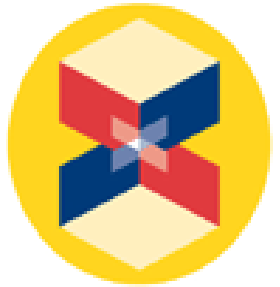
Digitronix Nepal
Innovation from Engineers...

FPGA Design with Matlab & Simulink

Course Prepared by
Digitronix Nepal
www.digitronixnepal.com
Email: digitronixnepali@gmail.com



XILINX
SYSTEM
GENERATOR™
For DSP

Sections and Lecturers of the Course:

Section 1. Installing Matlab/Simulink and Xilinx VIVADO/ISE Design Suit

Lecture 1 : Downloading and Intalling Matlab/Simulink and Xilinx ISE

Lecture 2 : Version Compatibility for Matlab/Simulink and ISE/VIVADO

Section 2. Introduction to Matlab/Simulink and HDL Coder/System Generator

Lecture 1 : Matlab and Simulink Introduction

Lecture 2 : HDL Coder and System Generator Introduction

Lecture 3 : Program Xilinx Zynq SoC Devices with Embedded Coder and HDL Coder

Lecture 4 : Program Xilinx FPGAs Using HDL Coder with Xilinx System Generator

Topic 2: Introduction to MATLAB/Simulink and System Generator

a. Matlab and Simulink(tools and methodology)
b. System Generator and System Generator Design flow with Basic Introduction of Methodology of System Generator
c. Program Xilinx FPGAs using HDL Coder
d. Program Xilinx FPGAs using HDL Coder with Xilinx System Generator
e. Program Xilinx Zynq SoC Devices with Embedded Coder and HDL Coder
f. Verify Xilinx FPGAs using HDL Verifier

MATLAB

1. MATLAB is a high-performance language for technical computing.
2. MATLAB integrates computation, visualization and programming in an easy-to-use einvironment where problems and solutions are expressed in familiar mathematical notation.
3. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows us to solve many technical computing problems, especially those with matrix and vector formulations in a fraction of the time.

MATLAB

The MATLAB system consists of five main parts:

1. The MATLAB language
   - This is a high-level matrix language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and throw-away programs, and “programming in the large” to create complete large and complex application programs.
2. The MATLAB working environment
   - This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in workspace and importing and exporting data.
3. Handle Graphics
   - This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation and presentation graphics. It also includes low-level commands that allows us to fully customize the appearance of graphics as well as to build complete GUI on MATLAB applications.

MATLAB

4. The MATLAB mathematical function library
   - This a vast collection of computational algorithms ranging from elementary functions like sum, sine and complex arithmetic, to more advance functions like matrix inverse, Bessel functions and Fast fourier transforms.

5. The MATLAB API
   - This is a library that allows us to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB, calling MATLAB as a computational engine, and for reading and writing MAT-files.

Simulink

1. Simulink is a simulation and model-based design environment for dynamic and embedded systems integrated with MATLAB.
2. It is basically a graphical block diagramming tool with customizable set of block libraries.
3. Simulink supports following functions:
   - System-level design
   - Simulation
   - Automatic code generation
   - Testing and verification of embedded systems

System Generator for DSP

1. The Xilinx System Generator for DSP is a plug-in to Simulink that enables designers to develop high-performance DSP systems for Xilinx FPGAs.
2. Designers can design and simulate a system using MATLAB, Simulink and Xilinx library of bit/cycle-true models.
3. The tool will then automatically generate synthesizable Hardware Description Language(HDL) code mapped to Xilinx pre-optimized algorithms. This HDL design can then be synthesized for implementation on Xilinx FPGAs and All Programmable SoCs.
4. Thus the designers can define system-level design at higher level and easily transform this source code into gate-level representation.

