

Официальный партнер Xilinx в России

# Компания МАКРО ГРУПП

---

## Семинар

*Маршруты проектирования*

*и*

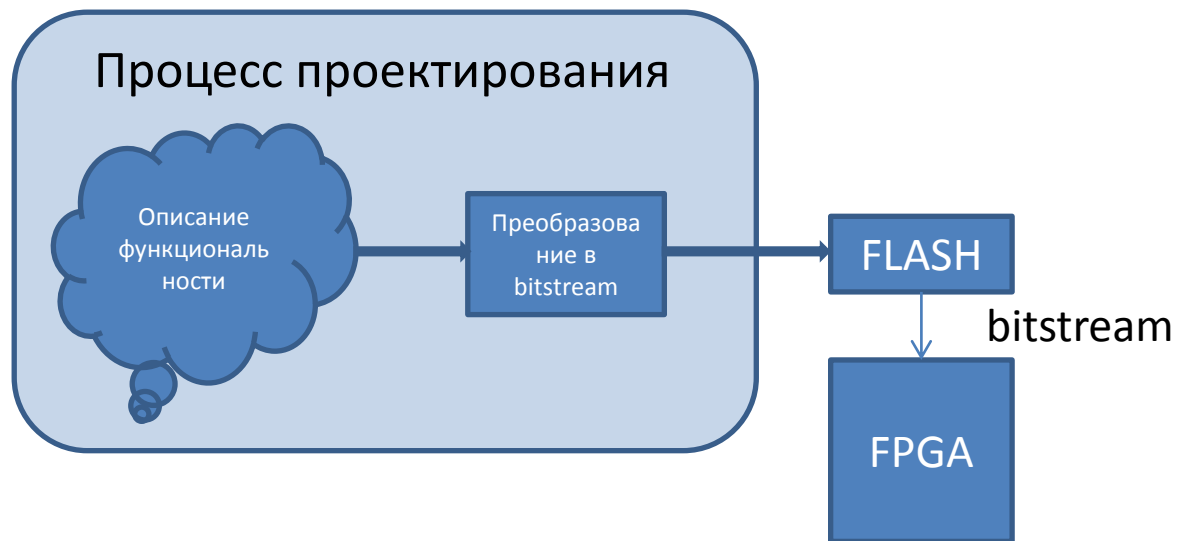
*методология сверхбыстрого  
проектирования для ПЛИС и СнК Xilinx*



Москва, 9.10.18, Спб., 11.10.18



# I. Маршруты проектирования ПЛИС и СнК



[UG892](#)

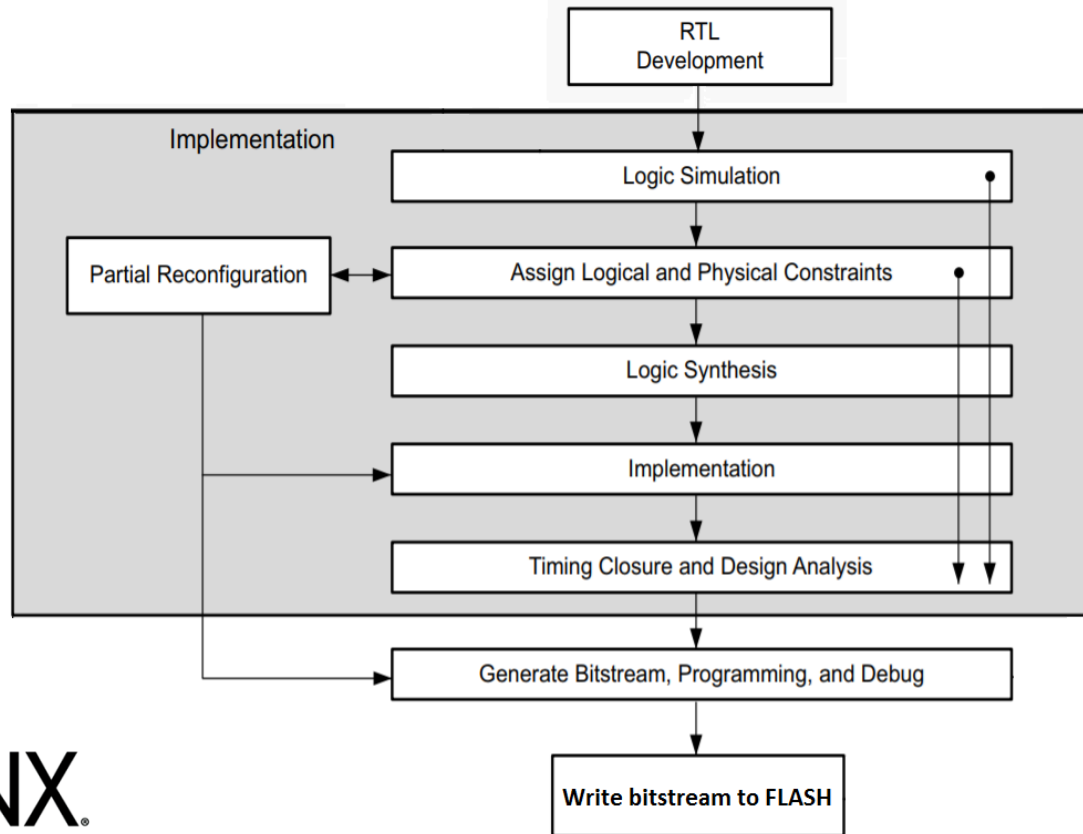
# Описание функциональности ПЛИС

Языки для описания RTL

“Классический” подход	“Высокоуровневое” проектирование
<ul style="list-style-type: none"><li>• Verilog/System Verilog (1985)</li><li>• VHDL (1987)</li></ul>	<ul style="list-style-type: none"><li>• System C</li><li>• C/C++</li><li>• Matlab/Symulink</li><li>• LabView</li></ul>



# Базовый маршрут: RTL to bitstream



[UG895](#)

# Базовый маршрут: RTL to bitstream

---

- Состоит из нескольких этапов
- Каждый этап состоит из подэтапов
- На каждом этапе возможны проблемы, трудности и задержки
- Огромное количество конфигурационных параметров

Как быстро получить гарантированный результат?

Использовать UFDМ!

[UG949](#)



# Этапы базового маршрута проектирования ПЛИС

---

1. Разработка RTL кода
2. Подключение IP-ядер
3. Логическая симуляция
4. Задание констрейнтов (ограничений)
5. Логический синтез
6. Имплементация
7. Анализ результата/улучшение тайминга
8. Генерация битстрима
9. Загрузка битстрима на носитель



# Другие маршруты проектирования

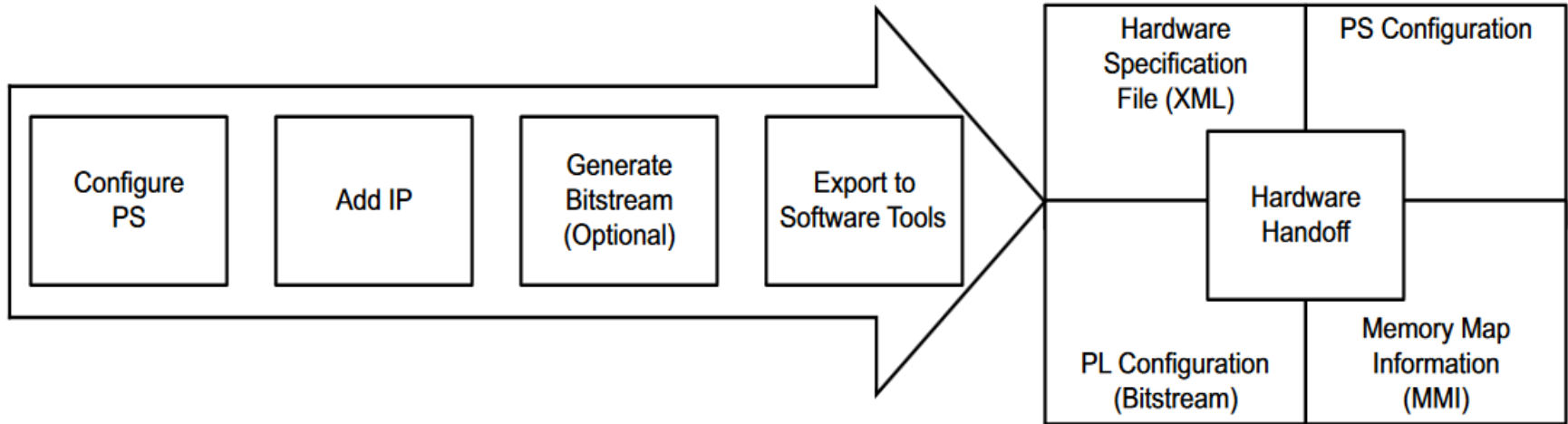
Средства разработки	Назначение
Vivado	Incremental Design; Hierarchical design; Partial reconfiguration
SDK PetaLinux	Системы на кристалле
Vivado HLS	IP-ядра на C/C++/System C
SDSoC	Системы на C/C++/System C
Vivado System Edition	Реализация математических моделей
SDNET	Разработка коммуникационных систем
SDAccel	Датацентры/Математические расчеты/Майнинг)
ReVision	ИИ, нейросети



# Этапы маршрута проектирования СнК

Создание HDF

[ug898](#)





# Этапы маршрута проектирования СнК

1. Создание HDF
2. Экспорт HDF
3. Создание BSP для разработки программного кода
4. Сборка Linux (option)
5. Создание программного кода
6. Построение проекта
7. Отладка
8. Профилирование (опционально)
9. Создание BSP для загрузочных файлов
10. Создание загрузочных файлов и запись их на носитель

[ug898](#)



# Маршрут проектирования Petalinux

Инсталляция и конфигурирование Petalinux [ug1144](#)

Инсталлировать:

