

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Лабораторная №9

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Оптимизация работы с массивами

Задание 2

Студенты:

Соболь В.

Темнова А.С.

Группа: 13541/3

Преподаватель:

Антонов А.П.

Санкт-Петербург
2019

Содержание

1. Задание	4
2. Часть 1	12
2.1. Исходный код	12
2.2. Скрипт выполнения	14
2.3. Моделирование	15
2.4. Решение a1	15
2.4.1. Директивы	15
2.4.2. Синтез	16
2.4.3. Анализ решения	19
2.5. Решение a2	19
2.5.1. Директивы	19
2.5.2. Синтез	20
2.5.3. Анализ решения	23
2.6. Решение a3	23
2.6.1. Директивы	23
2.6.2. Синтез	23
2.6.3. Анализ решения	26
2.7. Решение a4	26
2.7.1. Директивы	26
2.7.2. Синтез	27
2.7.3. Анализ решения	30
2.8. Решение a5	30
2.8.1. Директивы	30
2.8.2. Синтез	31
2.8.3. Анализ решения	34
2.9. Решение a6	35
2.9.1. Директивы	35
2.9.2. Синтез	35
2.9.3. Анализ решения	38
2.10. Решение a7	39
2.10.1. Директивы	39
2.10.2. Синтез	39
2.10.3. Анализ решения	43
2.11. Вывод	43
3. Часть 2	43
3.1. Исходный код	43
3.2. Скрипт выполнения	45
3.3. Моделирование	46
3.4. Решение b1	47
3.4.1. Директивы	47
3.4.2. Синтез	47
3.4.3. Анализ решения	50
3.5. Решение b2	50
3.5.1. Директивы	50

3.5.2.	Синтез	51
3.5.3.	Анализ решения	53
3.6.	Решение b3	54
3.6.1.	Директивы	54
3.6.2.	Синтез	54
3.6.3.	Анализ решения	58
3.7.	Решение b4	58
3.7.1.	Директивы	58
3.7.2.	Синтез	58
3.7.3.	Анализ решения	62
3.8.	Решение b5	62
3.8.1.	Директивы	62
3.8.2.	Синтез	63
3.8.3.	Анализ решения	66
3.9.	Решение b6	67
3.9.1.	Директивы	67
3.9.2.	Синтез	67
3.9.3.	Анализ решения	70
3.10.	Решение b7	71
3.10.1.	Директивы	71
3.10.2.	Синтез	71
3.10.3.	Анализ решения	75
3.11.	Решение b8	76
3.11.1.	Директивы	76
3.11.2.	Синтез	76
3.11.3.	Анализ решения	79
3.12.	Вывод	80

1. Задание

- Создать проект lab9_2
- Микросхема: xa7a12tcsg325-1q

ЧАСТЬ 1

- Создать функцию

```
foo_a: входной массив short d_in[N]; выходной массив short d_out [N/4].
for (short i=0; i<N/4; i++){
d_out[i] = d_in[i]*d_in[i+8] + d_in[i+4]*d_in[i+12];
}
N=16
```

- Создать тест lab9_2_test.c для проверки функции. Осуществить моделирование (с выводом результатов в консоль)
- Исследование:
- Solution_1a

- задать: clock period 10; clock_uncertainty 0.1
- установить реализацию bram; RAM_1P_BRAM для входного (и выходного) массива
- осуществить синтез
- привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval

- Solution_2a

- задать: clock period 10; clock_uncertainty 0.1
- установить реализацию ap_memory; RAM_1P для входного (и выходного) массива
- осуществить синтез
- привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary

- * performance Profile
- * Resource profile
- * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_1a и solution_2a) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_3a
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_2P для входного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_2a и solution_3a) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_4a
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного массива
 - установить array_partition; block; factor =2 для входного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile

- * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_3a и solution_4a) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_5a
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного и выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
-
- Сравнить два решения (solution_4a и solution_5a) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_6a
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_2P для входного и выходного массивов
 - установить array_partition; block; factor =2 для входного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary

- * performance Profile
- * Resource profile
- * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
-
- Сравнить два решения (solution_5a и solution_6a) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_7a
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_2P для входного и выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
-
- Сравнить два решения (solution_6a и solution_7a) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...

Сделать сводную таблицу ($S_x/Latency/II$ – номер решения/ $Latency/II$)

	RAM_1P	RAM_2P
Без block	$S_x/Latency/II$	
block; factor =2		
block; factor =4		

ЧАСТЬ 2

- Создать функцию

```
foo_b: входной массив short d_in[N]; выходной массив short d_out [N].
for (short i=0; i<N/4; i++){
d_out[i] = d_in[i]*d_in[i+4];
d_out[i+1]= d_in[i+8]*d_in[i+12];
d_out[i+2]= d_in[i]*d_in[i+12];
d_out[i+3]= d_in[i+4]*d_in[i+8];
}
N=16
```

- Solution_1b

- задать: clock period 10; clock_uncertainty 0.1
- установить реализацию ap_memory; RAM_1P для входного и выходного массивов
- осуществить синтез
- привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval

- Solution_2b

- задать: clock period 10; clock_uncertainty 0.1
- установить реализацию ap_memory; RAM_1P для входного и выходного массивов
- установить array_partition; block; factor =4 для входного массива
- осуществить синтез
- привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval

- * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_1b и solution_2b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_3b
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного и выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - установить array_partition; cyclic; factor =2 для выходного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_2b и solution_3b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_4b
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного и выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - установить array_partition; cyclic; factor =4 для выходного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile

- * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_3b и solution_4b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_5b
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного и RAM_2P для выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - установить array_partition; cyclic; factor =1 для выходного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_3b и solution_5b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
- Solution_6b
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного и RAM_2P для выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - установить array_partition; cyclic; factor =2 для выходного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)

- * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_5b и solution_6b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...
 - Solution_7b
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию ap_memory; RAM_1P для входного и RAM_2P для выходного массивов
 - установить array_partition; block; factor =4 для входного массива
 - установить array_partition; cyclic; factor =4 для выходного массива
 - осуществить синтез
 - привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - Сравнить два решения (solution_6b и solution_7b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...

Сделать сводную таблицу ($S_x/Latency/II$ – номер решения/ $Latency/II$)

	RAM_1P	RAM_2P
Без cyclic	$S_x/Latency/II$	
cyclic; factor =2		
cyclic; factor =4		

- Solution_8b

- задать: clock period 10; clock_uncertainty 0.1
- установить реализацию ap_memory; RAM_1P для входного и выходного массивов
- установить array_partition; block; factor =4 для входного массива
- установить array_partition; complete для выходного массива
- осуществить синтез
- привести в отчете:
 - * performance estimates=>summary (timing, latency)
 - * utilization estimates=>summary
 - * performance Profile
 - * Resource profile
 - * scheduler viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
 - * resource viewer (выполнить Zoom to Fit)
 - На скриншоте показать Latency
 - На скриншоте показать Initiation Interval
- Сравнить два решения (solution_2b и solution_8b) и сделать выводы: зависимость от типа интерфейса; объяснить количество использованных умножителей; объяснить (посчитать) число циклов Latency, II...

2. Часть 1

2.1. Исходный код

Для выполнения работы был написан код устройства и код теста, которые приведены ниже.

```

1 #include "lab9_2_1.h"
2
3 void foo_a(short d_in[N], short d_out[N/4]) {
4     for (short i=0; i<N/4; i++){
5         d_out[i] = d_in[i]*d_in[i+8] + d_in[i+4]*d_in[i+12];
6     }
7 }

```

Рис. 2.1. Код устройства

```

1 #include <stdio.h>
2 #include "lab9_2_1.h"
3
4
5 void generate_test_data(short scale, short d_in[N], short d_out[N/4]) {
6     short i;
7     for (i = 0; i < N; ++i){
8         d_in[i] = (i + 1) * scale;
9     }
10
11     for (i=0; i<N/4; i++){
12         d_out[i] = d_in[i]*d_in[i+8] + d_in[i+4]*d_in[i+12];
13     }
14 }
15
16 int compare_array_eq(short actual[N/4], short expected[N/4]){
17     for (int i = 0; i < N/4; ++i) {
18         if (actual[i] != expected[i]) {
19             fprintf(stdout, "%d: _Expeced_%d_ Actual_%d\n", i, expected[i], actual[i]);
20             ↪ return 0;
21         }
22     }
23     return 1;
24 }
25
26 int main() {
27     int pass = 1;
28     short d_in[N];
29     short d_out[N/4];
30     short expected_out[N/4];
31
32
33     for (int i = 1; i < 4; ++i) {
34         generate_test_data(i, d_in, expected_out);
35
36         foo_a(d_in, d_out);
37
38         if (!compare_array_eq(d_out, expected_out)){
39             pass = 0;
40         }
41     }
42
43     if (pass) {
44         fprintf(stdout, "—————Pass!—————\n");
45         return 0;
46     } else {
47         fprintf(stderr, "—————Fail!—————\n");
48         return 1;
49     }
50 }
51 }

```

Рис. 2.2. Код теста

```

1 #define N 16

```

Рис. 2.3. Заголовочный файл

2.2. Скрипт выполнения

Для автоматизации выполнения работы был написан следующий скрипт:

```
1 open_project -reset lab9_2_1
2 add_files lab9_2_1.c
3 set_top foo_a
4 add_files -tb lab9_2_1_test.c
5 set_solutions [list 1a 2a 3a 4a 5a 6a 7a]
6 foreach sol $solutions {
7   open_solution solution_$sol -reset
8   set_part {xa7a12tcs325-1q}
9   create_clock -period 10
10  set_clock_uncertainty 0.1
11  if {$sol == "1a"} {
12    set_directive_interface -mode bram foo_a d_in
13    set_directive_resource -core RAM_1P_BRAM foo_a d_in
14    set_directive_interface -mode bram foo_a d_out
15    set_directive_resource -core RAM_1P_BRAM foo_a d_out
16  }
17  if {$sol == "2a"} {
18    set_directive_interface -mode ap_memory foo_a d_in
19    set_directive_resource -core RAM_1P foo_a d_in
20    set_directive_interface -mode ap_memory foo_a d_out
21    set_directive_resource -core RAM_1P foo_a d_out
22  }
23  if {$sol == "3a"} {
24    set_directive_interface -mode ap_memory foo_a d_in
25    set_directive_resource -core RAM_2P foo_a d_in
26  }
27  if {$sol == "4a"} {
28    set_directive_interface -mode ap_memory foo_a d_in
29    set_directive_resource -core RAM_1P foo_a d_in
30    set_directive_array_partition -type block -factor 2 foo_a d_in
31  }
```

Рис. 2.4. Скрипт выполнения, часть 1

```

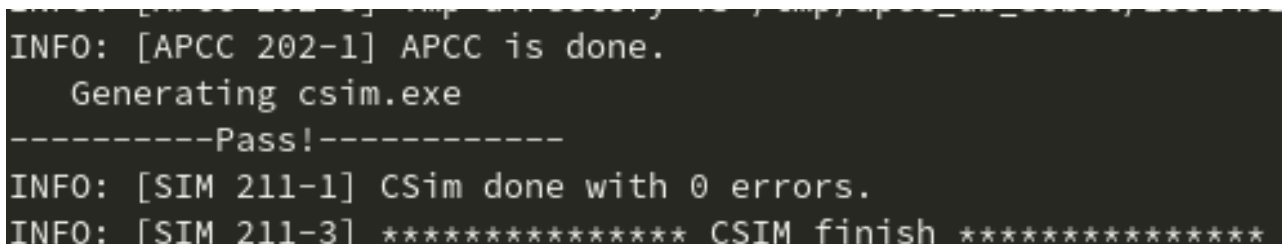
1  if {$sol == "5a"} {
2      set_directive_interface -mode ap_memory  foo_a d_in
3      set_directive_resource -core RAM_1P foo_a d_in
4      set_directive_interface -mode ap_memory  foo_a d_out
5      set_directive_resource -core RAM_1P foo_a d_out
6      set_directive_array_partition -type block -factor 4 foo_a d_in
7  }
8  if {$sol == "6a"} {
9      set_directive_interface -mode ap_memory  foo_a d_in
10     set_directive_resource -core RAM_2P foo_a d_in
11     set_directive_interface -mode ap_memory  foo_a d_out
12     set_directive_resource -core RAM_2P foo_a d_out
13     set_directive_array_partition -type block -factor 2 foo_a d_in
14 }
15 if {$sol == "7a"} {
16     set_directive_interface -mode ap_memory  foo_a d_in
17     set_directive_resource -core RAM_2P foo_a d_in
18     set_directive_interface -mode ap_memory  foo_a d_out
19     set_directive_resource -core RAM_2P foo_a d_out
20     set_directive_array_partition -type block -factor 4 foo_a d_in
21 }
22 csim_design
23 csynth_design
24 }
25 exit

