HDL Simulation Framework

Tips and Tricks for HDL simulation

Dirk Hutter

Motivation

- With bigger FPGAs system complexity increases
- Time spend for simulation and verification also increases

Question:
How can simulation be handles in an efficient way?

Outlook

- Simulation basics
- HDL Simulation Framework
- Gatel-level simulations
- Tips and Ticks

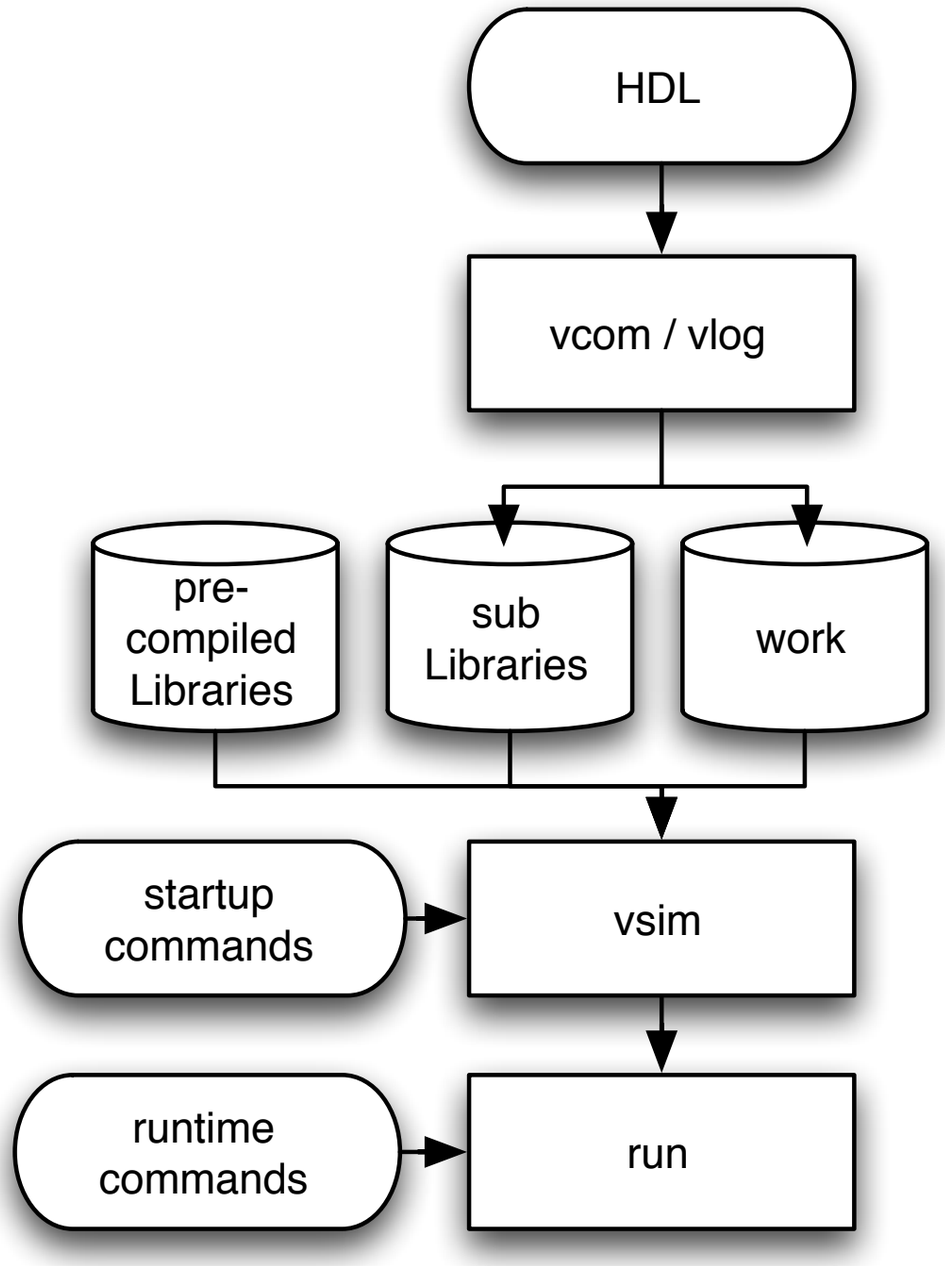
Concepts for efficient Simulations

- Simulation flow
  - Use an automated simulation flow
  - Add automated tests to your simulations
  - Use GUIs only for investigations and debug
  - Start simulations from basic sources
  - Be able to simulate your design at each build step

