ORCONF2015
Timo D. Hämäläinen
Esko Pekkarinen

# Kactus2: Open Source IP-XACT tool

TAMPERE UNIVERSITY OF TECHNOLOGY

# Current research

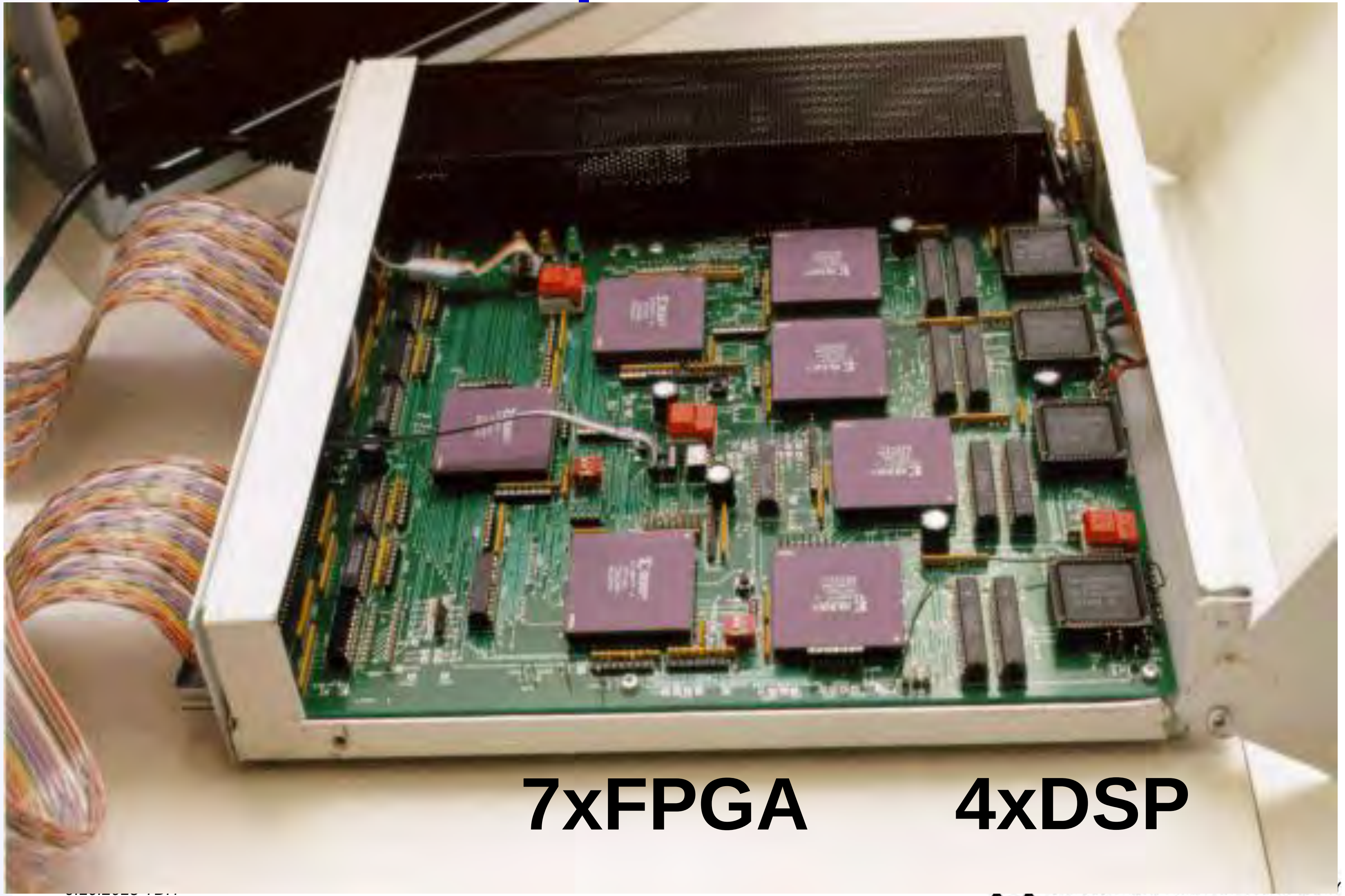
**SoC design methods and tools**
*Kactus2 open source IP-XACT tool*

Kactus2

**Parallel video encoder implementations**
DSP, FPGA, Multicore
*Kvazaar open source HEVC encoder*

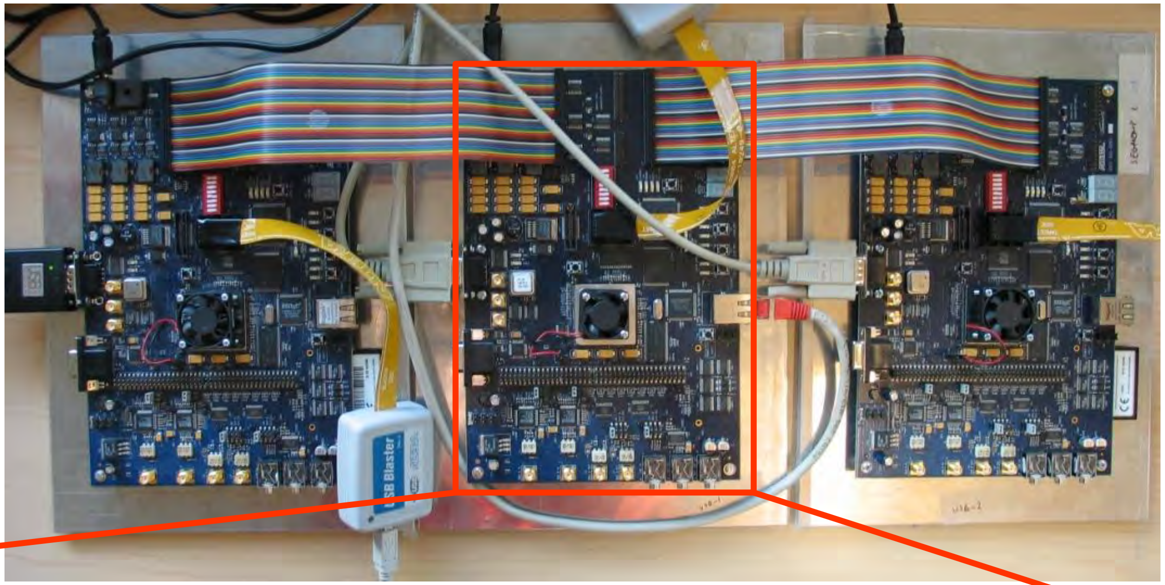
KVAZAAR HEVC ENCODER

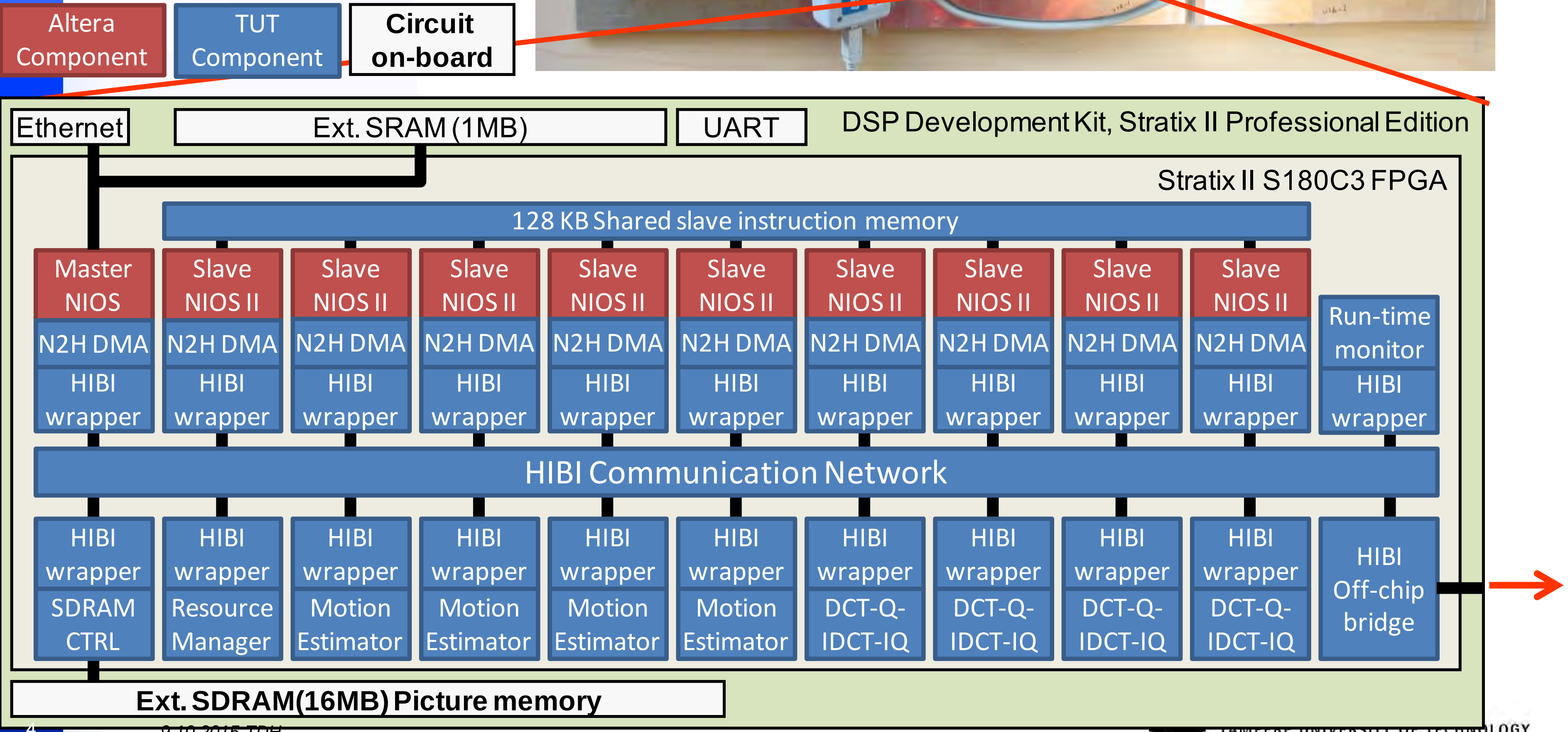
## Neural Network & Genetic Algorithm computer TUTNC (1996)



