

Презентация для вебинара

“ Основы верификации и отладки ПЛИС XILINX. на примере отладочной платы ARTY ”

Владимир Викулин, инженер по применению Xilinx
Vladimir.Vikulin@macrogroup.ru



**МАКРО
ГРУПП**

2017

Компания



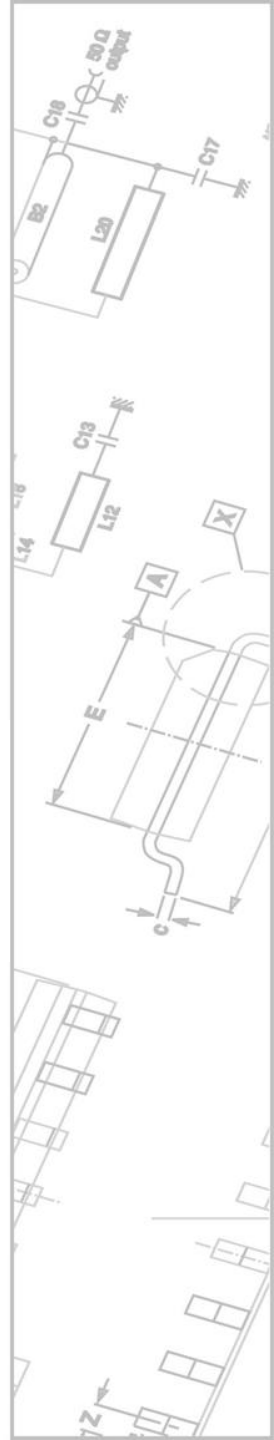
**МАКРО
ГРУПП**

Официальный партнер



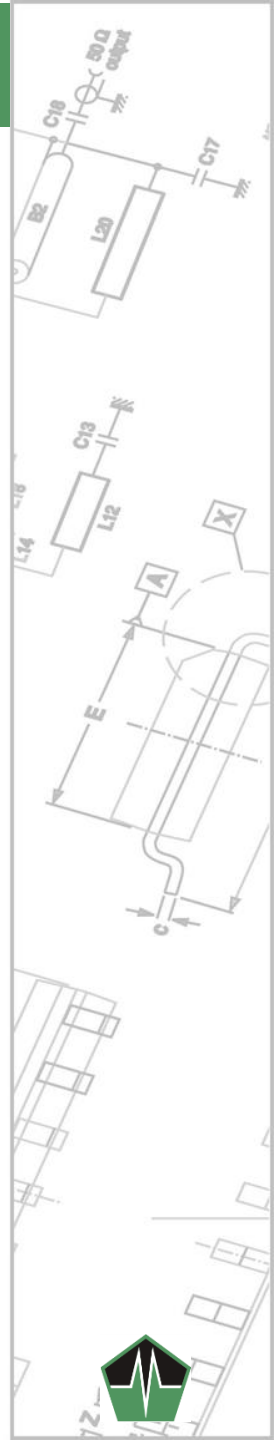
XILINX®

В РОССИИ



Программа вебинара

1. Верификация и отладка – что это такое ?
2. Отладка на аппаратуре в среде Xilinx Vivado
 - Общее представление
 - Практическая демонстрация
3. Симуляция
 - Общее представление
 - Где искать дополнительную информацию
4. Симуляция проекта в среде Xilinx Vivado (с практической демонстрацией)
5. Вопросы - ответы

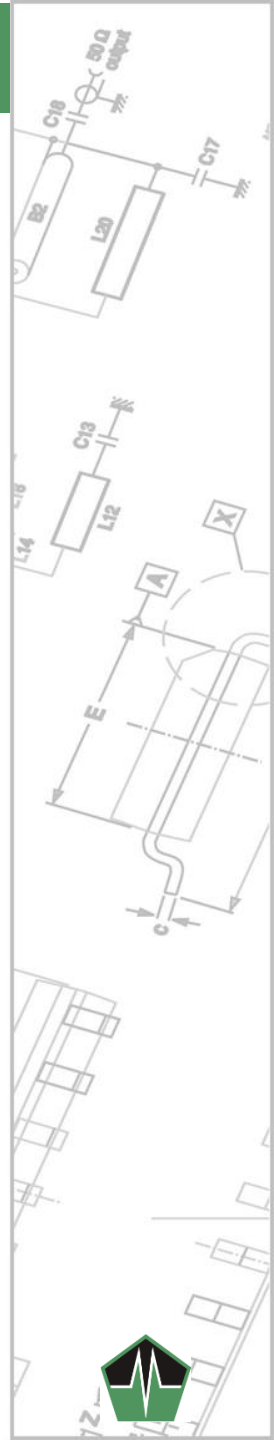


1. Верификация и отладка – что это такое (1)

Верификация – термин многозначный.

