

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

Лабораторная №14

Предмет: Проектирование реконфигурируемых гибридных вычислительных  
систем

Тема: Указатели

Задание 3

**Студенты:**

Соболь В.

Темнова А.С.

Группа: 13541/3

**Преподаватель:**

Антонов А.П.

Санкт-Петербург  
2019

# Содержание

<b>1. Задание</b>	<b>3</b>
<b>2. Исходный код</b>	<b>4</b>
<b>3. Скрипт</b>	<b>6</b>
<b>4. Решение 1a</b>	<b>6</b>
4.1. Моделирование . . . . .	6
4.2. Синтез . . . . .	7
4.3. C/RTL моделирование . . . . .	9
<b>5. Решение 2a</b>	<b>9</b>
5.1. Моделирование . . . . .	9
5.2. Синтез . . . . .	9
5.3. C/RTL моделирование . . . . .	11
<b>6. Вывод</b>	<b>12</b>

# 1. Задание

1. Создать проект lab14\_3
2. Микросхема: xa7a12tcsg325-1q
3. В папке source текст функции pointer\_stream\_better  
*Познакомьтесь с ним (посмотрите в лекции часть Multi-Access Pointers)*
4. Познакомьтесь с тестом.
5. Исследование:
6. Solution\_1a

- Создать версию pointer\_stream\_ , в которой будет убран volatile

```
#include "pointer_stream_better.h"

void pointer_stream_better ( volatile dout_t *d_o, volatile din_t *d_i) {
    din_t acc = 0;

    acc += *d_i;
    acc += *d_i;
    *d_o = acc;
    acc += *d_i;
    acc += *d_i;
    *d_o = acc;
}
```

7. Осуществить моделирование (при необходимости изменить тест) – обратить внимание на раздел тестирования в лекции
8. задать: clock period 10; clock\_uncertainty 0.1
9. установить реализацию ПО УМОЛЧАНИЮ
10. осуществить синтез для:
  - 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
11. Выполнить cosimulation и привести временную диаграмму
12. Solution\_2a

- Использовать исходную функцию `pointer_stream_better`
- Осуществить моделирование – обратить внимание на раздел тестирования в лекции
- задать: `clock period 10; clock_uncertainty 0.1`
- установить реализацию ПО УМОЛЧАНИЮ
- 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
- Выполнить cosimulation и привести временную диаграмму

13. Сравнить два решения (`solution_1a` и `solution_2a`) и сделать выводы

## 2. Исходный код

Ниже приведен исходный код устройства и теста.

```

1 #include "pointer_stream_better.h"
2
3
4 #ifdef USE_VOLATILE
5
6 void pointer_stream_better ( volatile dout_t *d_o, volatile din_t *d_i)
7
8 #else
9
10 void pointer_stream_better ( dout_t *d_o, din_t *d_i)
11
12 #endif
13 {
14     din_t acc = 0;
15
16     acc += *d_i;
17     acc += *d_i;
18     *d_o = acc;
19     acc += *d_i;
20     acc += *d_i;
21     *d_o = acc;
22 }
```

Рис. 2.1. Исходный код устройства

```

1 #ifndef _POINTER_STREAM_BETTER_H_
2 #define _POINTER_STREAM_BETTER_H_
3
4 #include <stdio.h>
5
6 typedef int din_t;
7 typedef int dout_t;
8
9
10 #ifdef USE_VOLATILE
11
12 void pointer_stream_better ( volatile dout_t *d_o, volatile din_t *d_i);
13
14 #else
15
16 void pointer_stream_better ( dout_t *d_o, din_t *d_i);
17
18 #endif
19
20 #endif

```

Рис. 2.2. Заголовочный файл

```

1 #include "pointer_stream_better.h"
2
3 int main () {
4     din_t d_i;
5     dout_t d_o;
6     int retval=0;
7     FILE *fp;
8
9     // Open a file for the output results
10    fp=fopen("result.dat","w");
11    fprintf(fp, "Din_Dout\n");
12
13    // Call the function to operate on the data
14    for (d_i=0;d_i<4;d_i++) {
15        pointer_stream_better(&d_o,&d_i);
16        fprintf(fp, "%d__%d\n", d_i, d_o);
17    }
18    fclose(fp);
19
20    // Compare the results file with the golden results
21    retval = system("diff --brief -w result.dat result.golden.dat");
22    if (retval != 0) {
23        printf("Test_failed_!!!\n");
24        retval=1;
25    } else {
26        printf("Test_passed_!\n");
27    }
28
29    // Return 0 if the test passed
30    return retval;
31 }

```

Рис. 2.3. Исходный код теста

### 3. Скрипт

Ниже приводится скрипт, для автоматизации выполнения лабораторной работы.

