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

Отчет по лабораторной работе №7_2
Курс: «Проектирование реконфигурируемых гибридных вычислительных систем»
Тема: Pipeline

Выполнил студент гр. 3540901/81501 Селиверстов С.А.
(подпись)

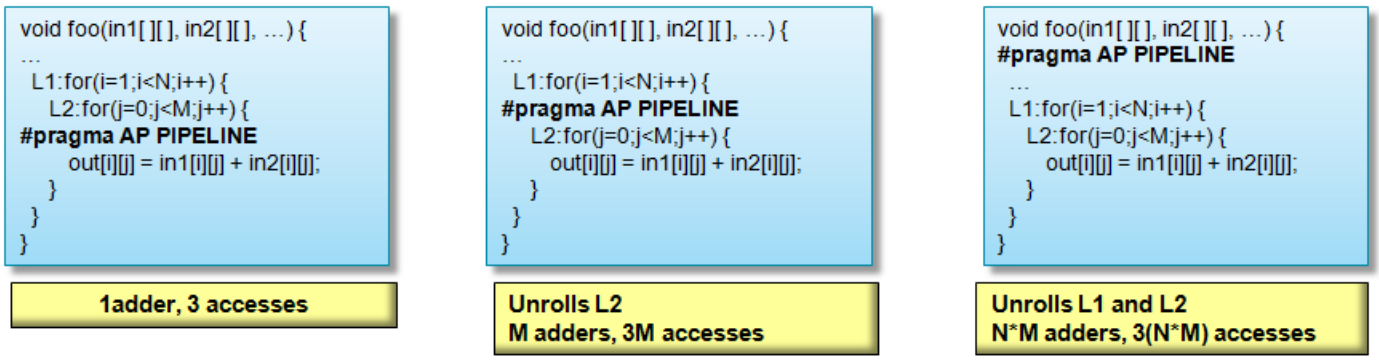
Руководитель Антонов А.П.
(подпись)
“___” _____2019 г.

Санкт – Петербург
2019

ОГЛАВЛЕНИЕ

1. Задание

- Создать проект lab7_2
- Микросхема: xa7a12tcsg325-1q
- Создать функцию на основе приведенных ниже слайдов.



- Создать тест lab7_2_test.c для проверки функций выше.
  - осуществить моделирование (с выводом результатов в консоль)

- Сделать свой solution (для каждого варианта задания директивы и для варианта без директивы)
  - задать: clock period 10; clock_uncertainty 0.1
  - осуществить синтез
    - привести в отчете:
      - performance estimates=>summary
      - utilization estimates=>summary
      - 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
  - Осуществить C|RTL моделирование (для каждого варианта задания директивы)
    - Привести результаты из консоли
    - Открыть временную диаграмму (все сигналы)
      - Отобразить два цикла обработки на одном экране
        - На скриншоте показать Latency
        - На скриншоте показать Initiation Interval
- Выводы
  - Привести обобщенную таблицу зависимости utilization и performance от каждого варианта задания директивы и для варианта без директивы.
  - Объяснить отличие процедур обращения к элементам массива для каждого случая

1.2. Исходный код

Представим код программ lab7_z2.c и lab7_z2_test.c на листинге 1 и 2.

```
  void foo (int in1[10][10], int in2[10][10], int
out[10][10]) {
    int i, j;
    L1:for (i = 0; i < 10; i++) {
        L2:for (j = 0; j < 10; j++) {
            out[i][j] = in1[i][j] + in2[i][j];
        }
    }
  }
```

Листинг 1.

```
  #include <stdio.h>
  int main(){
    int in1[10][10];
    int in2[10][10];
    int out[10][10];
    int exp_out[10][10];
    int i, j;
    for (i = 0; i < 10; i++) {
        for (j = 0; j < 10; j++) {
            in1[i][j] = i + j;
            in2[i][j] = i * 10;
            out[i][j] = 0;
            exp_out[i][j] = in1[i][j] + in2[i][j];
        }
    }
    foo(in1,in2,out);
    for (i = 0; i < 10; i++) {
        for (j = 0; j < 10; j++) {
            printf("%d + %d == Out %d == Exp %d\n",
in1[i][j],in2[i][j],out[i][j], exp_out[i][j]);
            if (out[i][j] != exp_out[i][j]) {
                printf("-------ERROR-------\n");
                return -1;
            }
        }
    }
    printf("-------Test Pass-------\n");
    return 0;
  }
```

Листинг 2.

