

Section 2

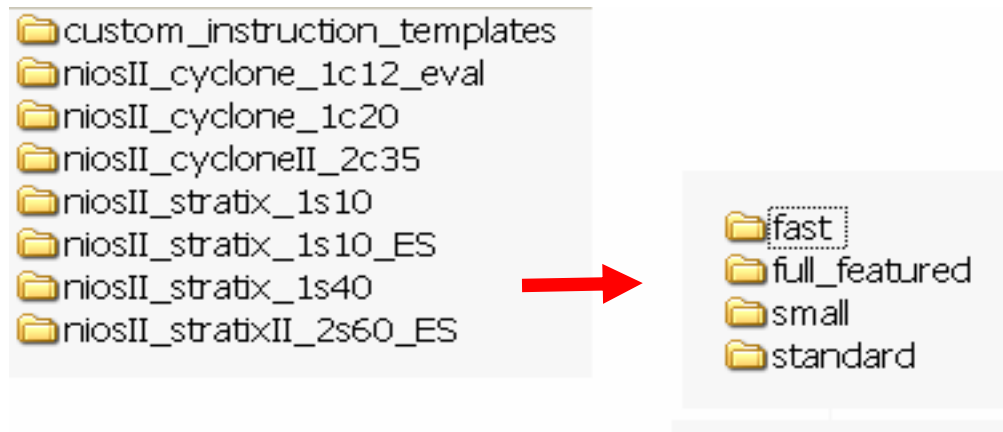
Hardware Design with SOPC Builder

Section 2 - Topics

- Create a Quartus Project
- Create a NiosII system in SOPC Builder
- Create a NiosII software project
- Run as Instruction Set Simulator (ISS)
- RTL Simulation

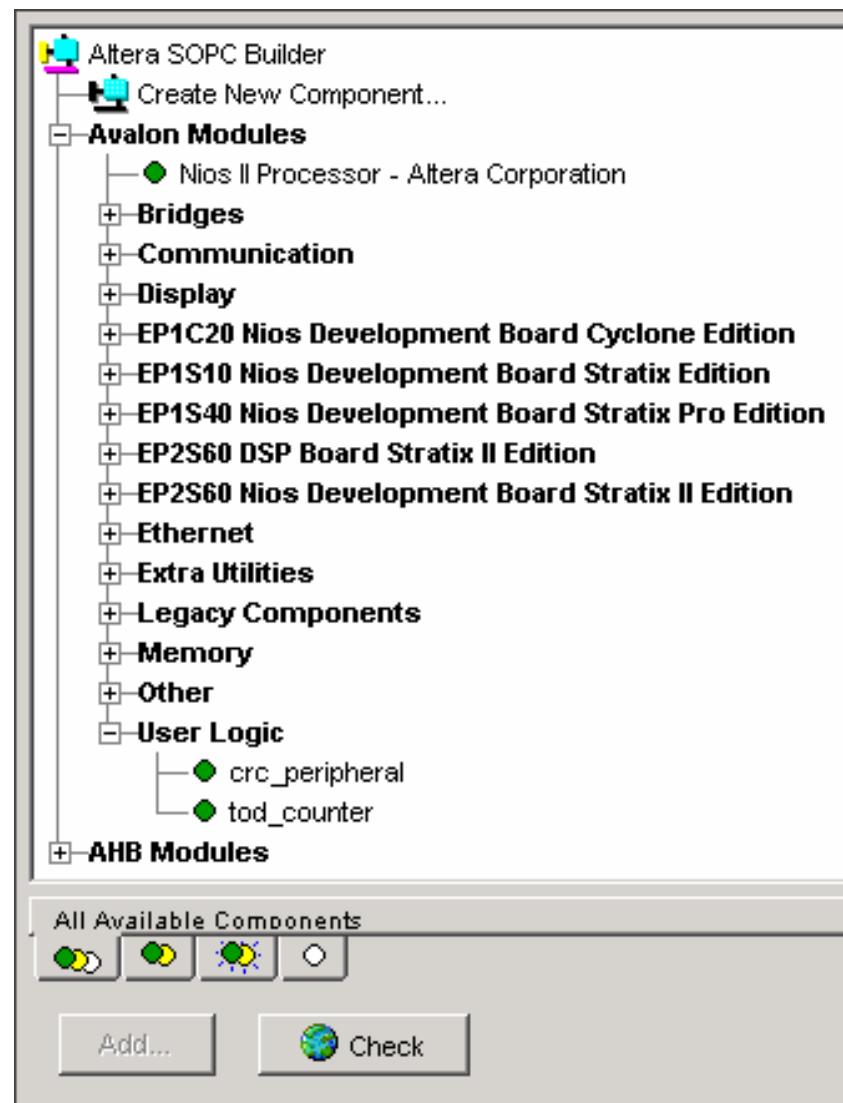
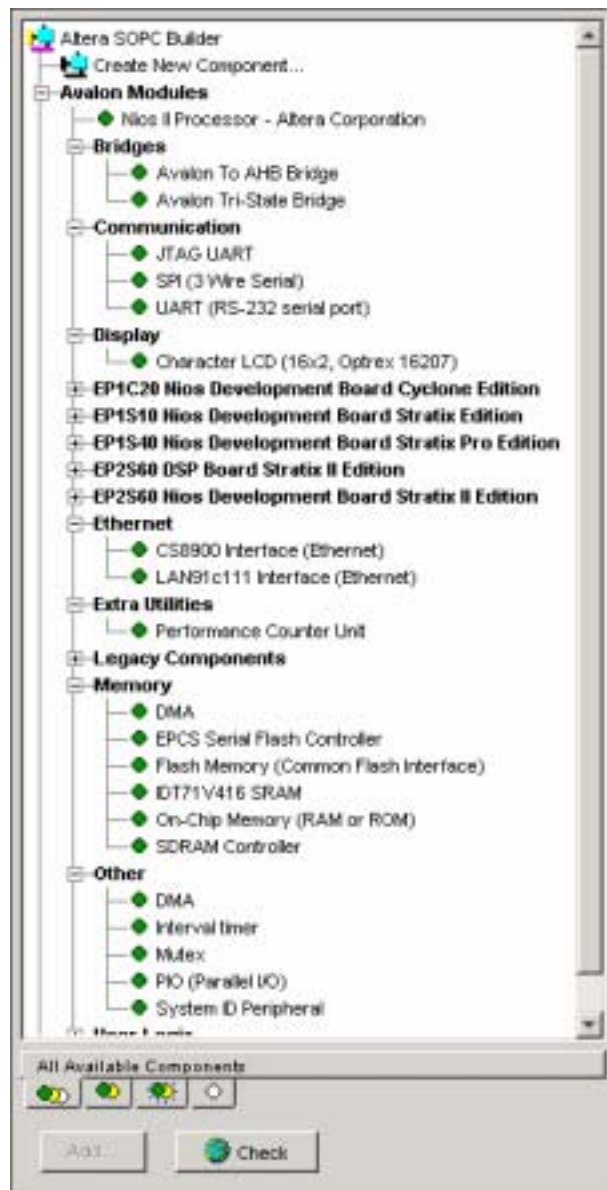
Reference Designs For Dev Kits

- Several reference designs are available
- See **altera\kits\nios2\examples\verilog** or **altera\kits\nios2\examples\vhdl**



- Can be used as-is in final hardware platform
- Otherwise, customize system to specific needs

Standard and Custom Peripherals



SOPC Builder Development Tool

Altera SOPC Builder - std_1c20

File Module System View Tools Help

System Contents Nios II More "cpu" Settings System Generation

Altera SOPC Builder

- Create New Component...
- Avalon Modules
 - Nios II Processor - Altera Corporation
 - Bridges
 - Avalon To AHB Bridge
 - Avalon Tri-State Bridge
 - Communication
 - JTAG UART
 - SPI (3 Wire Serial)
 - UART (RS-232 serial port)
 - Display
 - EP1C20 Nios Development Board Cycle
 - EP1K10 Nios Development Board Strati
 - EP1K10 Nios Development Board Strati
 - EP2K10 DSP Board Stratix II Edition
 - EP2K10 Nios Development Board Strati
 - Ethernet
 - Extra Utilities
 - Legacy Components
 - Memory
 - Other
 - DMA
 - Interval timer
 - Muxes

Board: Target: Nios Development Board, Cyclone (EP1C20) Target Device Family: Cyclone

Clock (MHz): clk: 50.0

| Use | Module Name | Description | Clock | Base | End | IRQ |
|-------------------------------------|--|--|-------|------------|------------|-----|
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> cpu | Nios II Processor - Altera Corporation | clk | | | |
| | instruction_master | Master port | | | | |
| | data_master | Master port | | | | |
| | jtag_debug_module | Slave port | | | | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> ext_ram_bus | Avalon Tri-State Bridge | clk | | | |
| | avalon_slave | Slave port | | | | |
| | tristate_master | Master port | | | | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> ext_flash | Flash Memory (Common Flash Interface) | | 0x00000 | 0x007FFFFF | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> ext_ram | IDT71V416 SRAM | | 0x02000 | 0x020FFFFF | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> epcs_controller | EPCS Serial Flash Controller | clk | 0x0210000 | 0x021007FF | 5 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> lan91c111 | LAN91c111 Interface (Ethernet) | | 0x0210000 | 0x0211FFFF | 6 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> sys_clk_timer | Interval timer | clk | 0x0210000 | 0x0212031F | 0 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> jtag_uart | JTAG UART | clk | 0x0212080 | 0x021208B7 | 1 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> button_pio | PIO (Parallel IO) | clk | 0x0212080 | 0x021208BF | 2 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> led_pio | PIO (Parallel IO) | clk | 0x02120870 | 0x0212087F | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> lcd_display | Character LCD (16x2, Optrex 16207) | clk | 0x02120880 | 0x021208BF | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> high_res_timer | Interval timer | clk | 0x02120820 | 0x0212083F | 3 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> seven_seg_pio | PIO (Parallel IO) | clk | 0x02120890 | 0x0212089F | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> reconfig_request_pio | PIO (Parallel IO) | clk | 0x021208A0 | 0x021208AF | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> uart1 | UART (RS-232 serial port) | clk | 0x02120840 | 0x0212085F | 4 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> sysid | System ID Peripheral | clk | 0x02120888 | 0x021208BF | |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> sdram | SDRAM Controller | clk | 0x01000 | 0x01FFFFFF | |

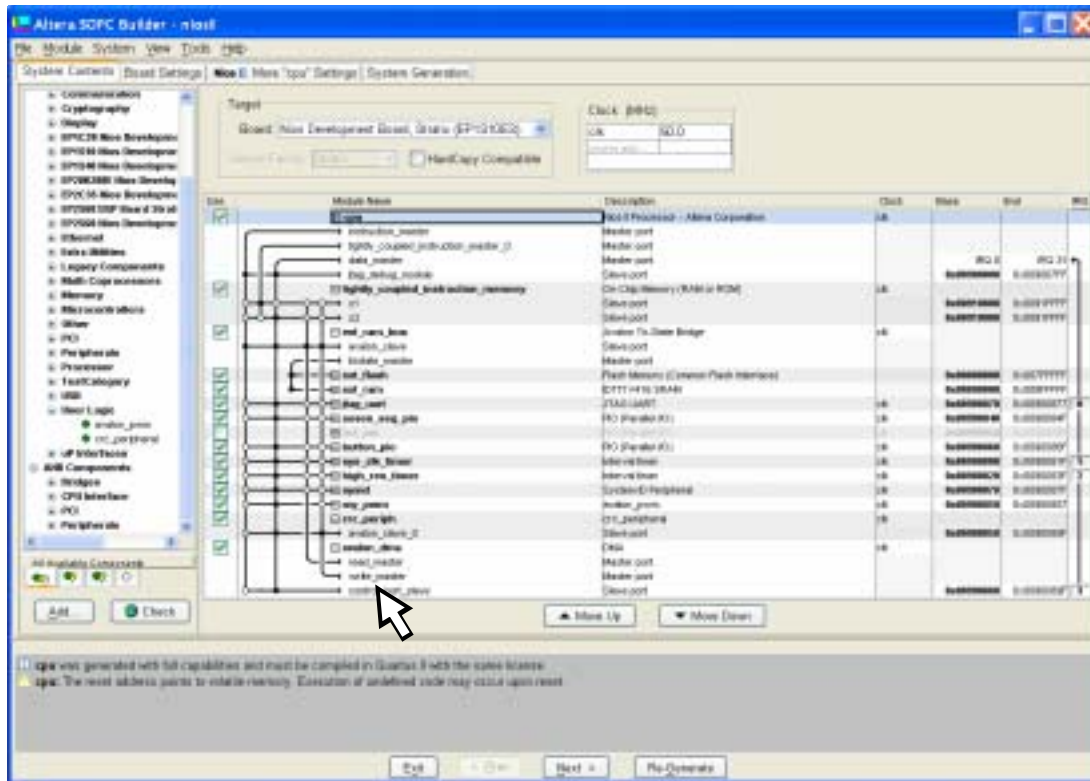
Move Up Move Down

Problems checking for web-updates (Unable to download component catalog, try again later.)

