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

Лабораторная №13

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Сравнение типов данных

Задание 1

Студенты:
Соболь В.
Темнова А.С.
Группа: 13541/3
Преподаватель:
Антонов А.П.

Санкт-Петербург
2019

Содержание

1. Задание

1. Создать проект lab13_1
2. Микросхема: xa7a12tcsg325-1q
3. В папке Source имеется 3 папки с описанием одной функции, но разными типами данных
4. Ознакомиться с описаниями функций
5. Ознакомиться с тестами
6. Исследование:
7. Solution_1a – для функции types_standard
    - Осуществить моделирование (с выводом результатов в консоль)
    - 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. Solution_2a – для функции types_float_double
    - Осуществить моделирование (с выводом результатов в консоль)
    - 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

9. Сравнить два решения (solution_1a и solution_2a) и сделать выводы
10. Solution_3a – для функции apint_arith

    - Осуществить моделирование (с выводом результатов в консоль)
    - 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

11. Сравнить два решения (solution_1a и solution_3a) и сделать выводы
12. Сравнить два решения (solution_2a и solution_3a) и сделать выводы

2. Скрипт

Ниже приводится скрипт, для автоматизации выполнения лабораторной работы.

```
set solutions [list types_standard types_float_double apint_arith]

foreach sol $solutions {
  open_project -reset proj_$sol

  set test_file $sol\_test

  add_files "$sol/$sol.c"
  add_files -tb "$sol/$test_file.c"
  add_files -tb "$sol/result.golden.dat"
  set_top $sol

  open_solution solution_$sol -reset
  set_part {xa7a12tcsg325-1q}
  create_clock -period 10ns
  set_clock_uncertainty 0.1

  csim_design
  csynth_design
  # cosim_design -trace_level all
}

exit
```

Рис. 2.1. Скрипт

3. Решение 1a

3.1. Исходный код

Ниже приведен исходный код устройства и теста.

```
#include "types_standard.h"

void types_standard(
  din_A  inA,
  din_B  inB,
  din_C  inC,
  din_D  inD,
  dout_1 *out1,
  dout_2 *out2,
  dout_3 *out3,
  dout_4 *out4
  ) {

  // Basic arithmetic operations
  *out1 = inA * inB;
  *out2 = inB + inA;
  *out3 = inC / inA;
  *out4 = inD % inA;

}
```

Рис. 3.1. Исходный код устройства

```
1  #ifndef _TYPES_STANDARD_H_
2  #define _TYPES_STANDARD_H_
3
4  #include <stdio.h>
5  #include <stdint.h>
6
7  #define N 9
8
9  typedef char din_A;
10 typedef short din_B;
11 typedef int din_C;
12 typedef long long din_D;
13
14 typedef int dout_1;
15 typedef unsigned char dout_2;
16 typedef int32_t dout_3;
17 typedef int64_t dout_4;
18
19 void types_standard(din_A inA,din_B inB,din_C inC,din_D inD,dout_1 *out1,dout_2
       ↪ *out2,dout_3 *out3,dout_4 *out4);
20
21 #endif
```

Рис. 3.2. Заголовочный файл

```
1 #include "types_standard.h"
2
3 int main () {
4   din_A  inA;
5   din_B  inB;
6   din_C  inC;
7   din_D  inD;
8   dout_1 out1;
9   dout_2 out2;
10  dout_3 out3;
11  dout_4 out4;
12
13  int i, retval=0;
14  FILE        *fp;
15
16  // Save the results to a file
17  fp=fopen("result.dat","w");
18
19  for (i=0;i<N;i++) {
20    // Create input data
21    inA = i+2;
22    inB = i+23;
23    inC = i+234;
24    inD = i+2345;
25
26        // Call the function to operate on the data
27        types_standard(inA,inB,inC,inD,&out1,&out2,&out3,&out4);
28 #ifndef __MINGW32__
29    fprintf(fp, "%d*%d=%d; %d+%d=%d; %d/%d=%d; %lld mod %d=%d;\n", inA, inB,
      ↪ out1, inB, inA, out2, inC, inA, out3, inD, inA,out4);
30 #else
31    fprintf(fp, "%d*%d=%d; %d+%d=%d; %d/%d=%d; %I64d mod %d=%d;\n", inA, inB,
      ↪ out1, inB, inA, out2, inC, inA, out3, inD, inA,out4);
32 #endif
33  }
34  fclose(fp);
35
36  // Compare the results file with the golden results
37  retval = system("diff --brief -w result.dat result.golden.dat");
38  if (retval != 0) {
39    printf("Test failed  !!!\n");
40    retval=1;
41  } else {
42    printf("Test passed !\n");
43  }
44
45  // Return 0 if the test passed
46  return retval;
47 }
```