2.Решение №1

2.1 Моделирование

На рисунке 2.1. приведем результаты логи успешного моделировании.

```
14 + 90 == Out 104 == Exp 104
15 + 90 == Out 105 == Exp 105
16 + 90 == Out 106 == Exp 106
17 + 90 == Out 107 == Exp 107
18 + 90 == Out 108 == Exp 108
-------Test Pass-------
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Рисунок 2.1. Логи моделирования

2.2 Синтез

Результаты синтеза с оценкой производительности и используемых ресурсов представлены на рисунках 2.1 и 2.2. соответственно.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.216	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
221	221	221	221	none

Рисунок 2.2. Performance estimates – summary

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	113	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	45	-
Register	-	-	35	-	-
Total	0	0	35	158	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	1	0

Рисунок 2.3. Utilization estimates – summary

Диаграмма операционного расписания с указанием Latency и диаграмма оперционного просмоторщика ресурсов приведены на рисунках 2.4. и 2.5.

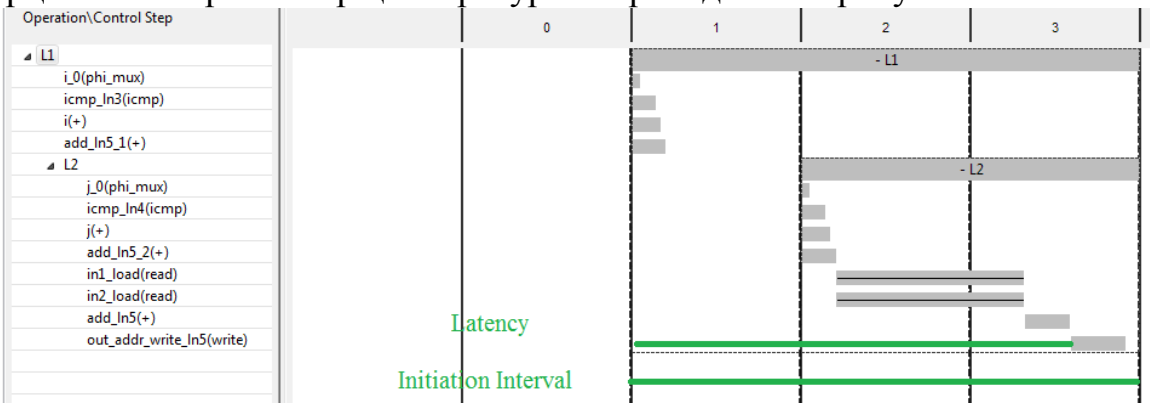


Рисунок 2.4. Schedule viewer

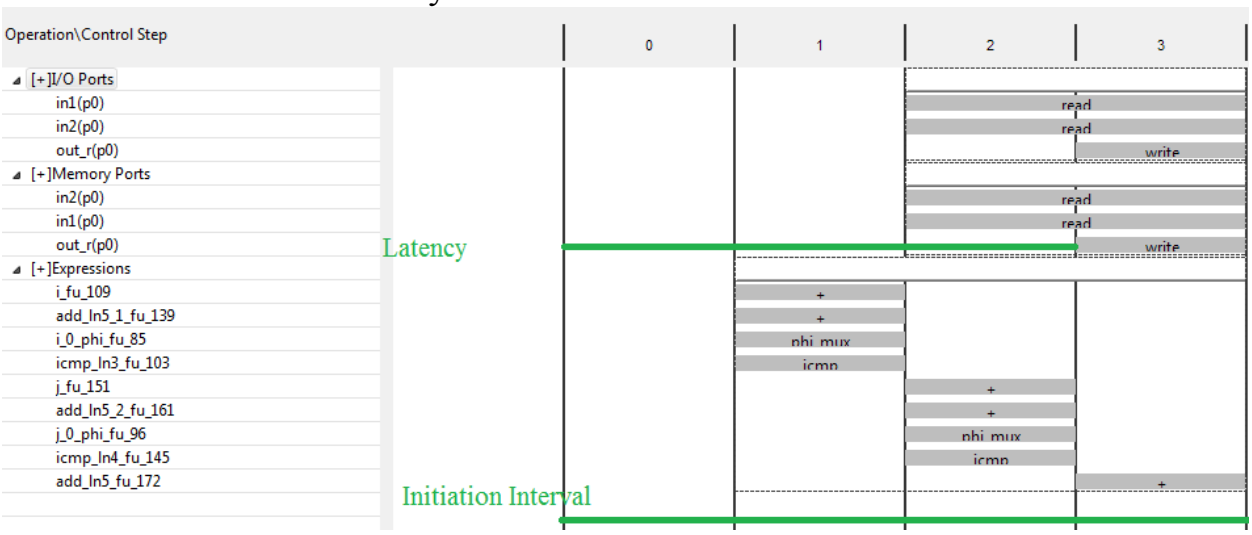


Рисунок 2.5. Resourse viewer

2.3 C|RTL моделирование

Результаты C|RTL приведены на рисунке 2.6.

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	221	221	221	NA	NA	NA

Export the report(.html) using the Export Wizard

Рисунок 2.6. Отчет о моделировании

Временная диаграмма приведена на рисунке 2.7.

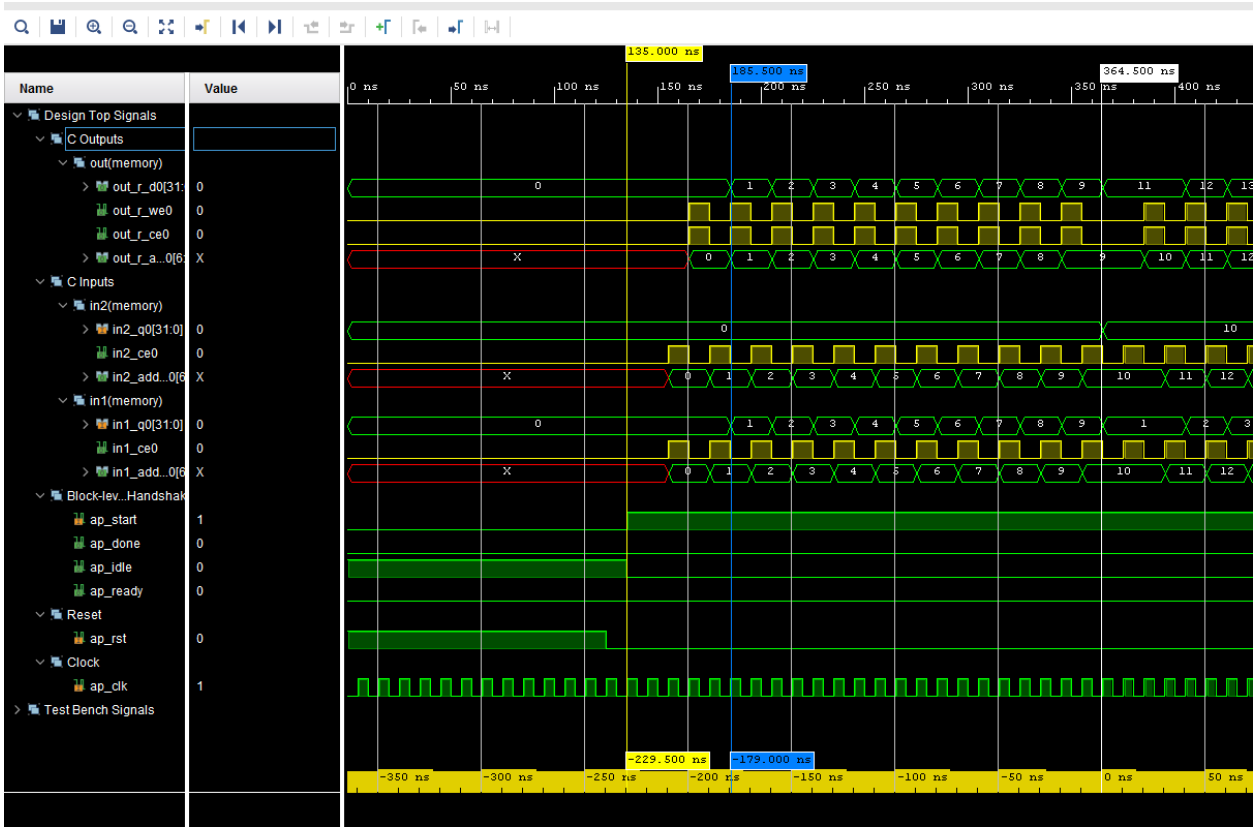


Рисунок 2.7. Временная диаграмма

3. Решение №2

3.1 Параметры второго решения

Пропишем директиву PIPELINE внутреннего цикла, рисунок 3.1.

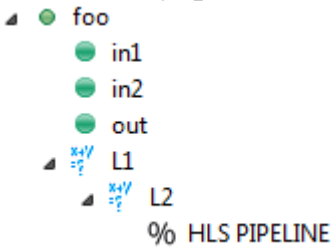


Рисунок 3.1. Директива PIPELINE примененная внутри внутреннего цикла

3.2 Синтез

Результаты синтеза с оценкой производительности и используемых ресурсов представлены на рисунках3.2 и 3.3 соответственно.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.702	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
102	102	102	102	none

Рисунок 3.2. Performance estimates – summary

