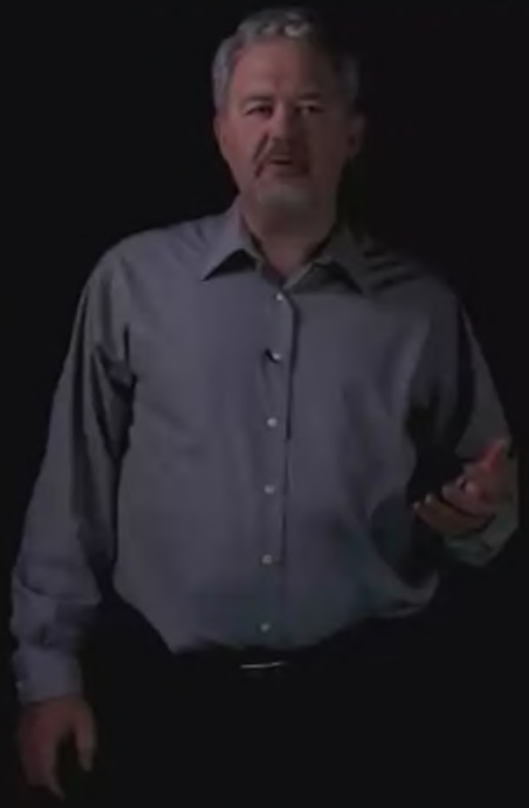
FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

Software for Soft Processors – Series Introduction



- Software Interface to Soft Processor
- Software Design Flow
- Nios II Software Build Tools for Eclipse
- Programming the FPGA with Software

Software Interface to Soft Processors

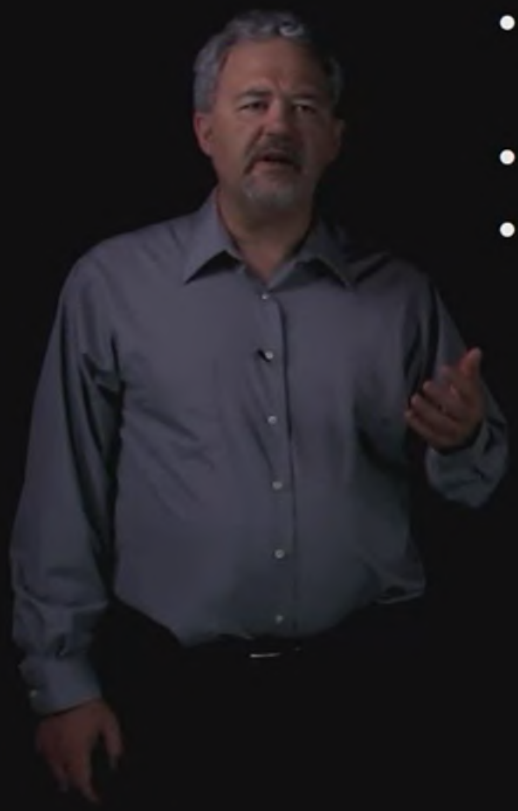
Custom soft processors provide code development support through active tool flows.

- GNU based C/C++ compilers and tool chain
- JTAG interface to live processor core
- Program, Run, Debug

Software development flow based on industry standards.

- Utilities, scripts, device drivers
- Operating Systems, Network Stacks

Software Interface to Soft Processors



Tool Flow : Integrated Development Environment (IDE)

- Hardware Abstraction Layer (HAL)
  - American National Standards Institute (ANSI) C standard libraries (e.g., stdio.h, math.h)
- GNU (GCC) compiler, assembler, linker, and makefile facilities
- Flash Program
  - Data
  - Software
  - FPGA Image

Videos in this module

1. Series Introduction -
2. Software for Soft Processors
3. Introduction to the IDE
4. C programming for Soft Processors
5. Building C programs in the IDE
6. Programming the Software
7. Memory in the Nios II
8. Accessing Custom instructions

https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

https://www.intel.com/content/www/us/en/products/programmable/processor/nios-ii/design-tools.html

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

Software for Soft Processors

In this video, you will learn:

- The range of software development tools available for Intel Altera FPGAs, and which to use for NIOS II soft processor.
- A description of the NIOS II EDS and NIOS II SBT for Eclipse, the IDE we will use for software development.

Software for Soft Processors

Intel Altera provides several software development tools for both hard and soft processors

- Intel SoC EDS (Embedded Development Suite) for ARM hard-core processors featuring ARM DS5
- Intel EDS (Embedded Design Suite) for NIOS II, featuring the NIOS II SBT for eclipse
- Altera Monitor Program, a free tool from the Altera University program

Software Development Tools

NIOS II Software Build Tools (SBT) for Eclipse. Eclipse is an open-source industry standard Integrated Development Environment (IDE).

- Software developers can create and debug applications without further knowledge of FPGAs.
- The SBT automates creation of a Board Support Package (BSP) for the C/C++ interactive runtime environment. GNU compilers are used (GCC)
- Using the BSP libraries, developers can create code to communicate with any Nios II processor system.
- The SBT creates the project makefiles for code builds.

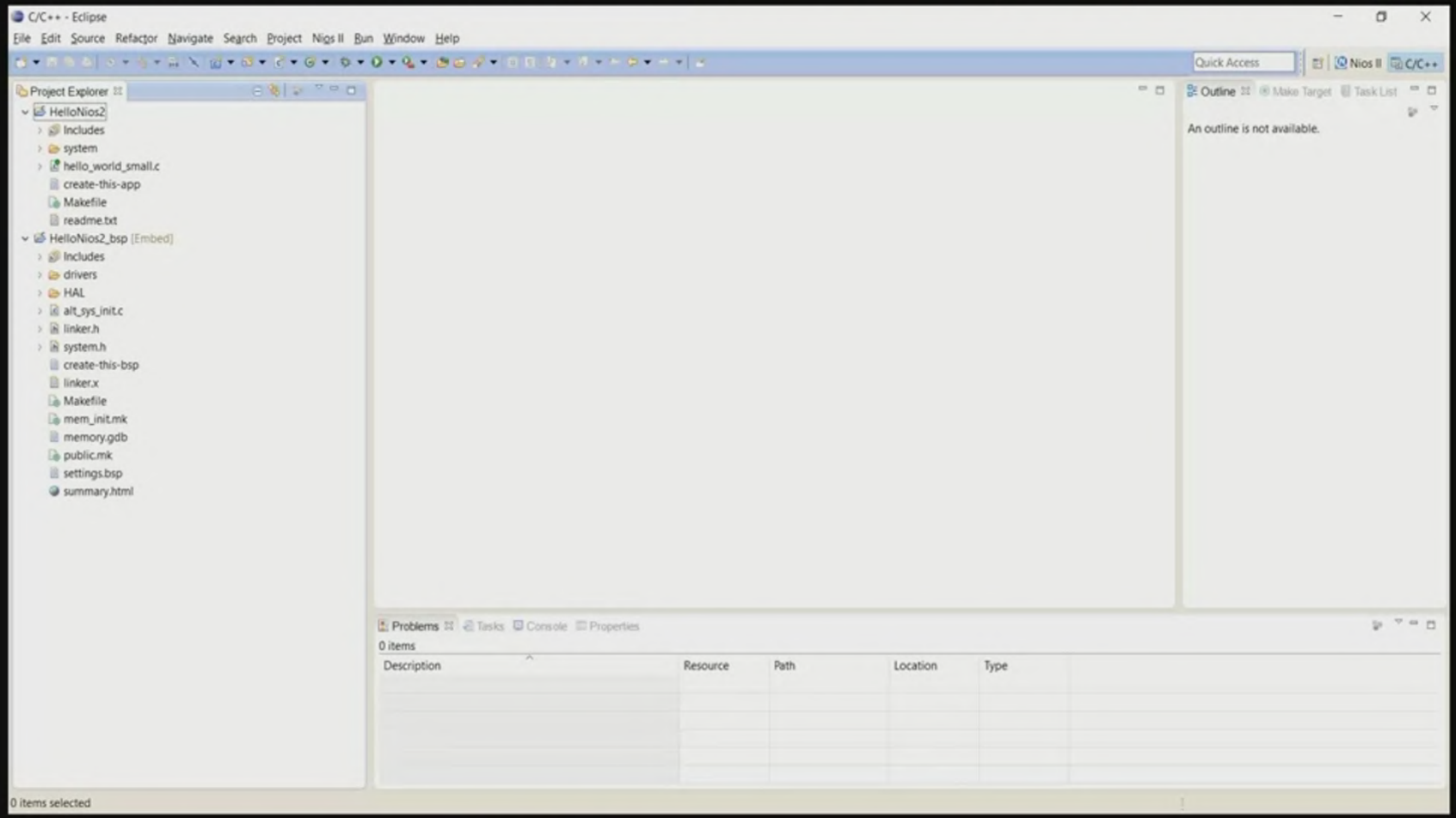
https://www.eclipse.org/

Software Development Tools

- Eclipse is a popular integrated development environment (IDE) used for embedded programming.
- The Eclipse open source Software Development Kit (SDK) workspace provides :
  - A plugin system for C/C++ source code editing, compiling, debugging, and live terminal. GNU tools provide the compiler, linker and assembler.
  - A GUI workbench window with perspectives. Each perspective provides a set of capabilities for accomplishing a specific type of task, like debugging.
- Eclipse is the IDE of choice for RISC-V soft processor development

Note : Starting with NIOS II version 19.1 Eclipse requires manual installation.
Follow the Eclipse manual in n2sw_nii5v2gen2.pdf

View of NIOS II Software Build Tools (SBT) for Eclipse



Summary

In this video, you have learned:

- Which tools are available for Intel Altera FPGAs, and which to use for NIOS II soft processor.
- A description of the NIOS II EDS and NIOS II SBT for Eclipse, the IDE we will use for software development.

References

[1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

[2] Intel/Altera Staff. (2020/Feb/20), *Altera Monitor Program Tutorial for Nios II.* [Online]. Available: ftp.intel.com › fpgaup › pub › Intel_Material › Tutorials

[3] Intel/Altera Staff. (2020/Jul/11), *Embedded Design Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

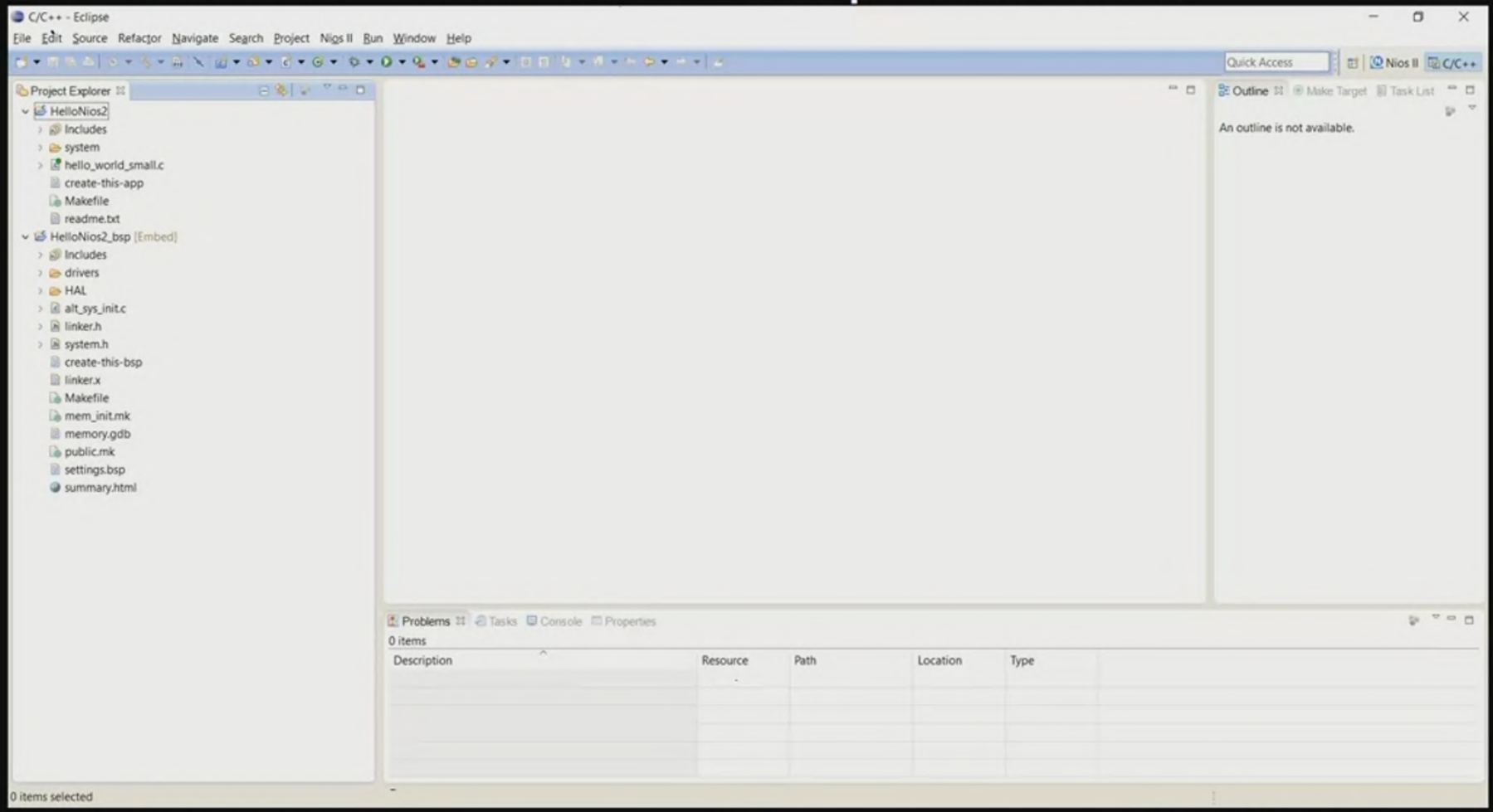
University of Colorado **Boulder**

Introduction to the NIOS IDE

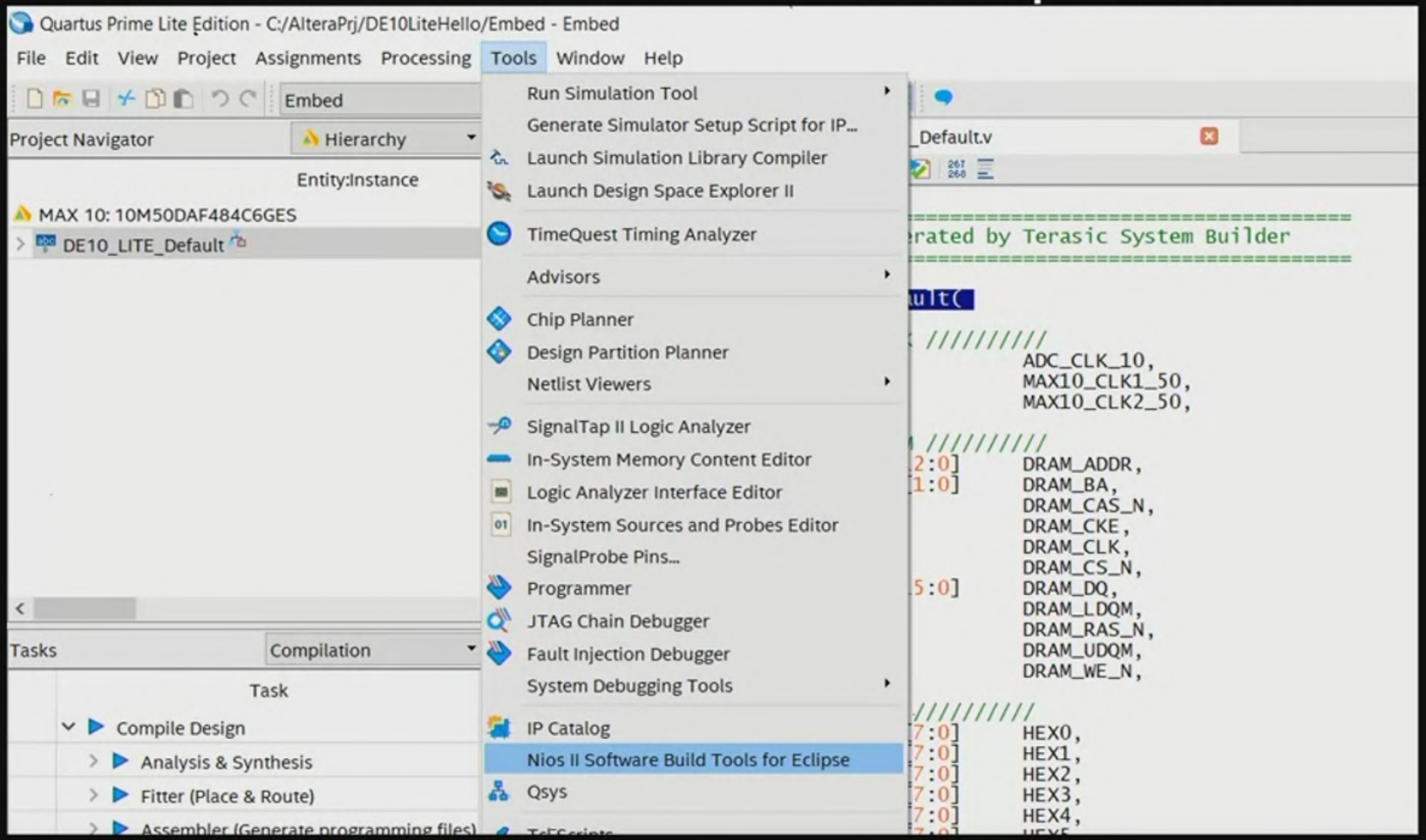
In this video, you will learn:

- How the Eclipse IDE appears using different perspectives.
- What views are shown in the C/C++ and Nios II perspectives.
- How to use the Eclipse IDE to start a software project.