- Vivado
- SDK
- Petalinux

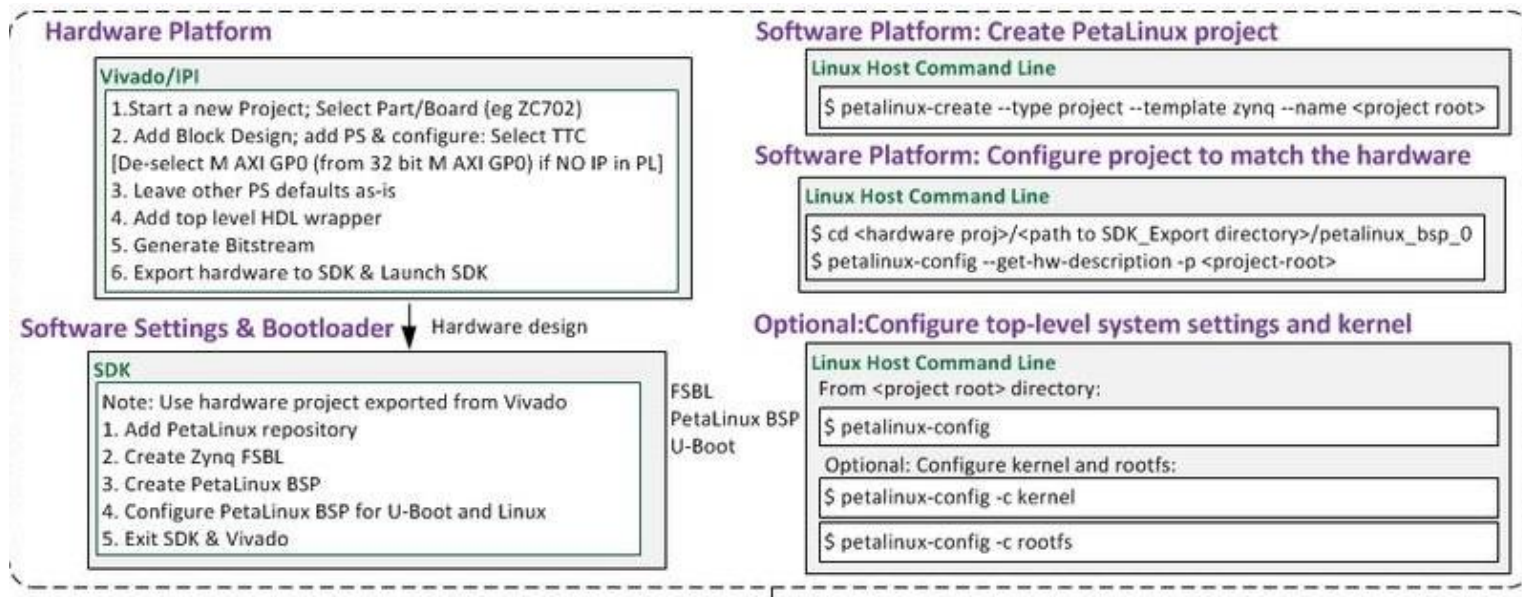


Установить окружение:  
Source <PetaLinux installation>/settings.sh



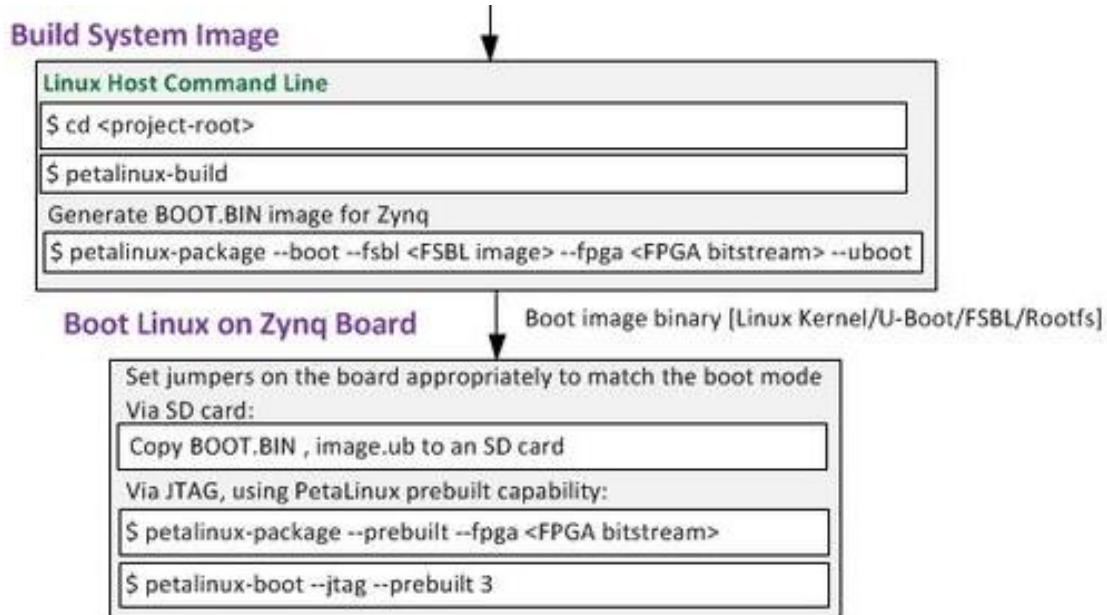
# Маршрут проектирования Petalinux

## Разработка



# Маршрут проектирования Petalinux

Построение образа системы и загрузочного образа  
(Требуется подключение к сети Интернет!)

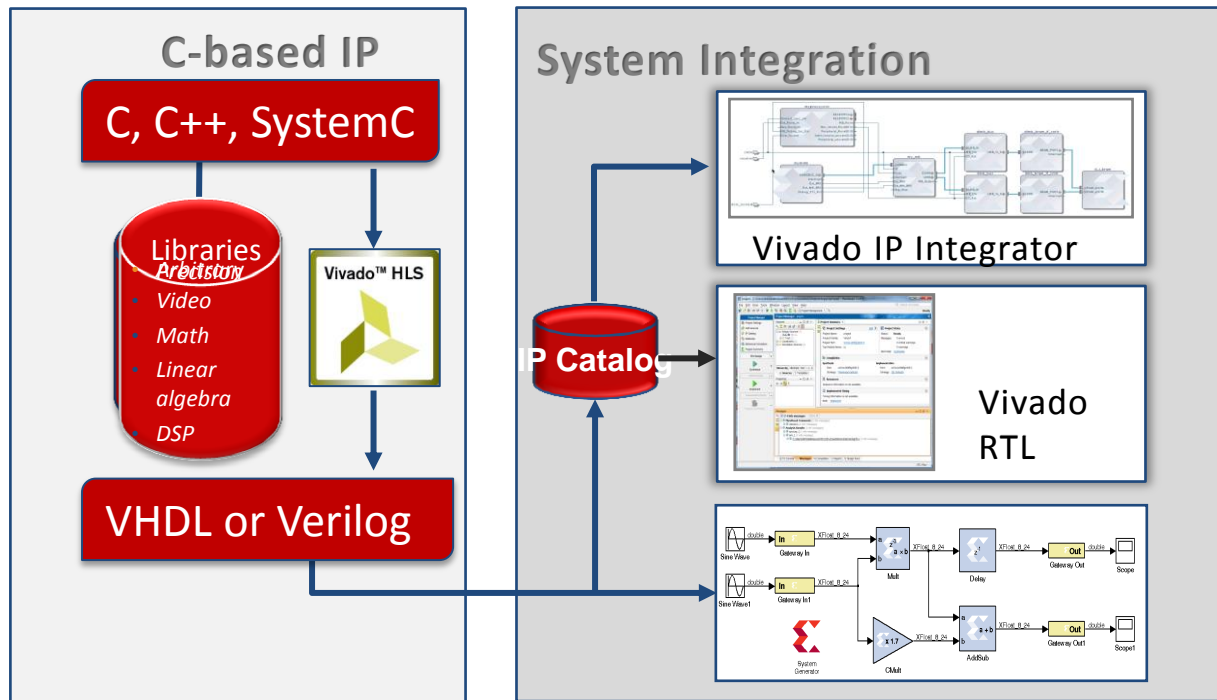


# Маршруты высокоуровневого проектирования

Проектирование аппаратуры (либо СнК) на языках (программирования) высокого уровня

Языки	C, C++, System C
Средства разработки	Vidado HLS SDSoC SDAccel/ReVision

# Маршрут проектирования Vivado HLS

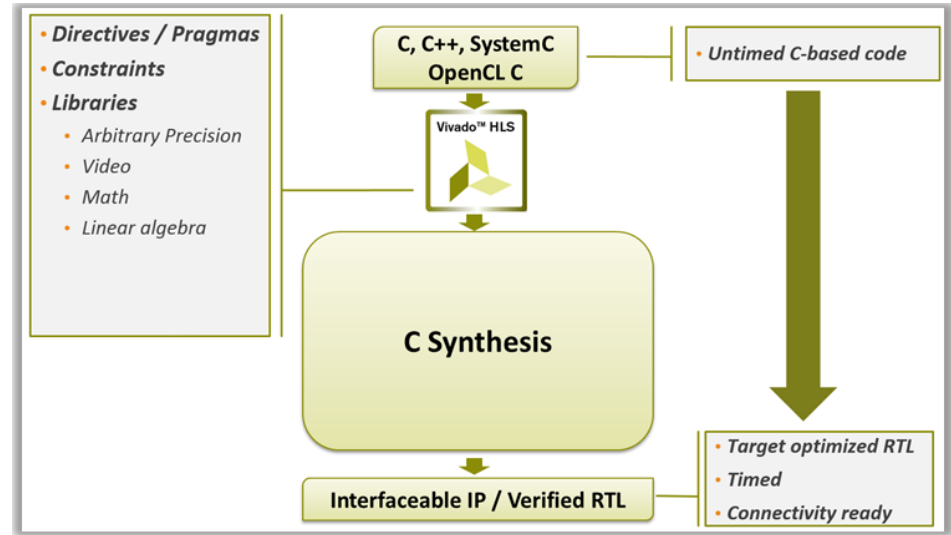
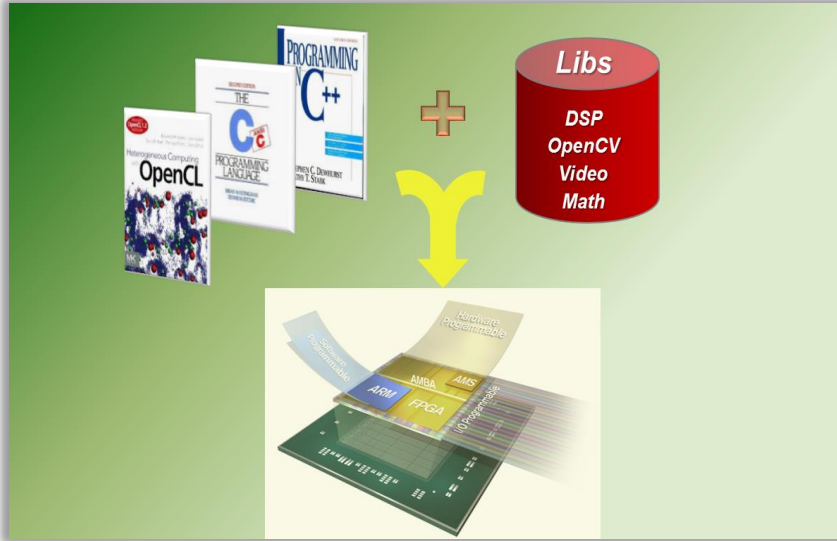


Входит в дистрибутив Vivado. Не требует отдельной лицензии!

# Этапы маршрута проектирования Vivado HLS

1. Откомпилировать, выполнить (отсимулировать), и отладить алгоритм на С.  
Прим: при высокоуровневом синтезе запуск скомпилированной С-программы симуляцией. Выполнение алгоритма на С симулирует функцию для проверки ее функциональной корректности
2. Синтезировать С-код в реализацию RTL, опционально используя директивы оптимизации (прагмы).
3. Сгенерировать отчеты и проанализировать дизайн.
4. Отверифицировать RTL-имплементацию (выполняется одним нажатием кнопки).
5. Упаковать RTL-имплементацию в набор IP-ядер

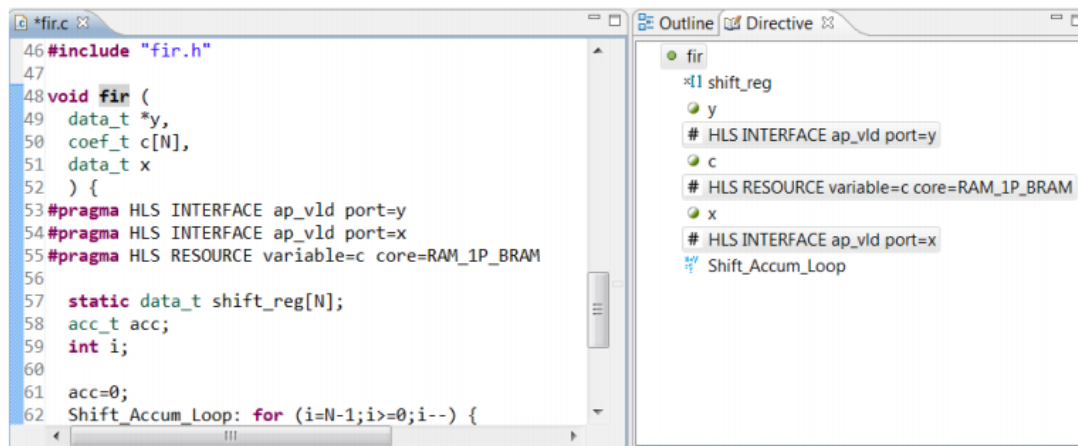
# Этапы маршрута проектирования Vivado HLS





# Этапы маршрута проектирования Vivado HLS

Пишем код на C/C++



The screenshot displays the Vivado IDE interface. On the left, the source code editor shows a C file named `*fir.c` with the following code:

```
46 #include "fir.h"
47
48 void fir (
49     data_t *y,
50     coef_t c[N],
51     data_t x
52 ) {
53     #pragma HLS INTERFACE ap_vld port=y
54     #pragma HLS INTERFACE ap_vld port=x
55     #pragma HLS RESOURCE variable=c core=RAM_1P_BRAM
56
57     static data_t shift_reg[N];
58     acc_t acc;
59     int i;
60
61     acc=0;
62     Shift_Accum_Loop: for (i=N-1;i>=0;i--) {
```

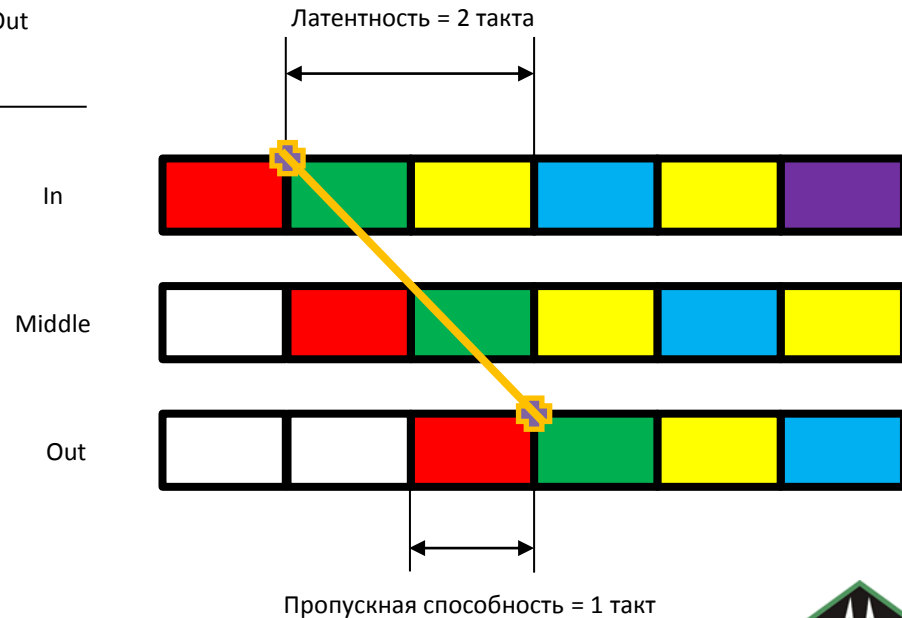
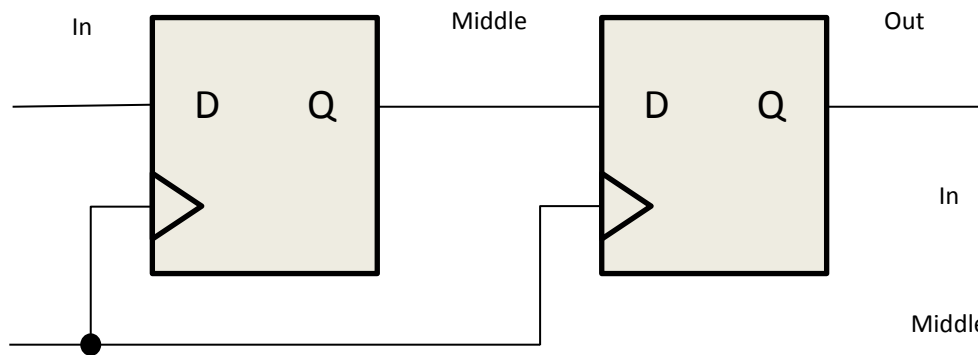
On the right, the 'Outline' pane shows the hierarchical structure of the design:

- `fir`
  - `shift_reg`
  - `y`
  - `# HLS INTERFACE ap_vld port=y`
  - `c`
  - `# HLS RESOURCE variable=c core=RAM_1P_BRAM`
  - `x`
  - `# HLS INTERFACE ap_vld port=x`
  - `Shift_Accum_Loop`

Отлаживаем его и определяем достигнутые характеристики, напр. такие как “латентность” и “пропускная способность”

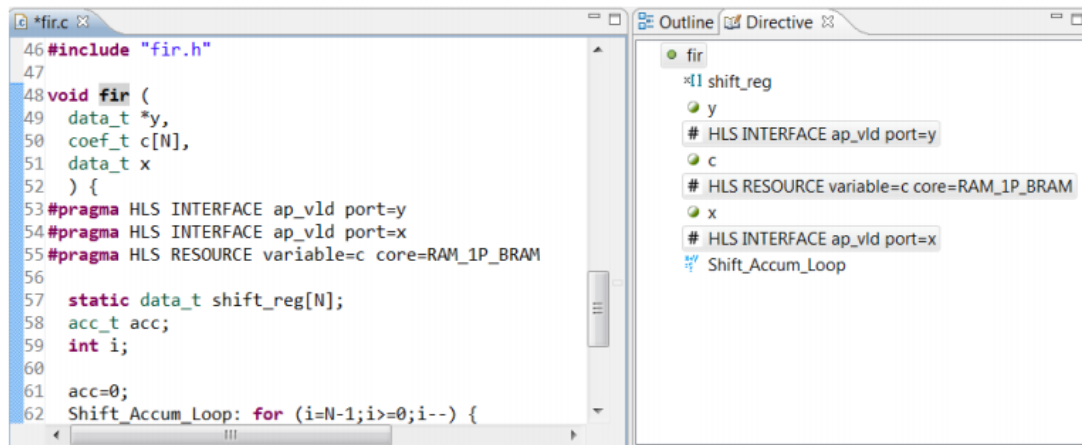


# Этапы маршрутирования проектирования Vivado HLS



# Этапы маршрута проектирования Vivado HLS

Пишем код на C/C++



The screenshot shows the Vivado HLS IDE interface. The left pane displays a C source file named `*fir.c` with the following code:

```
46 #include "fir.h"
47
48 void fir (
49     data_t *y,
50     coef_t c[N],
51     data_t x
52 ) {
53     #pragma HLS INTERFACE ap_vld port=y
54     #pragma HLS INTERFACE ap_vld port=x
55     #pragma HLS RESOURCE variable=c core=RAM_1P_BRAM
56
57     static data_t shift_reg[N];
58     acc_t acc;
59     int i;
60
61     acc=0;
62     Shift_Accum_Loop: for (i=N-1;i>=0;i--) {
```

The right pane shows the 'Outline' view of the project, listing the following components:

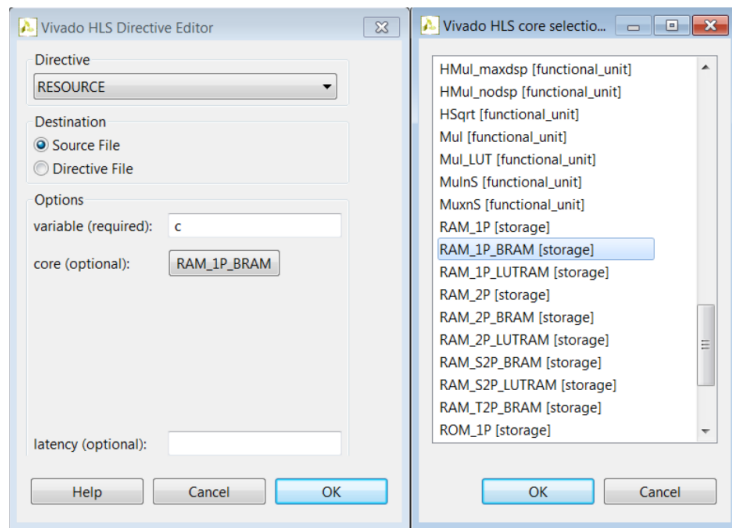
- `fir`
- `shift_reg`
- `y`
- `# HLS INTERFACE ap_vld port=y`
- `c`
- `# HLS RESOURCE variable=c core=RAM_1P_BRAM`
- `x`
- `# HLS INTERFACE ap_vld port=x`
- `Shift_Accum_Loop`

Доступные библиотеки функций:

- Arbitrary Precision Data Types Library
- HLS Stream Library
- HLS Math Library
- HLS Video Library
- HLS IP Library
- HLS Linear Algebra Library
- HLS DSP Library

# Этапы маршрута проектирования Vivado HLS

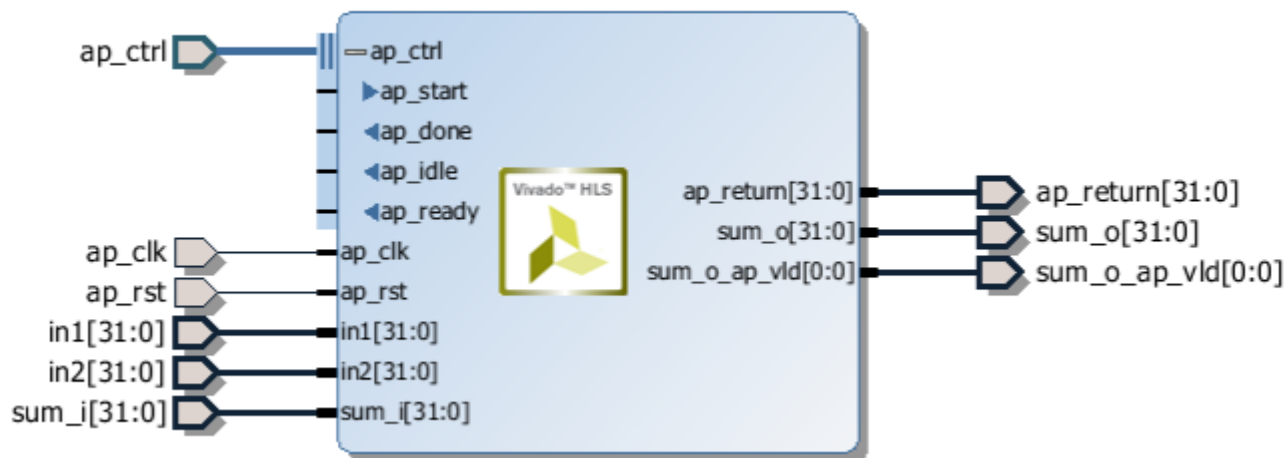
Задаем директивы имплементации (#Pragma)



```
dout_t sum_io
(din_t in1, din_t in2, dio_t *sum)
{
    dout_t temp;
    *sum = in1 + in2 + *sum;
    temp = in1 + in2;
    return temp;
}
```

# Этапы маршрутизации проектирования Vivado HLS

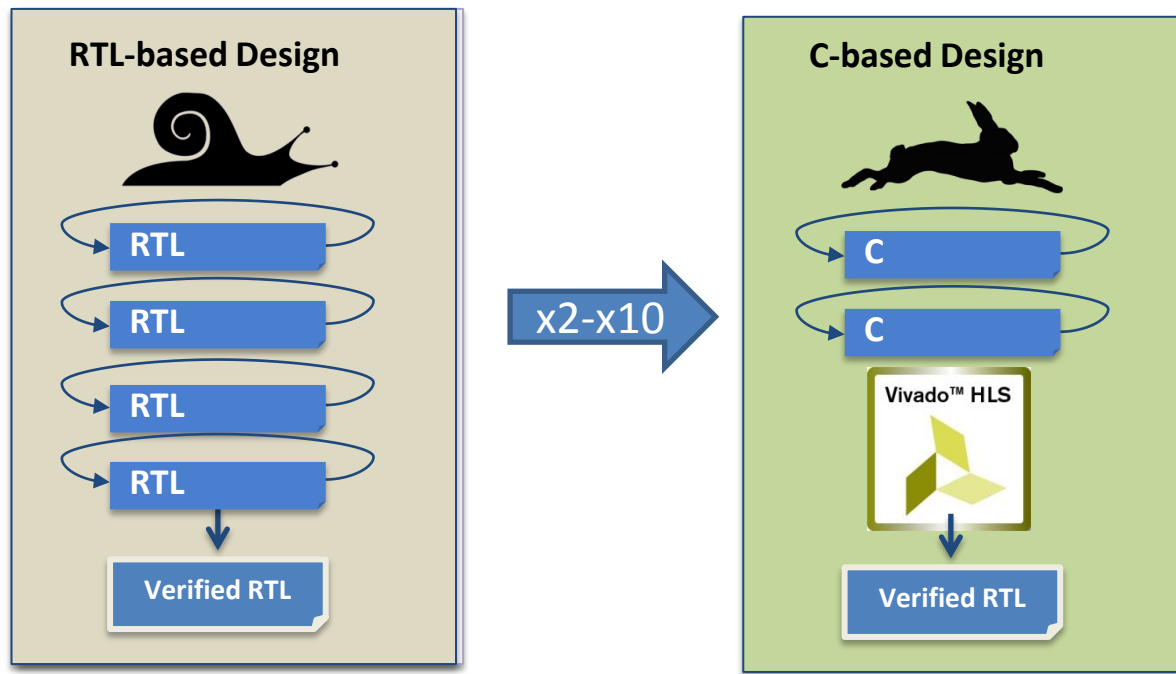
Получаем IP-ядро и сохраняем его в репозитории



# Этапы маршрута проектирования Vivado HLS

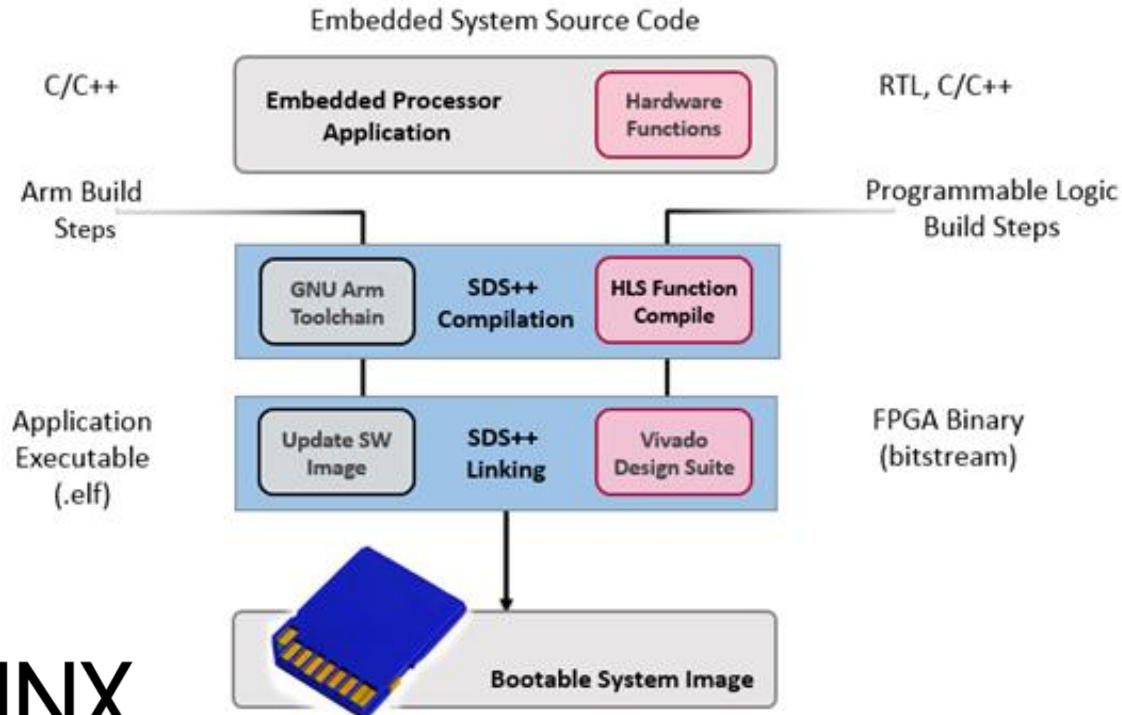
- [UG902](#) – VIVADO High Level Synthesis
- [UG1197](#) - UltraFast High-Level Productivity Design Methodology Guide
- [UG871](#) – VIVADO High Level Synthesis Tutorial

