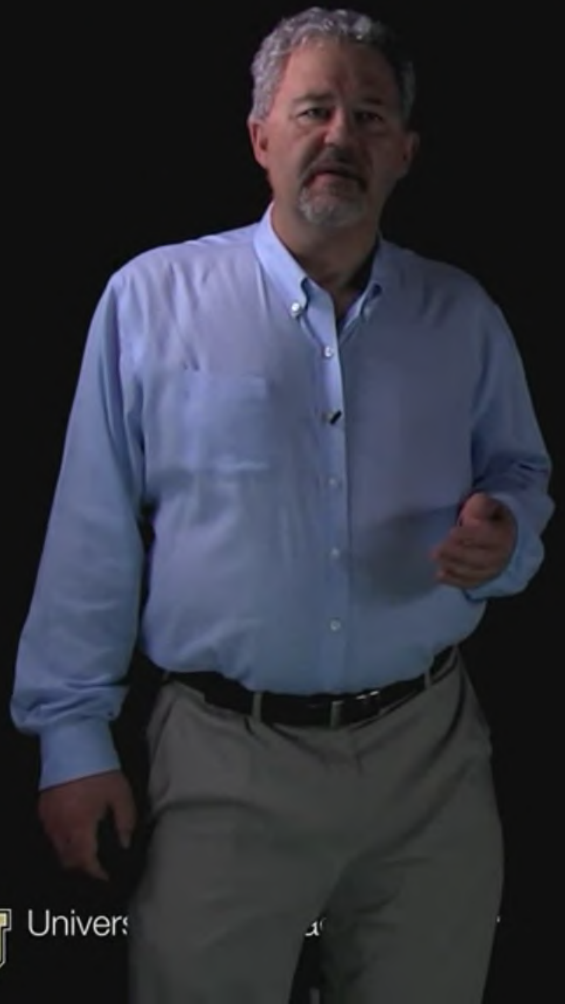
FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

FPGA Softcore Processors and IP Acquisition

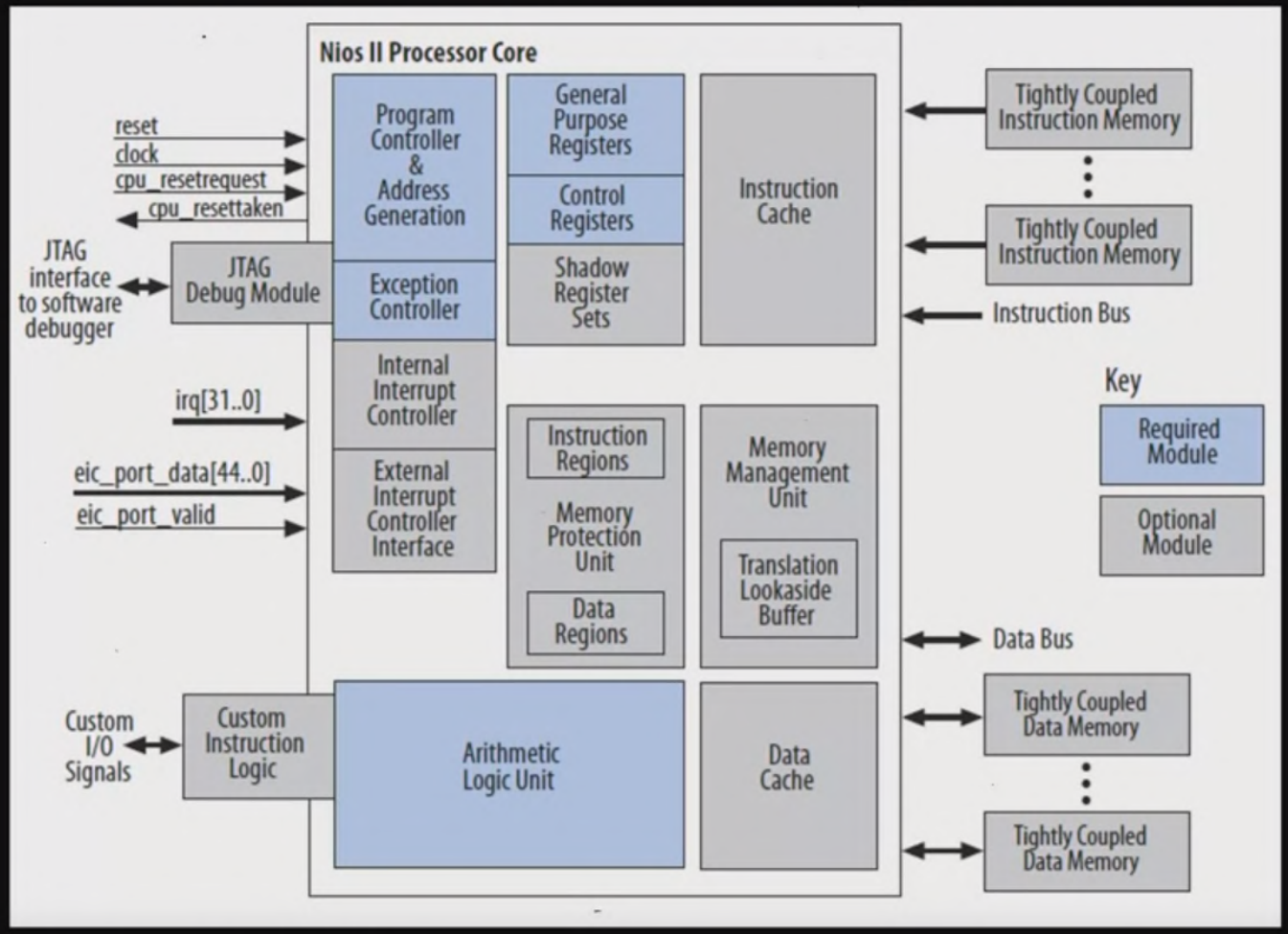


Why learn about Soft Processors and IP?

- Soft Processors have many unique advantages:
  - Flexibility : Hardware vs Software tradeoffs are easy to make
  - No Obsolescence: Infinitely migratable into future FPGA devices
  - Mature tools: Simulators, IDEs, Debuggers, and even Internal Logic Analyzers
  - Customization: Custom instructions and Custom Hardware can easily by added

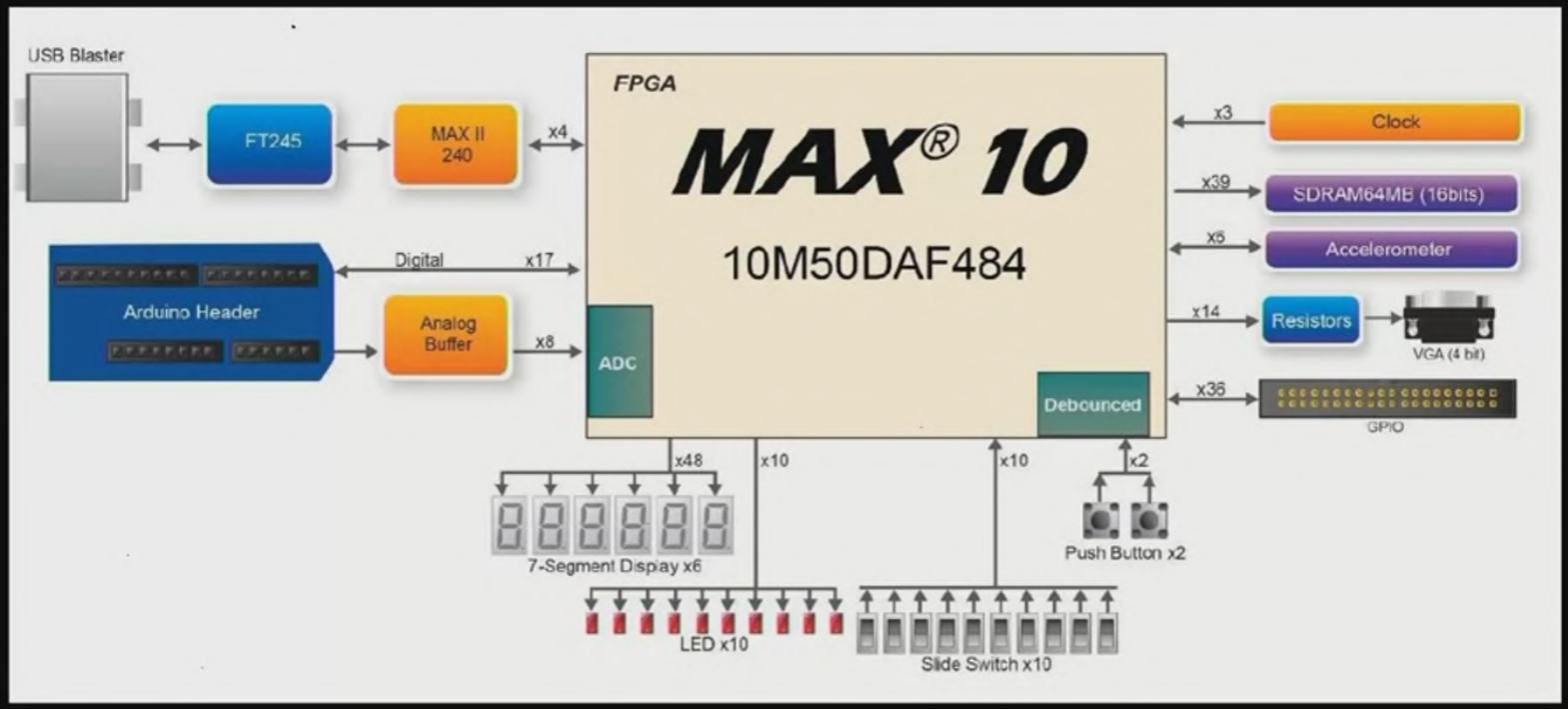
Feature	Altera Nios II	ARM Cortex-M1	Xilinx Microblaze	Xilinx Picoblaze	Microsemi Core ABC	Microsemi Core8051	Lattice Mico32	Lattice Mico8
Datapath	32 bits	32 bits	32 bits	8 bits	8/16/32	8 bits	32 bits	8 bits
Pipeline Stages	1-6	3	3-5	1	1	1	6	1
Frequency (MHz)	340	200	332	240	92	52	115	52.2
Gate Count	26k – 72k	2.6k-4k	30k – 60k	500 – 2k	420-4k	3k-5k	20k-25k	2k-3k
Register File	32 gen. + 32 special	16	32 gen. + 32 special	16	1	32	32	16 or 32
Instruction Word	32 bits	16/32	32 bits	18 bits	8/16	8/16	32	18
Instruction Cache	Optional	No	Optional	No	No	No	Option	No
Hardware Multiply	Optional	Yes	Optional	No	No	No	Yes	No
Hardware Floating Point	Optional	No	Optional	No	No	No	No	No
OS Support	eCOS, uC/OSII, Linux, uClinux	RTX	Linux	--	--	--	--	--

FPGA Softcore Processors and IP Acquisition



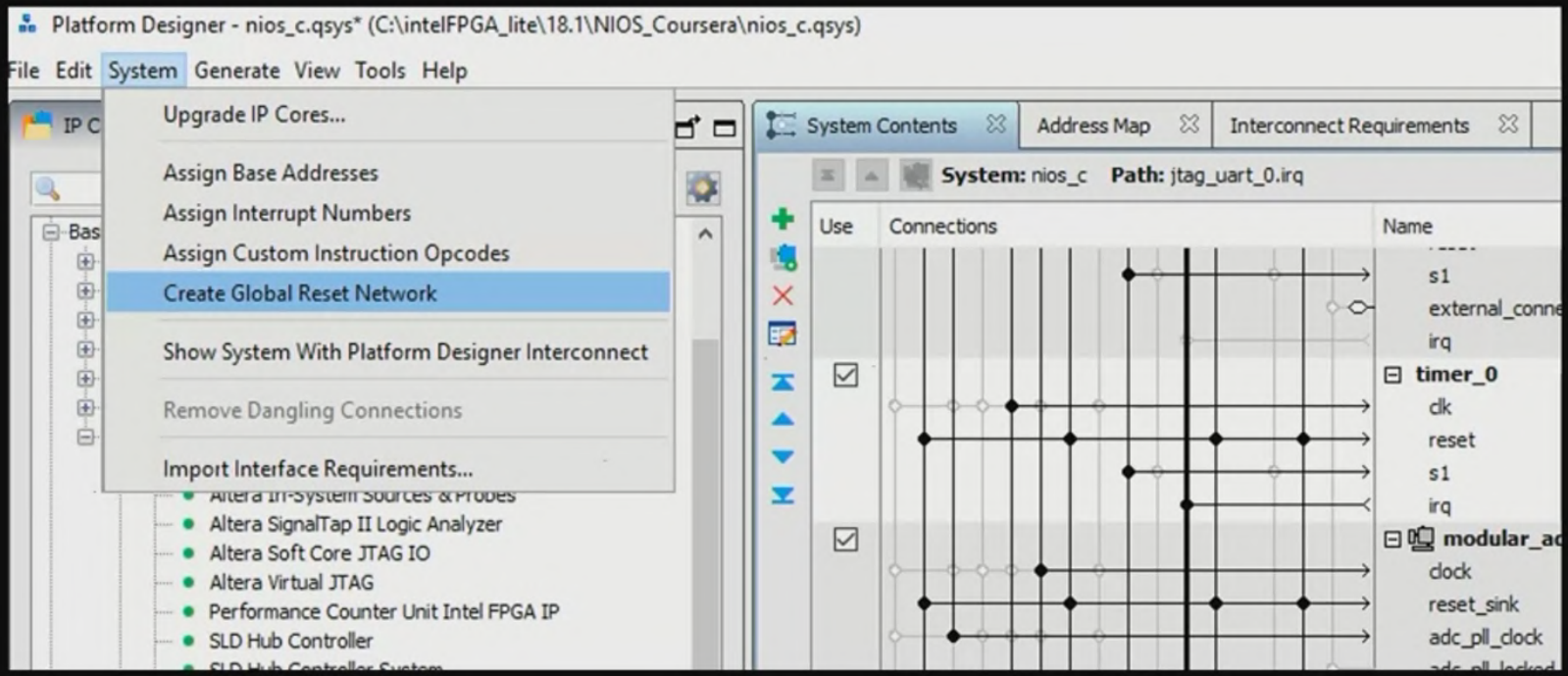
Interesting Architectures!

FPGA Softcore Processors and IP Acquisition



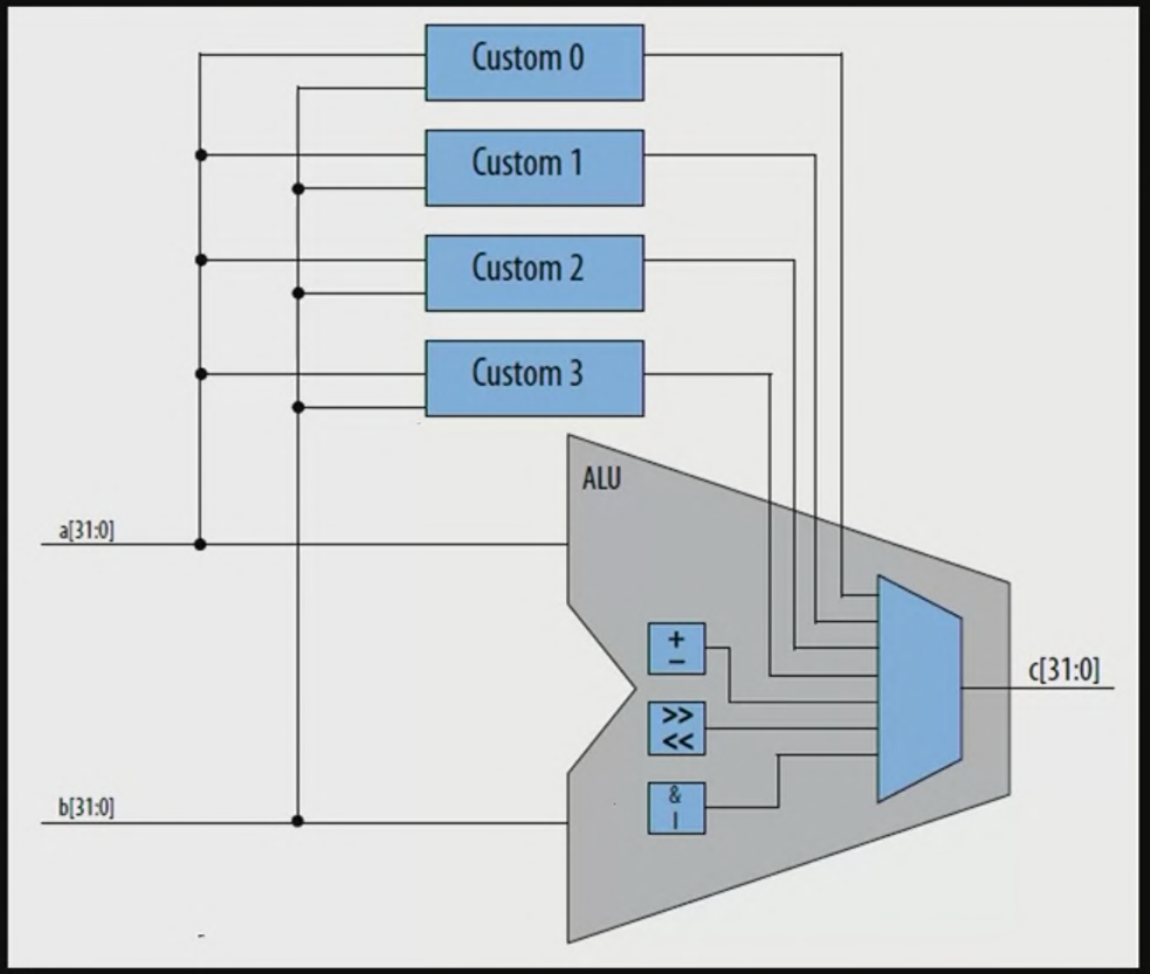
Fun Dev Tools!

