

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

Лабораторная работа 10

Предмет: Проектирование реконфигурируемых гибридных вычислительных систем

Тема: Упаковка данных

Задание 1

Студент: Ерниязов Т.Е.
Гр. № 3540901/81502

Преподаватель: Антонов А.П.

Санкт-Петербург
2019

Оглавление

1. Задание	4
2. Моделирование	7
3. Первое решение	7
3.1. Синтез	7
3.2. C/RTL моделирование.....	10
4. Второе решение.....	11
4.1. Синтез	11
4.2. C\RTL моделирование.....	14
5. Третье решение	15
5.1. Синтез	16
5.2. C\RTL моделирование.....	19
6. Четвёртое решение	20
6.1. Синтез	21
6.2. C\RTL моделирование.....	24
7. Выводы.....	25

1. Задание

- Создать проект `lab10_z1`
- Микросхема: `ха7a12tcsг325-1q`
- Познакомиться с исходным кодом `struct_port.c`
- Познакомиться с исходным кодом `struct_port_test.c` для проверки функции. Осуществить моделирование (с выводом результатов в консоль)
- Исследование:
 - `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 (интерес представляет количество и тип портов)
 - `Solution_2a`
 - задать: `clock period 10; clock_uncertainty 0.1`
 - установить реализацию `DATA_PACK`
 - 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 (интерес представляет количество и тип портов)
 - Сравнить два решения (`solution_1a` и `solution_2a`) и сделать выводы: зависимость от `DATA_PACK`; объяснить (посчитать) число циклов Latency, П...
 - `Solution_3a`
 - задать: `clock period 10; clock_uncertainty 0.1`
 - установить реализацию `DATA_PACK` with `struct_level`

- 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 (интерес представляет количество и тип портов)
- Сравнить два решения (solution_2a и solution_3a) и сделать выводы: зависимость от типа интерфейса; объяснить (посчитать) число циклов Latency, П...
- Solution_4a
 - задать: clock period 10; clock_uncertainty 0.1
 - установить реализацию DATA_PACK with field_level
 - 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 (интерес представляет количество и тип портов)
 - Сравнить два решения (solution_3a и solution_4a) и сделать выводы: зависимость от типа интерфейса; объяснить (посчитать) число циклов Latency, П...

Исходный текст подготовленной для синтеза функции и теста приведён ниже:

```
92 #include "struct_port.h"
93
94 data_t struct_port(data_t i_val, data_t *i_pt, data_t *o_pt) {
95     data_t o_val;
96     int i;
97
98     // Transfer pass-by-value structs
99     o_val.A = i_val.A+2;
100     for (i=0;i<4;i++) {
101         o_val.B[i] = i_val.B[i]+2;
102     }
103
104     // Transfer pointer structs
105     o_pt->A = i_pt->A+3;
106     for (i=0;i<4;i++) {
107         o_pt->B[i] = i_pt->B[i]+3;
108     }
109
110     return o_val;
111 }
```

Рис. 1.1. Исходный код синтезируемой функции

```
92 #ifndef _STRUCT_PORT_H_
93 #define _STRUCT_PORT_H_
94
95 #include <stdio.h>
96
97 typedef struct {
98     unsigned short A;
99     unsigned char B[4];
100 } data_t;
101
102 data_t struct_port(data_t i_val, data_t *i_pt, data_t *o_pt);
103
104 #endif
```

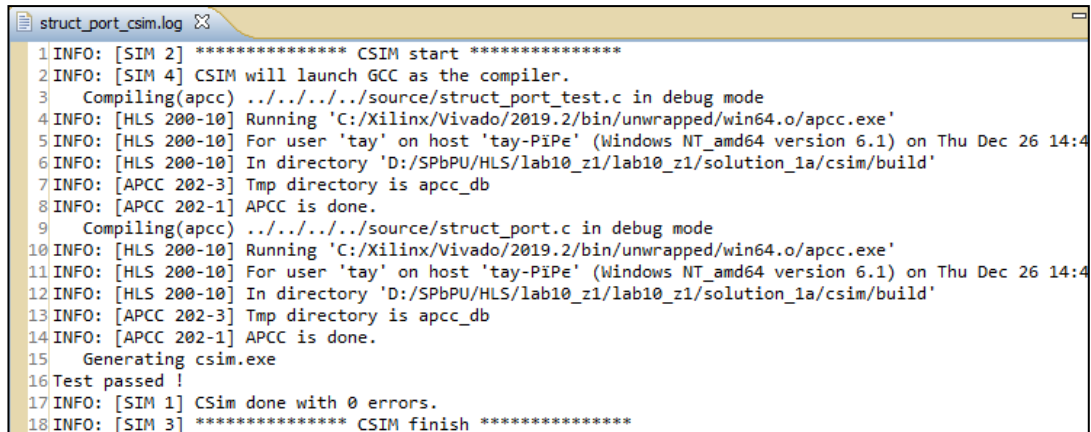
Рис. 1.2. Заголовочный файл

```
92 #include "struct_port.h"
93
94 int main () {
95     data_t d_ival, d_ipt;
96     data_t d_oval, d_opt;
97
98     int i, retval=0;
99     FILE *fp;
100
101     // Create input data
102     d_ival.A = 19;
103     d_ipt.A = 29;
104     for (i=0;i<4;i++) {
105         d_ival.B[i] = i+10;
106         d_ipt.B[i] = i+20;
107     }
108
109     // Call the function to operate on the data
110     d_oval = struct_port(d_ival, &d_ipt, &d_opt);
111
112     // Save the results to a file
113     fp=fopen("result.dat","w");
114     fprintf(fp, "Din Dout\n");
115
116     fprintf(fp, "%d %d\n", d_oval.A, d_opt.A);
117     for (i=0;i<4;i++) {
118         fprintf(fp, "%d %d\n", d_oval.B[i], d_opt.B[i]);
119     }
120     fclose(fp);
121
122     // Compare the results file with the golden results
123     retval = system("diff --brief -w result.dat result.golden.dat");
124     if (retval != 0) {
125         printf("Test failed !!!\n");
126         retval=1;
127     } else {
128         printf("Test passed !\n");
129     }
130
131     // Return 0 if the test passed
132     return retval;
133 }
```

Рис. 1.3. Исходный код теста

2. Моделирование

При запуске моделирования можно увидеть, что тест успешно пройден:



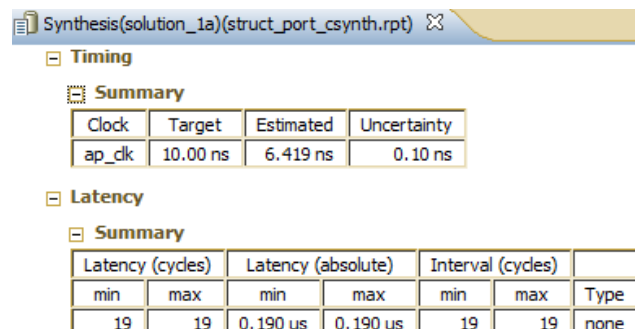
```
1 INFO: [SIM 2] ***** CSIM start *****
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3   Compiling(apcc) ../../../../source/struct_port_test.c in debug mode
4 INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2019.2/bin/unwrapped/win64.o/apcc.exe'
5 INFO: [HLS 200-10] For user 'tay' on host 'tay-PiPe' (Windows NT_amd64 version 6.1) on Thu Dec 26 14:4
6 INFO: [HLS 200-10] In directory 'D:/SPbPU/HLS/lab10_z1/lab10_z1/solution_1a/csim/build'
7 INFO: [APCC 202-3] Tmp directory is apcc_db
8 INFO: [APCC 202-1] APCC is done.
9   Compiling(apcc) ../../../../source/struct_port.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 26 14:4
12 INFO: [HLS 200-10] In directory 'D:/SPbPU/HLS/lab10_z1/lab10_z1/solution_1a/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 Test passed !
17 INFO: [SIM 1] CSim done with 0 errors.
18 INFO: [SIM 3] ***** CSIM finish *****
```

