

2019

Оглавление

Задание.....	3
Решение	5
Выводы	13

Задание

Создать проект, подключив готовые файлы исходного кода устройства и теста к нему.

Исходный код функции:

```
void lab1_2 (int in[3], char a, char b, char c, int out[3]) {
    int x,y;
    for(int i = 0; i < 3; i++) {
        x = in[i];
        y = a*x + b + c;
        out[i] = y;
    }
}
```

Исходный код теста:

```
#include <stdio.h>
int main()
{
    int In[3] = {1,3,9};
    int inA, inB, inC;
    int res[3];
    // For adders
    int refOut[9] = {30, 40, 70, 90, 140, 290, 150, 240, 510};
    int pass;
    int i;
    inA = 5;
    inB = 10;
    inC = 15;
    for (i=0; i<3; i++)
    {
        lab1_2(In, inA, inB, inC, res);
        for (int j=0; j<3; j++)
        {
            fprintf(stdout, " %d*d+%d+%d=%d \n", inA, In[j], inB, inC, res[j]);

            // Test the output against expected results
            if (res[j] == refOut[i*3+j])
                pass = 1;
            else
                pass = 0;
        }
        inA=inA+20;
        inB=inB+20;
        inC=inC+20;
    }
    if (pass)
    {
        fprintf(stdout, "-----Pass!-----\n");
        return 0;
    }
    else
    {
        fprintf(stderr, "-----Fail!-----\n");
        return 1;
    }
}
```

Создать 2 решения для синтеза устройства на основе *ха7a12tcsg325-1q*:
задать clock period 6, а также clock uncertainty 0.1.

Создать скрипт автоматизирующий процесс:

- Создать проект lab2_1
- Подключить файл lab2_1.c (папка source)
- Подключить тест lab2_1_test.c (папка source)
- Микросхема: ха7a12tcsg325-1q

Сделать solution1

- задать: clock period 6; clock_uncertainty 0.1
- осуществить моделирование
- осуществить синтез
- открыть GUI
- проверить работу созданного скрипта.

Не стирая результаты работы предыдущего запуска скрипта, запустить скрипт еще раз и проверить корректность работы при повторном запуске.

Решение

При создании решения зададим настройки: clock period 6, clock uncertain 0.1, устройство xa7a12tcs325-1q.

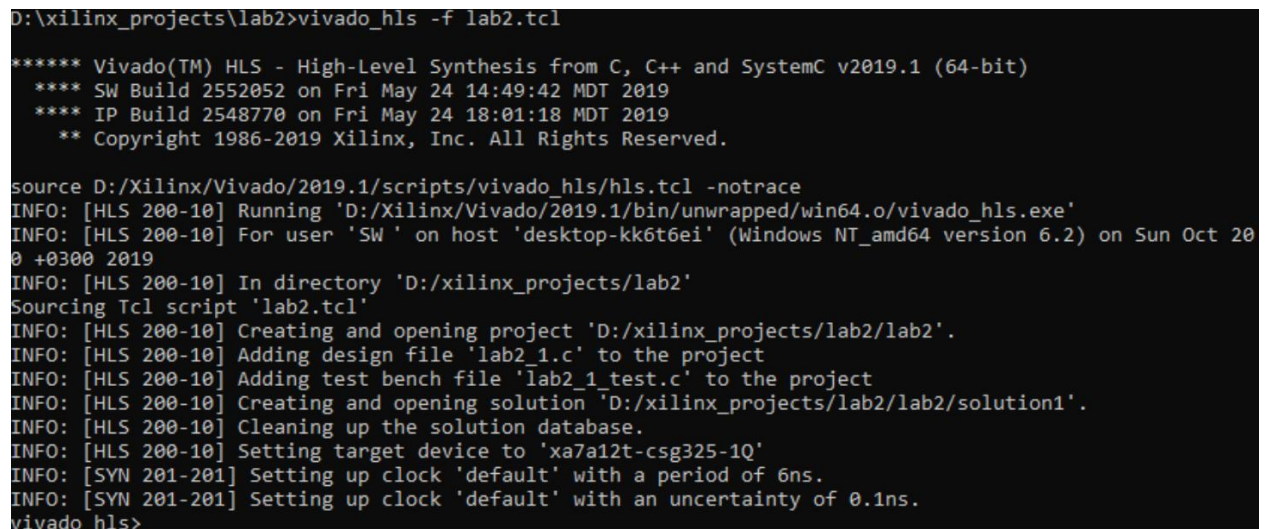
Для автоматизации создания проекта, напишем tcl скрипт, содержащий следующие команды:

```
open_project -reset lab2
set_top lab1_1
add_files lab2_1.c
add_files -tb lab2_1_test.c
open_solution solution1 -reset
set_part {xa7a12tcs325-1q}
create_clock -period 6ns
set_clock_uncertainty 0.1ns
```

Рис.1. Скрипт создания проекта

Команды скрипта полностью соответствуют настройкам в GUI.

