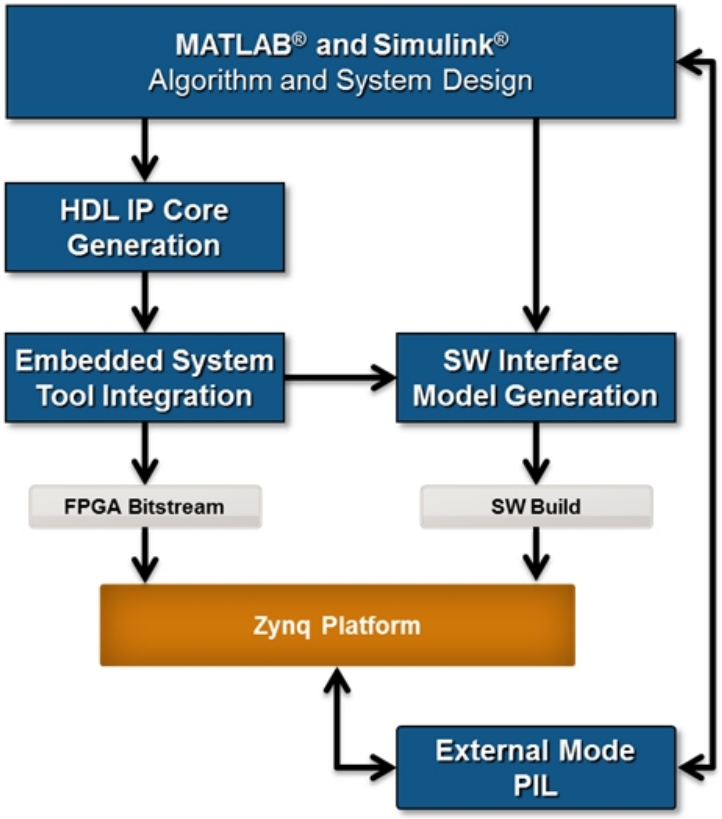
Lab 2: Model-Based System Design on Zynq Platform using MATLAB/Simulink

3 points

In this lab we use the HDL Coder™ to generate a custom HDL IP core on Xilinx® Zynq® ZedBoard, and we also use Embedded Coder® to generate C code that runs on the ARM® processor of Zynq Platform.



Use the following link to The MathWorks tutorial videos:
https://de.mathworks.com/campaigns/products/partner/xilinx-zynq-design-with-simulink.html

- A Guided Workflow for Zynq Using MATLAB and Simulink: video 20 min
- Run a Simulink Model on Zynq: example videos 1-4, 25min

You automatically generate HDL code for the programmable logic using HDL Coder, generate C code for the ARM using Embedded Coder, and implement the design on the Xilinx Zynq Platform.
In this workflow, you perform the following steps:

1. Set up your Zynq hardware and tools.
2. Partition your design for hardware and software implementation.
3. Generate an HDL IP core using HDL Workflow Advisor.
4. Integrate the IP core into a Xilinx EDK project and program the Zynq hardware.
5. Generate a software interface model.
6. Generate C code from the software interface model and run it on the ARM Cortex-A9 processor.
7. Tune parameters and capture results from the Zynq hardware using External Mode.

Requirements:

MATLAB/Simulink R2019a [1] **+ Toolboxes**

MathWorks®

Software

Required

MATLAB
Simulink
MATLAB Coder
Simulink Coder for C-code generation
Embedded Coder for C-code generation
Fixed-Point Designer for HDL-code generation
HDL Coder for HDL-code generation

Recommended

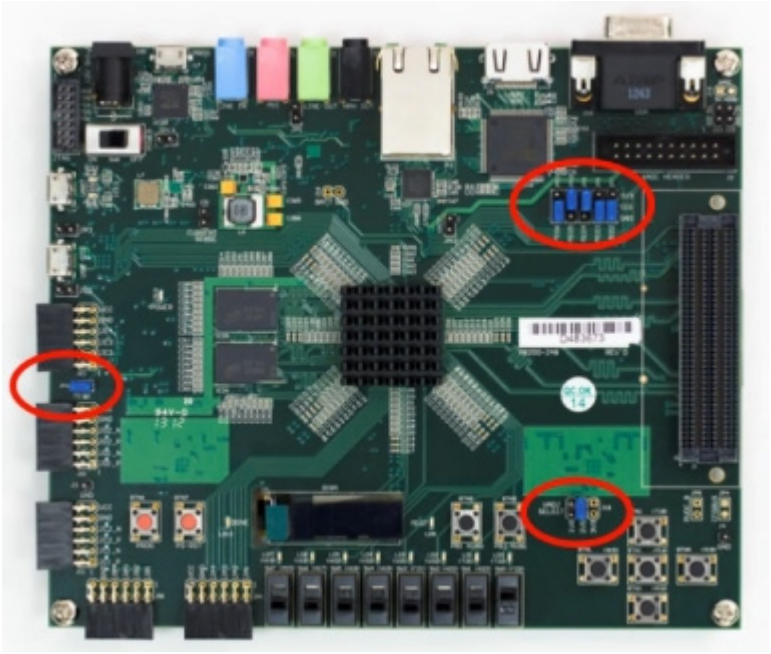
Communications System Toolbox
DSP System Toolbox
Signal Processing Toolbox

Xilinx Vivado Design Suite 2018.2 WebPACK+ SDK

ZedBoard with SD-card, 2xMicro-USB cables for JTAG and UART, Ethernet cable

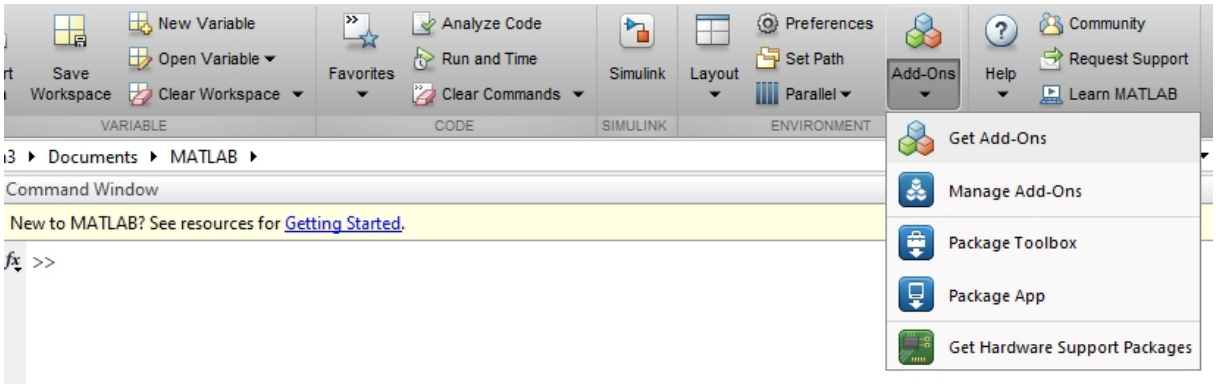
1. Setup Zynq hardware and software tools

a) Jumper settings of ZedBoard



b) Start MATLAB (Note: MATLAB working directory must not contain white spaces!)

