Design flow, tools for development and debug with FPGAs

Davide Falchieri

Data driven front-end electronics for highly segmented radiation detectors
25-27 November 2013

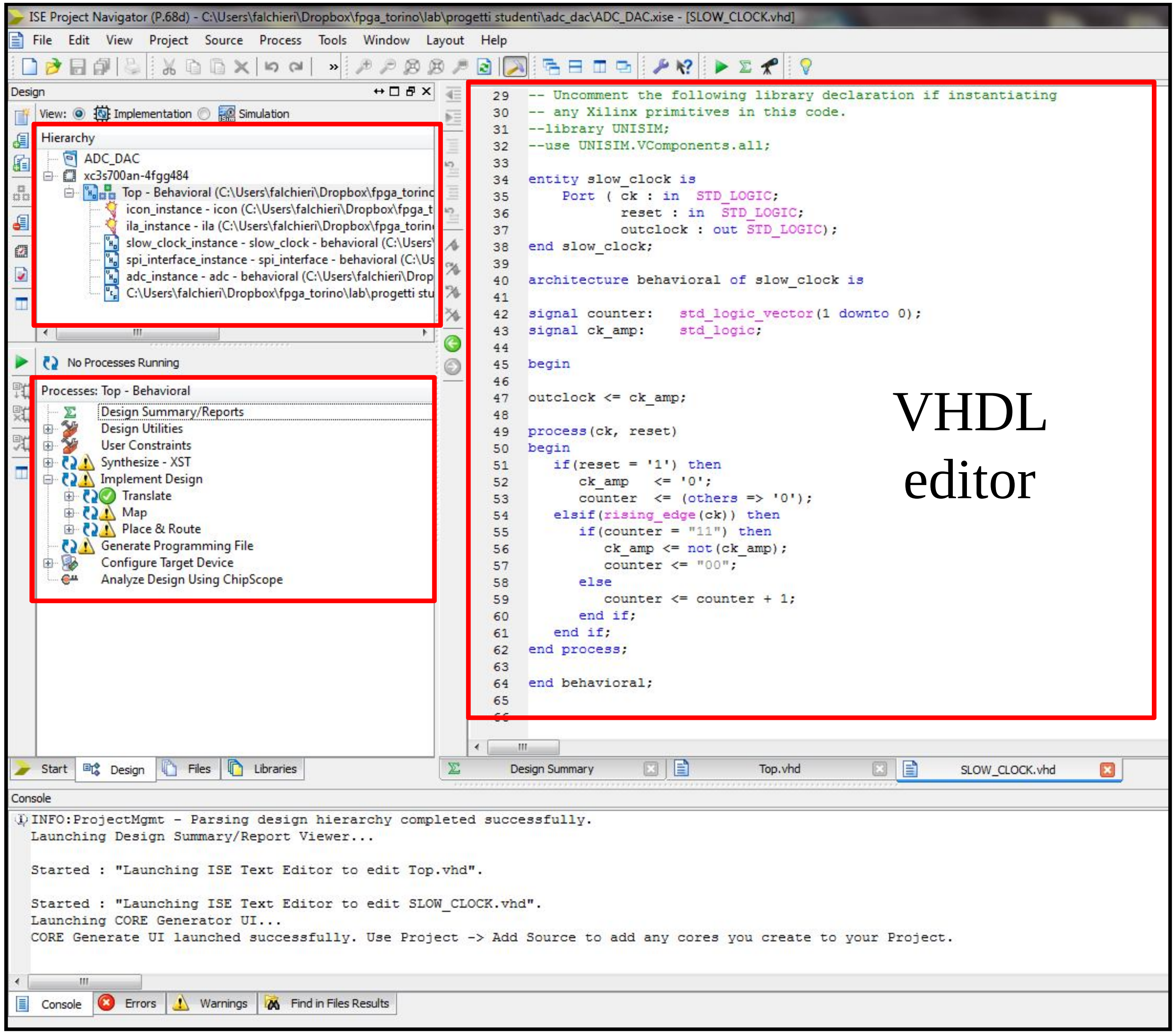
Outline

- FPGA Design flow
- FPGA debug
- Embedded processor design flow
- Simple design example on a demo board

Xilinx FPGA design flow: ISE

design hierarchy

design flow



ISE Project Navigator (P.68d) - C:\Users\falchieri\Dropbox\fpga_torino\lab\progetti studenti\adc_dac\ADC_DAC.xise - [SLOW_CLOCK.vhd]

Hierarchy

- ADC_DAC
  - xc3s700an-4fgg484
    - Top - Behavioral (C:\Users\falchieri\Dropbox\fpga_torinc
      - icon_instance - icon (C:\Users\falchieri\Dropbox\fpga_t
      - ila_instance - ila (C:\Users\falchieri\Dropbox\fpga_torin
      - slow_clock_instance - slow_clock - behavioral (C:\Users\
      - spi_interface_instance - spi_interface - behavioral (C:\Us
      - adc_instance - adc - behavioral (C:\Users\falchieri\Drop
      - C:\Users\falchieri\Dropbox\fpga_torino\lab\progetti stu

No Processes Running

Processes: Top - Behavioral

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
  - Translate
  - Map
  - Place & Route
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

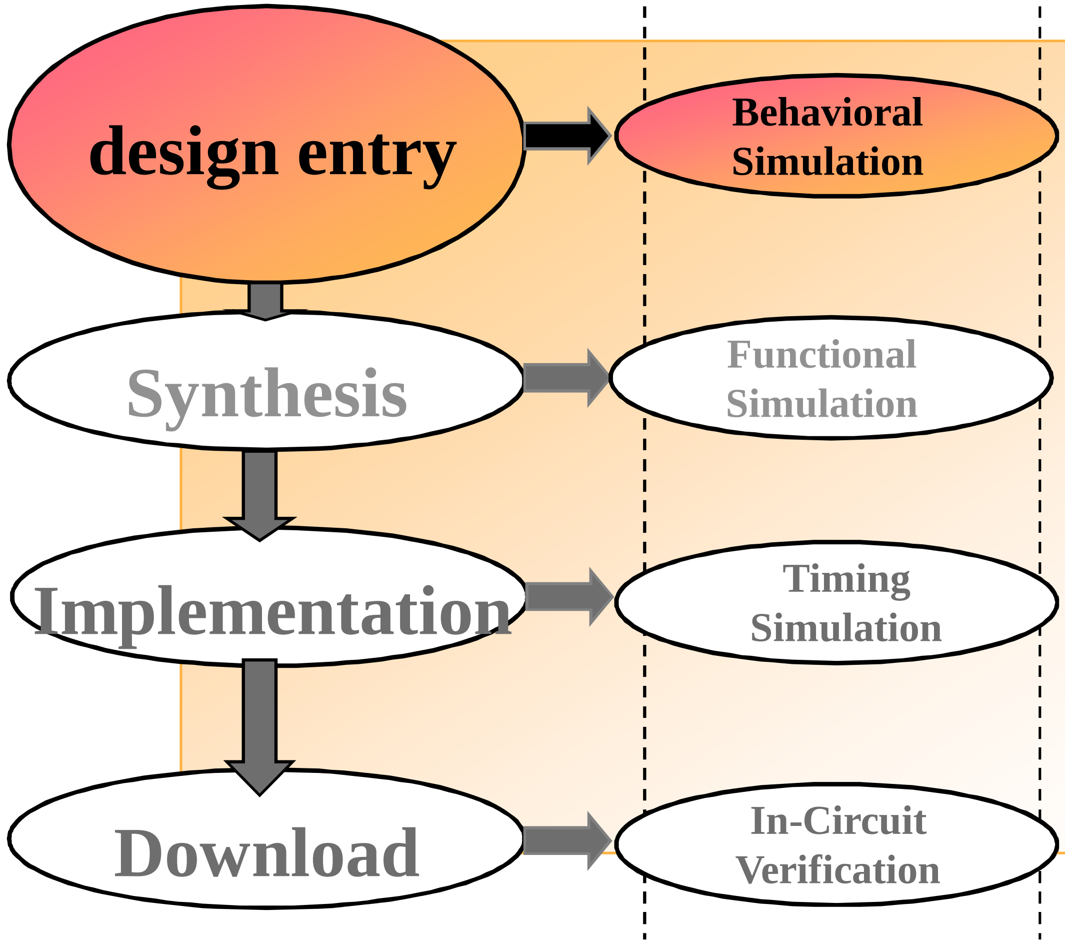
```vhdl
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity slow_clock is
    Port ( ck : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           outclock : out STD_LOGIC);
end slow_clock;

architecture behavioral of slow_clock is

signal counter:   std_logic_vector(1 downto 0);
signal ck_amp:    std_logic;

begin

outclock <= ck_amp;

process(ck, reset)
begin
   if(reset = '1') then
      ck_amp    <= '0';
      counter   <= (others => '0');
   elsif(rising_edge(ck)) then
      if(counter = "11") then
         ck_amp <= not(ck_amp);
         counter <= "00";
      else
         counter <= counter + 1;
      end if;
   end if;
end process;

end behavioral;
```

VHDL editor

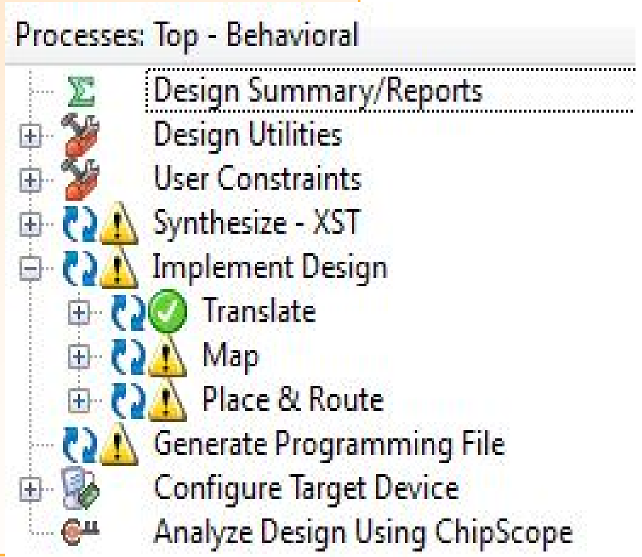
Console

```
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...

Started : "Launching ISE Text Editor to edit Top.vhd".

Started : "Launching ISE Text Editor to edit SLOW_CLOCK.vhd".
Launching CORE Generator UI...
CORE Generate UI launched successfully. Use Project -> Add Source to add any cores you create to your Project.
```

Xilinx FPGA design flow



Implement your design using VHDL or Verilog

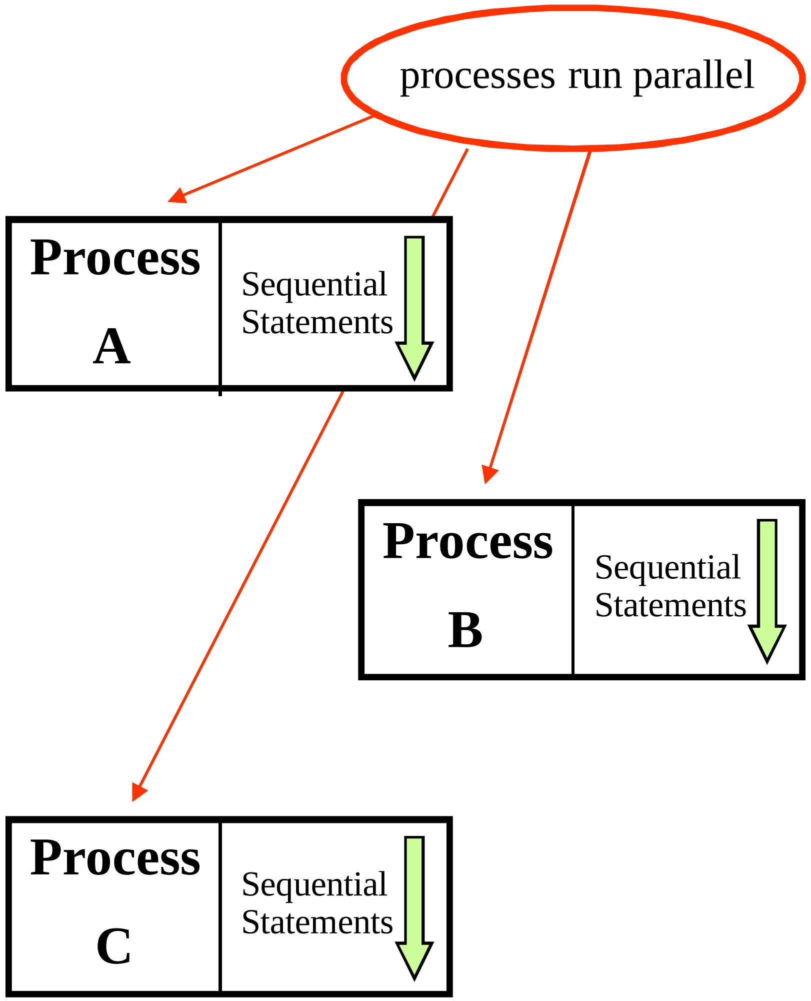


Processes: Top - Behavioral
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
  - Translate
  - Map
  - Place & Route
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

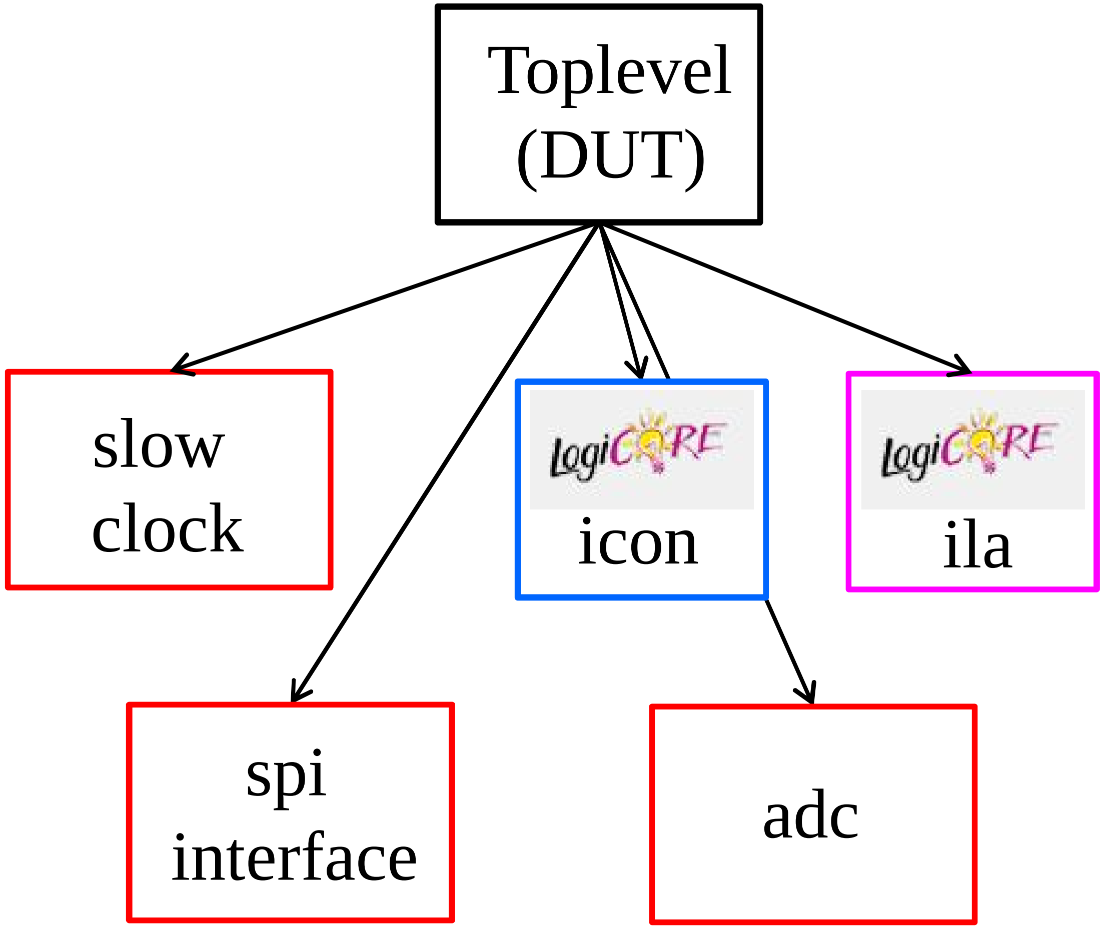
VHDL for modeling digital systems

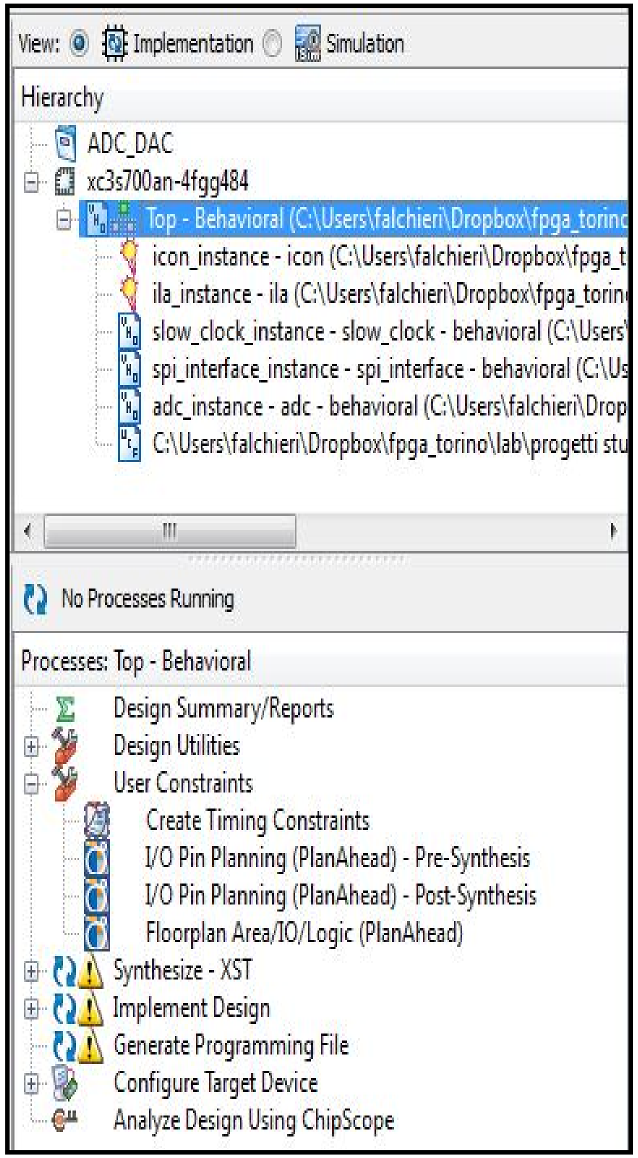
VHDL is intended for describing and modeling a digital system at various levels from the most abstract down to the gate level. VHDL is meant as a modeling language for specification and simulation, but can also be used for synthesis.