The NIOS II SBT IDE - Eclipse for NIOS

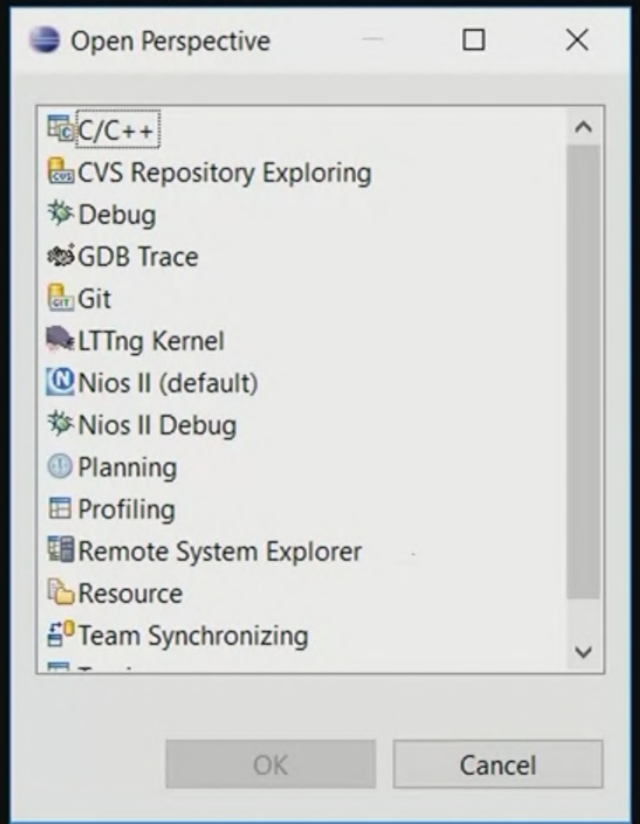


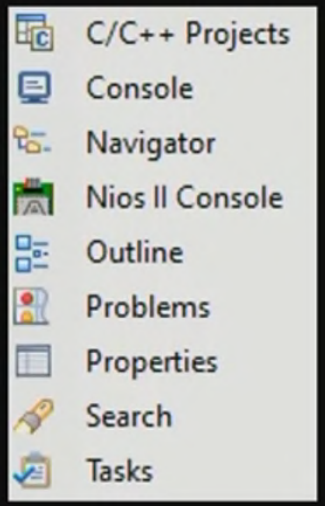
The NIOS II SBT IDE - Eclipse for NIOS



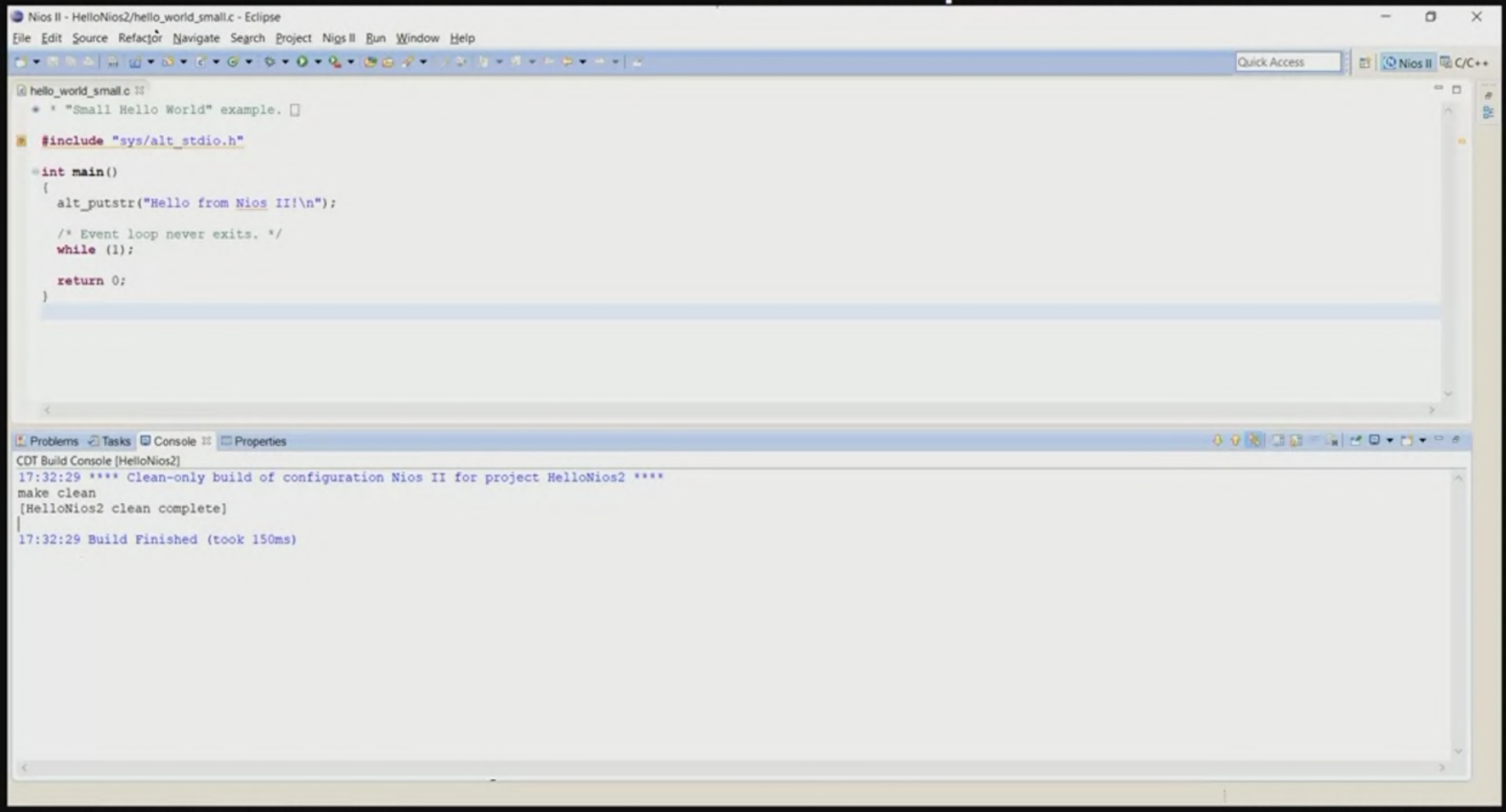
The NIOS II SBT IDE

- To get to the Nios II perspective : Window (menu) -> Open Perspective -> Nios II
- Perspectives Views

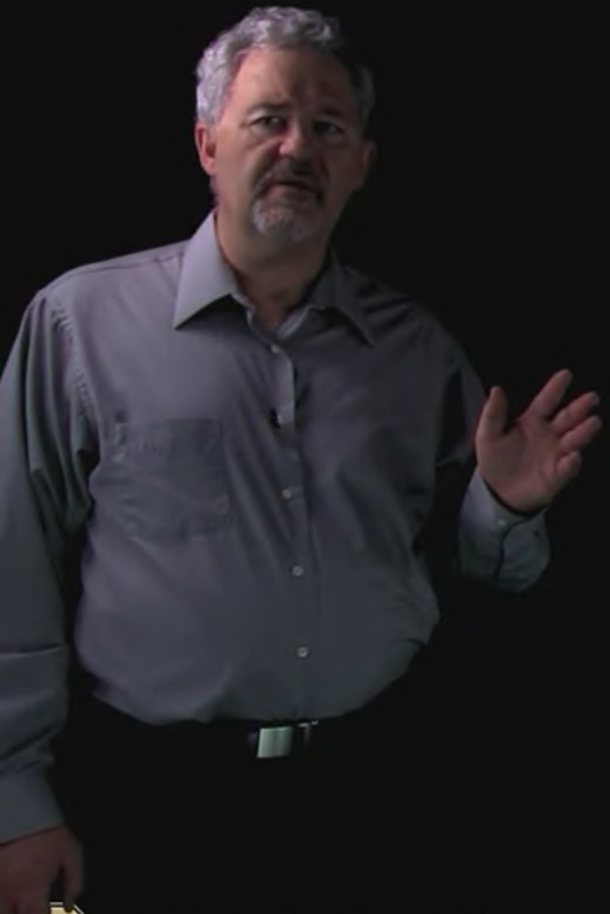




The NIOS II SBT IDE - Eclipse for NIOS



Eclipse for NIOS

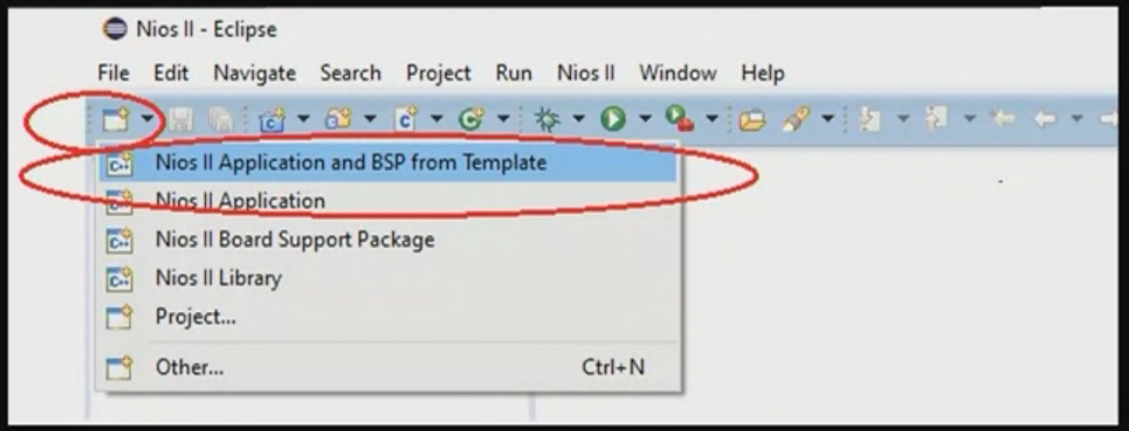


- Eclipse provides a Workbench :
  - A **perspective** is a group of views and editors in the Workbench window. Each perspective contains one or more views and editors. Within a window, each perspective may have a different set of views but all perspectives share the same set of editors.
  - A **view** is a visual component within the Workbench. It is typically used to navigate a list or hierarchy of information (such as the resources in the Workbench), or display properties for the active editor. Modifications made in a view are saved immediately.
  - An **editor** is also a visual component within the Workbench. It is typically used to edit or browse a resource

https://help.eclipse.org/2020-06/index.jsp

Starting a Software Project

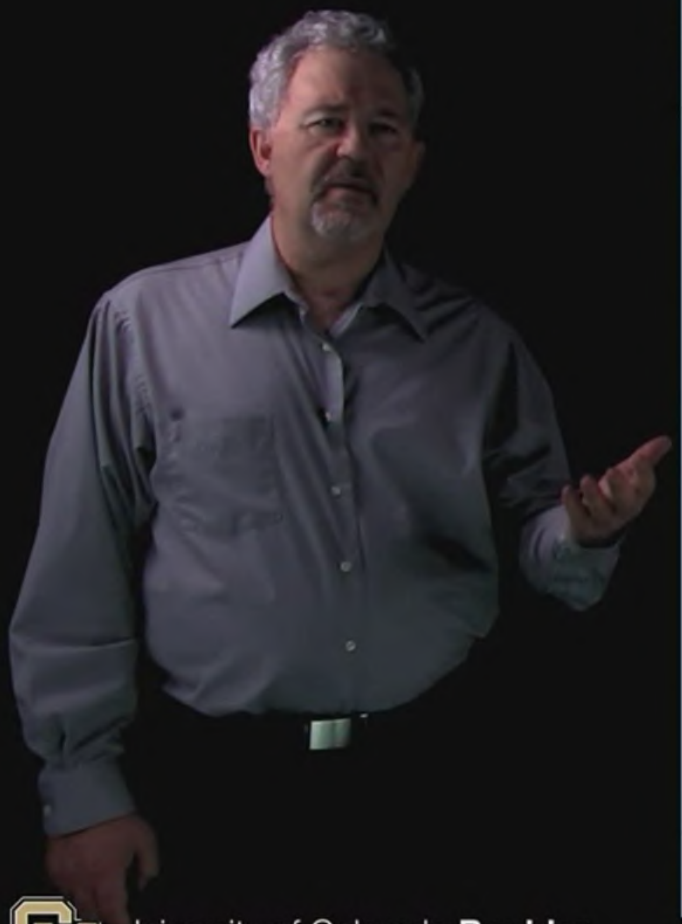
- From top menu, File -> New -> Nios II Application and BSP From Template, or from the icon:

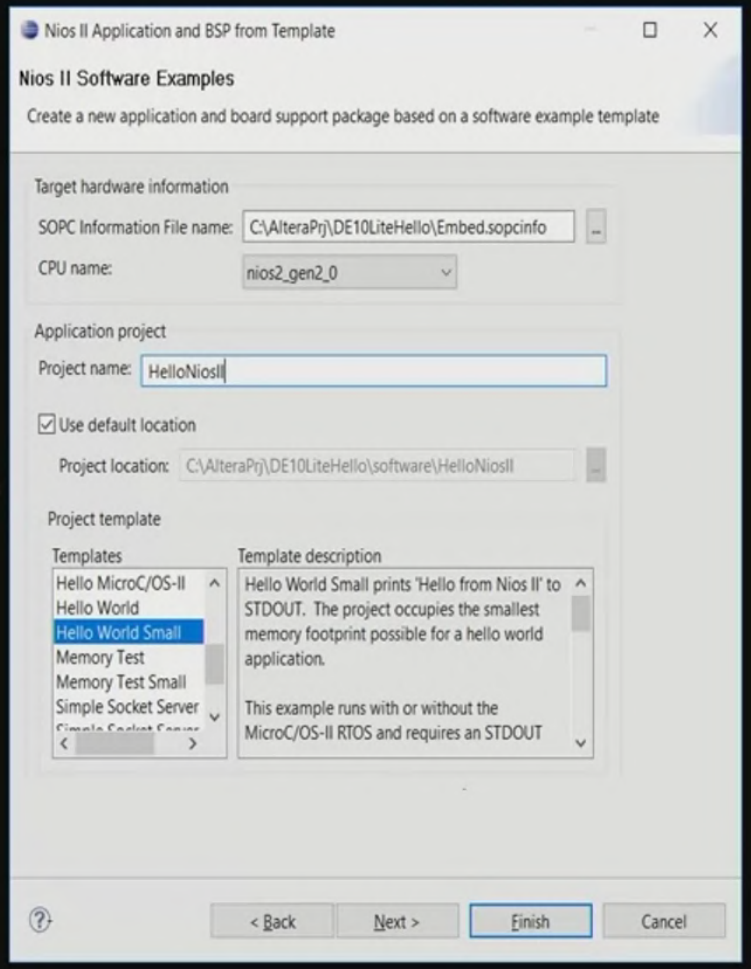


https://help.eclipse.org/2020-06/index.jsp

University of Colorado **Boulder**

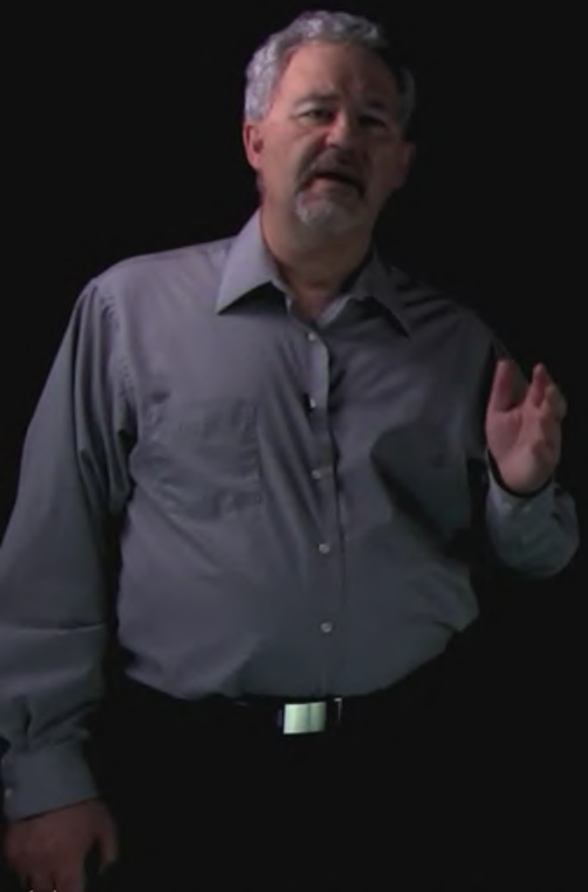
Starting a Software Project





- Sopcinfo file was created by Qsys to give system information, memory map to the software tools.
- Choose Template appropriate for your project
- Click Finish and your project is started!

Summary for the NIOS IDE



In this video, you have learned:

- How the Eclipse IDE appears using different perspectives.
- What views are shown in the C/C++ and Nios II perspectives.
- How to use the Eclipse IDE to start a software project.

References

[1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

[2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an717.pdf

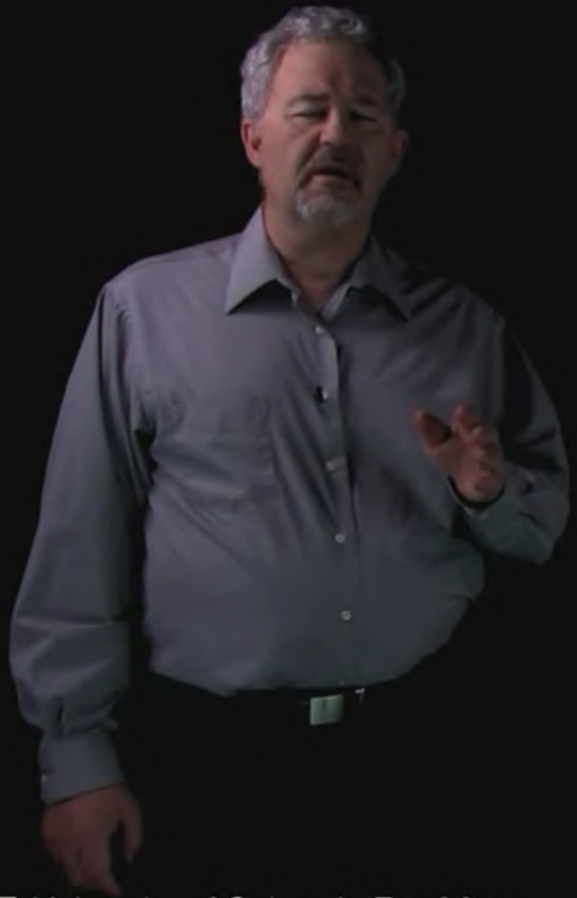
[3] Intel/Altera Staff. (2020/Jul/11), *Embedded Design Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

C programming for Soft Processors



In this video, you will learn:

- How Embedded C programming differs from application programming
- How the challenges of C programming for soft processors are met by provision of the BSP.
- How to use BSP Editor to control and customize soft processor software.

C programming for Soft Processors

Embedded Programming Unique Challenges

...Different from Application programming

	Embedded Programming	Applications Programming
1	Memory – Little Control of Memory Usage is Critical Stack size, Heap Size, Code Size, allocation, Memory Map	Memory – Lots. Not a concern in many cases. Don't care how much memory my word processor uses
2	Power Control of Power Consumption is Critical, especially in battery operated systems.	Power – Not usually a concern, except in data centers now
3	I/O Control of hardware essential. Setup of peripherals, timers, etc. Manage bandwidth, latency, speed of access.	I/O - No worries, Hardware is taken care of by OS
4	Speed Control of speed and efficient code critical in real-time systems	Speed – Don't care how fast my word processor is, it mostly waits for key press.