System Generator in windows after installation

goto: *Start menu →All Programs*

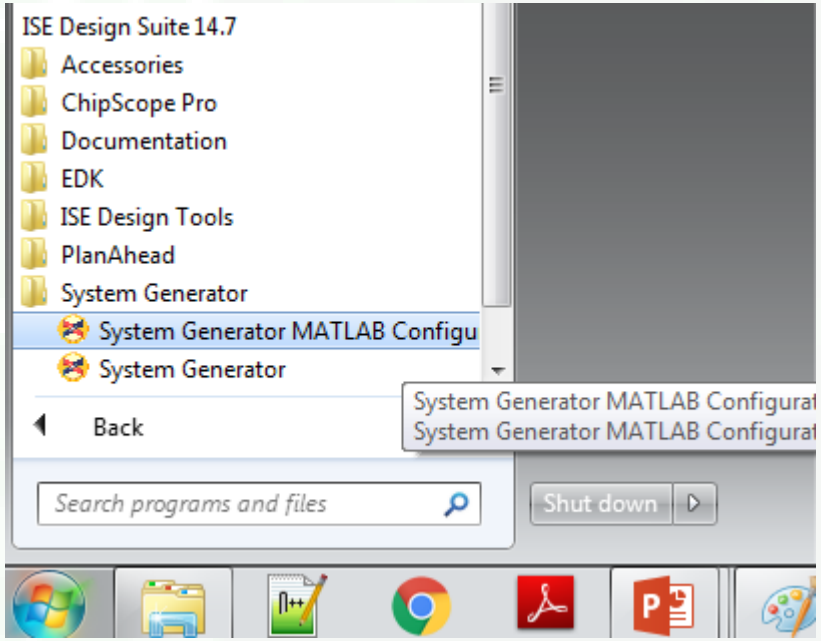


Figure: System Generator with ISE Design Suit
In Windows 7 O.S

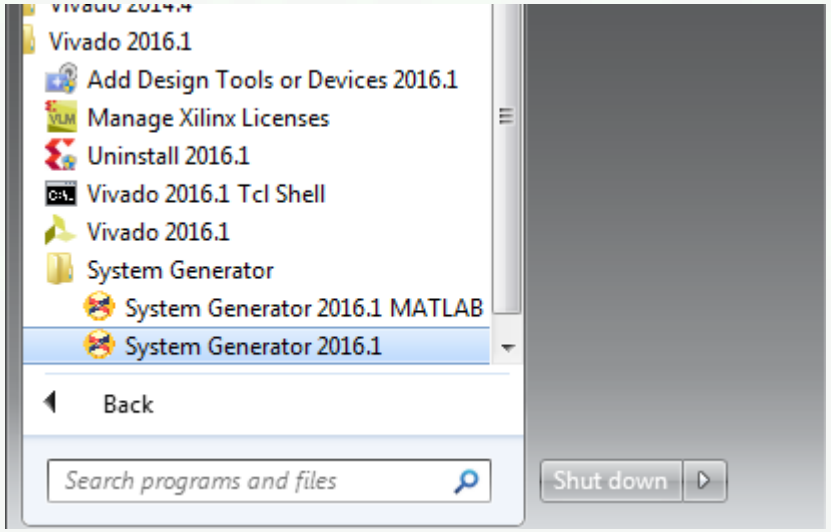
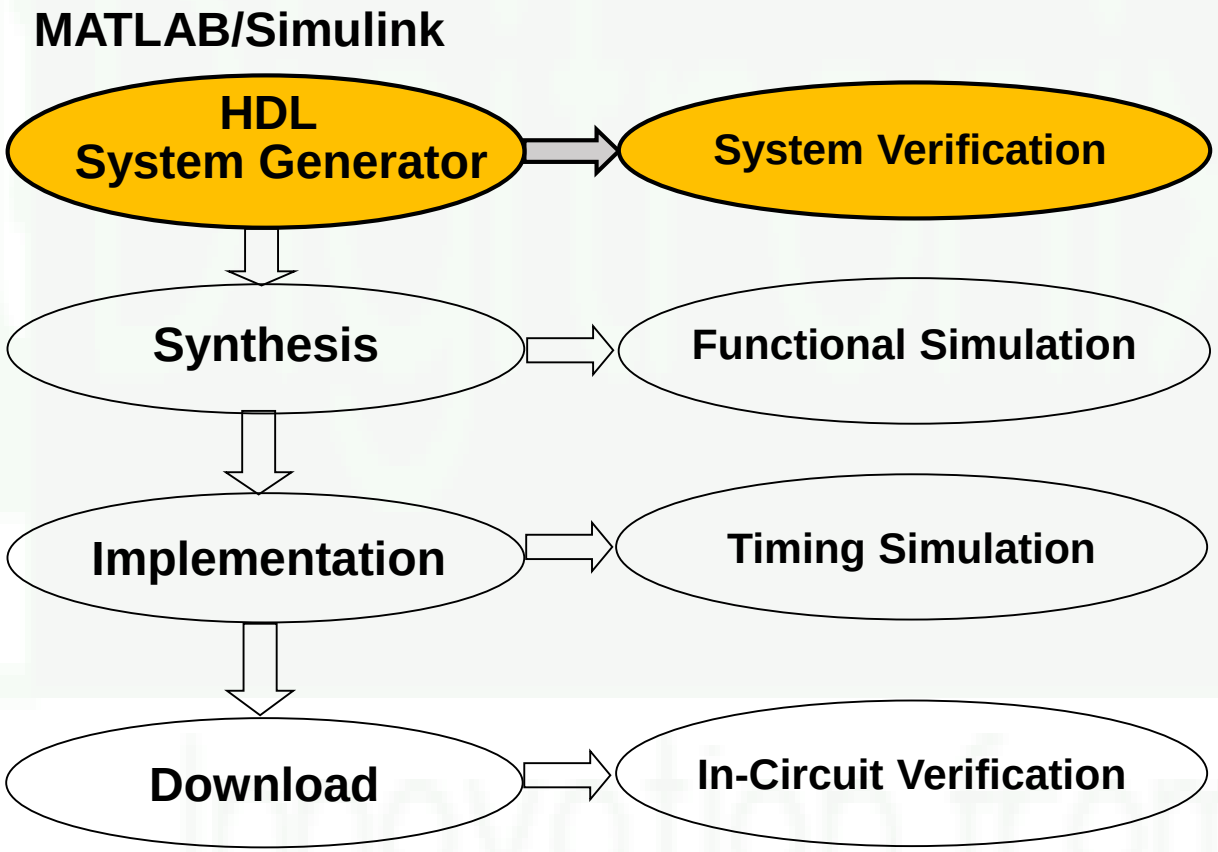
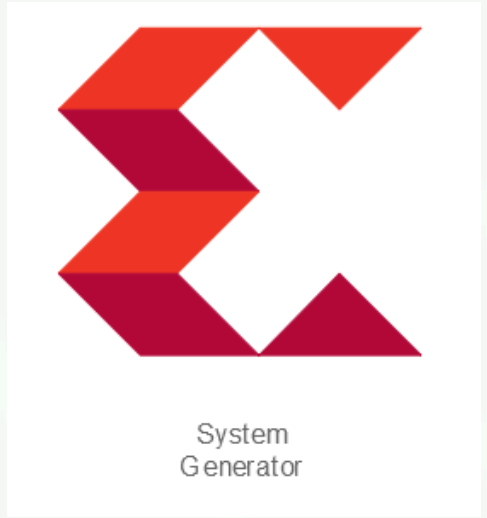


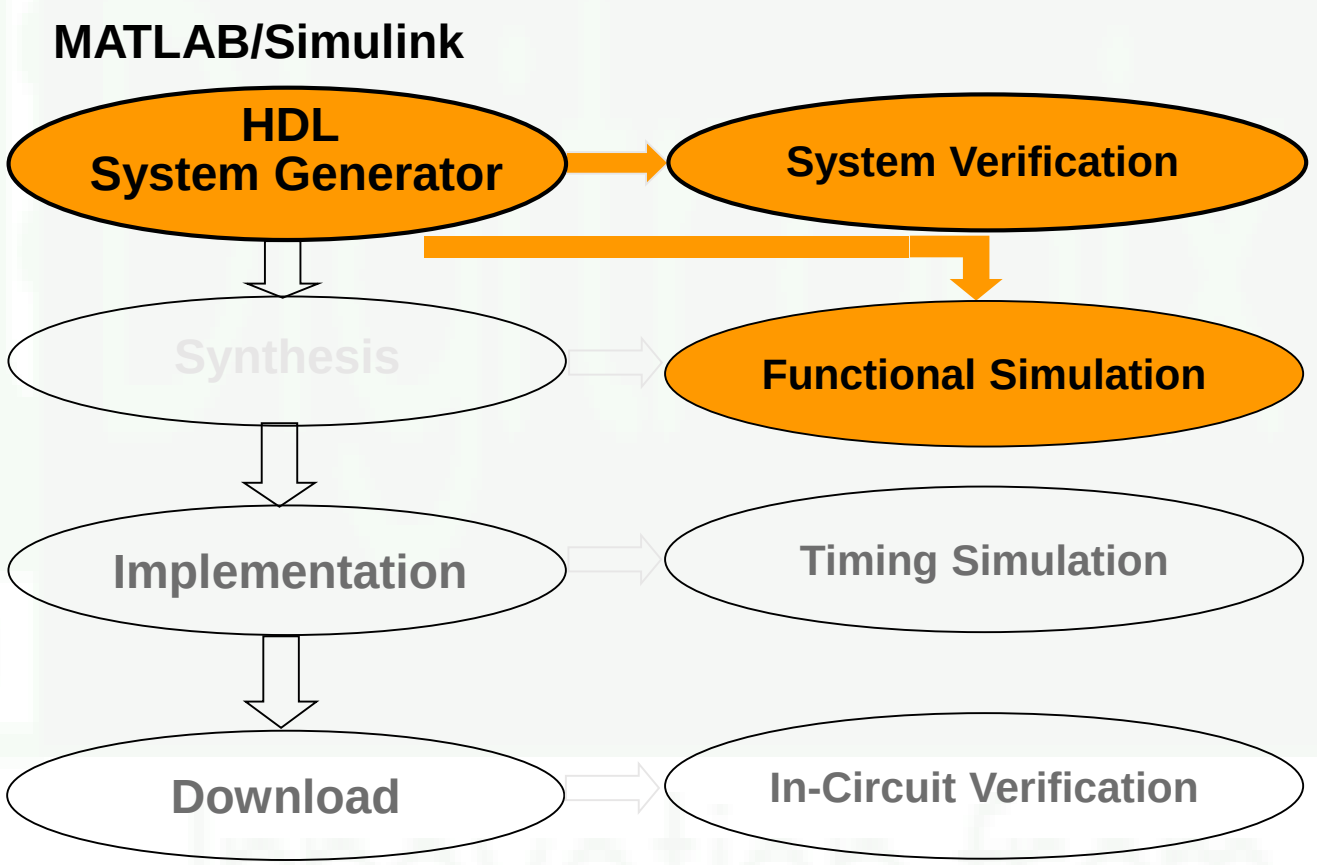
Figure: System Generator with VIVADO
Design Suit, In Windows 7 O.S