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	128	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	72	-
Register	-	-	33	-	-
Total	0	0	33	200	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	2	0

Рисунок 3.3. Utilization estimates – summary

Диаграмма операционного расписания с указанием Latency и диаграмма оперционного просмоторщика ресурсов приведены на рисунках 3.4. и 3.5.

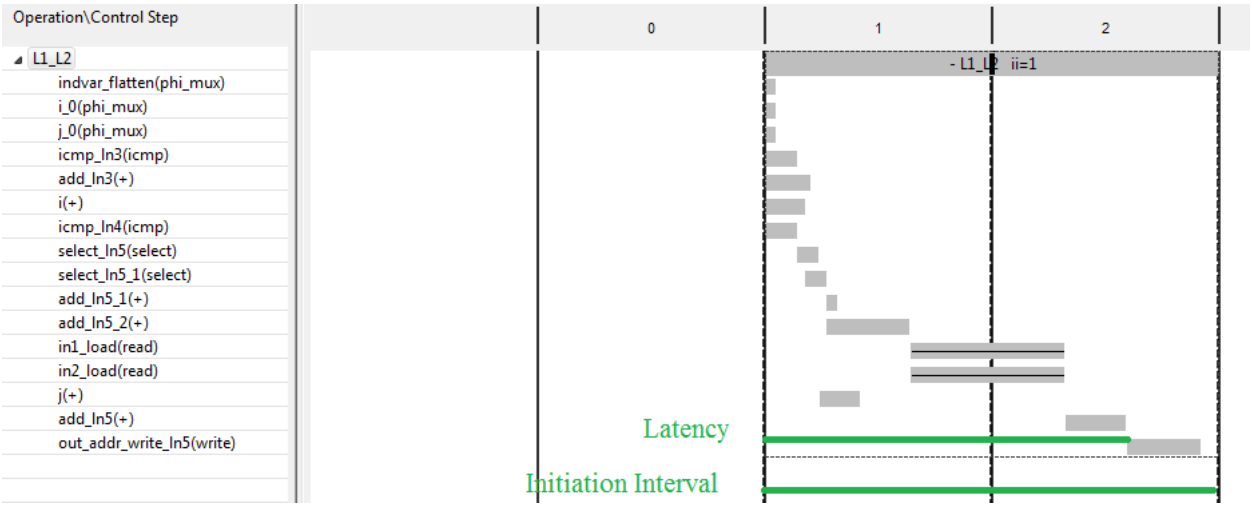


Рисунок 3.4. Schedule viewer

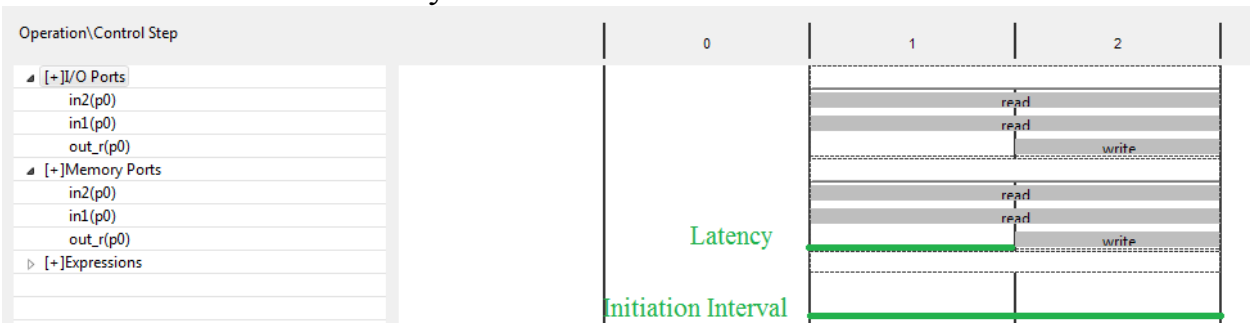


Рисунок 3.5. Resource viewer

3.3 C|RTL моделирование

Результаты C|RTL приведены на рисунке 3.6.

Cosimulation Report for 'foo'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	102	102	102	NA	NA	NA

Рисунок 3.6. Отчет о моделировании

Временная диаграмма приведена на рисунке 3.7.

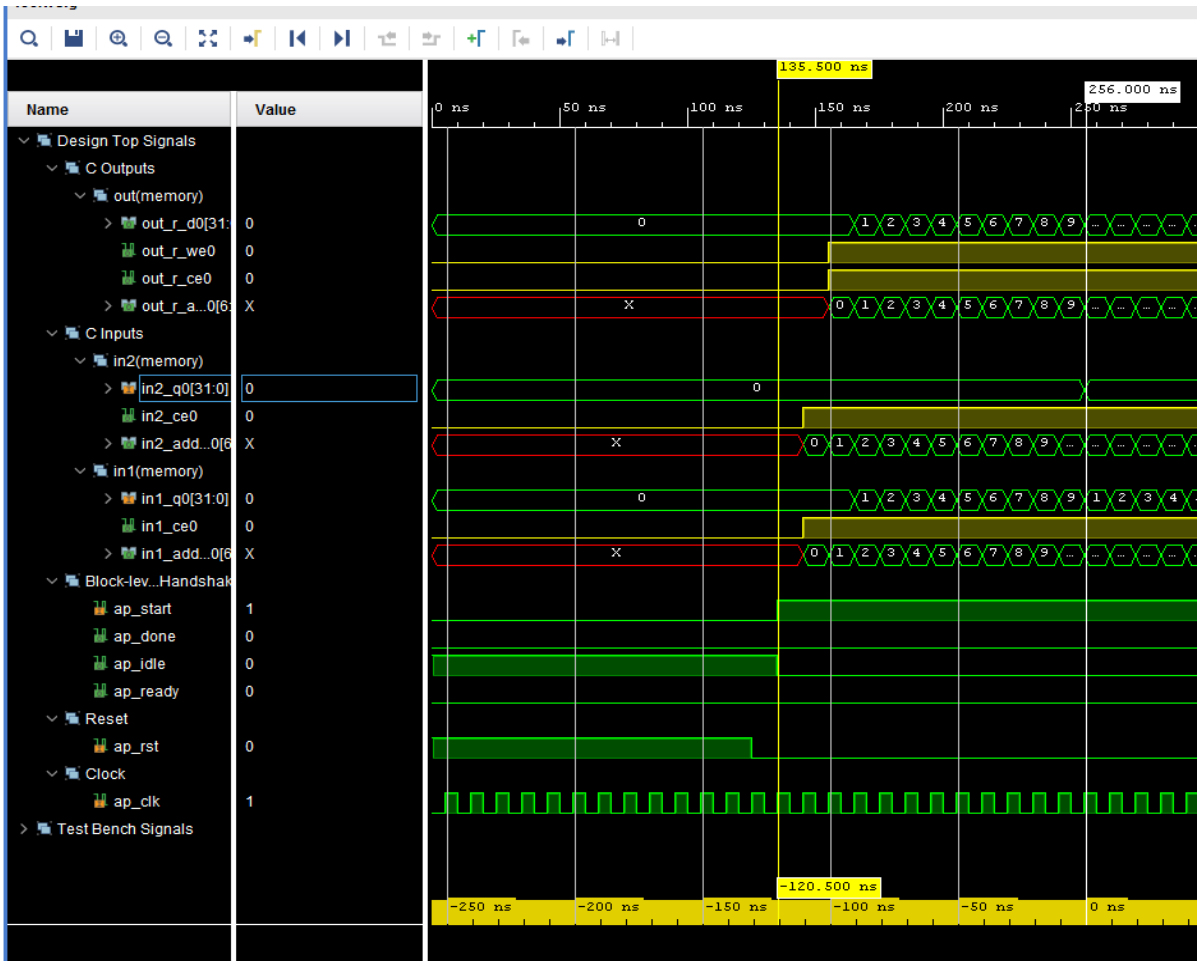


Рисунок 3.7. Временная диаграмма

4. Решение №3

4.1 Параметры третьего решения

Пропишем директиву PIPELINE внешнего цикла, рисунок 4.1.

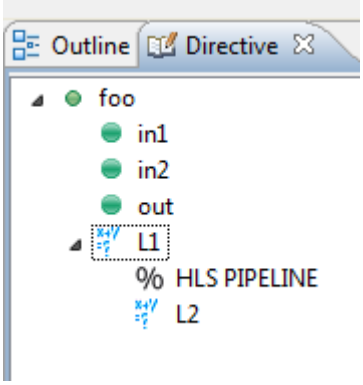


