Санкт-Петербургский Политехнический Университет Петра Великого
Институт Компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Лабораторная работа 5
Предмет: Проектирование реконфигурируемых гибридных вычислительных систем
Тема: Интерфейс AXI
Задание 1

Студент: Ерниязов Т.Е.

Гр. № 3540901/81502

Преподаватель: Антонов А.П.

Санкт-Петербург
2019

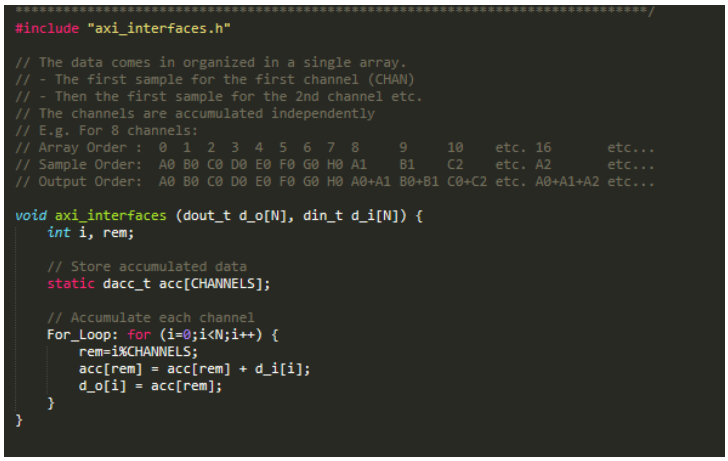
Оглавление

1. Задание

- Создать проект axi_interfaces_prj
- Подключить файл axi_interfaces.c (папка source)
- Подключить тест axi_interfaces_test.c
- Микросхема: xa7a12tcsg325-1q
- Задать: clock period 4; clock_uncertainty 0.1
- Установить директивы:
    - Порт d_o:
        - Array partition: type cyclic, factor 8, dimension 1
        - Interface: mode axis, register_mode both
    - Порт d_i:
        - Array partition: type cyclic, factor 8, dimension 1
        - Interface: mode axis, register_mode both
    - Block level
        - Unroll: factor 8
        - Pipeline: rewind
        - Interface: mode s_axilite
- 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
    - Открыть временную диаграмму (все сигналы)
        - Отобразить два цикла обработки на одном экране
            - На скриншоте показать Latency
            - На скриншоте показать Initiation Interval

Исходные файлы с кодом устройства и теста приведены ниже.

Исходный код:



```
#include "axi_interfaces.h"

// The data comes in organized in a single array.
// - The first sample for the first channel (CHAN)
// - Then the first sample for the 2nd channel etc.
// The channels are accumulated independently
// E.g. For 8 channels:
// Array Order :  0  1  2  3  4  5  6  7  8     9     10    etc. 16       etc...
// Sample Order:  A0 B0 C0 D0 E0 F0 G0 H0 A1    B1    C2    etc. A2       etc...
// Output Order:  A0 B0 C0 D0 E0 F0 G0 H0 A0+A1 B0+B1 C0+C2 etc. A0+A1+A2 etc...

void axi_interfaces (dout_t d_o[N], din_t d_i[N]) {
    int i, rem;

    // Store accumulated data
    static dacc_t acc[CHANNELS];

    // Accumulate each channel
    For_Loop: for (i=0;i<N;i++) {
        rem=i%CHANNELS;
        acc[rem] = acc[rem] + d_i[i];
        d_o[i] = acc[rem];
    }
}
```

Рис. 1.1. Исходный код

Код теста:

```
#include "axi_interfaces.h"

int main () {

    // Create input data
  din_t d_i[N] = {10, 20, 30, 40, 50, 60, 70, 80,
                                  11, 21, 31, 41, 51, 61, 71, 81,
                                  12, 22, 32, 42, 52, 62, 72, 82,
                                  13, 23, 33, 43, 53, 63, 73, 83};

  dout_t d_o[N];
    int i, retval=0;
    FILE        *fp;


    // Call the function to operate on the data
  axi_interfaces(d_o,d_i);

    // Save the results to a file
    fp=fopen("result.dat","w");
    fprintf(fp, "Din Dout\n");

    for (i=0;i<N;i++) {
    fprintf(fp, "%d   %d\n", d_i[i], d_o[i]);
    }
    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 passes
  return retval;
}
```