System Generator Based Design Flow

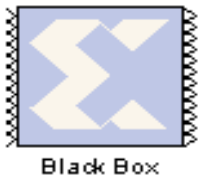




System Generator Based Design Flow

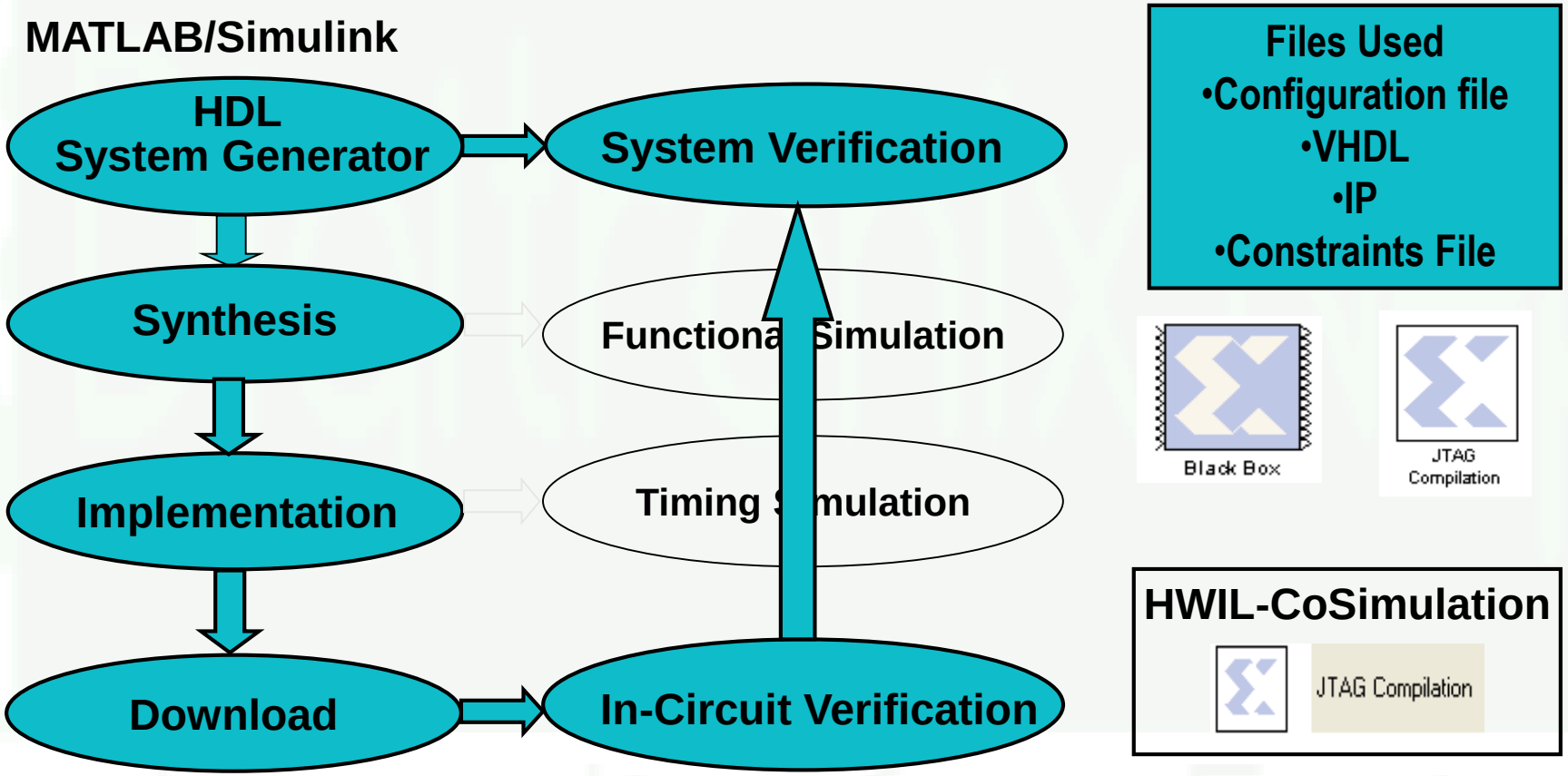


Files Used
- Configuration file
- VHDL
- IP
- Constraints File



HDL-CoSimulation

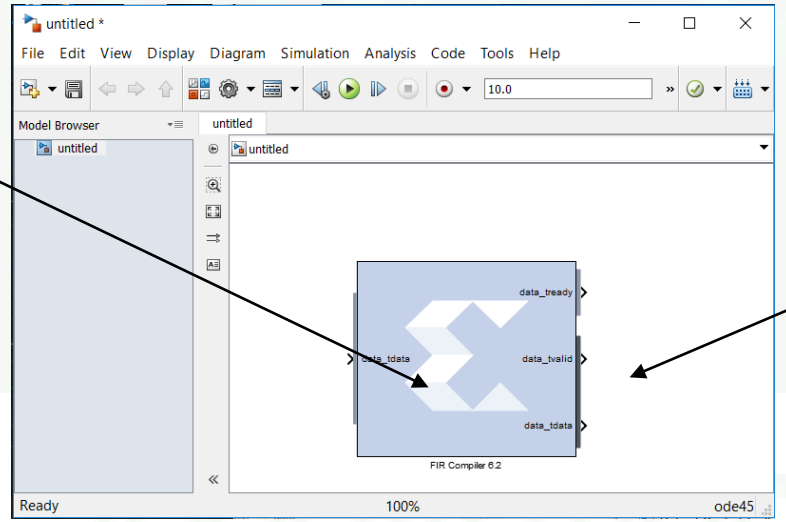
System Generator Based Design Flow

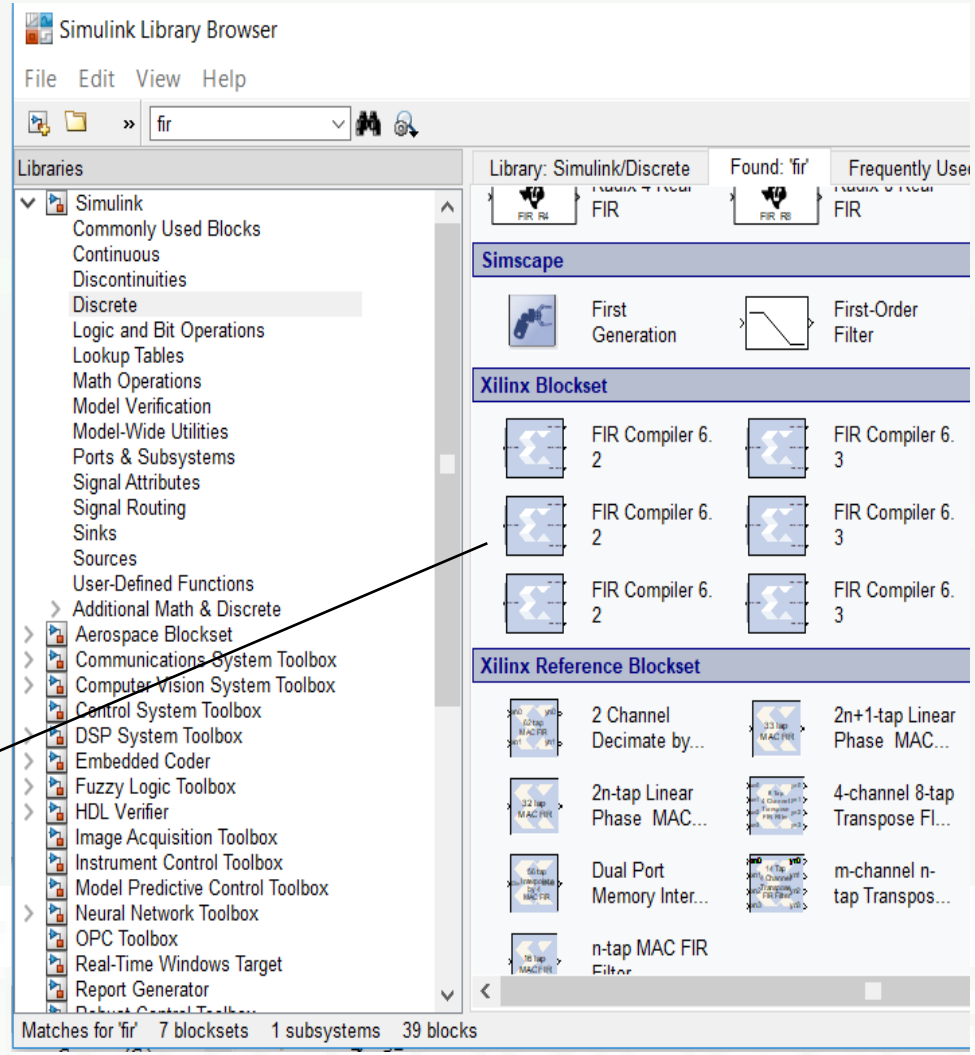


Creating a System Generator Design

- Build the design by dragging and dropping blocks from the Xilinx blockset onto your new sheet.
- Design Entry is similar to a circuit schematic editor.