Рисунок 4.1. Директива PIPELINE примененная внутри внешнего цикла

4.2 Синтез

Результаты синтеза с оценкой производительности и используемых ресурсов представлены на рисунках 4.2. и 4.3 соответственно.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.216	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
52	52	52	52	none

Detail

Instance

Loop

Рисунок 4.2. Performance estimates – summary

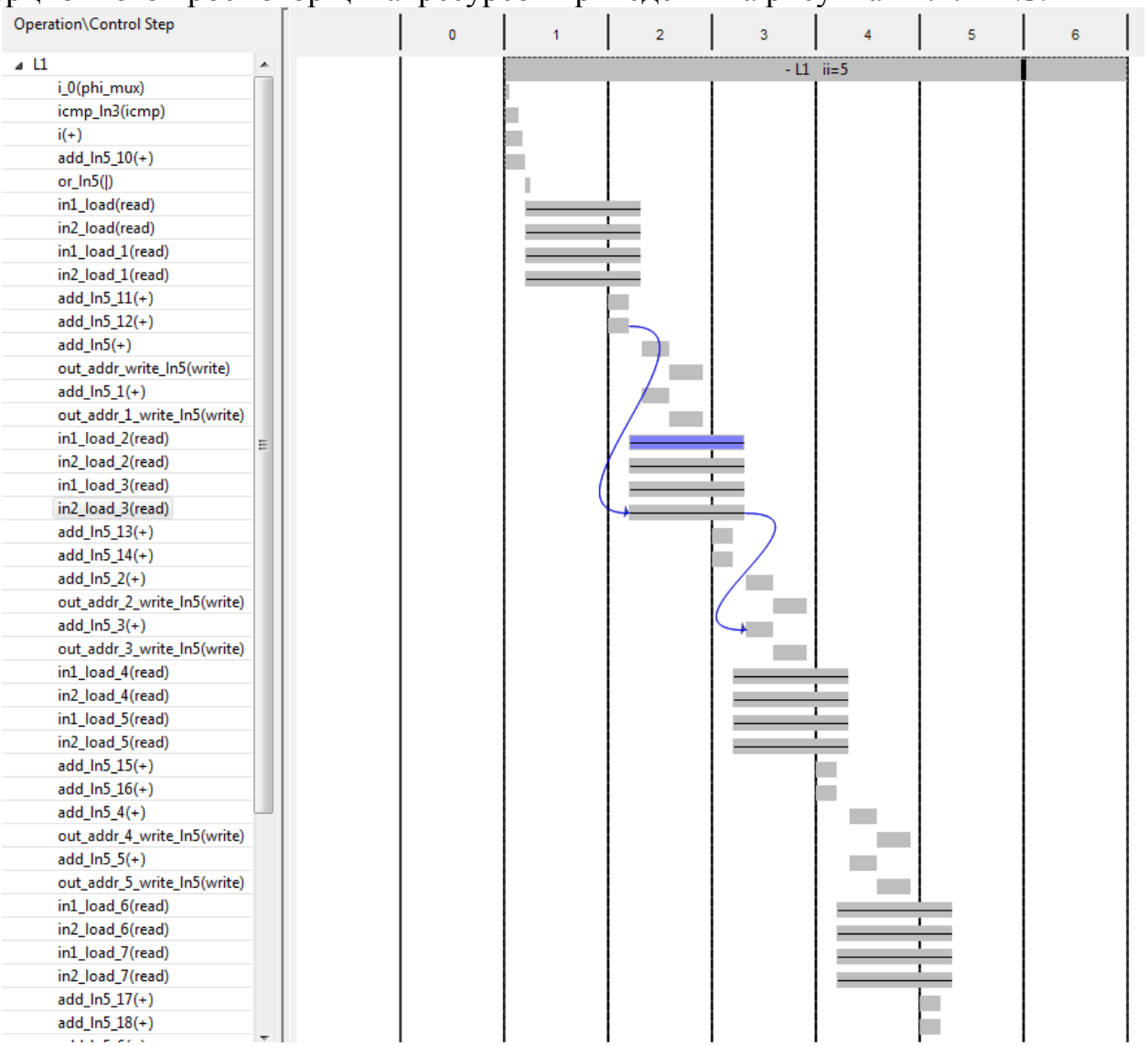
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	245	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	266	-
Register	-	-	543	-	-
Total	0	0	543	511	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	3	6	0

Рисунок 4.3. Utilization estimates – summary

Диаграмма операционного расписания с указанием Latency и диаграмма оперционного просмоторщика ресурсов приведены на рисунках 4.4. и 4.5.



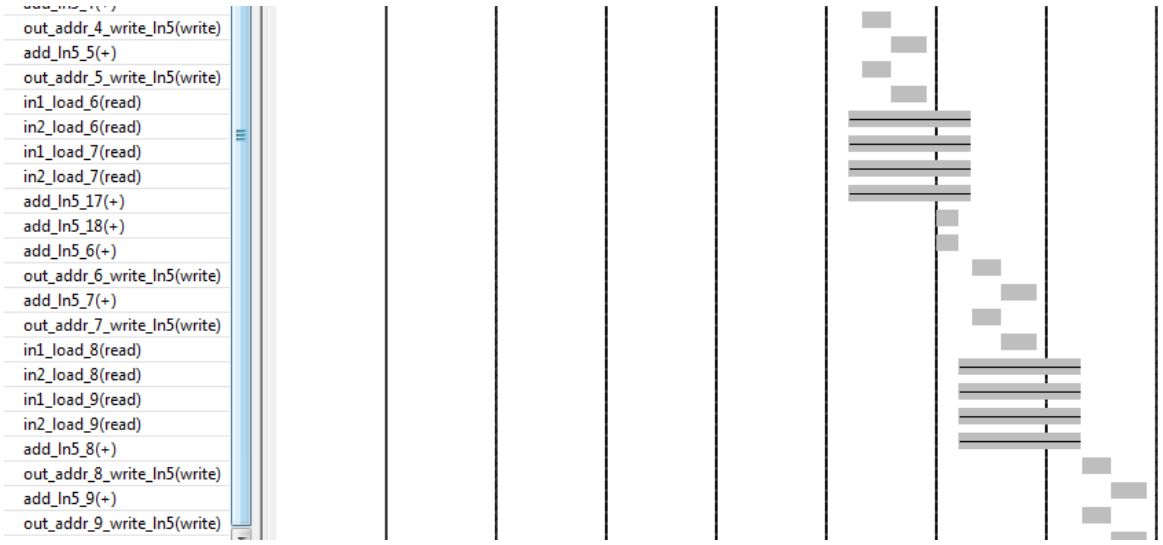


Рисунок 4.4. Schedule viewer

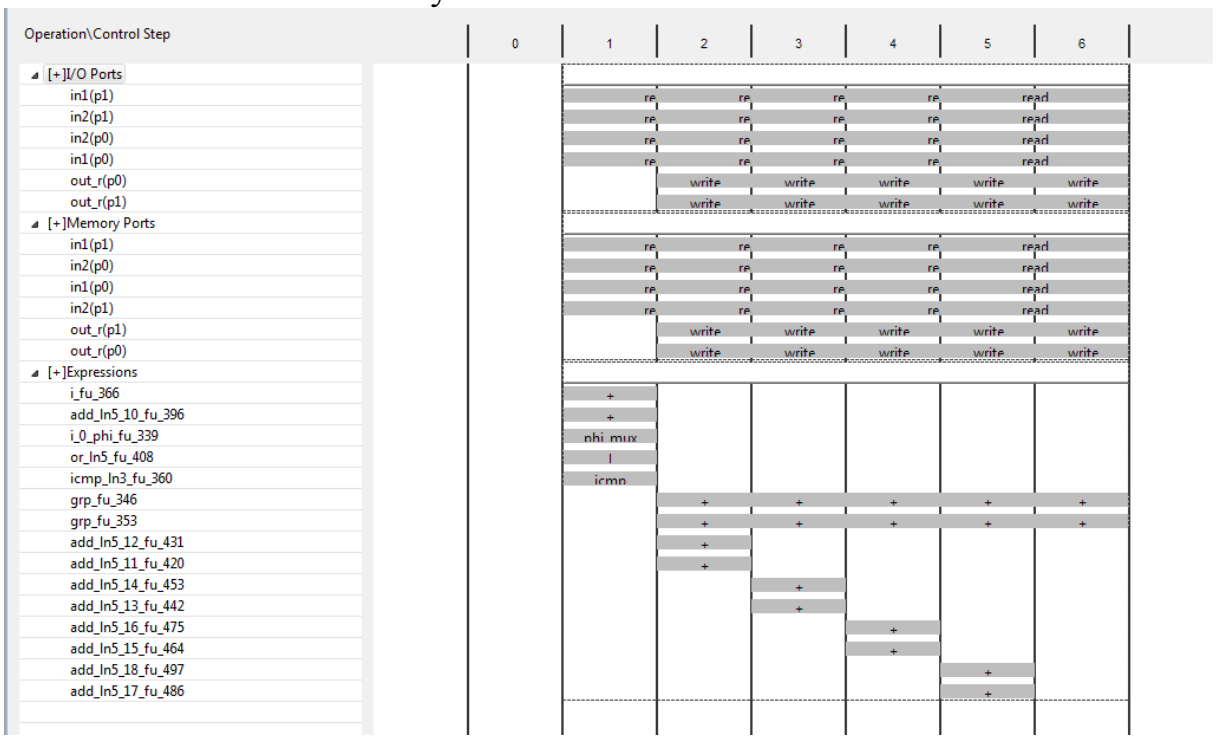


Рисунок 4.5. Resource viewer

4.3 C|RTL моделирование

Результаты C|RTL приведены на рисунке 4.6.