Рис. 1.2. Код теста

2. Скрипт для выполнения работы

Ниже приведён скрипт, который был написан для автоматизации выполнения работы.

```
open_project    -reset axi_interfaces_prj
add_files           axi_interfaces.c
add_files -tb   axi_interfaces_test.c
add_files -tb result.golden.dat
set_top     axi_interfaces

open_solution -reset solution1
set_part  {xa7a12tcsg325-1q}
create_clock -period 4

set_directive_array_partition -type cyclic -factor 8 -dim 1 "axi_interfaces" d_i
set_directive_array_partition -type cyclic -factor 8 -dim 1 "axi_interfaces" d_o
set_directive_unroll -factor 8 "axi_interfaces/For_Loop"
set_directive_interface -mode axis -register -register_mode both "axi_interfaces" d_i
set_directive_interface -mode axis -register -register_mode both "axi_interfaces" d_o
set_directive_pipeline -rewind "axi_interfaces/For_Loop"
set_directive_interface -mode s_axilite "axi_interfaces"

csim_design
csynth_design
cosim_design -trace_level all

exit
```

Рис 2.1. Скрипт выполнения работы

Ниже приведены установленные директивы, после выполнения скрипта. По рисунку видно, что директивы установлены корректно.

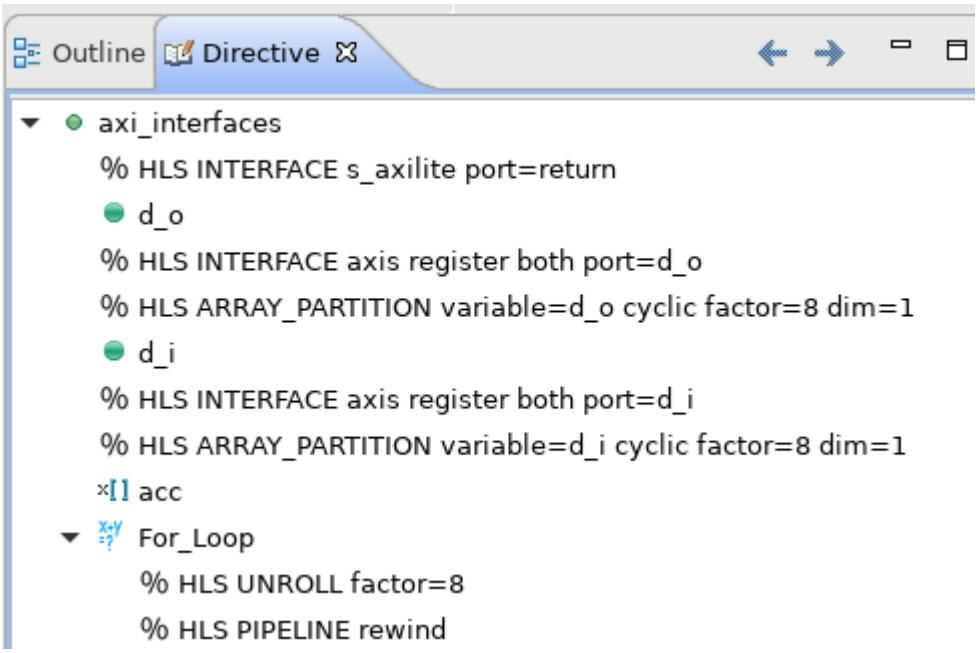


Рис. 2.2. Установленные директивы

3. Моделирование

По результатам моделирования, приведённым ниже, видно, что устройство проходит тесты.

```
INFO: [SIM 211-2] *************** CSIM start ***************
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
   Compiling(apcc) ../../../../axi_interfaces_test.c in debug mode
INFO: [HLS 200-10] Running 'D:/Xilinx/Vivado/2018.2/bin/unwrapped/win64.o/apcc.exe'