Concepts for efficient Simulations

- Testbench and stimuli
  - Simulations should be self checking and end automatically
  - Add pattern generators and checkers to your testbench or design
  - Integrate simulation debug output to your design
  - Prefer language constructs before simulator commands
  - Implement a full system simulation where possible

Modelsim Simulation Flow

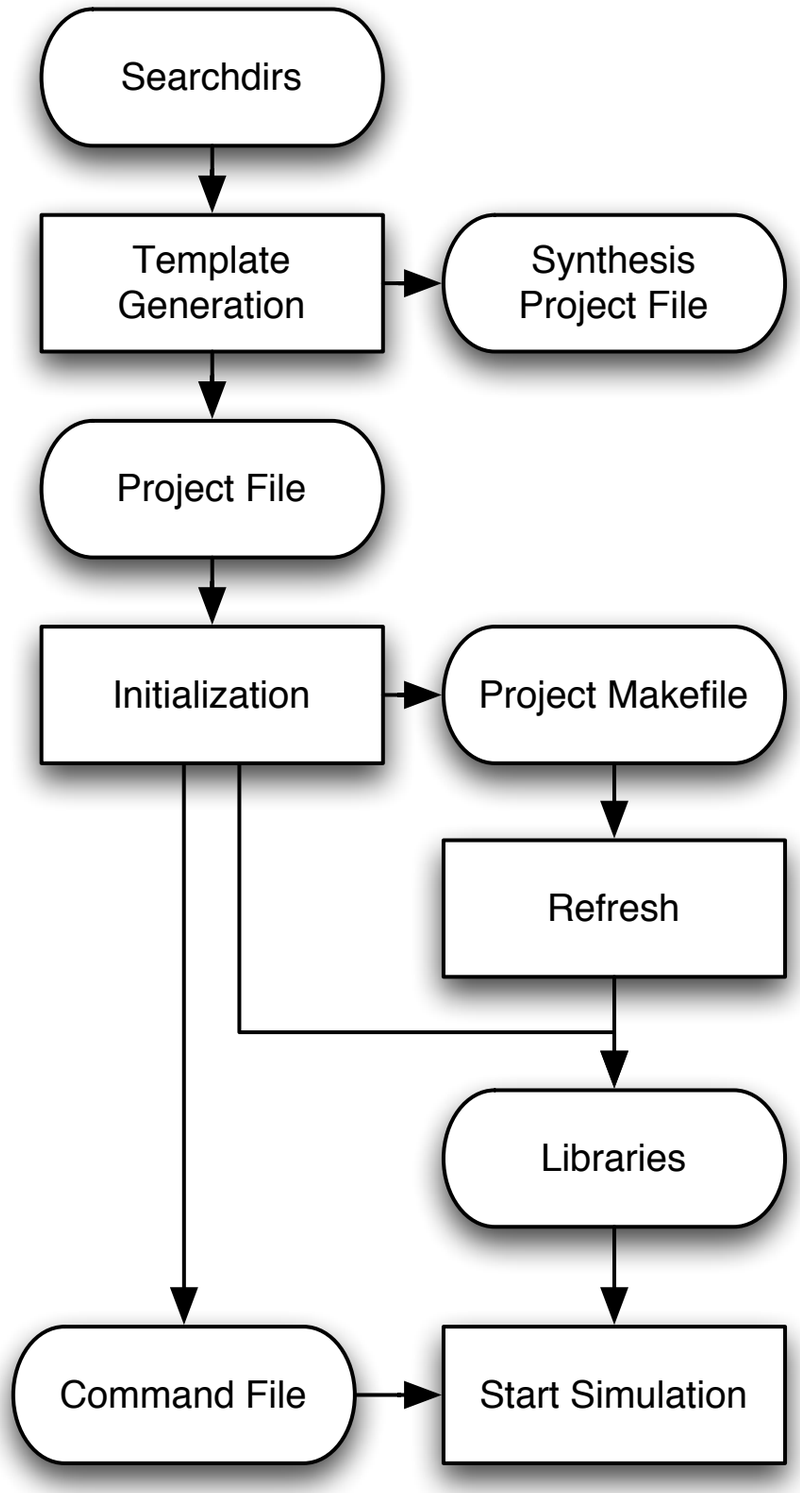


- HDL files are compiled to libraries
- For VHDL compile order is bottom up
- Library items are loaded for simulation using vsim
- During simulation runtime commands for manipulations and examinations are available
- How to automate?