Результат – существенное ускорение процесса разработки



# Маршрут проектирования SDSoC

Высокоуровневое проектирование на C/C++ для Zynq7000/ZynqUS+

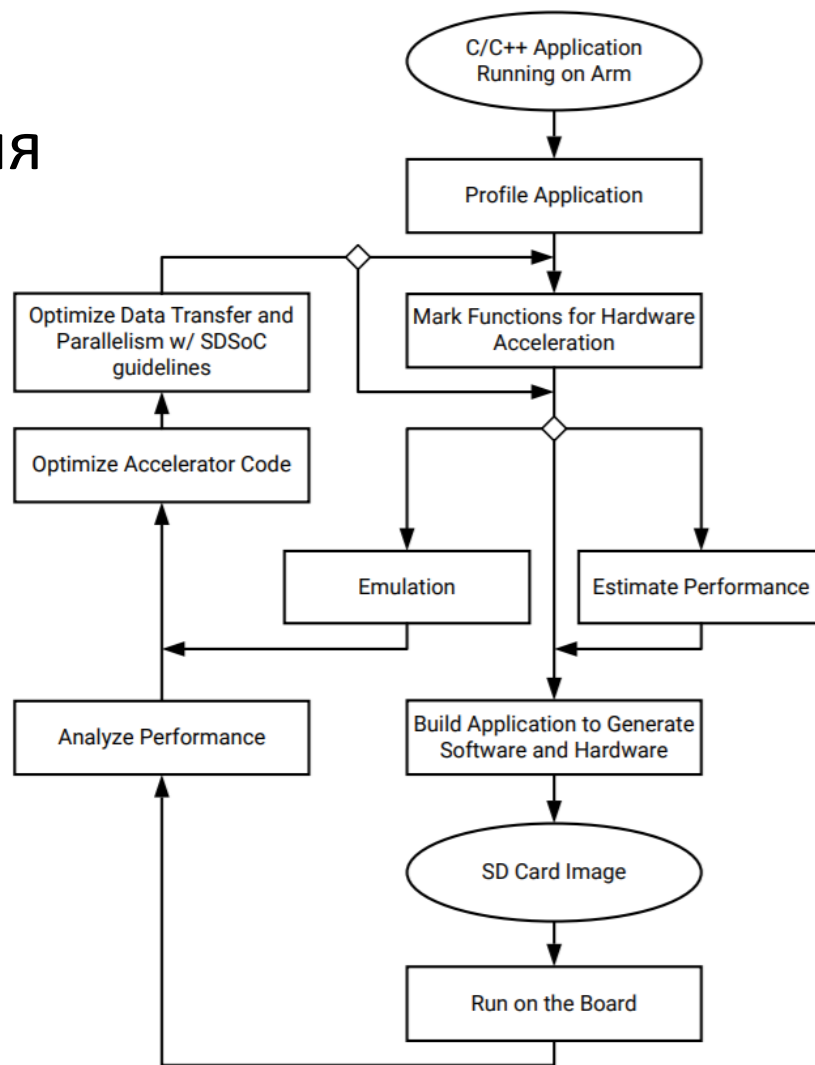


**Требуется  
отдельная  
лицензия !**

[UG1027](#) SDSoC  
Environment User  
Guide



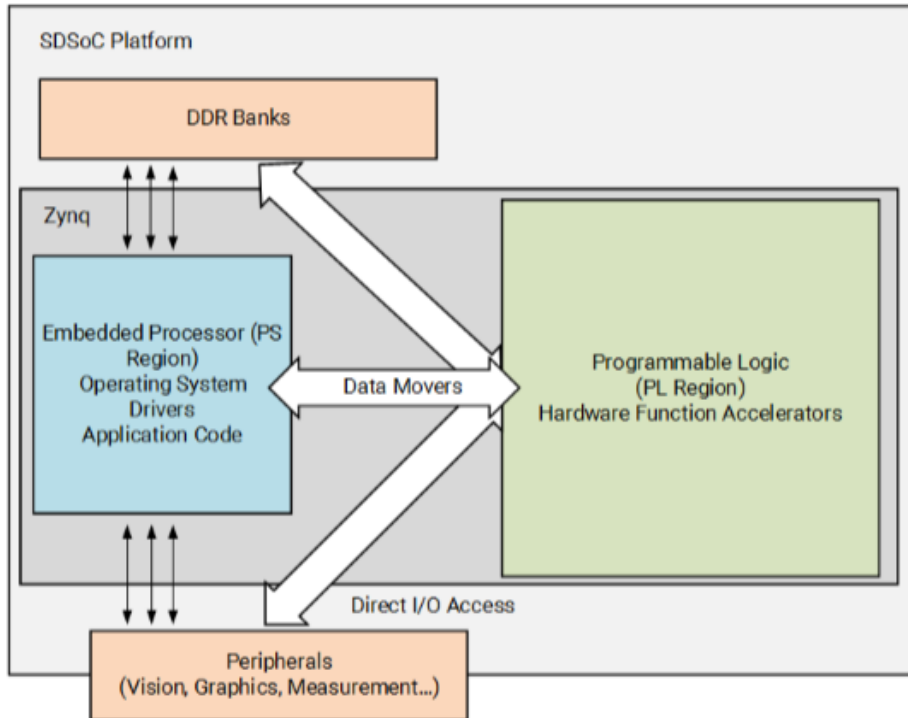
# Маршрут проектирования SDSoC





# Маршрут проектирования SDSoC

## Архитектура СнК



## Поддержка библиотек:

- DSP
- Video
- fixed point
- linear algebra
- BLAS
- OpenCV

## Поддержка ОС:

- ✓ Linux
- ✓ FreeRTOS
- ✓ Bare Metal

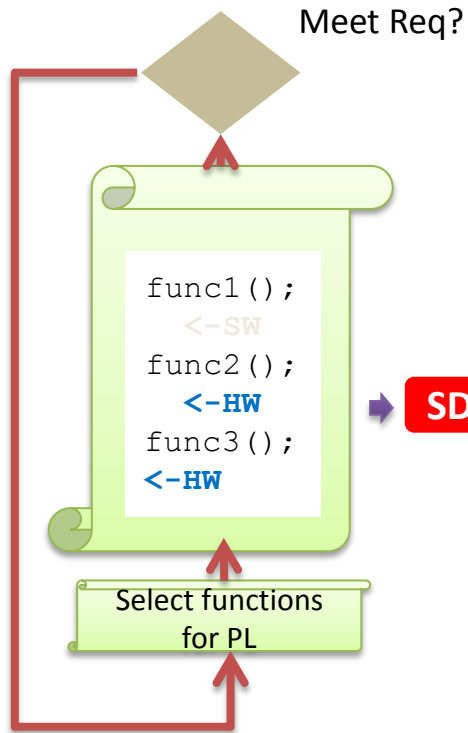


# Маршрут проектирования SDSoC

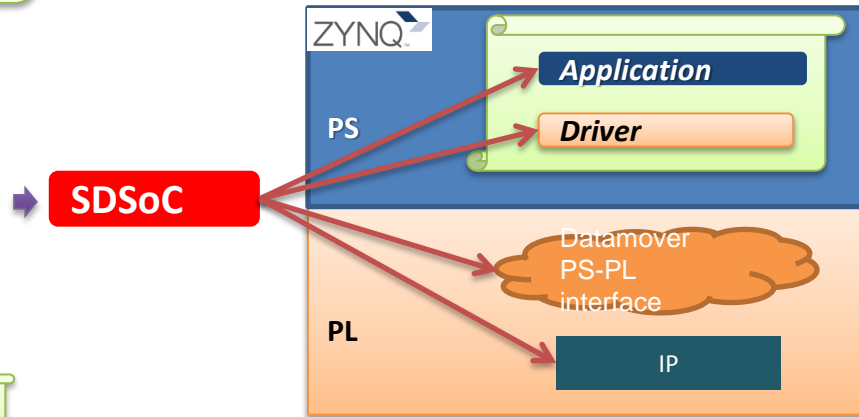
Маршрут проектирования до внедрения SDSoC



# Маршрут проектирования SDSoC



После внедрения SDSoC:  
Автоматическая генерация системы



Ускорение работы от 20х до 100 раз



# Маршрут проектирования SDSoC

---

Учебные материалы и документация:

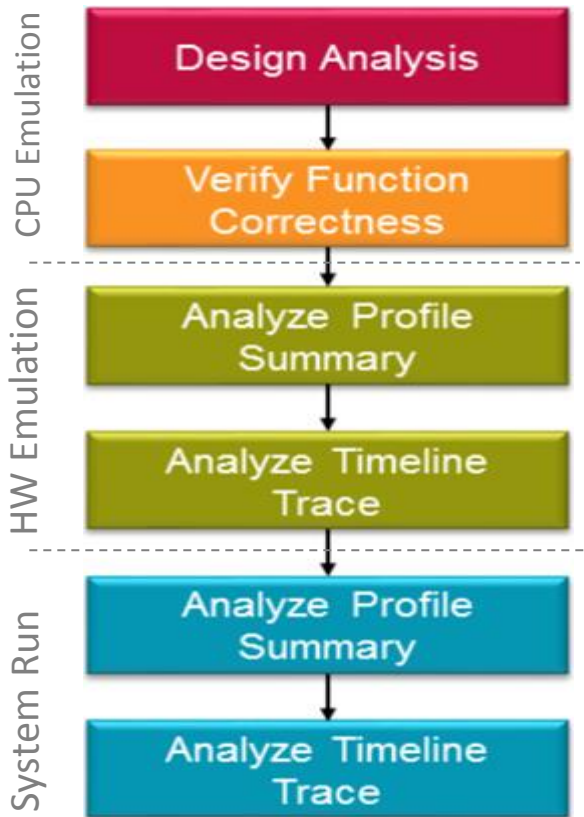
[На сайте github](#)

- Введение: Девять [лабораторных работ](#)
- Документация: [UG1027](#), [readme file](#)
- Учебное видео:
  - <https://www.youtube.com/watch?v=SiOXTJ8IkJA>
  - <https://vimeo.com/251893932>

# Маршрут проектирования SDAccel

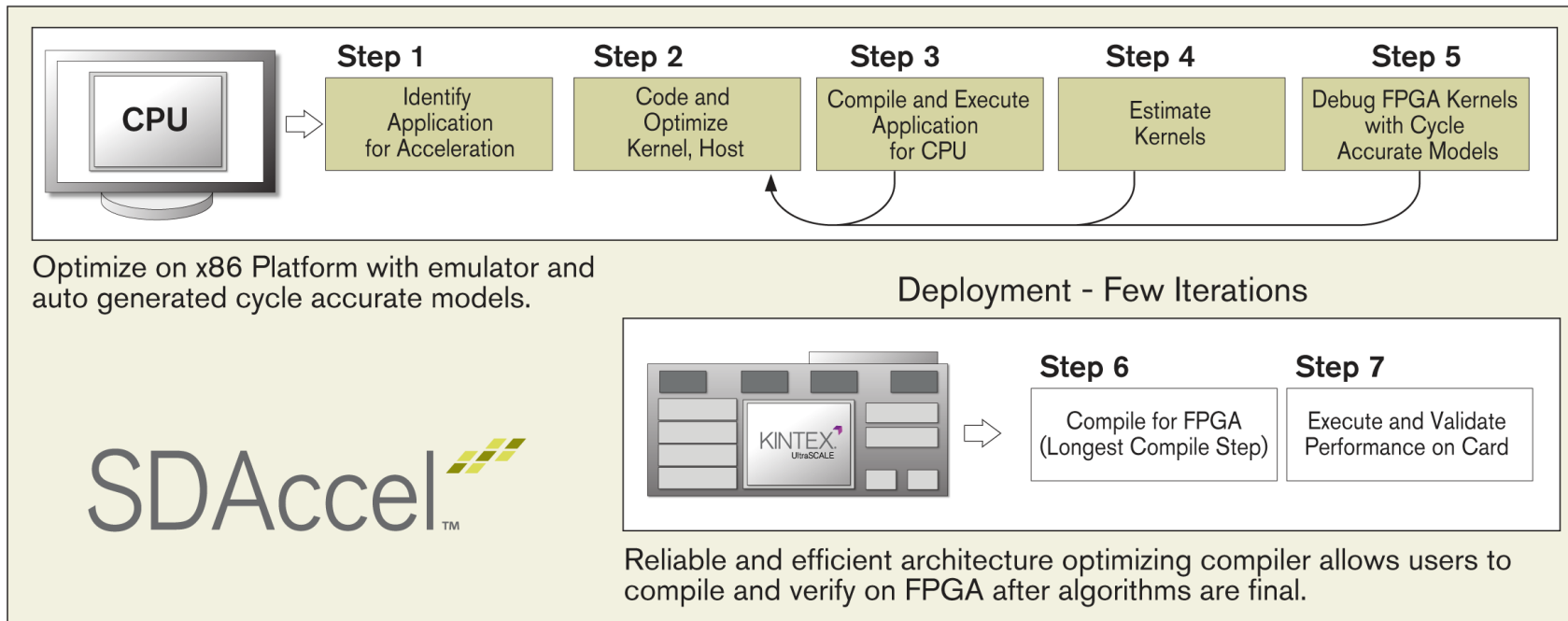
Поддержка OpenCL

[UG1023](#)