INFO: [HLS 200-10] For user 'napst' on host 'eterium' (Windows NT_amd64 version 6.2)
INFO: [HLS 200-10] In directory 'D:/10Semester/vivado/lab5_port_level_axi_interface/s
csim/build'
INFO: [APCC 202-3] Tmp directory is apcc_db
INFO: [APCC 202-1] APCC is done.
   Compiling(apcc) ../../../../axi_interfaces.c in debug mode
INFO: [HLS 200-10] Running 'D:/Xilinx/Vivado/2018.2/bin/unwrapped/win64.o/apcc.exe'
INFO: [HLS 200-10] For user 'napst' on host 'eterium' (Windows NT_amd64 version 6.2)
INFO: [HLS 200-10] In directory 'D:/10Semester/vivado/lab5_port_level_axi_interface/s
csim/build'
INFO: [APCC 202-3] Tmp directory is apcc_db
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 3.1. Результаты моделирования

4. Синтез

Ниже приведены оценки производительности. По ним видно, что оценочное время выполнения одного такта 3.3 нс, а latency составляет 5-6 тактов.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	3.319	0.50

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
5	6	4	4	loop rewind(delay=0 initiation interval(s))

Рис. 4.1. Оценка производительности

Оценка использования ресурсов показывает, что будут использованы 1579 LUT.

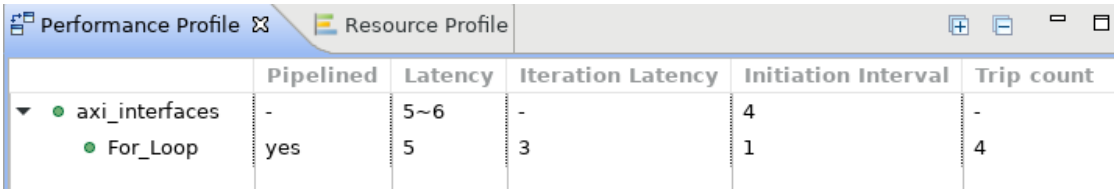
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	954
FIFO	-	-	-	-
Instance	0	-	36	40
Memory	-	-	-	-
Multiplexer	-	-	-	585
Register	-	-	848	-
Total	0	0	884	1579
Available	40	40	16000	8000
Utilization (%)	0	0	5	19

Рис. 4.2. Оценка использования ресурсов

По профилю производительности, можно сказать, что latency составляет 5-6 тактов, а II 4 такта.