Существующий набор команд можно получить, выполнив команду help при работе в режиме интерпретатора (vivado_hls -i).



```
D:\xilinx_projects\lab2>vivado_hls -f lab2.tcl

***** Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC v2019.1 (64-bit)
***** SW Build 2552052 on Fri May 24 14:49:42 MDT 2019
**** IP Build 2548770 on Fri May 24 18:01:18 MDT 2019
** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

source D:/Xilinx/Vivado/2019.1/scripts/vivado_hls/hls.tcl -notrace
INFO: [HLS 200-10] Running 'D:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/vivado_hls.exe'
INFO: [HLS 200-10] For user 'SW' on host 'desktop-kk6t6ei' (Windows NT_amd64 version 6.2) on Sun Oct 20
0 +0300 2019
INFO: [HLS 200-10] In directory 'D:/xilinx_projects/lab2'
Sourcing Tcl script 'lab2.tcl'
INFO: [HLS 200-10] Creating and opening project 'D:/xilinx_projects/lab2/lab2'.
INFO: [HLS 200-10] Adding design file 'lab2_1.c' to the project
INFO: [HLS 200-10] Adding test bench file 'lab2_1_test.c' to the project
INFO: [HLS 200-10] Creating and opening solution 'D:/xilinx_projects/lab2/lab2/solution1'.
INFO: [HLS 200-10] Cleaning up the solution database.
INFO: [HLS 200-10] Setting target device to 'xa7a12t-csg325-1Q'
INFO: [SYN 201-201] Setting up clock 'default' with a period of 6ns.
INFO: [SYN 201-201] Setting up clock 'default' with an uncertainty of 0.1ns.
vivado_hls>
```

Рис. 2 Загрузка tcl скрипта

Проверим на рис.3, что проект создался верно: подключены исходные и тестовые файлы, появился раздел Solution.

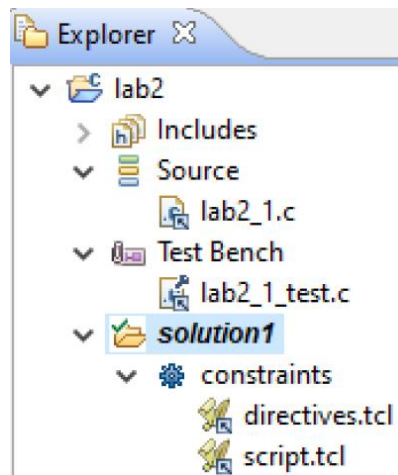


Рис. 3: Структура созданного проекта

Запустим интерактивный режим и выполним моделирование. Результаты успешного моделирования представлены на Рис. 4.

```
D:\xilinx_projects\lab2\lab2\solution1>vivado_hls -i

***** Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC v2019.1 (64-bit)
**** SW Build 2552052 on Fri May 24 14:49:42 MDT 2019
**** IP Build 2548770 on Fri May 24 18:01:18 MDT 2019
** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

source D:/Xilinx/Vivado/2019.1/scripts/vivado_hls/hls.tcl -notrace
INFO: [HLS 200-10] Running 'D:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/vivado_hls.exe'
INFO: [HLS 200-10] For user 'SW' on host 'desktop-kk6t6ei' (Windows NT_amd64 version 6.2) on Sun Oct 20 17:14
8 +0300 2019
INFO: [HLS 200-10] In directory 'D:/xilinx_projects/lab2/lab2/solution1'
vivado_hls> csim_design
INFO: [SIM 211-2] ***** CSIM start *****
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
make: 'csim.exe' is up to date.
  10*20+30+40=270
  20*30+40+50=690
  30*40+50+60=1310
-----Pass!-----
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
```

Рис. 4. Результаты успешного моделирования.

На следующем шаге выполним синтез с помощью команды `csynth_design`.

Результаты синтеза программы представлены на Рис.5.