Рис. 2.1. Результаты моделирования

3. Первое решение

3.1. Синтез

Приведем в отчете требуемые данные о проекте:



Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.419 ns	0.10 ns

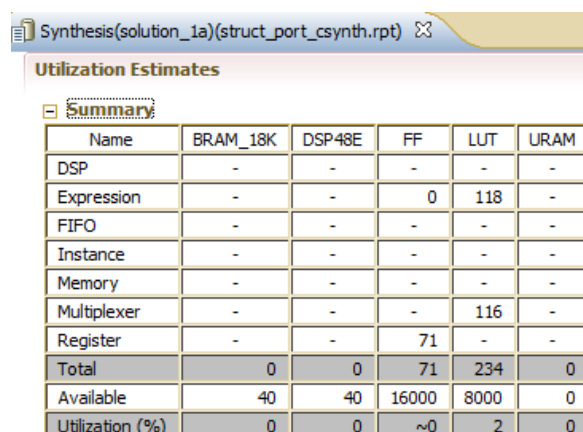
Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
19	19	0.190 us	0.190 us	19	19	none

Рис. 3.1. Производительность

Здесь можно увидеть, что достигнутая задержка равна $6.419 + 0.1$, что укладывается в заданные нами требования к тактовой частоте.



Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	118	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	116	-
Register	-	-	71	-	-
Total	0	0	71	234	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	2	0

Рис. 3.2. Занимаемые ресурсы

Synthesis(solution_1a)(struct_port_csynth.rpt) ⌕

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	struct_port	return value
ap_rst	in	1	ap_ctrl_hs	struct_port	return value
ap_start	in	1	ap_ctrl_hs	struct_port	return value
ap_done	out	1	ap_ctrl_hs	struct_port	return value
ap_idle	out	1	ap_ctrl_hs	struct_port	return value
ap_ready	out	1	ap_ctrl_hs	struct_port	return value
agg_result_A	out	16	ap_vid	agg_result_A	pointer
agg_result_A_ap_vid	out	1	ap_vid	agg_result_A	pointer
agg_result_B_address0	out	2	ap_memory	agg_result_B	array
agg_result_B_ce0	out	1	ap_memory	agg_result_B	array
agg_result_B_we0	out	1	ap_memory	agg_result_B	array
agg_result_B_d0	out	8	ap_memory	agg_result_B	array
agg_result_B_address1	out	2	ap_memory	agg_result_B	array
agg_result_B_ce1	out	1	ap_memory	agg_result_B	array
agg_result_B_we1	out	1	ap_memory	agg_result_B	array
agg_result_B_d1	out	8	ap_memory	agg_result_B	array
i_val_A	in	16	ap_none	i_val_A	scalar
i_val_B_address0	out	2	ap_memory	i_val_B	array
i_val_B_ce0	out	1	ap_memory	i_val_B	array
i_val_B_q0	in	8	ap_memory	i_val_B	array
i_pt_A	in	16	ap_none	i_pt_A	pointer
i_pt_B_address0	out	2	ap_memory	i_pt_B	array
i_pt_B_ce0	out	1	ap_memory	i_pt_B	array
i_pt_B_q0	in	8	ap_memory	i_pt_B	array
o_pt_A	out	16	ap_vid	o_pt_A	pointer
o_pt_A_ap_vid	out	1	ap_vid	o_pt_A	pointer
o_pt_B_address0	out	2	ap_memory	o_pt_B	array
o_pt_B_ce0	out	1	ap_memory	o_pt_B	array
o_pt_B_d0	out	1	ap_memory	o_pt_B	array
o_pt_B_d0	out	8	ap_memory	o_pt_B	array

Рис. 3.3. Применяемые интерфейсы

Performance Profile ⌕ Resource Profile

	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
struct_port	-	19	-	20	-
Loop 1	no	8	2	-	4
Loop 2	no	8	2	-	4

Рис. 3.4. Профиль производительности

На этом рисунке видно, что задержка получения первого выходного значения составляет 8 тактов с момента старта (всех данных – 19), а задержка после старта до готовности приема новых данных – 20:

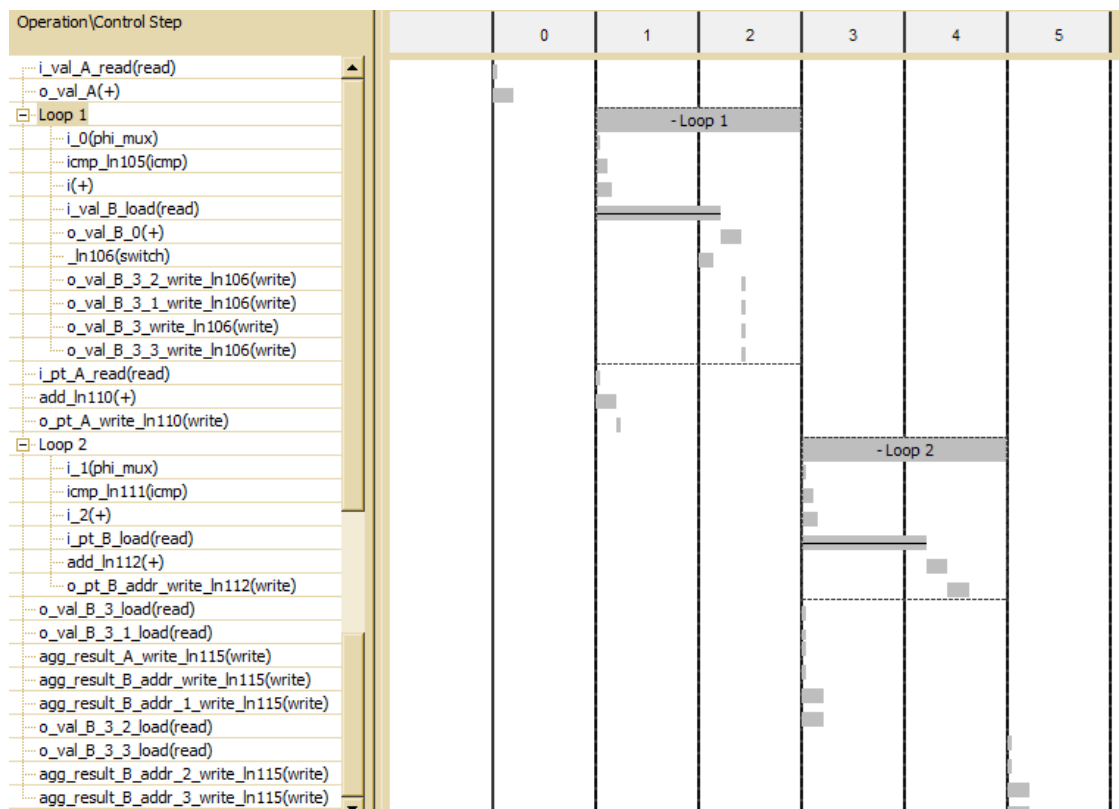


Рис. 3.5. Временная диаграмма

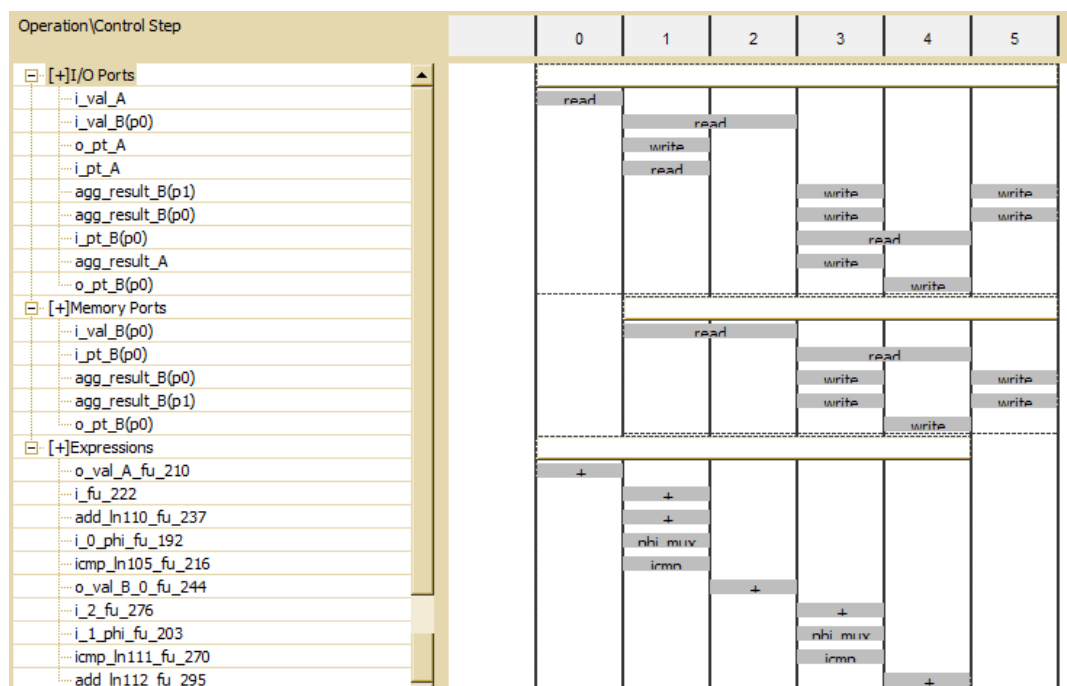


Рис. 3.6. Диаграмма использования ресурсов

	B...	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth
struct_port	0	0	71	234				
I/O Ports(8)					96			
Instances(0)	0	0	0	0				
Memories(0)	0		0	0	0			0
Expressions(8)	0	0	0	118	54	24	0	
Registers(12)			71		132			
Channels(0)	0		0	0	0			0
Multiplexers(7)	0		0	116	27			0
DSP(0)		0						

Рис. 3.7. Профиль ресурсов

Здесь можно увидеть те же числа, что и в отчете синтезатора.

3.2. C/RTL моделирование

При совместном моделировании, программа отобразила те же самые, ожидаемые нами значения Latency и II:

Cosimulation Report for 'struct_port'							
Result							
		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	19	19	19	NA	NA	NA

Рис.3.8. Результаты C\RTL моделирования

Покажем временную диаграмму совместного моделирования с отмеченными на ней Latency и Initiation Interval:

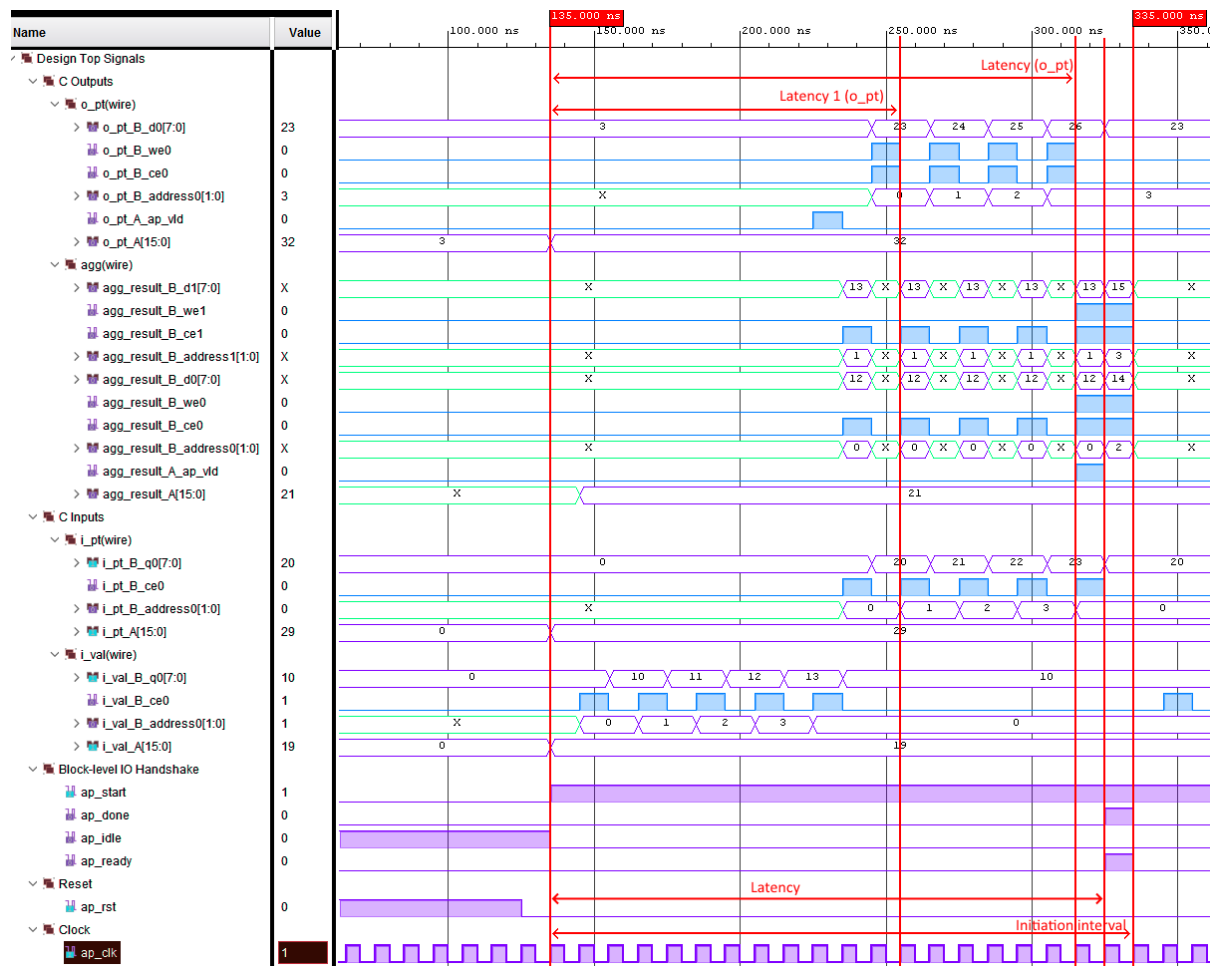


Рис. 3.9. Временная диаграмма совместного моделирования

4. Второе решение

Добавим директиву, которая изменяет способ передачи данных.

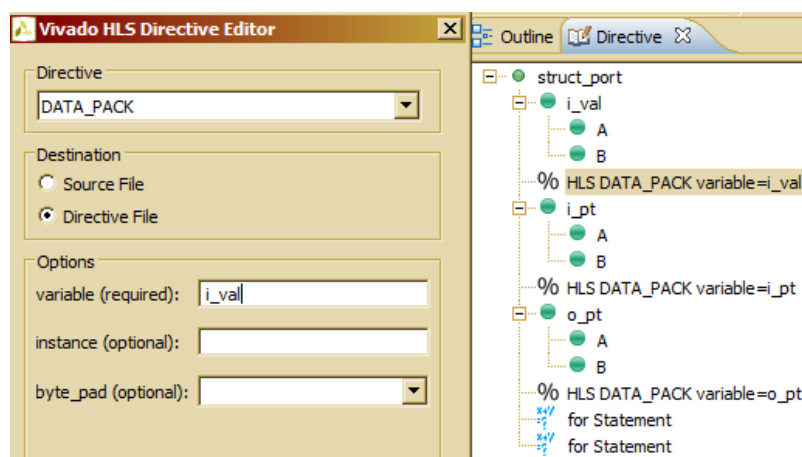


Рис. 4.1. Добавление директивы

4.1. Синтез

Приведем в отчете требуемые данные о проекте:

Synthesis(solution_2a)(struct_port_csynth.rpt) ✕

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.331 ns	0.10 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
6	6	60.000 ns	60.000 ns	6	6	none

Рис. 4.2. Производительность

Здесь можно увидеть, что достигнутая задержка равна $8.331 + 0.1$, что укладывается в заданные нами требования к тактовой частоте.

