

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 1

**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>3</b> |
| <b>3. Скрипт</b>             | <b>5</b> |
| <b>4. Решение 1a</b>         | <b>6</b> |
| 4.1. Моделирование . . . . . | 6        |
| 4.2. Синтез . . . . .        | 6        |
| <b>5. Вывод</b>              | <b>9</b> |

## 1. Задание

1. Создать проект lab14\_1
2. Микросхема: xa7a12tcsg325-1q
3. В папке source имеется описание функции с преобразованием malloc для синтеза. Ознакомьтесь с текстом.
4. Ознакомиться с тестом
5. Осуществить моделирование (с выводом результатов в консоль)
6. Исследование:
7. Solution\_1a
  - задать: clock period 10; clock\_uncertainty 0.1
  - установить реализацию ПО УМОЛЧАНИЮ
  - 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
  - Выполнить cosimulation и привести временную диаграмму
8. Сделать выводы и пояснить, что было сделано в коде.

## 2. Исходный код

Ниже приведен исходный код устройства и теста.

```

1 #include "malloc_removed.h"
2 #include <stdlib.h>
3
4 dout_t malloc_removed(din_t din[N], dsel_t width) {
5
6 #ifdef NO_SYNTH
7     long long *out_accum = malloc (sizeof(long long));
8     int* array_local = malloc (64 * sizeof(int));
9 #else
10    long long _out_accum;
11    long long *out_accum = &_out_accum;
12    int _array_local[64];
13    int* array_local = &_array_local[0];
14 #endif
15    int i, j;
16
17    LOOP_SHIFT: for (i=0; i<N-1; i++) {
18        if (i<width)
19            *(array_local+i)=din[i];
20        else
21            *(array_local+i)=din[i]>>2;
22    }
23
24    *out_accum=0;
25    LOOP_ACCUM: for (j=0; j<N-1; j++) {
26        *out_accum += *(array_local+j);
27    }
28
29    return *out_accum;
30 }

```

Рис. 2.1. Исходный код устройства

```

1 #ifndef _MALLOC_REMOVED_H_
2 #define _MALLOC_REMOVED_H_
3
4 #include <stdio.h>
5 #define N 32
6
7 typedef int din_t;
8 typedef long long dout_t;
9 typedef int dsel_t;
10
11 dout_t malloc_removed(din_t din[N], dsel_t width);
12
13 #endif

```

Рис. 2.2. Заголовочный файл

```

1 #include "malloc_removed.h"
2
3 int main () {
4     din_t A[N];
5     dout_t accum;
6
7     int i, retval=0;
8     FILE *fp;
9
10    for(i=0; i<N;++i) {
11        A[i]=i+200;
12    }
13    // Save the results to a file
14    fp=fopen("result.dat","w");
15
16    // Call the function
17    for(i=0; i<N;++i) {
18        accum = malloc_removed(A,i);
19        fprintf(fp, "%lld\n", (long long)accum);
20    }
21    fclose(fp);
22
23    // Compare the results file with the golden results
24    retval = system("diff --brief -w result.dat result.golden.dat");
25    if (retval != 0) {
26        printf("Test failed !!!\n");
27        retval=1;
28    } else {
29        printf("Test passed!\n");
30    }
31
32    // Return 0 if the test passed
33    return retval;
34 }

