Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе №4\_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**

[Задание 3](#_Toc23434767)

[Решение №1 5](#_Toc23434768)

[Моделирование 5](#_Toc23434769)

[Синтез 6](#_Toc23434770)

[C/RTL моделирование 9](#_Toc23434771)

[Решение №2 11](#_Toc23434772)

[Добавление протокола: 11](#_Toc23434773)

[Моделирование 11](#_Toc23434774)

[Синтез 12](#_Toc23434775)

[C/RTL моделирование 14](#_Toc23434776)

[Выводы 15](#_Toc23434777)

# **Задание**

1) Создать проект;

2) Подключить файлы исходного кода устройства и тест.

Листинг. Исходный код функции и теста

|  |
| --- |
| int lab4\_1**(** int a**,** int b**,** int **\***c**,** int **\***d**,** int **\***p\_y**)** **{**  int y**;**  y **=** a**\***b **+** **(\***c**);**  **\***p\_y **=** a**\***b **+** **(\***d**);**  **return** y**;**  **}** |
| #include <stdio.h>  int main**()**  **{**  int res**;**  // For adders  int refOut**[**3**]** **=** **{**230**,** 640**,** 1250**};**  int pOut**[**3**]** **=** **{**240**,** 650**,** 1260**};**  int pass**;**  int i**;**  int inA **=** 10**;**  int inB **=** 20**;**  int inC **=** 30**;**  int inD **=** 40**;**  int inP **=** 1**;**  // Call the adder for 5 transactions  **for** **(**i**=**0**;** i**<**3**;** i**++)**  **{**  res **=** lab4\_1**(**inA**,** inB**,** **&**inC**,** **&**inD**,** **&**inP**);**  fprintf**(**stdout**,** " %d\*%d+%d=%d \n"**,** inA**,** inB**,** inC**,** res**);**  fprintf**(**stdout**,** " %d\*%d+%d=%d \n"**,** inA**,** inB**,** inD**,** inP**);**  // Test the output against expected results  **if** **(**res **==** refOut**[**i**]** **&** inP **==** pOut**[**i**])**  pass **=** 1**;**  **else**  pass **=** 0**;**  inA**=**inA**+**10**;**  inB**=**inB**+**10**;**  inC**=**inC**+**10**;**  inD**=**inD**+**10**;**  **}**  **if** **(**pass**)**  **{**  fprintf**(**stdout**,** "----------Pass!------------\n"**);**  **return** 0**;**  **}**  **else**  **{**  fprintf**(**stderr**,** "----------Fail!------------\n"**);**  **return** 1**;**  **}**  **}** |

**Задание:**

* Создать проект lab4\_1
* Подключить файл lab4\_1.c (папка source)
* Создать тест lab4\_1\_test.c на основе теста Подключить тест lab1\_1\_test.c
* Микросхема: xa7a12tcsg325-1q
* Сделать solution1
  + задать: clock period 6; clock\_uncertainty 0.1
  + 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
* Сделать solution2
  + Задать протоколы
    - a: ap\_hs
    - b: ap\_ask
    - \*c: ap\_hs
    - \*d: ap\_vld
    - \*p\_y: ap\_ask
  + 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
* Выводы
  + Объяснить отличие протоколов port\_level

# **Решение №1**

При создании решения зададим настройки: clock period 6, clock uncertain 0.1, устройство *xa7a12tcsg325-1q*.

## Моделирование

При запуске моделирования можно увидеть, что тест успешно пройден:

|  |
| --- |
| INFO**:** **[**SIM 2**]** **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** CSIM start **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  INFO**:** **[**SIM 4**]** CSIM will launch GCC as the compiler**.**  Compiling**(**apcc**)** **../../../../**lab4\_1\_test**.**c in debug mode  INFO**:** **[**HLS 200**-**10**]** Running 'D:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'  INFO**:** **[**HLS 200**-**10**]** For user 'AlexKomarov' on host 'desktop-kk6t6ei' **(**Windows NT\_amd64 version 6.2**)** on Thu Oct 31 15**:**44**:**34 **+**0300 2019  INFO**:** **[**HLS 200**-**10**]** In directory 'D:/xilinx\_projects/lab4/lab4/solution1/csim/build'  INFO**:** **[**APCC 202**-**3**]** Tmp directory is apcc\_db  INFO**:** **[**APCC 202**-**1**]** APCC is done**.**  Compiling**(**apcc**)** **../../../../**lab4\_1**.**c in debug mode  INFO**:** **[**HLS 200**-**10**]** Running 'D:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'  INFO**:** **[**HLS 200**-**10**]** For user 'AlexKomarov' on host 'desktop-kk6t6ei' **(**Windows NT\_amd64 version 6.2**)** on Thu Oct 31 15**:**44**:**43 **+**0300 2019  INFO**:** **[**HLS 200**-**10**]** In directory 'D:/xilinx\_projects/lab4/lab4/solution1/csim/build'  INFO**:** **[**APCC 202**-**3**]** Tmp directory is apcc\_db  INFO**:** **[**APCC 202**-**1**]** APCC is done**.**  Generating csim**.**exe  10**\***20**+**30**=**230  10**\***20**+**40**=**240  20**\***30**+**40**=**640  20**\***30**+**50**=**650  30**\***40**+**50**=**1250  30**\***40**+**60**=**1260  **----------**Pass**!------------**  INFO**:** **[**SIM 1**]** CSim done with 0 errors**.**  INFO**:** **[**SIM 3**]** **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** CSIM finish **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** |

## Синтез

Приведем в отчете требуемые данные о проекте:

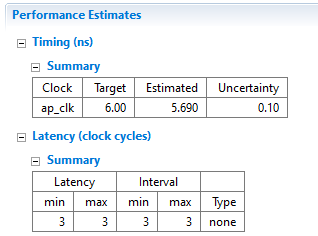


Рис. 1. Информация о производительности

Здесь можно увидеть, что достигнутая задержка равна 5.690 + 0.1, что укладывается в заданные нами требования к тактовой частоте.

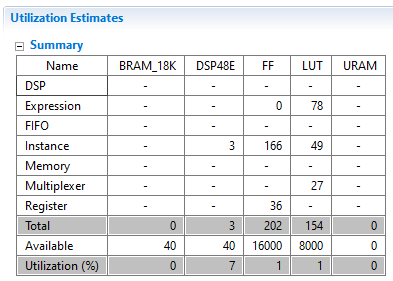


Рис. 2. Занимаемые ресурсы

Данный проект займет на микросхеме 3 DSP блока, 202 регистра для хранения чисел, и 154 LUT.

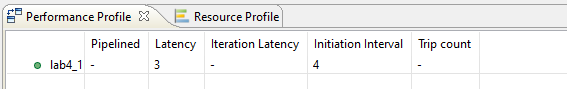


Рис. 3. Профиль производительности

На этом рисунке видно, что задержка получения выходного значения составляет 3 такта с момента старта, а задержка после старта до готовности приема новых данных – 4. Покажем эти интервалы на временной диаграмме:

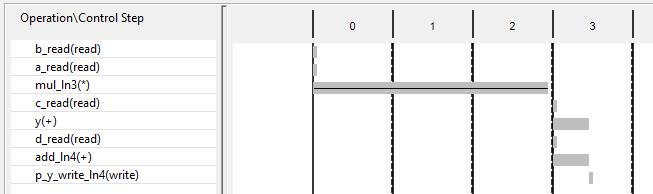


Рис. 4. Временная диаграмма

Здесь мы видим весь процесс получения результата. На первом такте происходит считывание операторов А и В, а также начинается их умножение. На четвертом такте начинается считывание C и D, а также их сложение. Таким образом суммарная задержка latency = 3, а со следующего 4-го такта можно подавать следующие данные (II = 4).

Таблица интерфейсов:

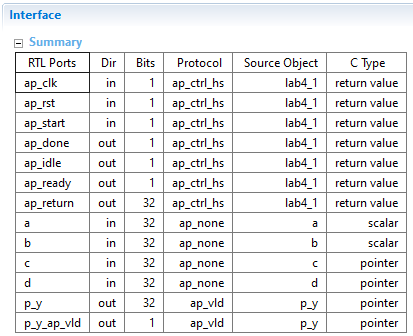


Рис. 5. Таблица интерфейсов

Для расчета схемы требуется более одного такта, поэтому в схему были добавлены ap\_clk и ap\_rst. Оба являются однобитовыми входами. Протокол управления вводом / выводом на уровне блоков был добавлен для управления RTL. Порты: ap\_start, ap\_done, ap\_idle и ap\_ready. Конструкция имеет 7 портов данных.

Входные порты: a, b, c, d являются 32-битными входами и имеют входы / выходы, протокол ap\_none.

Конструкция имеет 32-битный выходной порт для возврата функции ap\_return.