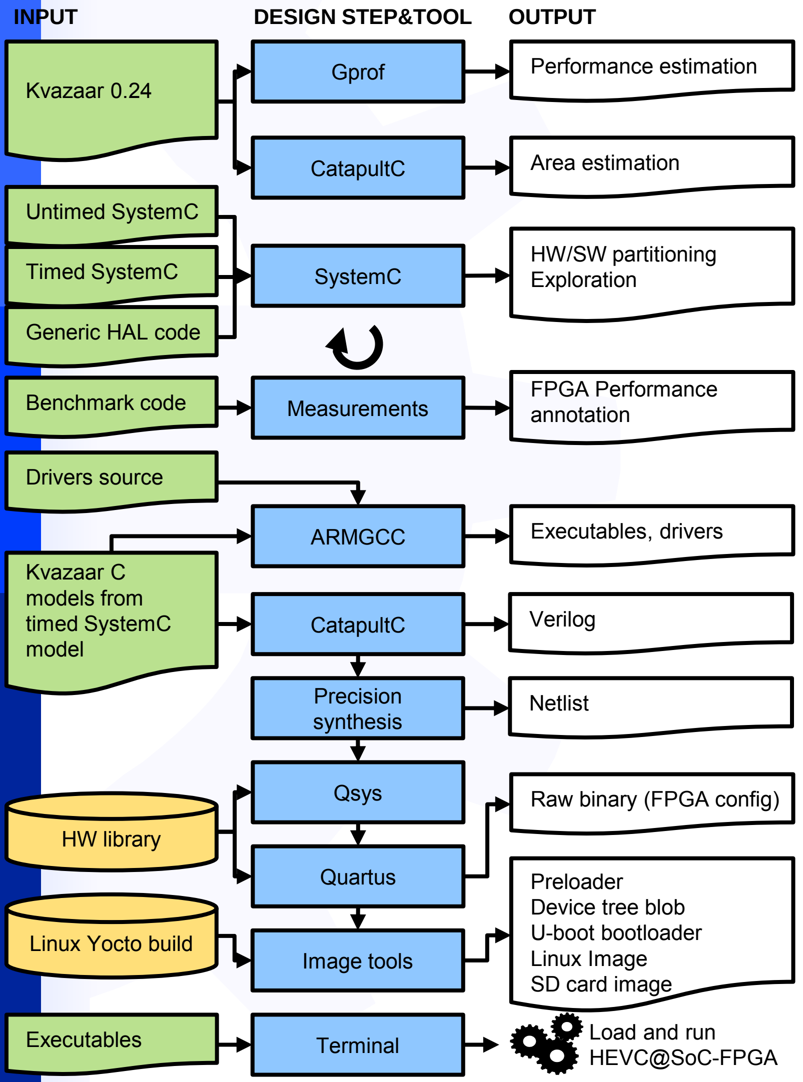
7xFPGA 4xDSP

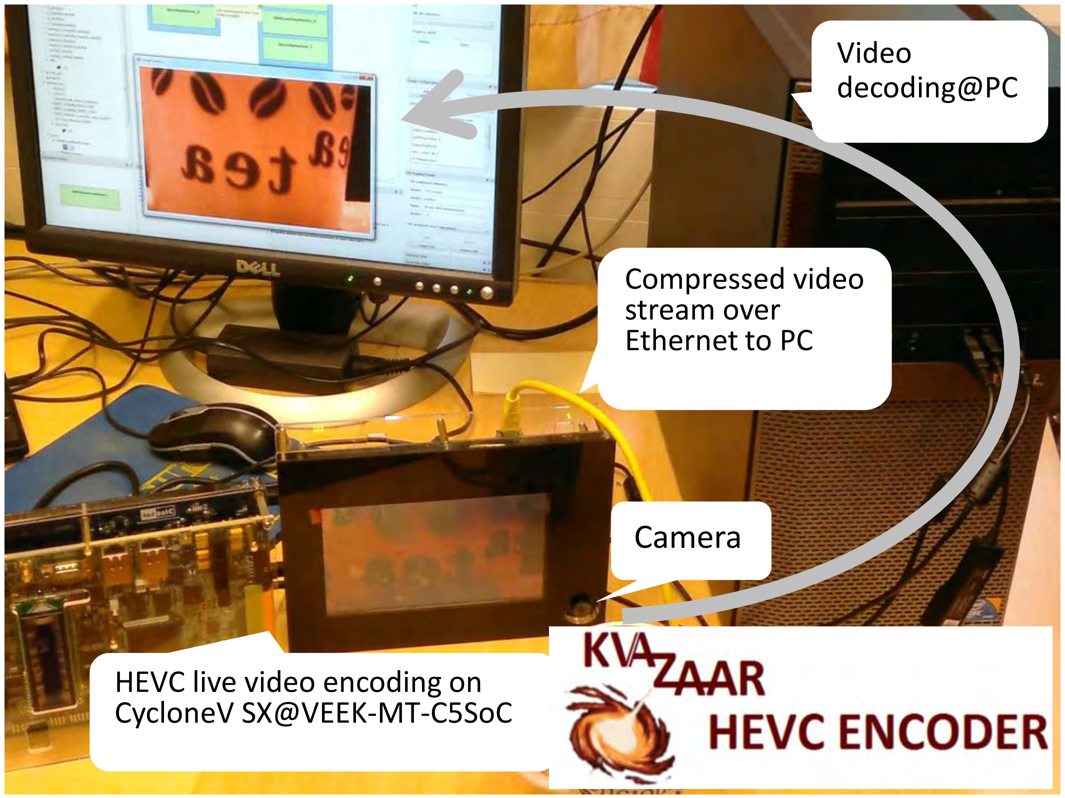
# "HIBI" on-chip network 3x 10xNIOS (2005)





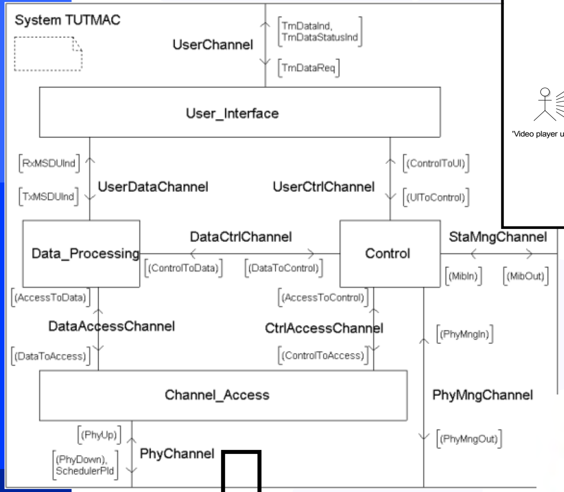
# Kvazaar open source HEVC encoder on CycloneV SoC-FPGA (DSD'2015)

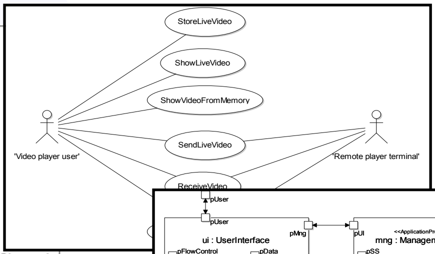


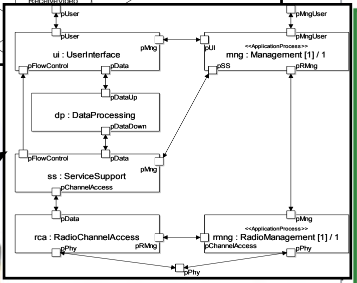

Video decoding@PC
Compressed video stream over Ethernet to PC
Camera
HEVC live video encoding on CycloneV SX@VEEK-MT-C5SoC
KVAZAAR HEVC ENCODER