Применительно к ПЛИС мы будем понимать его как проверку (доказательство) работоспособности проектируемого устройства в соответствие со спецификацией.

Важным этапом верификации является **симуляция** – проверка созданного HDL-кода (а так же нетлистов, созданных на его основе) с помощью компьютерного моделирования работы системы



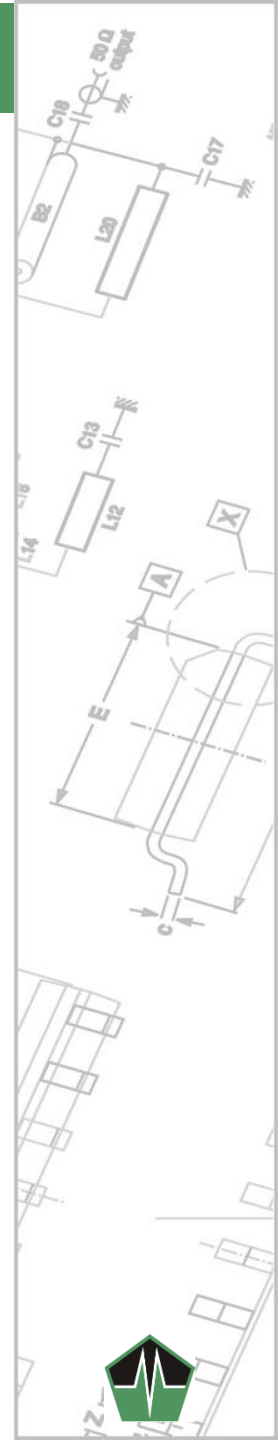
1. Верификация и отладка – что это такое (2)

Отладка – поиск ошибок и их устранение.

Отладка может вестись как непосредственно на аппаратуре, так и на моделях – на этапе симуляции.

Мы рассмотрим обе эти возможности

Замечание: Несмотря на различие, термины **верификация** и **отладка** часто употребляются как синонимы.



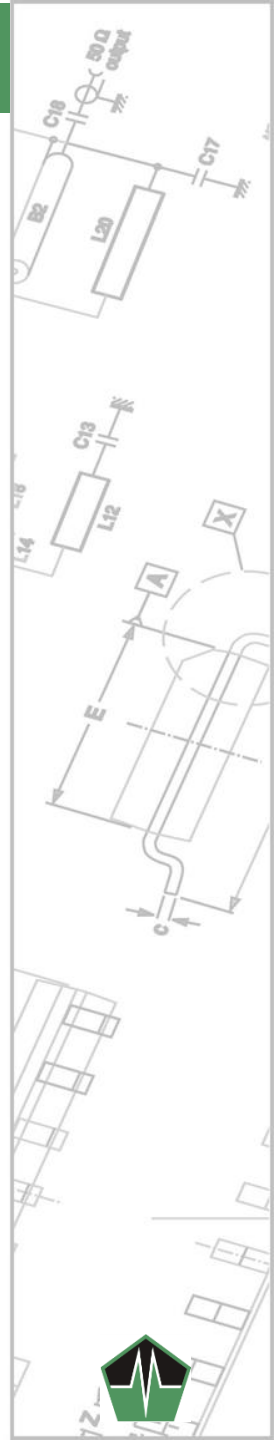
2. Отладка на аппаратуре в Xilinx Vivado (1)

Общее представление

1. Отладчик интегрирован в Vivado
2. Связь с аппаратурой производится через jtag-интерфейс
3. Имеется режим удаленной отладки через сеть: Hardware Server/XVC)
4. В аппаратуру имплементируются специальные ядра
5. Имеются несколько способов имплементации таких ядер:
 - Явно – на этапе кодирования
 - Автоматически – с помощью Wizard_a после завершения этапа синтеза
 - Через констрейнты

Больше информации: <https://www.xilinx.com/products/design-tools/vivado/debug.html>

