

Макро Групп – официальный партнёр Xilinx в России

# High Level Synthesis

## высокоуровневое проектирование для ПЛИС

Владимир Викулин,  
Инженер по применению Xilinx

26.09.2019

# HLS – что это такое и зачем это нужно

HLS или высокоуровневый синтез – относительно новая технология проектирования для ПЛИС, при которой проектирование аппаратной части ведется на языках программирования, главным образом на C и C++

Технология HLS позволяет:

- ◆ проектировать IP ядра на C/C++ и отлаживать их как обычную программу
- ◆ напрямую переносить в ПЛИС алгоритмы, написанные для процессоров
- ◆ оптимизировать дизайн для СнК (напр. Zynq), разрабатывая всю систему на базе программного кода и повышая производительность, реализуя критические модули в программируемой логике ПЛИС
- ◆ все перечисленное повышает производительность разработчиков **в 10-100 раз!**

# План вебинара

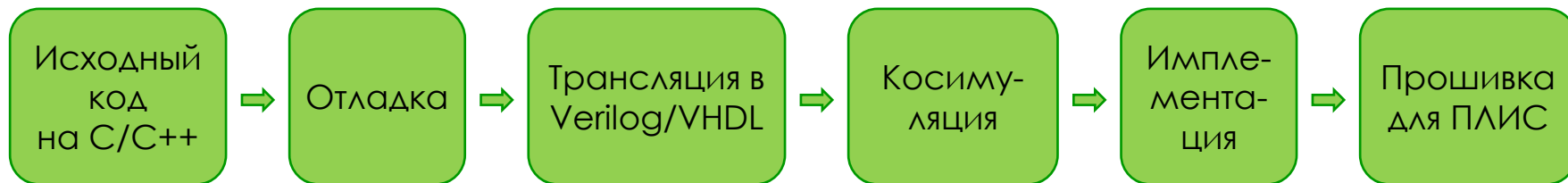
1. Разновидности реализации HLS от Xilinx
2. Vivado HLS – назначение и основные функции
3. Пример HLS проектирования на Vivado HLS и Vivado
4. Основные документы
5. Оптимизация и повышение производительности генерируемого кода
6. Портирование унаследованного C/C++ кода в ПЛИС
7. Итоги, перспективы, выводы

# 1. Реализации HLS от Xilinx

# Реализации HLS от Xilinx (1)

HLS от Xilinx – технология, позволяющая проектировать функциональность ПЛИС на языках высокого уровня – C/C++ (а так же System C).

Упрощенный маршрут проектирования



**Внимание!** Не все стандартные возможности C/C++ используются! (См. раздел 6)

# Реализации HLS от Xilinx (2)

Средства разработки Xilinx с поддержкой HLS

Название	Назначение	Ускорение разработки	Ускорение работы
<b>Vivado_HLS</b>	Разработка IP-ядер для дальнейшего использования в САПР Vivado	10-20 раз	до 2-3
<b>SDSoC</b>	C/C++ ориентированное проектирование для чипов Zynq, ZynqUS+ и MicroBlaze	10-100 раз	до 10
<b>SDAccel</b>	Программирование для серверов/рабочих станций с ускорительными картами Alveo	нет	до 50 (при понижении потребления)

SDSoC и SDAccel основаны на технологиях Vivado\_HLS

# Реализации HLS от Xilinx (3)

Продукты-аналоги других компаний

Компания	Продукт
Intel	Intel High Level Synthesis Compiler
Synopsys	Synphony C
Mentor Graphics	Catapult C