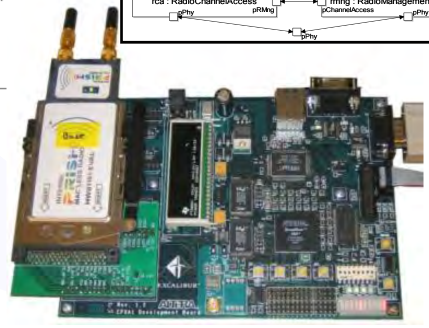
# Design methods & tools roadmap

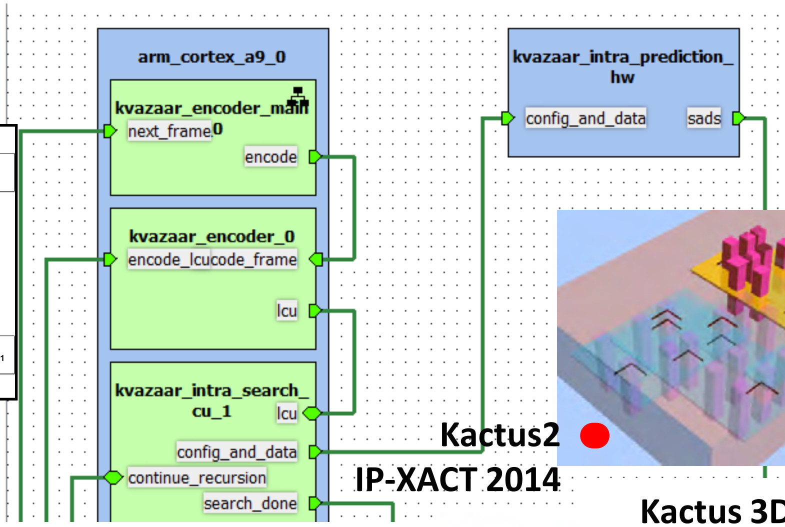
**SDL** **UML** **IP-XACT** **HLS**

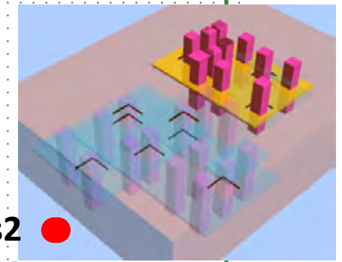












Kactus2 ● IP-XACT 2014

Kactus 3D ●

Kactus2 IP-XACT ● + extensions

Kactus1 tool ● IP-XACT 1.2

● SDL-based DSP+FPGA design for custom WLAN MAC

● UML2.0 based MP-SoC video+WLAN

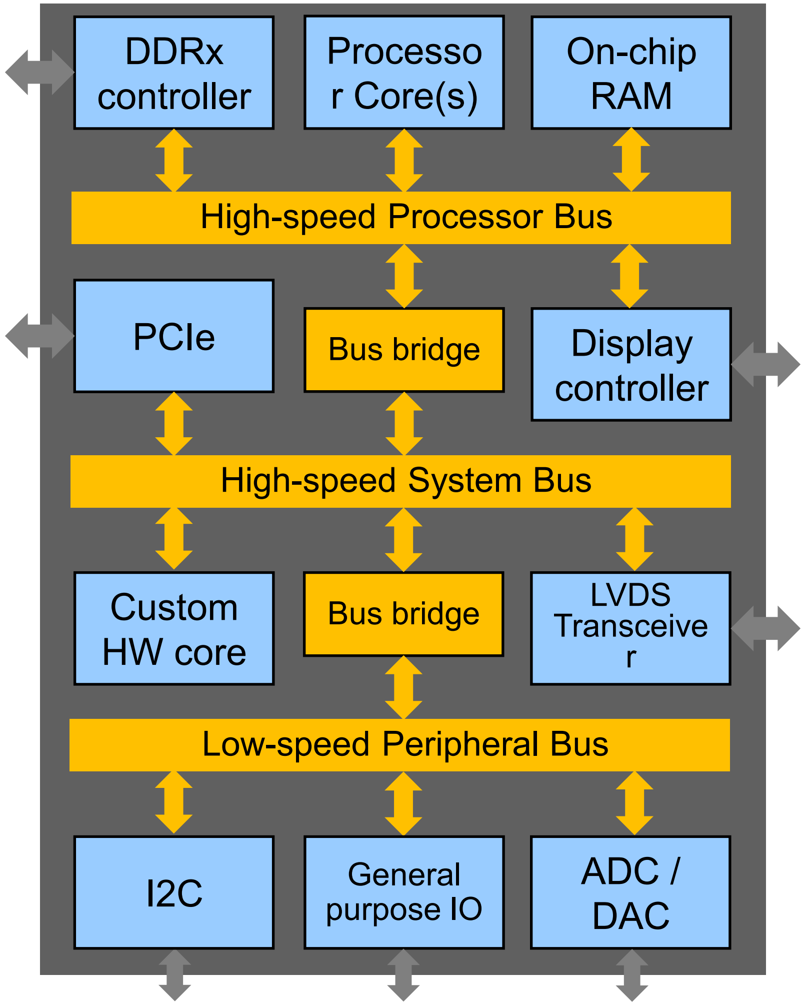
ARM + FPGA multicore ●

2000 2005 2010 2015 2020

## IP-XACT Motivation

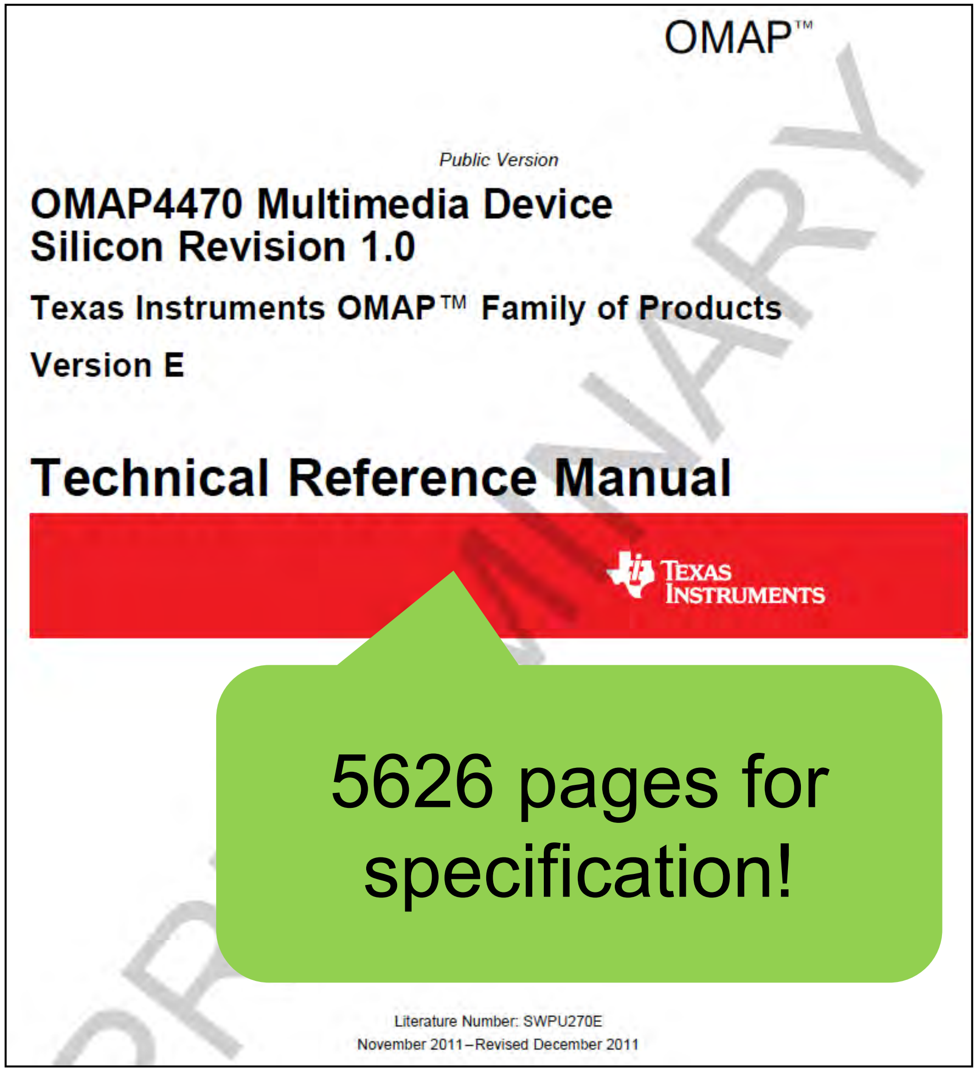
# ASIC/SoC-FPGA design challenges

- Multitude of tools, languages and specification styles
  - Different formats (syntax)
  - Meaning (semantics)
  - Intention (how language or tool is applied –or abused…)
- Abstractions above RTL must be used for design space exploration
- Design for deadline -> design for reuse
- SW development should start in parallel with HW design



## The HDL challenges

- VHDL/Verilog HDLs include three aspects mixed
  - Structural and behavioral descriptions
  - Control for configuration (functions, param. propagation, generators, conditions)
  - Virtual libraries (name based references)
- Implicit references that get evaluated late/somewhere in the design flow
- Vulnerable to errors if 100k files and multiple vendors
  - Wrong path/files, conflicts in (re)naming, scripts dependent on file version, …
- Coding style agreements does not seem to help