[1] HDL Coder of MATLAB R2019a supports Vivado 2018.2



Install two support packages in MATLAB (Add-Ons --> Get Hardware Support Packages)

HDL Coder Support Package for Xilinx Zynq Platform

Click on **Install**

Embedded Coder Support Package for Xilinx Zynq Platform

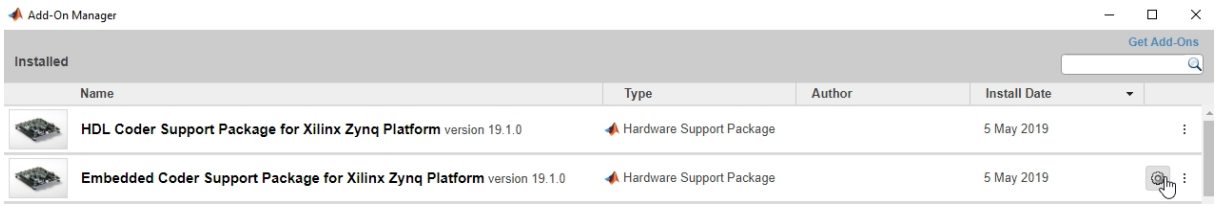
Click on **Install**

You need to login in your MathWorks account to get the support packages. In the next step you need to select a board.

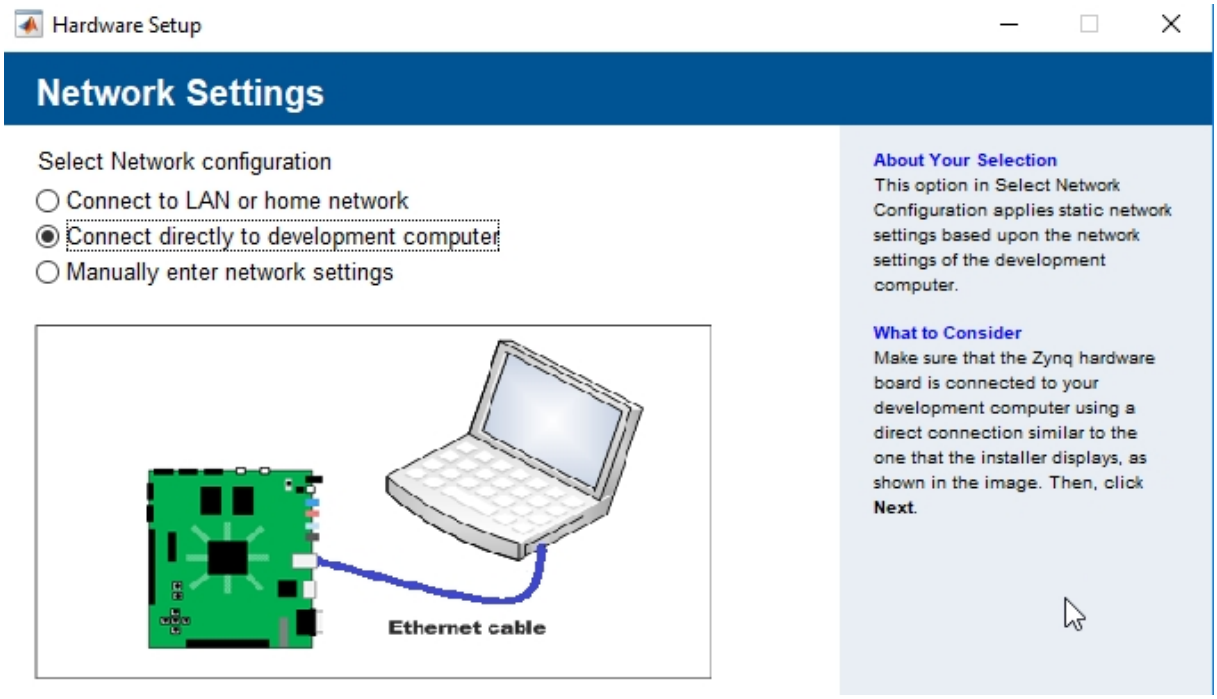
c) select ZedBoard

Add-Ons --> Manage Add-Ons

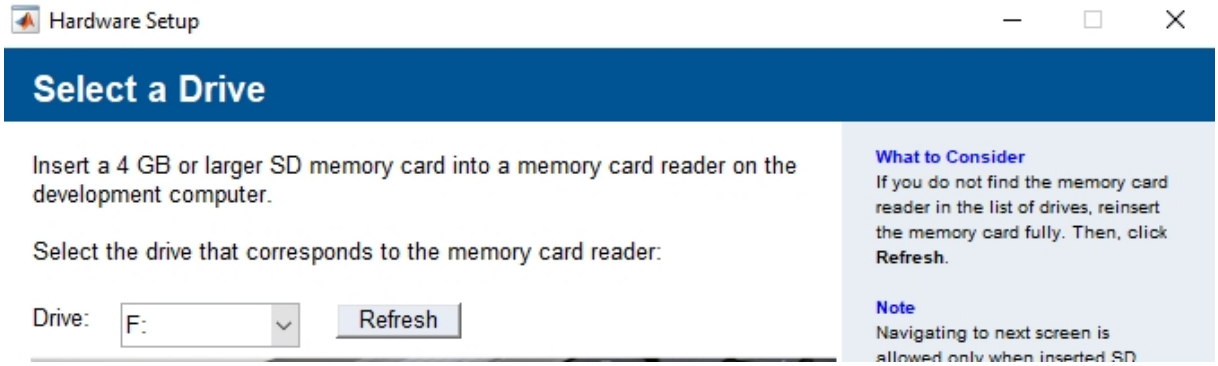
Click on Setup for Embedded Coder Support Package



Now connect ZedBoard to your computer via Ethernet-cable. Then click next.