На рисунке видны логи, демонстрирующее процесс имплементации `lab_1` и генерации RTL для модели `lab_1`.

```

vivado_hls> open_project lab2
INFO: [HLS 200-10] Opening project 'D:/xilinx_projects/lab2/lab2'.
v
vivado_hls> open_solution sol
No match found.
vivado_hls> open_solution solution1
INFO: [HLS 200-10] Opening solution 'D:/xilinx_projects/lab2/lab2/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 6ns.
INFO: [SYN 201-201] Setting up clock 'default' with an uncertainty of 0.1ns.
INFO: [HLS 200-10] Setting target device to 'xa7a12t-csg325-1Q'
D:/xilinx_projects/lab2/lab2/vivado_hls.app
vivado_hls> csynth_design
INFO: [SCHED 204-61] Option 'relax_ii_for_timing' is enabled, will increase II to
INFO: [HLS 200-10] Analyzing design file 'lab2_1.c' ...
INFO: [HLS 200-111] Finished Linking Time (s): cpu = 00:00:01 ; elapsed = 00:00:4
INFO: [HLS 200-111] Finished Checking Pragmas Time (s): cpu = 00:00:01 ; elapsed
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms Time (s): cpu = 00:00:01 ; elaps
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability Time (s): cpu = 00:00:01 ;
INFO: [HLS 200-111] Finished Pre-synthesis Time (s): cpu = 00:00:01 ; elapsed = 6
INFO: [HLS 200-111] Finished Architecture Synthesis Time (s): cpu = 00:00:01 ; el
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'lab1_1' ...
INFO: [HLS 200-10] -----
INFO: [HLS 200-42] -- Implementing module 'lab1_1'
INFO: [HLS 200-10] -----
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111] Elapsed time: 48.465 seconds; current allocated memory: 91.6
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
INFO: [BIND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111] Elapsed time: 0.027 seconds; current allocated memory: 91.67
INFO: [HLS 200-10] -----
INFO: [HLS 200-10] -- Generating RTL for module 'lab1_1'
INFO: [HLS 200-10] -----
INFO: [RTGEN 206-500] Setting interface mode on port 'lab1_1/a' to 'ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on port 'lab1_1/b' to 'ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on port 'lab1_1/c' to 'ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on port 'lab1_1/d' to 'ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on function 'lab1_1' to 'ap_ctrl_hs'.
INFO: [SYN 201-210] Renamed object name 'lab1_1_mac_muladd_8s_8s_9s_16_3_1' to '1
INFO: [RTGEN 206-100] Generating core module 'lab1_1_mac_muladdbkb': 1 instance(s)
INFO: [RTGEN 206-100] Finished creating RTL model for 'lab1_1'.
INFO: [HLS 200-111] Elapsed time: 0.042 seconds; current allocated memory: 91.84
INFO: [HLS 200-111] Finished generating all RTL models Time (s): cpu = 00:00:01 ;
INFO: [VHDL 208-304] Generating VHDL RTL for lab1_1.
INFO: [VLOG 209-307] Generating Verilog RTL for lab1_1.
vivado_hls>

```

Рис. 5: Лог-отчет результатов синтеза

Откроем GUI и рассмотрим результаты производительности.

Performance Estimates			
Timing (ns)			
Summary			
Clock	Target	Estimated	Uncertainty
ap_clk	6.00	3.820	0.10

Рис. 6. Производительность

Здесь можно увидеть, что достигнутая задержка равна $3.820 + 0.1$, что укладывается в заданные нами требования к тактовой частоте

Utilization Estimates					
Summary					
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	-	0	16	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	21	-
Register	-	-	12	-	-
Total	0	1	12	37	0
Available	40	40	16000	8000	0
Utilization (%)	0	2	~0	~0	0

Рис. 7. Занимаемые ресурсы

Данный проект займет на микросхеме 1 DSP блок (в котором будут использованы и сумматоры и умножитель), 12 триггеров для хранения чисел, и 37 LUT.

Профиль производительности и профиль ресурсов представлен на рисунке 8 и рисунке 9.

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
lab1_2	-	19	-	20	-
Loop 1	no	18	6	-	3

Рис. 8. Профиль производительности

Performance Profile		Resource Profile								
	BRAM	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth	Words	W*Bits*Banks
lab1_1	0	1	12	37						
> I/O Ports(4)					32					
> Instances(0)	0	0	0	0						
> Memories(0)	0		0	0	0			0	0	0
> Expressions(1)	0	0	0	16	9	9	0			
> Registers(2)			12		12					
> Channels(0)	0		0	0	0			0	0	0
> Multiplexers(1)	0		0	21	1			0		
> DSP(1)		1								

Рис. 9. Профиль ресурсов

Проанализировав отчет производительности, представленного на рисунке 9, можем сделать вывод, что задержка до получения результата — 2 такта, еще через 1 такт можно записывать новые данные.

