Altium

ALTIUMLIVE
Placing an FPGA on a board
Guidelines for efficient design

Willem Gruter
HDL Works
Product Manager

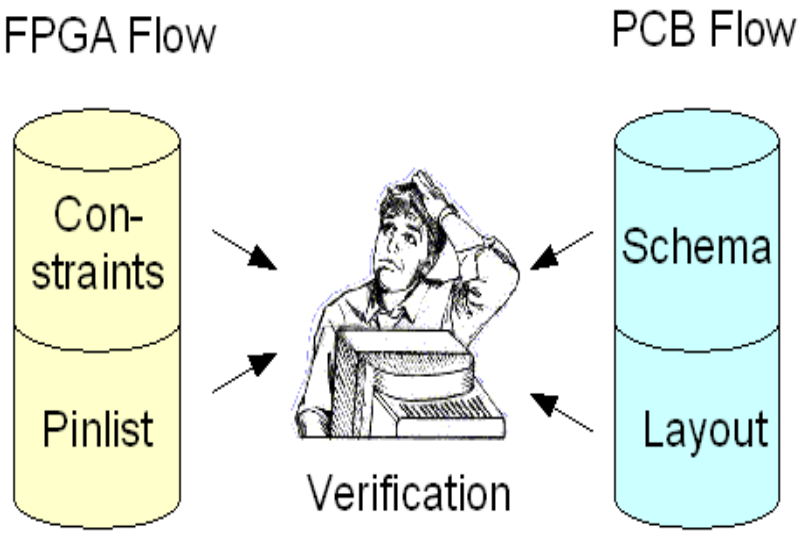
Münich
17 Jan

Agenda

1. **FPGA on a board**
2. Tasks to perform
3. Guidelines
4. Verification
5. Conclusions

FPGAs and PCBs

- Often different engineers
- Various development flows
  - PCB first / FPGA first / simultaneously
  - Generic vs Application specific symbols
  - FPGAs are flexible but pin assignment determines PCB performance
- Some errors require a board re-spin



FPGA board challenges

- Many different power pins

  Required voltage can be different based on IO types,

  frequencies and cores (PLLs, ethernet, highspeed pins) used on the devices
- Dual function pins: configuration pins become user IO after boot
- User IO pins can change after a synthesis run
- Memory interfaces can require specific pins
- Alternate device (smaller/larger device in the same package) can change pin functionality (Altera Cyclone)
- Pin swaps need to be synchronized on both sides of the development

FPGA pins

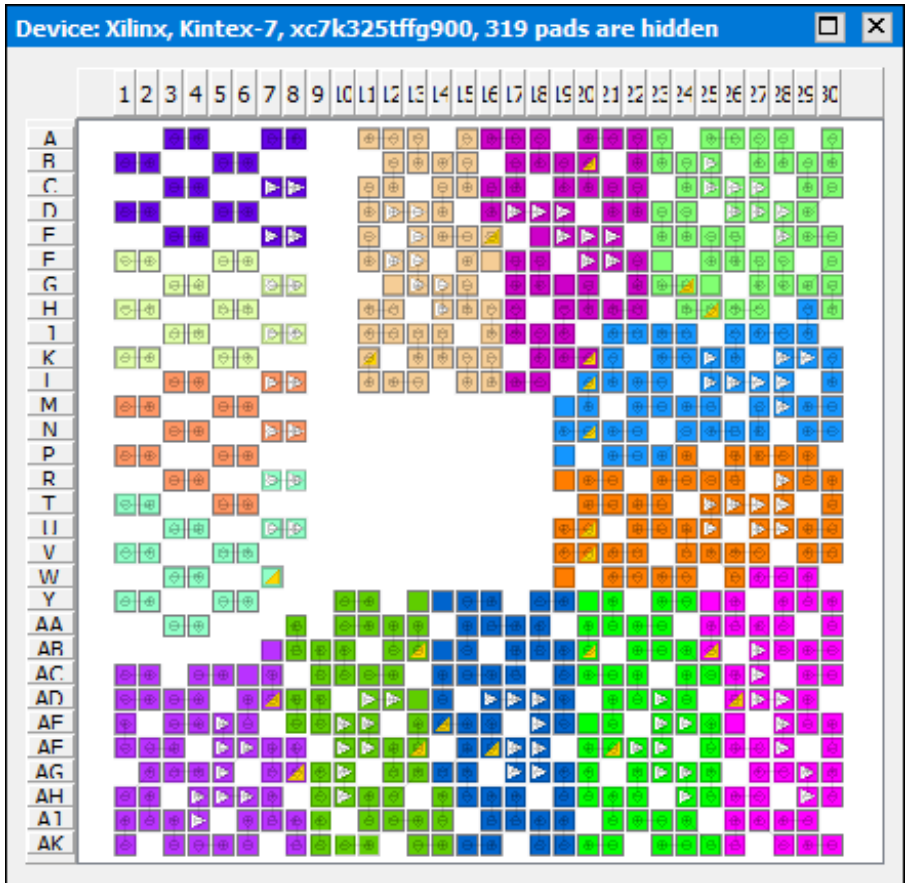
- Pin count of FPGA devices range from 32 to 2912 (Intel Stratix 10)
  - About 50 to 75% of the pins is user IO

