Design a Fixed Threshold Discriminator with SciCompiler

Abstract

This application note will show how to support a scientific application with an FPGA firmware, consisting of a signal discriminator with an arbitrary but fixed hardcoded threshold, designed and compiled with SciCompiler programming language and run on the SciDK development board.

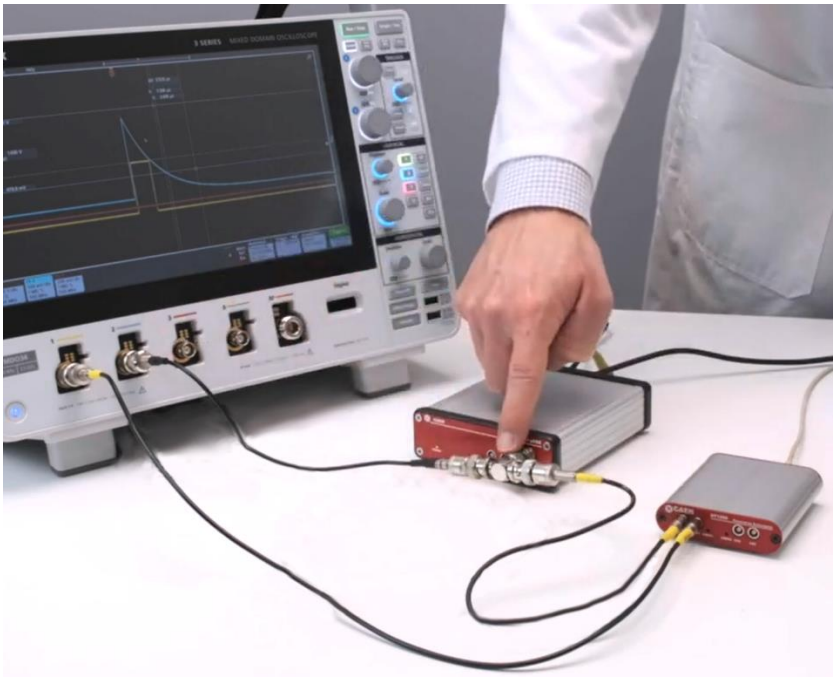
In order to build and demonstrate the project the CAEN DT4800 signal generator will also be used and connected to both the board and an oscilloscope via a split T.

The LEMO Digital I/O port available on the SciDK will send the discriminated signal to the oscilloscope for further analysis.

Index

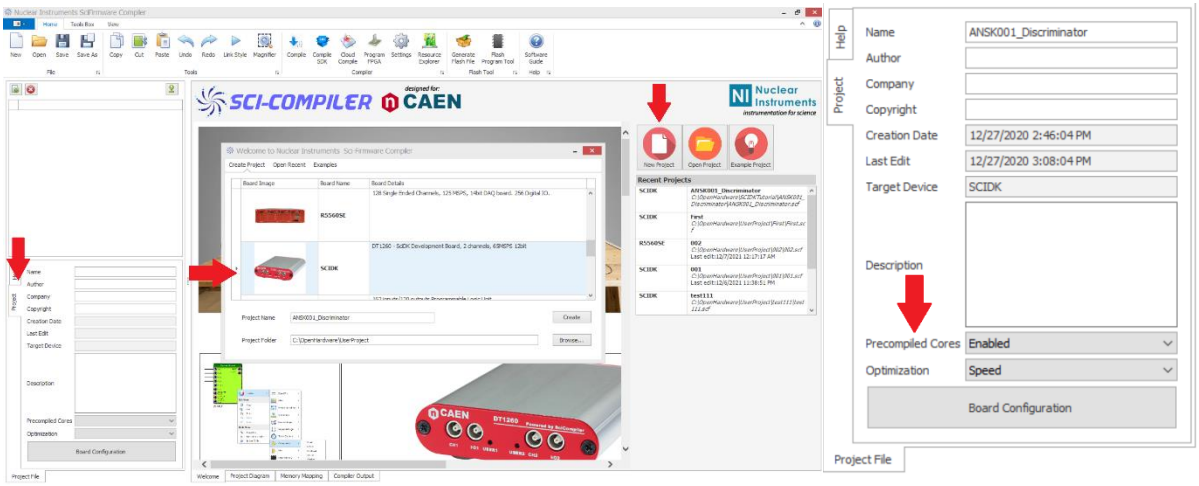
1 Preliminary setup

First make sure to have all the relevant connections in place and the latest available software upgrades installed. This is a reference of how your work area may look like:



SciCompiler project setup

Now open SciCompiler, select *New Project*, insert the name of the design we're going to create (for reference: *ANSK001_Discriminator*), select the relevant board (*SciDK*) and verify that under the *Project* tag the field *Precompiled Cores* is enabled; this may later speed up the compilation of the FPGA firmware by a significant margin:

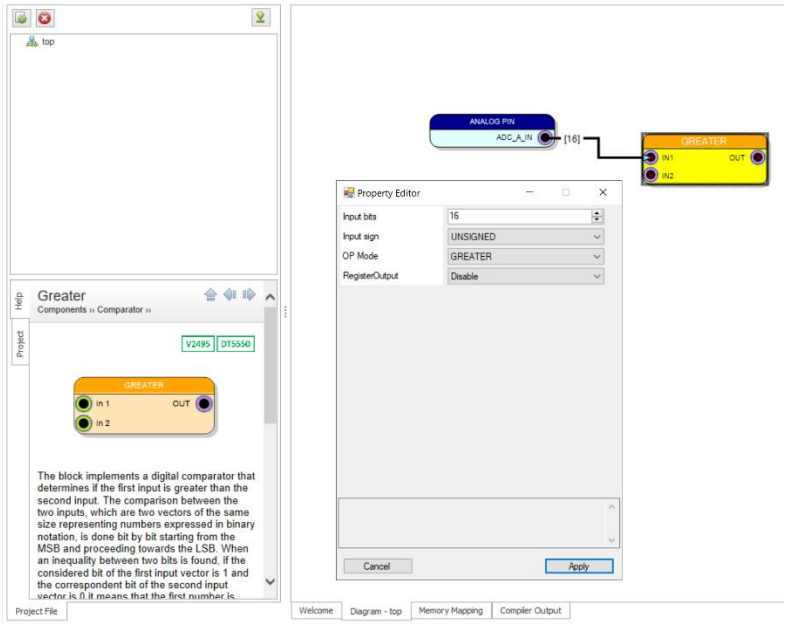


2 Block Diagram implementation

As the project revolves around a discriminator function we first need to pickup and place an **Analog Input** box selecting it from the *Board Pin* section of the *Tools Box* bar and a **Comparator** by alternatively opening the *Toolbar* via a right-click on the project area. For this presentation the **Comparator** of choice is **Greater**.

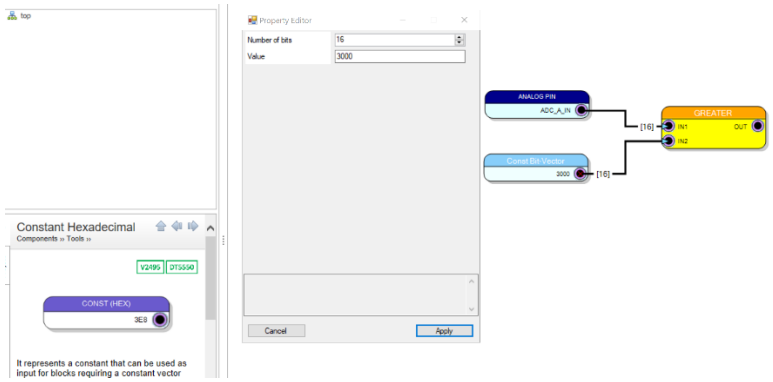
SciCompiler analog internal bus is always 16 bit whichever board it is attached to; for this design we leave the prompted *Propriety Editor* settings as default.

To connect the blocks just drag a virtual wire as if they were physical instruments, for any more information it is possible to refer to the *Context Help* :



Comparator finalization

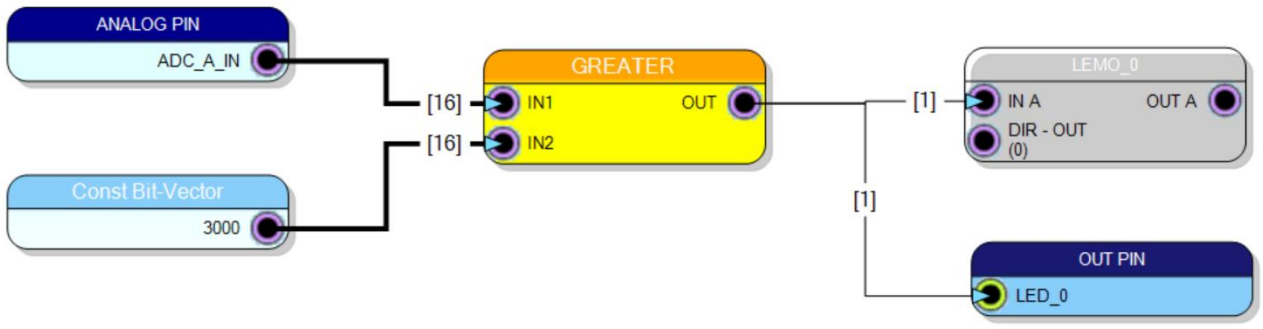
Proceeding further we need to create a **Constant-Bit Vector** box selecting it from the *Misc* section of the *Tools Box* and set it according to the *Propriety*(s) of the other boxes so that it's possible to finally assign the ~Fixed Threshold~ of the **Comparator**, in our design with a *Value* of 3000 :



3 Output section

To implement an effective output section we can use the *LEMO connector* on the SciDK board and to do so we just position the relevant **LEMO** block from *Toolbar* and then *Board Pin*. Please note that LEMO is a dual channel bidirectional port and can also be setup as a digital input as specified in the *Context Help*.