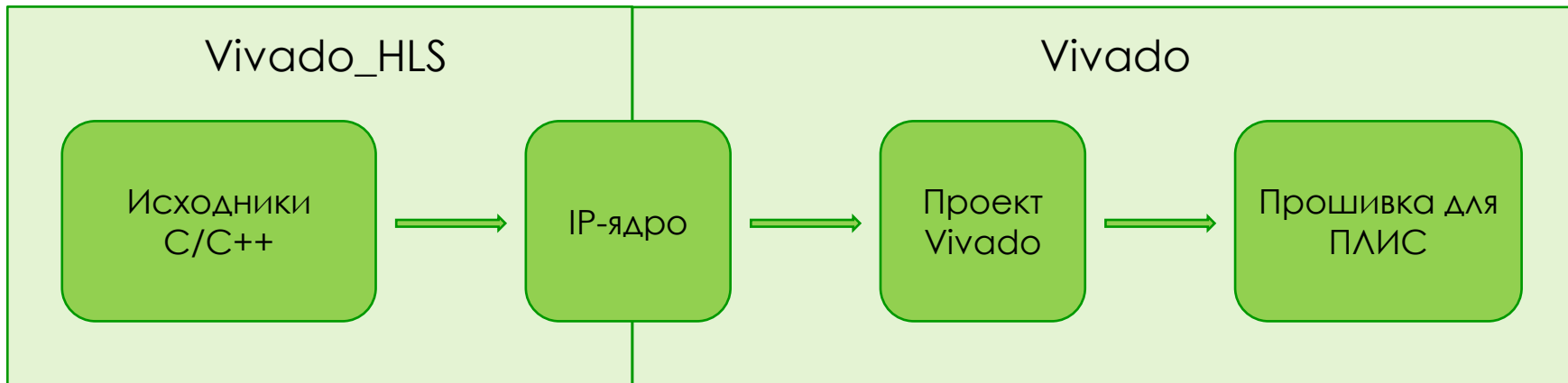
## 2. Vivado HLS – назначение и основные функции



# Vivado HLS – назначение и основные функции (1)

Назначение Vivado\_HLS – создание IP-ядра по описанию алгоритма его функционирования на языке высокого уровня (C/C++).

Основной маршрут проектирования



# Vivado HLS – назначение и основные функции (2)

Технические детали: как функция C/C++ преобразуется в модуль на HDL

```
int add_func(  
    int a, int b  
) {  
    return (a+b);  
}
```



```
module add_func(  
    input wire clk,  
    input wire rst,  
    input wire ap_start,  
    output wire ap_idle,  
    output wire ap_done,  
    output wire ap_ready,  
    input wire signed [31:0] a,  
    input wire signed [31:0] b,  
    output wire signed [31:0] ap_return  
)  
...  
    assign ap_return = a + b;  
endmodule
```



Порты,  
добавленные  
Vivado HLS



Порты,  
унаследо-  
ванные  
от ИСХОДНОЙ  
функции

# Vivado HLS – назначение и основные функции (3)

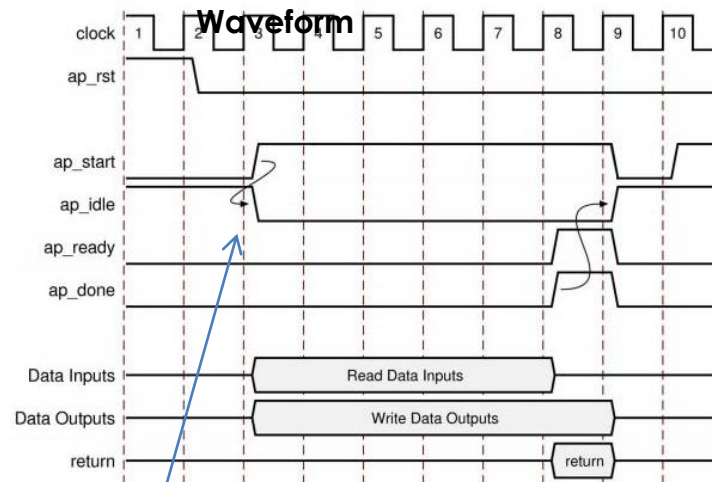
Соответствие между программной и аппаратной реализациями

```
#include <stdlib.h>
#include <stdio.h>
```

```
int main {
    int opA = 1;
    int opB = 2;

    int result = add_func(opA, opB);
    printf("result = %d\n", result);
    return (0);
}
```

```
module topmodule(
    input wire clk,
    ....
)
    ...
    add_func inst0(
        .clk ( clk),
        .rst ( rst),
        .ap_start (rg_ap_start),
        .ap_idle (w_ap_idle),
        .ap_done (w_ap_done),
        .ap_ready (w_ap_ready),
        .a (opA),
        .b (opB),
        .ap_return (result)
    );
    ...
endmodule
```



