003f8fb3	add	x31 , x31 , x3
13	80000024:	fe0a14e3	bne	x20,x0,8000000c <1p>
14	80000028:	001f8f93	addi	×31 , ×31 ,1
15				
16	8000002c < lp2	>:		
17	8000002c:	0000006 f	jal	x0,8000002c < lp2>
L	l .			

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
            do {
9
10
                 int x2 = x1[0];
11
                 int x3 = x1[1];
12
                 x1 += enroll;
13
                 \times 20 --
14
                 x31 += x2;
15
                 x31 += x3;
16
            } while (x20 != 0);
17
            x31++;
            while(1){}
18
19
       }
```

Трасса работы программы

Трасса работы программы представлена на рисунке 4.

100	Код команды	200	\neg																	Н	омег	та	кта										_
Адрес		Команда	1d	1 2	3 4	5 6	5 7	8 9	10 11	1 12 1	13 14	15 16	17 1	18 19	20 2	1 22	23 24	25	26 27					3 34	35 3	6 37	38 39	40 4	1 42	43 44	45 46	6 47	48 49
80000000<_start>	00400a13	addi x20,x0,4	0		D Al																												
80000004	00000097	auipc x1,0x0	1	F	ID D	AL				П			П					П															
80000008	02c08093	addi x1,x1,44 # 80000030 <_x>	2		FI	DA	L																										
8000000c <lp></lp>	0000a103	lw x2,0(x1)	3		F	ID [
80000010	0040a183	lw x3,4(x1)	4			FI	DD	41 M2	МЗ																								
80000014	808093	addi x1,x1,8	5			1	ID																										
80000018	fffa0a13	addi x20,x20,-1	6 7				F:	ID D	AL																								
8000001c	002f8fb3	add x31,x31,x2	7			П	П		D AI				П					П						П									
80000020	003f8fb3	add x31,x31,x3	0					F	ID D																								
80000024	fe0a14e3	one x20,x0,8000000c <lp></lp>	1						FII	D D	В																						
80000028	001f8f93	addi x31,x31,1	2						F	ID	D X																						
8000002c <lp2></lp2>	0000006f	al x0,8000002c <lp2></lp2>	3							F 3	ID DX							П						П									
80000030	1		4								FX																						
8000000c <lp></lp>	0000a103	lw x2,0(x1)	3			П	П			П		F II	d C	41 M2	МЗ			П		П	-			П								П	Т
80000010	0040a183	lw x3,4(x1)	4									F	ID	D M1	M2 M	13																	
80000014	808093	addi x1,x1,8	5			П	П			П			F]	D D	AL			П		П				П	Т							П	
80000018	fffa0a13	addi x20,x20,-1	6											F ID	DA	L																	
8000001c	002f8fb3	add x31,x31,x2	7			П	П			П			П	F	ID [) AL		П		П	т			П	Т	П						П	т
80000020	003f8fb3	add x31,x31,x3	0												FI	D D	AL																
80000024	fe0a14e3	one x20,x0,8000000c <lp></lp>	1			П	П	Т		П			П		1	F ID	D B	П		П	т			П	Т	П						П	Т
8000000c <lp></lp>	0000a103	lw x2,0(x1)	2													F	ID D	M1 I	12 M3	3													
80000010	0040a183	lw x3,4(x1)	3			П	П			П			П				F II	DI	41 M2	М3	\neg			П								П	т
80000014	808093	addi x1,x1,8	4														F	ID	D AL														
80000018	fffa0a13	addi x20,x20,-1	5				П			П			П					F:	D D	AL	т			П		П						П	\top
8000001c	002f8fb3	add x31,x31,x2	6																FIL	D	AL												
80000020	003f8fb3	add x31,x31,x3	7				П						П					П	F	ID	D A	L											\top
80000024	fe0a14e3	bne x20,x0,8000000c <lp></lp>	0																		ID C												
