Advanced FPGA design

Andrea Borga
andrea.borga@nikhef.nl

Outline

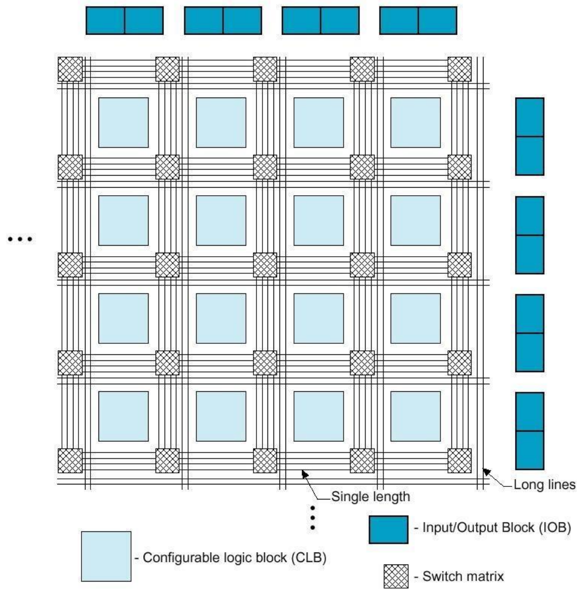
- First part: theory
  - … from the previous lesson
  - Considerations on Hardware Description
  - Gateware workflow
  - Takeaway thoughts
- Second part: practice
  - Eye diagrams
  - Pseudo Random Bit Sequences (PRBS)
  - FPGA serializers and deserializers

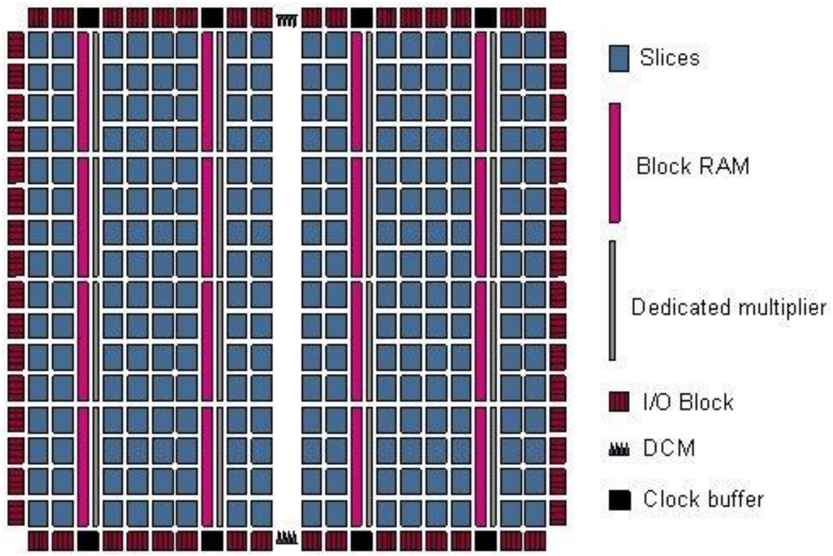
… from the previous lesson

FPGAs : Field Programmable Gate Arrays

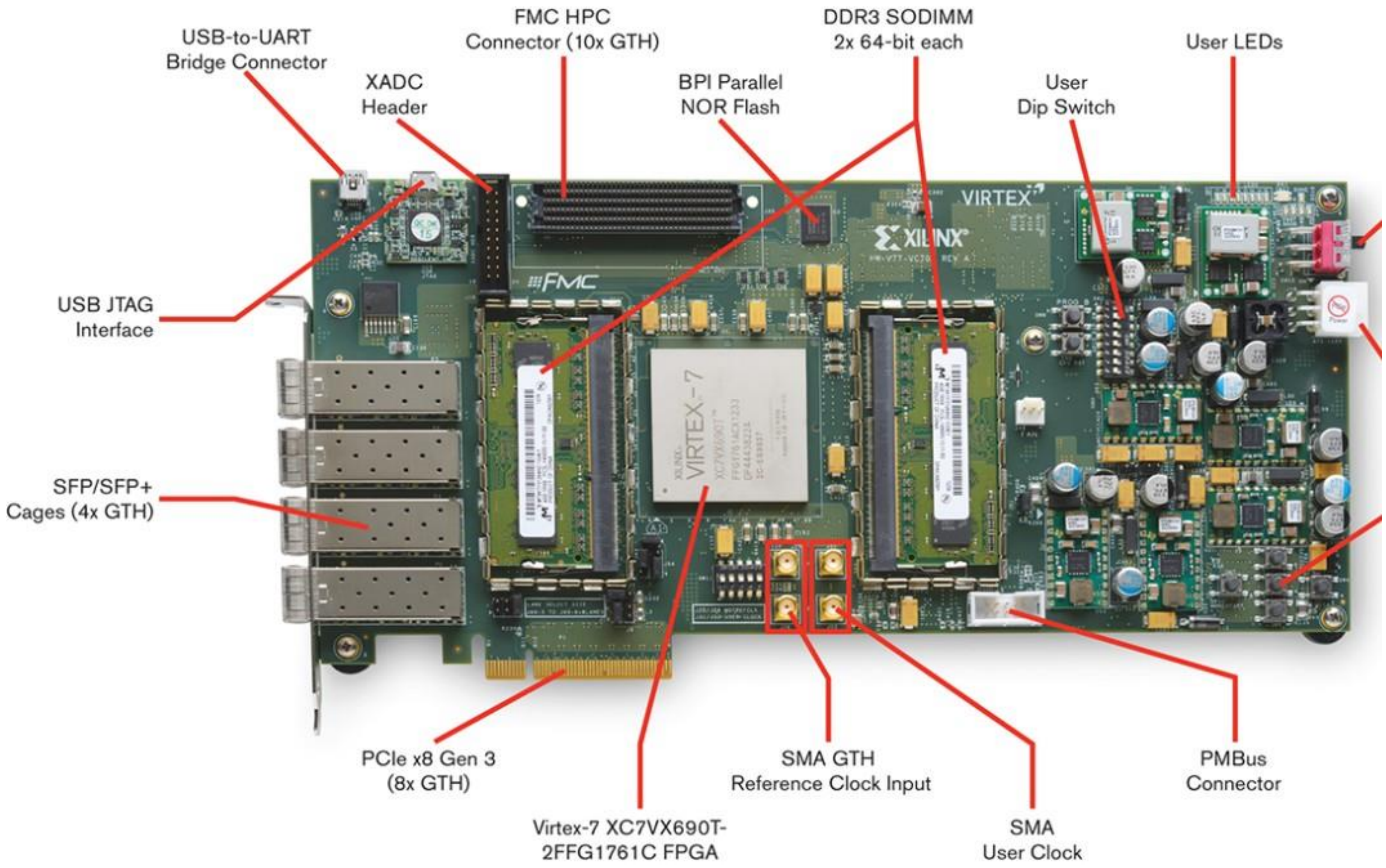
- Array (Matrix) like structure made of:
  - Look-Up-Table (LUT) to implement combinatorial logic
  - Flip-Flops (FF) to implement sequential logic
  - Routing network to interconnect the logic resources
  - I/O logic to communicate with outside logic
  - Clock Management: Phase Locked Loops (PLLs), Digital Clock Managers (DCMs)
  - Hard-Macros: Digital Signal Processing (DSP) cells, SRAMs, PCIe, Gigabit Transceivers, etc.

} Configurable Logic Block (CLB)



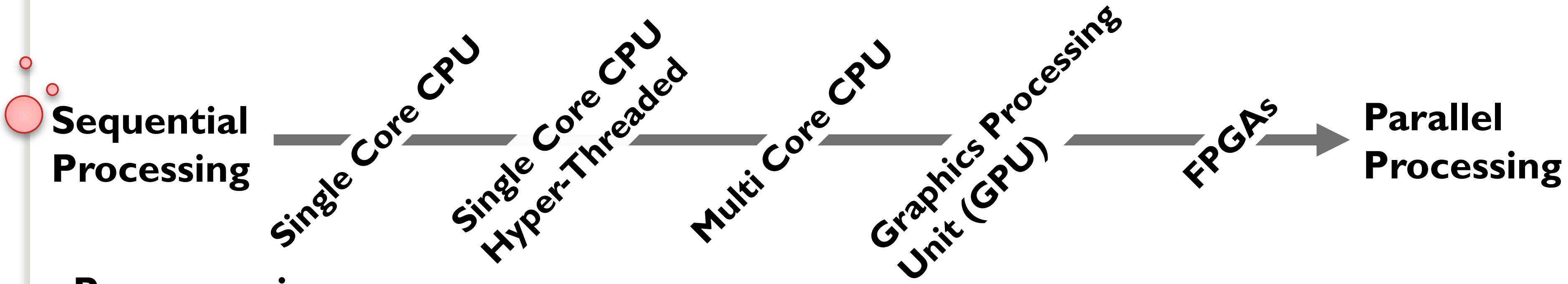


Example: Xilinx Virtex-7 development board



http://www.xilinx.com/products/boards-and-kits/dk-v7-vc709-g.html

Digital (Gateware) Design is NOT programming



Programming

- Code is written and translated into instructions
- Instructions are executed **sequentially** by the CPU(s)
- Parallelism is achieved by running instructions on multiple threads/cores
- Processing structures and instructions sets are **fixed by the architecture of the system**

VS.