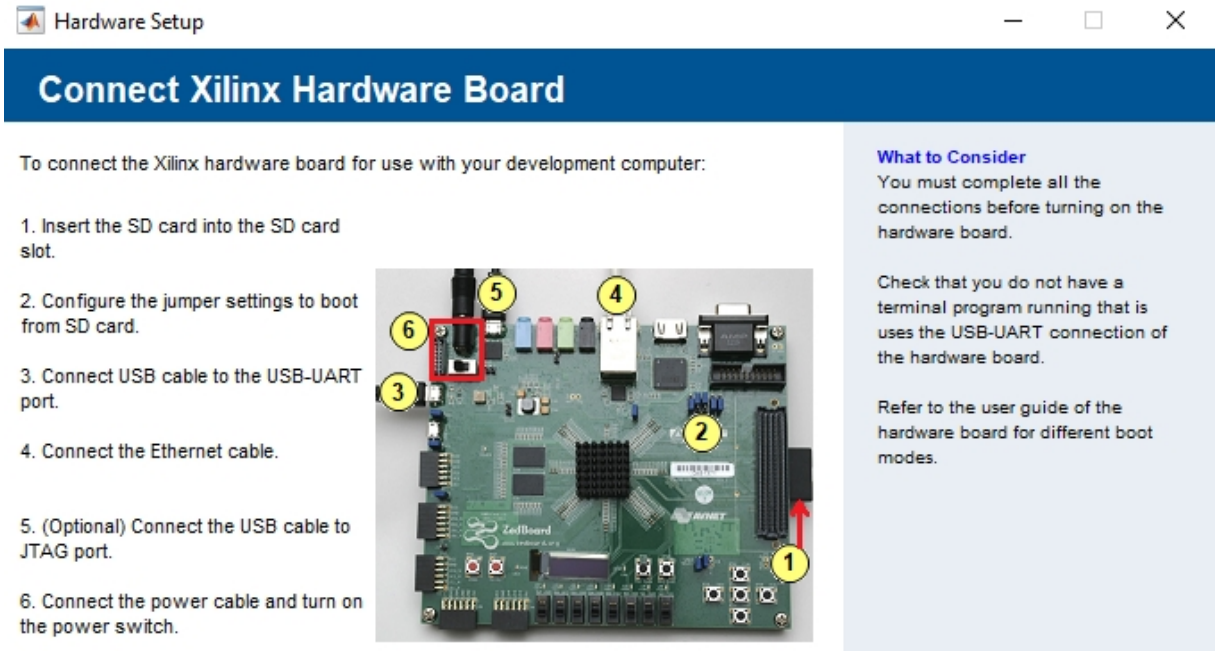


Insert the SD-card on your computer. Click next.



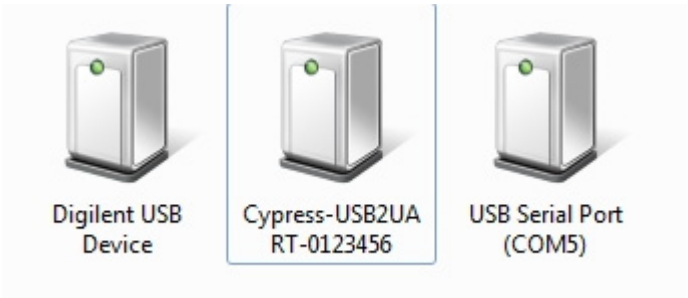
Write firmware to SD-card. Click next.

Insert SD-card into ZedBoard, connect ZedBoard to host computer as shown below.

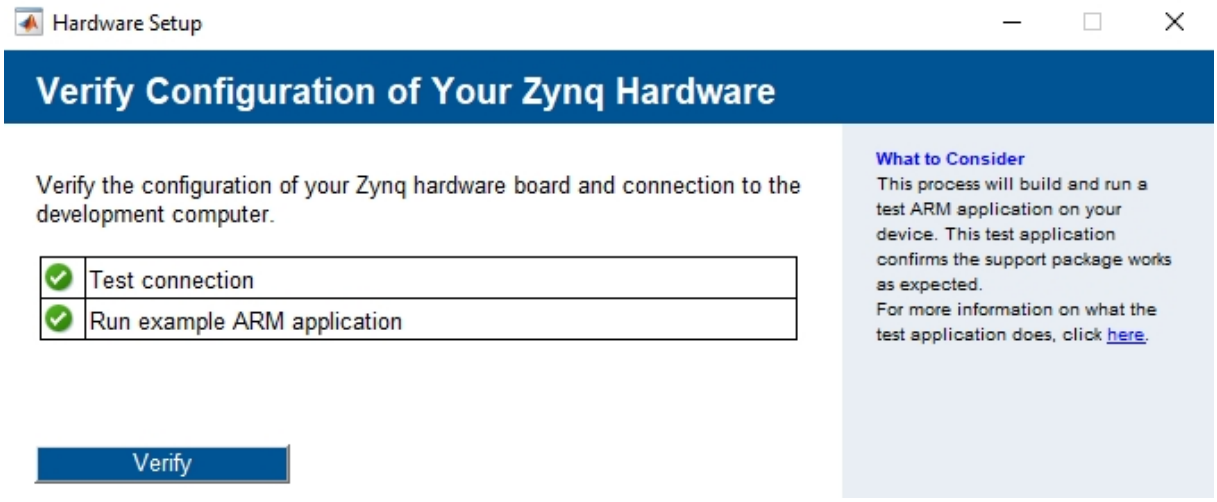


Turn on power switch of ZedBoard(before check: jumper settings, USB cable to JTAG junction, USB-to-UART connection, Ethernet cable on board to host computer, SD-card inserted).

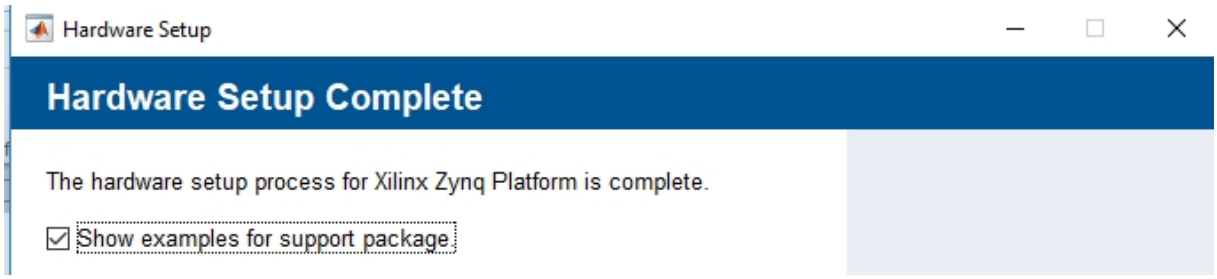
Check if the devices are active:



click Next. Verify configuration of Zynq Hardware. Click Verify!