FPGA Softcore Processors and IP Acquisition



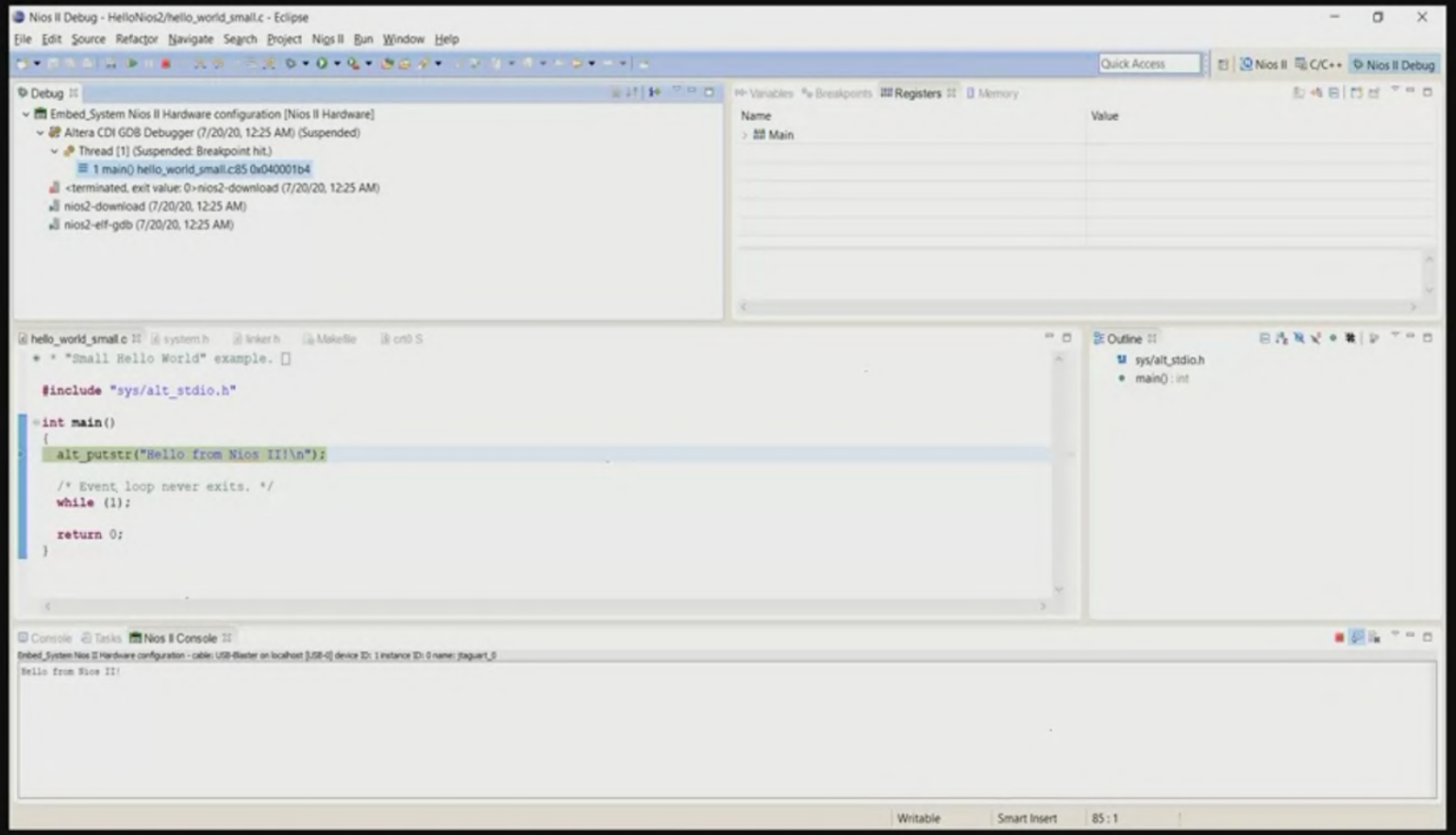
System Level Design!

FPGA Softcore Processors and IP Acquisition



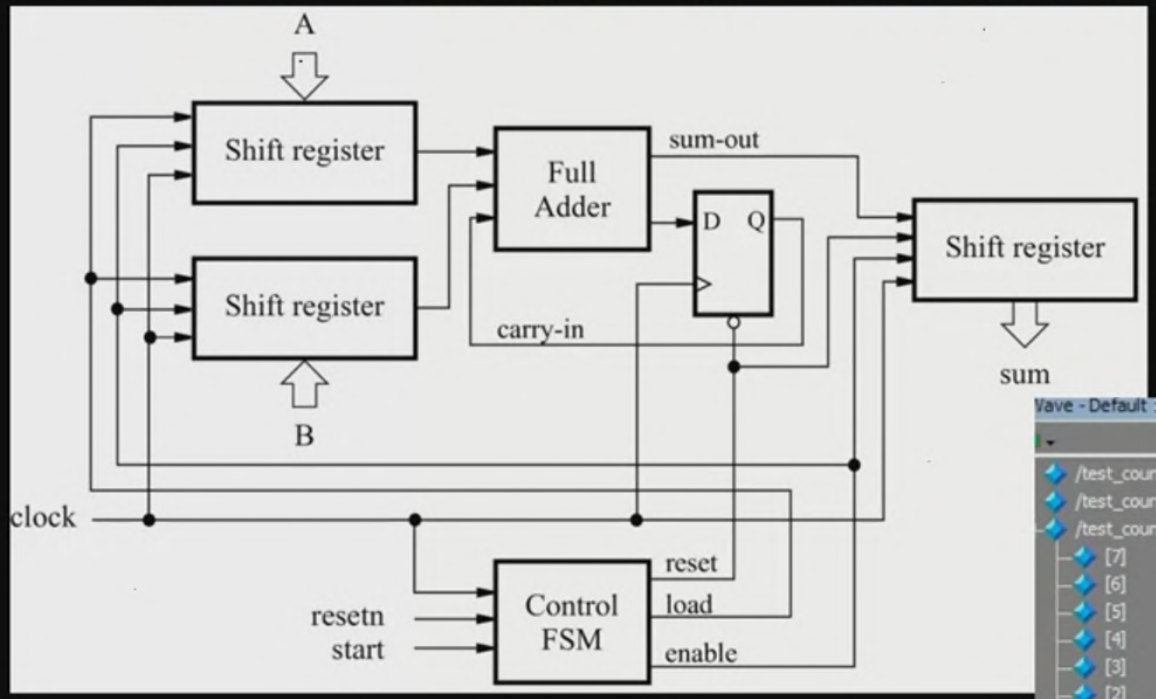
Customizing!

FPGA Softcore Processors and IP Acquisition

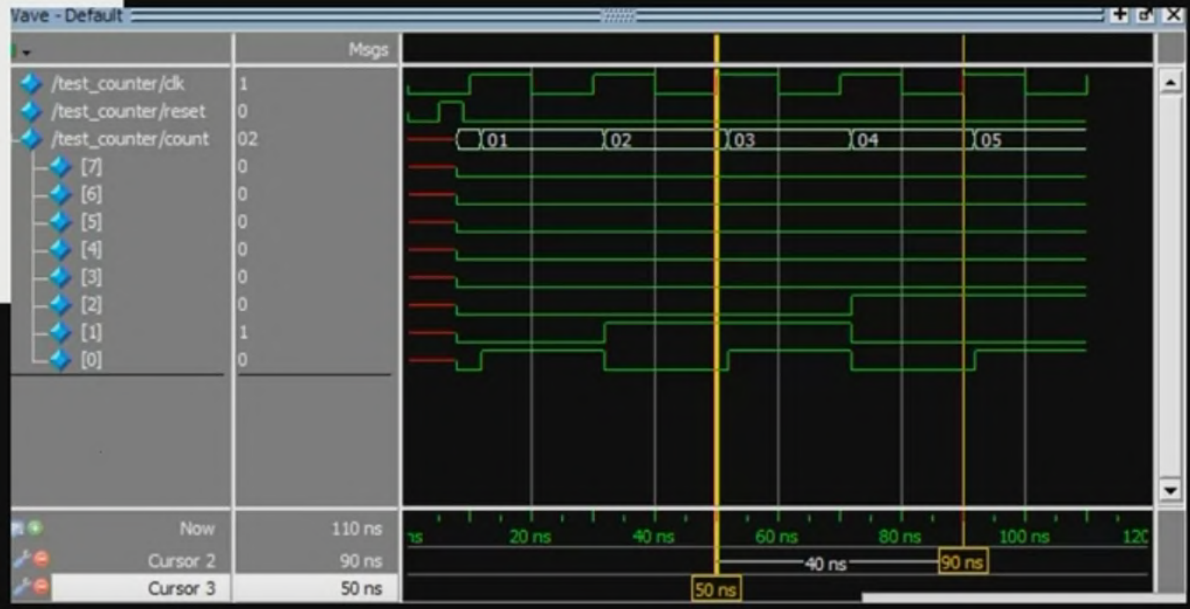


Coding in Eclipse!

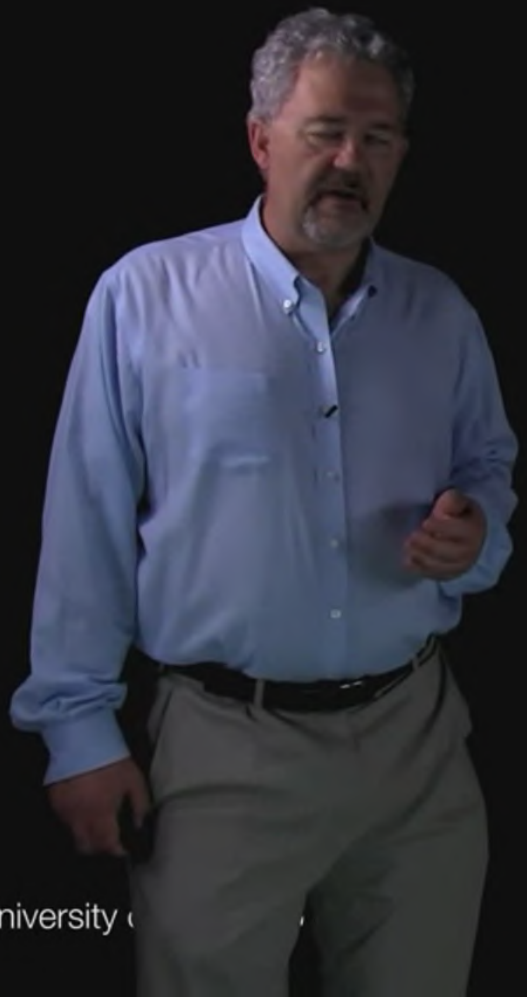
FPGA Softcore Processors and IP Acquisition



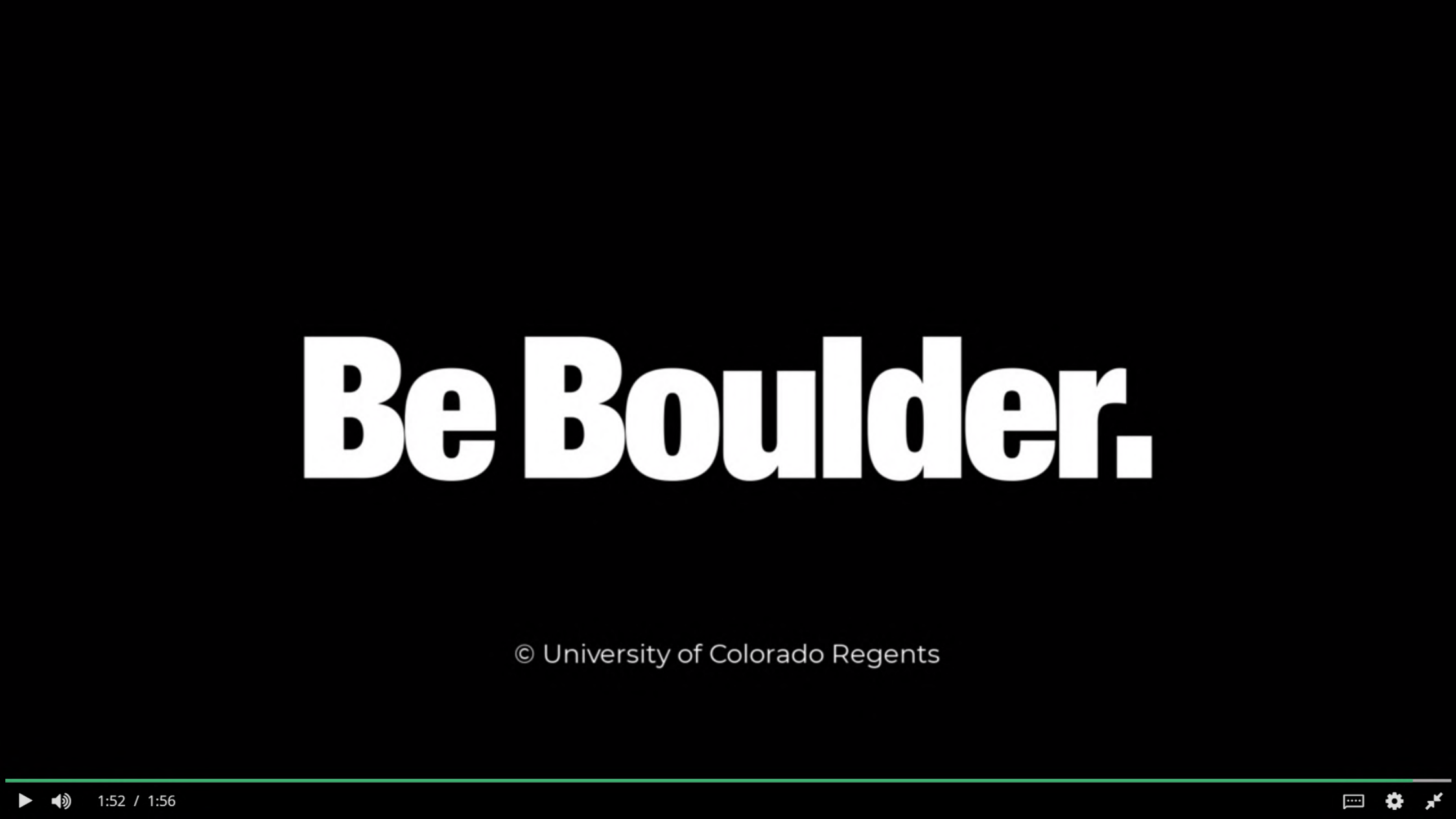
Lots of Examples



Modules in this Course



1. Make Your Own Processor – Softcore Processor Development Flow
2. Writing Software for Softcore Processors
3. IP Acquisition and Integration
4. Introducing ModelSim and Simulation for Verification


Be Boulder.
© University of Colorado Regents
1:52 / 1:56

University of Colorado **Boulder**

FPGA Design for Embedded Systems

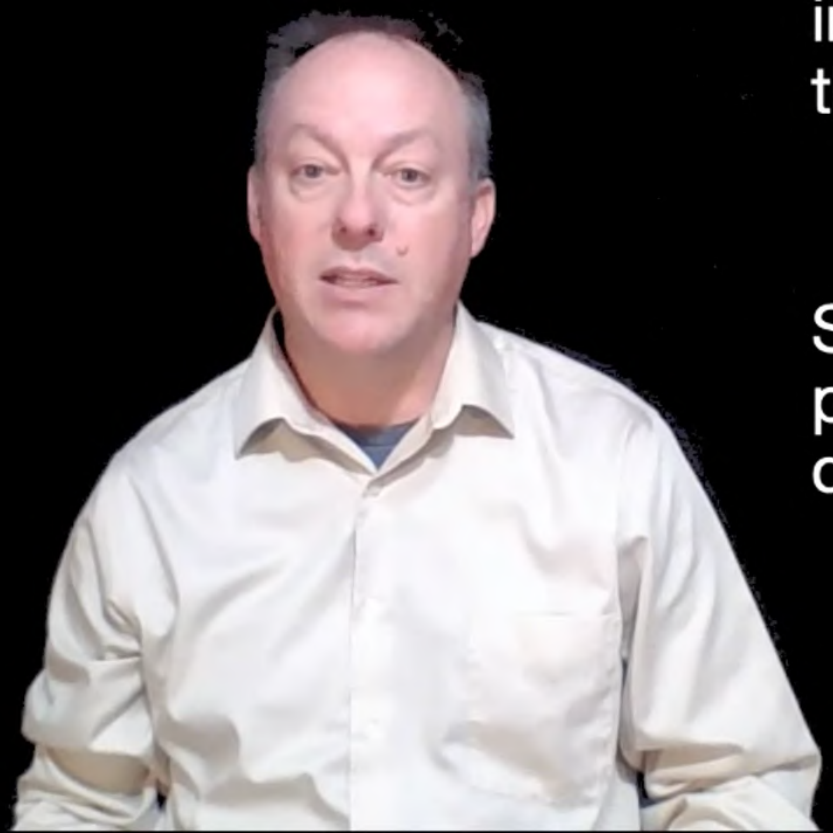
FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

Create a Soft Processor – Series Introduction

- Learn Architecture of a processor
- Build a processor system
- Customize the processor

Create a Soft Processor



FPGAs offer the flexibility to create and integrate a custom microprocessor into the hardware IP in the system.

- Peripherals, Memories, IO

Software development of soft processors is supported by a flow like discrete microcontroller designs.

