

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

Отчет по лабораторной работе №6_1
Курс: «Проектирование реконфигурируемых гибридных
вычислительных систем»
Тема: «Port-level IO protocols»

Выполнил студент гр. 3540901/81501

Селиверстов Я.А.

(подпись)

Руководитель

Антонов А.П.

(подпись)

“ ____ ” _____ 2019 г.

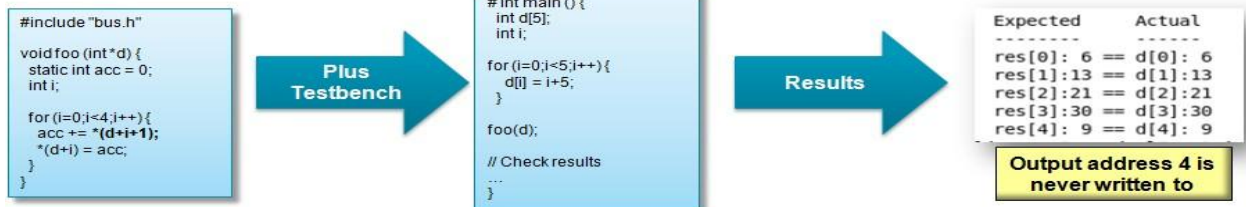
Санкт – Петербург
2019

ОГЛАВЛЕНИЕ

1. Задание.....	3
2. Первое решение.....	5
2.1. Исходный код программы и теста.....	5
2.2. Моделирование	6
2.3. Синтез.....	6
2.4. C RTL моделирование	8
3. Второе решение	10
3.1. Настройки второго решения	10
3.2. Моделирование	10
3.3. Синтез.....	11
3.4. C RTL моделирование	13
4. Выводы.....	14

1. Задание

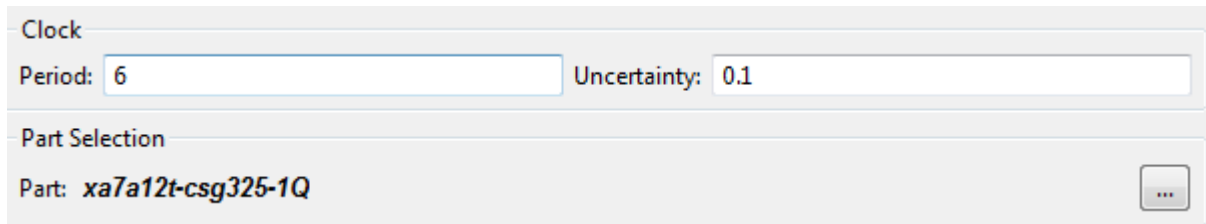
- Создать проект lab6_1
- Микросхема: ха7a12tcsg325-1q
- Создать Си код на основе слайда (функция foo)



- Создать тест lab6_1_test.c на основе слайда выше.
- Сделать solution1
 - задать: clock period 6; clock_uncertainty 0.1
 - осуществить моделирование (на основе слайда выше, с выводом результатов в консоль)
 - осуществить синтез (с настройками по умолчанию – интерфейс ap-fifo)
 - привести в отчете:
 - performance estimates=>summary
 - utilization estimates=>summary
 - Performance Profile
 - interface estimates=>summary
 - объяснить какой интерфейс использован для блока (и какие сигналы входят) и для портов (и какие сигналы входят).
 - 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
 - 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

- Сделать solution2
 - Задать протокол
 - a: ap_bus
 - осуществить моделирование
 - осуществить синтез
 - привести в отчете:
 - performance estimates=>summary
 - utilization estimates=>summary
 - Performance Profile
 - interface estimates=>summary
 - объяснить какой интерфейс использован для блока (и какие сигналы входят) и для портов (и какие сигналы входят).
 - 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
 - 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
 - Объяснить отличие протоколов