<u>Advantages</u>:
- able to describe concurrent instructions
- the code can be re-used from one project to an other, the same for single blocks (cores)
- can be simulated and synthesized
- takes less time than schematics



Design hierarchy




View: Implementation Simulation

Hierarchy

- ADC_DAC
- xc3s700an-4fgg484
  - Top - Behavioral (C:\Users\falchieri\Dropbox\fpga_torino
    - icon_instance - icon (C:\Users\falchieri\Dropbox\fpga_t
    - ila_instance - ila (C:\Users\falchieri\Dropbox\fpga_torin
    - slow_clock_instance - slow_clock - behavioral (C:\Users\
    - spi_interface_instance - spi_interface - behavioral (C:\Us
    - adc_instance - adc - behavioral (C:\Users\falchieri\Drop
    - C:\Users\falchieri\Dropbox\fpga_torino\lab\progetti stu

No Processes Running

Processes: Top - Behavioral

- Design Summary/Reports
- Design Utilities
- User Constraints
  - Create Timing Constraints
  - I/O Pin Planning (PlanAhead) - Pre-Synthesis
  - I/O Pin Planning (PlanAhead) - Post-Synthesis
  - Floorplan Area/IO/Logic (PlanAhead)
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

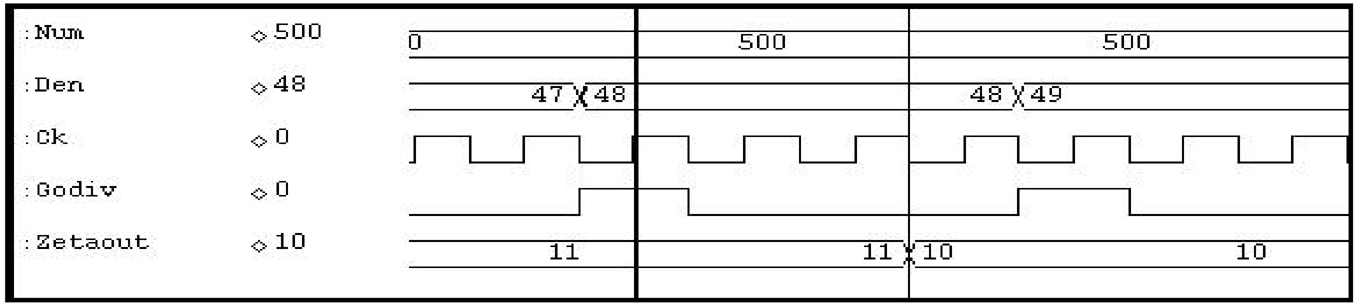
VHDL entry

• Write your own VHDL

```
run0:process
  begin
  if(rising_edge(ck)) then
       if (godiv = '1') then
                 num_temp <= num;
                 den_temp <= den;
       end if;
  end if;
end process run0;
```

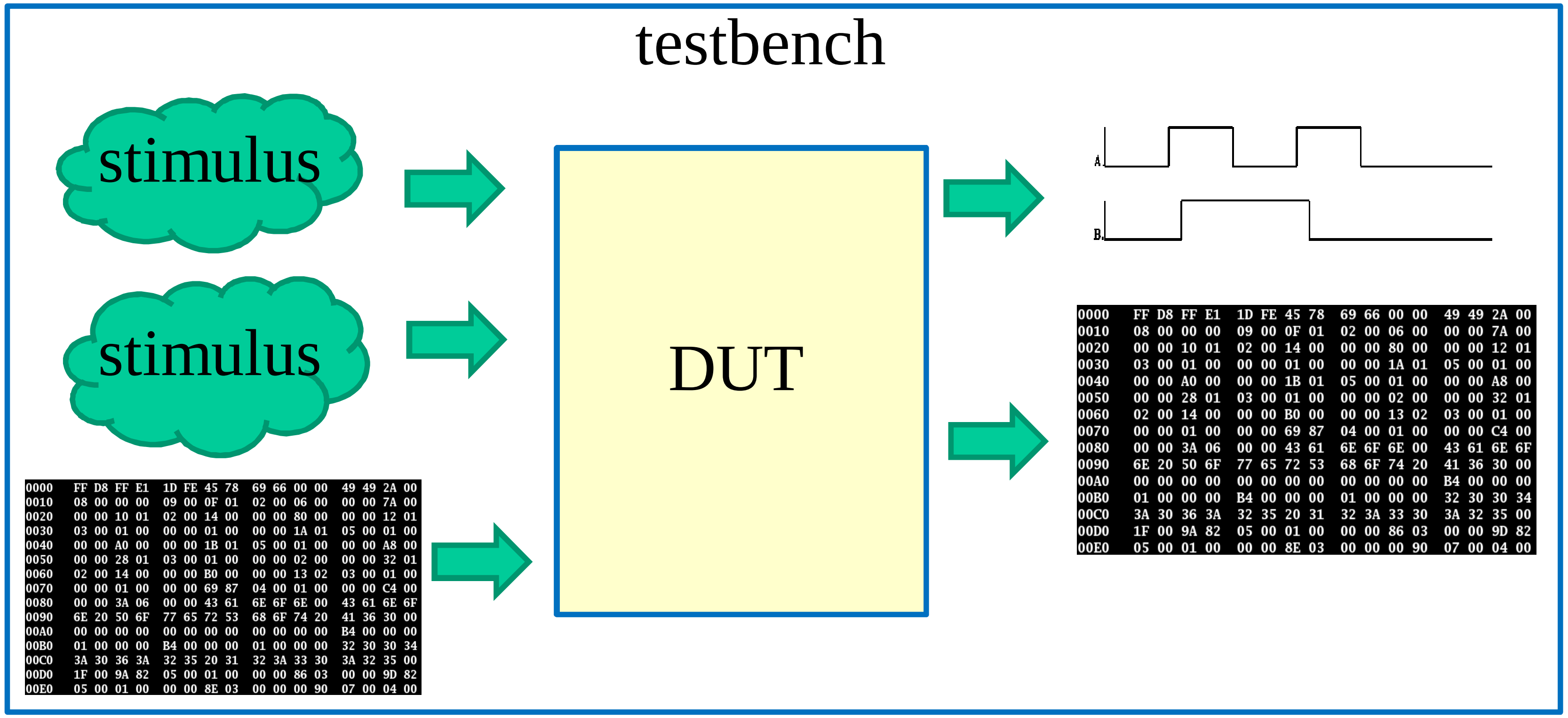
• Make extensive use of the soft cores available for free from manufacturers or directly on the Web (have a look to www.opencores.org or www.ohwr.org if interested)
For example use the Xilinx Core Generator if you need a fixed point divider block. A wizard allows you to choose the divider parameters and produces a synthesizable core.

• Simulate the code!



ISIM

Xilinx **ISIM** is a Hardware Description Language (HDL) simulator that lets you perform behavioral and timing simulations for VHDL, Verilog and mixed VHDL/Verilog language designs

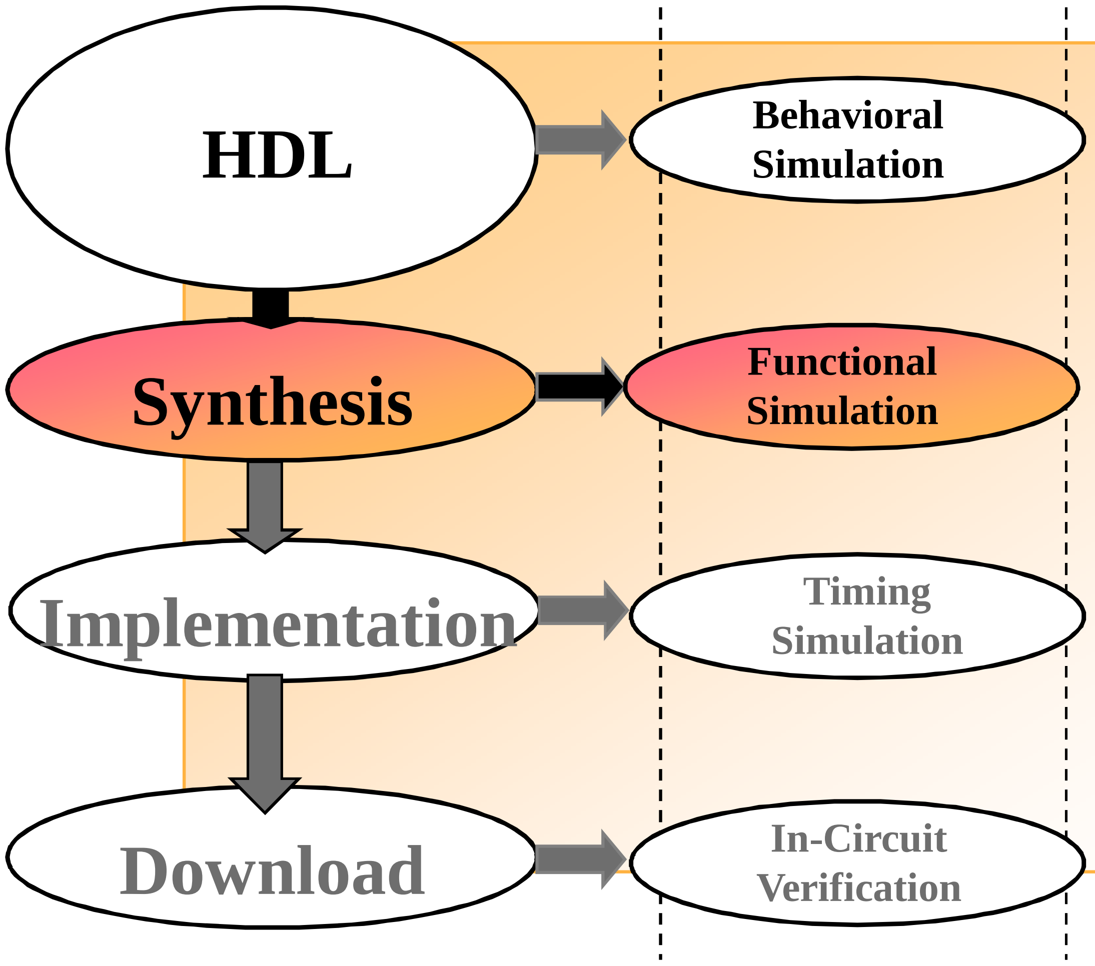


```
0000    FF D8 FF E1  1D FE 45 78  69 66 00 00  49 49 2A 00
0010    08 00 00 00  09 00 0F 01  02 00 06 00  00 00 7A 00
0020    00 00 10 01  02 00 14 00  00 00 80 00  00 00 12 01
0030    03 00 01 00  00 00 01 00  00 00 1A 01  05 00 01 00
0040    00 00 A0 00  00 00 1B 01  05 00 01 00  00 00 A8 00
0050    00 00 28 01  03 00 01 00  00 00 02 00  00 00 32 01
0060    02 00 14 00  00 00 B0 00  00 00 13 02  03 00 01 00
0070    00 00 01 00  00 00 69 87  04 00 01 00  00 00 C4 00
0080    00 00 3A 06  00 00 43 61  6E 6F 6E 00  43 61 6E 6F
0090    6E 20 50 6F  77 65 72 53  68 6F 74 20  41 36 30 00
00A0    00 00 00 00  00 00 00 00  00 00 00 00  B4 00 00 00
00B0    01 00 00 00  B4 00 00 00  01 00 00 00  32 30 30 34
00C0    3A 30 36 3A  32 35 20 31  32 3A 33 30  3A 32 35 00
00D0    1F 00 9A 82  05 00 01 00  00 00 86 03  00 00 9D 82
00E0    05 00 01 00  00 00 8E 03  00 00 00 90  07 00 04 00
```

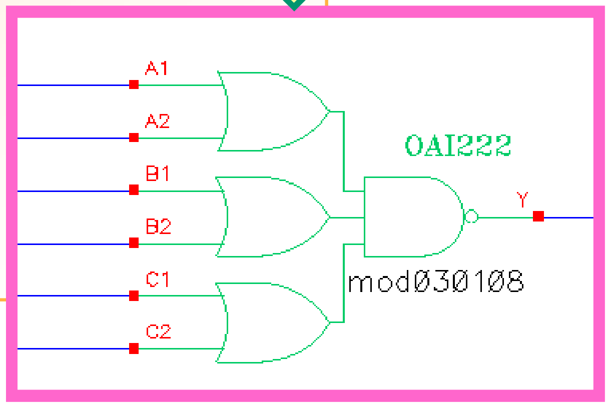
```
0000    FF D8 FF E1  1D FE 45 78  69 66 00 00  49 49 2A 00
0010    08 00 00 00  09 00 0F 01  02 00 06 00  00 00 7A 00
0020    00 00 10 01  02 00 14 00  00 00 80 00  00 00 12 01
0030    03 00 01 00  00 00 01 00  00 00 1A 01  05 00 01 00
0040    00 00 A0 00  00 00 1B 01  05 00 01 00  00 00 A8 00
0050    00 00 28 01  03 00 01 00  00 00 02 00  00 00 32 01
0060    02 00 14 00  00 00 B0 00  00 00 13 02  03 00 01 00
0070    00 00 01 00  00 00 69 87  04 00 01 00  00 00 C4 00
0080    00 00 3A 06  00 00 43 61  6E 6F 6E 00  43 61 6E 6F
0090    6E 20 50 6F  77 65 72 53  68 6F 74 20  41 36 30 00
00A0    00 00 00 00  00 00 00 00  00 00 00 00  B4 00 00 00
00B0    01 00 00 00  B4 00 00 00  01 00 00 00  32 30 30 34
00C0    3A 30 36 3A  32 35 20 31  32 3A 33 30  3A 32 35 00
00D0    1F 00 9A 82  05 00 01 00  00 00 86 03  00 00 9D 82
00E0    05 00 01 00  00 00 8E 03  00 00 00 90  07 00 04 00
```

Its use is convenient when the DUT equals to the ISE projects. Otherwise when the simulation blocks are much larger, other simulation tools perform better, for instance **Modelsim**

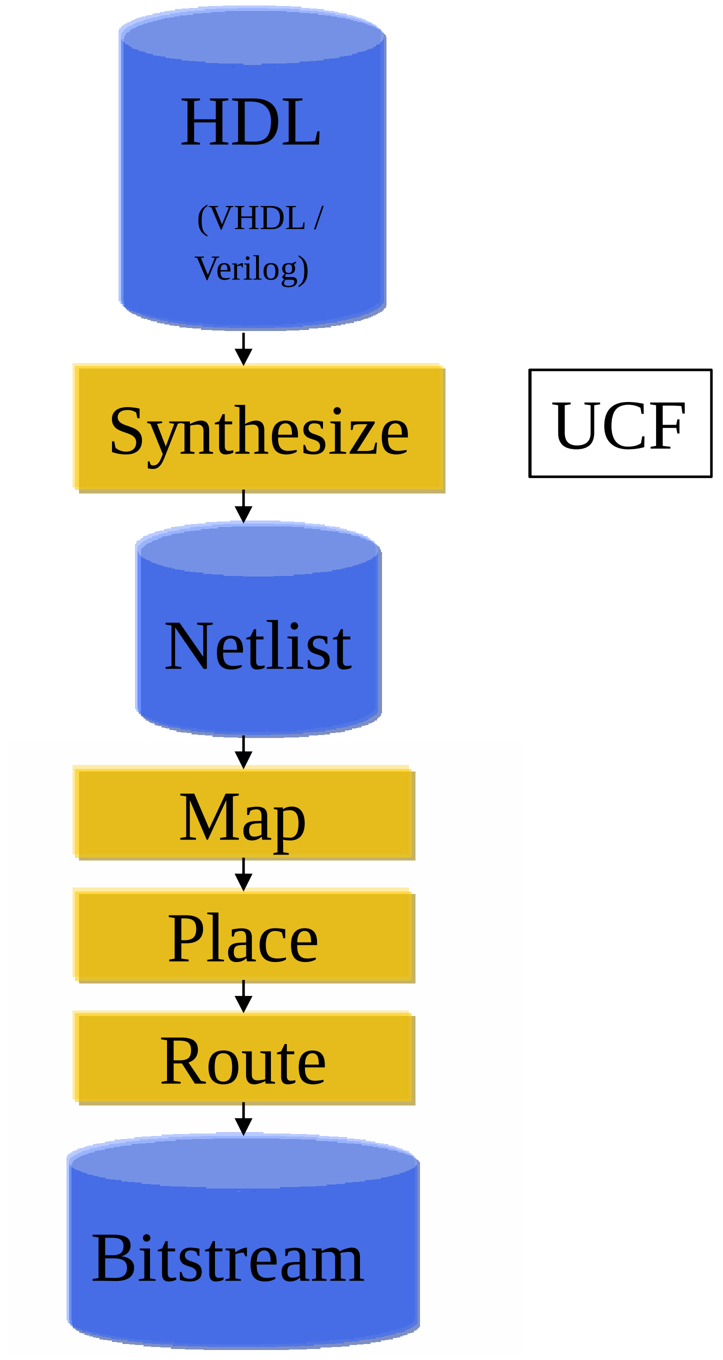
FPGA design flow



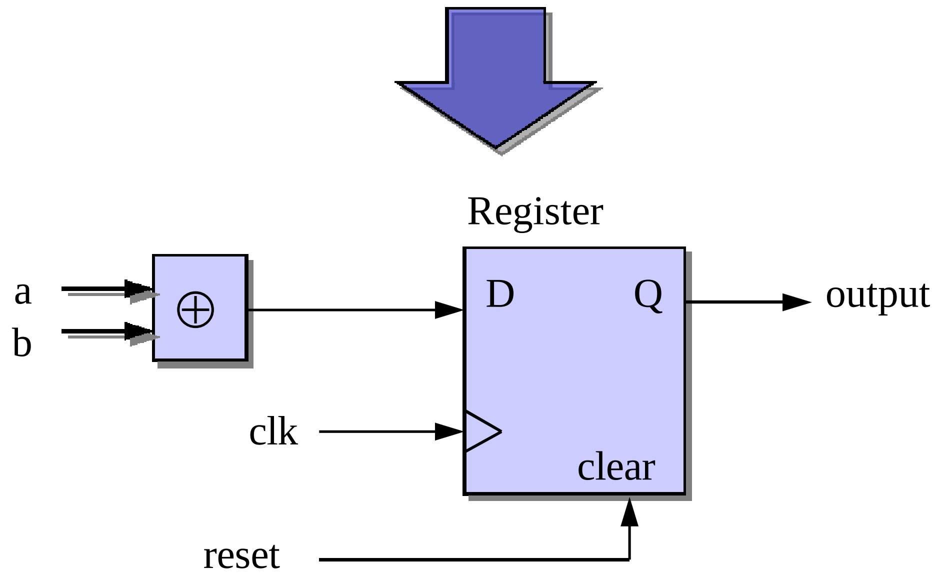
Synthesize the design to create an FPGA netlist



Logic Synthesis

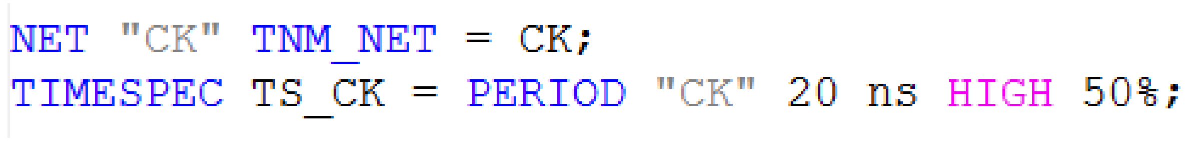


```
process(clk, reset)
begin
        if reset = ‘1‘ then
                output <= ‘0‘;
        elsif rising_edge(clk) then
                output <= a XOR b;
        end if;
end process;
```



