

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.

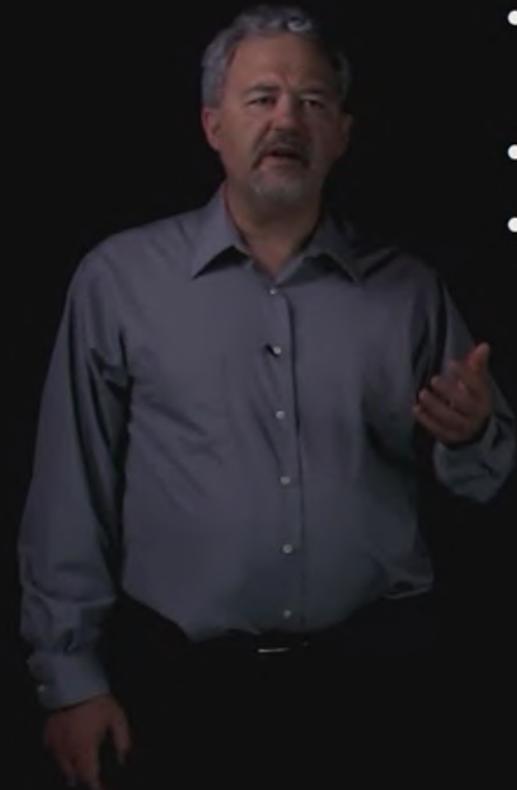
- GNU based C/C++ compilers and tool chain
- JTAG interface to live processor core
- Program, Run, Debug

Software development flow based on industry standards.

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

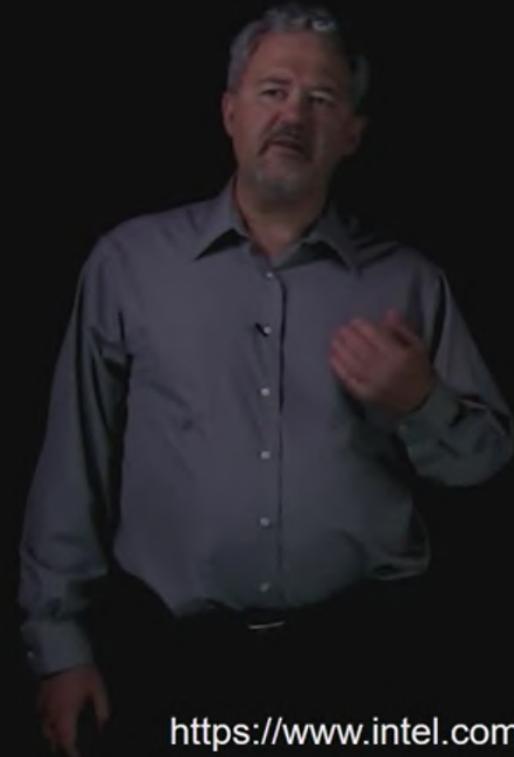
Software Interface to Soft Processors

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>

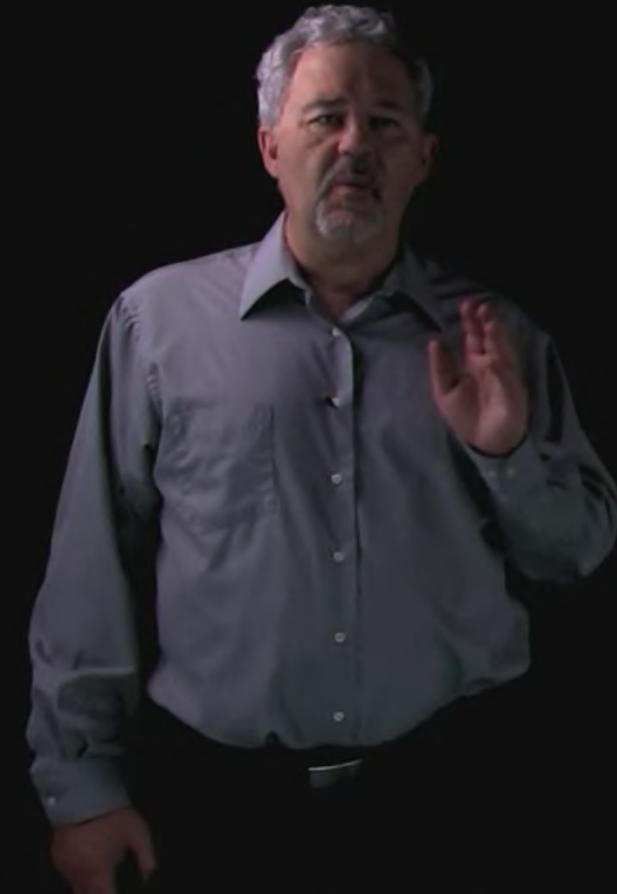


FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition



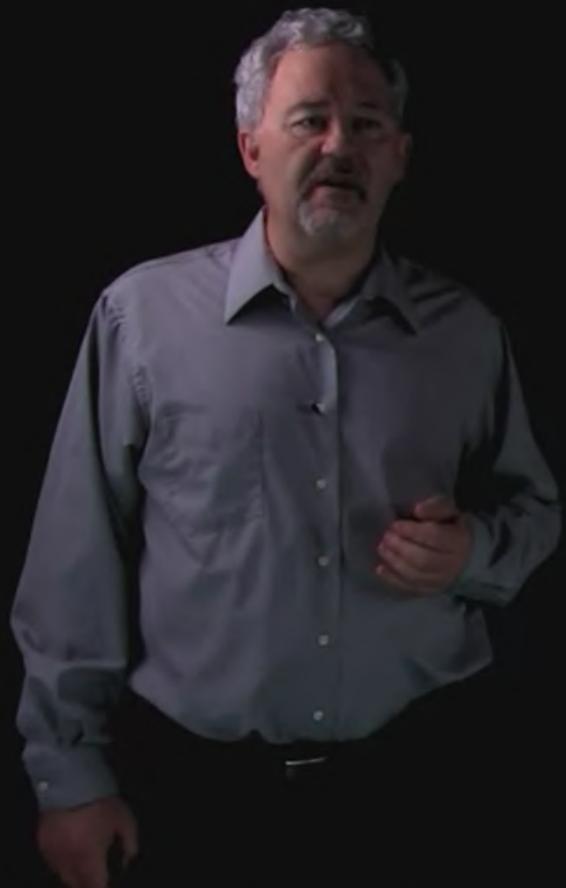
Software for Soft Processors



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

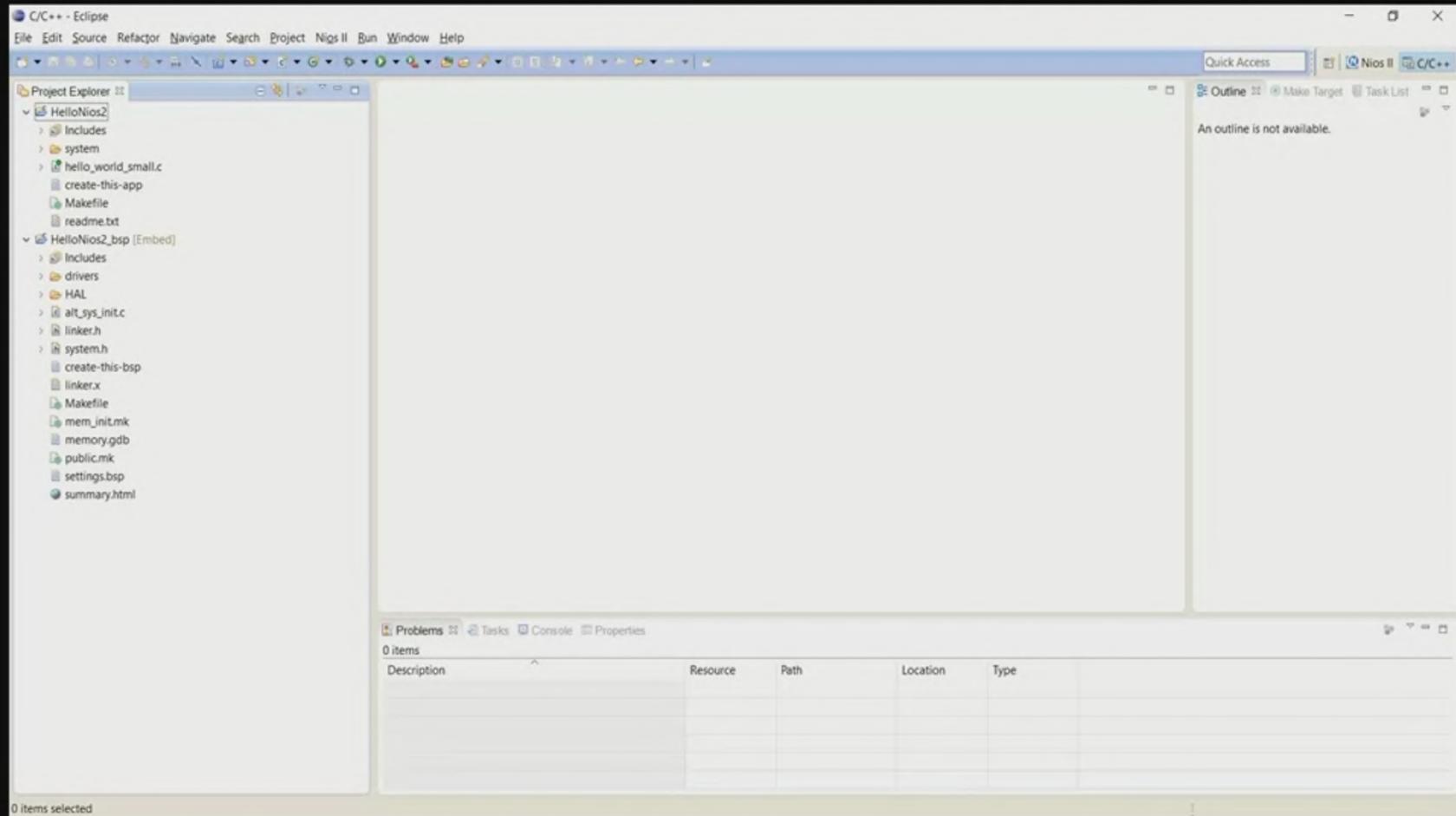


- Eclipse is a popular integrated development environment (IDE) used for embedded programming.
- The Eclipse open source Software Development Kit (SDK) workspace provides :
 - A plugin system for C/C++ source code editing, compiling, debugging, and live terminal. GNU tools provide the compiler, linker 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



University of Colorado **Boulder**



<https://www.eclipse.org/>

Copyright © 2020 University of Colorado

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://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



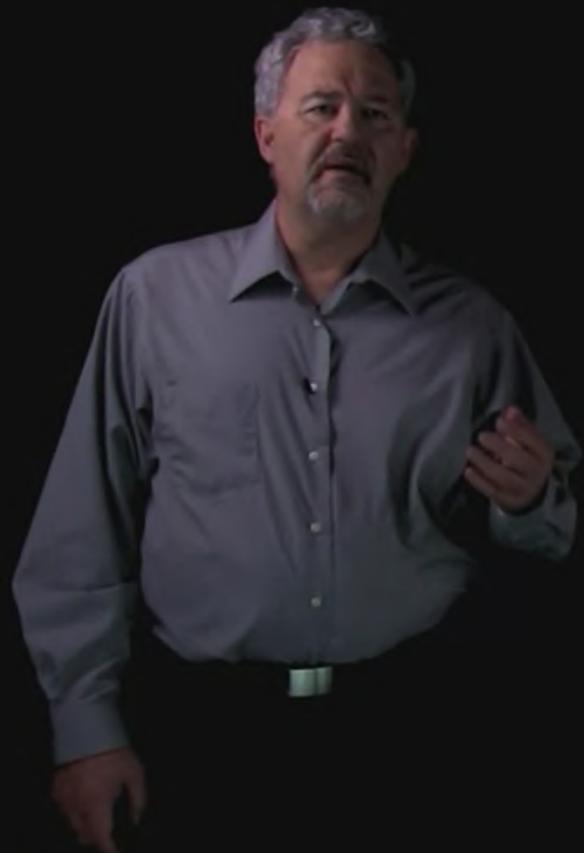
University of Colorado **Boulder**

0:01 / 3:53

Copyright © 2020 University of Colorado



Introduction to the NIOS IDE

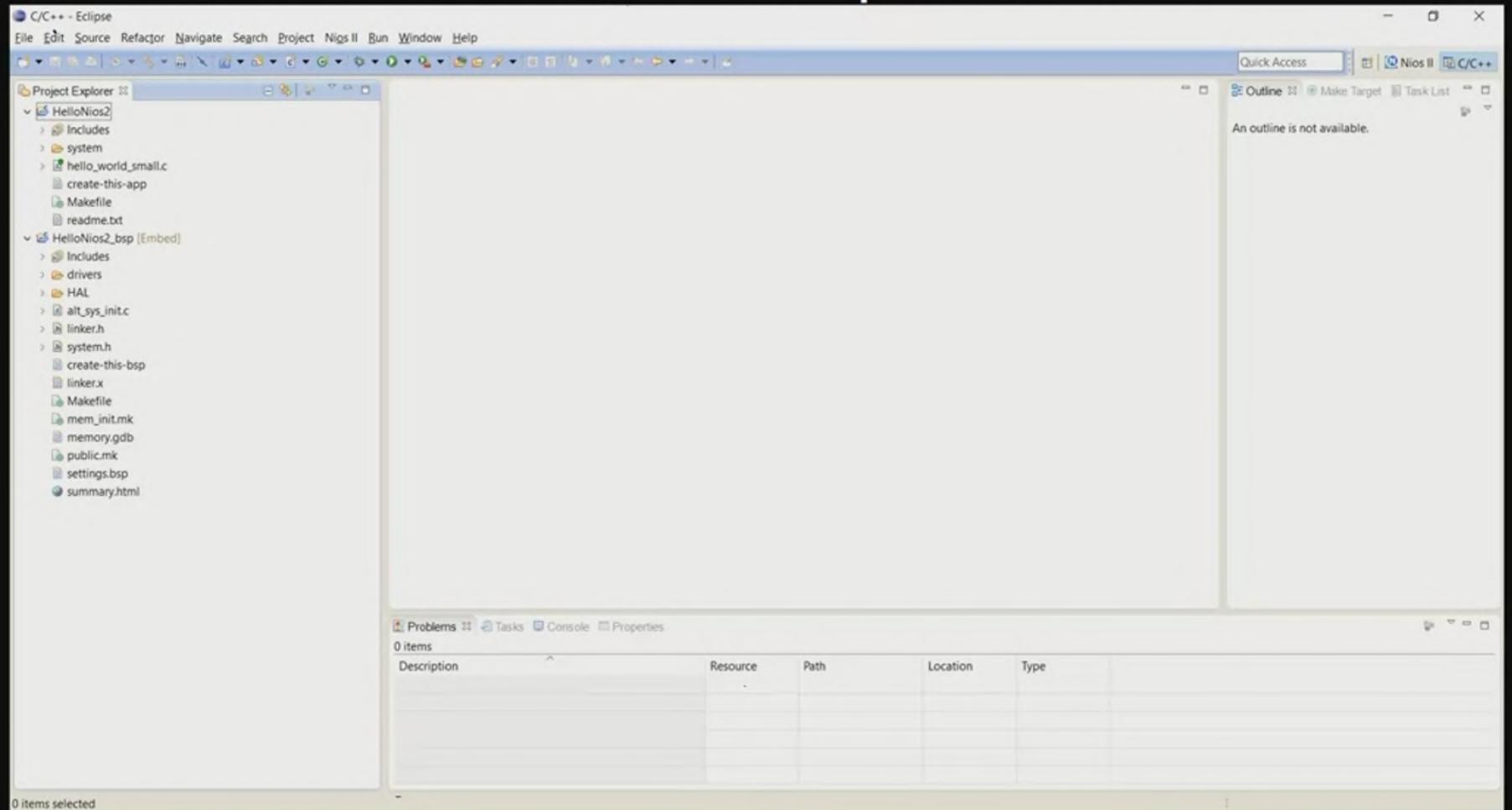


In this video, you will learn:

- 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



Quartus Prime Lite Edition - C:/AlteraPrj/DE10LiteHello/Embed - Embed

File Edit View Project Assignments Processing Tools Window Help

Project Navigator Entity:Instance

MAX 10: 10M50DAF484C6GES DE10_LITE_Default

Tasks Compilation

Task

- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
- Assembler (Generate programming files)

Embed

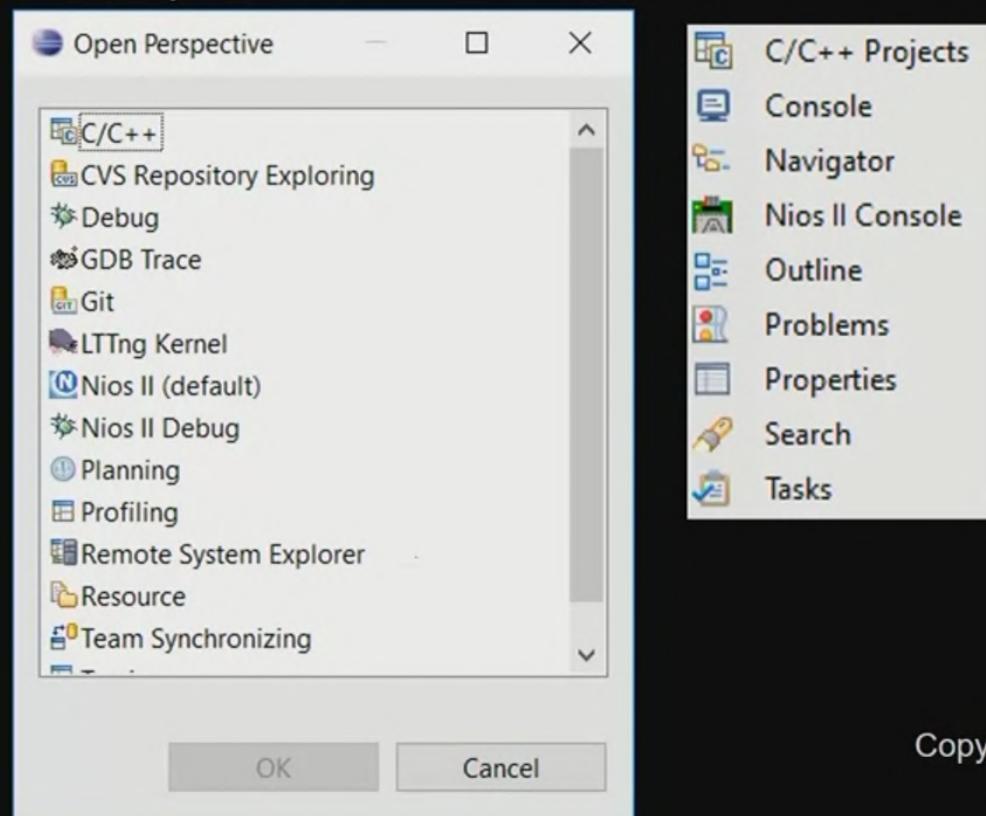
- Run Simulation Tool
- Generate Simulator Setup Script for IP...
- Launch Simulation Library Compiler
- Launch Design Space Explorer II
- TimeQuest Timing Analyzer
- Advisors
- Chip Planner
- Design Partition Planner
- Netlist Viewers
- SignalTap II Logic Analyzer
- In-System Memory Content Editor
- Logic Analyzer Interface Editor
- In-System Sources and Probes Editor
- SignalProbe Pins...
- Programmer
- JTAG Chain Debugger
- Fault Injection Debugger
- System Debugging Tools
- IP Catalog
- Nios II Software Build Tools for Eclipse
- Qsys
- Tcl Scripts