Временная диаграмма выполнения операций представлена на Рис.10.

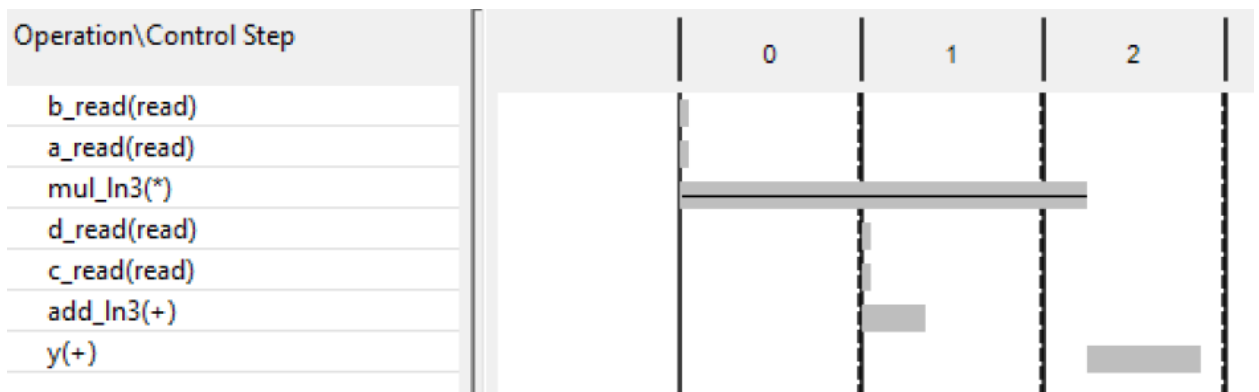


Рис. 10. Временная диаграмма выполнения операций

Проведем RTL моделирование с использованием команды `cosim_design`. Результат лог-отчета об успешном моделировании представлен на рисунке 11.

```

**** IP Build 2548770 on Fri May 24 18:01:18 MDT 2019
** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

source xsim.dir/lab1_1/xsim_script.tcl
# xsim {lab1_1} -autoloadwcfg -tclbatch {lab1_1.tcl}
Vivado Simulator 2019.1
Time resolution is 1 ps
source lab1_1.tcl
## run all
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @ "Simulation Time"
//
// RTL Simulation : 0 / 3 [0.00%] @ "117000"
// RTL Simulation : 1 / 3 [100.00%] @ "141000"
// RTL Simulation : 2 / 3 [100.00%] @ "159000"
// RTL Simulation : 3 / 3 [100.00%] @ "177000"
//
$finish called at time : 201 ns : File "D:/xilinx_projects/lab2/lab2/solution1/sim/verilog/lab1_1.autotb.v" Line 440
## quit
INFO: [Common 17-206] Exiting xsim at Sun Oct 20 17:32:07 2019...
INFO: [COSIM 212-316] Starting C post checking ...
10*20+30+40=270
20*30+40+50=690
30*40+50+60=1310
-----Pass!-----
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
vivado_hls>

```

Рис. 11: Лог-отчет результатов моделирования

Выполним экспорт с использованием команды `export_design`. Результат лог-отчета об успешном экспорте представлен на рисунке 12.

```

Implementation tool: Xilinx Vivado v.2019.1
Project:             lab2
Solution:            solution1
Device target:       xa7a12t-csg325-1Q
Report date:         Sun Oct 20 17:36:02 +0300 2019

#=== Post-Implementation Resource usage ===
SLICE:               4
LUT:                 11
FF:                  3
DSP:                 1
BRAM:                0
SRL:                 0
#=== Final timing ===
CP required:         6.000
CP achieved post-synthesis: 2.154
CP achieved post-implementation: 2.154
Timing met

HLS EXTRACTION: generated D:/xilinx_projects/lab2/lab2/solution1/impl/report/verilog/lab1_1_export.rpt
INFO: [Common 17-206] Exiting Vivado at Sun Oct 20 17:36:03 2019...
vivado_hls>

```

Рис. 12: Лог-отчет результатов экспорта

Дерево структуры выполненного проекта представлено на Рис. 13.