Click Next. Click Finish.



d) MATLAB setup

When using Vivado 2018.2 enter the following command in MATLAB command window:

```
hdlsetuptoolpath('ToolName','Xilinx Vivado','ToolPath','C:\Xilinx\Vivado\2018.2\bin\vivado.bat');
```

This is to setup the Xilinx Vivado synthesis tool path.

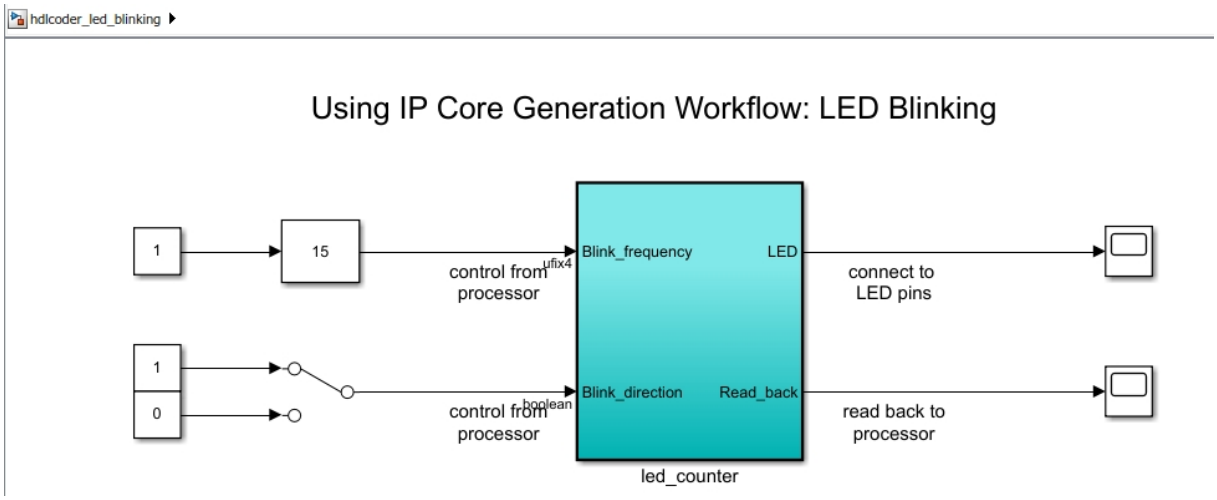
```
z = zynq
```

This is to setup the Zynq hardware connection

2. Perform HW/SW Workflow in MATLAB/Simulink

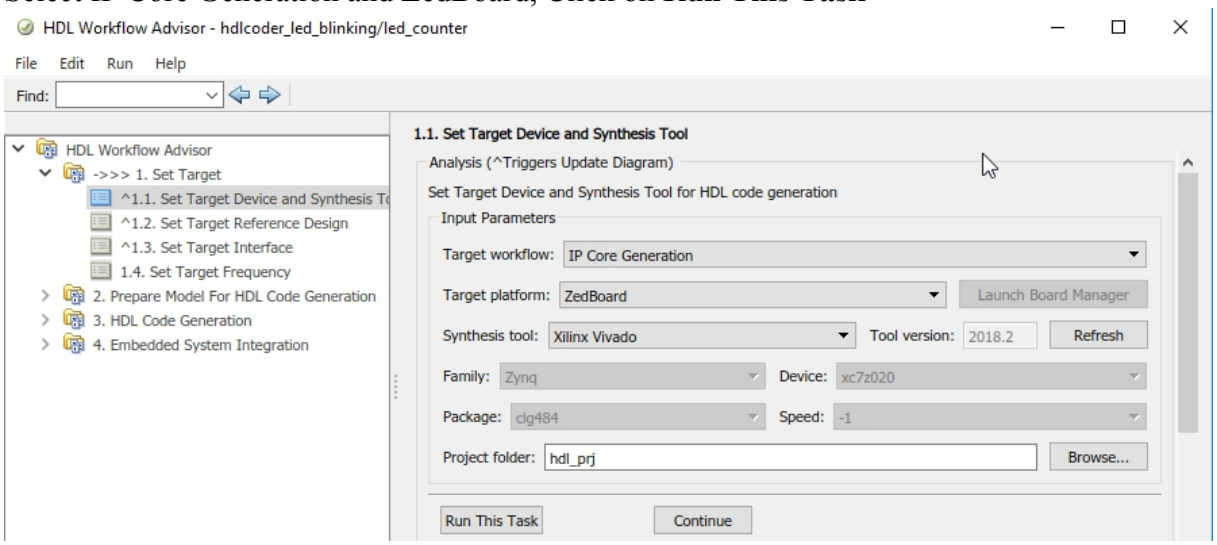
The first step of the Zynq hardware-software co-design workflow is to decide which parts of your design to implement on the programmable logic, and which parts to run on the ARM processor. Start the example design:

```
>>hdlcoder_led_blinking
```

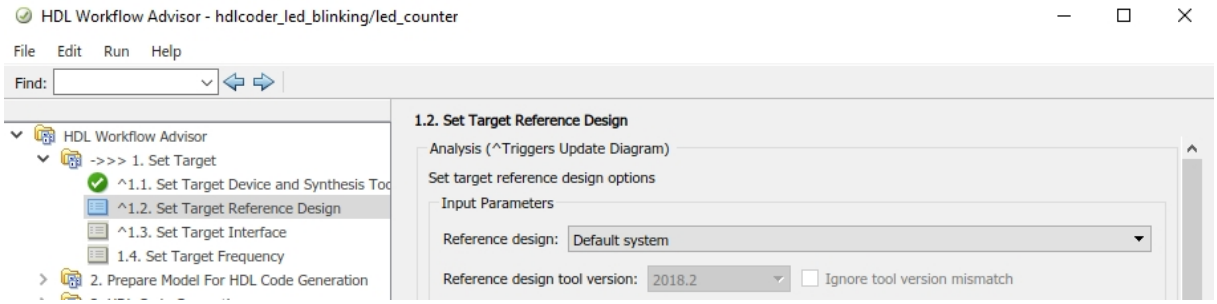


In this example, the subsystem **led_counter** is the hardware subsystem. It models a counter that blinks the LEDs on an FPGA board. Two input ports, **Blink_frequency** and **Blink_direction**, are control ports that determine the LED blink frequency and direction. All the blocks outside of the subsystem **led_counter** are for software implementation running on ARM processor. HDL Workflow Advisor enables you to automatically generate a sharable and reusable IP core.