Exit < Prev Next > Generate

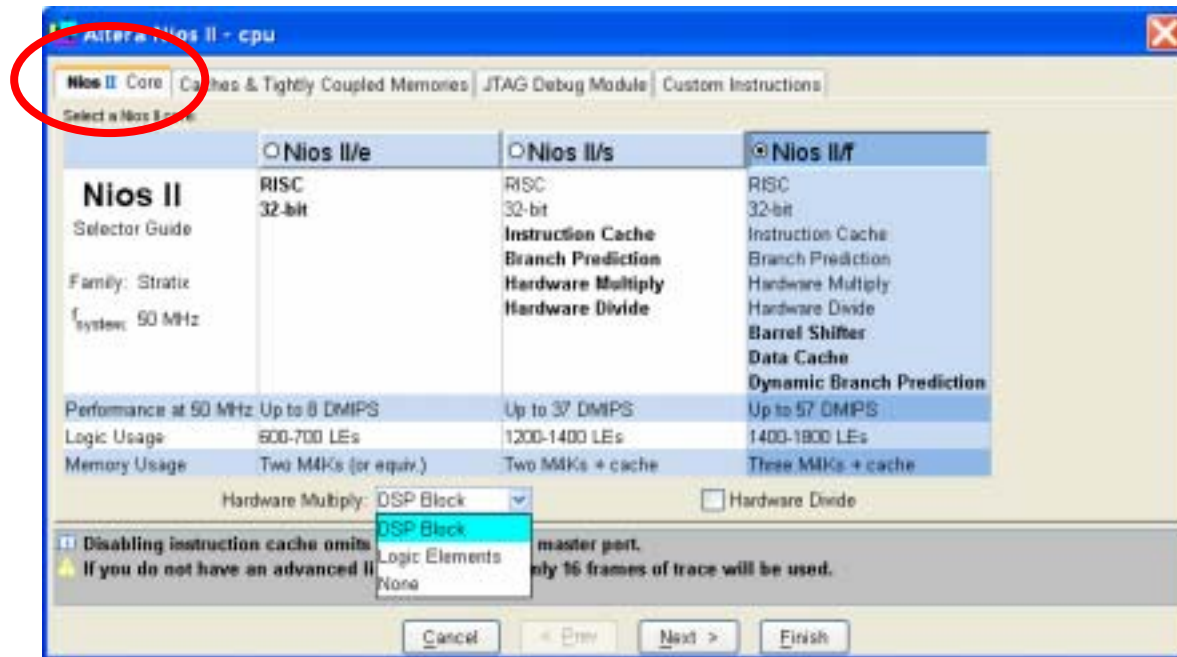
Avalon Master-Slave Connections

- **View => Show Master Connections**
- Observe and configure Avalon connections



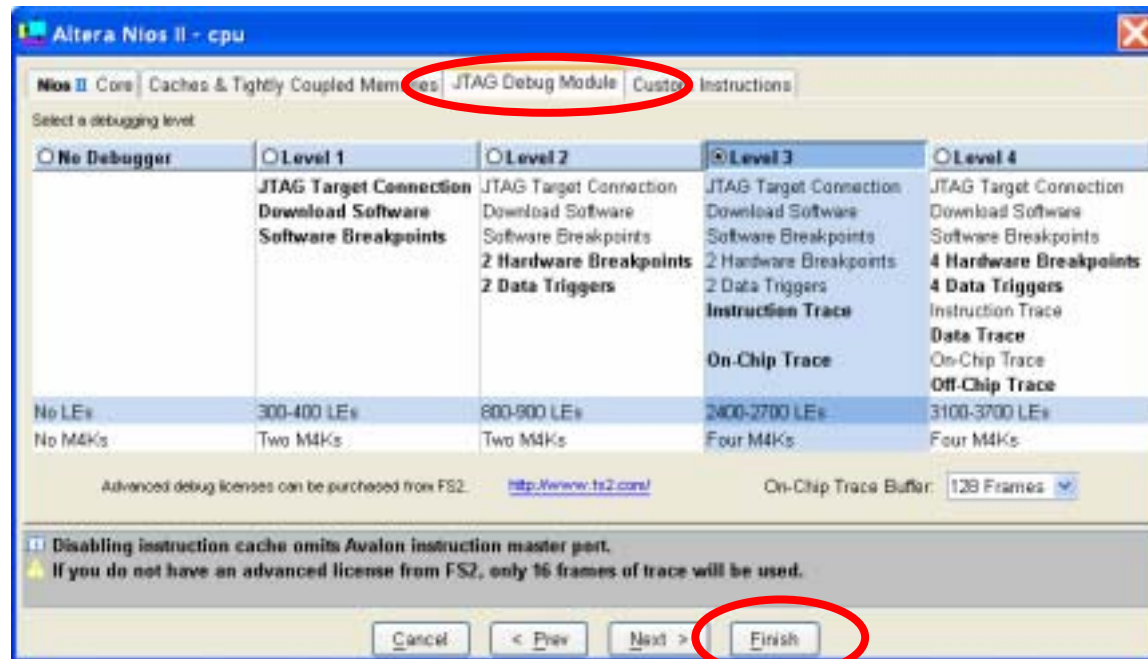
Configure the Nios II CPU

- Hardware designer selects Nios II version
 - economy, standard, or fast



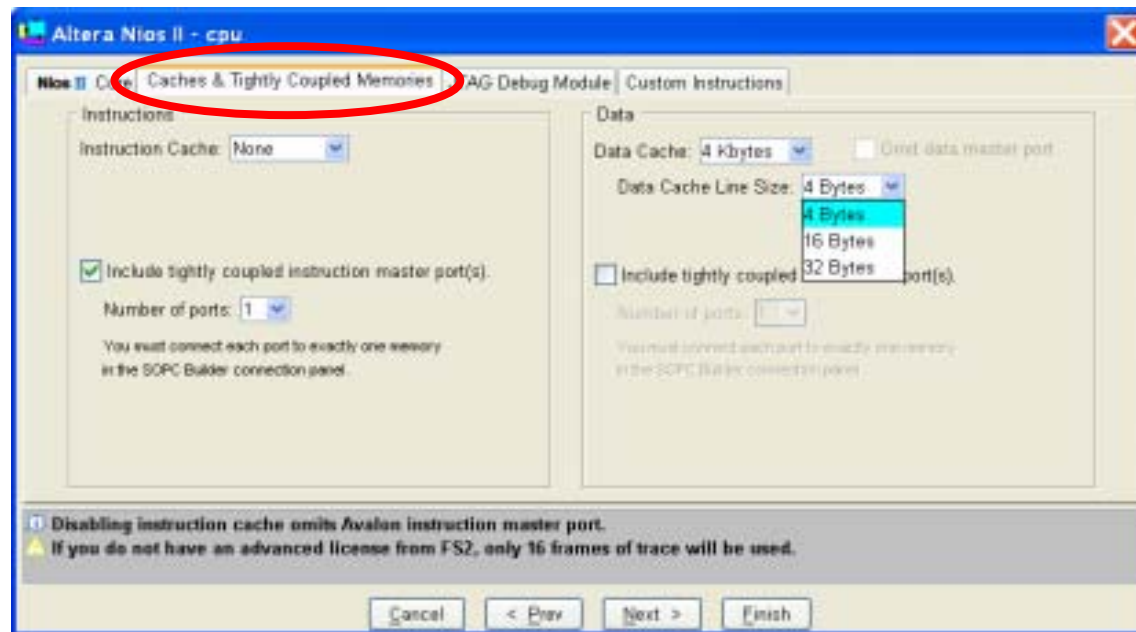
Choose JTAG Debug Core

- Select appropriate JTAG Debug level when configuring Nios II processor core



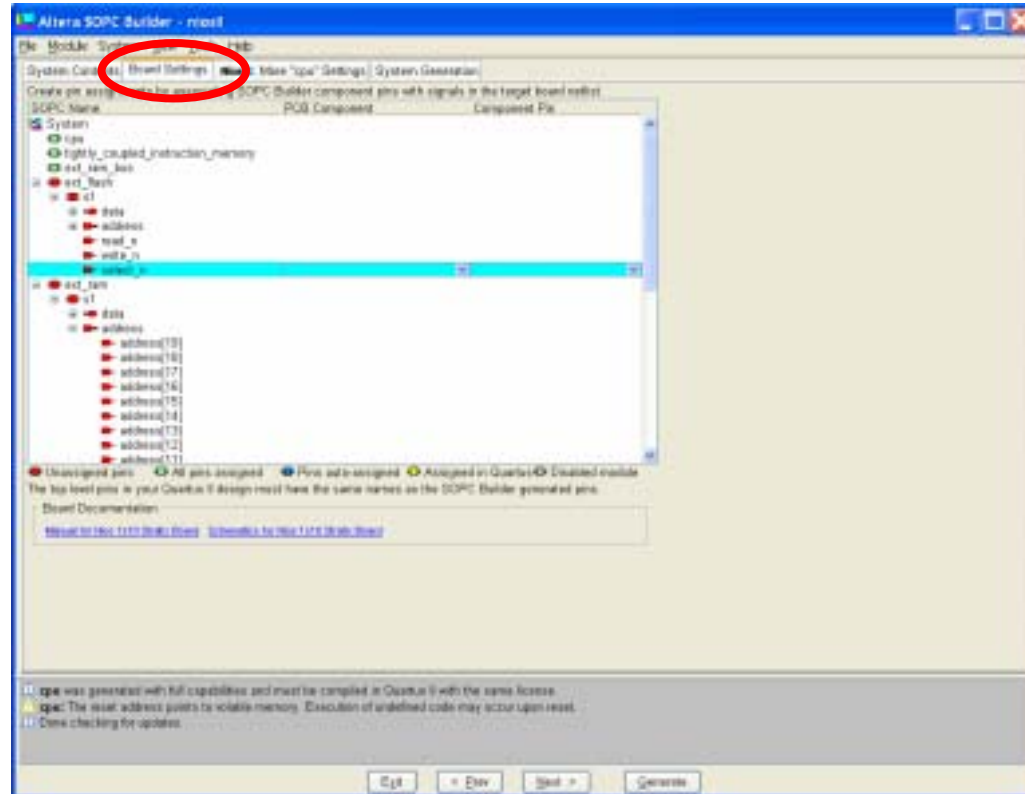
Select Cache and TCM Settings

- Adjust Size of Instruction and Data Cache Memory
 - Can now completely disable data cache on fast core
 - And also disable instruction cache as long as TCM used
- Enable Instruction / Data **Tightly Coupled Memory** masters
- Control Data Cache Width
 - Up to 32 byte cache line width now possible for better burst support



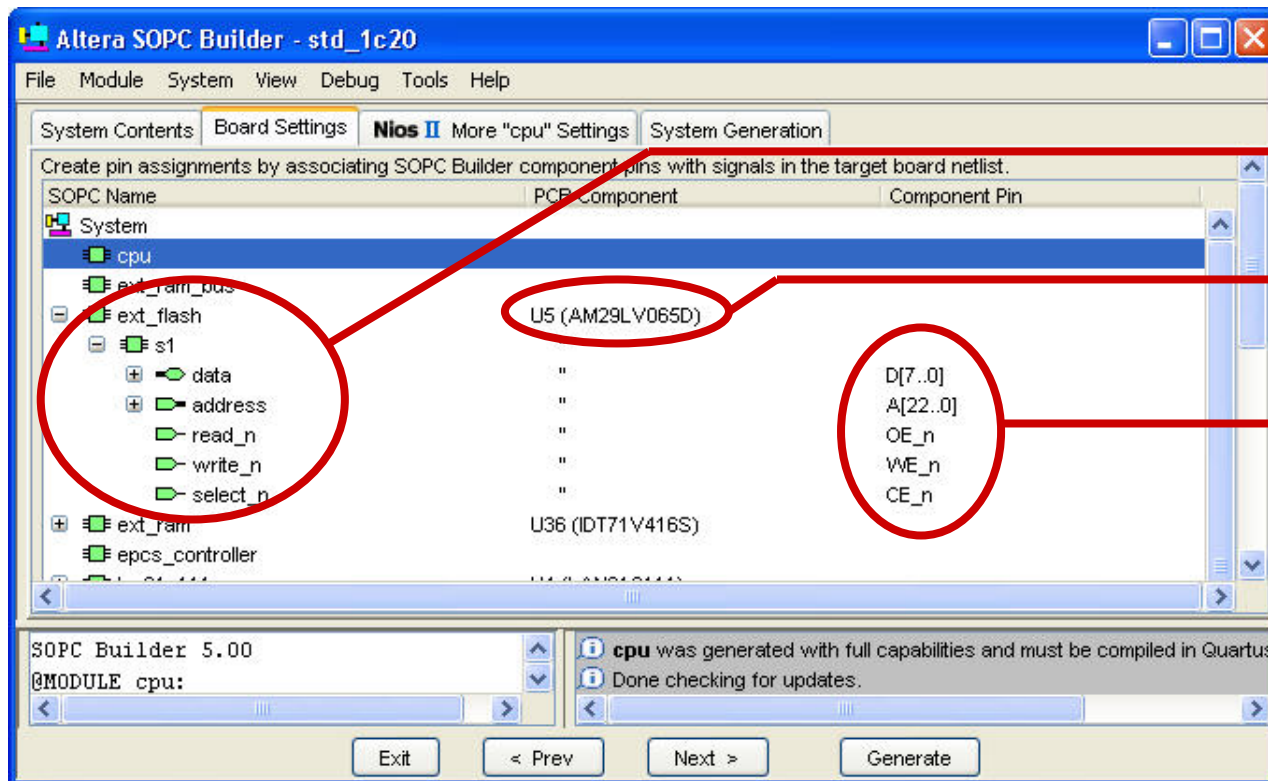
SOPC Builder – Board Settings

- Allows you to connect SOPC Builder components directly to board components using **Pin Mapper** utility



Using Pin Mapper

- Pin Editor contains three columns
 - SOPC Name, PCB Component, Component Pin



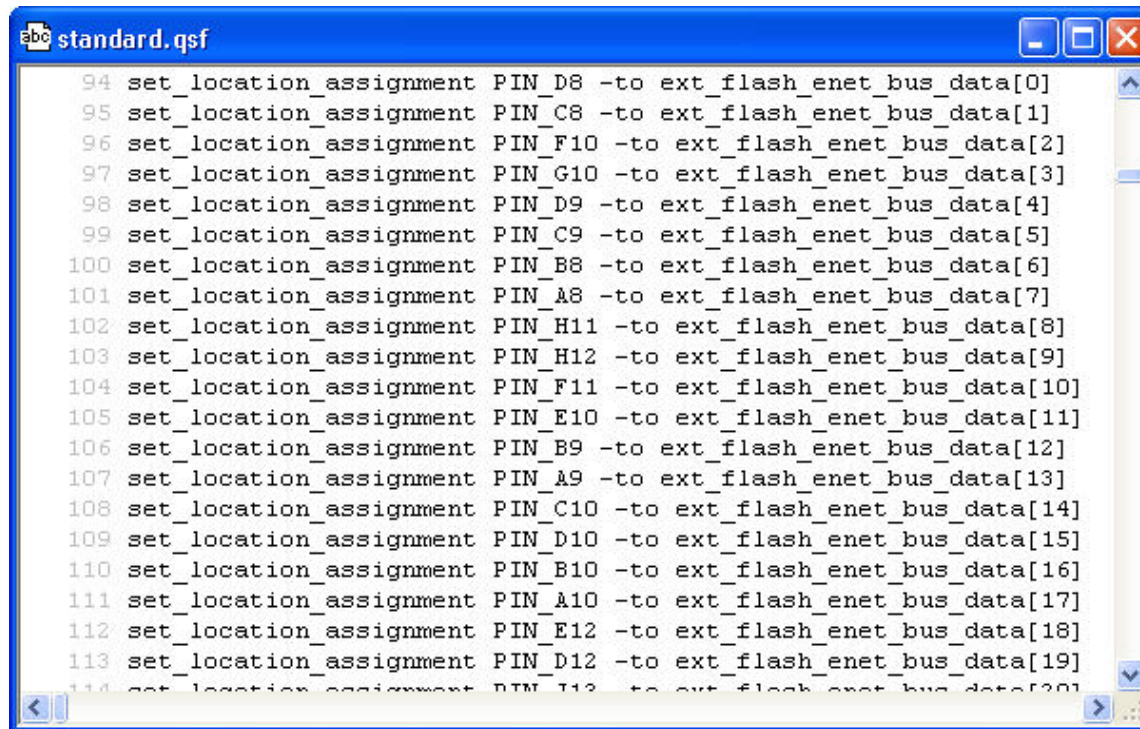
SOPC Builder component name and port names

PCB Component Name and Ref-Des

PCB pin names

Assignments get made in QSF

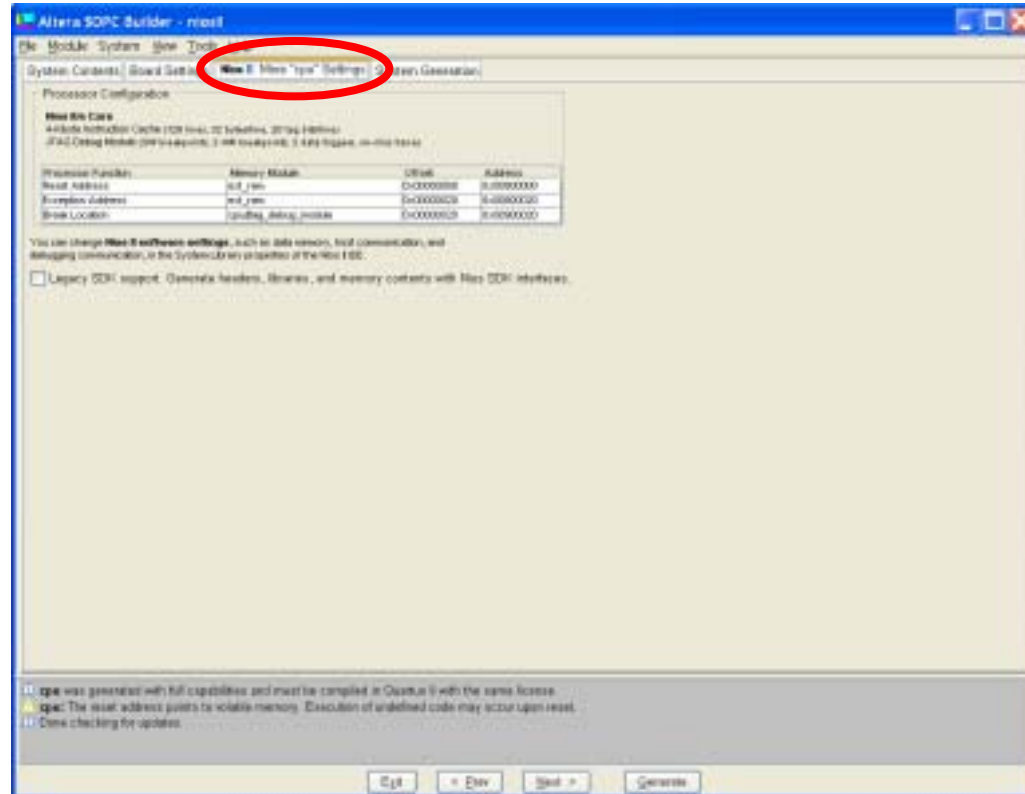
- During generation, all corresponding Quartus II pin assignments get made in the .QSF file.

A screenshot of a text editor window titled 'standard.qsf'. The window contains a list of pin assignments, each preceded by a line number from 94 to 114. The assignments are: 94 set_location_assignment PIN_D8 -to ext_flash_enet_bus_data[0], 95 set_location_assignment PIN_C8 -to ext_flash_enet_bus_data[1], 96 set_location_assignment PIN_F10 -to ext_flash_enet_bus_data[2], 97 set_location_assignment PIN_G10 -to ext_flash_enet_bus_data[3], 98 set_location_assignment PIN_C9 -to ext_flash_enet_bus_data[4], 99 set_location_assignment PIN_D9 -to ext_flash_enet_bus_data[5], 100 set_location_assignment PIN_B8 -to ext_flash_enet_bus_data[6], 101 set_location_assignment PIN_A8 -to ext_flash_enet_bus_data[7], 102 set_location_assignment PIN_H11 -to ext_flash_enet_bus_data[8], 103 set_location_assignment PIN_H12 -to ext_flash_enet_bus_data[9], 104 set_location_assignment PIN_F11 -to ext_flash_enet_bus_data[10], 105 set_location_assignment PIN_E10 -to ext_flash_enet_bus_data[11], 106 set_location_assignment PIN_B9 -to ext_flash_enet_bus_data[12], 107 set_location_assignment PIN_A9 -to ext_flash_enet_bus_data[13], 108 set_location_assignment PIN_C10 -to ext_flash_enet_bus_data[14], 109 set_location_assignment PIN_D10 -to ext_flash_enet_bus_data[15], 110 set_location_assignment PIN_B10 -to ext_flash_enet_bus_data[16], 111 set_location_assignment PIN_A10 -to ext_flash_enet_bus_data[17], 112 set_location_assignment PIN_E12 -to ext_flash_enet_bus_data[18], 113 set_location_assignment PIN_D12 -to ext_flash_enet_bus_data[19], and 114 set_location_assignment PIN_I12 -to ext_flash_enet_bus_data[20]. The window has a blue title bar and standard Windows window controls (minimize, maximize, close) on the right. A vertical scrollbar is on the right side of the text area.

```
94 set_location_assignment PIN_D8 -to ext_flash_enet_bus_data[0]
95 set_location_assignment PIN_C8 -to ext_flash_enet_bus_data[1]
96 set_location_assignment PIN_F10 -to ext_flash_enet_bus_data[2]
97 set_location_assignment PIN_G10 -to ext_flash_enet_bus_data[3]
98 set_location_assignment PIN_C9 -to ext_flash_enet_bus_data[4]
99 set_location_assignment PIN_D9 -to ext_flash_enet_bus_data[5]
100 set_location_assignment PIN_B8 -to ext_flash_enet_bus_data[6]
101 set_location_assignment PIN_A8 -to ext_flash_enet_bus_data[7]
102 set_location_assignment PIN_H11 -to ext_flash_enet_bus_data[8]
103 set_location_assignment PIN_H12 -to ext_flash_enet_bus_data[9]
104 set_location_assignment PIN_F11 -to ext_flash_enet_bus_data[10]
105 set_location_assignment PIN_E10 -to ext_flash_enet_bus_data[11]
106 set_location_assignment PIN_B9 -to ext_flash_enet_bus_data[12]
107 set_location_assignment PIN_A9 -to ext_flash_enet_bus_data[13]
108 set_location_assignment PIN_C10 -to ext_flash_enet_bus_data[14]
109 set_location_assignment PIN_D10 -to ext_flash_enet_bus_data[15]
110 set_location_assignment PIN_B10 -to ext_flash_enet_bus_data[16]
111 set_location_assignment PIN_A10 -to ext_flash_enet_bus_data[17]
112 set_location_assignment PIN_E12 -to ext_flash_enet_bus_data[18]
113 set_location_assignment PIN_D12 -to ext_flash_enet_bus_data[19]
114 set_location_assignment PIN_I12 -to ext_flash_enet_bus_data[20]
```

SOPC Builder – More “cpu” Settings

- Set **Reset**, **Exception**, and **Break Location** addresses
 - Note: Reset must be in non-volatile memory