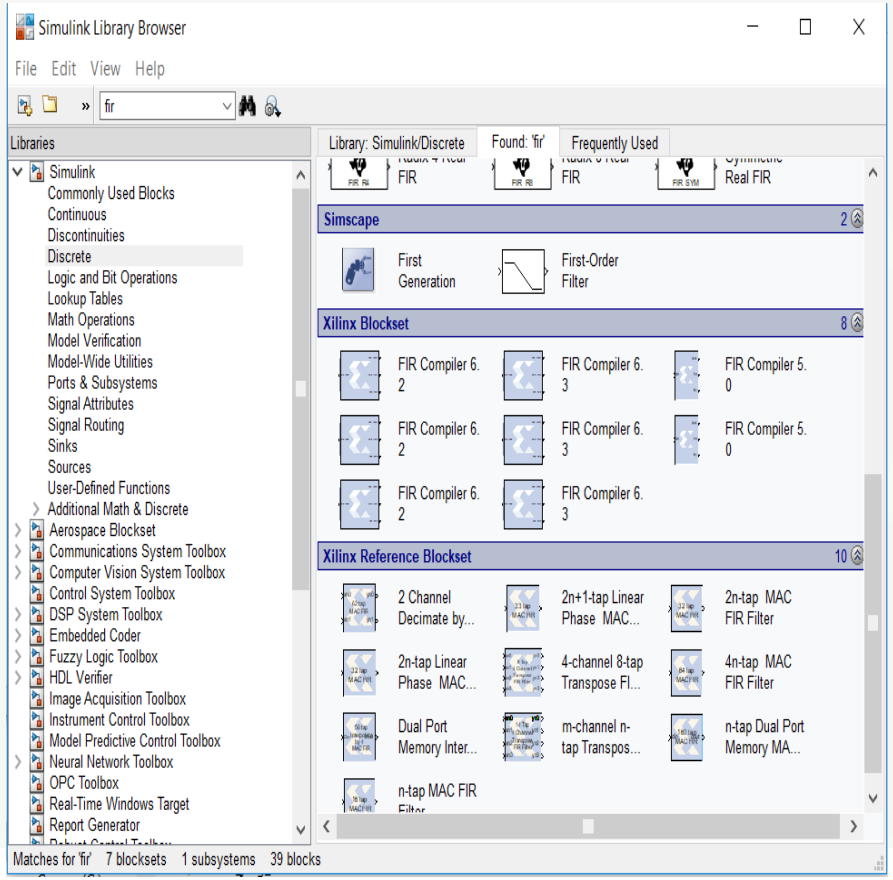
Connect up block by pulling the arrows on the sides of each block





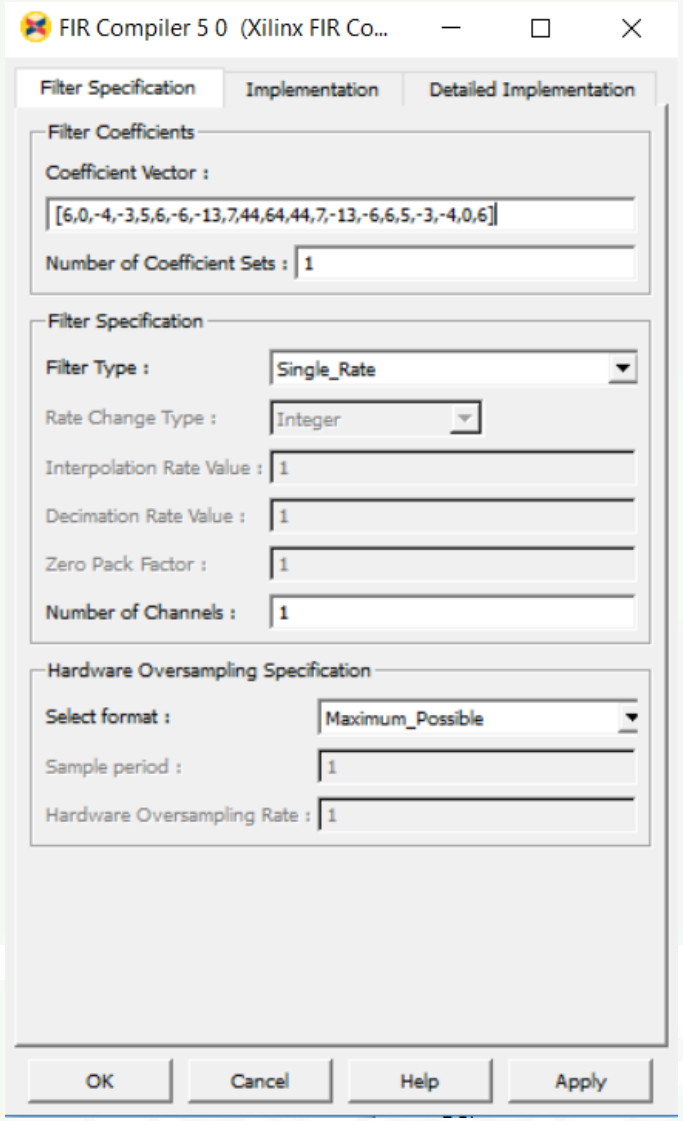
Finding Blocks

1. Use the Find feature to search ALL Simulink libraries
2. Xilinx blockset has nine major sections
   a. Basic elements
      - Counters, delays
   b. Communication
      - Error correction blocks
   c. Control Logic
      - MCode, Black Box
   d. Data Types
      - Convert, Slice
   e. DSP
      - FDATool, FFT, FIR
   f. Index
      - All Xilinx blocks – quick way to view all blocks
   g. Math
      - Multiply, accumulate, inverter
   h. Memory
      - Dual Port RAM, Single Port RAM
   i. Tools
      - ModelSim, Resource Estimator, WaveScope



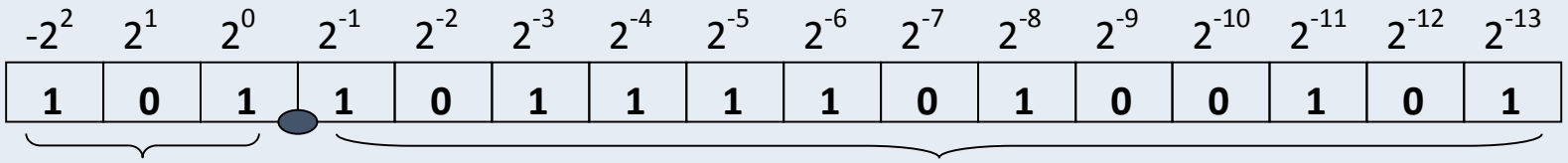
Configure Blocks

- **Double-click or go to Block Parameters to view a block's configurable parameters**
  - Arithmetic Type: Unsigned or twos complement
  - Implement with Xilinx Smart-IP Core (if possible)/ Generate Core
  - Latency: Specify the delay through the block
  - Overflow and Quantization: Users can saturate or wrap overflow. Truncate or Round Quantization
  - Override with Doubles: Simulation only
  - Precision: Full or the user can define the number of bits and where the decimal point is for the block
  - Sample Period: Can be inherent with a "-1" or must be an integer value
- **Note: While all parameters can be simulated, not all are realizable**



Number Format in System Generator

1. Simulink uses a “double” to represent numbers in a simulation. A double is a “64-bit twos complement floating point number”
   - Because the binary point can move, a double can represent any number between +/- 9.223×10^{18} with a resolution of 1.08×10^{-19} …a wide desirable range, but not efficient or realistic for FPGAs
2. Xilinx Blockset uses n-bit fixed point number (twos complement optional)



