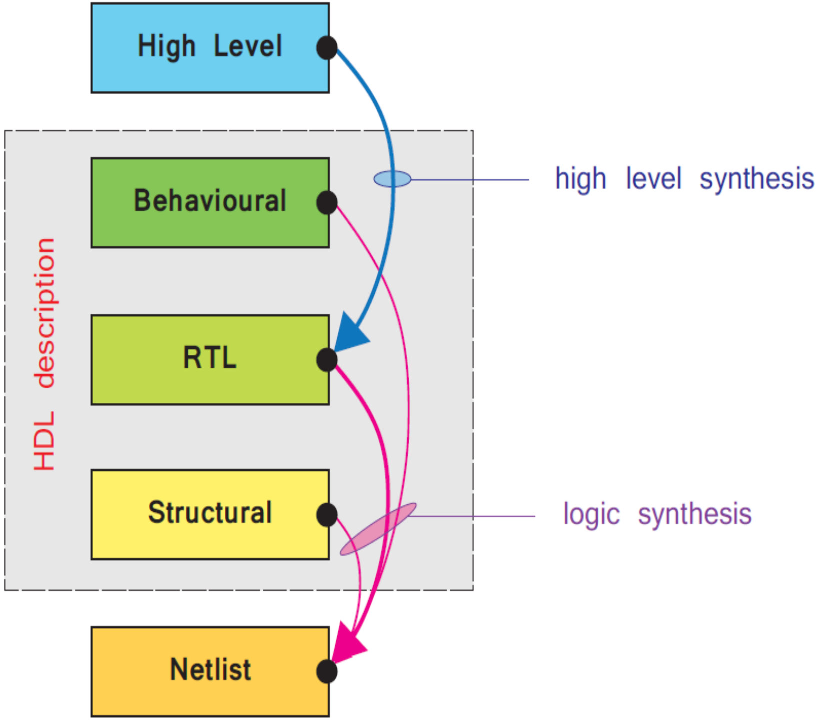
Очень коротко об HLS

Что такое HLS?

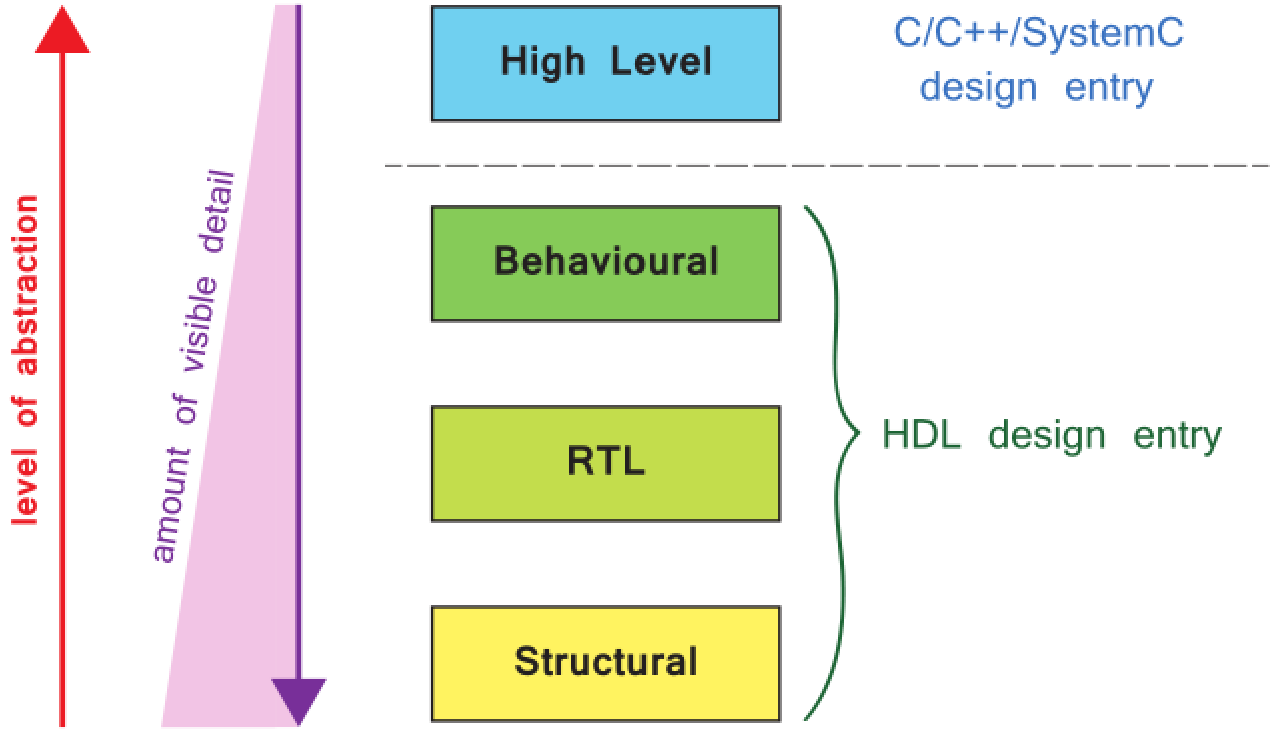
High-**L**evel **S**ynthesis (HLS) - технология, основной задачей которой является генерация качественной реализации аппаратного обеспечения уровня регистровых передач (RTL) из спецификации высокого уровня.



