Ein Jahr Designerfahrung mit Xilinx VIVADO & wesentliche Unterschiede zur ISE aus der Praxis

User Comparison of the FPGA Development Tools, Werner Feith, Sensor to Image GmbH, www.sensor-to-image.com

Xilinx ISE vs Vivado and ISE vs ALTERA Quartus II

Note: Some of the statements below do not need to be really true. These just express impressions how the tools appear to a group of users inside Sensor to Image GmbH.

- ALTERA Quartus II was released in 2002 and is available with about the same user interface since 2002
- XILINX ISE has been released in Version 5 in 2003 and the last mature release is dated November 2013 in version 14.7. Bug fixed shall be around until 2020 but (almost) no 7series devices are supported
- XILINX VIVADO is stable since mid of 2013 supporting only 7series devices. There is no support in VIVADO (jet) for eg. Spartan6

Summary – Overall Impression

Vivado	*ISE*	*Quartus II*
• Huge Java-based GUI executing particular design jobs using its internal Tcl interpreter and Vivado specific Tcl library. • Whole design stored in a design database. Each design step retrieves its inputs from the database and stores results back. • Perfect scripting capabilities. Everything could be controlled using Tcl commands, only subset of the functions is available using the GUI.	• Native code GUI calling command line tools for particular jobs. • Each design step takes file(s) as an input, produces output file(s) and bunch of temporary/work files. • Some tools like Chipscope are missing inside the GUI and have to be called separately	• Native code GUI calling command line tools for particular jobs. • Each design step takes file(s) as an input, produces output file(s) and bunch of temporary/work files. • All necessary steps in one GUI with more details then ISE

File Formats

Vivado	*ISE*	*Quartus II*
• Constraints in XDC format what is Xilinx extension of the standard SDC file format. XDC allows entering even placement and location constraints, not only timing related ones. • SystemVerilog is supported for both synthesis and simulation. • High-level C based synthesis is available using an external tool Vivado-HLS.	• Constraints entered in proprietary UCF format. • No support for newer versions of HDLs.	• Timing constraints in SDC format (Synopsis Design Constraint) • Location constraints in proprietary QSF format • SystemVerilog as VHDL2008 is supported for both synthesis and simulation.

Tools

Task	*Vivado*	*ISE*	*Quartus II*
High-level synthesis	Vivado-HLS	–	-
Embedded system design	Block Design	Platform Studio	QSYS/SOPC Builder
IP cores management	IP Integrator	CoreGen	MegaWizard Plugin Manager
IP cores packaging	IP Packager	–	-
Logic simulation	Vivado Simulator	ISim	ModelSim ALTERA Edition
FPGA analysis/editing	Device View	FPGA Editor	net list viewers, like: RTL Viewer, State Machine Viewer, Technology Map Viewer, …
Device programming	Hardware Manager	iMPACT	1. Convert Programming File 2. Quartus II Programmer
In-hardware debugging	Integrated Logic Analyzer	ChipScope Pro	SignalTap II Logic Analyzer
In-software debugging	SDK based on XMD and GDB	XMD Console	1. System Console 2. In-System Memory Content Editor
Timing Analysis	Tcl commands: report_timing_summary report_timing report_clocks report_clock_networks report_clock_interaction report_clock_utilization	Timing Analyzer	Timequest Timing Analyzer
Software development	SDK (Eclipse)	SDK (Eclipse)	Nios II Software Build Tools for Eclipse

- All tools in Vivado listed in the table except Vivado-HLS are integrated part of the GUI.
- The iMPACT tool is still included in Vivado installation so far. The Hardware Manager probably does not cover its full functionality jet

- Vivado 2013.4 might be the last version to import ISE14.7 EDK designs
- Timing Analyzer is a report tool

Additional Differences

Vivado	*ISE*	*Quartus II*
• All clock domains are related by default. It is necessary to explicitly unrelate asynchronous clock domains in XDC otherwise the implementation tries to meet the non-existing timing requirements. • Implementation run time very long and poor quality of results for designs with no or incomplete timing constraints. • Much less synthesis and implementation options compared to ISE. Some properties could be controlled using HDL attributes. • Though the implementation tasks run faster than in ISE, there are much longer "preparation" and "finalization" phases. Probably retrieving appropriate data from the design database and preparing it for processing takes long time. The same applies for storing the results back to design database and cleanup. • High memory consumption for small design. The memory occupation does not grow as fast with growing design size as in ISE. • Much better multiprocessing support than in ISE, though still not ideal. • More and better design and implementation analysis tools. • Perfect scripting in Tcl.	• All clock domains explicitly unrelated. • Possibility to get working design with no timing constraints.	• All clock domains explicitly unrelated. • Possibility to get working design with no timing constraints • Lots of scripting in Tcl.