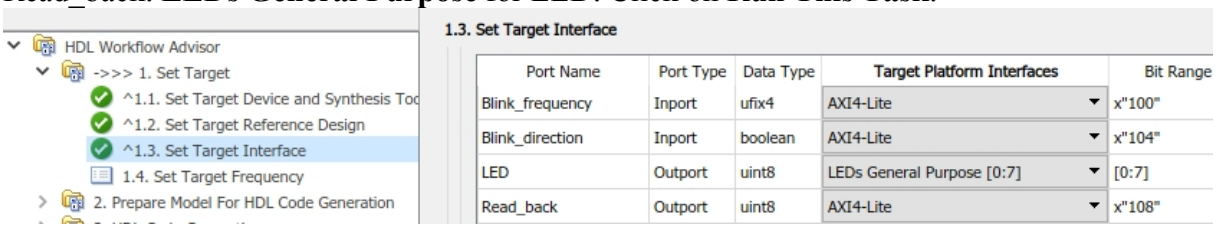
Click on block **led_counter**, right clicking: HDL code -> HDL workflow advisor
Select Set Target -> Set Target Device and Synthesis Tool
Select IP Core Generation and ZedBoard, Click on **Run This Task**



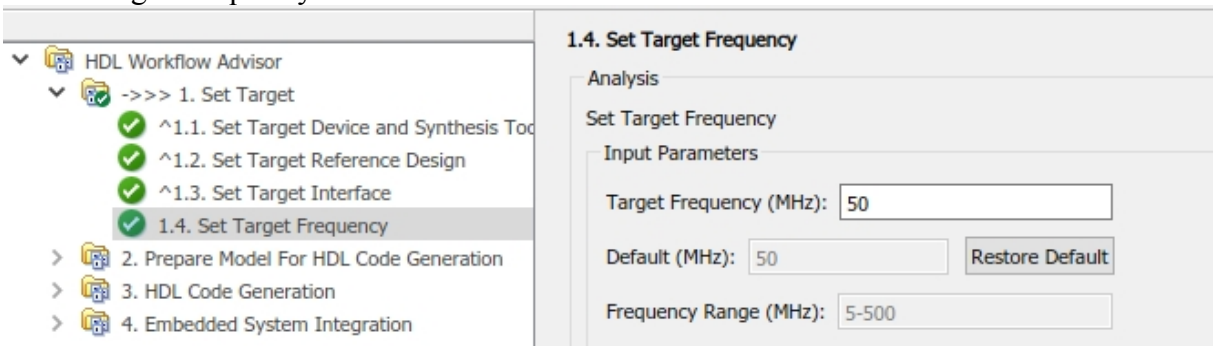
In Set Target Reference Design choose Default system. Click on **Run This Task**.



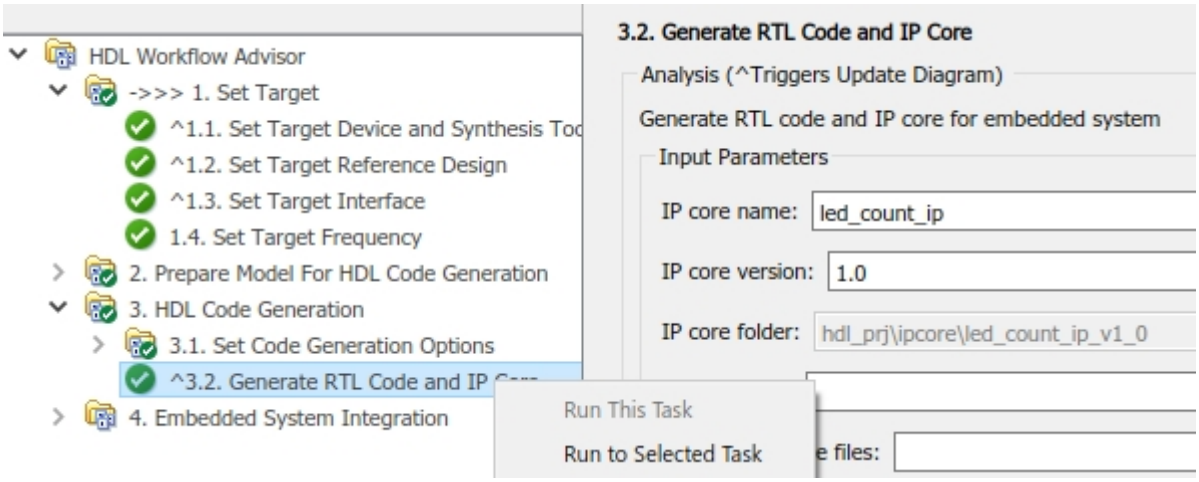
In Set Target Interface choose **AXI4-Lite** for Blink_frequency**,** Blink_direction, and Read_back. **LEDs General Purpose** for LED. Click on **Run This Task**.



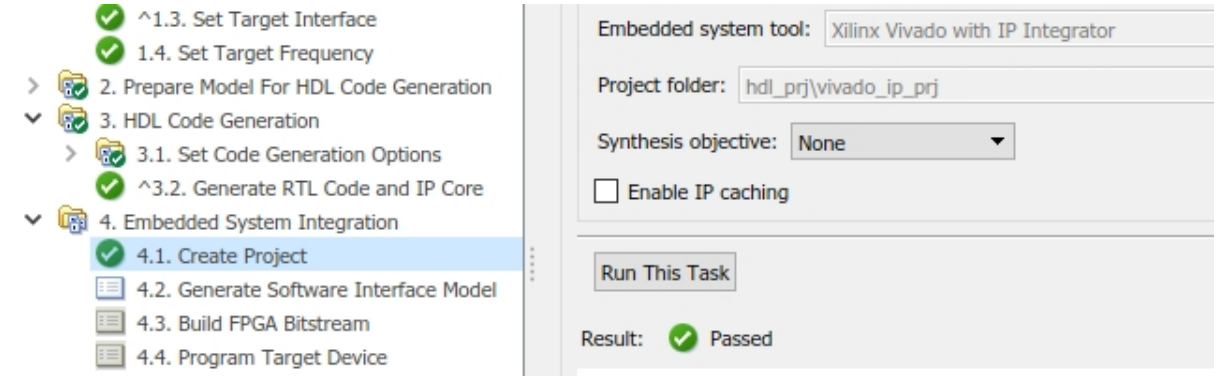
In Set Target Frequency choose 50 MHz. Click on **Run This Task**.