	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
axi_interfaces	-	5~6	-	4	-
For_Loop	yes	5	3	1	4

Рис. 4.3. Профиль производительности

Ниже приведён список портов устройства с указанием их протокола.

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_AXILiteS_AWVALID	in	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_AWREADY	out	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_AWADDR	in	4	s_axi	AXILiteS	return void
s_axi_AXILiteS_WVALID	in	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_WREADY	out	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_WDATA	in	32	s_axi	AXILiteS	return void
s_axi_AXILiteS_WSTRB	in	4	s_axi	AXILiteS	return void
s_axi_AXILiteS_ARVALID	in	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_ARREADY	out	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_ARADDR	in	4	s_axi	AXILiteS	return void
s_axi_AXILiteS_RVALID	out	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_RREADY	in	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_RDATA	out	32	s_axi	AXILiteS	return void
s_axi_AXILiteS_RRESP	out	2	s_axi	AXILiteS	return void
s_axi_AXILiteS_BVALID	out	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_BREADY	in	1	s_axi	AXILiteS	return void
s_axi_AXILiteS_BRESP	out	2	s_axi	AXILiteS	return void
ap_clk	in	1	ap_ctrl_hs	axi_interfaces	return value
ap_rst_n	in	1	ap_ctrl_hs	axi_interfaces	return value
interrupt	out	1	ap_ctrl_hs	axi_interfaces	return value

d_o_0_TREADY	in	1	axis	d_o_0	pointer
d_o_0_TDATA	out	16	axis	d_o_0	pointer
d_o_0_TVALID	out	1	axis	d_o_0	pointer
d_o_1_TREADY	in	1	axis	d_o_1	pointer