Cosimulation Report for 'foo'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	52	52	52	NA	NA	NA

Export the report(.html) using the Export Wizard

Рисунок 4.6. Отчет о моделировании

Временная диаграмма приведена на рисунке 4.7.

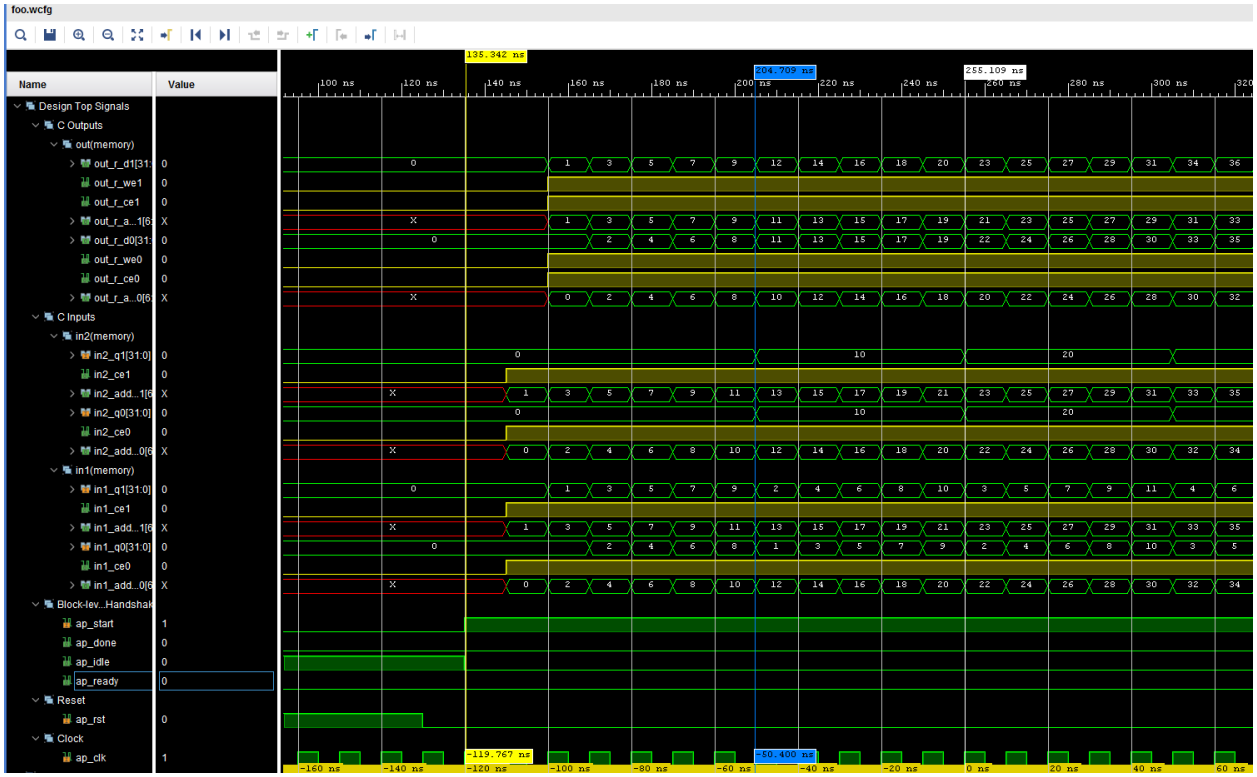


Рисунок 4.7. Временная диаграмма

5. Решение №4

5.1 Параметры четвертого решения

Пропишем директиву PIPELINE перед внешним циклом, рисунок 3.1.

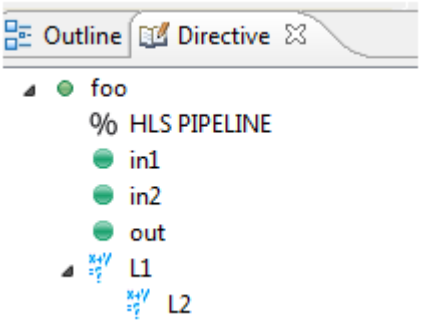


Рисунок 5.1. Директива PIPELINE примененная перед внешним циклом

5.2 Синтез

Результаты синтеза с оценкой производительности и используемых ресурсов представлены на рисунке 5.2. и 5.3. соответственно.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.216	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
50	50	50	50	function

Detail

Instance

Loop

Рисунок 5.2. Performance estimates – summary

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	82	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	1565	-
Register	-	-	52	-	-