Default.v

```
Generated by Terasic System Builder
multC
    //////////////
    ADC_CLK_10,
    MAX10_CLK1_50,
    MAX10_CLK2_50,
    //////////////
    [2:0] DRAM_ADDR,
    DRAM_BA,
    DRAM_CAS_N,
    DRAM_CKE,
    DRAM_CLK,
    DRAM_CS_N,
    [1:0] DRAM_DQ,
    DRAM_LDQM,
    DRAM_RAS_N,
    DRAM_UDQM,
    DRAM_WE_N,
    //////////////
    [5:0] HEX0,
    [7:0] HEX1,
    [7:0] HEX2,
    [7:0] HEX3,
    [7:0] HEX4,
    [7:0] HEX5
```

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



The screenshot shows the Eclipse for NIOS IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Nios II, Run, Window, and Help. The left pane displays the code for `hello_world_small.c`:

```
hello_world_small.c
/*
 * "Small Hello World" example.
 */
#include "sys/alt_stdio.h"

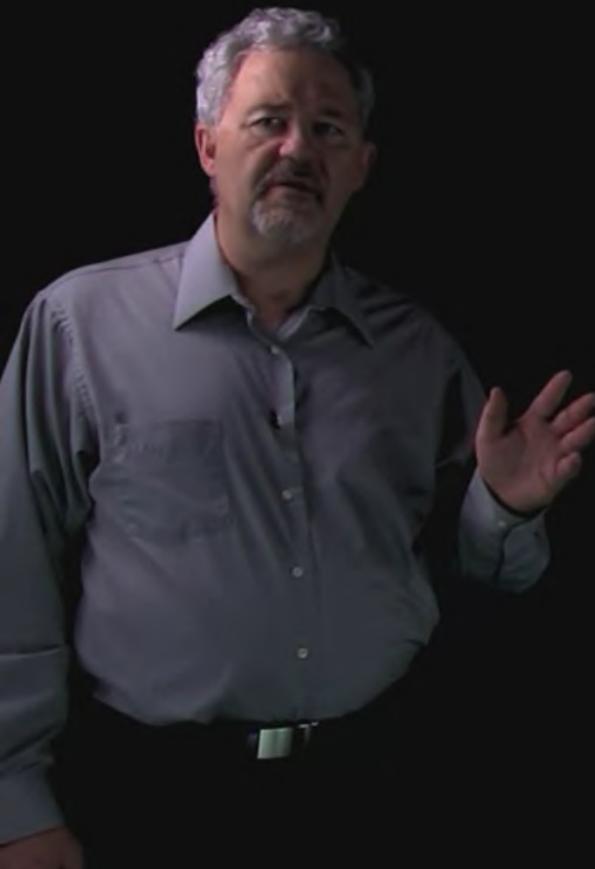
int main()
{
    alt_putstr("Hello from Nios II!\n");
    /* Event loop never exits. */
    while (1);

    return 0;
}
```

The bottom pane shows the CDT Build Console output:

```
CDT Build Console [HelloNios2]
17:32:29 **** Clean-only build of configuration Nios II for project HelloNios2 ****
make clean
[HelloNios2 clean complete]
17:32:29 Build Finished (took 150ms)
```





- 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

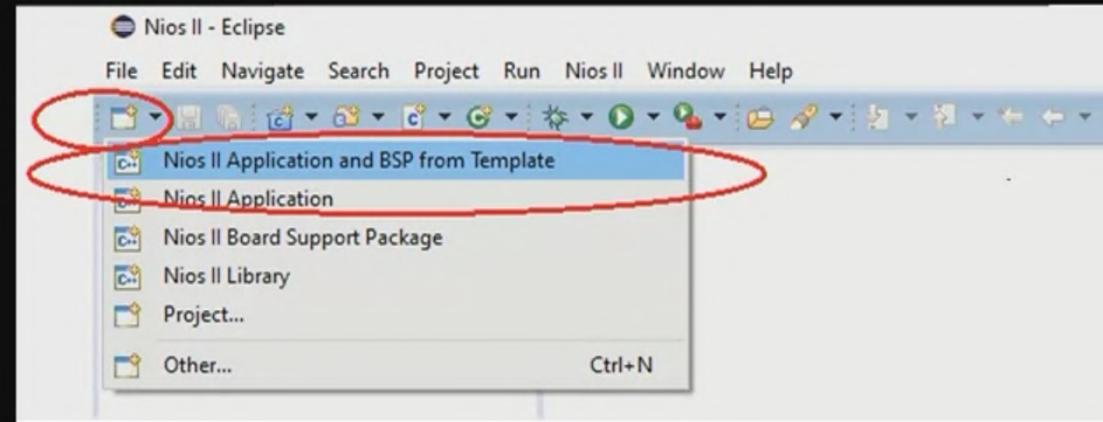
<https://help.eclipse.org/2020-06/index.jsp>

Copyright © 2020 University of Colorado

Starting a Software Project



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



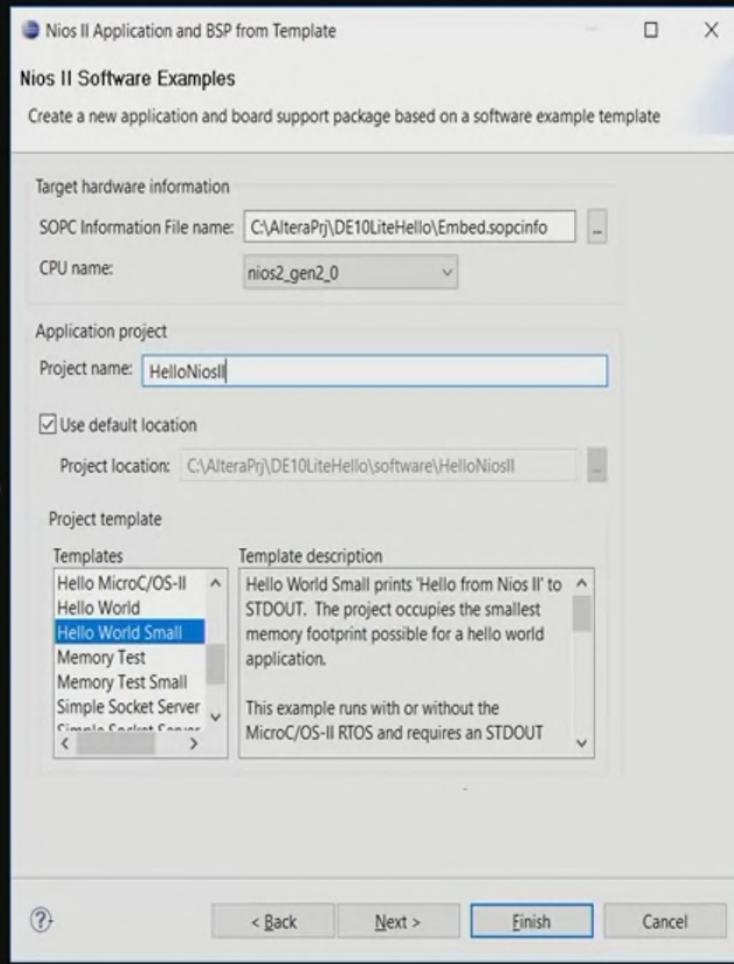
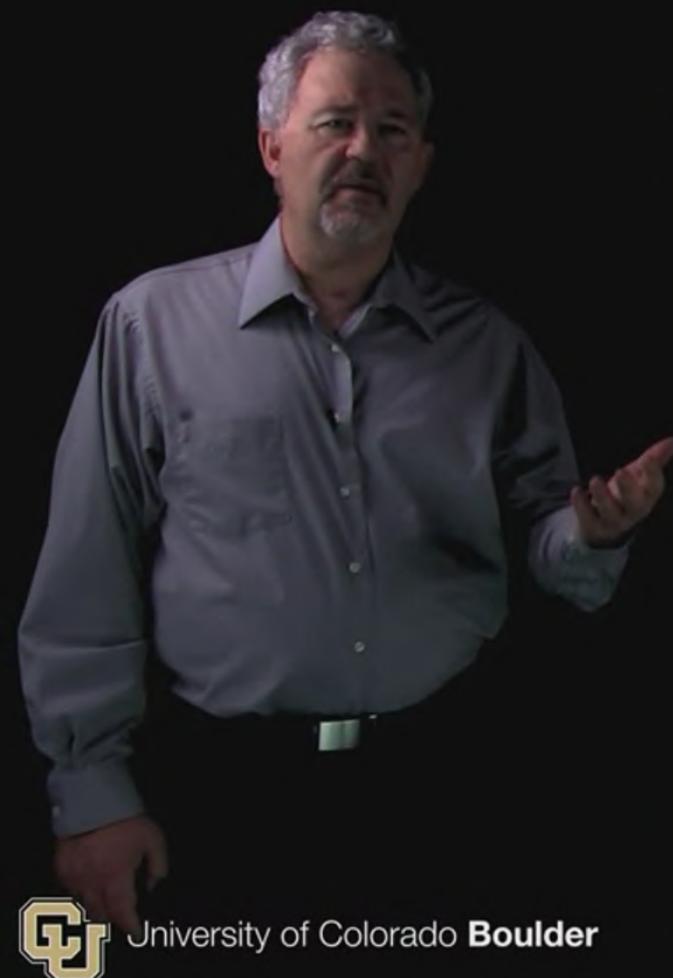
<https://help.eclipse.org/2020-06/index.jsp>

Copyright © 2020 University of Colorado



University of Colorado **Boulder**

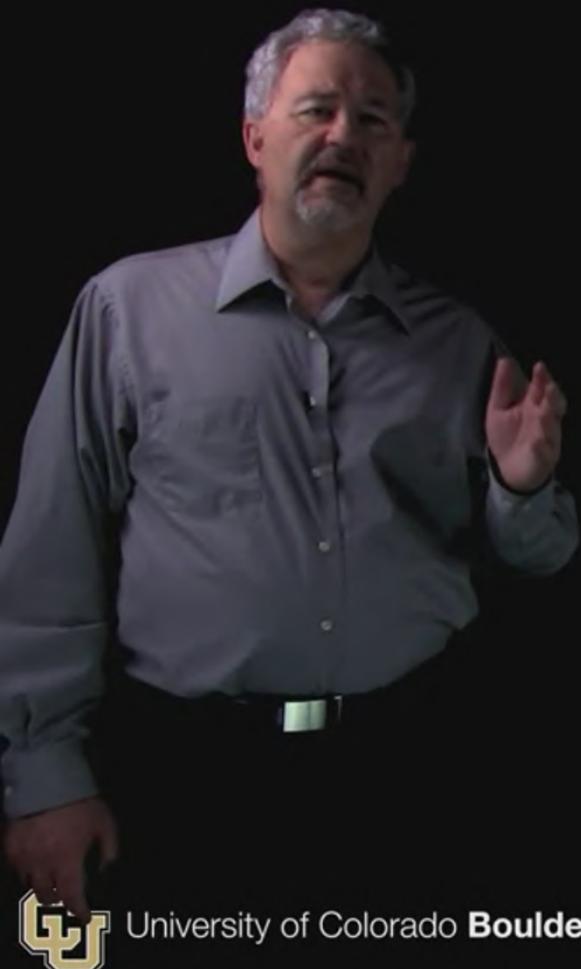
Starting a Software Project



- **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

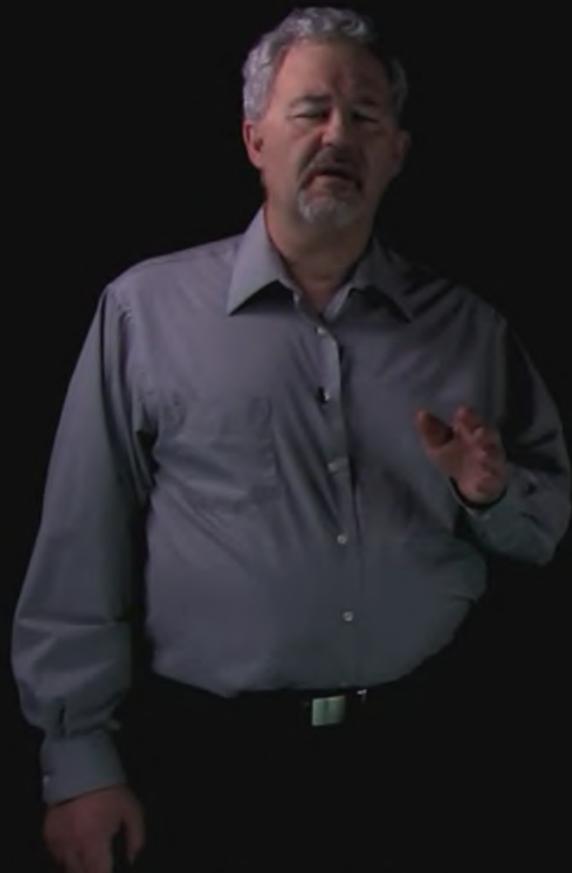
- [1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook*. [Online]. Available: <https://www.intel.com/content/www/us/en/programmable/products/processors/support.html>
- [2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial*. [Online]. Available: <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



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.

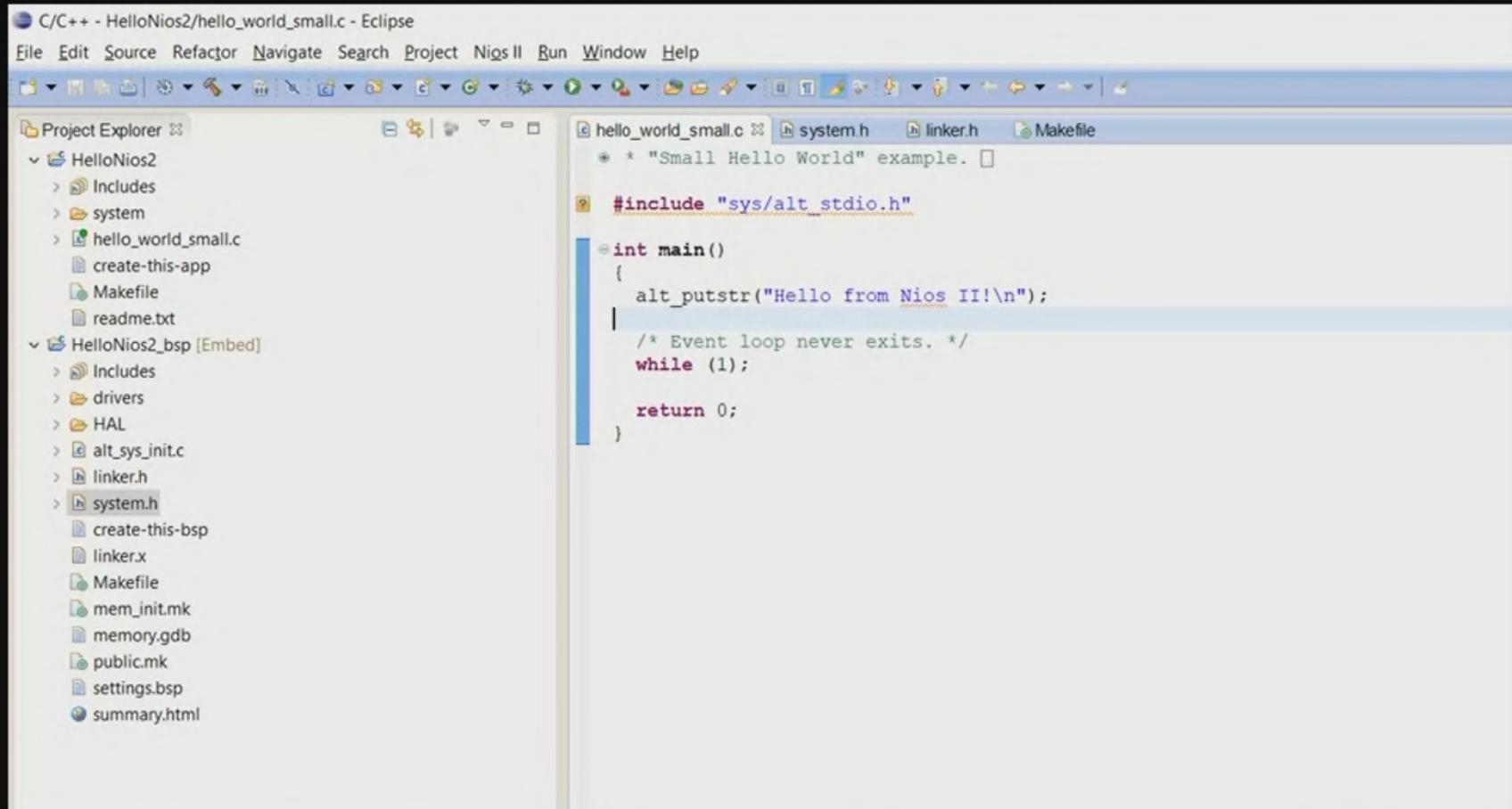
C programming for Soft Processors

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



The screenshot shows the Eclipse C/C++ IDE interface. The title bar reads "C/C++ - HelloNios2/hello_world_small.c - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Nios II, Run, Window, and Help. The toolbar has various icons for file operations. The left pane is the "Project Explorer" showing the project structure:

- HelloNios2
 - Includes
 - system
 - hello_world_small.c
 - create-this-app
 - Makefile
 - readme.txt
- HelloNios2_bsp [Embed]
 - Includes
 - drivers
 - HAL
 - alt_sys_init.c
 - linker.h
 - system.h
 - create-this-bsp
 - linker.x
 - Makefile
 - mem_init.mk
 - memory.gdb
 - public.mk
 - settings.bsp
 - summary.html

The right pane displays the source code for "hello_world_small.c". The code includes a header file "sys/alt_stdio.h" and defines the main function:

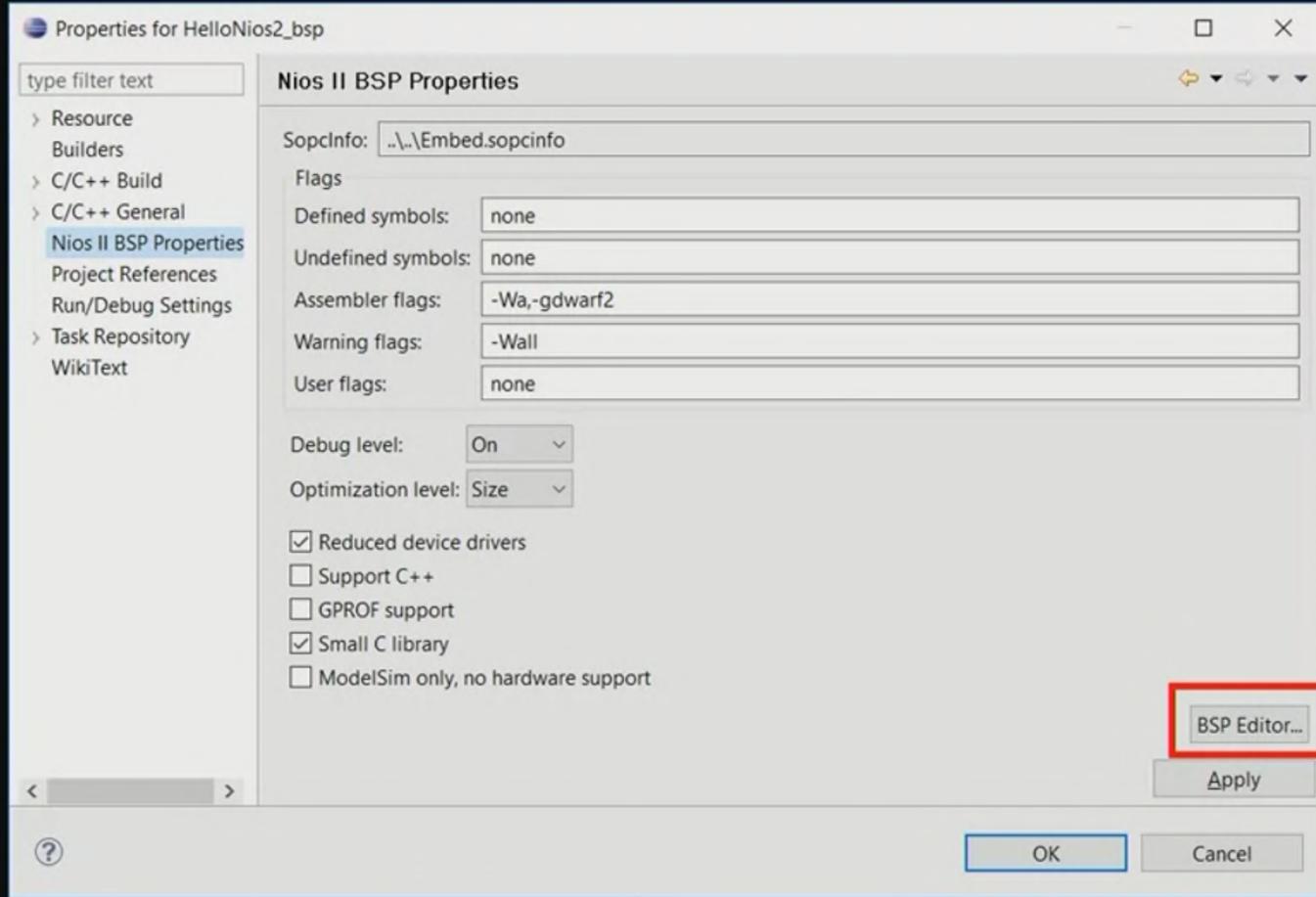
```
#include "sys/alt_stdio.h"

int main()
{
    alt_putstr("Hello from Nios II!\n");

    /* Event loop never exits. */
    while (1);

    return 0;
}
```

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

The screenshot shows the BSP Editor interface with the following configuration details:

- hal** settings:
 - sys_clk_timer: timer_0
 - timestamp_timer: none
 - stdin: jtag_uart
 - stdout: jtag_uart
 - stderr: jtag_uart
- hal.linker** settings:
 - enable_small_c_library
 - enable_gprof
 - enable_reduced_device_drivers
 - enable_sim_optimize
 - enable_exception_stack

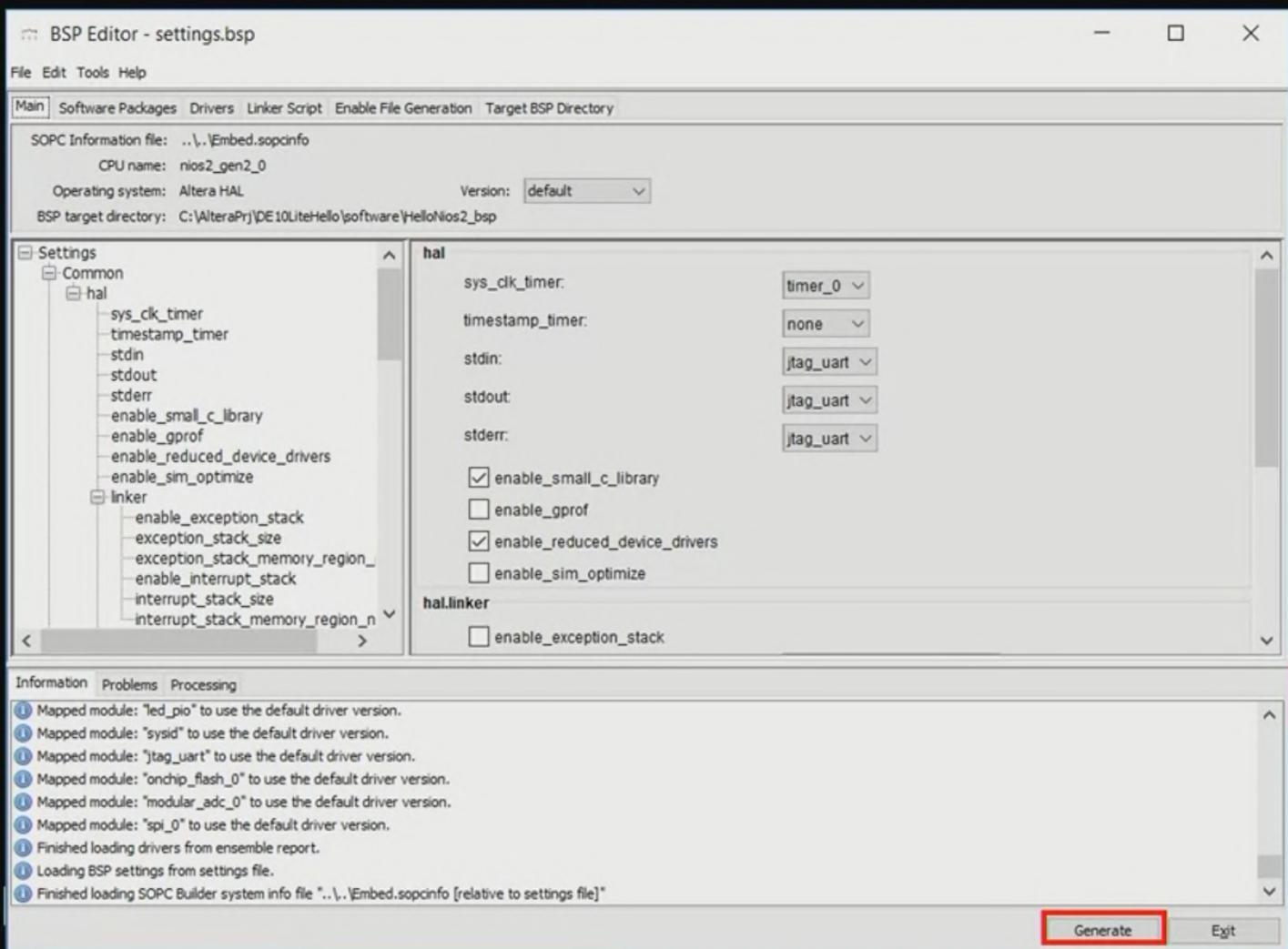
Information tab content (log messages):

- ① Mapped module: "led_pio" to use the default driver version.
- ① Mapped module: "sysid" to use the default driver version.
- ① Mapped module: "jtag_uart" to use the default driver version.
- ① Mapped module: "onchip_flash_0" to use the default driver version.
- ① Mapped module: "modular_adc_0" to use the default driver version.
- ① Mapped module: "spi_0" to use the default driver version.
- ① Finished loading drivers from ensemble report.
- ① Loading BSP settings from settings file.
- ① Finished loading SOPC Builder system info file "..\..\Embed.sopcinfo [relative to settings file]"

With the BSP Editor, the programmer can

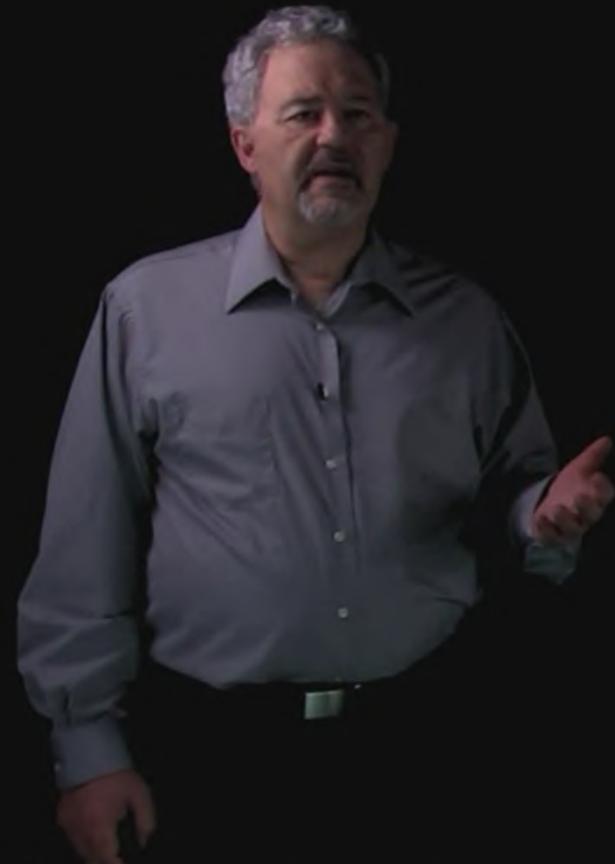
- Control system timers
- Specify stderr/in/out (in this case to the jtag uart which will show on the console)
- Control code size (small library and drivers chosen here)
- Enable and size stacks

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/>

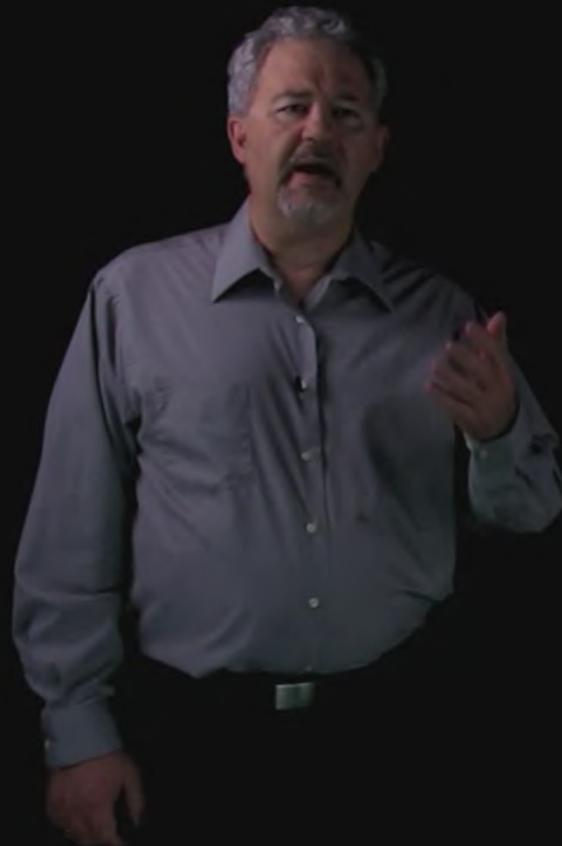


FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition



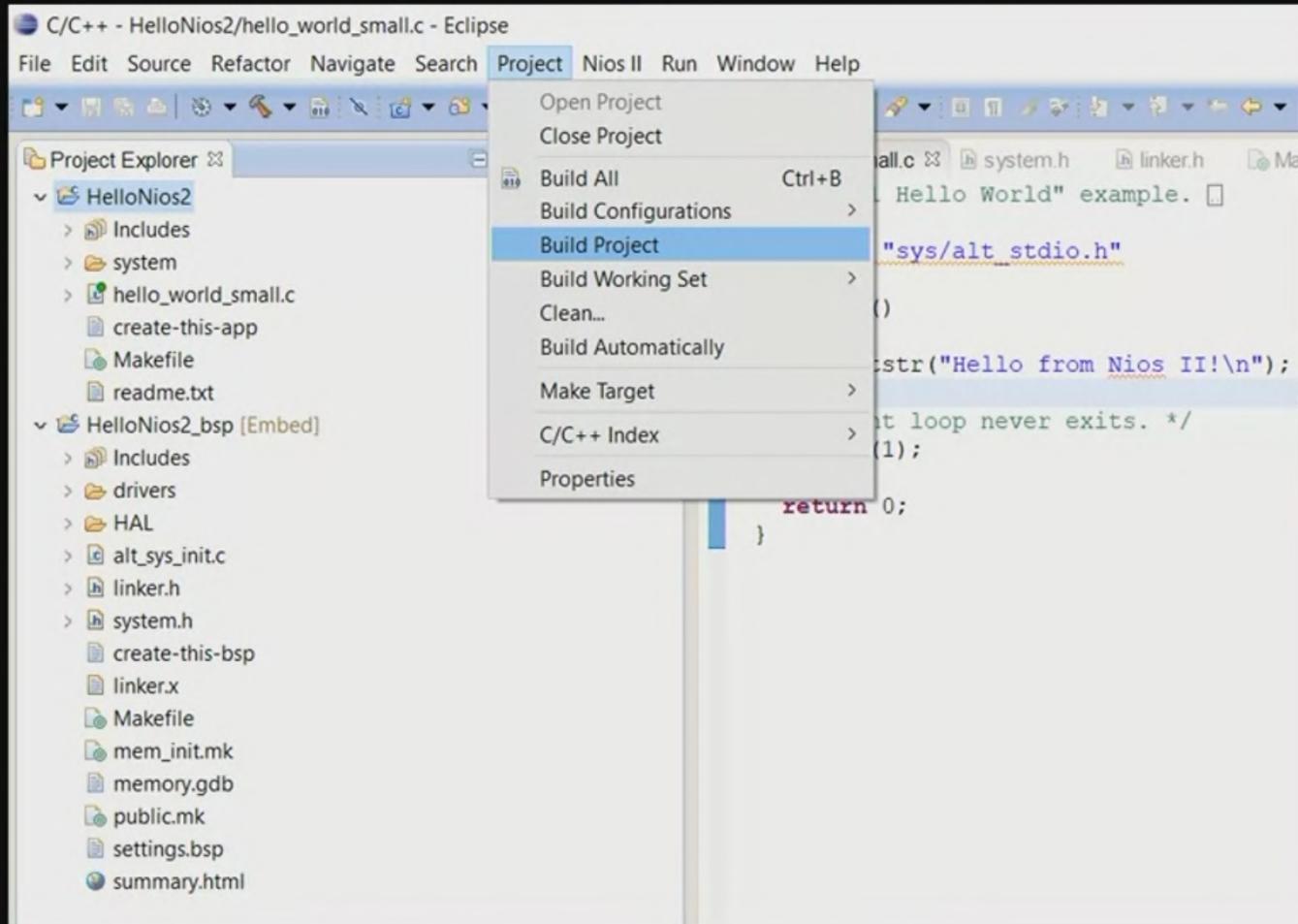
Building C Programs



In this video, you will learn:

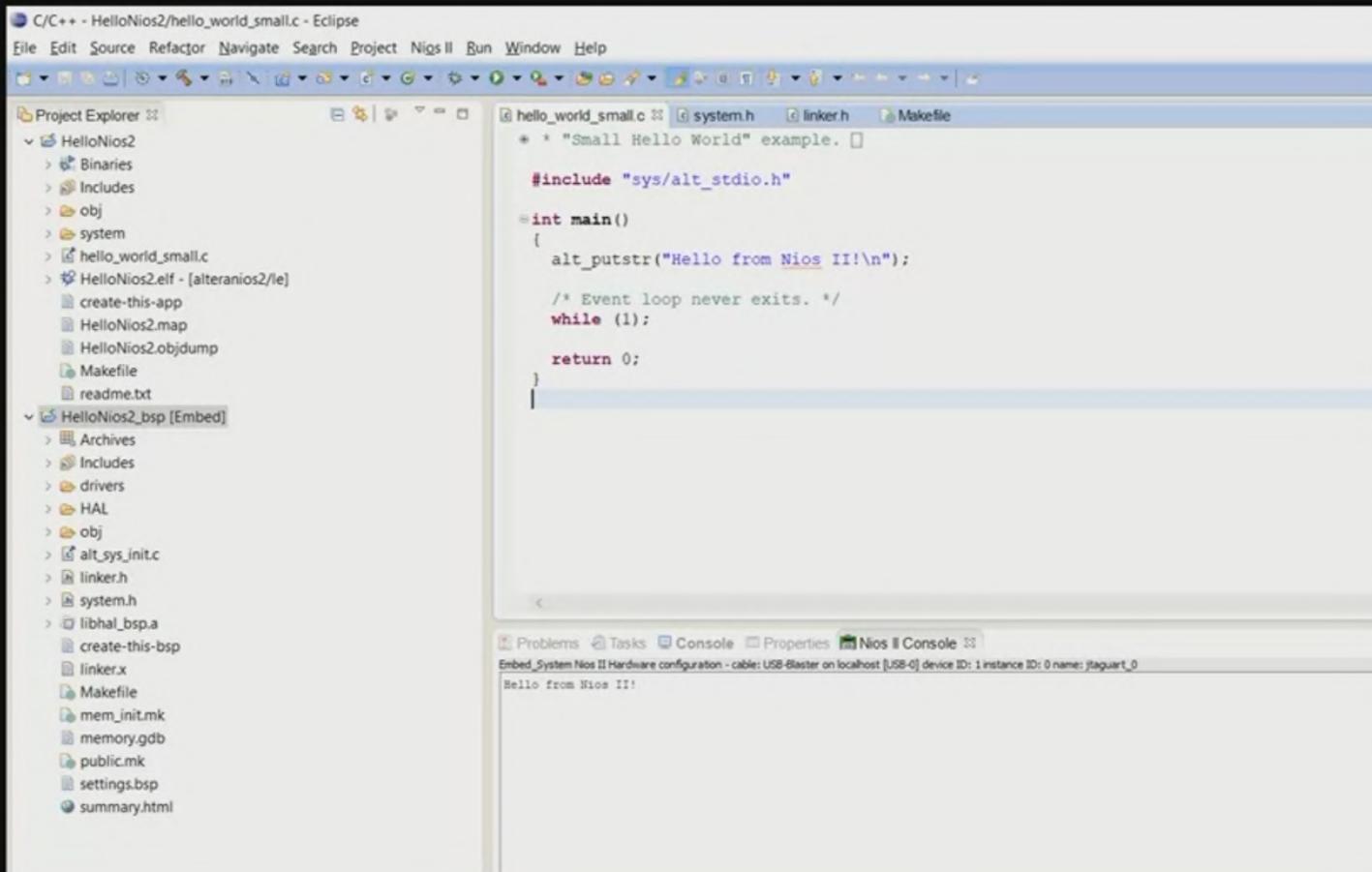
- How to build and run C programs using the NIOS II soft processor.
- How to handle potential issues in Run Configuration
- How to start a Debugging session in the Debug Perspective