8000000c <lp></lp>	0000a103	lw x2,0(x1)	1										П					П			FΙ	D D	M1 M	2 M3								П	т
80000010	0040a183	lw x3,4(x1)	2																		F	ID	D M	1 M2	М3								
80000014	808093	addi x1,x1,8	2			П	П			П			П					П		П		F	ID [AL					П			П	т
80000018	fffa0a13	addi x20.x201	4																				FI	D D	AL								
8000001c	002f8fb3	add x31,x31,x2	5			П	П			П			П					П						ID		L			П			П	_
80000020	003f8fb3	add x31,x31,x3	6																							AL							
80000024	fe0a14e3	bne x20,x0,8000000c <lp></lp>	7				П			П			П					П								D D	В						\top
8000000c <lp></lp>	0000a103	lw x2,0(x1)	0																						F	ID	D DX						
80000010	0040a183	lw x3,4(x1)	1				т						П					П									ID X						
80000014	808093	addi x1,x1,8	2																								FX						
80000028	001f8f93	addi x31,x31,1	1				П			П			П					П						П				ΕI	D D	AL		П	_
8000002c <lp2></lp2>	0000006f	ial x0,8000002c <lp2></lp2>	2																											D B			
80000030	1	Joseph Lings of the Line of th	3																											ID D	х		
80000034	2		4																											F ID			
80000034	3		5																												X		
8000003c 8000002c <lp2></lp2>	0000006f	ial x0.8000002c <lp2></lp2>	4																												**	TD.	D AL
	Код команды			1 2	3 4	5 6	5 7	8 9	10 11	1 12 1	13 14	15 16	17 1	18 19	20 2	1 22	23 24	25	26 27	28	29 3	0 31	32 3	3 34	35 3	6 37	38 39	40 4	1 42	43 44			_
Anpec	3000 3000000000	Команда	id	-	1517	- 1	1.1			1	-1		150	-1-0		1-2	- 1-		. 1-0,	1	-10	- *		15.7	- 10	1-7	- 3 (1.4	1	061	1	-4-4

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.

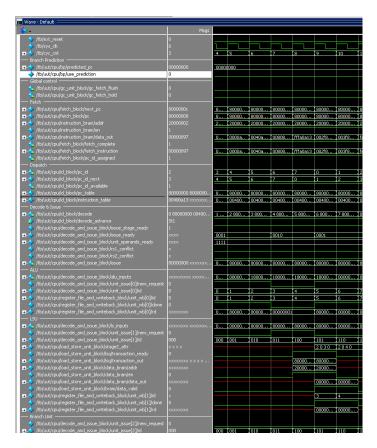


Рисунок 5 – Временные диаграммы сигналов

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
       lw x3, 4(x1) #!
12
13
       addi x20, x20, -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
       _x:
                .4 byte 0x1
29
       .4 byte 0x2
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

VINCIAIII O	дизассемолерный лис	тиш оптимизиро	bannon npor pammar
80000000 <_s	tart >:		
80000000:	00200a13	a d d i	x20 , x0 , 2
80000004:	00000097	auipc	×1 , 0 ×0
80000008:	03c08093	a d d i	x1,x1,60 # 80000040
<_x>			
8000000c < p	>:		
8000000c:	0000a103	lw	x2 ,0 (x1)
80000010:	0040a183	lw	×3 ,4 (×1)
8000014:	fffa0a13	a d d i	×20 , ×20 , —1
80000018:	002 f8 fb3	add	x31 , x31 , x2
8000001c:	003 f8 fb 3	add	x31 , x31 , x3
80000020:	0080a103	lw	x2 ,8 (x1)
80000024:	00c0a183	lw	x3 ,12(x1)
80000028:	01008093	a d d i	×1,×1,16
8000002c:	002 f8 fb 3	add	x31 , x31 , x2
80000030:	003 f8 fb3	add	x31 , x31 , x3
80000034:	fc0a1ce3	bne	x20,x0,8000000c <lp></lp>
80000038:	001f8f93	a d d i	x31 , x31 , 1
8000003c < lp2	2>:		
8000003c:	0000006 f	jal	x0,8000003c < lp2>
	80000000 < _ s 80000000 : 80000004: 80000008:	80000000 <_start >: 80000000: 00200a13 80000004: 00000097 80000008: 03c08093	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 add 80000030: 003f8fb3 add 80000034: fc0a1ce3 bne 80000038: 001f8f93 addi

Трасса работы программы

Трасса работы оптимизированной программы представлена на рисунке 6.

