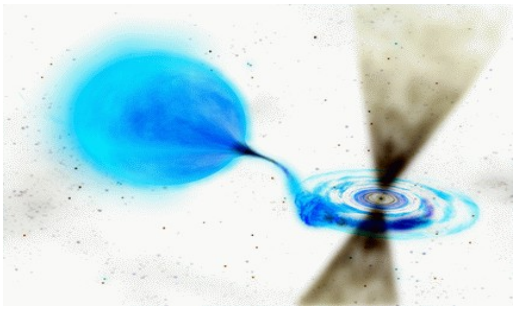


CICADA Note #6
CASPER Tool-flow Development Notes

Brandon Rumberg, Patrick Brandt
NRAO Green Bank
8/9/07

This document is meant to compile some of the things about using the CASPER tool-flow that we didn't find intuitive. By CASPER tool-flow we mean the bee_xps build environment and the MATLAB/Simulink design environment.

This knowledge was gained partially through trial and error, but largely thanks to the help of the folks at CASPER.

Several files are referenced in this document. A summary of these files can be found at:
https://wikio.nrao.edu/bin/viewfile/CICADA/CicadaNotes?rev=1;filename=README.note6

Included are tricks, suggestions, and interpretations of error messages...

Table of Contents

1. Error Messages

Vast numbers of messages are generated during a design build. What follows are interpretations of the error messages that we got hung up on at some point.

1.1 RAM full message

One error that shows up occasionally during the software part of the build deals with the RAM being full. (Or at least this is how we have interpreted it.) If you encounter this error message you should first try changing the compiler optimization level as described in the CASPER memo:

http://seti.berkeley.edu/casper/memos/xps_optimization.pdf
https://wikio.nrao.edu/bin/viewfile/CICADA/CicadaNotes?rev=1;filename=xps_optimization.pdf

To change the compiler optimization level, open the file:

<design_directory>\XPS_iBOB_base\system.xmp

Near the bottom of the file where it says "CompilerOptLevel: 2", change the '2' to a '4'. If you still receive the same error message try taking functions out of the software.

Here is an example of what this error message looks like:

XPS% Evaluating file run_xps.tcl
powerpc-eabi-gcc -O2 Software/main.c Software/tinysh.c drivers/core_util.c
drivers/xps_xsg/clk.c drivers/xps_xsg/devices.c drivers/xps_xsg/memory.c
drivers/xps_adc/adc.c drivers/xps_bram/bram.c drivers/xps_lwip/fifo.c
drivers/xps_lwip/lwipinit.c drivers/xps_lwip/lwiputil.c drivers/xps_sw_reg/reg.c
drivers/core_info.c -o Software/executable.elf
-Wl,-llwip4 -Wl,-T -Wl,Software/LinkerScript.lwip
-I./ppc405_1/include/ -ISoftware/ -Idrivers/ -Idrivers/xps_xsg/
-Idrivers/xps_adc/ -Idrivers/xps_bram/ -Idrivers/xps_lwip/ -Idrivers/xps_sw_reg/
-L./ppc405_1/lib/
-DLWIP_ENABLE
/cygdrive/c/EDK/gnu/powerpc-eabi/nt/bin/../lib/gcc/powerpc-eabi/3.4.1/../../../.
./powerpc-eabi/bin/ld: region plb_bram_if_cntlr_1 is full
(Software/executable.elf section .got2
/cygdrive/c/EDK/gnu/powerpc-eabi/nt/bin/../lib/gcc/powerpc-eabi/3.4.1/../../../.
./powerpc-eabi/bin/ld: section .sdata [ffff0000 -> ffff001f] overlaps section
.text [ffff0000 -> ffffdbaf
/cygdrive/c/EDK/gnu/powerpc-eabi/nt/bin/../lib/gcc/powerpc-eabi/3.4.1/../../../.
./powerpc-eabi/bin/ld: section .boot0 [ffff0020 -> ffff002f] overlaps section
.text [ffff0000 -> ffffdbaf
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x48): In function `plug_holes'
lwip/src/core/mem.c:105: relocation truncated to fit: R_PPC_EMB_SDA21 lfre
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x6c):lwip/src/core/mem.c:94: relocation
truncated to fit: R_PPC_EMB_SDA21 ram_en
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x78):lwip/src/core/mem.c:95: relocation
truncated to fit: R_PPC_EMB_SDA21 lfre
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0xb4):lwip/src/core/mem.c:106:
relocation truncated to fit: R_PPC_EMB_SDA21 lfre