OMAP™

*Public Version*

**OMAP4470 Multimedia Device Silicon Revision 1.0**

**Texas Instruments OMAP™ Family of Products**

**Version E**

**Technical Reference Manual**

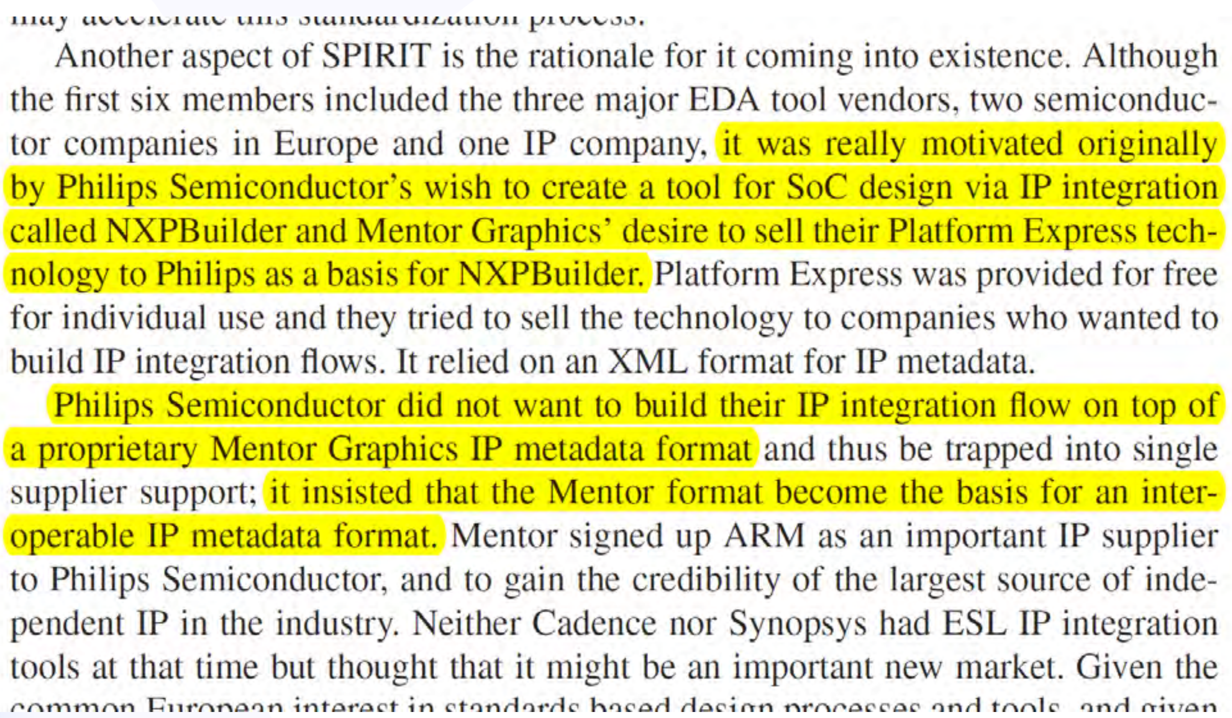
Literature Number: SWPU270E

November 2011–Revised December 2011

5626 pages for specification!

## The IP-XACT Story

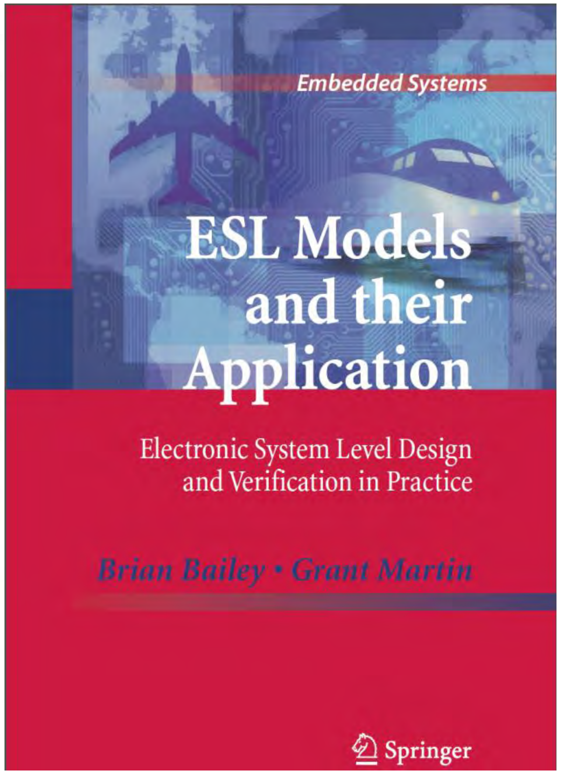
- The idea was published in DAC’2003 conference
- Mentor’s closed tool format + Philips’ IP assembly tool = IP-XACT 1.0
- Now IP-XACT is the IEEE standard 1685-2014



may accelerate this standardization process.

Another aspect of SPIRIT is the rationale for it coming into existence. Although the first six members included the three major EDA tool vendors, two semiconductor companies in Europe and one IP company, it was really motivated originally by Philips Semiconductor’s wish to create a tool for SoC design via IP integration called NXPBuilder and Mentor Graphics’ desire to sell their Platform Express technology to Philips as a basis for NXPBuilder. Platform Express was provided for free for individual use and they tried to sell the technology to companies who wanted to build IP integration flows. It relied on an XML format for IP metadata.

Philips Semiconductor did not want to build their IP integration flow on top of a proprietary Mentor Graphics IP metadata format and thus be trapped into single supplier support; it insisted that the Mentor format become the basis for an interoperable IP metadata format. Mentor signed up ARM as an important IP supplier to Philips Semiconductor, and to gain the credibility of the largest source of independent IP in the industry. Neither Cadence nor Synopsys had ESL IP integration tools at that time but thought that it might be an important new market. Given the common European interest in standards based design processes and tools, and given


Embedded Systems
ESL Models and their Application
Electronic System Level Design and Verification in Practice
Brian Bailey • Grant Martin
Springer

**Brian Bailey, Grant Martin, ESL Models and their Application, Springer, 2009**
**http://link.springer.com/chapter/10.1007%2F978-1-4419-0965-7_2**

# IP-XACT

- Standard
  - IP-block model
  - SoC design model
  - Integration and configuration flow
  - Tool interfaces
- IEEE1685-2014 include
  - 24 XSD files
  - 790 elements, 241 attributes
  - Vendor extensions
    - E.g. Xilinx adds ~200 elements
- Attempts to be a methodology, not as yet another exchange file format

IEEE

IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

IEEE Computer Society
and the
IEEE Standards Association Corporate Advisory Group

Sponsored by the
Design Automation Standards Committee

1685™

IP-XACT™

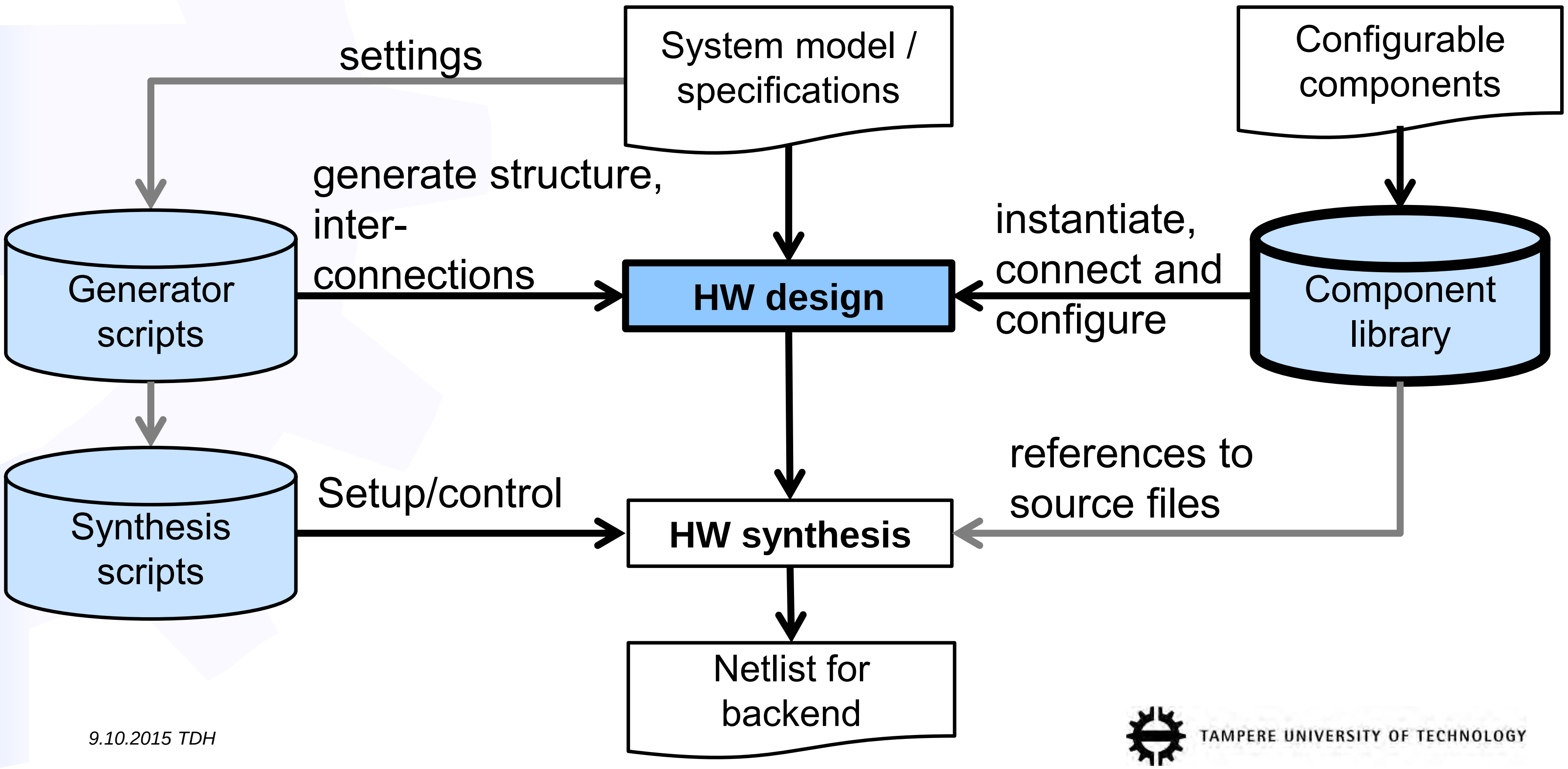
"Vendor, implementation language, abstraction level and tool independent description of IP-blocks and SoC designs"