Рис. 3.3. Исходный код теста

3.2. Моделирование

Ниже приведены результаты моделирования.

```
INFO: [HLS 200-10] In directory '/home/sobol/Downloads/labs_from_8/lab13_z1/sour
standard/csim/build'
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_sobol/174375157589625330 3997
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 ***************
INFO: [HLS 200-10] Analyzing design file 'types_standard/types_standard.c' ...
```

Рис. 3.4. Результаты моделирования

По результатам моделирования видно, что устройство работает корректно.

3.3. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.180	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
67	67	67	67	none

Рис. 3.5. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	1	-	-
Expression	-	-	0	15
FIFO	-	-	-	-
Instance	-	-	1173	707
Memory	-	-	-	-
Multiplexer	-	-	-	309
Register	-	-	68	-
Total	0	1	1241	1031
Available	40	40	16000	8000
Utilization (%)	0	2	7	12

Рис. 3.6. Utilization estimates

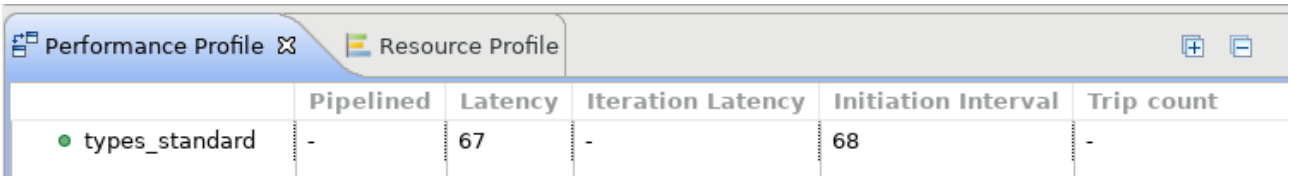

Рис. 3.7. Performance profile

Рис. 3.8. Scheduler viewer

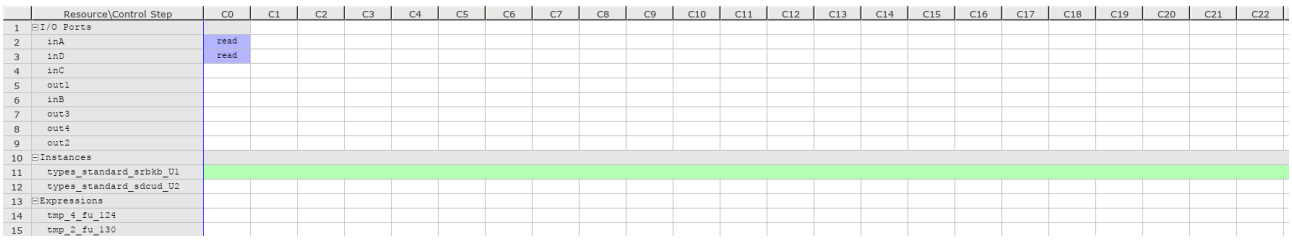

Рис. 3.9. Resource viewer

Исходя из диаграммы видно, что дольше всего выполняются операции деления нацело и деление с остатком что связано с используемым типом данных.

