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»

Отчет по лабораторной работе №2 по курсу "Архитектура ЭВМ"

Тема Изучение принципов работы микропроцессорного ядра RISC-V

Студент Прянишников А. Н.

Группа ИУ7-55Б

Оценка (баллы) _____

Преподаватели Ибрагимов С.В.

Москва — 2021 г.

Цели работы

Основная цель работы – 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.

Ход работы

Код по общему заданию

Код программы по общему заданию представлен на листинге 1.

Листинг 1: Код общего задания

```
        .section .text (1)
        .globl _start; (2)
        len = 8
        enroll = 4
        elem_sz = 4
_start: (4)
        addi x20, x0, len/enroll (5)
        la x1, _x (6)
loop:
        lw x2, 0(x1) (7)
        add x31, x31, x2 (8)
        lw x2, 4(x1)
        add x31, x31, x2
        lw x2, 8(x1)
        add x31, x31, x2
        lw x2, 12(x1)
        add x31, x31, x2
        addi x1, x1, elem_sz*enroll (9)
        addi x20, x20, -1 (10)
        bne x20, x0, loop (11)
        addi x31, x31, 1
forever: j forever (12)

        .section .data (13)
_x:     .4byte 0x1 (14)
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

Дизассемблированный код представлен на листинге 2

Листинг 2: Дизассемблированный код общего задания

```
Disassembly of section .text:

80000000 <_start>:
```

```
80000000: 00200a13 addi x20,x0,2
80000004: 00000097 auipc x1,0x0
80000008: 03c08093 addi x1,x1,60 # 80000040 <_x>

8000000c <loop>:
8000000c: 0000a103 lw x2,0(x1)
80000010: 002f8fb3 add x31,x31,x2
80000014: 0040a103 lw x2,4(x1)
80000018: 002f8fb3 add x31,x31,x2
8000001c: 0080a103 lw x2,8(x1)
80000020: 002f8fb3 add x31,x31,x2
80000024: 00c0a103 lw x2,12(x1)
80000028: 002f8fb3 add x31,x31,x2
8000002c: 01008093 addi x1,x1,16
80000030: fffa0a13 addi x20,x20,-1
80000034: fc0a1ce3 bne x20,x0,8000000c <loop>
80000038: 001f8f93 addi x31,x31,1

8000003c <forever>:
8000003c: 0000006f jal x0,8000003c <forever>

Disassembly of section .data:

80000040 <_x>:
80000040: 0001 c.addi x0,0
80000042: 0000 unimp
80000044: 0002 0x2
80000046: 0000 unimp
80000048: 00000003 lb x0,0(x0) # 0 <elem_sz-0x4>
8000004c: 0004 c.addi4spn x9,x2,0
8000004e: 0000 unimp
80000050: 0005 c.addi x0,1
80000052: 0000 unimp
80000054: 0006 0x6
80000056: 0000 unimp
80000058: 00000007 0x7
8000005c: 0008 c.addi4spn x10,x2,0
```

Псевдокод на языке C представлен на листинге 3.

Листинг 3: Псевдокод общего задания

```
#define len 8
#define enroll 4
#define elem_sz 4
int _x[]={1,2,3,4,5,6,7,8};
void _start() {
    int x20 = len/enroll;
    int *x1 = _x;
```