```

Рис. 2.3. Исходный код теста

### 3. Скрипт

Ниже приводится скрипт, для автоматизации выполнения лабораторной работы.

```

1 open_project -reset lab14_1
2
3 add_files malloc_removed.c -cflags "-Wno-cpp"
4 add_files -tb malloc_removed_test.c
5 add_files -tb result.golden.dat
6 set_top malloc_removed
7
8 open_solution -reset solution_1a
9
10 set_part {xa7a12tcsg325-1q}
11 create_clock -period 10ns
12 set_clock_uncertainty 0.1
13
14 csim_design
15 csynth_design
16 cosim_design -trace_level all

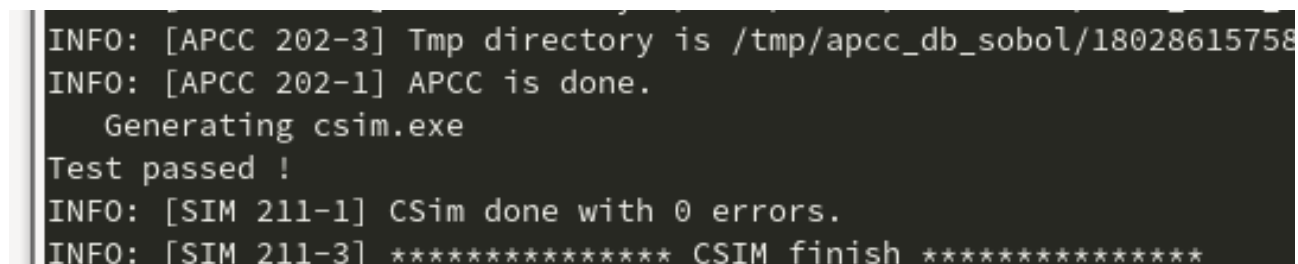
```