2. Первое решение



Clock

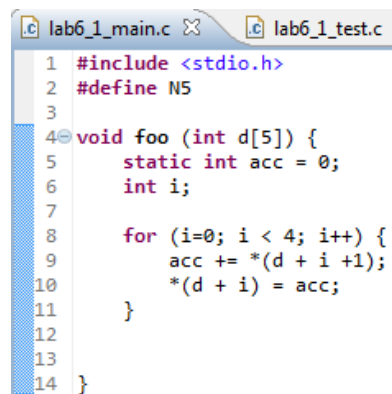
Period: 6 Uncertainty: 0.1

Part Selection

Part: *xa7a12t-csg325-1Q*

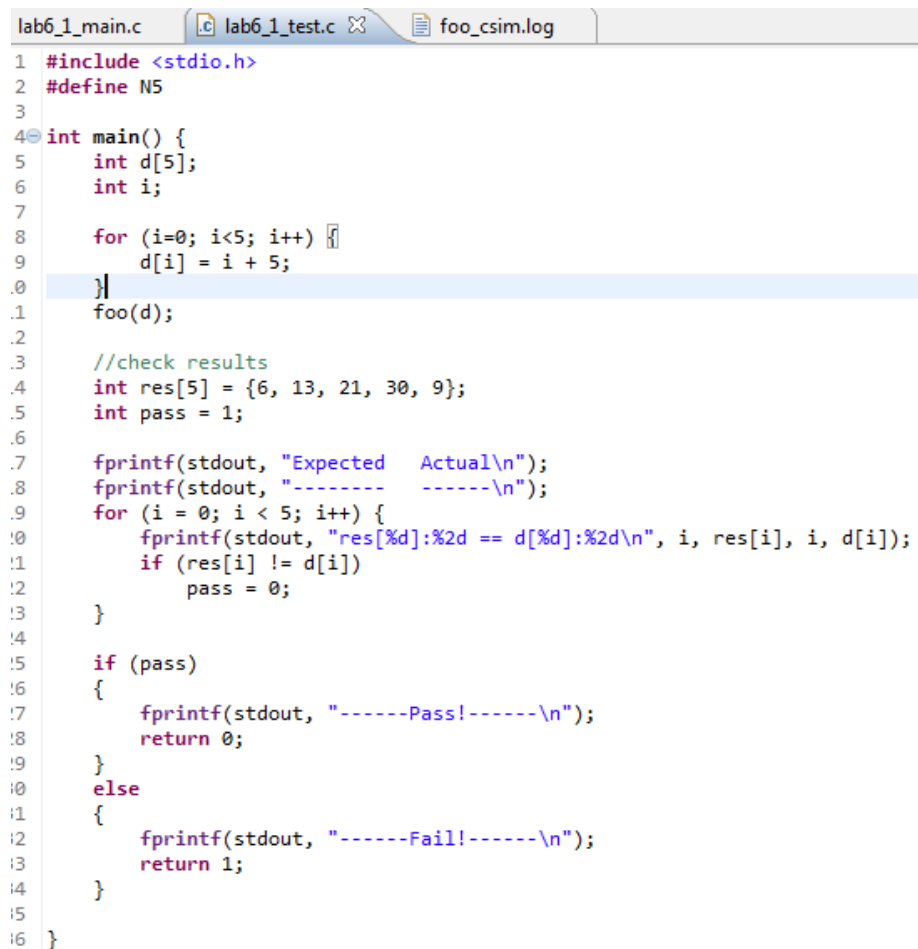
Рисунок 1.1. Параметры первого решения

2.1. Исходный код программы и теста



```
1 #include <stdio.h>
2 #define N5
3
4 void foo (int d[5]) {
5     static int acc = 0;
6     int i;
7
8     for (i=0; i < 4; i++) {
9         acc += *(d + i + 1);
10        *(d + i) = acc;
11    }
12
13
14 }
```

Рисунок 1.2. Исходный код синтезируемой функции



```
1 #include <stdio.h>
2 #define N5
3
4 int main() {
5     int d[5];
6     int i;
7
8     for (i=0; i<5; i++) {
9         d[i] = i + 5;
10    }
11    foo(d);
12
13    //check results
14    int res[5] = {6, 13, 21, 30, 9};
15    int pass = 1;
16
17    fprintf(stdout, "Expected    Actual\n");
18    fprintf(stdout, "-----    -----\\n");
19    for (i = 0; i < 5; i++) {
20        fprintf(stdout, "res[%d]:%2d == d[%d]:%2d\\n", i, res[i], i, d[i]);
21        if (res[i] != d[i])
22            pass = 0;
23    }
24
25    if (pass)
26    {
27        fprintf(stdout, "-----Pass!-----\\n");
28        return 0;
29    }
30    else
31    {
32        fprintf(stdout, "-----Fail!-----\\n");
33        return 1;
34    }
35 }
```