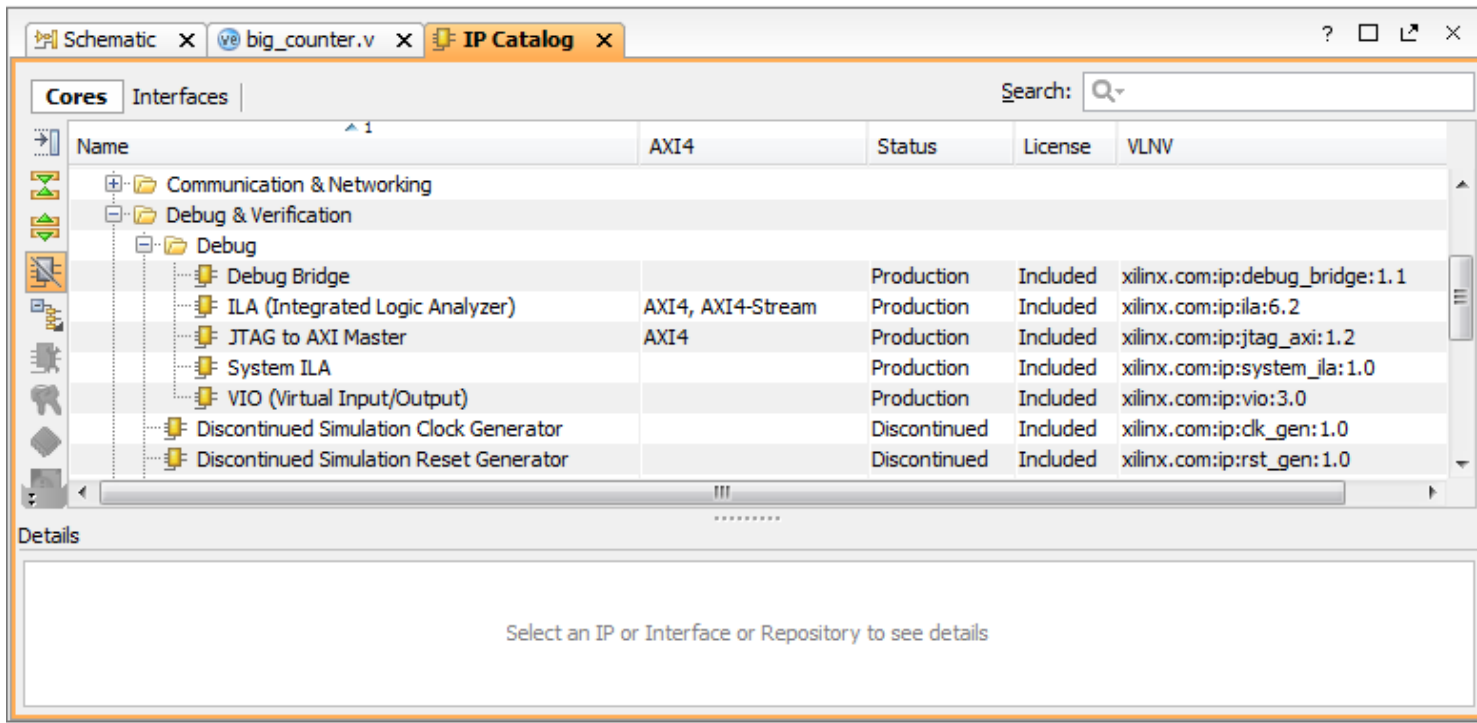
[Vivado Programming and Debugging User Guide \(UG908\)](#)



2. Отладка на аппаратуре в Xilinx Vivado (2)

Отладочные ядра

- ILA – Internal Logic Analyzer
- VIO – Virtual Input/Output
- Jtag AXI Master
- IBERT
- Serial I/O Analyzer
- Memory IP - Калибровка динамической памяти



The screenshot shows the Xilinx Vivado IP Catalog window. The 'Cores' tab is active, and the 'Debug & Verification' category is expanded. The following table lists the available cores:

Name	AXI4	Status	License	VLNV
Communication & Networking				
Debug & Verification				
Debug				
Debug Bridge		Production	Included	xilinx.com:ip:debug_bridge:1.1
ILA (Integrated Logic Analyzer)	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip:ila:6.2
JTAG to AXI Master	AXI4	Production	Included	xilinx.com:ip:jtag_axi:1.2
System ILA		Production	Included	xilinx.com:ip:system_ila:1.0
VIO (Virtual Input/Output)		Production	Included	xilinx.com:ip:vio:3.0
Discontinued Simulation Clock Generator		Discontinued	Included	xilinx.com:ip:clk_gen:1.0
Discontinued Simulation Reset Generator		Discontinued	Included	xilinx.com:ip:rst_gen:1.0