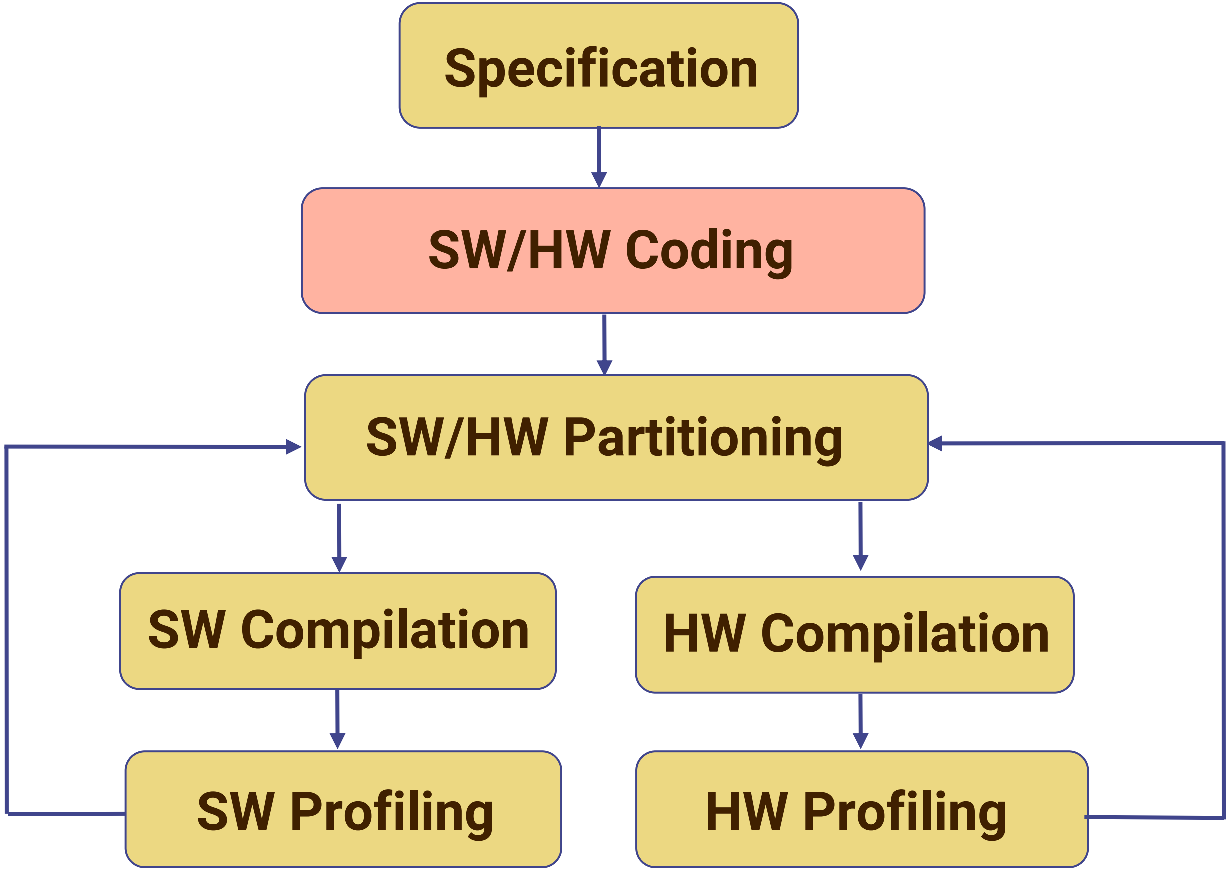
Зачем нужен HLS?

- Higher level of abstraction
- Modeling complex designs
- Reduce design efforts
- Fast turnaround time
- Technology independence
- Ease of HW/SW partitioning

Абстракция


level of abstraction
amount of visible detail
High Level
C/C++/SystemC
design entry
Behavioural
RTL
Structural
HDL design entry

Программно/Аппаратный подход



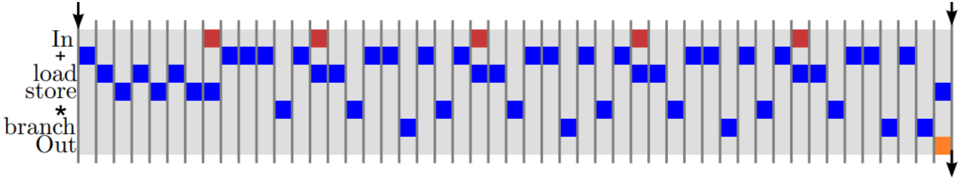
HLS есть у многих

Status	Compiler	Owner	License	Input	Output	Year
	AUGH	TIMA Lab.	Academic	C subset	VHDL	2012
	eXCite	Y Explorations	Commercial	C	VHDL/Verilog	2001
	Bambu	PoliMi	Academic	C	VHDL/Verilog	2012
	Bluespec	BlueSpec Inc.	Commercial	BSV	SystemVerilog	2007
	CHC	Altium	Commercial	C subset	VHDL/Verilog	2008
	CoDeveloper	Impulse Accelerated	Commercial	Impulse-C	VHDL	2003
	HDL Coder	MathWorks	Commercial	MATLAB, Simulink, Stateflow, Simscape	VHDL / Verilog	2003
	Stratus	Cadence	Commercial	C/C++ SystemC	RTL	2015
	CyberWorkbench	NEC	Commercial	BDL, SystemC	VHDL/Verilog	2011
	Catapult	Mentor (Siemens business)	Commercial	C, C++, SystemC	VHDL/Verilog	2004
	DWARV	TU. Delft	Academic	C subset	VHDL	2012
	GAUT	U. Bretagne	Academic	C/C++	VHDL	2010
	Hastlayer	Lombiq Technologies	Commercial	C#/C++/F#... (.NET)	VHDL	2015
	…					
Aban-doned	DEFACTO	U. South Cailf.	Academic	C	RTL	1999
	Garp	U. Berkeley	Academic	C subset	bitstream	2000
	MATCH	U. Northwest	Academic	MATLAB	VHDL	2000
	Napa-C	Sarnoff Corp.	Academic	C subset	VHDL/Verilog	1998
	PipeRench	U.Carnegie M.	Academic	DIL	bistream	2000
	SA-C	U. Colorado	Academic	SA-C	VHDL	2003
	SeaCucumber	U. Brigham Y.	Academic	Java	EDIF	2002
	SPARK	U. Cal. Irvine	Academic	C	VHDL	2003

CPU/DSP vs FPGA

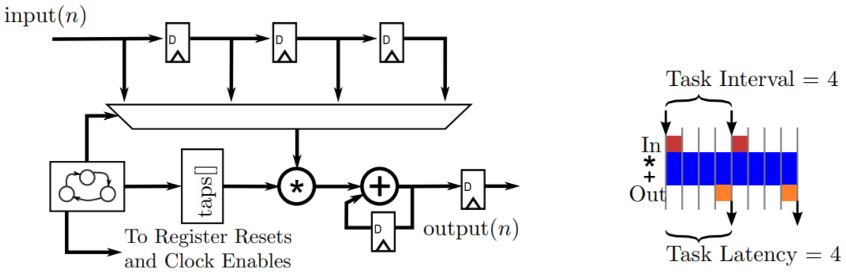
```
#define NUM_TAPS 4

void fir(int input, int *output, int taps[NUM_TAPS])
{
  static int delay_line[NUM_TAPS] = {};

  int result = 0;
  for (int i = NUM_TAPS − 1; i > 0; i−−) {
    delay_line[i] = delay_line[i − 1];
  }
  delay_line[0] = input;

  for (int i = 0; i < NUM_TAPS; i++) {
    result += delay_line[i] * taps[i];
  }

  *output = result;
}
```