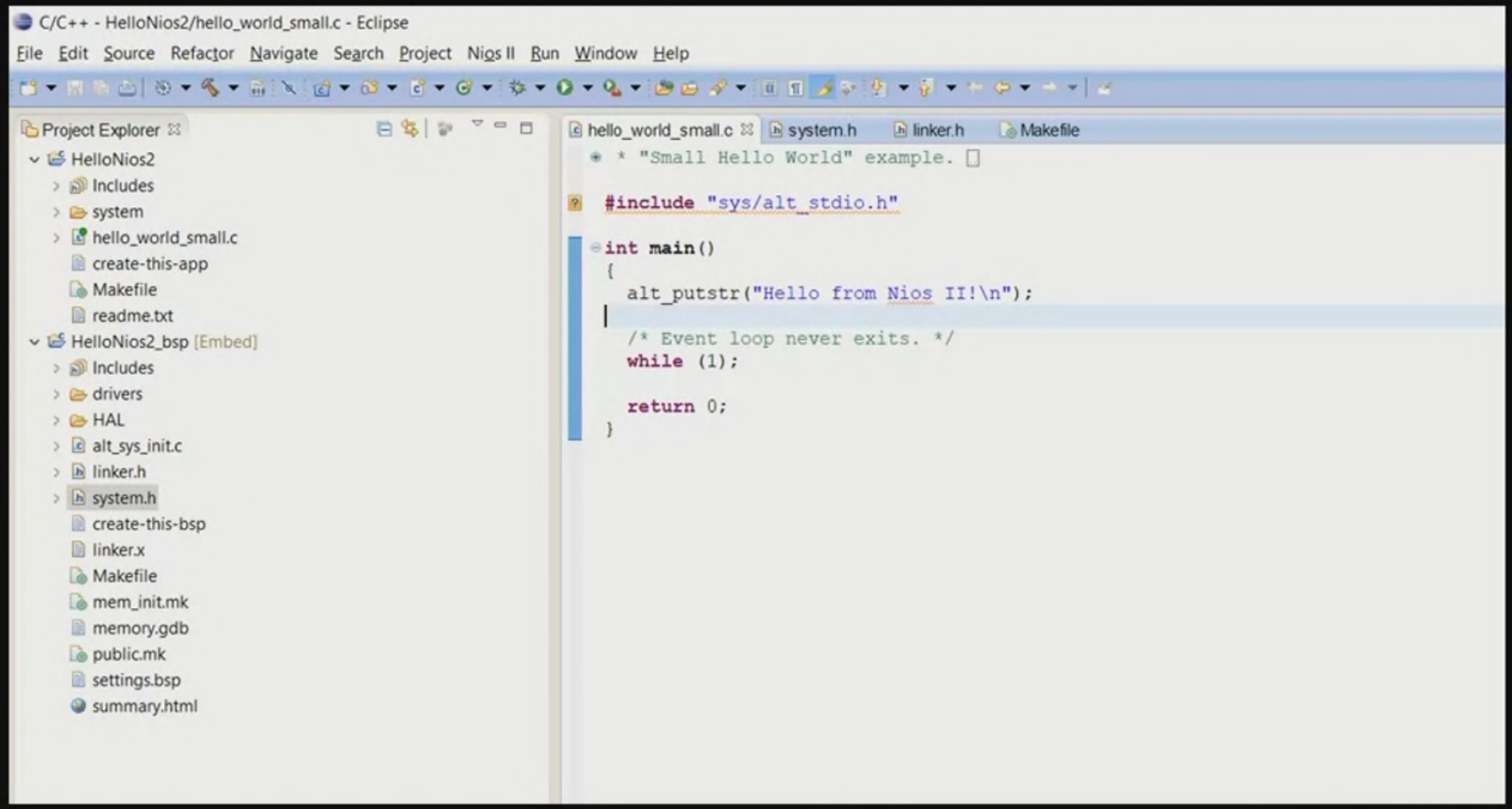
C programming for Soft Processors

Embedded Programming Unique Challenges

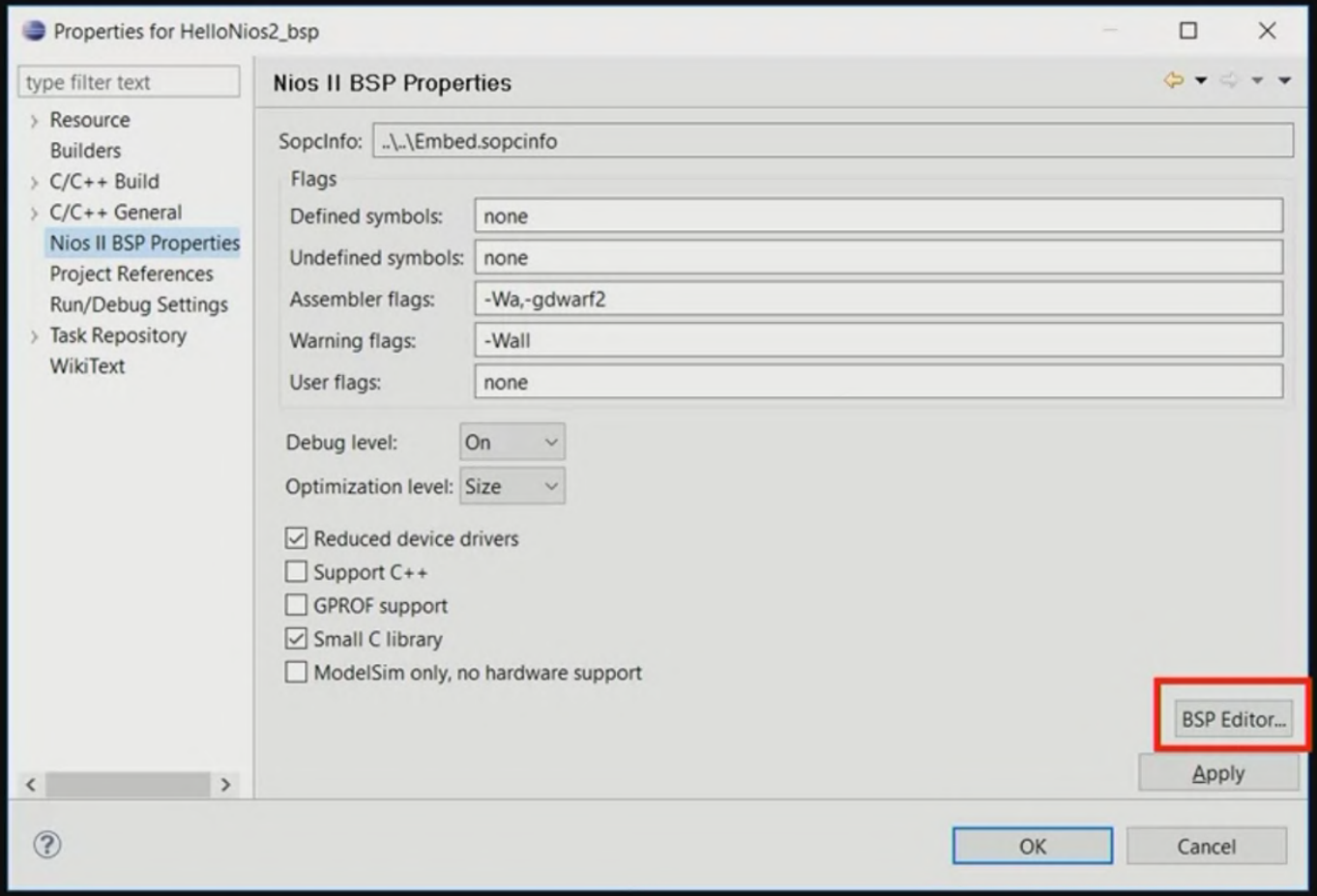
...Different from Application programming

	Embedded Programming	Applications Programming
5	Flow of Control Must handle Exceptions, Interrupts	Flow of Control – OS concern
6	Security – important to get this right to assure higher levels not vulnerable	Security – A concern, but techniques to achieve are different
7	Scope Visibility of Variables, Functions, etc. must be controlled	Scope – Fine control not as important
8	Timing Meeting timing deadlines, priorities must be controlled	Timing – OS, not an applications concern
9	Compiler Performance Know your compiler. How it performs can be tuned, careful with optimization level	Compiler Performance – Can be important, but not critical.

BSP is key to software system control

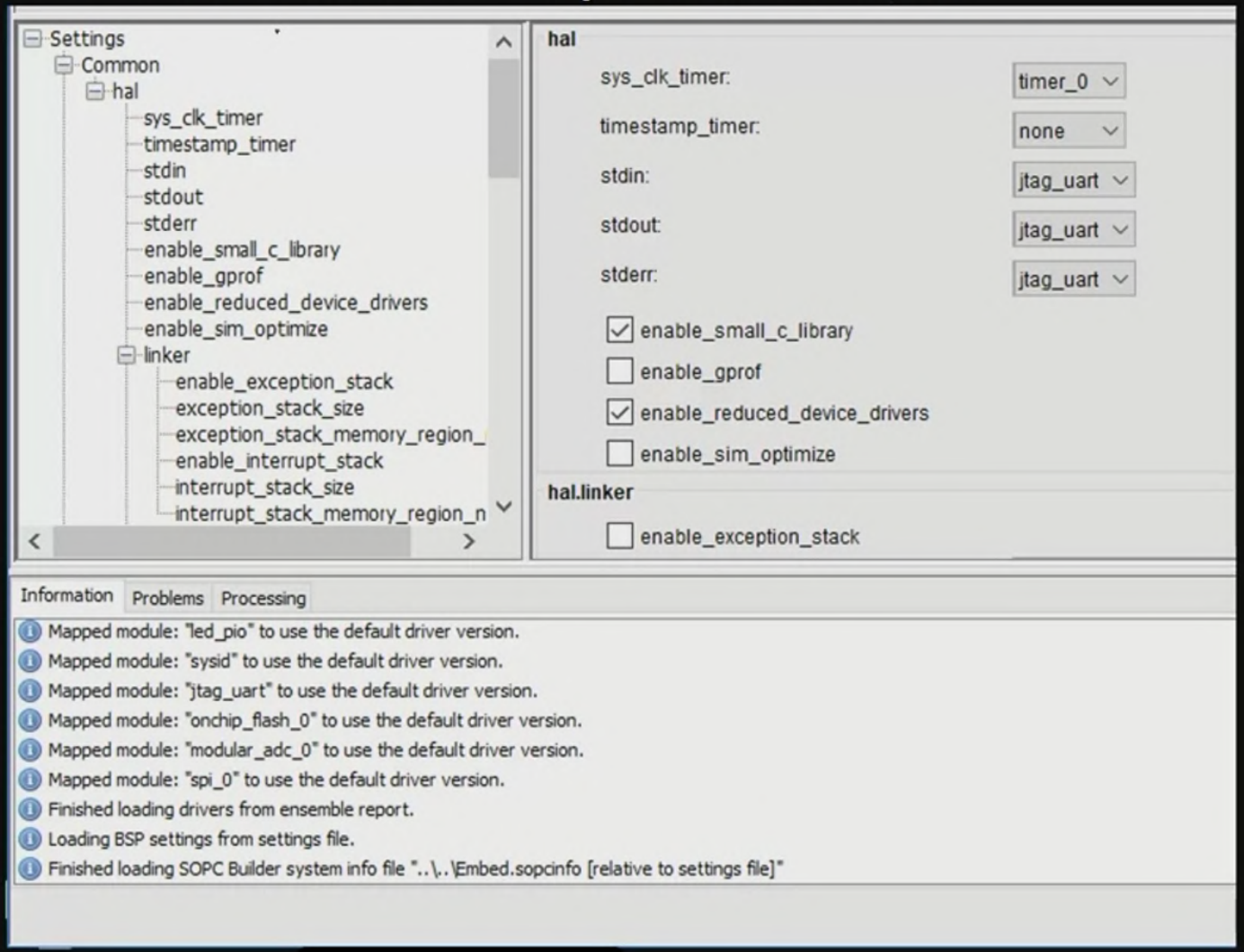


BSP Properties



- Right Click on the BSP project in the Project Explorer window and scroll all the way down to properties.
- This gives you access to the BSP Editor button on the lower right

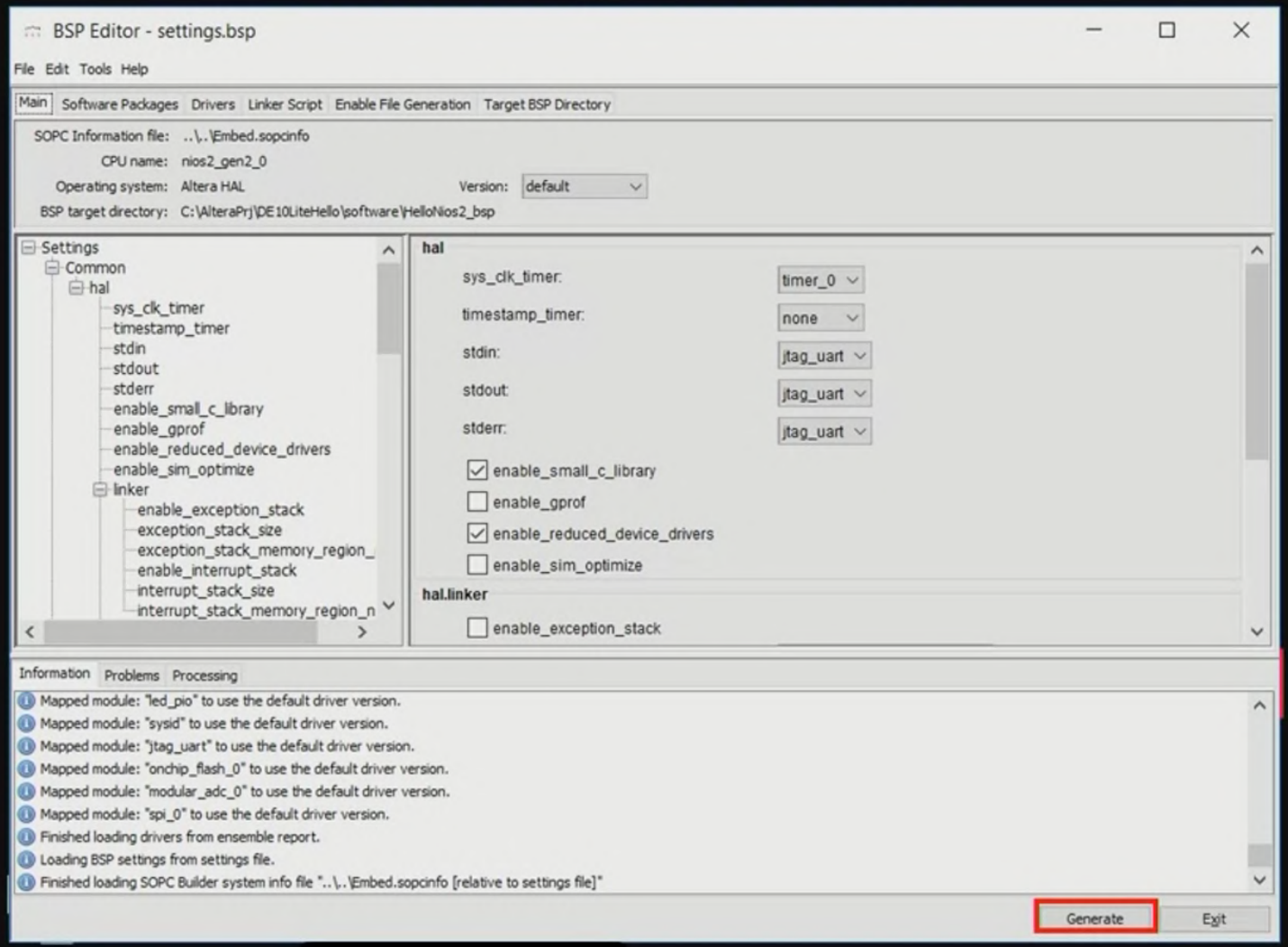
BSP Editor – a powerful tool



With the BSP Editor, the programmer can

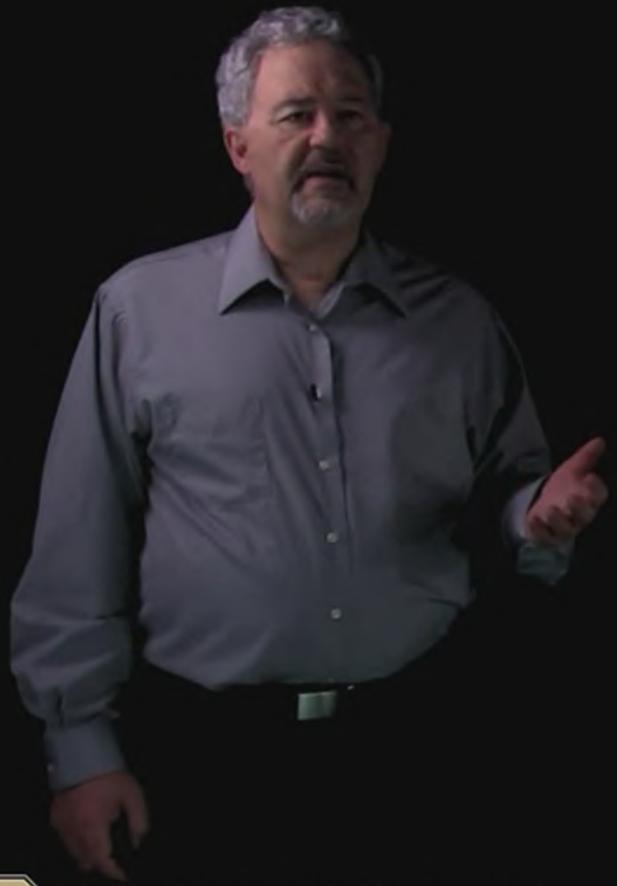
- Control system timers
- Specify stderr/in/out (in this case to the jtag uart which will show on the console)
- Control code size (small library and drivers chosen here)
- Enable and size stacks

BSP Editor – a powerful tool



After making any changes to the BSP, click Generate and the BSP will be updated with code size and options you have chosen.

Summary for Soft Processor C Programming



In this video, you have learned:

- How Embedded C programming differs from application programming
- How the challenges of C programming for soft processors are met by provision of the BSP.
- How to use BSP Editor to control and customize soft processor software.

References

[1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

[2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an717.pdf

[3] Intel/Altera Staff. (2020/Jul/11), *Embedded Design Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

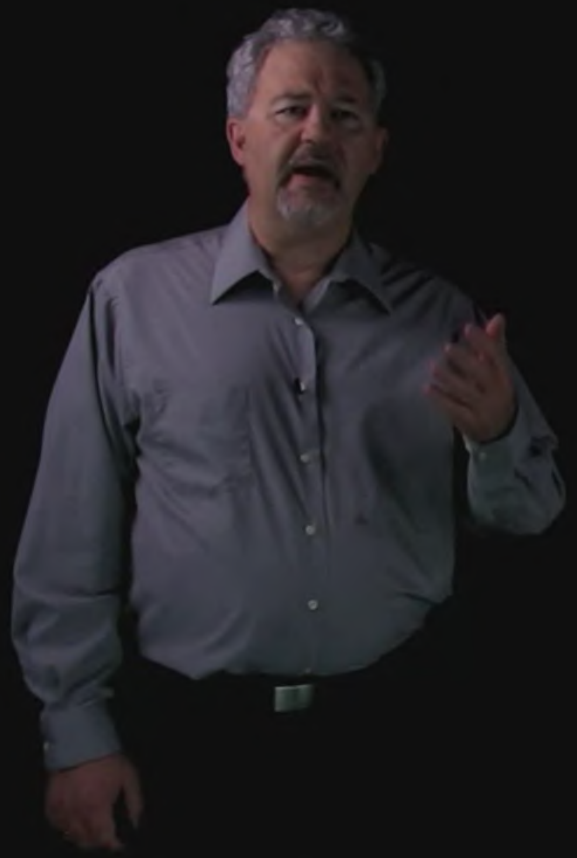
[4] Intel/Altera Staff. (2020/Jul/11), *MAX 10 Nios II Embedded "Hello World" Lab: DE10-Lite Development Kit.* [Online]. Available: https://fpgacloud.intel.com/devstore/platform/16.1.0/Standard/nios-ii-qsys-hello-world-lab-max10-de10-lite/

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

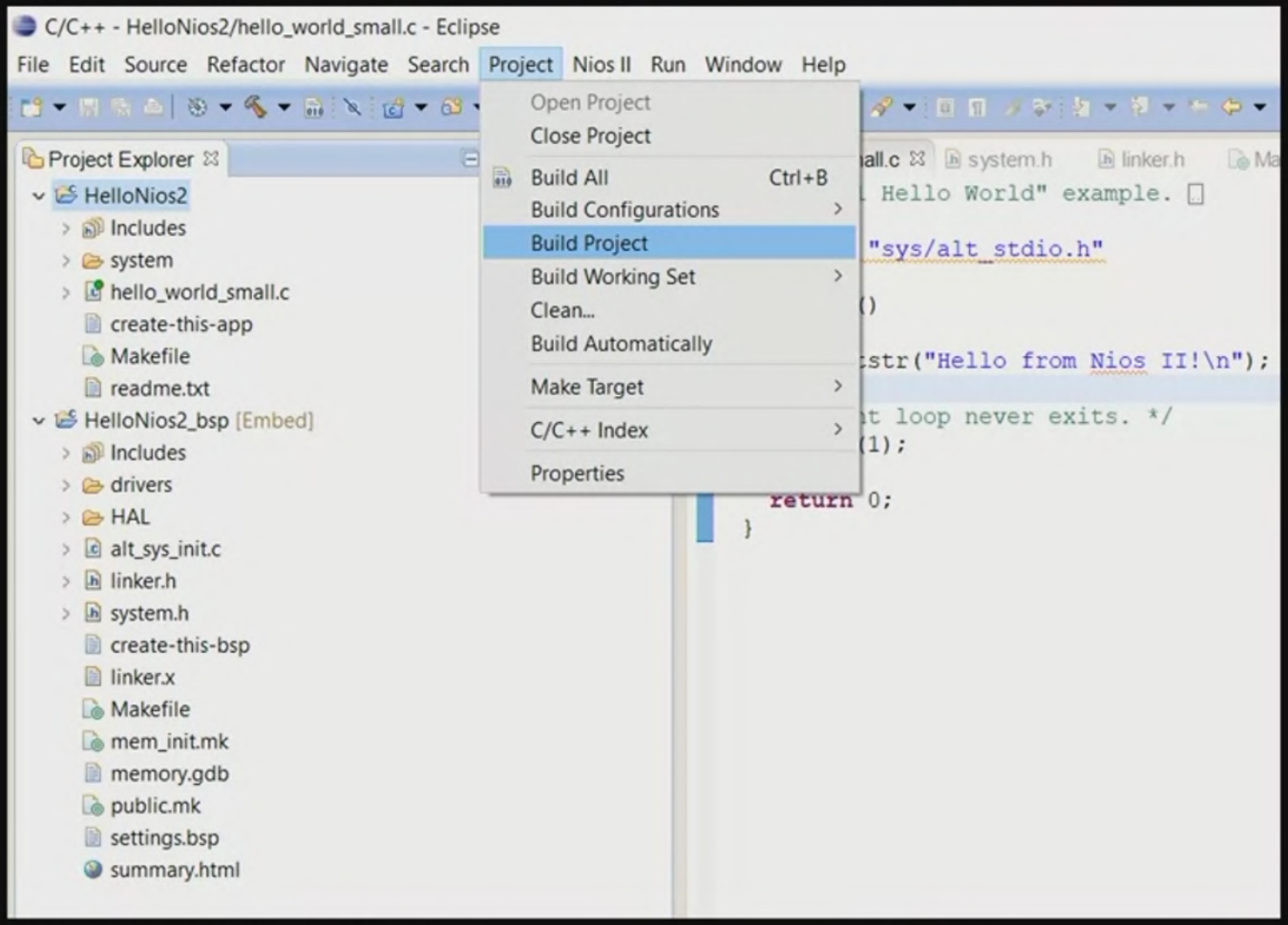
Building C Programs



In this video, you will learn:

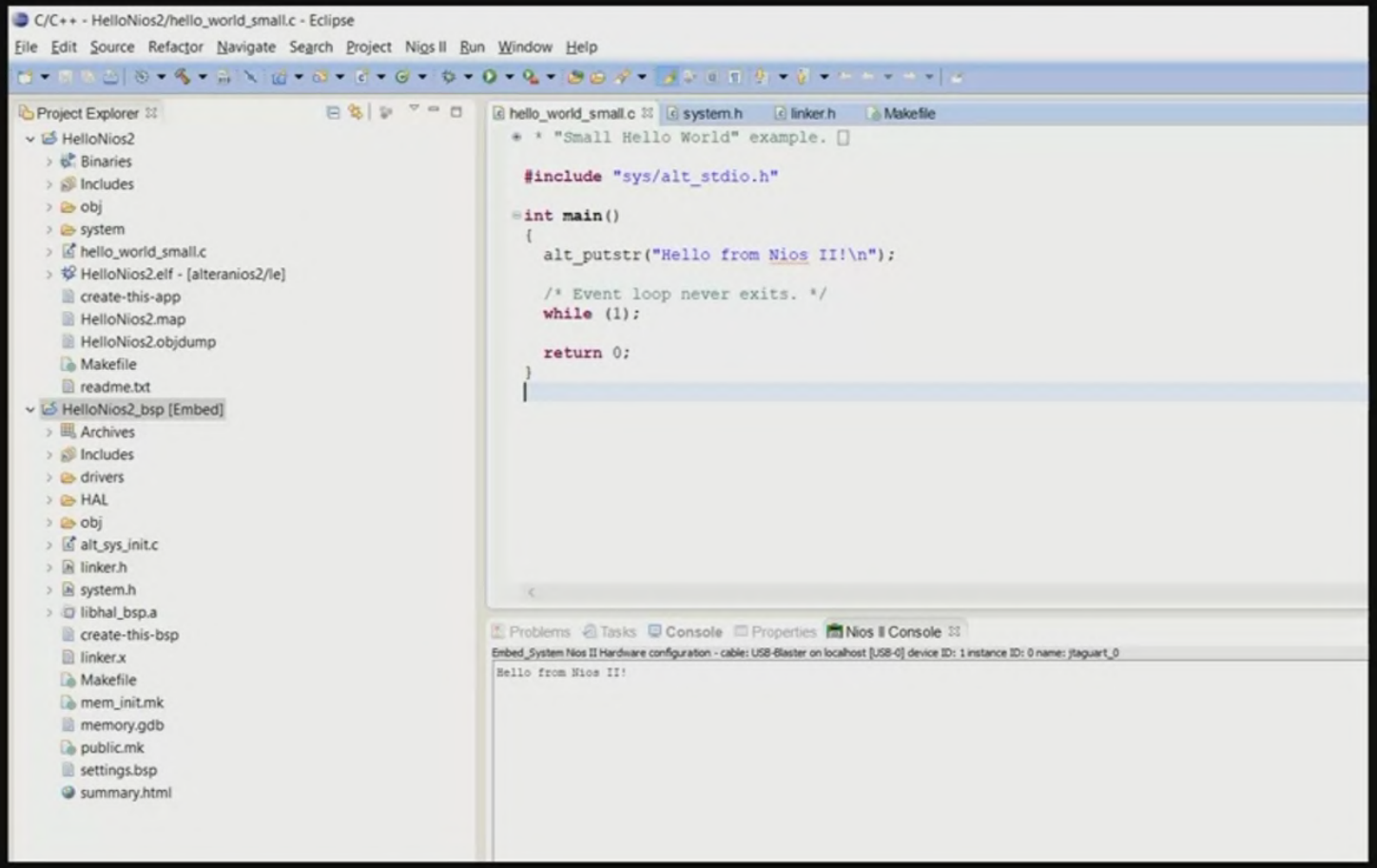
- How to build and run C programs using the NIOS II soft processor.
- How to handle potential issues in Run Configuration
- How to start a Debugging session in the Debug Perspective

Building Programs for Nios II



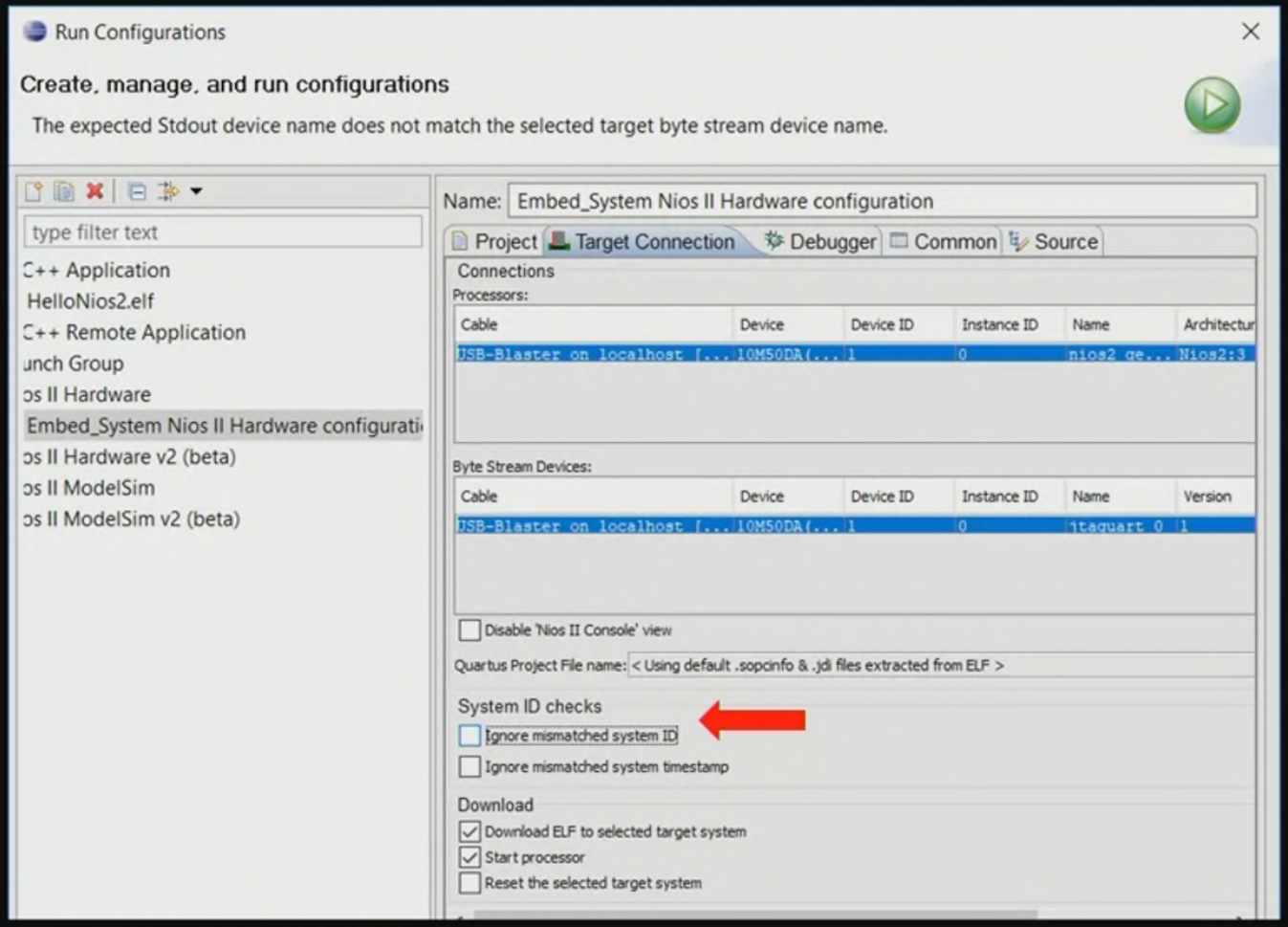
- Click on the application project (HelloNios2) in the Project Explorer window, and then from the top menu Project -> Build Project
- Repeat for the BSP Project

Running Programs for Nios II



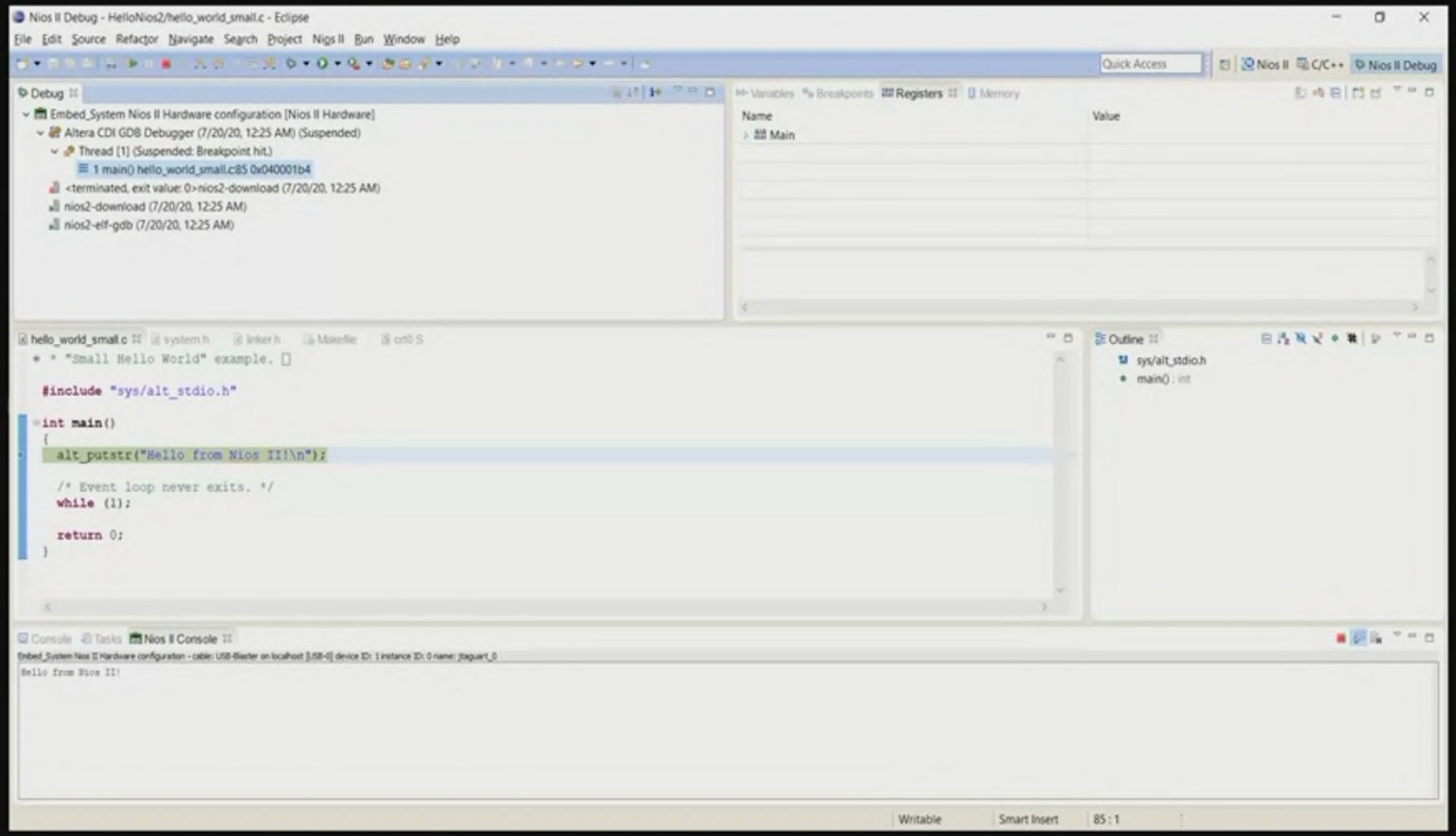
- From the top menu, click on Run-> Run Configurations and choose the configuration
- In this case, the hardware was connected, so hardware configuration was run. Simulation is also possible.

Running Programs for Nios II



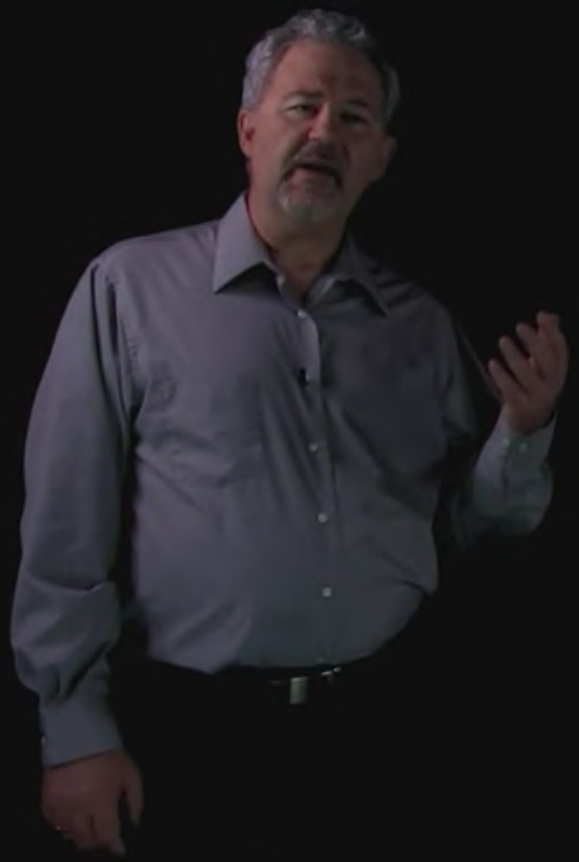
- Possible issue: System ID mismatch. Ignore System ID Mismatch control is in Run-> Run Configurations -> Target Connection

Debugging Programs for Nios II



- From the top menu, click on Debug-> Debug Configurations and choose the configuration
- In this case, the hardware was connected, so hardware configuration was run.

Summary for Soft Processor C Programming



In this video, you have learned:

- How to build and run C programs using the NIOS II soft processor.
- How to handle potential issues in Run Configuration
- How to start a Debugging session in the Debug Perspective

References

[1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

[2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an717.pdf

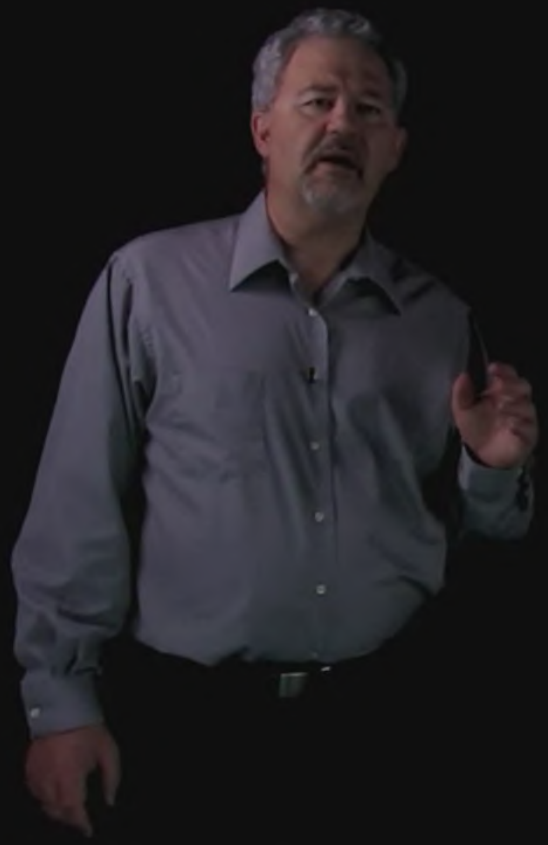
[3] Intel/Altera Staff. (2020/Jul/11), *Embedded Design Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

[4] Intel/Altera Staff. (2020/Jul/11), *MAX 10 Nios II Embedded "Hello World" Lab: DE10-Lite Development Kit.* [Online]. Available: https://fpgacloud.intel.com/devstore/platform/16.1.0/Standard/nios-ii-qsys-hello-world-lab-max10-de10-lite/

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

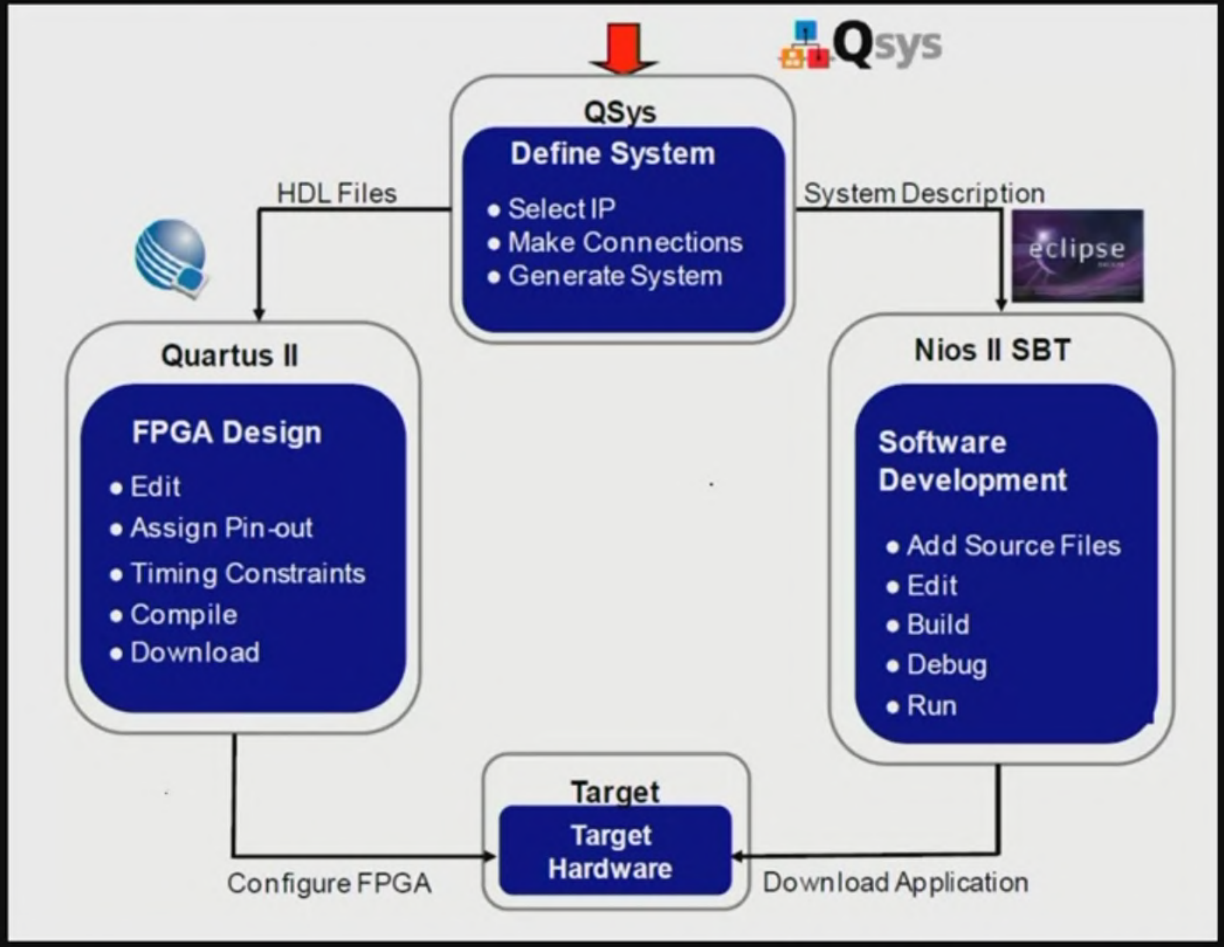
Programming Soft Processors



In this video, you will learn:

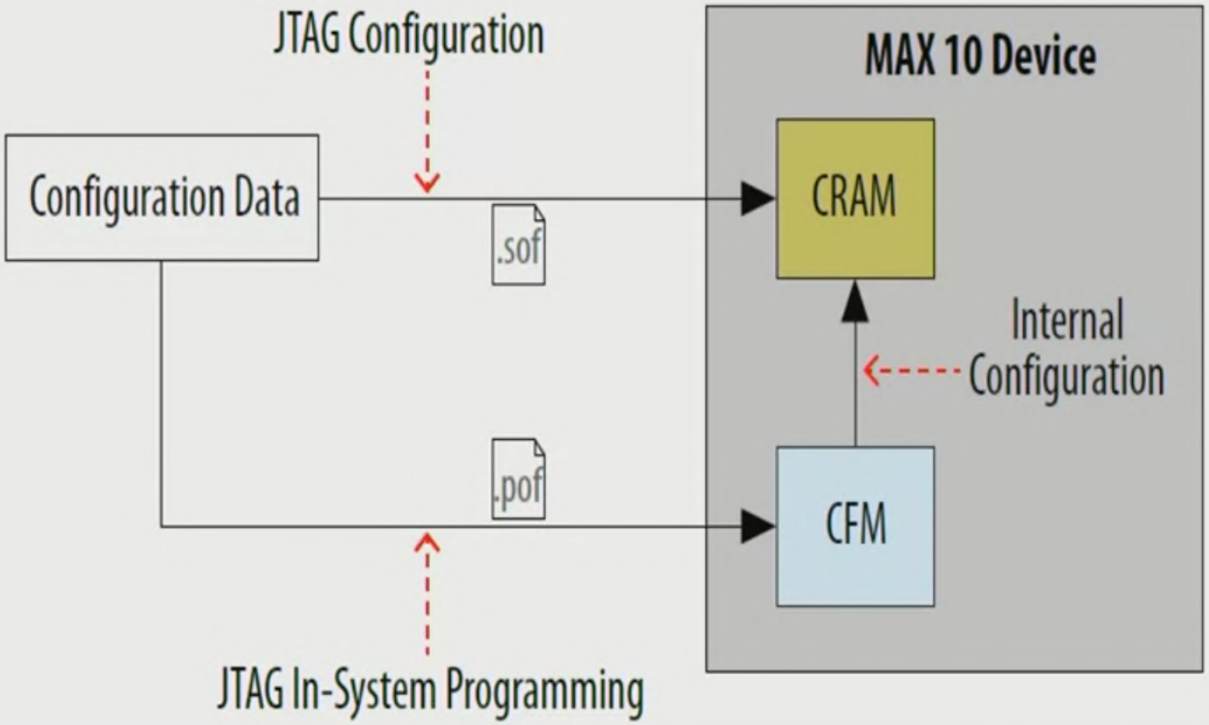
- How to program the target FPGA with hardware and software in the same development session.
- Limitations of the free evaluation version of the Nios II.
- How to resolve basic programming issues

Soft Processor Design Flow



Programming the Nios II

Figure 29: Overview of JTAG Configuration and Internal Configuration for MAX 10 Devices

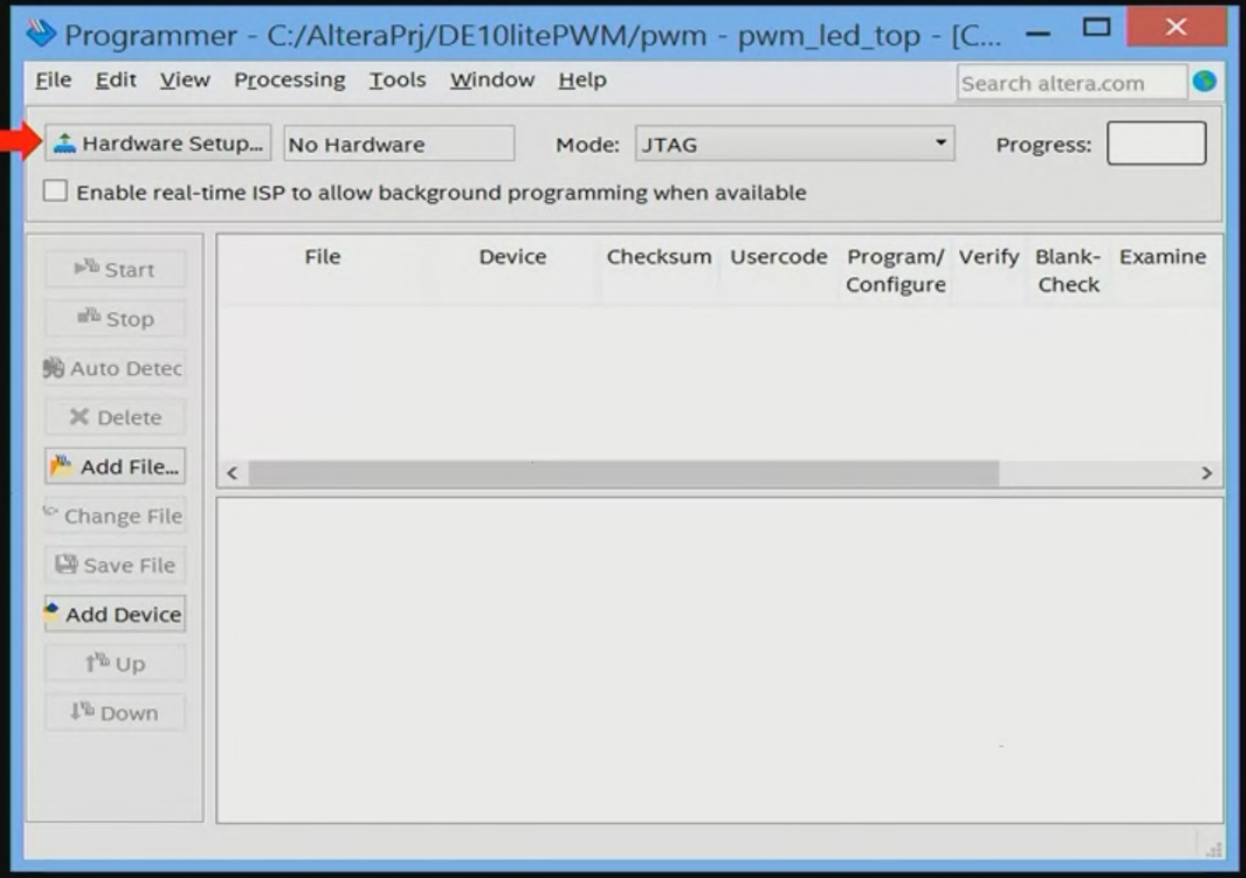


Programming the Nios II

In general, the hardware must first be programmed before loading the software because the software download checks for the presence of a processor. It is a 5-step process to program the hardware:

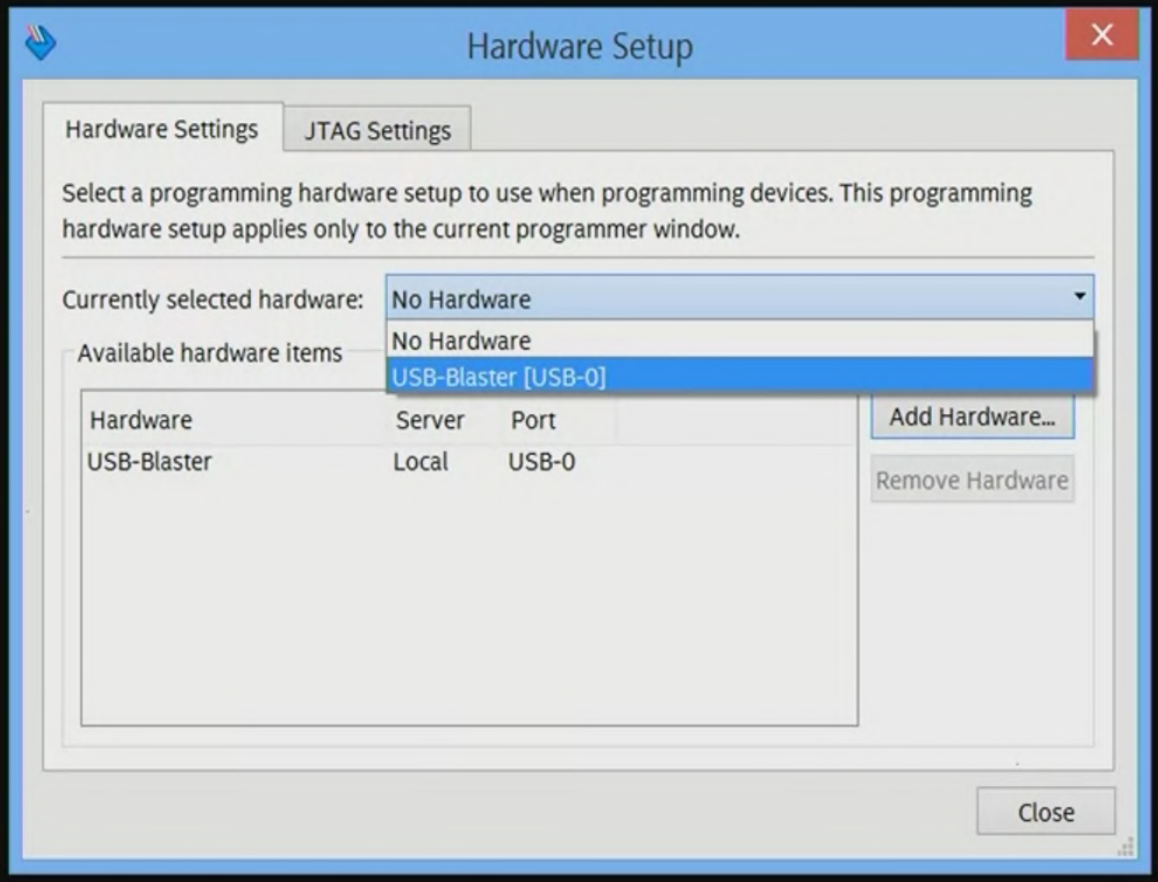
1. Connect the USB cable between your DE10-Lite kit and your computer.
2. Launch the Quartus Programmer, via the icon or through the Tools menu (**Tools -> Programmer**)
3. Setup the programming hardware for the USB Blaster.
4. Select the programming file
5. Click Start in the programmer

Programming the Nios II



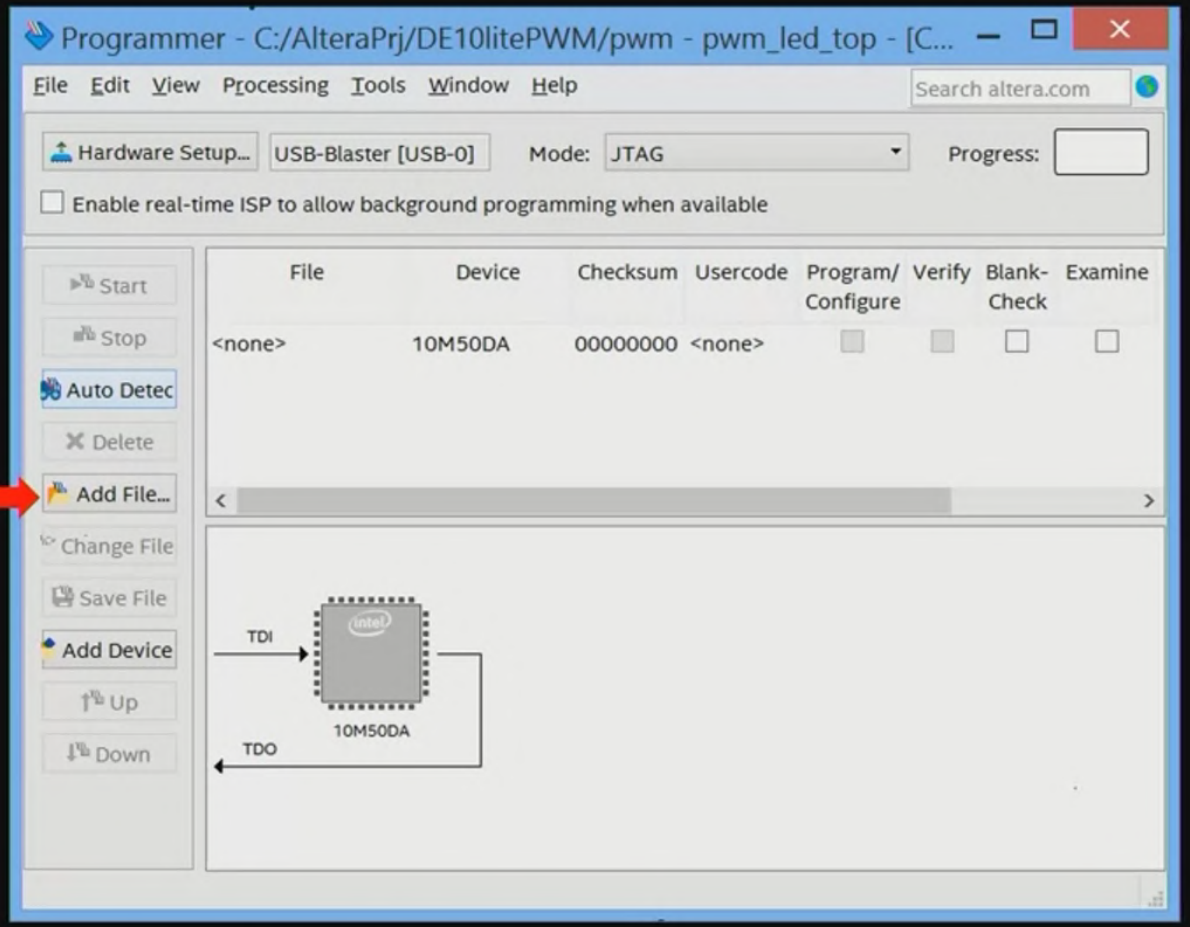
Step 2: Launch the Programmer Tool, select hardware setup

Programming the Nios II



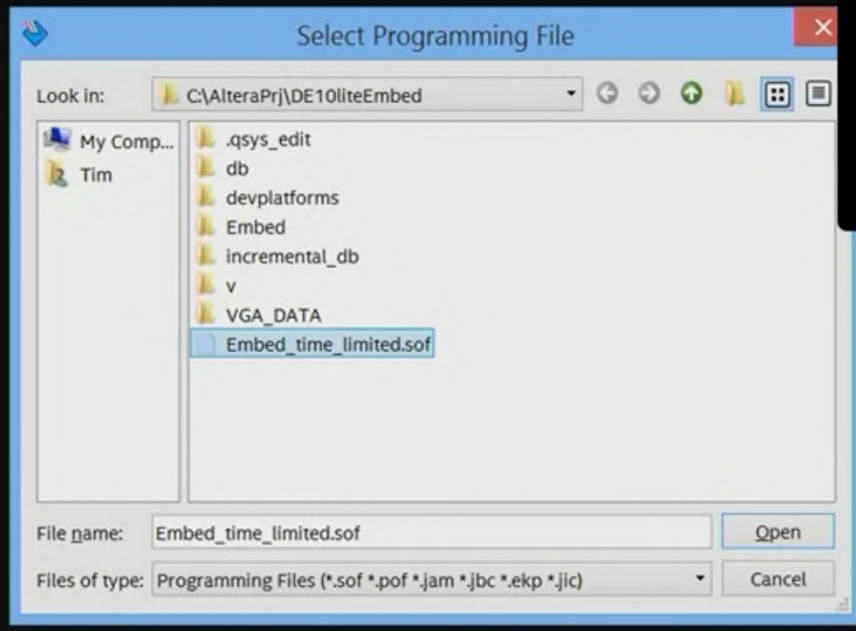
Step 3: Complete the hardware setup by selecting the USB Blaster

Programming the Nios II

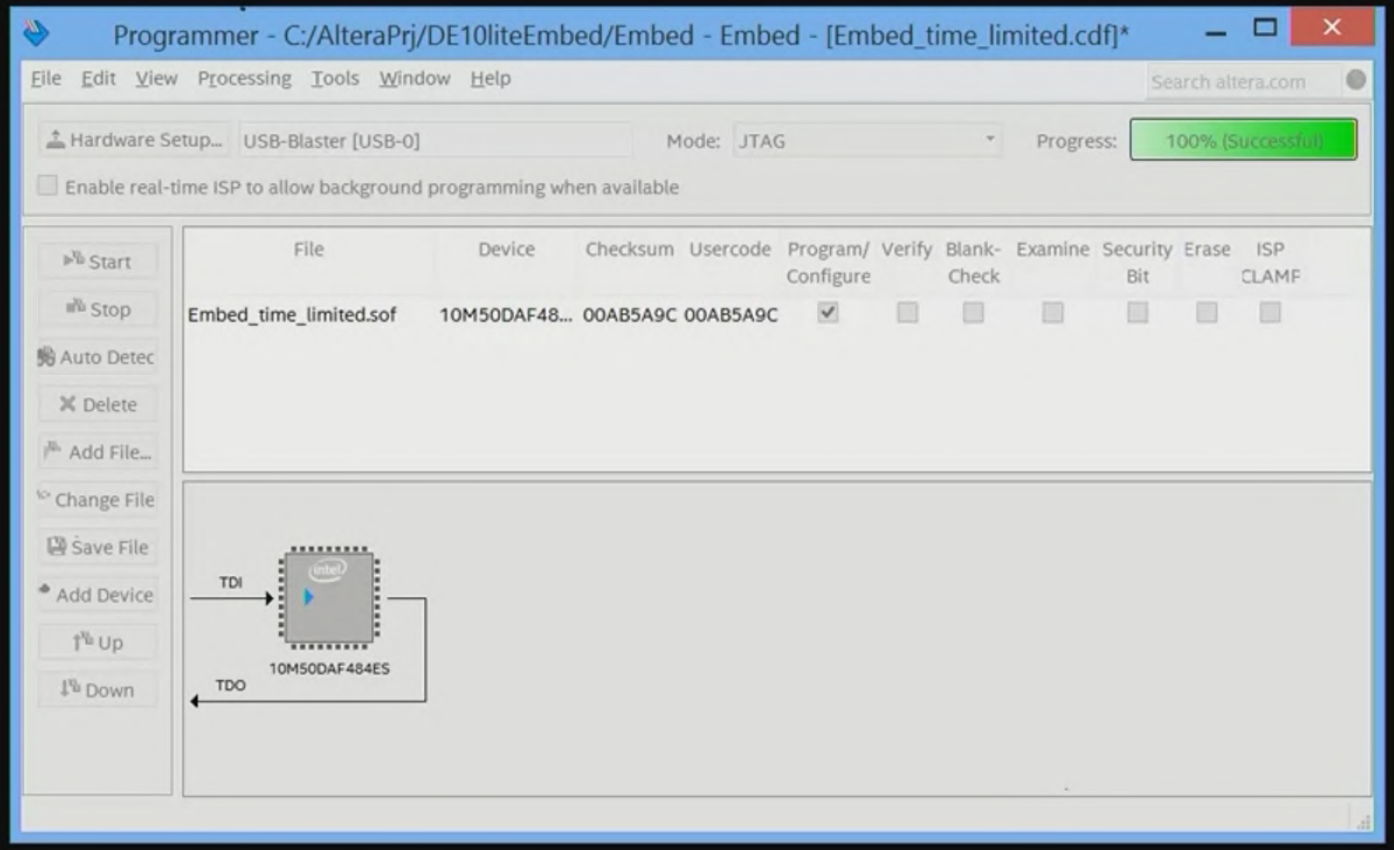


Step 4: To select the programming file, click Add File

Note the file is time-limited.