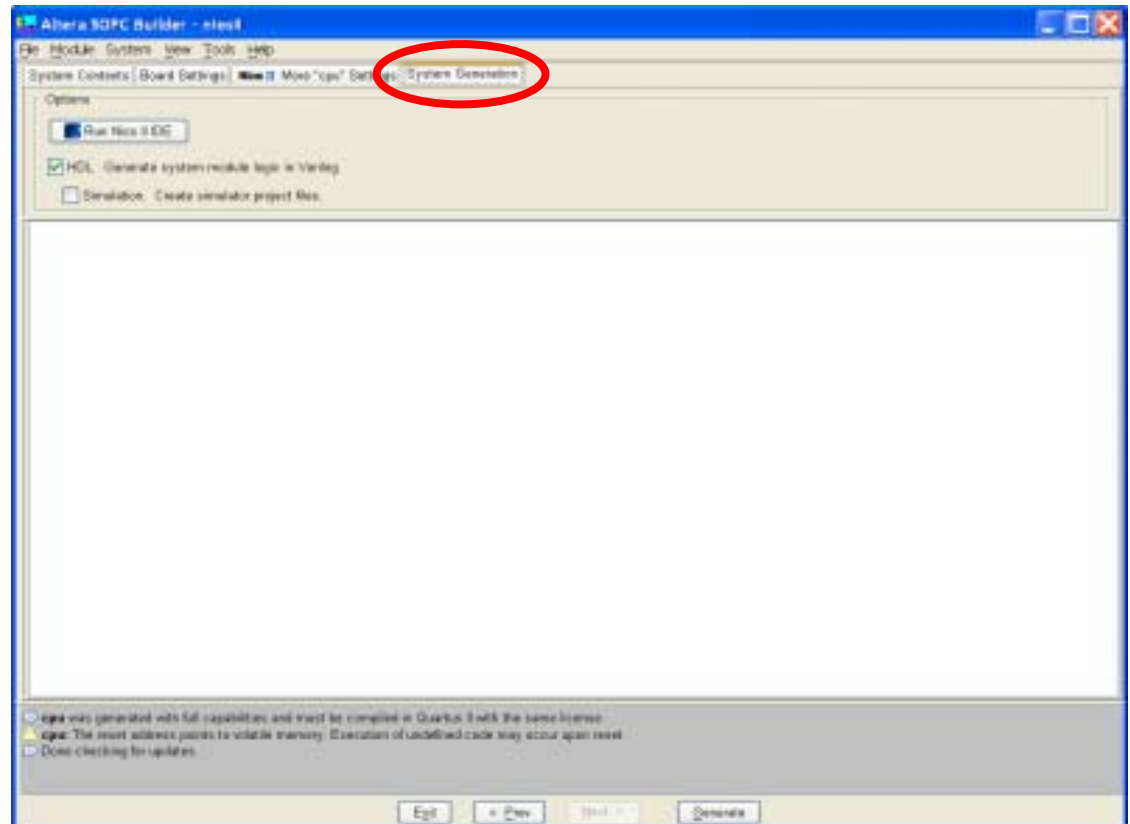


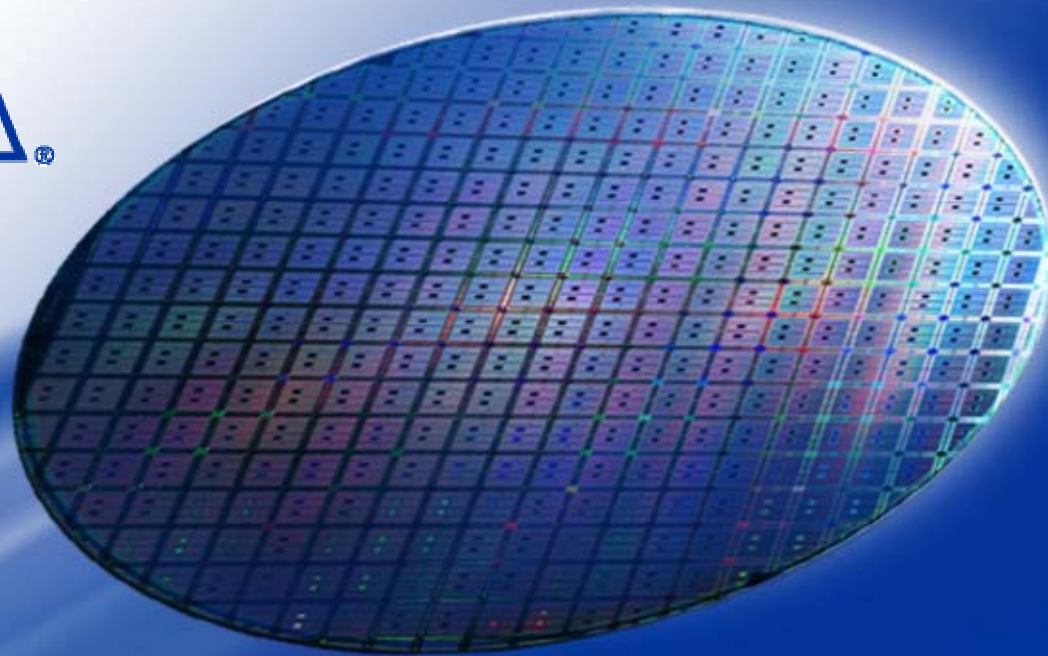
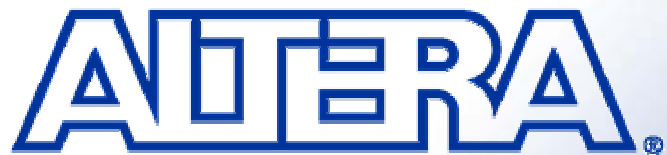
Nios II Exception Address

- All exceptions processed by code at “exception location”
 - Provided by HAL system library
- Supported Exception Types
 - Software Exceptions
 - Software Traps (currently, not implemented)
 - Unimplemented instructions
 - Maintains compatibility between Nios II cores
 - Hardware Interrupts
 - 32 Level-sensitive interrupts are supported.
 - More exceptions will be supported as features are added

SOPC Builder – System Generation

- Generate HDL Code/block symbol file and/or ModelSim Simulation Project
 - Note: Can also launch **Nios II IDE** from here





Peripherals for Nios II

Some Important Peripherals for Nios II

■ System ID Peripheral

- Used to Ensure Hardware/Software Version Synchronization
- Simple 2 read-only register peripheral containing hardware ID tags.
 - Register 1 contains random number
 - Register 2 contains time and date when system was generated in SOPC Builder
- Can be checked at runtime to ensure that the software to be downloaded matches the hardware image

■ Memory Interfaces

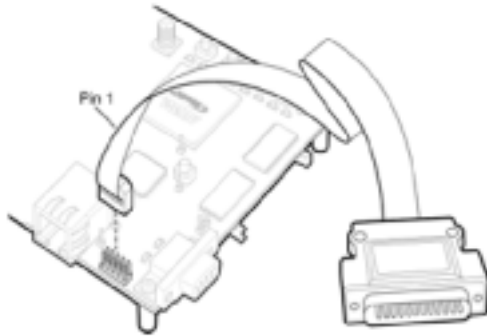
- EPCS Serial Flash Controller
- On-Chip
 - RAM, ROM
- Off-Chip
 - SRAM
 - CFI Flash

■ LCD Display

Other Important Peripherals for Nios II

■ JTAG UART

- Single JTAG Connection For:
 - Device Configuration
 - Flash Programming
 - Code Download
 - Debug
 - Target STDIO (printing)



■ Compact Flash Interface

- Mass Storage Support
 - True IDE Mode
 - Compact Flash Mode
- Software Supports
 - Low-Level API
 - MicroC/OS-II File System Support
 - μ CLinux File System Support

Peripheral Now Provided
with the Nios II IDE and
Supported through the
Nios Forum

www.niosforum.com



New Peripherals for Nios II 5.0

■ **SSRAM Controller**

- Cypress CY7C1380C Sync SRAM controller
 - Provided to support SSRAM component on Cyclone II dev kit board
 - Not a fully configurable general purpose controller

■ **Support for DDR/DDR2 in SOPC Builder GUI**

- With burst adapter
 - Sequential master to interleaved slave enhancement
- Separate READ/Write duplex slaves
 - Automatically matches address of read/write slaves
 - Arbitration logic connects read/write masters to both slaves

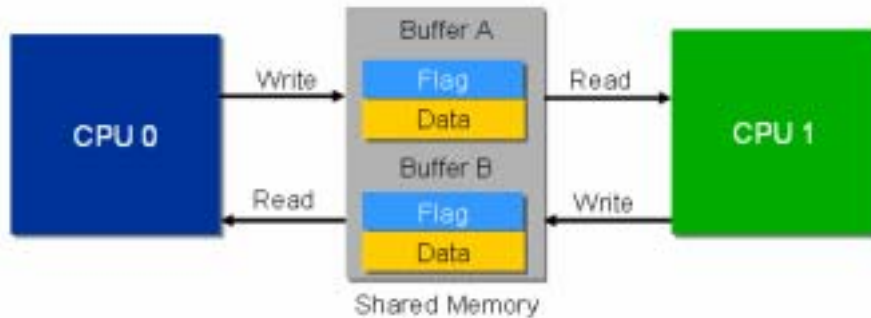
■ **Support for PCI and Bursting DMA in SOPC Builder GUI**

- Higher bandwidth transfers through PCI

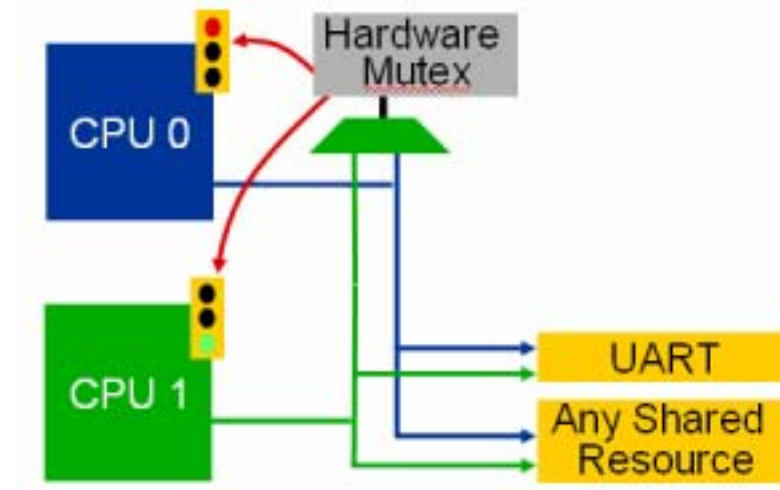
Sharing Data & Peripherals

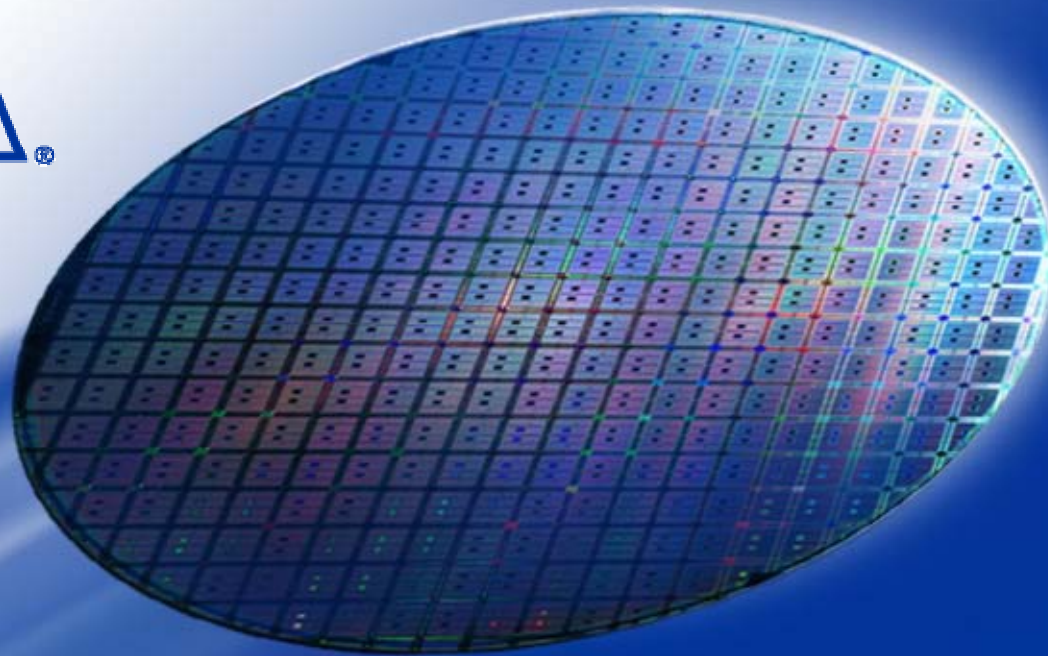
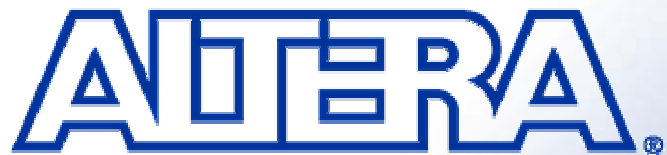
- **Mailboxes** For Inter-Processor (and Multi-Threaded) Message Passing
- Hardware **Mutex** For Safe Resource Sharing

Mailboxes (New in 5.0)



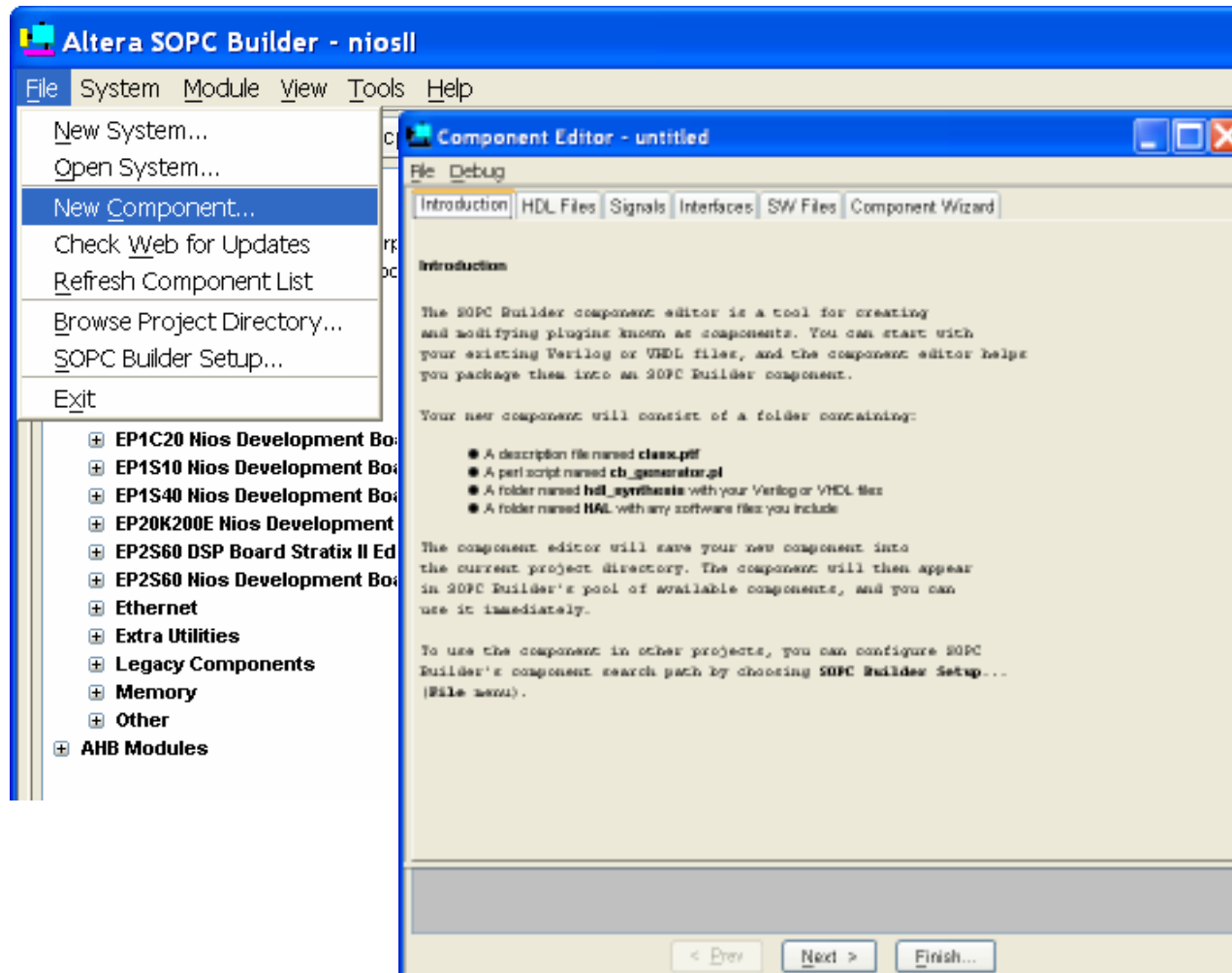
Mutex (New in 1.1)





Custom Peripherals

New Component Editor



Create External Component Interface

- To communicate with off-chip peripherals
- Base interface type on data sheet

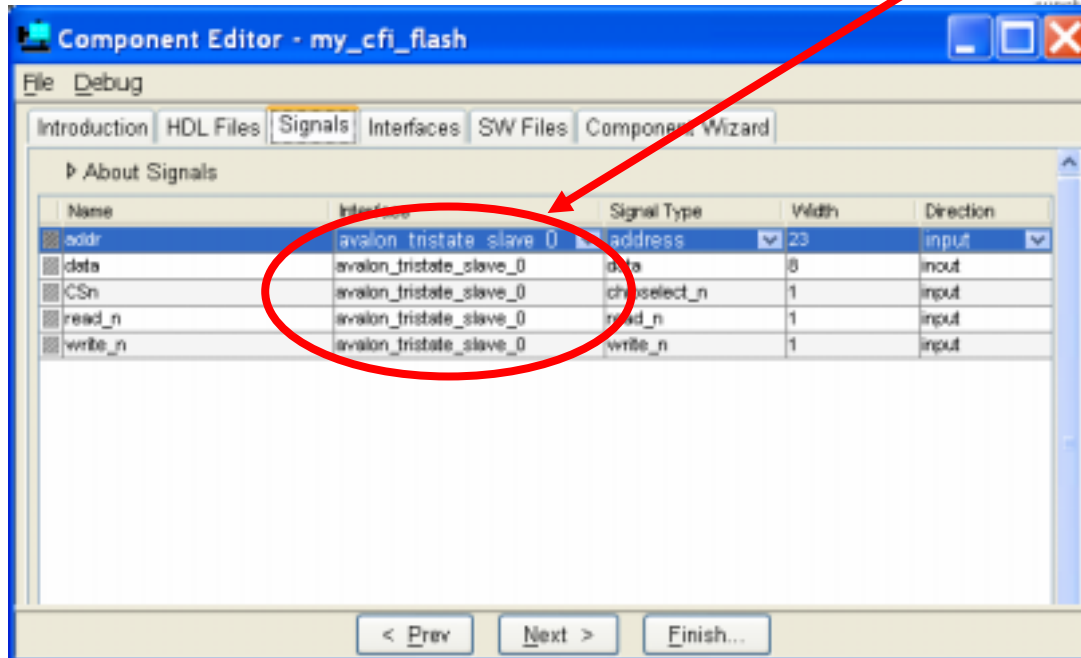
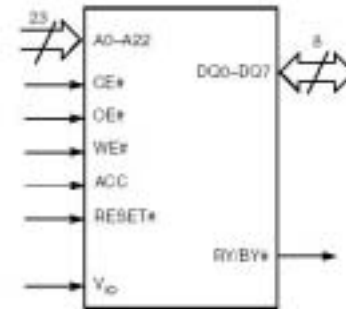