```

Рис. 2.5. Скрипт выполнения, часть 2

2.3. Моделирование

Ниже приведены результаты моделирования, по которым видно, что тест проходит успешно.



```

INFO: [APCC 202-1] APCC is done.
      Generating csim.exe
-----Pass!-----
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****

```

Рис. 2.6. Результаты моделирования

2.4. Решение a1

2.4.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```

▼ ● foo_a
    ● d_in
    % HLS RESOURCE variable=d_in core=RAM_1P_BRAM
    % HLS INTERFACE bram port=d_in
    ● d_out
    % HLS RESOURCE variable=d_out core=RAM_1P_BRAM
    % HLS INTERFACE bram port=d_out
     $x+y = ?$  for Statement

```

Рис. 2.7. Директивы

2.4.2. Синтез

По оценке производительности видно, что устройство **HE** соответствует заданным критериям.

Performance Estimates

□ Timing (ns)

□ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	10.437	0.10

□ Latency (clock cycles)

□ Summary

Latency		Interval		
min	max	min	max	Type
25	25	25	25	none

Рис. 2.8. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	32
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	86
Register	-	-	67	-
Total	0	2	67	118
Available	40	4016000	8000	
Utilization (%)	0	5	~0	1

Рис. 2.9. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	25	-	26	-
Loop 1	no	24	6	-	4

Рис. 2.10. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_Addr_A	out	32	bram	d_in	array
d_in_EN_A	out	1	bram	d_in	array
d_in_WEN_A	out	2	bram	d_in	array
d_in_Din_A	out	16	bram	d_in	array
d_in_Dout_A	in	16	bram	d_in	array
d_in_Clk_A	out	1	bram	d_in	array
d_in_Rst_A	out	1	bram	d_in	array
d_out_Addr_A	out	32	bram	d_out	array
d_out_EN_A	out	1	bram	d_out	array
d_out_WEN_A	out	2	bram	d_out	array
d_out_Din_A	out	16	bram	d_out	array
d_out_Dout_A	in	16	bram	d_out	array
d_out_Clk_A	out	1	bram	d_out	array
d_out_Rst_A	out	1	bram	d_out	array

Рис. 2.11. Interface estimates

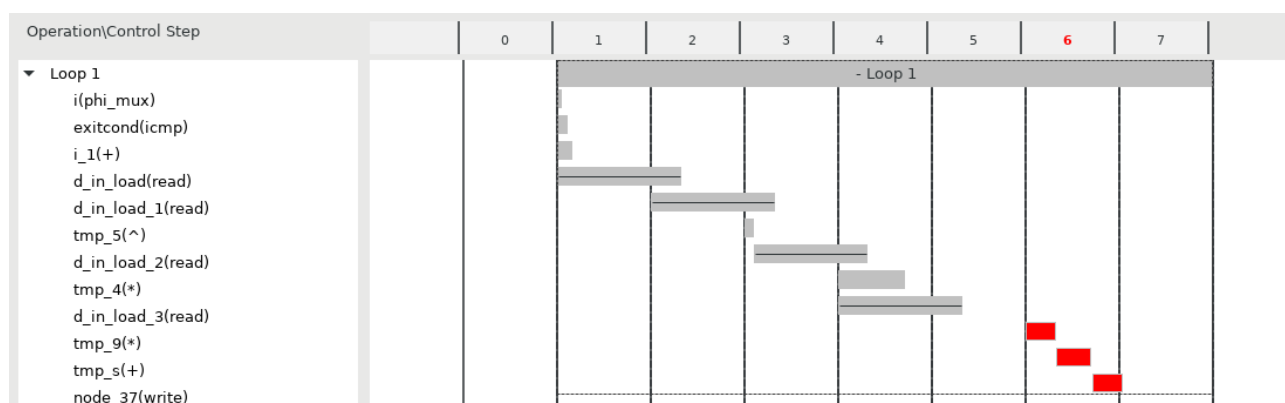


Рис. 2.12. Scheduler viewer

На рисунке выше видны блоки, которые не укладываются во временной интервал.

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6
1	⊟I/O Ports							
2	d_in(p0)		read	read	read	read		
3	d_out(p0)							write
4	⊟Memory Ports							
5	d_in(p0)		read	read	read	read		
6	d_out(p0)							write
7	⊟Expressions							
8	i_l_fu_116		+					
9	i_phi_fu_94		phi_mux					
10	exitcond_fu_110		icmp					
11	tmp_5_fu_140				^			
12	tmp_4_fu_159					*		
13	grp_fu_165							+

Рис. 2.13. Resource viewer

2.4.3. Анализ решения

На каждой итерации для двух операций умножения в устройстве используются 2 умножителя .

25 тактов задержки – это 1 начальный такт инициализации и 4 итерации цикла, каждый из которых занимает 6 тактов.

Интервал инициализации совпадает с задержкой, так как устройство не конвейеризировано и делает все последовательно.

Стоит отметить, что устройство не уложилось в требуемые 10 нс периода тактовой частоты.

2.5. Решение a2

2.5.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```

▼ ● foo_a
    ● d_in
    % HLS RESOURCE variable=d_in core=RAM_1P
    % HLS INTERFACE ap_memory port=d_in
    ● d_out
    % HLS RESOURCE variable=d_out core=RAM_1P
    % HLS INTERFACE ap_memory port=d_out
    X+Y
    =? for Statement

```

Рис. 2.14. Директивы

2.5.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
25	25	25	25	none

Рис. 2.15. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	32
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	77
Register	-	-	67	-
Total	0	2	67	109
Available	40	40	16000	8000
Utilization (%)	0	5	~0	1

Рис. 2.16. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	25	-	26	-
Loop 1	no	24	6	-	4

Рис. 2.17. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_address0	out	4	ap_memory	d_in	array
d_in_ce0	out	1	ap_memory	d_in	array
d_in_q0	in	16	ap_memory	d_in	array
d_out_address0	out	2	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 2.18. Interface estimates

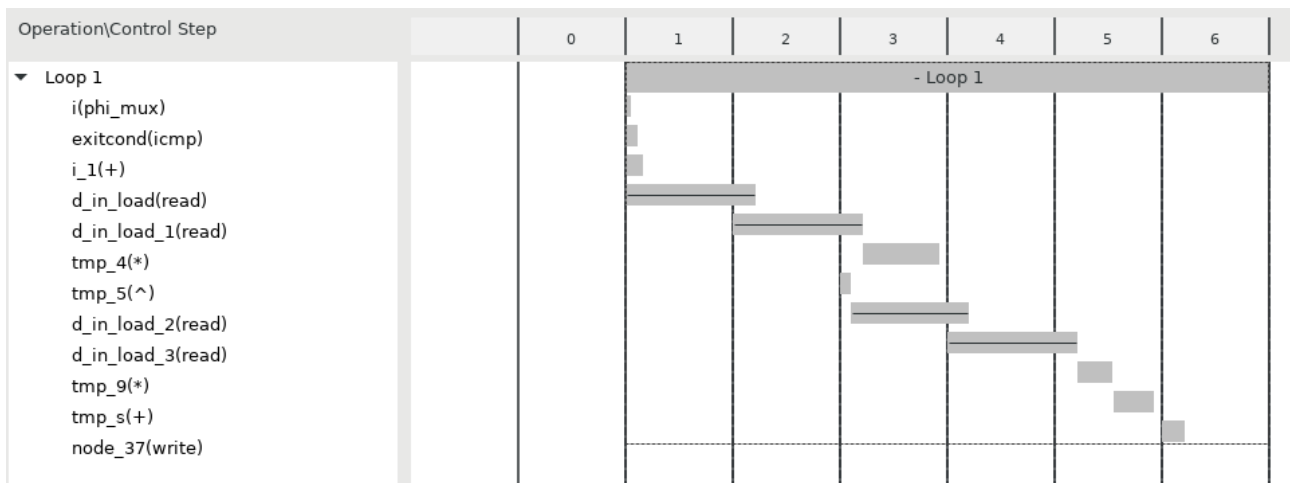


Рис. 2.19. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6
1	I/O Ports							
2	d_in(p0)		read	read	read	read		
3	d_out(p0)							write
4	Memory Ports							
5	d_in(p0)		read	read	read	read		
6	d_out(p0)							write
7	Expressions							
8	i_phi_fu_94		phi_mux					
9	i_l_fu_112		+					
10	exitcond_fu_106		icmp					
11	tmp_4_fu_155				*			
12	tmp_5_fu_136				^			
13	grp_fu_161						+	

Рис. 2.20. Resource viewer

2.5.3. Анализ решения

В сравнении с предыдущим, данное решение укладывается в отведённый временной интервал. Это связано с другими используемыми элементами памяти. Остальные параметры не отличаются.

2.6. Решение a3

2.6.1. Директивы

Ниже приведены директивы, установленные для данного решения.

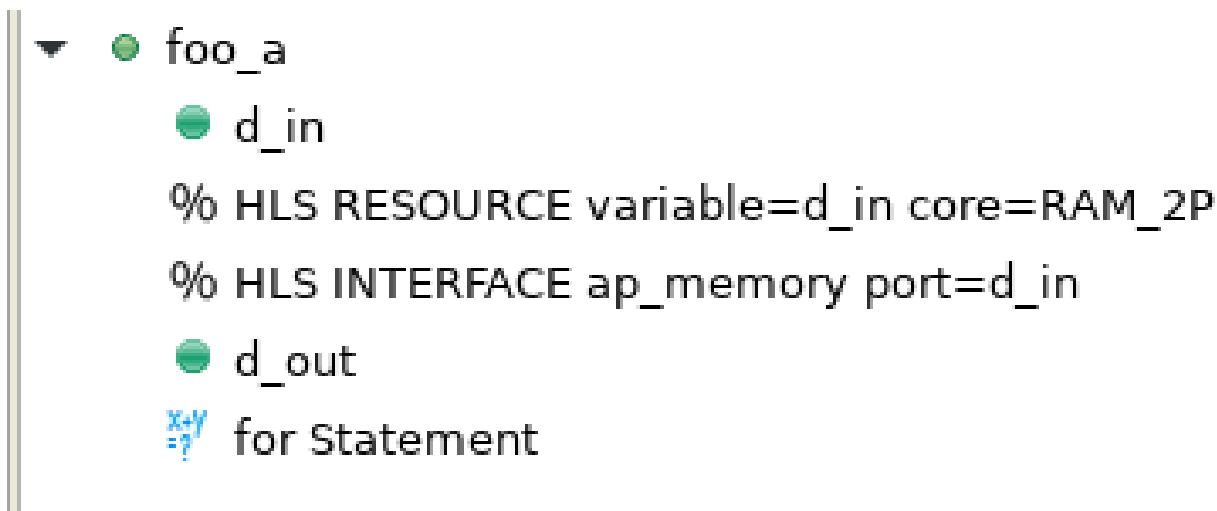


Рис. 2.21. Директивы

2.6.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
17	17	17	17	none

Рис. 2.22. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	32
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	72
Register	-	-	46	-
Total	0	2	46	104
Available	40	40	16000	8000
Utilization (%)	0	5	~0	1

Рис. 2.23. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	17	-	18	-
Loop 1	no	16	4	-	4

Рис. 2.24. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_address0	out	4	ap_memory	d_in	array
d_in_ce0	out	1	ap_memory	d_in	array
d_in_q0	in	16	ap_memory	d_in	array
d_in_address1	out	4	ap_memory	d_in	array
d_in_ce1	out	1	ap_memory	d_in	array
d_in_q1	in	16	ap_memory	d_in	array
d_out_address0	out	2	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 2.25. Interface estimates