Format = **Sign_Width_Decimal point from the LSB**

Value = -2.261108…

Format = Fix_16_13

(***Sign:*** *Fix = Signed Value*

UFix = Unsigned value)

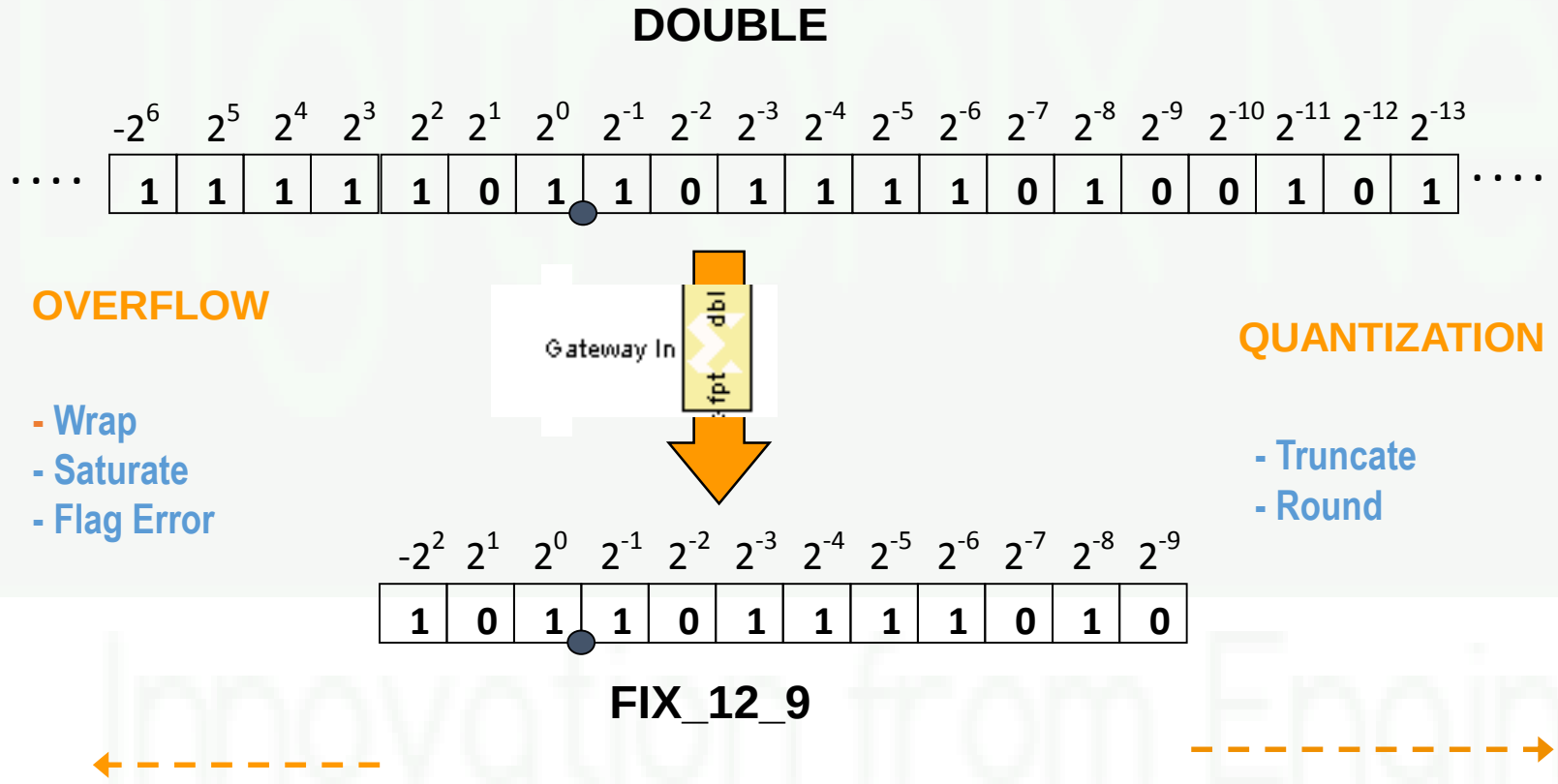
Always try to maximize the dynamic range of design by using only the required number of bits

Thus, a conversion is required when communicating with Xilinx blocks with Simulink blocks

(Xilinx blockset → MATLAB I/O → Gateway In/Out)

Gateway In/Out Blocks

- The Gateway In and Out blocks support parameters to control the conversion from double precision to N – bit fixed point precision



Sample Period in System Generator

1. Every System Generator signal must be “sampled”; transitions occur at equidistant discrete points in time called sample times
2. Each block in Simulink design has a “Sample Period” and it corresponds to how often that block’s function is calculated and the results outputted
3. This sample period must be set explicitly for:
   - Gateway in
   - Blocks with or without inputs
4. Sample period can be “derived” from input sample times for other blocks

Sample Period in System Generator

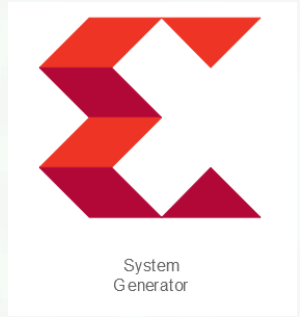
5. A sample period of 1/44100 means the block's function will be executed every 1/44100 of a sec

6. Remember Nyquist theorem($F_s \geq 2f_{max}$) when setting sample periods

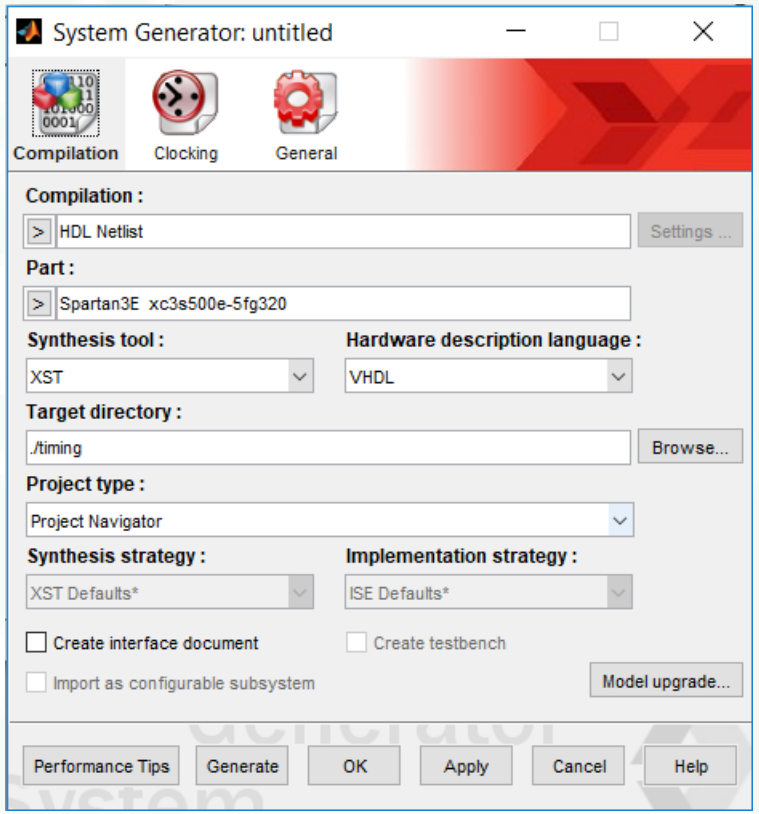
7. The sample period of a block directly relates to how that block will be clocked in the actual hardware.