CPU



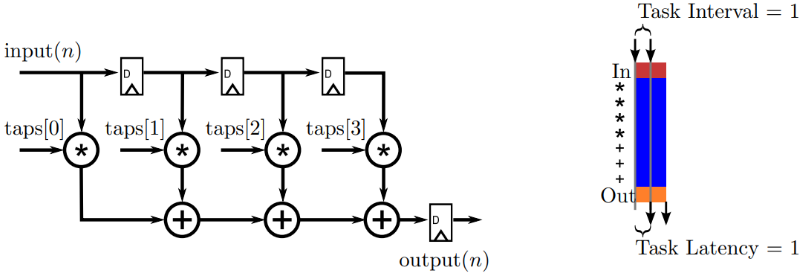
FPGA

последовательная реализация



FPGA

параллельная реализация



Целевые направления HLS

Aerospace and Defense
- Radar, Sonar
- Signals Intelligence

Communications
- LTE MIMO receiver
- Advanced wireless antenna positioning

Industrial, Scientific, Medical
- Ultrasound systems
- Motor controllers

Audio, Video, Broadcast
- 3D cameras
- Video transport

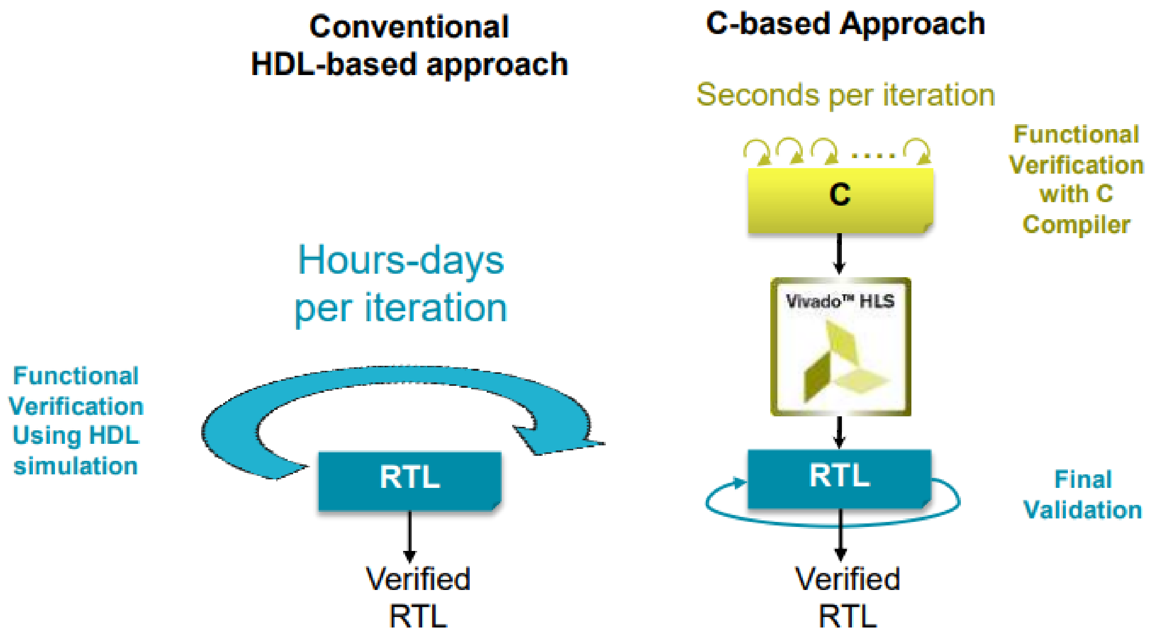
Automotive
- Infotainment
- Driver assistance

Consumer
- 3D television
- eReaders

Test & Measurement
- Communications instruments
- Semiconductor ATE

Computing & Storage
- High performance computing
- Database acceleration

Сокращаем время разработки



Optical flow video example

Input	RTL Simulation Time	C Simulation Time	Acceleration
10 frames of video data	~2 days	10 seconds	~12,000X

*RTL Simulations performed using ModelSim

Оптимизированные 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
- OpenCV libraries
- + множество других библиотек

Сокращаем время разработки

"In an **HDL** design, each scenario would likely cost an **additional day of writing code** …

With Vivado **HLS** these changes **took minutes**"

Nathan Jachimiec, R&D Engineer, Agilent Technologies

"I was able to design complex linear algebra algorithms **10x faster** than before with VHDL, and yet achieved **better QoR** with Vivado HLS."

Design Engineer, Major A&D contractor

"..we always use **C** to quickly build a **system-level model** for validation of **key algorithms**.. **problem** .. **quickly and efficiently convert C into a HDL**".

"With Xilinx Vivado HLS, …. … used **C to implement a key algorithm** … **into Verilog**. We **verified** both the functionality and performance in **Xilinx devices** … "

***Hengqi Liu**,Central R&D Data Center CTO, ZTE Inc.*

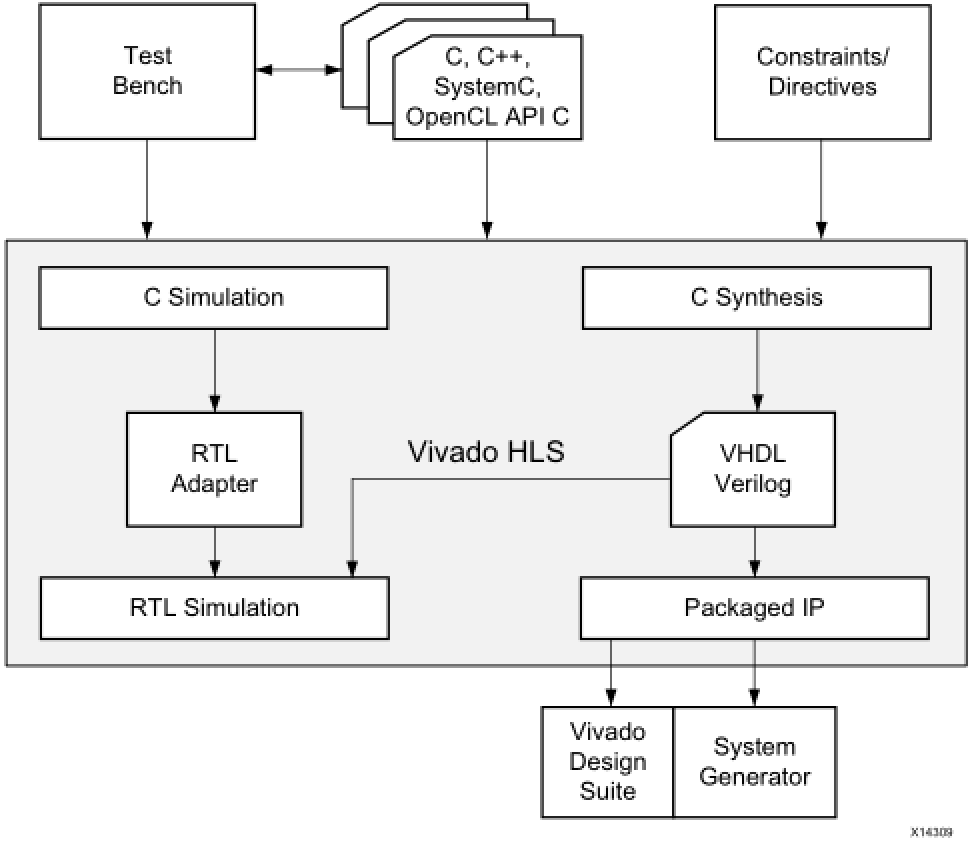
Radar Design 1024 x 64 QRD Floating-Point data path	Conventional Hand-coded HDL Approach	Using Vivado High Level Synthesis
Design Language	VHDL (RTL)	C
Design Time (weeks)	12	1
Latency (ms)	37	21
Memory (RAMB18E1)	134 (16%)	10 (1%)
Memory (RAMB36E1)	273 (65%)	138 (33%)
Registers	29686 (9%)	14263 (4%)
LUTs	28152 (18%)	24257 (16%)