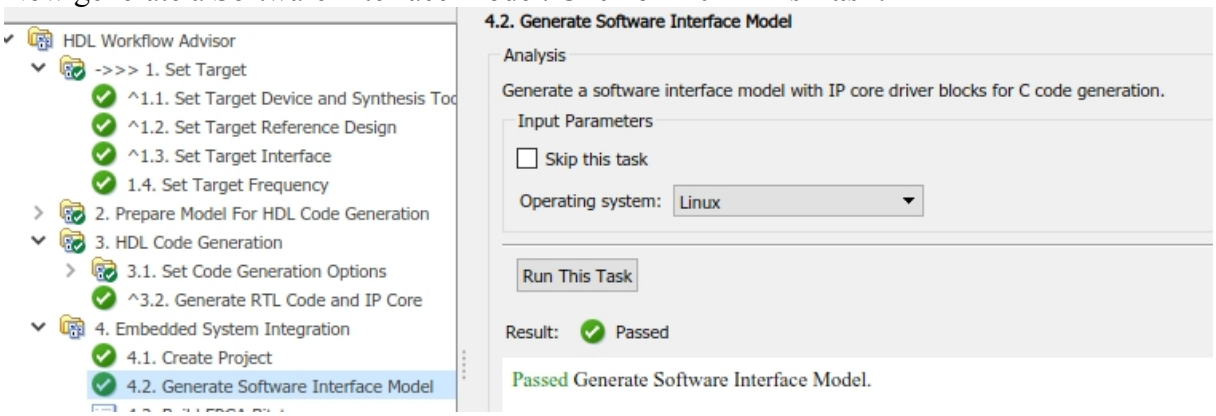
In HDL Code Generation right-click the **Generate RTL Code and IP Core** task and select **Run to Selected Task**



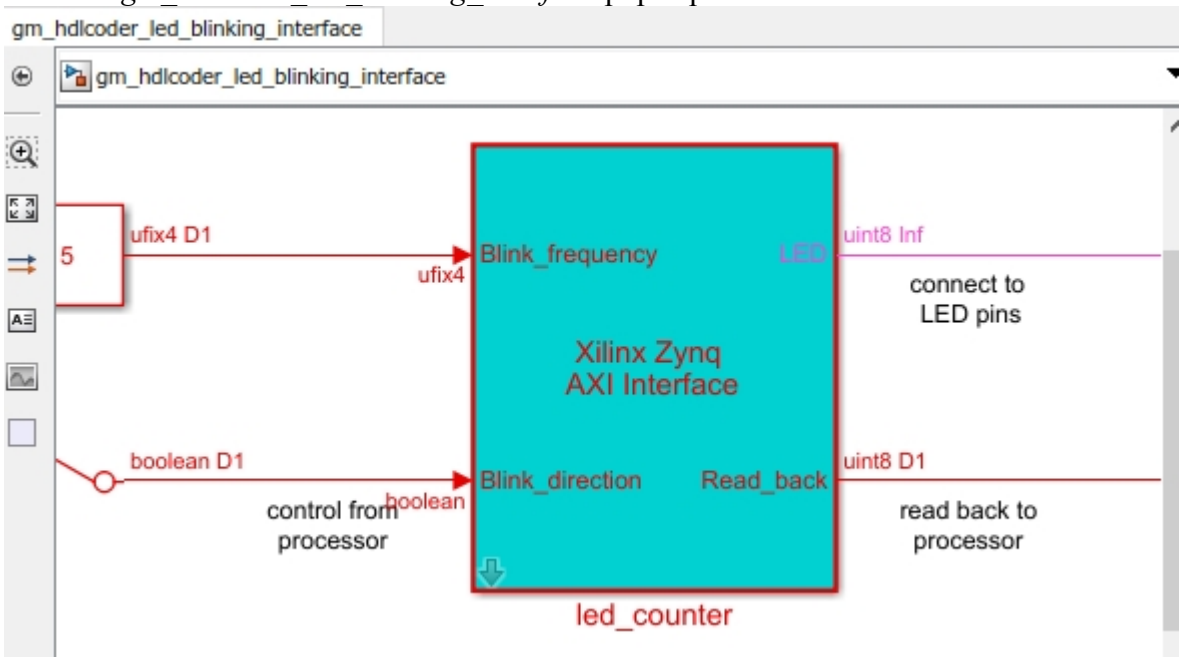
Now the HDL code is generated and you need to create a Xilinx project out of the HDLWorkflow Advisor by running step 4.1. In Create Project click on **Run This Task**.



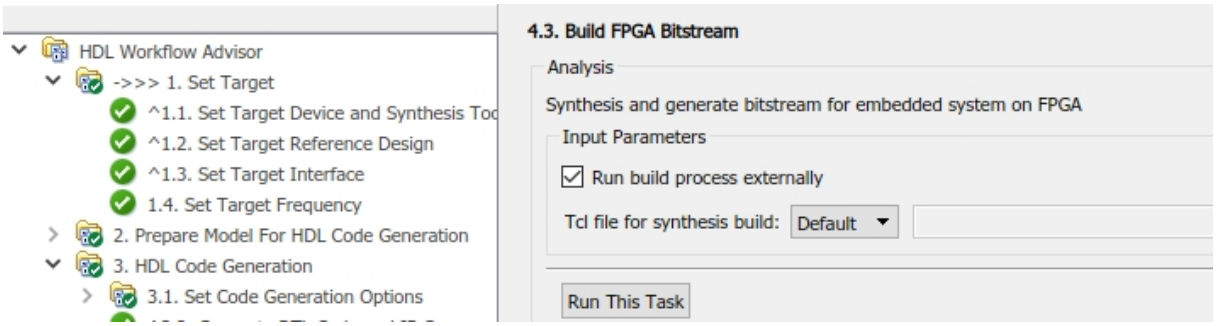
Now generate a Software Interface Model. Click on **Run This Task**.