System Generator Token

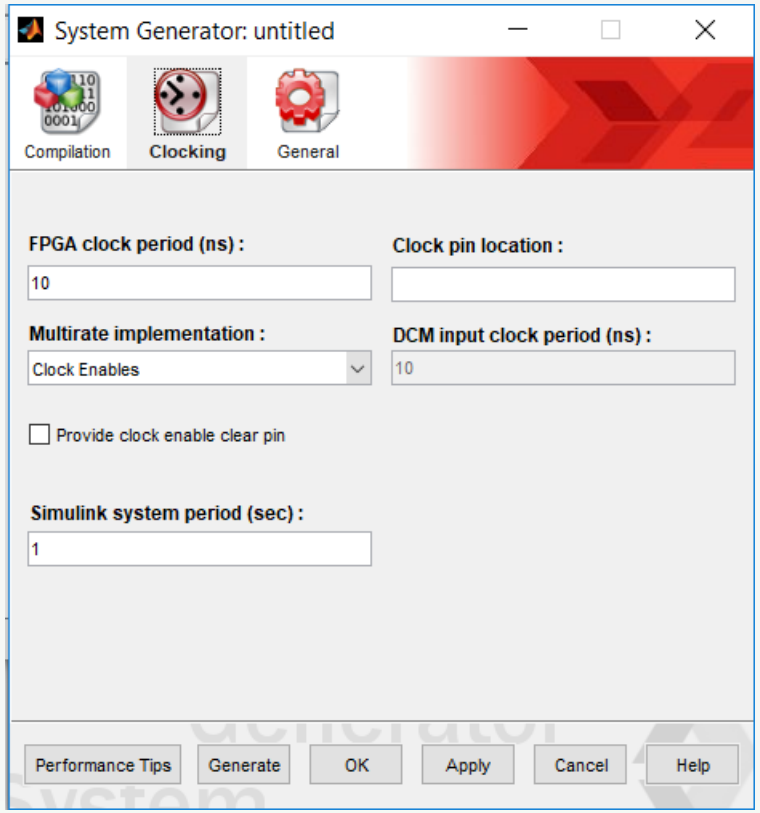
1. System Generator token is a system parameter block that contains all the settings related to model design.
2. It contains compilation settings to either generate HDL netlist, bitstream or to run co-simulation
3. It also contains system clock and sample period settings.



System Generator Token



Settings related to Compilation of Model



Settings related to clock period

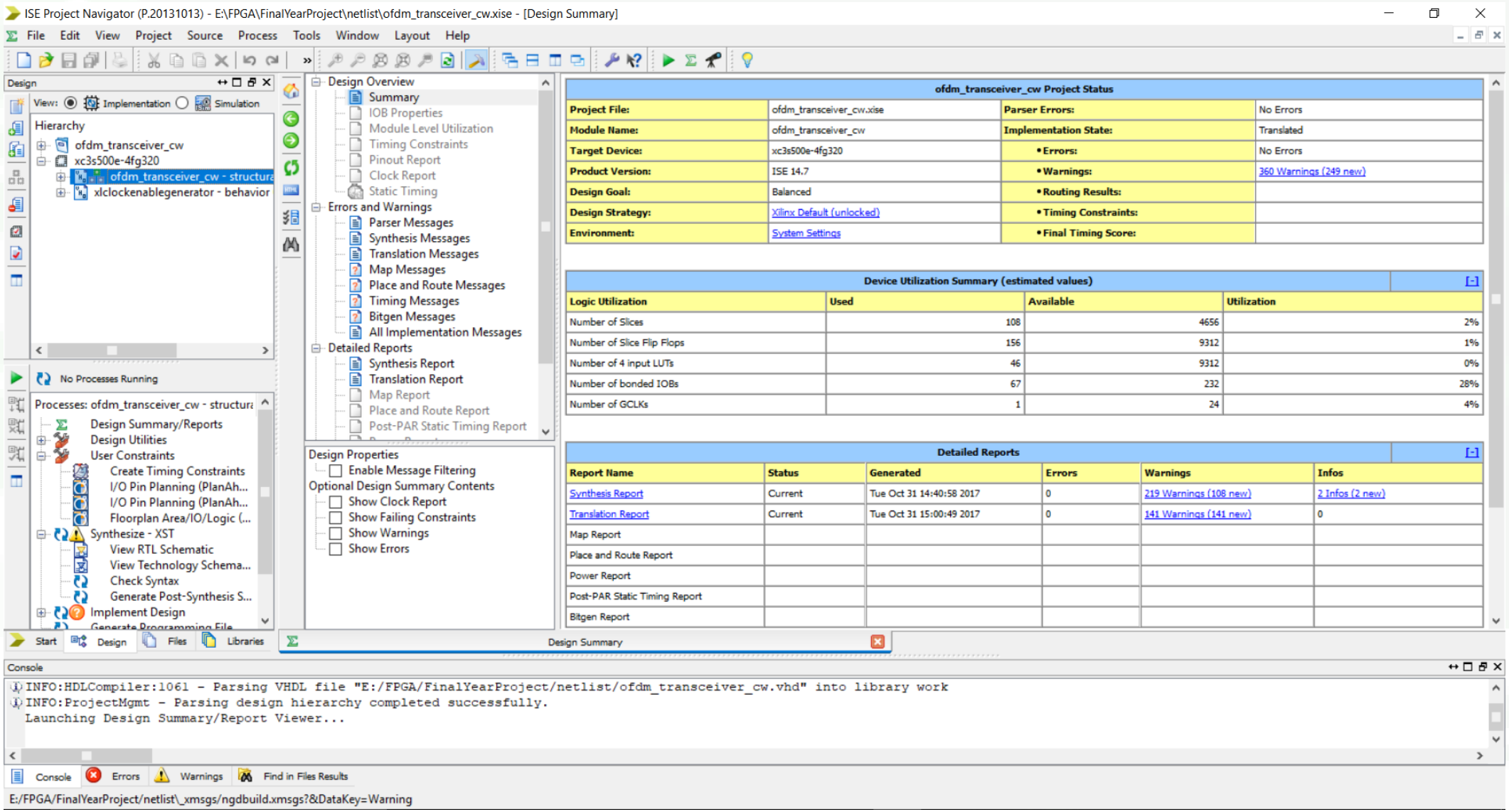
ISE Design Suite

1. Xilinx Integrated Synthesis Environment is a software tool for synthesis and analysis of HDL designs
2. It enables the developer to synthesize their designs
3. Perform timing analysis
4. Examine RTL diagrams and simulate a design's working in different situations
5. To configure the target device with programmer
6. ISE includes Embedded Development Kit, a Software Development Kit and ChipScope Pro.

ISE Design Suite

7. Xilinx System Generator can be first used to design system model
8. After designing the model, HDL netlist can be generated from the System Generator token
9. The generated netlist can be imported in ISE Design Suite as a project
10. We can then simulate the design, generate RTL diagram and run timing analysis.
11. Then we can generate bitstream and implement in the target FPGA board.