d_o_1_TDATA	out	16	axis	d_o_1	pointer
d_o_1_TVALID	out	1	axis	d_o_1	pointer
d_o_2_TREADY	in	1	axis	d_o_2	pointer
d_o_2_TDATA	out	16	axis	d_o_2	pointer
d_o_2_TVALID	out	1	axis	d_o_2	pointer
d_o_3_TREADY	in	1	axis	d_o_3	pointer
d_o_3_TDATA	out	16	axis	d_o_3	pointer
d_o_3_TVALID	out	1	axis	d_o_3	pointer
d_o_4_TREADY	in	1	axis	d_o_4	pointer
d_o_4_TDATA	out	16	axis	d_o_4	pointer
d_o_4_TVALID	out	1	axis	d_o_4	pointer
d_o_5_TREADY	in	1	axis	d_o_5	pointer
d_o_5_TDATA	out	16	axis	d_o_5	pointer
d_o_5_TVALID	out	1	axis	d_o_5	pointer
d_o_6_TREADY	in	1	axis	d_o_6	pointer
d_o_6_TDATA	out	16	axis	d_o_6	pointer
d_o_6_TVALID	out	1	axis	d_o_6	pointer
d_o_7_TREADY	in	1	axis	d_o_7	pointer
d_o_7_TDATA	out	16	axis	d_o_7	pointer
d_o_7_TVALID	out	1	axis	d_o_7	pointer
d_i_0_TDATA	in	16	axis	d_i_0	pointer
d_i_0_TVALID	in	1	axis	d_i_0	pointer
d_i_0_TREADY	out	1	axis	d_i_0	pointer
d_i_1_TDATA	in	16	axis	d_i_1	pointer
d_i_1_TVALID	in	1	axis	d_i_1	pointer
d_i_1_TREADY	out	1	axis	d_i_1	pointer
d_i_2_TDATA	in	16	axis	d_i_2	pointer
d_i_2_TVALID	in	1	axis	d_i_2	pointer
d_i_2_TREADY	out	1	axis	d_i_2	pointer
d_i_3_TDATA	in	16	axis	d_i_3	pointer
d_i_3_TVALID	in	1	axis	d_i_3	pointer
d_i_3_TREADY	out	1	axis	d_i_3	pointer
d_i_4_TDATA	in	16	axis	d_i_4	pointer
d_i_4_TVALID	in	1	axis	d_i_4	pointer
d_i_4_TREADY	out	1	axis	d_i_4	pointer
d_i_5_TDATA	in	16	axis	d_i_5	pointer
d_i_5_TVALID	in	1	axis	d_i_5	pointer
d_i_5_TREADY	out	1	axis	d_i_5	pointer
d_i_6_TDATA	in	16	axis	d_i_6	pointer
d_i_6_TVALID	in	1	axis	d_i_6	pointer
d_i_6_TREADY	out	1	axis	d_i_6	pointer