Digital (Gateware) Design

- **No fixed** architecture, the system is built according to the task
- Building is done by **describing/defining** system elements and their relations
- **Intrinsically parallel**, sequential behaviour is achieved by Finite-State-Machines (FSMs) and registers
- Description done by **schematics** or a **hardware description language** (HDL)

Hardware **Description** Language (HDL)

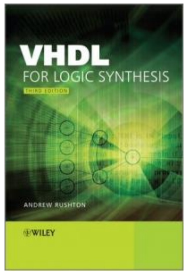
- **As the name suggests it is a language used to describe hardware:** so you have to use it to do so!
- Let's discuss the simple example of a wait statement
- In C (Unix, `#include <unistd.h>`)

```
sleep(5); // sleep 5 seconds
```

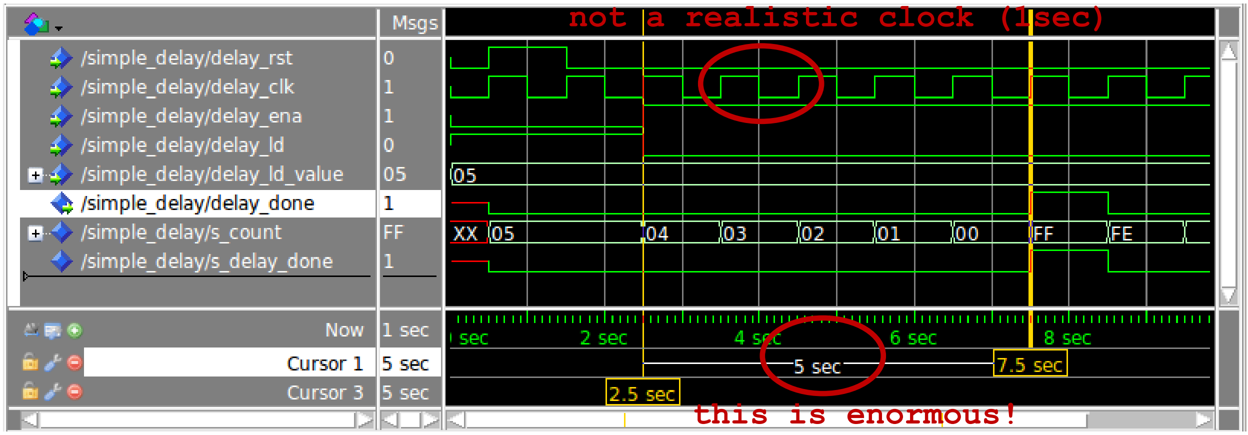
- In VHDL this is **not** synthesizable, but you can use it in test benches

```
wait for 5 sec;  -- handy for TB clocks
```

- This is (one) way to do it in synthesizable VHDL



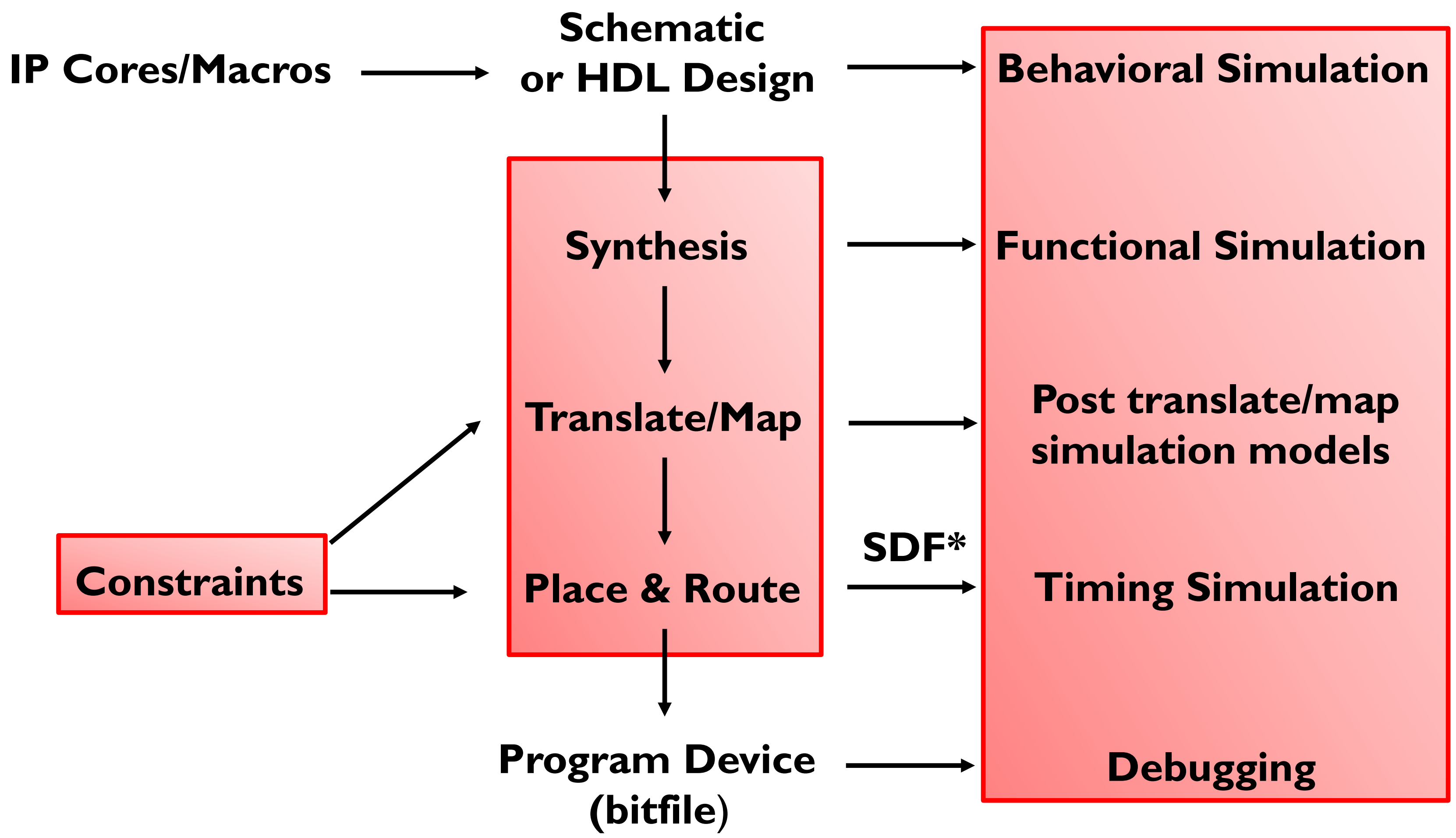
```
simple_delay_counter : process (delay_rst, delay_clk, delay_ena)
begin  -- process
  if delay_rst = '1' then
    s_count     <= delay_ld_value;
    s_delay_done <= '0';
  elsif rising_edge(delay_clk) then
    if delay_ena = '1' then
      if delay_ld = '1' then
        s_count <= delay_ld_value;
      else
        s_count <= s_count - 1;
      end if;
    end if;
    if s_count = 0 then
      s_delay_done <= '1';
    else
      s_delay_done <= '0';
    end if;
  end if;
end process;
```



	Msgs
/simple_delay/delay_rst	0
/simple_delay/delay_clk	1
/simple_delay/delay_ena	1
/simple_delay/delay_ld	0
/simple_delay/delay_ld_value	05
/simple_delay/delay_done	1
/simple_delay/s_count	FF
/simple_delay/s_delay_done	1
Now	1 sec
Cursor 1	5 sec
Cursor 3	5 sec

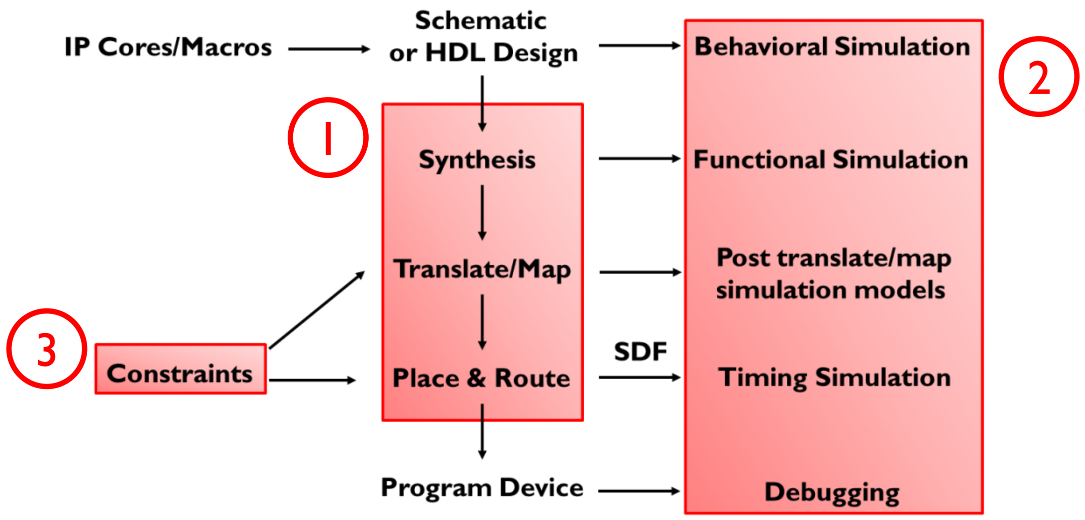
not a realistic clock (1sec)
05
XX 05 04 03 02 01 00 FF FE
sec 2 sec 4 sec 6 sec 8 sec
5 sec
7.5 sec
2.5 sec
this is enormous!

Gateware design workflow… a la carte!



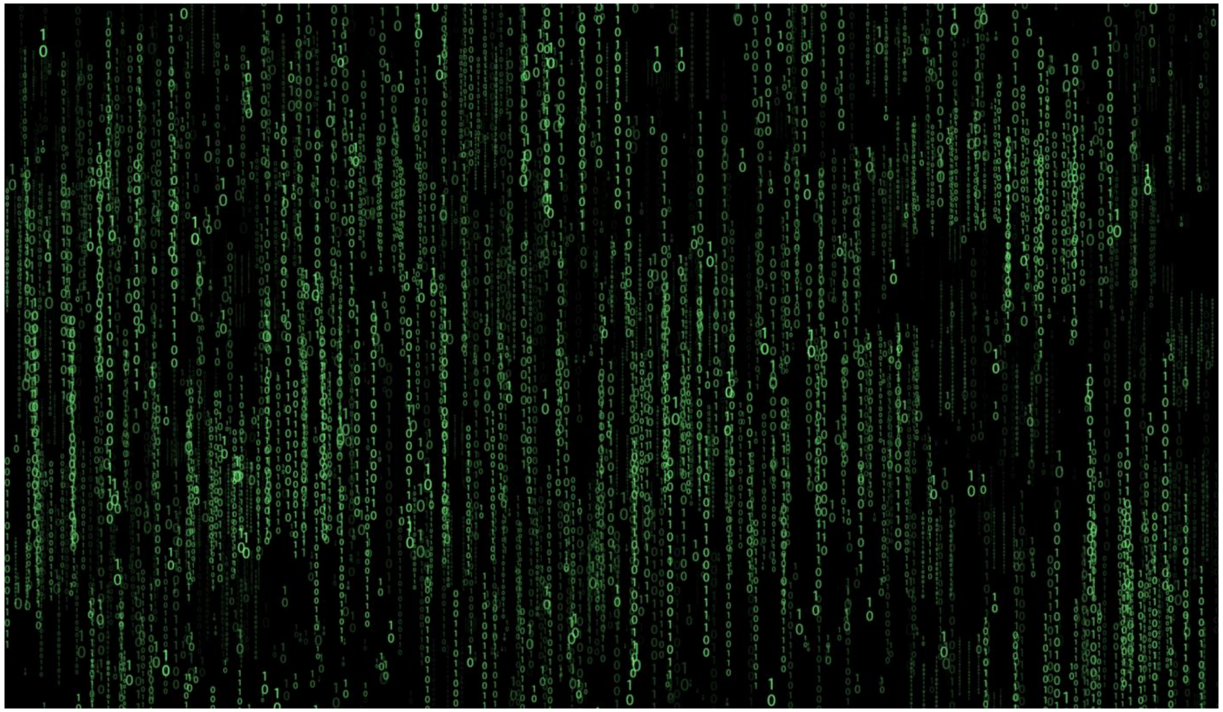
* Standard Delay Format

Gateware design workflow… a la carte!