./ppc405_1/lib//liblwip4.a(mem.o)(.text+0xbc):lwip/src/core/mem.c:96: relocation truncated to fit: R_PPC_EMB_SDA21 lfre
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x120): In function `mem_init'
lwip/src/core/mem.c:130: relocation truncated to fit: R_PPC_EMB_SDA21 lfre
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x138):lwip/src/core/mem.c:128: relocation truncated to fit: R_PPC_EMB_SDA21 mem_se
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x13c):lwip/src/core/mem.c:123: relocation truncated to fit: R_PPC_EMB_SDA21 ram_en
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x160): In function `mem_free'
lwip/src/core/mem.c:151: relocation truncated to fit: R_PPC_EMB_SDA21 ram_en
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x174):lwip/src/core/mem.c:165: relocation truncated to fit: R_PPC_EMB_SDA21 lfre
./ppc405_1/lib//liblwip4.a(mem.o)(.text+0x18c):lwip/src/core/mem.c:166: additional relocation overflows omitted from the outpu
collect2: ld returned 1 exit statu
*make: *** [Software/executable.elf] Error*
ERROR:MDT - Error while running "make -f system.make init_bram"
No changes to be saved in MSS file
No changes to be saved in XMP file

C:\brumberg\transient_event\event_cap22\XPS_iBOB_base>copy implementation\download.bit ..\bit_files\event_cap22_2007_Jul_26_1148.bit
The system cannot find the file specified.

1.2 Borph file generation

As of now (7/31/07), there are permissions issues with creating .BOF files. (These are the files that run on the BEE2 from the BORPH environment.) We currently have no solution. Here is what this message looks like:

C:\brumberg\transient_event\bee_test\XPS_BEE2_usr_base>mkbof.exe -o implementation\download.bof -s core_info.tab -v implementation\download.bit
elf=(null), bit=implementation\download.bit, sym=core_info.tab, bof=implementation\download.bof
Error, cannot open tmpfile: Permission denied
elf file open failed

C:\brumberg\transient_event\bee_test\XPS_BEE2_usr_base>copy implementation\download.bof ..\bit_files\bee_test_floating_2007_Jul_10_1645.bof
The system cannot find the file specified.

This message can be ignored when building for the iBOB, but is critical for BEE2 designs.

1.3 Timing Errors

Timing usually is not an issue when the clock speed is down around 100 Mhz, but when it is increased up to and beyond 200 Mhz it becomes a big headache.

If your build stops and says it can't build because the constraints for an individual block can't be met then the latency for some block needs to be increased. One thing to keep in mind is that greater bit widths (8 vs. 32 bits) require more latency. (Like an adder that needs to carry bits from the operations on less significant bits.) Sometimes a block is a critical part of a feedback loop and adding latency messes up the loop. We have dealt with this issue by reducing the precision of that block until the design will build. Some blocks don't have latency parameters like counters and accumulators. If you need a big counter and get timing errors refer to this:

http://direct.xilinx.com/bvdocs/appnotes/xapp023.pdf
https://wikio.nrao.edu/bin/viewfile/CICADA/CicadaNotes?rev=1;filename=xapp023.pdf

Once System Generator has finished the PlaceAndRoute (PAR) portion of the build, you can see your timing summary. Ideally there should not be any timing errors and the score should be zero. If this is not the case, the Xilinx Timing Analyzer is helpful. Please refer to section 3 of this document for more information on the Timing Analyzer.

Example (Bad) Timing Summary:

Timing summary:
---------- ---

Timing errors: 72 Score: 20476

NOTE: Data paths and computation blocks should not use more bits than required. For example, if the largest possible value for the inputs of a multiplier is 8 bits but the inputs are allowed to be 16 bits, then the multiplier will be 32 bits when 16 bits was all that was required.

Another thing to consider is that the software registers are by default 32 bits. If all 32 bits are not required then the unnecessary bits should be sliced off immediately after the register to avoid propagating unnecessary bits through the design.

Keeping this in mind will reduce the number of logic slices used and reduce the number of physical wires on the chip needed to route the data.

1.4 No sequential components.

Sometimes it helps to build small sections of a design to work out the timing errors without having to wait for the whole design to build. When doing this you may encounter this error:

MATLAB Seg Fault
No Counter
ERROR:NgdBuild:76 - File
"C:/brumberg/xaui_ex/XPS_iBOB_base/implementation/xaui_ex_xsg_core_config_wra
pper/xaui_ex.ngc" cannot be merged into block "xaui_ex_xsg_core_config"
(TYPE="xaui_ex") because one or more pins on the block, including pin "clk",
were not found in the file. Please make sure that all pins on the
instantiated component match pins in the lower-level design block
(irrespective of case). If there are bussed pins on this block, make sure
that the upper-level and lower-level netlists use the same bus-naming
convention.

This error occurs when there are no sequential components. The solution is to add an isolated counter to the design.

2. Simulink Tricks

Simulink is a part of MATLAB that is used to draw and simulate the designs.

2.1 "From Workspace" variables

Simulink offers several inputs: sine waves, step functions, noise, etc. But you may wish to simulate the design using a specific data set. We had some trouble loading simulated data from the workspace until we configured the GUI as follows.