***Source:** Design Engineer at Major A&D contractor*

"For each project where we used Vivado HLS, we **saved 2-3 weeks** of engineering **time**."

CTO, Major broadcast equipment company

Маршрут проектирования



Интерпретация функции

- **Functions**
  - Functions define hierarchy and control regions
- **Function Parameters:**
  - Define the RTL I/O Ports
- **Types:**
  - Data types define bitwidth requirements
  - HLS optimizes bitwidth except for function parameters
- **Loops:**
  - Define iterative execution regions that can share HW resources
- **Arrays:**
  - Main way of defining memory and data storage
- **Operators:**
  - Implementations optimized for performance
  - Automatically shared where possible to reduce area

```
void fir (
  data_t *y,
  coef_t c[4],
  data_t x
  ) {

  static data_t shift_reg[4];
  acc_t acc;
  int i;

  acc=0;
  loop: for (i=3;i>=0;i--) {
    if (i==0) {
      acc+=x*c[0];
      shift_reg[0]=x;
    } else {
      shift_reg[i]=shift_reg[i-1];
      acc+=shift_reg[i] * c[i];
}
  }
  *y=acc;
}
```

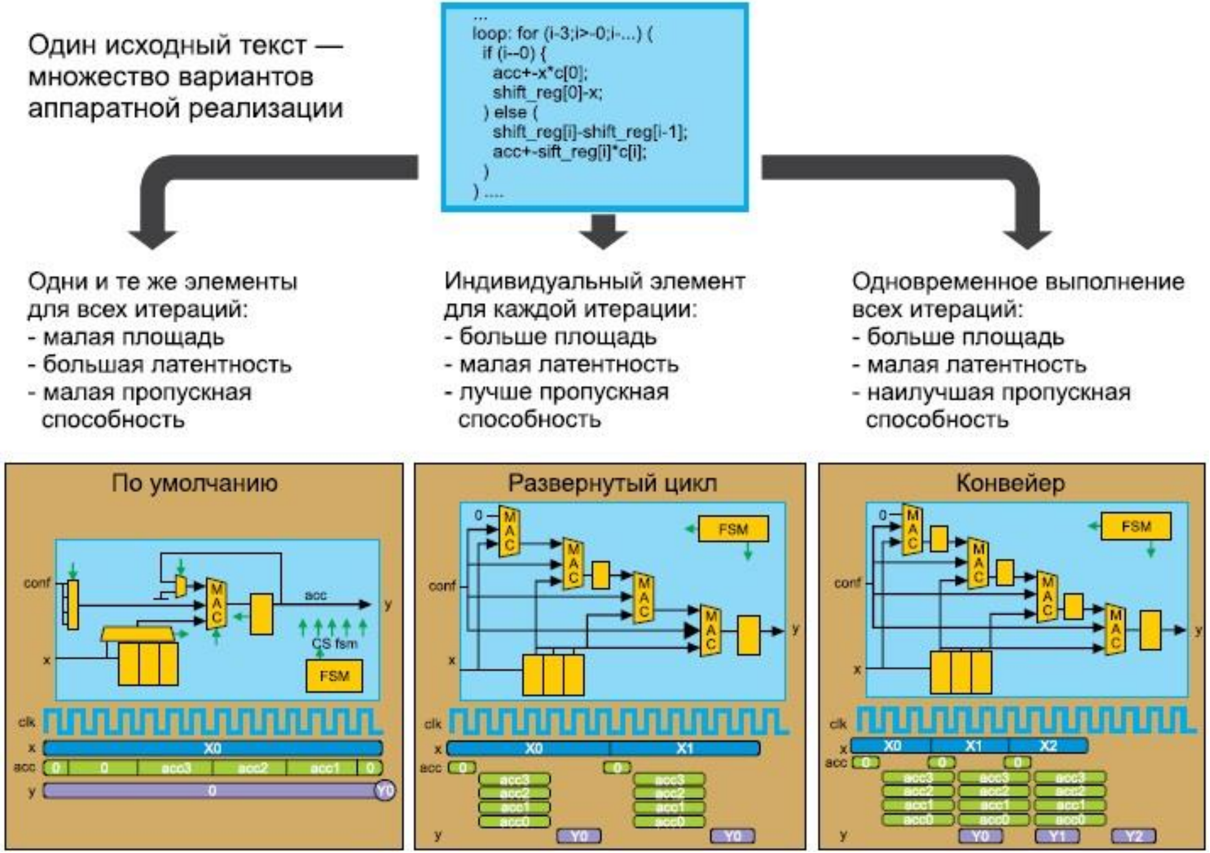
Интерфейс HDL модуля

Argument Type	Scalar		Array			Pointer or Reference			HLS:: Stream
Interface Mode	**Input**	**Return**	**I**	**I/O**	**O**	**I**	**I/O**	**O**	**I and O**
ap_ctrl_none									
ap_ctrl_hs		D							
ap_ctrl_chain									
axis									
s_axilite									
m_axi									
ap_none	D					D			
ap_stable									
ap_ack									
ap_vld								D	
ap_ovld							D		
ap_hs									
ap_memory			D	D	D				
bram									
ap_fifo									D
ap_bus									