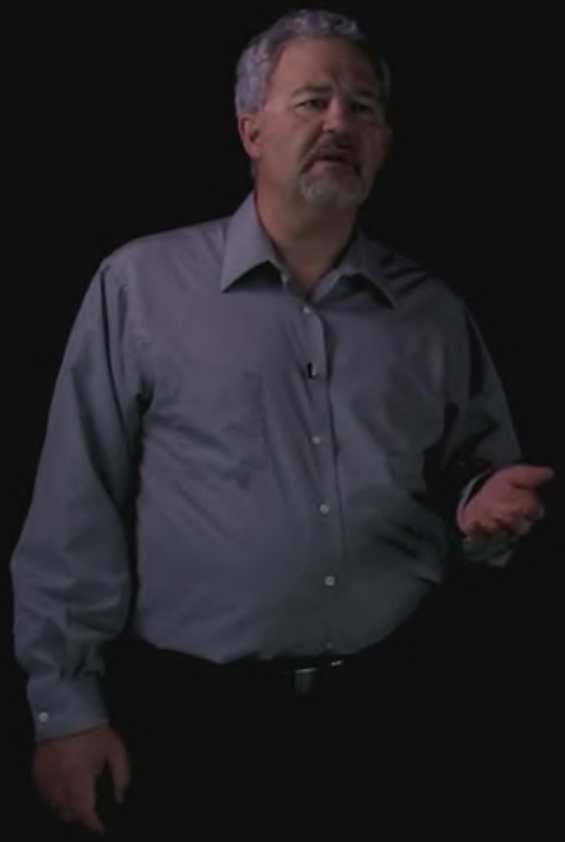


Programming the Nios II



Step 5: Back in the programming window, you may have to delete any other entries that are listed. Click Start in the upper left. You should see progress to 100% all in green.

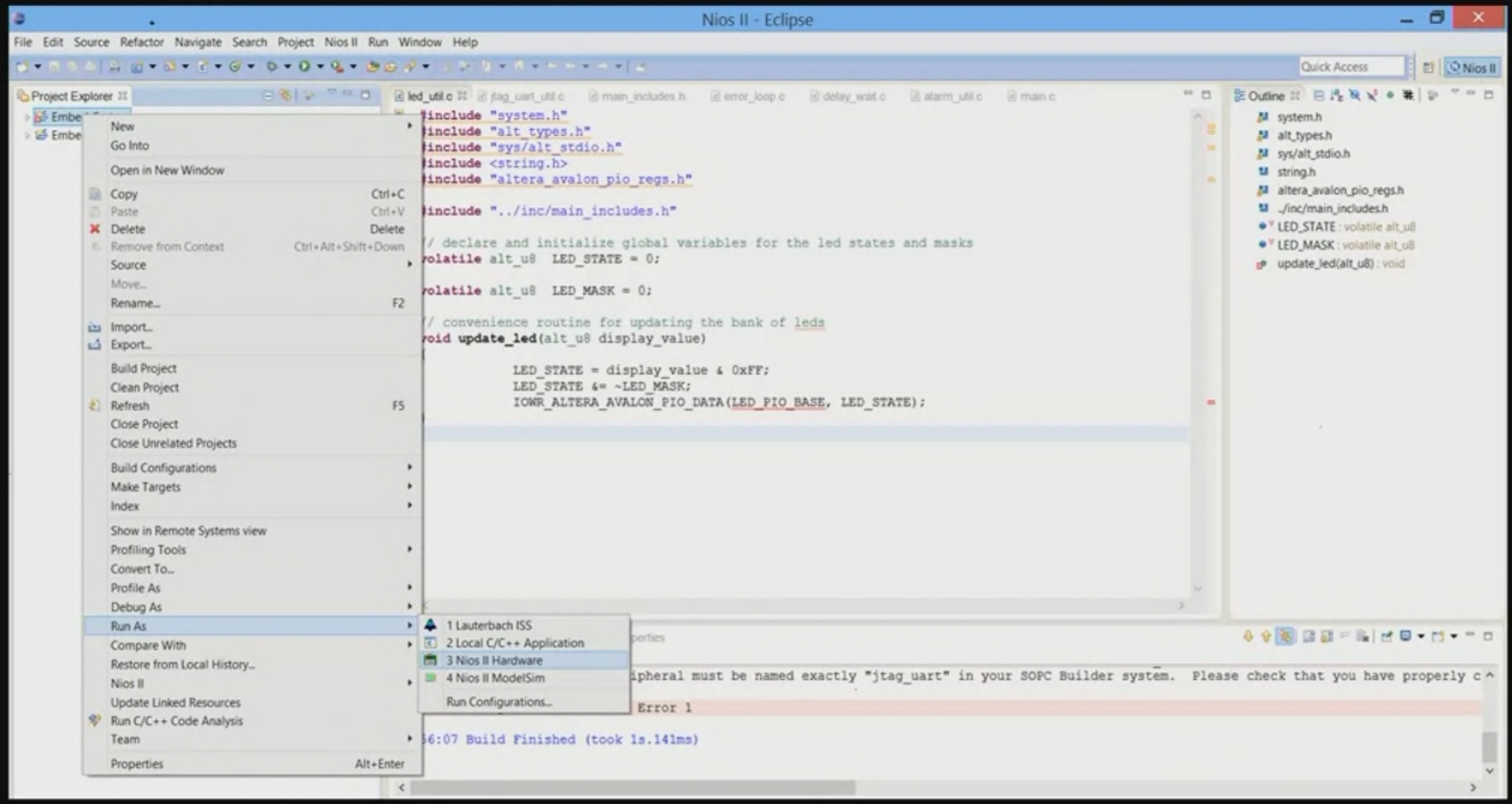
Programming the Nios II - Limitations



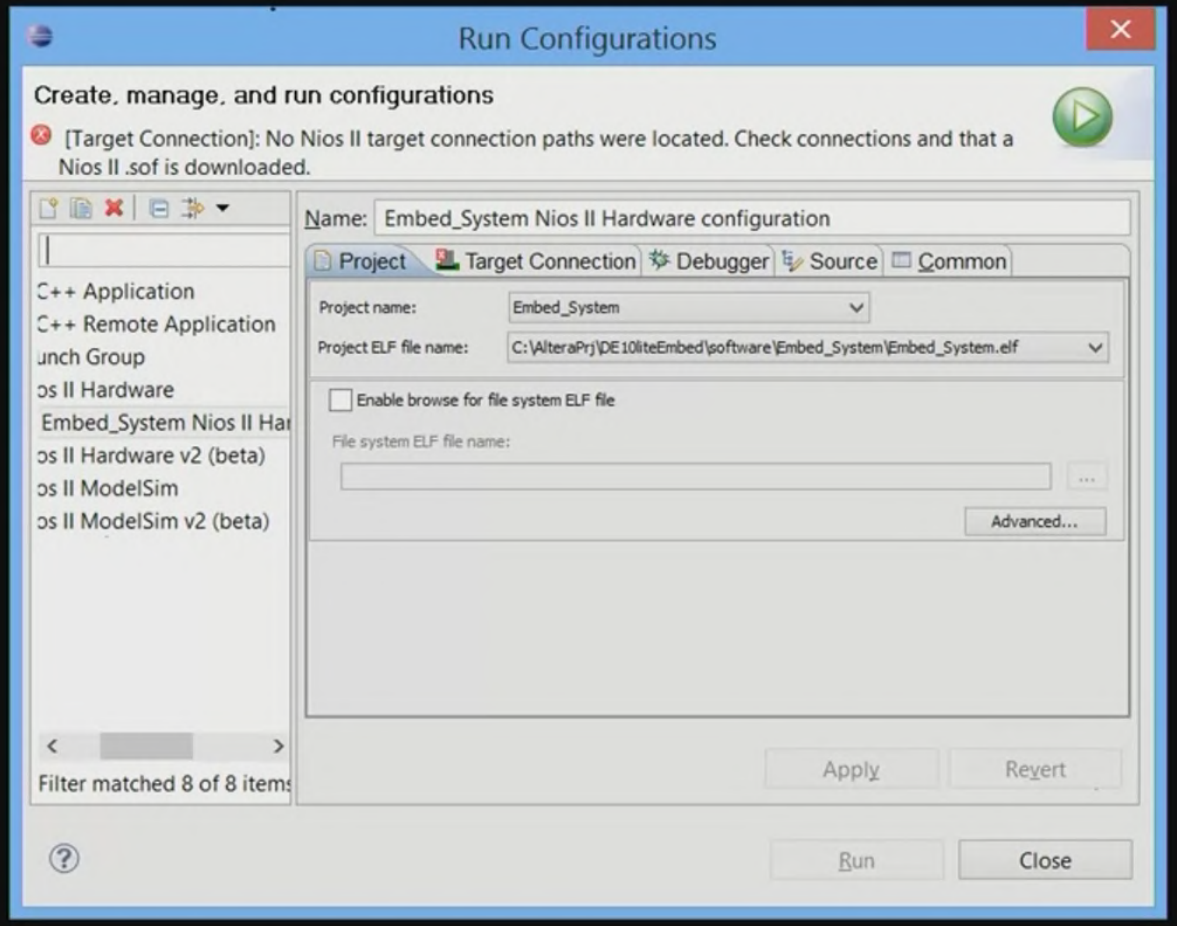
There are some limitations to the use of the NIOS II unless you have a license

- Unless you have a license for the Nios II processor, you will obtain a message about a time-limited megafunction. This message indicates operation will occur in "OpenCore Plus" evaluation mode.
- ***PLEASE NOTE:*** *OpenCore Plus evaluation mode prevents us from programming the flash of the MAX 10 device and evaluation must occur by programming the FPGA SRAM from the Quartus Programmer each time the kit is powered up using the .sof file.*
- A window will pop up on the Quartus Programmer. Just leave this up and ***do not press "Cancel"*** until you are finished using the hardware design that you just downloaded.

Programming the Nios II - software

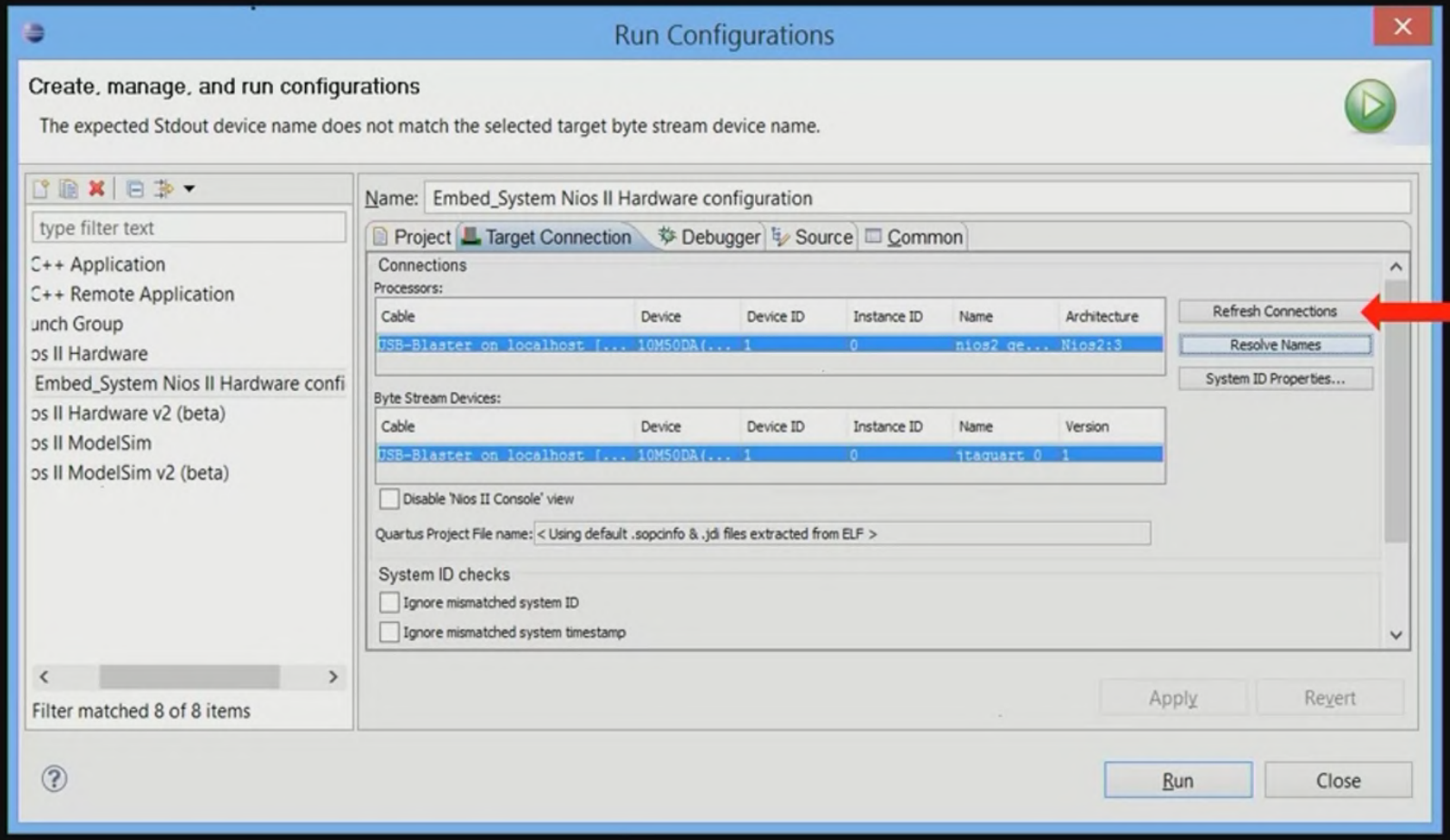


Programming the Nios II - software



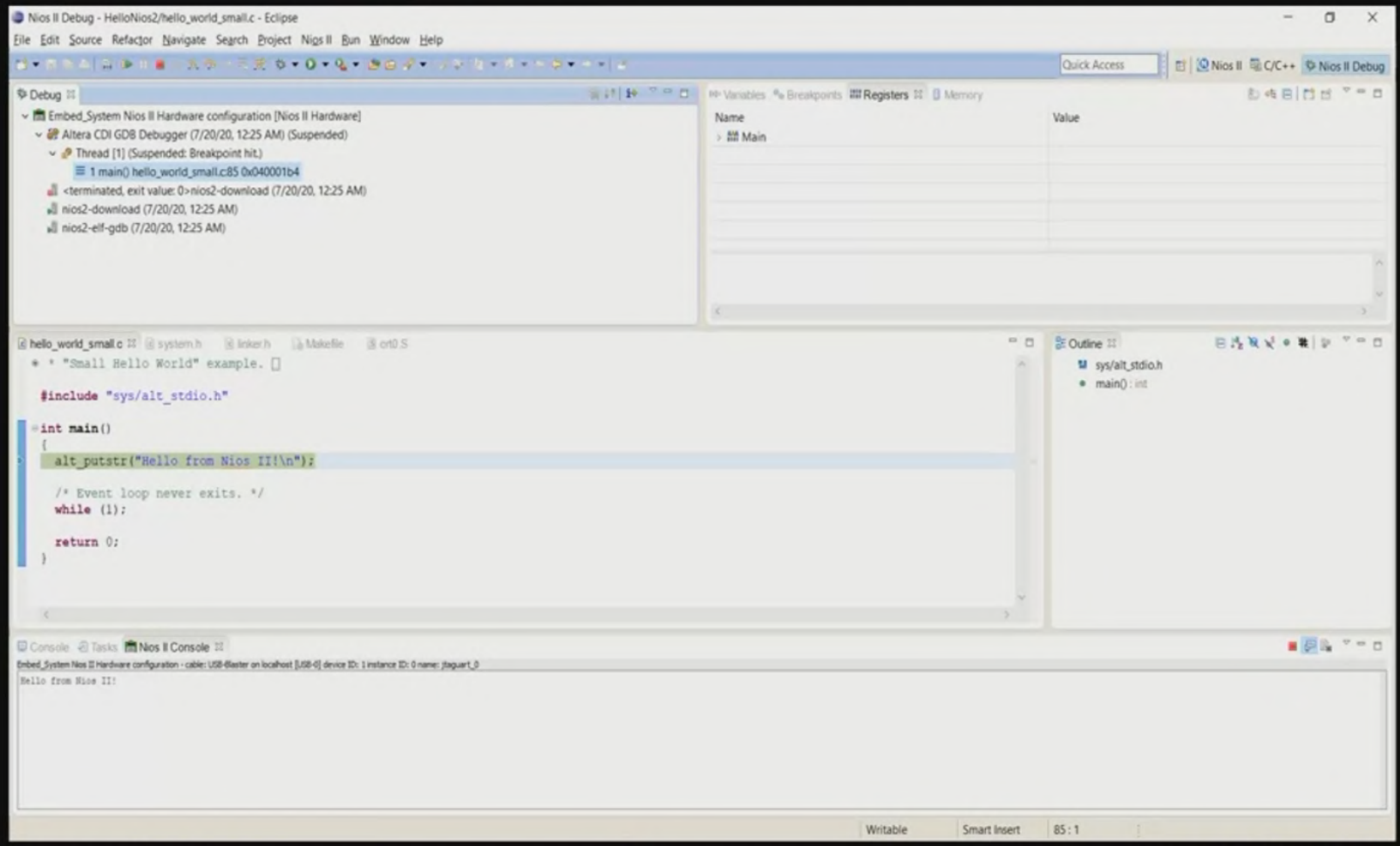
You may see a message
indicates no connection
found as shown here. If
on the target connectio
Click on Refresh Connec
and then hit Run

Programming the Nios II - software



Click on Refre
Connections a
hit Run

Programming the Nios II - software



Summary for Soft Processor C Programming

In this video, you have learned:

- How to program the target FPGA with hardware and software in the same development session.
- Limitations of the free evaluation version of the Nios II.
- How to resolve basic programming issues

References

[1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

[2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an717.pdf

[3] Intel/Altera Staff. (2020/Jul/11), *Embedded Design Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

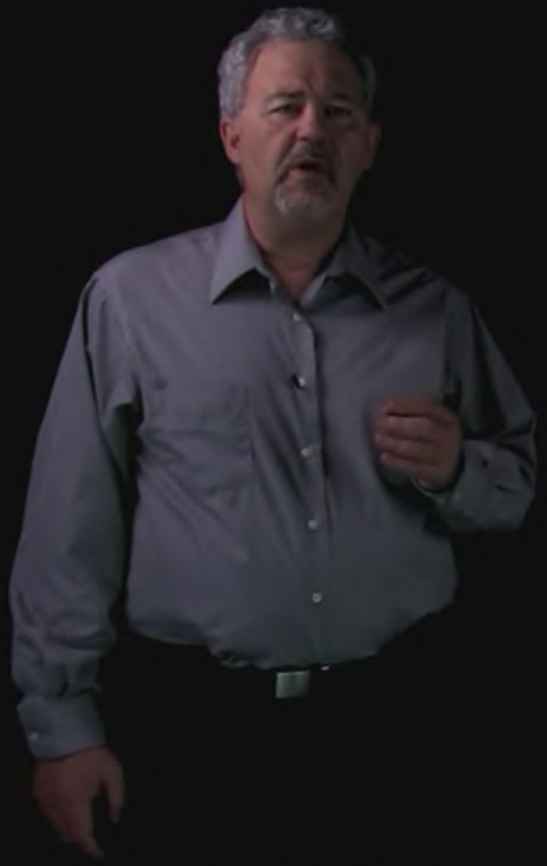
[4] Intel/Altera Staff. (2020/Jul/11), *MAX 10 Nios II Embedded "Hello World" Lab: DE10-Lite Development Kit.* [Online]. Available: https://fpgacloud.intel.com/devstore/platform/16.1.0/Standard/nios-ii-qsys-hello-world-lab-max10-de10-lite/

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

Memory in the Nios II



In this video, you will learn:

- How memory is used by the Nios II for data and instructions in a flexible manner that helps optimize performance
- How to view and make changes to the memory map in Qsys and in Eclipse using the BSP Editor
- How memory allocation is controlled in software using system.h and linker.h

Nios Memory Organization.

The flexible nature of the Nios II memory and I/O organization are the most notable difference between Nios II processor systems and traditional microcontrollers. Because Nios II processor systems are **configurable**, the memories and peripherals vary from system to system.

Nios Memory Organization.