Window *gm_hdlcoder_led_blinking_interface* pops up.

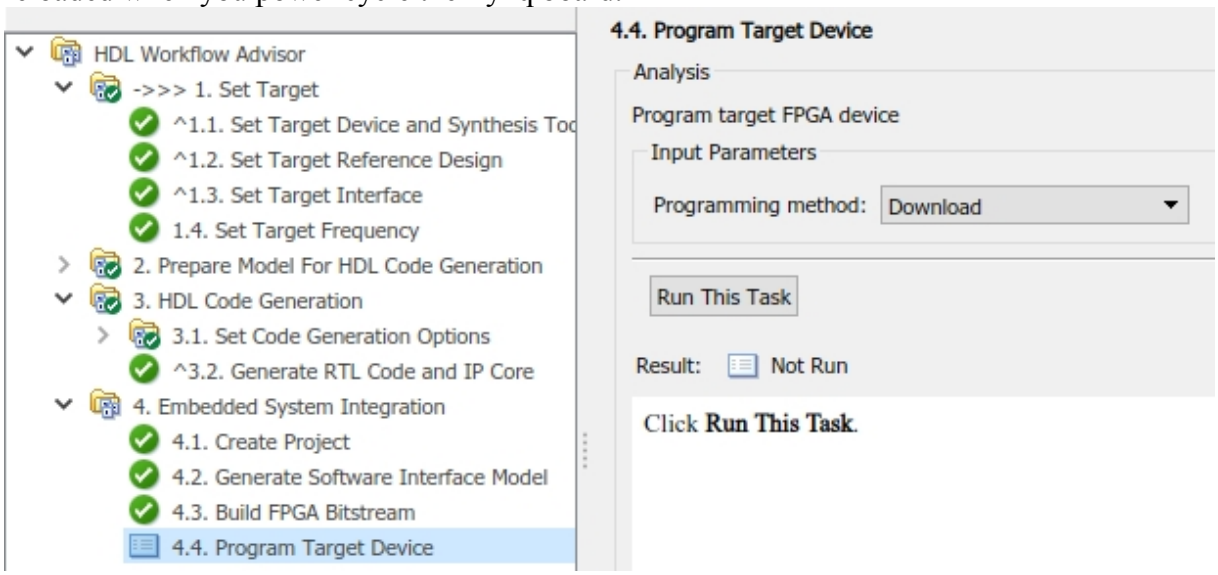


Click on Build FPGA Bitstream: **Run build process externally, Run This Task**



External Console is used for info on bitstream generation. Wait for message: Embedded system build completed

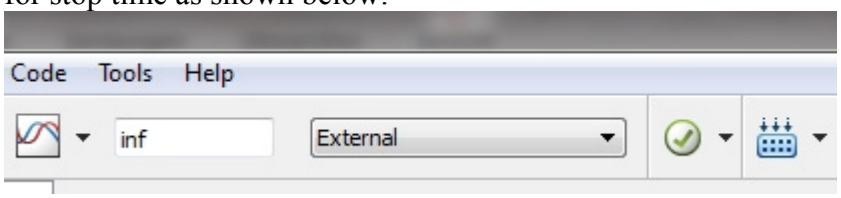
Program Target Device: Choose **Download** for **Programming method** to download the FPGA bitstream onto the SD card on the Zynq board, so your design will be automatically reloaded when you power cycle the Zynq board.



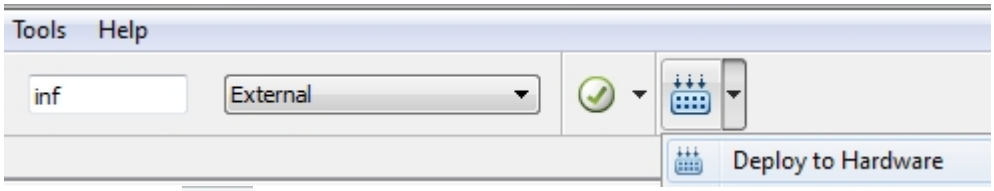
LEDs start blinking on ZedBoard!

Next, you configure the software interface model and generate C-code to run on the ARM processor to control the LED blink frequency and direction. Your hardware is linked to the Simulink model on the host computer through Ethernet cable. So you can tune and monitor the algorithm running on ZedBoard.

In *gm_hdlcoder_led_blinking_interface* window select External as simulation mode and inf for stop time as shown below:

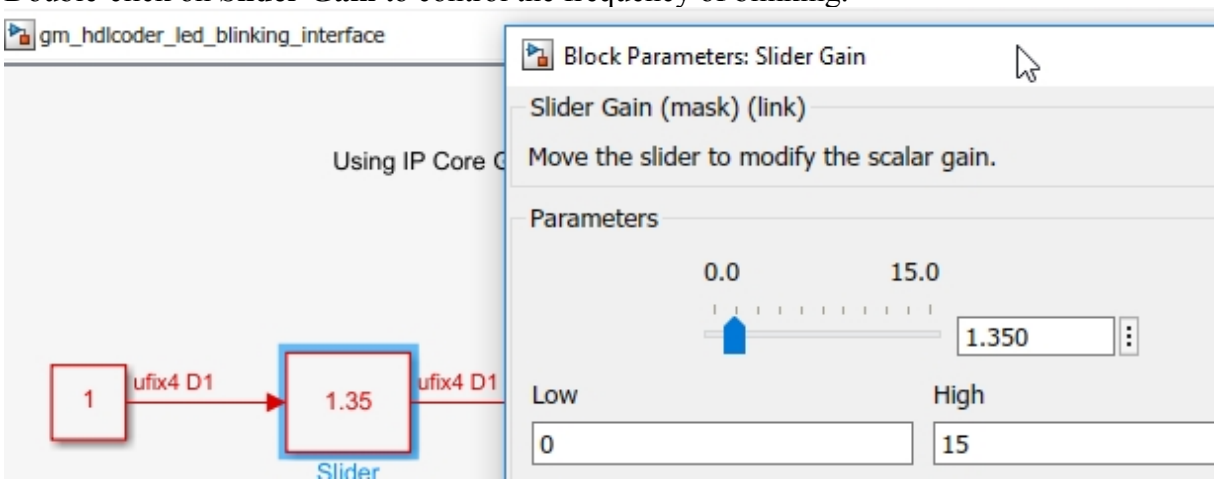


Now you **Deploy to Hardware**:



Click on Run 

Double-click on **Slider Gain** to control the frequency of blinking.