1) **Implementation flow:**
what turns a line of code into a blinking LED?
2) **Verification flow:**
why is the statement above not (always) true!
3) **Design constraining:**
how to force your game rules

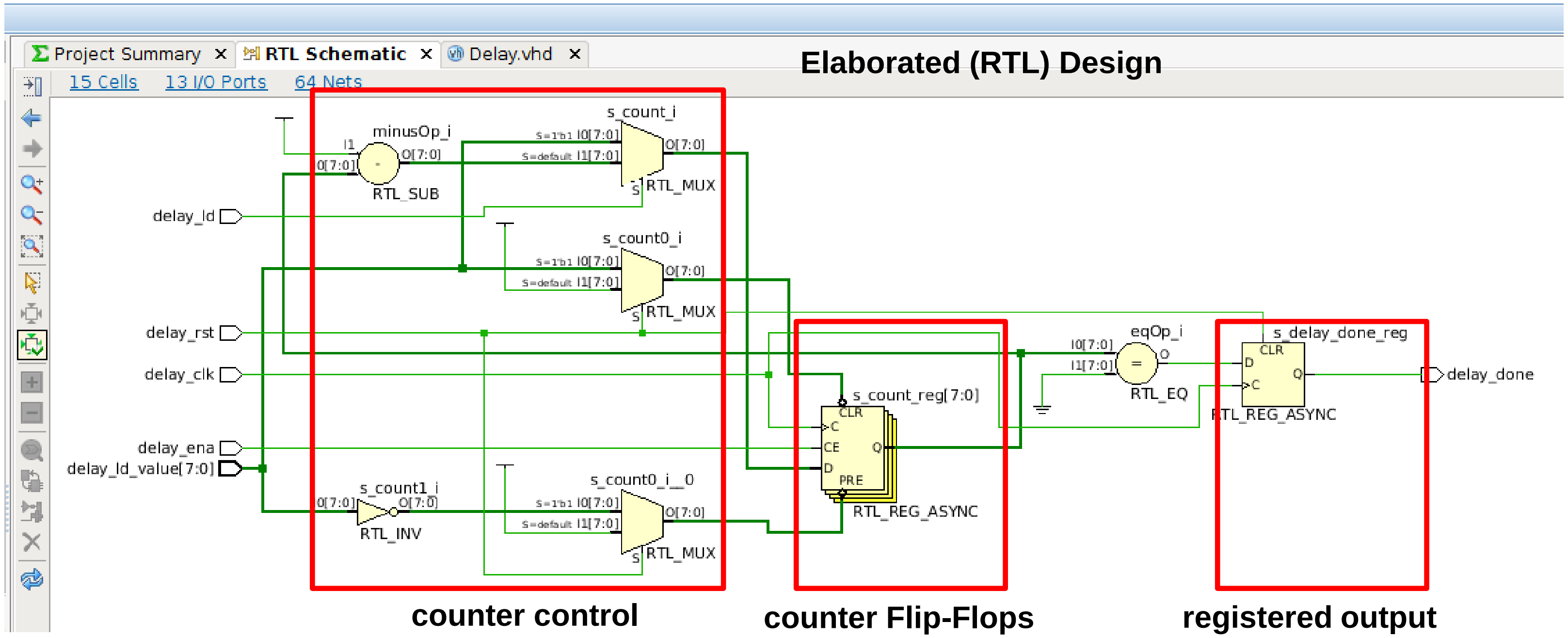
Implementation flow



Implementation flow: synthesis

Register Transfer Level (RTL)

- a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between registers and logical operations performed on those signals. (http://en.wikipedia.org/wiki/Register-transfer_level)



counter control

counter Flip-Flops

registered output

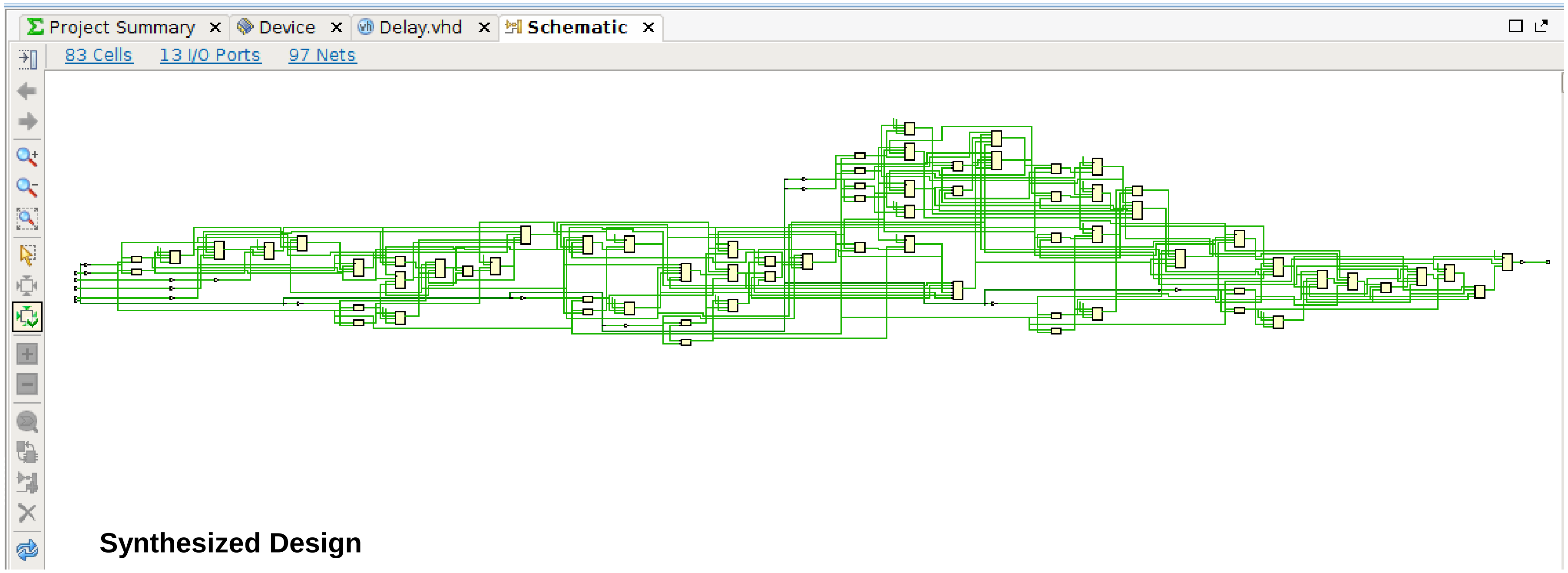
Implementation flow: synthesis

Synthesis

- translates the schematic or HDL code into elementary logic functions
- defines the connection of these elementary functions
- uses Boolean Algebra and Karnaugh maps to optimize logic functions
- generates a device independent **net list**

1815 - 1864

1924 -



Synthesized Design

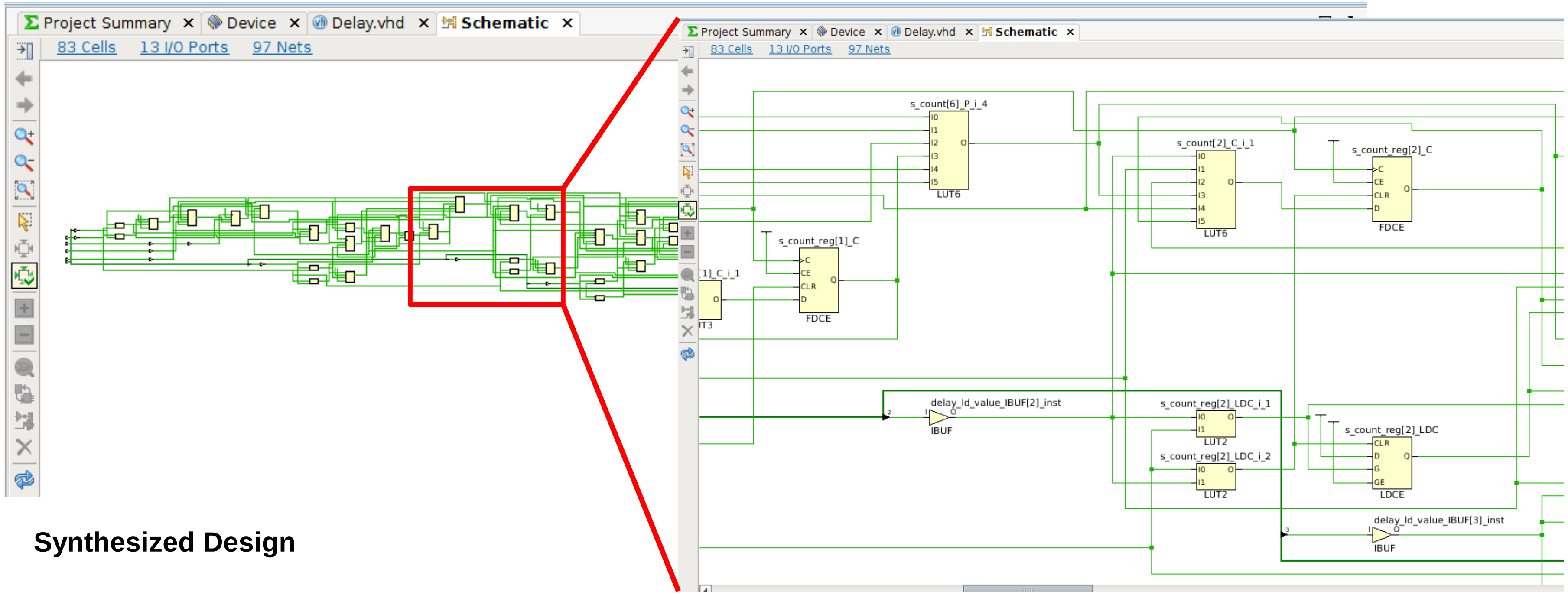
Implementation flow: synthesis

Synthesis

- translates the schematic or HDL code into elementary logic functions
- defines the connection of these elementary functions
- uses Boolean Algebra and Karnaugh maps to optimize logic functions
- generates a device independent **net list**

1815 - 1864

1924 -



Synthesized Design

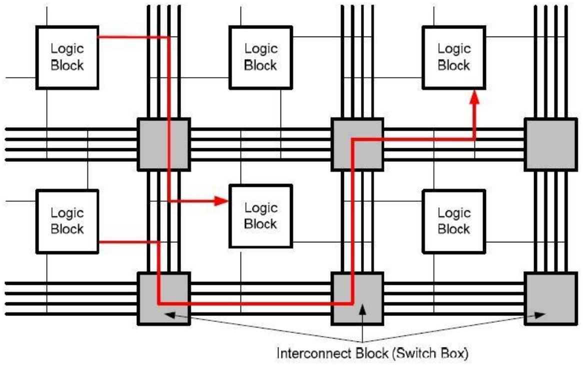
Implementation flow: mapping and routing