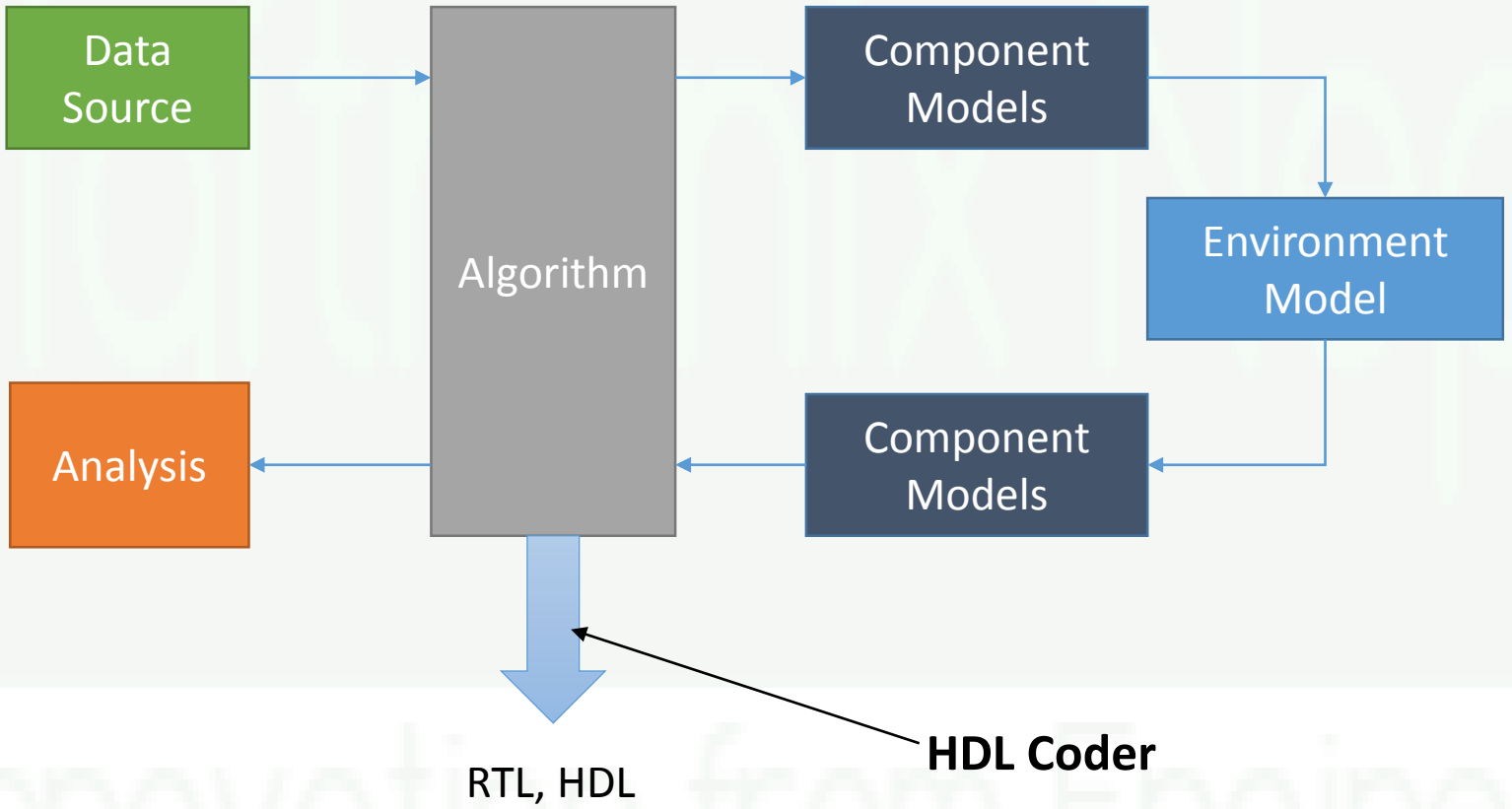
ISE Design Suite



Program Xilinx FPGAs using HDL Coder

1. HDL Coder is a MATLAB/Simulink tool which can be used to generate target-independent synthesizable VHDL and Verilog code
2. To implement the MATLAB algorithms and Simulink models in Xilinx FPGA boards, support package called Xilinx FPGA turnkey should be installed
3. HDL Coder contains a workflow advisor that guides through whole process
4. Workflow advisor helps in:
   - Preparing model for generation of HDL
   - Configuring HDL Generation options
   - Integrated with FPGA synthesis tools for timing annotation on model
   - Configuration for turnkey FPGA targets and IP core generation

Generation of HDL Source Code



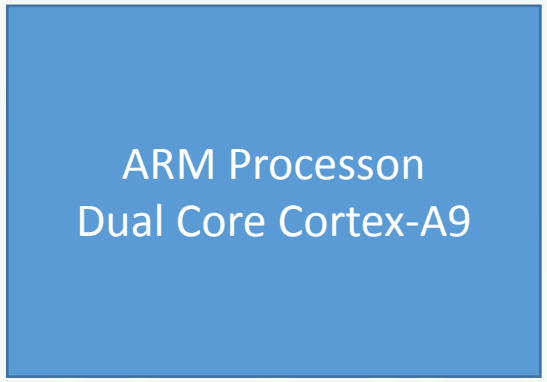
Program Xilinx FPGAs Using HDL Coder with Xilinx System Generator

1. HDL Coder can be used along with System Generator blocks
2. By creating a subsystem of Xilinx system generator blocks along with Simulink blocks HDL coder can generate HDL code for the model
3. Due to this integration we can use both the functionalities provided by the System Generator Blocks and Simulink blocks
4. HDL Coder supports Xilinx System Generator code generation with following settings only:
   - Compilation must be HDL Netlist.
   - Hardware description language must be the same as Target Language setting in HDL Coder
   - Create testbench must be unchecked
   - Multirate implementation must be Clock Enables
   - Synthesis strategy must be XST Defaults
   - Implementation Strategy must be ISE Defaults
   - Provide clock enable clear pin must be Checked

Program Xilinx Zynq SoC Devices with Embedded Coder and HDL Coder

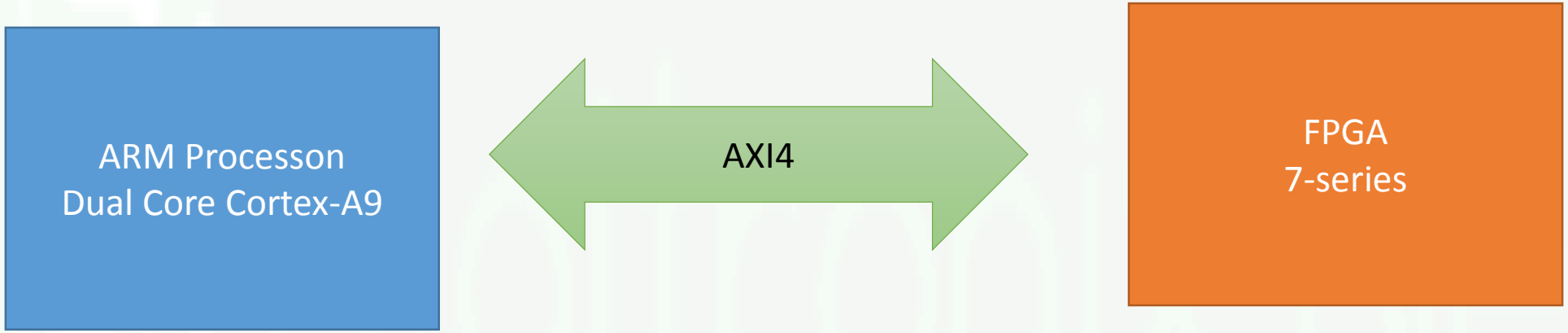
- Zynq is all programmable system on chip from Xilinx
- Zynq contains FPGA and ARM processor on one chip
- This enables high-performance system development.

Zynq Design Challenge



- ARM processors are programmed in C
- It oftens runs a Linux operating system
- FPGA Designers maynot be familiar with programming processors
- Confusions might arise regarding with portion of the system to run on the processor and which to run on FPGA