Рисунок 1.3. Исходный код теста

2.2. Моделирование

```
INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'
INFO: [HLS 200-10] For user 'Yaroslav' on host 'svyatoslavpc' (Windows NT amd64 version 6.1) on Thu Jan 02 20:08:33 +0300 2020
INFO: [HLS 200-10] In directory 'C:/Users/Yaroslav/AppData/Roaming/Xilinx/Vivado/lab6_1/solution1/csim/build'
INFO: [APCC 202-3] Tmp directory is apcc_db
INFO: [APCC 202-1] APCC is done.
Generating csim.exe
Expected Actual
-----
res[0]: 6 == d[0]: 6
res[1]:13 == d[1]:13
res[2]:21 == d[2]:21
res[3]:30 == d[3]:30
res[4]: 9 == d[4]: 9
-----Pass!-----
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
Finished C simulation.
```

Рисунок 1.4. Результат успешного моделирования

Моделирование прошло успешно.

2.3. Синтез

Данные о проекте

[-] **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	6.00	4.854	0.10

[-] **Latency (clock cycles)**

[-] **Summary**

Latency		Interval		
min	max	min	max	Type
13	13	13	13	none

Рисунок 1.5. Performance estimates – summary

Здесь можно увидеть, что достигнутая задержка равна $4.854 + 0.1$, что укладывается в заданные нами требования к тактовой частоте.

[-] **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	60	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	51	-
Register	-	-	74	-	-
Total	0	0	74	111	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	1	0

Рисунок 1.6. Utilization estimates – summary

Данный проект займет на микросхеме 74 регистра для хранения чисел и 111 LUT.

Interface					
Summary					
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo	return value
ap_rst	in	1	ap_ctrl_hs	foo	return value
ap_start	in	1	ap_ctrl_hs	foo	return value
ap_done	out	1	ap_ctrl_hs	foo	return value
ap_idle	out	1	ap_ctrl_hs	foo	return value
ap_ready	out	1	ap_ctrl_hs	foo	return value
d_address0	out	3	ap_memory	d	array
d_ce0	out	1	ap_memory	d	array
d_we0	out	1	ap_memory	d	array
d_d0	out	32	ap_memory	d	array
d_q0	in	32	ap_memory	d	array

Рисунок 1.7. Interface estimates – summary

Для расчета схемы требуется более одного такта. На рисунке 1.7 представлены интерфейсы, которые используются в синтезированном устройстве. Видно, что в схеме применяется протокол ap_memory. Порты d_d0 и d_q0 32-битные.

Performance Profile					
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo	-	13	-	14	-
Loop 1	no	12	3	-	4

Рисунок 1.8. Performance Profile

На рисунке 1.8 видно, что задержка получения первого выходного значения составляет 3 такта с момента старта (всех данных – 13), а задержка после старта до готовности приема новых данных – 14.

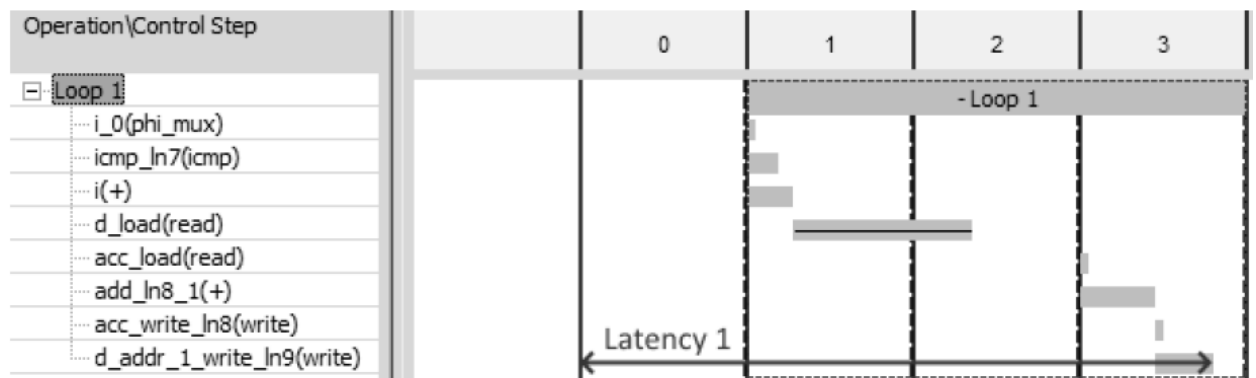


Рисунок 1.9. Schedule viewer

На рисунке 1.9 в первом такте происходит подготовка к запуску цикла, вероятно. Далее на первом такте каждой итерации цикла инициализация переменной-счетчика, проверка условия завершения, увеличение счётчика и начало считывания данных из массива d. На втором – завершение считывания данных из массива d. На третьем – чтение сохраненного значения аккумулятора, прибавление к нему значение элемента, прочитанного из d, запись результата обратно в аккумулятор и в массив d.