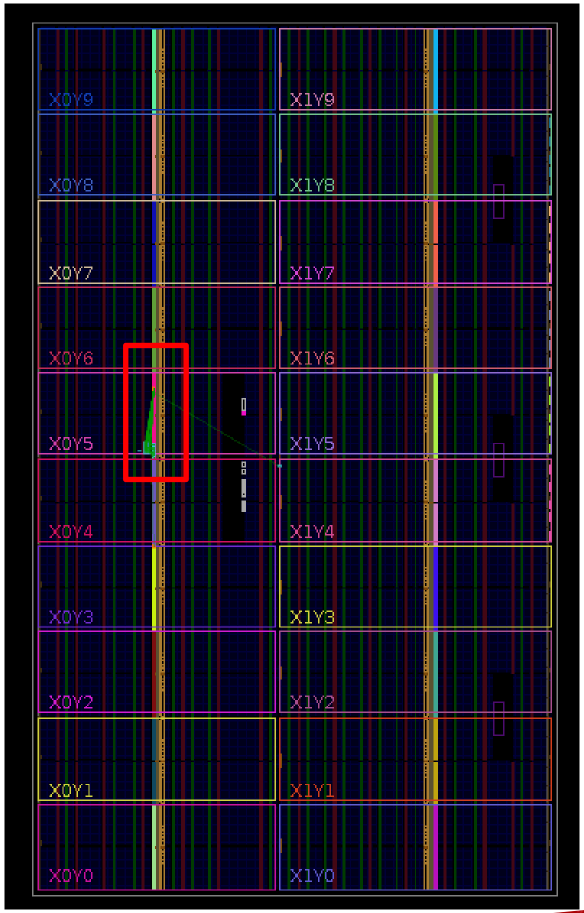
Translate / Mapping

- translates the device independent net list into technology specific elements
- checks the content of black boxes (e.g. IP cores)
- checks if the design can fit the target device
- maps these elements into the FPGA logic cells

Place and Route (P&R)

- places the basic elements on the logic cell grid
- routes the signals between the logic cells
- can be "guided" by constraints:
  - location constraints
  - timing constraints

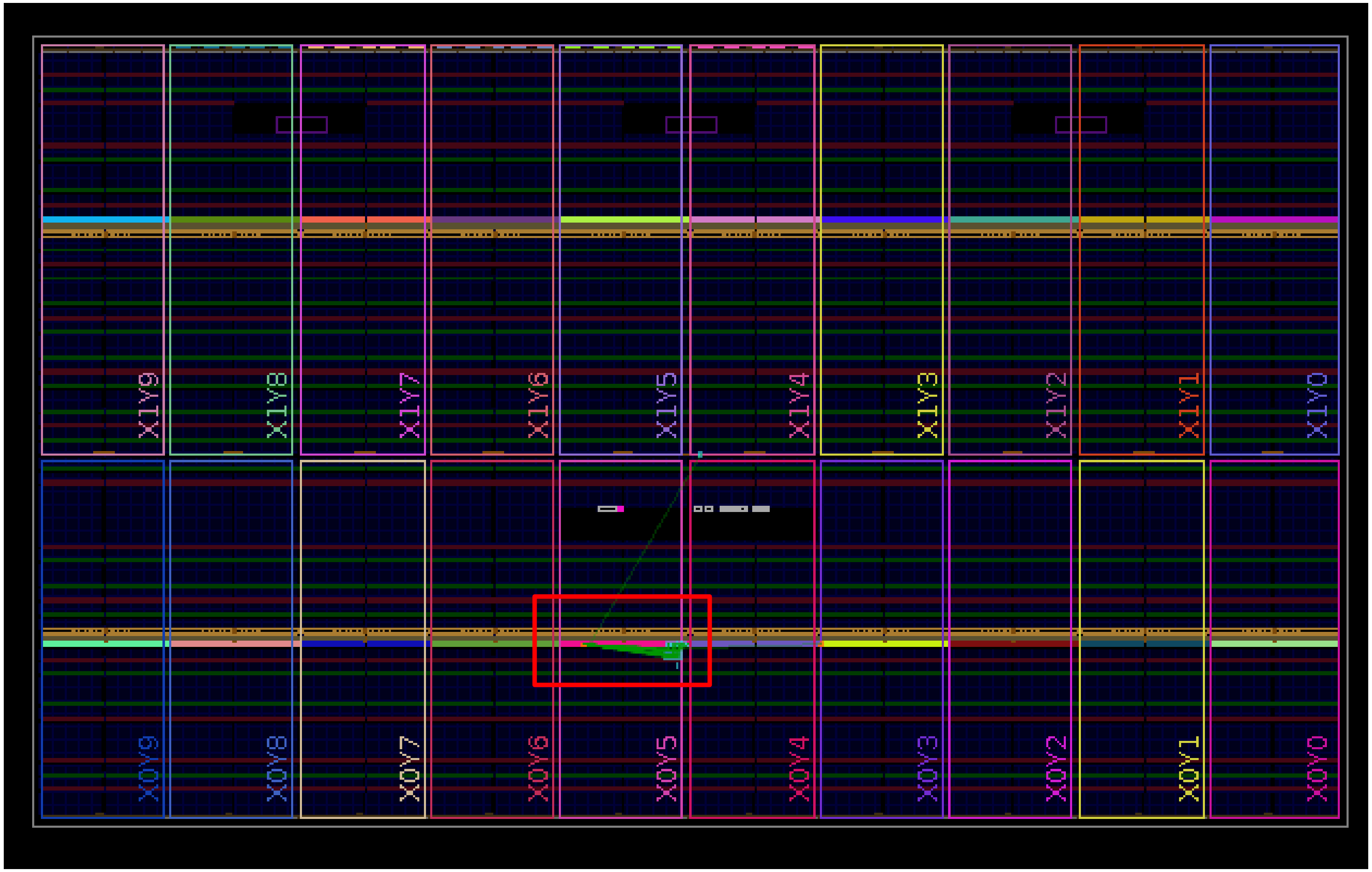




Virtex-7 690T floorplan

Xilinx Vivado 2014.4 design flow

Implementation flow: routing the counter



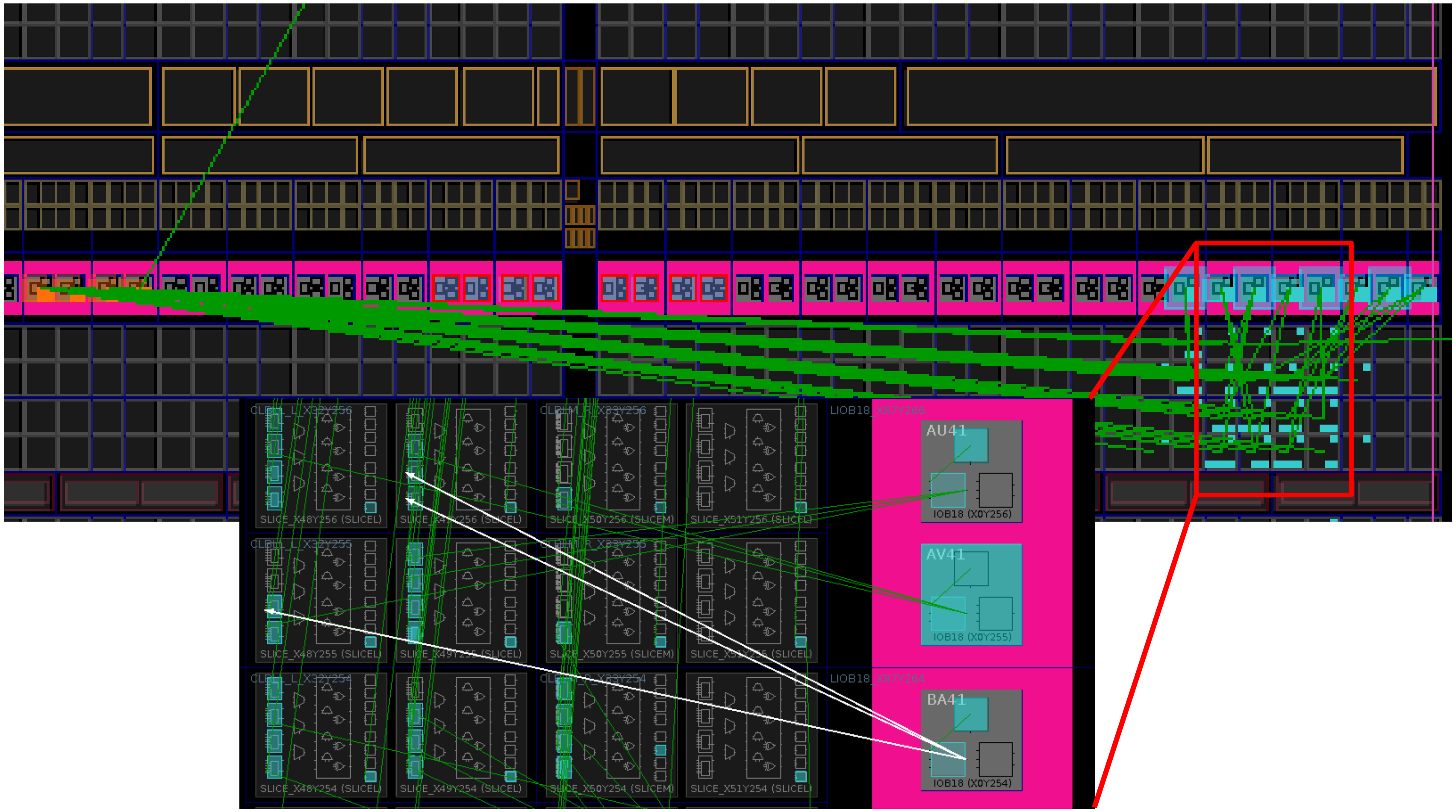
Virtex-7 690T floorplan

Xilinx Vivado 2014.4 design flow

Implementation flow: routing the counter

- Perfect example of a badly constrained design!☺

Virtex-7 690T floorplan

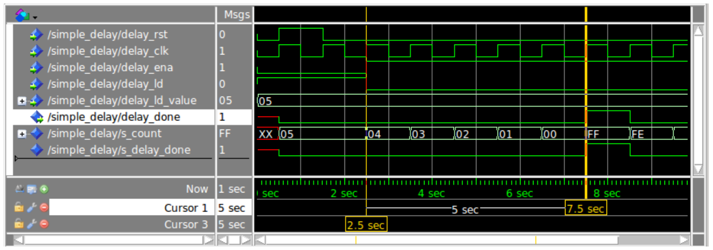


Xilinx Vivado 2014.4 design flow

Verification flow

Verification flow: simulation

- Verification of a design by an HDL simulator.
  - Industry standard → MentorGraphics Modelsim (or Questasim)
  - Try out some free alternatives → Icarus Verilog (http://iverilog.icarus.com/)
  - Try out some free alternatives → GHDL (http://ghdl.free.fr/)
- Event-based simulation to recreate the parallel nature of digital designs
  - The simulator time is sliced in delta delays
  - At each step of the delta delay all clauses (e.g. clock rising edge) are evaluated
  - The outcome of an event is computed and the logic updated
- Different levels of simulation:
  - **behavioral:** fastest, simulates only the behavior of the design