Simulation Framework - Features

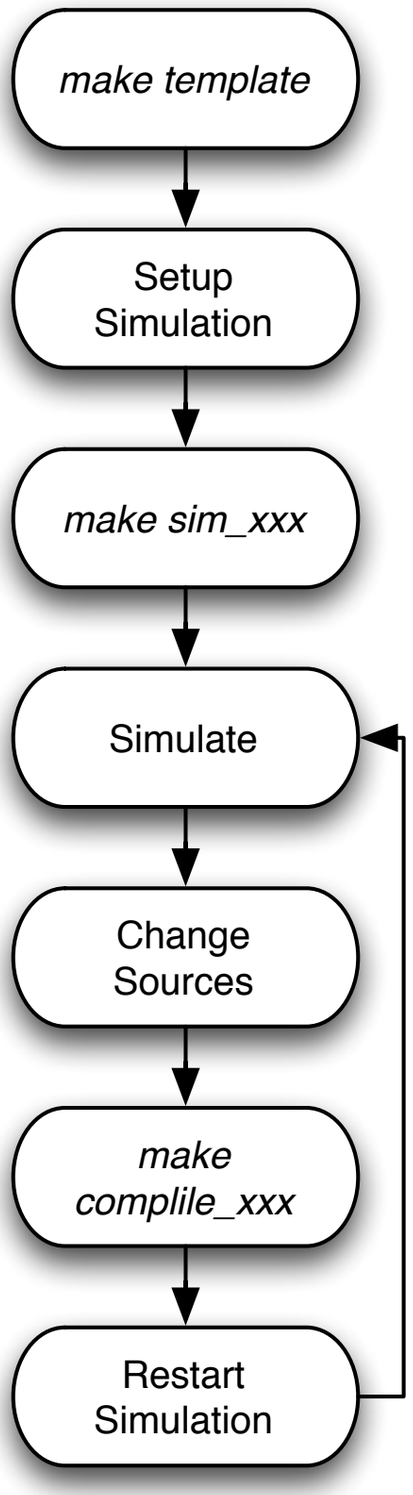
- Makefile based automated flow
- A single project file defines the whole simulation
- Ability to define
  - Sources and libraries
  - Back annotation paths for timing simulation
  - Startup and runtime commands or scripts
- Out of tree build is supported
- Each simulation is build into its own directory
- Libraries and framework files are in subdirectory
- Easy include of stimulus generation and output checking

Simulation Framework



- A templates is created from available sources and can be modified
- Initialization
  - Create needed libraries
  - Compile sources into chosen libraries
  - Create a makefile for each library using vmake and a common makefile for all libraries
  - Create a command file for vsim
- Refresh effected simulation sources
- Start the simulator with the given commands

How to use the Framework



- *make template* to crate a projectfile template
- Modify the template to match your simulation needs
- *make sim_xxx* to start your simulation
- Simulate and change sources
- *make compile_xxx* to refresh libraries
- For debug simulations restart the simulation in the GUI (without restarting the GUI)

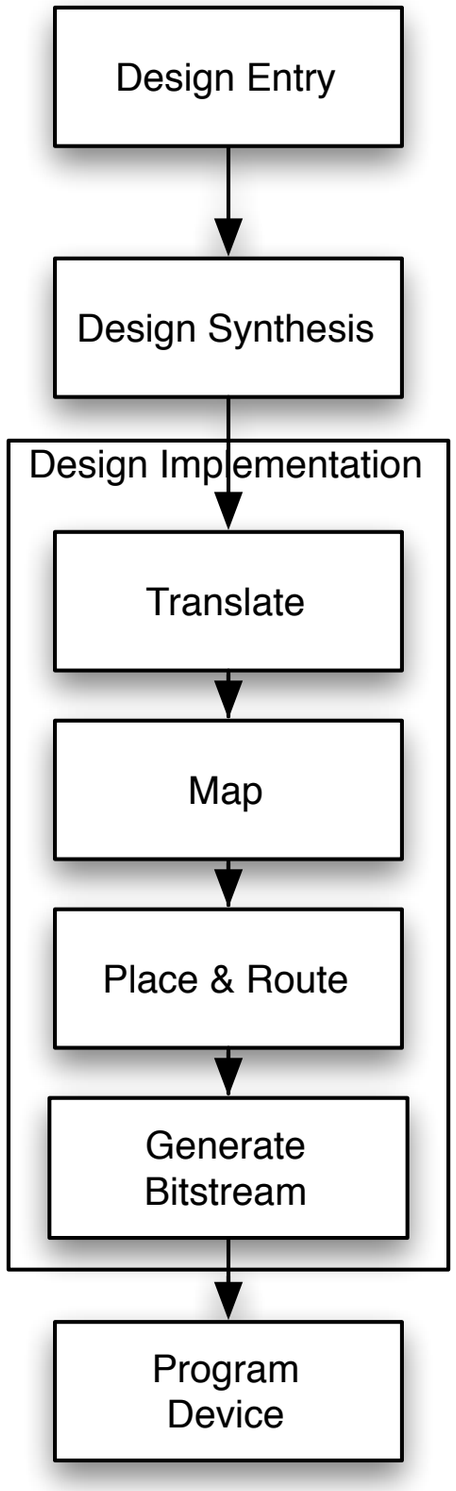
Project File

```
# Commands
[cmd]-do "do ../do/wave.do; do ../do/run.do"
#################################
[lib] TMU
# Synthesis files from ../../../script/files.txt
../../../src/misc/buildstamp.vhd
../../../src/top.vhd
#################################
# Simulation files from ./sim_files.txt
../../../common/src/fifo/fifo_dc18.vhd
../../../common/sim/src/CY7C1320BV18.vhd
# Board-Level Simulation Entities
./src/tmu.vhd
# Testbench Sources
[lib] work
./src/tmu_tb.vhd
```