AMD29LV065AD CFI Flash Chip

PIN DESCRIPTION

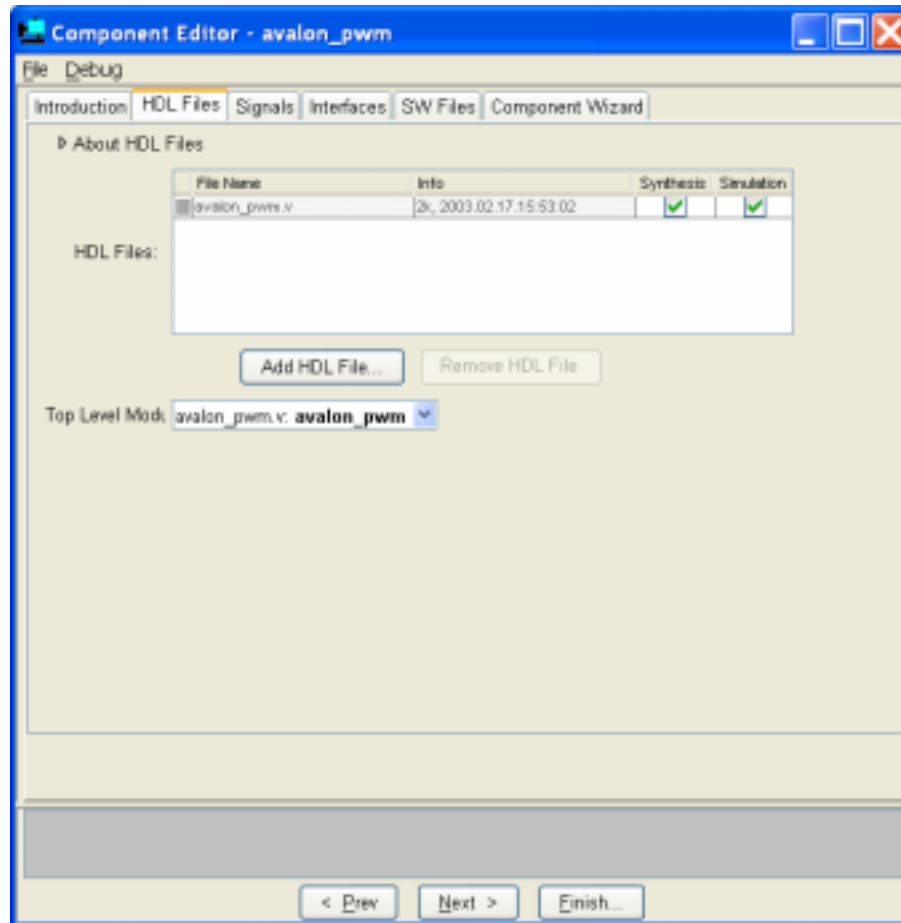
| | |
|-----------------|---|
| A0-A22 | = 23 Addresses inputs |
| DQ0-DQ7 | = 8 Data inputs/outputs |
| CE# | = Chip Enable input |
| OE# | = Output Enable input |
| WE# | = Write Enable input |
| ACC | = Acceleration Input |
| RESET# | = Hardware Reset Pin input |
| RY/BY# | = Ready/Busy output |
| V _{CC} | = 3.0 volt-only single power supply (see Product Selector Guide for speed options and voltage tolerances) |

LOGIC SYMBOL

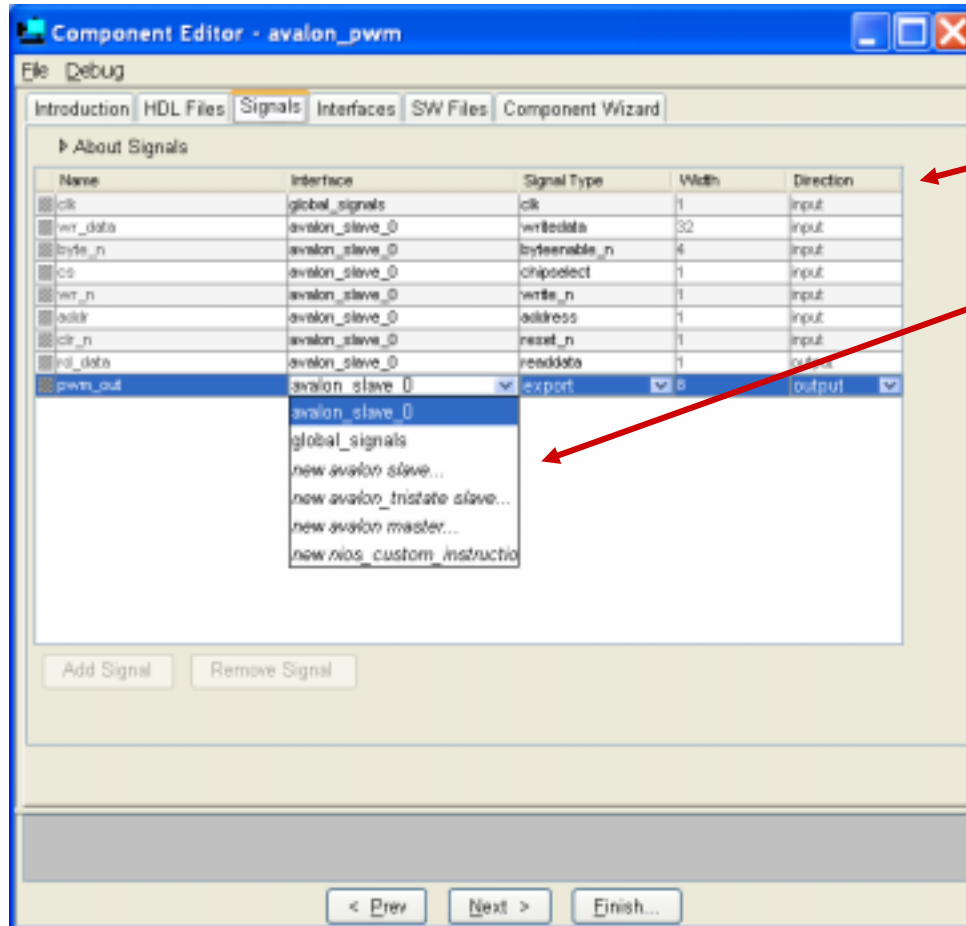


Or Add HDL Files

- For peripheral that has been encoded for FPGA



Define Component Signals

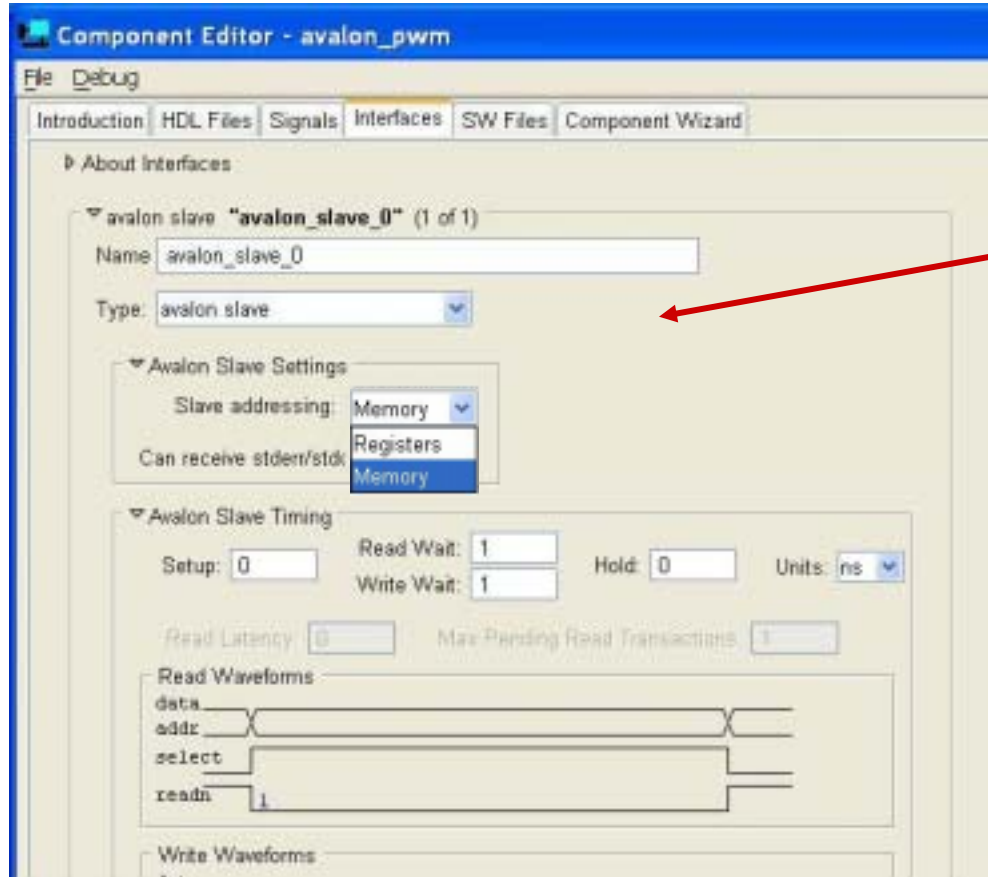


Automatically populates port table from design files

Enter port type here

Can also define ports manually

Define Interface for Each Signal Type



Choose interface type

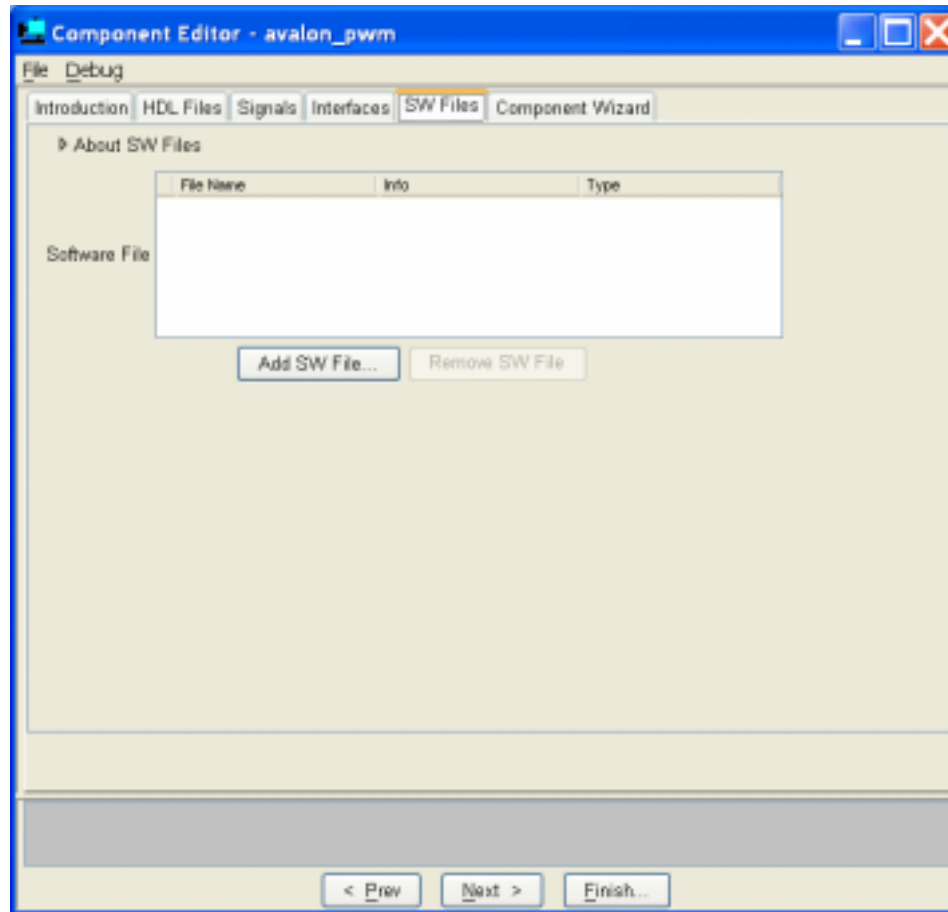
Register Slave uses native alignment, **Memory Slave** uses dynamic alignment

Control Read and Write Timing

Add wait and hold states View waveforms

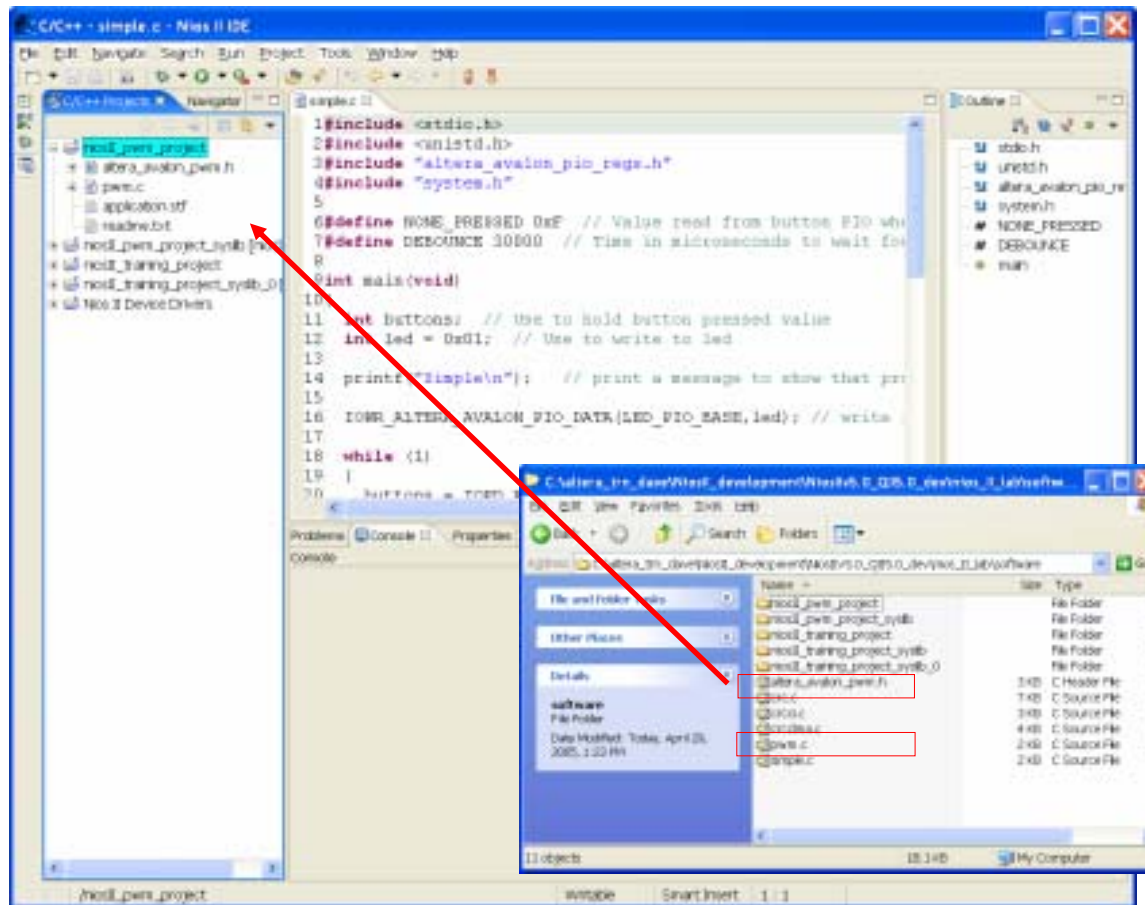
Add Software Files

- ie. Header files and drivers



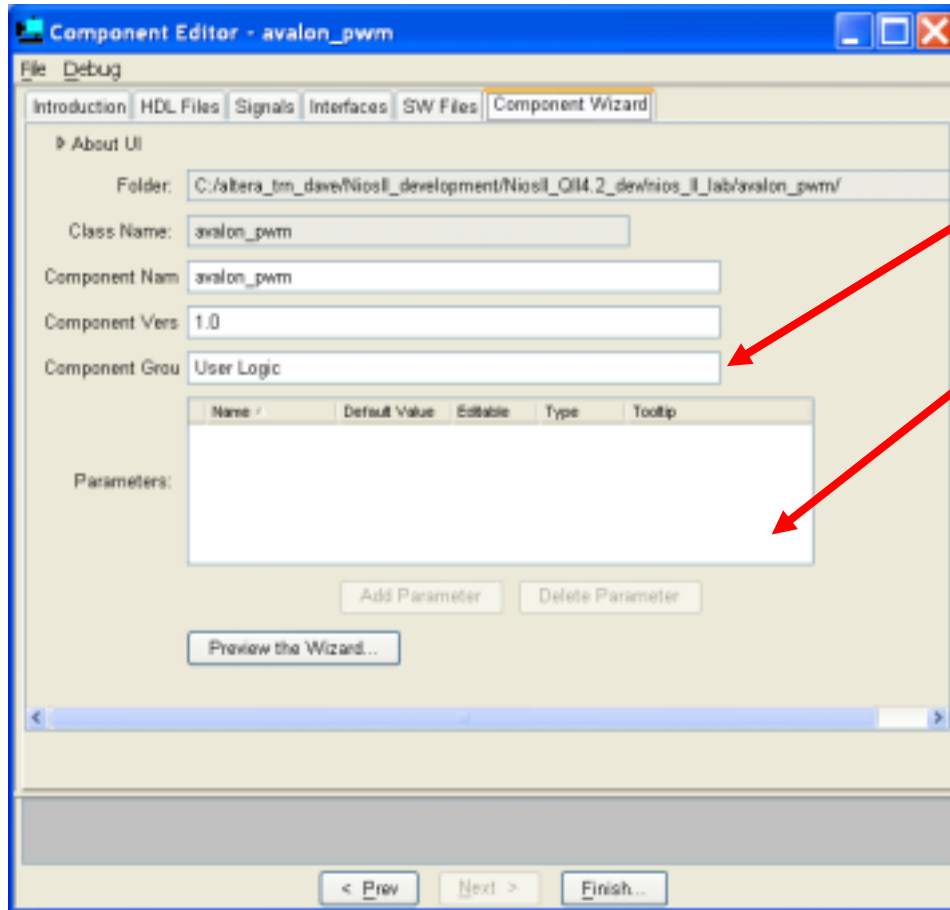
Add Software Files

- Header file and drivers can also be added directly to Application Project



Create Component Wizard

- Publish and create a wizard for your component



The screenshot shows the 'Component Editor - avalon_pwm' window with the 'Component Wizard' tab selected. The window contains the following fields and controls:

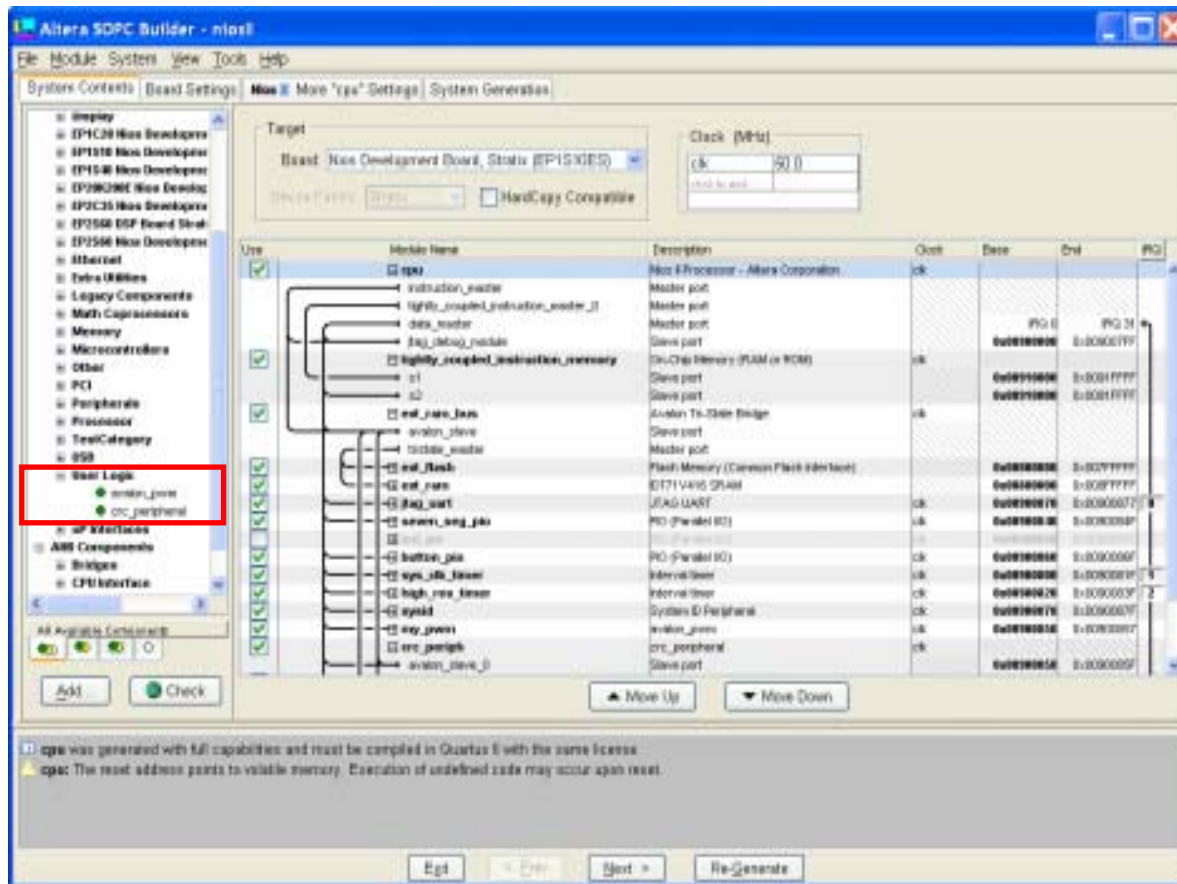
- Folder:** C:/altera_bin_dave/NiosII_development/NiosII_QII4.2_dev/nios_ii_lab/avalon_pwm/
- Class Name:** avalon_pwm
- Component Name:** avalon_pwm
- Component Vers:** 1.0
- Component Group:** User Logic
- Parameters:** A table with columns: Name, Default Value, Editable, Type, and Tooltip. The table is currently empty.
- Buttons:** 'Add Parameter', 'Delete Parameter', 'Preview the Wizard...', '< Prev', 'Next >', and 'Finish...'.

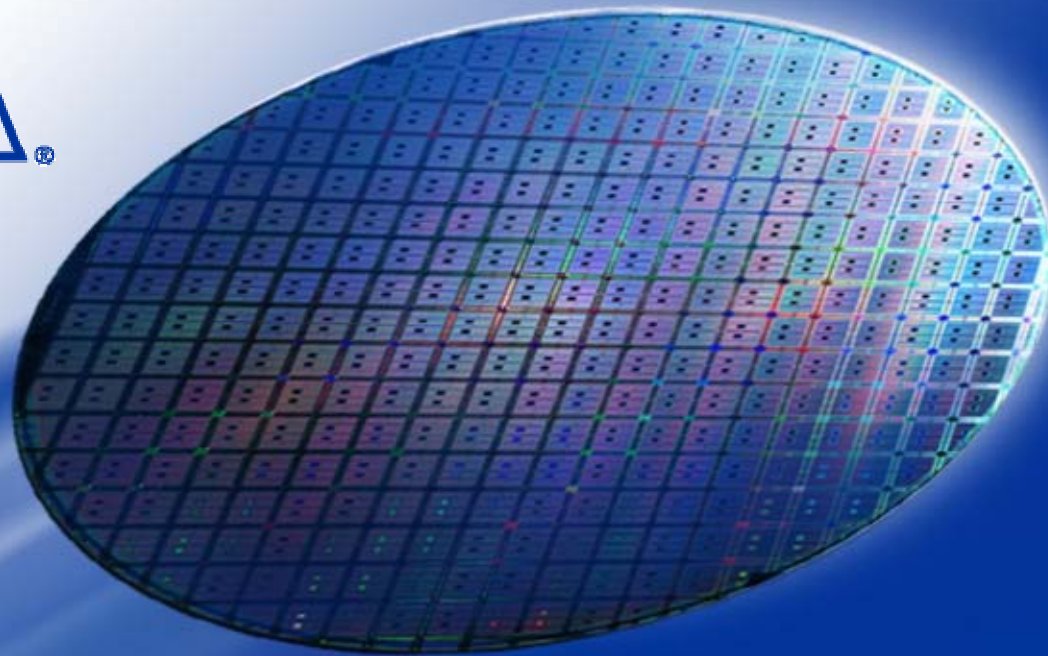
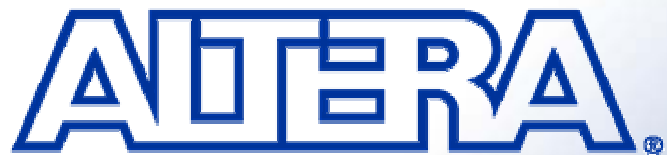
Two red arrows point from the text on the right to the 'Component Group' field and the 'Parameters' table.

- Fill in fields
- Add component to SOPC Builder portfolio
- Can add parameterizing capability to component

Add Component to SOPC System

- Default location is the **User Logic** folder

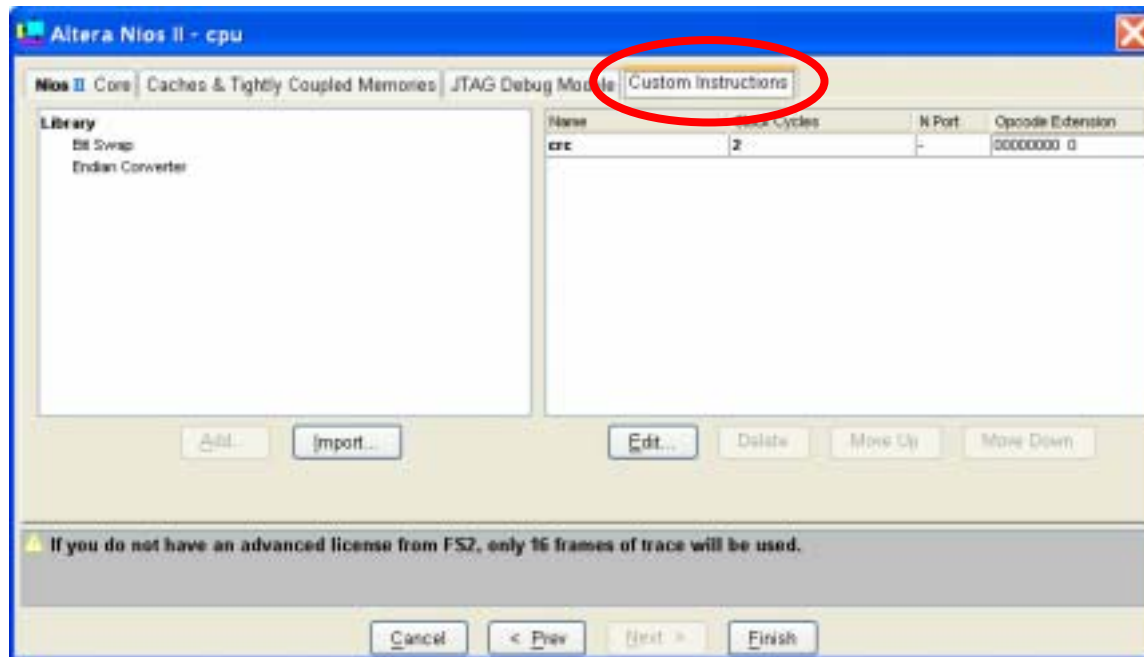




Custom Instructions

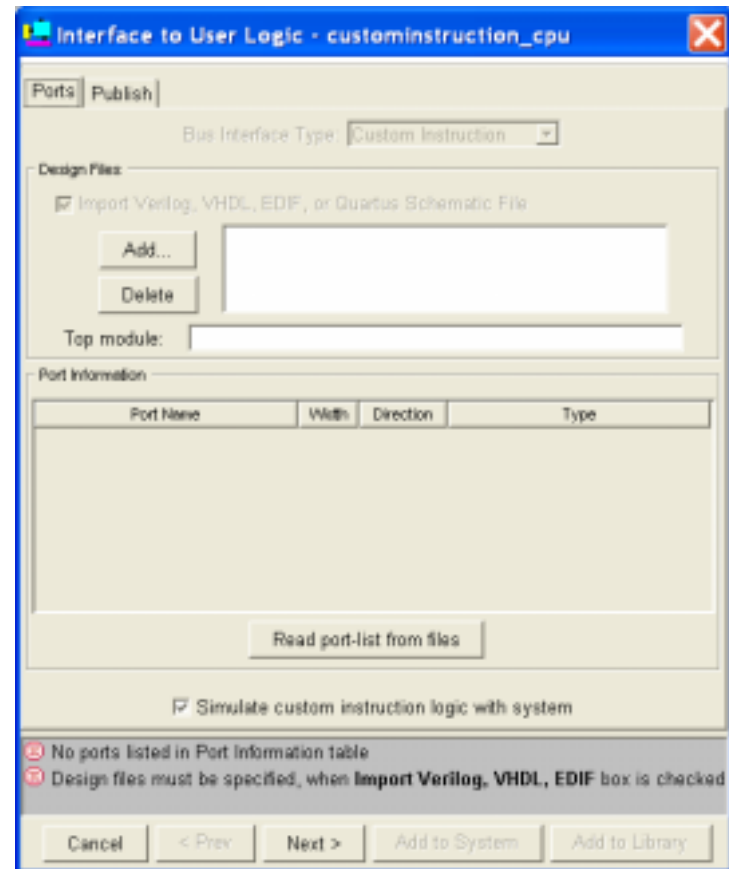
Custom Instructions Tab

- Enabled from the Custom Instructions tab in the Nios II CPU settings in SOPC Builder



Custom Instructions Tab

- Import logic for the custom instruction
- Custom Instruction module can be of following formats:
 - VHDL
 - Verilog HDL
 - EDIF
 - Quartus Block Diagram (.bdf)

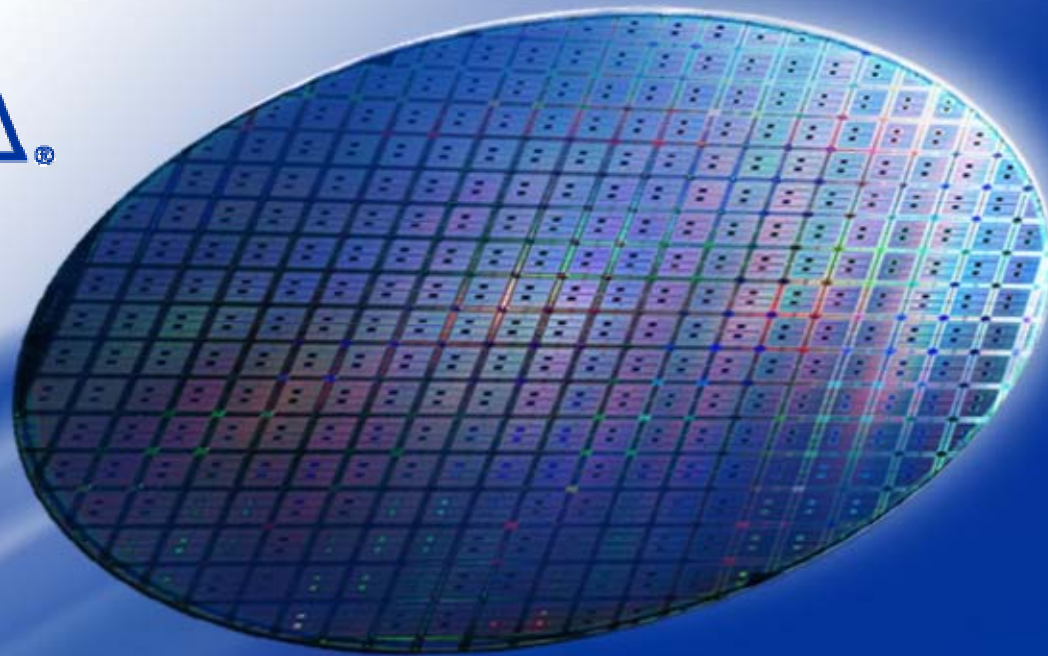


Software Interface - C

- NIOS II IDE generates macros automatically during build process
- Macros defined in **system.h** file
 - #define ALT_CI_<your instruction_name>(instruction arguments)
- Example of user C-code that references Bitswap custom instruction:

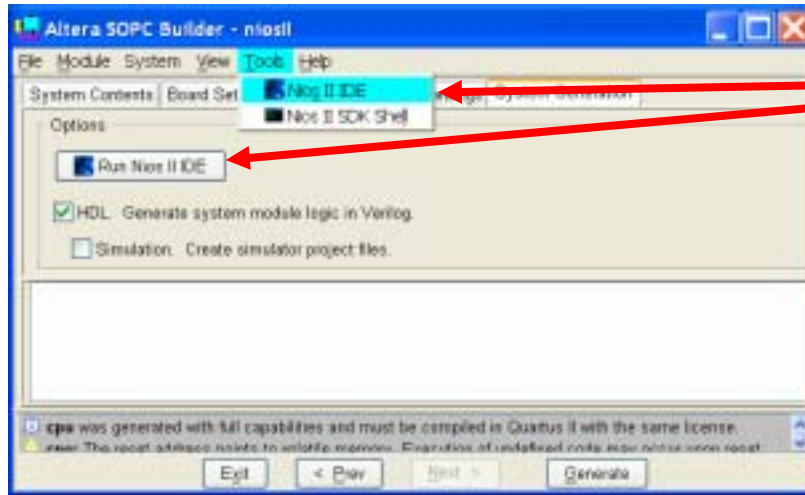
```
#include "system.h"
int main (void)
{
    int a = 0x12345678;
    int a_swap = 0;

    a_swap = ALT_CI_BSWAP(a);
    return 0;
}
```

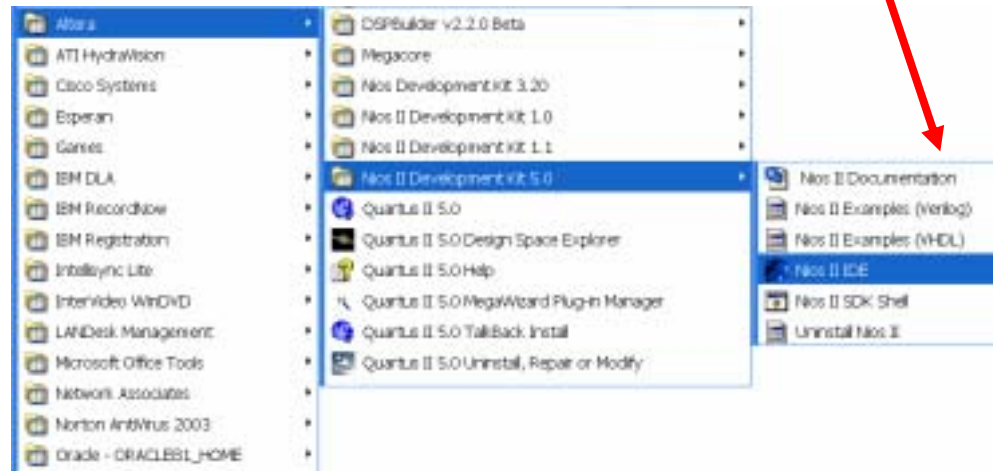


Software Project in NiosII IDE

Opening the Nios II IDE



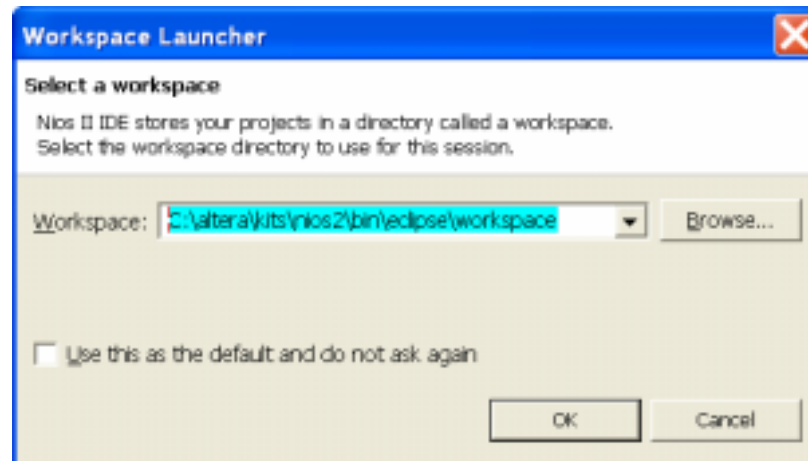
Launch the Nios II IDE from the SOPC Builder or from the Windows Start menu