Вызову функции  
соответствует строб **ap\_start**

### 3. Пример HLS проектирования на Vivado HLS и Vivado

# Пример HLS проектирования на Vivado HLS и Vivado (1)

Основные этапы, выполняемые в Vivado\_HLS

	Действие	Инструмент (функция)
1	Разработка кода на C/C++	<ul style="list-style-type: none"><li>встроенный редактор</li><li>компилятор (команда "Run C Simulation")</li></ul>
2	Отладка	<ul style="list-style-type: none"><li>встроенный отладчик</li></ul>
3	Синтез ядра	<ul style="list-style-type: none"><li>команда "Run C synthesis"</li></ul>
4	Ко-симуляция	<ul style="list-style-type: none"><li>команда "Run C/RTL cosimulation"</li></ul>
5	Оптимизация (опционально, см. раздел 5)	<ul style="list-style-type: none"><li>разработка вариантов решения (#prahmas, Solutions)</li><li>сравнение решений</li></ul>
6	Упаковка ядра	<ul style="list-style-type: none"><li>команда "Export RTL"</li></ul>

# Пример HLS проектирования на Vivado HLS и Vivado (2)

Основные этапы, выполняемые в Vivado

	Действие	Инструмент (функция)
1	Создание RTL-проекта	
2	Создание HDL-кода и HDL – тестбенча	◆ встроенный редактор
3	Имплементация ядра	◆ IP-catalog
4	RTL-симуляция	◆ встроенный отладчик ◆ внешний отладчик
5	Имплементация	
6	Отладка на аппаратуре	◆ Встроенный анализатор
7	Генерация файла прошивки конфигурационной памяти	

# Пример HLS проектирования на Vivado HLS и Vivado (3)

Пример работы в Vivado\_HLS -> Vivado

## Работа в Vivado\_HLS

- ◆ исходные файлы
- ◆ выполняемые действия
- ◆ IP-ядро как результат работы

## Работа в Vivado

- ◆ создание проекта
- ◆ инстантиация ядра
- ◆ отладка
- ◆ имплементация

# 4. Основные документы HLS



# Основные документы HLS

## Основные документы

- ◆ User Guide [UG902](#)
- ◆ Introduction [UG998](#)
- ◆ Tutorial [UG871](#)
- ◆ UFDM for Vivado\_HLS [UG1197](#)
- ◆ Vivado\_HLS optimization guide [UG1270](#)
- ◆ OpenCV guide [UG1233](#)

# 5. Оптимизация и повышение производительности генерируемого кода

# Оптимизация и повышение производительности генерируемого кода (1)

- ◆ Vivado\_HLS позволяет создавать варианты решений и выбирать из них оптимальный вариант (механизм solutions) Solutions отличаются вариантами синтеза, в то время как исходники на C/C++ у них одинаковые.
- ◆ Имеются критерии оптимизации
- ◆ Определены методы оптимизации
- ◆ Введены инструменты оптимизации:
  1. #Pragmas – управление синтезом
  2. Arbitrary Precision Types – типы данных с заданной разрядностью

Tip: большие проекты удобнее разрабатывать на C++

# Оптимизация и повышение производительности генерируемого кода (2)

## #Pragmas и Arbitrary Precision Types

Использование #pragmas (порядка 50 прагм) – см. [UG1270](#)

Метод оптимизации	Способ реализации
Параллелизм	#pragma HLS data_pack
Конвейерная обработка	#pragma HLS pipeline
Оптимизация циклов	#pragma HLS enroll
Оптимизация разрядности	Использование Arbitrary Precision Types (крайне желательно использовать C++)
Оптимизация вычислений	Использование встроенных библиотек (UG902 Ch2)

# Оптимизация и повышение производительности генерируемого кода (3)

## Arbitrary Precision Types (UG902 Ch2, Ch4):

С код:

- Добавляем путь “-I /<HLS\_HOME>/include” в опции gpp
- Добавляем в исходники на С строку `#include "ap_cint.h"`
- Получаем доступ к дополнительным типам данных: `[u]int#W` ( $1 \leq W < 1024$ )

Пример:

```
#include "ap_cint.h"
uint13 count;
int233 bignumber;
```

С++ код:

- Добавляем путь “-I /<HLS\_HOME>/include” в опции G++
- Добавляем заголовочный файл `ap_int.h`
- Получаем доступ к дополнительным типам данных:  
`ap_int<int_W>`, `ap_uint<int_W>`

Пример:

```
#define AP_INT_MAX_W 4096 // Must be defined
                          // before next line

#include "ap_int.h"
ap_int<4096> very_wide_var;
```