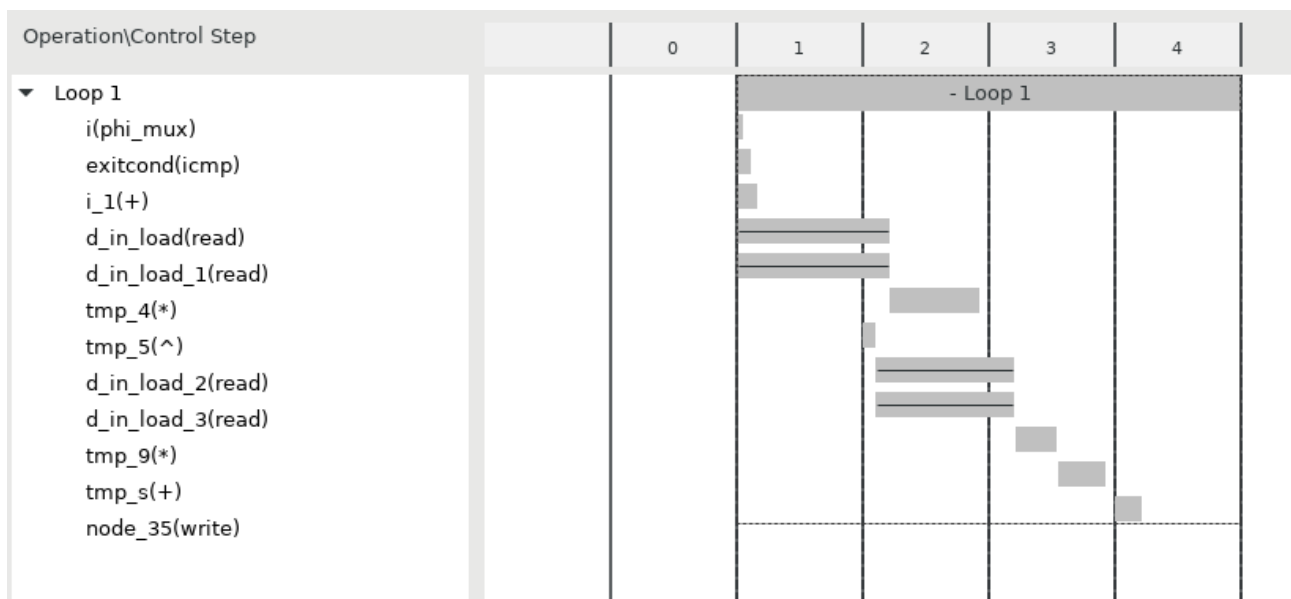


Рис. 2.26. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4
1	I/O Ports					
2	d_in(p1)		read	read		
3	d_in(p0)		read	read		
4	d_out(p0)					write
5	Memory Ports					
6	d_in(p1)		read	read		
7	d_in(p0)		read	read		
8	d_out(p0)					write
9	Expressions					
10	i_l_fu_112		+			
11	i_phi_fu_98		phi_mux			
12	exitcond_fu_106		icmp			
13	tmp_4_fu_156			*		
14	tmp_5_fu_136			^		
15	grp_fu_162				+	

Рис. 2.27. Resource viewer

2.6.3. Анализ решения


Как можно заметить, решение 3a не только быстрее, но и экономичнее, чем 2a. Оно используется столько же умножителей, однако благодаря двухпортовой памяти можно производить по 2 чтения из входного массива, что сокращает выполнение 1 итерации с 6 до 4 тактов. Соответственно, задержка и интервал инициализации меньше на 8 (минус 2 такта на 4-х итерациях) тактов.

2.7. Решение a4

2.7.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```

▼ ● foo_a
    ● d_in
    % HLS ARRAY_PARTITION variable=d_in block factor=2 dim=1
    % HLS RESOURCE variable=d_in core=RAM_1P
    % HLS INTERFACE ap_memory port=d_in
    ● d_out
     for Statement

```

Рис. 2.28. Директивы

2.7.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
17	17	17	17	none

Рис. 2.29. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	32
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	72
Register	-	-	46	-
Total	0	2	46	104
Available	40	40	16000	8000
Utilization (%)	0	5	~0	1

Рис. 2.30. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	17	-	18	-
Loop 1	no	16	4	-	4

Рис. 2.31. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_0_address0	out	3	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	3	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_out_address0	out	2	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 2.32. Interface estimates

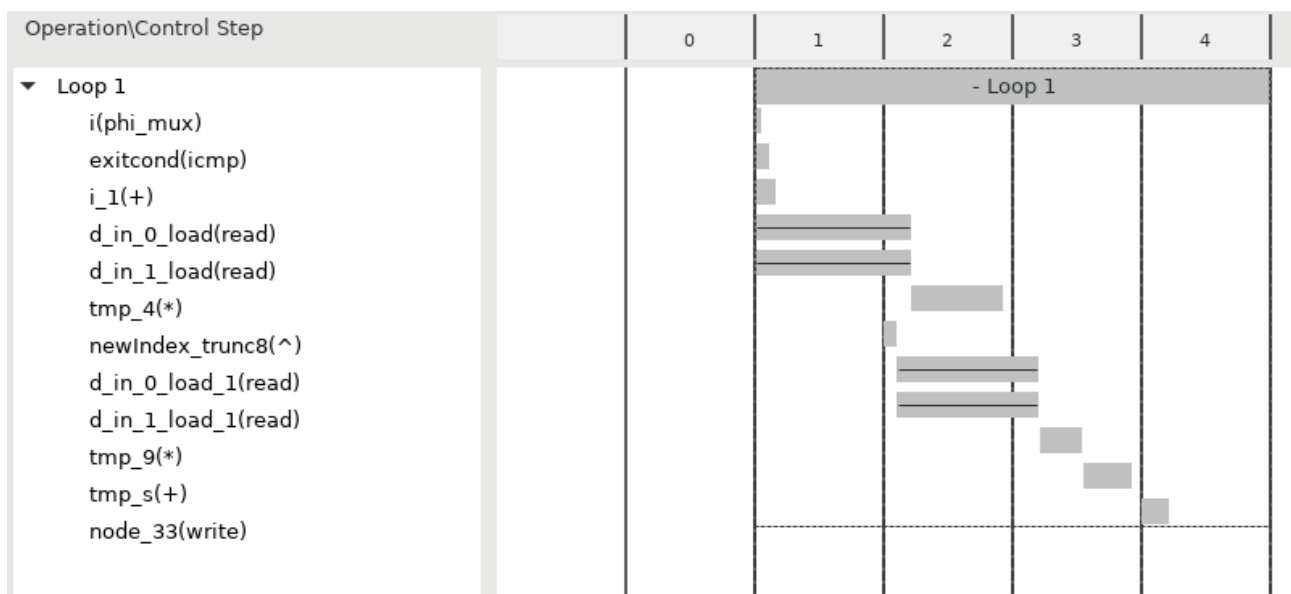


Рис. 2.33. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4
1	I/O Ports					
2	d_in_0(p0)		read	read		
3	d_in_1(p0)		read	read		
4	d_out(p0)					write
5	Memory Ports					
6	d_in_0(p0)		read	read		
7	d_in_1(p0)		read	read		
8	d_out(p0)					write
9	Expressions					
10	i_phi_fu_97		phi_mux			
11	i_l_fu_111		+			
12	exitcond_fu_105		icmp			
13	tmp_4_fu_135			*		
14	newIndex_trunc8_fu_123			^		
15	grp_fu_141				+	

Рис. 2.34. Resource viewer

2.7.3. Анализ решения

В сравнении с предыдущим решением, данное решение использует не один экземпляр двухпортовой памяти, а 2 экземпляра однопортовой. На результирующие характеристики устройства это не повлияло.

2.8. Решение а5

2.8.1. Директивы

Ниже приведены директивы, установленные для данного решения.

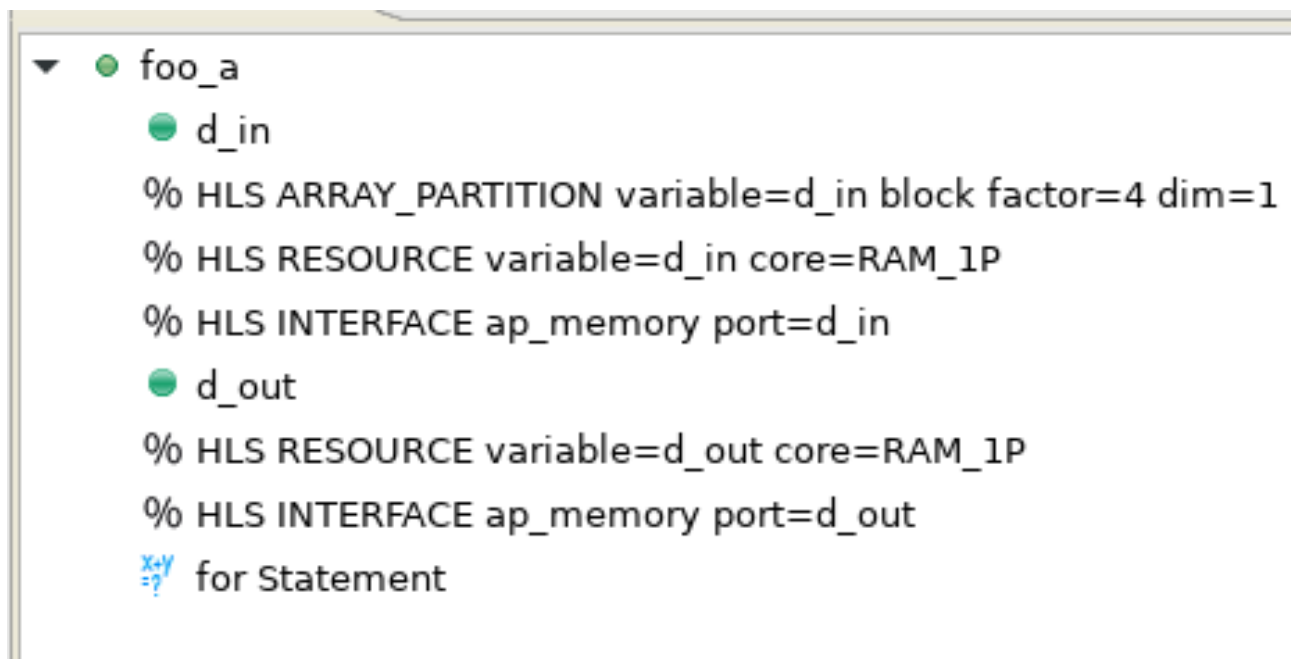


Рис. 2.35. Директивы

2.8.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
13	13	13	13	none

Рис. 2.36. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	21
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	36
Register	-	-	61	-
Total	0	2	61	57
Available	40	40	16000	8000
Utilization (%)	0	5	~0	~0

Рис. 2.37. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	13	-	14	-
Loop 1	no	12	3	-	4

Рис. 2.38. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_address0	out	2	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 2.39. Interface estimates



Рис. 2.40. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3
1	I/O Ports				
2	d_in_1(p0)		read		
3	d_in_2(p0)		read		
4	d_in_0(p0)		read		
5	d_in_3(p0)		read		
6	d_out(p0)				write
7	Memory Ports				
8	d_in_2(p0)		read		
9	d_in_3(p0)		read		
10	d_in_0(p0)		read		
11	d_in_1(p0)		read		
12	d_out(p0)				write
13	Expressions				
14	i_1_fu_124		+		
15	i_phi_fu_111		phi_mux		
16	exitcond_fu_118		icmp		
17	tmp_4_fu_138			*	
18	grp_fu_144				+

Рис. 2.41. Resource viewer

2.8.3. Анализ решения

Затраты по времени (задержка и интервал инициализации) сократились на 4 такта. Это исходит из того факта, что теперь используется 4 экземпляра однопортовой памяти и

все 4 операции чтения проходят одновременно. Это сокращает выполнение одной итерации на 1, что и создает улучшение в задержке на 4 такта.

2.9. Решение а6

2.9.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```

▼ ● foo_a
  ● d_in
    % HLS ARRAY_PARTITION variable=d_in block factor=2 dim=1
    % HLS RESOURCE variable=d_in core=RAM_2P
    % HLS INTERFACE ap_memory port=d_in
  ● d_out
    % HLS RESOURCE variable=d_out core=RAM_2P
    % HLS INTERFACE ap_memory port=d_out
  x=y
  =? for Statement

```

Рис. 2.42. Директивы

2.9.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

☐ Timing (ns)

☐ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

☐ Latency (clock cycles)

☐ Summary

Latency		Interval		
min	max	min	max	Type
13	13	13	13	none

Рис. 2.43. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	32
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	36
Register	-	-	61	-
Total	0	2	61	68
Available	40	40	16000	8000
Utilization (%)	0	5	~0	~0

Рис. 2.44. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	13	-	14	-
Loop 1	no	12	3	-	4

Рис. 2.45. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_0_address0	out	3	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_0_address1	out	3	ap_memory	d_in_0	array
d_in_0_ce1	out	1	ap_memory	d_in_0	array
d_in_0_q1	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	3	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_1_address1	out	3	ap_memory	d_in_1	array
d_in_1_ce1	out	1	ap_memory	d_in_1	array
d_in_1_q1	in	16	ap_memory	d_in_1	array
d_out_address1	out	2	ap_memory	d_out	array
d_out_ce1	out	1	ap_memory	d_out	array
d_out_we1	out	1	ap_memory	d_out	array
d_out_d1	out	16	ap_memory	d_out	array

Рис. 2.46. Interface estimates



Рис. 2.47. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3
1	I/O Ports				
2	d_in_0(p0)		read		
3	d_in_1(p0)		read		
4	d_in_1(p1)		read		
5	d_in_0(p1)		read		
6	d_out(p1)				write
7	Memory Ports				
8	d_in_0(p0)		read		
9	d_in_1(p1)		read		
10	d_in_1(p0)		read		
11	d_in_0(p1)		read		
12	d_out(p1)				write
13	Expressions				
14	i_phi_fu_109		phi_mux		
15	i_1_fu_122		+		
16	newIndex_trunc8_fu_134		^		
17	exitcond_fu_116		icmp		
18	tmp_4_fu_146			*	
19	grp_fu_152				+

Рис. 2.48. Resource viewer

2.9.3. Анализ решения

Задержка и интервал инициализации оказались одинаковыми. Использовать 4 однопортовых памяти или 2 двухпортовых оказалось практически одинаково. Разницу можно

заметить только в том, что в решении ба есть еще одна небольшая операция получения новых индексов для чтения из памяти. Однако эта операция очень быстрая, и все 4 чтения происходят практически одновременно.