4. Решение 2a

4.1. Исходный код

Ниже приведен исходный код устройства и теста.

```
#include "types_float_double.h"

void types_float_double(
  din_A  inA,
  din_B  inB,
  din_C  inC,
  din_D  inD,
  dout_1 *out1,
  dout_2 *out2,
  dout_3 *out3,
  dout_4 *out4
  ) {

  // Basic arithmetic and math.h sqrt()
  *out1 = inA * inB;
  *out2 = inB + inA;
  *out3 = inC / inA;
  *out4 = sqrtf(inD);

}
```

Рис. 4.1. Исходный код устройства

```
#ifndef _TYPES_FLOAT_DOUBLE_H_
#define _TYPES_FLOAT_DOUBLE_H_

#include <stdio.h>
#include <stdint.h>
#include <math.h>

#define N 9

typedef double din_A;
typedef double din_B;
typedef double din_C;
typedef float din_D;

typedef double dout_1;
typedef double dout_2;
typedef double dout_3;
typedef float dout_4;

void types_float_double(din_A inA,din_B inB,din_C inC,din_D inD,dout_1 *out1,
    ↪ dout_2 *out2,dout_3 *out3,dout_4 *out4);

#endif
```

Рис. 4.2. Заголовочный файл

```
#include "types_float_double.h"

int main () {
  din_A  inA;
  din_B  inB;
  din_C  inC;
  din_D  inD;
  dout_1 out1;
  dout_2 out2;
  dout_3 out3;
  dout_4 out4;

  int i, retval=0;
  FILE        *fp;

  // Save the results to a file
  fp=fopen("result.dat","w");

  for (i=0;i<N;i++) {
    // Create input data
    inA = i+23.1;
    inB = i+23.12;
    inC = i+234.123;
    inD = i+2345.1234;

    // Call the function to operate on the data
    types_float_double(inA,inB,inC,inD,&out1,&out2,&out3,&out4);
        // Compare the first 5 digits after the decimal point for sqrtf result
    fprintf(fp, "%f*%f=%f; %f+%f=%f; %f/%f=%f; %f sqrt =%.5f;\n", inA, inB, out1
    ↪ , inB, inA, out2, inC, inA, out3, inD, out4);
  }
  fclose(fp);

  // Compare the results file with the golden results
  retval = system("diff --brief -w result.dat result.golden.dat");
  if (retval != 0) {
    printf("Test failed  !!!\n");
    retval=1;
  } else {
    printf("Test passed !\n");
  }

  // Return 0 if the test passed
  return retval;
}
```