```
    do {
      int x2 = x1[0];
      x31 += x2;
      x2 = x1[1];
      x31 += x2;
      x2 = x1[2];
      x31 += x2;
      x2 = x1[3];
      x31 += x2;
      x1 += enroll;
      x20--;
    } while(x20 != 0);
    x31++;
    while(1){}
}
```

Задание 2

Задание: получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с указанным адресом.

Мой вариант: 6.

Адрес команды, номер итерации: 80000020, 1-я.

Код команды: 002f8fb3.

Команда: add x31,x31,x2.

Во время выборки команды снимается сигнал gc_fetch_hold разрешая работу блока выборки, а сигнал pc_id_available равен 1, что подтверждает готовность блока управления метаданными принять результат выборки. Сигнал pc равен адресу команды, также этот адрес выставляется на шину данных (addr).

Выставление сигнала en разрешает работу памяти команд. Одновременно с этим выставляется сигнал pc_id_assigned, указывающий блоку управления метаинформацией, что запрос в память отправлен, и информация о текущем pc должна быть записана в очередь команд. В начале следующего такта значение pc будет записано в pc_table по индексу, равному значению текущего первого свободного id, а pc_id увеличится на 1.

По фронту, завершающему такт 2, выбраный код команды (то есть, сигнал fetch_instruction) записывается в таблицу instruction_table по индексу. Это завершает операцию диспетчеризации.

На рисунке 1 приведён скрин по заданию с подписанными соответствующими сигналами:

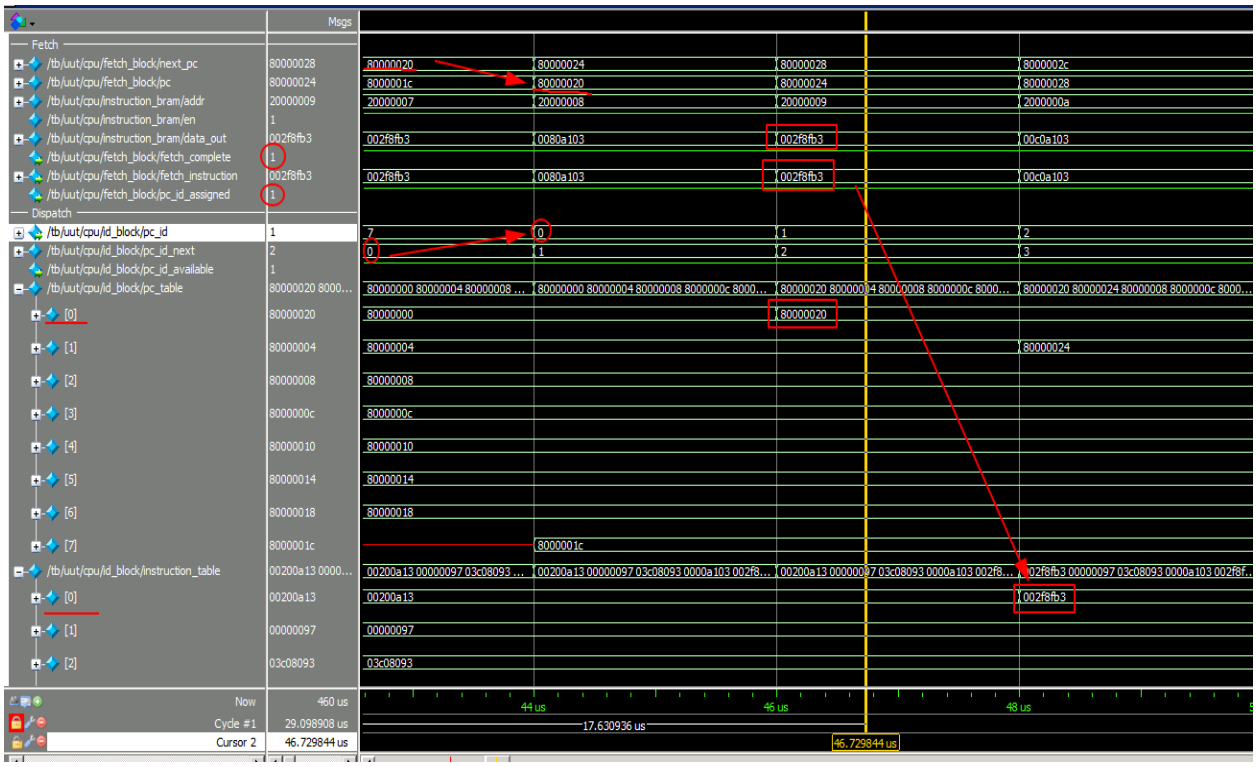

Рисунок. 1: Демонстрация выборки и диспетчеризации команды

Задание 3

Задание: получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с указанным адресом.

Мой вариант: 6.

Адрес команды, номер итерации: 8000002c, 1-я.

Код команды: 01008093.

Команда: addi x1,x1,16.

Так как возник конфликт по регистру x1 (rs2), выполнение стадии декодирования заняло 3 такта, а не один. Из-за конфликта сигнал decode_advance

выставлен в 0.

На рисунке 2 приведён скрин по заданию с подписанными соответствующими сигналами:

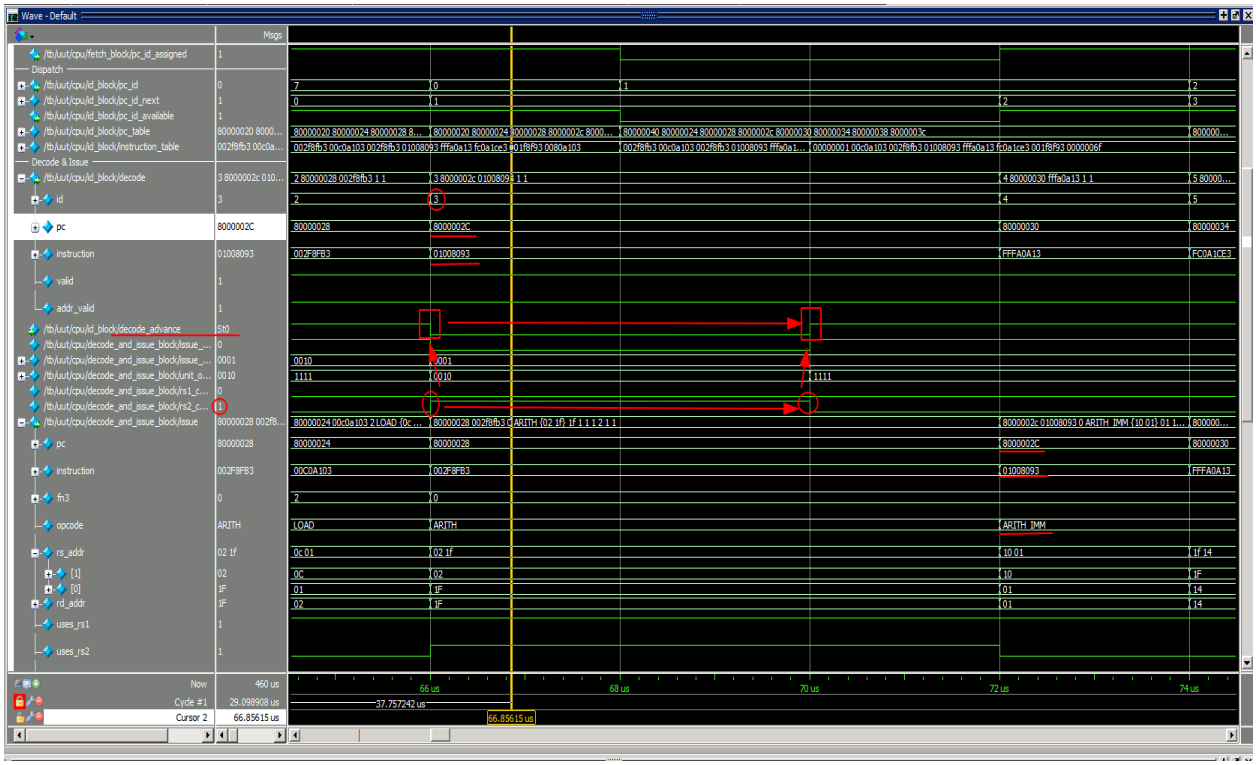

Рисунок. 2: Демонстрация выполнения стадии декодирования и планирования

Задание 4

Задание: получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с указанным адресом. Мой вариант: 6.

Адрес команды, номер итерации: 80000014, 1-я.

Код команды: 0040a103.

Команда: lw x2,4(x1).

Это команда блока обращения к памяти, поэтому выполнение заняло 3 такта.

На рисунках 3-4 приведены скрины по заданию с подписанными соответствующими сигналами:

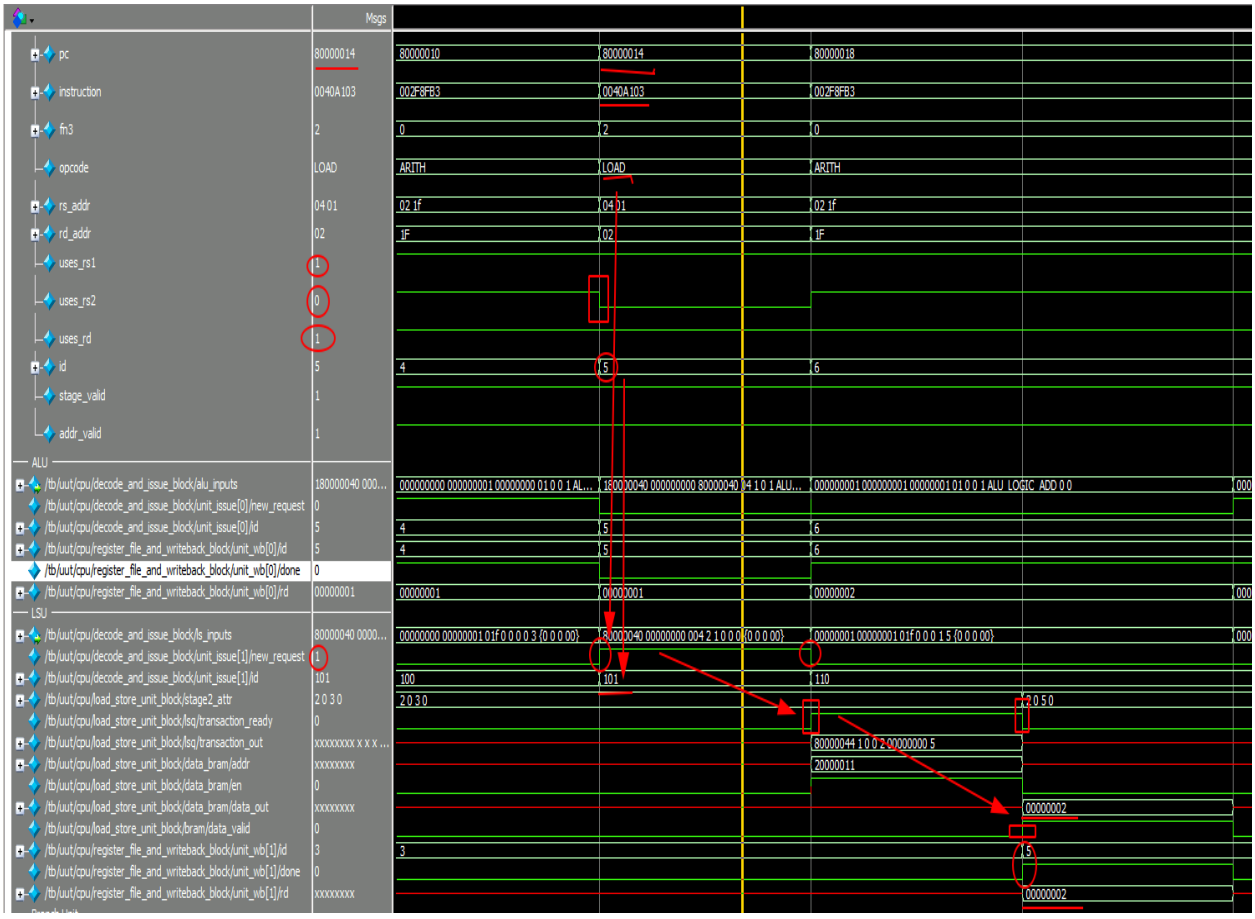

Рисунок. 3: Демонстрация выполнения стадии выполнения программы (часть 1)

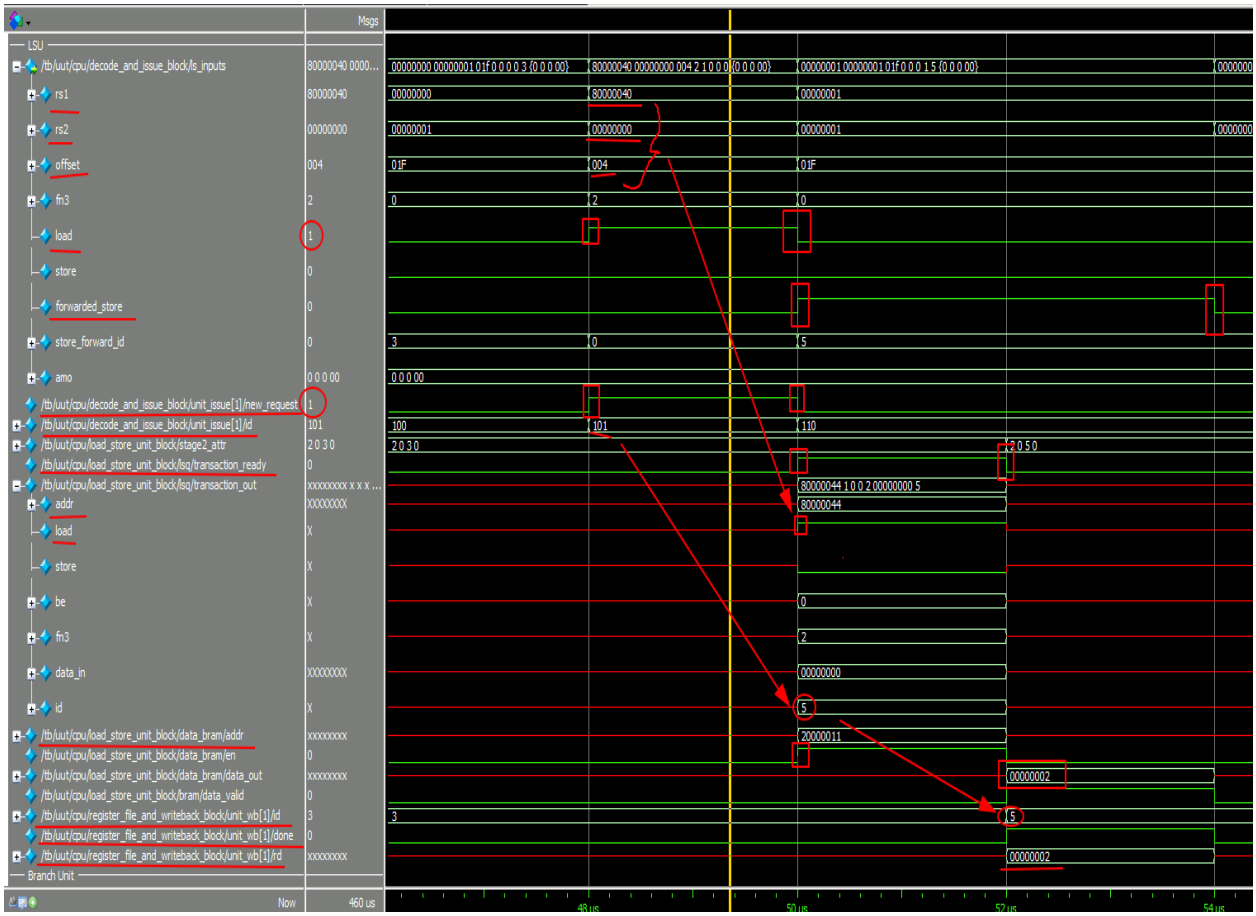

Рисунок. 4: Демонстрация выполнения стадии выполнения программы (часть 2)

Код по индивидуальному заданию

Код программы по индивидуальному заданию представлен на листинге 4.

Листинг 4: Код индивидуального задания