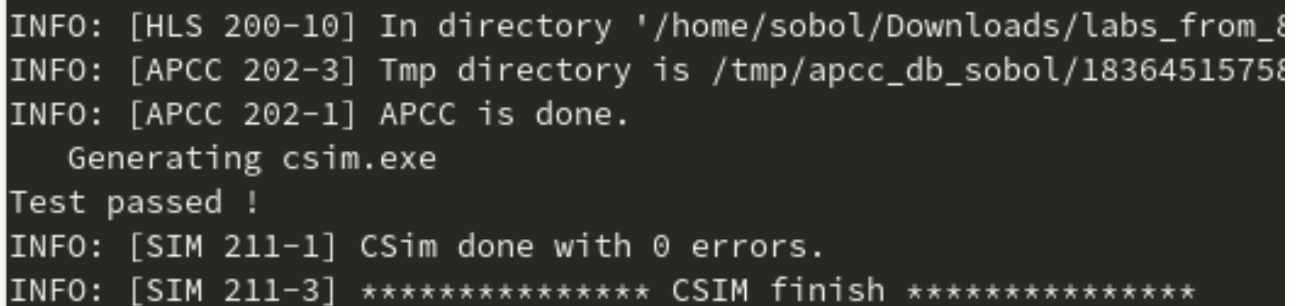
```
1 open_project -reset lab14_3
2
3 add_files pointer_stream_better.c
4 add_files -tb pointer_stream_better_test.c
5 add_files -tb result.golden.dat
6
7 set_top pointer_stream_better
8
9 open_solution -reset solution_1a
10
11 set_part {xa7a12tcsg325-1q}
12 create_clock -period 10ns
13 set_clock_uncertainty 0.1
14
15 csim_design
16 csynth_design
17 cosim_design -trace_level all
18
19 add_files pointer_stream_better.c -cflags "-DUSE_VOLATILE"
20 open_solution -reset solution_2a
21
22 set_part {xa7a12tcsg325-1q}
23 create_clock -period 10ns
24 set_clock_uncertainty 0.1
25
26 csim_design
27 csynth_design
28 cosim_design -trace_level all
```

Рис. 3.1. Скрипт

### 4. Решение 1a

#### 4.1. Моделирование

Ниже приведены результаты моделирования.



```
INFO: [HLS 200-10] In directory '/home/sobol/Downloads/labs_from_8
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_sobol/18364515758
INFO: [APCC 202-1] APCC is done.
    Generating csim.exe
Test passed !
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
```

Рис. 4.1. Результаты моделирования

По результатам моделирования видно, что устройство работает корректно.

## 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	0.000	0.10

#### Latency (clock cycles)

##### Summary

Latency		Interval		
min	max	min	max	Type
0	0	0	0	none

Рис. 4.2. Performance estimates

### Utilization Estimates

#### Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	-	-
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	-	-
Total	0	0	0	0
Available	40	40	16000	8000
Utilization (%)	0	0	0	0

Рис. 4.3. Utilization estimates

Performance Profile

Resource Profile

	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip co
<input checked="" type="radio"/> pointer_stream_better	-	0	-	1	-

Рис. 4.4. Performance profile

Operation\Control Step		0	
acc(read)			
acc_1(shl)			
node_8(write)			

Рис. 4.5. Scheduler viewer

	Resource\Control S...	C0
1	<input checked="" type="checkbox"/> I/O Ports	
2	d_o	write
3	d_i	read
4	<input checked="" type="checkbox"/> Expressions	
5	acc_1_fu_29	shl

Рис. 4.6. Resource viewer



### 4.3. C/RTL моделирование

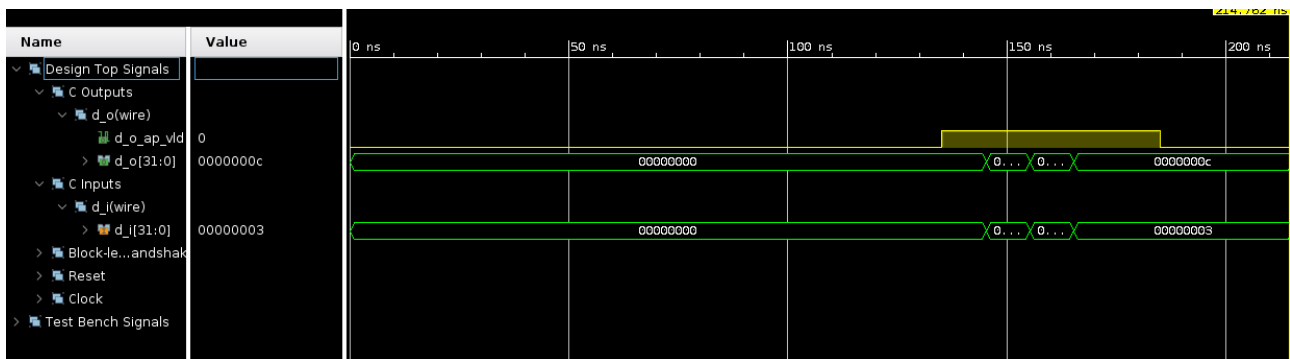


Рис. 4.7. Временная диаграмма

Как видно по результатам выше, полученное решение выполняется за один такт.

## 5. Решение 2а

### 5.1. Моделирование

Ниже приведены результаты моделирования.

```
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_sobol/1842291
INFO: [APCC 202-1] APCC is done.
    Generating csim.exe
Test passed !
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
```

Рис. 5.1. Результаты моделирования

По результатам моделирования видно, что устройство работает корректно.