Также имеются порты: p\_y – порт выхода для указателя и его однобитовый порт управления по протоколу wire handshake protocol.

Приведем профиль ресурсов:

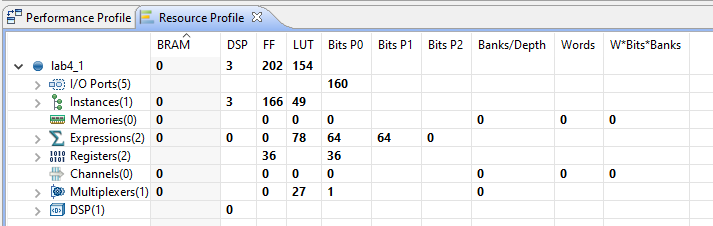


Рис. 6. Профиль ресурсов

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

## C/RTL моделирование

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

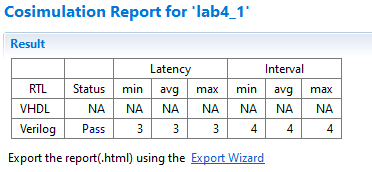


Рис. 7. Результаты C/RTL моделирования

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

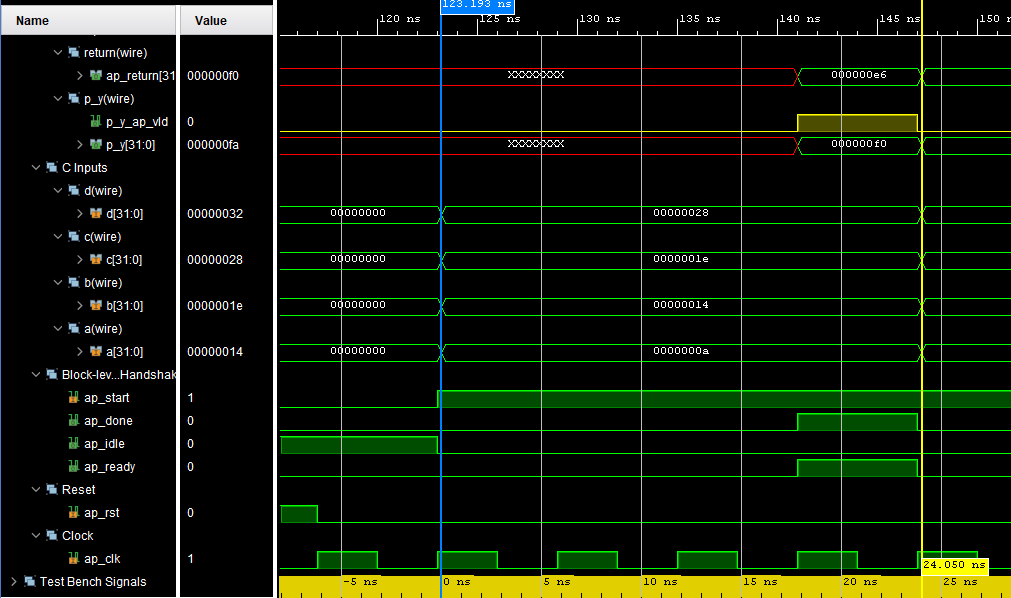


Рис. 8. Временная диаграмма совместного моделирования

# **Решение №2**

## Добавление протокола:

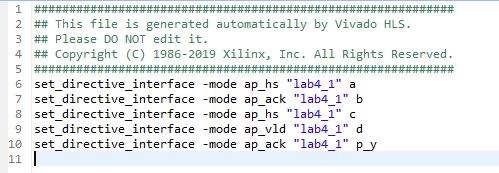


Рис. 9. Директивы

ap\_hs, ap\_ack и ap\_vld – это wire handshake протоколы ввода/вывода.

ap\_hs – протокол специализирующий валидацию значения на входе и потверждение значения на выходе.

ap\_ack – задает подтверждение на входном/выходном порте.

ap\_vld – задает валидацию на выходном/входном порте.

Значения считываются тогда, когда ap\_vld имеет высокое значение, а когда считывание закончено – ap\_ack принимает высокое значение.