```
        .section .text
        .globl _start;
        len = 8
        enroll = 2
        elem_sz = 4

_start:
        addi x20, x0, len/enroll
        la x1, _x
lp:
        lw x2, 0(x1)
        lw x3, 4(x1) #!
        addi x1, x1, elem_sz*enroll
        addi x20, x20, -1
        add x31, x31, x2
        add x31, x31, x3
        bne x20, x0, lp
        addi x31, x31, 1
lp2: j lp2

        .section .data
_x:     .4byte 0x1
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

Дизассемблированный код представлен на листинге 5:

Листинг 5: Дизассемблированный код индивидуального задания

```
Disassembly of section .text:

80000000 <_start>:
80000000: 00400a13 addi x20,x0,4
80000004: 00000097 auipc x1,0x0
80000008: 02c08093 addi x1,x1,44 # 80000030 <_x>
```

```
8000000c <lp>:
8000000c: 0000a103 lw x2,0(x1)
80000010: 0040a183 lw x3,4(x1)
80000014: 00808093 addi x1,x1,8
80000018: fffa0a13 addi x20,x20,-1
8000001c: 002f8fb3 add x31,x31,x2
80000020: 003f8fb3 add x31,x31,x3
80000024: fe0a14e3 bne x20,x0,8000000c <lp>
80000028: 001f8f93 addi x31,x31,1

8000002c <lp2>:
8000002c: 0000006f jal x0,8000002c <lp2>

Disassembly of section .data:

80000030 <_x>:
80000030: 0001 c.addi x0,0
80000032: 0000 unimp
80000034: 0002 0x2
80000036: 0000 unimp
80000038: 00000003 lb x0,0(x0) # 0 <enroll-0x2>
8000003c: 0004 c.addi4spn x9,x2,0
8000003e: 0000 unimp
80000040: 0005 c.addi x0,1
80000042: 0000 unimp
80000044: 0006 0x6
80000046: 0000 unimp
80000048: 00000007 0x7
8000004c: 0008 c.addi4spn x10,x2,0
```

Задание 5

Была получена временную диаграмму сигналов выполнения программы индивидуального варианта.

На рисунке 5 показан результат работы программы, который содержится в регистре x31:

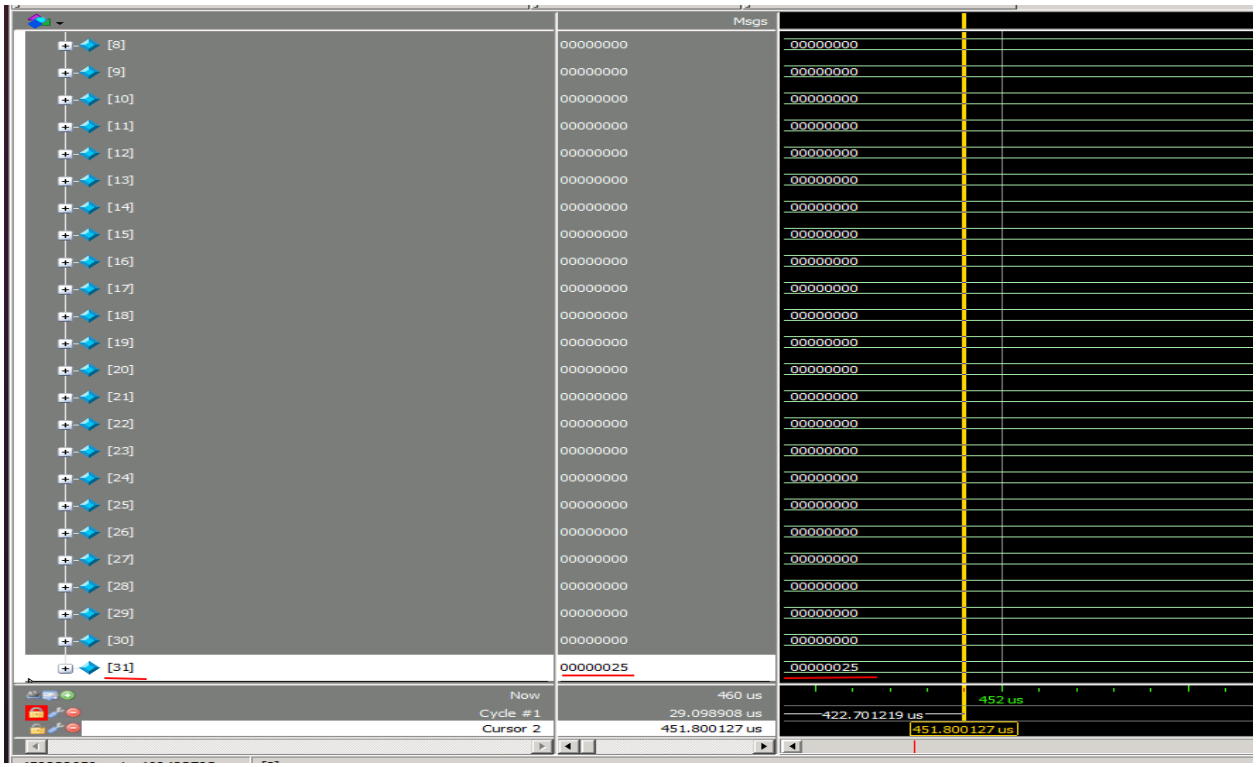

Рисунок. 5: Результат работы программы

Результат совпал с ожидаемым.

Задание: Получить снимок экрана, содержащий временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом !.

Мой вариант – 6.

Адрес команды: 80000010.

Код команды: 0040a183.

Команда: lw x3, 4(x1).