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



The screenshot shows the Eclipse C/C++ IDE interface. The title bar reads "C/C++ - HelloNios2/hello_world_small.c - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Nios II, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left shows a project structure for "HelloNios2" and "HelloNios2_bsp [Embed]". The "HelloNios2" folder contains Binaries, Includes, obj, system, hello_world_small.c, HelloNios2.elf, create-this-app, HelloNios2.map, HelloNios2.objdump, Makefile, and readme.txt. The "HelloNios2_bsp [Embed]" folder contains Archives, Includes, drivers, HAL, obj, alt_sys_init.c, linker.h, system.h, libhal_bsp.a, create-this-bsp, linker.x, Makefile, mem_init.mk, memory.gdb, public.mk, settings.bsp, and summary.html. The central workspace shows the file "hello_world_small.c" with the following code:

```
* * "Small Hello World" example.

#include "sys/alt_stdio.h"

int main()
{
    alt_putstr("Hello from Nios II!\n");

    /* Event loop never exits. */
    while (1);

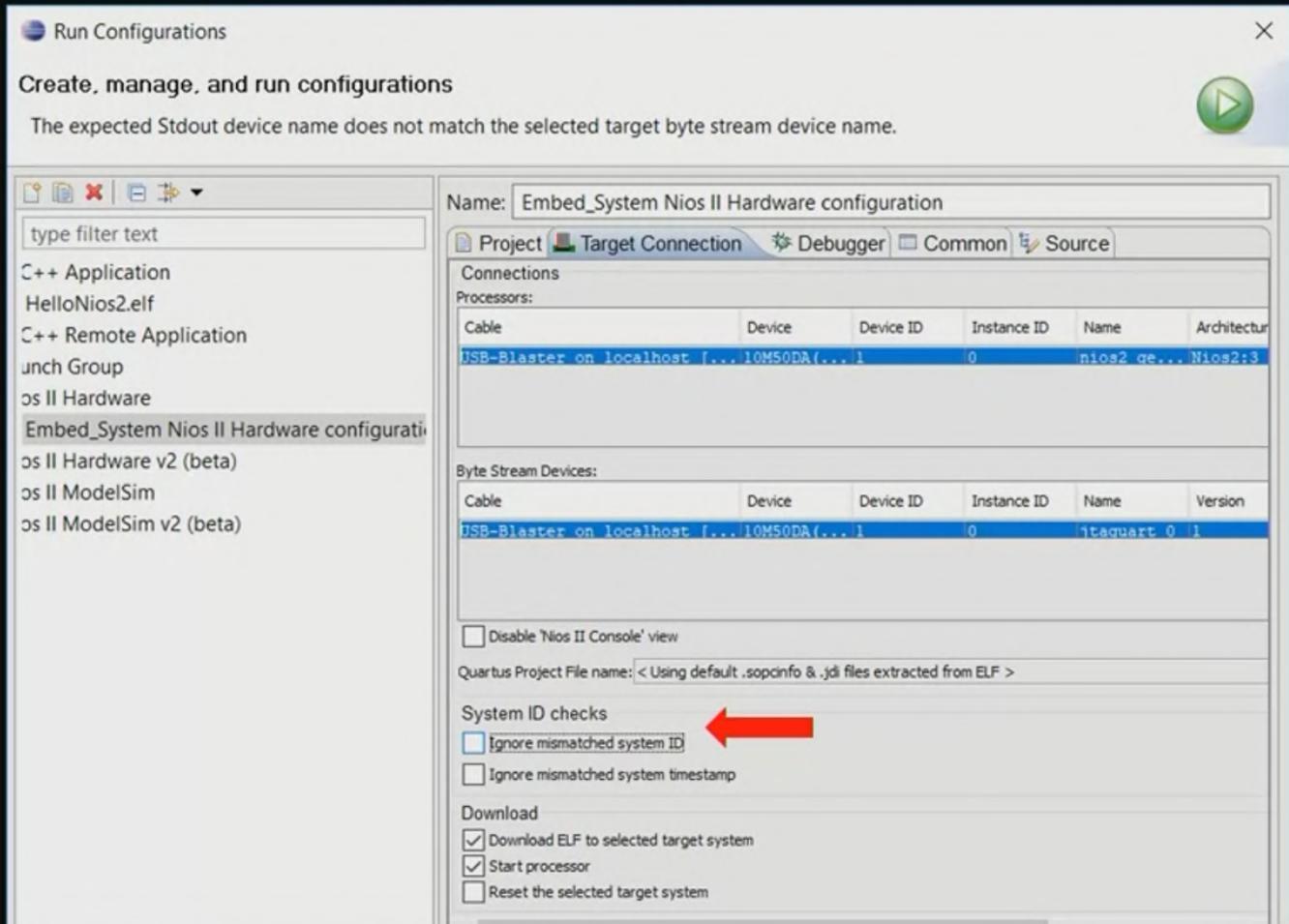
    return 0;
}
```

The bottom right pane is the "Nios II Console" showing the output: "Hello from Nios II!". The tabs at the bottom of the console pane are Problems, Tasks, Console, Properties, and Nios II Console.

- 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.



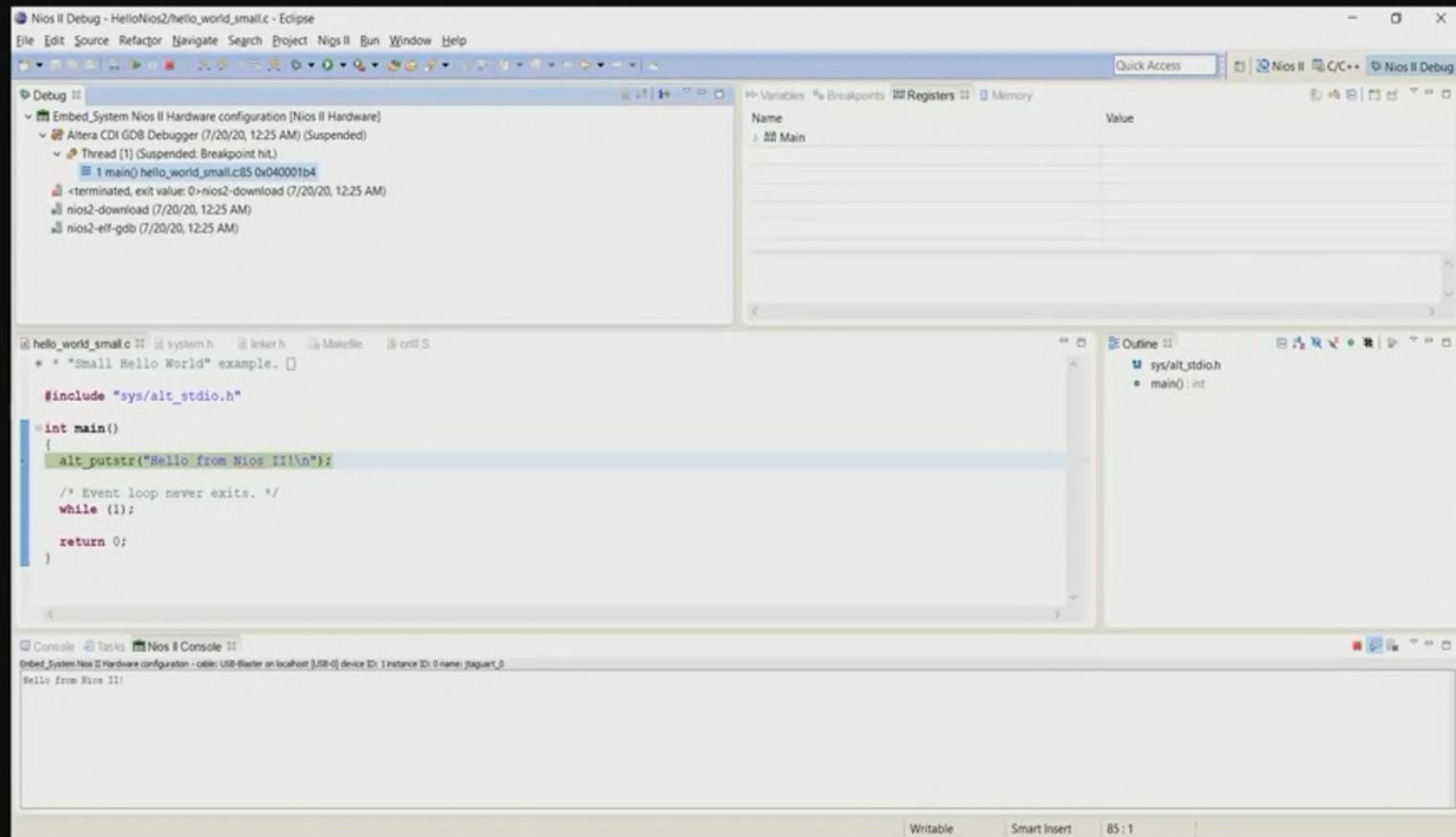
Running Programs for Nios II



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



Debugging Programs for Nios II



Nios II Debug - HelloNios2/hello_world_small.c - Eclipse

File Edit Source Refactor Navigate Project Nios II Run Window Help

Debug

Embed_System Nios II Hardware configuration [Nios II Hardware]

Altera CDI GDB Debugger (7/20/20, 12:25 AM) (Suspended)

Thread [1] (Suspended: Breakpoint hit)

1 main() hello_world_small.c:85 0x040001b4

<terminated, exit value: 0> nios2>download (7/20/20, 12:25 AM)

nios2>download (7/20/20, 12:25 AM)

nios2>elf-gdb (7/20/20, 12:25 AM)

hello_world_small.c

```
#include "sys/alt_stdio.h"

int main()
{
    alt_putstr("Hello from Nios II!\n");

    /* Event loop never exits. */
    while (1);

    return 0;
}
```

Variables

Registers

Memory

Outline

sys/alt_stdio.h

main(): int

Console

Nios II Console

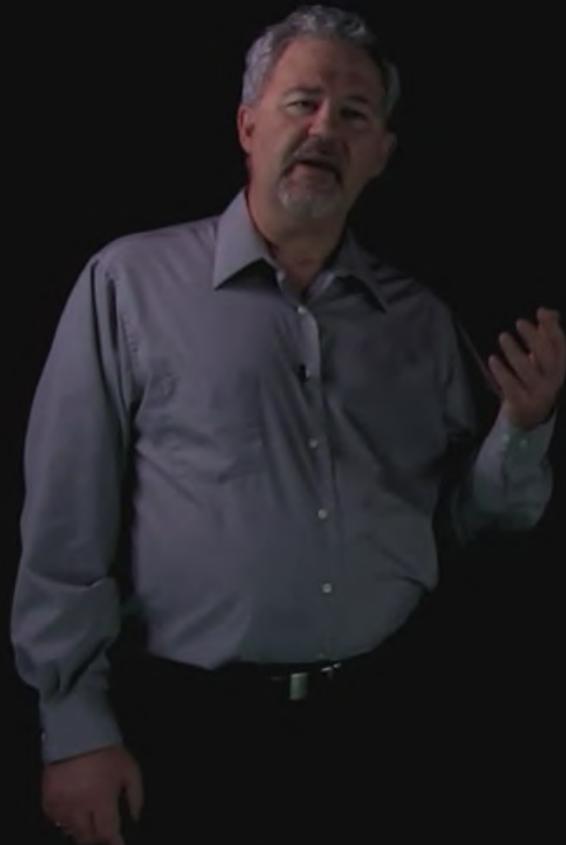
Embed_System Nios II Hardware configuration - cable: USB-Blaster on localhost [JTAG-0] device ID: 1 instance ID: 0 name: jaguar_0

Hello from 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

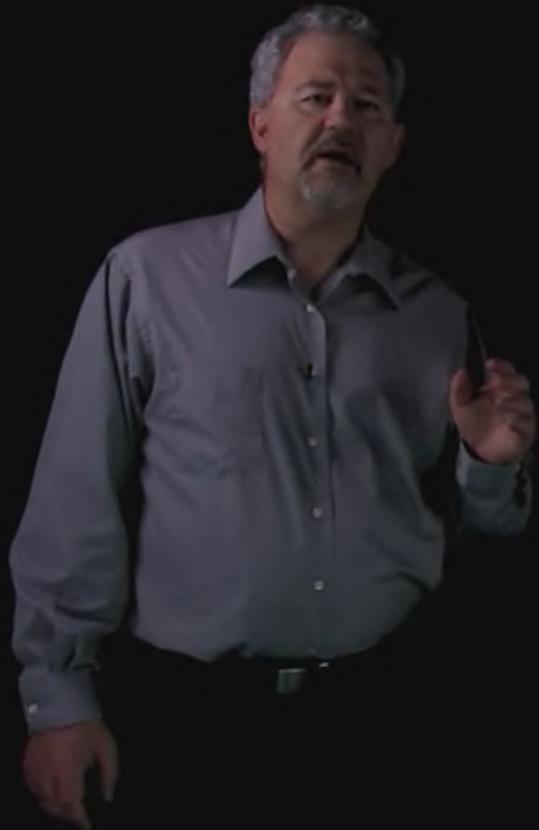
FPGA Softcore Processors and IP Acquisition



University of Colorado **Boulder**

Copyright © 2020 University of Colorado

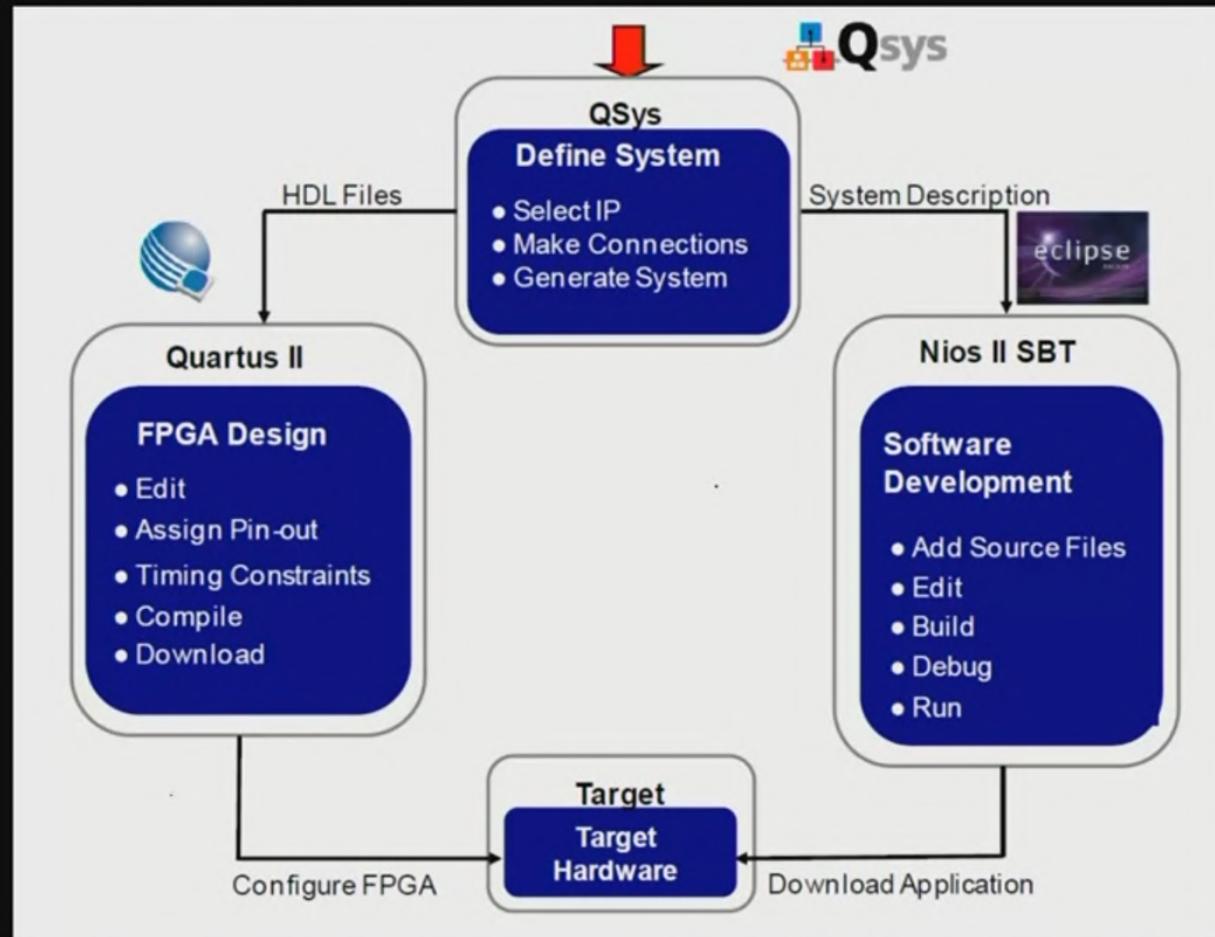
Programming Soft Processors



In this video, you will learn:

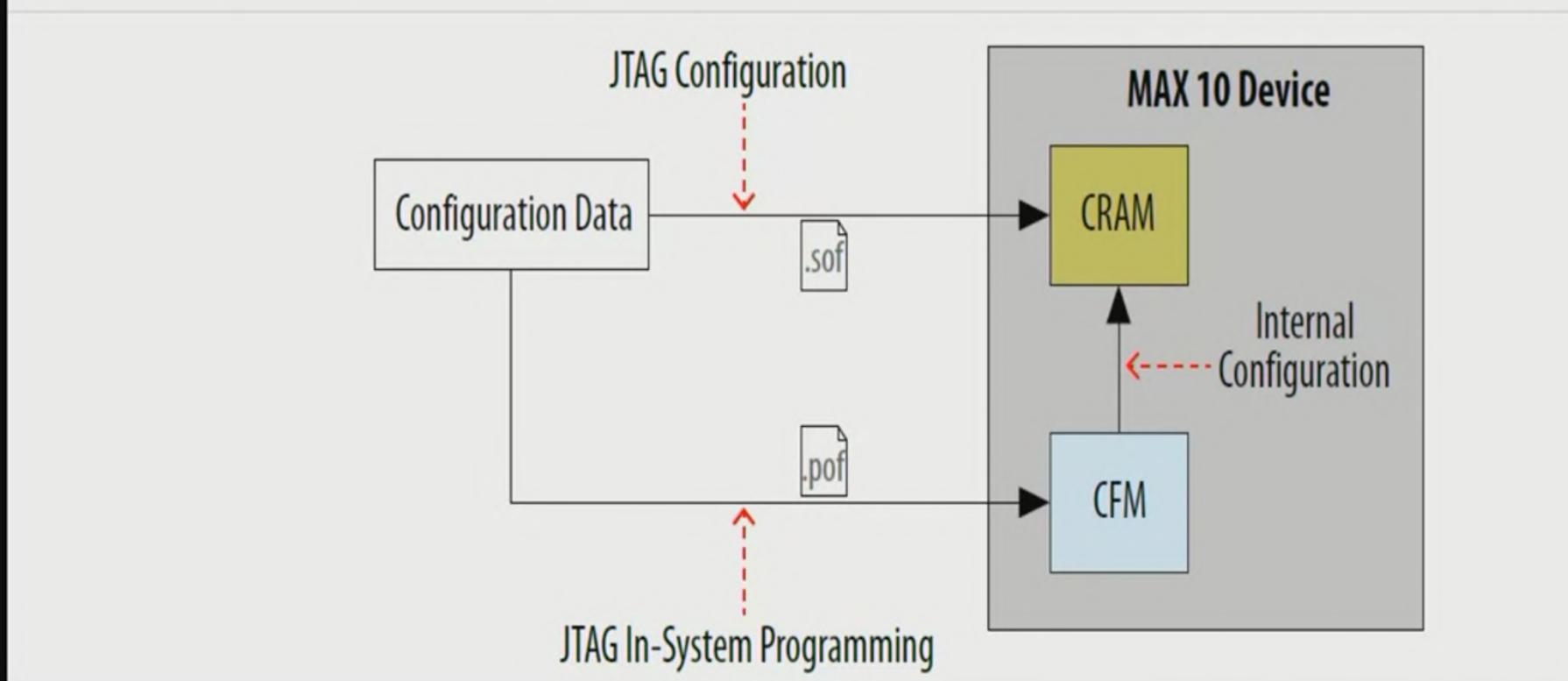
- How to program the target FPGA with hardware and software in the same development session.
- Limitations of the free evaluation version of the Nios II.
- How to resolve basic programming issues