- JTAG interface debugging, C compiler

Create a Soft Processor

Programmable Logic based processor design allows for design trade offs between:

- Discrete Fixed IP
- Software Control
- Custom FPGA Logic

Flexibility increases the design process complexity:

- Software offload verses Hardware effort

Videos in this module

1. Series Introduction
2. Soft Processor Advantages
3. Soft Processor Development Flows
4. Soft Processor Architectures, Part I
5. Soft Processor Architectures, Part II
6. Nios II Dev: Definitions
7. Nios II Dev: Connections (PD/Qsys Demo)
8. Nios II Dev: Compilation
9. Nios II Dev: Programming
10. Nios II Dev: Customization

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

Soft Processor Advantages

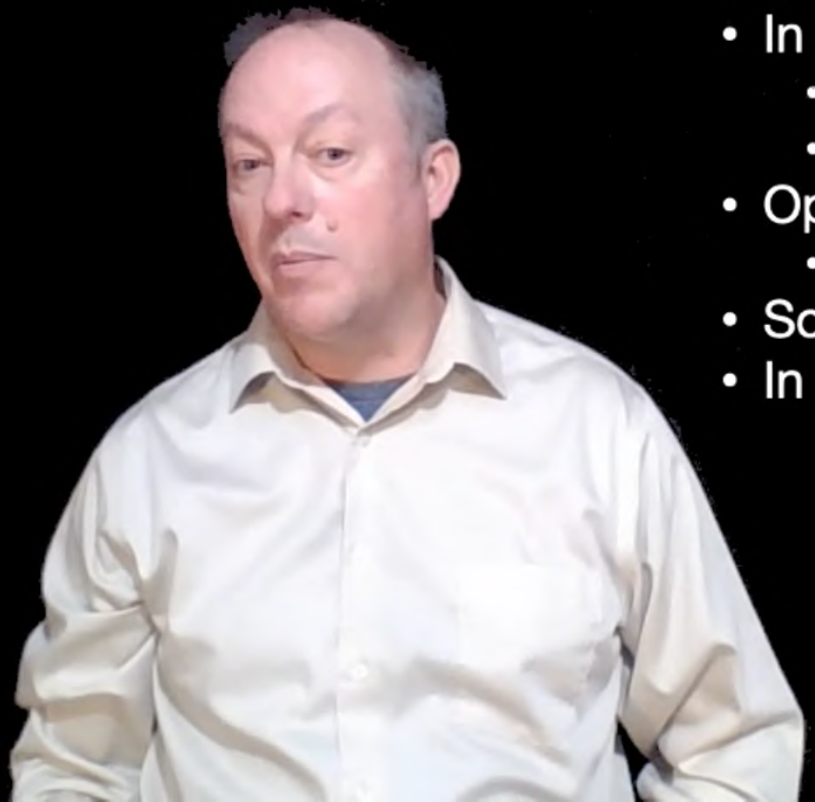
- Flexibility : Hardware vs Software
- Infinitely migratable into future devices
- Mature tools
  - Reference designs, flows, ecosystems

Soft Processor Advantages

- Flexibility : Hardware vs Software
  - Team : C programmers vs HDL programmers
    - Change management : Update C code or HDL
  - Software tasks : Dedicated FPGA Logic
  - Pre-Compile before selecting FPGA
    - Choose based on : Frequency, Size, Price, IO
  - Memory: cache size selectable
- Infinitely migratable into future FPGA devices
  - C code, processor configuration
  - Increased performance, lower power, lower cost
  - Move into larger or smaller device, or an ASIC*
  - Add another core for new features

*= License required

Soft Processor Advantages



- Mature tools and ecosystem
  - Reference designs and flows
  - In system JTAG code trace:
    - Multi (Green Hills), Trace32 (Lauterbach), XMTC (Xilinx)
    - Instruction and Data trace
  - Operating Systems and RTOS :
    - eCos, uClinux, FreeRTOS
  - Software : Simulink/Matlab (Mathworks)
  - In System Nodes : SignalTap, SmartDebug, ChipScope

Soft Processor Advantages

- Soft Processor type
  - (E) Economy
  - (F) Fast

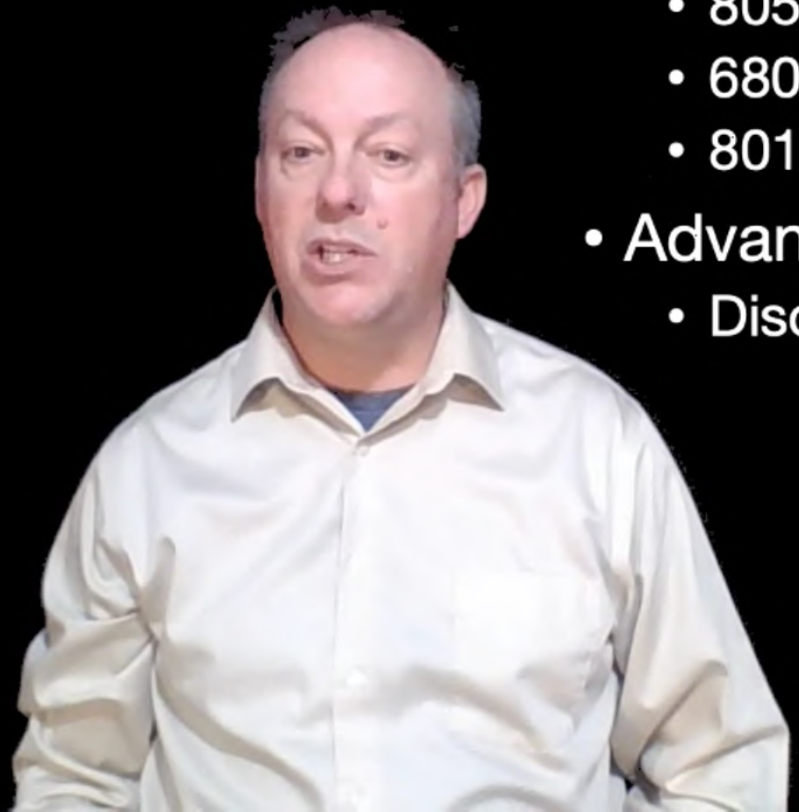
Soft Processor Advantages

- Soft Processor type*: (E) Economy, (F) Fast

	NIOS II (E)	NIOS II (F)
Pipeline	1 Stage	6 Stages
Branch Prediction	None	Dynamic
HW Multiplication	Software	1 Cycle HW
Inst/Data cache	None	0.5 - 64 Kbyte
MMU	None	Available
Area	540 LEs	1600 LEs
Freq	195 MHz	140 MHz
Perf	18 MIPS	145 MIPS

*Intel/Altera Cyclone II Device Comparison

Soft Processor Advantages



- Variety of 3rd party cores
  - 8051 (Cast)
  - 68000 (Digital Core Design)
  - 80186 (iWave)
- Advantage is software code compatible
  - Discrete device no longer available

Soft Processor Advantages

- PicoBlaze 8bit (Xilinx, very small area and code)

Soft Processor Advantages

- Dedicated Bus Structure
  - Intel-Altera NIOS : Avalon
  - Xilinx MicroBlaze : AXI
  - Microchip-Microsemi ARM Cortex-M1 : AHB

Soft Processor Advantages

- Dedicated and Custom Peripherals addressable through the bus fabric

• Bus Structure: Xilinx MicroBlaze : AXI

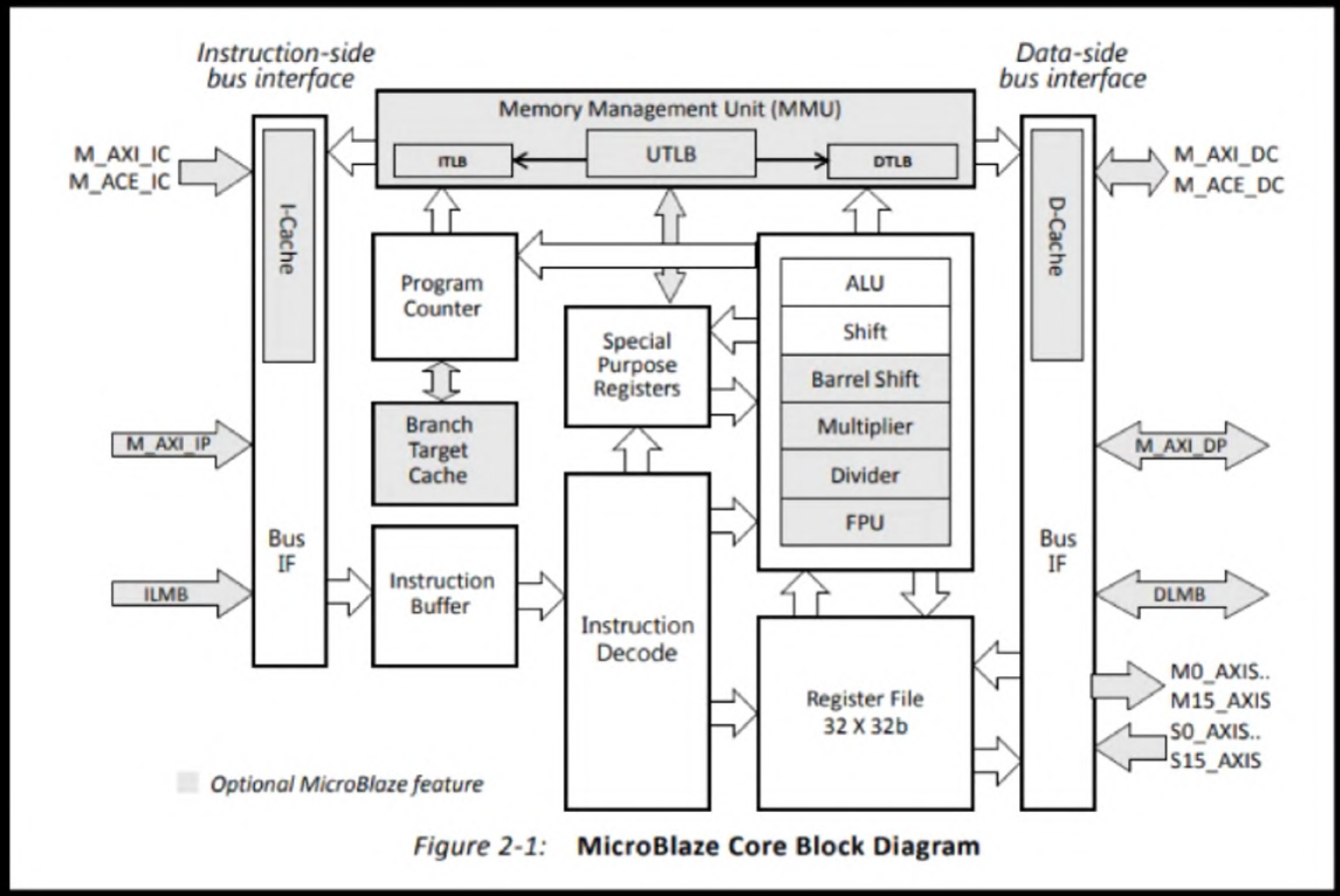
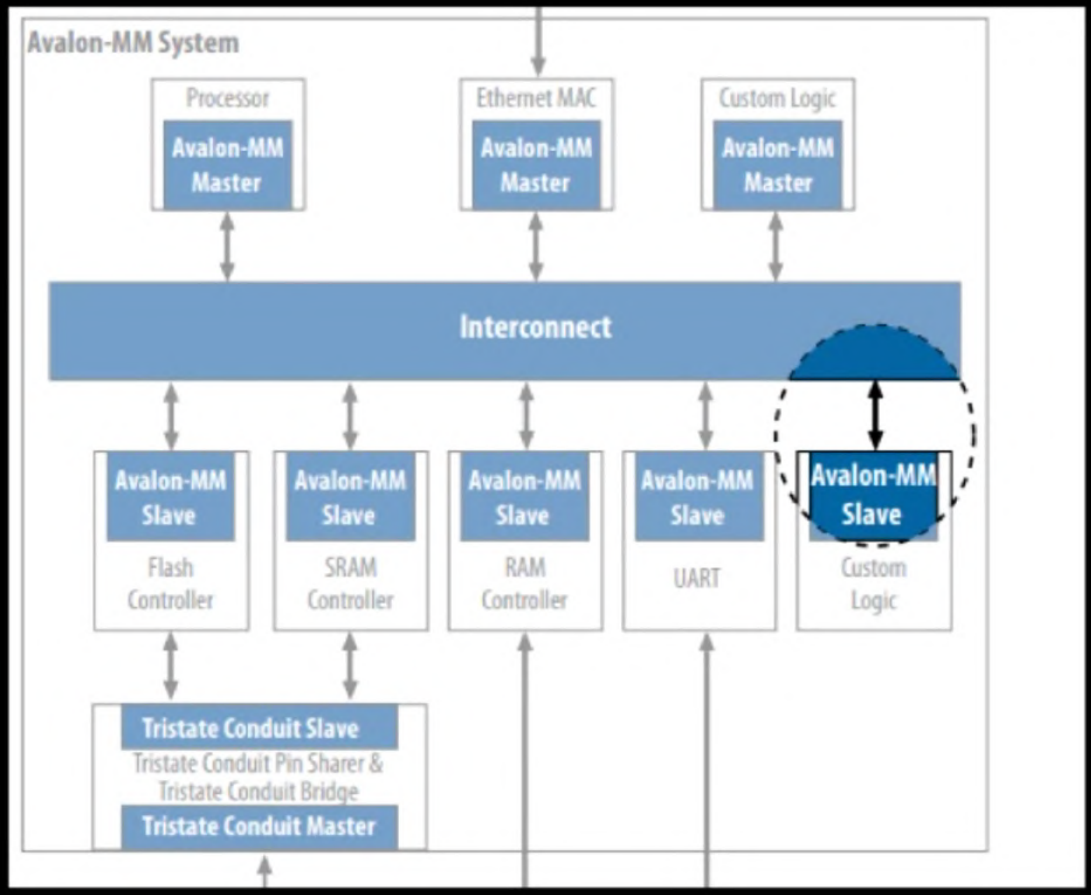


Figure 2-1: **MicroBlaze Core Block Diagram**

• Bus Structure: Intel-Altera NIOS : Avalon



Soft Processor Advantages

- Flexibility : Hardware vs Software
- Infinitely migratable into future devices
- Mature tools

Create a Soft Processor

FPGA References

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

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

https://www.xilinx.com/products/intellectual-property/xmtc.html

https://www.xilinx.com/products/intellectual-property/nav-embedded/nav-embedded-processors.html

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug984-vivado-microblaze-ref.pdf

"Embedded SoPC Design with NIOS II Processor and Verilog Examples", Pong P. Chu, pg 231 Table 9.1 NIOS II Comparison

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug984-vivado-microblaze-ref.pdf

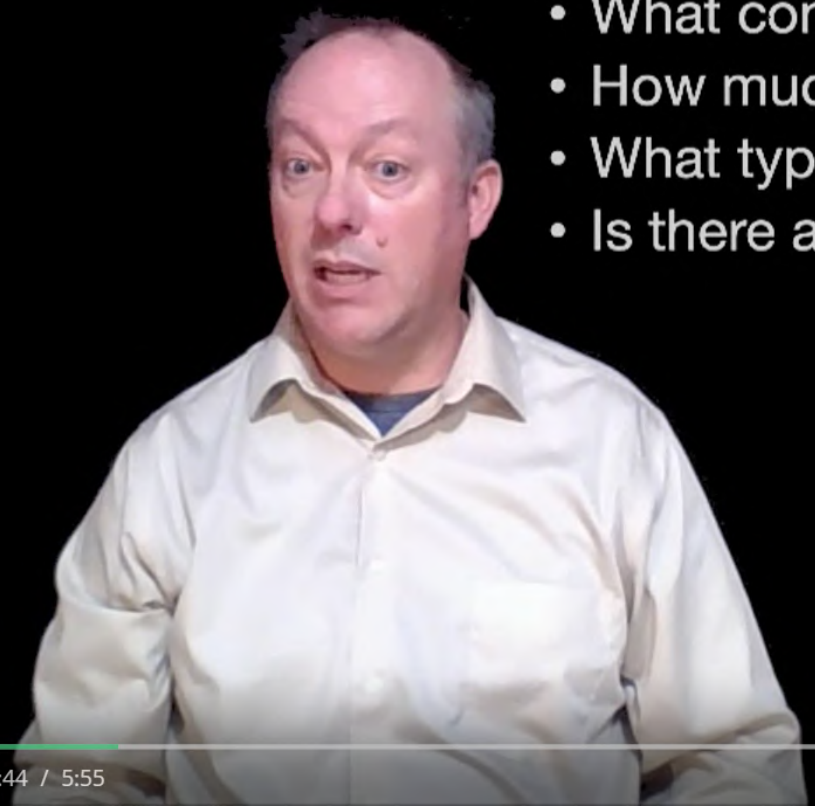
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

Soft Processor Flows

- NIOS II System Definition for Component Selection
- Questions for System Design:
  - What computational requirements does the design have?
  - How much bandwidth is needed? DDR3, DDR4?
  - What types of interfaces and how many IO are needed?
  - Is there a multi-threaded application or Operating System?

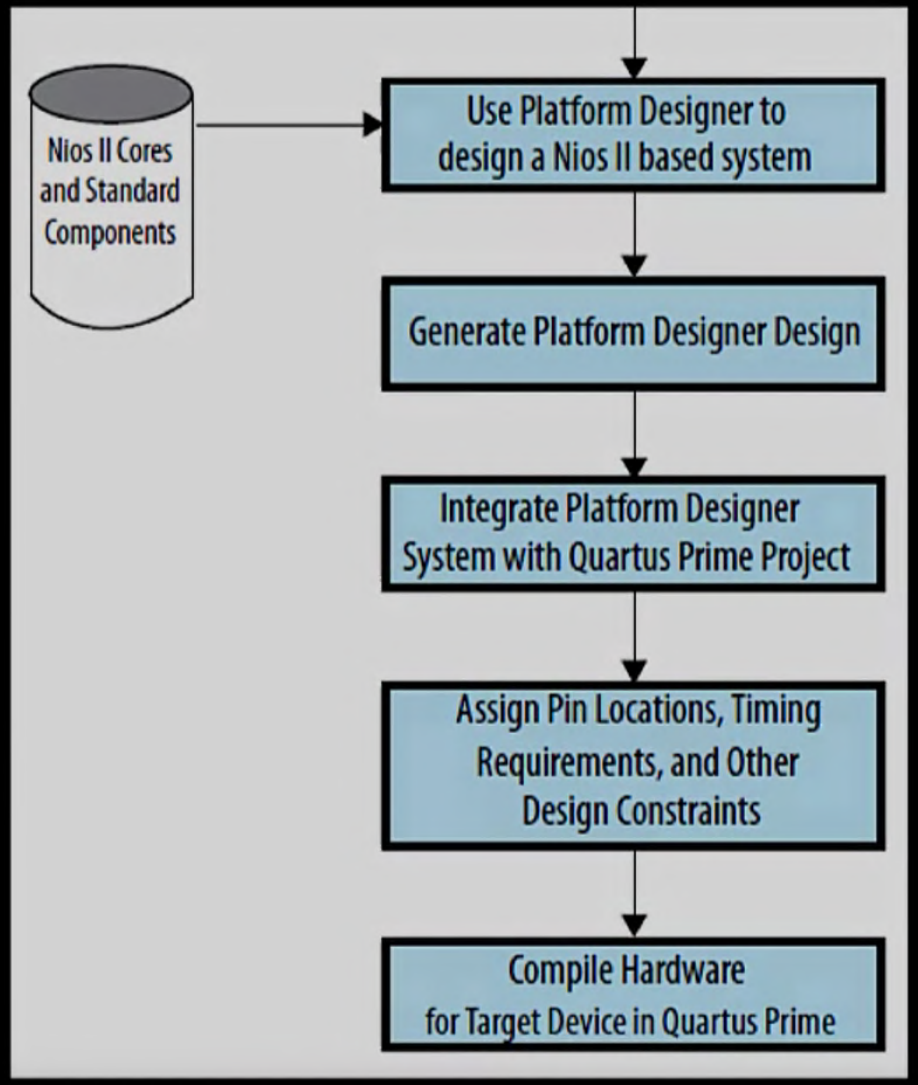


Soft Processor Flows

- Based on these questions, one can determine:
  - Faster or smaller processor? Interfaces?
  - Hardware acceleration or off-loading?
    - DMA needed to eliminate processor cycles copying data?
    - Could a custom instruction remove a loop from a DSP algorithm?