Details

Select an IP or Interface or Repository to see details



2. Отладка на аппаратуре в Xilinx Vivado (3)

Практический пример

The screenshot shows the Xilinx Vivado 2016.3 IDE interface. The main window displays a synthesized design for a circuit named 'big_counter.v'. The circuit schematic includes several components: 'btn_IBUF[0]_inst', 'btn_IBUF[3]_inst', 'l_sys_dk_IBUF_inst', 'l_sys_dk_IBUF_BUFG_inst', 'pause_d1_reg', 'rst_d1_reg', 'pause_reg', 'rst_reg', 'big_counter_1', and 'led_OBUF[0]_inst' through 'led_OBUF[3]_inst'. The 'Flow Navigator' on the left side of the interface has the 'Synthesis' section expanded, and the 'Set Up Debug' option is highlighted with a red circle. The 'Debug' window at the bottom shows a table with the following structure:

Name	Driver Cell	Driver Pin	Probe Type
Unassigned Debug Nets (0)			



2. Отладка на аппаратуре в Xilinx Vivado (4)

Отладочные сигналы задаются на этапе синтеза с помощью специального мастера

The screenshot shows the Xilinx Vivado 2016.3 interface. The 'Set Up Debug' dialog box is open, displaying a table of nets to be debugged. The table has columns for Name, Clock Domain, Driver Cell, and Probe Type. The 'big_counter_I/rst' net is selected. The background shows the 'Synthesized Design' window with a netlist and the 'Debug' window.

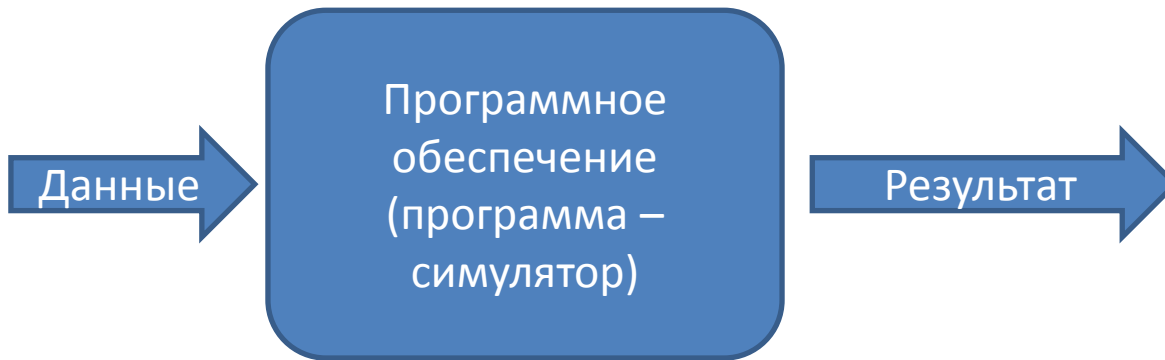
Name	Clock Domain	Driver Cell	Probe Type
big_counter_I/CL_reg (16)	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [0]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [1]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [2]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [3]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [4]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [5]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [6]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [7]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [8]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/CH_reg_n_0 [9]	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/PE	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/pause	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger
big_counter_I/rst	i_sys_clk_IBUF_BUF	FDRE	Data and Trigger

3. Симуляция (1)

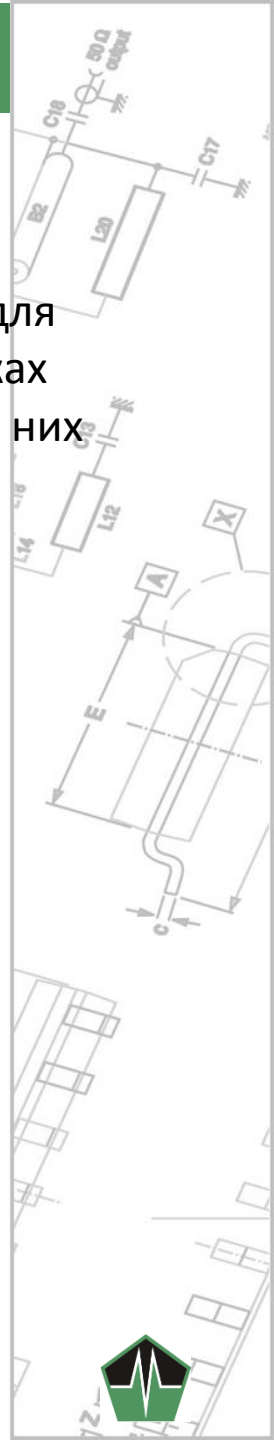
Общее представление

Симулятор – специальная программа, предназначенная для моделирования поведения систем, описываемых на языках высокого уровня ([System]Verilog, VHDL) и полученных из них нетлистов

- HDL-код (или нетлисты)
- Тестовые воздействия
- Библиотеки элементов и пр.