Рассмотрим профиль ресурсов:

	BRAM	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth	Words	W*Bits*Banks
foo	0	0	74	111						
I/O Ports(1)					32					
Instances(0)	0	0	0	0						
Memories(0)	0		0	0	0		0		0	0
Expressions(3)	0	0	0	60	38	37	0			
Registers(5)			74		74					
Channels(0)	0		0	0	0		0		0	0
Multiplexers(3)	0		0	51	7		0			
DSP(0)		0								

Рисунок 1.10. Resource Profile.

Значения в Resource Profile на рис. 1.10 совпадают с результатами синтеза на рис. 1.6.

2.4. C|RTL моделирование

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	13	13	13	NA	NA	NA

Рисунок 1.11. Cosimulation Report

При совместном моделировании, программа на рисунке 1.11 отобразила те же самые, ожидаемые нами значения Latency и П.

Покажем на рисунке 1.12 временную диаграмму совместного моделирования с отмеченными на ней Latency и II:



Рисунок 1.12. Design Top Signals

3. Второе решение

3.1. Настройки второго решения

The screenshot shows the 'Clock' section with 'Period' set to 6 and 'Uncertainty' set to 0.1. Below it, the 'Part Selection' section shows the part 'xa7a12t-csg325-1Q' selected. A small button with three dots is visible on the right side of the part selection area.

Рисунок 2.1. Параметры второго решения

Добавим директиву, которая изменяет используемый port-level протокол

The screenshot shows the 'Vivado HLS Directive Editor' window. The 'Directive' dropdown is set to 'INTERFACE'. Under 'Destination', 'Directive File' is selected. In the 'Options' section, 'mode (optional)' is set to 'ap_bus', 'depth (optional)' is empty, 'latency (optional)' is empty, and 'port (required)' is set to 'd'.

Рисунок 2.2. Directive Interface

3.2. Моделирование

```
5 INFO: [HLS 200-10] For user 'tay' on host 'tay-PiPe' (Windows NT_amd64 version 6.1) on Thu Dec 05 13:
6 INFO: [HLS 200-10] In directory 'D:/SPbPU/HLS/lab6_z1/lab6_z1/solution2/csim/build'
7 INFO: [APCC 202-3] Tmp directory is apcc_db
8 INFO: [APCC 202-1] APCC is done.
9   Compiling(apcc) ../../../../source/lab6_z1.c in debug mode
10 INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2019.2/bin/unwrapped/win64.o/apcc.exe'
11 INFO: [HLS 200-10] For user 'tay' on host 'tay-PiPe' (Windows NT_amd64 version 6.1) on Thu Dec 05 13:
12 INFO: [HLS 200-10] In directory 'D:/SPbPU/HLS/lab6_z1/lab6_z1/solution2/csim/build'
13 INFO: [APCC 202-3] Tmp directory is apcc_db
14 INFO: [APCC 202-1] APCC is done.
15   Generating csim.exe
16 Expected   Actual
17 -----
18 res[0]: 6 == d[0]: 6
19 res[1]:13 == d[1]:13
20 res[2]:21 == d[2]:21
21 res[3]:30 == d[3]:30
22 res[4]: 9 == d[4]: 9
23 -----Pass!-----
24 INFO: [SIM 1] CSim done with 0 errors.
25 INFO: [SIM 3] ***** CSIM finish *****
```

Рисунок 2.3 Результат успешного моделирования

Моделирование второго решения также прошло успешно.

3.3. Синтез

Performance Estimates

[-] Timing (ns)

[-] Summary

Clock	Target	Estimated	Uncertainty
ap_clk	6.00	5.900	0.10

[-] Latency (clock cycles)

[-] Summary

Latency		Interval		
min	max	min	max	Type
25	25	25	25	none

Рисунок 2.4. Performance estimates – summary

На рисунке 2.4. можно увидеть, что достигнутая задержка равна $5.900 + 0.1$, укладывается в заданные требования тактовой частоты.