If you have data in a file, then you can read it in using a Matlab script and save the data in a Simulink *"From Workspace"* variable. Included is an example Matlab script called *data_in.m:*

https://wikio.nrao.edu/bin/viewfile/CICADA/CicadaNotes?rev=1;filename=data_in.m

This script reads input from *data.file:*

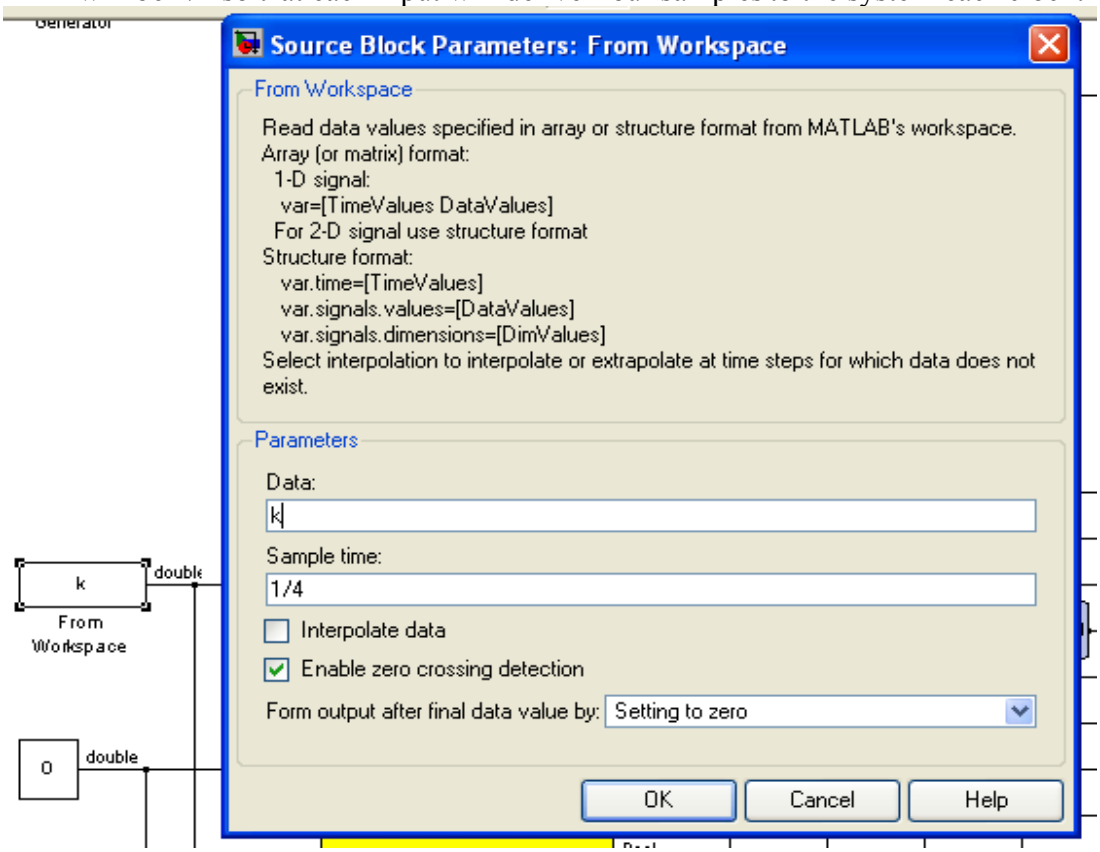
https://wikio.nrao.edu/bin/viewfile/CICADA/CicadaNotes?rev=1;filename=data.file

The data file has three columns and no header information. The script saves the second column to a workspace using the Simulink format. Of course you can also create your own data set using Matlab.

The parameters when you double click on the from workspace variable should be set to:

Data: the variable that has your signal in it

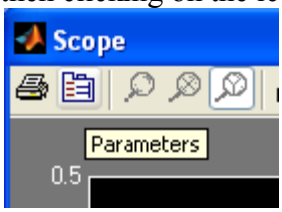
Sample time: If the input is not going into an ADC you will probably want the sample time to be 1. If it is going into an interleaved ADC you will want the sample time to be 1/8 so the system will get eight samples a clock. If the ADC is not interleaved then the sample time will be 1/4 so that each input will deliver four samples to the system each clock.



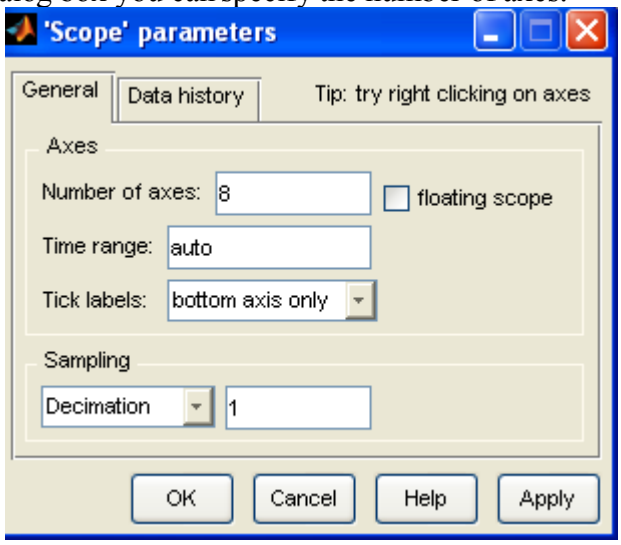
If you receive errors using from workspace variables, we know that it works with the above parameters.

