Altium

Verifying that your FPGA is correctly connected on a PCB

AltiumLive 2017:
ANNUAL PCB DESIGN SUMMIT

Willem Gruter
Product manager

München
October

Agenda

1. **Problem description**
2. IO Checker approach
3. Verification
4. Design
5. Conclusions

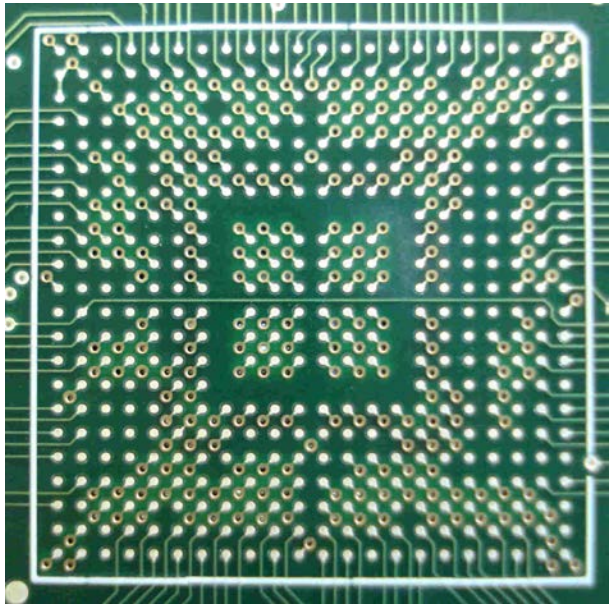
The problem

The FPGA and PCB are designed using different tools.
With over a thousand pins how do you verify each pin is correctly connected ?

- User IO
- LVDS pairs
- Power & ground

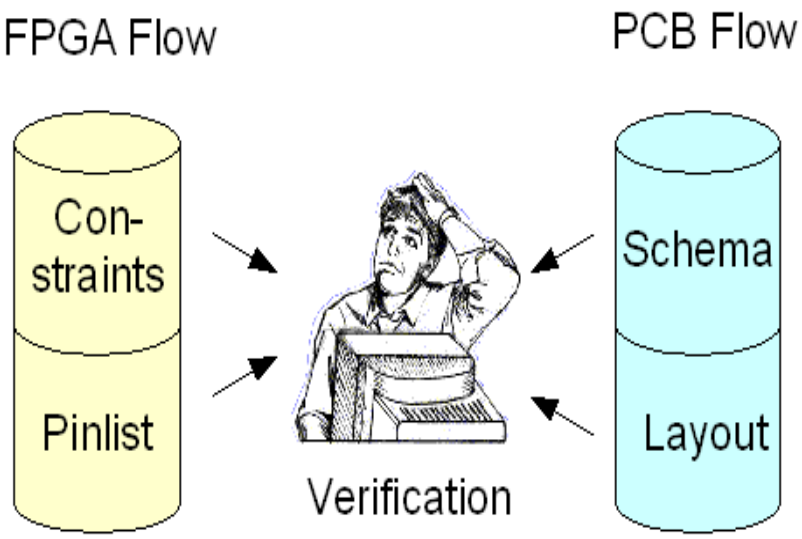
How often do you need to repeat this process:

- Because changes were made to the FPGA
- Because pins were swapped



FPGAs and PCBs

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



Manual verification procedure

- Extract netlist from schematic capture / PCB
- Limit netlist to the FPGA (reference designator)
- Sort by FPGA pin name
- Compare
  - By hand
  - Using a script (needs to be developed / maintained)

Manual verification is

- Time consuming
- Error prone
- Dull work

- What to do after pin swaps or ECOs
  - Repeat steps ?
  - Assume it is OK ?