Для сложных проектов симуляция – необходимый этап, она требует больших ресурсов и может занимать очень много времени



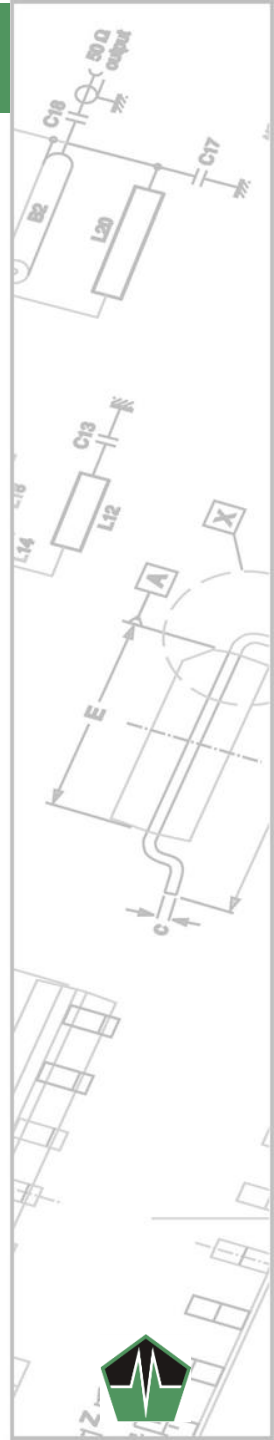
3. Симуляция (2)

ПО для симуляции

Симуляторы для ПЛИС Xilinx:

- Собственный встроенный в Vivado симулятор от Xilinx
- Универсальные симуляторы от сторонних разработчиков
 - QuestaSim/Modelsim от MentorGraphics
 - Nc-sim от Cadence Design Systems
 - Active-HDL от компании Aldec
 - VCS от компании Synopsys

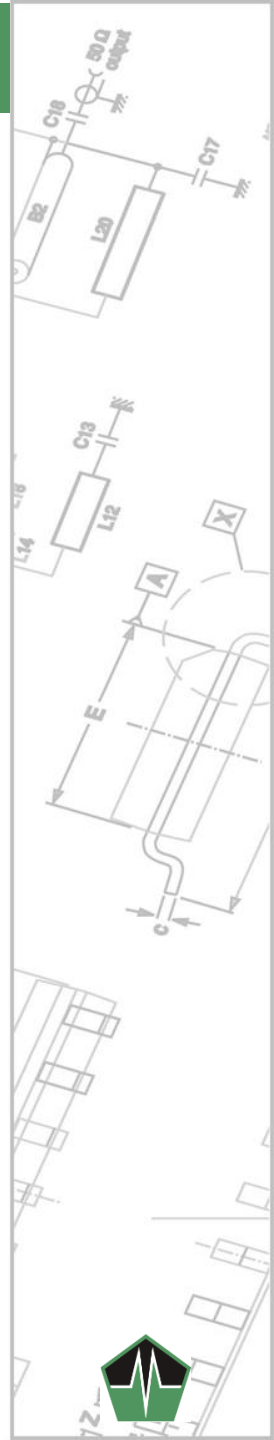
Все эти симуляторы поддерживаются Xilinx, но симуляторы сторонних разработчиков требуется покупать отдельно от Vivado за дополнительные деньги.



3. Симуляция (3)

Симулятор Xilinx Vivado Свойства и характеристики

- ✓ Одновременная поддержка языков
 - SystemVerilog
 - Verilog 2001
 - VHDL -93 и VHDL -2008
- ✓ Поддержка точек останова в исходниках
- ✓ Ко-симуляция с поддержкой языка C
- ✓ Симуляция нетлистов с учетом задержек (формат SDF3.0)
- ✓ Автоматическая компиляция исходников и библиотек
- ✓ Отображение в различных форматах, включая аналоговые сигналы
- ✓ Объединение сигналов в виртуальные шины
- ✓ Управление посредством TCL