Synthesis(solution_2a)(struct_port_csynth.rpt) ✕

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	487	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	54	-	-
Total	0	0	54	577	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	7	0

Рис. 4.3. Затрачиваемые ресурсы

Synthesis(solution_2a)(struct_port_csynth.rpt) ✕

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	struct_port	return value
ap_rst	in	1	ap_ctrl_hs	struct_port	return value
ap_start	in	1	ap_ctrl_hs	struct_port	return value
ap_done	out	1	ap_ctrl_hs	struct_port	return value
ap_idle	out	1	ap_ctrl_hs	struct_port	return value
ap_ready	out	1	ap_ctrl_hs	struct_port	return value
agg_result_A	out	16	ap_vld	agg_result_A	pointer
agg_result_A_ap_vld	out	1	ap_vld	agg_result_A	pointer
agg_result_B_address0	out	2	ap_memory	agg_result_B	array
agg_result_B_ce0	out	1	ap_memory	agg_result_B	array
agg_result_B_we0	out	1	ap_memory	agg_result_B	array
agg_result_B_d0	out	8	ap_memory	agg_result_B	array
agg_result_B_address1	out	2	ap_memory	agg_result_B	array
agg_result_B_ce1	out	1	ap_memory	agg_result_B	array
agg_result_B_we1	out	1	ap_memory	agg_result_B	array
agg_result_B_d1	out	8	ap_memory	agg_result_B	array
i_val	in	48	ap_none	i_val	scalar
i_pt	in	48	ap_none	i_pt	pointer
o_pt	out	48	ap_vld	o_pt	pointer
o_pt_ap_vld	out	1	ap_vld	o_pt	pointer

Рис. 4.4. Применяемые интерфейсы

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
struct_port	-	6	-	7	-
Loop 1	no	4	1	-	4

Рис. 4.5. Профиль производительности

На этом рисунке видно, что задержка получения первого выходного значения составляет 4 такта с момента старта (6 для всех), а задержка после старта до готовности приема новых данных – 7:

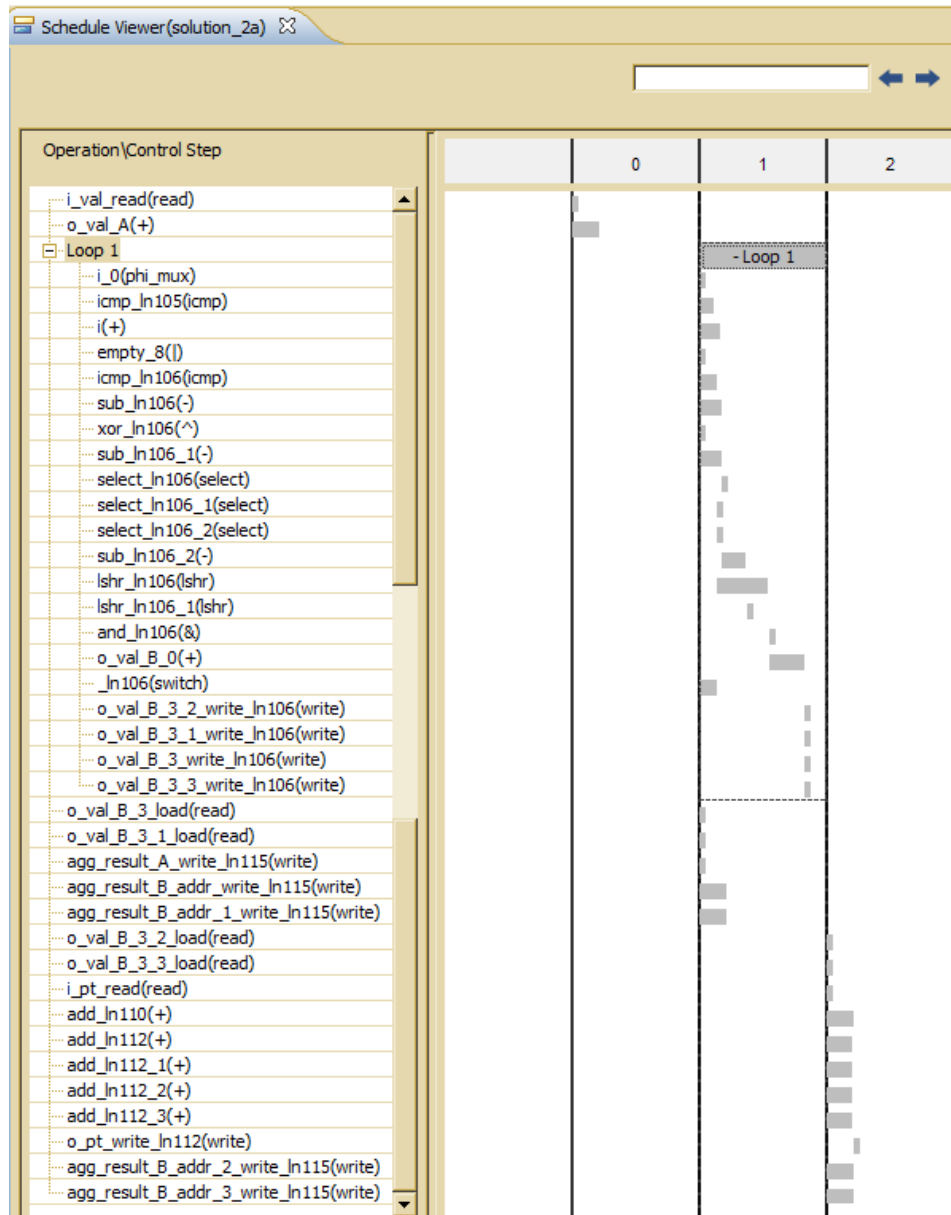


Рис. 4.6. Временная диаграмма

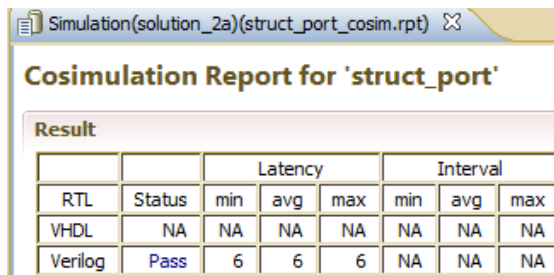


Рис. 4.9. C/RTL моделирование

Покажем временную диаграмму совместного моделирования с отмеченными на ней Latency и Initiation Interval:

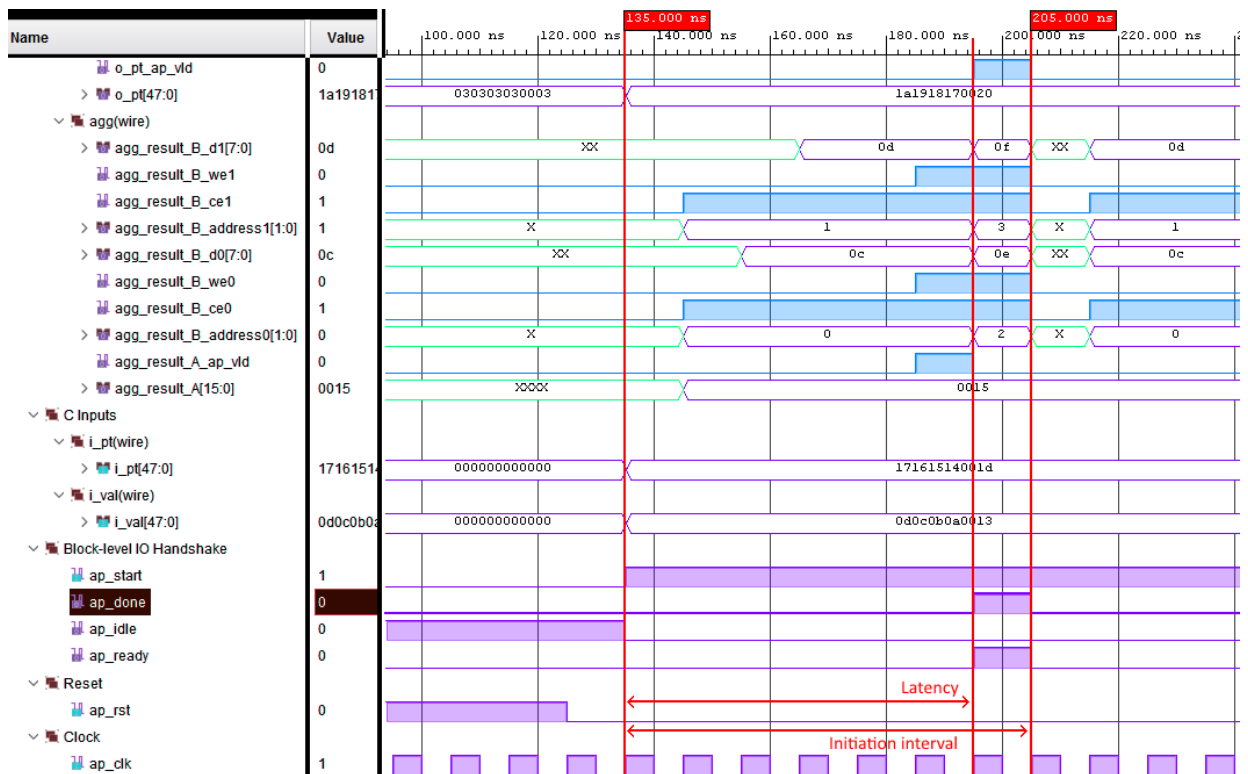


Рис. 4.10. Временная диаграмма совместного моделирования

Здесь также видны отличия во времени выполнения итераций и протоколе работы. Видно, директива была успешно применена к аргументам функции `сделав` из них один порт шириной 48 бит, это позволило получить одновременный доступ ко всем элементам структуры. Однако, структуру `o_val` развернуть не удалось т.к. не удаётся применить директиву к «return», вследствие чего не удалось выполнить распараллеливание первый цикл.

5. Третье решение

Добавим директиву, которая изменяет способ передачи данных.

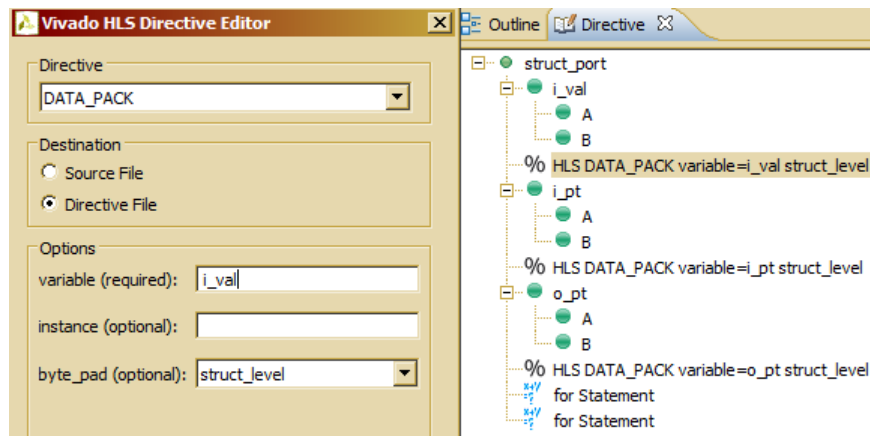


Рис. 5.1. Добавление директивы

5.1. Синтез

Приведем в отчете требуемые данные о проекте:

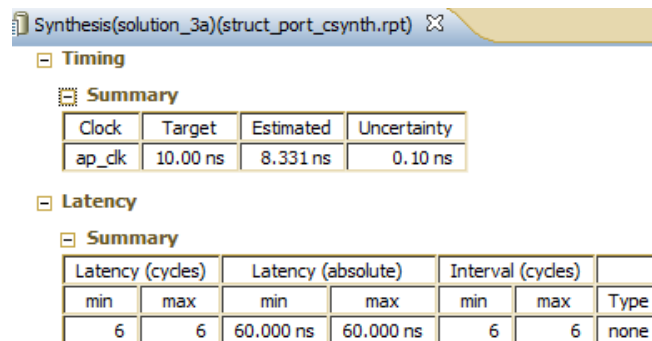


Рис. 5.2. Производительность

Здесь можно увидеть, что достигнутая задержка равна $8.331 + 0.1$, что укладывается в заданные нами требования к тактовой частоте.

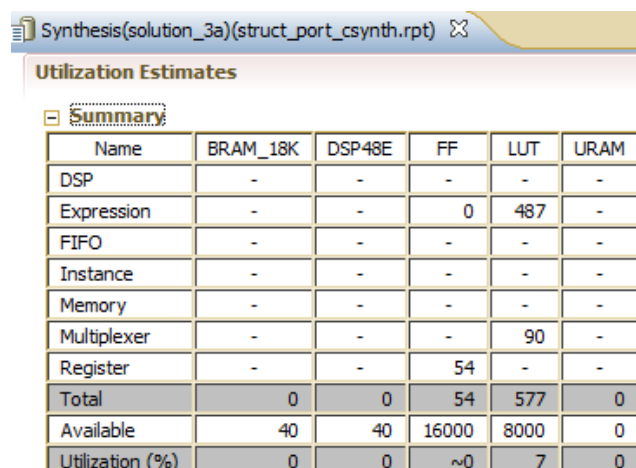


Рис. 5.3. Затрачиваемые ресурсы

Synthesis(solution_3a)(struct_port_csynth.rpt) ⌵

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	struct_port	return value
ap_rst	in	1	ap_ctrl_hs	struct_port	return value
ap_start	in	1	ap_ctrl_hs	struct_port	return value
ap_done	out	1	ap_ctrl_hs	struct_port	return value
ap_idle	out	1	ap_ctrl_hs	struct_port	return value
ap_ready	out	1	ap_ctrl_hs	struct_port	return value
agg_result_A	out	16	ap_vld	agg_result_A	pointer
agg_result_A_ap_vld	out	1	ap_vld	agg_result_A	pointer
agg_result_B_address0	out	2	ap_memory	agg_result_B	array
agg_result_B_ce0	out	1	ap_memory	agg_result_B	array
agg_result_B_we0	out	1	ap_memory	agg_result_B	array
agg_result_B_d0	out	8	ap_memory	agg_result_B	array
agg_result_B_address1	out	2	ap_memory	agg_result_B	array
agg_result_B_ce1	out	1	ap_memory	agg_result_B	array
agg_result_B_we1	out	1	ap_memory	agg_result_B	array
agg_result_B_d1	out	8	ap_memory	agg_result_B	array
i_val	in	48	ap_none	i_val	scalar
i_pt	in	48	ap_none	i_pt	pointer
o_pt	out	48	ap_vld	o_pt	pointer
o_pt_ap_vld	out	1	ap_vld	o_pt	pointer

Рис. 5.4. Применяемые интерфейсы

Performance Profile ⌵ Resource Profile

	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
struct_port	-	6	-	7	-
Loop 1	no	4	1	-	4

Рис. 5.5. Профиль производительности

На этом рисунке видно, что задержка получения первого выходного значения составляет 4 такта с момента старта (6 для всех), а задержка после старта до готовности приема новых данных – 7:

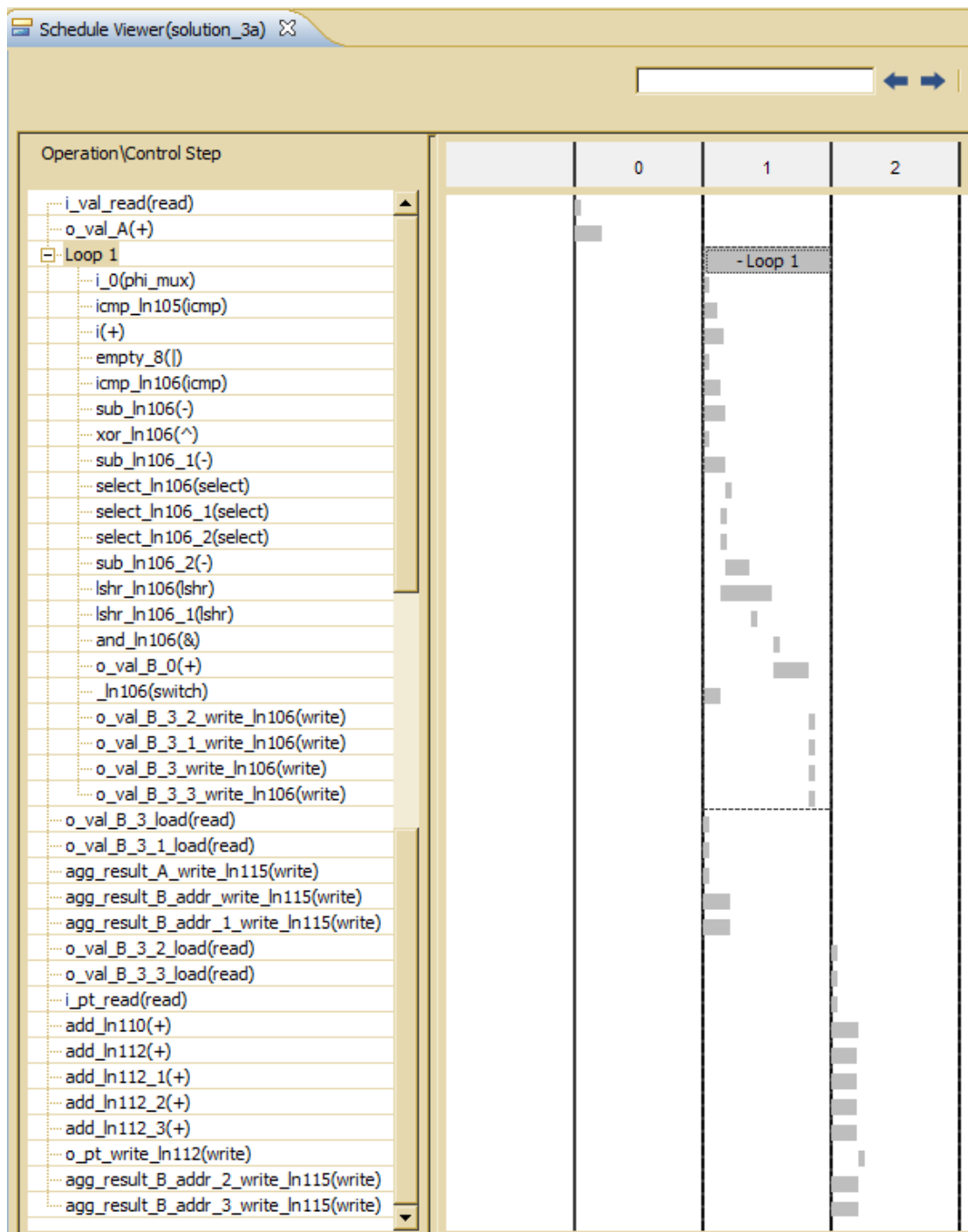


Рис. 5.6. Временная диаграмма

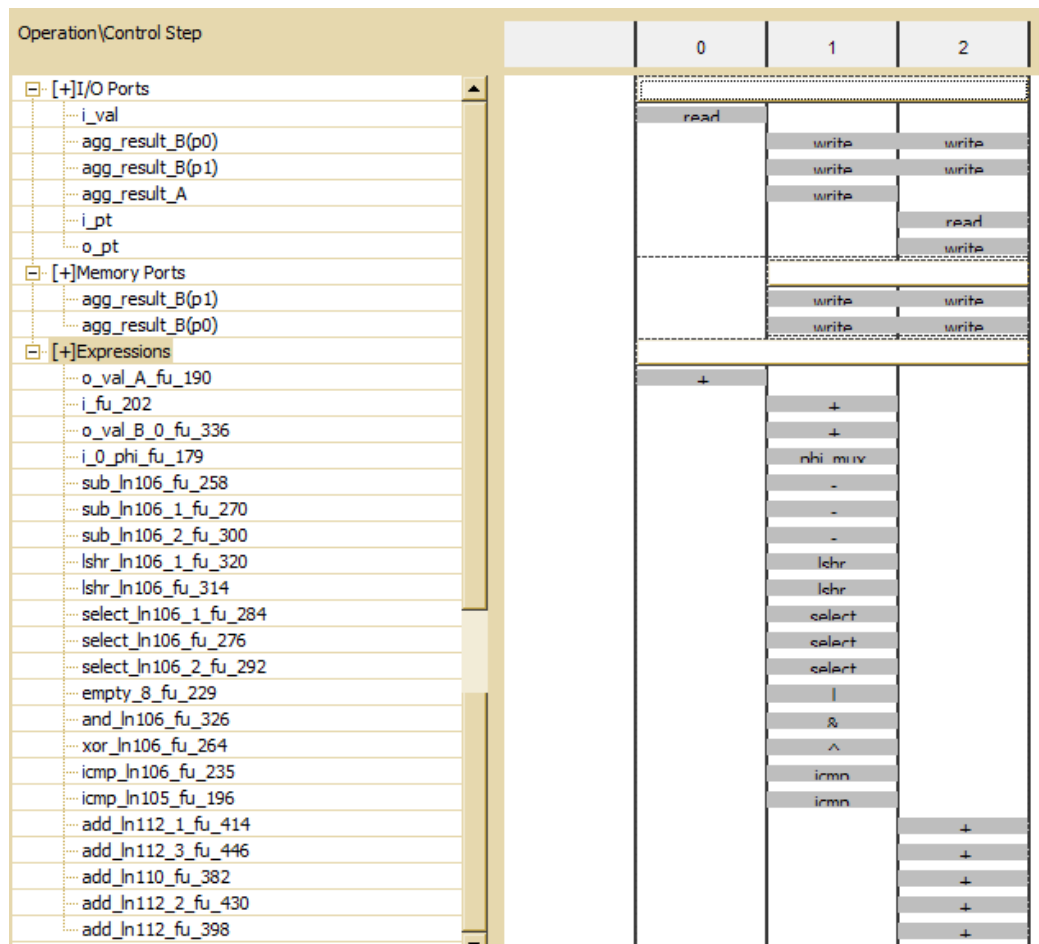


Рис. 5.7. Диаграмма использования ресурсов

Наконец покажем профиль ресурсов:


Performance Profile		Resource Profile						
	B...	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth
struct_port	0	0	54	577				
I/O Ports(5)					168			
Instances(0)	0	0	0	0				
Memories(0)	0		0	0	0			0
Expressions(21)	0	0	0	487	122	248	44	
Registers(7)			54		54			
Channels(0)	0		0	0	0			0
Multiplexers(6)	0		0	90	24			0
DSP(0)		0						

Рис. 5.8. Профиль ресурсов

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

5.2. C\RTL моделирование

При совместном моделировании, программа отобразила те же самые, ожидаемые нами значения Latency и П:

Simulation(solution_3a)(struct_port_cosim.rpt) 

Cosimulation Report for 'struct_port'

Result							
		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	6	6	6	NA	NA	NA

Рис. 5.9. C/RTL моделирование

Покажем временную диаграмму совместного моделирования с отмеченными на ней Latency и Initiation Interval:

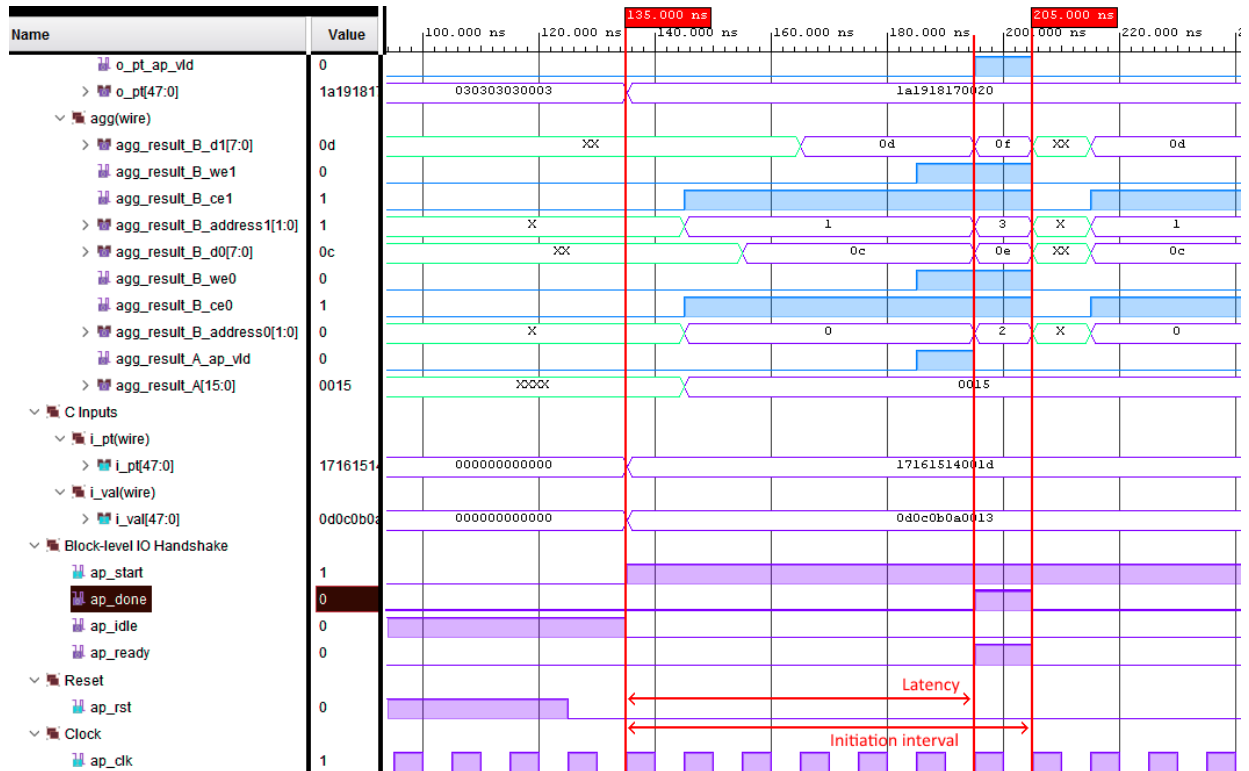


Рис. 5.10. Временная диаграмма совместного моделирования

Результаты полученного решения совпадает с предыдущим.

6. Четвёртое решение

Добавим директиву, которая изменяет способ передачи данных.

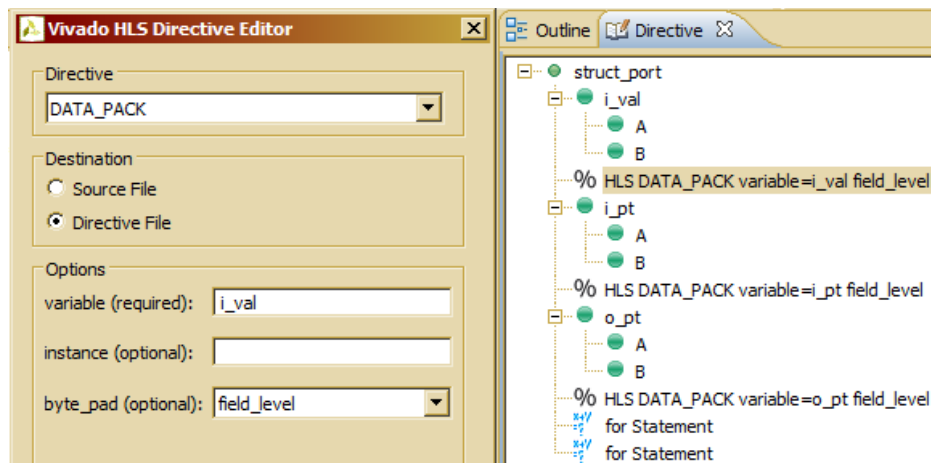


Рис. 6.1. Добавление директивы

6.1. Синтез

Приведем в отчете требуемые данные о проекте:

Synthesis(solution_4a)(struct_port_csynth.rpt)

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.331 ns	0.10 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
6	6	60.000 ns	60.000 ns	6	6	none

Рис. 6.2. Производительность

Здесь можно увидеть, что достигнутая задержка равна $8.331 + 0.1$, что укладывается в заданные нами требования к тактовой частоте.

Synthesis(solution_4a)(struct_port_csynth.rpt)

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	487	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	54	-	-
Total	0	0	54	577	0
Available	40	40	16000	8000	0
Utilization (%)	0	0	~0	7	0

Рис. 6.3. Затрачиваемые ресурсы

Synthesis(solution_4a)(struct_port_csynth.rpt) ⌕

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	struct_port	return value
ap_rst	in	1	ap_ctrl_hs	struct_port	return value
ap_start	in	1	ap_ctrl_hs	struct_port	return value
ap_done	out	1	ap_ctrl_hs	struct_port	return value
ap_idle	out	1	ap_ctrl_hs	struct_port	return value
ap_ready	out	1	ap_ctrl_hs	struct_port	return value
agg_result_A	out	16	ap_vld	agg_result_A	pointer
agg_result_A_ap_vld	out	1	ap_vld	agg_result_A	pointer
agg_result_B_address0	out	2	ap_memory	agg_result_B	array
agg_result_B_ce0	out	1	ap_memory	agg_result_B	array
agg_result_B_we0	out	1	ap_memory	agg_result_B	array
agg_result_B_d0	out	8	ap_memory	agg_result_B	array
agg_result_B_address1	out	2	ap_memory	agg_result_B	array
agg_result_B_ce1	out	1	ap_memory	agg_result_B	array
agg_result_B_we1	out	1	ap_memory	agg_result_B	array
agg_result_B_d1	out	8	ap_memory	agg_result_B	array
i_val	in	48	ap_none	i_val	scalar
i_pt	in	48	ap_none	i_pt	pointer
o_pt	out	48	ap_vld	o_pt	pointer
o_pt_ap_vld	out	1	ap_vld	o_pt	pointer

Рис. 6.4. Применяемые интерфейсы

Performance Profile ⌕ Resource Profile

	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
struct_port	-	6	-	7	-
Loop 1	no	4	1	-	4

Рис. 6.5. Профиль производительности

Заметно сходство с результатами предыдущего решения.