Timing constraints

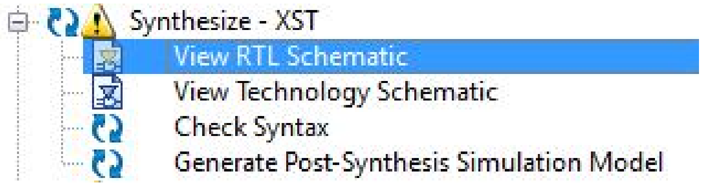
The synthesis process tries to satisfy the constraints put by the designer in the UCF (User Constraint File):
it is usually a timing constraint

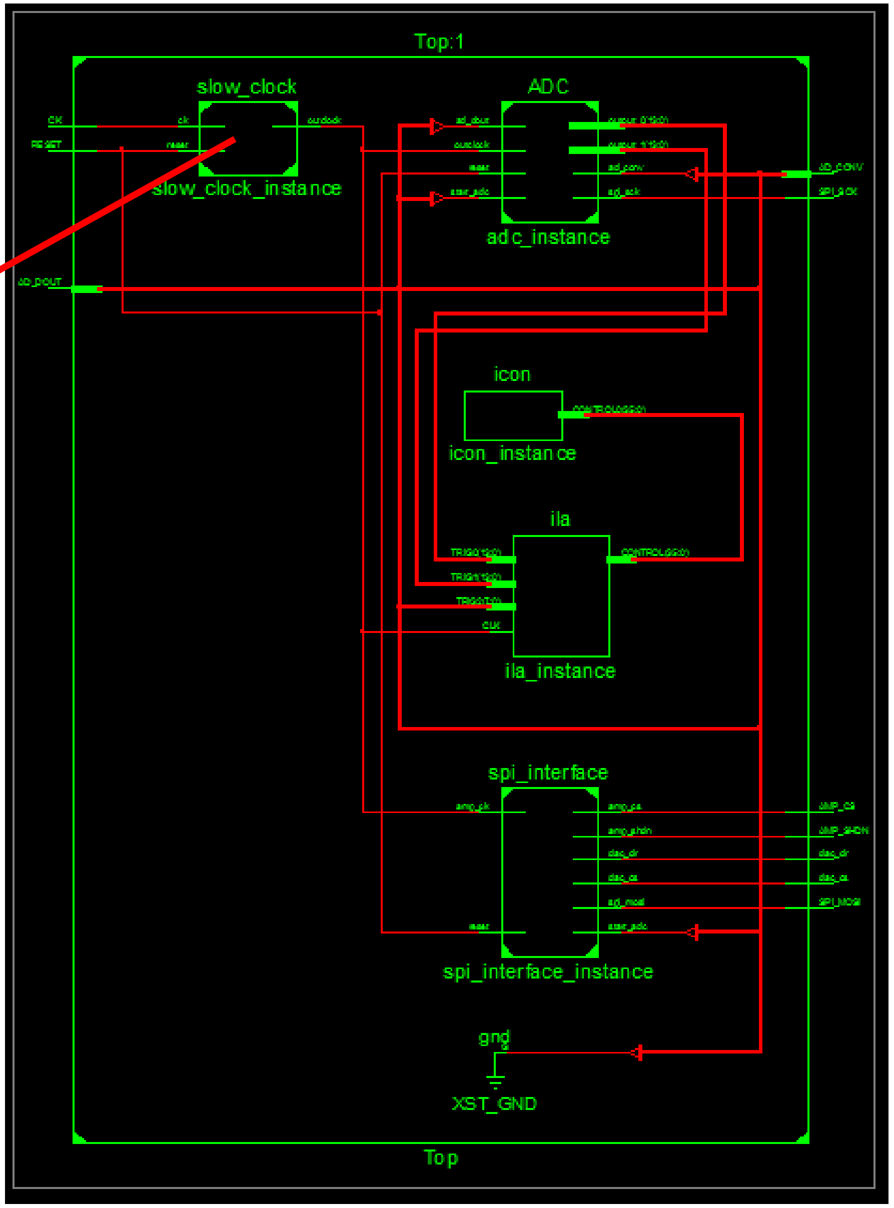


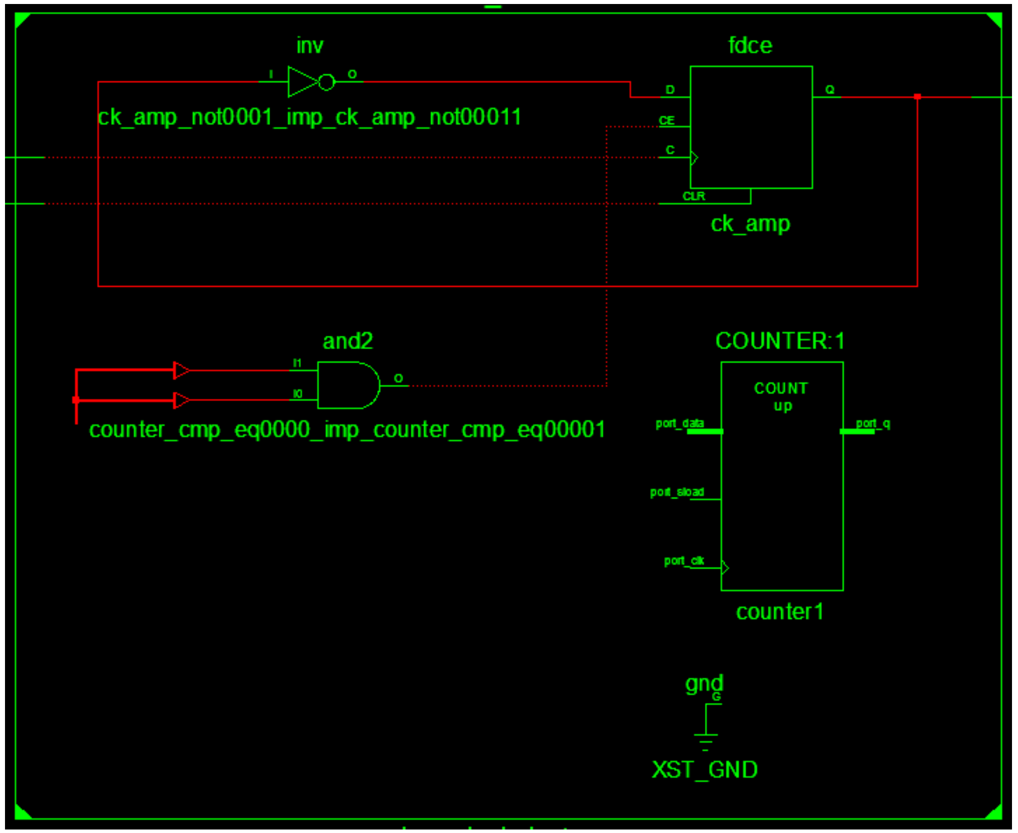
```
NET "CK" TNM_NET = CK;
TIMESPEC TS_CK = PERIOD "CK" 20 ns HIGH 50%;
```

This timing constraint asks the synthesizer (and later to the place & route tool) to build a circuit able to work at 50 MHz without having setup/hold violations.

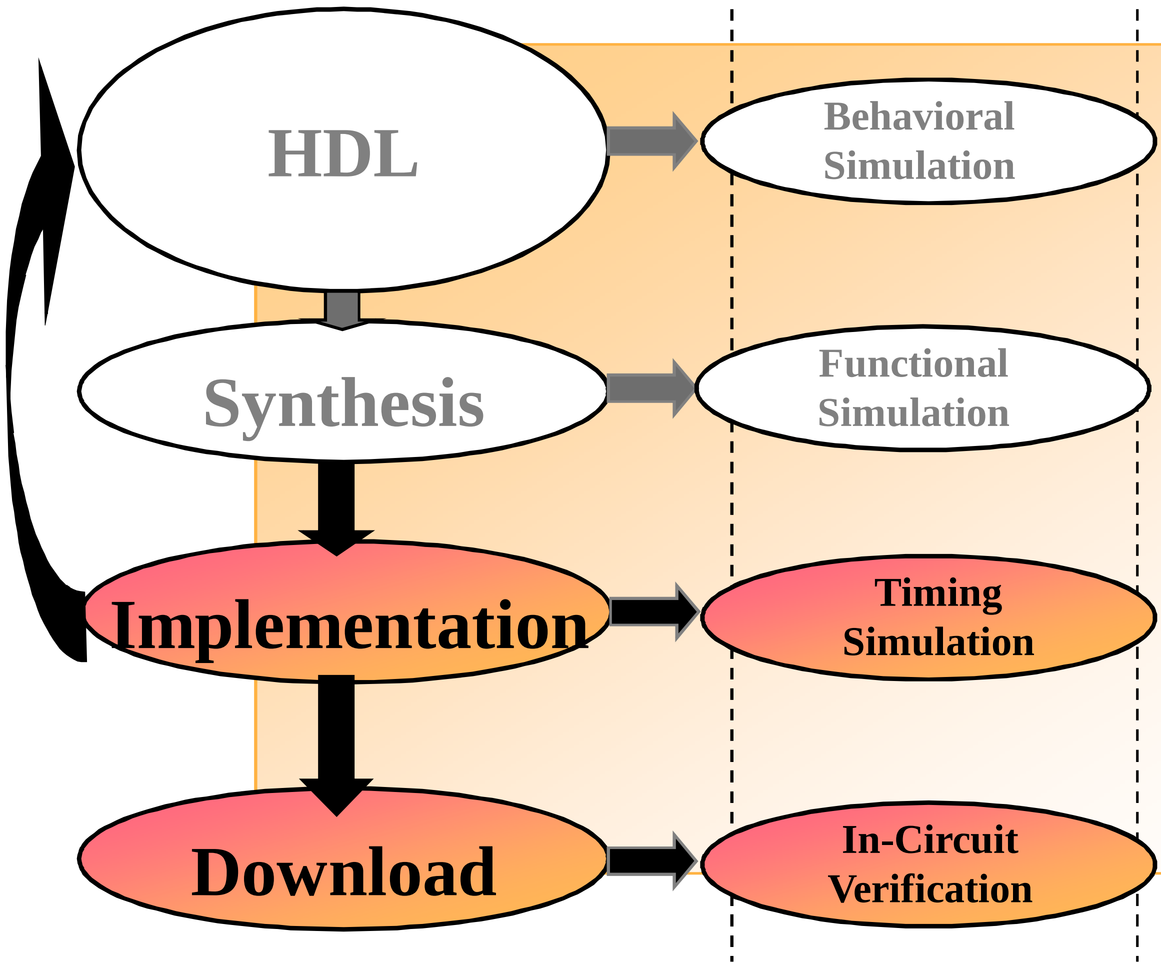
Navigating in the schematics

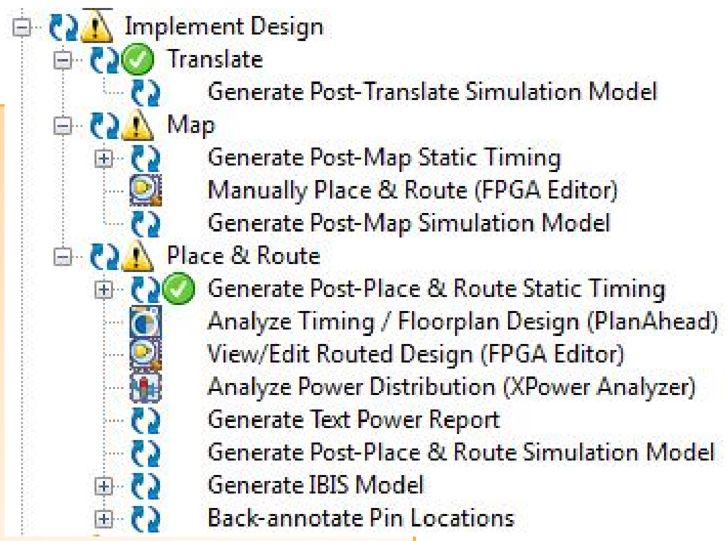

Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model




inv
fdce
ck_amp_not0001_imp_ck_amp_not00011
ck_amp
and2
counter_cmp_eq0000_imp_counter_cmp_eq00001
COUNTER:1
COUNT up
counter1
gnd
XST_GND

FPGA design flow



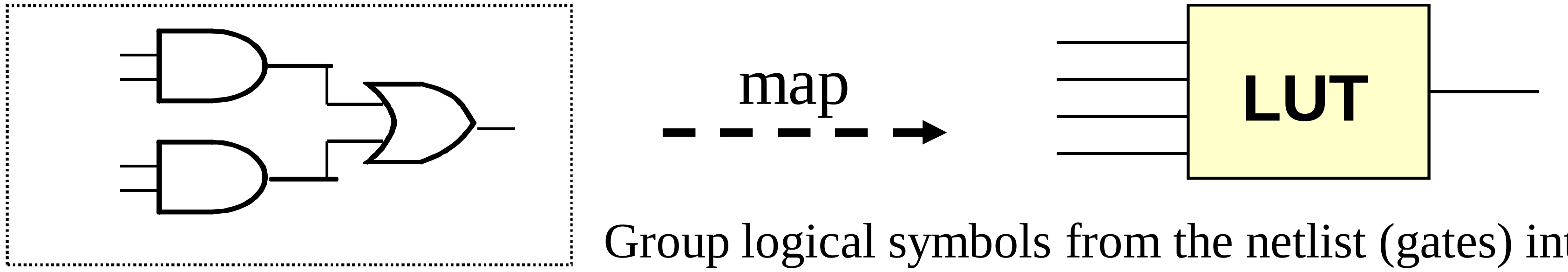


- Implement Design
  - Translate
    - Generate Post-Translate Simulation Model
  - Map
    - Generate Post-Map Static Timing
    - Manually Place & Route (FPGA Editor)
    - Generate Post-Map Simulation Model
  - Place & Route
    - Generate Post-Place & Route Static Timing
    - Analyze Timing / Floorplan Design (PlanAhead)
    - View/Edit Routed Design (FPGA Editor)
    - Analyze Power Distribution (XPower Analyzer)
    - Generate Text Power Report
    - Generate Post-Place & Route Simulation Model
    - Generate IBIS Model
    - Back-annotate Pin Locations

Translate, place and route, and generate a bitstream to download in the FPGA

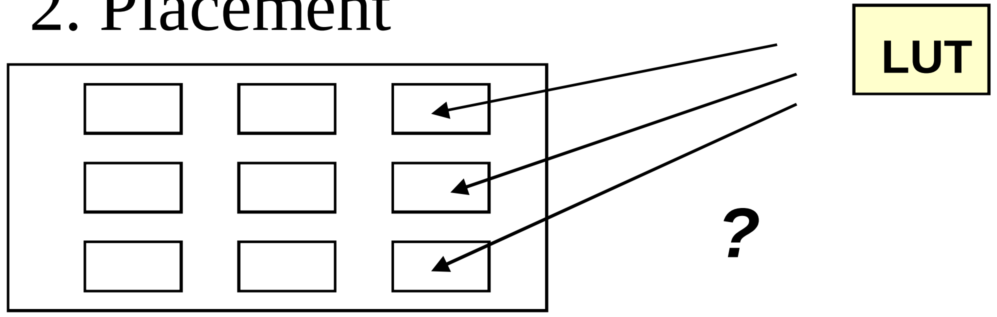
Implementation

1. Technology Mapping



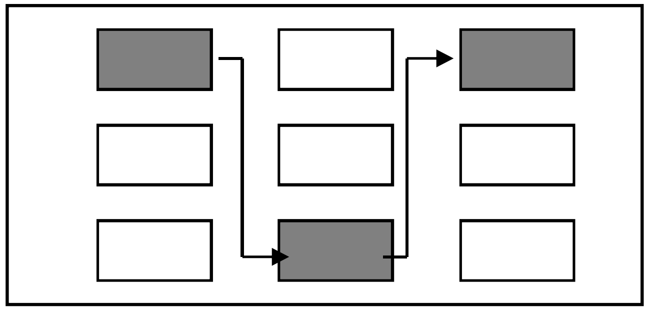
Group logical symbols from the netlist (gates) into physical components (slices and IOBs)

2. Placement



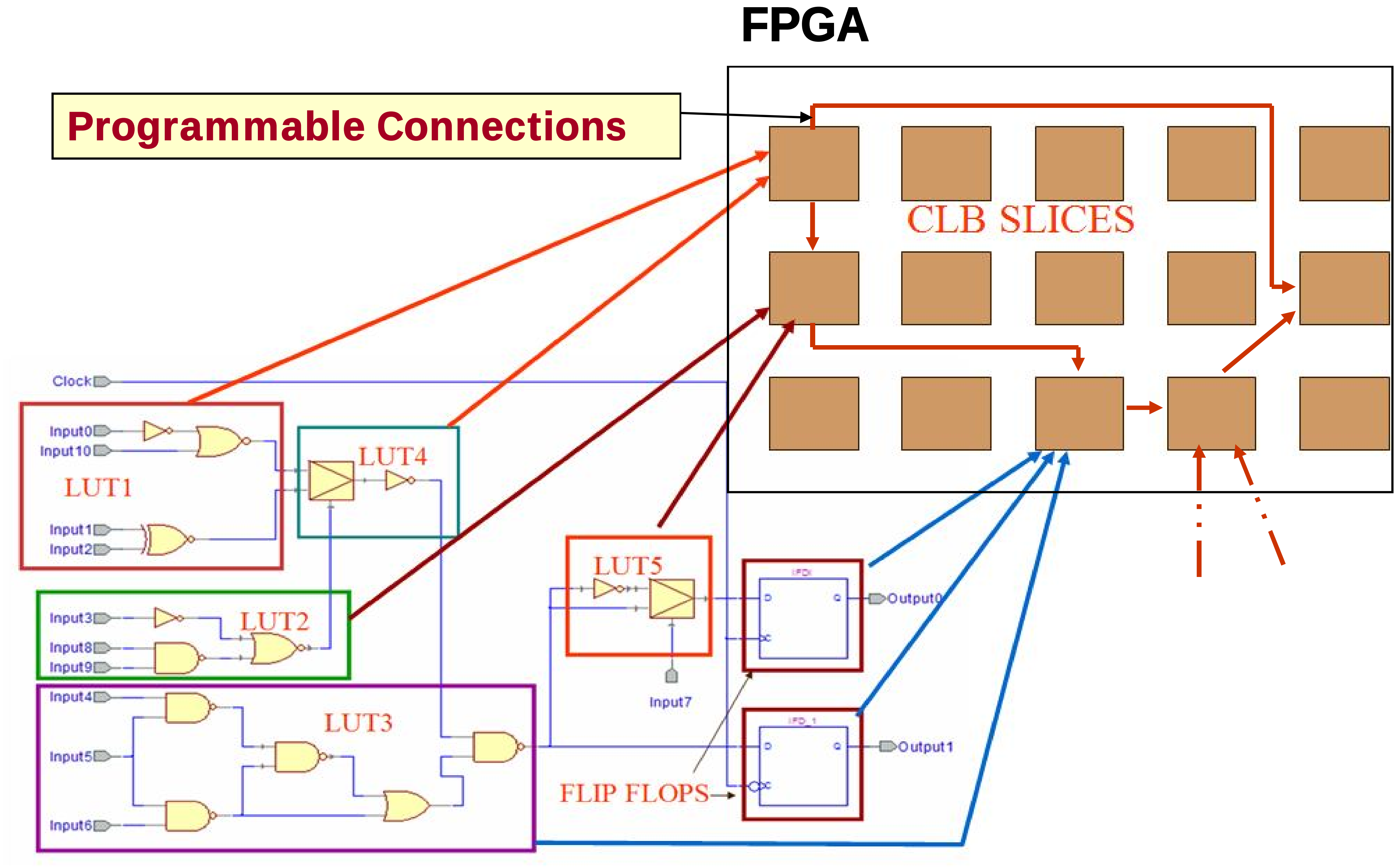
Assign a logical LUT to a physical location