## Kactus2 motivation

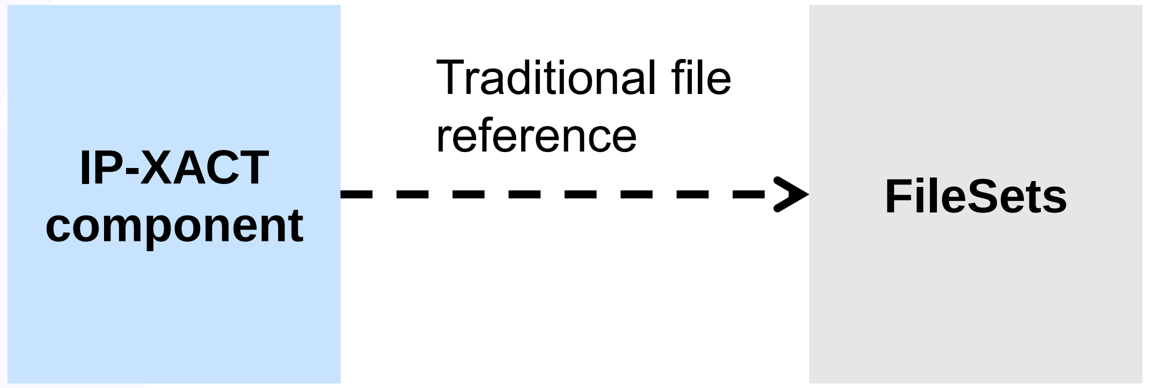
- Launched from the needs of embedded system SMEs – and need to re-use the results of student projects
- Subcontractor SMEs need (affordable) tool to packetize their IP for integrator companies
- Thus:
- Users may not be IP-XACT experts
- Tool must have much better usability than EDA tools on average
- Tool must be easy to install and ready to use right from the start
- We keep every release as stable as possible even if it lacks features

# Scope of IP-XACT

- Model for IP/design exchange and reuse
- Placeholder for *generators* configuring/affecting designs
- Not a language nor e.g. definition of automation scripts



# IP-block = The IP-XACT component



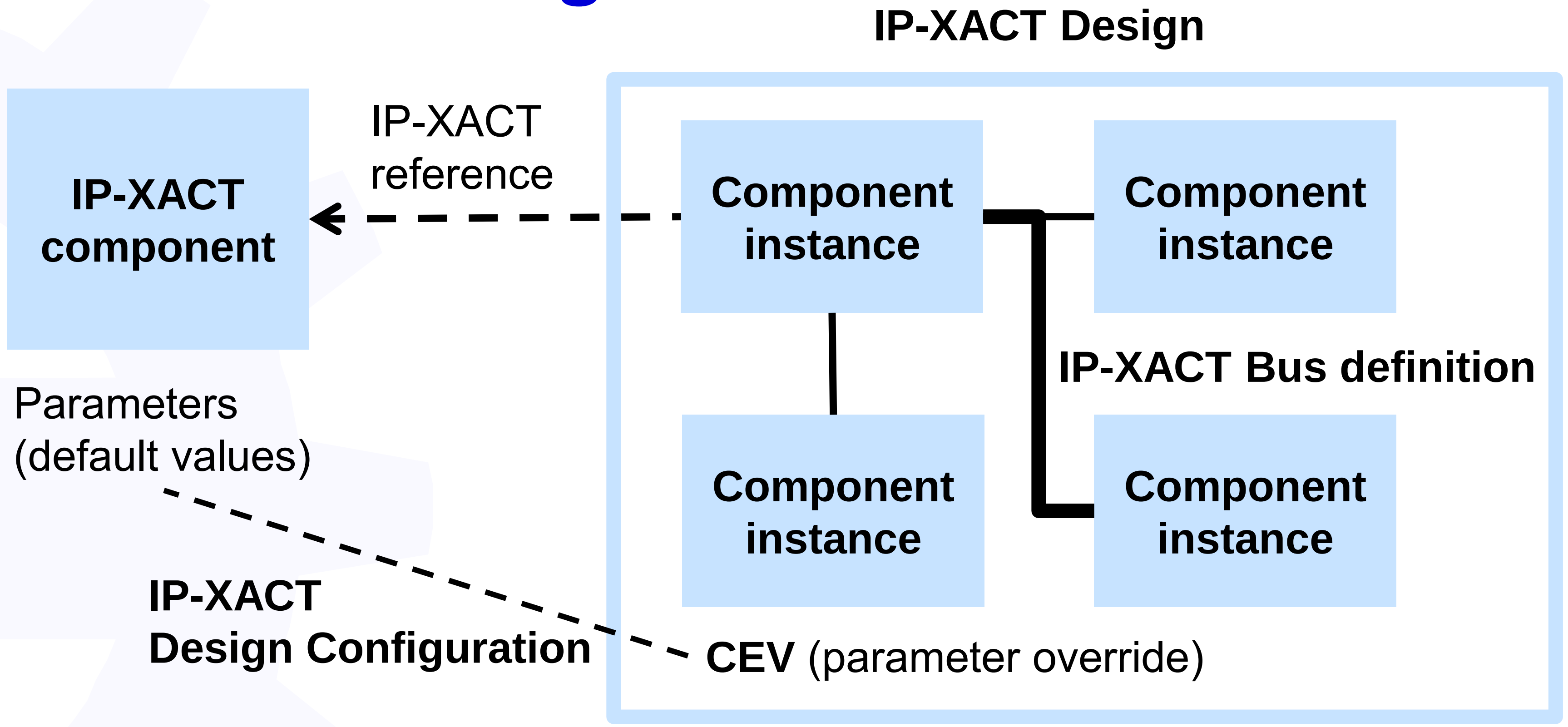
Common structure and definitions:
- Interfaces
- Parameters
- Registers (for SW)
- (Non-functional properties)

Associated files for
- Behavior
- Documentation
- Whatever out there but the interface

**Analogy: VHDL Entity = interface** **VHDL Architecture = functionality**

- **IP-XACT component** is a structural model of the IP-block
- Several implementations at different abstraction levels and languages can be included as **Views** and **FileSets**
  - All implement the same external interface
- Reusable: self-containing, explicit definitions, **parameters must be evaluated within the component**