На этом рисунке видно, что задержка получения первого выходного значения составляет 4 такта с момента старта (6 для всех), а задержка после старта до готовности приема новых данных – 7:

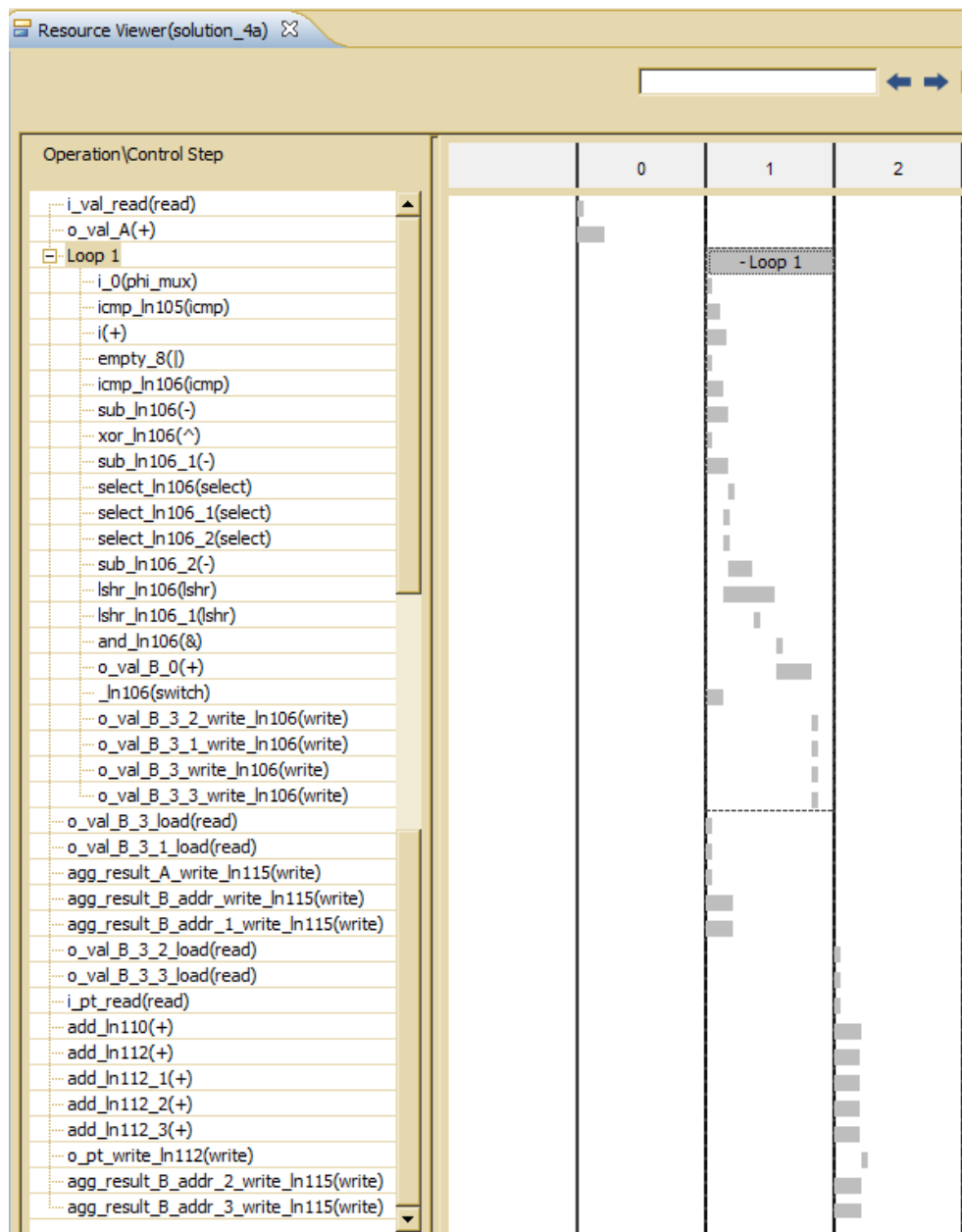


Рис. 6.6. Временная диаграмма

Simulation(solution_4a)(struct_port_cosim.rpt) X

Cosimulation Report for 'struct_port'

Result

		Latency			Interval			
	RTL	Status	min	avg	max	min	avg	max
VHDL		NA	NA	NA	NA	NA	NA	NA
Verilog		Pass	6	6	6	NA	NA	NA

Рис. 6.9. C/RTL моделирование

Покажем временную диаграмму совместного моделирования с отмеченными на ней Latency и Initiation Interval:

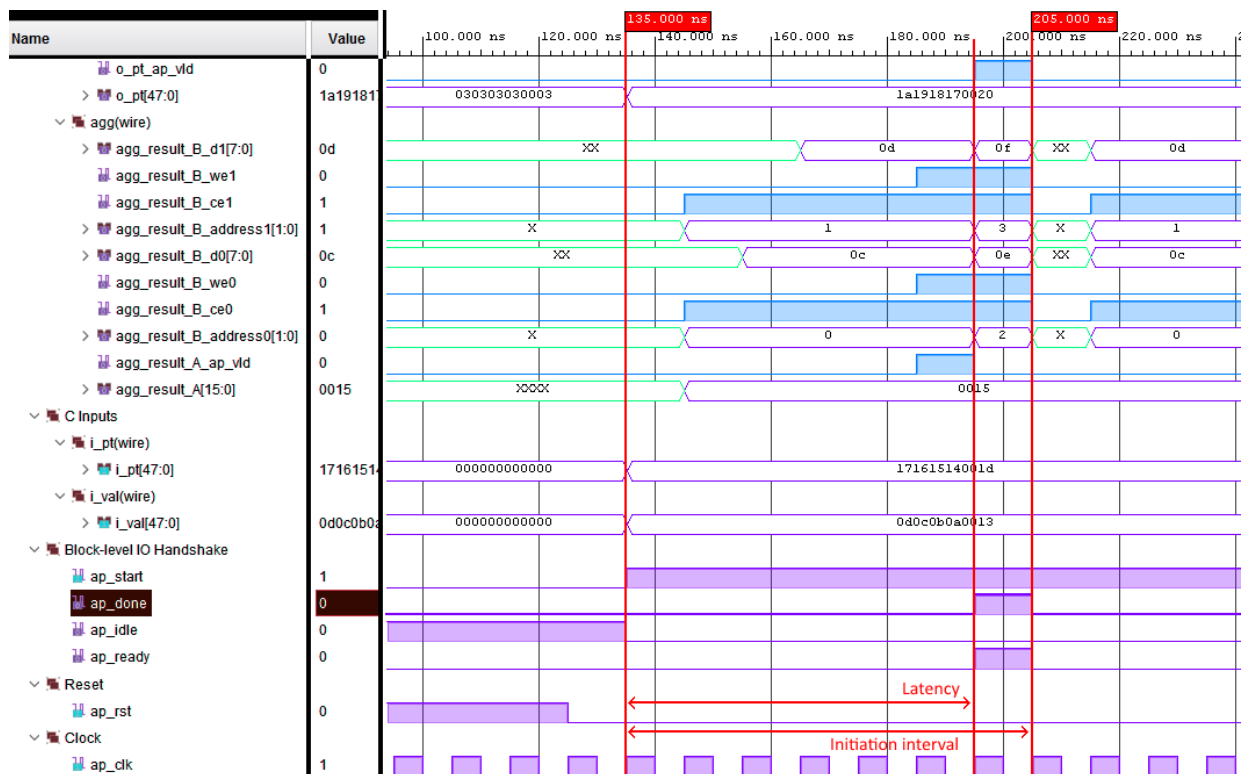


Рис. 6.10. Временная диаграмма совместного моделирования

Результаты полученного решения совпадает с предыдущим.

7. Выводы

В данной работе было проведено исследование влияния директивы DATA_PACK на функции, оперирующие со структурами данных. Данная директива позволяет развернуть структуру в один порт соответствующей ширины, однако это требует большего количества ресурсов. Изменения параметра byte_rad в данной работе никак не сказалось на результате, вследствие, вероятно, простоты синтезируемой функции.