2.10. Решение а7

2.10.1. Директивы

Ниже приведены директивы, установленные для данного решения.

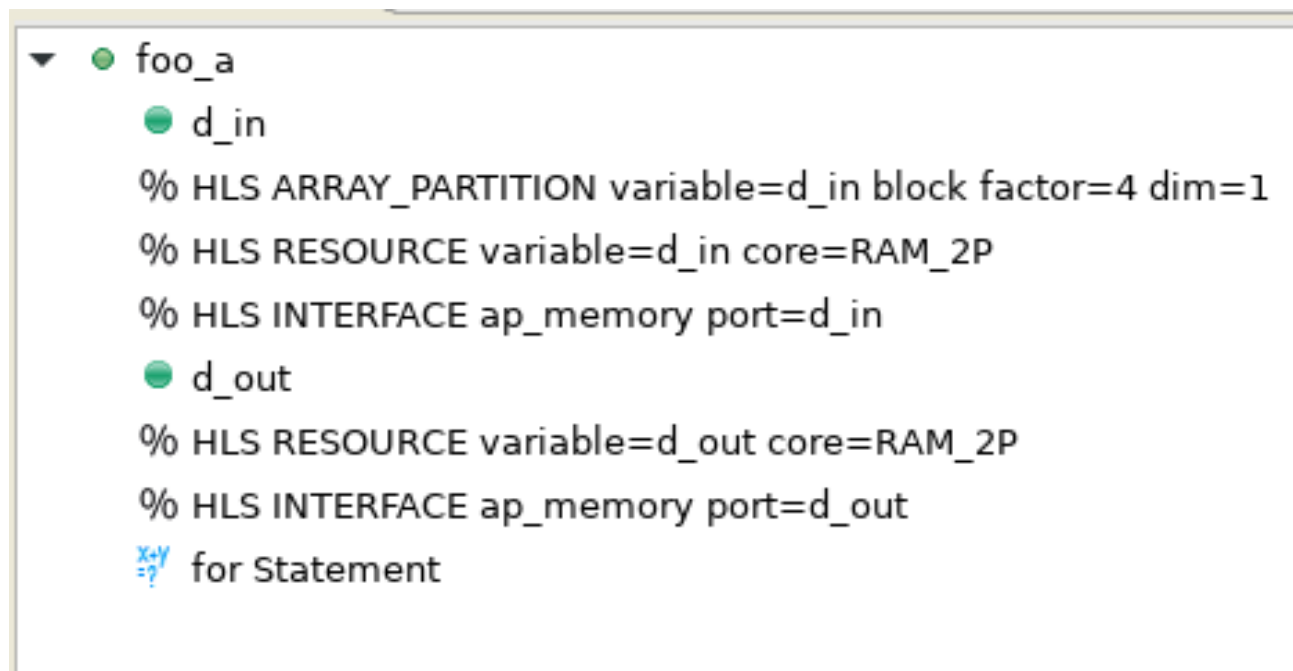


Рис. 2.49. Директивы

2.10.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
13	13	13	13	none

Рис. 2.50. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	2	-	-
Expression	-	-	0	21
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	36
Register	-	-	61	-
Total	0	2	61	57
Available	40	40	16000	8000
Utilization (%)	0	5	~0	~0

Рис. 2.51. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_a	-	13	-	14	-
Loop 1	no	12	3	-	4

Рис. 2.52. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_a	return value
ap_rst	in	1	ap_ctrl_hs	foo_a	return value
ap_start	in	1	ap_ctrl_hs	foo_a	return value
ap_done	out	1	ap_ctrl_hs	foo_a	return value
ap_idle	out	1	ap_ctrl_hs	foo_a	return value
ap_ready	out	1	ap_ctrl_hs	foo_a	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_address1	out	2	ap_memory	d_out	array
d_out_ce1	out	1	ap_memory	d_out	array
d_out_we1	out	1	ap_memory	d_out	array
d_out_d1	out	16	ap_memory	d_out	array

Рис. 2.53. Interface estimates



Рис. 2.54. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3
1	<input type="checkbox"/> I/O Ports				
2	d_in_1(p0)		read		
3	d_in_2(p0)		read		
4	d_in_3(p0)		read		
5	d_in_0(p0)		read		
6	d_out(p1)				write
7	<input type="checkbox"/> Memory Ports				
8	d_in_0(p0)		read		
9	d_in_1(p0)		read		
10	d_in_3(p0)		read		
11	d_in_2(p0)		read		
12	d_out(p1)				write
13	<input type="checkbox"/> Expressions				
14	i_phi_fu_115		phi_mux		
15	i_1_fu_128		+		
16	exitcond_fu_122		icmp		
17	tmp_4_fu_142			*	
18	grp_fu_148				+

Рис. 2.55. Resource viewer

2.10.3. Анализ решения

Видно, что задержки остались прежними. Поведение по диаграмме аналогично решению 5а. Вообще это решение аналогично по всем параметрам 5а, что означает, что для данного устройства нет необходимость иметь 4 двухпортовых памяти, и программа автоматически создала 4 однопортовых, как более экономичное решение.

2.11. Вывод

Ниже приведено сравнение всех решений первой части.

	RAM_1P	RAM_2P	BRAM_1P
Без block	2a/25/25	3a/17/17	1a/25/25
block; factor =2	4a/17/17	6a/13/13	
block; factor =4	5a/13/13	7a/13/13	

Рис. 2.56. Сравнение решений

По таблице видно, что наименьшие задержки получили те решения, где есть возможность осуществлять более 4 чтений массива памяти. Решение, которое имеет больше 4 чтений (7а), реализуется как 5а, так как оно избыточно, ведь в устройстве на каждой итерации всего 4 операции чтения. Такая избыточность может понадобиться только если использовать конвейеризацию, а пока все 4 итерации проходят последовательно, этого не требуется.

3. Часть 2

3.1. Исходный код

Для выполнения работы был написан код устройства и код теста, которые приведены ниже.

```
1 #include "lab9_2_2.h"
2
3 void foo_b(short d_in[N], short d_out[N/4 + 3]) {
4     for (short i=0; i<N/4; i++){
5         d_out[i] = d_in[i]*d_in[i+4];
6         d_out[i+1] = d_in[i+8]*d_in[i+12];
7         d_out[i+2] = d_in[i]*d_in[i+12];
8         d_out[i+3] = d_in[i+4]*d_in[i+8];
9     }
10 }
```

Рис. 3.1. Код устройства

```

1 #include <stdio.h>
2 #include "lab9_2_2.h"
3
4
5 void generate_test_data(short scale, short d_in[N], short d_out[N/4 + 3]) {
6     short i;
7     for (i = 0; i < N; ++i){
8         d_in[i] = (i + 1) * scale;
9     }
10
11     for (i=0; i<N/4; i++){
12         d_out[i] = d_in[i]*d_in[i+4];
13         d_out[i+1] = d_in[i+8]*d_in[i+12];
14         d_out[i+2] = d_in[i]*d_in[i+12];
15         d_out[i+3] = d_in[i+4]*d_in[i+8];
16     }
17 }
18
19 int compare_array_eq(short actual[N/4 + 3], short expected[N/4 + 3]) {
20     for (int i = 0; i < N/4 + 3; ++i) {
21         if (actual[i] != expected[i]) {
22             fprintf(stdout, "%d: Expected %d Actual %d\n", i, expected[i], actual[i]);
23             ↪ return 0;
24         }
25     }
26     return 1;
27 }
28
29 int main() {
30     int pass = 1;
31     short d_in[N];
32     short d_out[N/4 + 3];
33     short expected_out[N/4 + 3];
34
35
36     for (int i = 1; i < 4; ++i) {
37         generate_test_data(i, d_in, expected_out);
38
39         foo_b(d_in, d_out);
40
41         if (!compare_array_eq(d_out, expected_out)){
42             pass = 0;
43         }
44     }
45
46
47     if (pass) {
48         fprintf(stdout, "—————Pass!—————\n");
49         return 0;
50     } else {
51         fprintf(stderr, "—————Fail!—————\n");
52         return 1;
53     }
54 }

```

Рис. 3.2. Код теста

```
1 #define N 16
```

Рис. 3.3. Заголовочный файл

3.2. Скрипт выполнения

Для автоматизации выполнения работы был написан следующий скрипт:

```
1 open_project -reset lab9_2_2
2 add_files lab9_2_2.c
3 set_top foo_b
4 add_files -tb lab9_2_2_test.c
5
6 set solutions      [list 1b 2b 3b 4b 5b 6b 7b 8b]
7
8 foreach sol $solutions {
9   open_solution solution_$sol -reset
10  set_part {xa7a12tcs325-1q}
11  create_clock -period 10
12  set_clock_uncertainty 0.1
13
14  if {$sol == "1b"} {
15    set_directive_interface -mode ap_memory  foo_b d_in
16    set_directive_resource -core RAM_1P foo_b d_in
17    set_directive_interface -mode ap_memory  foo_b d_out
18    set_directive_resource -core RAM_1P foo_b d_out
19  }
20  if {$sol == "2b"} {
21    set_directive_interface -mode ap_memory  foo_b d_in
22    set_directive_resource -core RAM_1P foo_b d_in
23    set_directive_interface -mode ap_memory  foo_b d_out
24    set_directive_resource -core RAM_1P foo_b d_out
25    set_directive_array_partition -type block -factor 4 foo_b d_in
26  }
27  if {$sol == "3b"} {
28    set_directive_interface -mode ap_memory  foo_b d_in
29    set_directive_resource -core RAM_1P foo_b d_in
30    set_directive_interface -mode ap_memory  foo_b d_out
31    set_directive_resource -core RAM_1P foo_b d_out
32    set_directive_array_partition -type block -factor 4 foo_b d_in
33    set_directive_array_partition -type block -factor 2 foo_b d_out
34  }
```

Рис. 3.4. Скрипт выполнения, часть 1

```

1  if {$sol == "4b"} {
2      set_directive_interface -mode ap_memory  foo_b d_in
3      set_directive_resource -core RAM_1P foo_b d_in
4      set_directive_interface -mode ap_memory  foo_b d_out
5      set_directive_resource -core RAM_1P foo_b d_out
6      set_directive_array_partition -type block -factor 4 foo_b d_in
7      set_directive_array_partition -type block -factor 4 foo_b d_out
8  }
9  if {$sol == "5b"} {
10     set_directive_interface -mode ap_memory  foo_b d_in
11     set_directive_resource -core RAM_1P foo_b d_in
12     set_directive_interface -mode ap_memory  foo_b d_out
13     set_directive_resource -core RAM_2P foo_b d_out
14     set_directive_array_partition -type block -factor 4 foo_b d_in
15     set_directive_array_partition -type cyclic -factor 1 foo_b d_out
16 }
17 if {$sol == "6b"} {
18     set_directive_interface -mode ap_memory  foo_b d_in
19     set_directive_resource -core RAM_1P foo_b d_in
20     set_directive_interface -mode ap_memory  foo_b d_out
21     set_directive_resource -core RAM_2P foo_b d_out
22     set_directive_array_partition -type block -factor 4 foo_b d_in
23     set_directive_array_partition -type cyclic -factor 2 foo_b d_out
24 }
25 if {$sol == "7b"} {
26     set_directive_interface -mode ap_memory  foo_b d_in
27     set_directive_resource -core RAM_1P foo_b d_in
28     set_directive_interface -mode ap_memory  foo_b d_out
29     set_directive_resource -core RAM_2P foo_b d_out
30     set_directive_array_partition -type block -factor 4 foo_b d_in
31     set_directive_array_partition -type cyclic -factor 4 foo_b d_out
32 }
33 if {$sol == "8b"} {
34     set_directive_interface -mode ap_memory  foo_b d_in
35     set_directive_resource -core RAM_1P foo_b d_in
36     set_directive_interface -mode ap_memory  foo_b d_out
37     set_directive_resource -core RAM_1P foo_b d_out
38     set_directive_array_partition -type block -factor 4 foo_b d_in
39     set_directive_array_partition -type complete foo_b d_out
40 }
41
42 csim_design
43 csynth_design
44 }
45
46 exit