Msgs
/simple_delay/delay_rst 0
/simple_delay/delay_clk 1
/simple_delay/delay_ena 1
/simple_delay/delay_ld 0
/simple_delay/delay_ld_value 05
/simple_delay/delay_done 1
/simple_delay/s_count FF
/simple_delay/s_delay_done 1
05
XX 05 04 03 02 01 00 FF FE
Now 1 sec
Cursor 1 5 sec
Cursor 3 5 sec
sec 2 sec 4 sec 6 sec 8 sec
5 sec
7.5 sec
2.5 sec

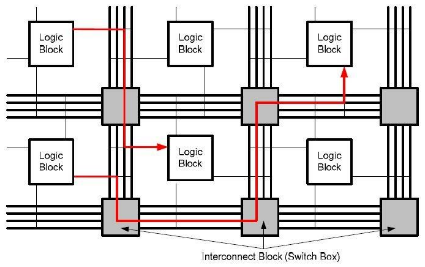
Example: MentorGraphics Questasim

Verification flow: simulation

- **functional:** fast, uses realistic functional models for the target technology the least used by HDL designers … why?
  - Mostly because these days you can (almost) trust your tools (a bit) more
  - What happens if you use the VHDL statement?

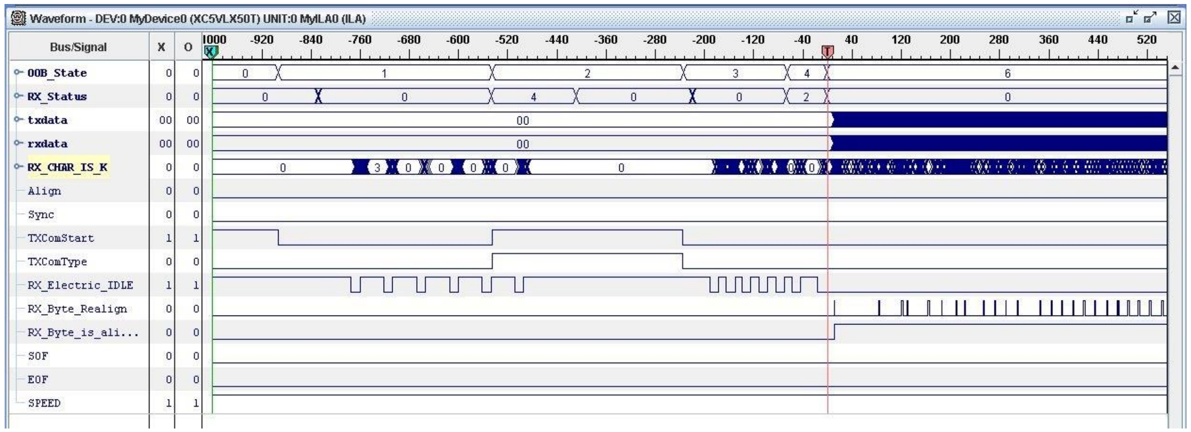
```
signal <= 'X'; -- unknown (misused to connect to anything)
```

- **post translate and map simulation models:** similar to the above but with information (about the actual primitives) of the translation and mapper steps
- **timing:** slow, most accurate. Uses Place & Route design + SDF (Standard Delay Format)
  - in the past was used to detect routers errors in placing designs… when routers where not so smart and FPGAs where not so fast!
  - what if the propagation delays of the bits of our counter where not equal?
  - or greater than the clock speed?

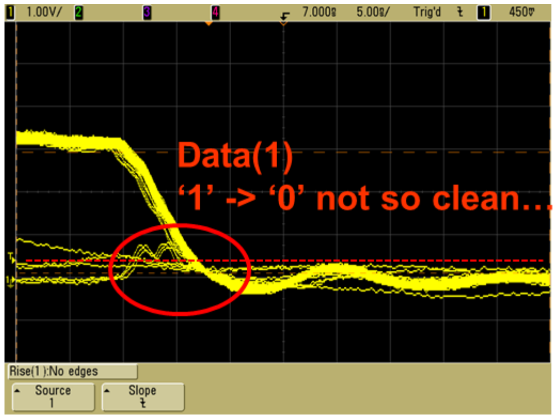


Verification flow: debugging

- Your desing is up… and also running?
- Most FPGA vendors provide internal logic analyzer cores
  - ISE ChipScope, Vivado Set up Debug (Xilinx)
  - SignalTap (Altera)
- Can be embedded into the design and controlled by JTAG
- Allow also the injection of signals
- It is at times extremely useful to spy inside the FPGA… but this doesn't replace an oscilloscope… as signal integrity issues can be on the PCB
- Remember… it's hardware!



Example: ChipScope waveform window (Xilinx ISE)



Design constraining

Design constraining

- Remember: you are describing your hardware!

- Constraining is becoming so important that it is turning into a (not yet) standardized language of it own:
  - .qsf: Quartus II Setting File (Altera)
  - .sdc: Synopsis Design Constraints (de facto standard)
  - .ucf: User Constraint File → .xdc Xilinx Constraint File (Xilinx)

- Two types of constraints:

Location constraints

- Geographical position and pin related

Timing constraints

- clock and timing related

Design constraining: location

- FPGAs usually provide a large number of I/O pins for communication with the outside world
- Large variety of I/O standards supported: 3.3V CMOS, 2.5V LVDS, SSTL, …
- I/O pins can be assigned more or less freely

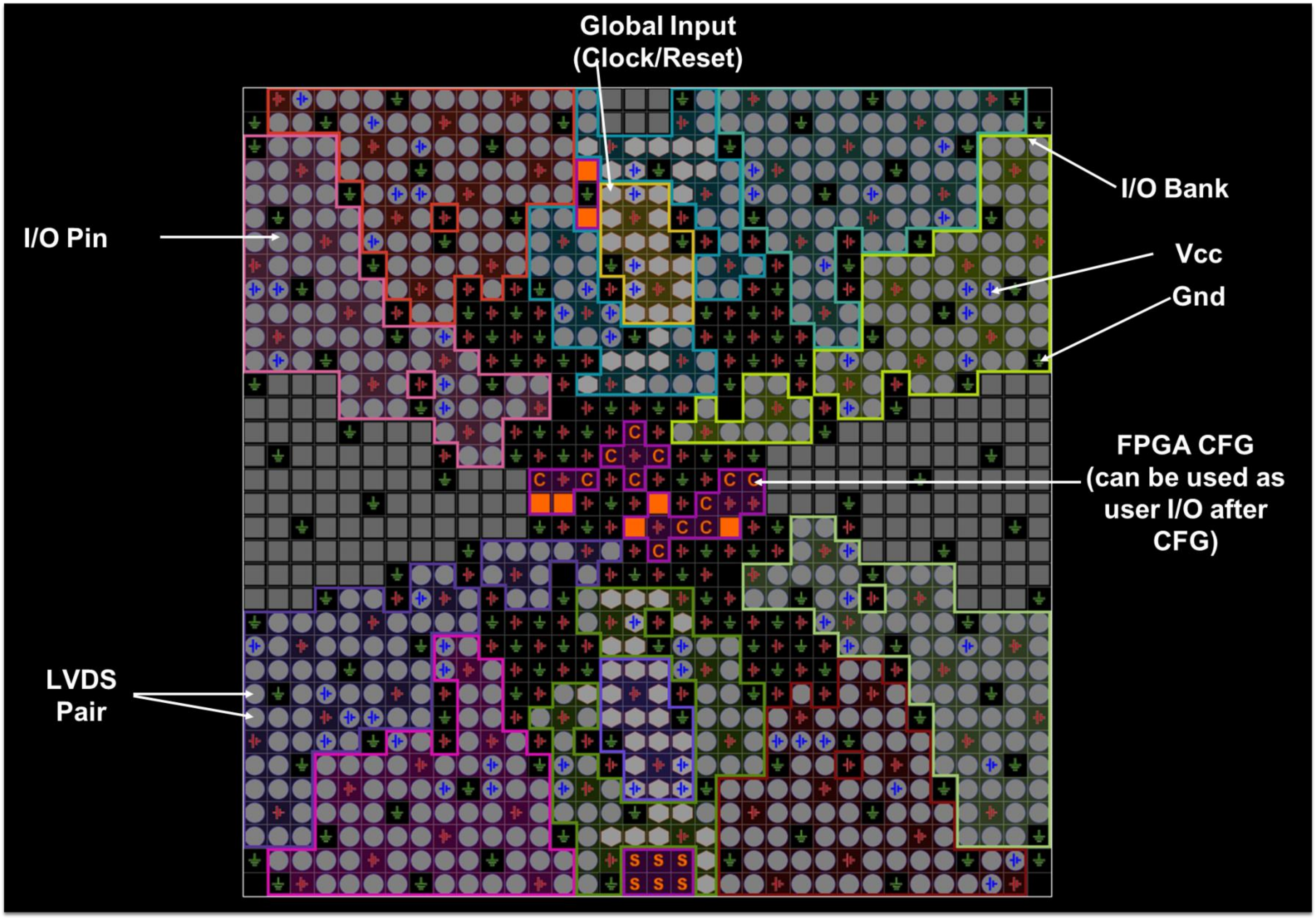
BUT

- I/O cells are grouped in **I/O banks** → All cells in an I/O bank need to use either the same standard or a similar one (with the same voltage level), e.g. 3.3V CMOS is not compatible with LVDS
- LVDS signals always come in **dedicated pairs**
- **Clock** signals should use dedicated clock input pins → routed internally over a dedicated network
- High-Speed serial interfaces (PCIe, Gigabit-Transceivers) or hard macros might need dedicated pins as well

Good Practice

- Try to locate pins belonging to one design module close to each other → **avoid routing** across chip
- PCB Designers:
  - Check your I/O assignment with a **preliminary** design with only I/O pins instantiated
  - Check for **SSN** (Simultaneous Switching Noise)
  - Use **back-annotation** of I/O pins to optimise fan-out and routing of signals