    (the more complex the device the more

    none user IO pins)
- Non-user IO

  Control pins, JTAG, Power, ground, do-not-use
- User IO
  - Special (Clock capable, high speed, VREF)
  - LVDS pairs
  - Unused

    (leave floating, connect to ground ?)



Design steps

- Symbols for your FPGA
- Place those symbols on schematic pages
- Add the connectivity
- Verify schematic design with FPGA design
- Board design
- Re-verify design with FPGA design

FPGA Vendors

Main suppliers:

- Intel (Altera)
- Lattice Semiconductor
- Microsemi
- Xilinx

Small players:

- Achronix
- Cypress
- Gowin (New, Chinese)
- Quicklogic

FPGA Tooling

Common base:

- Verilog / VHDL for functional behaviour
- Constraints file(s)

Differences:

- Working with LVDS signals
- Quality of the pin out reporting
- Where they put the data

Intel (Altera)

- Project file has extension ‘qpf’
- Constraint file has extension ‘qsf’
- Pin report is output_files/<project_name>.pin
- CSV file can be generated in the Pin Planner: File → Export

Most complete pin report, specifies all user IO, control, power and ground.

No specification for dedicated control signals which should be GND or VCCIO (Like MSEL1)

Lattice Semiconductor

- Project file has extension ‘ldf’
- Constraint file has extension ‘lpf’
- Standard pin report is <project>_impl.pad

A better report file is the Pin Layout file. This file needs to be generated explicitly.

To generate this file open the Spreadsheet View in Diamond and select 'File ->Export -> Pin Layout file...' from the menu bar.

The Diamond software does not report the required voltage on regular power pins (like VCC, VCCA and VCCAUX)

No specification for dedicated control signals which should be GND or VCCIO (Like CFG1)

Microsemi

- Project file has extension ‘prjx’
- Constraint files have the extension 'pdc'
- Two pin report files(‘txt‘ and ‘rpt‘):
  - By name (Only reports user IO)
  - By pin number (Reports all)

The Microsemi Libero software does not report any voltage information for power pins

No specification for dedicated control signals which should be GND or VCCIO (Like M0_1)

Xilinx

- Project file has extension ‘xpr’
- Constraint files have extension ‘xdc’
- Pin reports are in <project_name>.runs/impl_1(2,3,4)/

  ‘<project_name>_io_placed.rpt‘ for the user IO

  ‘<project_name>power_routed.rpt‘ for most power (voltage) signals
- CSV file can be generated using File → Export

Voltage information for most pins (but not all)

No specification for dedicated control signals which should be GND or VCCIO (Like M0_1)

Why the guidelines

- Verification is a time-consuming job
- Quite often it is done only once

  After design changes you assume it is done correctly on both sides
- Even when doing a manual verification consistency pays off
- Much higher gains with automated verification

  (Allows re-verification with little effort)

Guidelines on wire names

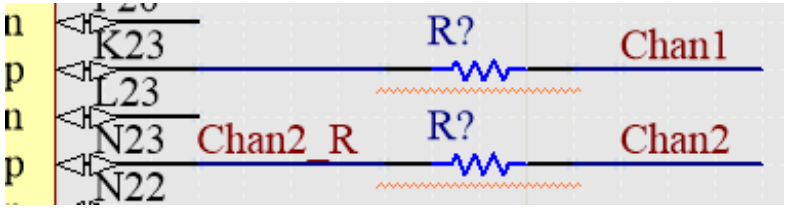
- Consistency
  - User IO
  - Power
  - Control
  - LVDS
- Put information in the signal names
- Unconnected pins:
  - Put a note or noDrc marker
  - Place net with <pin_nr>_NC name