Soft Processor Flows

- NIOS II Platform Designer Flow, formerly called Qsys

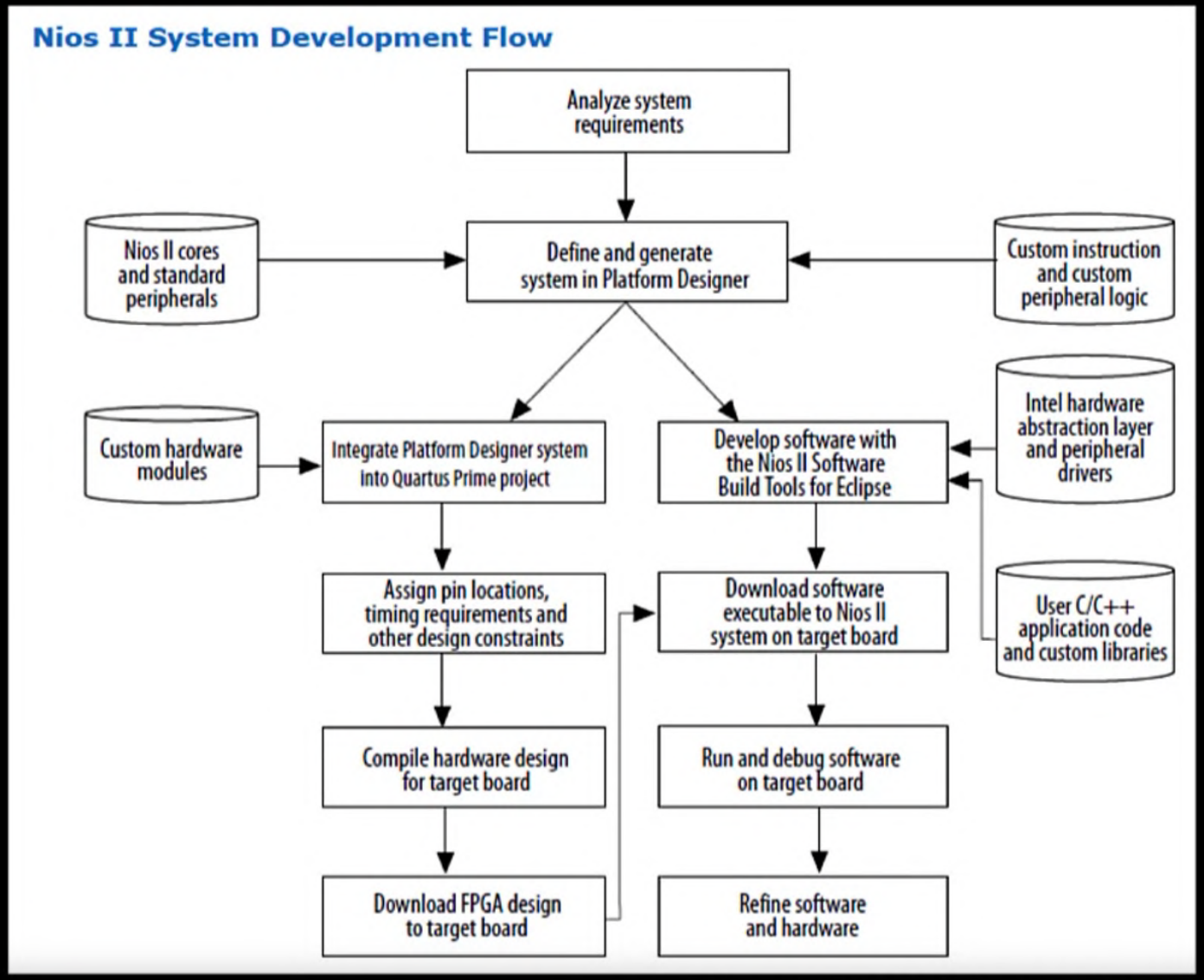


Soft Processor Flows

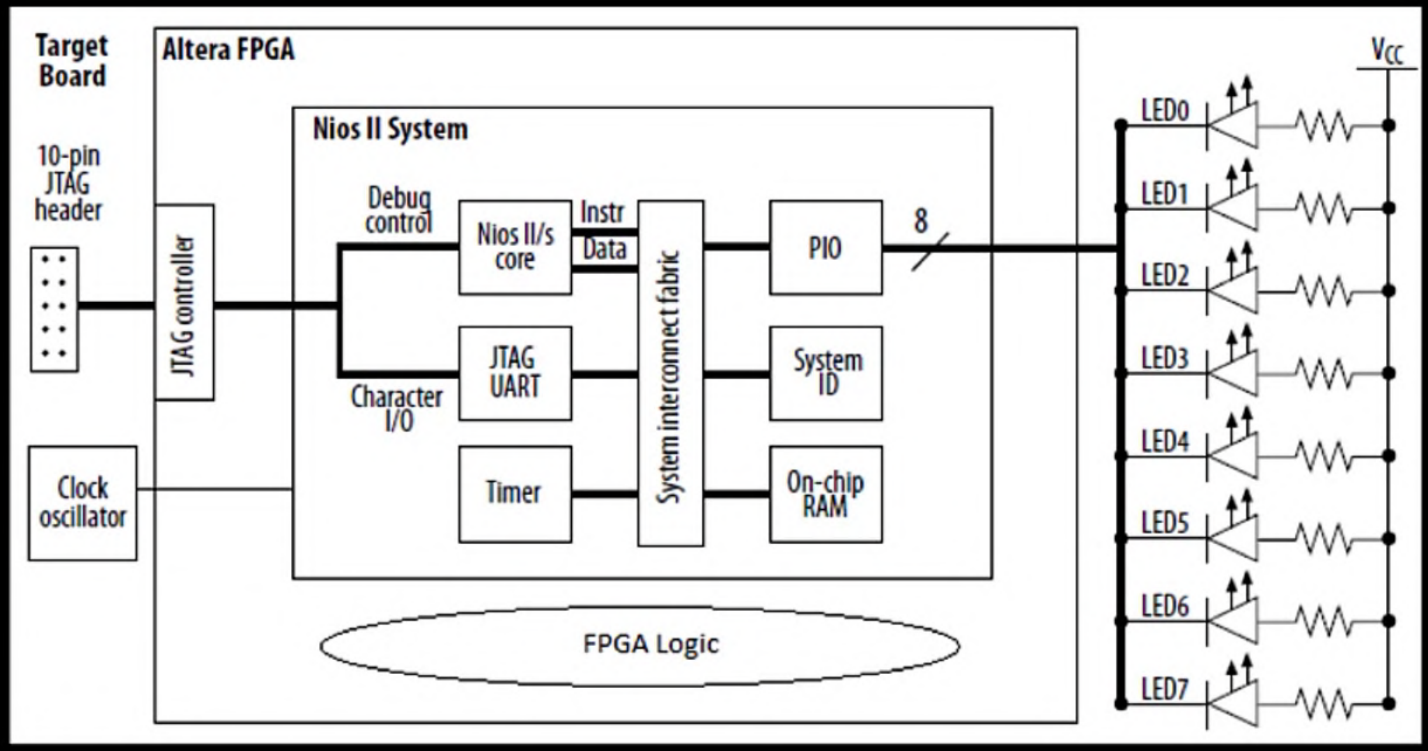
- NIOS II Platform Designer Flow, formerly called Qsys
- Platform Designer (Qsys) is a system development tool for creating processors, peripherals, and memories.
- Once compiled, the logic cells and flip-flops inside the FPGA are loaded with the processor configuration.
- The design is volatile and is re-programmed at power ON.

- NIOS II Platform Designer Detailed Flow

- hardware design steps
- software design steps
- system design steps, involving both hardware and software



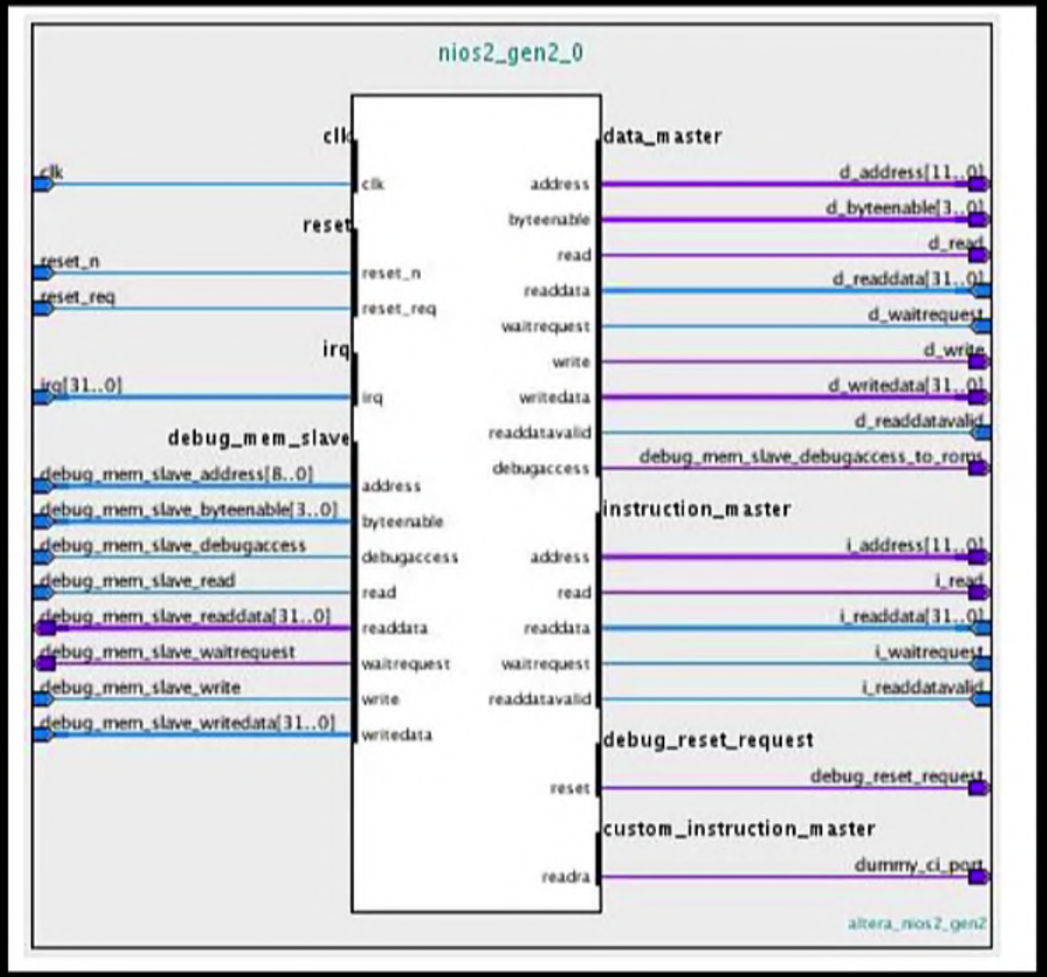
- Board
  - JTAG
  - Clocks
  - IO
  - Power
- FPGA
  - Nios Core
  - Logic



- NIOS II Block Level Design

Soft Processor Flows

- Signals in Platform Designer
- Clock and reset pins are defined in Platform Designer.
- Each IO that is required for the NIOS system is added.
- Constraints, like clock frequency and reset pin definitions are added.



Soft Processor Flows

- Software Development in Eclipse
- After generating and compiling the system, the C/C++ code development can begin with the Nios II Software Build Tools for Eclipse.

- Eclipse uses information from the .sopcinfo file
- system.h defines symbols in the hardware
- Executable and Linking Format File (.elf) result of compiling a C/C++
- Hexadecimal (Intel-Format) File (.hex) initialization information for on-chip memories
- Flash memory programming data, boot code

Soft Processor Flows

- System Configuration definition is needed for Processor Hardware development
- Once the Hardware is defined, the Software will depend on the Hardware definition
- When the Hardware definition is loaded into the Software C/C++ coding can begin

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

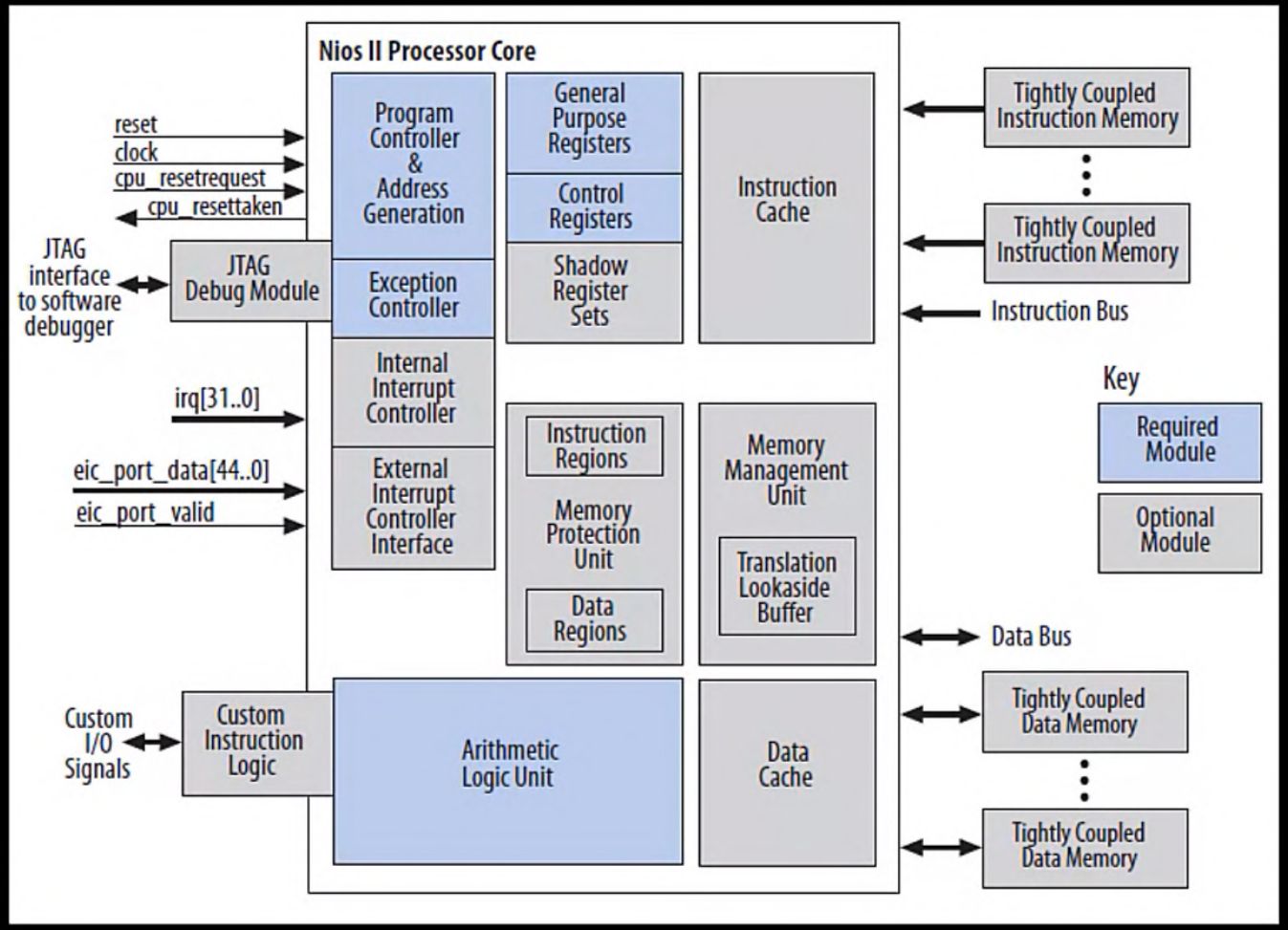
Overview

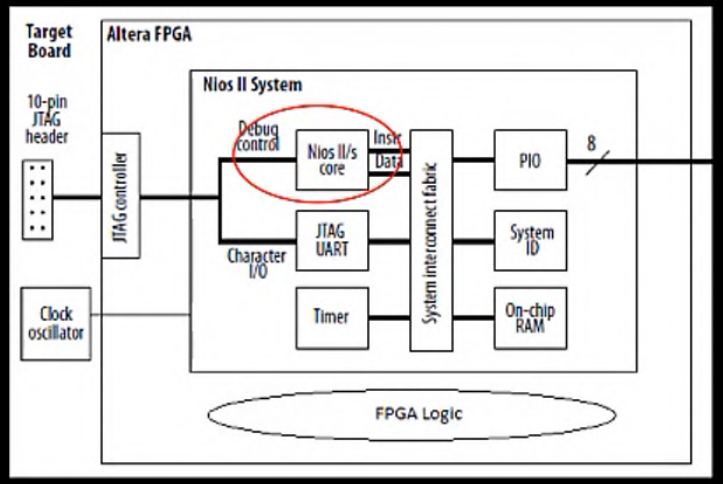
- Review of soft processors on the market.
- Describe the architecture of the NIOS II soft core processor.
- Specific components of the core are described in detail.

Feature	Altera Nios II	ARM Cortex-M1	Xilinx Microblaze	Xilinx Picoblaze	Microsemi Core ABC	Microsemi Core8051	Lattice Mico32	Lattice Mico8
Datapath	32 bits	32 bits	32 bits	8 bits	8/16/32	8 bits	32 bits	8 bits
Pipeline Stages	1-6	3	3-5	1	1	1	6	1
Frequency (MHz)	340	200	332	240	92	52	115	52.2
Gate Count	26k – 72k	2.6k-4k	30k – 60k	500 – 2k	420-4k	3k-5k	20k-25k	2k-3k
Register File	32 gen. + 32 special	16	32 gen. + 32 special	16	1	32	32	16 or 32
Instruction Word	32 bits	16/32	32 bits	18 bits	8/16	8/16	32	18
Instruction Cache	Optional	No	Optional	No	No	No	Option	No
Hardware Multiply	Optional	Yes	Optional	No	No	No	Yes	No
Hardware Floating Point	Optional	No	Optional	No	No	No	No	No
OS Support	eCOS, uC/OSII, Linux, uClinux	RTX	Linux	--	--	--	--	--

Soft Processor Architecture : Part I

- NIOS II Processor Core





Soft Processor Architecture

- NIOS II Core

- A FPGA based processor system is equivalent to a microcontroller or “computer on a chip” that includes a processor and a combination of peripherals and memory on a single chip.
- A processor system includes:
  - processor core
  - on-chip peripherals and on-chip memory
  - interfaces to off-chip memory
- All processor systems use a consistent instruction set and programming model.