Gate-level Simulations

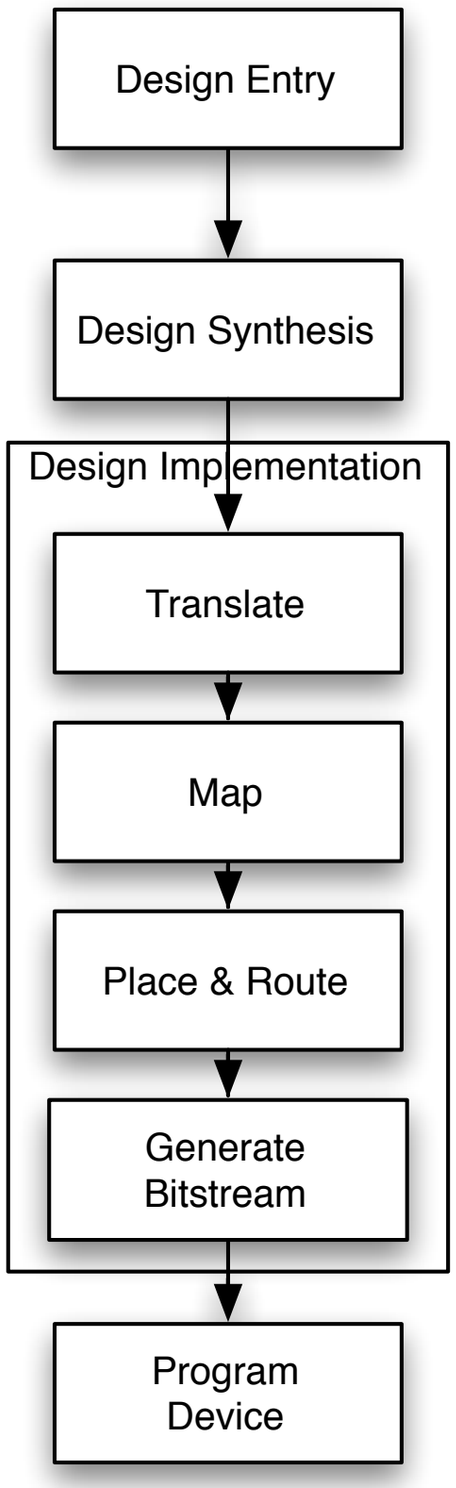
- Behavioral / functional simulations are not sufficient
- Gate-level and timing simulations get more and more important for FPGAs
- Examples:
  - Differences between RTL and implemented design due to (unintended) optimizations
  - Incomplete or wrong constraints
  - Static timing analyzer used for implementation won‘t find runtime problems like BRAM collisions

Xilinx Design Flow



- Synthesis (XST)
  - Translates HDL design into a structural netlist of primitives
- Translate (NGDbuild)
  - Reduces netlists to Xilinx primitives
- Mapping (MAP)
  - Maps logic to FPGA components, pre placement
- Place & Route (PAR)
  - Place and route of mapped design

Simulation Points



- Behavioral:

  Check RTL sources for intended function

- Post syn /trans:

  Check for unintended syntheses results

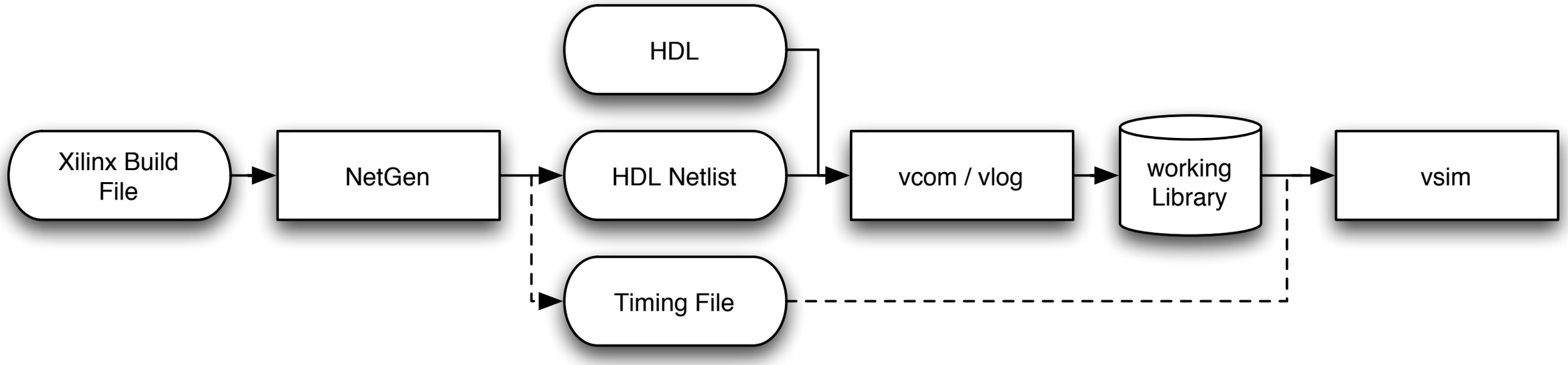
- Post PAR

  Gate-level simulation with partial timing informations (only block delay)

- Post MAP

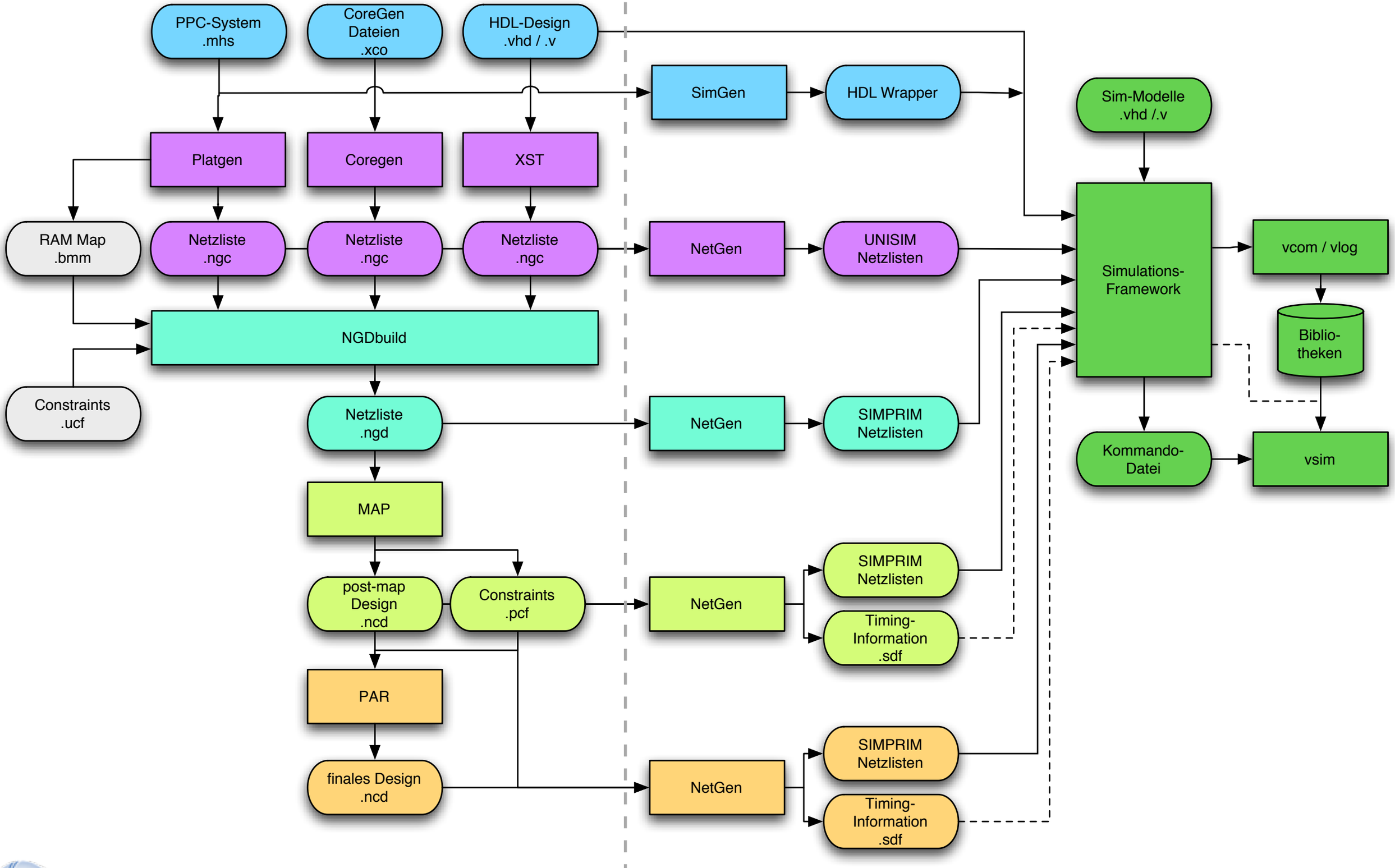
  Gate-level simulation with full timing information (block and net delay)