```

Рис. 3.5. Скрипт выполнения, часть 2

3.3. Моделирование

Ниже приведены результаты моделирования, по которым видно, что тест проходит успешно.

```
INFO: [APCC 202-1] APCC is done.  
Generating csim.exe  
-----Pass!-----  
INFO: [SIM 211-1] CSim done with 0 errors.  
INFO: [SIM 211-3] ***** CSIM finish *****
```

Рис. 3.6. Результаты моделирования

3.4. Решение b1

3.4.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```
▼ ● foo_b  
    ● d_in  
    % HLS RESOURCE variable=d_in core=RAM_1P  
    % HLS INTERFACE ap_memory port=d_in  
    ● d_out  
    % HLS RESOURCE variable=d_out core=RAM_1P  
    % HLS INTERFACE ap_memory port=d_out  
    x=y  
    =? for Statement
```

Рис. 3.7. Директивы

3.4.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
29	29	29	29	none

Рис. 3.8. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	56
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	134
Register	-	-	116	-
Total	0	4	116	190
Available	40	40	16000	8000
Utilization (%)	0	10	~0	2

Рис. 3.9. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_b	-	29	-	30	-
Loop 1	no	28	7	-	4

Рис. 3.10. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_breturn value	
ap_rst	in	1	ap_ctrl_hs	foo_breturn value	
ap_start	in	1	ap_ctrl_hs	foo_breturn value	
ap_done	out	1	ap_ctrl_hs	foo_breturn value	
ap_idle	out	1	ap_ctrl_hs	foo_breturn value	
ap_ready	out	1	ap_ctrl_hs	foo_breturn value	
d_in_address0	out	4	ap_memory	d_in	array
d_in_ce0	out	1	ap_memory	d_in	array
d_in_q0	in	16	ap_memory	d_in	array
d_out_address0	out	3	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 3.11. Interface estimates

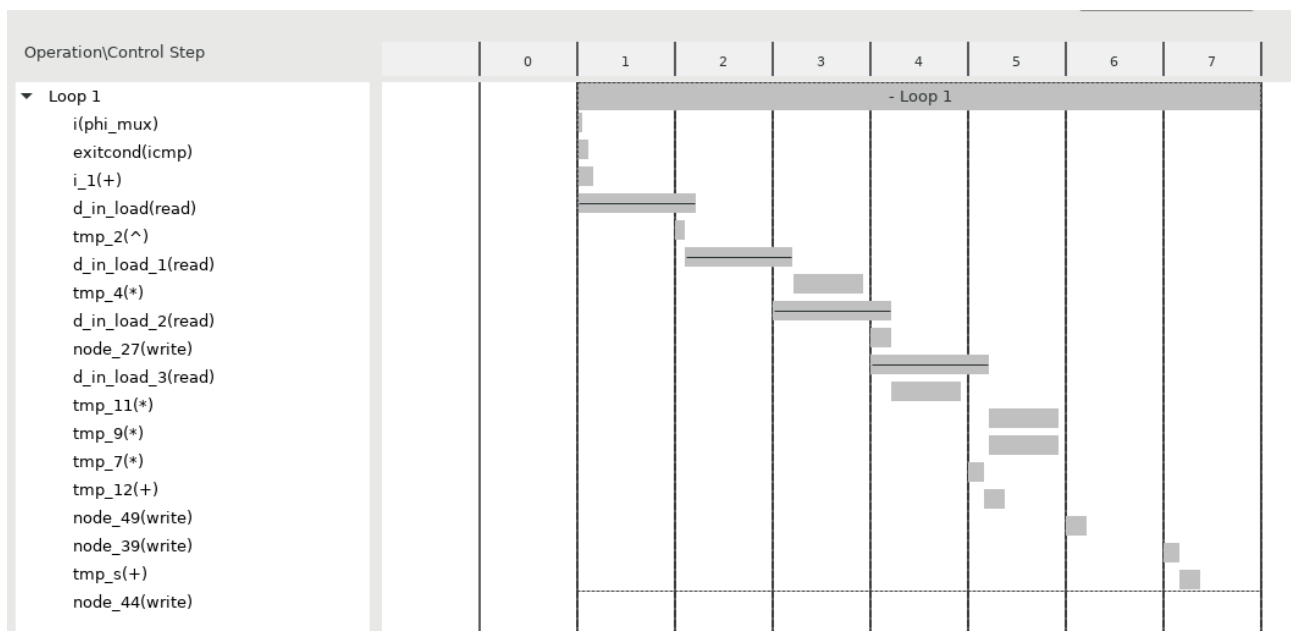


Рис. 3.12. Scheduler viewer

На рисунке выше видны блоки, которые не укладываются во временной интервал.

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6	C7
1	I/O Ports								
2	d_in(p0)		read	read	read	read			
3	d_out(p0)					write	write	write	write
4	Memory Ports								
5	d_in(p0)		read	read	read	read			
6	d_out(p0)					write	write	write	write
7	Expressions								
8	i_1_fu_140		+						
9	i_phi_fu_122		phi_mux						
10	exitcond_fu_134		icmp						
11	tmp_2_fu_151			^					
12	tmp_4_fu_209				*				
13	tmp_11_fu_214					*			
14	tmp_12_fu_183						+		
15	tmp_9_fu_220						*		
16	tmp_7_fu_226						*		
17	tmp_s_fu_198								+

Рис. 3.13. Resource viewer

3.4.3. Анализ решения

Согласно ожиданиям устройство использует 4 умножителя, а также имеет большие задержки, так как 4 чтения выполняются последовательно.

3.5. Решение b2

3.5.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```

▼ ● foo_b
  ● d_in
    % HLS ARRAY_PARTITION variable=d_in block factor=4 dim=1
    % HLS RESOURCE variable=d_in core=RAM_1P
    % HLS INTERFACE ap_memory port=d_in
  ● d_out
    % HLS RESOURCE variable=d_out core=RAM_1P
    % HLS INTERFACE ap_memory port=d_out
  X-Y
  =? for Statement

```

Рис. 3.14. Директивы

3.5.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

[-] Timing (ns)

[-] Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

[-] Latency (clock cycles)

[-] Summary

Latency		Interval		
min	max	min	max	Type
25	25	25	25	none

Рис. 3.15. Performance estimates

Utilization Estimates				
[-] Summary				
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	45
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	104
Register	-	-	112	-
Total	0	4	112	149
Available	40	40	16000	8000
Utilization (%)	0	10	~0	1

Рис. 3.16. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
▼ foo_b	-	25	-	26	-
○ Loop 1	no	24	6	-	4

Рис. 3.17. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_b	return value
ap_rst	in	1	ap_ctrl_hs	foo_b	return value
ap_start	in	1	ap_ctrl_hs	foo_b	return value
ap_done	out	1	ap_ctrl_hs	foo_b	return value
ap_idle	out	1	ap_ctrl_hs	foo_b	return value
ap_ready	out	1	ap_ctrl_hs	foo_b	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_address0	out	3	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 3.18. Interface estimates

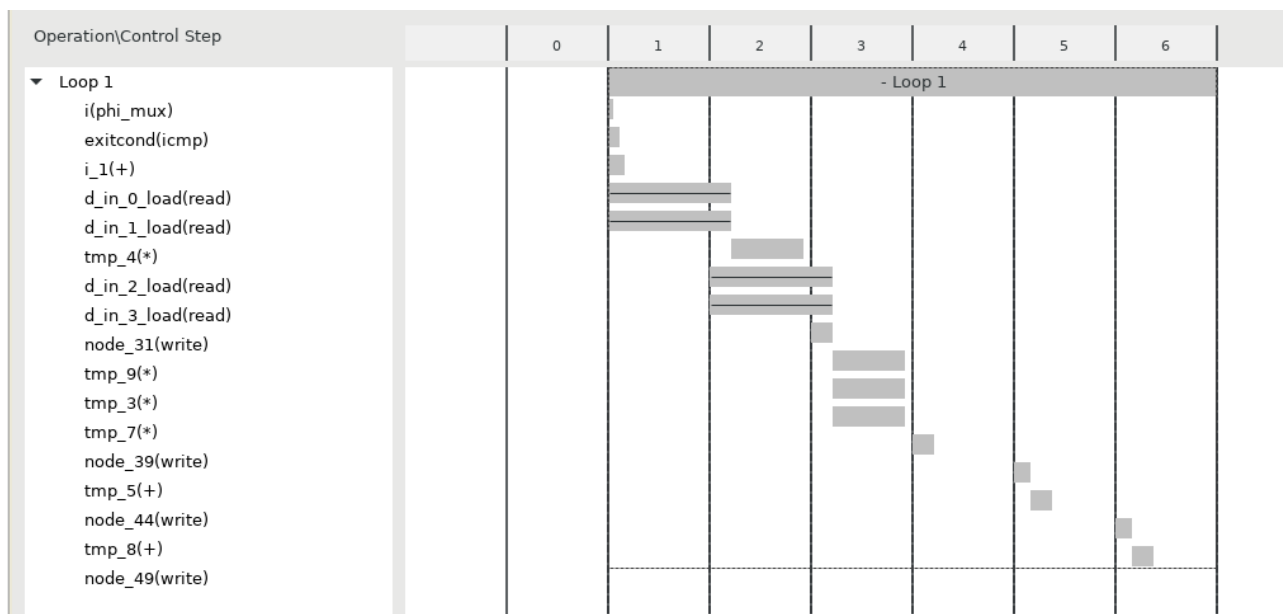


Рис. 3.19. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6
1	I/O Ports							
2	d_in_0(p0)		read					
3	d_in_1(p0)		read					
4	d_in_3(p0)			read				
5	d_in_2(p0)			read				
6	d_out(p0)			write	write	write	write	
7	Memory Ports							
8	d_in_1(p0)		read					
9	d_in_0(p0)		read					
10	d_in_2(p0)			read				
11	d_in_3(p0)			read				
12	d_out(p0)			write	write	write	write	
13	Expressions							
14	i_1_fu_153		+					
15	i_phi_fu_139		phi_mux					
16	exitcond_fu_147		icmp					
17	tmp_4_fu_191			*				
18	tmp_3_fu_203				*			
19	tmp_9_fu_197				*			
20	tmp_7_fu_208				*			
21	tmp_5_fu_169					+		
22	tmp_8_fu_180							+

Рис. 3.20. Resource viewer

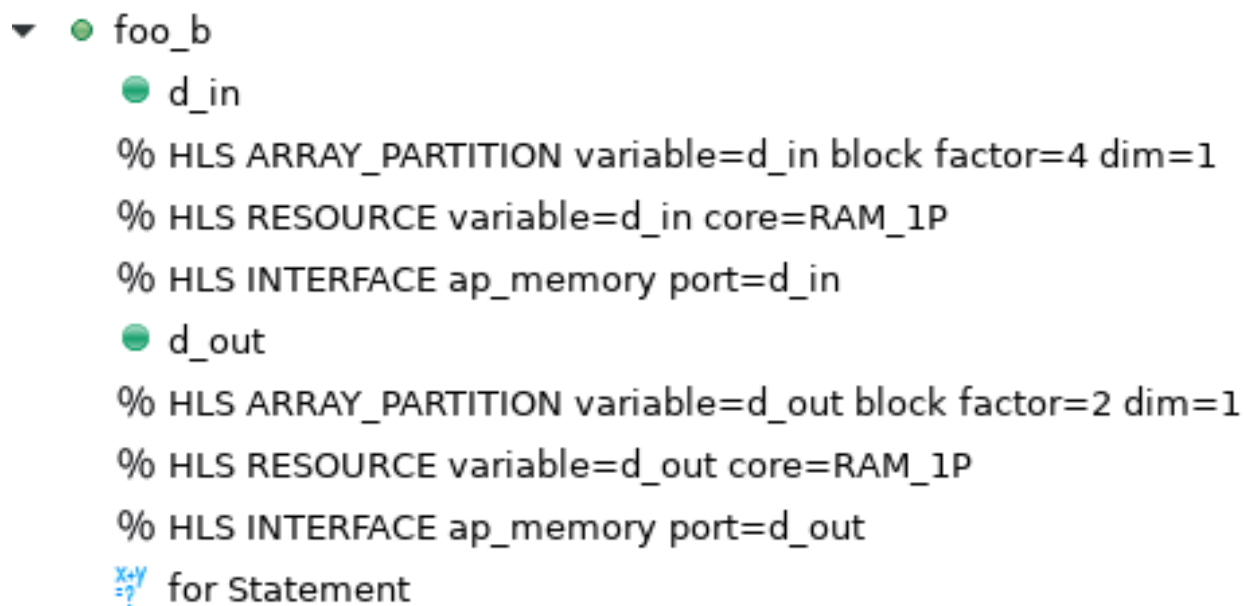
3.5.3. Анализ решения

В сравнении с предыдущим решением, количество умножителей не изменилось, однако за счет возможности двух параллельных чтений (хотя их 4 на самом деле) сократились задержки (цикл за 6 тактов, а не за 7).