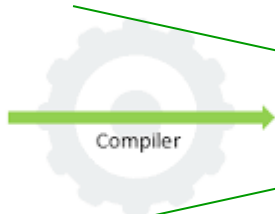
# 6. Портирование унаследованного C/C++ кода в ПЛИС

# Портирование унаследованного С/С++ кода в ПЛИС (1)

С помощью Vivado HLS можно преобразовать компьютерную программу в аппаратную реализацию на ПЛИС!

```
#include <stdio.h>
int main()
{
    printf("Hello")
    ;
    return 0;
}
```

Source code



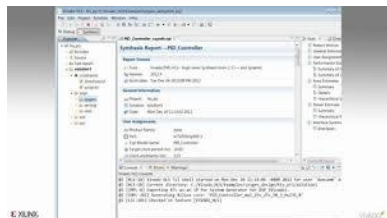
Compiler

```
100010101010101
000100101010111
011111100110000
001011001101010
0101110111100011
011111001111000
000110011110101
010010010101000
```

Executable code

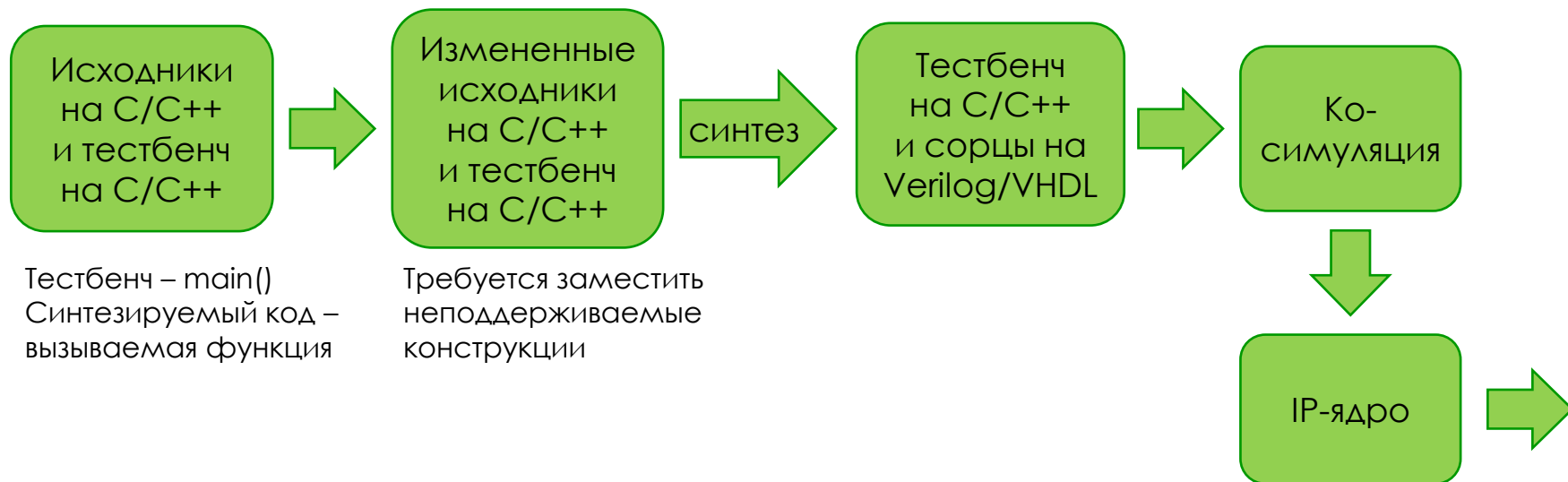


## Vivado\_HLS



# Портирование унаследованного C/C++ кода в ПЛИС (2)

Действия в Vivado HLS



На каждом этапе производится верификация и сравнение результатов



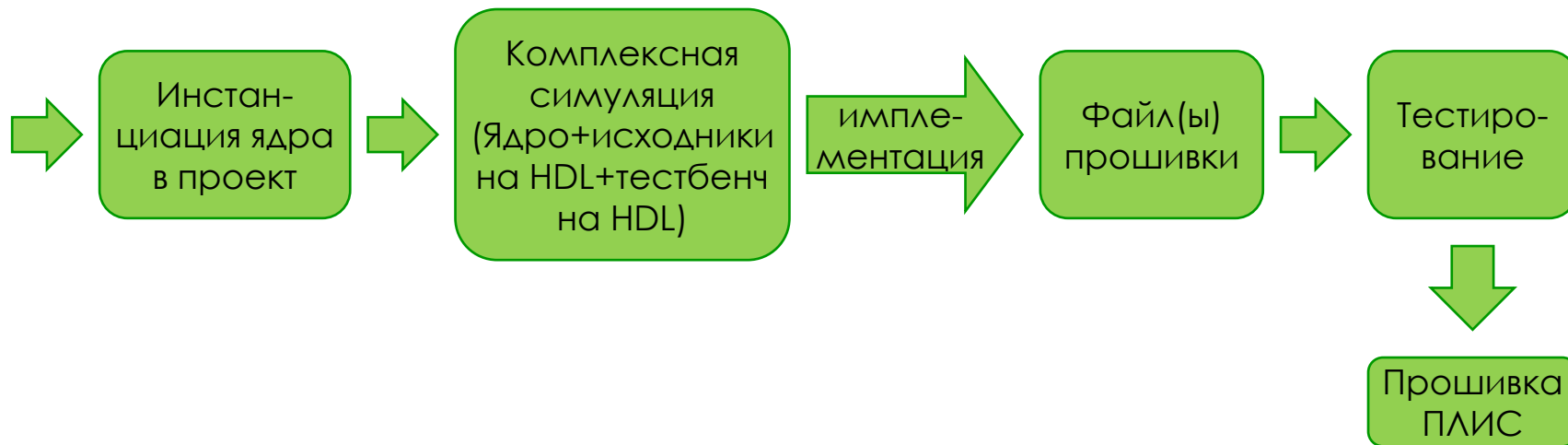
# Портирование унаследованного C/C++ кода в ПЛИС (3)

Поддерживаемые и неподдерживаемые типы данных

Свойство	Программная реализация	Реализация HLS	Замена
Динамическое распределение памяти	Поддерживается	Не поддерживается	Статическое распределение
Указатели, адресная арифметика	Поддерживаются	Частично поддерживаются	Заменить на индексы и прямые ссылки
Произвольная разрядность	Не поддерживается	Поддерживается	Можно использовать для оптимизации

# Портирование унаследованного С/С++ кода в ПЛИС (4)

Действия в Vivado



# Портирование унаследованного С/С++ кода в ПЛИС (5)

Success story: портирование ядра кодера G722.1 в ПЛИС Artix-7

Плата AES-A7EV-7A50T-G  