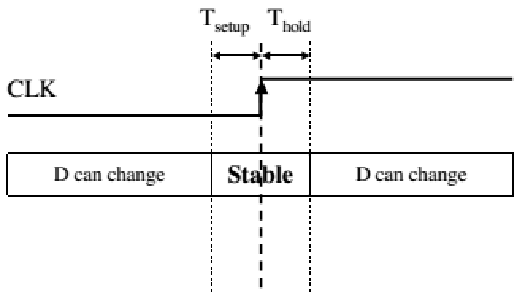
Design constraining: location


Global Input (Clock/Reset)
I/O Bank
I/O Pin
Vcc
Gnd
FPGA CFG (can be used as user I/O after CFG)
LVDS Pair
C
S

Example: Virtex-4 LX40-FF1148 Package & Pinout View

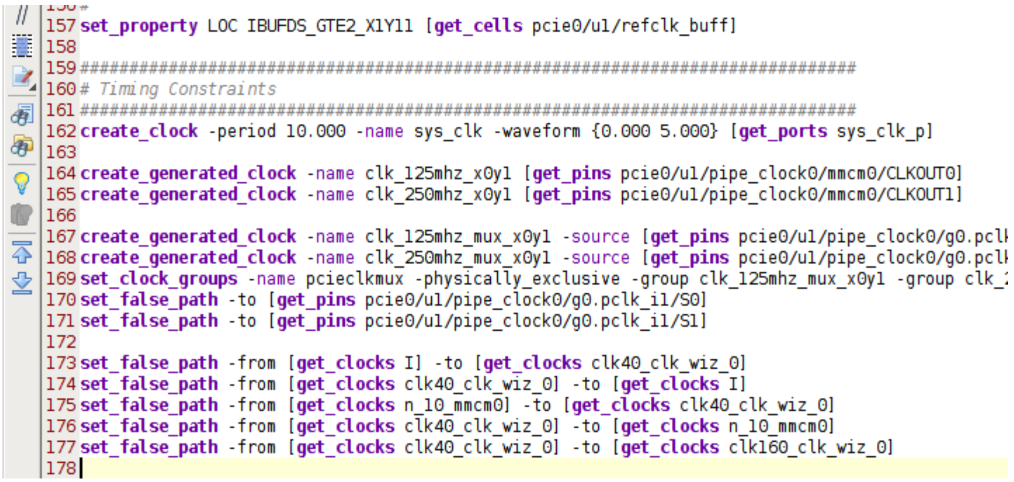
Design constraining: timing

- <u>**Timing constraints**</u>
  - clock period
  - setup and hold times
  - path delays: highlight critical connections
  - false paths: force ignoring some connections



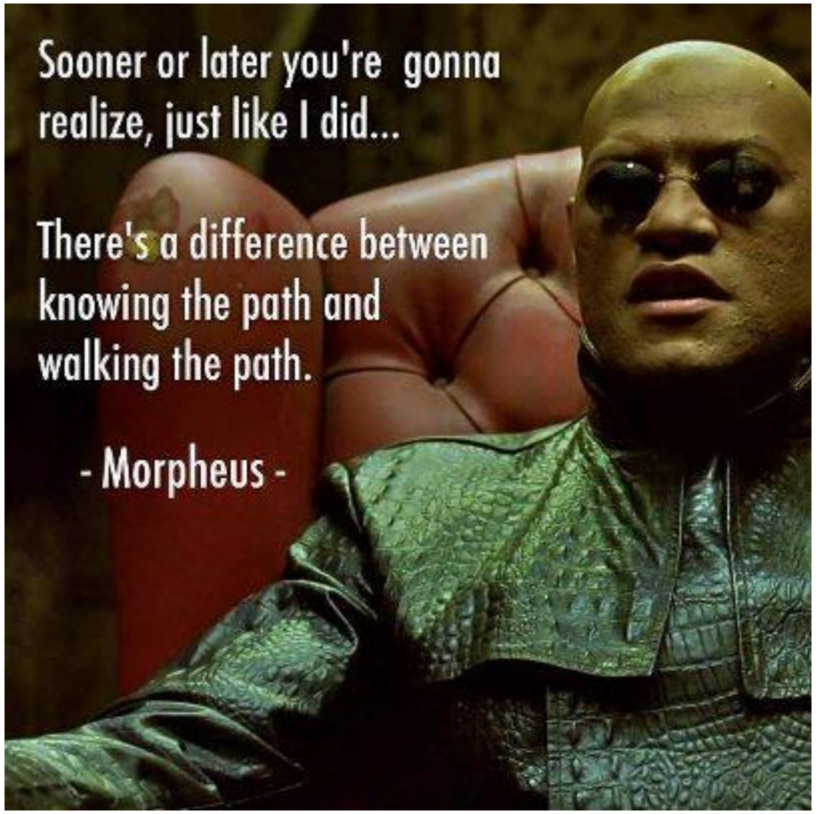
- See for example: Xilinx Vivado Using Constraints (UG903)

- Remember the story of the propagation delay in the timing simulation?
(the positioning of Flip-Flops?)
- If you do a good job constraining… you can spare yourself the timing simulation!



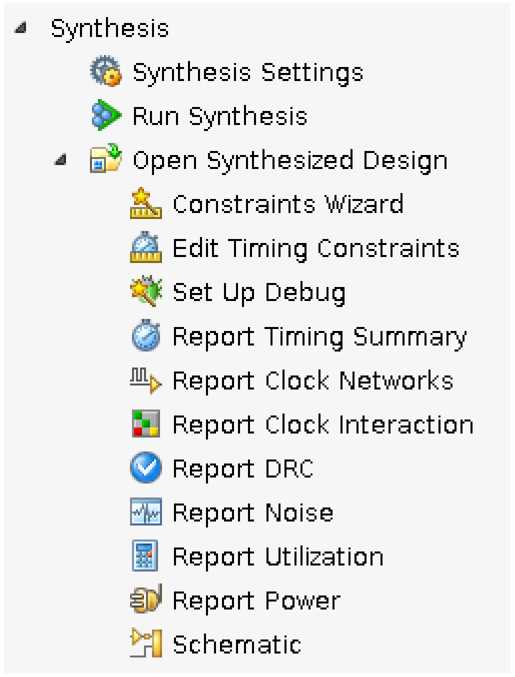
```
set_property LOC IBUFDS_GTE2_X1Y11 [get_cells pcie0/u1/refclk_buff]

##############################################################################
# Timing Constraints
##############################################################################
create_clock -period 10.000 -name sys_clk -waveform {0.000 5.000} [get_ports sys_clk_p]

create_generated_clock -name clk_125mhz_x0y1 [get_pins pcie0/u1/pipe_clock0/mmcm0/CLKOUT0]
create_generated_clock -name clk_250mhz_x0y1 [get_pins pcie0/u1/pipe_clock0/mmcm0/CLKOUT1]

create_generated_clock -name clk_125mhz_mux_x0y1 -source [get_pins pcie0/u1/pipe_clock0/g0.pcl
create_generated_clock -name clk_250mhz_mux_x0y1 -source [get_pins pcie0/u1/pipe_clock0/g0.pcl
set_clock_groups -name pcieclkmux -physically_exclusive -group clk_125mhz_mux_x0y1 -group clk_
set_false_path -to [get_pins pcie0/u1/pipe_clock0/g0.pclk_i1/S0]
set_false_path -to [get_pins pcie0/u1/pipe_clock0/g0.pclk_i1/S1]

set_false_path -from [get_clocks I] -to [get_clocks clk40_clk_wiz_0]
set_false_path -from [get_clocks clk40_clk_wiz_0] -to [get_clocks I]
set_false_path -from [get_clocks n_10_mmcm0] -to [get_clocks clk40_clk_wiz_0]
set_false_path -from [get_clocks clk40_clk_wiz_0] -to [get_clocks n_10_mmcm0]
set_false_path -from [get_clocks clk40_clk_wiz_0] -to [get_clocks clk160_clk_wiz_0]
```

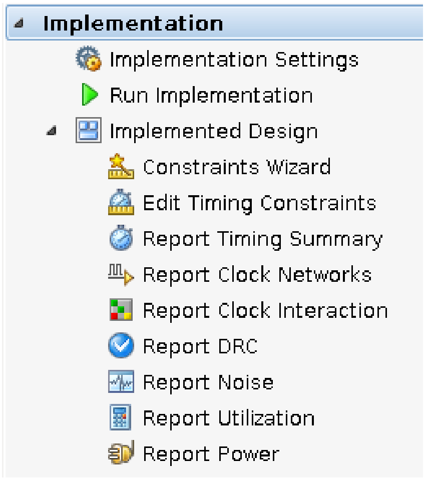
Takeaway thoughts

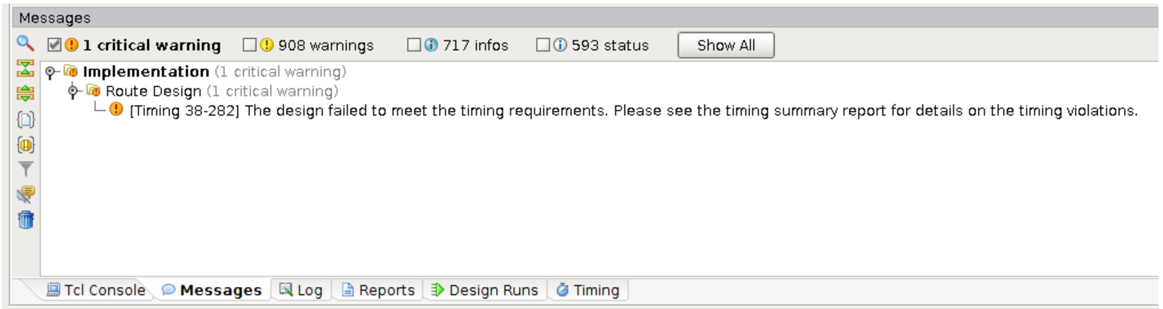

Sooner or later you're gonna realize, just like I did...
There's a difference between knowing the path and walking the path.
- Morpheus -

Takeaway: don't ignore reports!

- Learn to carefully review reports
- The reason why your design is not functioning as intended…
  can be right in front of your eyes!
- Especially check timing and… **don't** run designs that haven't met timing!