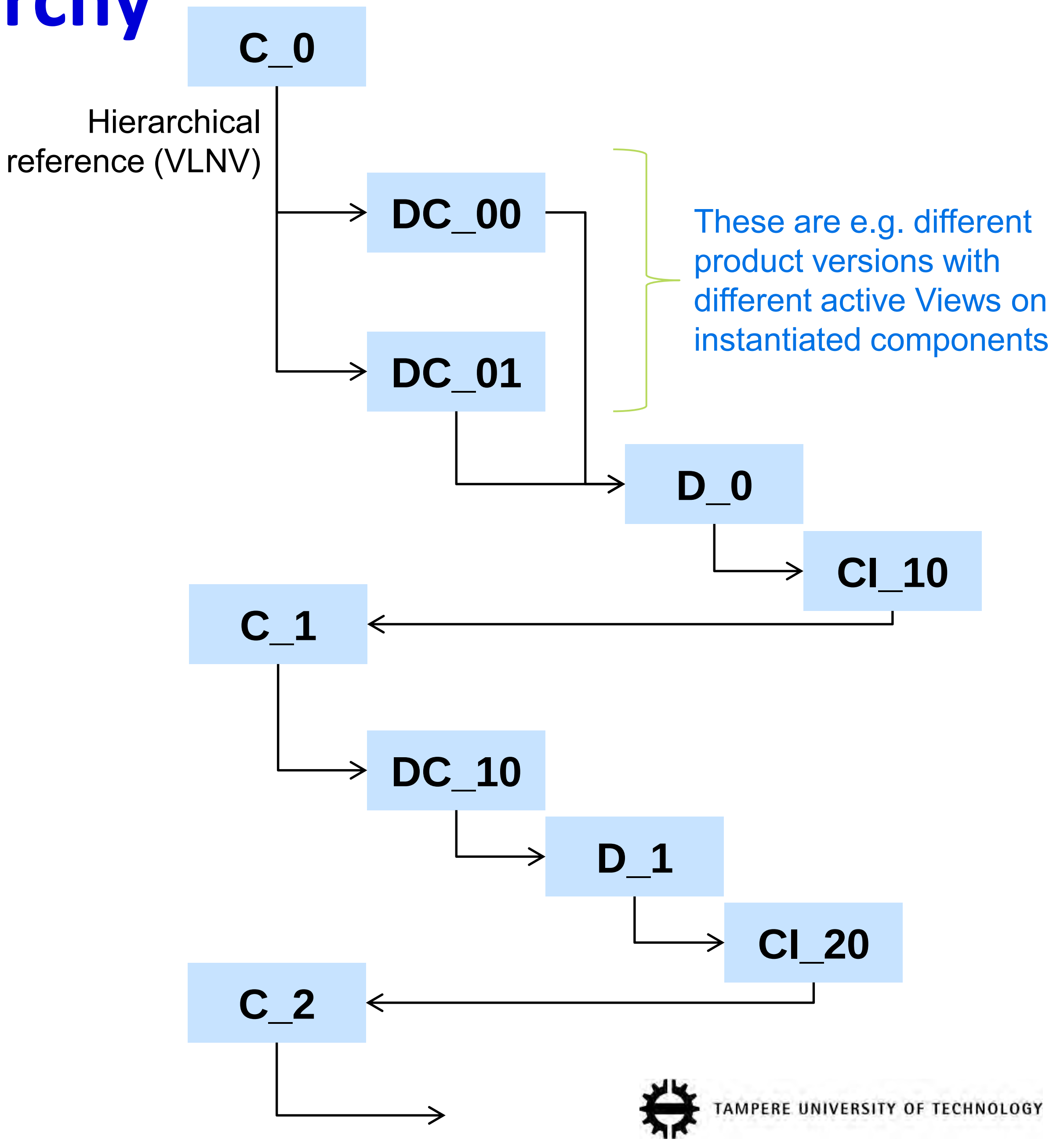
# The IP-XACT Design



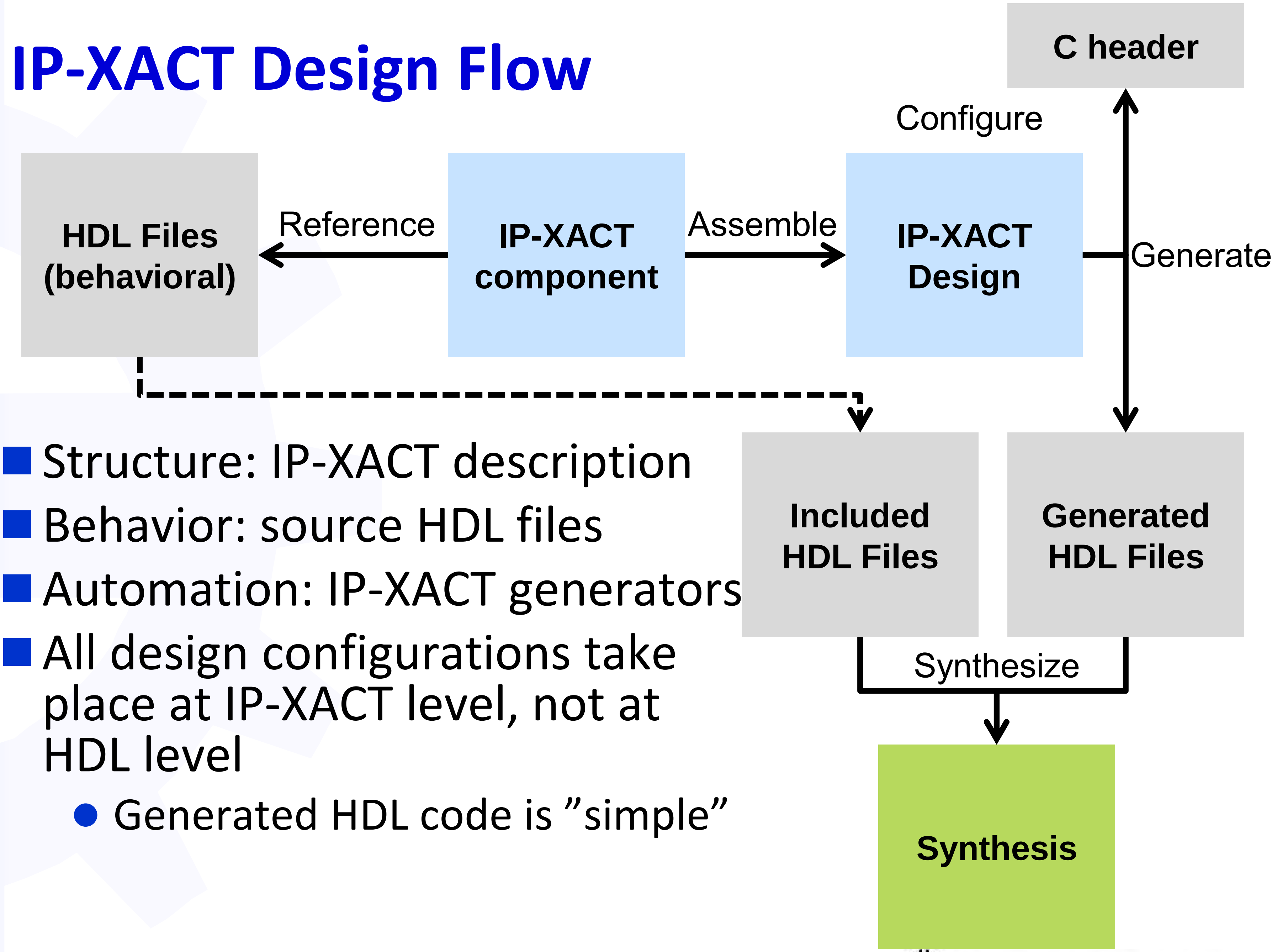
- **IP-XACT design** is a model of the SoC
  - Component instances
  - Interconnections, formalized by **Bus definitions**
  - Design-wide and instance specific component configurations by **Configurabe Element Values** that are specific to **Design Configurations**
- Parameters propagate only one level downwards in hierarchy
  - Component must always be independently reusable independent of context

# IP-XACT hierarchy

- C: Component
- DC: Design configuration
- D: Design
- CI: component instance

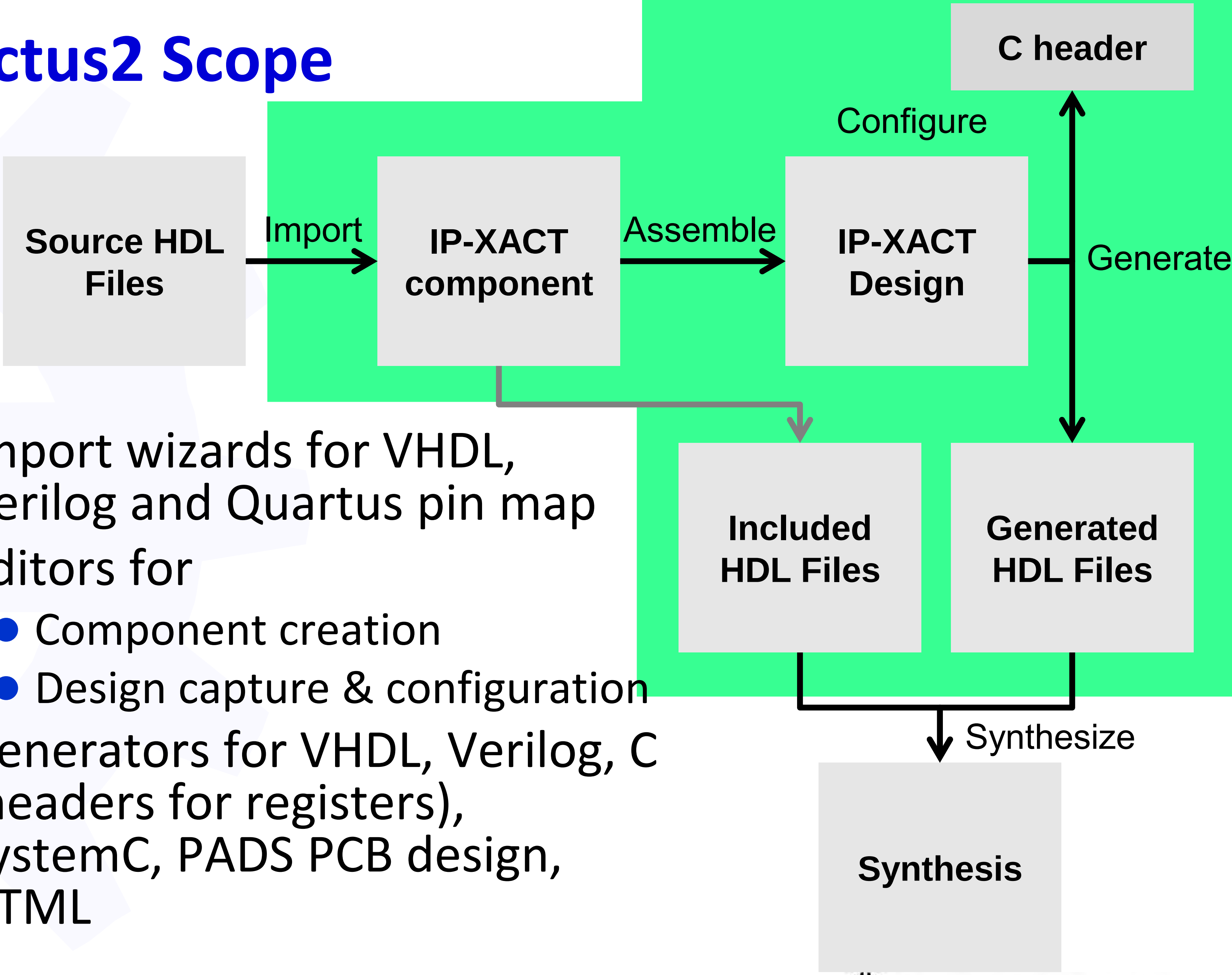

C_0
Hierarchical reference (VLNV)
DC_00
DC_01
These are e.g. different product versions with different active Views on instantiated components
D_0
CI_10
C_1
DC_10
D_1
CI_20
C_2