Использование ресурсов:

Utilization Estimates					
[-] Summary					
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	60	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	65	-
Register	-	-	109	-	-
Total	0	0	109	125	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	1	0

Рисунок 2.5. Utilization estimates – summary

На рисунке 2.5. видно, что данный проект теперь займет на микросхеме 109 регистров для хранения чисел, и 125 LUT.

Interface					
Summary					
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	foo	return value
ap_rst	in	1	ap_ctrl_hs	foo	return value
ap_start	in	1	ap_ctrl_hs	foo	return value
ap_done	out	1	ap_ctrl_hs	foo	return value
ap_idle	out	1	ap_ctrl_hs	foo	return value
ap_ready	out	1	ap_ctrl_hs	foo	return value
d_req_din	out	1	ap_bus	d	pointer
d_req_full_n	in	1	ap_bus	d	pointer
d_req_write	out	1	ap_bus	d	pointer
d_rsp_empty_n	in	1	ap_bus	d	pointer
d_rsp_read	out	1	ap_bus	d	pointer
d_address	out	32	ap_bus	d	pointer
d_datain	in	32	ap_bus	d	pointer
d_dataout	out	32	ap_bus	d	pointer
d_size	out	32	ap_bus	d	pointer

Рисунок 2.6. Interface Summary.

Performance Profile					
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
foo	-	25	-	26	-
Loop 1	no	24	6	-	4

Рисунок 2.7. Performance Profile

На рисунке 2.7. видно, что задержка получения первого выходного значения составляет 7 тактов с момента старта, для остальных -25, а задержка после старта до готовности приема новых данных – 26:

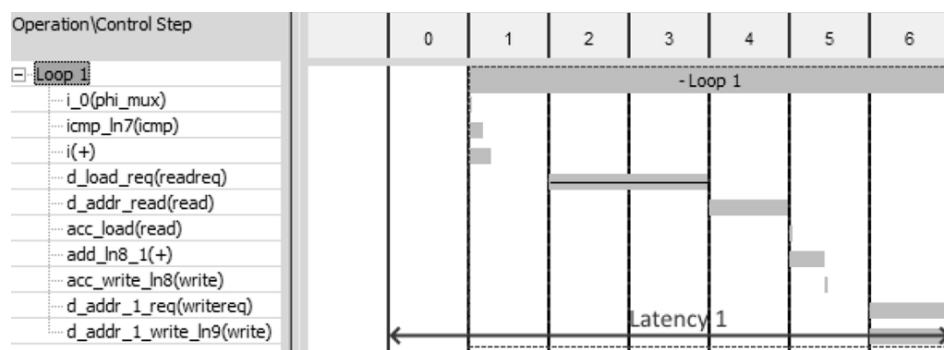


Рисунок 2.8. Schedule viewer

Последовательность работы близка к решению 1, однако, операции чтения и записи в массив d теперь выполняются отдельно от всех остальных операций, и на это тратится больше тактов.

Рассмотрим профиль ресурсов:

Performance Profile		Resource Profile									
		BRAM	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth	Words	W*Bits*Banks
▲	foo	0	0	109	125						
▶	I/O Ports(1)					32					
	Instances(0)	0	0	0	0						
	Memories(0)	0		0	0	0			0	0	0
▶	Expressions(3)	0	0	0	60	38	37	0			
▶	Registers(6)			109		109					
	Channels(0)	0		0	0	0			0	0	0
▶	Multiplexers(3)	0		0	65	36			0		
	DSP(0)		0								

Рисунок 2.9. Resource Profile

Здесь мы также видим отличия, согласно общему отчету о затраченных ресурсах.