# Маршрут проектирования SDAccel

## SDAccel - Accelerated OpenCL Programming and Deployment



# Маршрут проектирования SDAccel

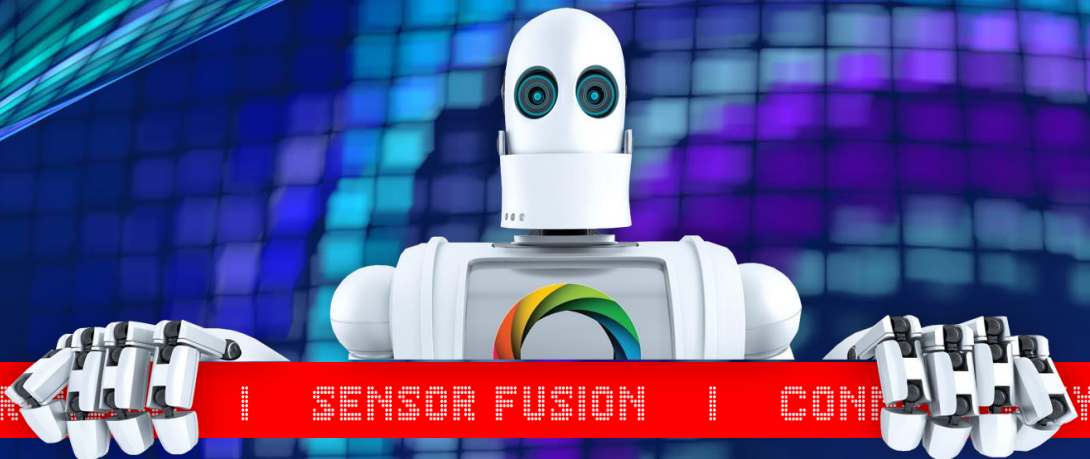
---

Поддержка:

- ✓ Nimbix
- ✓ Amazon
- ✓ Xilinx XBB: Платформы для датацентров (на базе отладочных плат):
  - работа в режиме 24x7 в течение 3-5 лет

# reVISION™

Responsive and Reconfigurable Vision Systems



- ✓ Разработка систем ИИ
- ✓ Выделение и распознавание объектов в видеопотоке
- ✓ Нейронные сети, для классификации и принятия решений
- ✓ Функционирование в реальном времени

**Все это возможно уже сейчас!**





# Стек reVISION - назначение

Разработка систем ИИ на базе нейронных сетей и систем обработки видеопотока:

- ✓ Системы искусственного интеллекта
- ✓ Обработка видео
- ✓ Распознавание образов
- ✓ Системы принятия решения на базе нейросетей

Применение элементной базы и технологий Xilinx позволяет получать решения в области систем искусственного интеллекта, функционирующие в режиме реального времени



Что означает понятие “СТЕК”?

“СТЕК” (в данном контексте) - это нечетко определенный термин для описания набора иерархически связанных друг с другом средств, систем и технологий для решения каких-либо задач

Состав стека Revision (см. Слайд ниже):

# Стек reVISION™



Caffe

Разработка  
приложений



SDSoC  
Environment

DNN

CNN

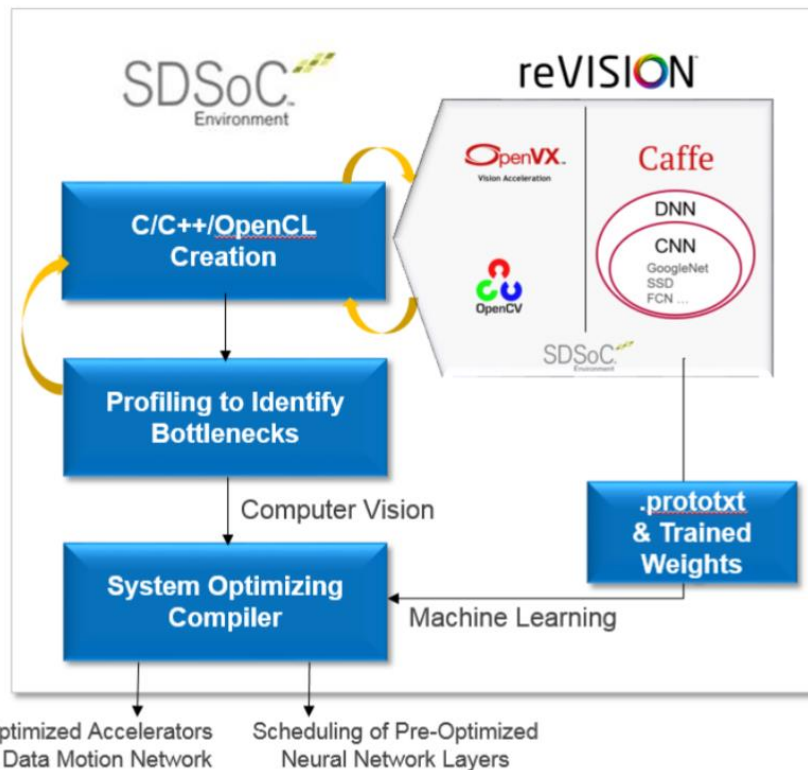
GoogLeNet  
SSD  
FCN ...

Разработка  
алгоритмов



Разработка  
аппаратуры

# Маршрут проектирования ReVision



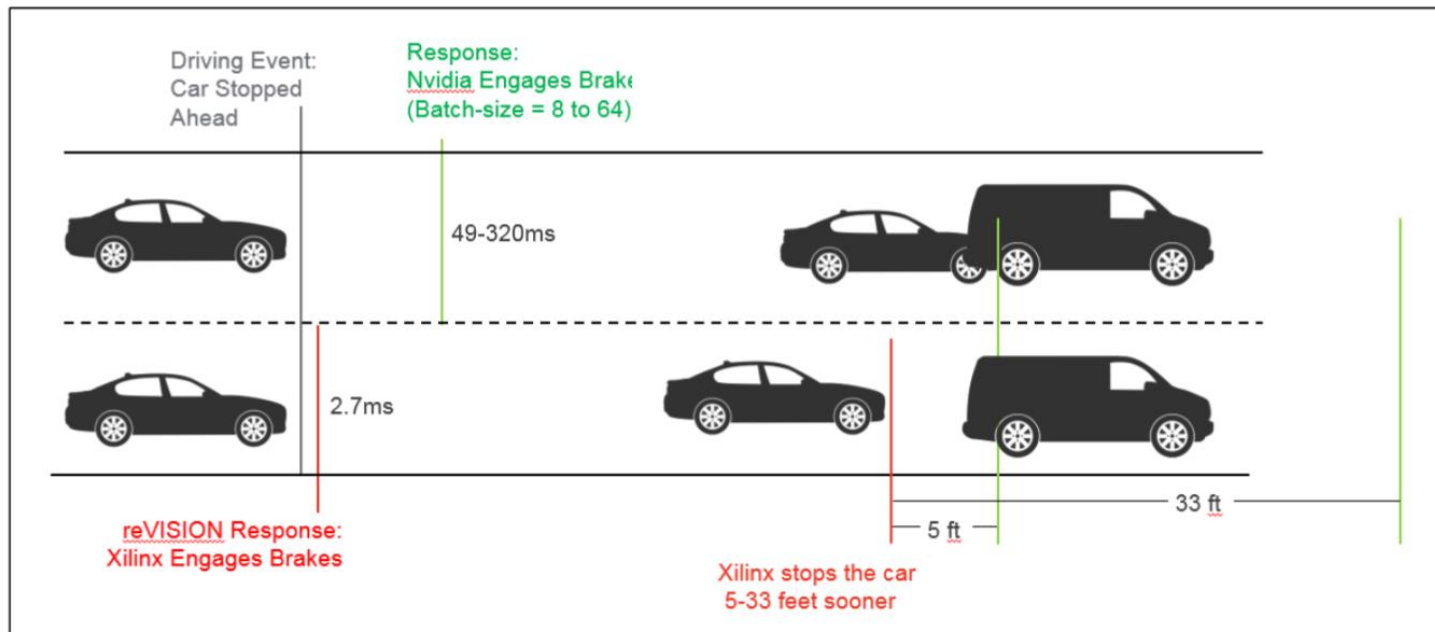
Принцип:  
80% - Xilinx  
20% - Разработчик

# Стек reVISION™

Преимущества  
систем ИИ Xilinx:

Nvidia:

 **XILINX.**  
reVISION™



- Xilinx: ZU9 running GoogLeNet @ batch = 1
- Nvidia TX1: 256 Cores; running GoogLeNet @ batch = 8
- Assuming the car was driving at 65 mph

Figure 11: Why Response Time Matters: Xilinx vs Nvidia Tegra X1

 **XILINX.**

Больше информации [здесь](#)



# Маршруты проектирования Xilinx

## Резюме:

- ✓ Xilinx развивает большое количество систем проектирования => много маршрутов проектирования
- ✓ Будущее – за системами HLS
- ✓ Имеется отличная документация – подробные пошаговые руководства охватывают все этапы процесса разработки
- ✓ Доступны многочисленные обучающие материалы
- ✓ Проблема – выбор из огромного числа документов

Где найти дополнительную информацию: Design HUBs

<https://www.xilinx.com/support/documentation-navigation/design-hubs.html>



# II. Методология сверхбыстрого проектирования

---

- Назначение
- Основные принципы
- Методические рекомендации по всем этапам маршрута проектирования



# Методология сверхбыстрого проектирования

---

Зачем это нужно?

- Объем ПЛИС с каждым годом возрастает
- Значительно повысилась сложность разработки
- ПЛИС находится в центре системы, а не как раньше, выполняет функции glue logic





# Методология сверхбыстрого проектирования

---

Что делать?

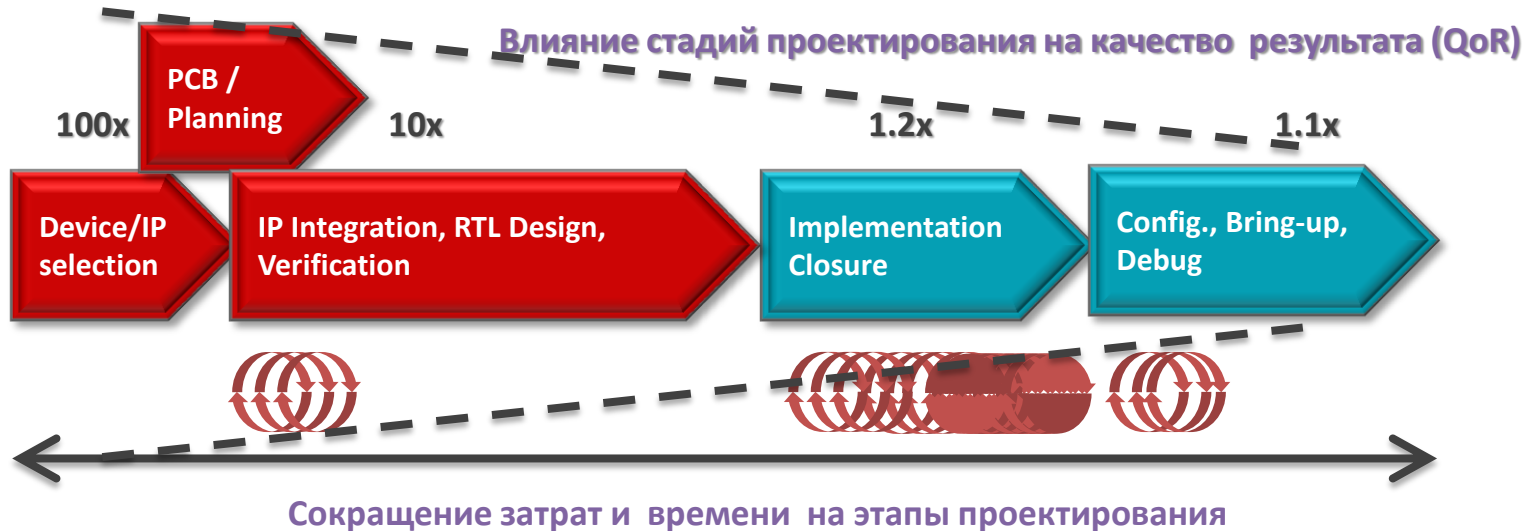
➤ Применять наилучшую методологию разработки

# UFDM!

# Методология сверхбыстрого проектирования

## ➤ Многократно возрастает цена ошибки и трудоемкость ее исправления

- Предварительный анализ
- Усилия по завершению этапа
  - Небольшие, быстрые итерации → более быстрая верификация
  - Более короткие детерминированные циклы отладки



# Методология сверхбыстрого проектирования

---

Xilinx разработал такую методологию и назвал ее UFSM. Что дает разработчику ее применение?