d_i_7_TDATA	in	16	axis	d_i_7	pointer
d_i_7_TVALID	in	1	axis	d_i_7	pointer
d_i_7_TREADY	out	1	axis	d_i_7	pointer

Рис. 4.4. Список портов

Ниже приводится таблица использования ресурсов на каждом шаге выполнения.

	Resource\Control Step	C0	C1	C2	C3	C4
1	⊟I/O Ports					
2	d_i_1		read			
3	d_i_6		read			
4	d_i_0		read			
5	d_i_2		read			
6	d_i_4		read			
7	d_i_5		read			
8	d_i_3		read			
9	d_i_7		read			
10	d_o_0				write	
11	d_o_2				write	
12	d_o_1				write	
13	d_o_4				write	
14	d_o_7				write	
15	d_o_5				write	
16	d_o_3				write	
17	d_o_6				write	
18	⊟Expressions					
19	i_1_7_fu_236		+			
20	i1_phi_fu_222		phi_mux			
21	do_init_phi_fu_206		phi_mux			
22	exitcond_fu_246		icmp			
23	StgValue_72_fu_252			return		
24	tmp_4_fu_278				+	
25	tmp_4_1_fu_309				+	
26	tmp_4_6_fu_464				+	
27	tmp_3_2_fu_328				+	
28	tmp_3_3_fu_359				+	
29	tmp_3_7_fu_483				+	
30	tmp_4_4_fu_402				+	
31	tmp_3_1_fu_297				+	
32	tmp_4_7_fu_495				+	
33	tmp_3_fu_266				+	
34	tmp_4_3_fu_371				+	
35	tmp_3_6_fu_452				+	
36	tmp_4_2_fu_340				+	
37	tmp_4_5_fu_433				+	
38	tmp_3_4_fu_390				+	
39	tmp_3_5_fu_421				+	

Рис. 4.5. Использование ресурсов

Ниже приведён результат работы планировщика вычислений.

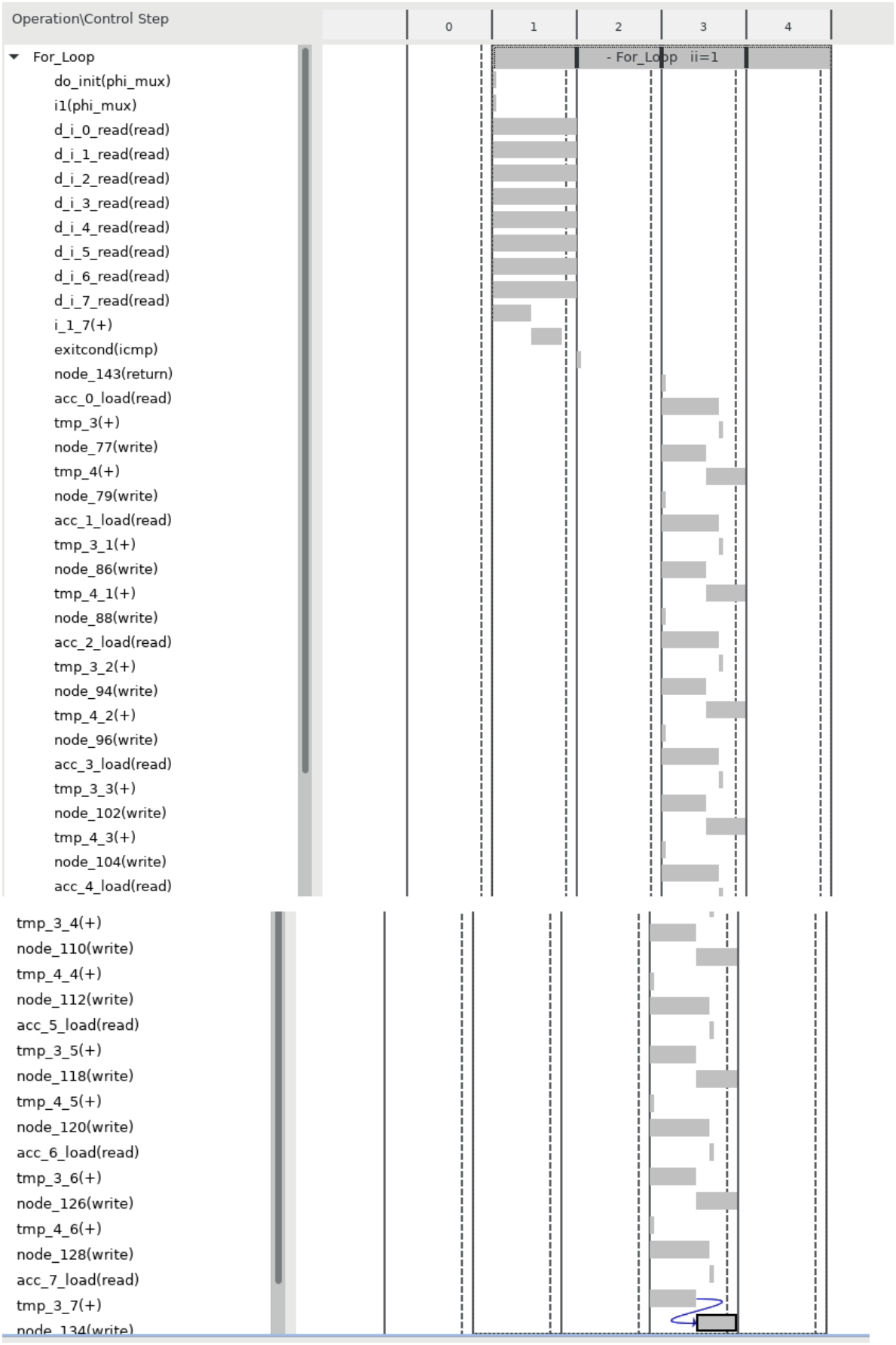


Рис. 4.6. Планировщик вычислений

5. C|RTL моделирование

Результат C|RTL моделирования приведён ниже.

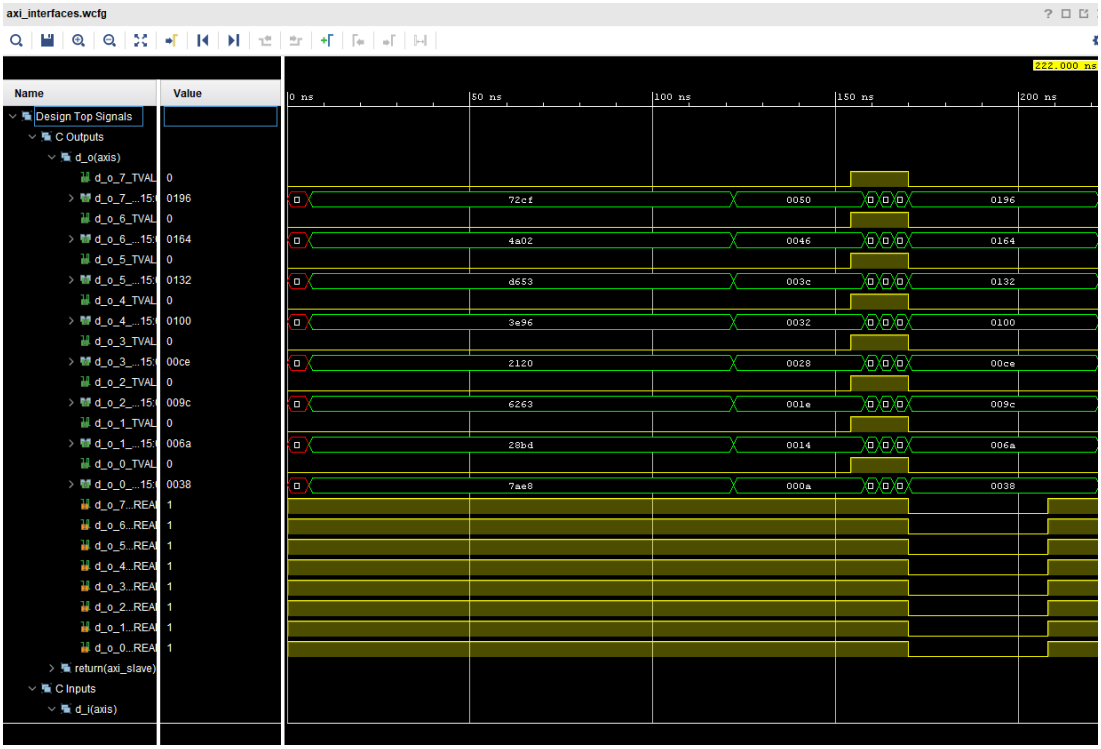


Рис. 5.1. Результат моделирования

6. Выводы

В данной работе был получен опыт использования AXI Interface, а также были рассмотрены несколько директив, позволяющих выполнять оптимизацию проекта.