2.2 Scope stuff: multiple inputs, holding all the data points

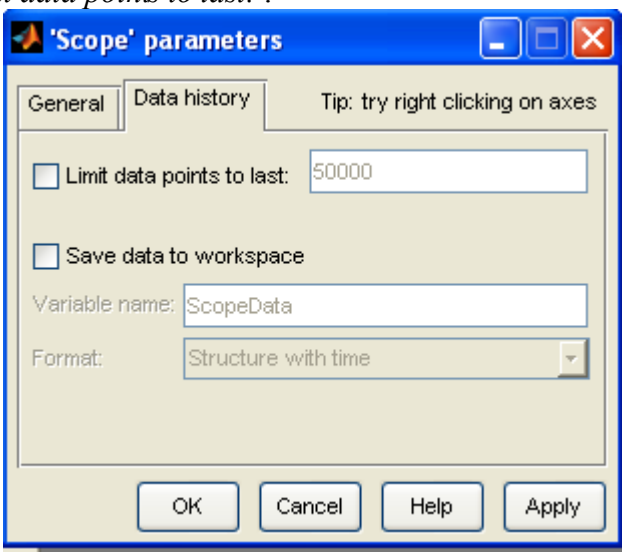
It is often helpful to have a Simulink scope show multiple signals. You can specify the number of inputs to a scope by double clicking on it then clicking on the icon in the top left next to the printer icon.



In the *'Scope' parameters* dialog box you can specify the number of axes.

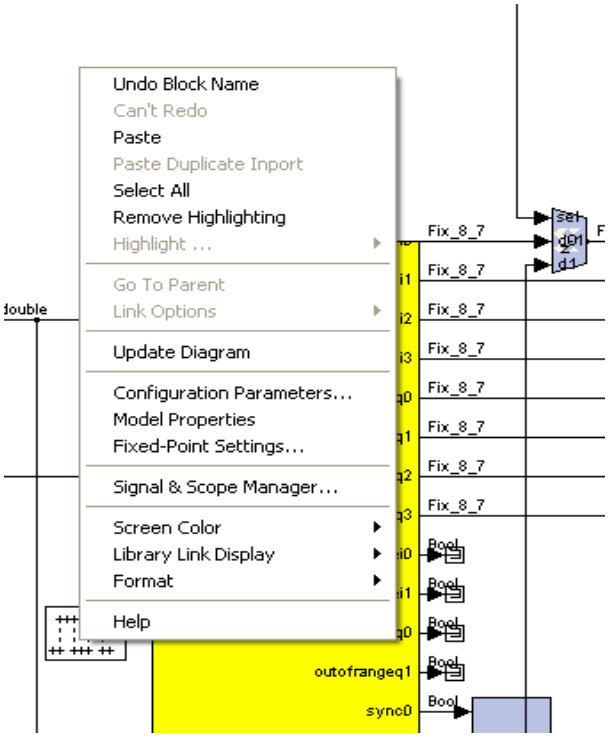


Also, if you are running a long simulation you will likely want to click on the *Data history* tab and un-check the box that says *Limit data points to last:* .

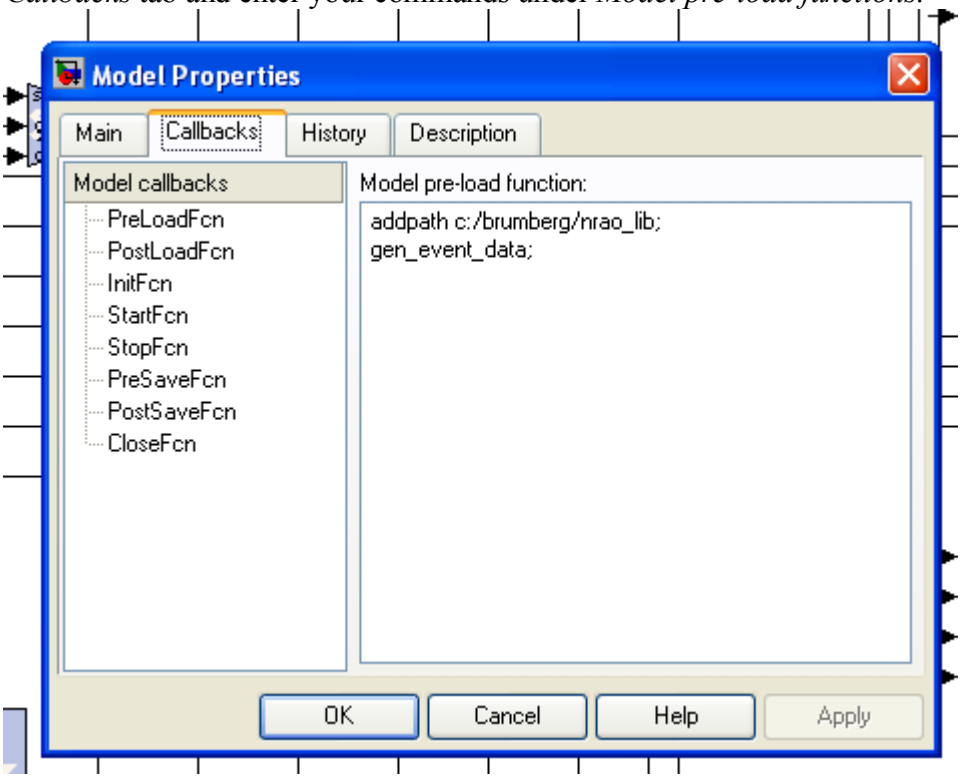


2.3 Automatically run files when opening a design

You can have Matlab commands run automatically every time you open a design. This is helpful if the simulation input is generated by a Matlab function or if you want to add the path to a library used by the design. With the design open, right click on a blank part of the design and choose *Model Properties* from the list.