Verification issues

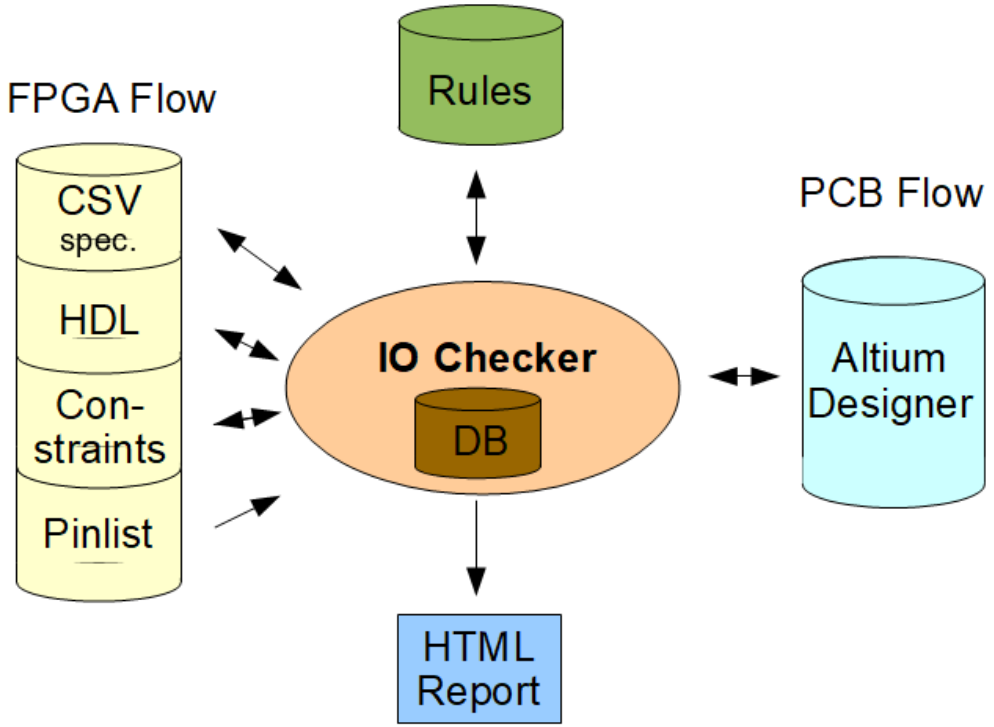
- User IO
  - Signals names in both environments are not exactly the same
  - Bus signals
  - LVDS signals
- Power, ground, NC pins
  - P3V3 or +3.0V in the schematic capture versus 3.00 or 'any**' in the report file
  - Decoupled circuits
  - Pull-up / pull-down circuit
- Needle / haystack problem
  - A lot of pins / data and probably a few (potential) problems

IO Checker

- Verifies FPGA vs PCB pin assignment
  - Fuzzy matching, rule based matching, user accepted
  - Power, ground, VCCIO banks
  - LVDS pairs
- A smart way to concentrate on the differences
- Report
  - Out-of-date files
  - Changes in pin assignment (FPGA and PCB)

The approach

- Collect the data:
  - FPGA pin file(s)
  - FPGA constraints file(s)
  - PCB netlist + reference designator
- Match signals
- View / sort problems
- Improve rules
- Identify the potential problems



Matching

Matching steps:

- Direct matches
- Fuzzy matching
- Family / Power matching
- Rule matching
- User accepted mismatches
- User accepted power mismatches

Fuzzy matching

Automatically ignore differences in:

- Bus notation: (2) , [2], <2>
- Underscore characters
- Bus index and flattened names

DataAddress<15>
dataaddress[15]
data_address15
data_address_15

Are all the same

Family / Power matching

Automatically match:

- GND pins
- POWER (when voltage is recognized)
- NC
- Pins connected to ground like GND+ GND*

Rule matching

The rule engine allows you to

- Validate signals which have different names but are correct

  (Requires a relation between pin and PCB signal name)
- Use regular expressions with references to matched parts
- Allow +, - operators to match non aligned busses

databus_(\w)_<(\d+)> databus_pci_12	data_$1_$2 data_pci_12	Matches 'databus_<letter>_<digits>' with 'data_<same_letter>_<same digits>'
main_\w+(\d+) main_driver46	slice_$1-16 slice_30	Matches main_<letters><digits> with slice_<(digits - 16)>