A Nios II core uses one or more of the following to provide memory and I/O access:

- **Instruction master port**—An Avalon® Memory-Mapped (Avalon-MM) master port that connects to instruction memory via system interconnect fabric
- **Instruction cache**—Fast cache memory internal to the Nios II core
- **Data master port**—An Avalon-MM master port that connects to data memory and peripherals via system interconnect fabric
- **Data cache**—Fast cache memory internal to the Nios II core
- **Tightly-coupled instruction or data memory** port—Interface to fast on-chip memory outside the Nios II core

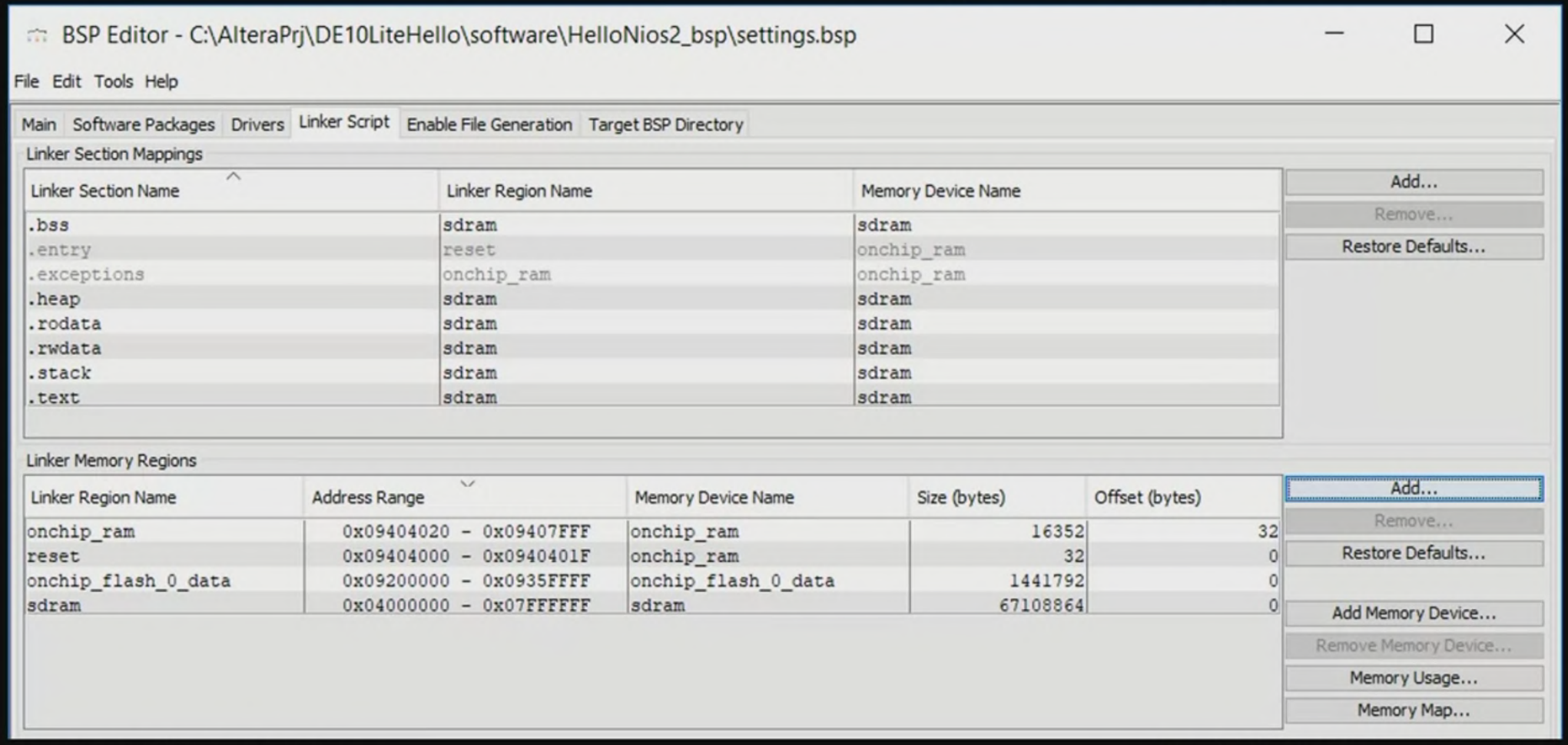
Qsys Defines the Memory Map

System Contents | Address Map | Interconnect Requirements

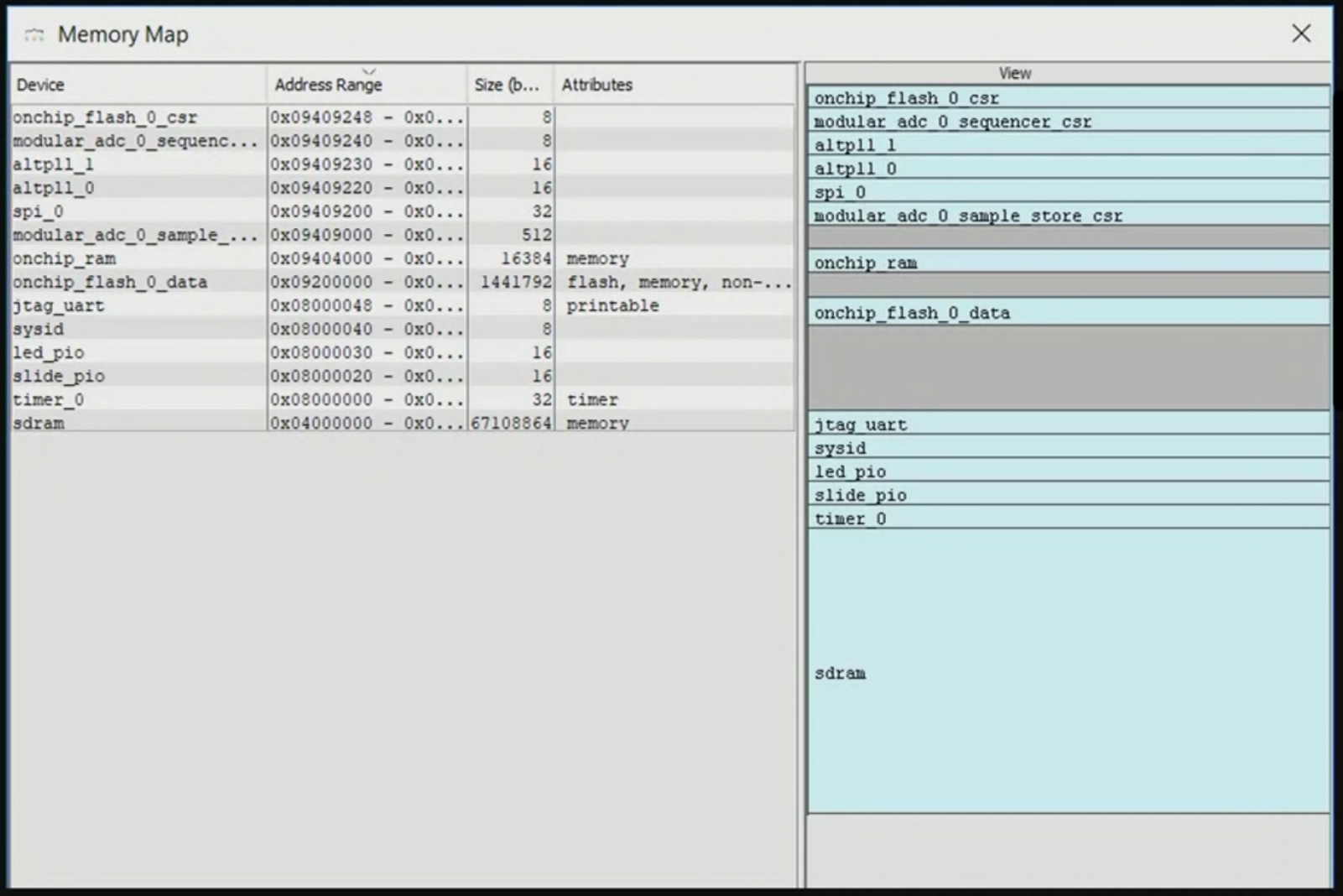
System: Embed **Path:** clk_0

	master_0.master	mm_clock_crossing_bridge_0.m0	nios2_gen2_0.data_master	nios2_gen2_0.instruction_master
altpll_0.pll_slave				0x0940 9220 - 0x0940 922f
altpll_1.pll_slave				0x0940 9230 - 0x0940 923f
jtag_uart.avalon_jtag_slave		0x0048 - 0x004f		
led_pio.s1		0x0030 - 0x003f		
mm_clock_crossing_bridge_0.s0			0x0800 0000 - 0x0800 0fff	
modular_adc_0.sequencer_csr	0x0940 9240 - 0x0940 9247		0x0940 9240 - 0x0940 9247	
modular_adc_0.sample_store_csr			0x0940 9000 - 0x0940 91ff	
nios2_gen2_0.debug_mem_slave			0x0940 8800 - 0x0940 8fff	0x0940 8800 - 0x0940 8fff
onchip_flash_0.data			0x0920 0000 - 0x0935 ffff	0x0920 0000 - 0x0935 ffff
onchip_flash_0.csr			0x0940 9248 - 0x0940 924f	
onchip_ram.s1			0x0940 4000 - 0x0940 7fff	0x0940 4000 - 0x0940 7fff
sdram.s1			0x0400 0000 - 0x07ff ffff	0x0400 0000 - 0x07ff ffff
slide_pio.s1		0x0020 - 0x002f		
spi_0.spi_control_port			0x0940 9200 - 0x0940 921f	
sysid.control_slave		0x0040 - 0x0047		
timer_0.s1		0x0000 - 0x001f		
led_pio.s1 via mm_clock_crossin...			0x0800 0030 - 0x0800 003f	
slide_pio.s1 via mm_clock_cross...			0x0800 0020 - 0x0800 002f	
timer_0.s1 via mm_clock_crossi...			0x0800 0000 - 0x0800 001f	
sysid.control_slave via mm_cloc...			0x0800 0040 - 0x0800 0047	
jtag_uart.avalon_jtag_slave via ...			0x0800 0048 - 0x0800 004f	

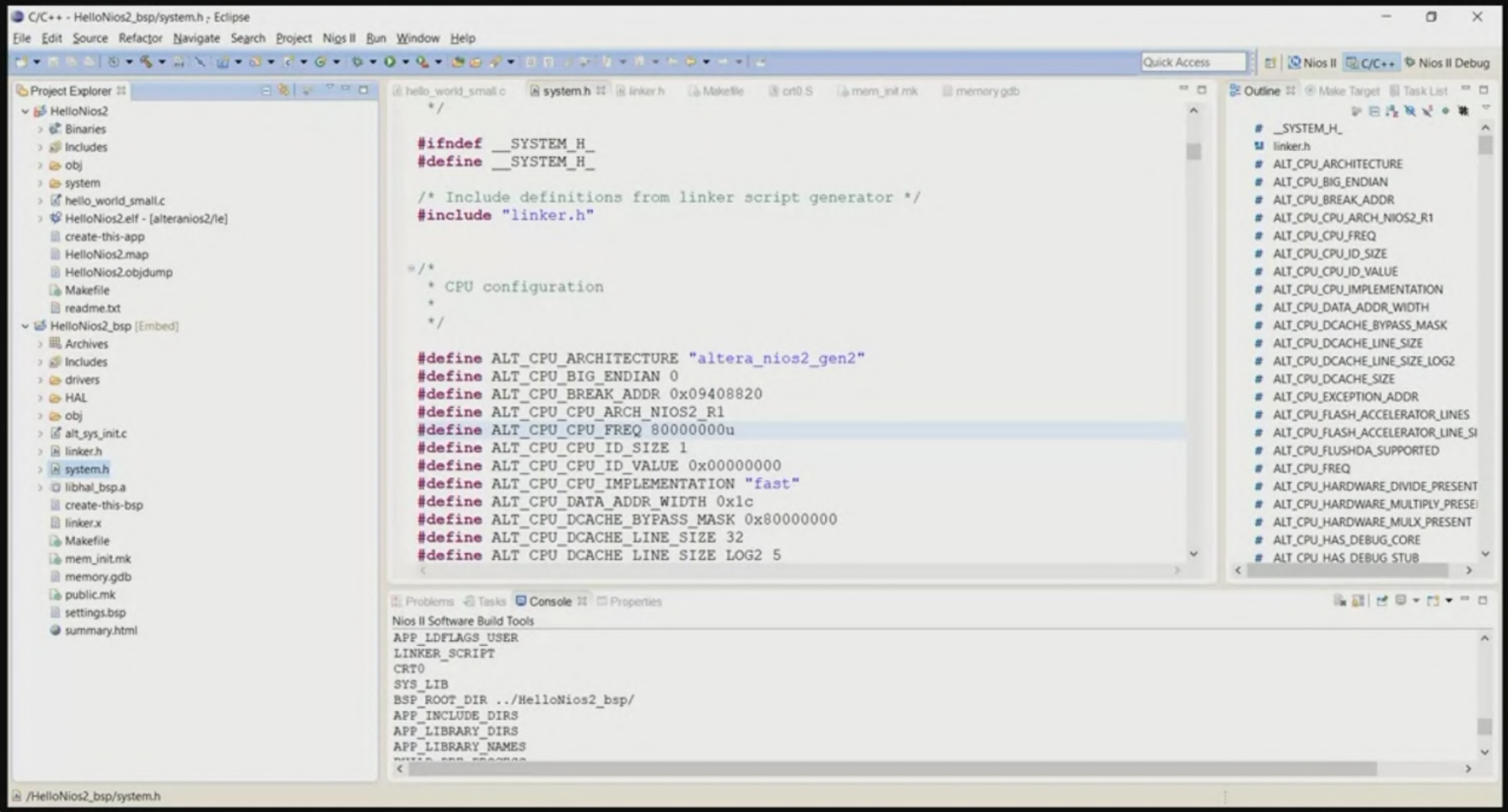
BSP Editor allows Memory Changes



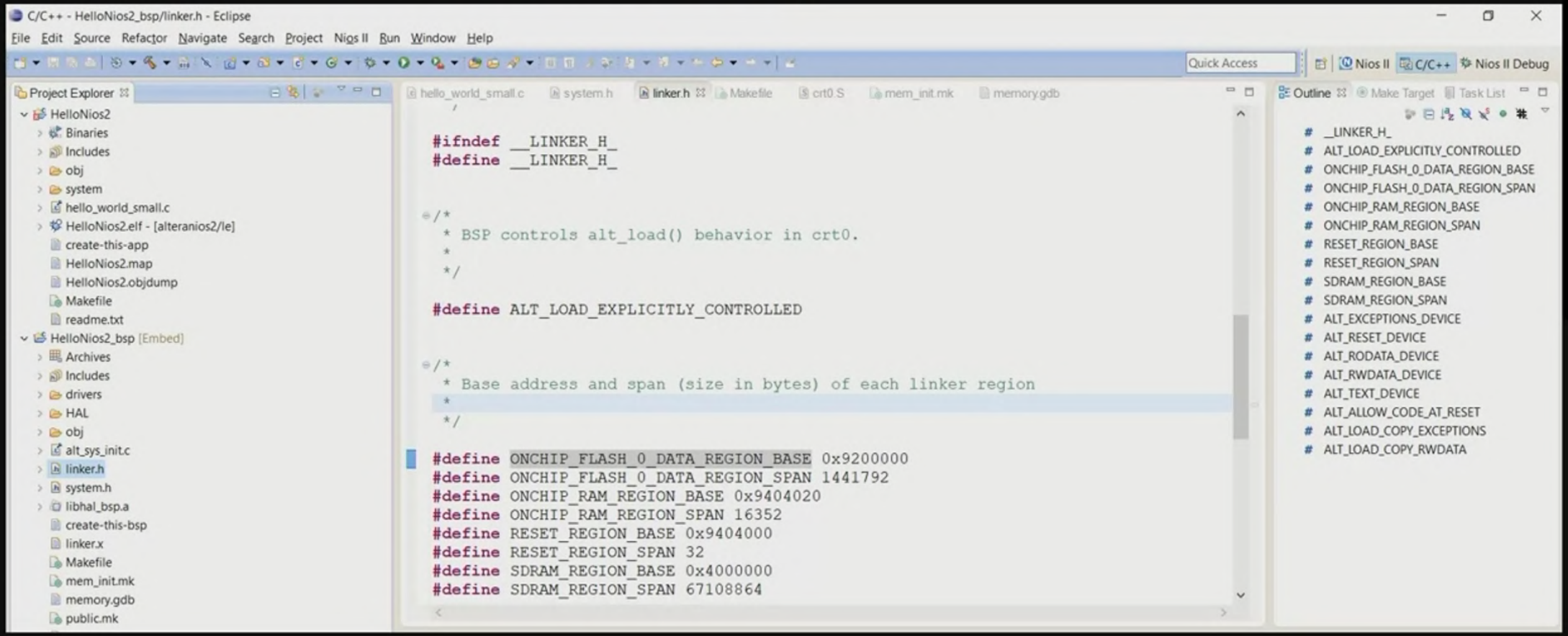
BSP Memory Map



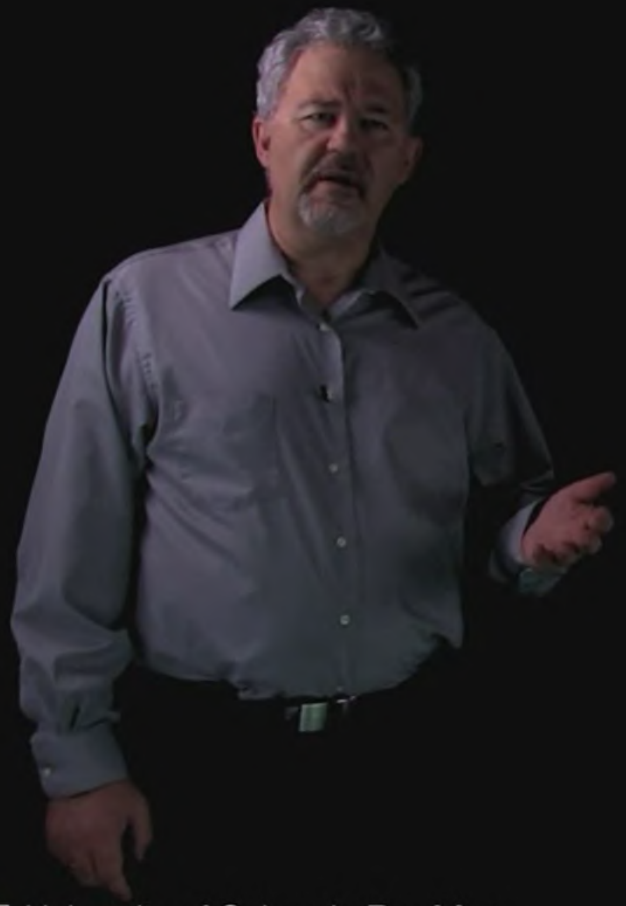
System.h and Memory



Linker.h and Memory



Summary for Soft Processor C Programming



In this video, you have learned:

- How memory is used by the Nios II for data and instructions in a flexible manner that helps optimize performance
- How to view and make changes to the memory map in Qsys and in Eclipse using the BSP Editor
- How memory allocation is controlled in software using system.h and linker.h

References

[1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html

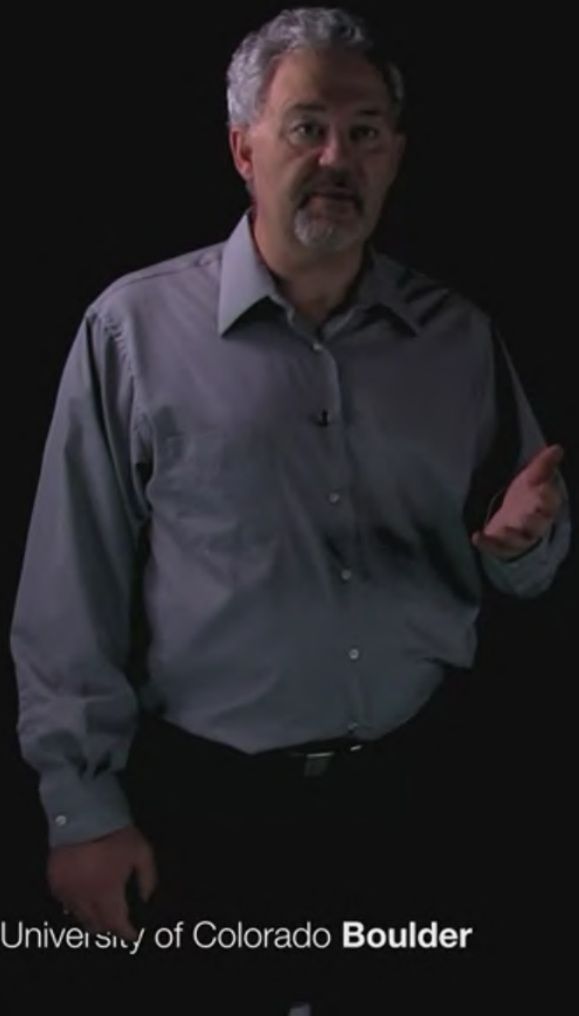
[2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an717.pdf