Then go to the *Callbacks* tab and enter your commands under *Model pre-load functions*.



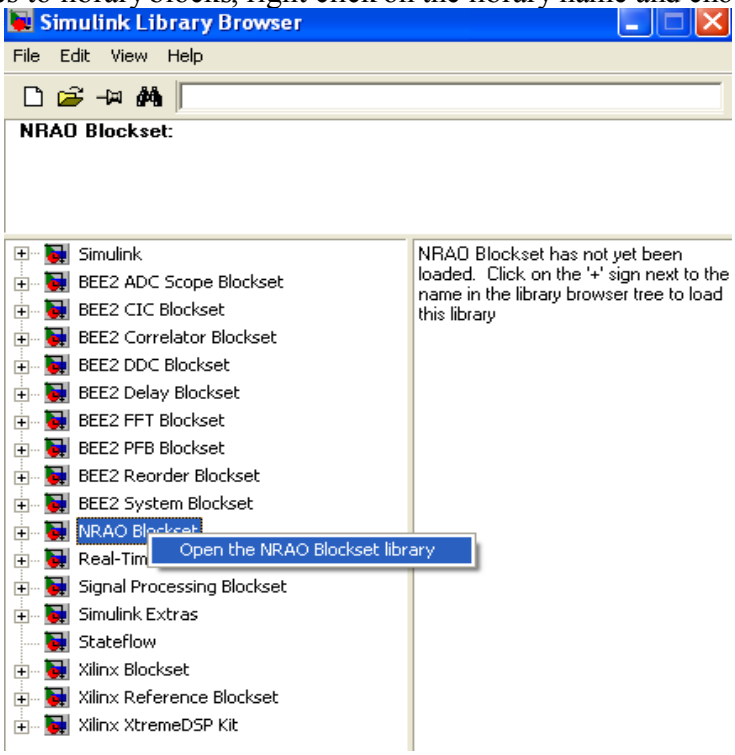
2.4 Creating a library

To create a library, first save a model file with a masked subsystem in it. In the same directory with this model file, place the slblocks.m file:

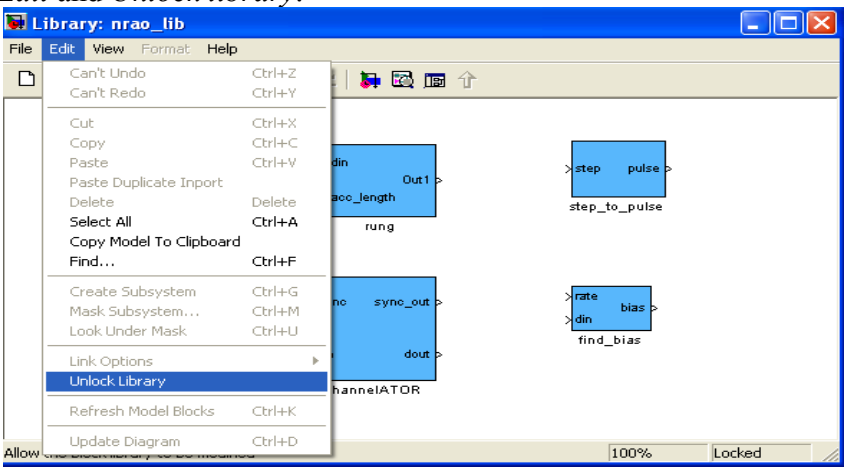
https://wikio.nrao.edu/bin/viewfile/CICADA/GcadaNotes?rev=1;filename=slblocks.m

Edit slblocks.m according to the comments in the file. Now, at the Matlab prompt enter *addpath <path_to_library_file>*.

To make changes to library blocks, right click on the library name and choose *Open blockset.*



Then go to *Edit* and *Unlock library*.