Soft Processor Architecture

- NIOS II Core Features
  - 32 bit RISC – Reduced Instruction Set Core
  - Full 32-bit instruction set, data path, and address space
  - Quantity 32 general-purpose registers
  - Quantity 32 interrupt sources
  - External interrupt controller for more interrupt sources
  - Single-instruction 32 × 32 multiply and divide producing a 32-bit result
  - Dedicated instructions for computing 64-bit and 128-bit products of multiplication
  - Optional floating-point instructions for single-precision floating-point operations
  - Single-instruction barrel shifter

Soft Processor Architecture

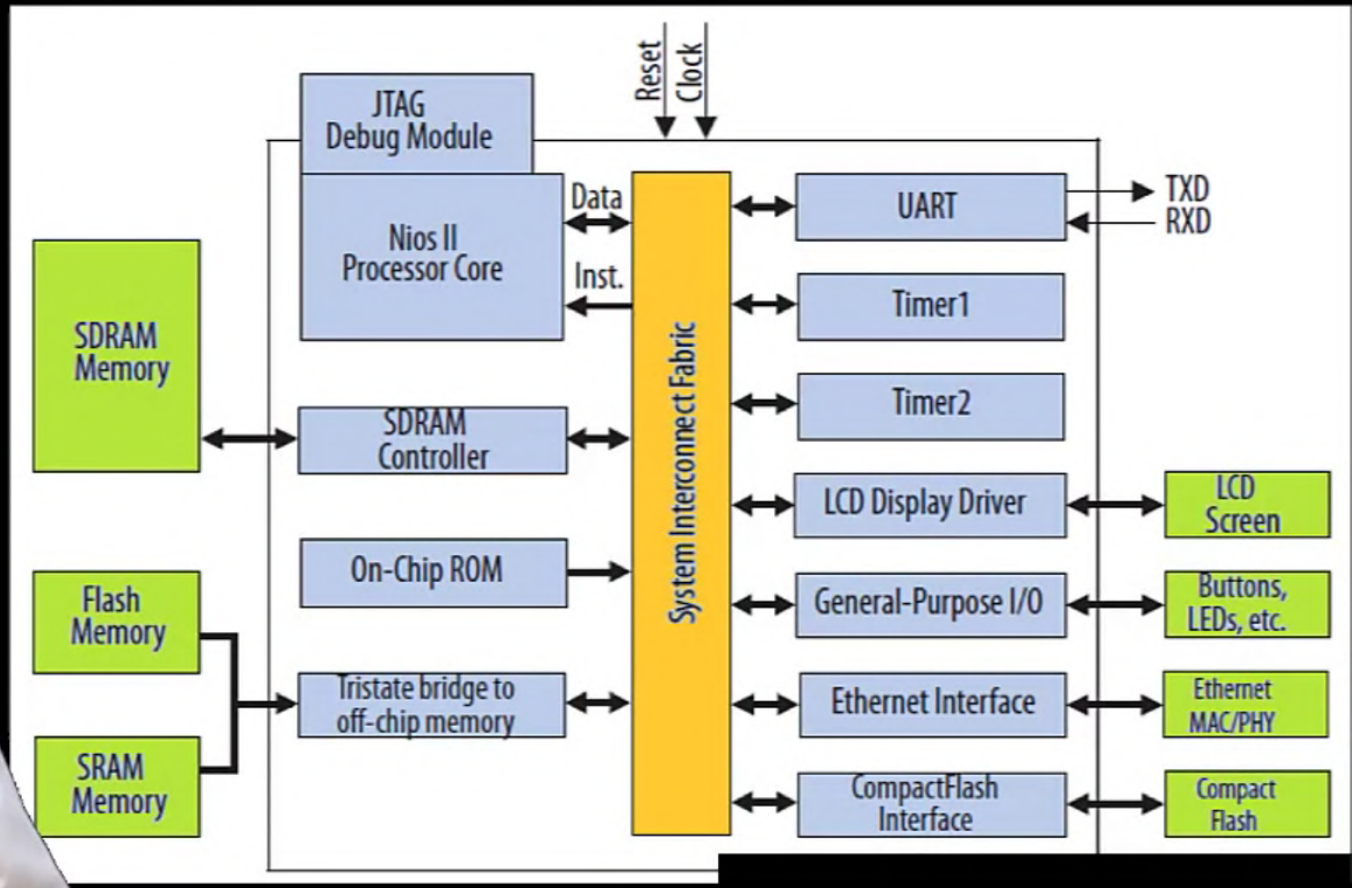
- NIOS II Core Features
  - Access to on-chip peripherals, and interfaces to off-chip memories and peripherals
  - Hardware-assisted debug module enabling processor start, stop, step, and trace
  - Optional memory management unit (MMU) to support operating systems that require MMUs
  - Optional memory protection unit (MPU)

Soft Processor Architecture

- NIOS II Core Features
  - Software development environment based on the GNU C/C++ tool chain
  - Integration with FPGA's Signal Tap Embedded Logic Analyzer, enabling real-time analysis of instructions and data along with other signals in the FPGA design
  - Instruction set architecture (ISA) compatible across all processor systems
  - Performance up to 250 DMIPS
  - Optional error correcting code (ECC) support for a subset of processor internal RAM blocks

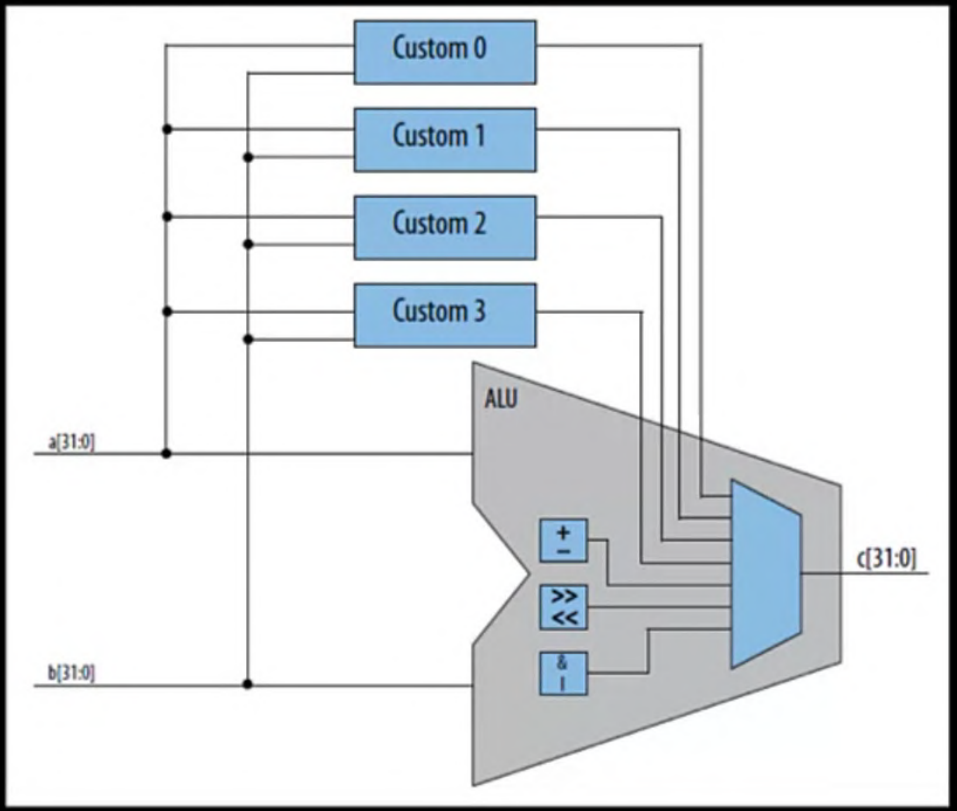
Soft Processor Architecture

- NIOS II Processor System
  - Internal
  - External
  - Bus Fabric



Soft Processor Architecture

- NIOS II ALU
  - Standard
    - Arithmetic
    - Relational <, >, !=
    - Logical AND, OR
    - Shift
  - Customizable
    - DSP
    - FIR



Soft Processor Architecture

- NIOS II ALU
- Hardware implementation or software computation?
  - Applications that rarely perform complex arithmetic, the division instruction can be computed by many cycles in software
  - Removing the divide hardware conserves on-chip resources but increases the execution cycles of division operations.

Soft Processor Architecture

- NIOS II Core : Register File
  - Quantity thirty-two 32-bit GP integer registers
  - Quantity thirty-two 32-bit control registers

  - Optional : one or more shadow register sets. A shadow register set is a complete set of Nios II general-purpose registers
  - Typical use of shadow register sets is to accelerate context switching
  - Shadow register sets are typically manipulated by an operating system kernel, and are transparent to application code

Soft Processor Architecture

- NIOS II Core : Reset control

reset: global hardware reset signal that forces the processor core to reset immediately

cpu_resetrequest: optional, local reset signal that causes the processor to reset without affecting other components in the system

debug_reset_request: when the JTAG Debug module is enabled, allows the JTAG debuggerto reset the processor

debugreq: optional signal that temporarily suspends the processor for debugging purposes

reset_req: optional signal prevents the memory corruption by performing a reset handshake before the processor resets

Soft Processor Architecture

- NIOS II Core : Exception and Interrupt
  - External interrupt controller (EIC) interface enables a speed up of interrupt handling in a complex system by adding a custom interrupt controller, software configurable
  - EC : a non-vectored exception controller handles all exception types and each exception causes the processor to transfer execution to an exception address
  - IC : 32 level-sensitive interrupt request (IRQ) inputs, irq0 through irq31, providing a unique input for each interrupt source

Architecture : Summary

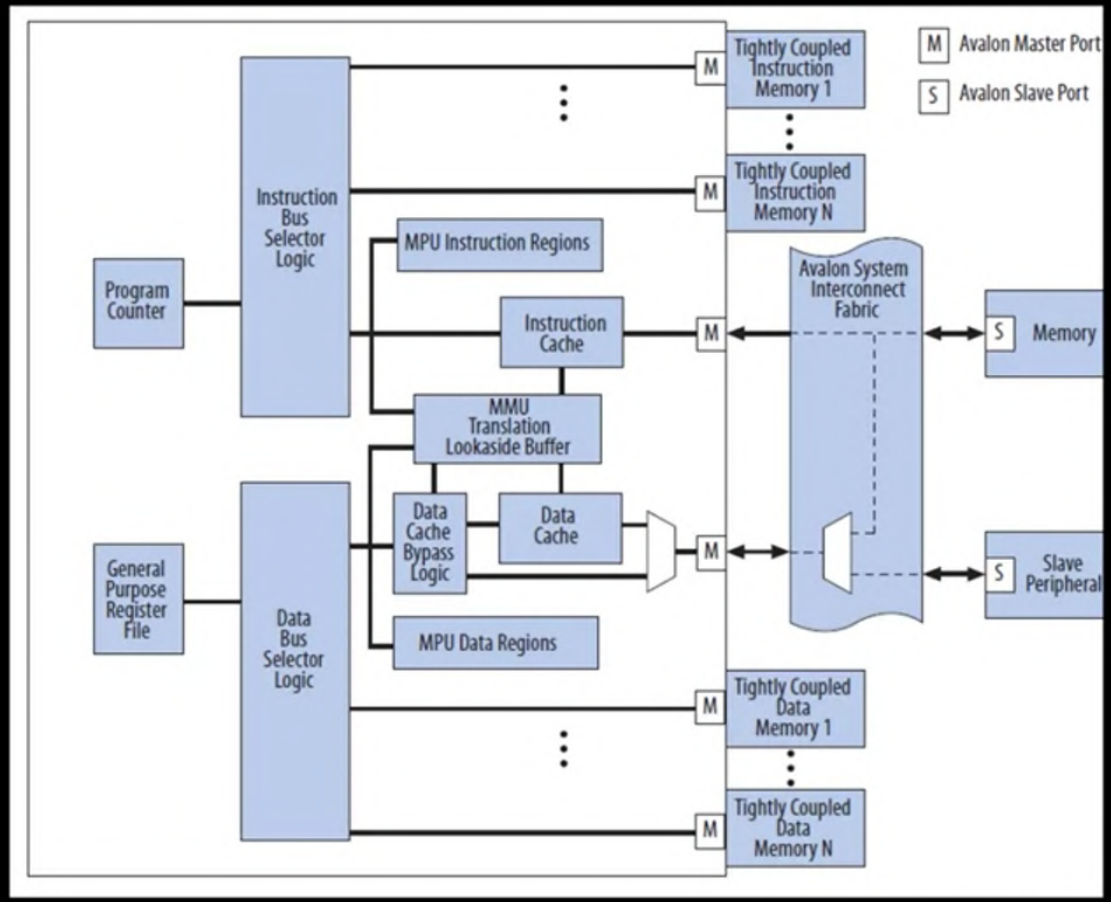
- Definition of an FPGA Processor System
- NIOS Core features

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

Soft Processor Architecture : Part II

- NIOS II Memory and IO
  - Instruction
  - Data
  - Memory
    - Tightly Coupled
    - External
  - IO
    - External



Soft Processor Architecture

- NIOS II Core : Memory and IO
  - Memory Mapped :
    - Instruction Memory Mapped master port that connects to instruction memory by system interconnect fabric
    - Instruction cache: Fast cache memory internal to the Nios II core
    - Data master port: A mapped 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 to the
    - Nios II core
  - IO organization varies per FPGA device

Soft Processor Architecture

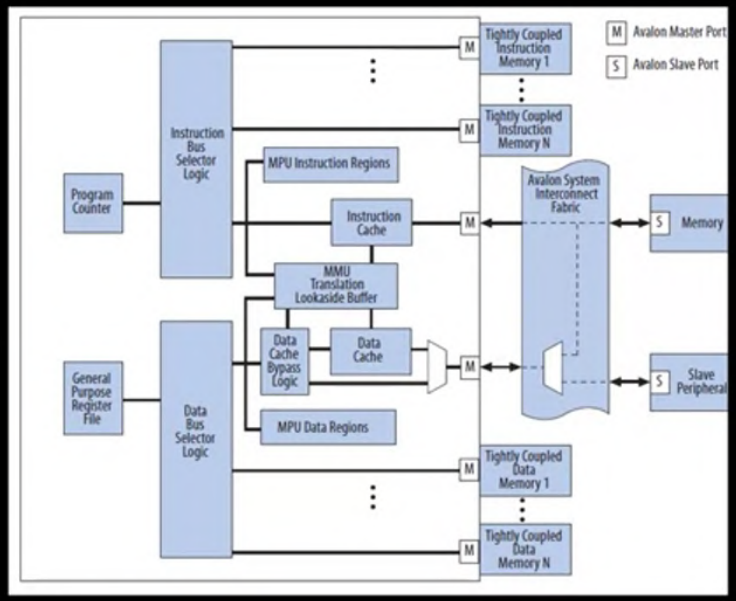
- NIOS II Core : Memory and IO
  - Memory Mapping :
    - The architecture supports separate instruction and data buses, classifying it as a Harvard architecture.
    - Both the instruction and data buses are implemented as master ports.
    - The data master port connects components
      - memory
      - peripheral components
    - Instruction master port connects only
      - memory
    - The instruction and data masters have a combined address map and instructions and data are in the same address space

Soft Processor Architecture

- NIOS II Core : Instruction and Data
  - Instruction Port:
    - Fetches instructions to be executed by the processor. The instruction master port is pipelined, and does not perform any write operations
  - Data Port:
    - Reads data from memory or a peripheral when the processor executes a load instruction
    - Writes data to memory or a peripheral when the processor executes a store instruction
  - Cache and TCMs supported by both I and D

Soft Processor Architecture

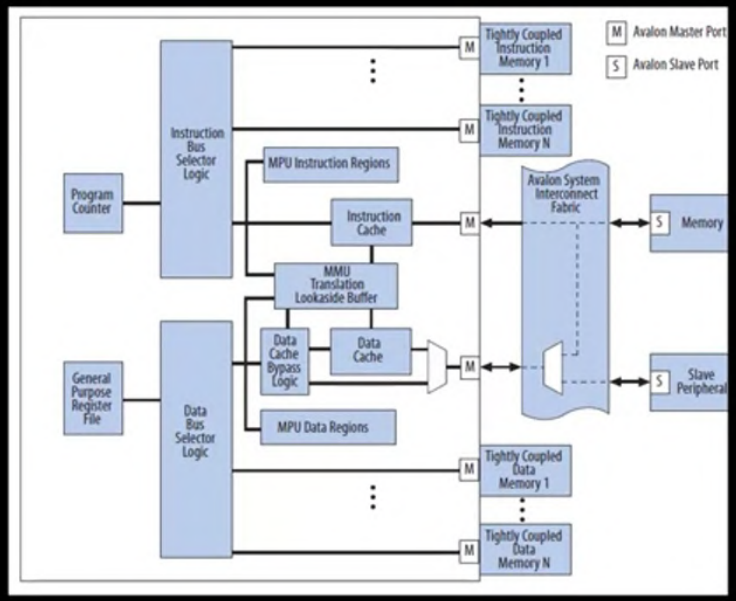
- NIOS II Core : Cache Memory
  - Instruction (optional)
  - Data (optional)
  - Cache memories can improve the average memory access time for processor systems that use slow off-chip memory such as SDRAM for program and data storage
  - Caches are enabled perpetually at run-time, but methods are provided for software to bypass
  - Cache management and cache coherency are handled by software



**Note: If the critical loop of a program is 2 KB, but the size of the instruction cache is 1 KB, an instruction cache does not improve execution speed. In fact, an instruction cache may degrade performance in this situation. Usage of Cache is Application Dependent.

Soft Processor Architecture

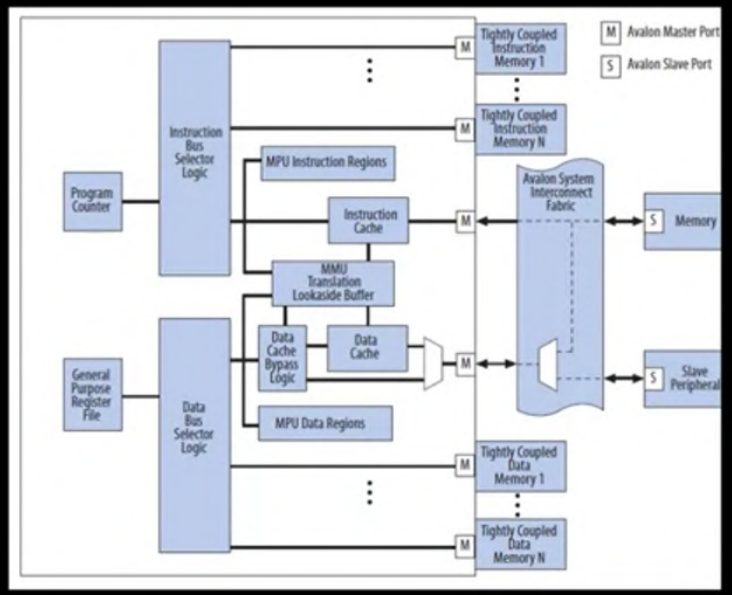
- NIOS II Core : TCM (on-chip, external to core)
- Tightly Coupled Memory
  - Instruction (optional)
  - Data (optional)
  - TCM ports are a separate master port on the processor core.
  - Performance critical code or data is located in a TCM.
  - Compute-intensive digital signal processing (DSP) applications can place data buffers into TCMs for the fastest possible data access
  - Normal address space, the same as other memory addressed devices connected via system interconnect fabric.