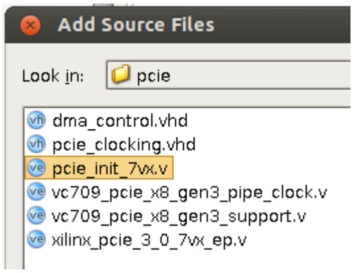


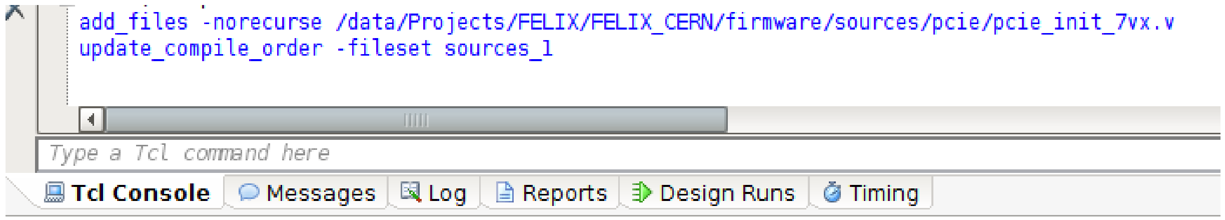
```
Implementation (1 critical warning)
  Route Design (1 critical warning)
    [Timing 38-282] The design failed to meet the timing requirements. Please see the timing summary report for details on the timing violations.
```

Xilinx Vivado 2014.4 design flow

Takeaway: scripting for Gateware designs

- Design tools can be scripted: **Tool Command Language (TCL)**
- Parameters/Options can be passed via command-line (makefile, shell scripts)
- You have **much more control and reproducibility** on your procedures (you can forget about checking a tick-box, and you will, sooner or later…)
- allows for complete automation → design servers and nightly build

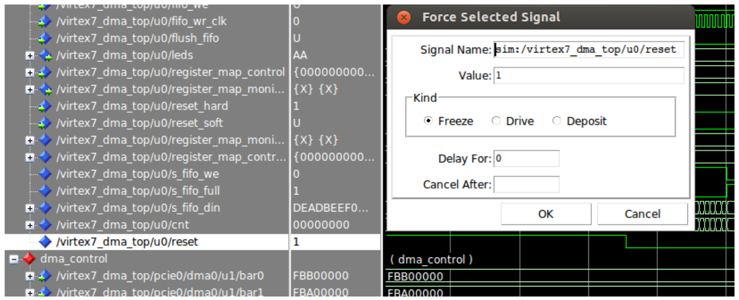




```
add_files -norecurse /data/Projects/FELIX/FELIX_CERN/firmware/sources/pcie/pcie_init_7vx.v
update_compile_order -fileset sources_1
```

Xilinx Vivado 2014.4

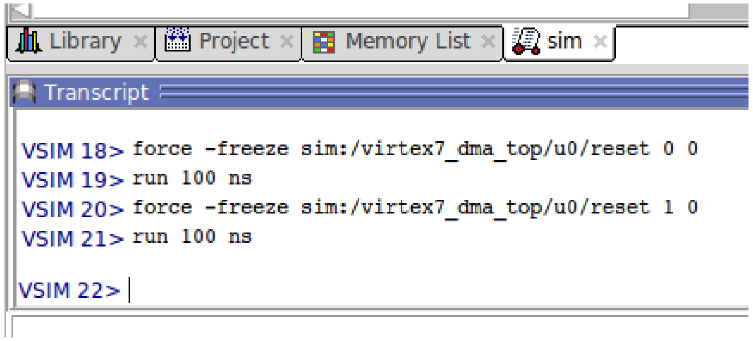
- Simulators can be controlled with TCL and even used to create test benches (slower but extremely flexible)



/virtex7_dma_top/u0/fifo_wr_clk	0
/virtex7_dma_top/u0/flush_fifo	U
/virtex7_dma_top/u0/leds	AA
/virtex7_dma_top/u0/register_map_control	{000000000...
/virtex7_dma_top/u0/register_map_moni...	{X} {X}
/virtex7_dma_top/u0/reset_hard	1
/virtex7_dma_top/u0/reset_soft	U
/virtex7_dma_top/u0/register_map_moni...	{X} {X}
/virtex7_dma_top/u0/register_map_contr...	{000000000...
/virtex7_dma_top/u0/s_fifo_we	0
/virtex7_dma_top/u0/s_fifo_full	1
/virtex7_dma_top/u0/s_fifo_din	DEADBEEF0...
/virtex7_dma_top/u0/cnt	00000000
/virtex7_dma_top/u0/reset	1
dma_control	
/virtex7_dma_top/pcie0/dma0/u1/bar0	FBB00000
/virtex7_dma_top/pcie0/dma0/u1/bar1	FBA00000

Force Selected Signal

Signal Name: sim:/virtex7_dma_top/u0/reset
Value: 1
Kind: [✓ Freeze] Drive Deposit
Delay For: 0
Cancel After:

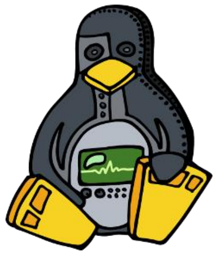


```
VSIM 18> force -freeze sim:/virtex7_dma_top/u0/reset 0 0
VSIM 19> run 100 ns
VSIM 20> force -freeze sim:/virtex7_dma_top/u0/reset 1 0
VSIM 21> run 100 ns

VSIM 22>
```

MentorGraphics Questasim v10.2

Takeaway: more tips

- Describe your hardware: **think hard**…ware!
- RT…M! Seriously… you HAVE to, especially with FPGAs (family overview, DC and Switching, clock resource, Transceiver Guides, Package and pinout)
- Consider your FPGA full at 70% or you'll get nice surprises from your router…
- Digital designs are analog in essence (especially with ever higher clock frequencies)
- Share share share…
- Celebrate your achievements!

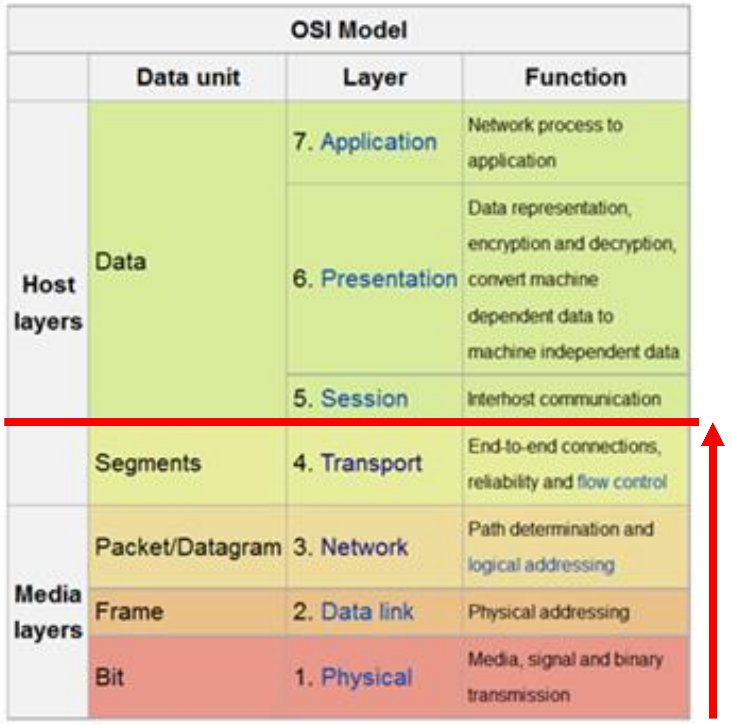


FPGAs… so what?
Practical example

ISO / OSI model: You are here…

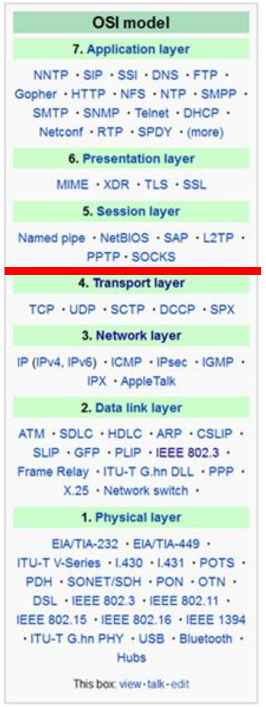
- International Organization for Standardization / Open System Interconnection: if you are talking about engineering, can't do a talk without!☺
- It is a conceptual model that characterizes and standardizes the internal functions of a communication system by partitioning it into abstraction layers
- A layer serves the layer above it and is served by the layer below it

by function



OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections, reliability and flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

by protocol



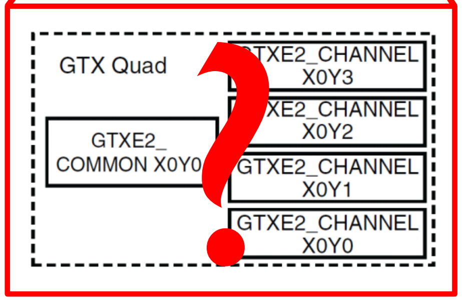
OSI model
7. Application layer
NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP · NFS · NTP · SMPP · SMTP · SNMP · Telnet · DHCP · Netconf · RTP · SPDY · (more)
6. Presentation layer
MIME · XDR · TLS · SSL
5. Session layer
Named pipe · NetBIOS · SAP · L2TP · PPTP · SOCKS
4. Transport layer
TCP · UDP · SCTP · DCCP · SPX
3. Network layer
IP (IPv4, IPv6) · ICMP · IPsec · IGMP · IPX · AppleTalk
2. Data link layer
ATM · SDLC · HDLC · ARP · CSLIP · SLIP · GFP · PLIP · IEEE 802.3 · Frame Relay · ITU-T G.hn DLL · PPP · X.25 · Network switch ·
1. Physical layer
EIA/TIA-232 · EIA/TIA-449 · ITU-T V-Series · I.430 · I.431 · POTS · PDH · SONET/SDH · PON · OTN · DSL · IEEE 802.3 · IEEE 802.11 · IEEE 802.15 · IEEE 802.16 · IEEE 1394 · ITU-T G.hn PHY · USB · Bluetooth · Hubs
This box: view · talk · edit

http://en.wikipedia.org/wiki/OSI_model

System Architecture: You are here…

	Segment	4. Transport
Media Layers	Packet	3. Network
	Frame	2. Data Link
	Bit	1. Physical

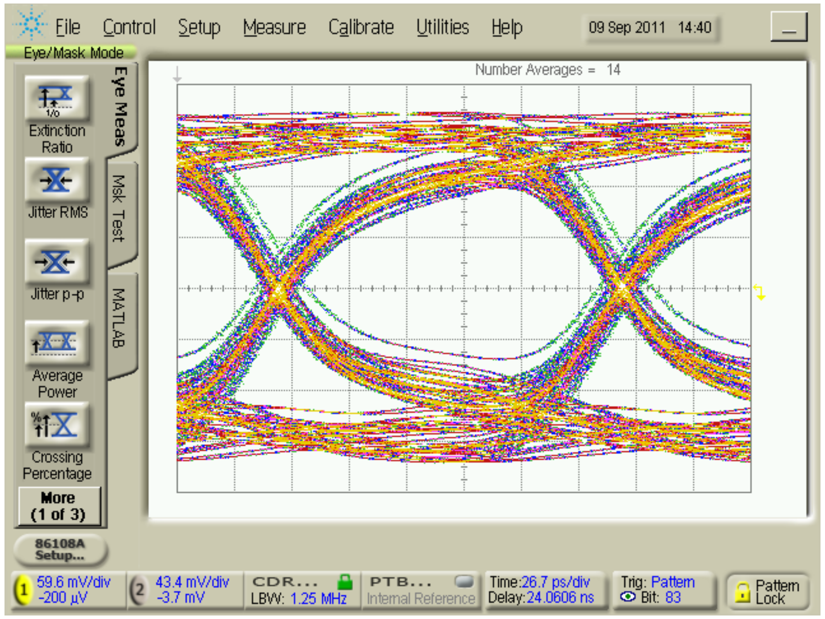
- A Gigabit-Transceiver is one of the many gadgets surrounding gate logic widely available in modern FPGAs
- Very popular since a lot of applications have demanding and fast (serial) I/O requirements