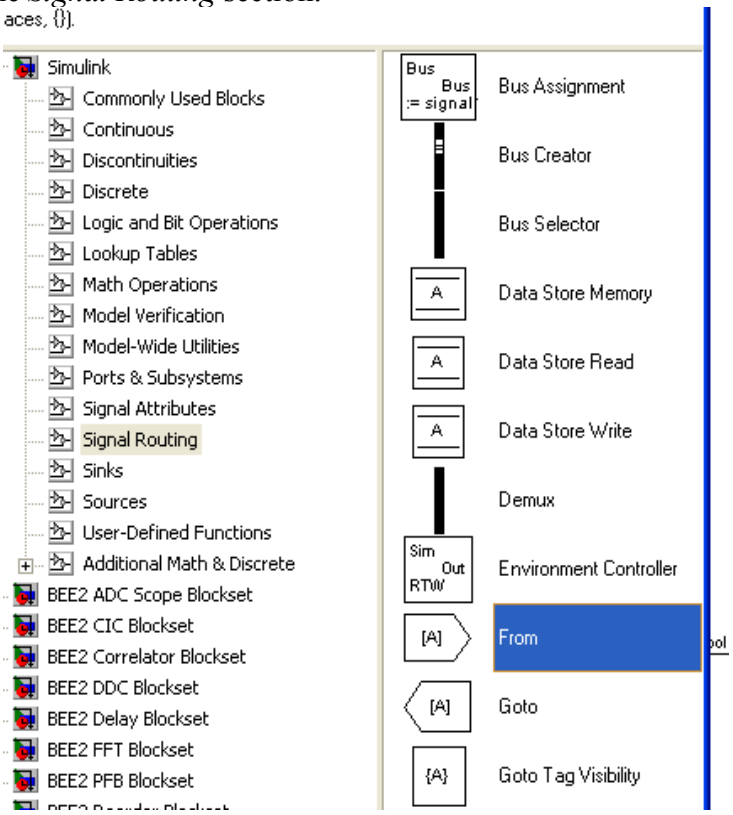
Now you can edit the blocks.

If you would like to add blocks to a library, unlock the library as described above and drag a block into the library.

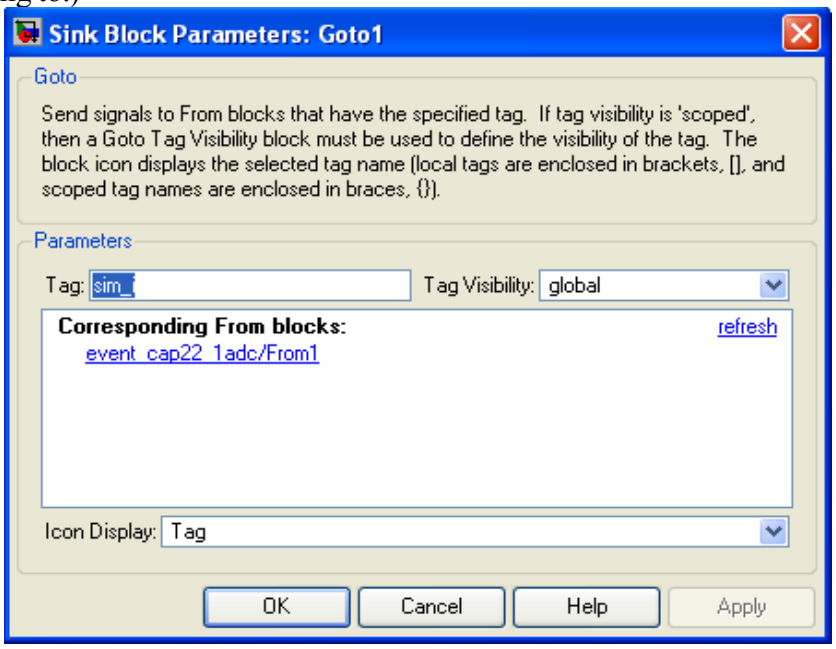
NOTE: A subsystem must be masked to properly add it to a library. Subsystem masking is well documented in the Simulink Help Browser.

2.5 From/goto blocks

Often one must route signals from one side of a design to the other, crossing many other signal paths. These extra lines confuse the design layout. These lines can be hidden using the Simulink blocks *From* and *Goto* under the *Signal Routing* section.