Note: The TCM master requires a fixed memory latency of 1 cycle

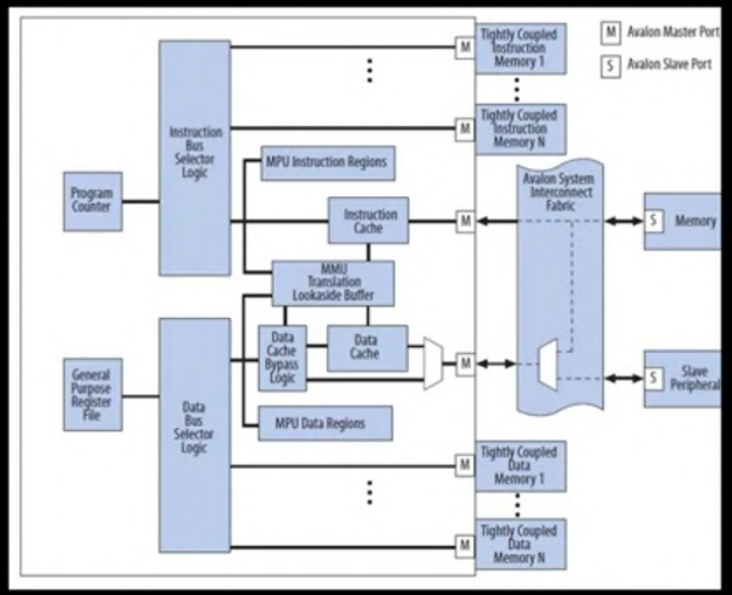
Soft Processor Architecture

- NIOS II Core : MMU or (exclusive) MPU
- Memory Management/Protection Unit
- MMU
  - 4 GB of virtual to physical memory 4-KB page and frame size control
  - Translation Lookaside Buffers (TLBs), accelerating address translation
    - TLBs acting as n-way set-associative caches for software page tables



Soft Processor Architecture

- MPU
  - Read and write access permissions for data regions
  - Execute access permissions for instruction regions



Soft Processor Architecture

- NIOS II Core : JTAG Debug Module
  - PC to core control
    - Downloading programs to memory
    - Starting and stopping execution
    - Setting breakpoints and watchpoints (break Instruction)
    - Analyzing registers and memory contents
    - Collecting real-time execution trace data
  - The “soft” processor allows you to debug a system in development using a full-featured debug core, and later remove the debug features to conserve logic resources

Note: While the processor has no minimum clock frequency requirements, it is recommended that the design’s system clock frequency be at least 4X the JTAG clock frequency

Soft Processor Architecture

- NIOS II Core : JTAG Debug Module
  - Triggers

Control	Bus	Description
Specific Address	D / I	When the bus accesses a specific address.
Specific Data	D	When a specific data value appears on the bus.
Read Cycle	D	On a read bus cycle.
Write Cycle	D	On a write bus cycle.
Armed	D / I	After an armed trigger event.
Range	D	On a range of address or data values, or both.

Note: Off-chip trace buffer requires additional 3rd party debug software and hardware :
Imagination Technologies
Lauterbach

Architecture : Summary

- Definition of an FPGA Processor System
- NIOS Core features
- Internal and External Memory Support

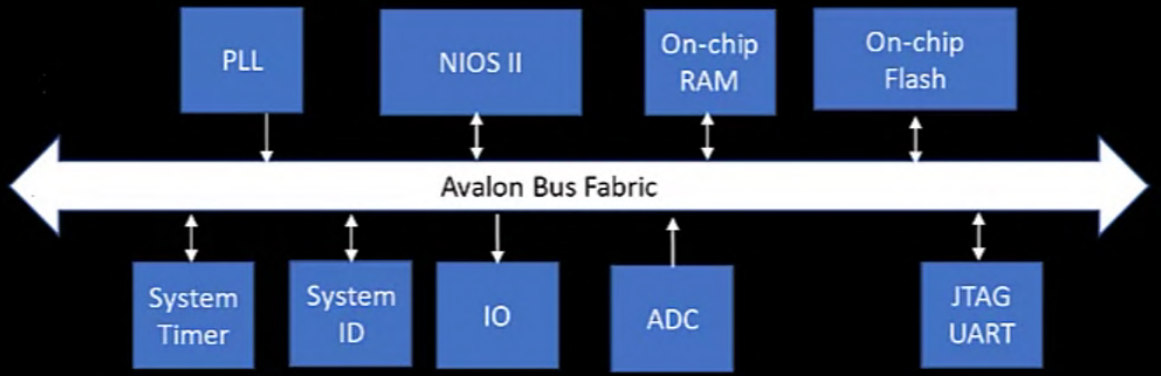
FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

NIOS II Development : Definitions

- Create Example Design : Max10
  - NIOS II and Fabric to components
  - ADC to Bus to IO 7 Segment Display
  - This video is a tool “Overview”, next video is a Demo

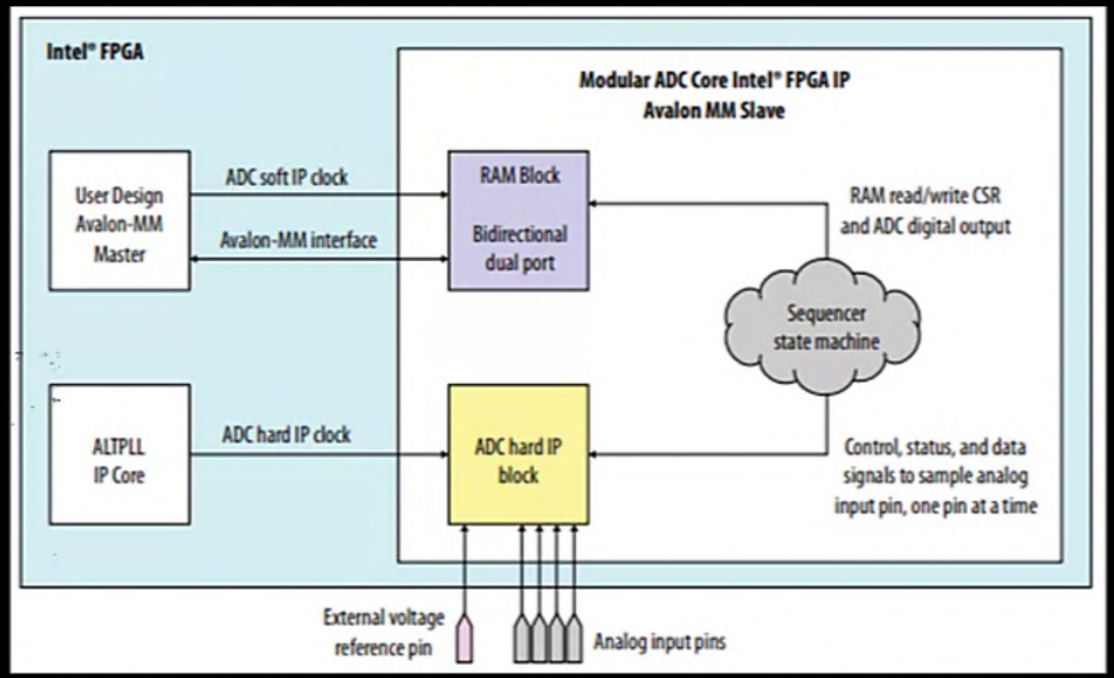


Max10 Device :
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/ug_m10_ufm.pdf

Max10 Example :
https://fpgacloud.intel.com/devstore/platform/16.0.0/Standard/nios-ii-on-die-temperature-sensor-max10-de10-lite/

NIOS II Development :

- Max 10 ADC

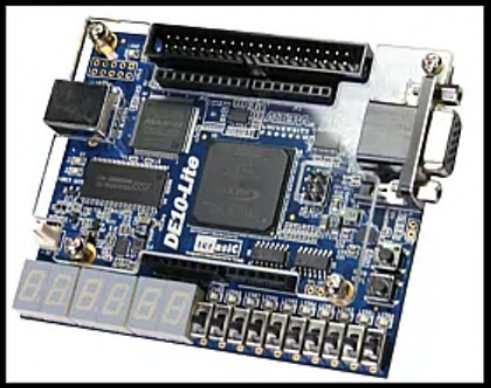


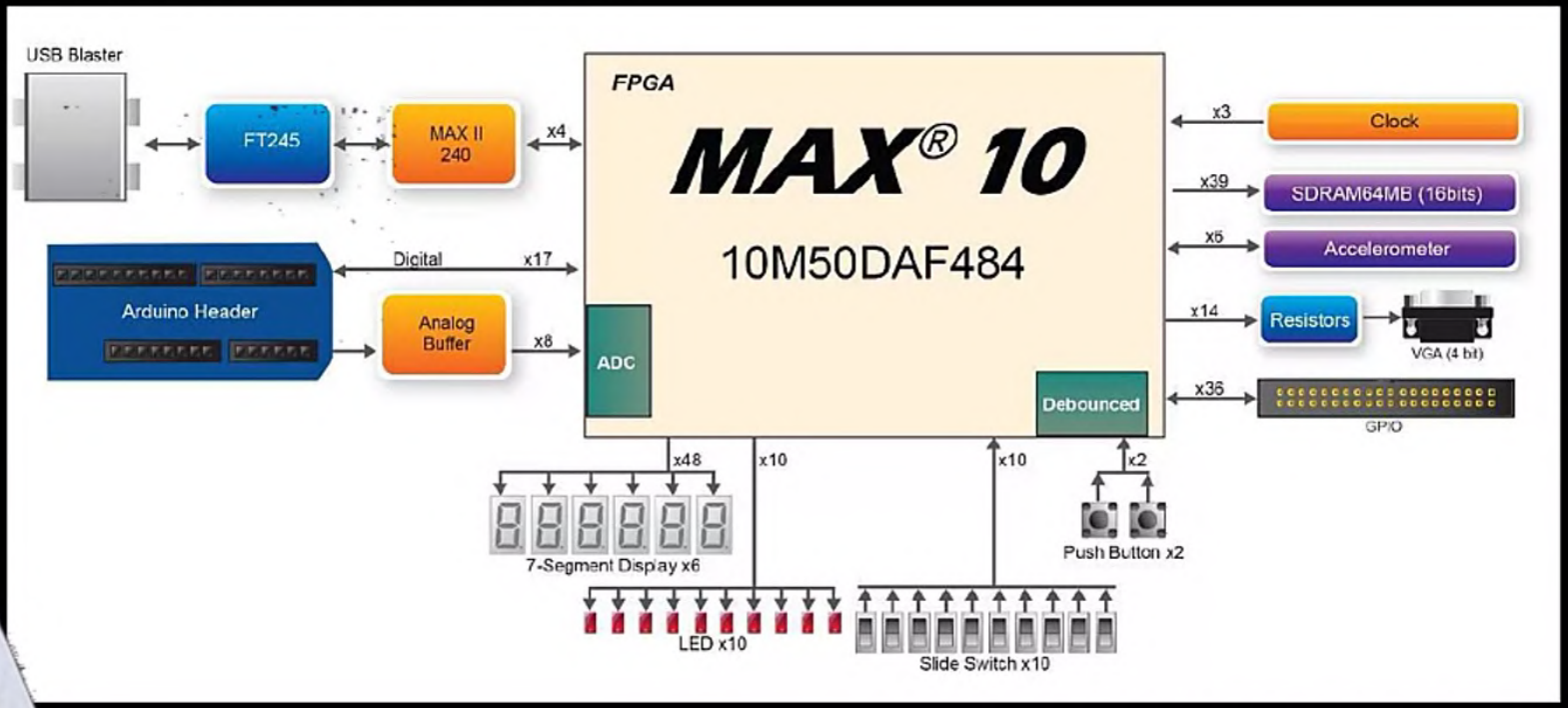
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/ug_m10_adc.pdf

NIOS II Development :

- Create Example Design : Max10
  - DE10-Lite Development Kit : (used later)
  - ~50k LEs, ADC, 2x20 GPIO

https://www.terasic.com.tw/





NIOS II Development :

- New Project Wizard : Board (tab)
  - DE10-Lite Template
  - IOs pre-defined

New Project Wizard

Family, Device & Board Settings

Device | Board

Select the board/development kit you want to target for compilation.

Family: MAX 10 Development Kit: Any

Available boards:

Name	Version	Family	Device	Vendor	LEs
Arrow MAX 10 DECA	0.9	MAX 10	10M50DAF484C6GES	Arrow	49760
BeMicro MAX 10 FPGA Evaluat...	1.0	MAX 10	10M08DAF484C8GES	Arrow	8064
MAX 10 DE10 - Lite	1.0	MAX 10	10M50DAF484C6GES	Altera	49760
MAX 10 FPGA 10M08 Evaluati...	1.0	MAX 10	10M08SAE144C8GES	Altera	8064
MAX 10 FPGA Development Kit	1.0	MAX 10	10M50DAF256C7G	Altera	49760
MAX 10 NEEK	1.0	MAX 10	10M50DAF484I7G	Terasic	49760
Odyssey MAX 10 FPGA Kit	1.0	MAX 10	10M08SAU169C8GES	Macnica Ameri...	8064

Create top-level design file.

Can't find your board? Check the Design Store for additions and search for baseline under Design Examples.

NIOS II Development :

- New Project Wizard :
  - Check Device and Board

New Project Wizard

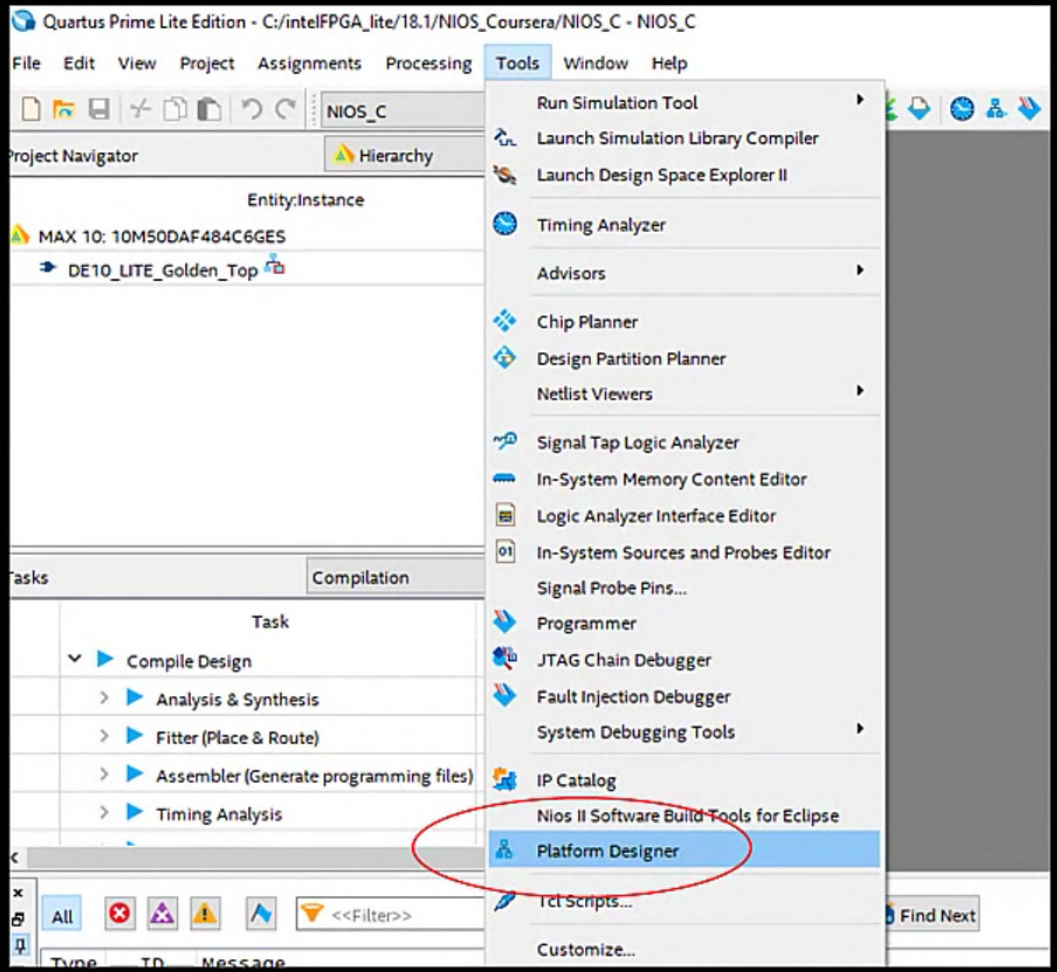
Summary

When you click Finish, the project will be created with the following settings:

Project directory:	C:\intelFPGA_lite\18.1\NIOS_Coursera
Project name:	NIOS_C
Top-level design entity:	NIOS_C
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	MAX 10
Device:	10M50DAF484C6GES
Board:	MAX 10 DE10 - Lite

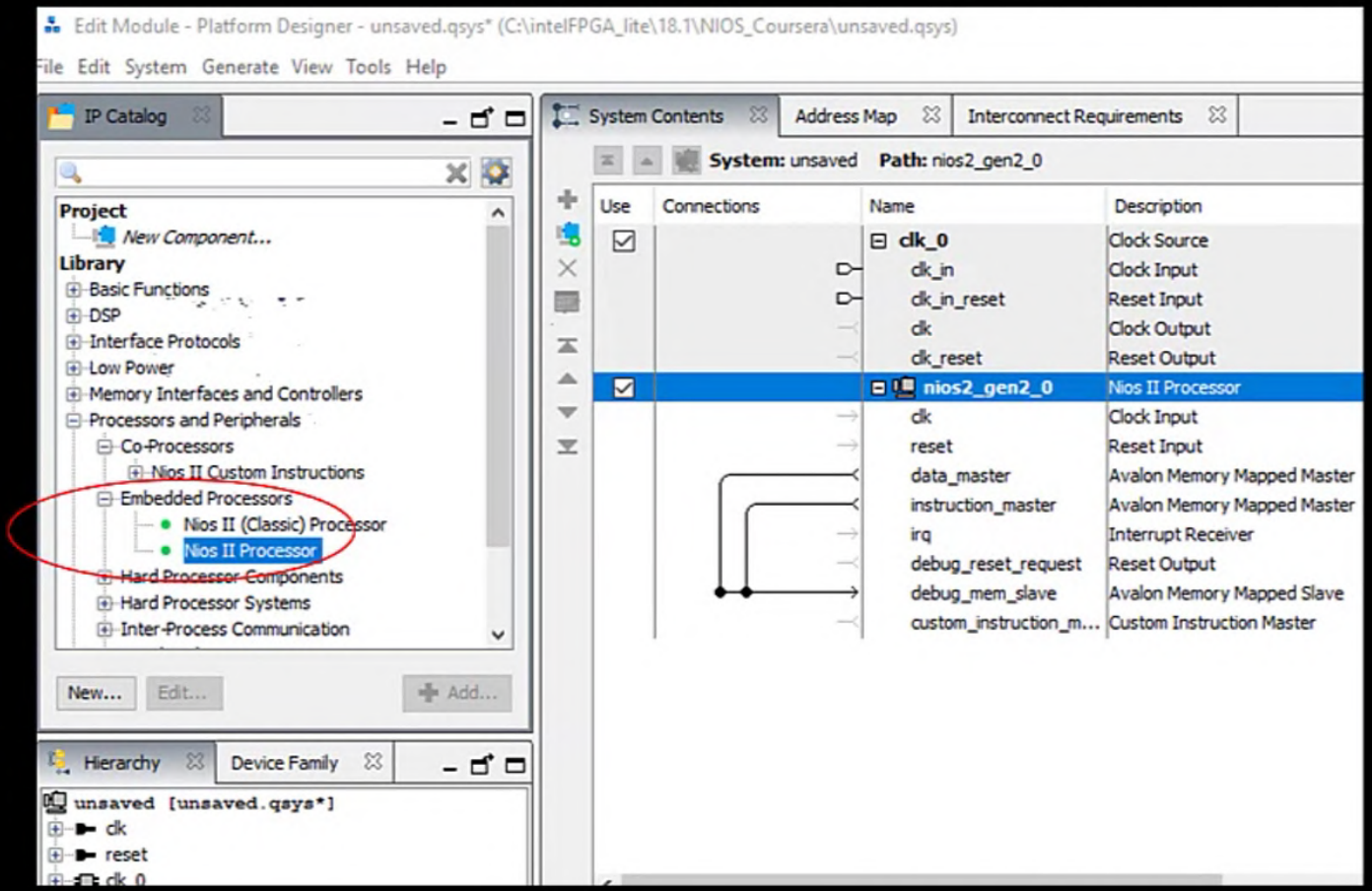
NIOS II Development :