Soft Processor Design Flow

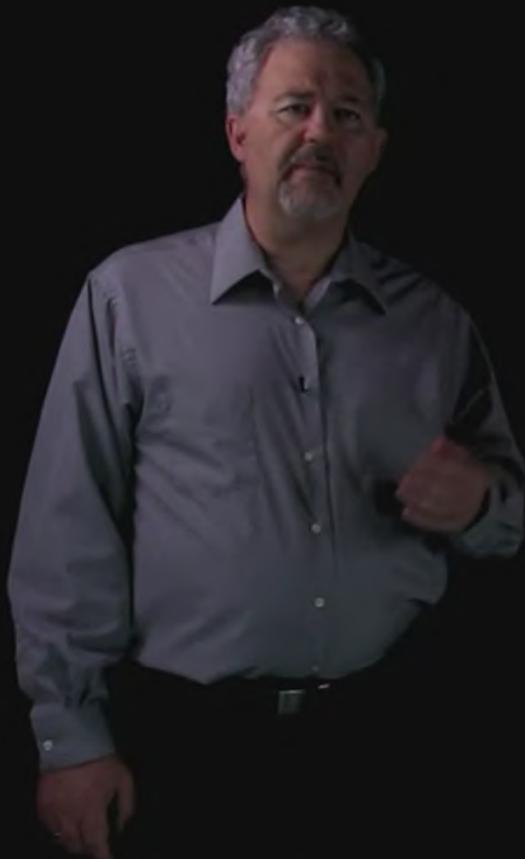


Programming the Nios II

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



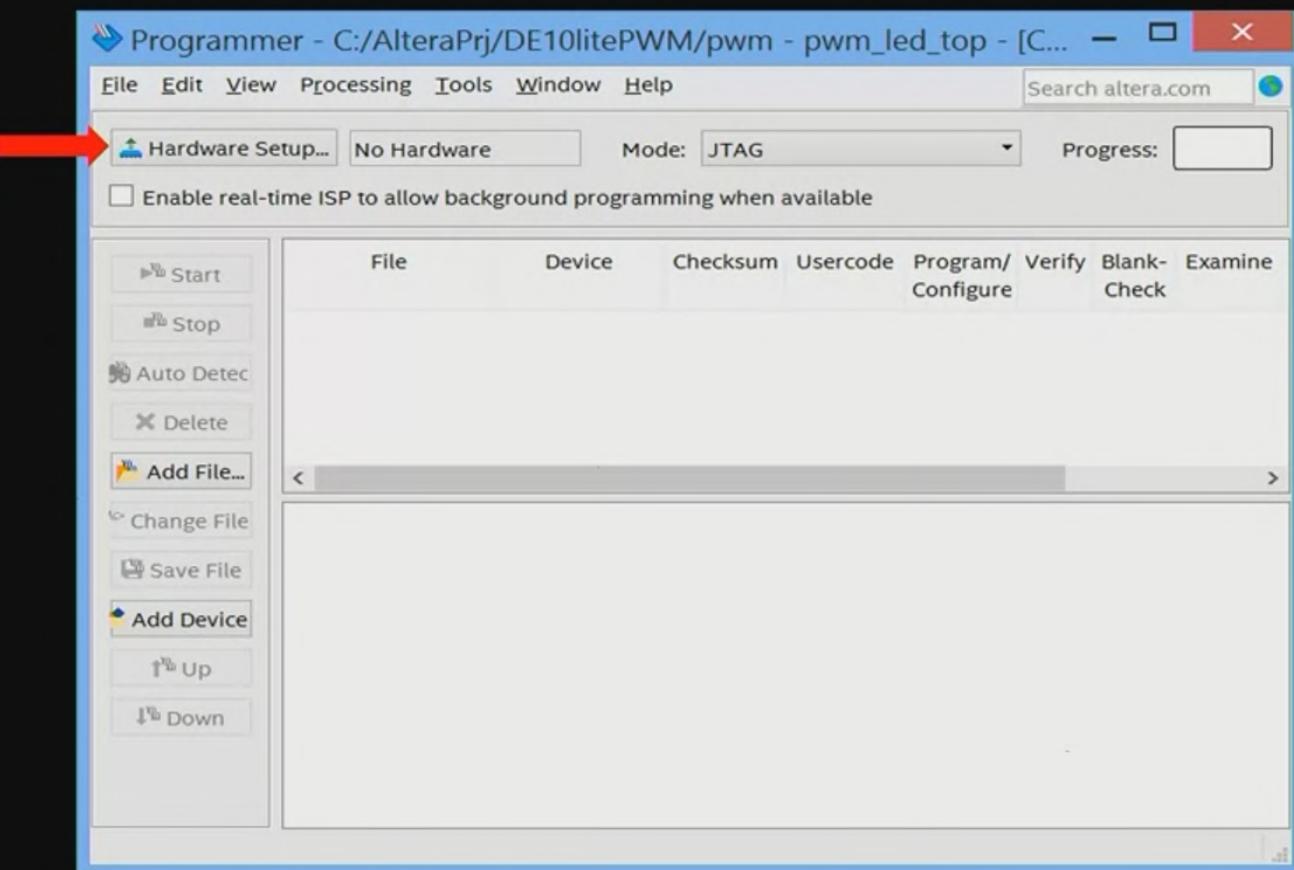
Programming the Nios II



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

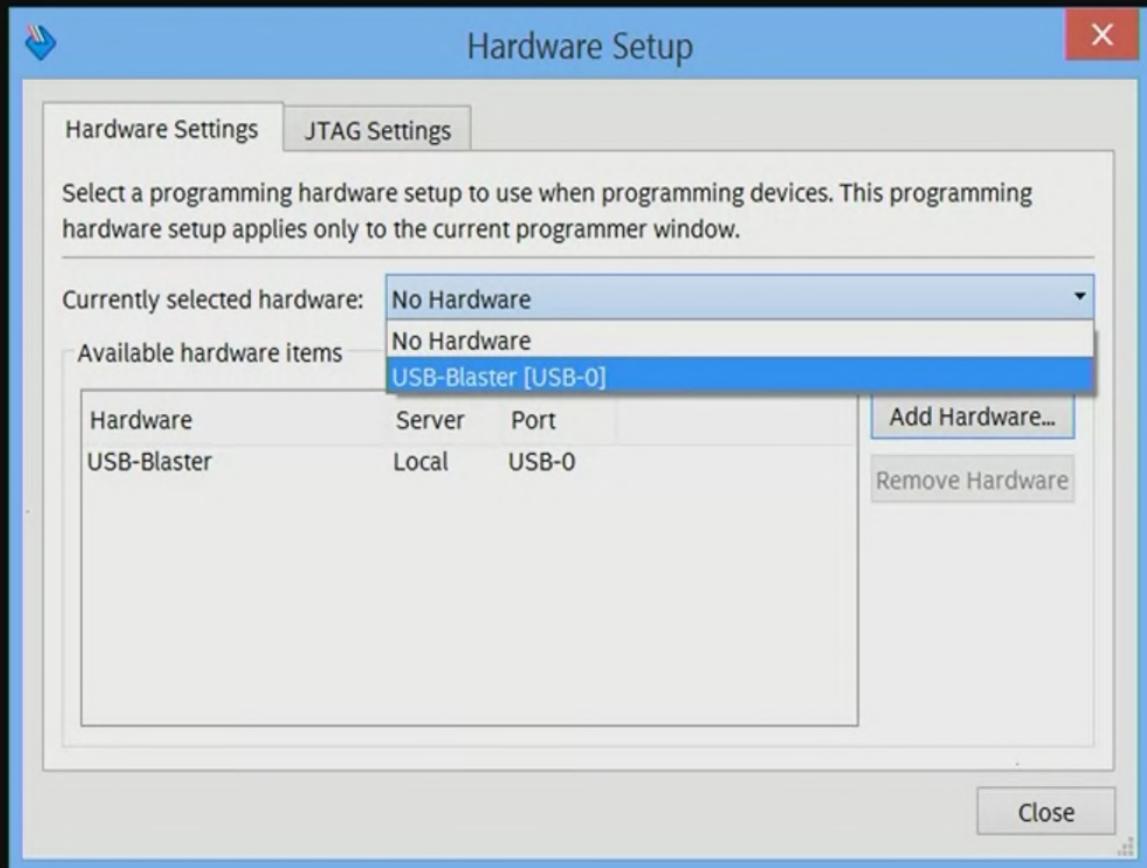
Programming the Nios II



Step 2: Launch the Programmer Tool, select hardware setup



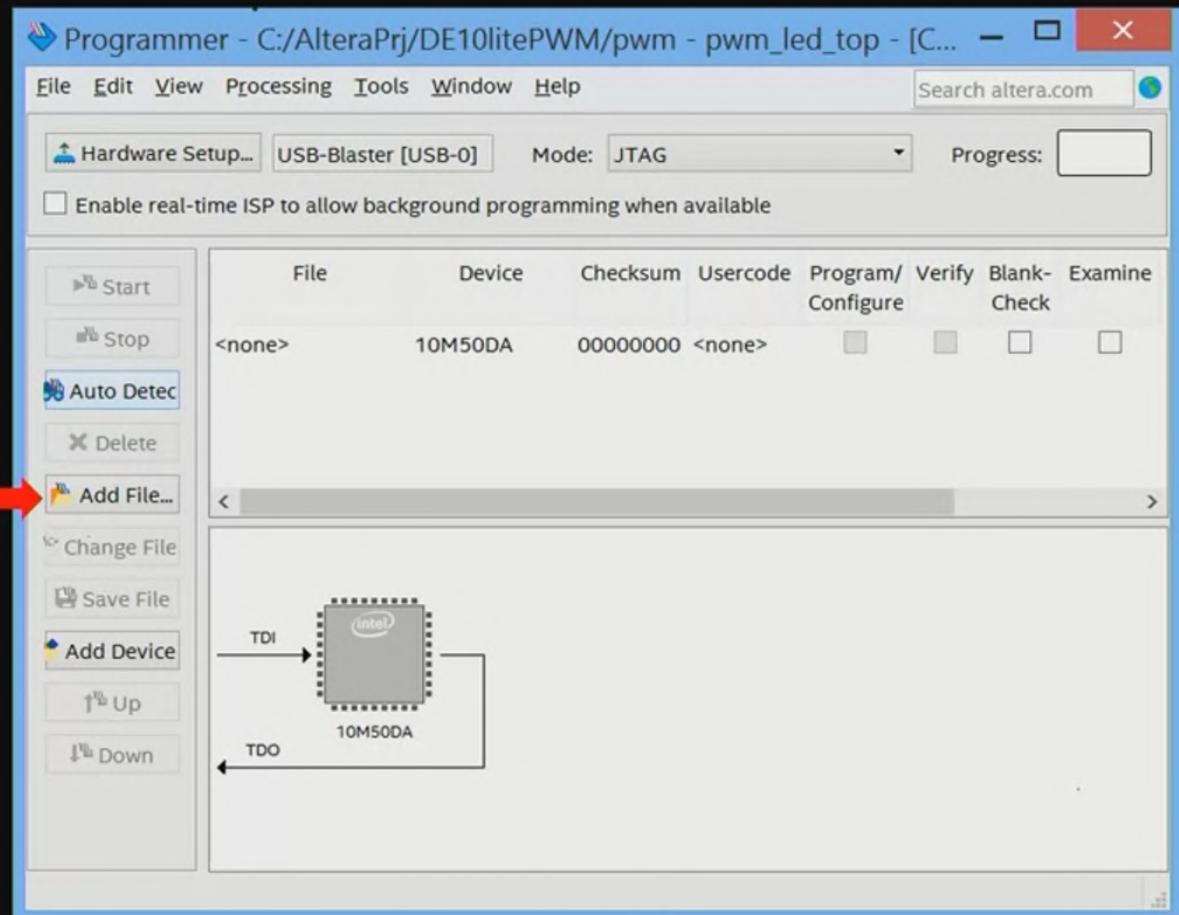
Programming the Nios II



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

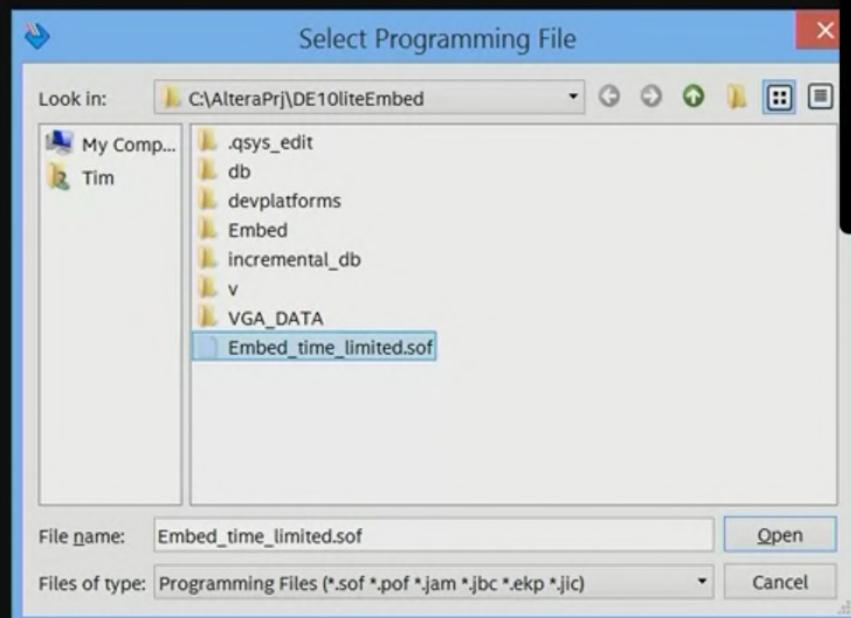


Programming the Nios II

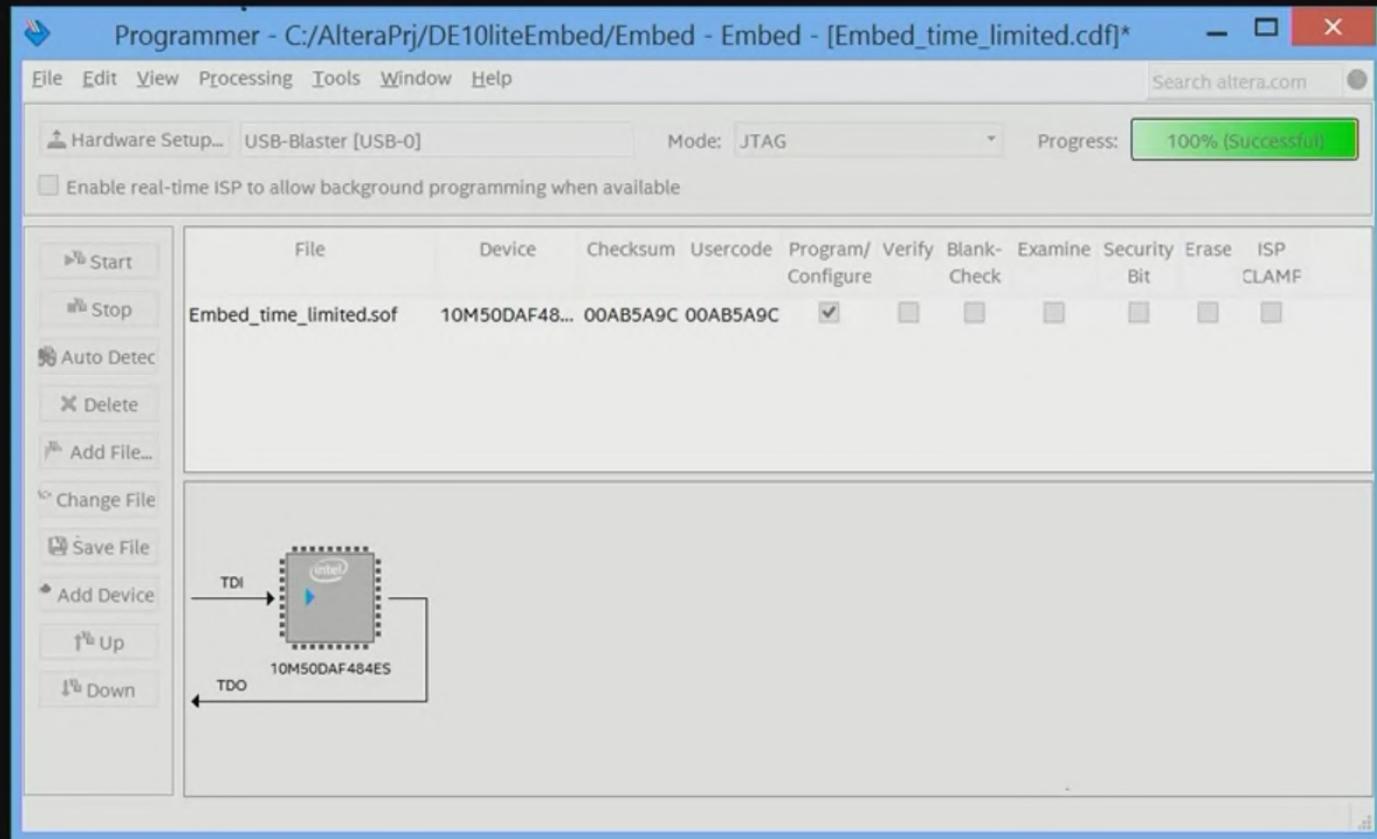


Step 4: To select the programming file, click Add File

Note the file is time-limited.