Рис. 4.3. Исходный код теста

4.2. Моделирование

Ниже приведены результаты моделирования.

```
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_sobol/174548157589627033l275
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. 4.4. Результаты моделирования

По результатам моделирования видно, что устройство работает корректно.

4.3. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.546	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
30	30	30	30	none

Рис. 4.5. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	-	-
FIFO	-	-	-	-
Instance	-	14	4379	5977
Memory	-	-	-	-
Multiplexer	-	-	-	145
Register	-	-	31	-
Total	0	14	4410	6122
Available	40	40	16000	8000
Utilization (%)	0	35	27	76

Рис. 4.6. Utilization estimates

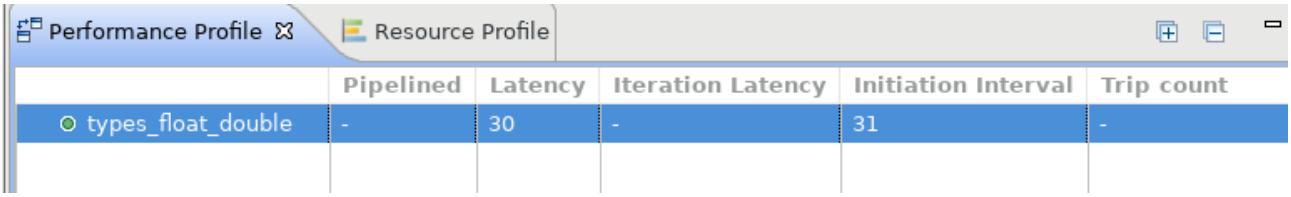

Рис. 4.7. Performance profile

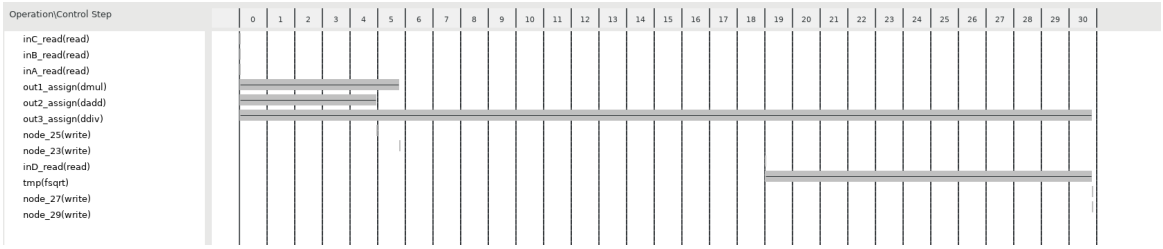

Рис. 4.8. Scheduler viewer

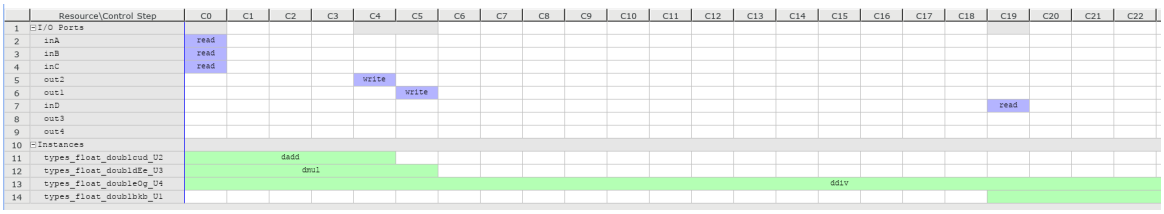

Рис. 4.9. Resource viewer

Как видно, в данном проекте используется наибольшее количество ресурсов в связи с тем что используются 64 и 32-битные типы данных для синтеза которых требуется больше FF и LUT чем для типов данных char, short.

5. Решение 3a

5.1. Исходный код

Ниже приведен исходный код устройства и теста.

```
#include "apint_arith.h"

void apint_arith(dinA_t  inA, dinB_t  inB, dinC_t  inC, dinD_t  inD,
                 dout1_t *out1, dout2_t *out2, dout3_t *out3, dout4_t *out4
  ) {

  // Basic arithmetic operations
  *out1 = inA * inB;
  *out2 = inB + inA;
  *out3 = inC / inA;
  *out4 = inD % inA;

}
```

Рис. 5.1. Исходный код устройства

```
#ifndef _APINT_ARITH_H_
#define _APINT_ARITH_H_

#include <stdio.h>
#include "ap_cint.h"

#define N 9

typedef int6 dinA_t;
typedef int12 dinB_t;
typedef int22 dinC_t;
typedef int33 dinD_t;

typedef int18 dout1_t;
typedef uint13 dout2_t;
typedef int22 dout3_t;
typedef int6 dout4_t;

void apint_arith(dinA_t inA,dinB_t inB,dinC_t inC,dinD_t inD,dout1_t *out1,
    ↪ dout2_t *out2,dout3_t *out3,dout4_t *out4);