На рисунке 6 показан скрин выполнения стадий выборки и диспетчеризации для команды:

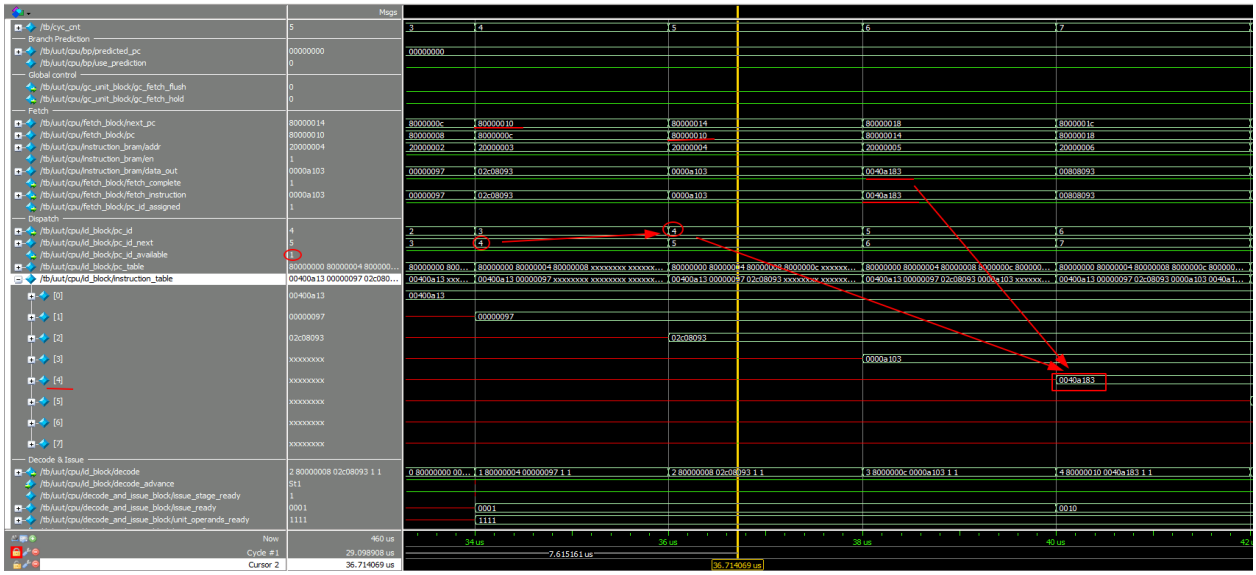

Рисунок. 6: Демонстрация выполнения стадий выборки и диспетчеризации для команды по индивидуальному варианту

На рисунке 7 показан скрин выполнения стадий декодирования и планирования для команды:

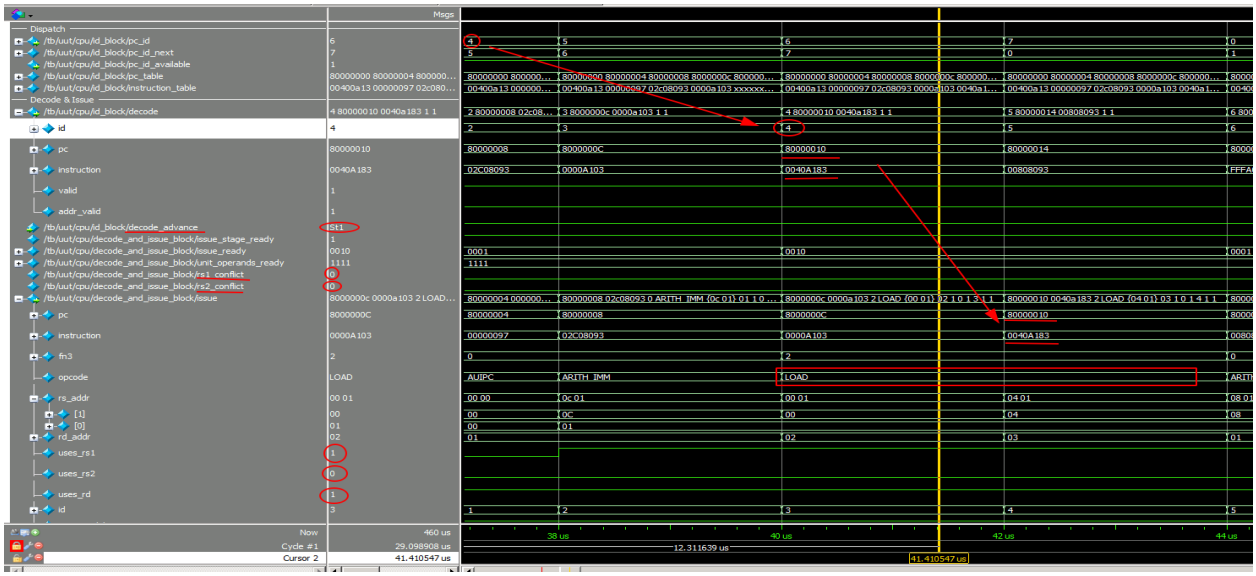

Рисунок. 7: Демонстрация выполнения стадий декодирования и планирования для команды по индивидуальному варианту

На рисунках 8-9 показаны скрины выполнения стадии выполнения для команды:

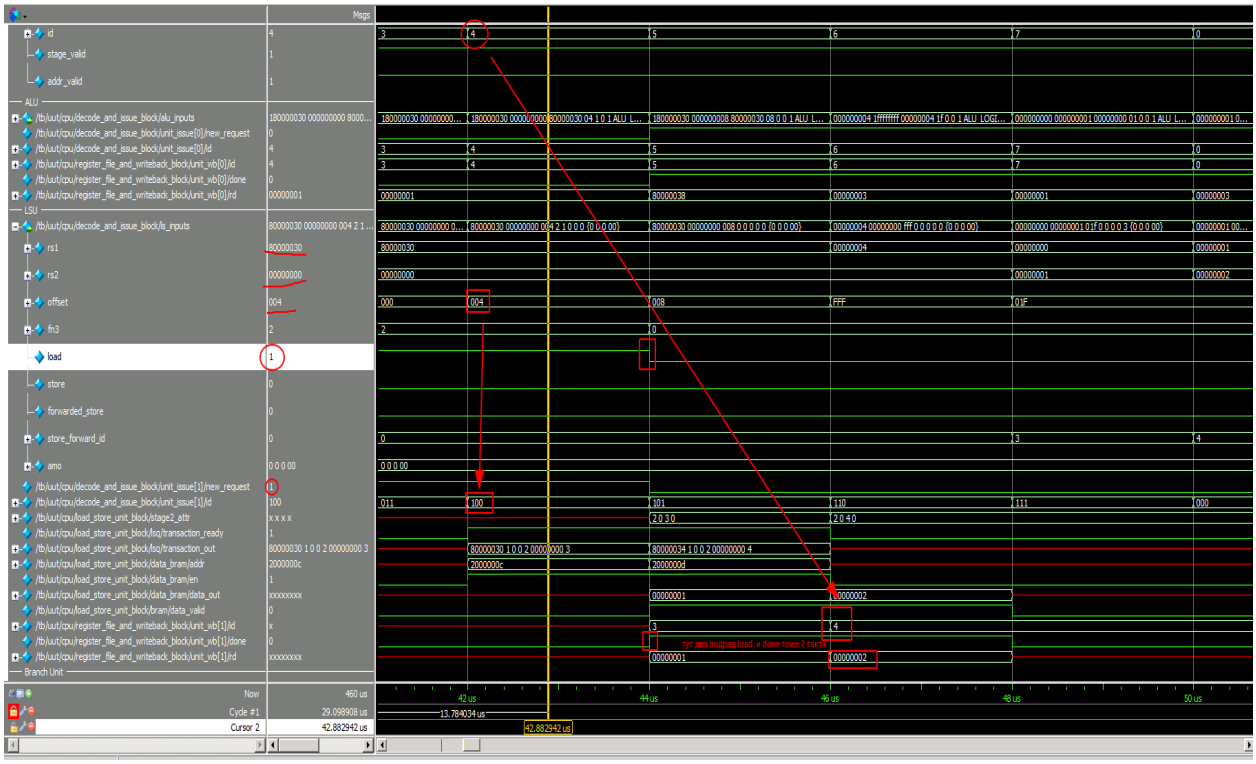

Рисунок. 8: Демонстрация выполнения стадии выполнения для команды по индивидуальному варианту (часть 1)

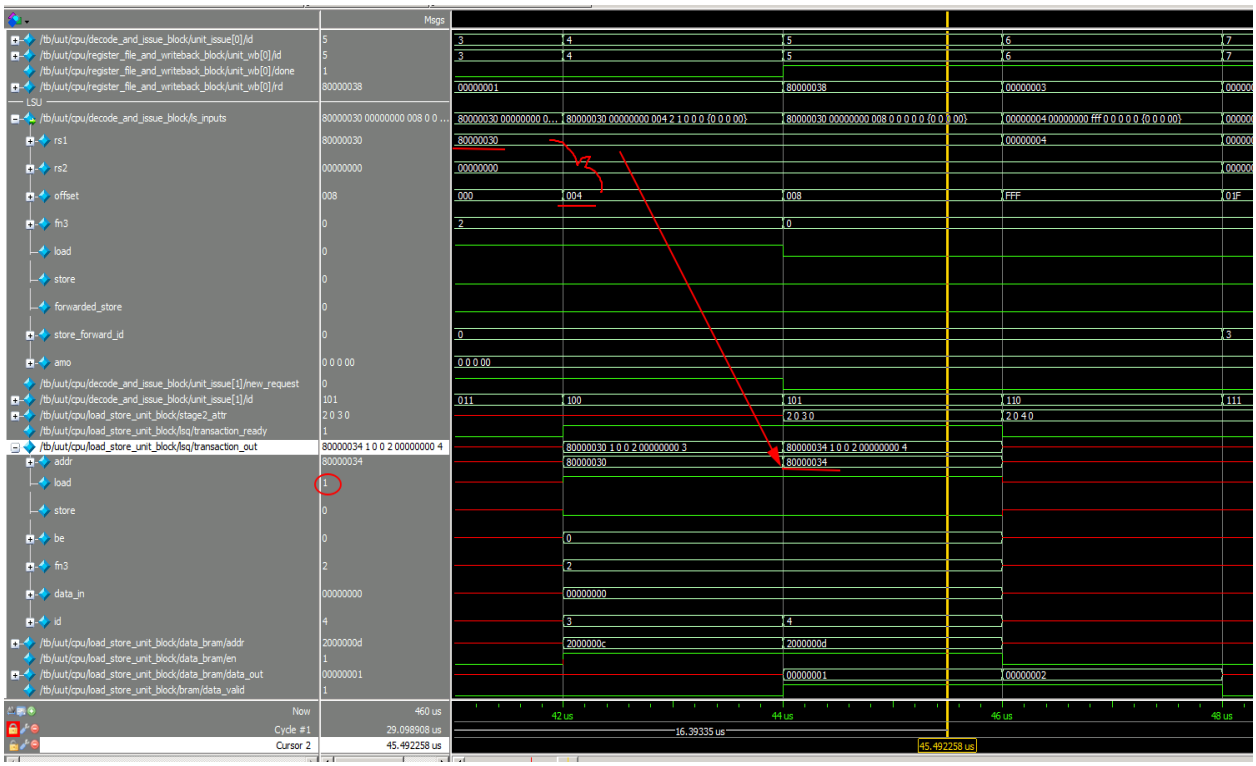

Рисунок. 9: Демонстрация выполнения стадии выполнения для команды по индивидуальному варианту (часть 2)

Трасса выполнения программы

На рисунке 10 приведена трасса выполнения программы:

Адрес	Код команды	Команда	id	Номер такта																																																		
				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	47	48	49	50	51
80000000<_start>	0x00400a13	addi x20,x0,4	0	F	ID	D	AL																																															
80000004	0x00000097	auipc x1,0x0	1		F	ID	D	AL																																												
80000008	0x02c08093	addi x1,x1,44 # 80000030 <	2			F	ID	D	AL																																										
8000000c<lp>	0x0000a103	lw x2,0(x1)	3				F	ID	D	M1	M2	M3																																						
80000010	0x0040a183	lw x3,4(x1)	4					F	ID	D	M1	M2	M3																																					
80000014	0x00808093	addi x1,x1,8	5						F	ID	D	AL																																						
80000018	0xfffa0a13	addi x20,x20,-1	6							F	ID	D	AL																																					
8000001c	0x002f8fb3	add x31,x31,x2	7								F	ID	D	AL																																				
80000020	0x003f8fb3	add x31,x31,x3	0									F	ID	D	AL																																			
80000024	0xfe0a14e3	bne x20,x0,8000000c	1										F	ID	D	B																																		
80000028	0x001f8f93	addi x31,x31,1	2											F	ID	D	X																																	
8000002c<lp2>	0x0000006f	jal x0,8000002c	3												F	ID	DX																																	
80000030	0x00000001	<invalid operation>	4													F	X																																	
8000000c<lp>	0x0000a103	lw x2,0(x1)	3															F	ID	D	M1	M2	M3																													
80000010	0x0040a183	lw x3,4(x1)	4																F	ID	D	M1	M2	M3																												
80000014	0x00808093	addi x1,x1,8	5																			F	ID	D	AL																													
80000018	0xfffa0a13	addi x20,x20,-1	6																				F	ID	D	AL																												
8000001c	0x002f8fb3	add x31,x31,x2	7																					F	ID	D	AL																											
80000020	0x003f8fb3	add x31,x31,x3	0																						F	ID	D	AL																										
80000024	0xfe0a14e3	bne x20,x0,8000000c	1																							F	ID	D	B																									
8000000c<lp>	0x0000a103	lw x2,0(x1)	2																								F	ID	D	M1	M2	M3																						
80000010	0x0040a183	lw x3,4(x1)	3																									F	ID	D	M1	M2	M3																					
80000014	0x00808093	addi x1,x1,8	4																										F	ID	D	AL																						
80000018	0xfffa0a13	addi x20,x20,-1	5																											F	ID	D	AL																					
8000001c	0x002f8fb3	add x31,x31,x2	6																												F	ID	D	AL																				
80000020	0x003f8fb3	add x31,x31,x3	7																													F	ID	D	AL																			
80000024	0xfe0a14e3	bne x20,x0,8000000c	0																														F	ID	D	B																		
8000000c<lp>	0x0000a103	lw x2,0(x1)	1																															F	ID	D	M1	M2	M3															
80000010	0x0040a183	lw x3,4(x1)	2																																F	ID	D	M1	M2	M3														
80000014	0x00808093	addi x1,x1,8	3																																	F	ID	D	AL															
80000018	0xfffa0a13	addi x20,x20,-1	4																																		F	ID	D	AL														
8000001c	0x002f8fb3	add x31,x31,x2	5																																			F	ID	D	AL													
80000020	0x003f8fb3	add x31,x31,x3	6																																				F	ID	D	AL												
80000024	0xfe0a14e3	bne x20,x0,8000000c	7																																					F	ID	D	B											
8000000c<lp>	0x0000a103	lw x2,0(x1)	0																																						F	ID	D	DX										
80000010	0x0040a183	lw x3,4(x1)	1																																							F	ID	DX										
80000014	0x00808093	addi x1,x1,8	2																																								D	X										
80000028	0x001f8f93	addi x31,x31,1	1																																										F	ID	D	AL						
8000002c<lp2>	0x0000006f	jal x0,8000003c<forever>	2																																											F	ID	D	B					
80000030	0x00000001	<invalid operation>	3																																												F	ID	D	X				
80000034	0x00000002	<invalid operation>	4																																													F	ID	DX				
80000038	0x00000003	<invalid operation>	5																																														F	X				
8000002c<lp2>	00000002	jal x0,8000003c<forever>	4																																																F	ID	D	AL
Адрес	Код команды	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	47	48	49	50	51
				Номер такта																																																		

Рисунок. 10: Трасса выполнения программы

Конфликтов по операндам в программе не было, но программа дважды неправильно угадывала результат сравнения в цикле, из-за чего выполнение было дольше.

Оптимизация программы

Так как задержка шла из-за циклов, я принял решение убрать цикл. Программа считает сумму всех элементов массива, поэтому я сначала все значения массива в регистр, а затем сложил их и получил ответ.

Код программы по общему заданию представлен на листинге 6.

Листинг 6: Оптимизированный код индивидуального задания

```
.section .text
 .globl _start;


_start:
        la x1, _x
        lw x2, 0(x1)
        lw x3, 4(x1)
        lw x4, 8(x1)
        lw x5, 12(x1)
        lw x6, 16(x1)
        lw x7, 20(x1)
        lw x8, 24(x1)
        lw x9, 28(x1)
        addi x31, x31, x2
        addi x31, x31, x3
        addi x31, x31, x4
        addi x31, x31, x5
        addi x31, x31, x6
        addi x31, x31, x7
        addi x31, x31, x8
        addi x31, x31, x9
        addi x31, x31, 1
lp2: j lp2

        .section .data
_x:     .4byte 0x1
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

Дизассемблированный код представлен на листинге 7:

Листинг 7: Дизассемблированный код

```
SYMBOL TABLE:
80000000 l    d  .text  00000000 .text
80000050 l    d  .data  00000000 .data
00000000 l    df *ABS*  00000000 opt.o
80000050 l       .data  00000000 _x
8000004c l       .text  00000000 lp2
80000000 g       .text  00000000 _start
80000070 g       .data  00000000 _end



Disassembly of section .text:

80000000 <_start>:
80000000:       0x00000097              auipc x1,0x0
80000004:       0x05008093              addi  x1,x1,80 # 80000050 <_x>
80000008:       0x0000a103              lw x2,0(x1)
8000000c:       0x0040a183              lw x3,4(x1)
80000010:       0x0080a203              lw x4,8(x1)
80000014:       0x00c0a283              lw x5,12(x1)
80000018:       0x0100a303              lw x6,16(x1)
8000001c:       0x0140a383              lw x7,20(x1)
80000020:       0x0180a403              lw x8,24(x1)
80000024:       0x01c0a483              lw x9,28(x1)
80000028:       0x002f8fb3              add x31,x31,x2
8000002c:       0x003f8fb3              add x31,x31,x3
80000030:       0x004f8fb3              add x31,x31,x4
80000034:       0x005f8fb3              add x31,x31,x5
80000038:       0x006f8fb3              add x31,x31,x6
8000003c:       0x007f8fb3              add x31,x31,x7
80000040:       0x008f8fb3              add x31,x31,x8
80000044:       0x009f8fb3              add x31,x31,x9
80000048:       0x001f8f93              addi x31,x31,1

8000004c <lp2>:
8000004c:       0x0000006f              jal x0,8000004c <lp2>

Disassembly of section .data:

80000050 <_x>:
80000050:       0001                    c.addi  x0,0
80000052:       0000                    unimp
80000054:       0002                    0x2
80000056:       0000                    unimp
80000058:       00000003                lb      x0,0(x0) # 0 <_start-0
   x80000000>
8000005c:       0004                    c.addi4spn      x9,x2,0
8000005e:       0000                    unimp
```

```
80000060:       0005                    c.addi   x0,1
80000062:       0000                    unimp
80000064:       0006                    0x6
80000066:       0000                    unimp
80000068:       00000007                0x7
8000006c:       0008                    c.addi4spn       x10,x2,0
```

На рисунке 11 представлен результат работы программы:

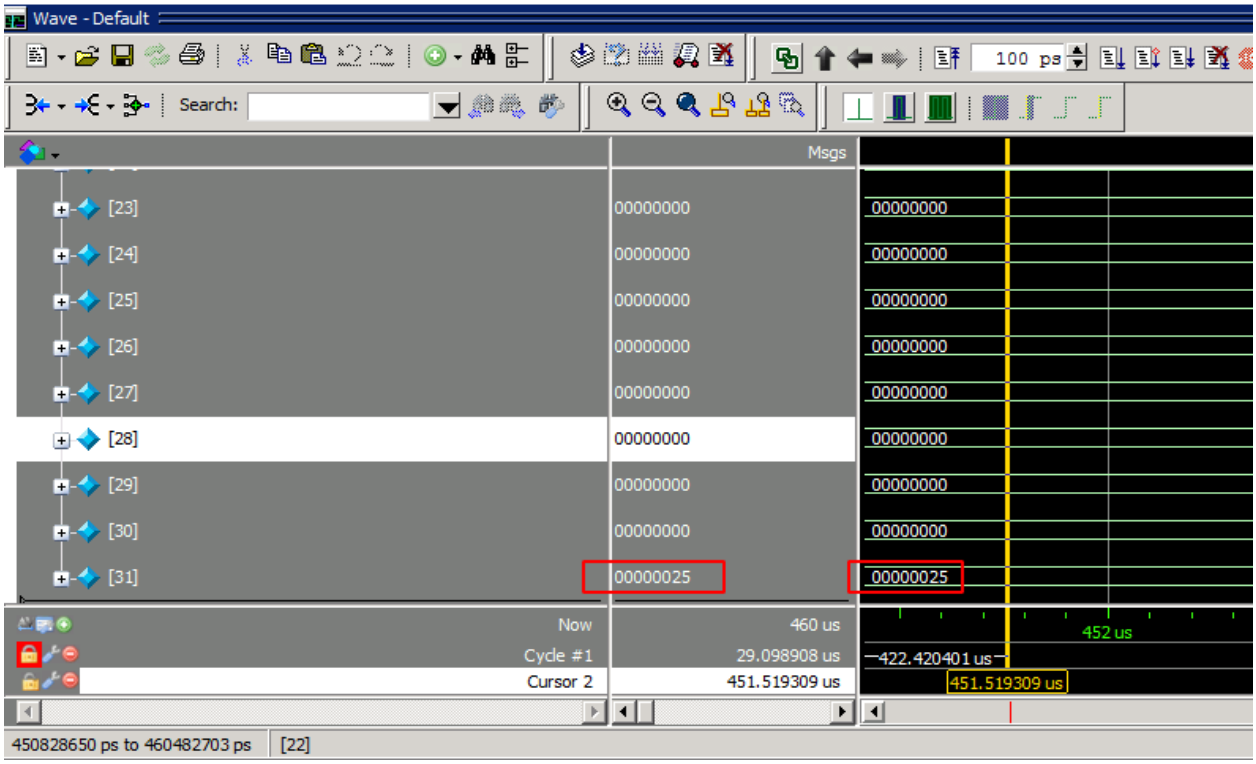

Рисунок. 11: Результат работы оптимизированной программы

Результат совпал с ожидаемым и полученным для начальной программы.

На рисунке 12 представлена трасса выполнения оптимизированной программы

Адрес	Код команды	Команда	id	Номер такта																										
				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
80000000<_start>	0x00000097	auipc x1,0x0	0	F	ID	D	AL																								
80000004	0x05008093	addi x1,x1,80 # 80000050	1		F	ID	D	AL																							
80000008	0x0000a103	lw x2,0(x1)	2			F	ID	D	M1	M2	M3																				
8000000c	0x0040a183	lw x3,4(x1)	3				F	ID	D	M1	M2	M3																			
80000010	0x0080a203	lw x4,8(x1)	4					F	ID	D	M1	M2	M3																		
80000014	0x00c0a283	lw x5,12(x1)	5						F	ID	D	M1	M2	M3																	
80000018	0x0100a303	lw x6,16(x1)	6							F	ID	D	M1	M2	M3																
8000001c	0x0140a383	lw x7,20(x1)	7								F	ID	D	M1	M2	M3															
80000020	0x0180a403	lw x8,24(x1)	0									F	ID	D	M1	M2	M3														
80000024	0x01c0a483	lw x9,28(x1)	1										F	ID	D	M1	M2	M3													
80000028	0x002f8fb3	add x31,x31,x2	2											F	ID	D	AL														
8000002c	0x003f8fb3	add x31,x31,x3	3												F	ID	D	AL													
80000030	0x004f8fb3	add x31,x31,x4	4													F	ID	D	AL												
80000034	0x005f8fb3	add x31,x31,x5	5														F	ID	D	AL											
80000038	0x006f8fb3	add x31,x31,x6	6															F	ID	D	AL										
8000003c	0x007f8fb3	add x31,x31,x7	7																F	ID	D	AL									
80000040	0x008f8fb3	add x31,x31,x8	0																	F	ID	D	AL								
80000044	0x009f8fb3	add x31,x31,x9	1																		F	ID	D	AL							
80000048	0x001f8f93	addi x31,x31,1	2																			F	ID	D	AL						
8000004c<lp2>	0x0000006f	jal x0,8000004c <lp2>	3																				F	ID	D	B					
80000060	0x00000001	<invalid operation>	4																					F	ID	D	X				
80000064	0x00000002	<invalid operation>	5																						F	ID	DX				
80000068	0x00000003	<invalid operation>	6																							F	X				
8000004c<lp2>	0x0000006f	jal x0,8000004c <lp2>	4																									F	ID	D	AL
Адрес	Код команды	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
				Номер такта																											

Рисунок. 12: Трасса выполнения оптимизированной программы

Количество тактов сокращено практически вдвое. Цена — расширяемость 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.