- Tools (Menu) :
  - Platform Designer
    - Older Versions (Qsys)



NIOS II Development :

- NIOS II Platform Designer (Formerly Qsys)



NIOS II Development :

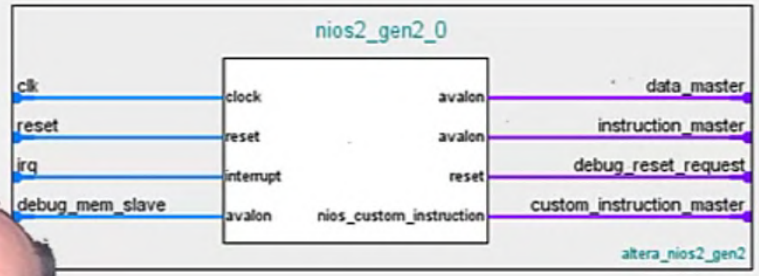
- NIOS II Processor (e) Economy or (f) Fast

Nios II Processor - nios2_gen2_0

Nios II Processor
altera_nios2_gen2

Block Diagram

Show signals



Main | Vectors | Caches and Memory Interfaces | Arithmetic Instructions | MMU and MPU Settings | JTAG Debug

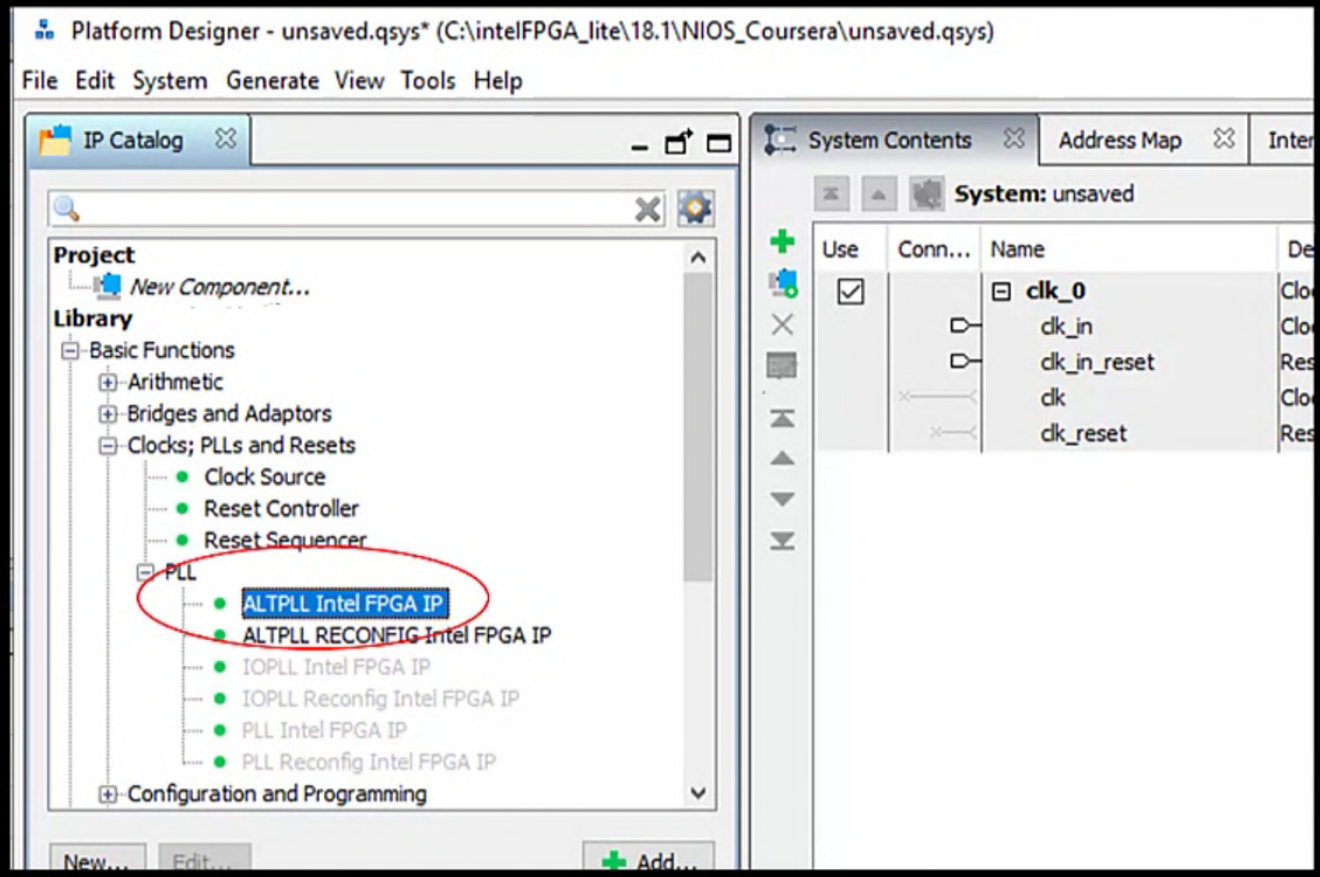
Select an Implementation

Nios II Core: ○ Nios II/e ◉ Nios II/f

	Nios II/e	Nios II/f
Summary	Resource-optimized 32-bit RISC	Performance-optimized 32-bit RISC
Features	JTAG Debug ECC RAM Protection	JTAG Debug Hardware Multiply/Divide Instruction/Data Caches Tightly-Coupled Masters ECC RAM Protection External Interrupt Controller Shadow Register Sets MPU MMU
RAM Usage	2 + Options	2 + Options

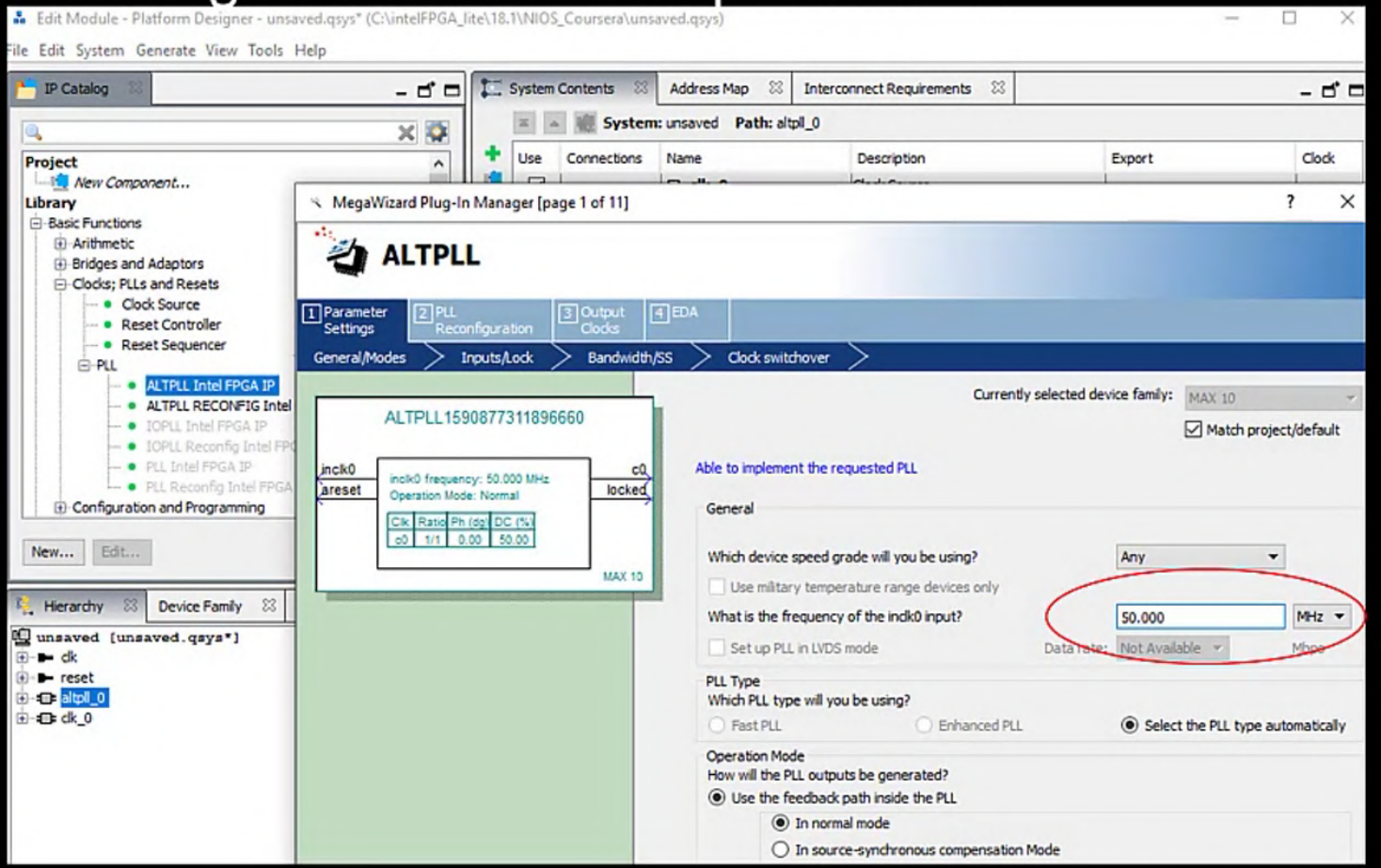
NIOS II Development :

- Add PLL : Double click



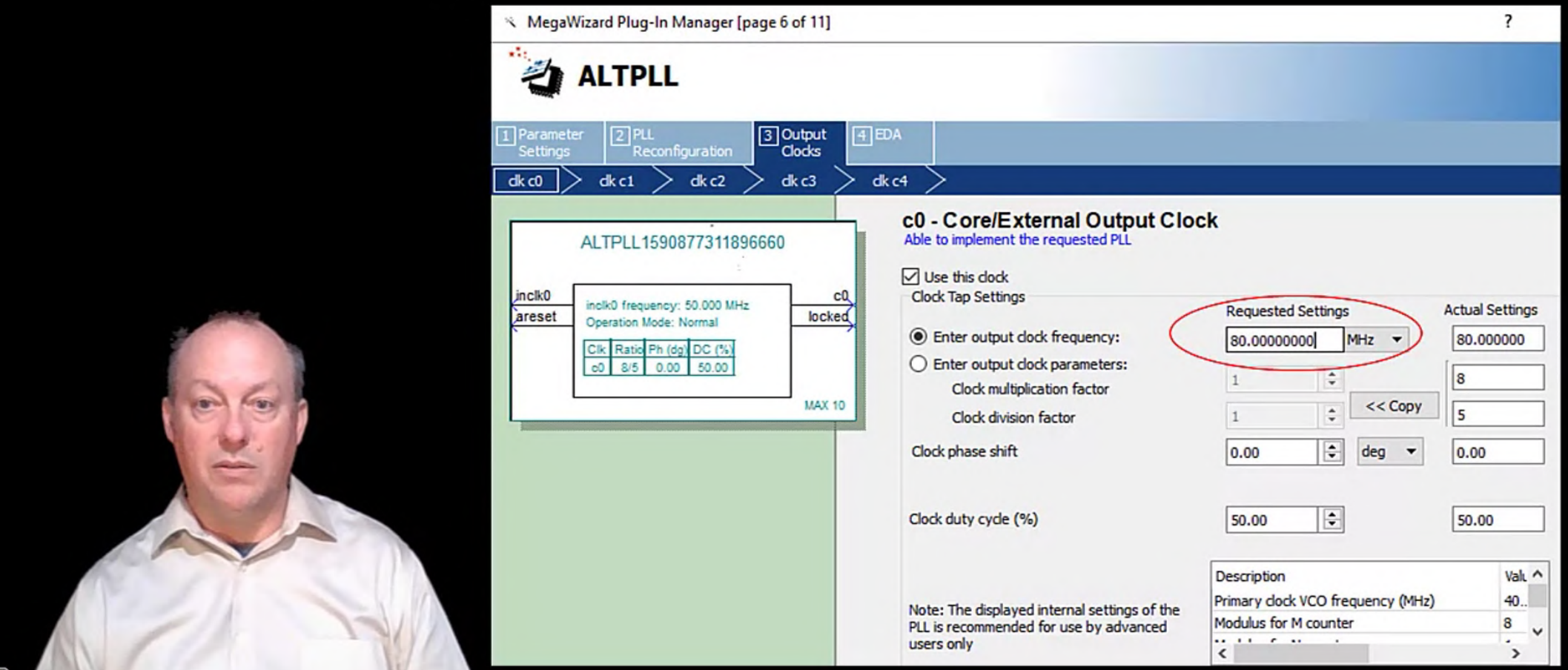
NIOS II Development :

- PLL Wizard : Assign Parameters : input 50MHz



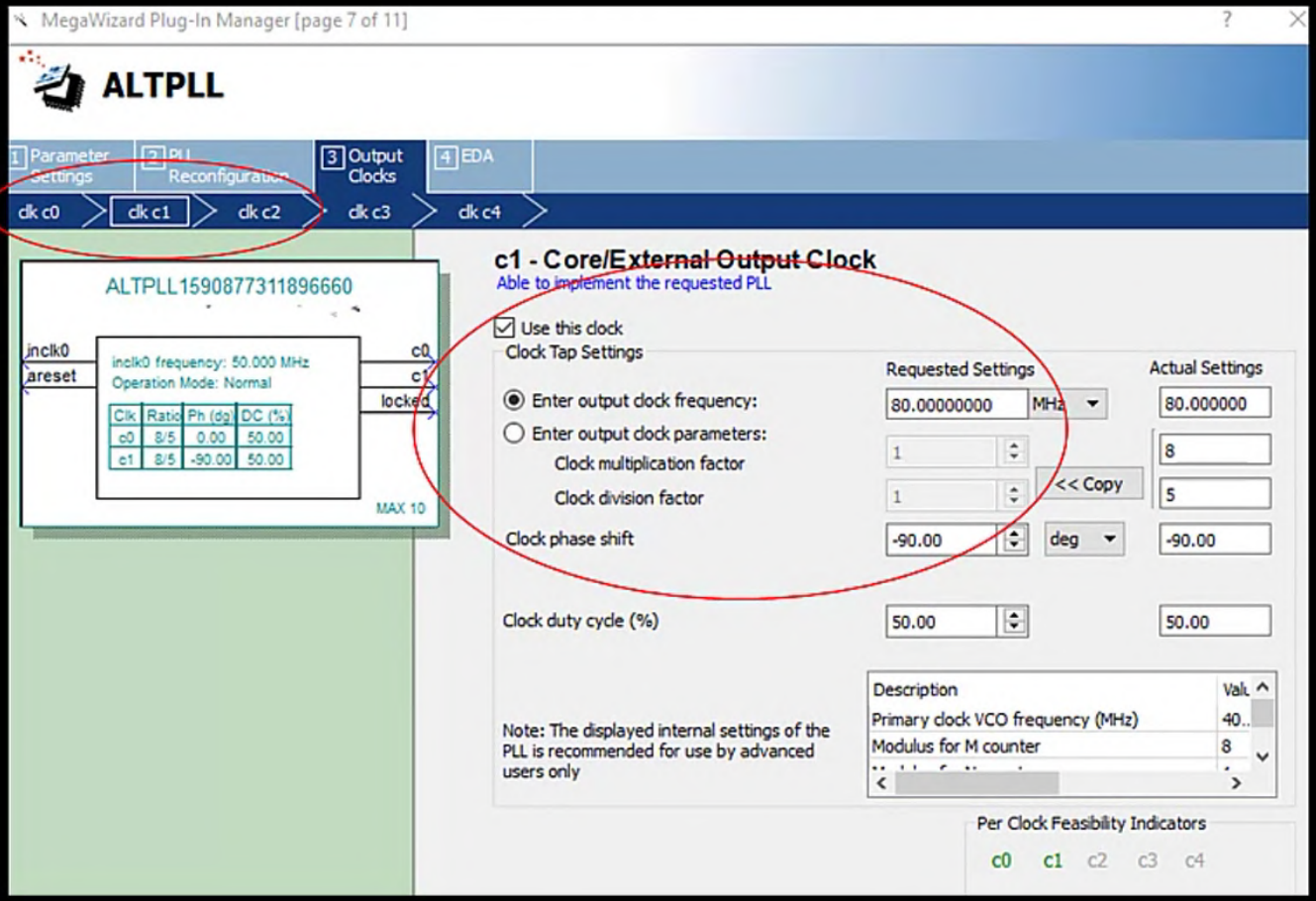
NIOS II Development :