- A rule generator helps you to define / refine rules

Last step

User accepted mismatches for

- Control signals
- Known differences

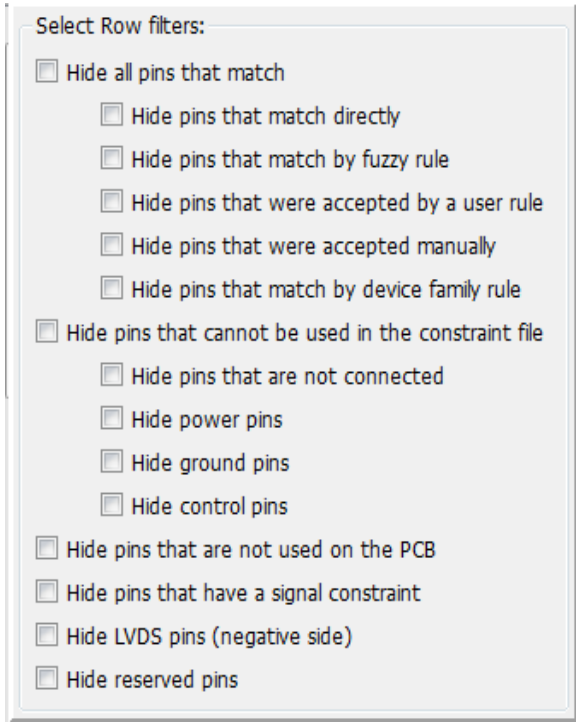
		Pin	Bank		Signal (Pin file)		Signal (PCB)	Rule
		AB7	3A		<AS_DATA3, DATA3>	☒		
		T8	9A		<MSEL0>		2.5V (=2.5V)	
		P9	9A		<MSEL1>		GROUND	
		G5	9A		<MSEL2>		GROUND	
		P7	9A		<MSEL3>		GROUND	

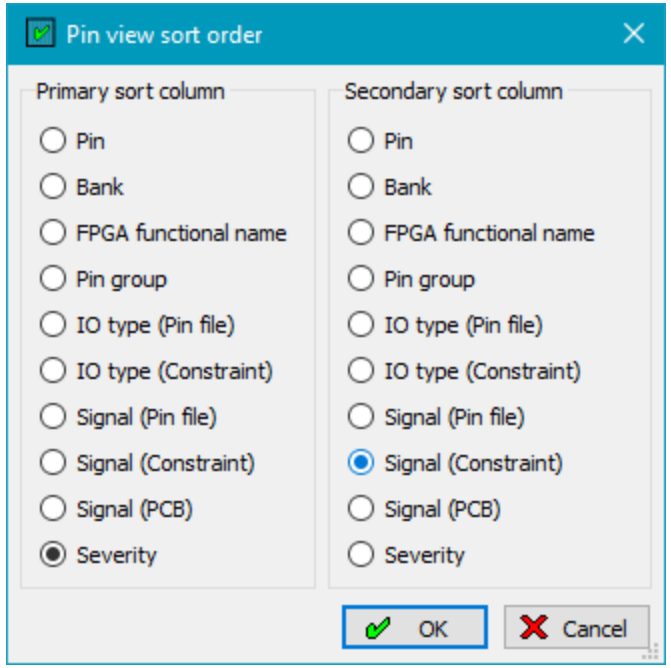
		Pin	Bank		Signal (Pin file)		Signal (PCB)	Rule
		AB7	3A		<AS_DATA3, DATA3>	☒		<User> (Not used)
		T8	9A		<MSEL0>		2.5V (=2.5V)	<User> (Control, verified)
		P9	9A		<MSEL1>		GROUND	<User> (Control, verified)
		G5	9A		<MSEL2>		GROUND	<User> (Control, verified)
		P7	9A		<MSEL3>		GROUND	<User> (Control, verified)

Organizing the view

You have a lot of data but you need to focus on the potential problems:

- Filters:
  - Hide direct / fuzzy / rules / user matches …
- Sort the view by problem severity instead of pin number





Sorting: unsorted

Pins: 672, unmatched: 19 (14 + 5), matched: 653 (210 exact, 65 ground, 81 power, 134 fuzzy, 7 device, 156 rule and 0 user)