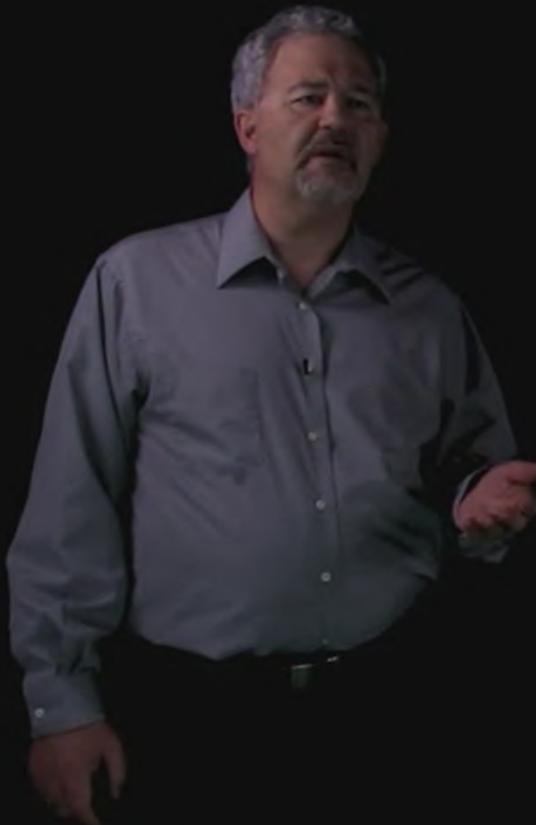
Programming the Nios II



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.



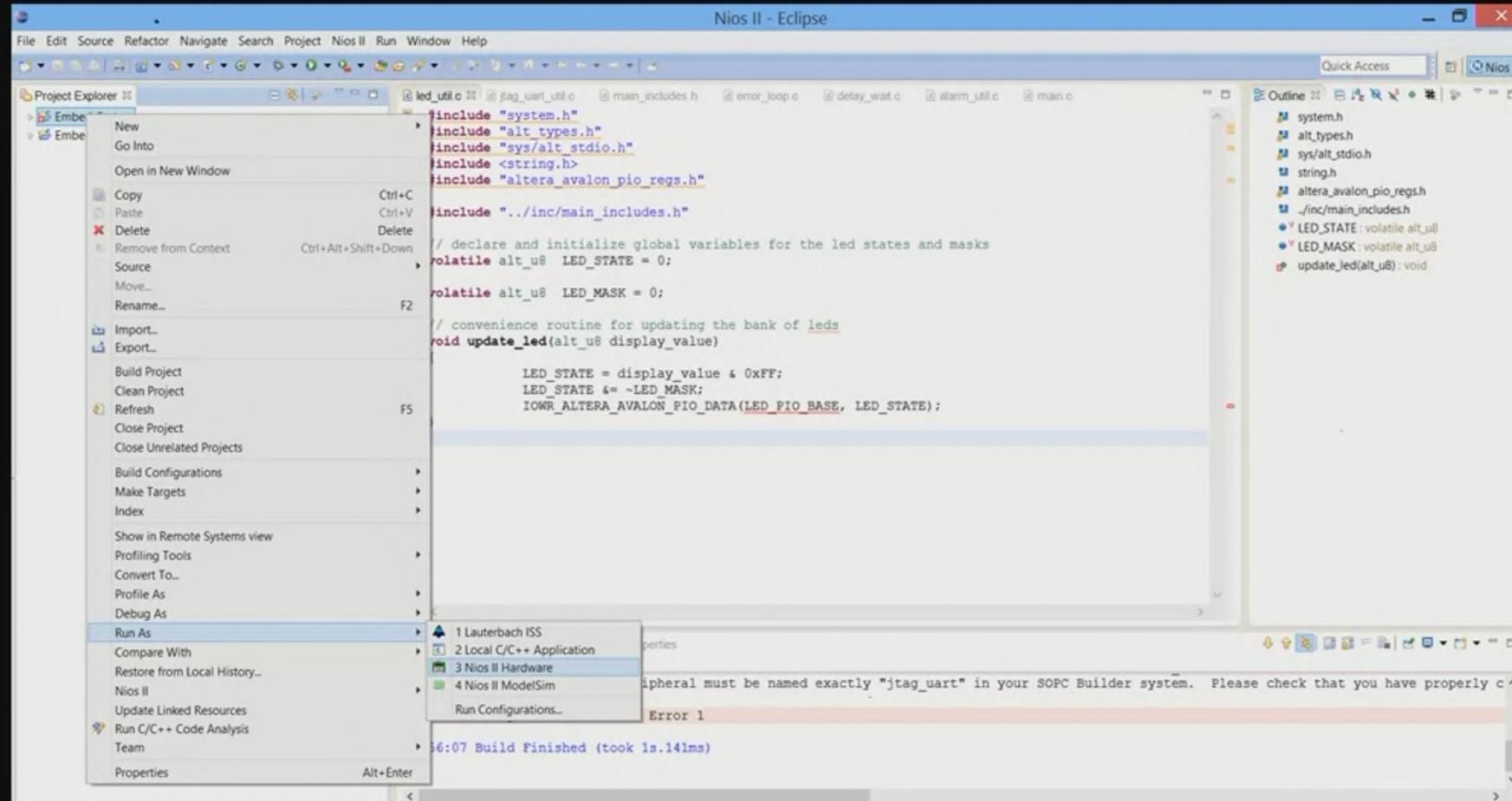
Programming the Nios II - Limitations



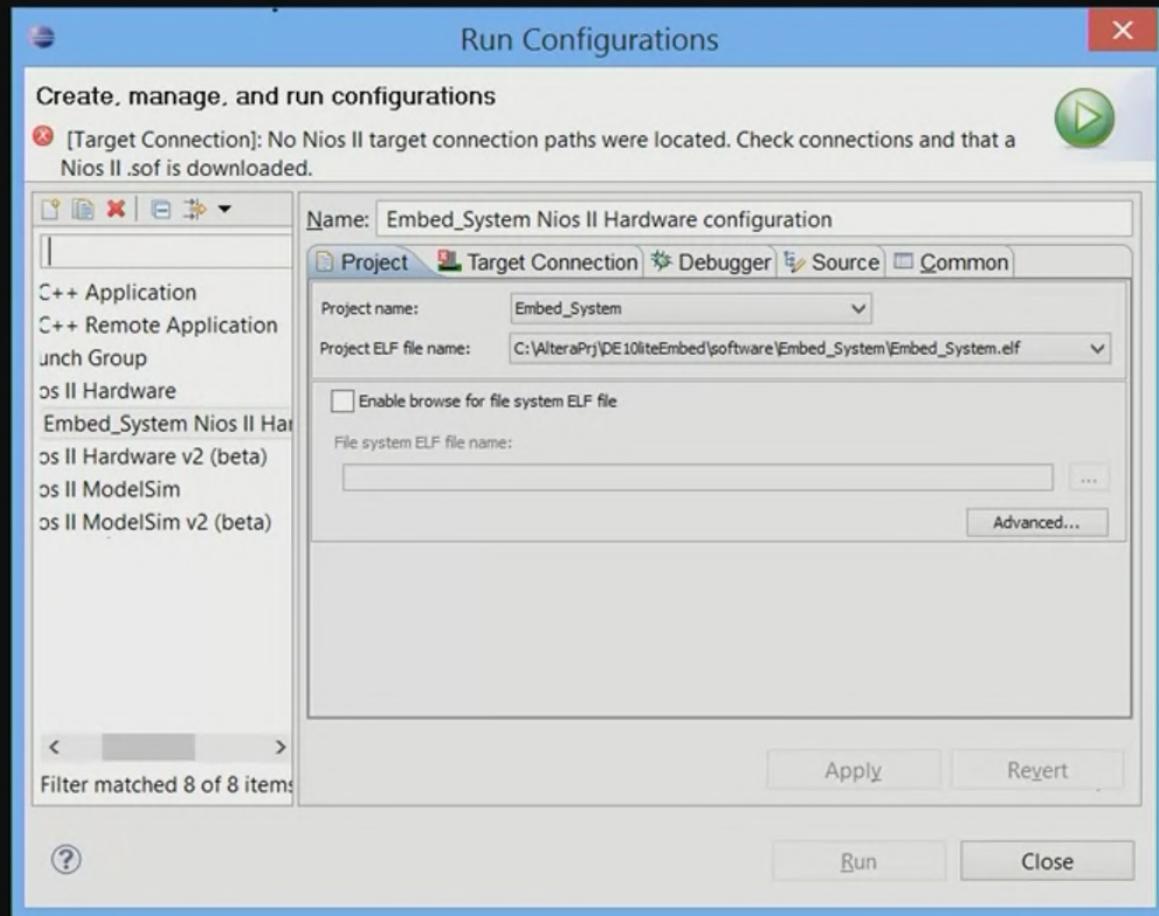
There are some limitations to the use of the NIOS II unless you have a license

- 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 MAX10 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



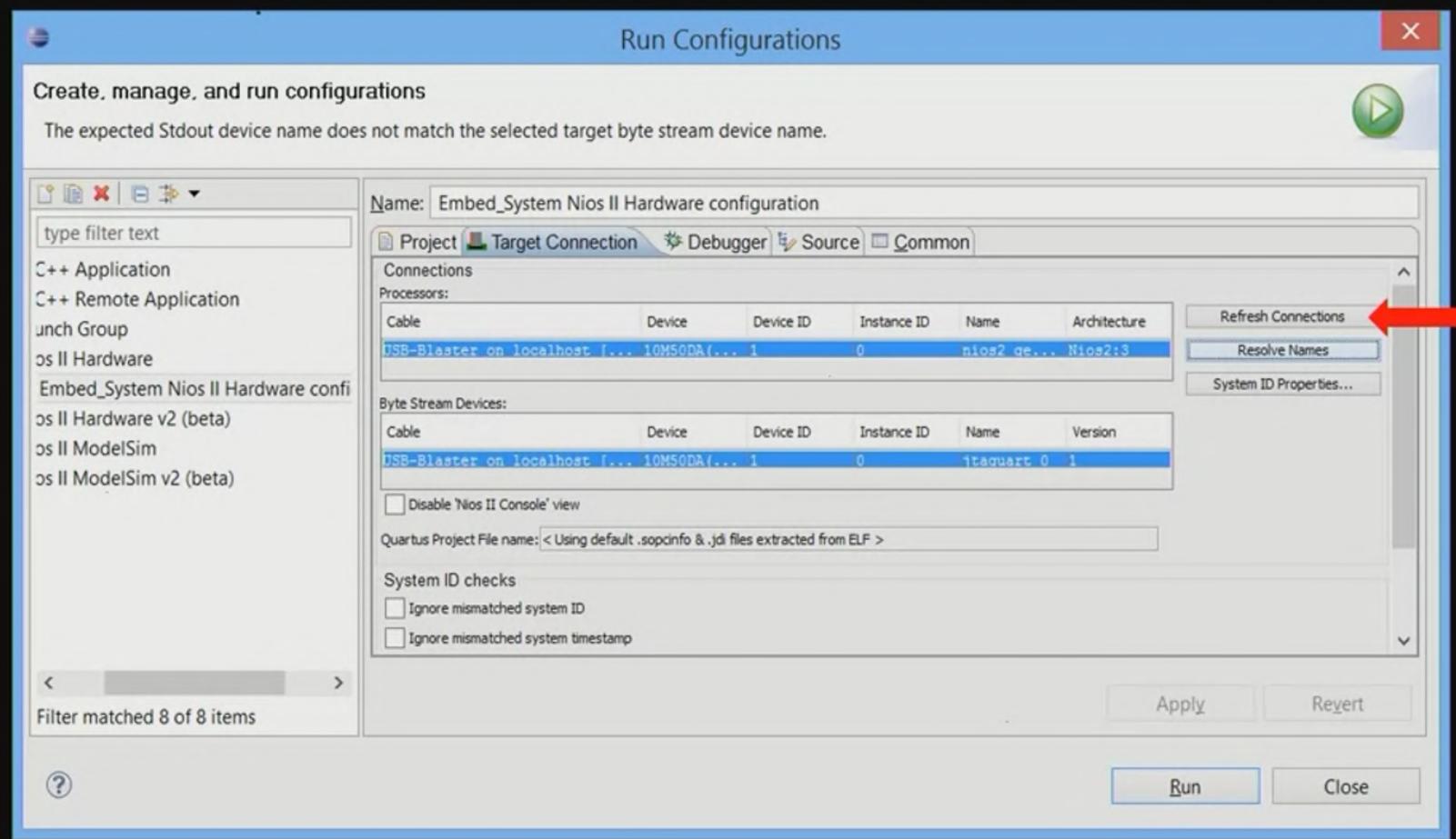
Programming the Nios II - software



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



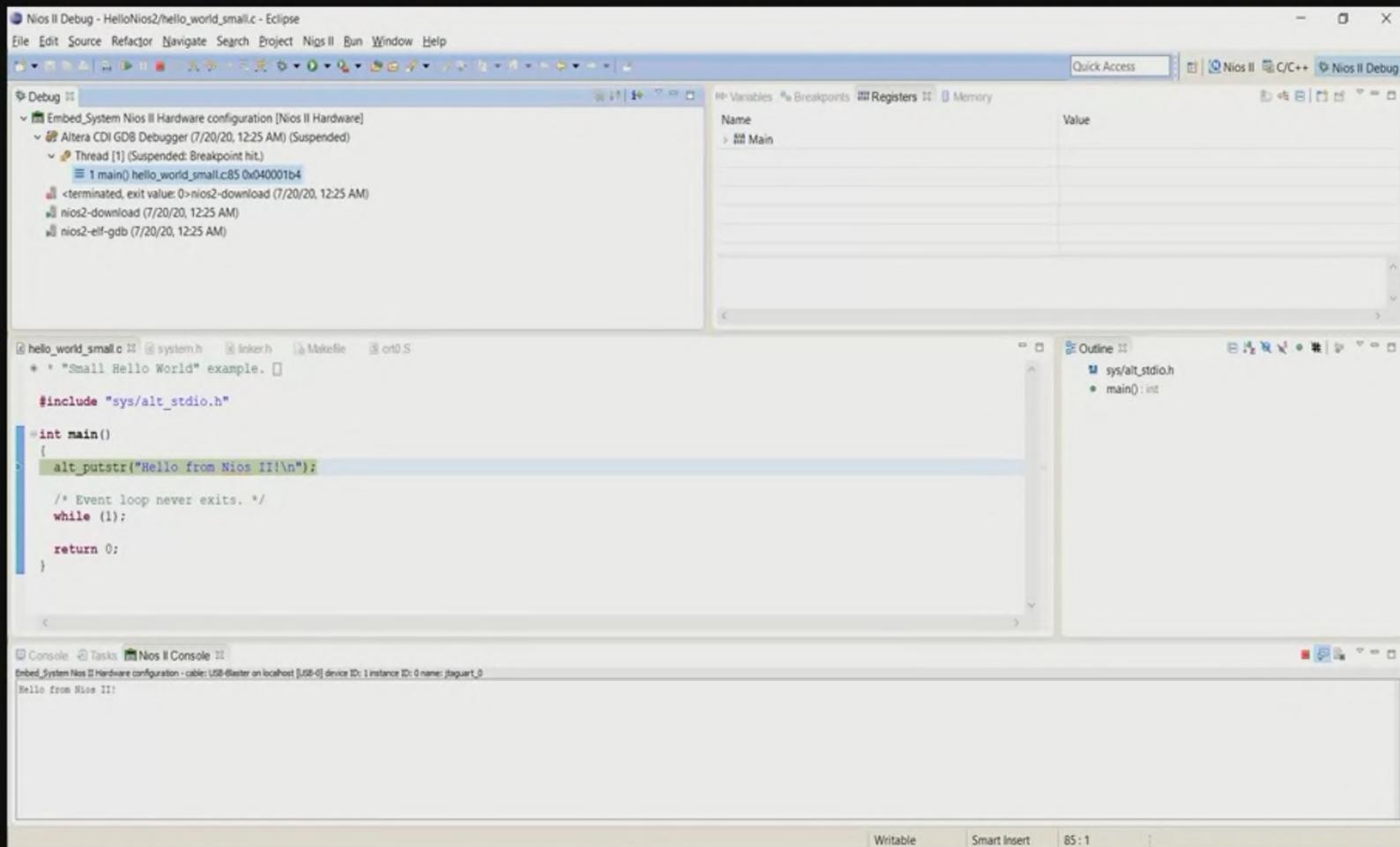
Programming the Nios II - software



Click on Refresh Connections & 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/>

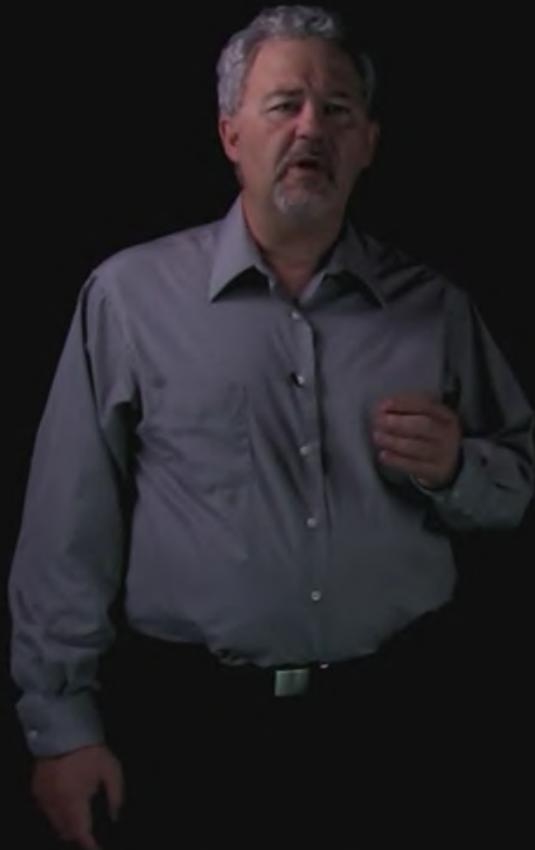


FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition



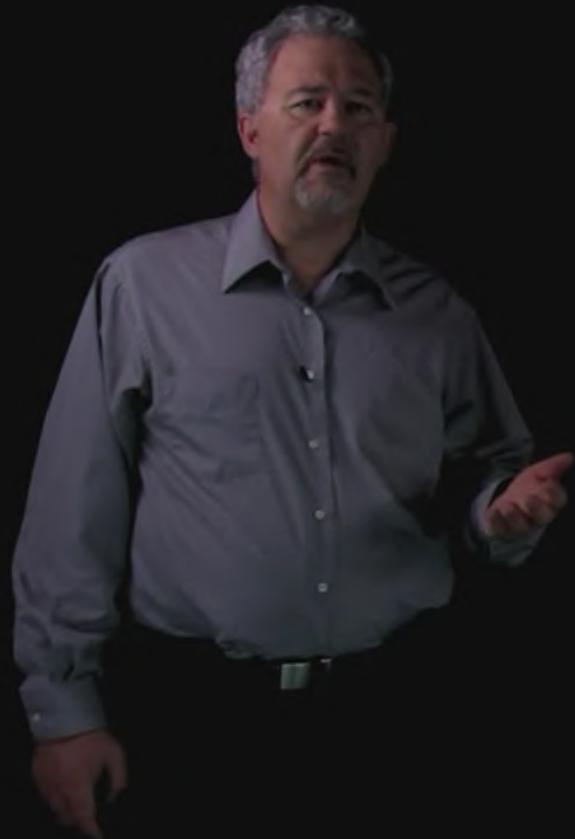
Memory in the Nios II



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.