# IP-XACT Design Flow



- Structure: IP-XACT description
- Behavior: source HDL files
- Automation: IP-XACT generators
- All design configurations take place at IP-XACT level, not at HDL level
    - Generated HDL code is ”simple”

# Kactus2 Scope



- Import wizards for VHDL, Verilog and Quartus pin map
- Editors for
  - Component creation
  - Design capture & configuration
- Generators for VHDL, Verilog, C (headers for registers), SystemC, PADS PCB design, HTML

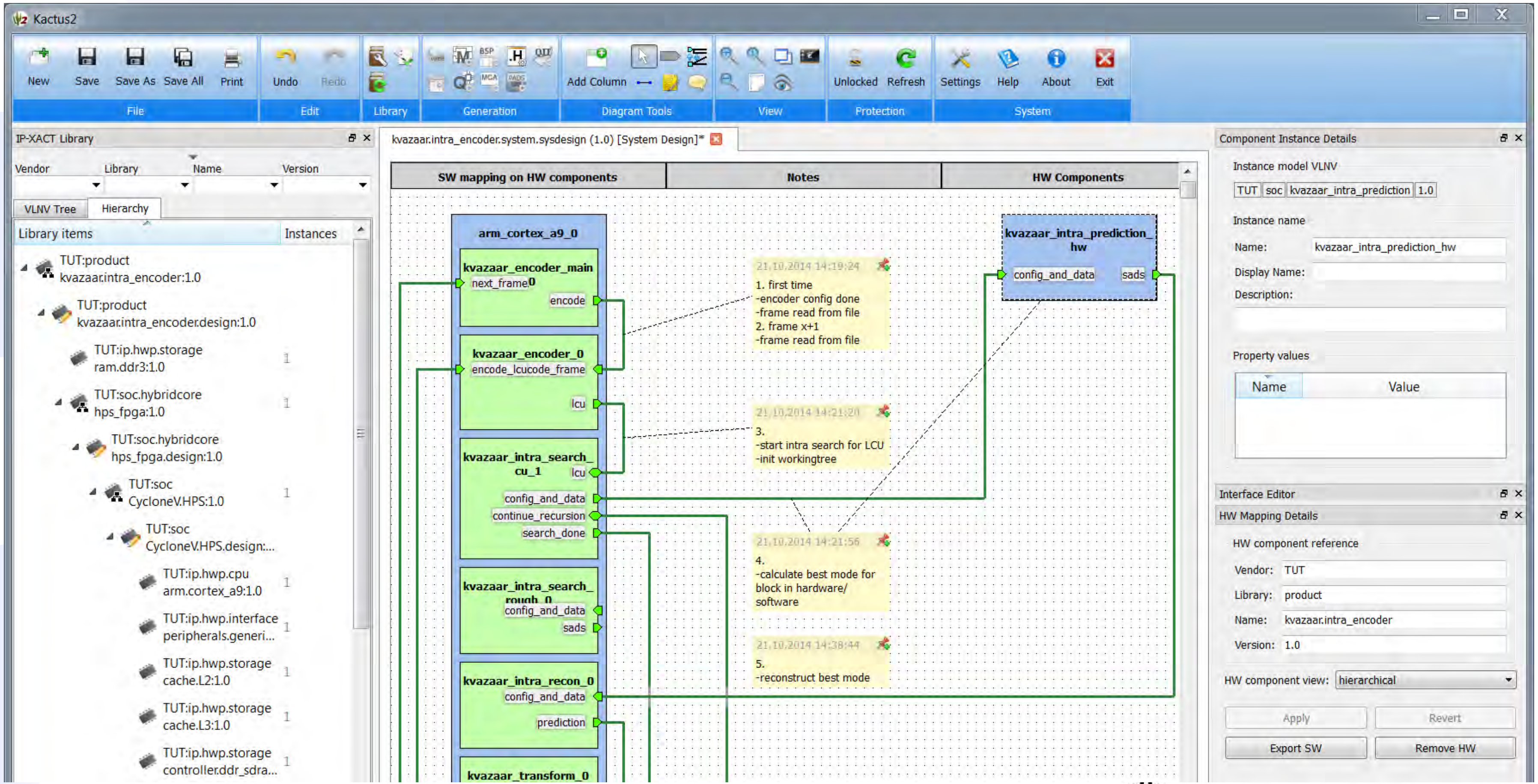
## Challenges in applying IP-XACT

- Standard does not ensure compatibility of bus/abstraction definitions from different vendors
- Vendor extensions complicate exchangeablility
- Re-generation of HDL code is problematic if it was manually fixed in between (general problem)
- Common problems:
  - User edits IP-XACT XML files manually and add comments to anywhere in the XML
  - Try to implement over-generic legacy HDL projects in IP-XACT as such
  - Make ruthless file/name dependencies and does not respect the IP-XACT way
  - Abuse IP-XACT elements, e.g. a parameter is NOT for a file path reference or for controlling product versions

## Kactus2 tool

# Kactus2 demo

- Component ports, address space, memory map
- Import from VHDL file
- (draft a new component and generate stub VHDL code)



kvazaar.intra_encoder.system.sysdesign (1.0) [System Design]*

TUT:product kvazaar.intra_encoder:1.0
TUT:product kvazaar.intra_encoder.design:1.0
TUT:ip.hwp.storage ram.ddr3:1.0 — 1
TUT:soc.hybridcore hps_fpga:1.0 — 1
TUT:soc.hybridcore hps_fpga.design:1.0
TUT:soc CycloneV.HPS:1.0 — 1
TUT:soc CycloneV.HPS.design:...
TUT:ip.hwp.cpu arm.cortex_a9:1.0 — 1
TUT:ip.hwp.interface peripherals.generi... — 1
TUT:ip.hwp.storage cache.L2:1.0 — 1
TUT:ip.hwp.storage cache.L3:1.0 — 1
TUT:ip.hwp.storage controller.ddr_sdra... — 1

arm_cortex_a9_0
kvazaar_encoder_main: next_frame, encode
kvazaar_encoder_0: encode_lcucode_frame, lcu
kvazaar_intra_search_cu_1: lcu, config_and_data, continue_recursion, search_done
kvazaar_intra_search_rough_0: config_and_data, sads
kvazaar_intra_recon_0: config_and_data, prediction
kvazaar_transform_0
kvazaar_intra_prediction_hw: config_and_data, sads

21.10.2014 14:19:24
1. first time
-encoder config done
-frame read from file
2. frame x+1
-frame read from file

21.10.2014 14:21:20
3.
-start intra search for LCU
-init workingtree

21.10.2014 14:21:56
4.
-calculate best mode for block in hardware/ software

21.10.2014 14:38:44
5.
-reconstruct best mode

Instance model VLNV: TUT | soc | kvazaar_intra_prediction | 1.0

Instance name
Name: kvazaar_intra_prediction_hw
Display Name:
Description:

Property values

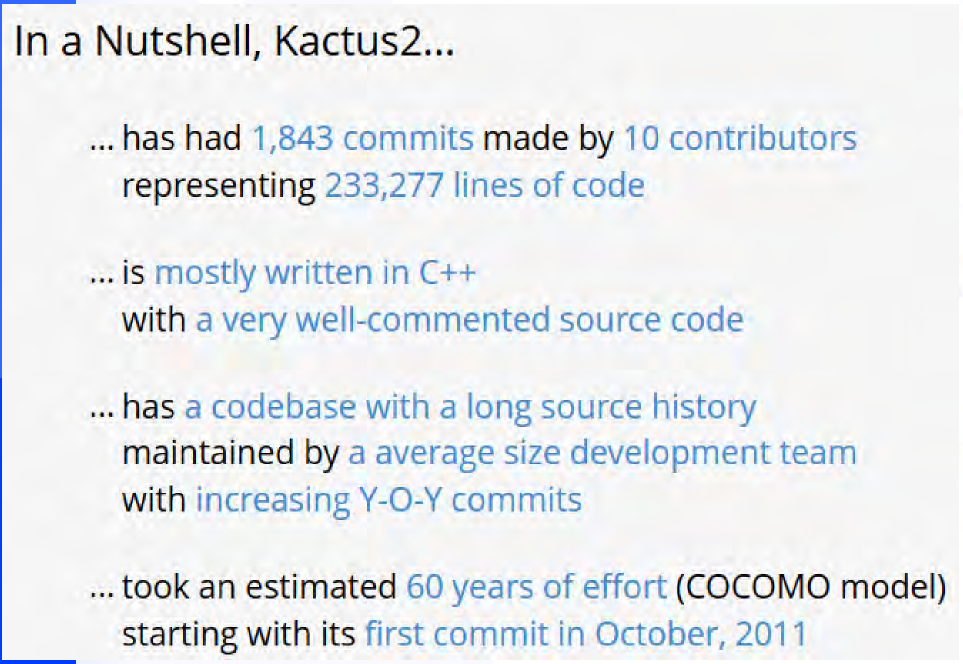
| Name | Value |
|---|---|

HW component reference
Vendor: TUT
Library: product
Name: kvazaar.intra_encoder
Version: 1.0