## Моделирование

Создадим второе решение для данного проекта. Его настройки аналогичны настройкам первого.

|  |
| --- |
| INFO**:** **[**SIM 2**]** **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** CSIM start **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  INFO**:** **[**SIM 4**]** CSIM will launch GCC as the compiler**.**  Compiling**(**apcc**)** **../../../../**lab4\_1\_test**.**c in debug mode  INFO**:** **[**HLS 200**-**10**]** Running 'D:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'  INFO**:** **[**HLS 200**-**10**]** For user 'AlexKomarov' on host 'desktop-kk6t6ei' **(**Windows NT\_amd64 version 6.2**)** on Thu Oct 31 16**:**33**:**05 **+**0300 2019  INFO**:** **[**HLS 200**-**10**]** In directory 'D:/xilinx\_projects/lab4/lab4/solution2/csim/build'  INFO**:** **[**APCC 202**-**3**]** Tmp directory is apcc\_db  INFO**:** **[**APCC 202**-**1**]** APCC is done**.**  Compiling**(**apcc**)** **../../../../**lab4\_1**.**c in debug mode  INFO**:** **[**HLS 200**-**10**]** Running 'D:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/apcc.exe'  INFO**:** **[**HLS 200**-**10**]** For user 'AlexKomarov' on host 'desktop-kk6t6ei' **(**Windows NT\_amd64 version 6.2**)** on Thu Oct 31 16**:**33**:**10 **+**0300 2019  INFO**:** **[**HLS 200**-**10**]** In directory 'D:/xilinx\_projects/lab4/lab4/solution2/csim/build'  INFO**:** **[**APCC 202**-**3**]** Tmp directory is apcc\_db  INFO**:** **[**APCC 202**-**1**]** APCC is done**.**  Generating csim**.**exe  10**\***20**+**30**=**230  10**\***20**+**40**=**240  20**\***30**+**40**=**640  20**\***30**+**50**=**650  30**\***40**+**50**=**1250  30**\***40**+**60**=**1260  **----------**Pass**!------------**  INFO**:** **[**SIM 1**]** CSim done with 0 errors**.**  INFO**:** **[**SIM 3**]** **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** CSIM finish **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  30\*40+50+60=1310  ----------Pass!------------  INFO: [SIM 1] CSim done with 0 errors.  INFO: [SIM 3] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CSIM finish \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

Моделирование второго решения также прошло успешно.

## Синтез

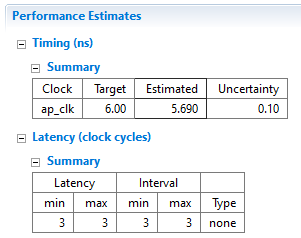


Рис. 11. Производительность

Значения задержки соответствуют значениям из предыдущего решения.

Использование ресурсов:

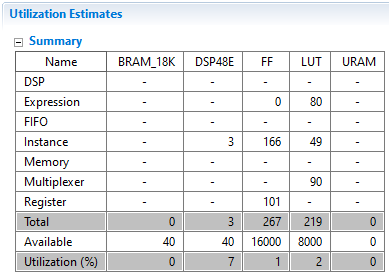


Рис. 12. Затрачиваемые ресурсы

Данный проект будет занимать на микросхеме:

* 3 DSP блока
* 267 регистров для хранения и считывания данных (чисел)
* 219 LUT

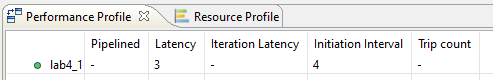


Рис. 13. Профиль производительности

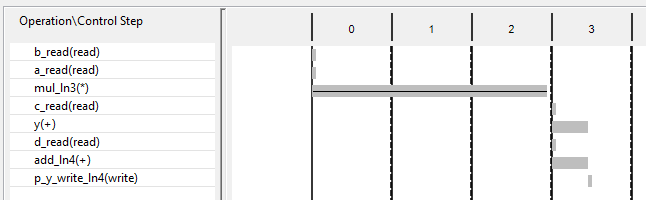


Рис. 14. Временная диаграмма

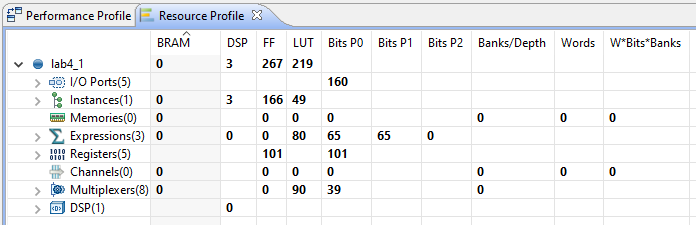
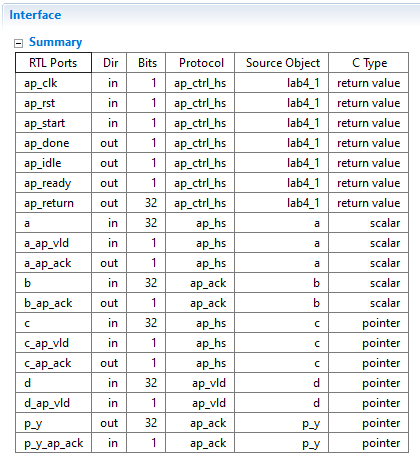