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

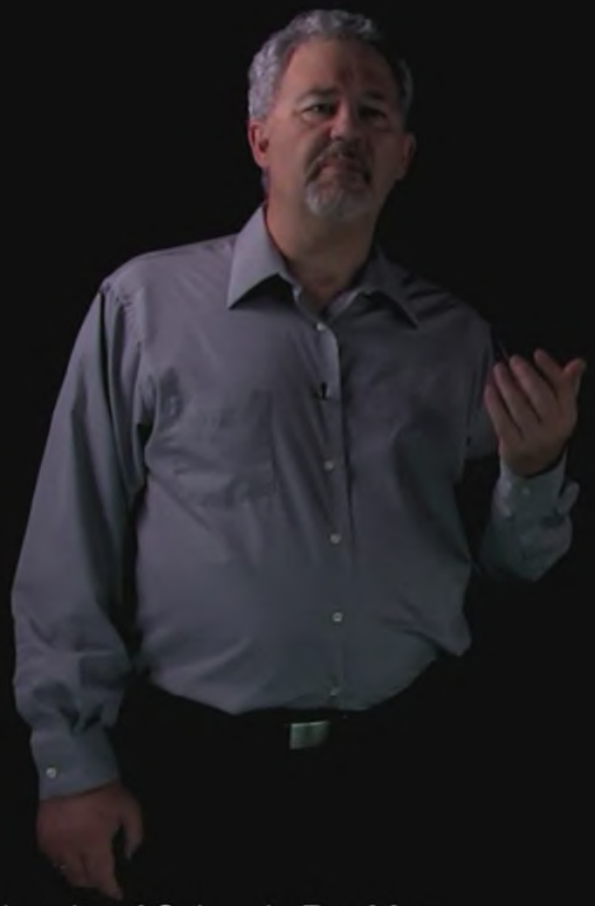
Accessing Custom Instructions



In this video, you will learn:

- How to radically improve performance of some algorithms by use of custom instructions in the Nios II
- How the software interface is enabled by information from Qsys and the BSP editor to allow use by a simple Macro
- How to use the custom instruction in an example design showing the definition and use of the Macro

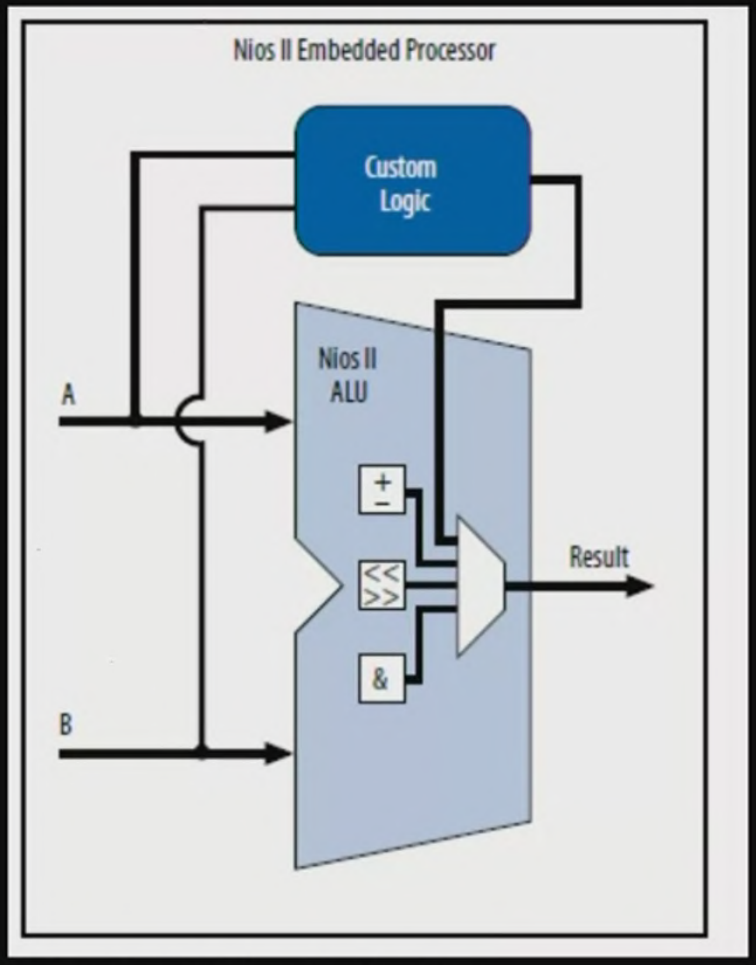
Nios Custom Instructions

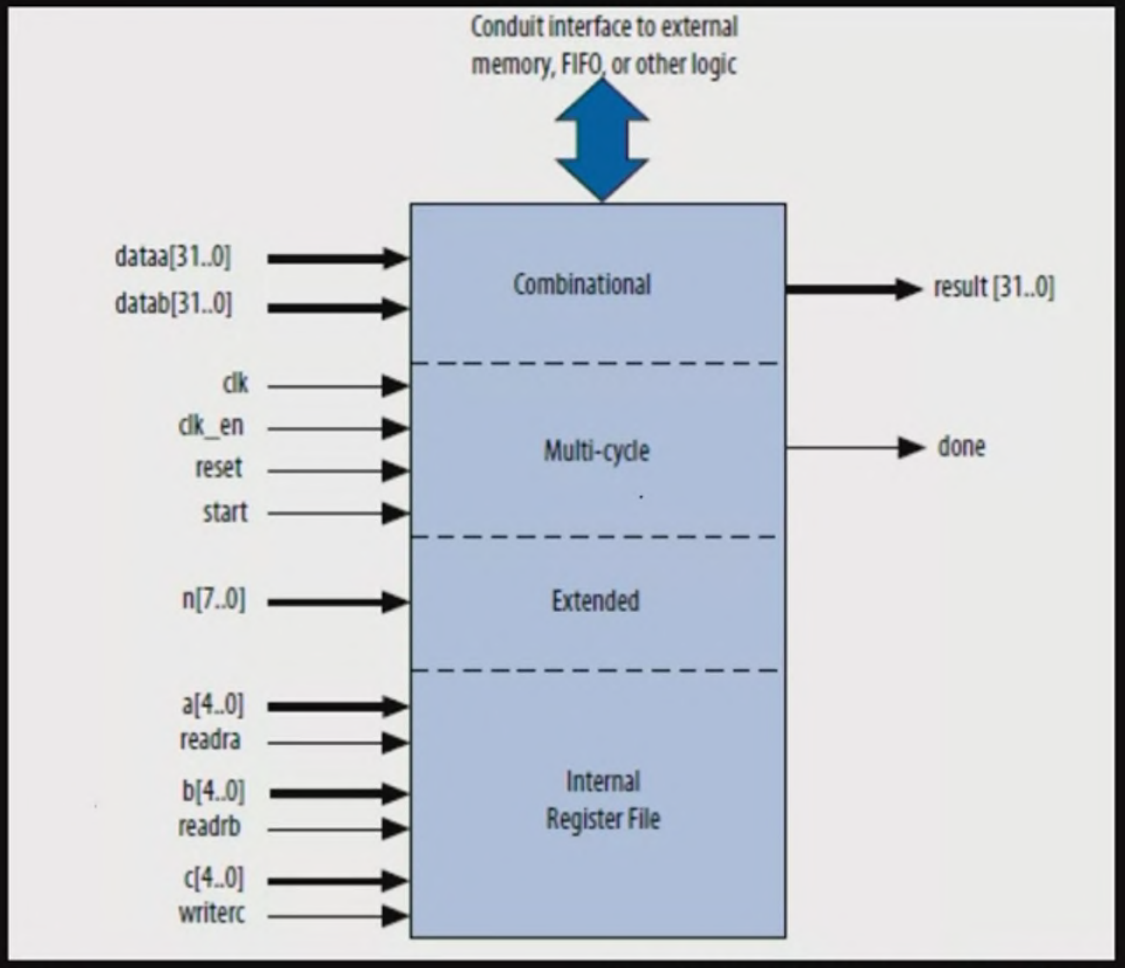


Custom instructions give you the ability to tailor the Nios II processor to meet the needs of a particular application. You can accelerate time critical software algorithms, as much as 10 to 100 times, by converting them to custom hardware logic blocks.

Because it is easy to alter the design of the FPGA-based Nios II processor, custom instructions provide an easy way to experiment with hardware-software tradeoffs at any point in the design process.

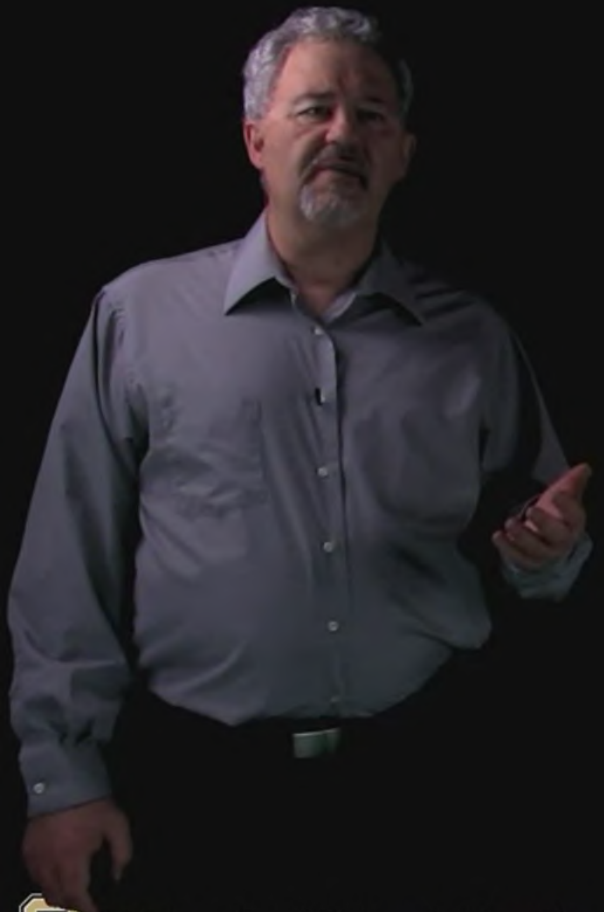
Nios Custom Instructions





[1]

Custom Instruction Software Interface



The Nios II custom instruction software interface is simple and abstracts the details of the custom instruction from the software developer.

For each custom instruction, the Nios II Embedded Design Suite (EDS) generates a macro in the system header file, system.h. You can use the macro directly in your C or C++ application code, and you do not need to program assembly code to access custom instructions. Software can also invoke custom instructions in Nios II processor assembly language.

Custom Instruction Use

hello_world_small.c | ci_crc.c | ci_crc.h | crc.h | crc.c

```
 *
 * Description: Custom instruction implementations of the CRC.
 *
 * Notes:       A macro is defined that is used to access the CRC custom
 *     .        instruction.
 ******************************************************************************/

#include "system.h"

/*The n values and their corresponding operation are as follow:
 * n = 0, Initialize the custom instruction to the initial remainder value
 * n = 1, Write  8 bits data to custom instruction
 * n = 2, Write 16 bits data to custom instruction
 * n = 3, Write 32 bits data to custom instruction
 * n = 4, Read  32 bits data from the custom instruction
 * n = 5, Read  64 bits data from the custom instruction
 * n = 6, Read  96 bits data from the custom instruction
 * n = 7, Read 128 bits data from the custom instruction*/
#define CRC_CI_MACRO(n, A)         __builtin_custom_ini(ALT_CI_CUSTOM_INSTRUCTION_0_N + (n & 0x7), (A))

unsigned long crcCI(unsigned char * input_data, unsigned long input_data_length)
{
  unsigned long index;
  /* copy of the data buffer pointer so that it can advance by different widths */
  void * input_data_copy = (void *)input_data;

  /* The custom instruction CRC will initialize to the inital remainder value */
  CRC_CI_MACRO(0,0);

  /* Write 32 bit data to the custom instruction.  If the buffer does not end
   * on a 32 bit boundary then the remaining data will be sent to the custom
   * instruction in the 'if' statement below.
   */
  for(index = 0; index < (input_data_length & 0xFFFFFFFC); index+=4)
  {
    CRC_CI_MACRO(3, *(unsigned long *)input_data_copy);
    input_data_copy += 4;  /* void pointer, must move by 4 for each word */
  }

  /* Write the remainder of the buffer if it does not end on a word boundary */
  if((input_data_length & 0x3) == 0x3)  /* 3 bytes left */
  {
```

Custom Instruction Use

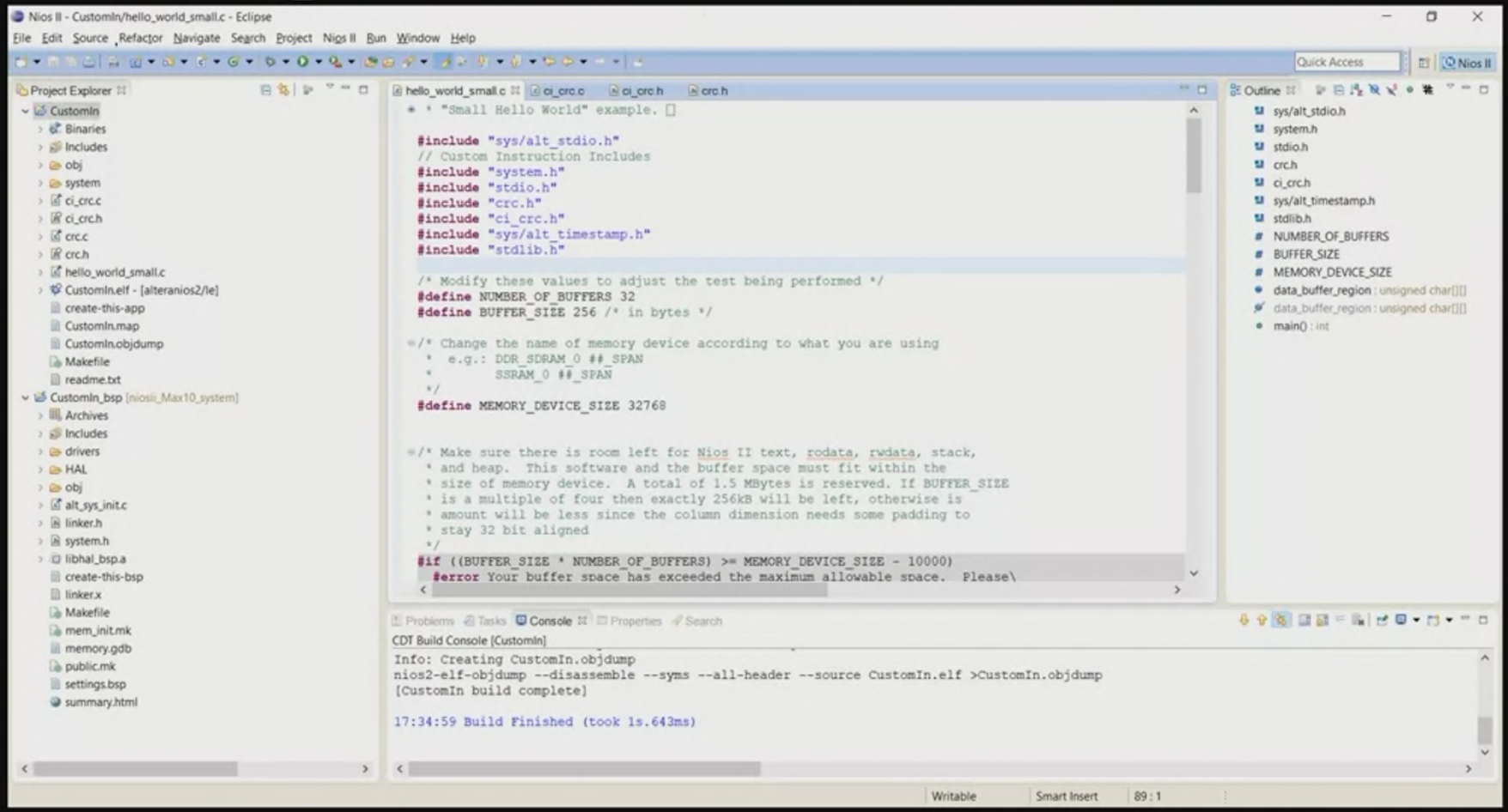
Definition:

```
#define CRC_CI_MACRO(n, A)
__builtin_custom_ini(ALT_CI_CUS
TOM_INSTRUCTION_0_N + (n &
0x7), (A))
```

Use:

```
/* The custom instruction CRC will initialize to the
initial remainder value */
    CRC_CI_MACRO(0,0);
```

Building the Custom Instruction SW

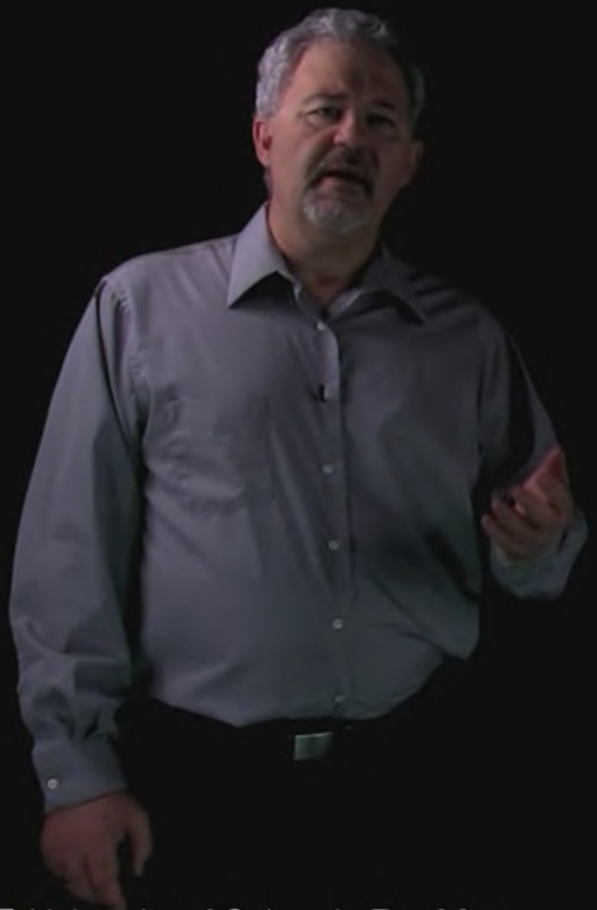


Running the Custom Instruction SW

```
Comparison between software and custom instruction CRC32
**********************************************************
System specification
--------------------
System clock speed = 50 MHz
Number of buffer locations = 32
Size of each buffer = 256 bytes
Initializing all of the buffers with pseudo-random data
--------------------------------------------------------
Initialization completed
Running the software CRC
------------------------
Completed
Running the optimized software CRC
----------------------------------
Completed
Running the custom instruction CRC
----------------------------------
Completed
```

```
Validating the CRC results from all implementations
---------------------------------------------------
All CRC implementations produced the same results
Processing time for each implementation
---------------------------------------
Software CRC = 34 ms
Optimized software CRC = 19 ms
Custom instruction CRC = 00 ms
Processing throughput for each implementation
---------------------------------------------
Software CRC = 2978 Mbps
Optimized software CRC = 32768 Mbps
Custom instruction CRC = 949 Mbps
Speedup ratio
-------------
Custom instruction CRC vs software CRC = 68
Custom instruction CRC vs optimized software CRC = 39
Optimized software CRC vs software CRC = 1
```

Summary for Soft Processor C Programming



In this video, you have learned:

- How to radically improve performance of some algorithms by use of custom instructions in the Nios II
- How the software interface is enabled by information from Qsys and the BSP editor to allow use by a simple Macro
- How to use the custom instruction in an example design showing the definition and use of the Macro

References

[1] Intel/Altera Staff. (2020/Jul/16), *Nios II Custom Instruction User Guide.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/archives/ug-nios2-custom-instruction-15.1.pdf

[2] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook.* [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/processors/support.html