Адрес	Код команды	Команда	id		2 4	151	0 7	1010	110	. 1 1 2	242	1 1 1 5	1001	710	* 010	2 21	201	22 2	·loc	200	7 0		ep			204	25 27		20	2014	2 41	142
	00400a13	addi x20,x0,4			D A		6 /	8 9	10	11 1.	2 13 .	14 15	16 1	/ 18	19/2	0 21	22 2	23 24	1 25	26 2	2/ 2	8 29	30.	313	2 33	34	35 St	3/	38	39 4	0 41	. 42
	00000097	auipc x1,0x0	1	200	ID D				П																							
	02c08093	addi x1,x1,60 # 80000040 < x>	2			D D A	Δ1																									
	0000a103	lw x2,0(x1)	3				-	M2 M	2																							
04.	0040a183	lw x3,4(x1)	4		,	_		M1 M	_																							
		addi x20.x201	5					D A																								
	002f8fb3	add x31,x31,x2	6					ID																								
	002f8fb3	add x31.x31.x2	7					2000	D D	ΔΙ																						
	0080a103	lw x2,8(x1)	0						ID		1 M2 I	N3																				
		w x3.12(x1)	1								M1																					
	1008093	addi x1,x1,16	2								D D																					
		add x31,x31,x2	3							0000	ID	000																				
	003f8fb3	add x31,x31,x3	4							ľ	-	ID D																				
	fc0a1ce3	bne x20,x0,8000000c <lp></lp>	5			П			П		100	F ID	20000	3																		т
	001f8f93	addi x31.x31.1	6										ID																			
	0000006f	ial x0,8000003c <lp2></lp2>	7										200000000000000000000000000000000000000	D DX																		
80000040 (ошиб.)			0											X																		
1000000	0000a103	lw x2,0(x1)	7												FΙ	D D	M1 N	42 M	3													П
	0040a183	lw x3,4(x1)	0														DI															
9000014	fffa0a13	addi x20,x20,-1	1													2020	ID	200	ES 25550													
	002f8fb3	add x31,x31,x2	2																AL													
00001c	002f8fb3	add x31,x31,x2	3														100	0000	D D													
0000020	0080a103	lw x2,8(x1)	4															F	ID	D M	11 M	2 M3										
0000024	00c0a183	w x3,12(x1)	5																F	ID I	D M	1 M2	МЗ									
000028		addi x1,x1,16	6																	FI	D D	AL										
000002c	002f8fb3 >	add x31,x31,x2	7																		F II	D D	AL									
000030	003f8fb3	add x31,x31,x3	0																		F	ID	D .	AL								
000034	fc0a1ce3	bne x20,x0,8000000c <lp></lp>	1																			F	ID	D E	3							
00000c <lp></lp>	0000a103	lw x2,0(x1)	2																				F:	ID [X							
000010	0040a183	w x3,4(x1)	3																					FI	D DX							
000014	fffa0a13	addi x20,x20,-1	4																					1	= X							
000038	001f8f93	addi x31,x31,1	3																							F	ID D	AL				
00003c <lp2></lp2>	0000006f	jal x0,8000003c <lp2></lp2>	4																								FI	D	AL			
900040 (ошиб.)			5																													
000044 (ошиб.)			6																													
000048 (ощиб.)			7																													Г
00003c <lp2></lp2>	0000006f	jal x0,8000003c <lp2></lp2>	6																											F	FID	D
Annes	Код команды	Verrauma	1,,	1 2	3 4	5	6 7	8 9	10	11 12	2 13	14 15	16 1	7 18	19 2	0 21	22	23 2	4 25	26 2	27 2	8 29	30	31 3	2 33	34	35 36	37	38	39 4	0 41	42
Адрес		Команда	id																			Ном	iep	так	та							_

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.