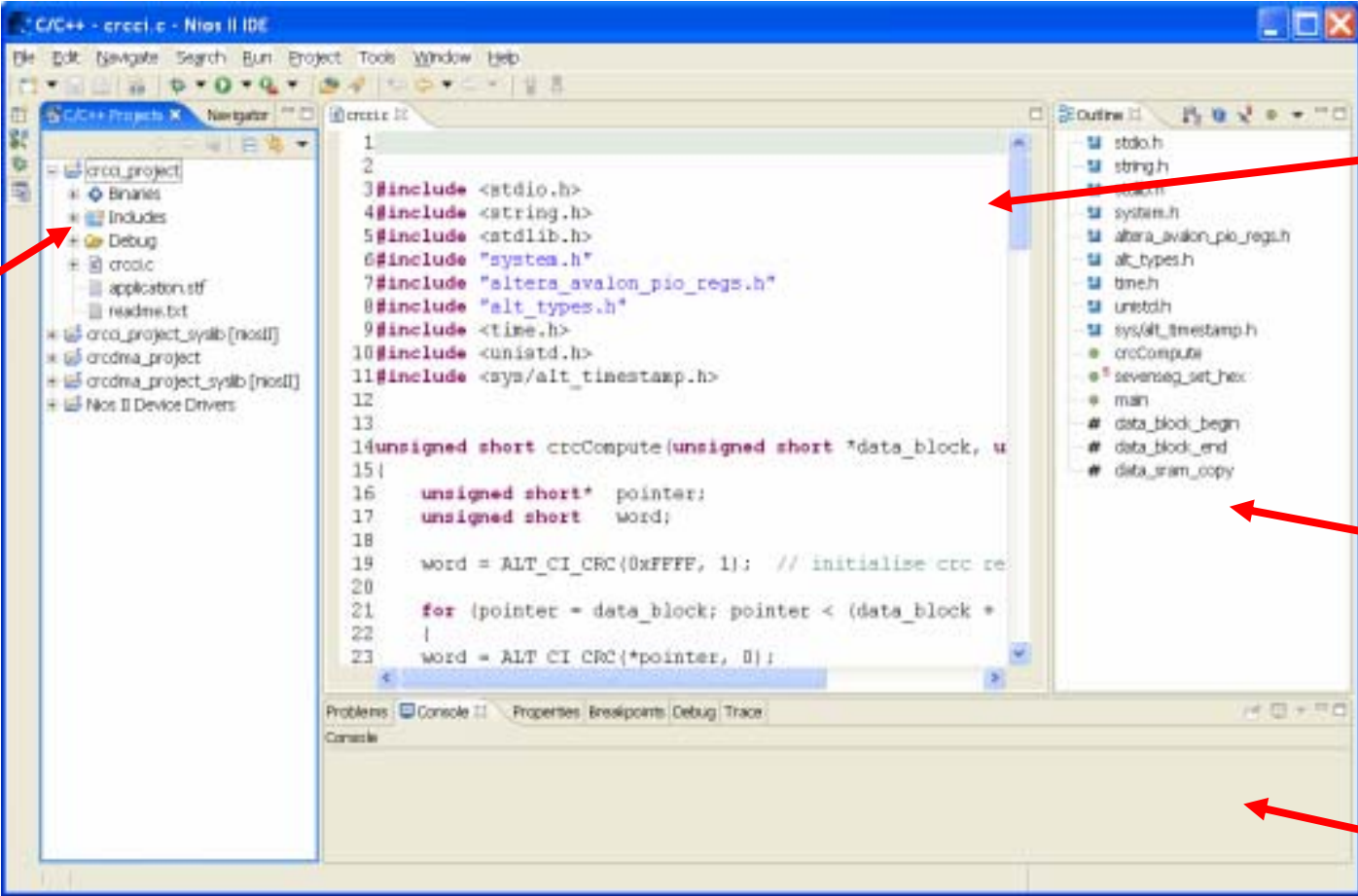
Workspace Dialog Box

New

- Appears when you first open Nios II IDE
 - Before main tool opens
- Can now open Multiple Nios II IDE sessions
 - Pick workspace each time Nios II IDE opened
 - Default C:\altera\kits\nios2\bin\eclipse\workspace
- Each “workspace” has its own settings



Nios II IDE Workbench



The screenshot displays the Nios II IDE Workbench interface. On the left is the **Navigator** window showing a project tree with folders like 'Binaries', 'Includes', 'Debug', and 'crocic'. The central **Editor** window shows a C++ source file with various include statements and a CRC computation function. On the right is the **Outline** window listing symbols such as 'stdio.h', 'string.h', 'system.h', and 'crcCompute'. At the bottom is the **Console** window, which is currently empty. Red arrows point from text labels to each of these four main components.

List of Open Projects

File Viewer Window
(for C code, C++, and assembly*)

Outline View
(view and/or open funcs, enums, class unions, struct typedefs, etc.)

Terminal window

• Note: C++ files must have extension `.cpp`
In-line assembly code offset by `asm()`;

This Creates Two Software Projects

- Application *and* System Library Project

Application Project

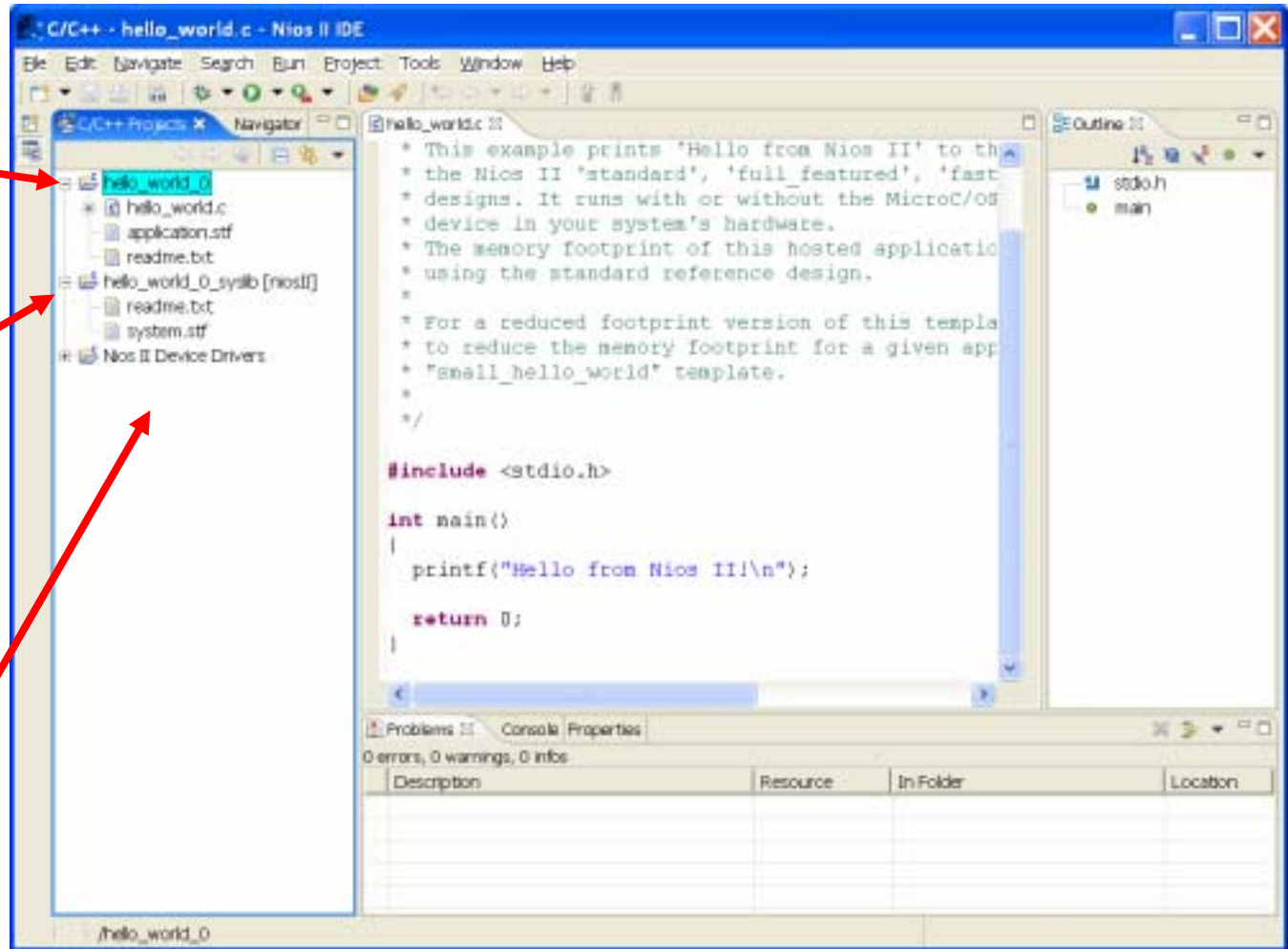
- contains application source code

System Library Project

- contains system header file, etc.

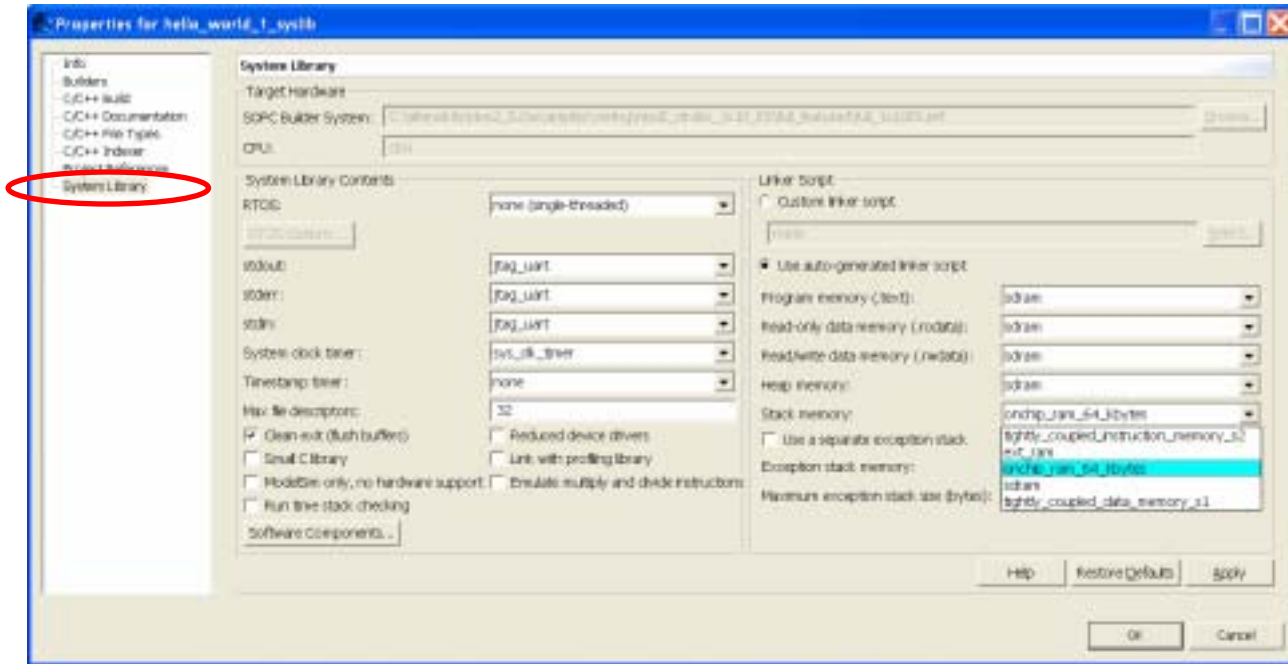
Drivers Directory

- contains all device drivers
- DO NOT DELETE !



System Library Project Properties

Specify stdio devices



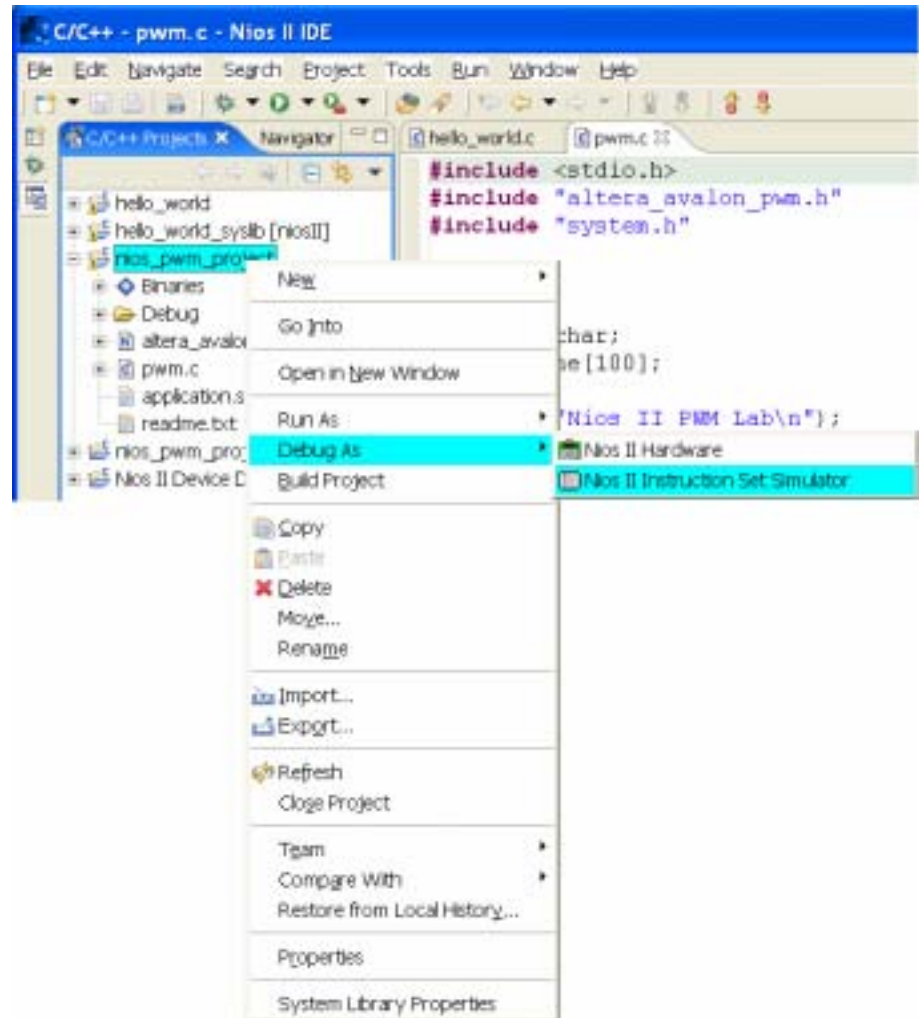
Partition the memory map

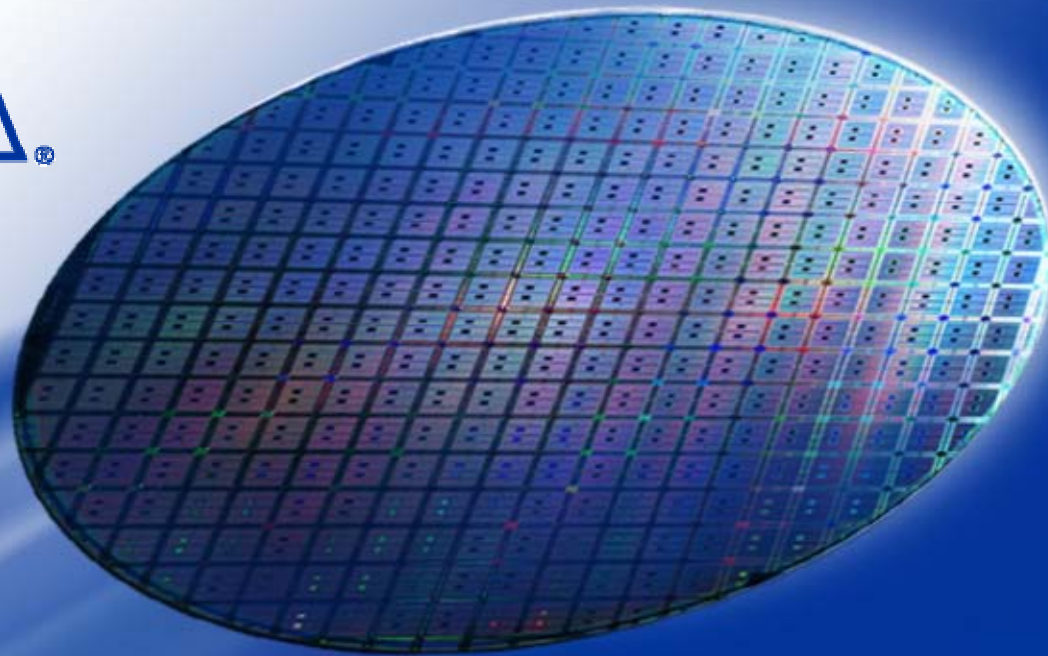
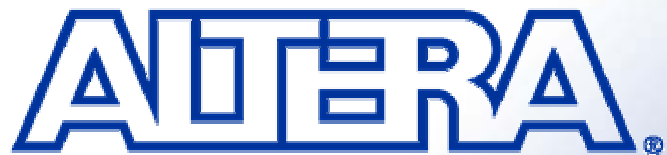
Nios II Instruction Set Simulator

- **Instruction Set Simulators** are software models of an Instruction Set Architecture
 - Generally used to debug code if a target board is unavailable
 - Provides limited models of a few hardware peripherals
 - Timer
 - UART
 - Memory (flash, SDRAM, on-chip, etc...)