(0) (3)

	Pin	Bank		IO type (Pin file)	Signal (Pin file)	Signal (Constraint)		Signal (PCB)	Rule
	R21		✔	<1.2V>				VCCA_PLL <= P1V2 (=1.2V)	<Power>
	R22		✔	<GND>	<GND>			GROUND	<Ground>
	R23	1		3.3-V LVTTL	ddr2_dimm_a<15>	ddr2_dimm_a<15>		ddr2_dimm_a_15	<Fuzzy>
	R24	1		3.3-V LVTTL	ddr2_dimm_a<14>	ddr2_dimm_a<14>		ddr2_dimm_a_14	<Fuzzy>
	R25	1		3.3-V LVTTL	ddr2_dimm_a<11>	ddr2_dimm_a<11>		ddr2_dimm_a_11	<Fuzzy>
	R26	1		3.3-V LVTTL	ddr2_dimm_a<12>	ddr2_dimm_a<12>		ddr2_dimm_a_12	<Fuzzy>
●	T1	6	✘	<3.3V>				P1V2 (=1.2V)	
	T2	6		3.3-V LVTTL	seg7_06	seg7_06		seg7_5	seg7
	T3	6		3.3-V LVTTL	seg7_05	seg7_05		seg7_4	seg7
	T4	6		3.3-V LVTTL	seg7_02	seg7_02		seg7_1	seg7
	T5	6		3.3-V LVTTL	rdn8a	rdn8a		rdn8a	<Direct>
	T6	6		3.3-V LVTTL	rdn6a	rdn6a		rdn6a	<Direct>
	T7	6		3.3-V LVTTL	rdn4a	rdn4a		rdn4a	<Direct>
●	T8	6		3.3-V LVTTL	rdn2a	rdn2a		GROUND	
	T9	6		3.3-V LVTTL	rdn1a	rdn1a		rdn1a	<Direct>
	T10		✔	<GND>	<GND>			GROUND	<Ground>
	T11		✔	<1.2V>				P1V2 (=1.2V)	<Power>
	T12		✔	<GND>	<GND>			GROUND	<Ground>
	T13		✔	<1.2V>				P1V2 (=1.2V)	<Power>

Sorting: by severity

Pins: 672, unmatched: 12 (7 + 5), matched: 660 (210 exact, 65 ground, 81 power, 134 fuzzy, 7 device, 156 rule and 7 user)

		Pin	Bank		IO type (Pin file)	Signal (Pin file)	Signal (Constraint)		Signal (PCB)	Rule
		R10	6		<3.3V>				P1V2 (=1.2V)	
		H12			<3.3V>				VCC_PLLOUT <= P1V2 (=1.2V)	
		AC1	6		<3.3V>				P1V2 (=1.2V)	
		T1	6		<3.3V>				P1V2 (=1.2V)	
		V13			<3.3V>				VCC_PLLOUT <= P1V2 (=1.2V)	
		A8	4		3.3-V LVTTL	databus_a<12>	databus_a<12>		data_a_21	
		F10	4		3.3-V LVTTL	databus_a<21>	databus_a<21>		data_a_12	
		T8	6		3.3-V LVTTL	rdn2a	rdn2a		GROUND	
		N20	2			<GND*>			rdn2a	
		AB5	7			<nCEO>				
		AB21	8			<TRST>			$2134	
		E16	3		3.3-V LVTTL	~DATA0~ / RESERVED_INPUT			RESERVED_INPUT	
		AE3	7		3.3-V LVTTL	speaker_out	speaker_out		speaker_out (dangling)	<Direct>
		G8	4			<MSEL0>			GROUND	<User> (Verified, c...
		B2	4			<MSEL1>			P1V2 (=1.2V)	<User> (Verified, c...
		E6	4			<MSEL2>			P1V2 (=1.2V)	<User> (Verified, c...
		F7	4			<MSEL3>			GROUND	<User> (Verified, c...
		E22	3			<nCE>			GROUND	<User> (Verified, c...
		F5	1			<TEMPDIODEn>				<User> (Verified, u...