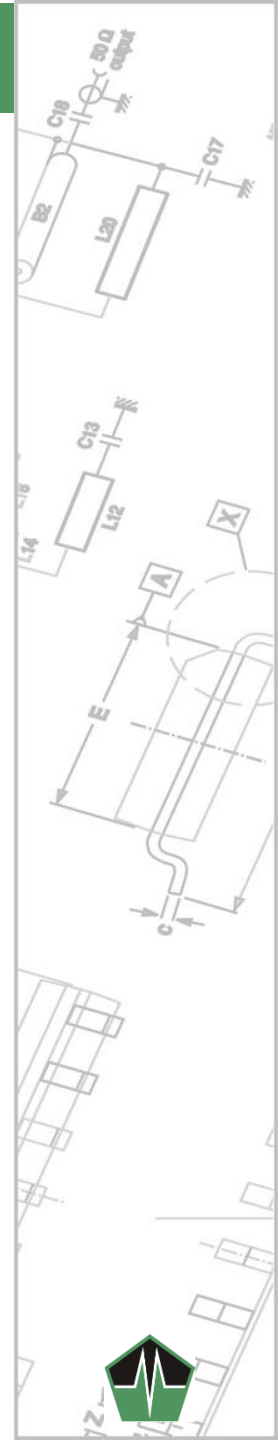
3. Симуляция (4)

Тестбенч – общее понятие

Чтобы отсимулировать проект, требуется написать т.н. “тестбенч”. Его можно рассматривать как совокупность синтезируемых файлов проекта и тестовых воздействий (стимулов)

Симуляция осуществляется потактово. Тестовые воздействия могут быть заданы как алгоритмически (при этом к тестовому коду требование синтезируемости не предъявляется, так и в виде тестовых векторов, задающих состояние всех входов системы на каждом такте выполнения.

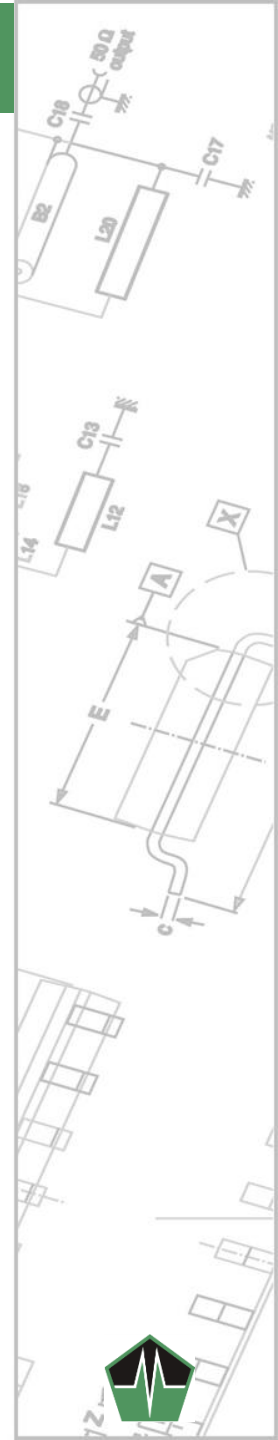
Важной характеристикой тестбенча, характеризующей полноту теста, является т.н. “покрытие”, (степень покрытия) т.е. отношение количества хотя бы раз переключившихся триггеров к общему количеству триггеров в проекте. Для качественного теста это отношение должно быть не ниже 100%, т.е. каждый триггер в процессе тестирования должен переключиться хотя бы один раз.



3. Симуляция (5)

Тестбенч – функциональное тестирование

На практике часто бывает достаточно провести т.н. “функциональное тестирование”, т.е. проверить правильность работы основных функций, выполняемых отлаживаемой системой.



2. Симуляция (6)

Тестбенч – варианты организации

Возможны варианты различные по построению и сложности.

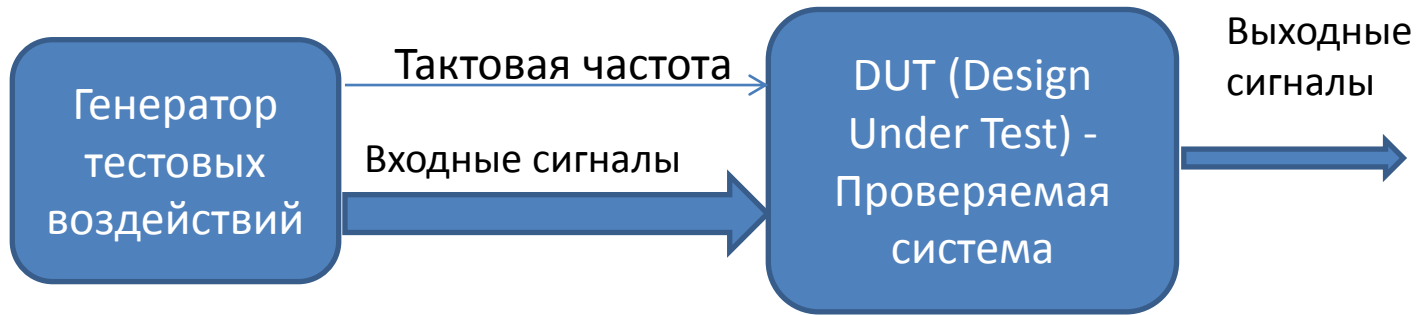
Возможная (неполная) классификация – от простого к сложному:

- Целенаправленная генерация входных воздействий и визуальная проверка корректности результата
- Целенаправленная генерация входных воздействий и автоматическая проверка корректности результата
- Автоматическая случайная генерация входных воздействий и автоматическая проверка корректности результата

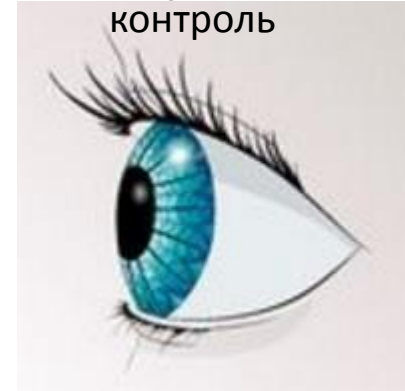


2. Симуляция (7)

Тестбенч – простой вариант

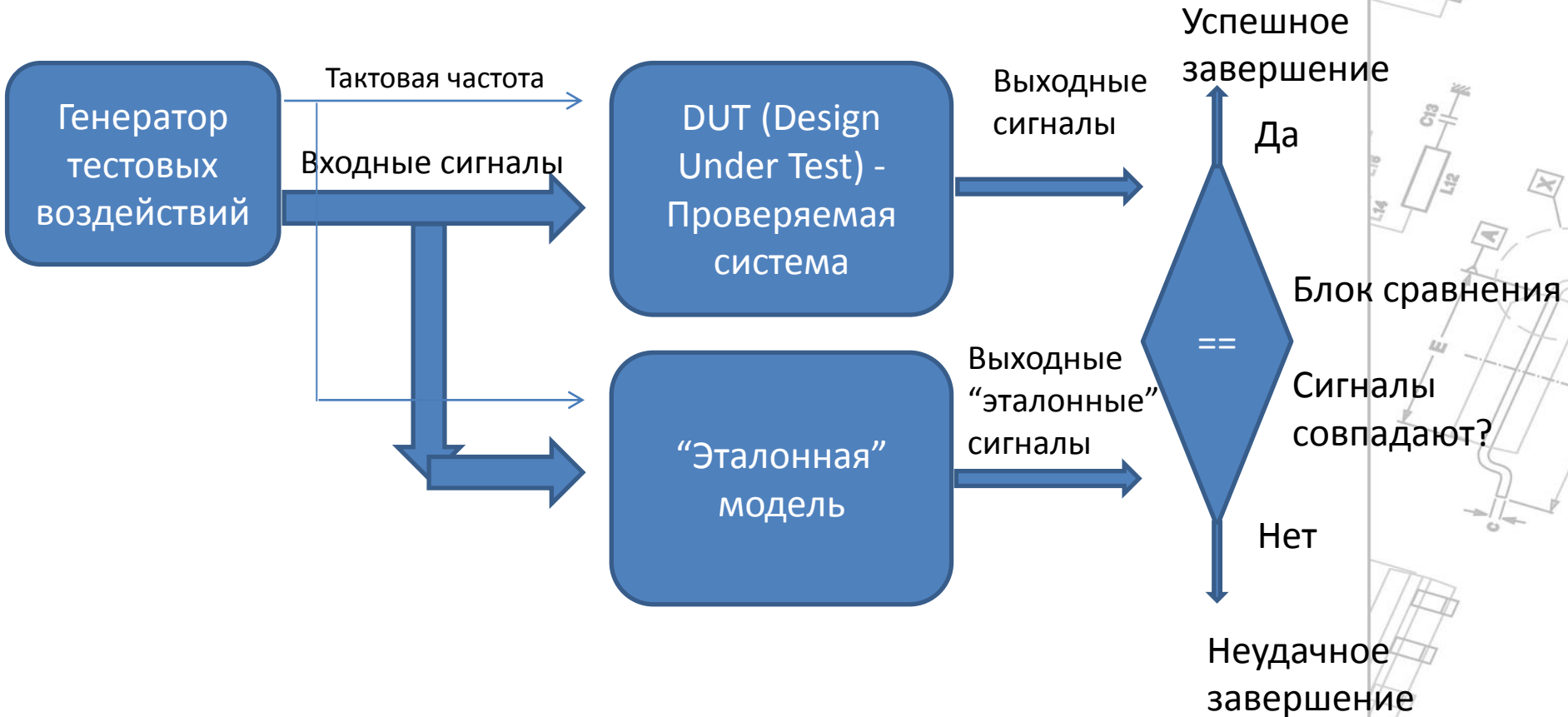


Визуальный контроль



3. Симуляция (8)

“Автоматический” тестбенч (вариант построения)

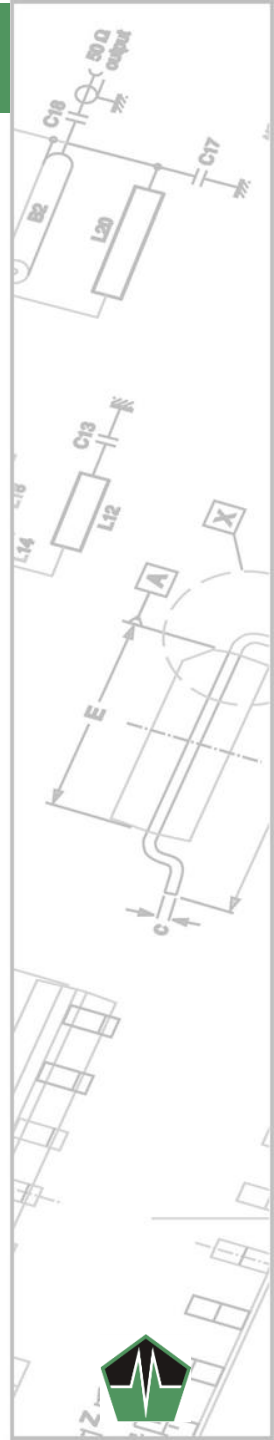


3. Симуляция (9)

Где искать дополнительную информацию?

- По симулятору Xilinx Vivado:
<https://www.xilinx.com/products/design-tools/vivado/simulator.html>