### 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	4.505	0.10

### Latency (clock cycles)

#### Summary

Latency		Interval		
min	max	min	max	Type
3	3	3	3	none

Рис. 5.2. Performance estimates

## Utilization Estimates

### Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	103
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	42
Register	-	-	68	-
Total	0	0	68	145
Available	40	40	16000	8000
Utilization (%)	0	0	~0	1

Рис. 5.3. Utilization estimates

Performance Profile		Resource Profile				
		Pipelined	Latency	Iteration Latency	Initiation Interval	Trip co
pointer_stream_better		-	3	-	4	-

Рис. 5.4. Performance profile

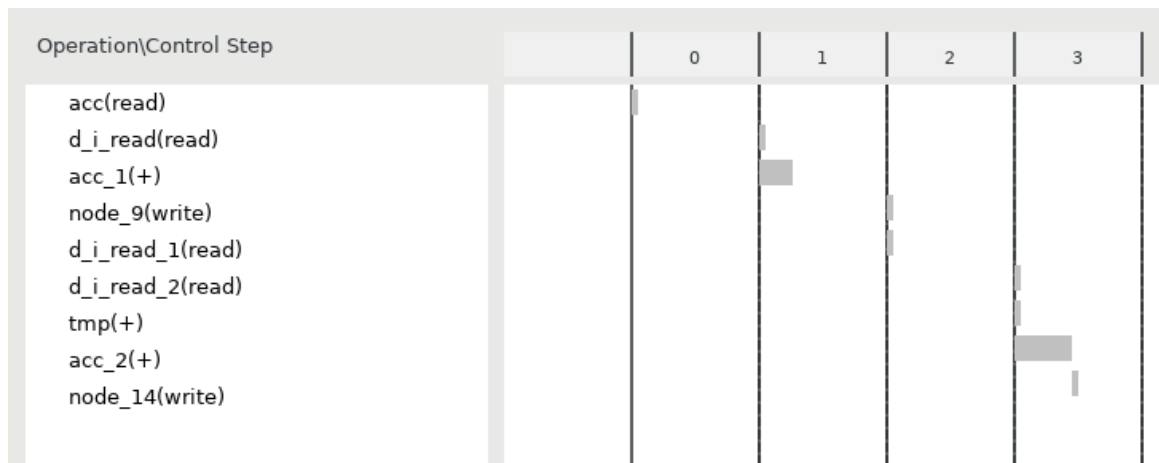


Рис. 5.5. Scheduler viewer

	Resource\Control S...	C0	C1	C2	C3
1	<input type="checkbox"/> I/O Ports				
2	d_i	read	read	read	read
3	d_o			write	write
4	<input type="checkbox"/> Expressions				
5	acc_1_fu_31		+		
6	acc_2_fu_42				+
7	tmp_fu_37				+

Рис. 5.6. Resource viewer

### 5.3. C/RTL моделирование

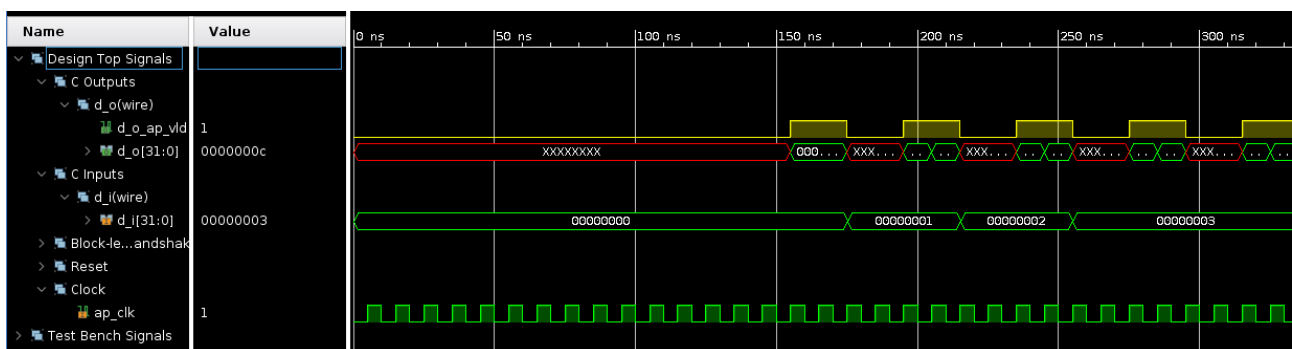


Рис. 5.7. Временная диаграмма

Как видим, при использовании ключевого слова `volatile` производительность проекта снизилась. Это связано с тем что отключена оптимизация операций и теперь у на выходе значения появляются 2 раза а не один раз в самом конце, как в предыдущем решении.

## 6. Вывод

Ключевое слово `volatile` используется когда требуется полный контроль над выполнением операций для избежания моментов когда компилятор оптимизирует исходный код тем самым убирает ненужные строки кода которые по-другому работают на аппаратуре нежели на процессоре.