Total	0	0	52	1647	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	20	0

Рисунок 5.3. Utilization estimates – summary

Диаграмма операционного расписания с указанием Latency и диаграмма оперционного просмоторщика ресурсов приведены на рисунках 5.4 и 5.5 соотвтственно.

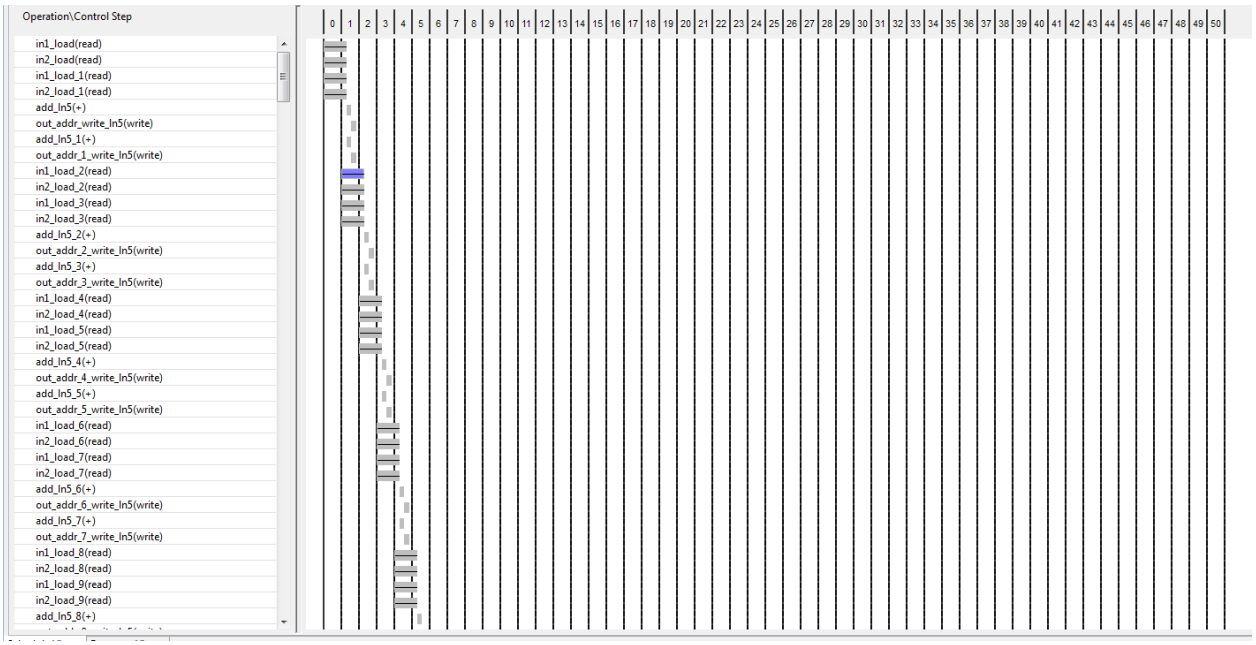


Рисунок 5.4. Schedule viewer

Рисунок 5.5. Resource viewer

5.3 C|RTL моделирование

Результаты C|RTL приведены на рисунке 5.6.

Cosimulation Report for 'foo'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	50	50	50	NA	NA	NA

Export the report(.html) using the Export Wizard

Рисунок 5.6. Отчет о моделировании

Временная диаграмма приведена на рисунке 5.7.

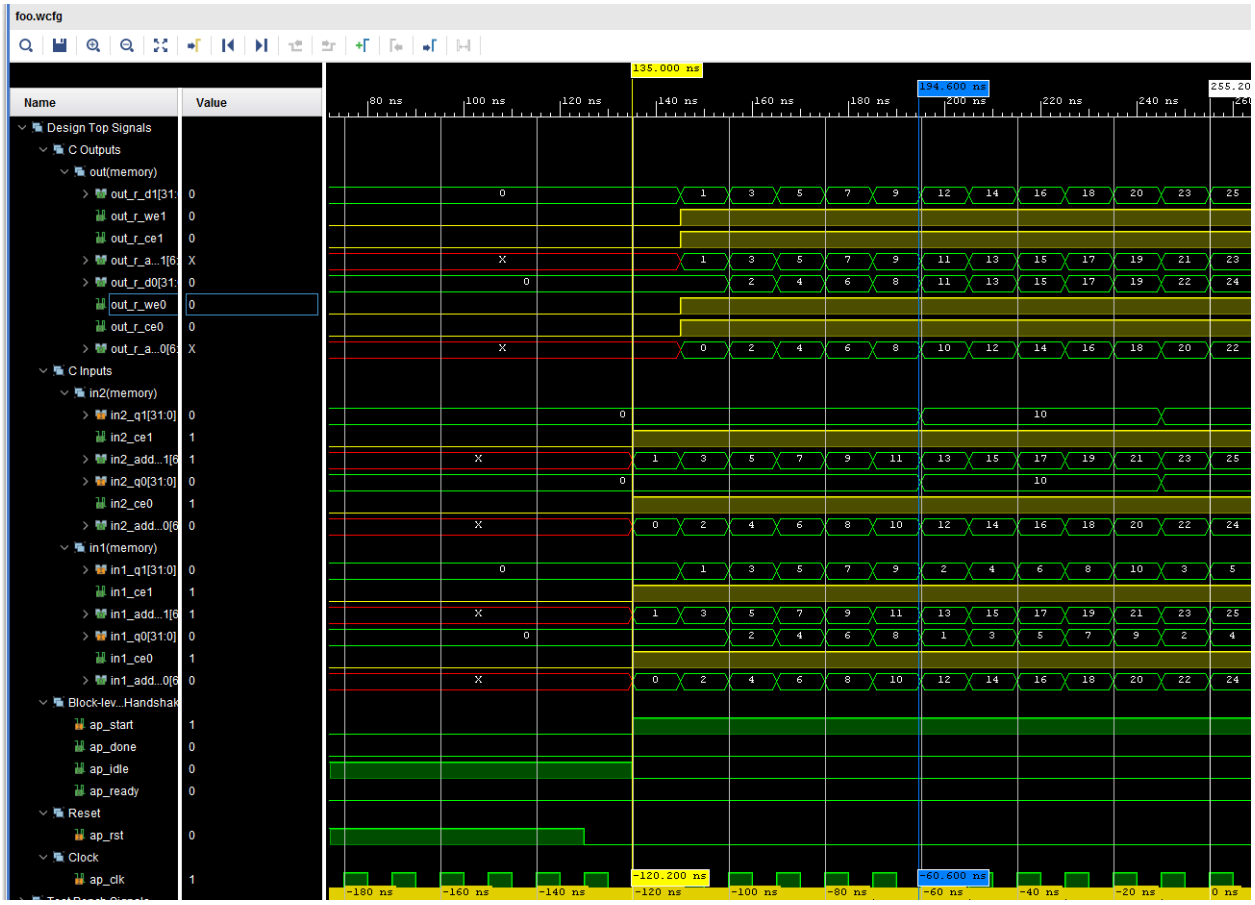


Рисунок 5.7. Временная диаграмма

6. ВЫВОД

При различных методах применения директивы PIPELINE изменяется и результат. Чем выше в иерархии находится директива, тем выше уровень параллелизма, выше пропускная способность и выше количество требуемых ресурсов для имплементации проекта.

Performance Estimates

Timing (ns)

Clock		solution1	solution2	solution3	solution4
ap_clk	Target	10.00	10.00	10.00	10.00
	Estimated	9.216	9.702	9.216	9.216

Latency (clock cycles)

		solution1	solution2	solution3	solution4
Latency	min	221	102	52	50
	max	221	102	52	50
Interval	min	221	102	52	50
	max	221	102	52	50

Utilization Estimates

	solution1	solution2	solution3	solution4
BRAM_18K	0	0	0	0
DSP48E	0	0	0	0
FF	35	33	543	52
LUT	158	200	511	1647
URAM	0	0	0	0

Рисунок 6.1. Сравнительный отчет решений

В первом решении не используется директивы, поэтому все операции выполняются последовательно.

Во втором решении конвейеризируется только операция сложения, что сокращает количество требуемых тактов для выполнения до 102. Единовременно получается доступ к 3 переменным и реализуется 1 операция сложения (см. рисунок 6.1).

В третьем решении конвейеризируется внутренний цикл. Единовременно получается доступ к 3*M переменным и реализуется M операций сложения.

В четвертом решении конвейеризируется внешний цикл. Единовременно получается доступ к 3*(N*M) переменным и реализуется N*M операций сложения.