- Информация по конкретному симулятору – на сайтах производителей
- Общая методология – в технической литературе

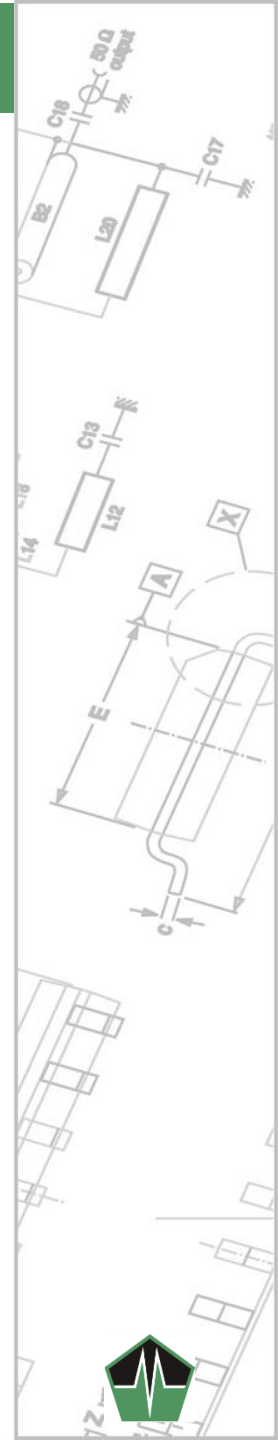
Отличная книжка: Chris Spear, [System Verilog for Verification](#) от Springer International Publishing AG – содержит описание современных методов верификации/симуляции с использованием методологий UVM, OVM и VMM



4. Симуляции проекта в среде Xilinx Vivado (1)

Демонстрационный пример

1. Создаем тестбенч
2. Запускаем на выполнение Simulation/Run Simulation
3. Выводим сигналы
4. Находим ошибки и устраняем их



Спасибо за внимание!



**МАКРО
ГРУПП**

www.macrogroupp.ru

fpga@macrogroupp.ru

8-800-333-0605

Владимир Викулин, инженер по применению Xilinx

Vladimir.Vikulin@macrogroupp.ru