Nios II Instruction Set Simulator

- Launch an ISS Debug session from the Run Menu
- Targets .elf file to ISS and opens debugger
 - Application can then be debugged as normal



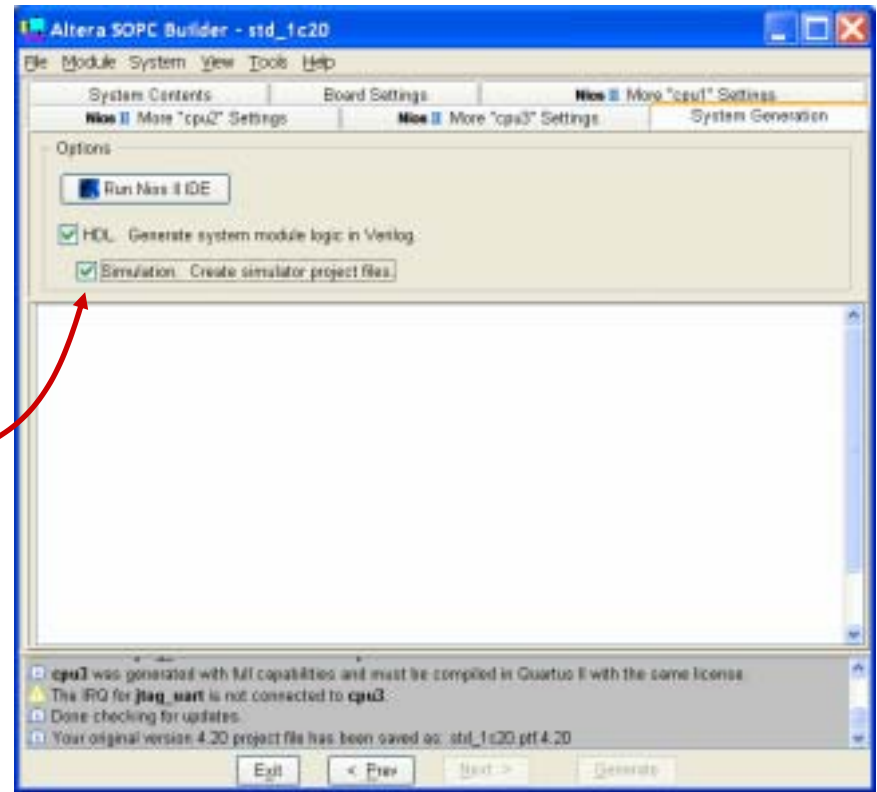


RTL Simulation

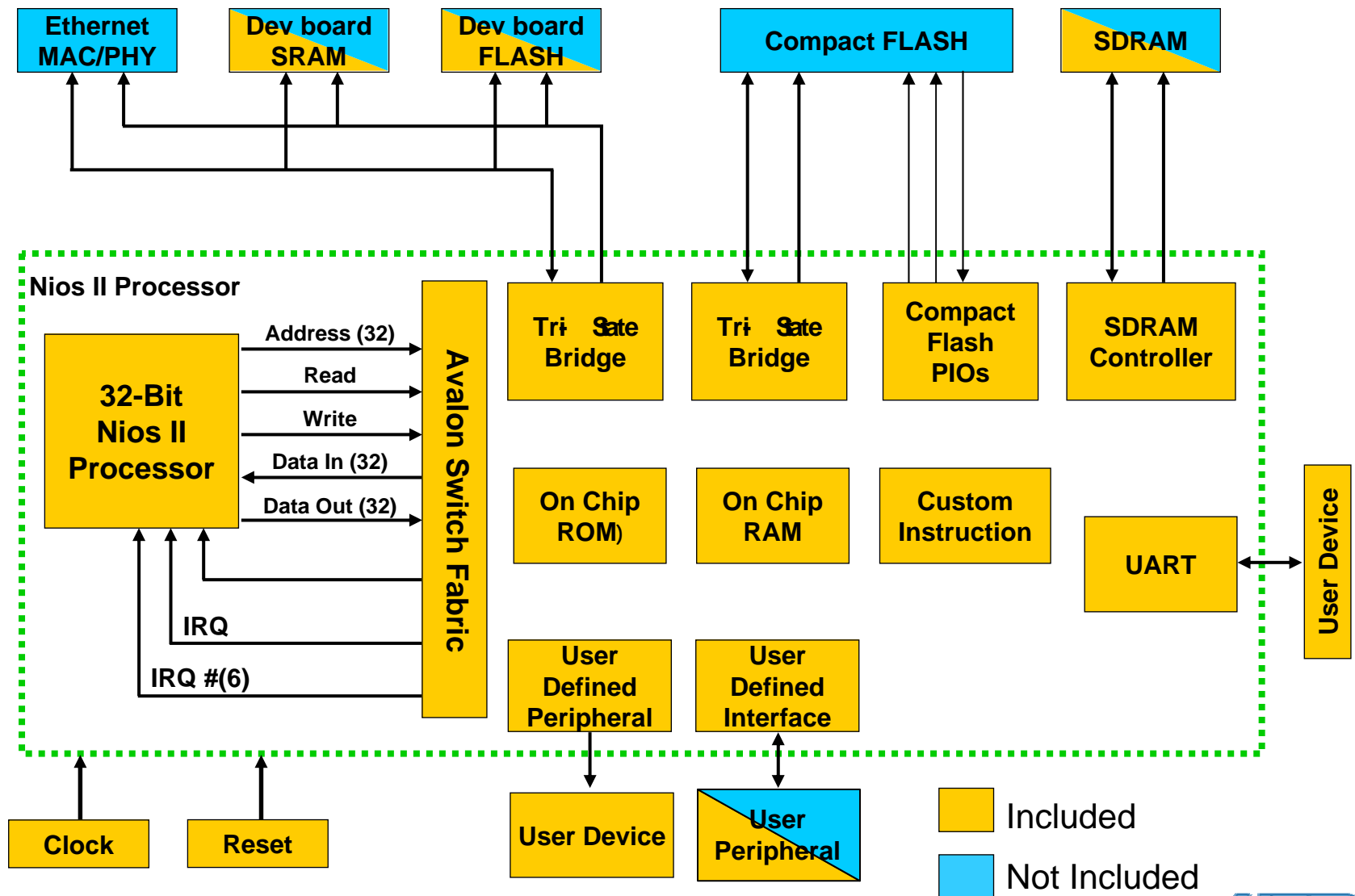
RTL Simulation

- Nios II SOPC Builder Automatically Creates Simulation Models Plus:
 - ModelSim Project
 - Testbench
 - Simulation Scripts

**Be Sure to Set
Simulation Option**



Simulation TestBench



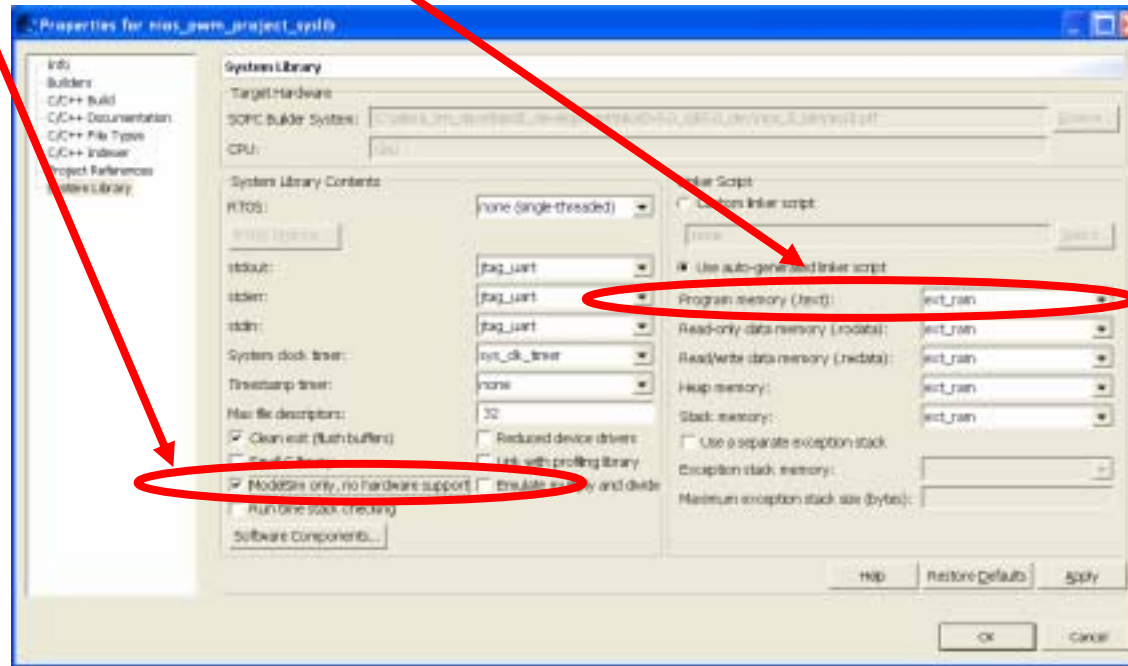
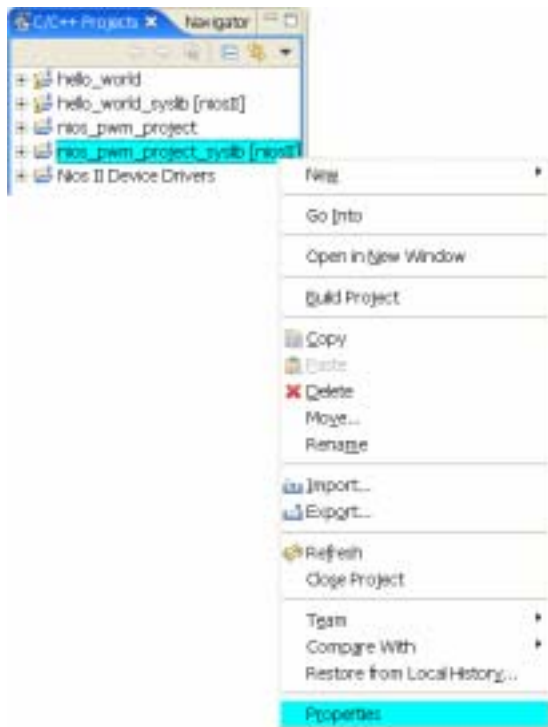
User Additions to Nios II TestBench

```
3784
3785 // <ALTERA_NOTE> CODE INSERTED BETWEEN HERE
3786 //include additional files here
3787 // AND HERE WILL BE PRESERVED </ALTERA_NOTE>
3788
3789
3790
3791 module test_bench ;
3792
3793
3794 wire    [ 11: 0] in_port_to_the_button_pio;
3795 reg     clk;
3796 wire    [  1: 0] bidir_port_to_and_from_the_led_pio;
3797 wire    [  3: 0] be_n;
3798 wire     write_n_to_the_ext_flash;
3799 wire    [ 15: 0] address;
3800 wire     select0_n_to_the_ext_ram;
3801 wire     uart1_sl_readyfordata_from_sa;
3802 wire    [ 19: 0] off_chip_bus_address;
3803 wire     write_n;
3804 wire    [  3: 0] off_chip_bus_byteenablen;
3805 reg     reset_n;
3806 wire    [ 31: 0] off_chip_bus_data;
3807 wire     select1_n_to_the_ext_ram;
3808 wire    [ 15: 0] out_port_from_the_seven_seg_pio;
3809 wire     select0_n;
3810 wire     off_chip_bus_readn;
3811 wire     read_n;
3812 wire     txd_from_the_uart1;
3813 wire     rxd_to_the_uart1;
3814 wire     select_n_to_the_ext_flash;
3815 wire     uart1_sl_dataavailable_from_sa;
3816 wire    [ 31: 0] data;
3817 wire     select1_n;
3818 wire     write_n_to_the_ext_ram;
3819
3820
3821 // <ALTERA_NOTE> CODE INSERTED BETWEEN HERE
3822 //  add your signals and additional architecture here
3823 // AND HERE WILL BE PRESERVED </ALTERA_NOTE>
3824
```

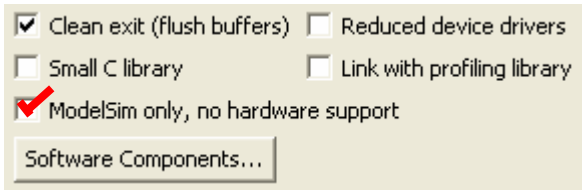
- SOPC Builder creates testbench embedded in top level file (eg. niosII.v)
- Sections within this file are reserved to add user files and code
- These sections are preserved if the SOPC builder is used to re-generate the Nios II system

Running an RTL Simulation

- Modify Nios II IDE System Library For Simulation:
 - Specify Program Memory
 - Set Up As Simulation Only



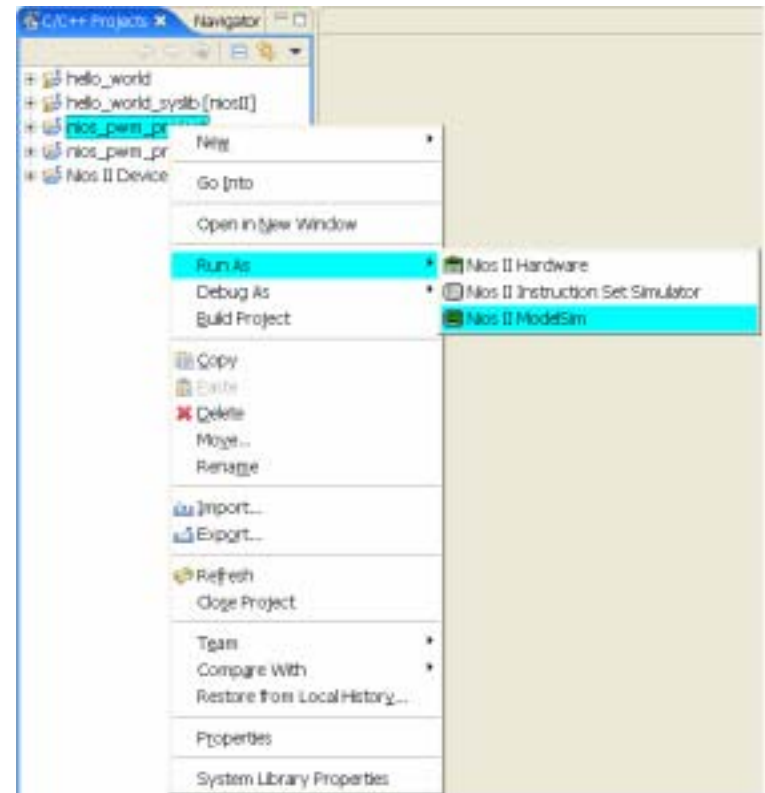
Running an RTL Simulation



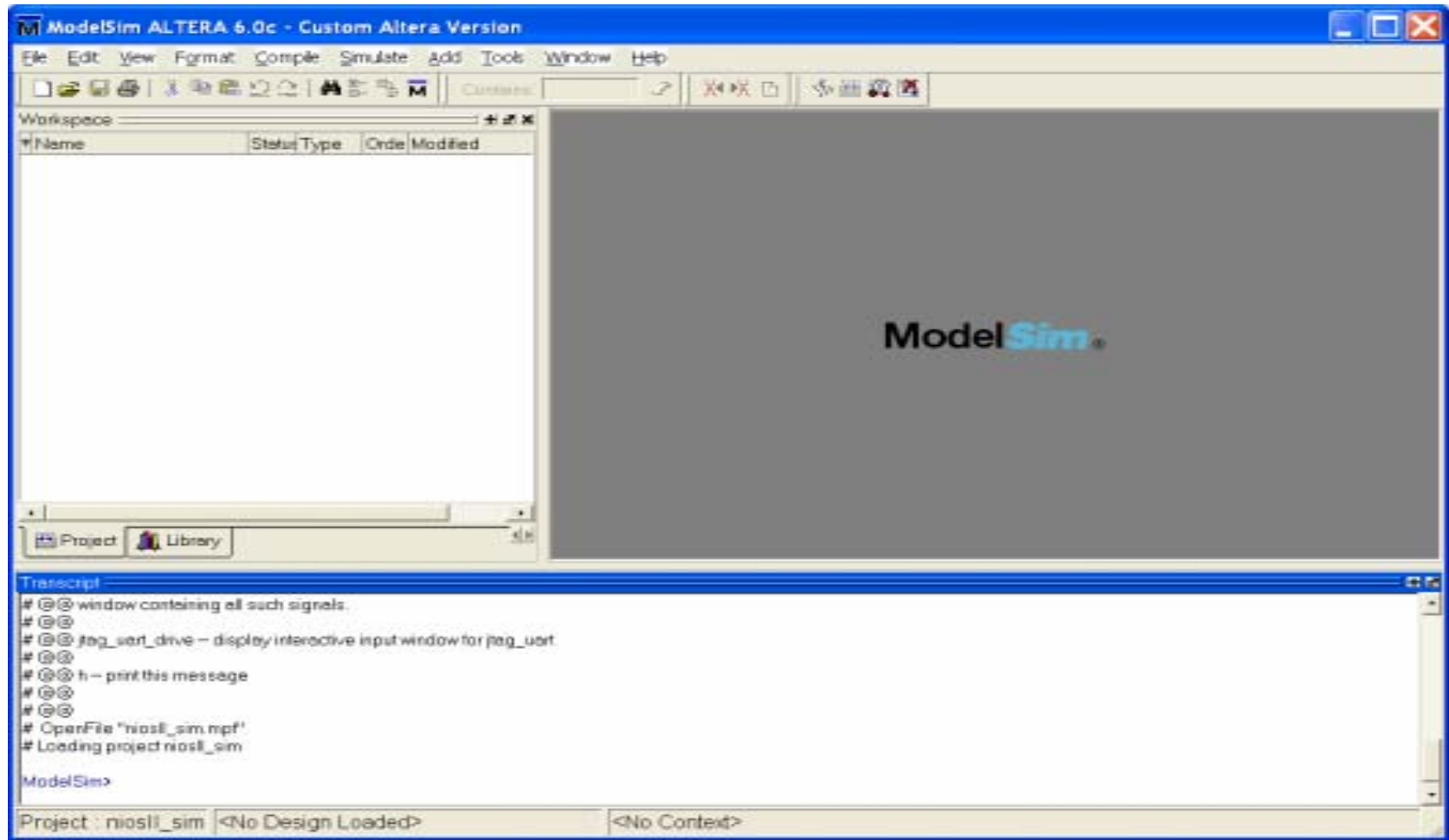
- Checking the “ModelSim only, no hardware support” button:
 - Leaves caches uninitialized
 - Does not initialize the .bss section
- As a result simulation speeds are increased
- You can still simulate with this button unchecked but simulation time will be much longer

Running an RTL Simulation

- Launch ModelSim from Nios II IDE:
 - Highlight Software Project In **C/C++ Projects** panel
 - Right click
 - Run As Nios II ModelSim