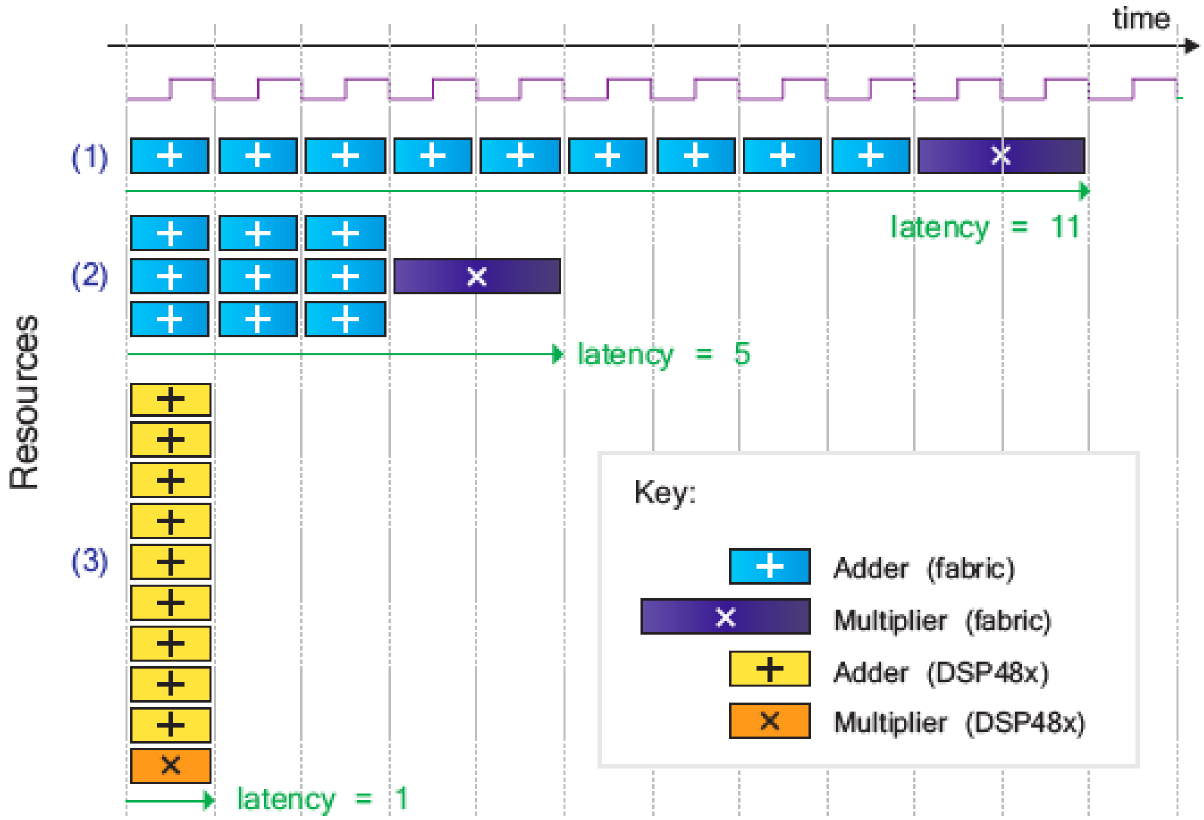
Supported D = Default Interface Not Supported