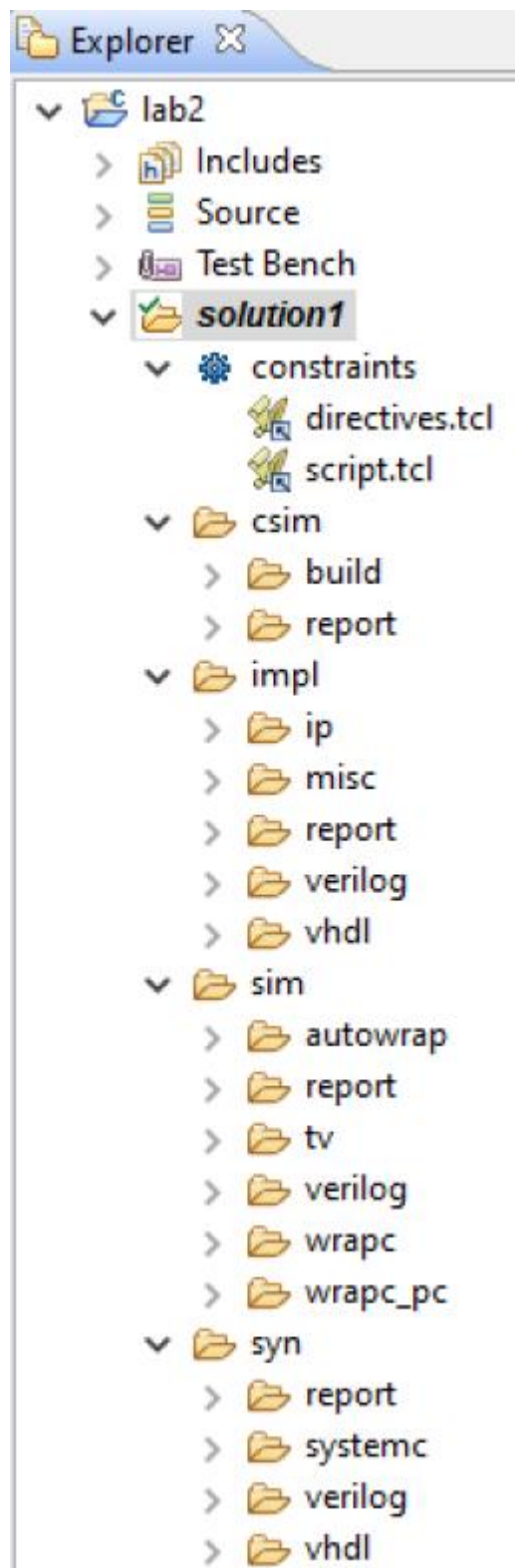


Рис. 13.Дерево проекта

Скрипт для автоматизации создания проекта можно расширить последними командами и проверить его работоспособность.

```
open_project -reset lab2
set_top lab1_1
add_files lab2_1.c
add_files -tb lab2_1_test.c
open_solution solution1 -reset
set_part {xa7a12tcsg325-1q}
create_clock -period 6ns
set_clock_uncertainty 0.1ns
csim_design
csynth_design
cosim_design
export_design -flow impl -format ip_catalog
```

Рис. 13. Скрипт для создания проекта

При запуске скрипта получили точно такие же результат, как и продемонстрированы выше.

Часть логов, отвечающая за C/RTL моделирование и генерацию приведена на рисунках 14 и 15.

```
INFO: [Common 17-206] Exiting xsim at Sun Oct 20 17:32:07 2019...
INFO: [COSIM 212-316] Starting C post checking ...
  10*20+30+40=270
  20*30+40+50=690
  30*40+50+60=1310
-----Pass!-----
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
```

Рис.14. Логи об успешном C/RTL моделировании

```
INFO: [APCC 202-1] APCC is done.
  Generating cosim.tv.exe
INFO: [COSIM 212-302] Starting C TB testing ...
  10*20+30+40=270
  20*30+40+50=690
  30*40+50+60=1310
-----Pass!-----
INFO: [COSIM 212-333] Generating C post check test bench ...
```

Рис.15. Логи об успешной генерации

Выводы

В ходе выполнения лабораторной работы были изучены методы работы с Vivado HLS Command Prompt. Был создан проект с решением по заданным исходным, тестовым файлам и параметрам. Процесс создания проекта и моделирования был автоматизирован при помощи tcl скрипта, который можно использовать для подгрузки проекта прямо из командной строки. Результаты работы скрипта не изменились после его перезапуска.