3. Routing



Select wire segments and switches for interconnection

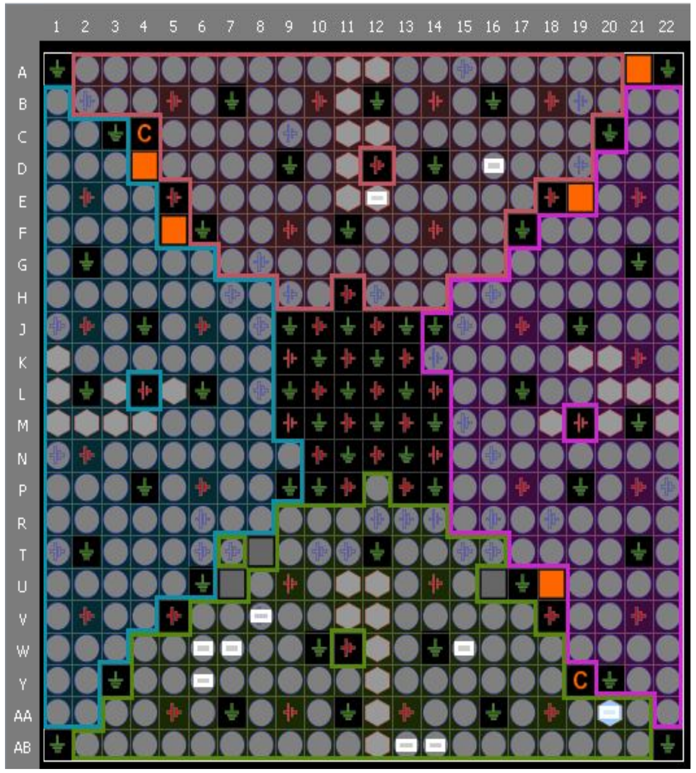
Routing Example



Physical constraints

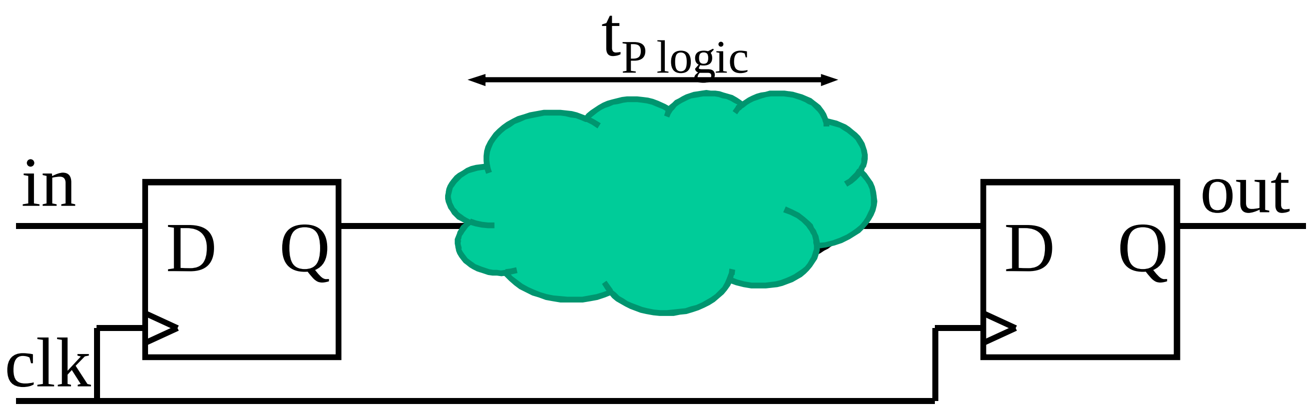
```
NET "SPI_MOSI" LOC = "AB14" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8;
NET "SPI_SCK" LOC = "AA20" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8;

# AMP pins
NET "AMP_CS" LOC = "W6" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8;
NET "AMP_SHDN" LOC = "W15" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8;

# ADC pins
NET "AD_CONV" LOC = "Y6" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8;
NET "AD_DOUT" LOC = "D16" |IOSTANDARD = LVCMOS33;

#DAC pins
NET "DAC_CS" LOC = "W7" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8 ;
NET "DAC_CLR" LOC = "AB13" |IOSTANDARD = LVCMOS33 |SLEW = SLOW |DRIVE = 8 ;

#system pins
NET "CK" LOC = "E12" |IOSTANDARD = LVCMOS33;
NET "RESET" LOC = "V8" |IOSTANDARD = LVCMOS33;
```

The place & route tools place the logical I/O signals (in the VHDL entity of the toplevel) in the IOBs, checking that the banking rules are respected



Static Timing Analyzer

- Performs static analysis of the circuit performance
- Reports critical paths with all sources of delays
- Determines maximum clock frequency
- Critical Path – The Longest Path From Outputs of Registers to Inputs of Registers



$$\boxed{t_{Critical} = t_{P\ FF} + t_{P\ logic} + t_{S\ FF}}$$

- Min. Clock Period = Length of The Critical Path
- Max. Clock Frequency = 1 / Min. Clock Period

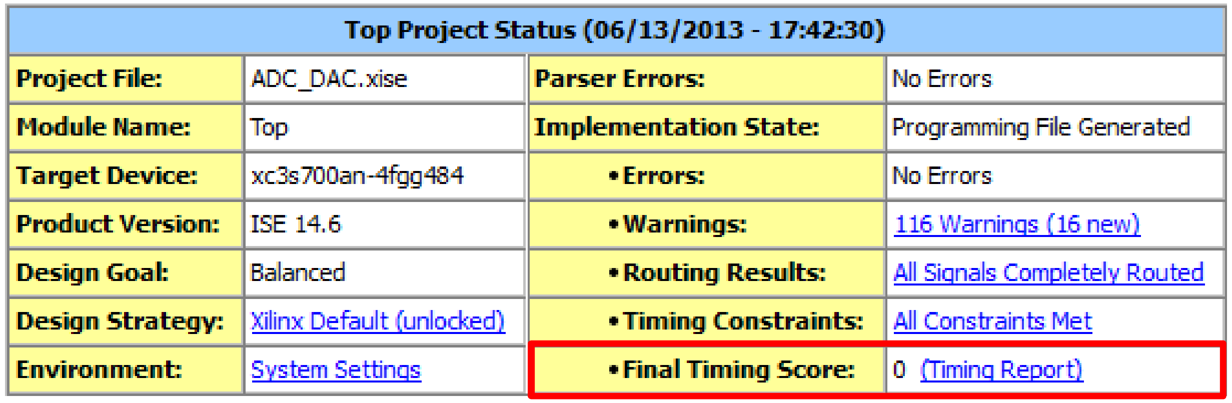
Timing closure

Timing closure is achieved when all timing constraints for a design are met under all legal operating conditions PVT:

- **Process**
- **Voltage**
- **Temperature**

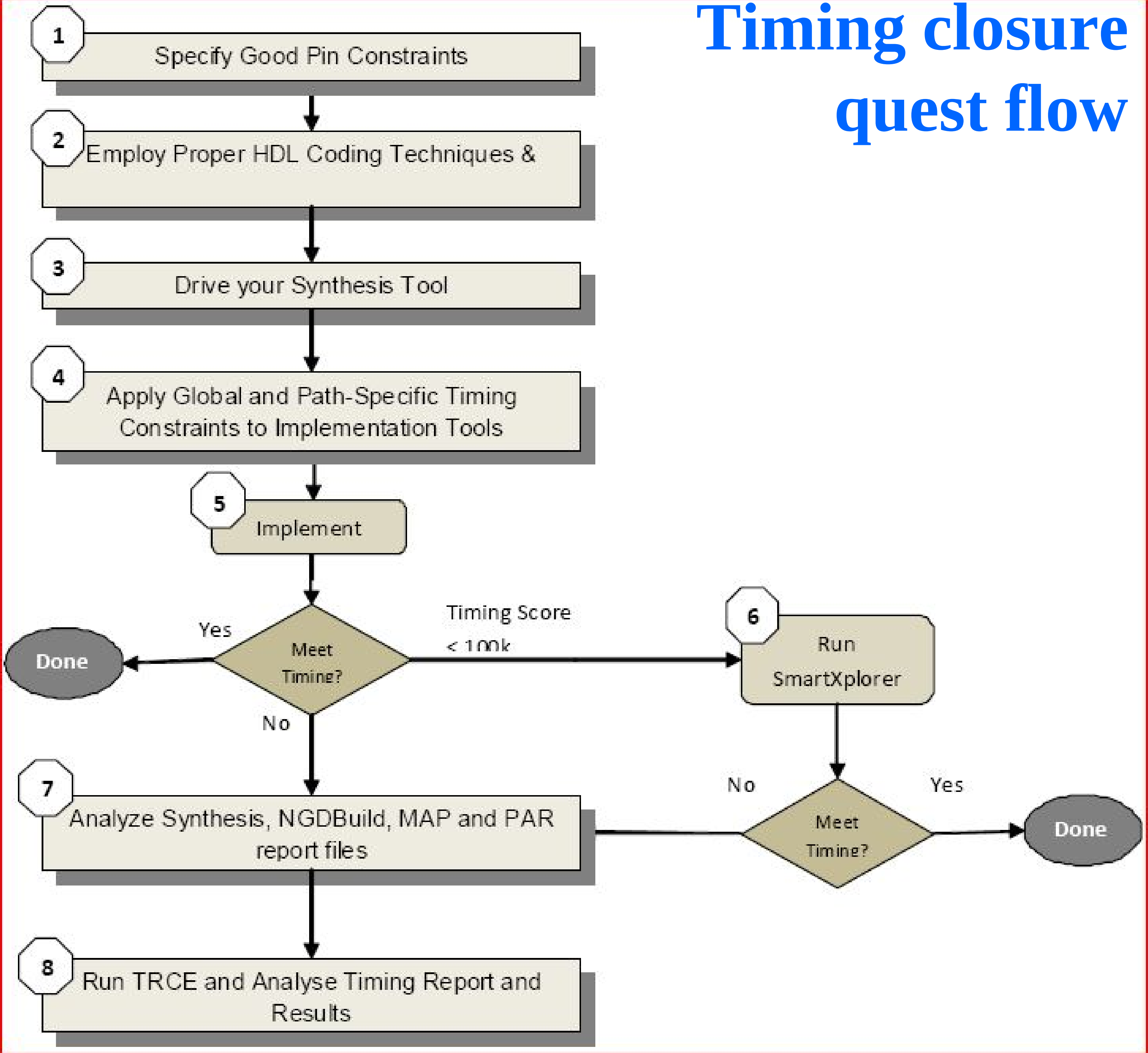
Timing closure is achieved when the design is fully constrained and the **timing score** is **zero**. The timing score:

- is the total value representing the timing analysis for all constraints and the amount by which the constraints are failing
- is the *sum in picoseconds of all timing constraints that have not been met*

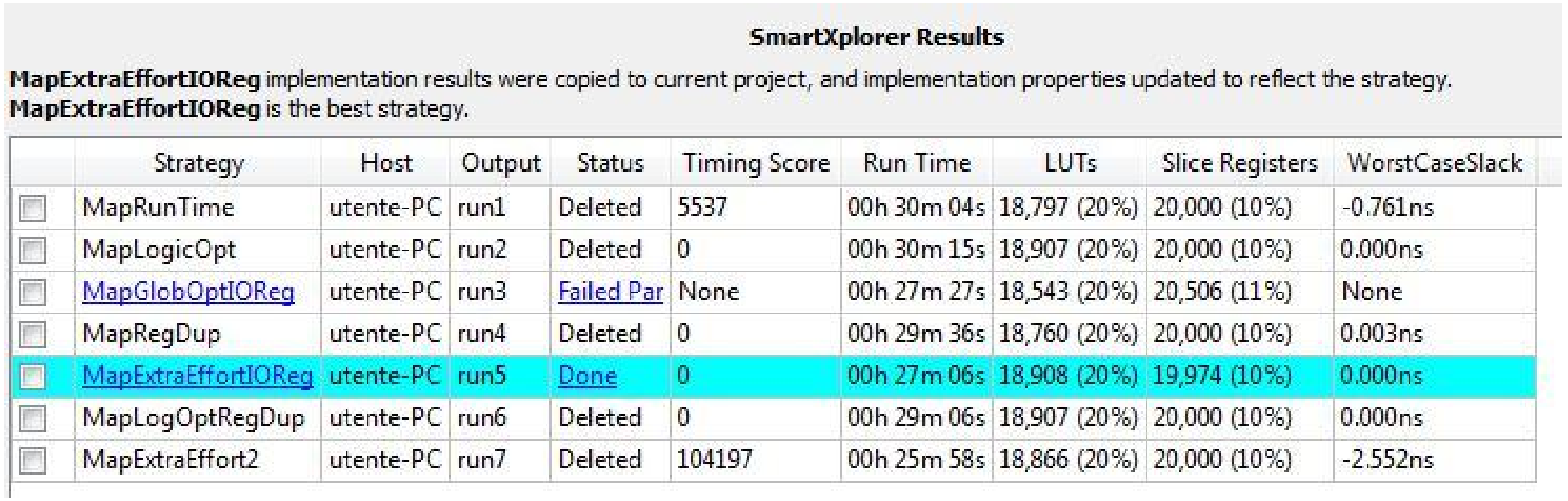


Top Project Status (06/13/2013 - 17:42:30)			
Project File:	ADC_DAC.xise	**Parser Errors:**	No Errors
Module Name:	Top	**Implementation State:**	Programming File Generated
Target Device:	xc3s700an-4fgg484	• **Errors:**	No Errors
Product Version:	ISE 14.6	• **Warnings:**	116 Warnings (16 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Timing closure quest flow


1
Specify Good Pin Constraints
2
Employ Proper HDL Coding Techniques &
3
Drive your Synthesis Tool
4
Apply Global and Path-Specific Timing Constraints to Implementation Tools
5
Implement
Meet Timing?
Yes
Done
Timing Score < 100k
6
Run SmartXplorer
No
7
Analyze Synthesis, NGDBuild, MAP and PAR report files
No
Yes
Meet Timing?
Done
8
Run TRCE and Analyse Timing Report and Results

SmartXplorer



SmartXplorer Results

MapExtraEffortIOReg implementation results were copied to current project, and implementation properties updated to reflect the strategy.
MapExtraEffortIOReg is the best strategy.

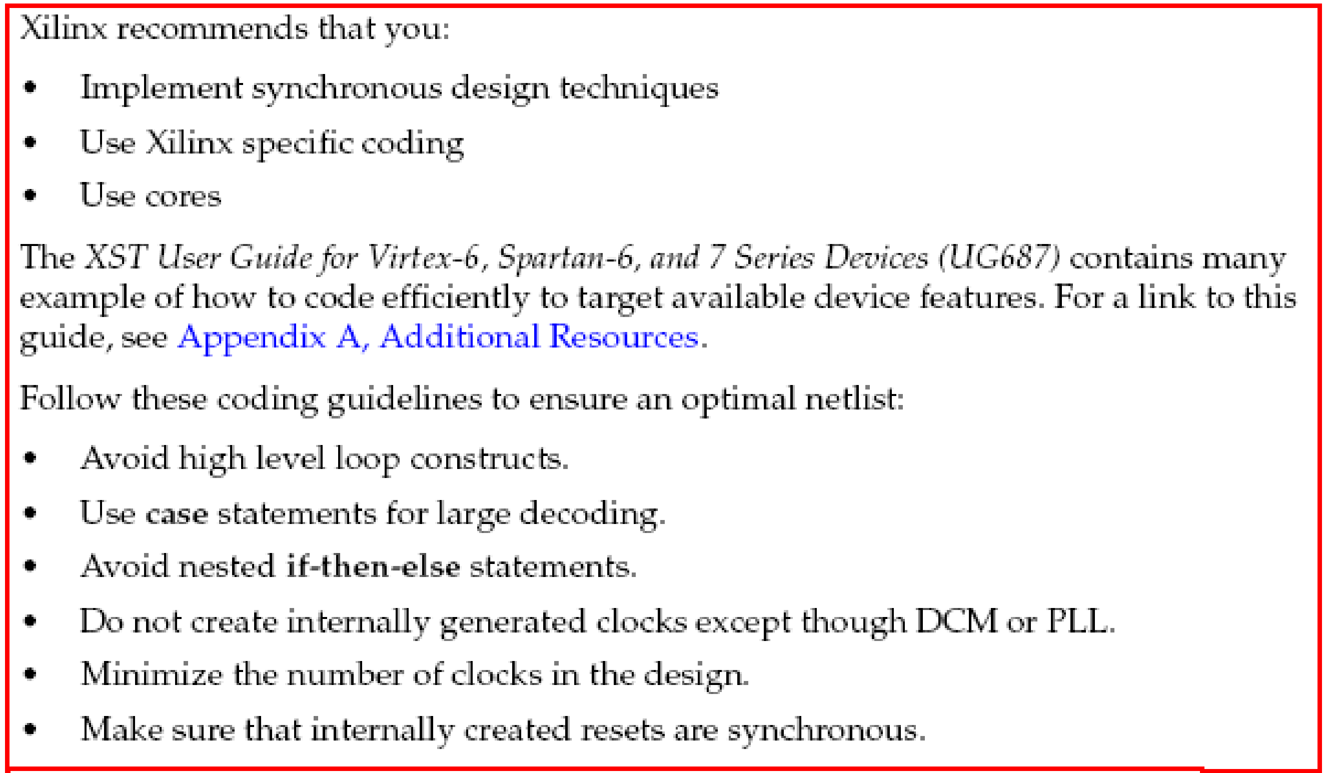
	Strategy	Host	Output	Status	Timing Score	Run Time	LUTs	Slice Registers	WorstCaseSlack
☐	MapRunTime	utente-PC	run1	Deleted	5537	00h 30m 04s	18,797 (20%)	20,000 (10%)	-0.761ns
☐	MapLogicOpt	utente-PC	run2	Deleted	0	00h 30m 15s	18,907 (20%)	20,000 (10%)	0.000ns
☐	MapGlobOptIOReg	utente-PC	run3	Failed Par	None	00h 27m 27s	18,543 (20%)	20,506 (11%)	None
☐	MapRegDup	utente-PC	run4	Deleted	0	00h 29m 36s	18,760 (20%)	20,000 (10%)	0.003ns
☐	MapExtraEffortIOReg	utente-PC	run5	Done	0	00h 27m 06s	18,908 (20%)	19,974 (10%)	0.000ns
☐	MapLogOptRegDup	utente-PC	run6	Deleted	0	00h 29m 06s	18,907 (20%)	20,000 (10%)	0.000ns
☐	MapExtraEffort2	utente-PC	run7	Deleted	104197	00h 25m 58s	18,866 (20%)	20,000 (10%)	-2.552ns

SmartXplorer tries up to 7 different implementation strategies until the timing closure is achieved, if possible at all
Timing closure is difficult when:

• the percentage of usage of the FPGA resources is higher than 60-70 %

• the timing constraints are close to the physical limits of the device

Coding guidelines



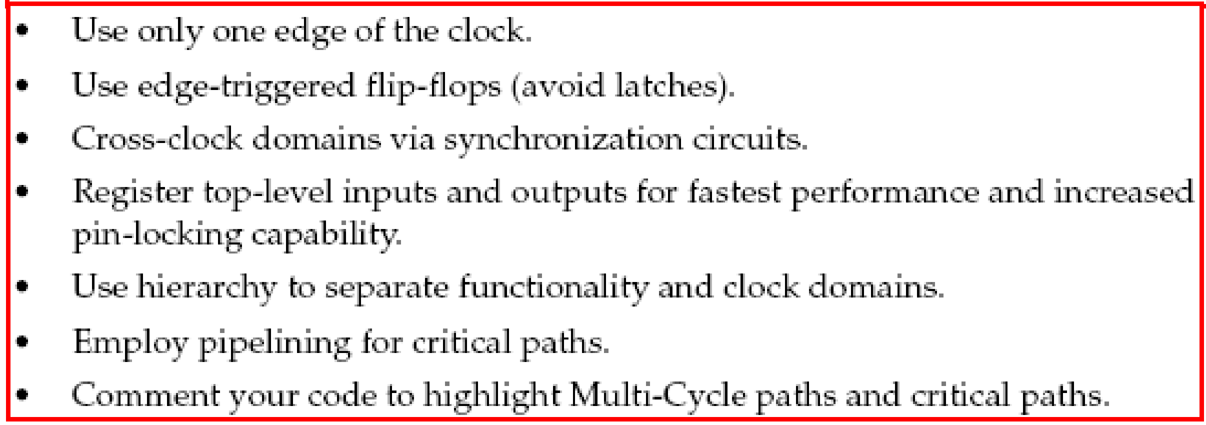
Xilinx recommends that you:

- Implement synchronous design techniques
- Use Xilinx specific coding
- Use cores

The *XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices (UG687)* contains many example of how to code efficiently to target available device features. For a link to this guide, see Appendix A, Additional Resources.