Рис. 3.1. Скрипт

## 4. Решение 1a

### 4.1. Моделирование

Ниже приведены результаты моделирования.



```

INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_sobol/18028615758
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  | 6.514     | 0.10        |

### Latency (clock cycles)

#### Summary

| Latency |     | Interval |     |      |
|---------|-----|----------|-----|------|
| min     | max | min      | max | Type |
| 126     | 126 | 126      | 126 | none |

Рис. 4.2. Performance estimates

## Utilization Estimates

### Summary

| Name            | BRAM_18K | DSP48E | FF    | LUT  |
|-----------------|----------|--------|-------|------|
| DSP             | -        | -      | -     | -    |
| Expression      | -        | -      | 0     | 114  |
| FIFO            | -        | -      | -     | -    |
| Instance        | -        | -      | -     | -    |
| Memory          | 1        | -      | 0     | 0    |
| Multiplexer     | -        | -      | -     | 96   |
| Register        | -        | -      | 68    | -    |
| Total           | 1        | 0      | 68    | 210  |
| Available       | 40       | 40     | 16000 | 8000 |
| Utilization (%) | 2        | 0      | ~0    | 2    |

--

Рис. 4.3. Utilization estimates

| Performance Profile |           | Resource Profile |                   |                     |            |
|---------------------|-----------|------------------|-------------------|---------------------|------------|
|                     | Pipelined | Latency          | Iteration Latency | Initiation Interval | Trip count |
| ▼ ● malloc_removed  | -         | 126              | -                 | 127                 | -          |
| ● LOOP_SHIFT        | no        | 62               | 2                 | -                   | 31         |
| ● LOOP_ACCUM        | no        | 62               | 2                 | -                   | 31         |

Рис. 4.4. Performance profile

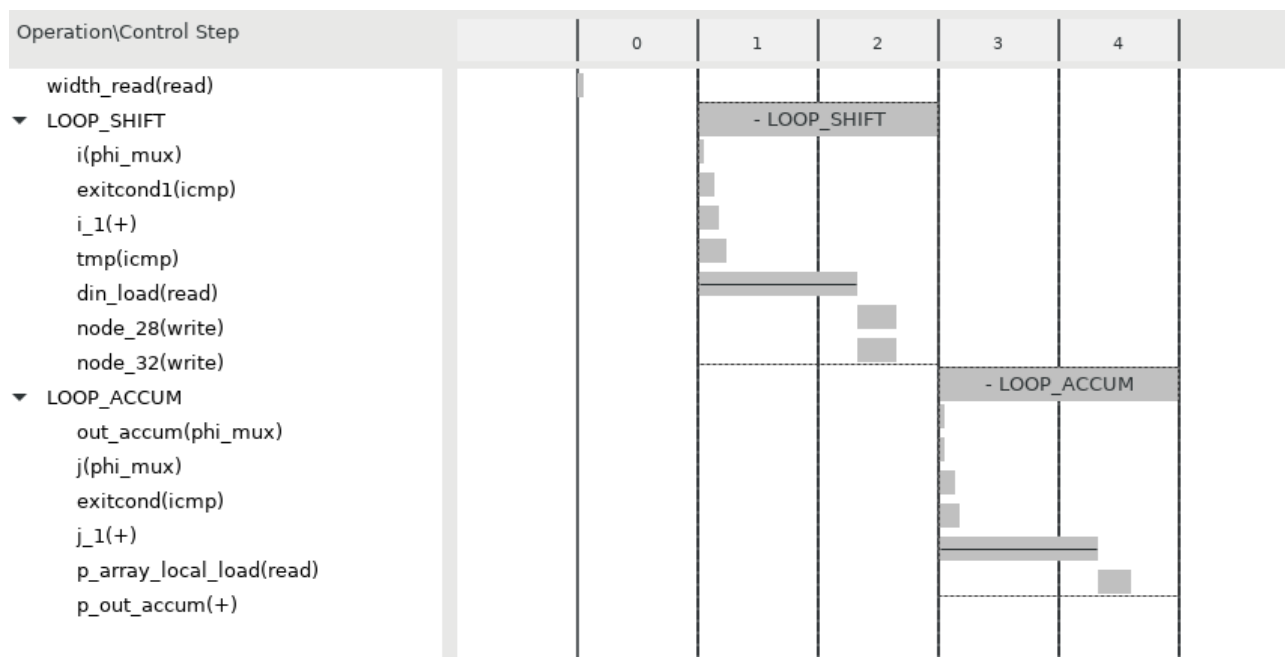


Рис. 4.5. Scheduler viewer

| Resource\Control Step   | C0   | C1      | C2    | C3      | C4 |
|-------------------------|------|---------|-------|---------|----|
| 1 I/O Ports             |      |         |       |         |    |
| 2 width                 | read |         |       |         |    |
| 3 din (p0)              |      | read    |       |         |    |
| 4 ap_return             |      |         |       | ret     |    |
| 5 Memory Ports          |      |         |       |         |    |
| 6 din (p0)              |      | read    |       |         |    |
| 7 p_array_local (p0)    |      |         | write | read    |    |
| 8 Expressions           |      |         |       |         |    |
| 9 i_phi_fu_94           |      | phi_mux |       |         |    |
| 10 i_l_fu_134           |      | +       |       |         |    |
| 11 exitcond1_fu_128     |      | icmp    |       |         |    |
| 12 tmp_fu_140           |      | icmp    |       |         |    |
| 13 j_phi_fu_117         |      |         |       | phi_mux |    |
| 14 out_accum_phi_fu_105 |      |         |       | phi_mux |    |
| 15 j_l_fu_171           |      |         |       | +       |    |
| 16 exitcond_fu_165      |      |         |       | icmp    |    |
| 17 p_out_accum_fu_190   |      |         |       |         | +  |

Рис. 4.6. Resource viewer

Как видно из диаграммы, функция верхнего уровня состоит из 2 циклов: цикл сдвига данных и цикл суммирования данных в аккумулятор. Цикл сдвига состоит из команды чтения, которая требует больше всего времени работы цикла, и сложения данных с аккумулятором. В итоге получаем 2 такта на 1 цикл \* количество циклов 31 и получаем значение Latency = 62 такта. Аналогично для цикла сдвига, дольше всего длится операция чтения данных, а далее 2 операции записи. В данном случае длительность одного



цикла также 2 такта на 1 цикл \* количество циклов, в итоге получаем значение  $\text{Latency} = 62$  Итоговое значение  $\text{Latency} = 62 + 62 + 1$  для инициализации цикла SHIFT + 1 для инициализации цикла ACCUM = 126 Данные будут доступны на выходе через 1 такт  $\text{Initiation Interval} = \text{Latency} + 1 = 127$

## 5. Вывод

В рамках данной работы был показан способ обойти невозможность выделения памяти на аппаратном уровне, путем создания массива данных фиксированного размера, достаточного для сохранения всех данных.