#endif
```

Рис. 5.2. Заголовочный файл

```
1  #include "apint_arith.h"
2
3  int main () {
4    dinA_t  inA;
5    dinB_t  inB;
6    dinC_t  inC;
7    dinD_t  inD;
8    dout1_t out1;
9    dout2_t out2;
10   dout3_t out3;
11   dout4_t out4;
12
13   int i, retval=0;
14   FILE        *fp;
15
16   // Save the results to a file
17   fp=fopen("result.dat","w");
18
19   for (i=0;i<N;i++) {
20     // Create input data
21     inA = i+2;
22     inB = i+23;
23     inC = i+234;
24     inD = i+2345;
25
26     // Call the function to operate on the data
27         apint_arith(inA,inB,inC,inD,&out1,&out2,&out3,&out4);
28 #ifndef __MINGW32__
29     fprintf(fp, "%d*%d=%d; %d+%d=%d; %d/%d=%d; %lld mod %d=%d;\n", inA, inB,
       ↪ out1, inB, inA, out2, inC, inA, out3, inD, inA,out4);
30 #else
31     fprintf(fp, "%d*%d=%d; %d+%d=%d; %d/%d=%d; %I64d mod %d=%d;\n", inA, inB,
       ↪ out1, inB, inA, out2, inC, inA, out3, inD, inA,out4);
32 #endif
33   }
34   fclose(fp);
35
36   // Compare the results file with the golden results
37   retval = system("diff --brief -w result.dat result.golden.dat");
38   if (retval != 0) {
39     printf("Test failed  !!!\n");
40     retval=1;
41   } else {
42     printf("Test passed !\n");
43   }
44
45   // Return 0 if the test
46   return retval;
47 }
```

Рис. 5.3. Исходный код теста

5.2. Моделирование

Ниже приведены результаты моделирования.

```
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_sobol/174703157589628778285O
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. 5.4. Результаты моделирования

По результатам моделирования видно, что устройство работает корректно.

5.3. Синтез

По оценке производительности видно, что устройство соответствует заданным критериям.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.180	0.10

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
36	36	36	36	none

Рис. 5.5. Performance estimates

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	1	-	-
Expression	-	-	0	20
FIFO	-	-	-	-
Instance	-	-	681	415
Memory	-	-	-	-
Multiplexer	-	-	-	169
Register	-	-	37	-
Total	0	1	718	604
Available	40	40	16000	8000
Utilization (%)	0	2	4	7

Рис. 5.6. Utilization estimates

Рис. 5.7. Performance profile

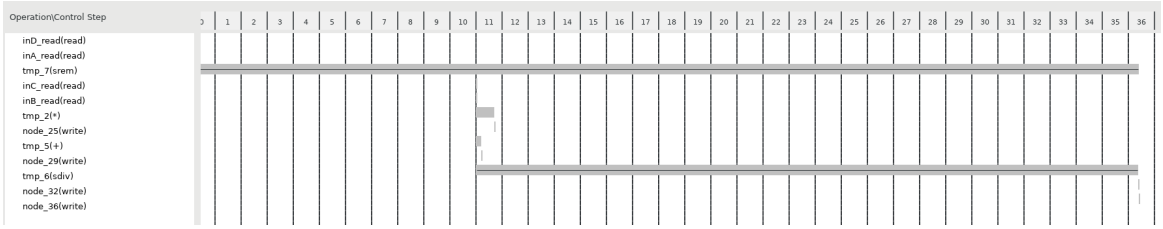

Рис. 5.8. Scheduler viewer

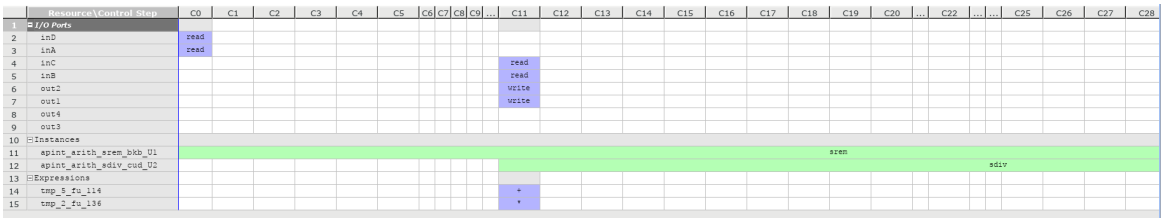

Рис. 5.9. Resource viewer

Как видно, в данном проекте значение Latency чуть больше, чем в лучшем решении и количество требуемых ресурсов чуть больше, чем в лучшем решении. Это связано с тем что используются оптимальные для задачи типы данных. В связи с этим требуется оптимальное количество ресурсов.

6. Вывод

В ходе работы была исследована одна функция с разными типами данных.

Было установлено, что чем лучше подобраны типы данных, на основе анализа возможных значений, тем оптимальнее синтезированная схема.