3.4. C|RTL моделирование

При осуществлении совместного моделирования программа показала ожидаемые результаты Latency

lab6_1_main.clab6_1_test.cfoo_csim.log

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	25	25	25	NA	NA	NA

Рисунок 2.10. Отчет о моделировании

Покажем временную диаграмму совместного моделирования

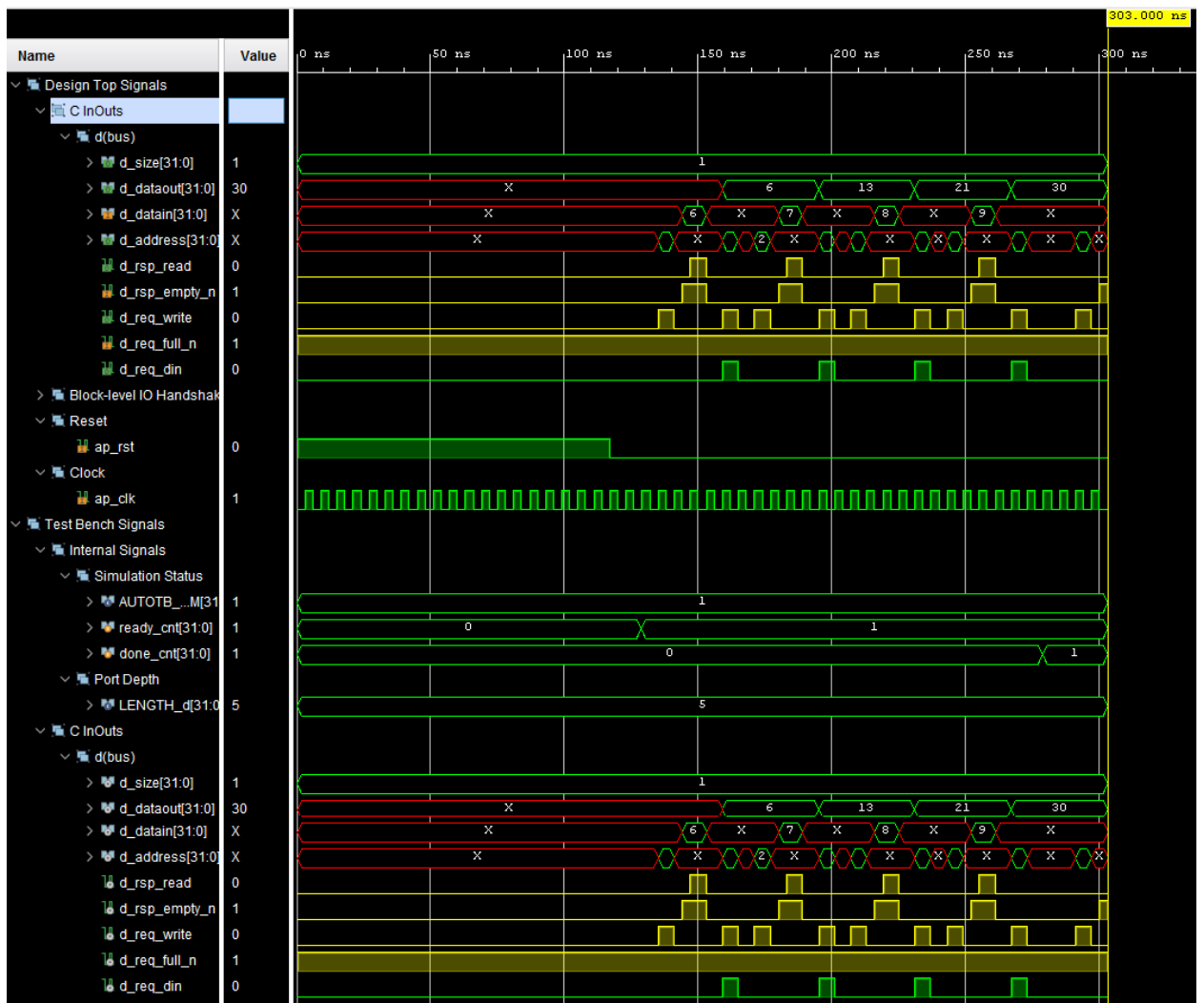


Рисунок 2.11. Design Top Signals

Здесь также видны отличия во времени выполнения итераций и протоколе работы с массивом D.

4. Выводы

В данной работе были рассмотрены различия, которые могут появиться при синтезе устройства с применением различных block-level протоколов (ap_memory и ap_bus). По умолчанию для массивов используется протокол типа тип ap_memory. Протокол типа ap_bus реализует переменные указателя и передачи по ссылке в виде шины общего назначения.

В результате получены 2 решения: первое – полный цикл выполнения 6 тактов, а максимальная задержка обработки сигнала на такте составляет 4.854 нс, и второе – полный цикл выполнения тоже 6 тактов, но задержка уже 5.900 нс и используется протокол ap_bus.