Follow these coding guidelines to ensure an optimal netlist:

- Avoid high level loop constructs.
- Use **case** statements for large decoding.
- Avoid nested **if-then-else** statements.
- Do not create internally generated clocks except though DCM or PLL.
- Minimize the number of clocks in the design.
- Make sure that internally created resets are synchronous.

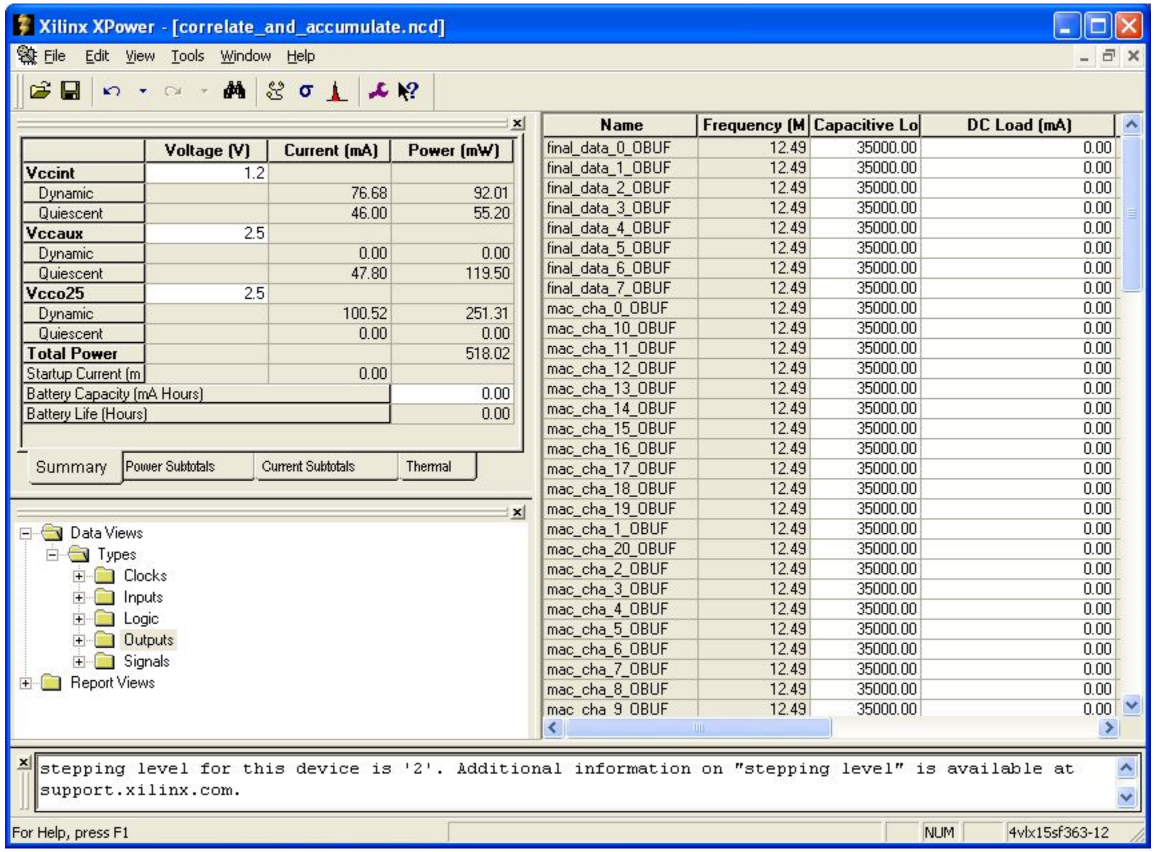


- Use only one edge of the clock.
- Use edge-triggered flip-flops (avoid latches).
- Cross-clock domains via synchronization circuits.
- Register top-level inputs and outputs for fastest performance and increased pin-locking capability.
- Use hierarchy to separate functionality and clock domains.
- Employ pipelining for critical paths.
- Comment your code to highlight Multi-Cycle paths and critical paths.

from UG612

Xpower analyzer

- XPower is used to estimate the power consumption and junction temperature of your FPGA
  - Reads an implemented design (NCD file) and timing constraint data
  - You supply activity rates, clock frequencies, capacitive loading on output pins, power supply data, and ambient temperature



Xilinx XPower - [correlate_and_accumulate.ncd]
File Edit View Tools Window Help

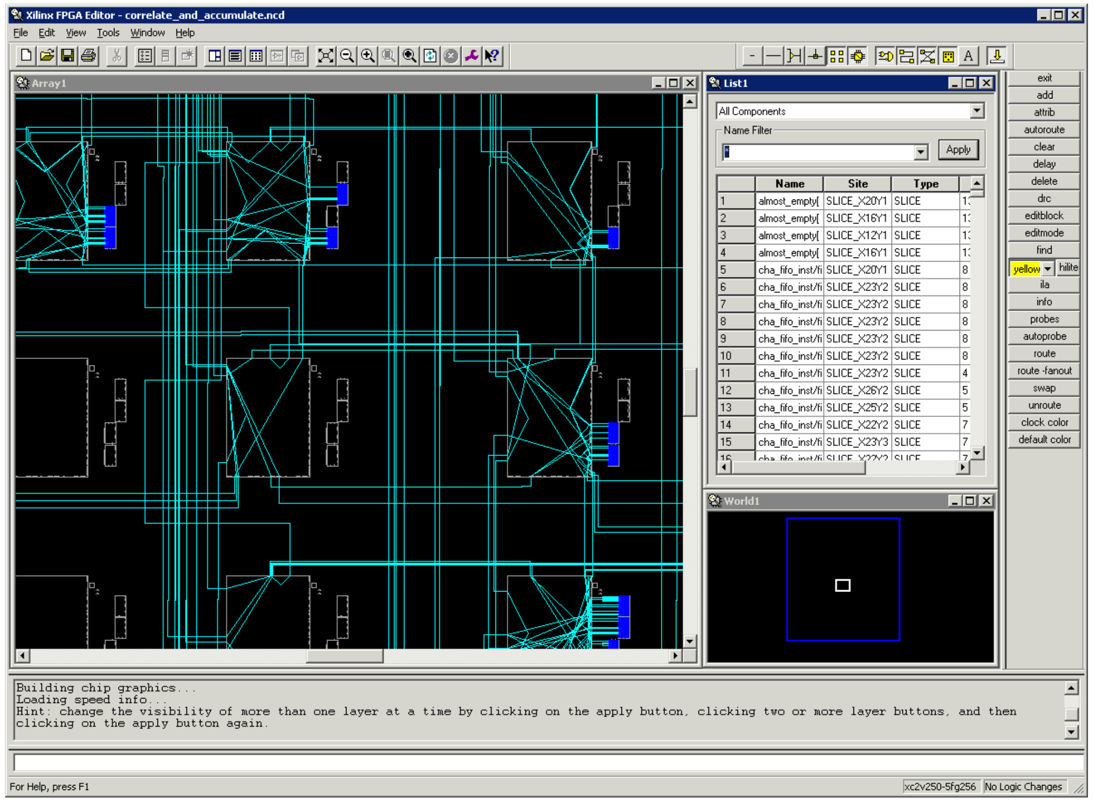
	Voltage (V)	Current (mA)	Power (mW)
Vccint	1.2		
Dynamic		76.68	92.01
Quiescent		46.00	55.20
Vccaux	2.5		
Dynamic		0.00	0.00
Quiescent		47.80	119.50
Vcco25	2.5		
Dynamic		100.52	251.31
Quiescent		0.00	0.00
Total Power			518.02
Startup Current (m		0.00	
Battery Capacity (mA Hours)			0.00
Battery Life (Hours)			0.00

Name	Frequency (M	Capacitive Lo	DC Load (mA)
final_data_0_OBUF	12.49	35000.00	0.00
final_data_1_OBUF	12.49	35000.00	0.00
final_data_2_OBUF	12.49	35000.00	0.00
final_data_3_OBUF	12.49	35000.00	0.00
final_data_4_OBUF	12.49	35000.00	0.00
final_data_5_OBUF	12.49	35000.00	0.00
final_data_6_OBUF	12.49	35000.00	0.00
final_data_7_OBUF	12.49	35000.00	0.00
mac_cha_0_OBUF	12.49	35000.00	0.00
mac_cha_10_OBUF	12.49	35000.00	0.00
mac_cha_11_OBUF	12.49	35000.00	0.00
mac_cha_12_OBUF	12.49	35000.00	0.00
mac_cha_13_OBUF	12.49	35000.00	0.00
mac_cha_14_OBUF	12.49	35000.00	0.00
mac_cha_15_OBUF	12.49	35000.00	0.00
mac_cha_16_OBUF	12.49	35000.00	0.00
mac_cha_17_OBUF	12.49	35000.00	0.00
mac_cha_18_OBUF	12.49	35000.00	0.00
mac_cha_19_OBUF	12.49	35000.00	0.00
mac_cha_1_OBUF	12.49	35000.00	0.00
mac_cha_20_OBUF	12.49	35000.00	0.00
mac_cha_2_OBUF	12.49	35000.00	0.00
mac_cha_3_OBUF	12.49	35000.00	0.00
mac_cha_4_OBUF	12.49	35000.00	0.00
mac_cha_5_OBUF	12.49	35000.00	0.00
mac_cha_6_OBUF	12.49	35000.00	0.00
mac_cha_7_OBUF	12.49	35000.00	0.00
mac_cha_8_OBUF	12.49	35000.00	0.00
mac_cha_9_OBUF	12.49	35000.00	0.00

```
stepping level for this device is '2'. Additional information on "stepping level" is available at
support.xilinx.com.
```

FPGA editor

- The FPGA Editor is a graphical application that displays
  - Device resources
  - Precise layout of the chosen device
- The FPGA Editor is commonly used to
  - View device resources
  - Make minor modifications
    - Done late in the design cycle
    - Does not require re-implementation of the design
    - Changes are NOT back-annotated to the source files
  - Insert probes
  - Make short-term functional changes for in-circuit verification



	Name	Site	Type	
1	almost_empty[	SLICE_X20Y1	SLICE	1
2	almost_empty[	SLICE_X16Y1	SLICE	1
3	almost_empty[	SLICE_X12Y1	SLICE	1
4	almost_empty[	SLICE_X16Y1	SLICE	1
5	cha_fifo_inst/fi	SLICE_X20Y1	SLICE	8
6	cha_fifo_inst/fi	SLICE_X23Y2	SLICE	8
7	cha_fifo_inst/fi	SLICE_X23Y2	SLICE	8
8	cha_fifo_inst/fi	SLICE_X23Y2	SLICE	8
9	cha_fifo_inst/fi	SLICE_X23Y2	SLICE	8
10	cha_fifo_inst/fi	SLICE_X23Y2	SLICE	8
11	cha_fifo_inst/fi	SLICE_X23Y2	SLICE	4
12	cha_fifo_inst/fi	SLICE_X26Y2	SLICE	5
13	cha_fifo_inst/fi	SLICE_X25Y2	SLICE	5
14	cha_fifo_inst/fi	SLICE_X22Y2	SLICE	7
15	cha_fifo_inst/fi	SLICE_X23Y3	SLICE	7
16	cha_fifo_inst/fi	SLICE_X22Y2	SLICE	7

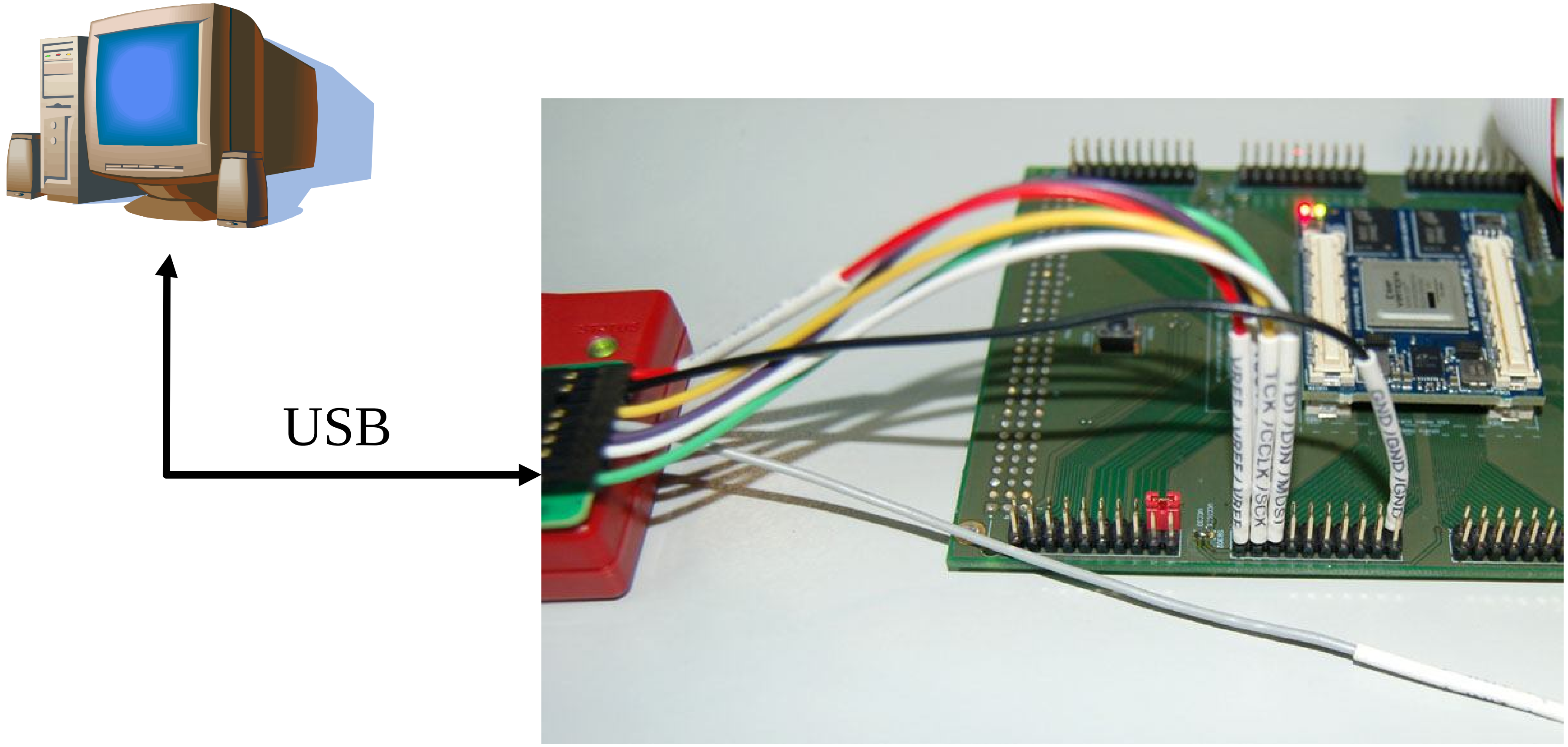
```
Building chip graphics...
Loading speed info...
Hint: change the visibility of more than one layer at a time by clicking on the apply button, clicking two or more layer buttons, and then
clicking on the apply button again.
```

For Help, press F1 | xc2v250-5fg256 | No Logic Changes

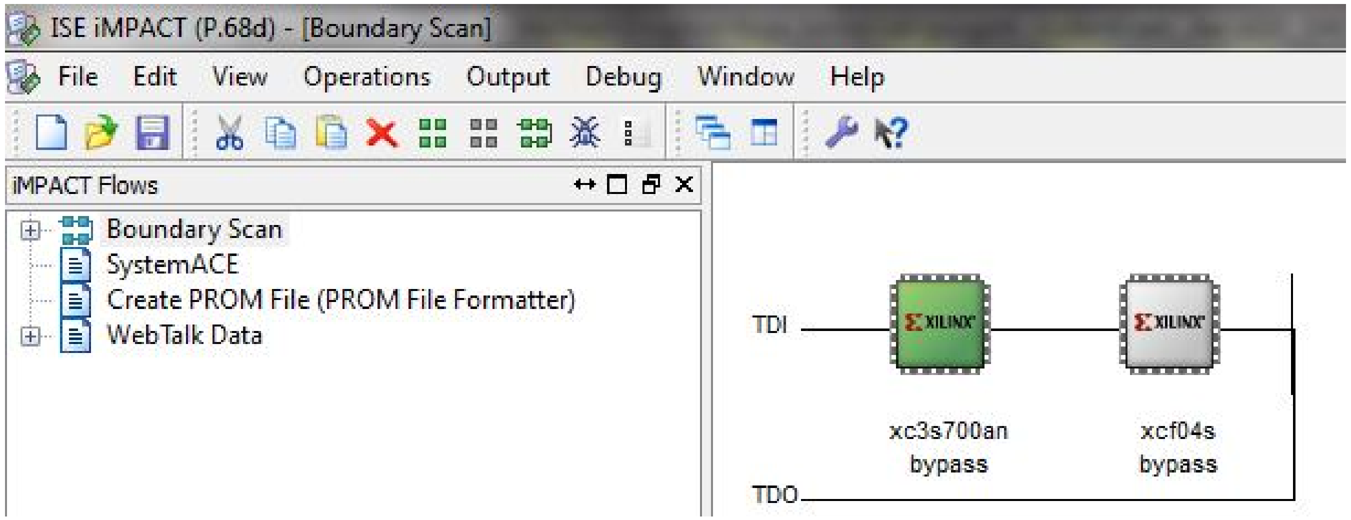
Xilinx programming cable

It allows to:

- program the FPGA
- debug its behavior by spying internal signals



ISE Impact



Impact allows to access via JTAG the devices on the chain, in this case one FPGA + one PROM.
Impact allows to:

- configure the FPGA
- program and readback the PROM
- play with the standard JTAG state machine in case of problems

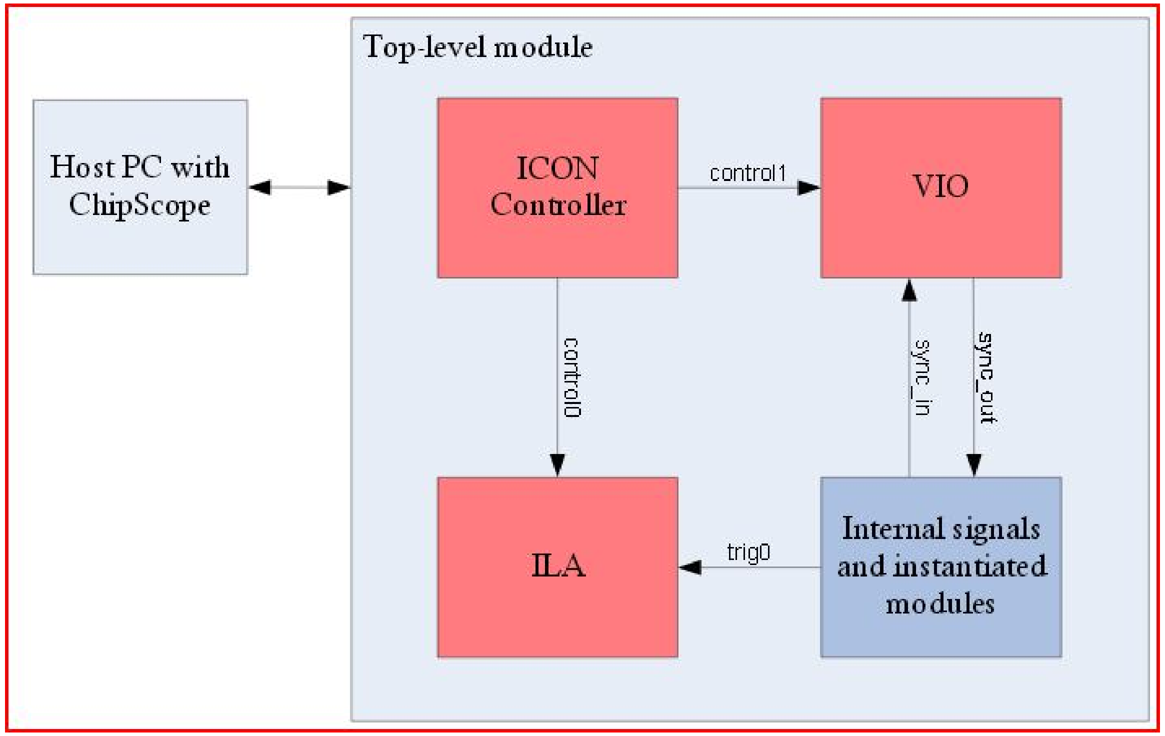
Chipscope PRO

ChipScope is an embedded, software based, logic analyzer, with 3 main blocks:

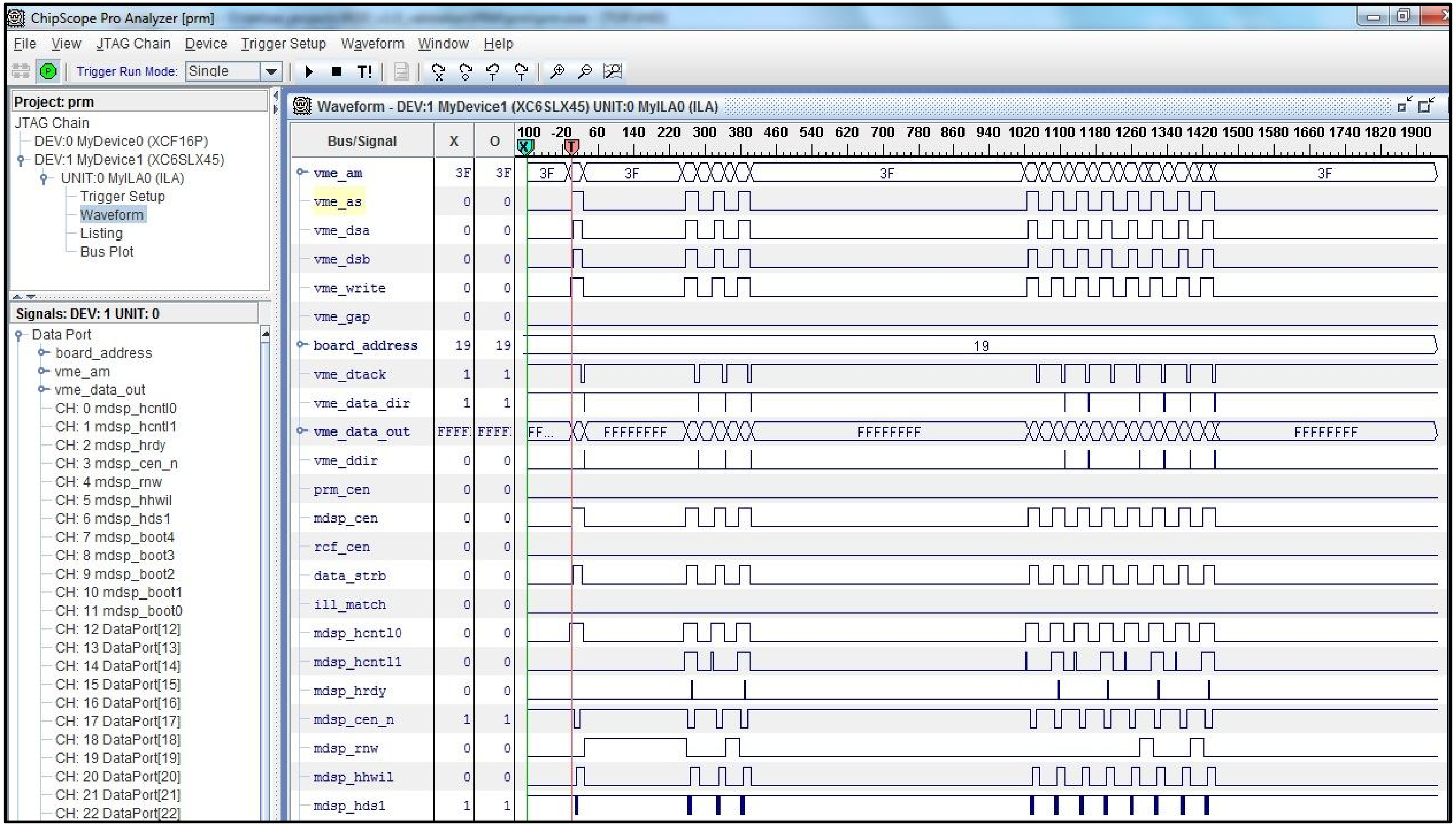
• **ICON** (Integrated CONtroller): A controller module that provides communication between the ChipScope host PC and ChipScope modules in the design (such as VIO and ILA).

• **VIO** (Virtual Input/Output): A module that can monitor and drive signals in your design in real-time. You can think of them as virtual push-buttons (for input) and LEDs (for output). These can be used for debugging purposes, or they can incorporated into your design as a permanent I/O interface.

• **ILA** (Integrated Logic Analyzer): A module that lets you view and trigger on signals in your hardware design. Think of it as a digital oscilloscope (like ModelSim's waveform viewer) that you can place in your design to aid in debugging.



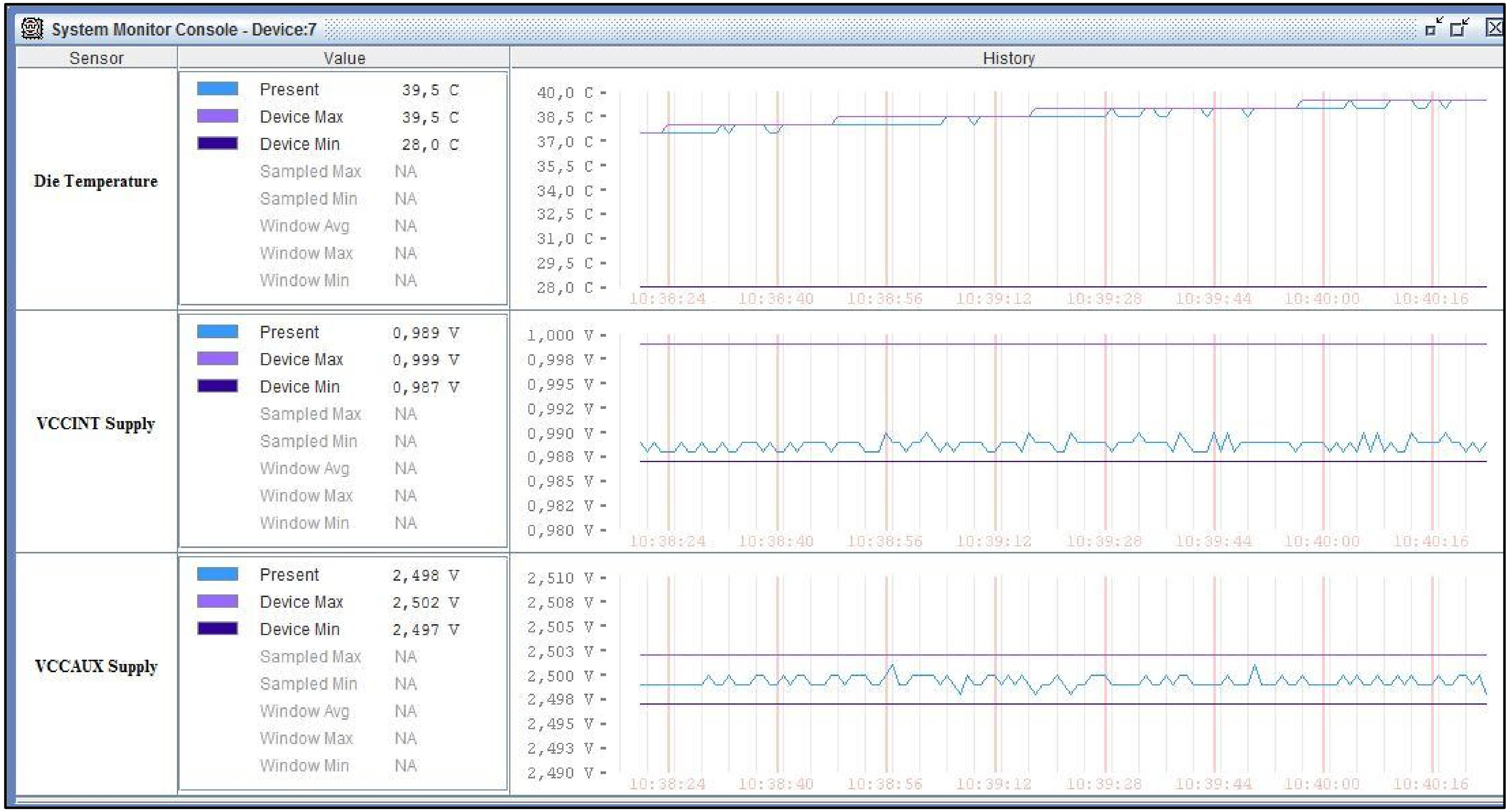
Chipscope PRO



Bus/Signal	X	O
vme_am	3F	3F
vme_as	0	0
vme_dsa	0	0
vme_dsb	0	0
vme_write	0	0
vme_gap	0	0
board_address	19	19
vme_dtack	1	1
vme_data_dir	1	1
vme_data_out	FFFF	FFFF
vme_ddir	0	0
prm_cen	0	0
mdsp_cen	0	0
rcf_cen	0	0
data_strb	0	0
ill_match	0	0
mdsp_hcntl0	0	0
mdsp_hcntl1	0	0
mdsp_hrdy	0	0
mdsp_cen_n	1	1
mdsp_rnw	0	0
mdsp_hhwil	0	0
mdsp_hds1	1	1

Using Chipscope to debug the behavior of a FPGA interfacing the VME bus

Chipscope PRO – system monitor



System Monitor Console - Device:7

Sensor	Value		
Die Temperature	🟦	Present	39,5 C
	🟪	Device Max	39,5 C
	■	Device Min	28,0 C
		Sampled Max	NA
		Sampled Min	NA
		Window Avg	NA
		Window Max	NA
		Window Min	NA
VCCINT Supply	🟦	Present	0,989 V
	🟪	Device Max	0,999 V
	■	Device Min	0,987 V
		Sampled Max	NA
		Sampled Min	NA
		Window Avg	NA
		Window Max	NA
		Window Min	NA
VCCAUX Supply	🟦	Present	2,498 V
	🟪	Device Max	2,502 V
	■	Device Min	2,497 V
		Sampled Max	NA
		Sampled Min	NA
		Window Avg	NA
		Window Max	NA
		Window Min	NA

ISIM – Chipscope interaction

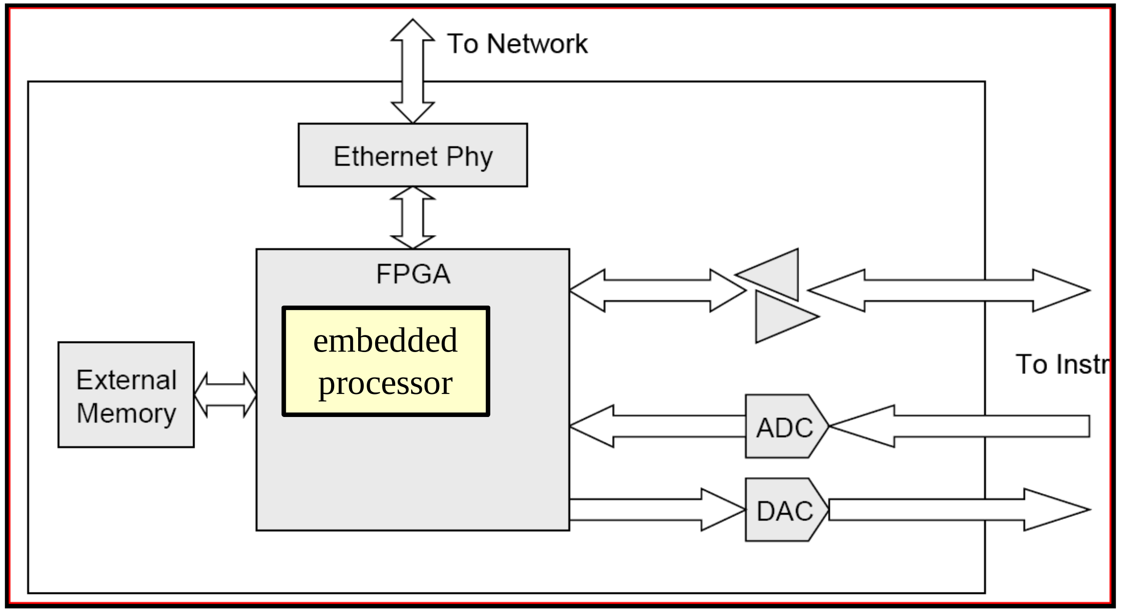
Real life is often different from what you see in simulation. What to do if simulation works fine, while live debug shows problems ?

One trick could be the following:
- spy with Chipscope the I/O signals of the faulty module,
- run ISIM using as a stimulus the inputs taken with Chipscope
- compare ISIM outputs with Chipscope outputs.

Usually a logical problem is not revealed by the sets of stimuli used in simulation, while it is immediately spotted in real life.

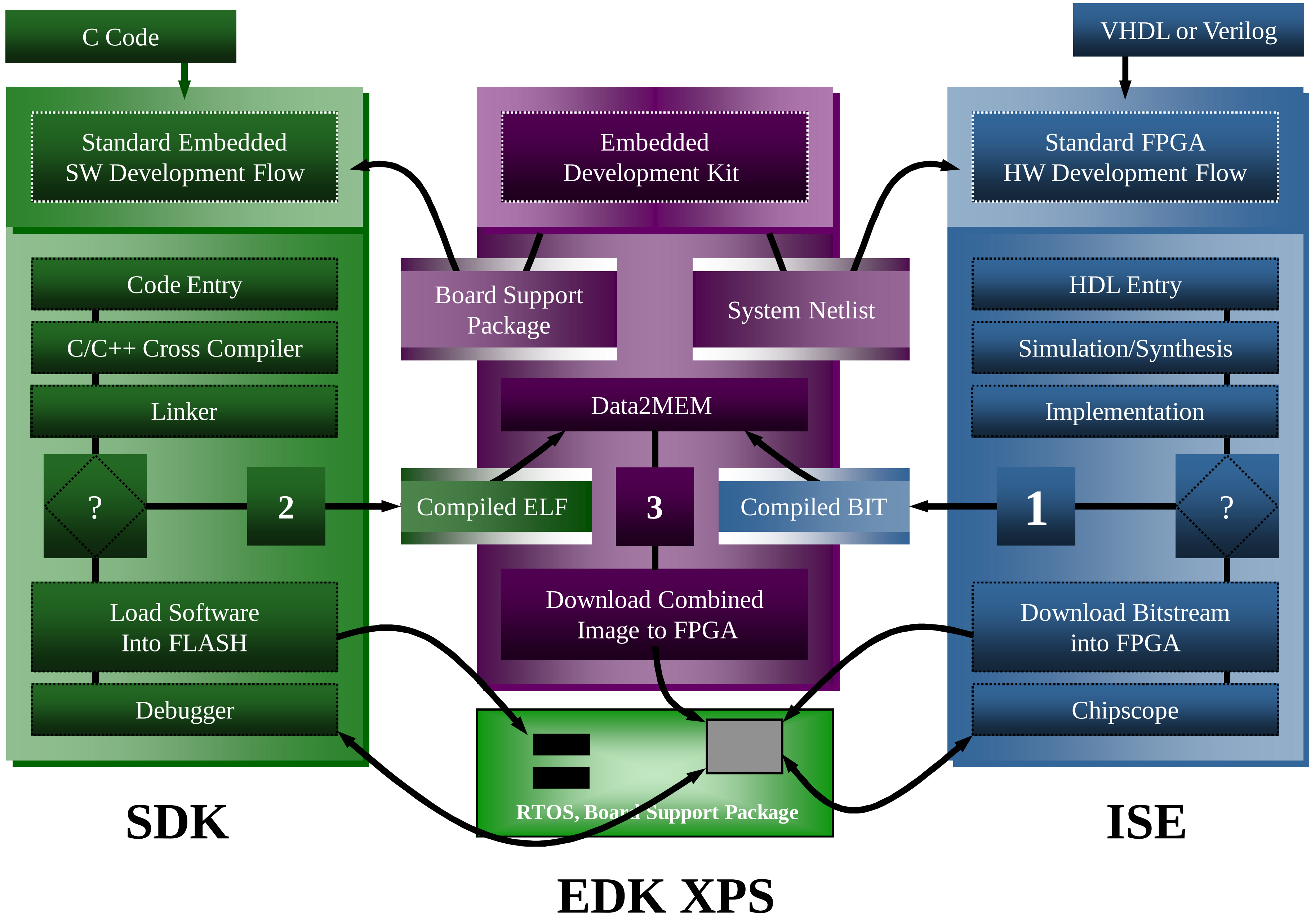
Embedded processors

Having an embedded processor in the FPGA can be convenient:
• the FPGA can handle all the high-throughput real-time tasks,
• the embedded processor can handle the common interfaces, like Ethernet, DDR2, UART, SPI, …