- Уменьшение риска неудачного завершения проекта
- Существенное сокращение времени разработки
- Повышение надежности разрабатываемой системы



# Методология сверхбыстрого проектирования

---

## Что представляет собой UFDМ?

- UFDМ – комплексная пошаговая методология, охватывающая все основные маршруты проектирования и нацеленная на гарантированное достижение результата при сокращении времени разработки (по сравнению с традиционными методами)
- Представляет собой рекомендации и указания по выполнению каждого шага маршрута проектирования
- Требуется планирование и документирование действий разработчика



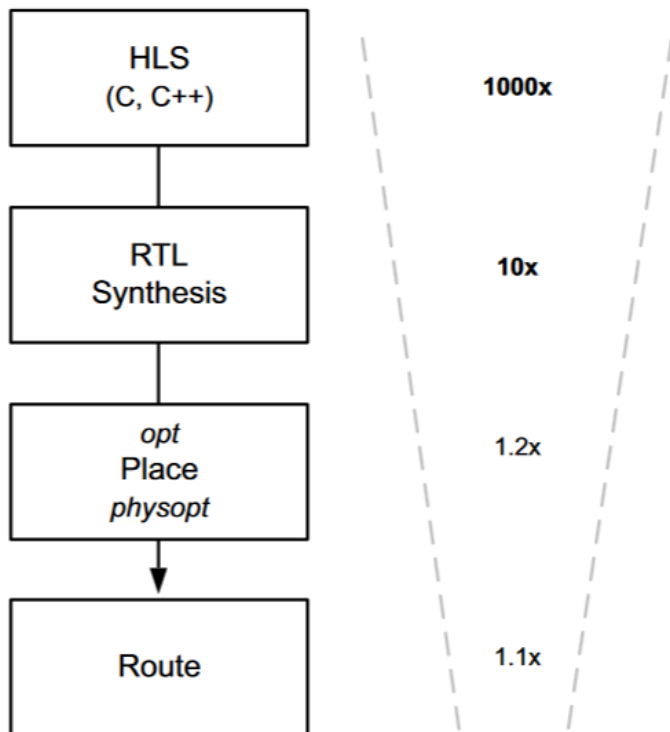
# Методология сверхбыстрого проектирования

## Концепции UDFM

1. Ранние стадии имеют максимальное влияние на производительность
2. Выполняйте проверки на каждой стадии
3. Используйте преимущество раннего выявления проблем



Влияние изменений на производительность



# Методология сверхбыстрого проектирования

## Основные документы

1. [UG949](#) UltraFast Design Methodology Guide for the Vivado Design Suite
2. [Xtp301](#) Design-methodology-checklist
3. [UG1046](#) UltraFast Methodology Guide for Embedded Design
4. [Xtp397](#) Embedded Design checklist
5. [UG1197](#) UltraFast High-Level Productivity Design Methodology Guide
6. [UG1228](#) Zynq UltraScale+ MPSoC Embedded Design Methodology Guide
7. Руководства по отдельным сериям
8. [UG1231](#) UltraFast Design Methodology Quick Reference Guid



# Методология сверхбыстрого проектирования

---

## Три основных документа

Краткое описание UFDМ

[UG1231 UltraFast Design Methodology Quick Reference Guid](#)

Основной документ

[UG949 UltraFast Design Methodology Guide for the Vivado Design Suite](#)

Основной список проверок (чеклист) для заполнения

[Xtp301 UltraFast Design Methodology Checklist](#)

Чеклисты можно загрузить непосредственно через DocNav



# Методология сверхбыстрого проектирования

---

## Основные документы для встроенных систем и СнК

UFDM для систем на кристалле: UltraFast Embedded Design Methodology Guide ([UG1046](#)) и чеклист: [xtp397](#)

## Основные документы для HLS

UFDM для высокоуровневого проектирования: High-Level Productivity Design Methodology Guide ([UG1197](#))



# Проектирование в соответствии с UFDМ

---

Этапы работы в соответствии с UFDМ ([UG1231](#))

1. Проектирование печатной платы и определение в/в ПЛИС
2. Проектирование и имплементация
3. Проверка корректности (Top-level Design Validation)
4. Анализ проекта
5. Достижение требуемых параметров (design closure)



# Проектирование в соответствии с UFDМ

---

Алгоритм работы (классический маршрут проектирования)

1. Руководствуемся [UG1231](#)
2. Заполняем [ХТР301](#)
3. Проходим все этапы, перечисленные в [UG1231](#), в соответствии с приведенными в нем документами Xilinx ([UG949](#) и др.)  
(Внимание: некоторые этапы выполняются параллельно!)



# Проектирование в соответствии с UFDМ

## Этап 1. Проектирование устройств и печатных платы (BOARD AND DEVICE PLANNING)

Подэтап (разработчик РСВ)	Подэтап (разработчик ПЛИС)
Определение основных интерфейсов	Планирование ввода/вывода
Разработка структуры печатной платы	Определение в/в для основных интерфейсов
Разработка и верификация схемы платы	Окончательное задание в/в
Изготовление и тестирование п.п.	Оценка потребляемой мощности



# Проектирование в соответствии с UFDМ

---

## Этап 1.

### Планирование В/В

- Создайте IO planning project
- Предварительно задайте пины
- Импортируйте пины, создайте графическое обозначение компонентов
- Преобразуйте в RTL-проект



# Проектирование в соответствии с UFDМ

## Этап 2. Разработка и имплементация (Design Entry and Implementation)

### Подэтап (разработчик логики ПЛИС)

Разработка и верификация RTL-модулей

Сборка и верификация проекта

Сборка и верификация топ-модуля



# Проектирование в соответствии с UFDМ

---

## Этап 2.

### Разработка RTL-модулей

- Перед кодированием собственных модулей изучите [UG949](#) Ch3 и [UG901](#) ch4
- Используйте готовые шаблоны (tools/language tempates)
- Загрузите примеры кода
- Избегайте лэтчей (анализируйте отчет о синтезе и убирайте лэтчи)
- Пользуйтесь уточняющими директивами



# Проектирование в соответствии с UFDМ

---

## Этап 2.

### Разработка RTL-модулей

- Задайте дерево клоков
- Установите необходимые клоковые буфера
- Элементы тактирования расположите на верхнем уровне
- Установите буфера для входных и выходных сигналов (напр. для дифф. входов) на верхнем уровне
- Планируйте сигналы сброса и разрешения так же, как и клоки

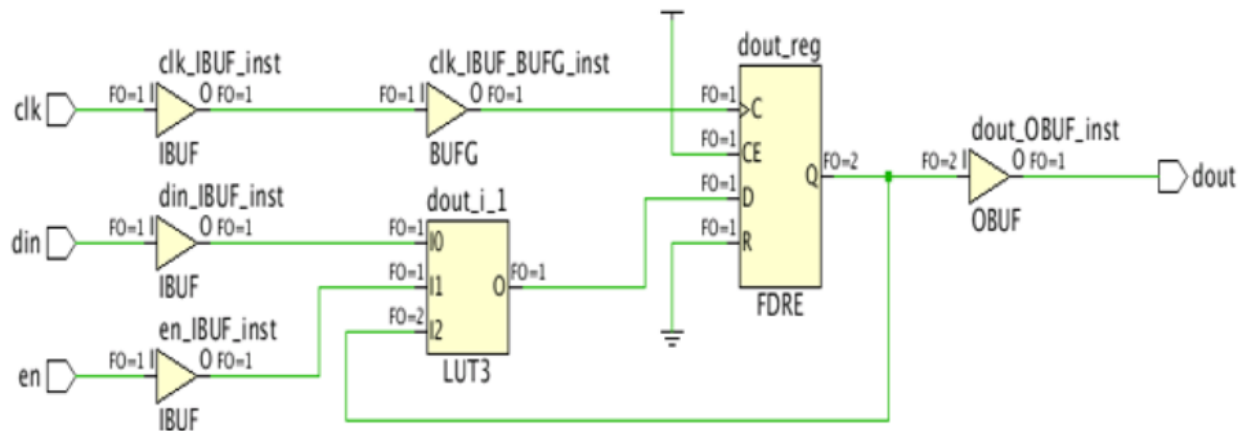


# Проектирование в соответствии с UFDM

## Этап 2: Разработка RTL-модулей (пример)

```
module test
(
input clk,
input en,
input din,
output reg dout
);

always@(posedge clk)
begin
    if(en)
        begin
            dout <= din;
        end
    end
endmodule
```





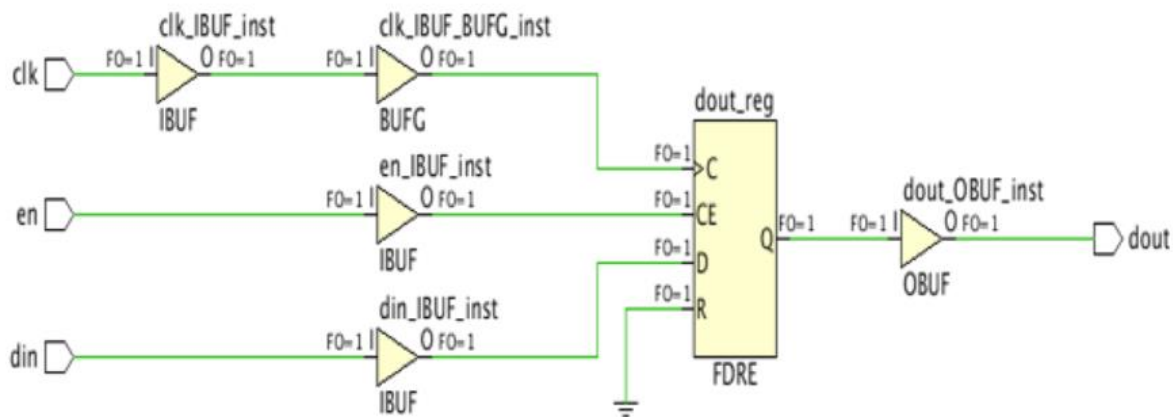
# Проектирование в соответствии с UFDМ

## Этап 2: Разработка RTL-модулей (пример)

```
module test
(
input clk,
(* direct_enable = "true" *) input en,
input din,
output reg dout
);

always@(posedge clk)
begin
if(en)
begin
dout <= din;
end
end

endmodule
```



# Проектирование в соответствии с UFDМ

---

## Этап 2.

### Верификация в Vivado

- Не пренебрегайте верификацией. Как минимум, проведите RTL-верификацию
- Доступен собственный симулятор Xilinx, а так же наиболее популярные сторонние симуляторы (QuestaSim, ActiveHDL ...)
- Руководствуйтесь [ug900](#)



# Проектирование в соответствии с UFDМ

## Этап 3. Задание и проверка констрейнтов (TOP-LEVEL CONSTRAINTS VALIDATION)

Подэтап
Задание базовых констрейнтов (Baselining)
Проверка констрейнтов тайминга



# Проектирование в соответствии с UFDМ

## Этап 3. Задание и проверка констрейнтов (TOP-LEVEL CONSTRAINTS VALIDATION)

### Baselinig

- Baselining – задание простейших констрейнтов, первоначально не учитывая тайминг В/В. Руководствуйтесь [UG949](#) chapter5.
- Начинается с завершения этапа синтеза и заканчивается по завершению этапа “Route”
- Минимальный набор: все тактовые констрейнты и описание пересечения тактовых доменов
- Используйте Timing Constraints wizard
- Руководствуйтесь [UG906](#), [UG903](#), [UG894](#)



# Проектирование в соответствии с UFDМ

---

## Этап 3. Проверка констрейнтов

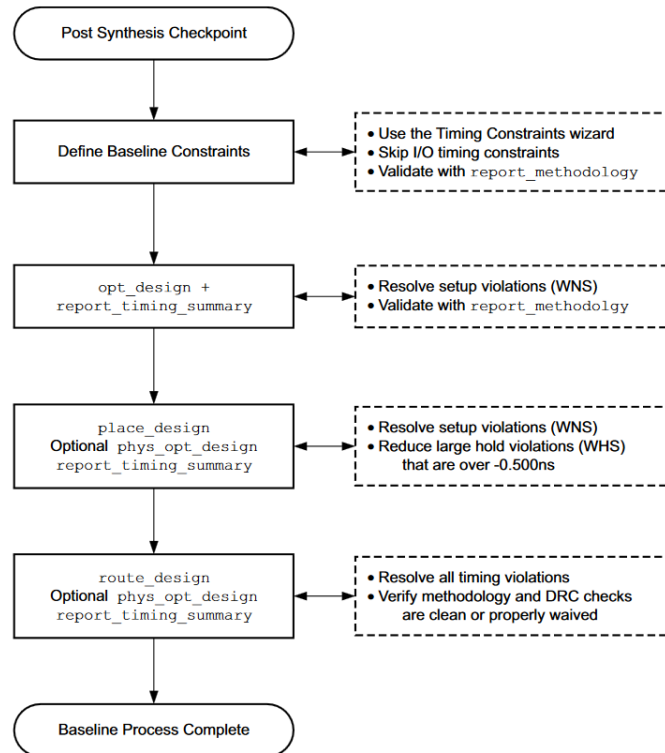
Используйте следующие отчеты:

- Report\_timing\_summary либо check\_timing
- Report\_methodology
- Report\_clock\_interaction
- Report\_cdc
- Report\_exceptions



# Проектирование в соответствии с UFDМ

## Baselining



# Проектирование в соответствии с UFDМ

## Этап 4. Анализ и оптимизация проекта (DESIGN ANALYSIS AND CLOSURE)

### Timing Closure and Power Closure

Подэтап
Выявление причин нарушения тайминга
Уменьшение задержек в логических элементах
Уменьшение задержек в линиях связи между логическими элементами
Уменьшение сдвига (Skew) тактовых сигналов
Уменьшение неопределенности тактовых сигналов (Clock Uncertainty)



# Проектирование в соответствии с UFDМ

## Этап 4. Анализ и оптимизация проекта (DESIGN ANALYSIS AND CLOSURE)

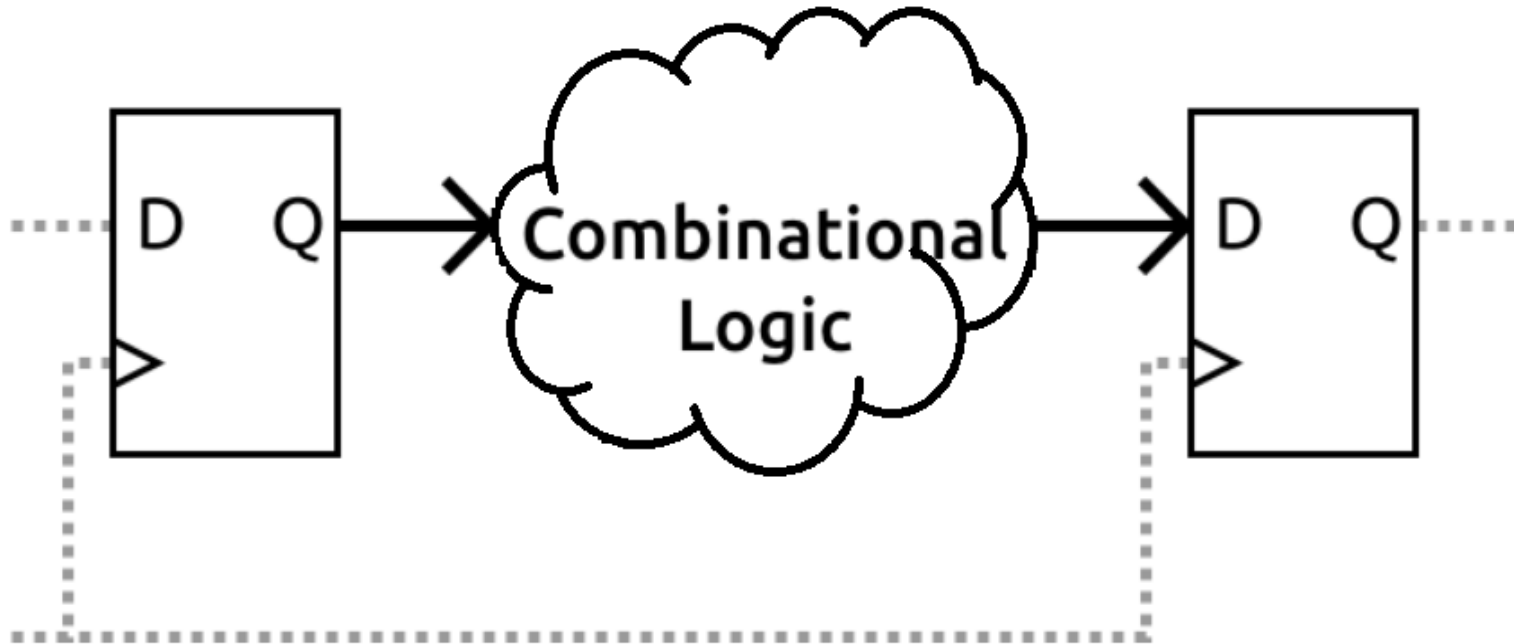
Подэтап
Уменьшение числа управляющих сигналов (Reduce Control Sets)
Оптимизация цепей с высоким фанавтомом
Оптимизация связей (Address Congestion)
Настройка параметров процесса компиляции
Анализ и оптимизация потребляемой мощности





# Проектирование в соответствии с UFDМ

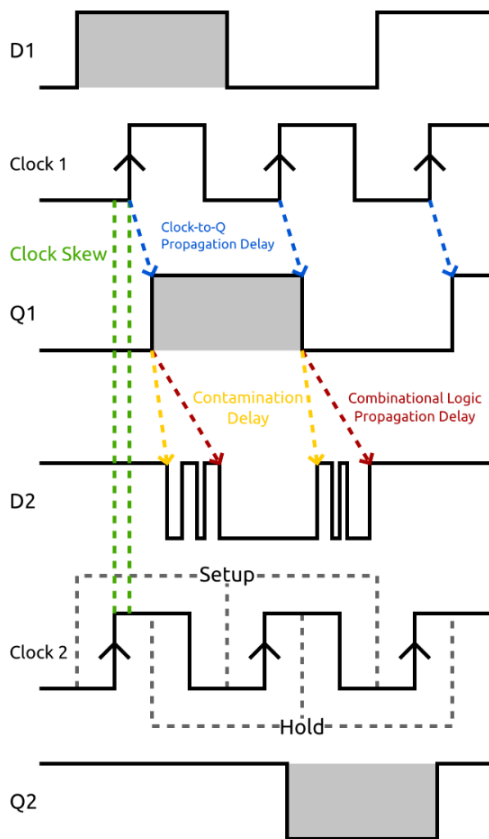
## Сведение тайминга (Timing Closure)



# Проектирование в соответствии с UFDМ

## Тайминг: Основные параметры

[UG612](#)



№	Английский термин	Русский термин
1	Propagation Delay	Задержка распространения
2	Setup Time	Время установки
3	Hold Time	Время удержания
4	Fclk (Tclk=1/Fclk)	Значение тактовой частоты (Период)
5	Clock jitter	Джиттер тактового сигнала
6	Clock Skew	“Перекас” тактового сигнала

[Краткий tutorial здесь:](#)



# Проектирование в соответствии с UFDМ

## Сведение тайминга (Timing Closure)

- Руководствуйтесь [UltraFast Design Methodology Timing Closure Quick Reference Guide \(UG1292\)](#)
- Генерируйте и анализируйте отчеты, согласно с [UG906](#) , находите и убирайте “слаки”

- Utilization Report
- Report DRC
- Report Clock Networks
- Report Clock Interaction
- Report Pulse Width
- Report Timing

- Report Datasheet
- Report Exceptions
- Report Clock Domain Crossings
- Report Bus Skew
- Report Methodology

Критерий завершения – все ограничения выполнены



# Проектирование в соответствии с UFDМ



## Этап 4.

### Анализ и оптимизация потребляемой мощности

- Анализ мощности возможен как в среде Vivado (см. [UG907](#)), так и дополнительными средствами ХРЕ \*) ( см. [UG440](#) - Xilinx Power Estimator User Guide)
- Терминология – см [UG907](#)

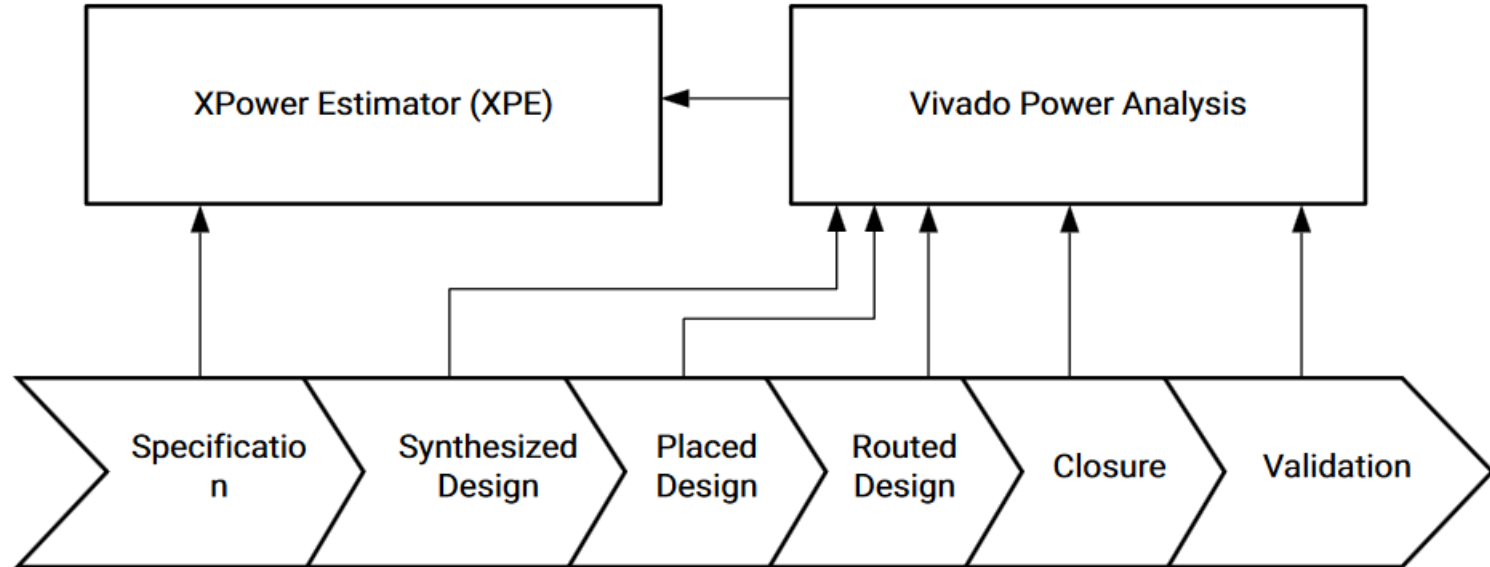
*\*) ХРЕ – приложение для Excel и различается по семействам ПЛИС. Чтобы ХРЕ корректно работало под Windows – установите в своем компьютере американские форматы даты и времени*



# Проектирование в соответствии с UFDМ

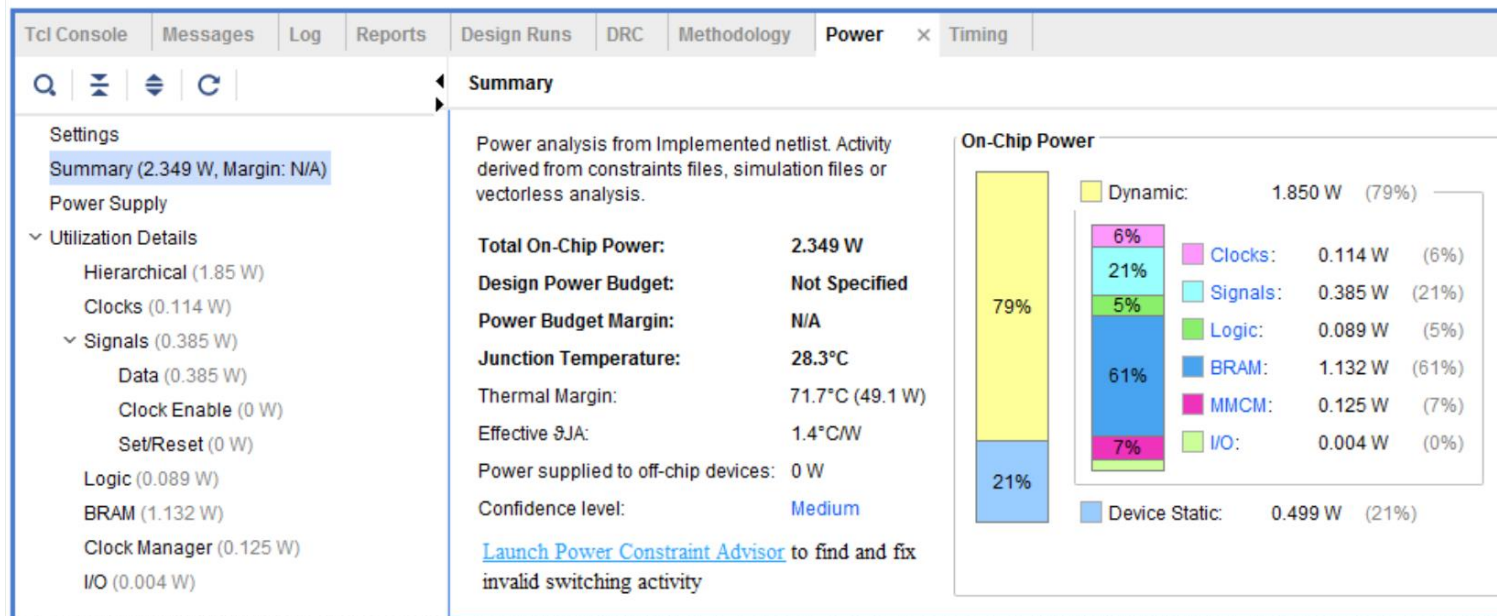
Анализ и оптимизация потребляемой мощности