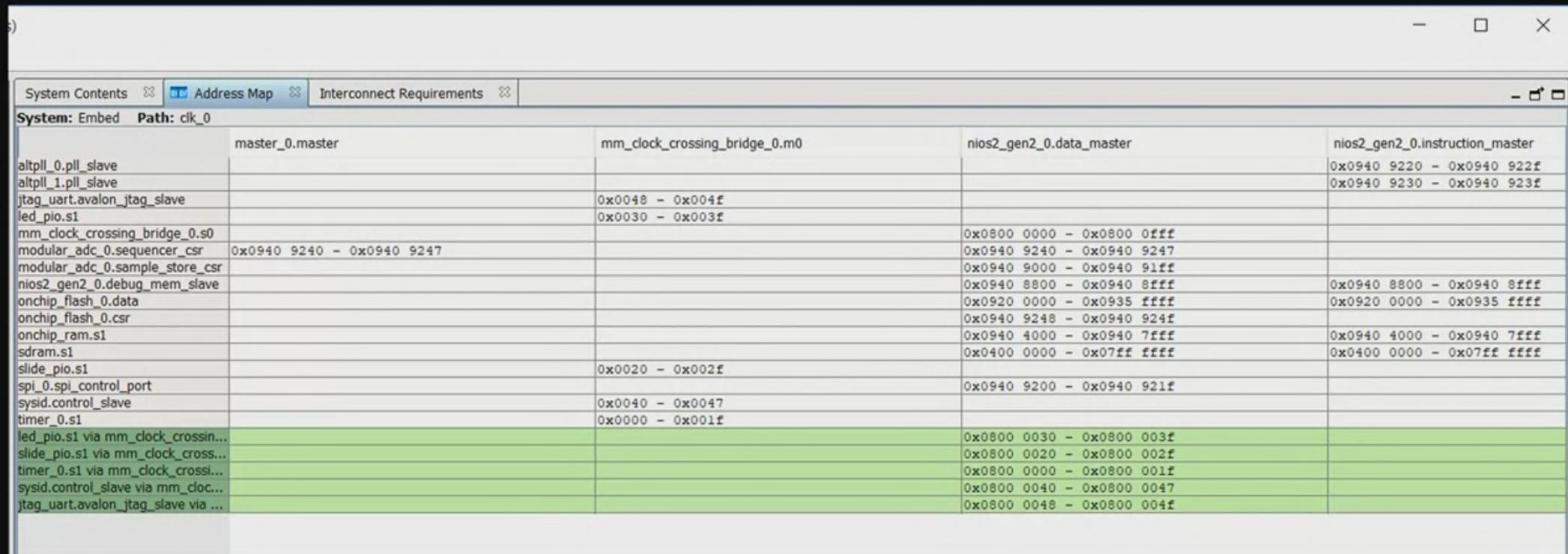


A Nios II core uses one or more of the following to provide memory and I/O access:

- **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



The screenshot shows the Qsys Address Map window for the clk_0 system. The window has tabs for System Contents, Address Map (which is selected), and Interconnect Requirements. The table displays memory mappings for various components:

Component	Master	Slave	Range	Master	Slave	Range
altpll_0 pll slave	master_0.master					
altpll_1 pll slave						
jtag_uart.avalon_jtag_slave			0x0048 - 0x004f			
led_pio.s1			0x0030 - 0x003f			
mm_clock_crossing_bridge_0.s0						
modular_adc_0_sequencer_csr	0x0940 9240 - 0x0940 9247					
modular_adc_0_sample_store_csr						
nios2_gen2_0_debug_mem_slave						
onchip_flash_0_data						
onchip_flash_0_csr						
onchip_ram.s1						
sdram.s1						
slide_pio.s1			0x0020 - 0x002f			
spi_0_spi_control_port						
sysid.control_slave			0x0040 - 0x0047			
timer_0.s1			0x0000 - 0x001f			
led_pio.s1 via mm_clock_crossin...						
slide_pio.s1 via mm_clock_cross...						
timer_0.s1 via mm_clock_crossi...						
sysid.control_slave via mm_cloc...						
jtag_uart.avalon_jtag_slave via ...						



BSP Editor allows Memory Changes

BSP Editor - C:\AlteraPrj\DE10LiteHello\software\HelloNios2_bsp\settings.bsp

File Edit Tools Help

Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory

Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	sdram	sdram
.entry	reset	onchip_ram
.exceptions	onchip_ram	onchip_ram
.heap	sdram	sdram
.rodata	sdram	sdram
.rwdta	sdram	sdram
.stack	sdram	sdram
.text	sdram	sdram

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_ram	0x09404020 - 0x09407FFF	onchip_ram	16352	32
reset	0x09404000 - 0x0940401F	onchip_ram	32	0
onchip_flash_0_data	0x09200000 - 0x0935FFFF	onchip_flash_0_data	1441792	0
sdram	0x04000000 - 0x07FFFFFF	sdram	67108864	0

Add...
Remove...
Restore Defaults...

Add...
Remove...
Restore Defaults...
Add Memory Device...
Remove Memory Device...
Memory Usage...
Memory Map...



BSP Memory Map

Memory Map

The screenshot shows a memory map interface with a table of device memory ranges and attributes, and a search view on the right.

Memory Map Table:

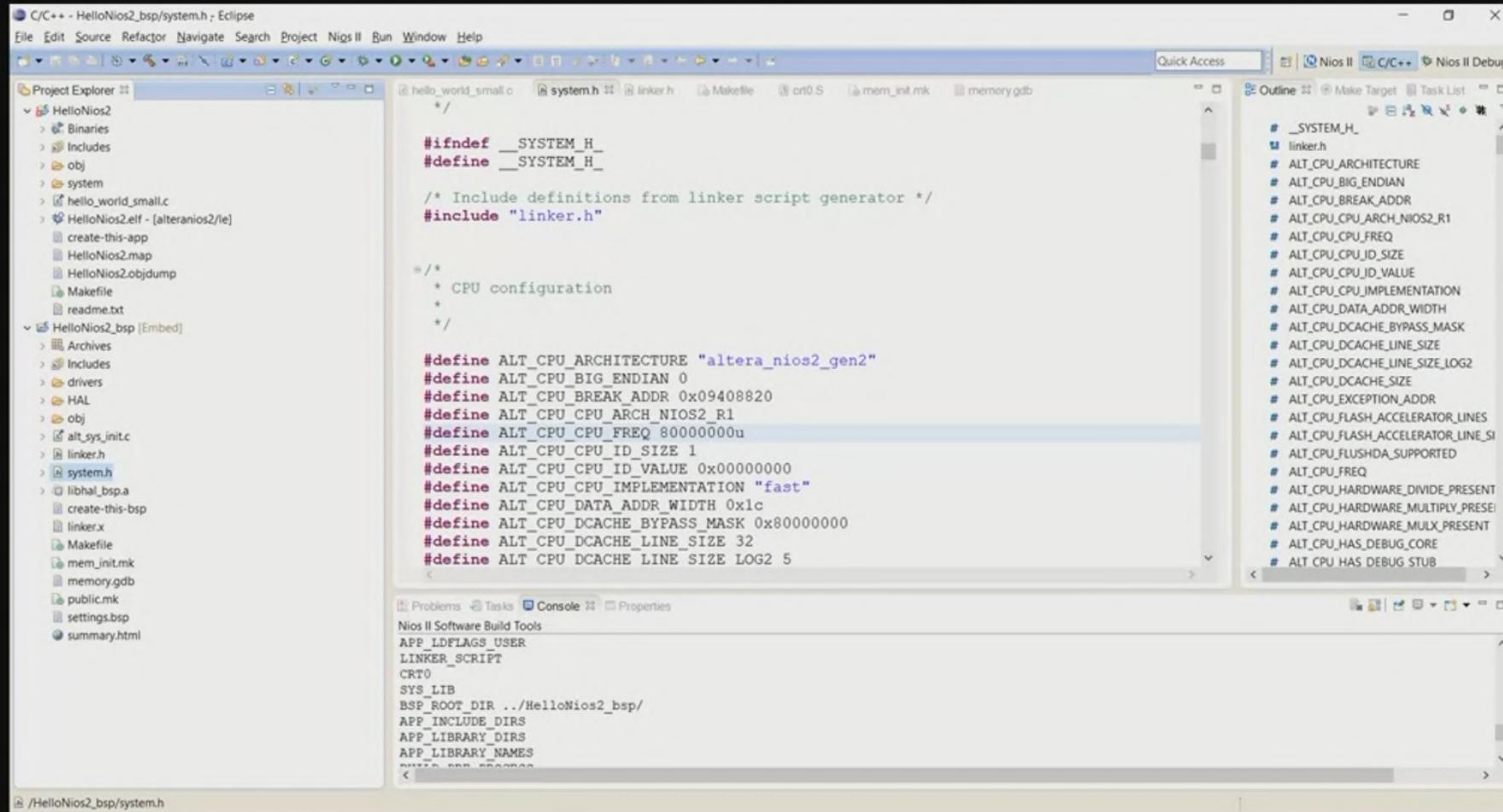
Device	Address Range	Size (b...)	Attributes
onchip_flash_0_csr	0x09409248 - 0x0...	8	
modular_adc_0_sequenc...	0x09409240 - 0x0...	8	
altpll_1	0x09409230 - 0x0...	16	
altpll_0	0x09409220 - 0x0...	16	
spi_0	0x09409200 - 0x0...	32	
modular_adc_0_sample_...	0x09409000 - 0x0...	512	
onchip_ram	0x09404000 - 0x0...	16384	memory
onchip_flash_0_data	0x09200000 - 0x0...	1441792	flash, memory, non-...
jtag_uart	0x08000048 - 0x0...	8	printable
sysid	0x08000040 - 0x0...	8	
led_pio	0x08000030 - 0x0...	16	
slide_pio	0x08000020 - 0x0...	16	
timer_0	0x08000000 - 0x0...	32	timer
sdram	0x04000000 - 0x0...	67108864	memory

View:

- onchip_flash_0_csr
- modular_adc_0_sequencer_csr
- altpll_1
- altpll_0
- spi_0
- modular_adc_0_sample_store_csr
- onchip_ram
- onchip_flash_0_data
- jtag_uart
- sysid
- led_pio
- slide_pio
- timer_0
- sdram



System.h and Memory

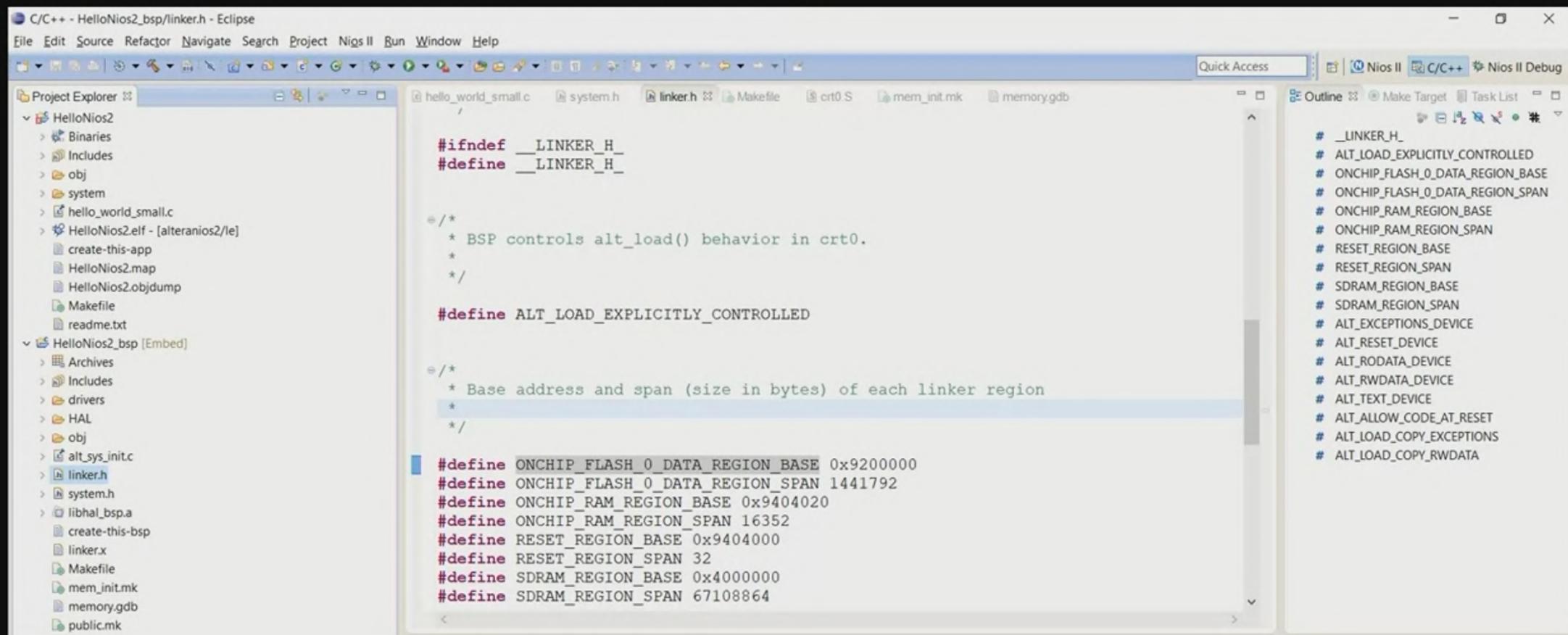


The screenshot shows the Eclipse C/C++ IDE interface with the following details:

- Project Explorer:** Shows the project structure for "HelloNios2_bsp". It includes a main "HelloNios2" folder containing Binaries, Includes, obj, system, and hello_world_small.c. Inside "HelloNios2_bsp", there are Archives, Includes, drivers, HAL, obj, alt_sys_init.c, linker.h, system.h, libhal_bsp.a, create-this-bsp, linker.x, Makefile, mem_init.mk, memory.gdb, public.mk, settings.bsp, and summary.html.
- Editor:** The main editor window displays the content of "system.h". The code includes defines for the Nios II architecture, such as ALT_CPU_ARCHITECTURE, ALT_CPU_BIG_ENDIAN, and ALT_CPU_BREAK_ADDR, along with memory-related defines like ALT_CPU_DCACHE_SIZE and ALT_CPU_DCACHE_LINE_SIZE.
- Outline View:** The right-hand margin shows the "Outline" view, which lists various symbols and defines, such as __SYSTEM_H_, linker.h, ALT_CPU_IMPLEMENTATION, and ALT_CPU_DCACHE_BYPASS_MASK.
- Properties View:** The bottom right shows the "Properties" view for the Nios II Software Build Tools, listing settings like APP_LDFLAGS_USER, LINKER_SCRIPT, CRT0, and SYS_LIB.
- Bottom Status Bar:** The status bar at the bottom shows the path "/HelloNios2_bsp/system.h".



Linker.h and Memory



The screenshot shows the Eclipse C/C++ IDE interface with the following details:

- Project Explorer:** Shows the project structure for "HelloNios2" and its sub-projects.
- Editor:** Displays the content of the "linker.h" file.
- Outline View:** Shows the list of macro definitions and region constants defined in the file.
- Toolbars and Menus:** Standard Eclipse toolbars and menus like File, Edit, Source, Refactor, Navigate, Search, Project, Nios II, Run, Window, Help.

```
#ifndef __LINKER_H_
#define __LINKER_H_

/*
 * BSP controls alt_load() behavior in crt0.
 *
 */

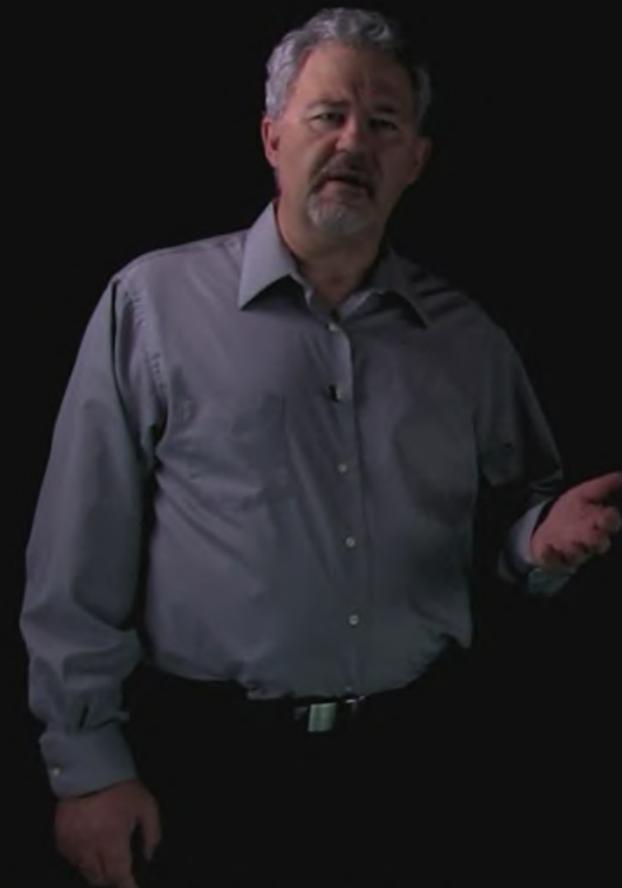
#define ALT_LOAD_EXPLICITLY_CONTROLLED

/*
 * Base address and span (size in bytes) of each linker region
 *
 */

#define ONCHIP_FLASH_0_DATA_REGION_BASE 0x9200000
#define ONCHIP_FLASH_0_DATA_REGION_SPAN 1441792
#define ONCHIP_RAM_REGION_BASE 0x9404020
#define ONCHIP_RAM_REGION_SPAN 16352
#define RESET_REGION_BASE 0x9404000
#define RESET_REGION_SPAN 32
#define SDRAM_REGION_BASE 0x4000000
#define SDRAM_REGION_SPAN 67108864
```