Is also possible to assign an output to one the two onboard LED which in this case will blink when the signal is over the threshold, they're labeled *Digital Pin* in the *Board Pin* section:

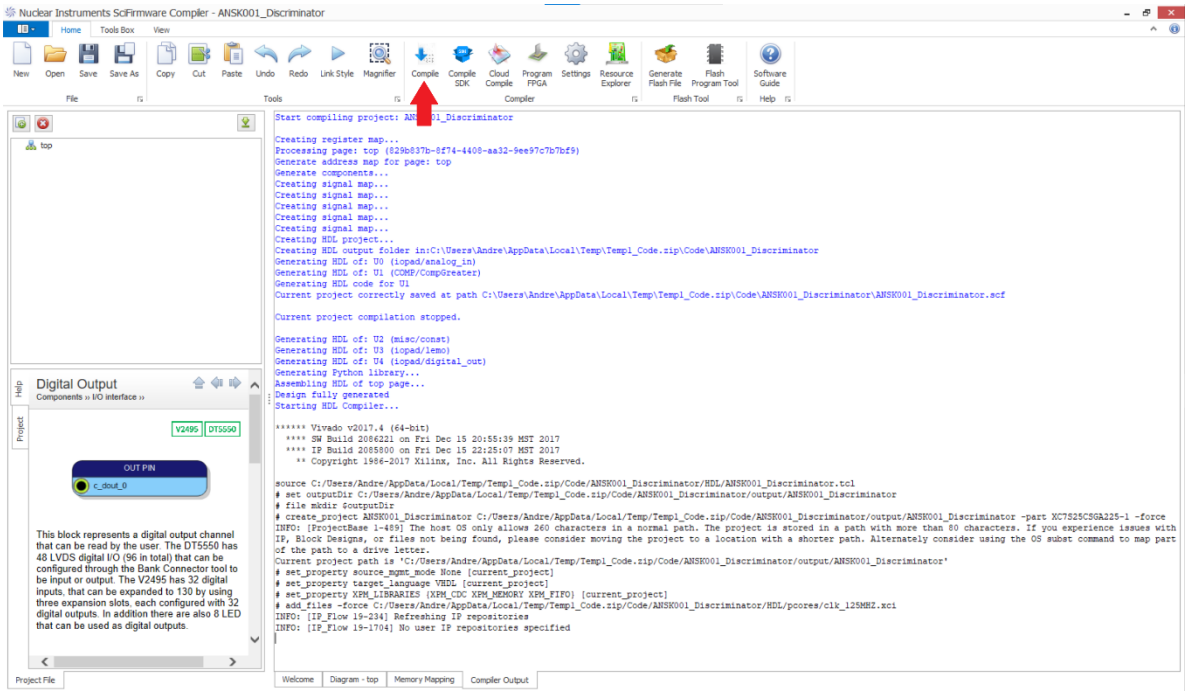


The output section is done and so, with no register or memory mapped pheriferals, is the whole design.

4 Compilator

Once the design is done you can leave the *Tools Box* for the *Home* menu bar, where you will click *Compile*.

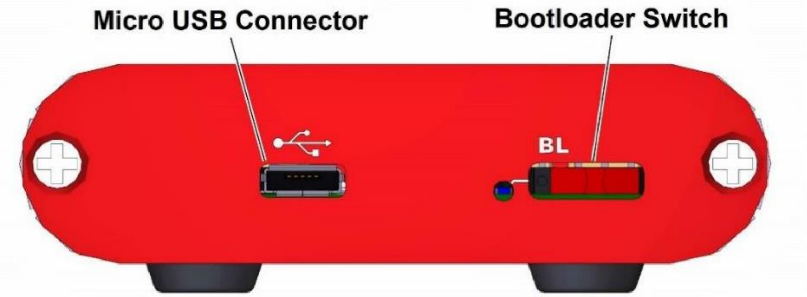
SciCompiler will prompt a confirmation box and then starts to sort out your design into an Hardware Description Language and further proceed to invoke Vivado generating the full bitstream needed to run your design on the FPGA:



As soon as the message `Successful compilation!` appears at the end of the bitstream generation the design has been fully finalized.