Running an RTL Simulation

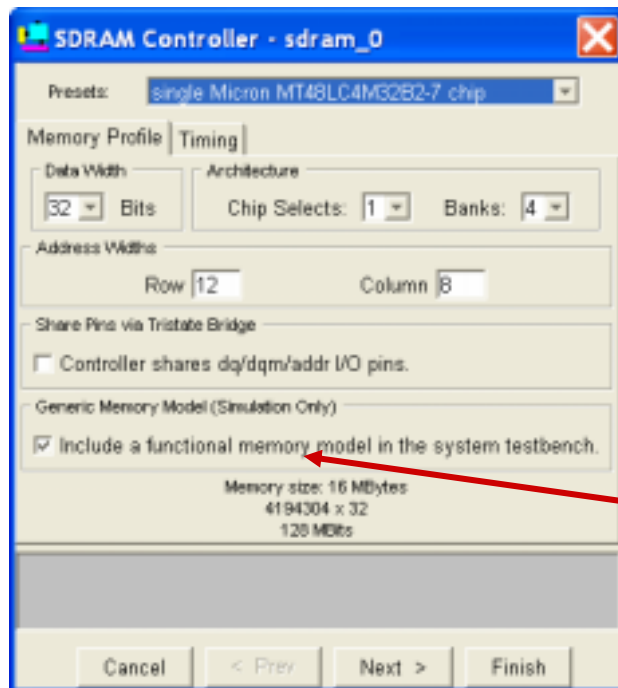


Simulation Scripts

- When ModelSim is started from the Nios II IDE a set-up script is run automatically which creates aliases for simulation scripts
- The set up script can also be run independently as follows:
 - do setup_sim.do
- Simulation Scripts
 - s↵ Compiles HDL source code and loads design
 - c↵ Rebuilds memory contents based on software code
 - Includes changes since Nios II generation
 - w↵ Opens Wave window with “useful” signals
 - l↵ Opens List window with “useful” signals
 - h↵ Displays help message describing scripts

Memory Device Simulation Models

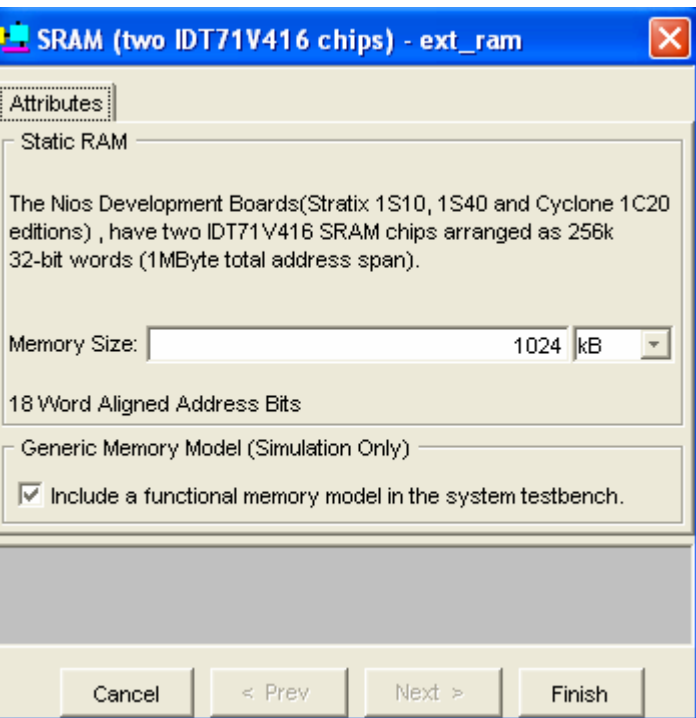
- Applies To The Following Nios II Memories
 - On Chip Memory (ROM or RAM)
 - SRAM
 - Flash Memory and now SDRAM



Include SDRAM Model
for Simulation

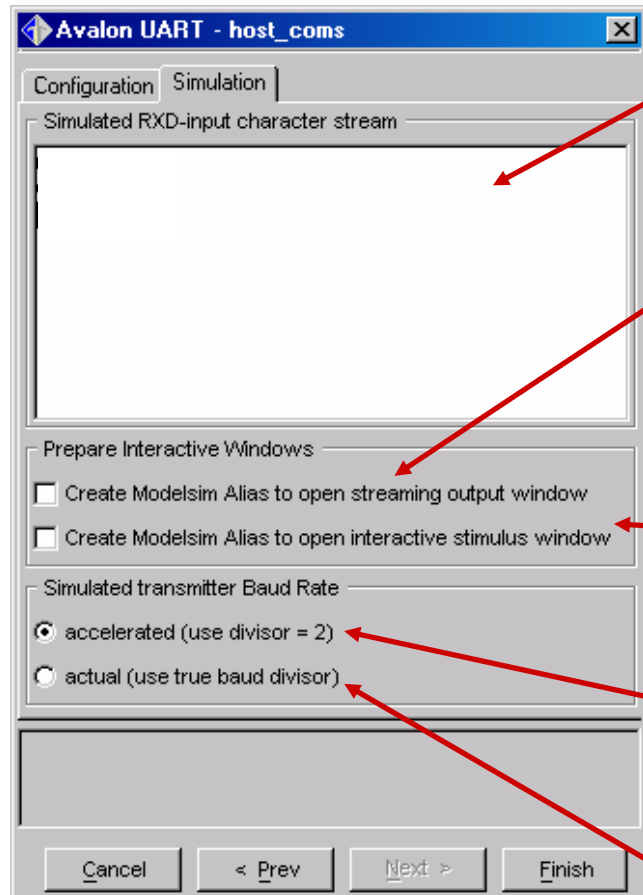
Memory Device Simulation Models

- You can no longer initialize memories in the SOPC Builder
 - Memory init file are created by the Nios II IDE



- ext_ram will be initialized for simulation with the ext_ram.dat file
- You must compile your software in the Nios II IDE to generate this file
- Onchip memories are initialized with <component_name>.hex
- Onchip memory init files can be created by an editor or by the Nios II IDE

UART Simulation

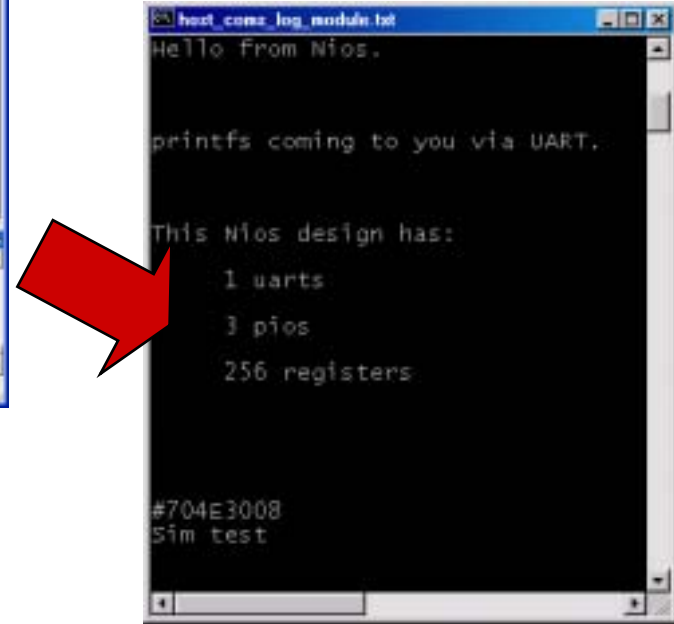
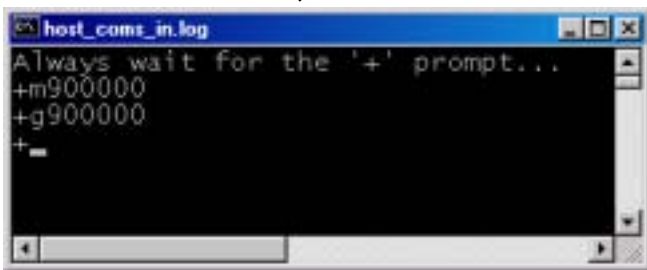
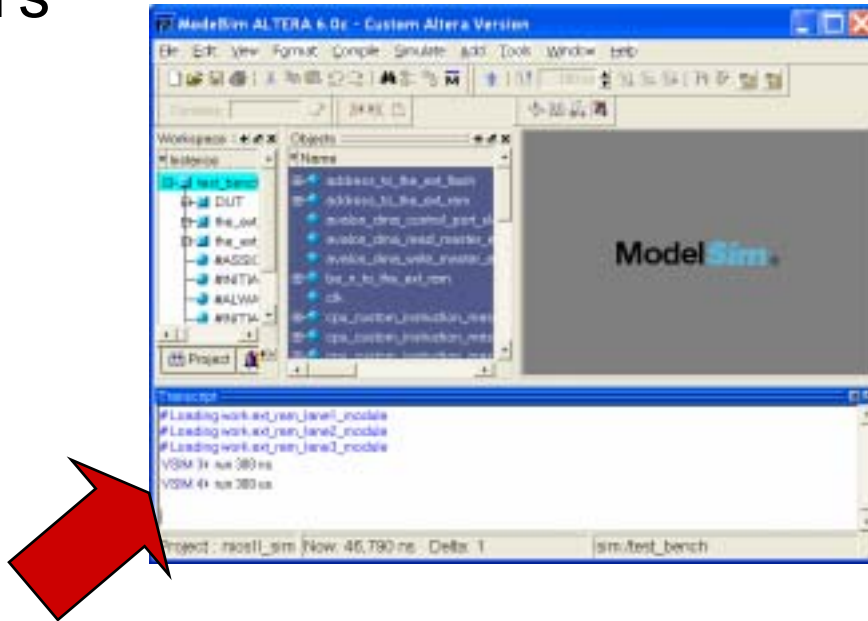


- Text is transmitted to UART during simulation
- Creates and saves txt file containing UART tx stream
- Creates window to input text at simulation run time

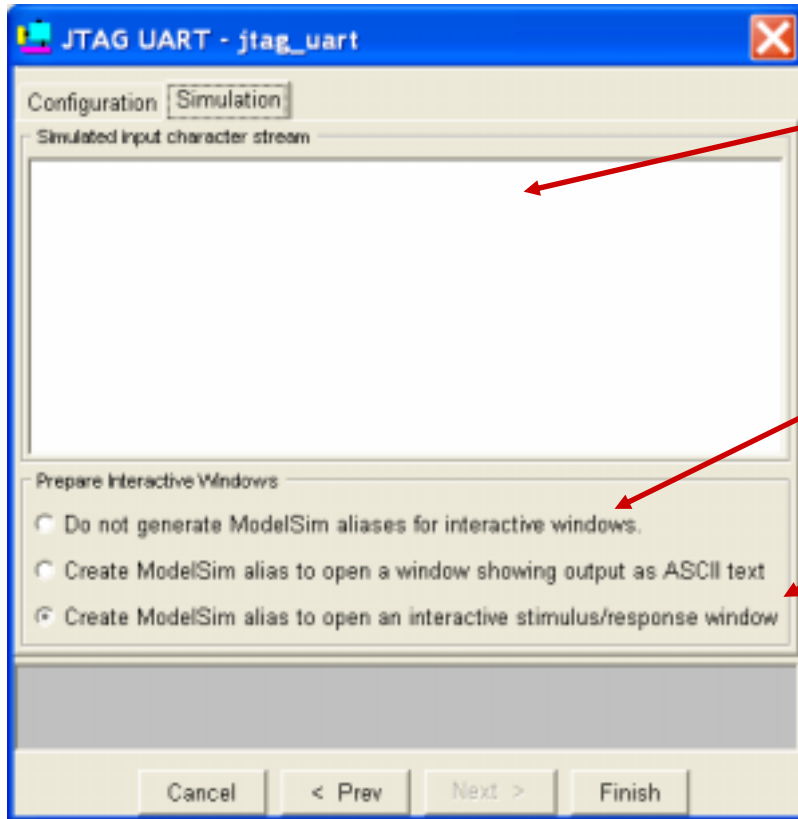
Note: ModelSim Options are mutually exclusive

UART Simulation

- Input is interactive or predefined
- Output is shown and saved independently for multiple UARTs



JTAG_UART Simulation

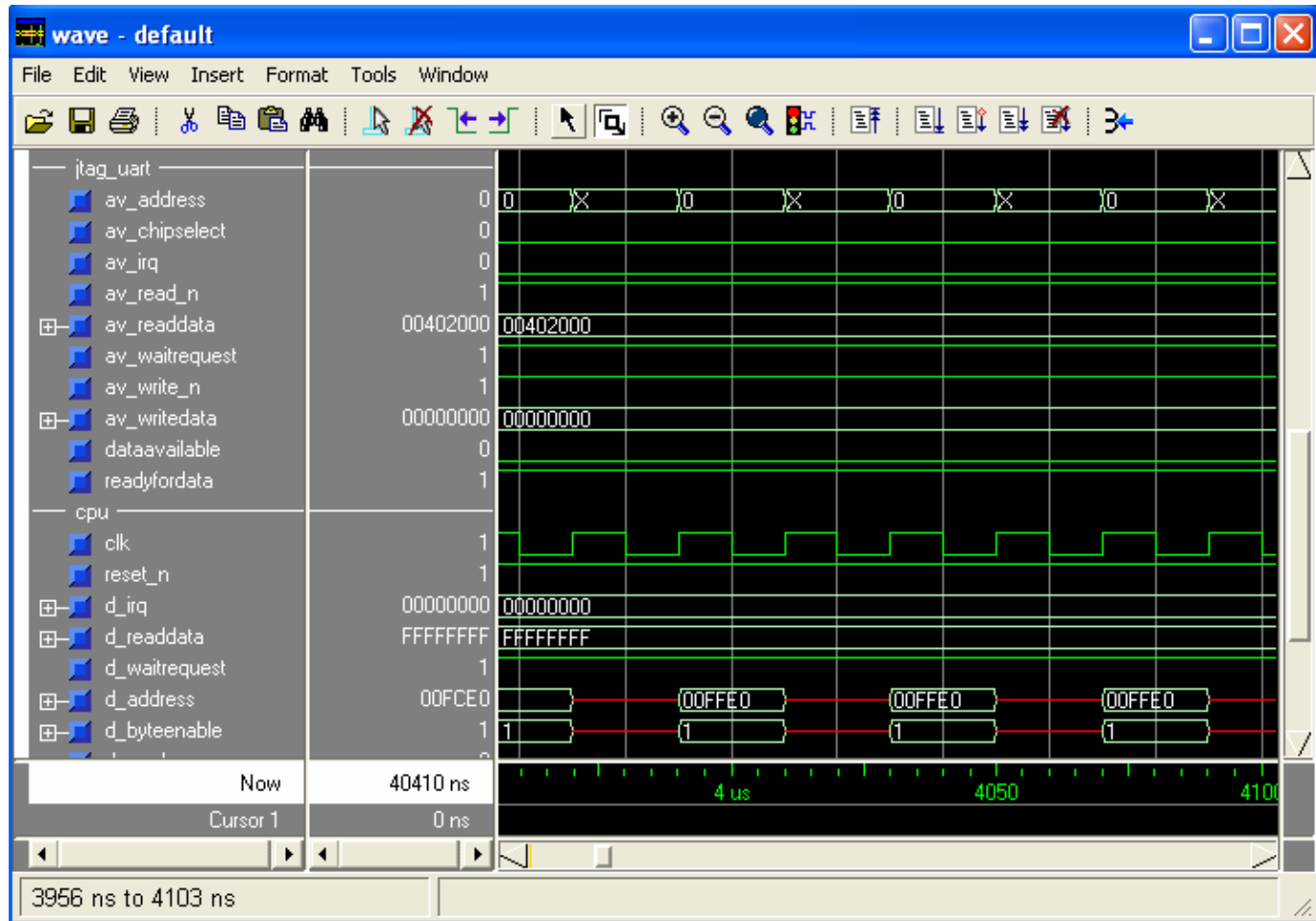


- Text is transmitted to the new JTAG_UART peripheral during simulation
- Creates and saves txt file containing UART tx stream
- Creates window to input text at simulation run time

Note: ModelSim Options are mutually exclusive

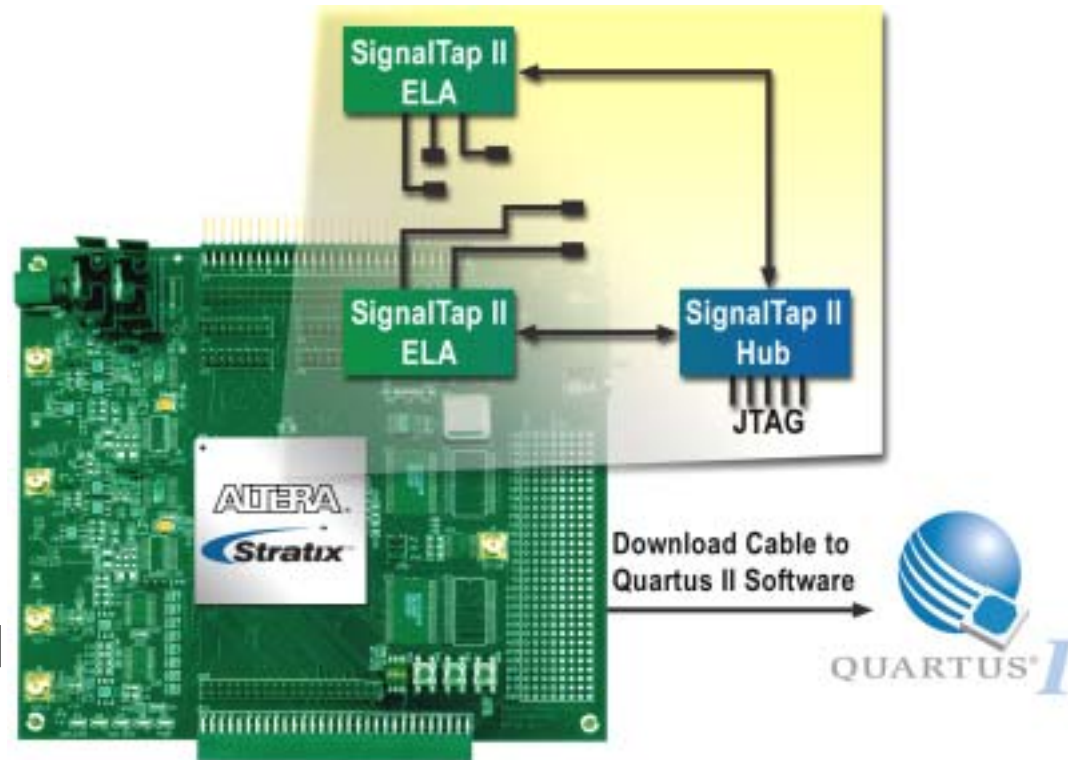
Wave Window

- Adds UART and CPU signals by default



SignalTap™ II Logic Analyzer

- Up to 200 MHz
- Multi-Analyzer Support
- 1,024 Channels
- 128K Samples
- 10 Trigger Levels
- No Probes!
- Can be used simultaneously with the Nios II IDE debugger and the FS2 console!



*Capture the state of internal nodes
In-system, at full system speeds*

SignalTap™ II Logic Analyzer