Рис. 15. Профиль ресурсов

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

Интерфейсы:



По сравнению с решением 1 появились следующие портовые протоколы ввода/вывода:

Помимо порта ввода для “a”, есть порты валидации для начала считывания и подтверждения по завершению считывания.

Помимо порта ввода для “b”, есть порт подтверждения по завершению считывания.

Помимо порта ввода для “с”, есть порты валидации для начала считывания и подтверждения по завершению считывания.

Помимо порта ввода для “d”, есть порт валидации для начала считывания.

Помимо порта вывода для “p\_y”, есть порт приема подтверждения считывания.

## C/RTL моделирование

При осуществлении совместного моделирования программа показала те же результаты для задержки.

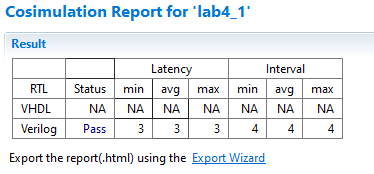


Рис. 16. C/RTL моделирование

Покажем временную диаграмму совместного моделирования:

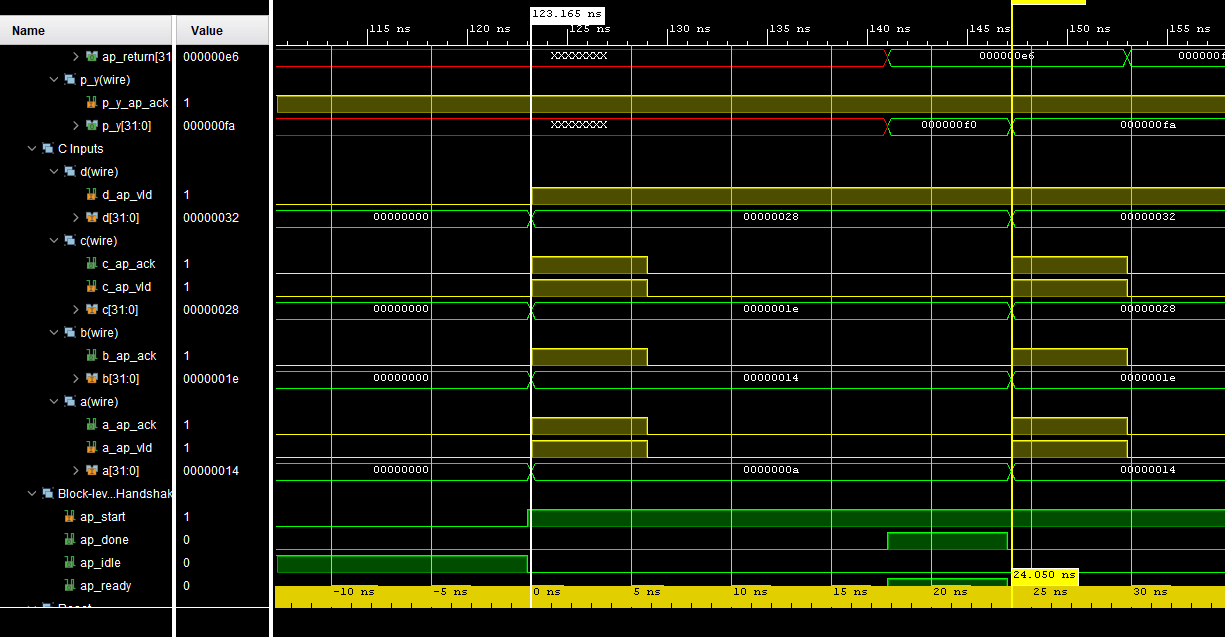


Рис. 17. Временная диаграмма совместного моделирования

# **Выводы**

Существуют следующие типы port-level wire handshake протоколов: ap\_vld, ap\_ack и ap\_hs. Они могут быть заданы портов ввода/вывода. Все порты для подтверждений валидаций являются однобитными. По умолчания для входов используется протокол ap\_none, а для выходов – ap\_vld.