Design methods: How to initialize the state of a FlipFlop

ISE6	*ISE11*	*Vivado2013.4*
• Asynchronous RESET in Spartan2 and Spartan2E • Synchronous RESET in Spartan3 with quite some effort to have a common RESET in several clock domains inside one chip	• Synchronous RESET in Spartan3x, Spartan6 with quite some effort to have a common RESET in several clock domains inside one chip • FF initialization in VHDL with INIT attribute (signal abc : std_logic_vector(3 downto 0):=(others=>'0');) works now, but is no widely used	• Synchronous RESET deprecated because: 1. Low global routing resources in new and big chips 2. High risk of having RESET due to signal routing at different clock cycles in opposite FPGA corners –or- limited bandwidth to avoid this behavior 3. Limited chip use as modern FPGA have now 4 FF/logic element with one=1 clock on this logic element. So with synchronous RESET all 4 FF in this logic element need to run at the same clock • FF initialization in VHDL with INIT attribute (signal abc : std_logic_vector(3 downto 0):=(others=>'0');) recommended for the use of GSR resources • Use HIGH active synchronous RST and do not mix HIGH and LOW active RST signals

Vivado /ISE, about project structure

Vivado	*ISE*
• Base directory: XPR Vivado project file • Below all needed directories for clear FPGA as C project elements • Vivado_gvrd-ac701-tpg\gvrd-ac701.srcs\sources_1\new → all VHDL • Vivado_gvrd-ac701-tpg\gvrd-ac701.srcs\sources_1\new → all VHDL • Vivado_gvrd-ac701-tpg\gvrd-ac701.srcs\sources_1bd → all uBlaze, ARM, IP Builder sources • Vivado_gvrd-ac701-tpg\gvrd-ac701.srcs\constrs_1\new → all constraints • Vivado_gvrd-ac701-tpg\gvrd-ac701.sdk\SDK\SDK_Export generated by Block design as SDK basis (and S2I software base directory) • Vivado_gvrd-ac701-tpg\gvrd-ac701.sdk\SDK\SDK_Export\hw standard location of Block diagram XML (as in ISE …) ✓ Good structure!	• Base directory: free to choose, S2I solution: ISE14_gvrd-ac701-v2\ise\ XISE project file • User is responsible for his project structure, no special character to be used like **Umlaute** • ISE14_gvrd-ac701-v2\ise\ XISE project file • ISE14_gvrd-ac701-v2\src\ → all VHDL and all constraints • ISE14_gvrd-ac701-v2\cpu\ → all EDK MHS file • ISE14_gvrd-ac701-v2\sdk\ → SDK base directory, has to be exported to this location manually from EDK ○ No/private structure!

Vivado/ISE, some GUI details

Vivado	*ISE*
• Synthesis → “Synthesis + Translate” in ISE due to uniform file formats for post Synthesis simulation. • Synthesis & BlockDesign → close Blockdesign → Generate Ouput Products -> Run Synthesis, else strange behavior • Implementation → “Map + Place&Route” due to simplification • No IMPACT, now Hardware manager: As we do not understand Hardware manger it is S2I solution to transfers ISE PEARL scripts to Vivado TCL scripts and use XMD as IMPACT replacement	• Synthesis, was able to generate and update EDK net lists if needed • Translate • Map • Place&Route • IMPACT

BlockDesign/EDK, some GUI details

Vivado	*ISE*
• Blockdesign (back again from Foundation!?) • All IP blocks are available as drawing blocks, block parameterization like in EDK • **No rule to start with CPU, bus, ... - but not limited to CPU centric systems** • No (real) rule on how to connect blocks • No general import available for ISE EDK projects • Again Unclear SDK/Eclipse integration • Anarchic approach with slow initial project progress, maybe needed as a good solution for universal and long term IP integration	• Block based design focused on uBlaze/ARM systems • Central element is CPU → Bus (OPB, PLB, AXI V1.x) → peripherals • Unclear SDK/Eclipse integration • ISE tool version not compatible to other major ISE versions due to unclear integration of GNU- and Eclipse tools