X14293

Один код – несколько реализаций

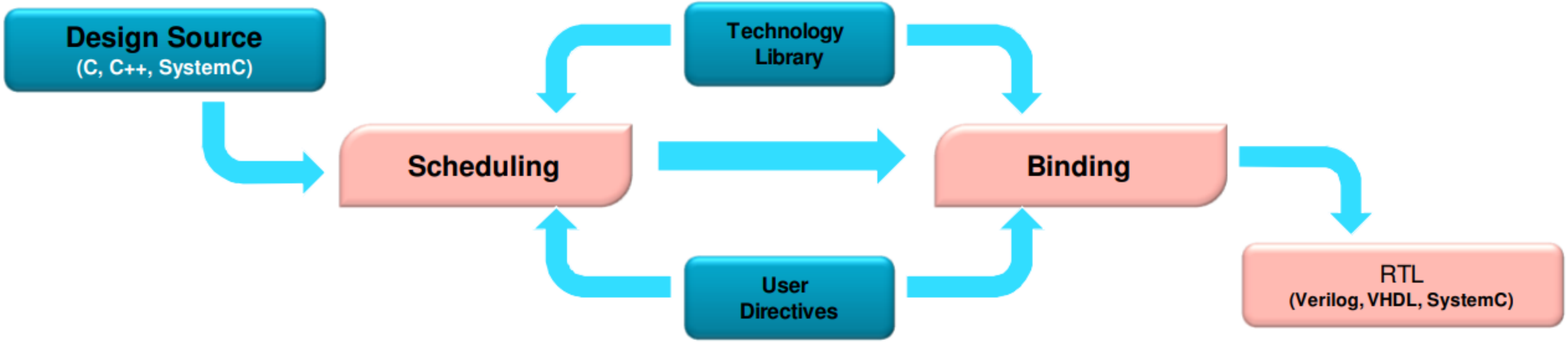


Баланс между ресурсами и задержкой

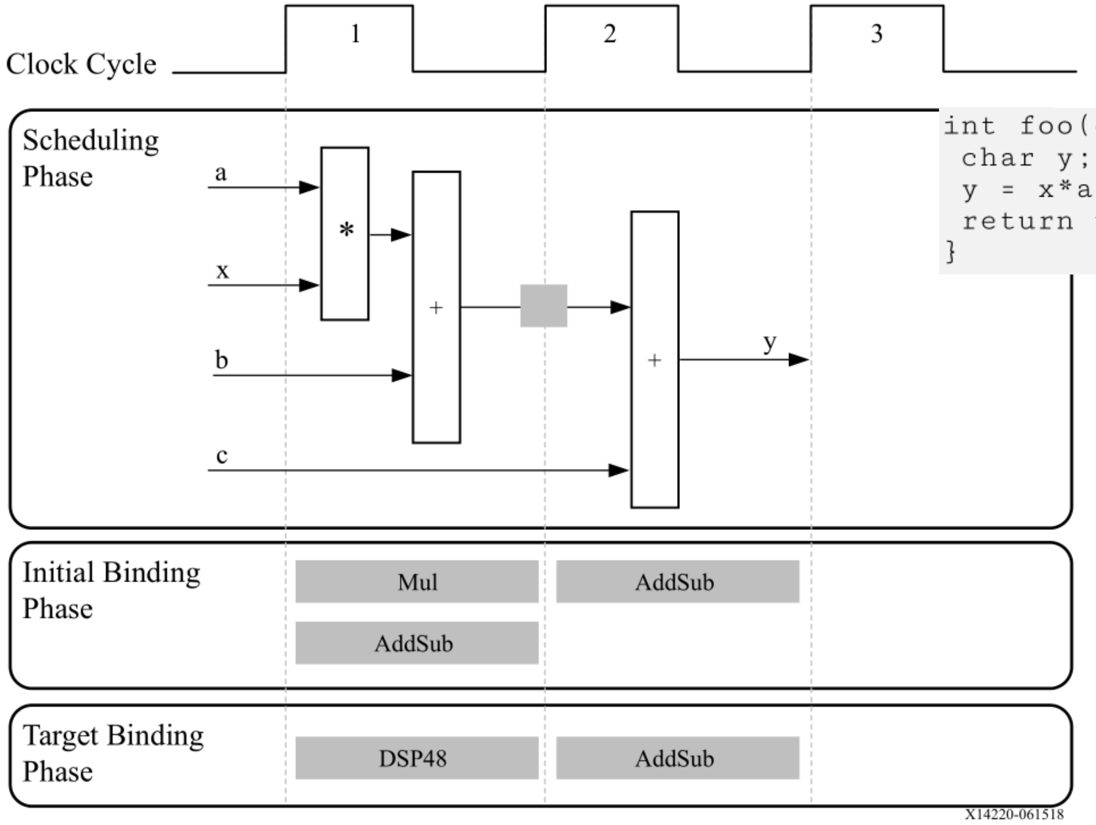


Два ключевых этапа

- **Scheduling & Binding**
  - Scheduling and Binding are at the heart of HLS
- **Scheduling determines in which clock cycle an operation will occur**
  - Takes into account the control, dataflow and user directives
  - The allocation of resources can be constrained
- **Binding determines which library cell is used for each operation**
  - Takes into account component delays, user directives



Scheduling & Binding




```
int foo(char x, char a, char b, char c) {
 char y;
 y = x*a+b+c;
 return y;
}
```

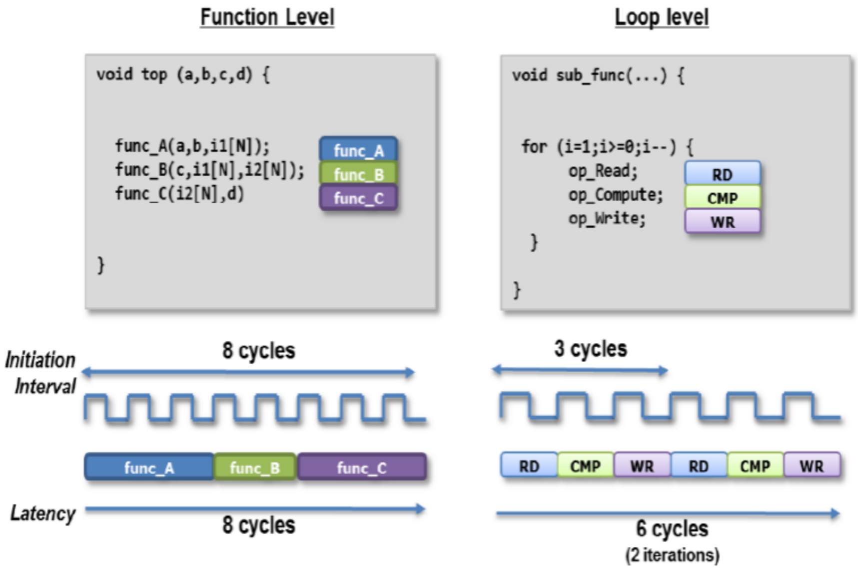
Три ключевых определения

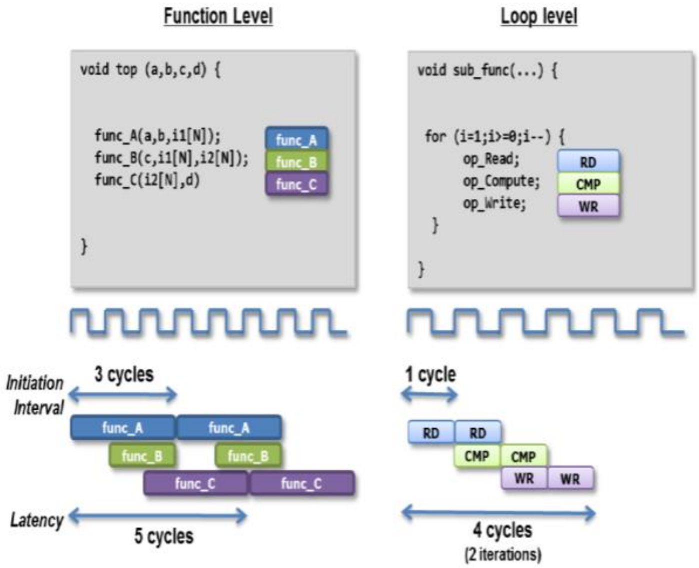
- Area: Amount of hardware resources required to implement the design based on the resources available in the FPGA, including look-up tables (LUT), registers, block RAMs, and DSP48s.

- Latency: Number of clock cycles required for the function to compute all output values.

- Initiation interval (II): Number of clock cycles before the function can accept new input data

Initiation Interval & Latency





Два способа управления синтезом

Using Tcl Commands or Embedded Pragmas

In the Vivado HLS Directive Editor dialog box, you can specify either of the following Destination settings:

- **Directive File**: Vivado HLS inserts the directive as a Tcl command into the file directives.tcl in the solution directory.
- **Source File**: Vivado HLS inserts the directive directly into the C source file as a pragma.

The following table describes the advantages and disadvantages of both approaches.

Table 2: **Tcl Commands Versus Pragmas**

Directive Format	Advantages	Disadvantages
Directives file (Tcl Command)	Each solution has independent directives. This approach is ideal for design exploration. If any solution is re-synthesized, only the directives specified in that solution are applied.	If the C source files are transferred to a third-party or archived, the directives.tcl file must be included. The `directives.tcl` file is required if the results are to be re-created.
Source Code (Pragma)	The optimization directives are embedded into the C source code. Ideal when the C sources files are shipped to a third-party as C IP. No other files are required to recreate the same results. Useful approach for directives that are unlikely to change, such as TRIPCOUNT and INTERFACE.	If the optimization directives are embedded in the code, they are automatically applied to every solution when re-synthesized.