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

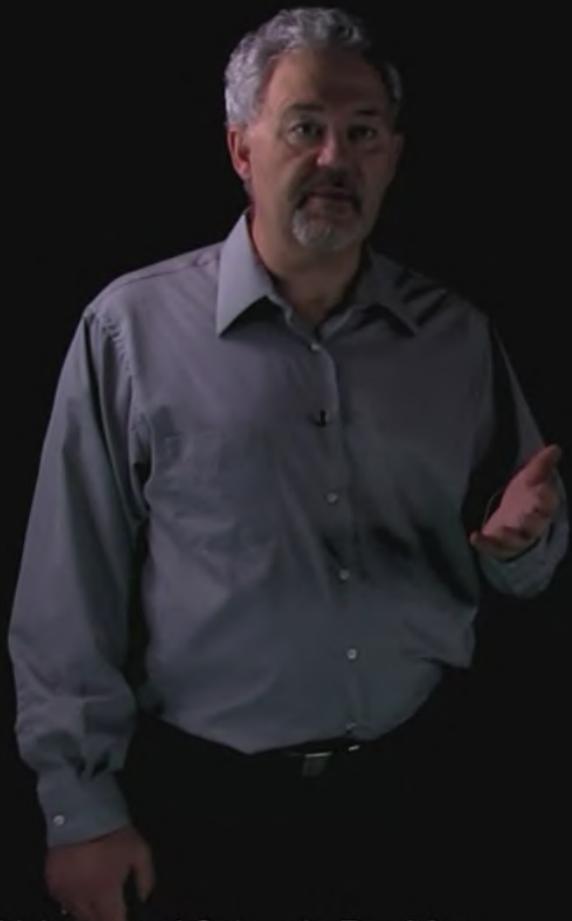
- [1] Intel/Altera Staff. (2020/Jul/11), *Nios II Gen2 Software Developer's Handbook*. [Online]. Available: <https://www.intel.com/content/www/us/en/programmable/products/processors/support.html>
- [2] Intel/Altera Staff. (2020/Feb/17), *Nios II Gen2 Hardware Development Tutorial*. [Online]. Available: <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

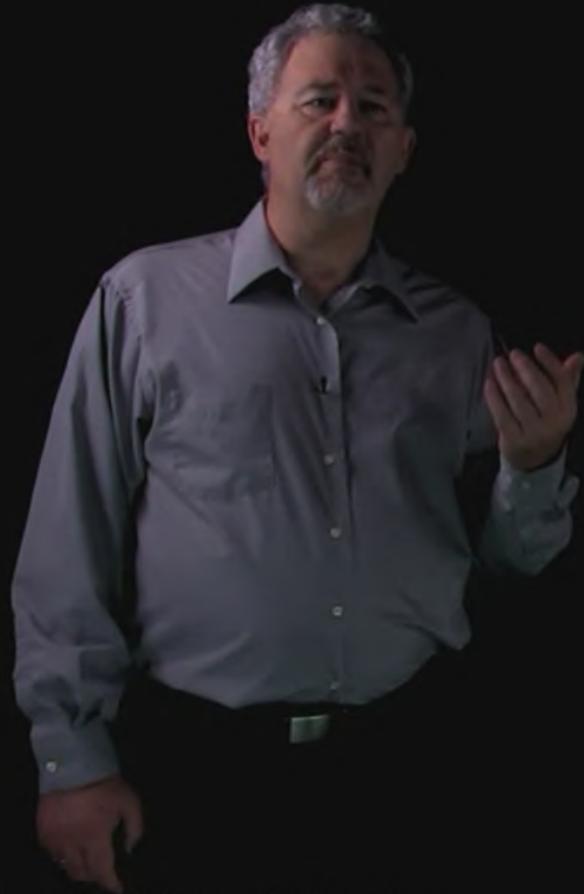
Accessing Custom Instructions



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

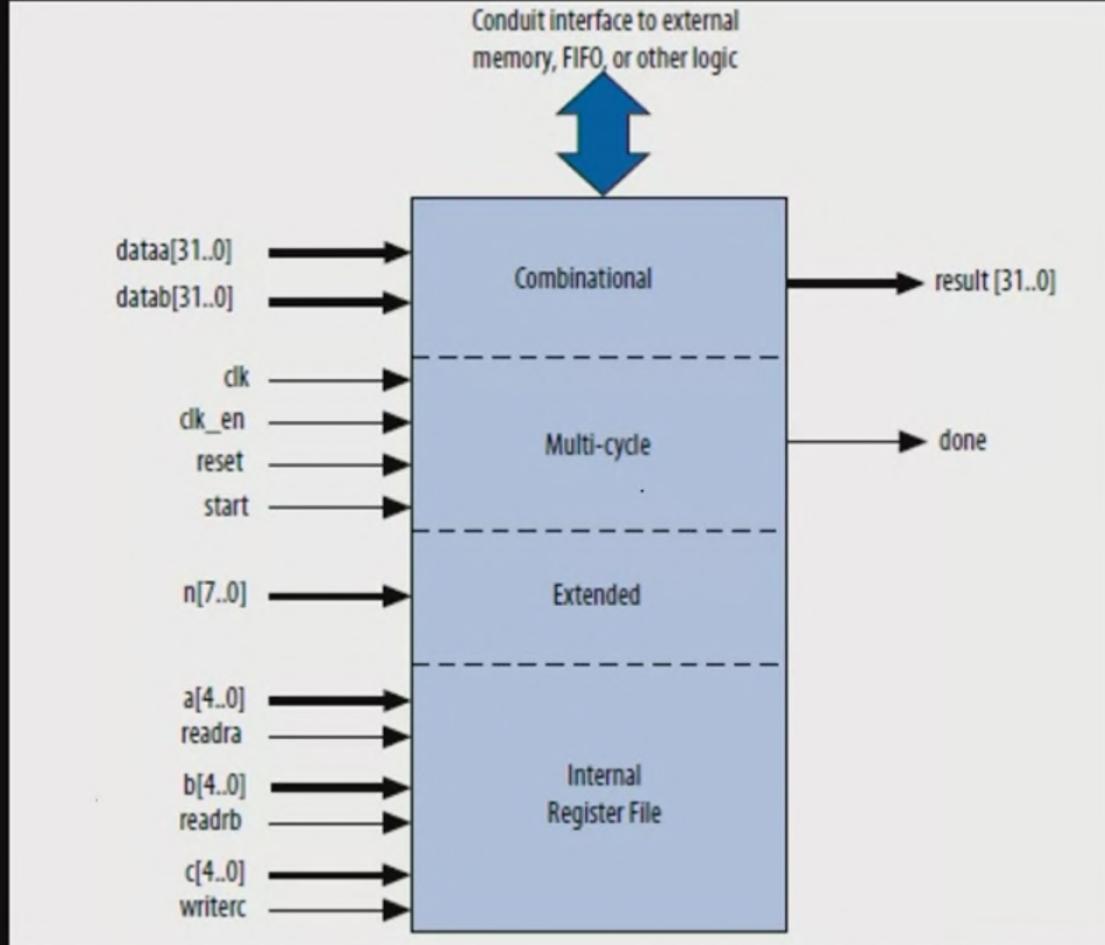
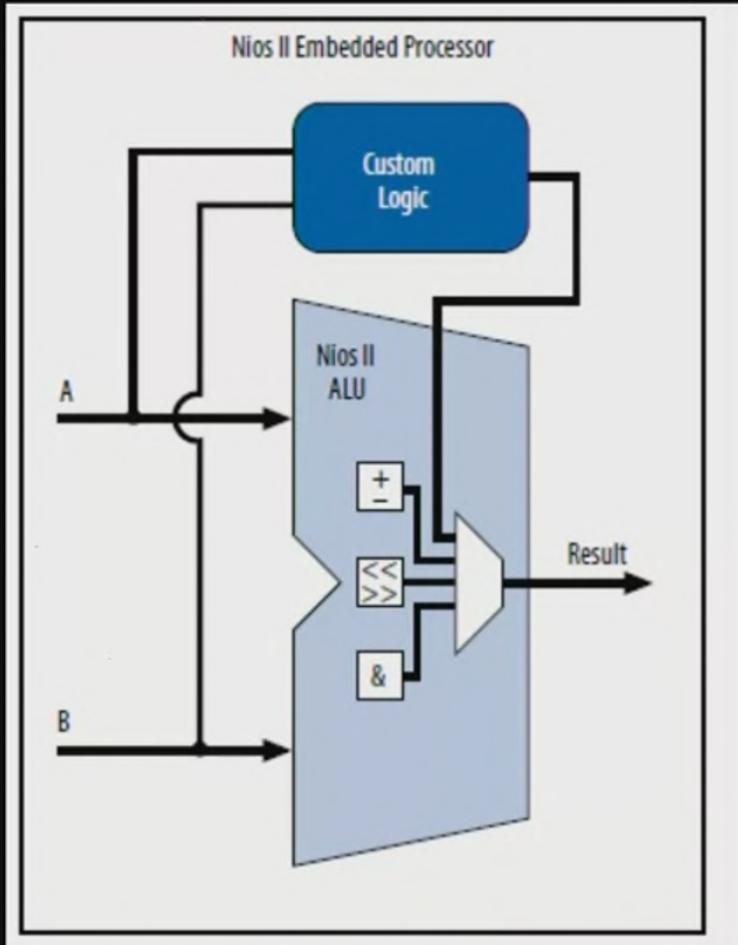
Nios Custom Instructions



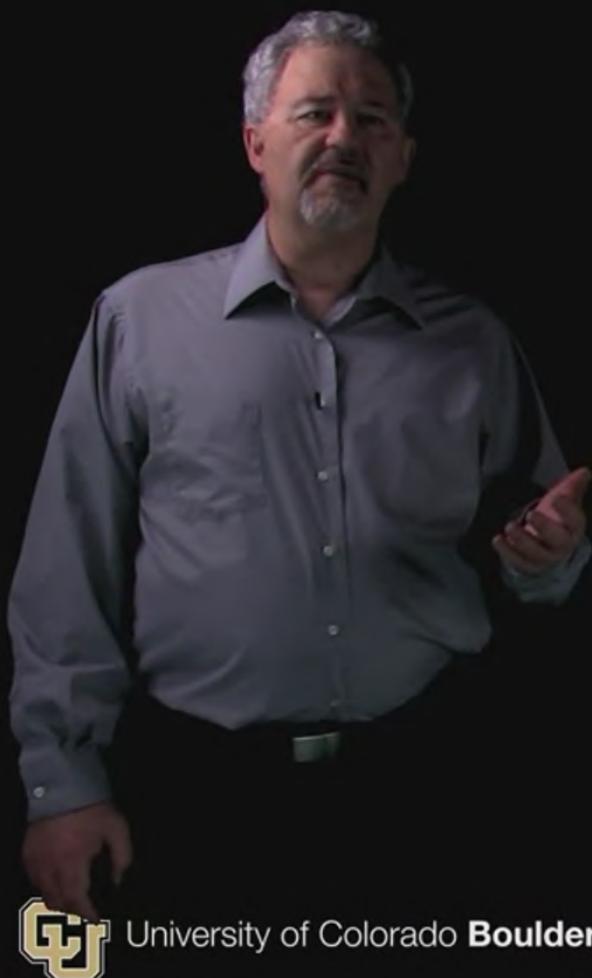
Custom instructions give you the ability to tailor the Nios II processor to meet the needs of a particular application. You can **accelerate** time critical software algorithms, as much as **10 to 100 times**, by converting them to custom hardware logic blocks.

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



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



Custom Instruction Use

Definition:

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

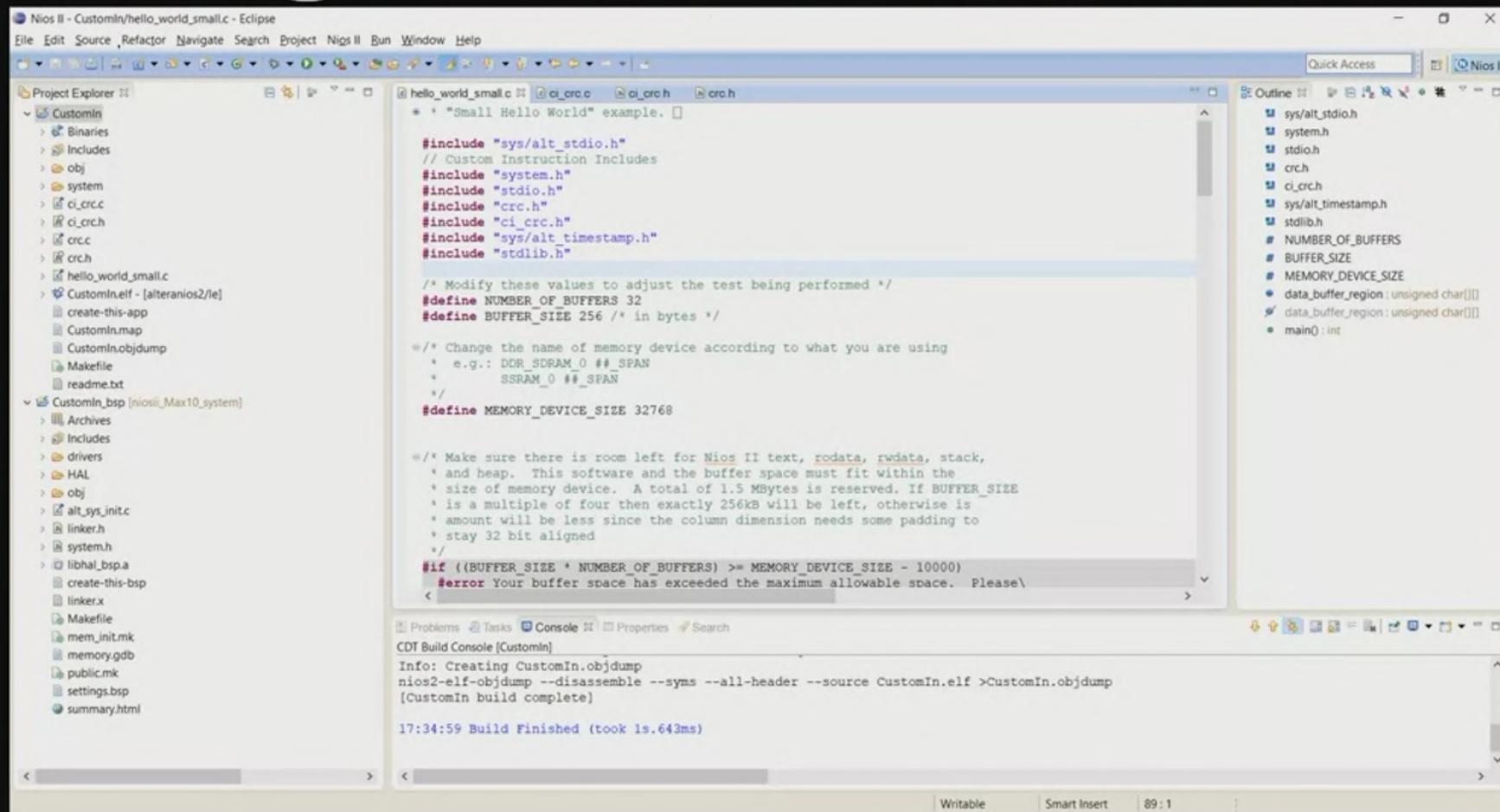
Use:

```
/* The custom instruction CRC will initialize to the
initial remainder value */
```

```
CRC_CI_MACRO(0,0);
```



Building the Custom Instruction SW



Nios II - CustomIn/hello_world_small.c - Eclipse

File Edit Source Refactor Navigate Search Project Nios II Run Window Help

Project Explorer

- CustomIn
- Binaries
- Includes
- obj
- system
- ci_crc.c
- ci_crc.h
- crc.c
- crc.h
- hello_world_small.c
- CustomIn.elf - [alteranios2/le]
- create-this-app
- CustomIn.map
- CustomIn.objdump
- Makefile
- readme.txt

hello_world_small.c

```
#include "sys/alt_stdio.h"
// Custom Instruction Includes
#include "system.h"
#include "stdio.h"
#include "crc.h"
#include "ci_crc.h"
#include "sys/alt_timestamp.h"
#include "stdlib.h"

/* Modify these values to adjust the test being performed */
#define NUMBER_OF_BUFFERS 32
#define BUFFER_SIZE 256 /* in bytes */

/* Change the name of memory device according to what you are using
 * e.g.: DDR_SDRAM_0 ##_SPAN
 *        SSRAM_0 ##_SPAN
 */
#define MEMORY_DEVICE_SIZE 32768

/* Make sure there is room left for Nios II text, rodata, rdata, stack,
 * and heap. This software and the buffer space must fit within the
 * size of memory device. A total of 1.5 MBytes is reserved. If BUFFER_SIZE
 * is a multiple of four then exactly 256kB will be left, otherwise the
 * amount will be less since the column dimension needs some padding to
 * stay 32 bit aligned
 */
#ifndef ((BUFFER_SIZE * NUMBER_OF_BUFFERS) >= MEMORY_DEVICE_SIZE - 10000)
    #error Your buffer space has exceeded the maximum allowable space. Please\
```

Outline

- sys/alt_stdio.h
- system.h
- stdio.h
- crc.h
- ci_crc.h
- sys/alt_timestamp.h
- stdlib.h
- NUMBER_OF_BUFFERS
- BUFFER_SIZE
- MEMORY_DEVICE_SIZE
- data_buffer_region : unsigned char[]
- data_buffer_region : unsigned char[]
- main() : int

Problems Tasks Console Properties Search

CDT Build Console [CustomIn]

```
Info: Creating CustomIn.objdump
nios2-elf-objdump --disassemble --syms --all-header --source CustomIn.elf >CustomIn.objdump
[CustomIn build complete]
```

17:34:59 Build Finished (took 1s.643ms)

Writable Smart Insert 89:1



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

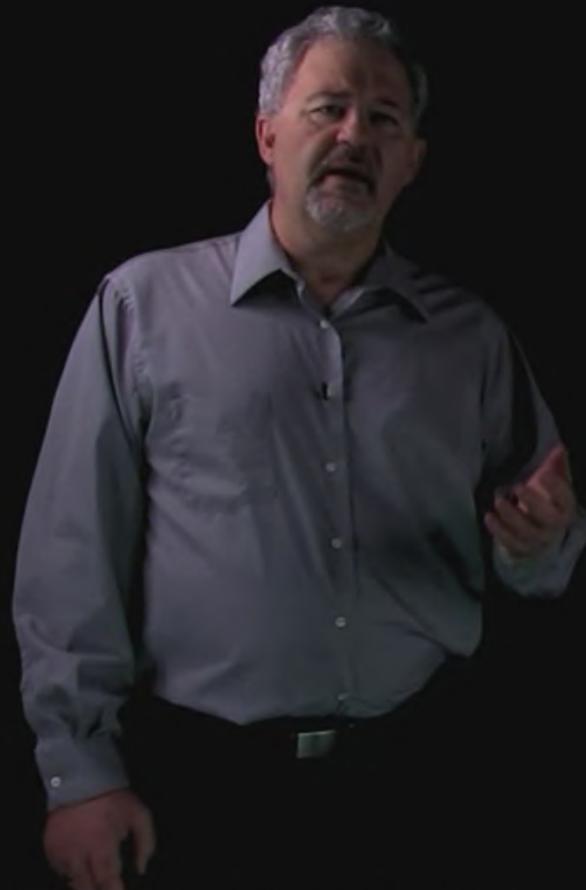
Custom instruction CRC vs software CRC = 68

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>