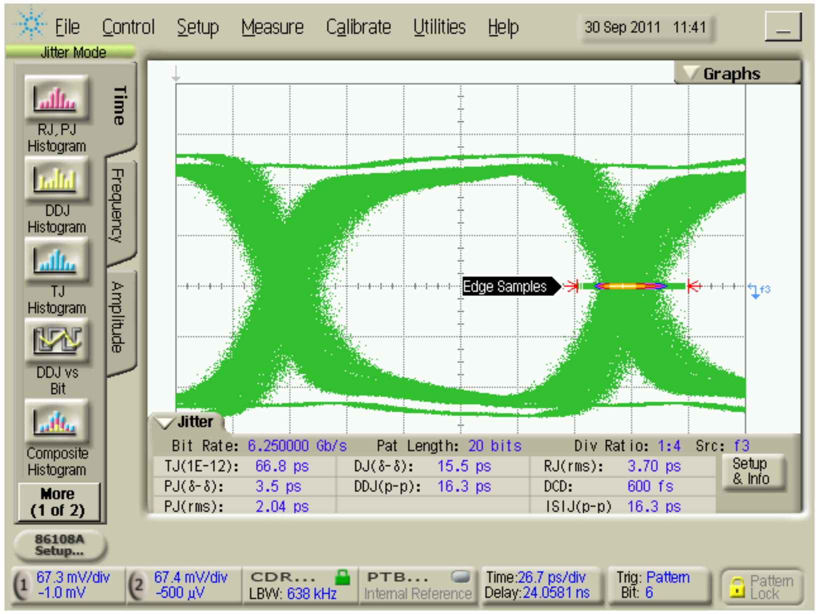
Gigabit-Transceiver X

Eye Diagrams

- An eye diagram (eye pattern) is the first measure of the quality of a transmission channel: how good are my “ones” and “zeroes”?



```
09 Sep 2011   14:40
Number Averages = 14
1  59.6 mV/div  -200 µV
2  43.4 mV/div  -3.7 mV
CDR...  LBW: 1.25 MHz
PTB...  Internal Reference
Time:26.7 ps/div  Delay:24.0606 ns
Trig: Pattern  Bit: 83
```



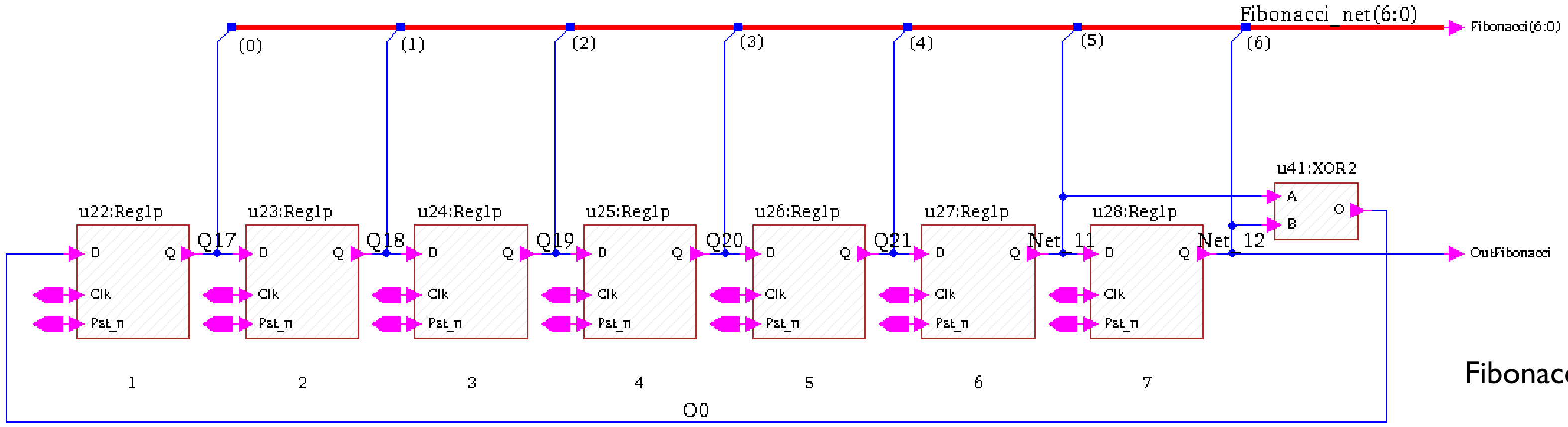
```
30 Sep 2011   11:41
Bit Rate: 6.250000 Gb/s    Pat Length: 20 bits    Div Ratio: 1:4   Src: f3
TJ(1E-12):  66.8 ps    DJ(δ-δ):   15.5 ps    RJ(rms):   3.70 ps
PJ(δ-δ):    3.5 ps     DDJ(p-p):  16.3 ps    DCD:       600 fs
PJ(rms):    2.04 ps                          ISIJ(p-p)  16.3 ps
1  67.3 mV/div  -1.0 mV
2  67.4 mV/div  -500 µV
CDR...  LBW: 638 kHz
PTB...  Internal Reference
Time:26.7 ps/div  Delay:24.0581 ns
Trig: Pattern  Bit: 6
```

- Essential information on transmission quality can be obtained from these diagrams : amplitude (voltage) stability, time stability,, etc.
- It is all about the probability to sample the signal correctly

Instrument: Agilent 86100C DCA-J

Pseudo Random Bit Sequence (PRBS)

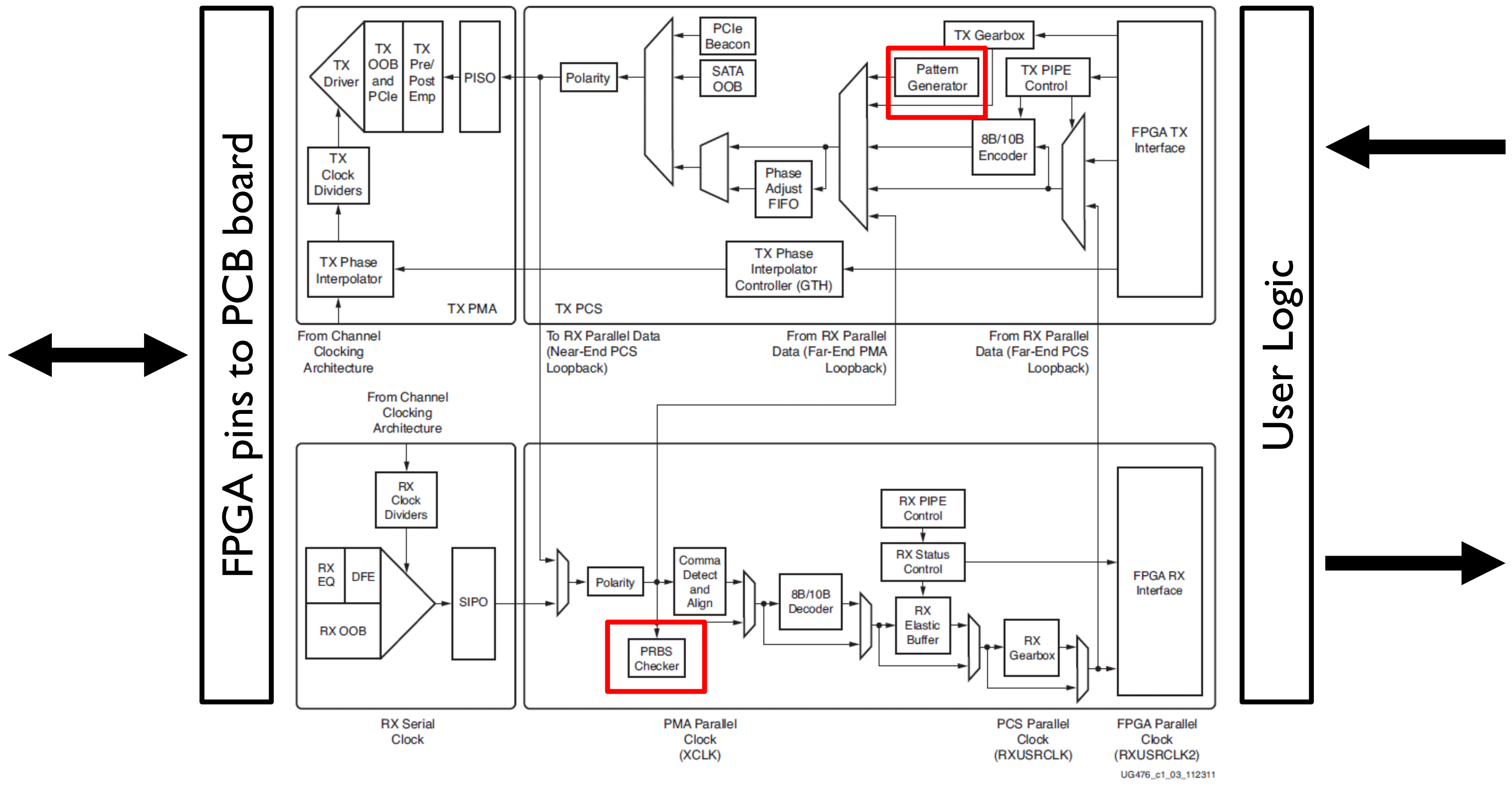
- A PRBS is a sequence of bits that are pseudo-random. That is, they are not really random but they can be used where a good approximation to random values is required → test vectors, white noise
- They are often implemented using Linear Feedback Shift Registers (LFSR)
- The arrangement of taps for feedback in an LFSR is a polynomial mod 2
  - PRBS = $x^7 + x^6 + 1$
- Maximum number of sequences: 2^n-1
- It starts from a "seed value", the only forbidden state is all-zeroes (no exit)



Fibonacci LFSR

http://en.wikipedia.org/wiki/Linear_feedback_shift_register

Xilinx Virtex-7 Serializers Deserializers



- Good news: the PRBS is a built-in function of most modern transceivers!
- The next step is to write your own code and drive a link!

www.xilinx.com/support/.../user.../ug476_7Series_Transceivers.pdf

Pitbullen!

- We may face (very) difficult problems…

- **Never let go!**
- It may take a while… but victory will be yours!
- Thank you very very much to:
  Torsten Alt (FIAS) and Peter Jansweijer (Nikhef)