Artix-7 50T FPGA Evaluation Kit



## Исходный код на С

<https://www.itu.int/rec/T-REC-G.722.1-200505-I/en> , пригл. 9000 строк

## Имплементация

ПЛИС: Artix-7 (7a50tftg256-1) @25 МГц  
Собственный код на Verilog – пригл. 1000 стр.

Name	BRAM_18K	DSP48E	FF	LUT	LUTRAM
Vivado_HLS: Ядро G722.1					
Used/Available	22/150	13/120	5668/65200	22767/32600	-
Utilization (%)	14,67	10	9	69	-
Vivado: проект в целом					
Used/Available	24/150	16/120	4151/65200	22767/32600	193/9600
Utilization (%)	16	13.33	6.37	69	2

# 7. Итоги, перспективы и выводы

## ◆ **Итоги**

- ✓ Технологии HLS от Xilinx реально обеспечивают ускорение процесса проектирования и производительности проектируемых систем на ПЛИС и СнК

## ◆ **Перспективы**

- ✓ Значение и популярность HLS методов от Xilinx и других компаний будет только увеличиваться
- ✓ Для перспективной линейки Versal технологии HLS будут основными средствами разработки

## ◆ **Выводы**

- ✓ Изучайте и активно применяйте HLS-технологии Xilinx – это просто, эффективно и выгодно!

# Спасибо за внимание!

Компания Макро Групп:

- ◆ официальный партнер Xilinx
- ◆ комплексная поставка электронных компонентов
- ◆ техническая поддержка по всем вопросам применения продукции и ПО Xilinx
- ◆ контрактное производство электроники

Обращайтесь:

- ◆ [Vladimir.Vikulin@macrogroup.ru](mailto:Vladimir.Vikulin@macrogroup.ru)
- ◆ [Dmitry.Khorkov@macrogroup.ru](mailto:Dmitry.Khorkov@macrogroup.ru)
- ◆ [fpga@macrogroup.ru](mailto:fpga@macrogroup.ru)