3.6. Решение b3

3.6.1. Директивы

Ниже приведены директивы, установленные для данного решения.



The screenshot shows a code editor with a tree view on the left and a code editor on the right. The tree view shows a folder 'foo_b' containing two files: 'd_in' and 'd_out'. The code editor shows the following directives:

```
% HLS ARRAY_PARTITION variable=d_in block factor=4 dim=1
% HLS RESOURCE variable=d_in core=RAM_1P
% HLS INTERFACE ap_memory port=d_in
% HLS ARRAY_PARTITION variable=d_out block factor=2 dim=1
% HLS RESOURCE variable=d_out core=RAM_1P
% HLS INTERFACE ap_memory port=d_out
for Statement
```

Рис. 3.21. Директивы

3.6.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
25	25	25	25	none

Detail

+ Instance

+ Loop

Рис. 3.22. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	75
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	146
Register	-	-	159	-
Total	0	4	159	221
Available	40	40	16000	8000
Utilization (%)	0	10	~0	2

Рис. 3.23. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
▼ foo_b	-	25	-	26	-
Loop 1	no	24	6	-	4

Рис. 3.24. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_breturn value	
ap_rst	in	1	ap_ctrl_hs	foo_breturn value	
ap_start	in	1	ap_ctrl_hs	foo_breturn value	
ap_done	out	1	ap_ctrl_hs	foo_breturn value	
ap_idle	out	1	ap_ctrl_hs	foo_breturn value	
ap_ready	out	1	ap_ctrl_hs	foo_breturn value	
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_0_address0	out	2	ap_memory	d_out_0	array
d_out_0_ce0	out	1	ap_memory	d_out_0	array
d_out_0_we0	out	1	ap_memory	d_out_0	array
d_out_0_d0	out	16	ap_memory	d_out_0	array
d_out_1_address0	out	2	ap_memory	d_out_1	array
d_out_1_ce0	out	1	ap_memory	d_out_1	array
d_out_1_we0	out	1	ap_memory	d_out_1	array
d_out_1_d0	out	16	ap_memory	d_out_1	array

Рис. 3.25. Interface estimates



Рис. 3.26. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6
1	I/O Ports							
2	d_in_1(p0)		read					
3	d_in_3(p0)		read					
4	d_in_0(p0)		read					
5	d_in_2(p0)		read					
6	d_out_0(p0)				write	write	write	write
7	d_out_1(p0)				write		write	write
8	Memory Ports							
9	d_in_0(p0)		read					
10	d_in_1(p0)		read					
11	d_in_3(p0)		read					
12	d_in_2(p0)		read					
13	d_out_0(p0)				write	write	write	write
14	d_out_1(p0)				write		write	write
15	Expressions							
16	i_1_fu_189		+					
17	i_phi_fu_175		phi_mux					
18	exitcond_fu_183		icmp					
19	tmp_4_fu_277			*				
20	tmp_9_fu_283			*				
21	newIndex_trunc_fu_215				+			
22	tmp_5_fu_227					+		
23	tmp_2_fu_289					*		
24	newIndex_trunc1_fu_233					^		
25	newIndex_trunc2_fu_258						+	
26	tmp_8_fu_252						+	
27	tmp_6_fu_293						*	

Рис. 3.27. Resource viewer

3.6.3. Анализ решения

В данном решении 4 чтения выполняются одновременно, однако итерация по-прежнему выполняется долго – за 6 тактов. Это вызвано тем, что устройство не может одновременно писать 4 значения в выходной массив.

3.7. Решение b4

3.7.1. Директивы

Ниже приведены директивы, установленные для данного решения.

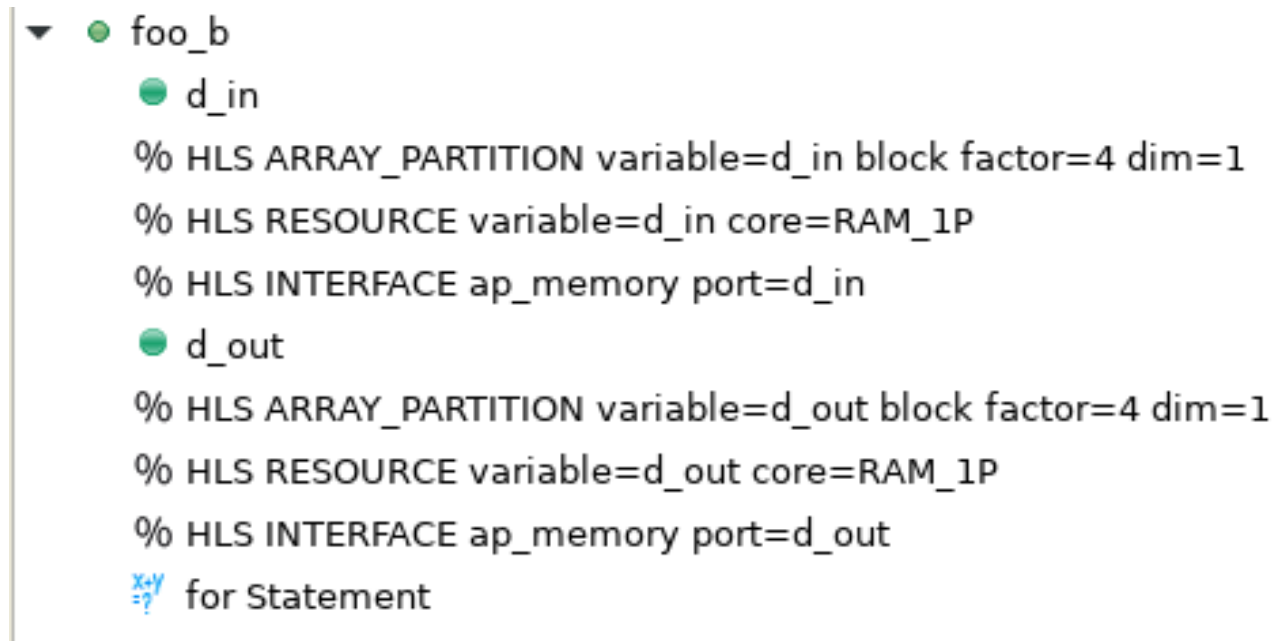


Рис. 3.28. Директивы

3.7.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
29	29	29	29	none

Рис. 3.29. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	70
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	161
Register	-	-	161	-
Total	0	4	161	231
Available	40	40	16000	8000
Utilization (%)	0	10	1	2

Рис. 3.30. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_b	-	29	-	30	-
Loop 1	no	28	7	-	4

Рис. 3.31. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_b	return value
ap_rst	in	1	ap_ctrl_hs	foo_b	return value
ap_start	in	1	ap_ctrl_hs	foo_b	return value
ap_done	out	1	ap_ctrl_hs	foo_b	return value
ap_idle	out	1	ap_ctrl_hs	foo_b	return value
ap_ready	out	1	ap_ctrl_hs	foo_b	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_0_address0	out	1	ap_memory	d_out_0	array
d_out_0_ce0	out	1	ap_memory	d_out_0	array
d_out_0_we0	out	1	ap_memory	d_out_0	array
d_out_0_d0	out	16	ap_memory	d_out_0	array
d_out_1_address0	out	1	ap_memory	d_out_1	array
d_out_1_ce0	out	1	ap_memory	d_out_1	array
d_out_1_we0	out	1	ap_memory	d_out_1	array
d_out_1_d0	out	16	ap_memory	d_out_1	array
d_out_2_address0	out	1	ap_memory	d_out_2	array
d_out_2_ce0	out	1	ap_memory	d_out_2	array
d_out_2_we0	out	1	ap_memory	d_out_2	array
d_out_2_d0	out	16	ap_memory	d_out_2	array
d_out_3_address0	out	1	ap_memory	d_out_3	array
d_out_3_ce0	out	1	ap_memory	d_out_3	array
d_out_3_we0	out	1	ap_memory	d_out_3	array
d_out_3_d0	out	16	ap_memory	d_out_3	array

Рис. 3.32. Interface estimates

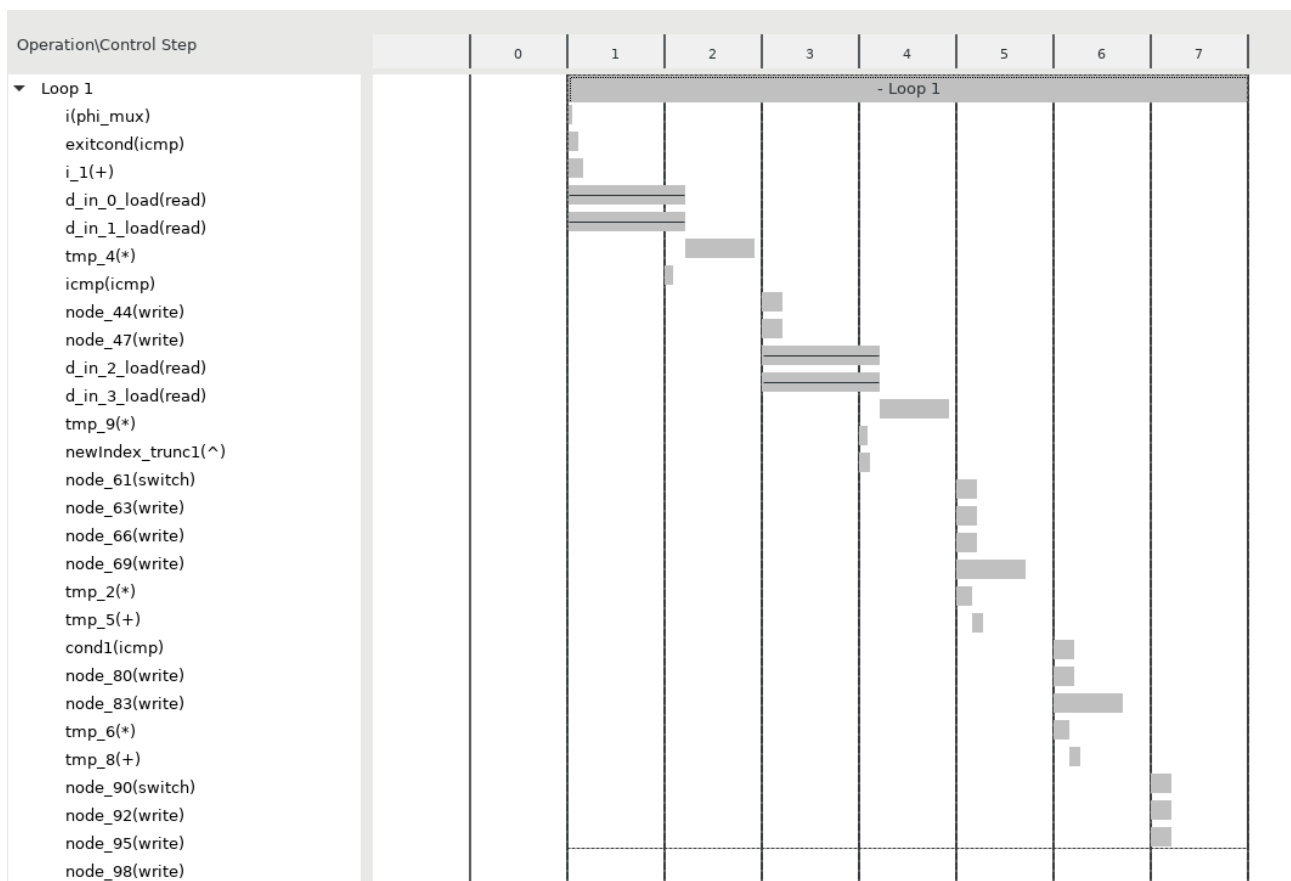


Рис. 3.33. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6	C7
1	I/O Ports								
2	d_in_1(p0)		read						
3	d_in_0(p0)		read						
4	d_in_3(p0)				read				
5	d_out_1(p0)				write		write	write	write
6	d_out_0(p0)				write		write		
7	d_in_2(p0)				read				
8	d_out_2(p0)						write	write	write
9	d_out_3(p0)								write
10	Memory Ports								
11	d_in_1(p0)		read						
12	d_in_0(p0)		read						
13	d_in_3(p0)				read				
14	d_out_1(p0)				write		write	write	write
15	d_out_0(p0)				write		write		
16	d_in_2(p0)				read				
17	d_out_2(p0)						write	write	write
18	d_out_3(p0)								write
19	Expressions								
20	i_1_fu_209		+						
21	i_phi_fu_195		phi_mux						
22	exitcond_fu_203		icmp						
23	tmp_4_fu_314			*					
24	icmp_fu_241			icmp					
25	tmp_9_fu_320					*			
26	newIndex_trunc1_fu_256					^			
27	tmp_5_fu_268						+		
28	tmp_2_fu_326						*		
29	cond1_fu_292						icmp		
30	tmp_8_fu_298							+	
31	tmp_6_fu_330							*	

Рис. 3.34. Resource viewer

3.7.3. Анализ решения

Можно сказать, что при получении возможности на запись 4 памяти сразу, все стало только хуже. В устройстве используется очень много перезаписей в регистры, а не сразу в выходные линии, что не дает ни делать 4 чтения, ни записывать 4 значения на выходы.