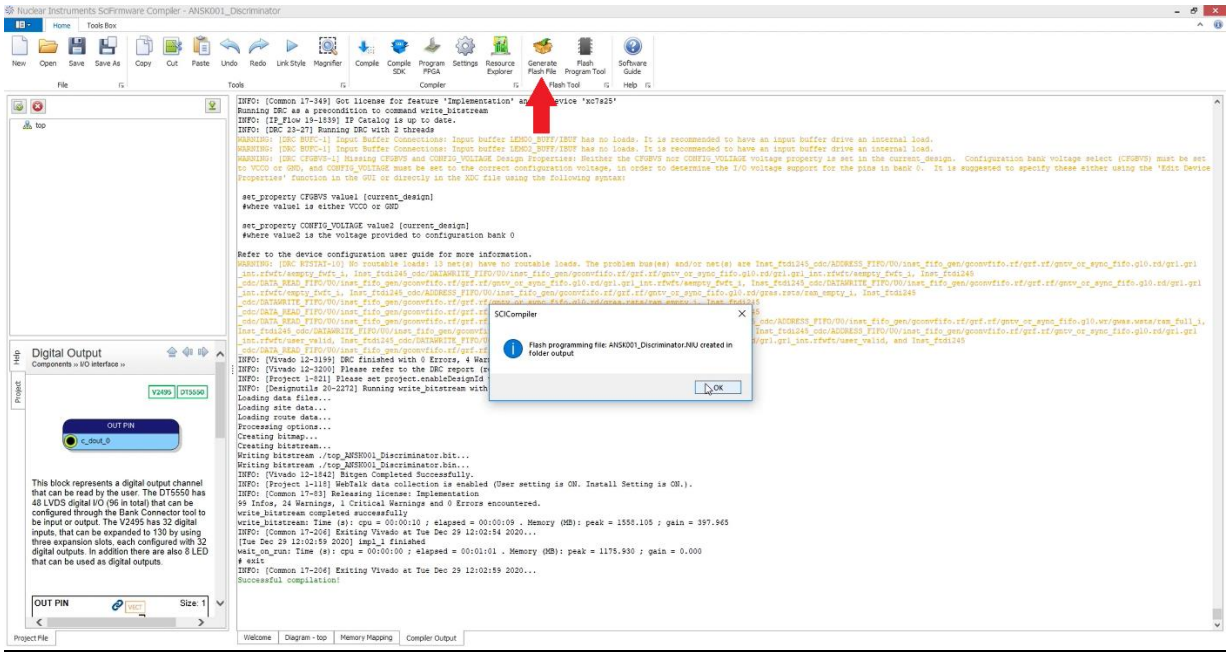
5 Board configuration

Before safely programming the board a few actions are needed:



First unplug the *Micro USB Connector* and then cycle the *Bootloader Switch* to the right. Reconnect the USB cable and check that the *Status Led* located next to the *Bootloader Switch* is no more blinking as when the SciDK is in normal operation but has now become solid blue, indicating the bootloader mode.

Go back to SciCompiler and click *Generate Flash File* , this will generate a FPGA code ready to be installed:



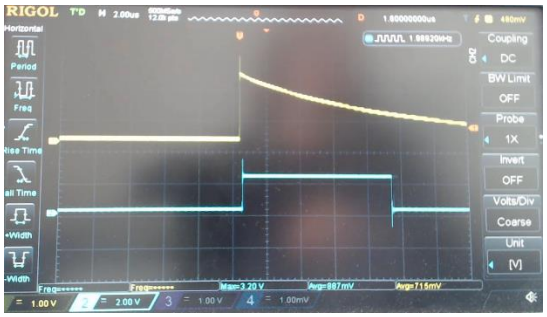
Installing the code

Then select *Flash Program Tool*, just next to *Generate Flash File*. In the next dialog window select SciDK and in the subsequent one check that the serial number matches the intended *SciDK device*; this is especially useful in complex projects. Then the relevant *Firmware File* is already selected, press *Start* to transfer it to the board:

When done you can go back disconnecting the USB cable, cycle again the *Bootloader Switch* and reconnect the cable or just press *Boot Application*. Make sure the blue *Status Led* is blinking, signaling that the procedure has been successfully completed.

6 Real world example

The next image gives you an example of the design working as intended:



The yellow line comes directly from the CAEN DT4800 signal generator while the blue one is drawn by the *LEMO Output* of the SciDK Development Board programmed with SciCompiler and represents the interval in witch the first signal has a value greater than the programmed **Fixed Threshold** of 3000.

This Application note ends here