Fixing user IO mismatches

IO Checker can propose changes in the constraints to resolve mismatches

Pins: 672, unmatched: 9 (4 + 5), matched: 663 (211 exact, 65 ground, 81 power, 134 fuzzy, 7 device, 158 rule and 7 user)

Pin	Bank		IO type (Pin file)	Signal (Pin file)	Signal (Constraint)		Signal (PCB)	Rule
H12		✗	<3.3V>				VCC_PLLOUT <= P1V2 (=1.2V)	
V13		✗	<3.3V>				VCC_PLLOUT <= P1V2 (=1.2V)	
T1	6	✗	<3.3V>				P1V2 (=1.2V)	
AC1	6	✗	<3.3V>				P1V2 (=1.2V)	
R10	6	✗	<3.3V>				P1V2 (=1.2V)	
T8	6		3.3-V LVTTL	rdn2a			GROUND	
AB5	7			<nCEO>		☒		
AB21	8			<TRST>			$2134	
E16	3		3.3-V LVTTL	~DATA0~ / RESERVED_INPUT			RESERVED_INPUT	
AE3	7		3.3-V LVTTL	speaker_out	speaker_out		speaker_out (dangling)	<Direct>
A8	4		3.3-V LVTTL	databus_a<12>	databus_a<21>		data_a_21	databus
F10	4		3.3-V LVTTL	databus_a<21>	databus_a<12>		data_a_12	databus
N20	2			<GND*>	rdn2a		rdn2a	<Direct>
G8	4			<MSEL0>			GROUND	<User> (Verified, c...
B2	4			<MSEL1>				
E6	4			<MSEL2>				
F7	4			<MSEL3>				
E22	3			<nCE>				
F5	1			<TEMPDIODEn>				

Constraint generation

Constraints suggested by generator

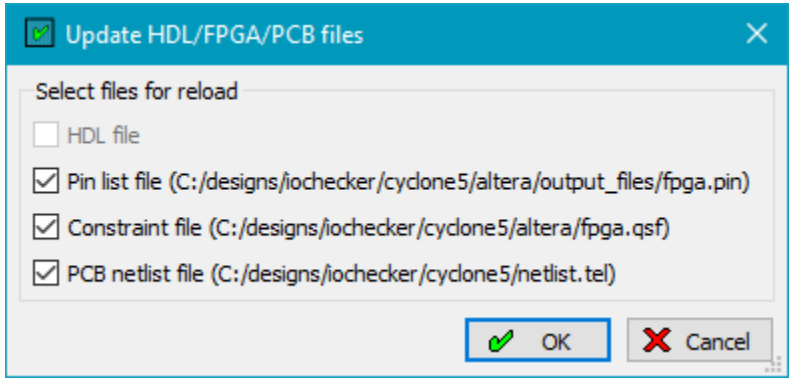
		Pin	Old FPGA Signal Name	New FPGA Signal Name	IO Standard	PCB Signal Name
1	☑	A8	databus_a<12>	databus_a<21>		data_a_21
2	☑	F10	databus_a<21>	databus_a<12>		data_a_12
3	☑	N20	GND*	rdn2a		rdn2a
4	☑	T8	rdn2a	<NONE>	<NONE>	GROUND

OK Cancel

Incremental update

Once a project has been setup you can:

- Easily re-import (changed) files
- Modified files are pre-selected
- Reports changed assignments, IO standards and power properties



From verification to design

When started with the FPFA design:

- Add wires / ports to symbols in a schematic

When started with schematic capture:

- Create FPGA constraints
- Create top level VHDL / Verilog unit

Adding wires / ports to symbols in a schematic

Sources:

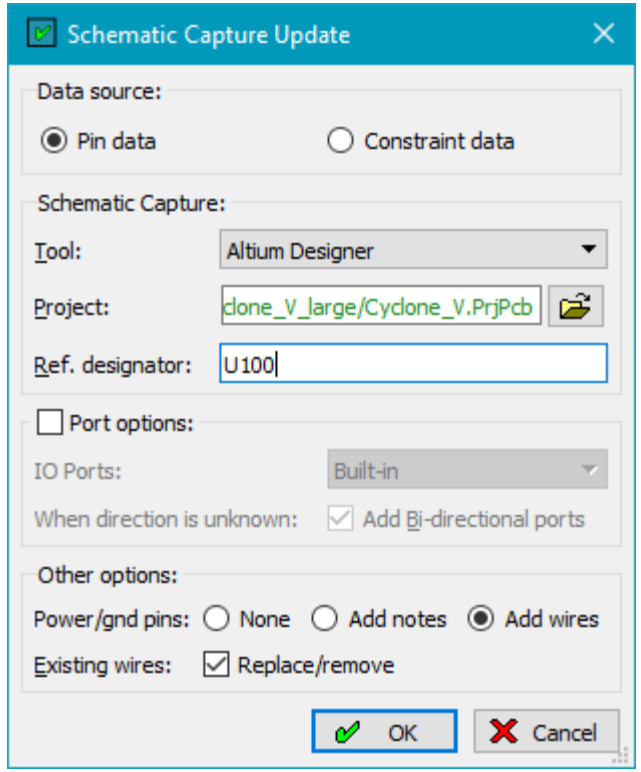
- Constraints, pin file, CSV file

Options:

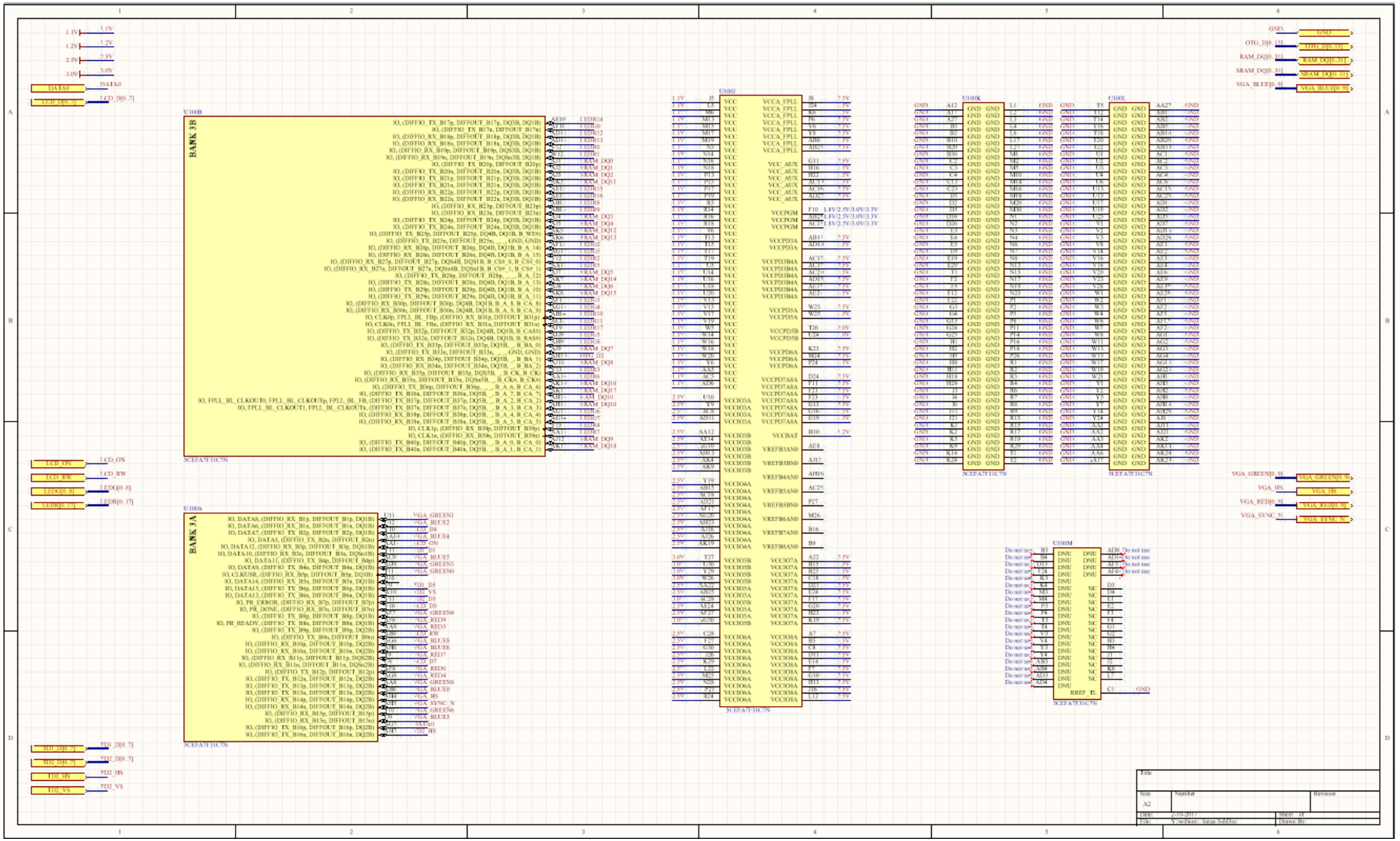
- Adding ports
- Adding notes or wires for power & ground pins
- Updating existing schematic