User IO

- Easiest: use equal names
- Use a consistent style for names
  - prefix
  - suffix
- For series resistor or capacitor use a prefix to denote signal name on opposite side

  reboot ↔ reboot_R or reboot ↔ reboot_C
- Prevent system generated names:

  1N201328



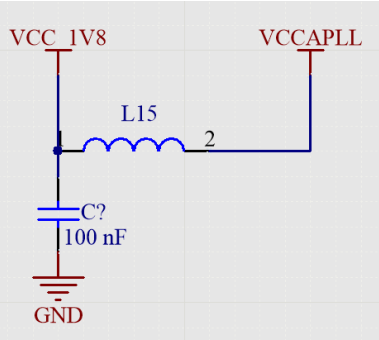
Power pins

- Put use in the signal names
  - Include the voltage level in the pin name:

    VCCINT_+1.2V, VCCIO3_1.8V, VACC_AUX_DP1V2
- Do this also for decoupled powers

  MGTAVTT_1.8V
- Could add physical location if useful (position of noise capacitors):

  top, NW (NorthWest)

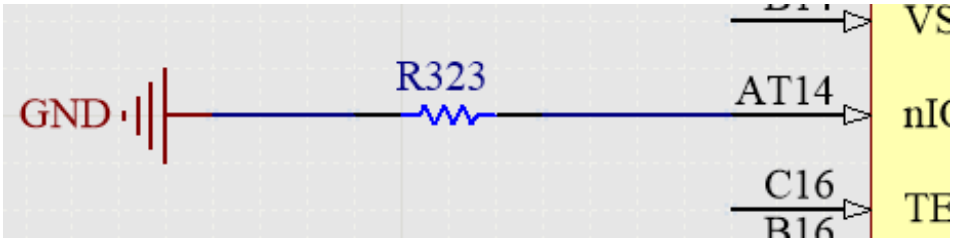


Control pins

- Use names that relate to pin functional name on the FPGA

  For JTAG pins FPGA_TCK, FPGA_TMS
- For control pins with direct pull up/down add the value to the name

  M1_0_to_GND or CFG0_3V3



Consistency is not always possible

- Company standards on VHDL or Verilog
  - i<name> for inputs
  - o<name> for outputs
  - b<name> for bidirectional
- FPGA control pins
- FPGA tools create different signal pair names
  - Lattice: mem_dqs0 → mem_dqs0+ and mem_dqs0-
  - Intel: mem_dqs0 → mem_dqs0 and mem_dqs(n)
- PCB tooling
  - AltiumDesigner requires ’_n’ and ’_p’ on LVDS pairs

Help in schemactic connectivity

- Altium: Pin Mapping
  - Uses CSV file from Intel or Xilinx
- IO Checker FPGA
  - Uses its own project file based on CSV, constraints or pin files

Verification issues

- User IO
  - Different signal names in both environments
  - Bus signals in FPGA vs flat signals in the schematics
  - LVDS signals (Altium requires _p/_n at the end of the signal name)
- Power, ground, NC pins
  - P3V0 or +3.0V in the schematic while 3.00 or ‘any**’ in the report file
  - Decoupled circuits
  - Pull-up / pull-down circuits
- Needle / haystack problem
  - A lot of pins / data and probably few (potential problems)

Pin connectivity verification

- Manual, iterating through all the FPGA symbols in the schematic

  Very time consuming

- Script based:

  Extract data from schematic / board (Altium Pin Mapping)

  Extract data from FPGA tool (Increasing difficulty: Intel, Xilinx, Lattice and Microsemi)

  Spreadsheet style comparison

  A lot faster, still requires organising data, doesn't help when signal names differ

