Demo RadioHDL ← hdltool_readme.txt, readme_libraries.txt

9 sep 2014

UniBoard_Fly / <application>
/ UniBoard ← $UNB
/ RadioHDL / ← $RADIOHDL

Environment

- bashrc — tool environment setup scripts for RadioHDL and 'old' style UNB
  - setup_radiohdl.sh
  - setup_unb (also for Aartfaac and Paasar)

- RadioHDL tool start scripts
  - run_modelsim unb1 (6.6c) unb2 (10.2)
  - run_quartus unb1 (11.1) unb2 (13.1a10)
  - unb2_*

lp <name>
mk <name>
as #
ds

Configuration files

- Tool config file ← sim_tool_name = modelsim, model_tech_dir, synth_tool_name = quartus
  - hdltool.cfg (one central)
- Library config file — modules, designs → now all called libraries
  - hdllib.cfg (one per library)

The configuration files define how the tools should build the HDL libraries to create the targets. The targets are:

t1 compile for simulation
t2 synthesize to creat hw image
t3 regression test in simulation of VHDL test benches
t4 " " " " " Python test cases (via MM
t5 " " on hardware " " " " " "
t6 zip file of library)

common_dict_file.py
hdl_config.py

modelsim_config.py
quartus_config.py

— import modelsim_config
help(

Build directory location

local
<hdl_lib_name>/hdllib.cfg
modelsim/
quartus/
src/
tb/

central
$HDL_BUILD_DIR/quartus/<hdl_lib_name
/modelsim/

rm -rf

▨ Simulation

```
> rm -rf build
> python modelsim_config.py     — finds all hdllib.cfg in rootdir
                                   and creates mpf for all libraries
> run_modelsim &
      >> lp eth
      >> lp all                      in $HDL_BUILD_DIR
      >> mk compile all
      >> double click tb.eth icon
      >> as 5
      >> .ds
      >> as 10
      >> run -a
```

self checking
self stopping
↳ tb_tb_tb_eth_regression.vhd.

▨ Synthesis

```
> python quartus_config.py     — finds all hdllib.cfg in root dir and
                                  creates qpf, qsf, qip for all
                                  design libraries that have a
                                  synth_top_level_entity key.

> run_quartus unb1 &
      >> open unb1_minimal

  or

  unb2_qcomp unb1_minimal.
```

- revisions in hdllib.cfg is possible but still to do.

▨ RadioHDL / tools / hdltool.cfg
modelsim
quartus
oneclick

/ libraries / base — common, dp, mm, uth ...
external — easics
io — epcs, i2c, eth, tr_10GbE
technology — ip_stratixIV
ip_arria10
fifo
flash
tse
transceiver
technology_pkg.vhd
hdllib.cfg.

▨ Technology independence

technology / ip_stratixIV / ip_arria10 — ram, fifo, ddr, asmi_parallel, gx
tse_sgmii_gx /
tse_sgmii_lvds /
ram (only ip_arria10_ram_crw_crw.vhd done)

transpose
ip_ = all IP components for 1 device technology
tech_ = all device technologies for 1 IP component

memory / tech_memory_ram_crw_crw.vhd
↓
io / memory / common_ram_crw_crw.vhd ← g_technology = c_tech_stratixIV
or c_tech_arria10

☑ UniBoard² io. test design

boards / uniboard1 / designs / unb1_minimal ← sopc qsys
/ unb1_test ← Leon
/ libraries / unb1_board / src / vhdl

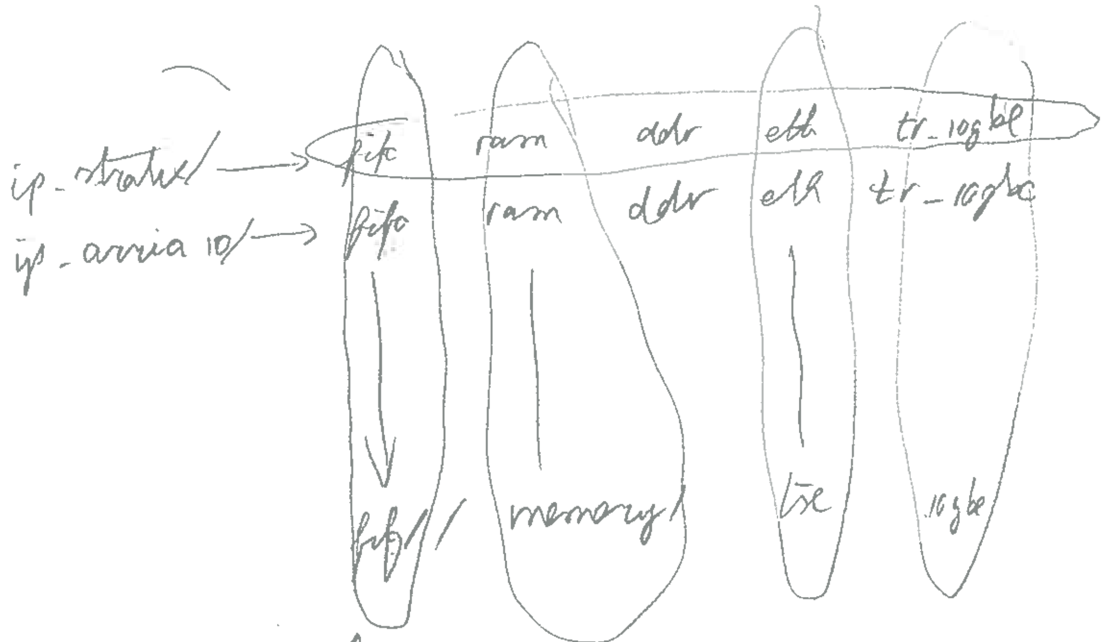
used to be in $UNB unb_common/.
e.g. ctrl_unb_common is now ctrl_unb1_board

/ uniboard2/ designs / unb2_pinning ← Jonathan
/ unb2_minimal

↳ same functionality as unb1_minimal but for Arria10 on UniBoard²

/ unb2 test.

- The transpose map between IP and tech needs to be done first before the unb2 equivalent of a unb1 design can be made.
- Trial designs for Arria10 like unb2_pinning can also be made if necessary to quickly investigate the DDR4, GX, flash IP for Arria 10.



component packages dip