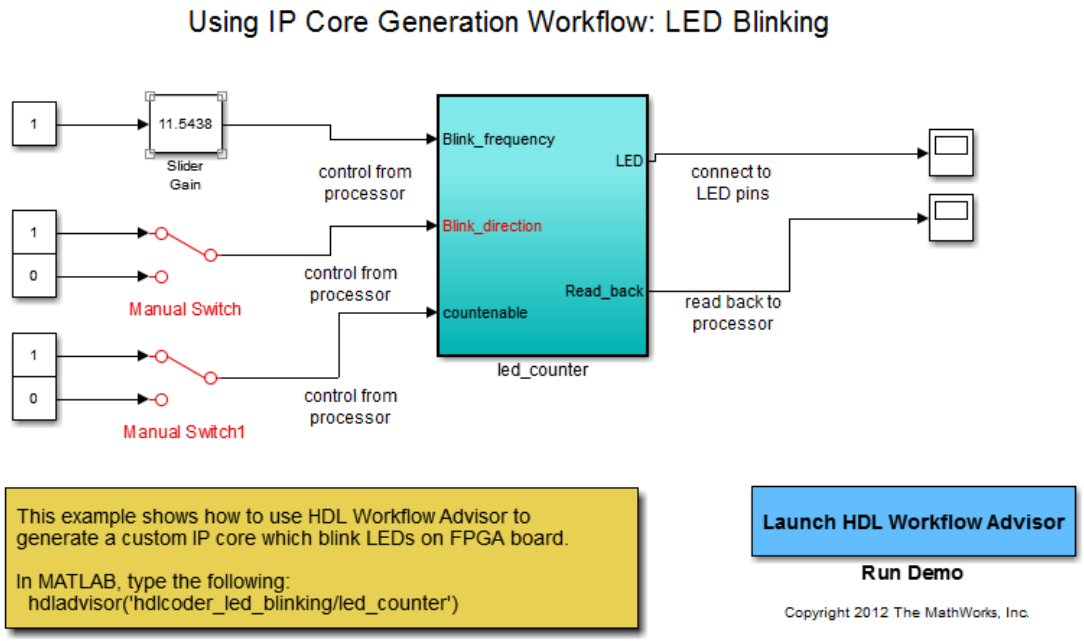
Double-click the **Manual Switch** to switch the direction of blinking LEDs!

Congratulation, the preparatory work is done! To understand the model-based design flow better, the following practical task might help.

Practical Tasks

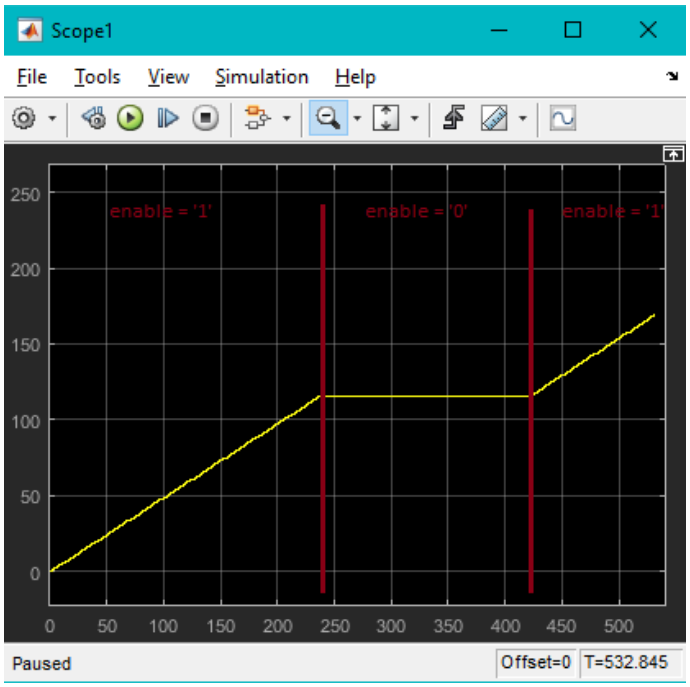
1. Perform the workflow of Model-Based Design for the provided example (p.2-10).
   Run your model on the ZedBoard in External mode.
   Answer the following questions:
   a) What is the meaning of the particular jumper settings for the ZedBoard (check in user guide)?
   b) What is the purpose of output "read back to processor"?
   c) After you generate the custom IP core, a HDL code generation report is getting generated. Open the high-level resource report: How many adders, registers and mux are used? Open the generated source file *led_counter.vhd*: what is the purpose of the two VHDL processes defined in this file?
   d) Open the HTML C-code generation report. Do a screenshot of the list of generated code (main file, model files, data files).

2. Your task is to enhance the current Simulink model with a counter enable signal to control the counter by an external signal from the ARM processor. The counter has to stop and continue again. A top-level design could look as following:



a) Double-click on *led_counter* to open design. Describe the Simulink counter model! Modify the *led_counter* block to the respective requirement! If you need additional Simulink blocks, you can find them under "Tools – Library Browser". **Save** your Simulink model within your MATLAB projects directory (don't save within hdlcoderdemos directory).

b) Now simulate your model by using the "Run" button.
The following scope view shows the required functionality.



c) If your Simulink model works fine, apply the new led_blinking implementation to the ZedBoard. Switch on power supply of your ZedBoard. Perform the MATLAB setup 1d) to set the synthesis tool path.
You have to run the **HDL Workflow Advisor** to generate HDL code using HDL Coder and the Software Interface model using Embedded Coder. Build FPGA bitstream and program the device. LEDs are not blinking yet! Now deploy your software model to hardware. You can now again control blinking frequency and direction. Also you have option to disable counting. Try this and show tutor in lab!

Prepare before lab: MATLAB setup b) , questions 1a)+b)
Note: Tutor will check on your preparation at the beginning of lab! No preparation means you need to leave lab with no points!

Lab report:

1. Screenshots of slider gain window and Simulink scope of practical task No.1!
2. Answers to questions of practical task No.1!
3. Screenshot of new Simulink model of practical task No.2! Explain the change of the model for implementing the *counter enable* function
4. Screenshot of Simulink scope to prove that the counter is stopping and continuing again.

Each lab group must submit a report (hardcopy) no later than June/7!