3.8. Решение b5

3.8.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```

▼ ● foo_b
  ● d_in
    % HLS ARRAY_PARTITION variable=d_in block factor=4 dim=1
    % HLS RESOURCE variable=d_in core=RAM_1P
    % HLS INTERFACE ap_memory port=d_in
  ● d_out
    % HLS ARRAY_PARTITION variable=d_out cyclic factor=1 dim=1
    % HLS RESOURCE variable=d_out core=RAM_2P
    % HLS INTERFACE ap_memory port=d_out
    x=y
    =? for Statement

```

Рис. 3.35. Директивы

3.8.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
25	25	25	25	none

Рис. 3.36. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	45
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	104
Register	-	-	112	-
Total	0	4	112	149
Available	40	40	16000	8000
Utilization (%)	0	10	~0	1

Рис. 3.37. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_b	-	25	-	26	-
Loop 1	no	24	6	-	4

Рис. 3.38. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_b	return value
ap_rst	in	1	ap_ctrl_hs	foo_b	return value
ap_start	in	1	ap_ctrl_hs	foo_b	return value
ap_done	out	1	ap_ctrl_hs	foo_b	return value
ap_idle	out	1	ap_ctrl_hs	foo_b	return value
ap_ready	out	1	ap_ctrl_hs	foo_b	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_address1	out	3	ap_memory	d_out	array
d_out_ce1	out	1	ap_memory	d_out	array
d_out_we1	out	1	ap_memory	d_out	array
d_out_d1	out	16	ap_memory	d_out	array

Рис. 3.39. Interface estimates



Рис. 3.40. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6
1	I/O Ports							
2	d_in_1(p0)		read					
3	d_in_0(p0)		read					
4	d_in_2(p0)			read				
5	d_in_3(p0)			read				
6	d_out(p1)				write	write	write	write
7	Memory Ports							
8	d_in_0(p0)		read					
9	d_in_1(p0)		read					
10	d_in_2(p0)			read				
11	d_in_3(p0)			read				
12	d_out(p1)				write	write	write	write
13	Expressions							
14	i_phi_fu_145		phi_mux					
15	i_l_fu_159		+					
16	exitcond_fu_153		icmp					
17	tmp_4_fu_197			*				
18	tmp_3_fu_209				*			
19	tmp_9_fu_203				*			
20	tmp_7_fu_214				*			
21	tmp_5_fu_175						+	
22	tmp_8_fu_186							+

Рис. 3.41. Resource viewer

3.8.3. Анализ решения

Данное решение с одной двух портовой памятью вышло эффективнее, чем 4 однопортовых. Тем не менее, одновременно 2 записи все также не производится.

3.9. Решение b6

3.9.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```
▼ ● foo_b
  ● d_in
    % HLS ARRAY_PARTITION variable=d_in block factor=4 dim=1
    % HLS RESOURCE variable=d_in core=RAM_1P
    % HLS INTERFACE ap_memory port=d_in
  ● d_out
    % HLS ARRAY_PARTITION variable=d_out cyclic factor=2 dim=1
    % HLS RESOURCE variable=d_out core=RAM_2P
    % HLS INTERFACE ap_memory port=d_out
    x=y
    =? for Statement
```

Рис. 3.42. Директивы

3.9.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

☐ Timing (ns)

☐ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

☐ Latency (clock cycles)

☐ Summary

Latency		Interval		
min	max	min	max	Type
29	29	29	29	none

Рис. 3.43. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	45
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	161
Register	-	-	164	-
Total	0	4	164	206
Available	40	40	16000	8000
Utilization (%)	0	10	1	2

Рис. 3.44. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_b	-	29	-	30	-
Loop 1	no	28	7	-	4

Рис. 3.45. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_b	return value
ap_rst	in	1	ap_ctrl_hs	foo_b	return value
ap_start	in	1	ap_ctrl_hs	foo_b	return value
ap_done	out	1	ap_ctrl_hs	foo_b	return value
ap_idle	out	1	ap_ctrl_hs	foo_b	return value
ap_ready	out	1	ap_ctrl_hs	foo_b	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_0_address1	out	2	ap_memory	d_out_0	array
d_out_0_ce1	out	1	ap_memory	d_out_0	array
d_out_0_we1	out	1	ap_memory	d_out_0	array
d_out_0_d1	out	16	ap_memory	d_out_0	array
d_out_1_address1	out	2	ap_memory	d_out_1	array
d_out_1_ce1	out	1	ap_memory	d_out_1	array
d_out_1_we1	out	1	ap_memory	d_out_1	array
d_out_1_d1	out	16	ap_memory	d_out_1	array

Рис. 3.46. Interface estimates

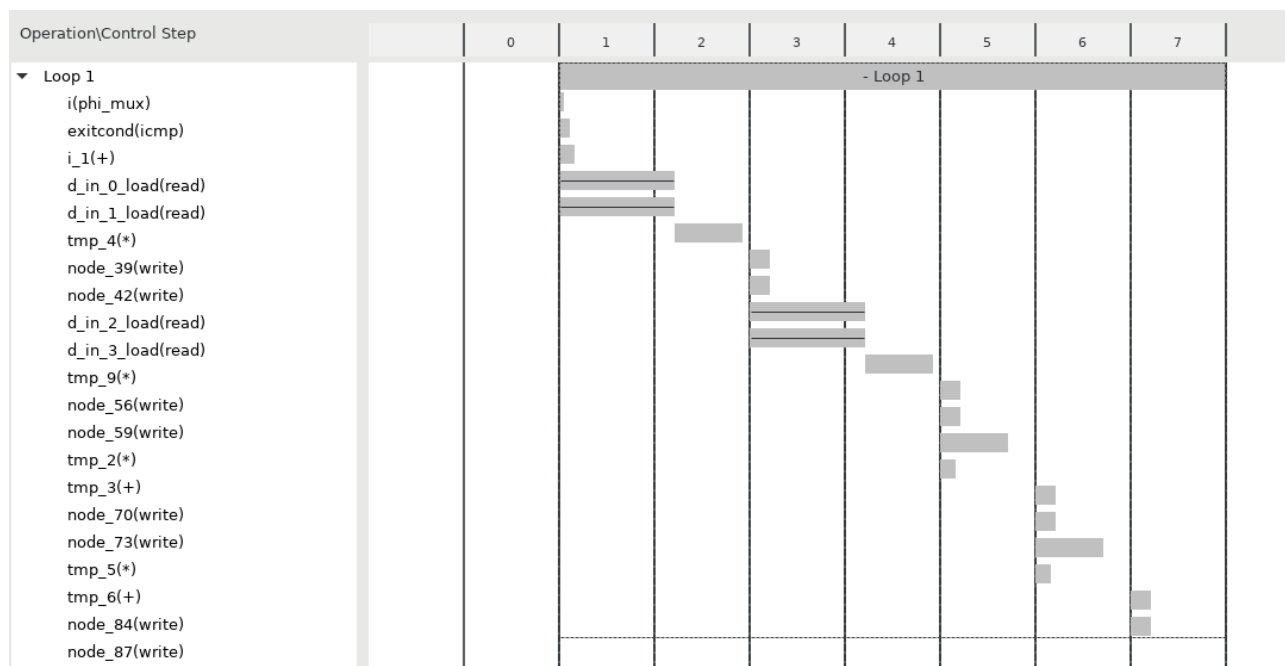


Рис. 3.47. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6	C7
1	I/O Ports								
2	d_in_0(p0)		read						
3	d_in_1(p0)		read						
4	d_out_1(p1)				write		write	write	write
5	d_in_3(p0)				read				
6	d_in_2(p0)				read				
7	d_out_0(p1)				write		write	write	write
8	Memory Ports								
9	d_in_1(p0)		read						
10	d_in_0(p0)		read						
11	d_in_2(p0)				read				
12	d_out_1(p1)				write		write	write	write
13	d_out_0(p1)				write		write	write	write
14	d_in_3(p0)				read				
15	Expressions								
16	i_phi_fu_186		phi_mux						
17	i_1_fu_200		+						
18	exitcond_fu_194		icmp						
19	tmp_4_fu_290			*					
20	tmp_9_fu_296					*			
21	tmp_3_fu_246						+		
22	tmp_2_fu_302						*		
23	tmp_6_fu_268							+	
24	tmp_5_fu_306							*	

Рис. 3.48. Resource viewer

3.9.3. Анализ решения

В данном решении, 4 чтения выполняются одновременно, а также присутствует одновременную запись в выходной массив, однако задержки остались прежними.

3.10. Решение b7

3.10.1. Директивы

Ниже приведены директивы, установленные для данного решения.

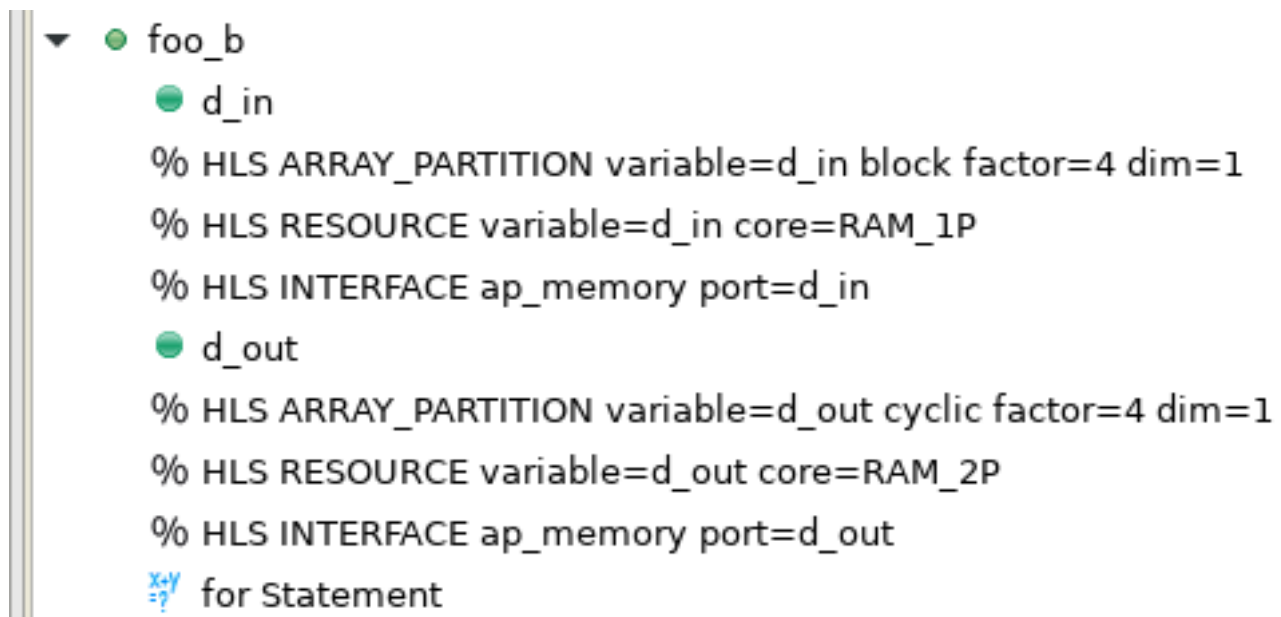


Рис. 3.49. Директивы

3.10.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

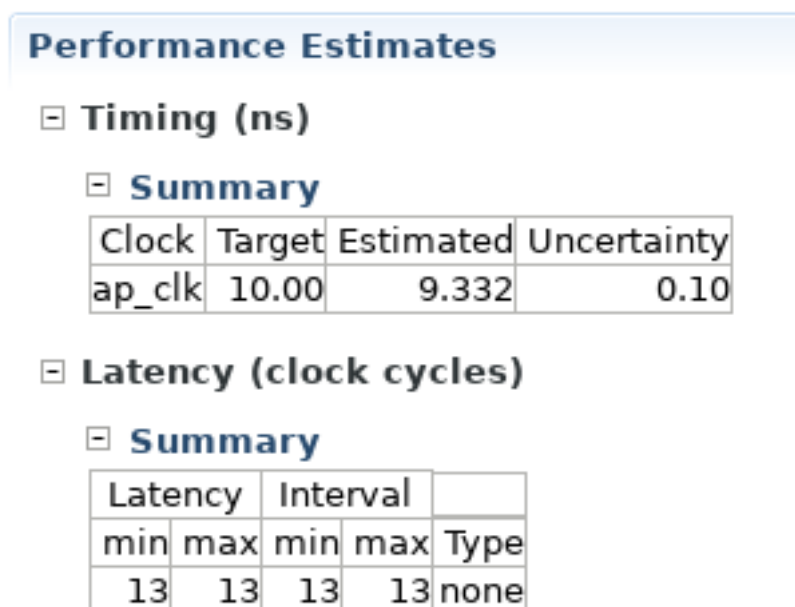


Рис. 3.50. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	45
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	252
Register	-	-	89	-
Total	0	4	89	297
Available	40	40	16000	8000
Utilization (%)	0	10	~0	3