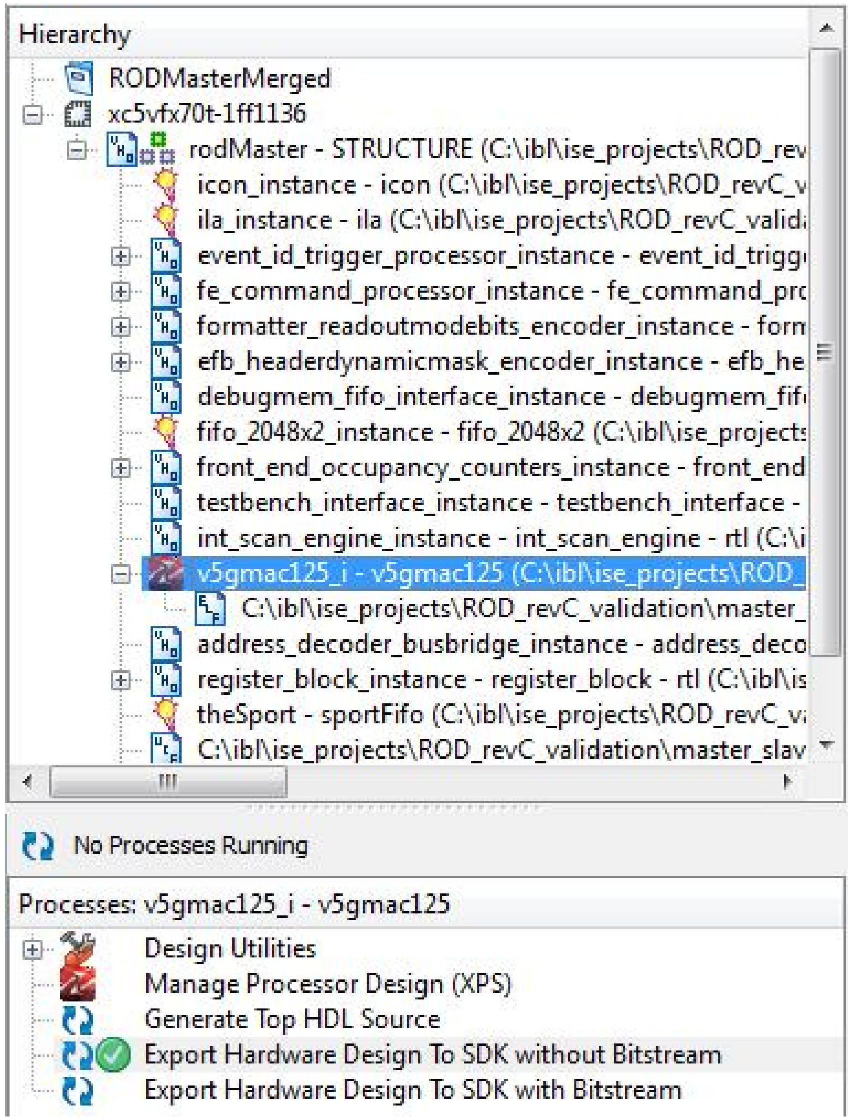


In this way the FPGA design flow changes a bit, providing some work to do also to SW designers

Embedded Development Tool Flow Overview



Embedded Development Tools



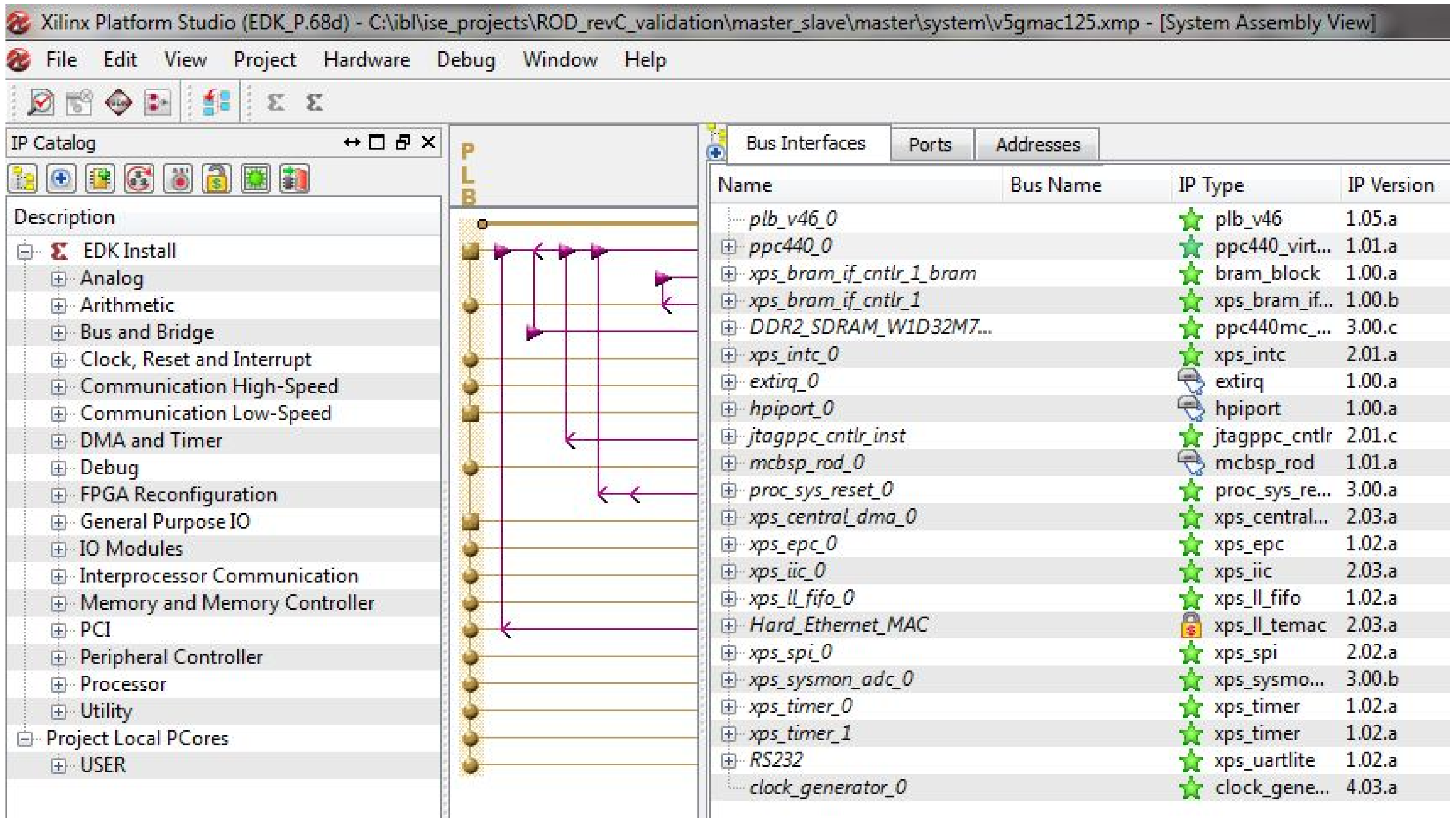
```
Hierarchy
  RODMasterMerged
  xc5vfx70t-1ff1136
    rodMaster - STRUCTURE (C:\ibl\ise_projects\ROD_rev
      icon_instance - icon (C:\ibl\ise_projects\ROD_revC_v
      ila_instance - ila (C:\ibl\ise_projects\ROD_revC_valid
      event_id_trigger_processor_instance - event_id_trigg
      fe_command_processor_instance - fe_command_prc
      formatter_readoutmodebits_encoder_instance - form
      efb_headerdynamicmask_encoder_instance - efb_he
      debugmem_fifo_interface_instance - debugmem_fif
      fifo_2048x2_instance - fifo_2048x2 (C:\ibl\ise_projects
      front_end_occupancy_counters_instance - front_end
      testbench_interface_instance - testbench_interface -
      int_scan_engine_instance - int_scan_engine - rtl (C:\i
      v5gmac125_i - v5gmac125 (C:\ibl\ise_projects\ROD_
        C:\ibl\ise_projects\ROD_revC_validation\master_
      address_decoder_busbridge_instance - address_deco
      register_block_instance - register_block - rtl (C:\ibl\is
      theSport - sportFifo (C:\ibl\ise_projects\ROD_revC_v
      C:\ibl\ise_projects\ROD_revC_validation\master_slav

No Processes Running

Processes: v5gmac125_i - v5gmac125
  Design Utilities
  Manage Processor Design (XPS)
  Generate Top HDL Source
  Export Hardware Design To SDK without Bitstream
  Export Hardware Design To SDK with Bitstream
```

From ISE and using the Core generator, it is possible to insert a soft or hard processor to the design hierarchy.
This lead to the use of 2 other tools.

➡ EDK XPS

➡ SDK

EDK XPS



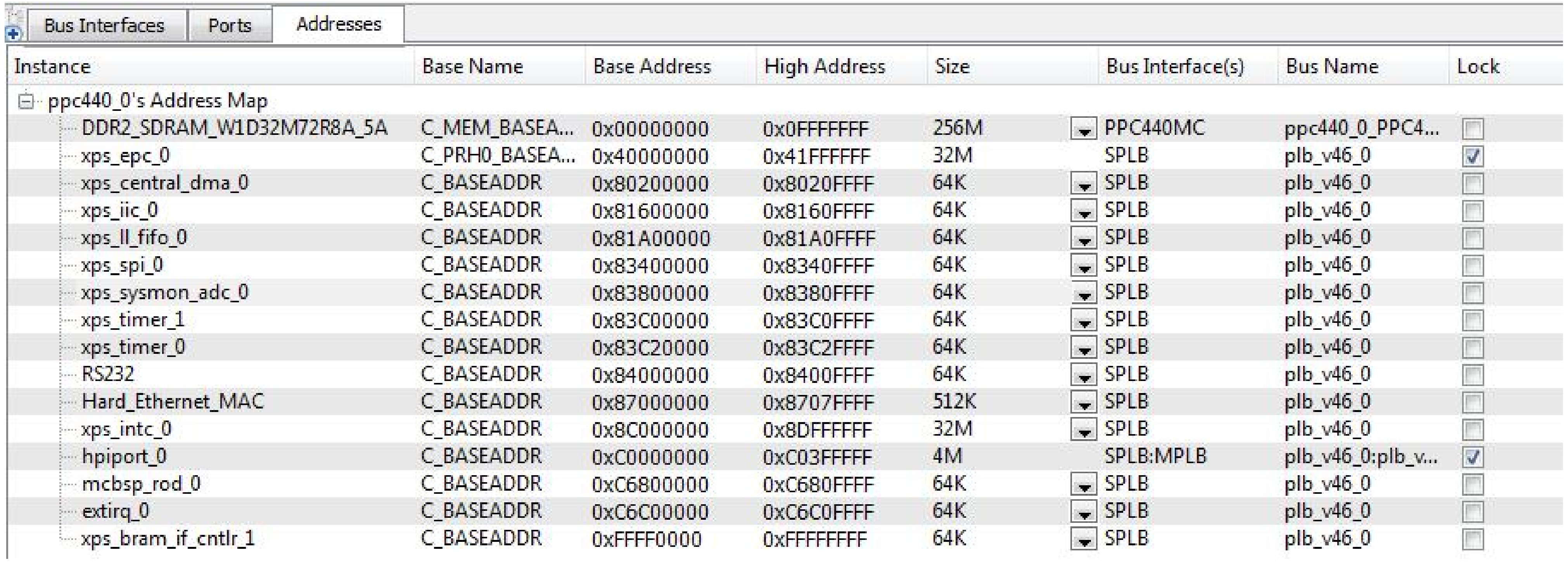
Xilinx Platform Studio (EDK_P.68d) - C:\ibl\ise_projects\ROD_revC_validation\master_slave\master\system\v5gmac125.xmp - [System Assembly View]

Description

- EDK Install
  - Analog
  - Arithmetic
  - Bus and Bridge
  - Clock, Reset and Interrupt
  - Communication High-Speed
  - Communication Low-Speed
  - DMA and Timer
  - Debug
  - FPGA Reconfiguration
  - General Purpose IO
  - IO Modules
  - Interprocessor Communication
  - Memory and Memory Controller
  - PCI
  - Peripheral Controller
  - Processor
  - Utility
- Project Local PCores
  - USER

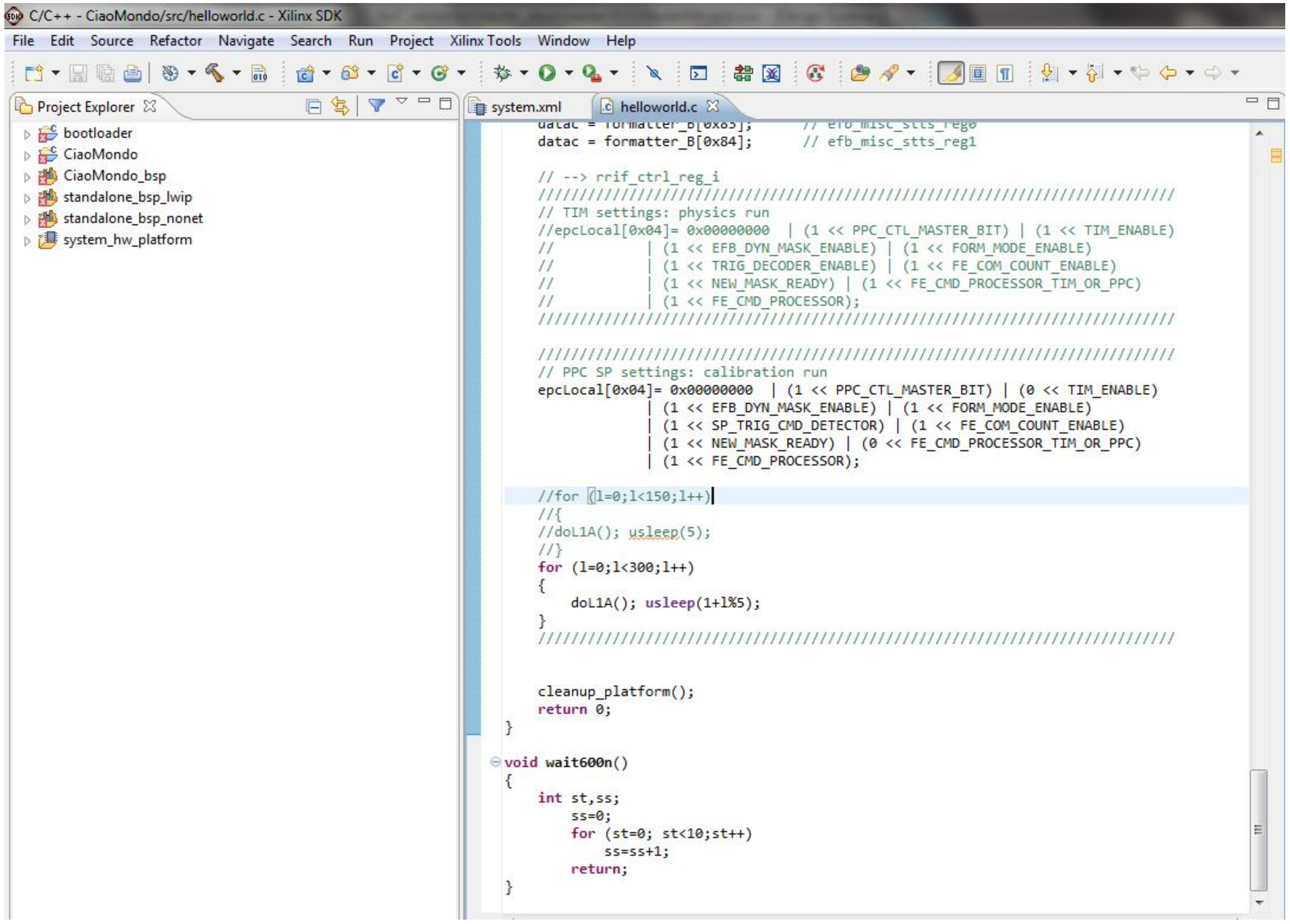
Name	Bus Name	IP Type	IP Version
plb_v46_0		plb_v46	1.05.a
ppc440_0		ppc440_virt...	1.01.a
xps_bram_if_cntlr_1_bram		bram_block	1.00.a
xps_bram_if_cntlr_1		xps_bram_if...	1.00.b
DDR2_SDRAM_W1D32M7...		ppc440mc_...	3.00.c
xps_intc_0		xps_intc	2.01.a
extirq_0		extirq	1.00.a
hpiport_0		hpiport	1.00.a
jtagppc_cntlr_inst		jtagppc_cntlr	2.01.c
mcbsp_rod_0		mcbsp_rod	1.01.a
proc_sys_reset_0		proc_sys_re...	3.00.a
xps_central_dma_0		xps_central...	2.03.a
xps_epc_0		xps_epc	1.02.a
xps_iic_0		xps_iic	2.03.a
xps_ll_fifo_0		xps_ll_fifo	1.02.a
Hard_Ethernet_MAC		xps_ll_temac	2.03.a
xps_spi_0		xps_spi	2.02.a
xps_sysmon_adc_0		xps_sysmo...	3.00.b
xps_timer_0		xps_timer	1.02.a
xps_timer_1		xps_timer	1.02.a
RS232		xps_uartlite	1.02.a
clock_generator_0		clock_gene...	4.03.a

- XPS provides an integrated environment for creating software and hardware specification flows for embedded processor systems based on MicroBlaze™ and PowerPC® processors.
- XPS offers customization of tool flow configuration options and provides a graphical system editor for connection of processors, peripherals, and buses.



Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
ppc440_0's Address Map							
DDR2_SDRAM_W1D32M72R8A_5A	C_MEM_BASEA...	0x00000000	0x0FFFFFFF	256M	PPC440MC	ppc440_0_PPC4...	☐
xps_epc_0	C_PRH0_BASEA...	0x40000000	0x41FFFFFF	32M	SPLB	plb_v46_0	✓
xps_central_dma_0	C_BASEADDR	0x80200000	0x8020FFFF	64K	SPLB	plb_v46_0	☐
xps_iic_0	C_BASEADDR	0x81600000	0x8160FFFF	64K	SPLB	plb_v46_0	☐
xps_ll_fifo_0	C_BASEADDR	0x81A00000	0x81A0FFFF	64K	SPLB	plb_v46_0	☐
xps_spi_0	C_BASEADDR	0x83400000	0x8340FFFF	64K	SPLB	plb_v46_0	☐
xps_sysmon_adc_0	C_BASEADDR	0x83800000	0x8380FFFF	64K	SPLB	plb_v46_0	☐
xps_timer_1	C_BASEADDR	0x83C00000	0x83C0FFFF	64K	SPLB	plb_v46_0	☐
xps_timer_0	C_BASEADDR	0x83C20000	0x83C2FFFF	64K	SPLB	plb_v46_0	☐
RS232	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	plb_v46_0	☐
Hard_Ethernet_MAC	C_BASEADDR	0x87000000	0x8707FFFF	512K	SPLB	plb_v46_0	☐
xps_intc_0	C_BASEADDR	0x8C000000	0x8DFFFFFF	32M	SPLB	plb_v46_0	☐
hpiport_0	C_BASEADDR	0xC0000000	0xC03FFFFF	4M	SPLB:MPLB	plb_v46_0:plb_v...	✓
mcbsp_rod_0	C_BASEADDR	0xC6800000	0xC680FFFF	64K	SPLB	plb_v46_0	☐
extirq_0	C_BASEADDR	0xC6C00000	0xC6C0FFFF	64K	SPLB	plb_v46_0	☐
xps_bram_if_cntlr_1	C_BASEADDR	0xFFFF0000	0xFFFFFFFF	64K	SPLB	plb_v46_0	☐

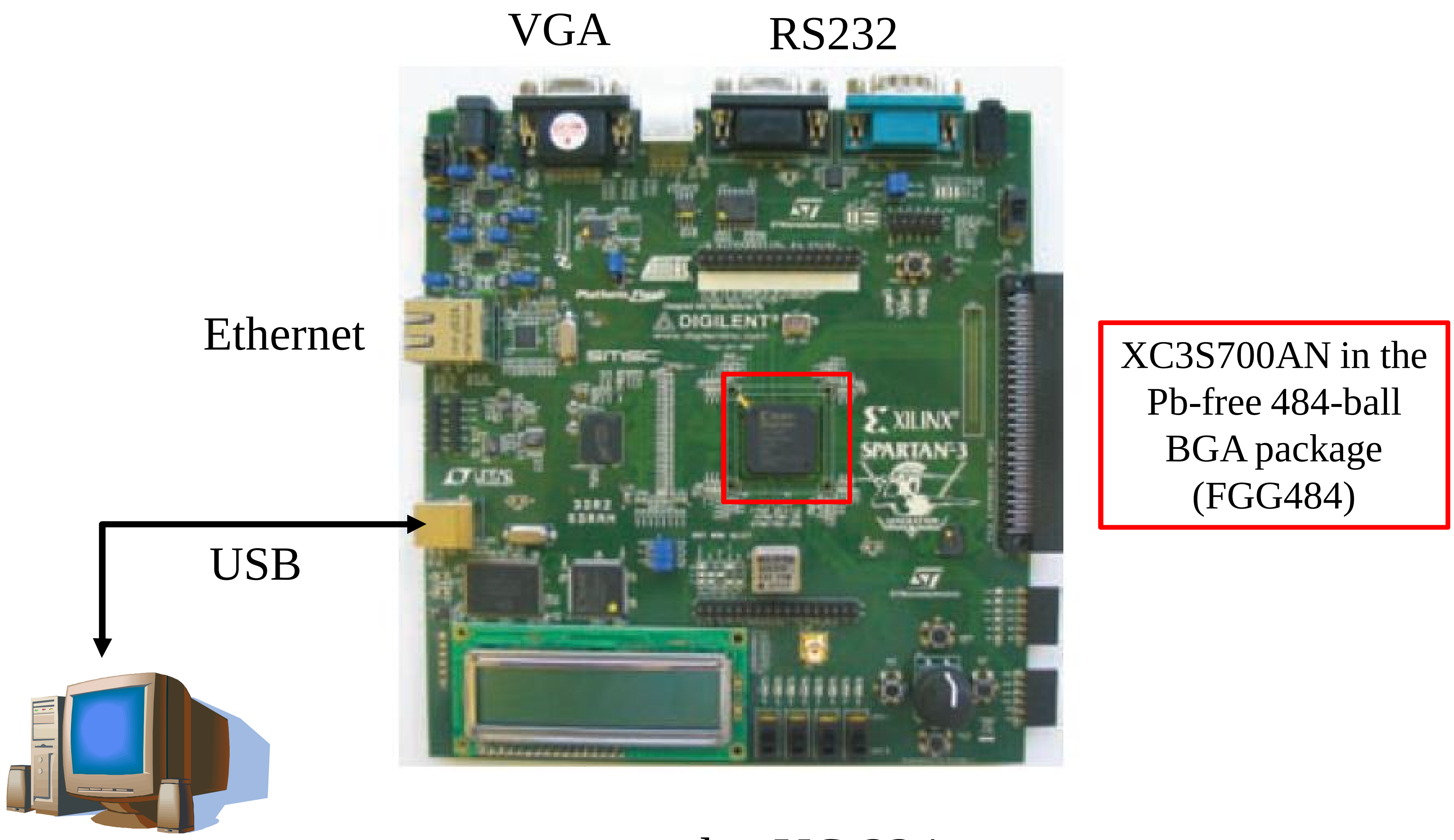
SDK



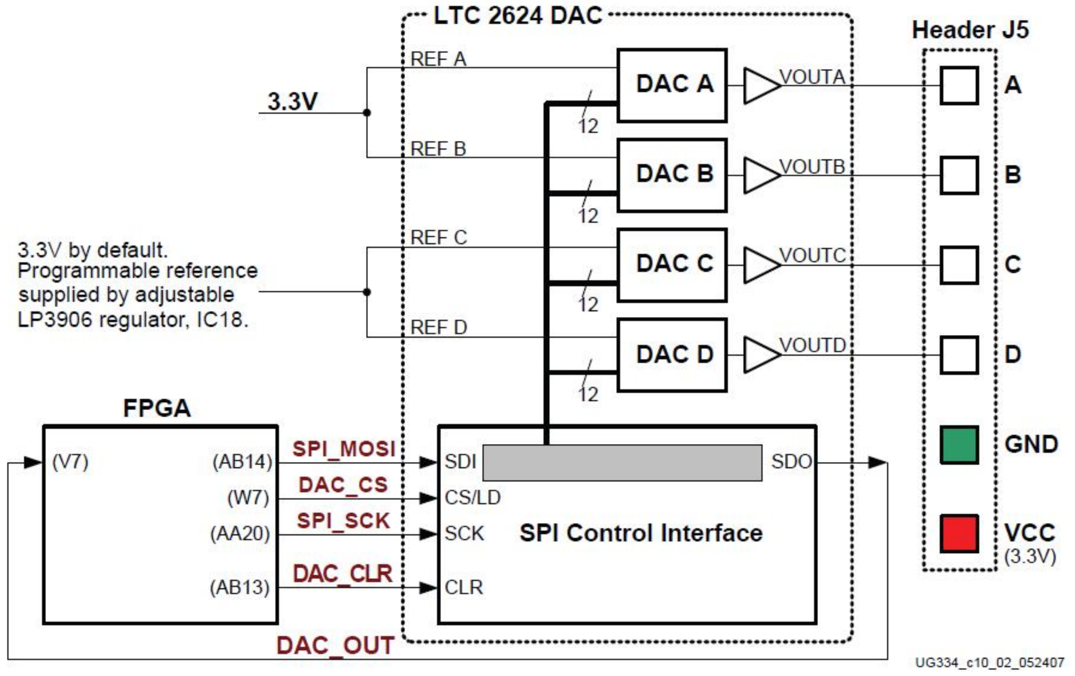
```c
    datac = formatter_B[0x83];      // efb_misc_stts_reg0
    datac = formatter_B[0x84];      // efb_misc_stts_reg1

    // --> rrif_ctrl_reg_i
    ////////////////////////////////////////////////////////////////////////////
    // TIM settings: physics run
    //epcLocal[0x04]= 0x00000000  | (1 << PPC_CTL_MASTER_BIT) | (1 << TIM_ENABLE)
    //              | (1 << EFB_DYN_MASK_ENABLE) | (1 << FORM_MODE_ENABLE)
    //              | (1 << TRIG_DECODER_ENABLE) | (1 << FE_COM_COUNT_ENABLE)
    //              | (1 << NEW_MASK_READY) | (1 << FE_CMD_PROCESSOR_TIM_OR_PPC)
    //              | (1 << FE_CMD_PROCESSOR);
    ////////////////////////////////////////////////////////////////////////////

    ////////////////////////////////////////////////////////////////////////////
    // PPC SP settings: calibration run
    epcLocal[0x04]= 0x00000000  | (1 << PPC_CTL_MASTER_BIT) | (0 << TIM_ENABLE)
                    | (1 << EFB_DYN_MASK_ENABLE) | (1 << FORM_MODE_ENABLE)
                    | (1 << SP_TRIG_CMD_DETECTOR) | (1 << FE_COM_COUNT_ENABLE)
                    | (1 << NEW_MASK_READY) | (0 << FE_CMD_PROCESSOR_TIM_OR_PPC)
                    | (1 << FE_CMD_PROCESSOR);

    //for (l=0;l<150;l++)
    //{
    //doL1A(); usleep(5);
    //}
    for (l=0;l<300;l++)
    {
        doL1A(); usleep(1+l%5);
    }
    ////////////////////////////////////////////////////////////////////////////


    cleanup_platform();
    return 0;
}

void wait600n()
{
    int st,ss;
        ss=0;
        for (st=0; st<10;st++)
            ss=ss+1;
        return;
}
```

The Xilinx Software Development Kit (**SDK**) is the recommended development environment for software application projects. SDK is based on the Eclipse open source standard.

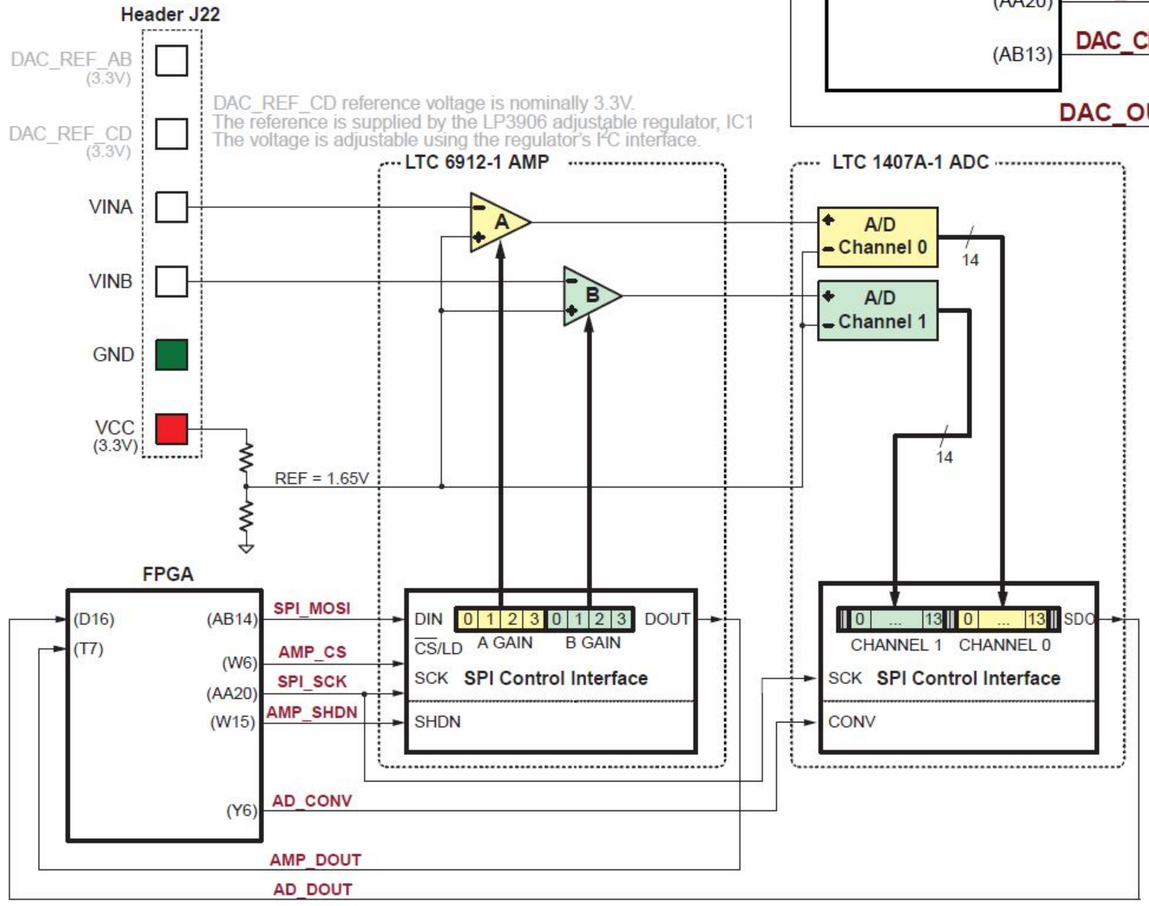
Spartan 3AN Starter Kit board



see also UG 334



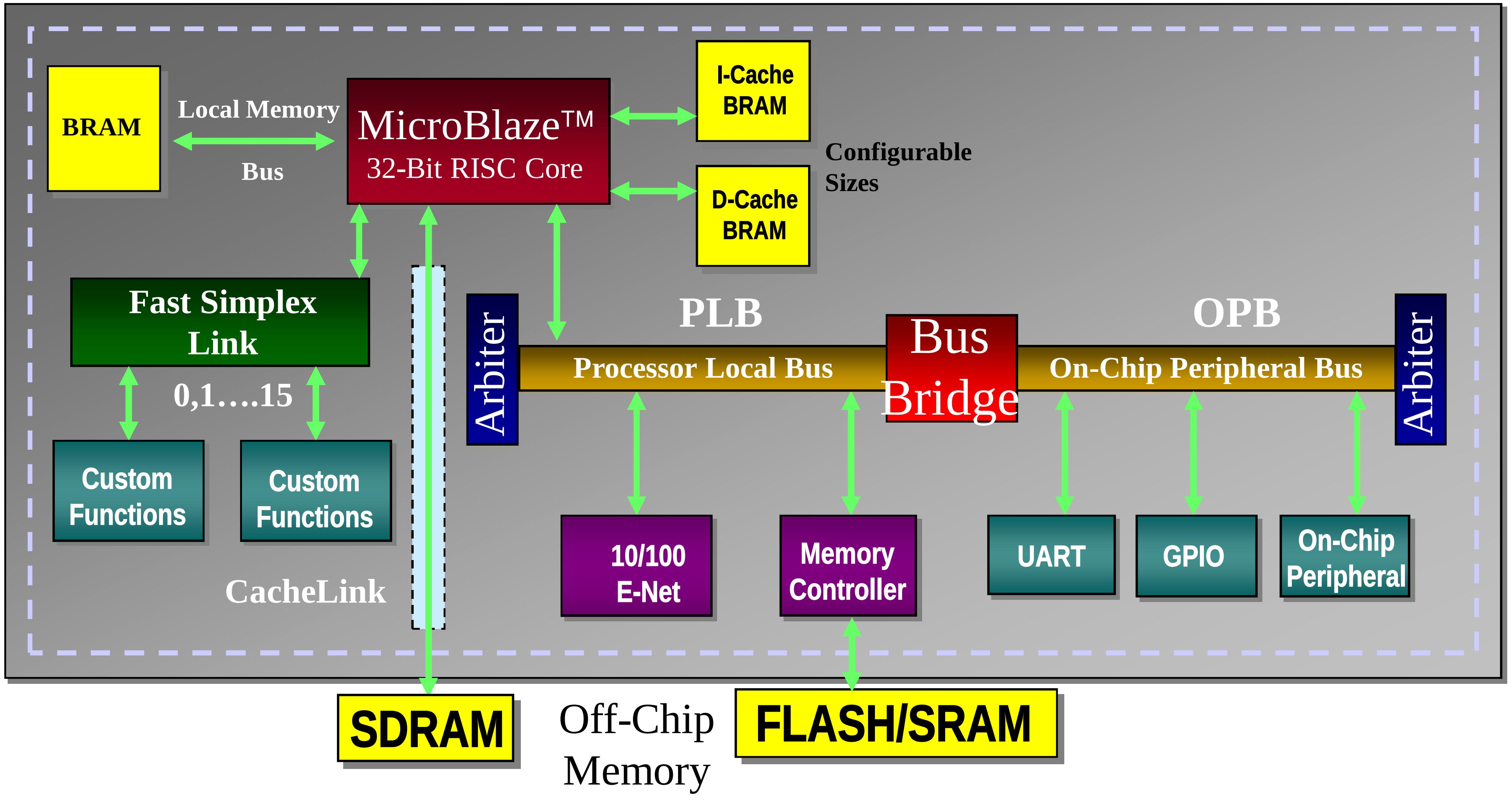
LTC 2624 DAC
Header J5
3.3V
REF A
REF B
REF C
REF D
DAC A
DAC B
DAC C
DAC D
12
VOUTA
VOUTB
VOUTC
VOUTD
A
B
C
D
GND
VCC
(3.3V)
3.3V by default.
Programmable reference
supplied by adjustable
LP3906 regulator, IC18.
FPGA
(V7)
(AB14)
(W7)
(AA20)
(AB13)
SPI_MOSI
DAC_CS
SPI_SCK
DAC_CLR
SDI
CS/LD
SCK
CLR
SDO
SPI Control Interface
DAC_OUT
UG334_c10_02_052407



Header J22
DAC_REF_AB
(3.3V)
DAC_REF_CD
(3.3V)
DAC_REF_CD reference voltage is nominally 3.3V.
The reference is supplied by the LP3906 adjustable regulator, IC1
The voltage is adjustable using the regulator's I²C interface.
VINA
VINB
GND
VCC
(3.3V)
REF = 1.65V
LTC 6912-1 AMP
A
B
LTC 1407A-1 ADC
A/D
Channel 0
A/D
Channel 1
14
FPGA
(D16)
(T7)
(AB14)
(W6)
(AA20)
(W15)
(Y6)
SPI_MOSI
AMP_CS
SPI_SCK
AMP_SHDN
AD_CONV
AMP_DOUT
AD_DOUT
DIN
CS/LD
SCK
SHDN
0 1 2 3 0 1 2 3
A GAIN
B GAIN
DOUT
SPI Control Interface
0 ... 13 0 ... 13
CHANNEL 1
CHANNEL 0
SDO
SCK
CONV
SPI Control Interface

Backup

MicroBlaze System


BRAM
Local Memory
Bus
MicroBlaze™
32-Bit RISC Core
I-Cache BRAM
D-Cache BRAM
Configurable Sizes
Fast Simplex Link
0,1….15
Custom Functions
Custom Functions
CacheLink
Arbiter
PLB
Processor Local Bus
Bus Bridge
OPB
On-Chip Peripheral Bus
Arbiter
10/100 E-Net
Memory Controller
UART
GPIO
On-Chip Peripheral
SDRAM
Off-Chip Memory
FLASH/SRAM