HW component view: hierarchical

## Kactus2 tool project

# Kactus2 Project



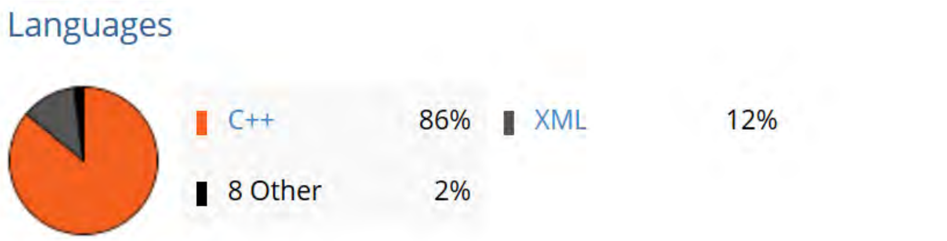
In a Nutshell, Kactus2...

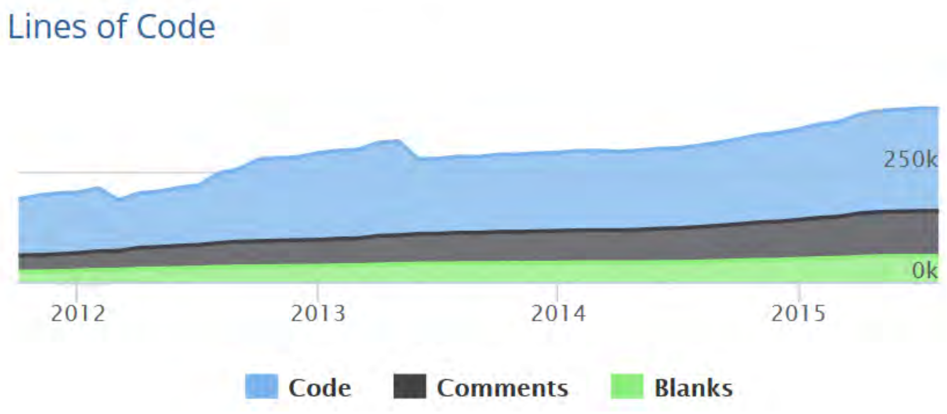
... has had 1,843 commits made by 10 contributors representing 233,277 lines of code

... is mostly written in C++ with a very well-commented source code

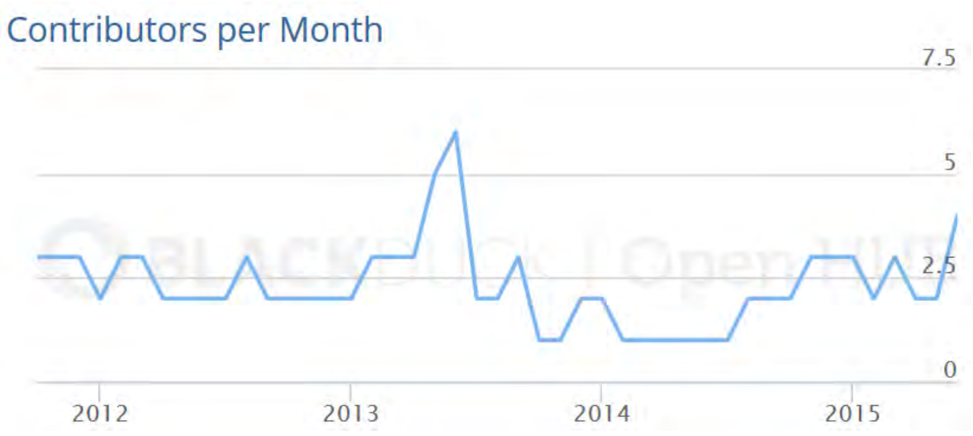
... has a codebase with a long source history maintained by a average size development team with increasing Y-O-Y commits

... took an estimated 60 years of effort (COCOMO model) starting with its first commit in October, 2011



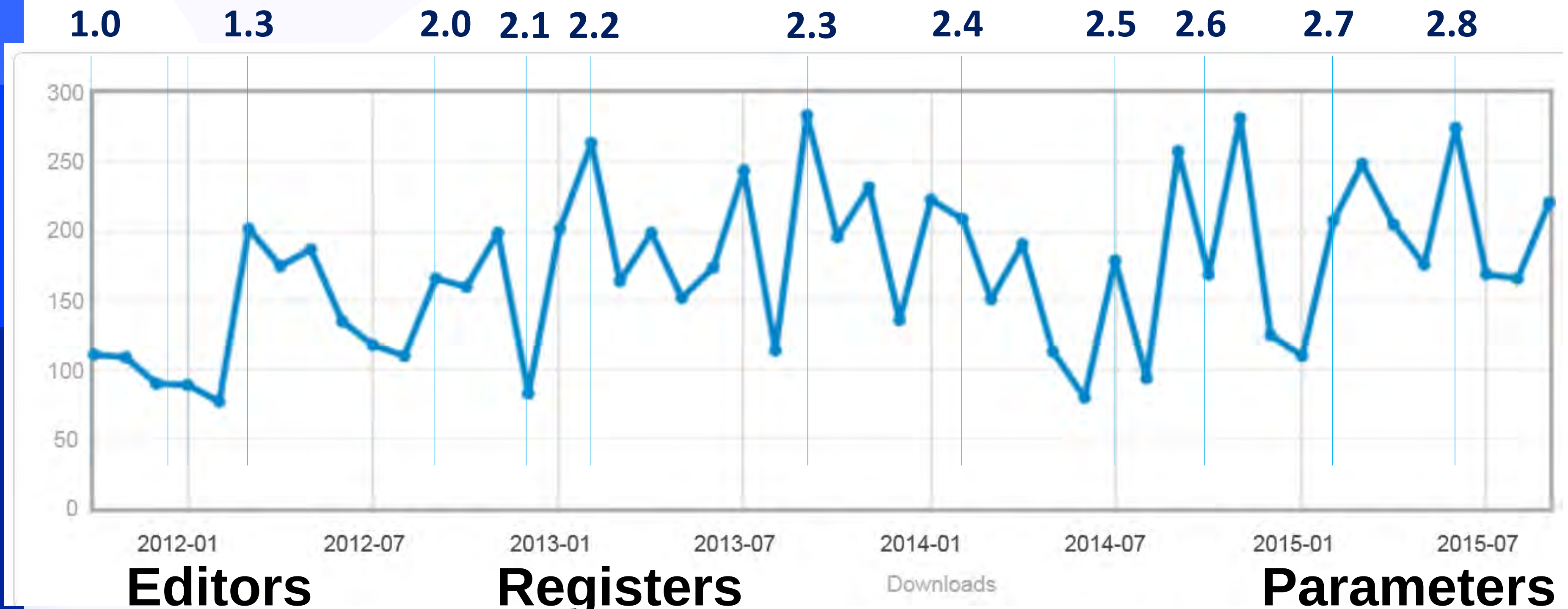


- Language: C++/Qt
- SDK: VisualStudio
- Runs on Win, Linux, (Anrdoid)
- Key contributors at TUT
- Requirements, feedback, support provided by companies



# Kactus2 Releases

- Three main waves of development
- 3.0 upgrades to 1685-2014 (Oct 2015)
- 8k+ downloads



**Editors**
**Generators**

**Registers**
**Import wizards**

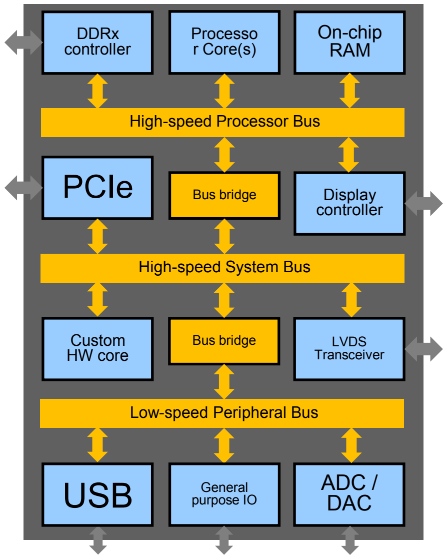
**Parameters**
**Configuration**

# Call for contributions

Kactus2 core development

Kactus2 plugins

**http://kactus2.cs.tut.fi**
**http://funbase.cs.tut.fi**
**kactus2@cs.tut.fi**



Best practise design examples

**Coding camp for your design?**

We are pleased to help you - maybe get together and code together?