Configuration file based options:

- Wire length
- Adding LVDS DIFFPAIR parameters, No ERC objects
- Symbol type of Power & ground ports

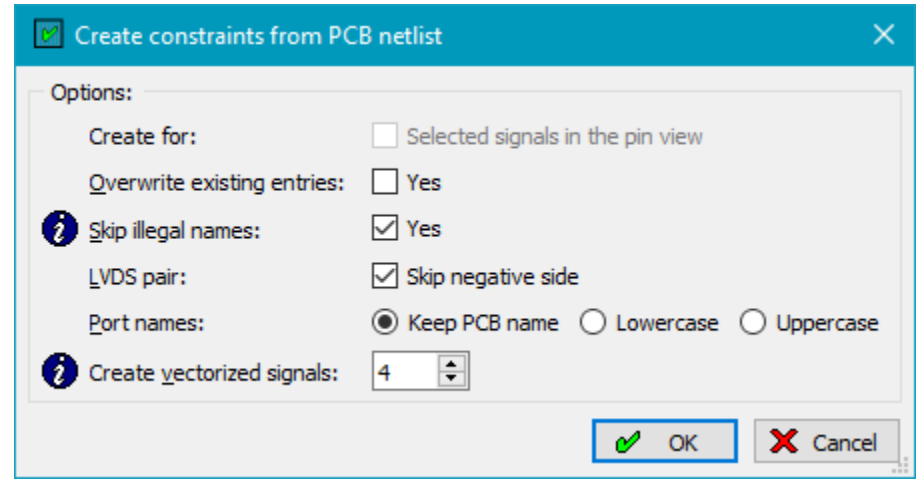


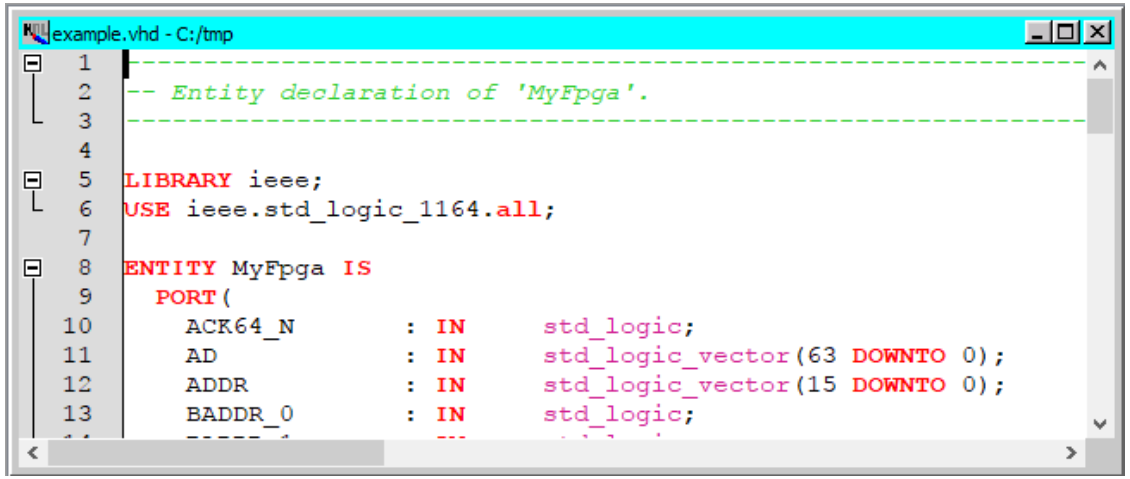
Example wiring



When started with schematic capture

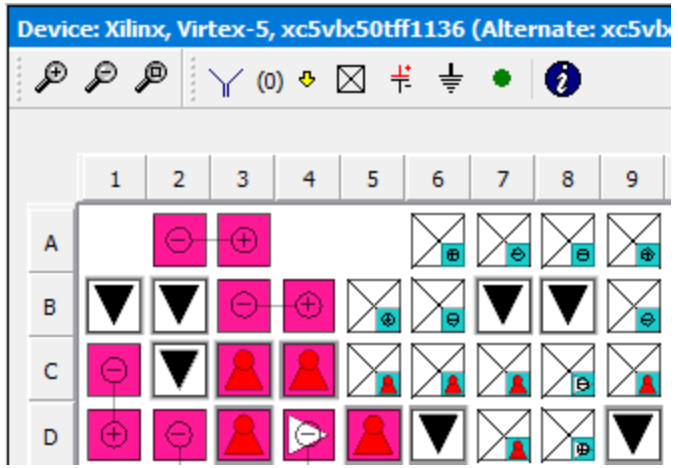
- Create FPGA constraints:
  - Using a dialog
  - By drag & drop
- Export data to CSV file
- Create toplevel VHDL or Verilog unit

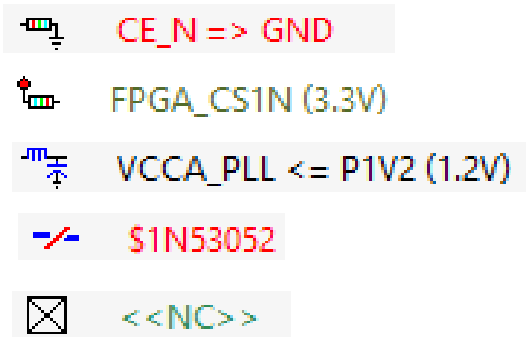




Other features

- CSV file import and export
- Device (alternate) view
- Rule generator
- Additional netlist information:
  - Pull-up- pull-down resistors
  - Decoupling circuits
  - Dangling, not connected
- TCL interface





FPGA support

- Altera (Intel)
  - Cyclone I, II, III, III LS, IV, IVE,V
  - Stratix I (GX), II (GX), III, IV, V
  - Arria GX, II GX, V, 10
  - Max II, V, 10
- Lattice
  - ECP2, ECP3, ECP5, Mach XO2, Mach XO3 and XP2
- Actel (Microsemi)
  - Axcelerator
  - Fusion, Smart Fusion, Smart Fusion 2
  - Igloo, Igloo+, IglooE, Igloo2
  - ProASIC3, E, L, RTG4
- Xilinx
  - Spartan 3 (all), Spartan 6, Spartan 7
  - Virtex 4, 5, 6
  - Artix 7, Kintex 7, Virtex 7, Zynq
  - Ultrascale (Kintex, Virtex)
  - Ultrascale+ (Kintex, Virtex, Zynq)

Future develop

- Generate application specific symbol(s)
- Optimize pin assignments based on physical board connectivity
- Multiple FPGA on a single board
- Tighten integration between Altium Designer and IO Checker

Conclusion

IO Checker:

- Fits in all flows
- Is easy to use
- Delivers results fast
- Allows quick re-verification after changes
- Saves you valuable time

Altium®

Thanks for your Attention!

Questions?

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