Из C/C++ в HLS

```c
#include "matrixmul.h"

void matrixmul(
        mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
        mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
        result_t res[MAT_A_ROWS][MAT_B_COLS])
{
  // Iterate over the rows of the A matrix
   Row: for(int i = 0; i < MAT_A_ROWS; i++) {
      // Iterate over the columns of the B matrix
      Col: for(int j = 0; j < MAT_B_COLS; j++) {
         res[i][j] = 0;
         // Do the inner product of a row of A and col of B
         Product: for(int k = 0; k < MAT_B_ROWS; k++) {
            res[i][j] += a[i][k] * b[k][j];
         }
      }
   }

}
```

Обычный C/C++ код
Синтезируемый, но не оптимальный
по II и Latency

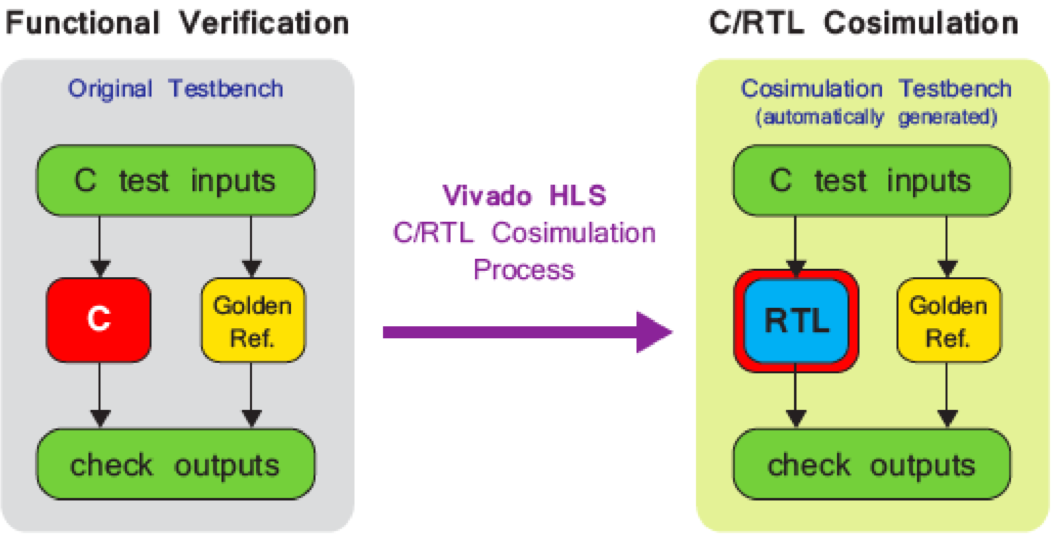
```c
#include "matrixmul.h"

void matrixmul(
      mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
      mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
      result_t res[MAT_A_ROWS][MAT_B_COLS])
{
#pragma HLS ARRAY_RESHAPE variable=b complete dim=1
#pragma HLS ARRAY_RESHAPE variable=a complete dim=2
#pragma HLS INTERFACE ap_fifo port=a
#pragma HLS INTERFACE ap_fifo port=b
#pragma HLS INTERFACE ap_fifo port=res
  mat_a_t a_row[MAT_A_ROWS];
  mat_b_t b_copy[MAT_B_ROWS][MAT_B_COLS];
  int tmp = 0;

  // Iterate over the rowa of the A matrix
  Row: for(int i = 0; i < MAT_A_ROWS; i++) {
    // Iterate over the columns of the B matrix
    Col: for(int j = 0; j < MAT_B_COLS; j++) {
#pragma HLS PIPELINE rewind
      // Do the inner product of a row of A and col of B
      tmp=0;
      // Cache each row (so it's only read once per function)
      if (j == 0)
        Cache_Row: for(int k = 0; k < MAT_A_ROWS; k++)
          a_row[k] = a[i][k];

       // Cache all cols (so they are only read once per function)
     if (i == 0)
            Cache_Col: for(int k = 0; k < MAT_B_ROWS; k++)
               b_copy[k][j] = b[k][j];

      Product: for(int k = 0; k < MAT_B_ROWS; k++) {
        tmp += a_row[k] * b_copy[k][j];
      }
      res[i][j] = tmp;
    }
  }
}
```