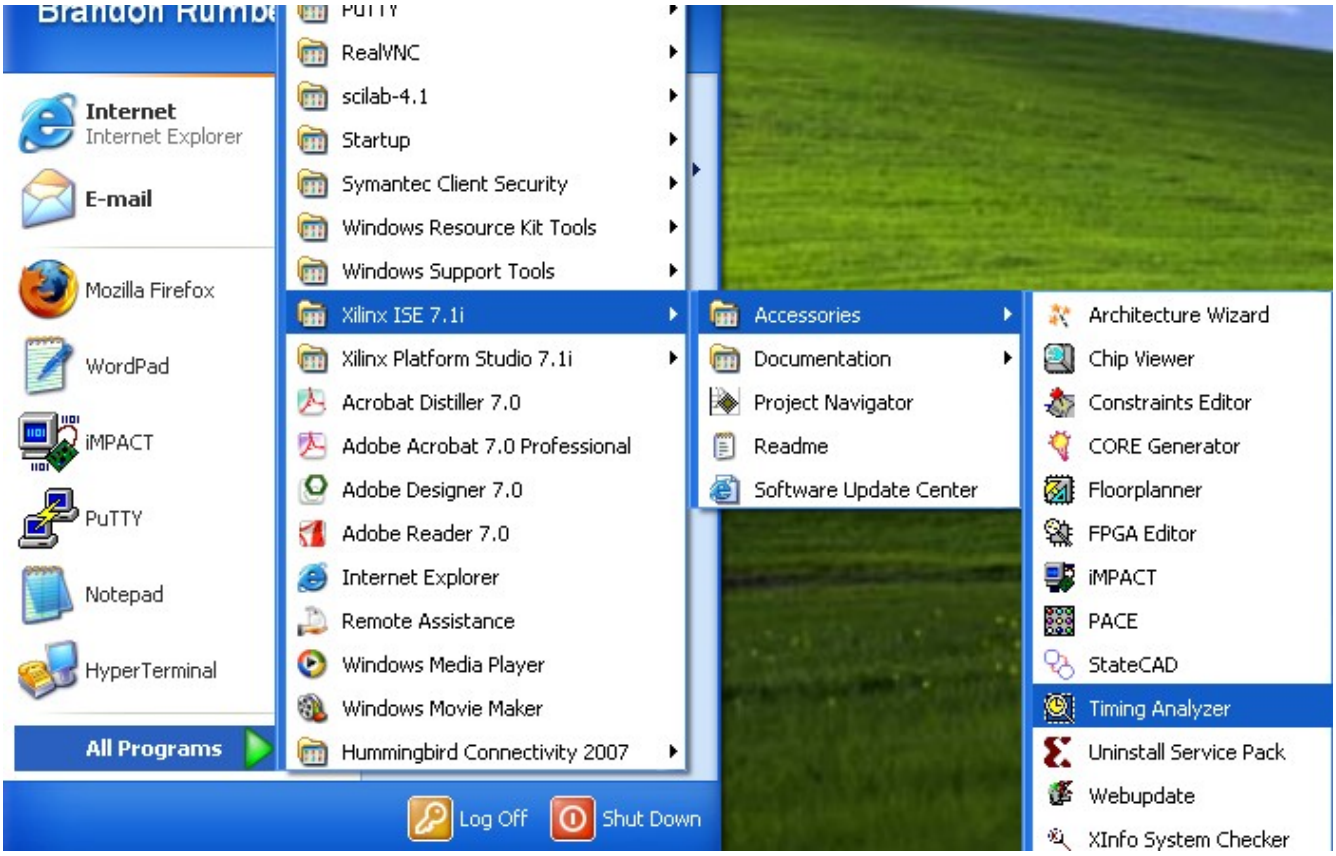


By setting the *Tag Visibility* (in the mask parameters) to global you can connect parts in different subsystems. (It is recommend including a note next to from/goto blocks, saying what subsystem they are coming from or going to.)



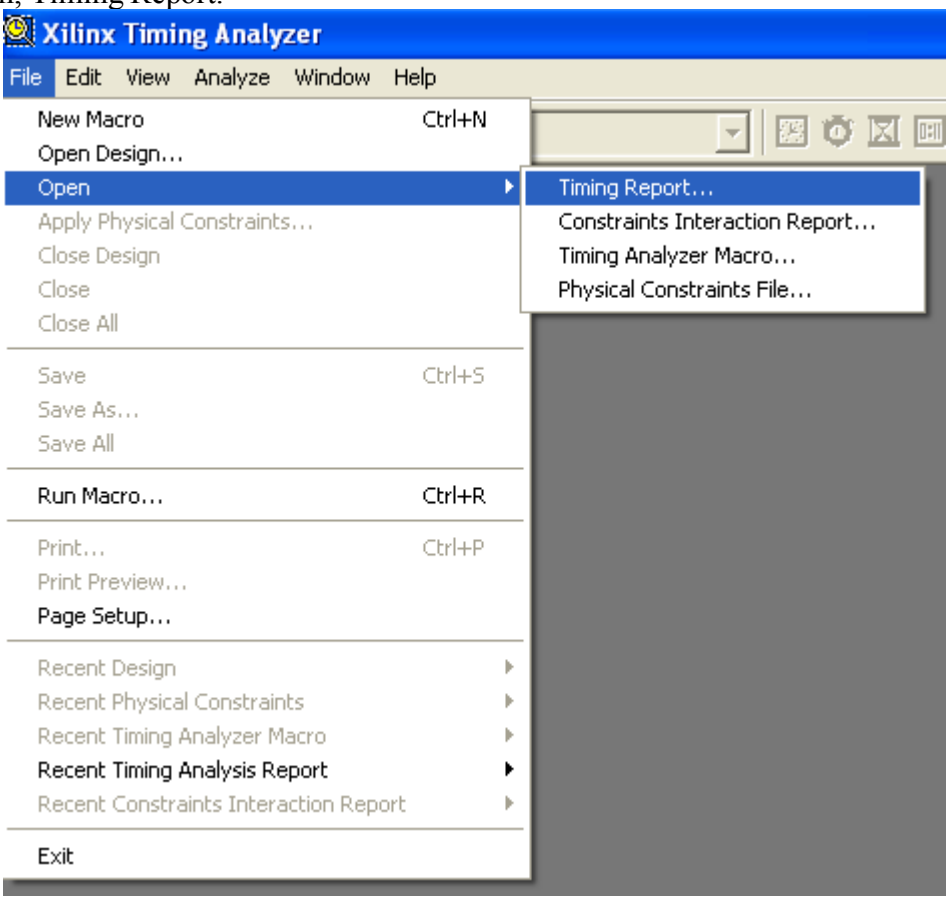
3. Timing Analyzer

Xilinx Timing Analyzer is useful to find the maximum routing and logic delays in your design to track down timing issues. You can run the timing analyzer by going to the start menu, Xilinx ISE, Accessories, Timing Analyzer.