Рис. 3.51. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_b	-	13	-	14	-
Loop 1	no	12	3	-	4

Рис. 3.52. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_b	return value
ap_rst	in	1	ap_ctrl_hs	foo_b	return value
ap_start	in	1	ap_ctrl_hs	foo_b	return value
ap_done	out	1	ap_ctrl_hs	foo_b	return value
ap_idle	out	1	ap_ctrl_hs	foo_b	return value
ap_ready	out	1	ap_ctrl_hs	foo_b	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_0_address1	out	1	ap_memory	d_out_0	array
d_out_0_ce1	out	1	ap_memory	d_out_0	array
d_out_0_we1	out	1	ap_memory	d_out_0	array
d_out_0_d1	out	16	ap_memory	d_out_0	array
d_out_1_address1	out	1	ap_memory	d_out_1	array
d_out_1_ce1	out	1	ap_memory	d_out_1	array
d_out_1_we1	out	1	ap_memory	d_out_1	array
d_out_1_d1	out	16	ap_memory	d_out_1	array
d_out_2_address1	out	1	ap_memory	d_out_2	array
d_out_2_ce1	out	1	ap_memory	d_out_2	array
d_out_2_we1	out	1	ap_memory	d_out_2	array
d_out_2_d1	out	16	ap_memory	d_out_2	array
d_out_3_address1	out	1	ap_memory	d_out_3	array
d_out_3_ce1	out	1	ap_memory	d_out_3	array
d_out_3_we1	out	1	ap_memory	d_out_3	array
d_out_3_d1	out	16	ap_memory	d_out_3	array

Рис. 3.53. Interface estimates

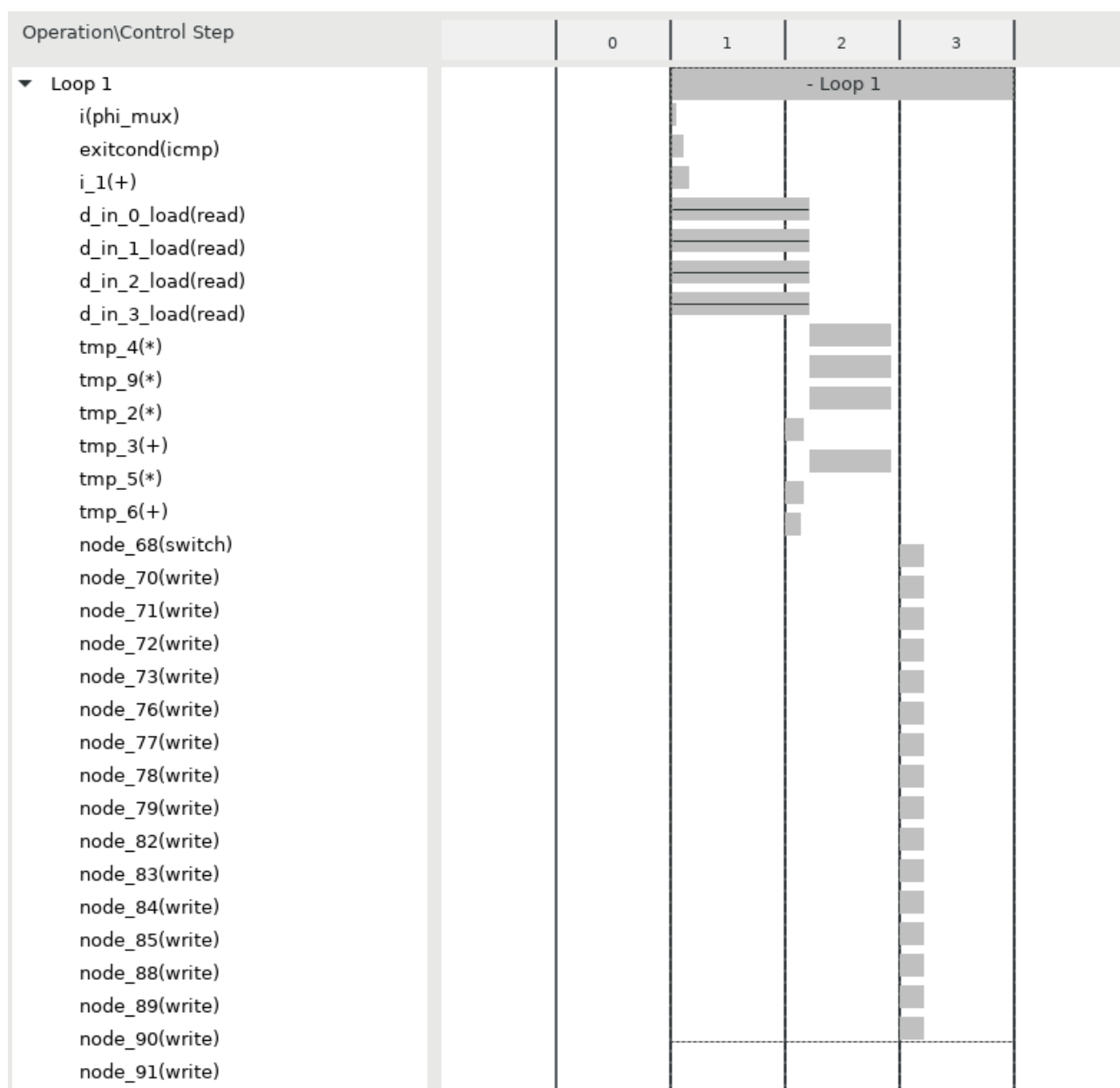


Рис. 3.54. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3
1	I/O Ports				
2	d_in_1(p0)		read		
3	d_in_0(p0)		read		
4	d_in_2(p0)		read		
5	d_in_3(p0)		read		
6	d_out_1(p1)				write
7	d_out_0(p1)				write
8	d_out_3(p1)				write
9	d_out_2(p1)				write
10	Memory Ports				
11	d_in_0(p0)		read		
12	d_in_3(p0)		read		
13	d_in_1(p0)		read		
14	d_in_2(p0)		read		
15	d_out_0(p1)				write
16	d_out_2(p1)				write
17	d_out_3(p1)				write
18	d_out_1(p1)				write
19	Expressions				
20	i_phi_fu_272		phi_mux		
21	i_l_fu_286		+		
22	exitcond_fu_280		icmp		
23	tmp_6_fu_341			+	
24	tmp_3_fu_319			+	
25	tmp_9_fu_369			*	
26	tmp_4_fu_363			*	
27	tmp_2_fu_375			*	
28	tmp_5_fu_381			*	

Рис. 3.55. Resource viewer

3.10.3. Анализ решения

Данное решение аналогично с RAM_1 хуже, чем предыдущее (с factor 2).

3.11. Решение b8

3.11.1. Директивы

Ниже приведены директивы, установленные для данного решения.

```
▼ ● foo_b
  ● d_in
  % HLS ARRAY_PARTITION variable=d_in block factor=4 dim=1
  % HLS RESOURCE variable=d_in core=RAM_1P
  % HLS INTERFACE ap_memory port=d_in
  ● d_out
  % HLS ARRAY_PARTITION variable=d_out complete dim=1
  % HLS RESOURCE variable=d_out core=RAM_1P
  % HLS INTERFACE ap_memory port=d_out
  x=y
  =? for Statement
```

Рис. 3.56. Директивы

3.11.2. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

☐ **Timing (ns)**

☐ **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.332	0.10

☐ **Latency (clock cycles)**

☐ **Summary**

Latency		Interval		Type
min	max	min	max	
25	25	25	25	none

Рис. 3.57. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	45
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	104
Register	-	-	112	-
Total	0	4	112	149
Available	40	40	16000	8000
Utilization (%)	0	10	~0	1

Рис. 3.58. Utilization estimates

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo_b	-	25	-	26	-
Loop 1	no	24	6	-	4

Рис. 3.59. Performance profile

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo_b	return value
ap_rst	in	1	ap_ctrl_hs	foo_b	return value
ap_start	in	1	ap_ctrl_hs	foo_b	return value
ap_done	out	1	ap_ctrl_hs	foo_b	return value
ap_idle	out	1	ap_ctrl_hs	foo_b	return value
ap_ready	out	1	ap_ctrl_hs	foo_b	return value
d_in_0_address0	out	2	ap_memory	d_in_0	array
d_in_0_ce0	out	1	ap_memory	d_in_0	array
d_in_0_q0	in	16	ap_memory	d_in_0	array
d_in_1_address0	out	2	ap_memory	d_in_1	array
d_in_1_ce0	out	1	ap_memory	d_in_1	array
d_in_1_q0	in	16	ap_memory	d_in_1	array
d_in_2_address0	out	2	ap_memory	d_in_2	array
d_in_2_ce0	out	1	ap_memory	d_in_2	array
d_in_2_q0	in	16	ap_memory	d_in_2	array
d_in_3_address0	out	2	ap_memory	d_in_3	array
d_in_3_ce0	out	1	ap_memory	d_in_3	array
d_in_3_q0	in	16	ap_memory	d_in_3	array
d_out_address0	out	3	ap_memory	d_out	array
d_out_ce0	out	1	ap_memory	d_out	array
d_out_we0	out	1	ap_memory	d_out	array
d_out_d0	out	16	ap_memory	d_out	array

Рис. 3.60. Interface estimates



Рис. 3.61. Scheduler viewer

	Resource\Control Step	C0	C1	C2	C3	C4	C5	C6
1	I/O Ports							
2	d_in_0(p0)		read					
3	d_in_1(p0)		read					
4	d_in_2(p0)			read				
5	d_in_3(p0)			read				
6	d_out(p0)				write	write	write	write
7	Memory Ports							
8	d_in_1(p0)		read					
9	d_in_0(p0)		read					
10	d_in_3(p0)			read				
11	d_in_2(p0)			read				
12	d_out(p0)				write	write	write	write
13	Expressions							
14	i_1_fu_153		+					
15	i_phi_fu_139		phi_mux					
16	exitcond_fu_147		icmp					
17	tmp_4_fu_191			*				
18	tmp_9_fu_197				*			
19	tmp_7_fu_208				*			
20	tmp_3_fu_203				*			
21	tmp_5_fu_169						+	
22	tmp_8_fu_180							+

Рис. 3.62. Resource viewer

3.11.3. Анализ решения

Можно увидеть, что проигнорировано разбиение выходного массива. Скорее всего это связано с особенностью устройства, в общем случае это должно дать наивысшую скорость записи выходных данных.

3.12. Вывод

Ниже приведено сравнение решений.

	RAM_1P	RAM_2P
Без cyclic	2b/25/26	5b/25/26
cyclic; factor =2	3b/25/26	6b/25/26
cyclic; factor =4	4b/29/30	7b/29/30

Рис. 3.63. Сравнение решений

По сравнению видно, что при разбиении выходной памяти не всегда получается ожидаемый прирост производительности, а часто даже ухудшение. Согласно временным диаграммам, запись в выходную память почти никогда не производится параллельно для нескольких ячеек. Скорее всего, это связано с особенностью устройства, где на выход подаются различные комбинации одних и тех же входов.