- Tool based:
  - IO Checker FPGA

    Very fast, rule engine for name differences and very easy re-verification

IO Checker FPGA

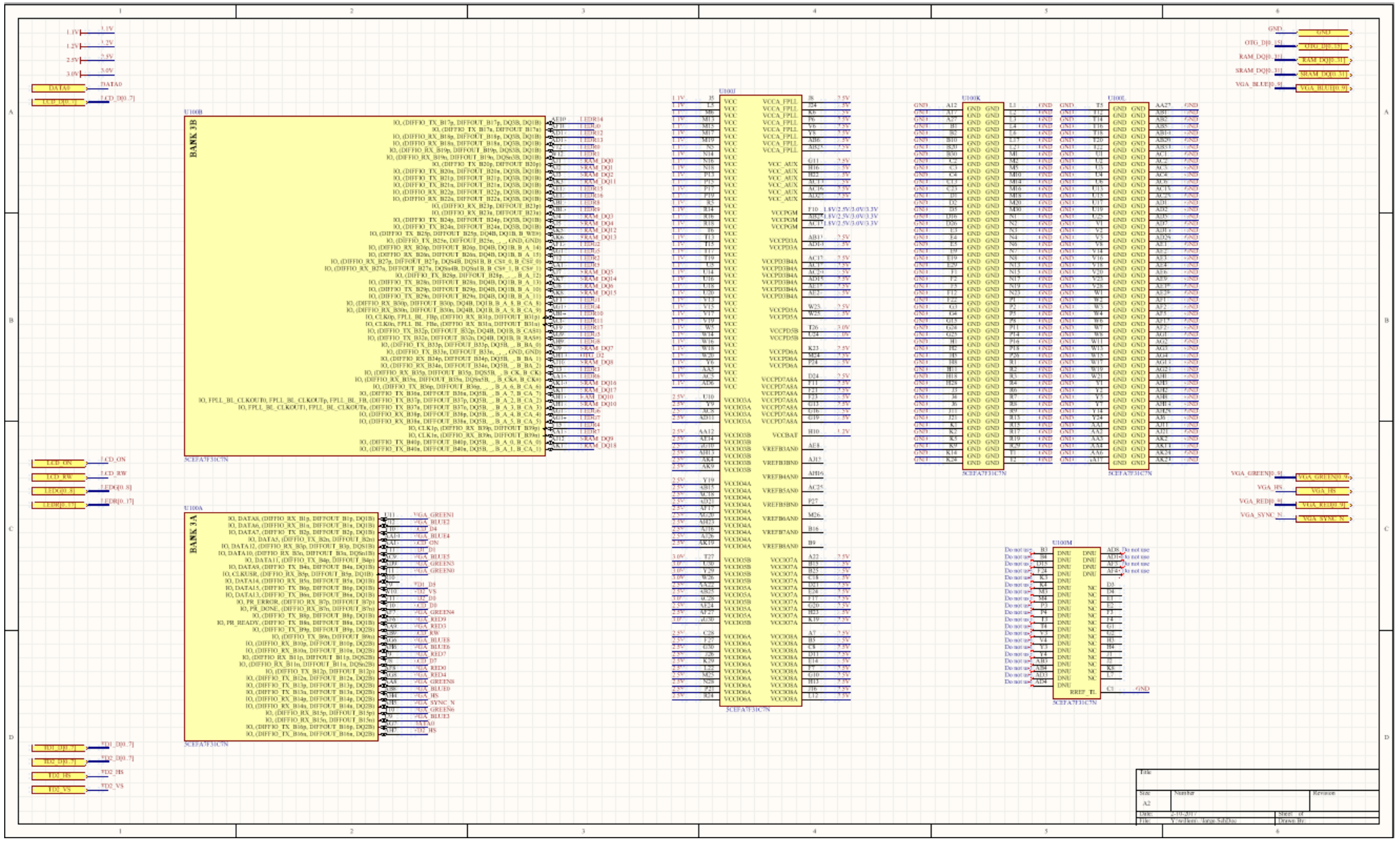
- Support all work-flows
- Works with CSV, constraints or pin data
- View showing all data: constraints, pin and pcb combined with filters and sorting.

  Additional info like pull-up, pull-down resistor, un-connected.
- Rule based verification

  Match groups like mem_ddr_dq<3> (fpga) to ddr3_dq<3> (pcb)
- Voltage verification for power pins
- Schematic wiring based on CSV, constraints and pin file

  with additional options like LVDS parameter, noDrc.
- Constraint generation based on schematic data

Example wiring



Altium®

Questions ?

You can find me in the Expo
for a demonstration or discussion !