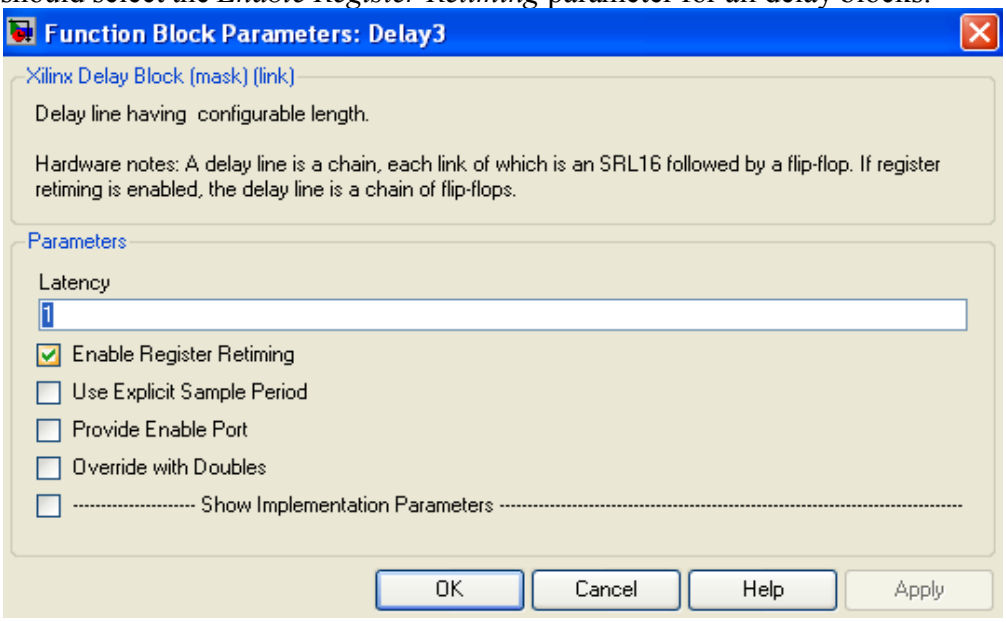
These delays must always be shorter than the timing constraints which are on the order of a few clock cycles. Designs that take a very long time to build (more than 5 hours) often have timing issues. When these timing issues are resolved the build time decreases dramatically.

Then File, Open, Timing Report.



And navigate to *<design directory>\XPS_iBOB_base\implementation\system.twx*. Now you can see what part of the design is causing timing errors. This ASCII file will show the routes with time in excess of the constraint. We found that adding delay blocks between paths causing problems would usually fix those problems. The build and modify process was repeated until there were no timing errors.

NOTE: You should select the *Enable Register Retiming* parameter for all delay blocks.



4. Clock speeds for the ADC and iBOB

The ADC clock should always be 4 times the desired system clock. If the ADC is not used in interleave mode then the FPGA will receive four samples for each clock for each channel. If the ADC is used in interleave mode then one input is sampled on both the rising and falling edge of the clock. Thus the FPGA will receive eight samples per clock for one channel.

5. Notes for coding iBOB software

There are a few nuances when it comes to programming C code to compile onto the iBOB's Power PC. Firstly, when connecting through either the ethernet or serial link to the IBOB (the serial connection was made in our tests using the Berkeley written "*openPort()*" function), if you do not send an escape code specifying the type of connection, Tinyshell will not send any command prompts or command echoes. To enable these replies for TCP, send the value '255' to the IBOB immediately following the *connect()* call. For more information on how the serial link was setup, see the code file for *openPort*.

We used the escape code so that Tinyshell would send the "IBOB %" prompt when a command finished. This way we could key off the prompt to know when the iBOB was finished sending data.

There are two ways to do both general I/O and data transmission from the iBOB to a PC. The first is the *xil_printf()* function. This function behaves exactly as the normal printf(). The second is *outbyte()*, which simply outputs (or sends) an 8 bit character. The *xil_printf* ultimately calls *outbyte*, so it may be more efficient to use *outbyte* directly, although the overall limiting factor for transmission speed is most likely Tinyshell itself. The main difference between *xil_printf* and *outbyte* is data type; *xil_printf* transmits in ASCII, thus it will appear on the screen as printable characters, whereas the *outbyte* function sends in pure binary. Both functions will send to either the serial port or the ethernet port (a check is done upon calling to determine which interface is active).

It is also worth noting that the iBOB Tinyshell is not threaded, so only one connection is supported at one time. Multiple simultaneous connections can be made, but the behavior is unpredictable since it only has a single input buffer (so any character typed on either connection is placed in the same buffer).