Gate-level Simulation for Xilinx FPGAs



- NetGen transforms files from the design flow into HDL netlists
- For each component with a „keep_hierarchy“ attribute a separate netlist can be created
- Timing information are written to a separate SDF file
- SDF file holds min, max and typ values of timing

Integration to Simulation Framwork



Tips and Tricks

Hierarchical Signals

- Often signals from other hierarchy levels are needed
- Records for debug signal or VHDL packages are often used
- VHDL-2008:

```
-- hierarchical signal name
<< signal .tb.uut.o_n : std_logic >>
-- signal a two levels above
<< signal ^.^.a : std_logic >>
-- variable in a package pack
<< variable @lib.pack.v : bit >>
```

- Can be used in testbenches in Modelsim 10.x

Unimportant Warnings

- Get rid of unimportant warnings to see crucial ones
- Arithmetic warnings before reset

```
# Switch off arithmetics warnings

set StdArithNoWarnings 1
set NumericStdNoWarnings 1

# Switch on arithmetics warnings after reset
when -label enable_StdWarn {rst = '0'} {
  echo "### Enabling arithmetic warnings" ;
  set StdArithNoWarnings 0;
  set NumericStdNoWarnings 0}
```

- BRAM collision checks:

  SIM_COLLISION_CHECK generic in BRAM primitive

Modelsim Breakpoints

- Breakpoints can be used to steer simulation progress
- E.g. stopping a simulation

```
onbreak {resume}
when {/testbench/done_condition = 1} {
  if {[examine /testbench/error] = 0} {
    echo "### SIMULATION PASSED"
  }
  if {[examine /testbench/error] != 0} {
    echo "### SIMULATION FAILED"
  }
  stop
  if [batch_mode] {
    quit -f
  }
}
run -all
```

Modelsim Virtual Signals

- Signals can be combined to virtual signal and buses
- Virtual types can be defined
- Useful for signals not available on testbench level and gate-level simulations

```
# create virtual intermediate bus of control signals
virtual signal -install /sim_tb_top { (context /sim_tb_top )
( ddr3_cs_n_fpga(0) & ddr3_ras_n_fpga & ddr3_cas_n_fpga &
ddr3_we_n_fpga )} controls

# define types for control signals
virtual type {{0b1000 DES} {0b0001 REF} {0b0011 ACT} {0b0101 RD}
{0b0100 WR} {0b0110 ZQCS} {default OTHER_STATE}} myCtrlType

# cast controls to myCtrlType and save as myCtrlState
virtual function {(myCtrlType)/sim_tb_top/controls} myCtrlState
```

Thank You for Your Attention!

Contact:

Dirk Hutter

hutter@compeng.uni-frankfurt.de

Prof. Dr. Volker Lindenstruth

Lehrstuhl für Architektur von Hochleistungsrechnern
Frankfurt Institute for Advanced Studies

http://www.compeng.de