Диаграмма процесса оценки потребляемой мощности



# Проектирование в соответствии с UFDМ

## Оценка потребляемой мощности в Vivado



## Детальный анализ потребляемой мощности в ХРЕ



# Проектирование в соответствии с UFDМ

---

## Оптимизация потребляемой мощности

- Установите бюджет энергопотребления
- Используйте команду `power_opt` для снижения энергопотребления
- Максимально используйте каскадирование блоков памяти
- Руководствуйтесь [UG907](#), Chapter 6 для уменьшения потребляемой мощности





# Проектирование в соответствии с UFDМ

---

Рекомендации и замечания по выполнению  
отдельных этапов и подэтапов (Tips & Tricks)



# Проектирование в соответствии с UFDМ

---

## Этап 1. BOARD AND DEVICE PLANNING/PCB Design

1. Если разрабатывается единичный экземпляр или мелкая партия – используйте покупные отладочные платы и FMC-модули
2. Если разрабатывается серийное устройство
  - ✓ Возьмите за основу схему и РСВ наиболее подходящей отладочной платы. (Как правило, схема и РСВ выкладывается в открытый доступ)
  - ✓ Руководствуйтесь соответствующими чеклистами



# Проектирование в соответствии с UFDМ

---

## Этап 2. DESIGN ENTRY AND IMPLEMENTATION

### Подэтап “Build and Validate RTL Submodules”

Пункт “Ensure design adheres to RTL coding guidelines”:

Следуйте рекомендациям по кодированию RTL-модулей для Вашего семейства согласно [UG949](#), chapter 3 и [UG901](#), chapter 4.

- Используйте шаблоны из библиотеки шаблонов Vivado
- Загрузите и изучите [ug901-vivado-synthesis-examples](#)



# Проектирование в соответствии с UFDМ

---

## Этап 4. Подэтап “Анализ и оптимизация потребляемой мощности”

- Чем меньше нанометров, тем выше статическое потребление и ниже динамическое (на мегагерцы)



# III. UFDM и СнК

---

## SDK и PetaLinux

- SDK входит в состав Vivado. Отдельной лицензии не требуется
- PetaLinux – полностью бесплатный продукт (поддержка - платная)
- Применение SDK в рамках UFDM: [UG1046](#) (chapter 6), [XTP397](#)



# IV. Высокоуровневое проектирование и UFDM

---

**Vivado HLS**

**SDSoC**

**(Краткий обзор)**

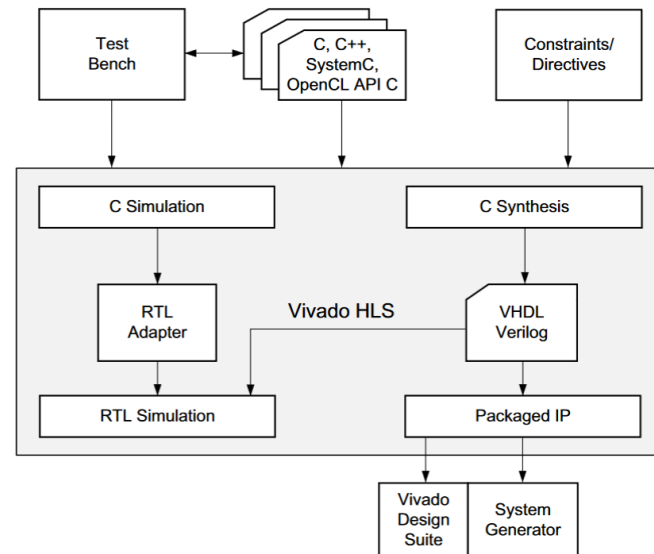


# Высокоуровневое проектирование и UFDМ

## Vivado HLS

- Входит в состав Vivado. Отдельной лицензии не требуется
- Основные документы:

[UG902](#)



# Высокоуровневое проектирование и UFDМ

---

## Vivado HLS

- Разработка ведется на C/C++/System C
- Код транслируется в модули на Verilog/VHDL и затем синтезируется
- Процесс синтеза задается директивами `#pragma`
- Вводятся дополнительные типы данных
- Имеются ограничения, например запрещено динамическое распределение памяти
- Имеется набор библиотек





# Высокоуровневое проектирование и UFDМ

---

## Vivado HLS

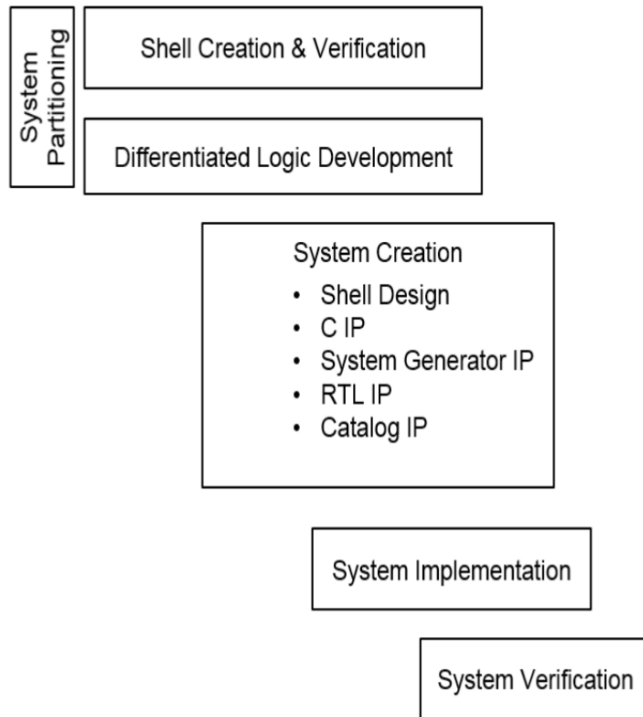
### Доступные библиотеки:

- Arbitrary Precision Data Types Library
- HLS Stream Library
- HLS Math Library
- HLS Video Library
- HLS IP Library
- HLS Linear Algebra Library
- HLS DSP Library



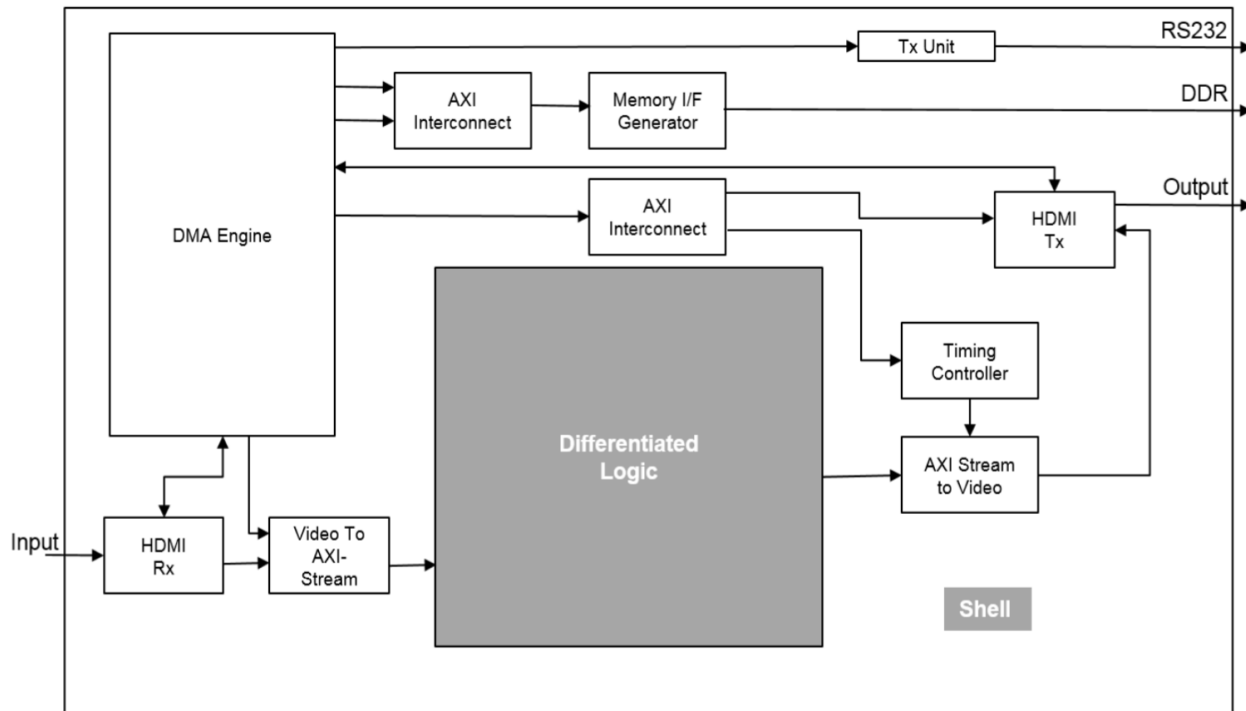
# Высокоуровневое проектирование и UFDМ

## Процесс разработки СнК ([ug1197](#))



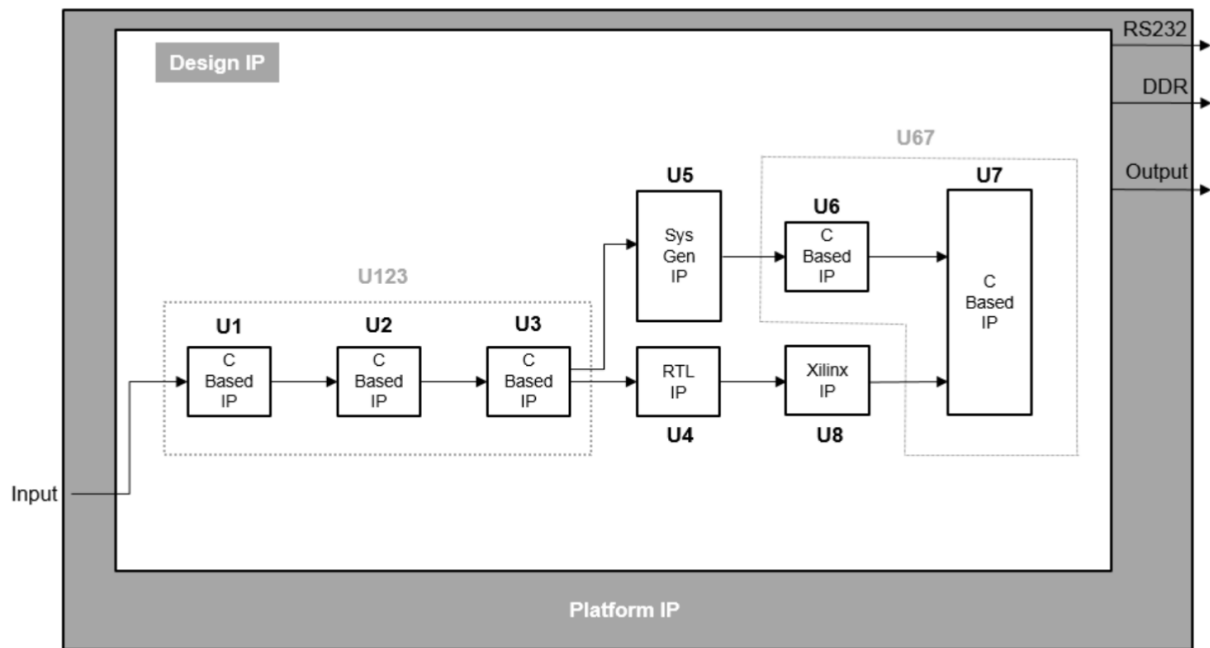
# Высокоуровневое проектирование и UFDМ

Методика: разделить систему на 2 части: **Shell** и Custom Logic IPs



# Высокоуровневое проектирование и UFDМ

Методика: разделить систему на 2 части: Shell и Custom Logic



✓ Эти части можно разрабатывать параллельно



# Высокоуровневое проектирование и UFDМ

---

## SDSoC

- Отдельный продукт с отдельной лицензией
- Основные документы: [UG1046](#) (Chapter 8), [XTP397](#), [UG1197](#)



# Спасибо за внимание!

---

## Компания Макро Групп:

- ✓ **Официальный партнер Xilinx**
- ✓ **Комплексная поставка электронных компонентов**
- ✓ **Техническая поддержка по всем вопросам применения продукции и ПО Xilinx**

## 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)