Оптимизированный C/C++ код для FPGA

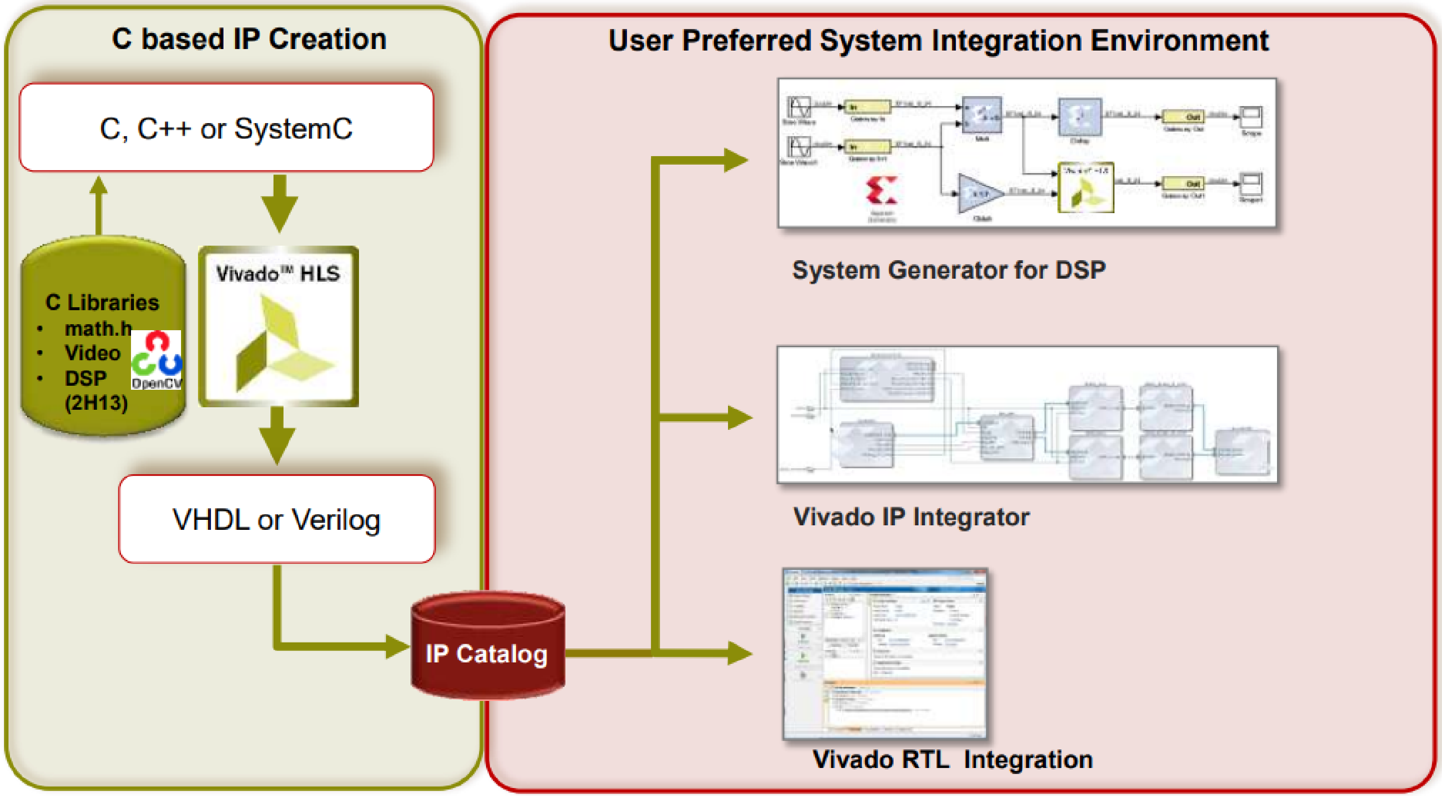
Методология оптимизации

Simulate Design	• Validate The C function
Synthesize Design	• Baseline design
1: Initial Optimizations	• Define interfaces (and data packing) • Define loop trip counts
2: Pipeline for Performance	• Pipeline and Dataflow
3: Optimize Structures for Performance	• Partition memories and ports • Remove false dependencies
4: Reduce Latency	• Optionally specify latency requirements
5: Improve Area	• Optionally recover resources through sharing

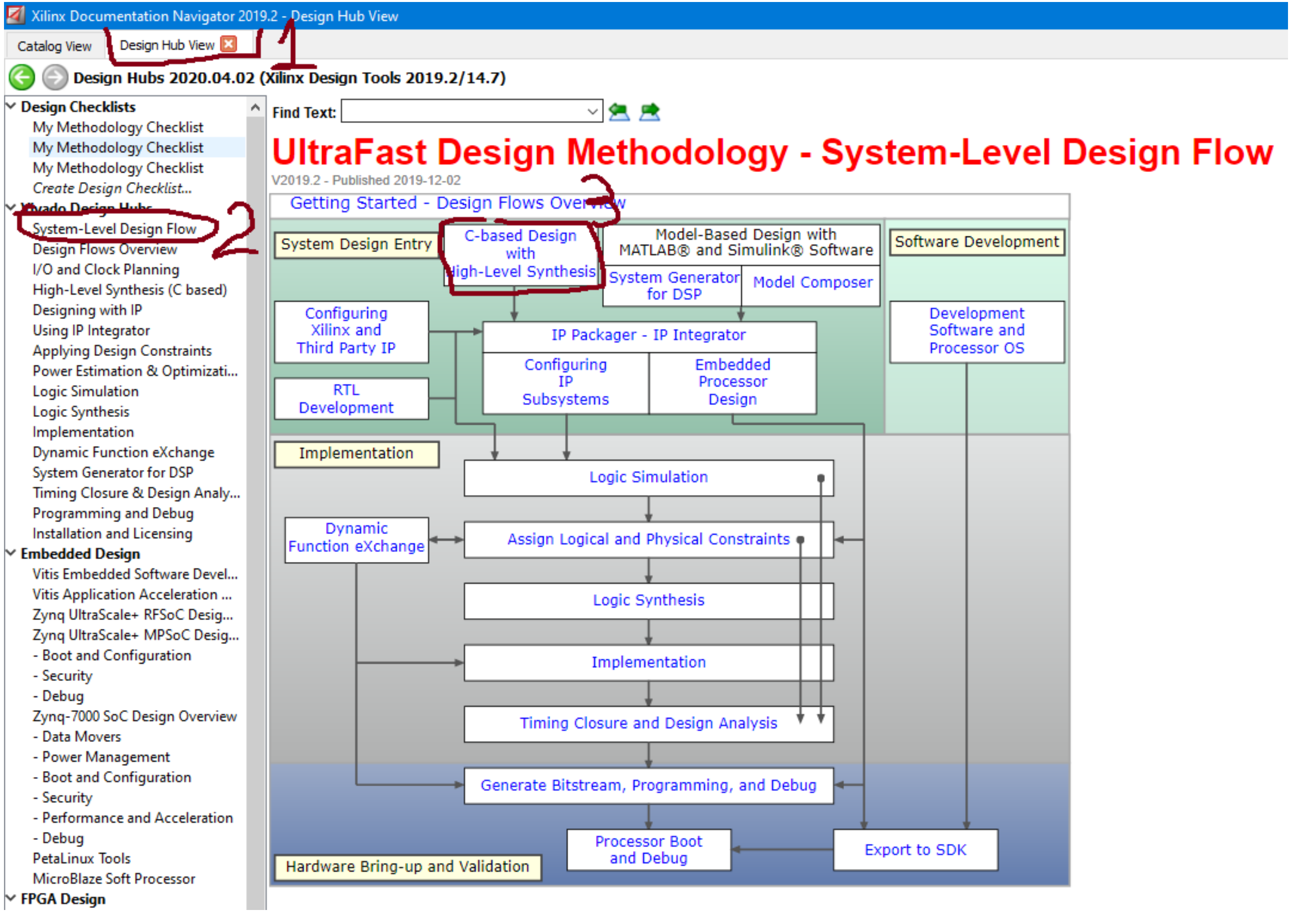
Верификация проекта



Сквозное проектирование в Vivado



Documentation Navigator



Литература

- Xilinx UG902 - High-Level Synthesis
- Xilinx UG871 - High-Level Synthesis: Tutorial - Лабораторные
- Xilinx UG1197 - UltraFast High-Level Productivity Design Methodology Guide
- Parallel Programming for FPGAs – **must have!** (бесплатная книга)
- ZynqBook – бесплатная книга
- Курсы по Vivado HLS – plis2.ru – Авторизованный тренинг центр Xilinx
- Статьи Adam Taylor на hackster.io

Присоединяйся

Youtube: www.youtube.com/c/fpgasystems

Сайт: www.fpga-systems.ru

Telegram: @fpgasystems

E-mail: fpga-systems@yandex.ru

Дистрибьютор **Xilinx**: **www.inline-ctc.ru**

Twitch: www.twitch.tv/fpgasystems

Facebook: www.facebook.com/groups/fpgasystems.ru/

Тренинг центр **Xilinx**: **www.plis2.ru**

Twitter: @fpgasystems

VK: vk.com/club185679360

Заголовок слайда

Текстовый блок