Zynq Design Challenge

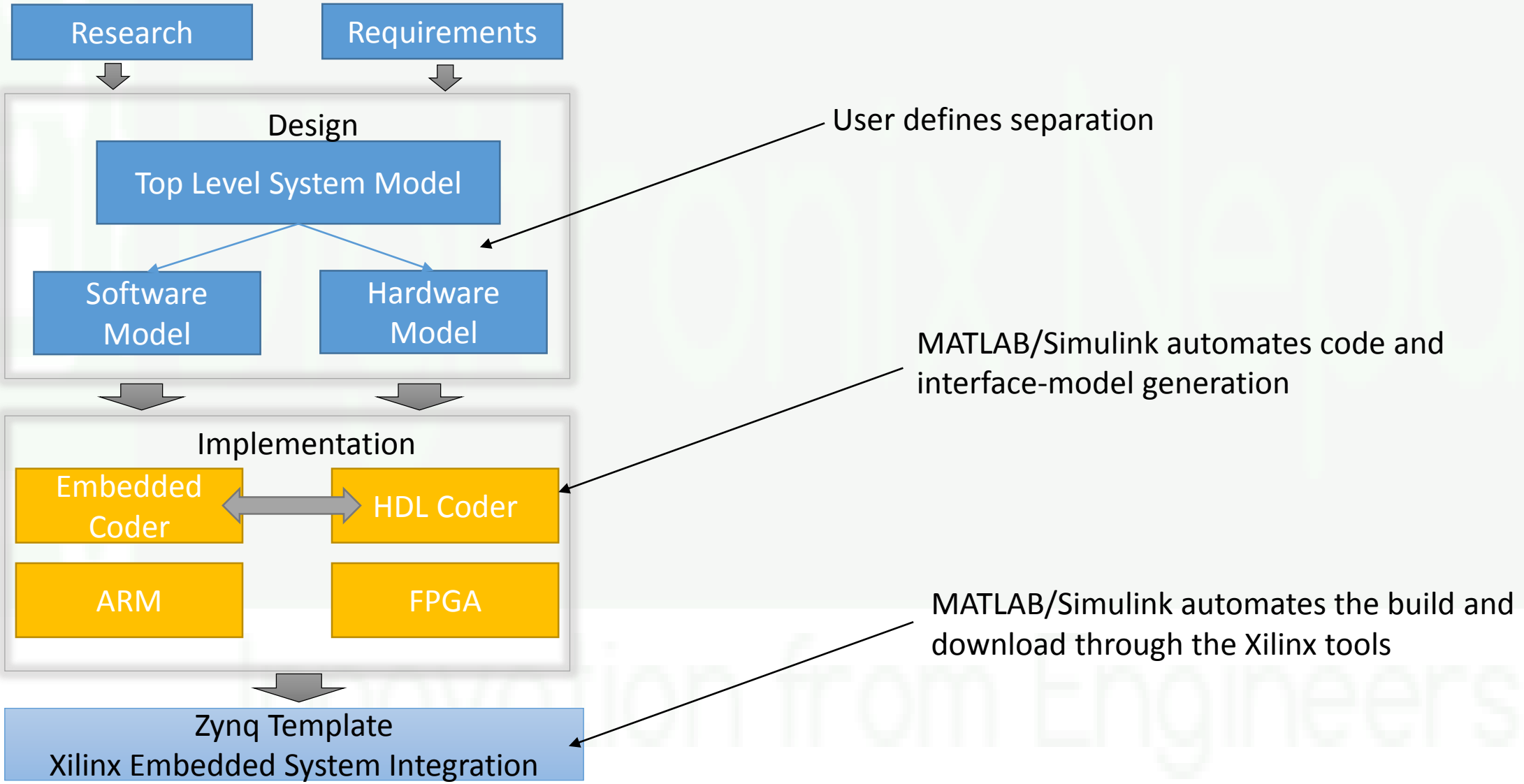


- FPGAs are programmed in VHDL or Verilog
- Has a established workflow and different from ARM design workflow
- Processor programmers may not be familiar with FPGA Design
- Zynq uses “standard” AXI4 interface between FPGA and ARM
- No established rules for hooking up the interface between FPGA and ARM processor

Zynq Design Challenge

- These challenges can be tackled easily using Model-Based Design approach with the help of Embedded coder and HDL Coder in Simulink.
- Embedded coder and HDL coder creates a high level zynq design flow for faster implementation of designs.
- Embedded coder will generate readable, compact and fast C and C++ code for use on ARM processor while HDL coder will be used to generate VHDL/Verilog code for FPGA.

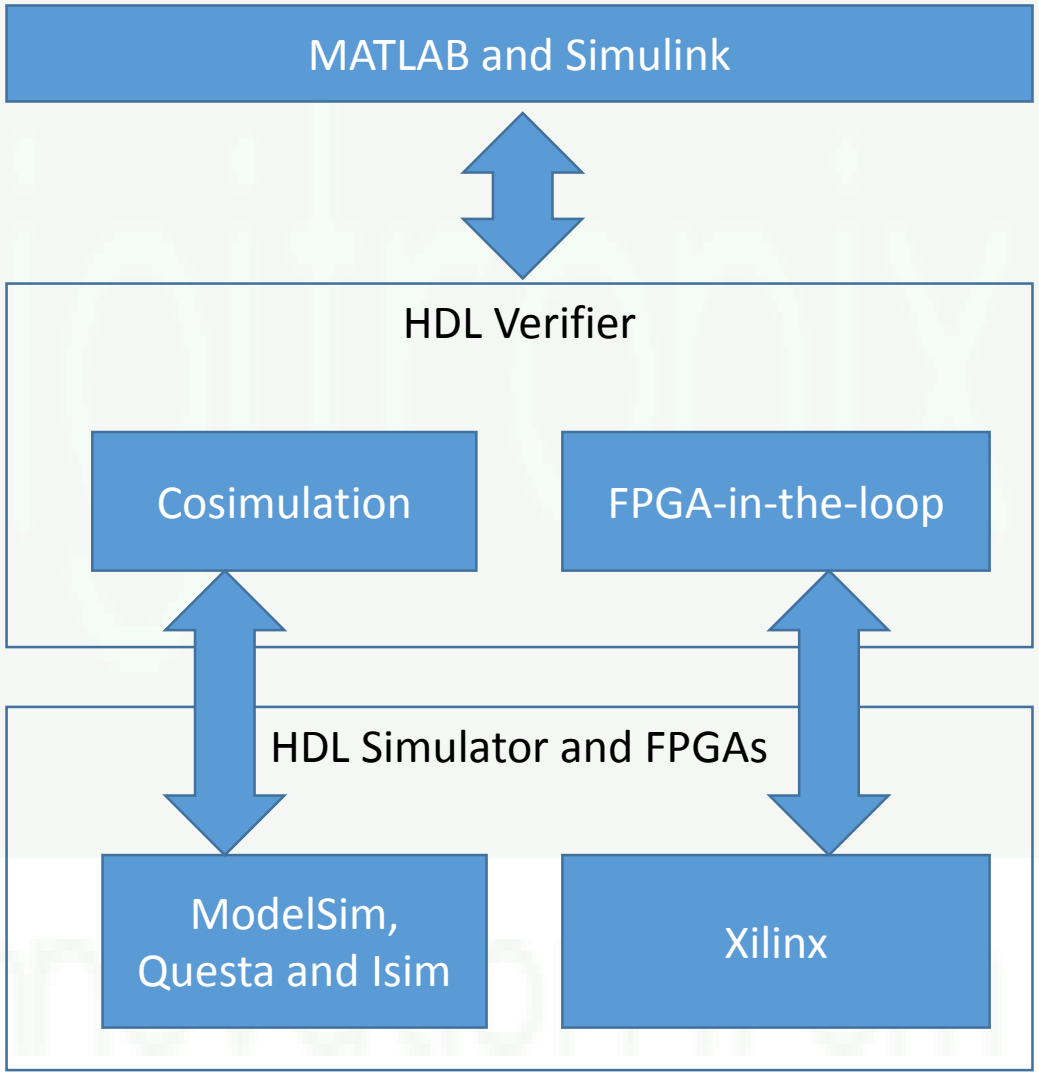
Zynq Design Flow



HDL Verifier

1. HDL verifier makes Verilog and VHDL design verification easy by using HDL simulators and FPGA hardware-in-loop
2. It interfaces MATLAB and Simulink with ModelSim, Cadence incisive and Questa HDL simulators
3. HDL verifier eliminates the need to create standalone test benches for verification
4. It helps us verify that our HDL implementation matches our MATLAB and Simulink system specifications.

HDL Verifier



References

1. https://www.mathworks.com/products/connections/product_detail/xilinx-system-generator-for-dsp.html
2. https://www.mathworks.com/
3. https://www.xilinx.com/products/design-tools/vivado/integration/sysgen.html
4. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_3/ug897-vivado-sysgen-user.pdf

Thank You!