- PLL Wizard : Output Clock c0 80MHz : NIOS



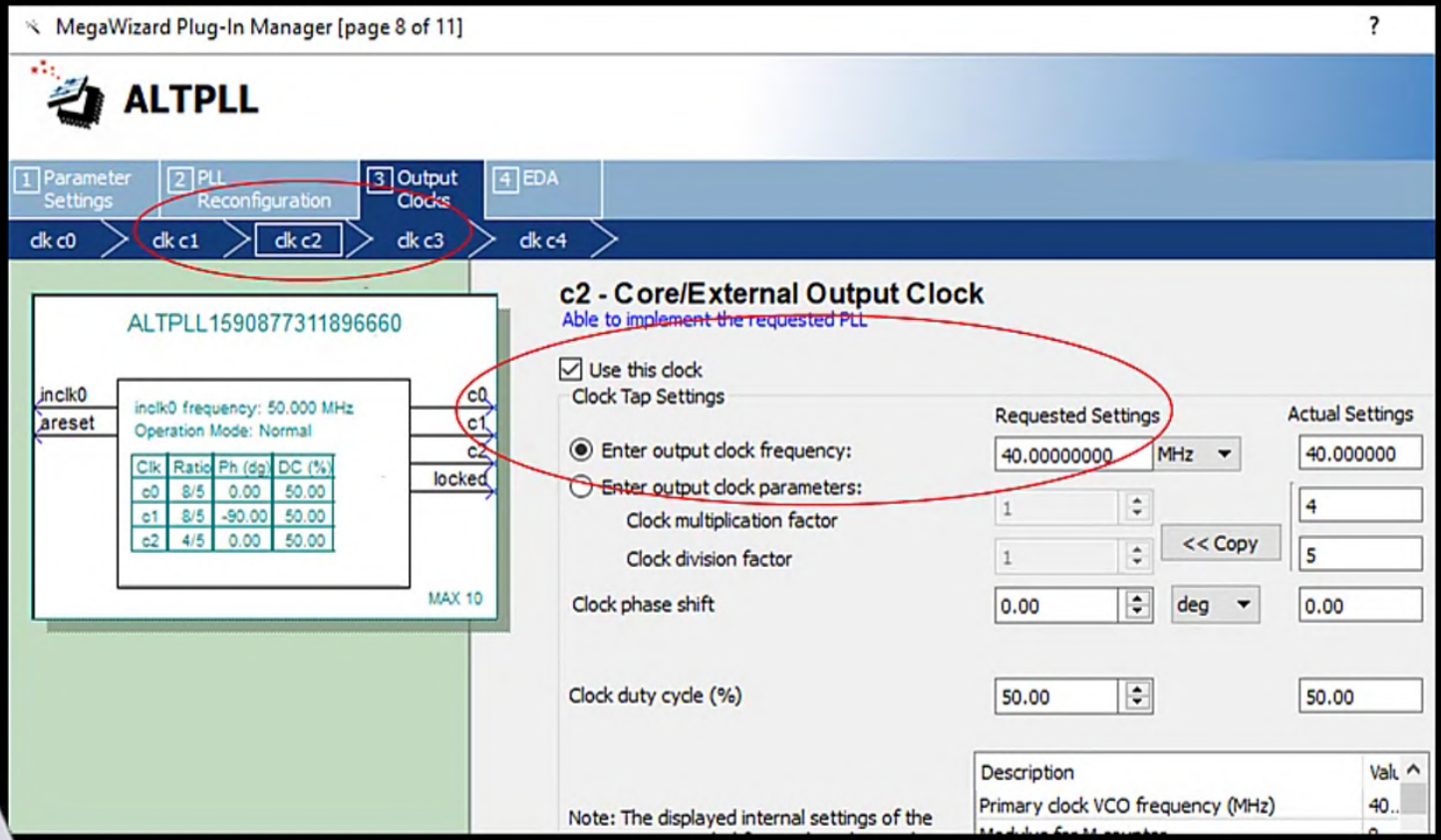
NIOS II Development :

- PLL Wizard : Output Clock c1 80MHz : SDRAM



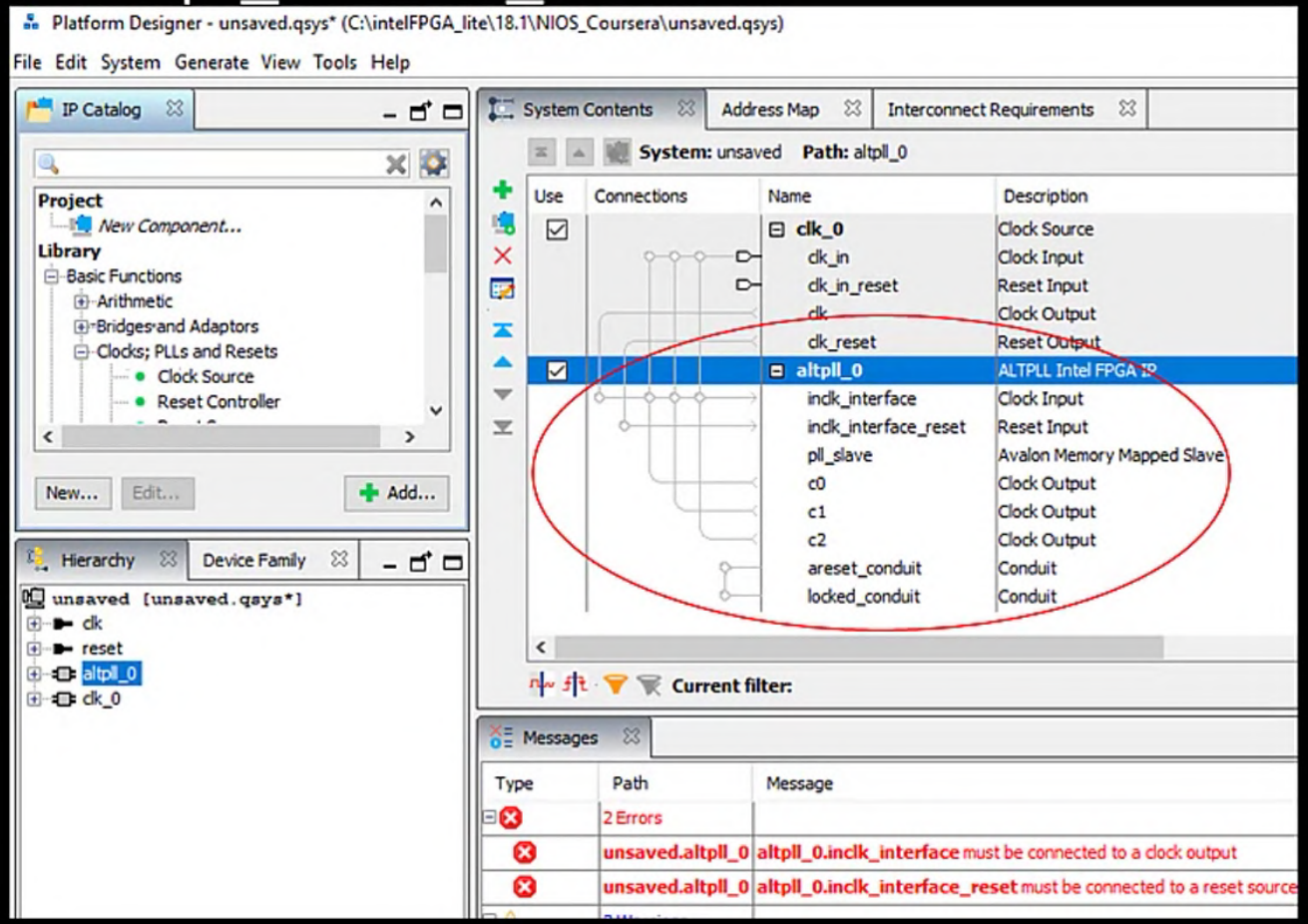
NIOS II Development :

- PLL Wizard : Output Clock c2 40MHz : Peripherals



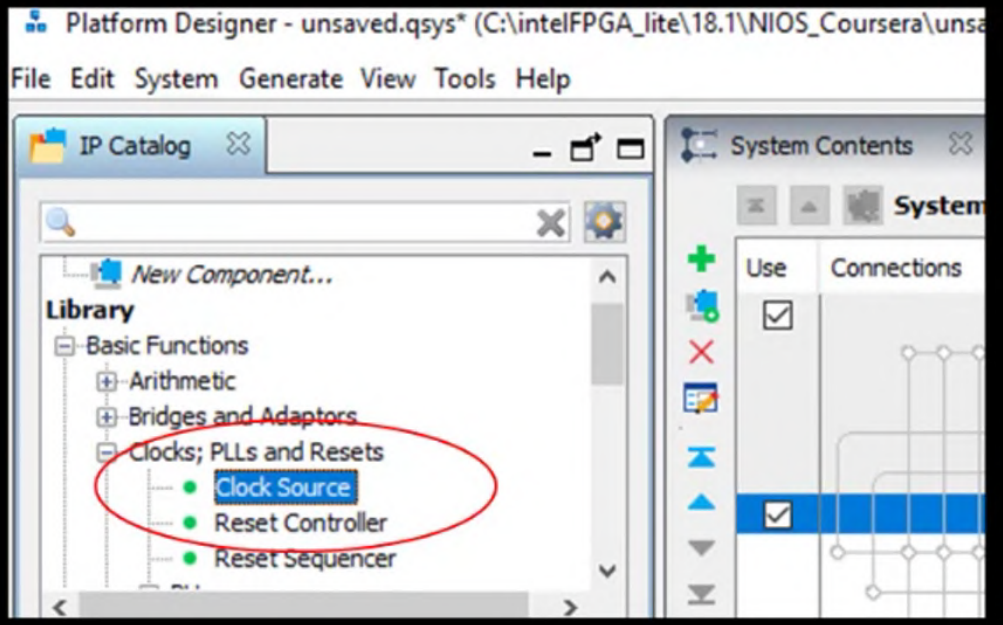
NIOS II Development :

- Platform Designer : altpll_0 to clk_in
- Errors are ok



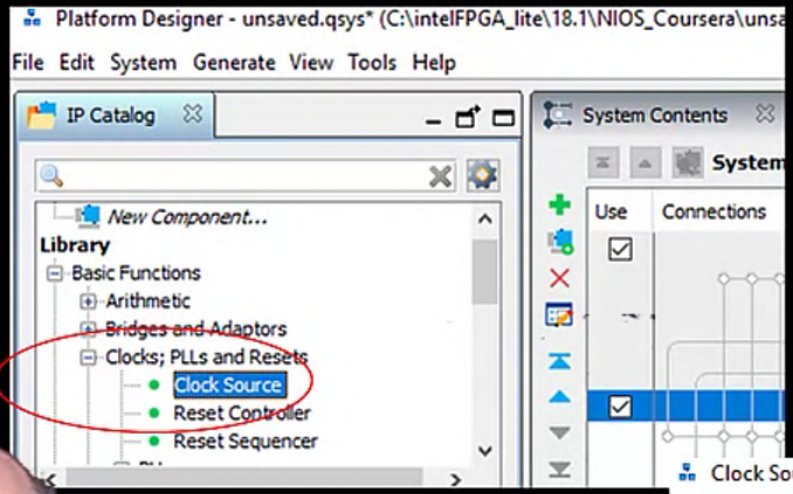
NIOS II Development :

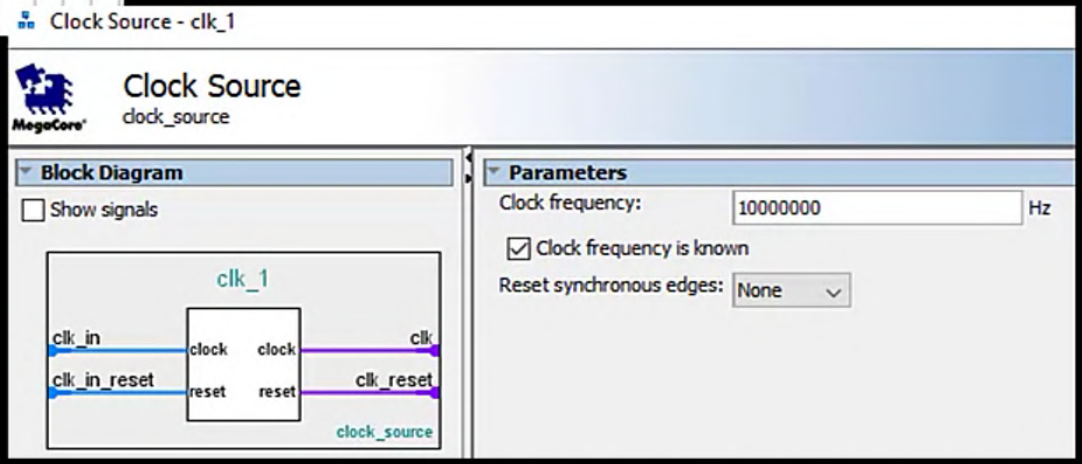
- Platform Designer : add External clock



NIOS II Development :

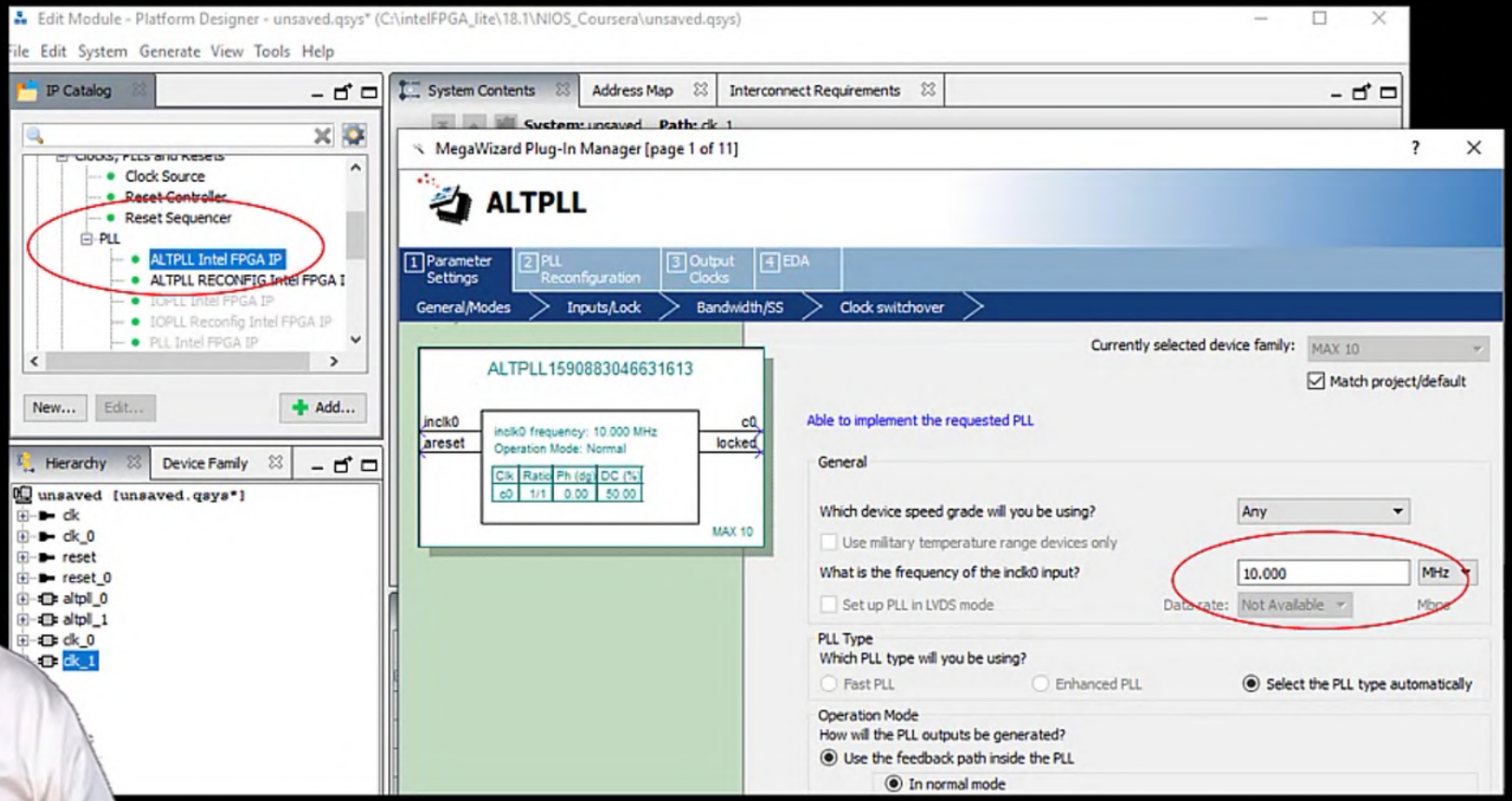
- Platform Designer : add External clock : 10MHz





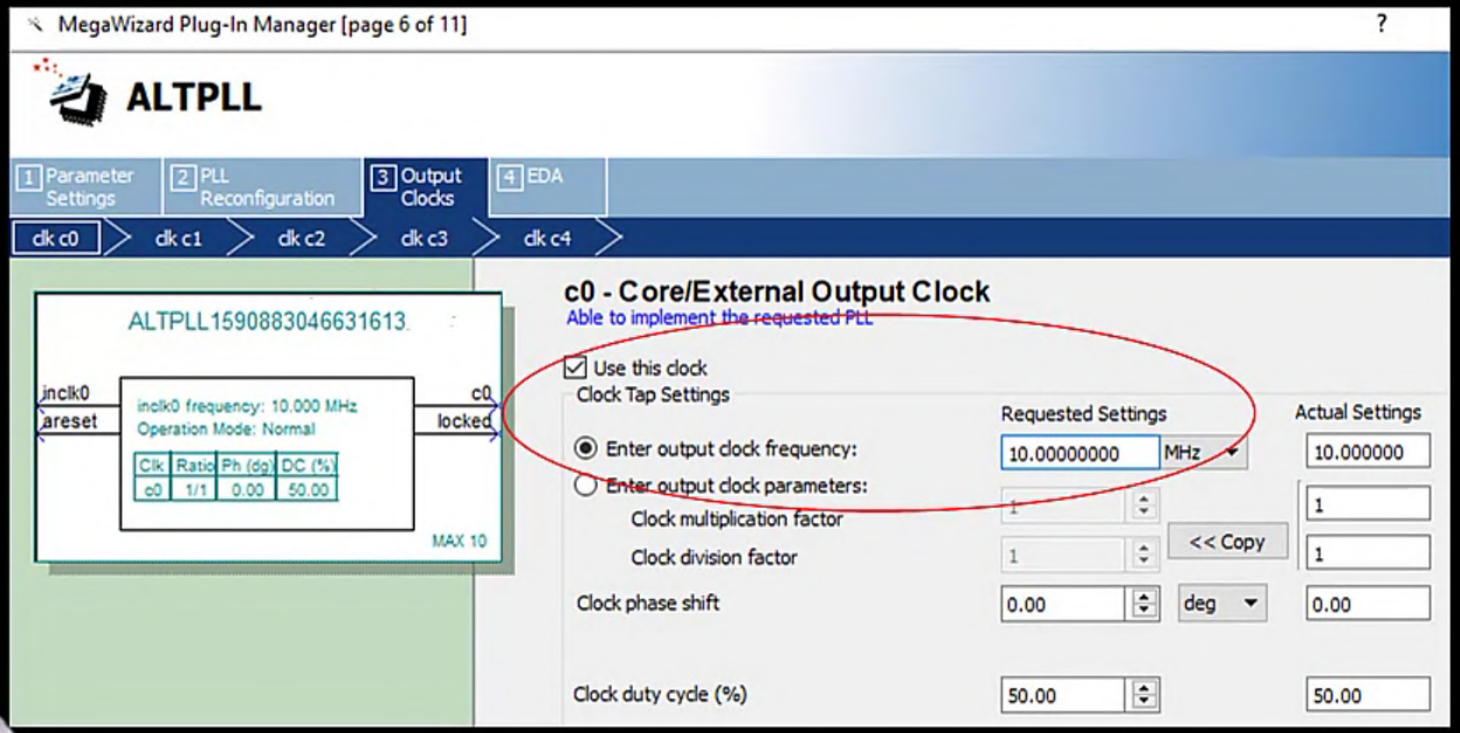
NIOS II Development :

- Platform Designer : add PLL for ADC : 10MHz



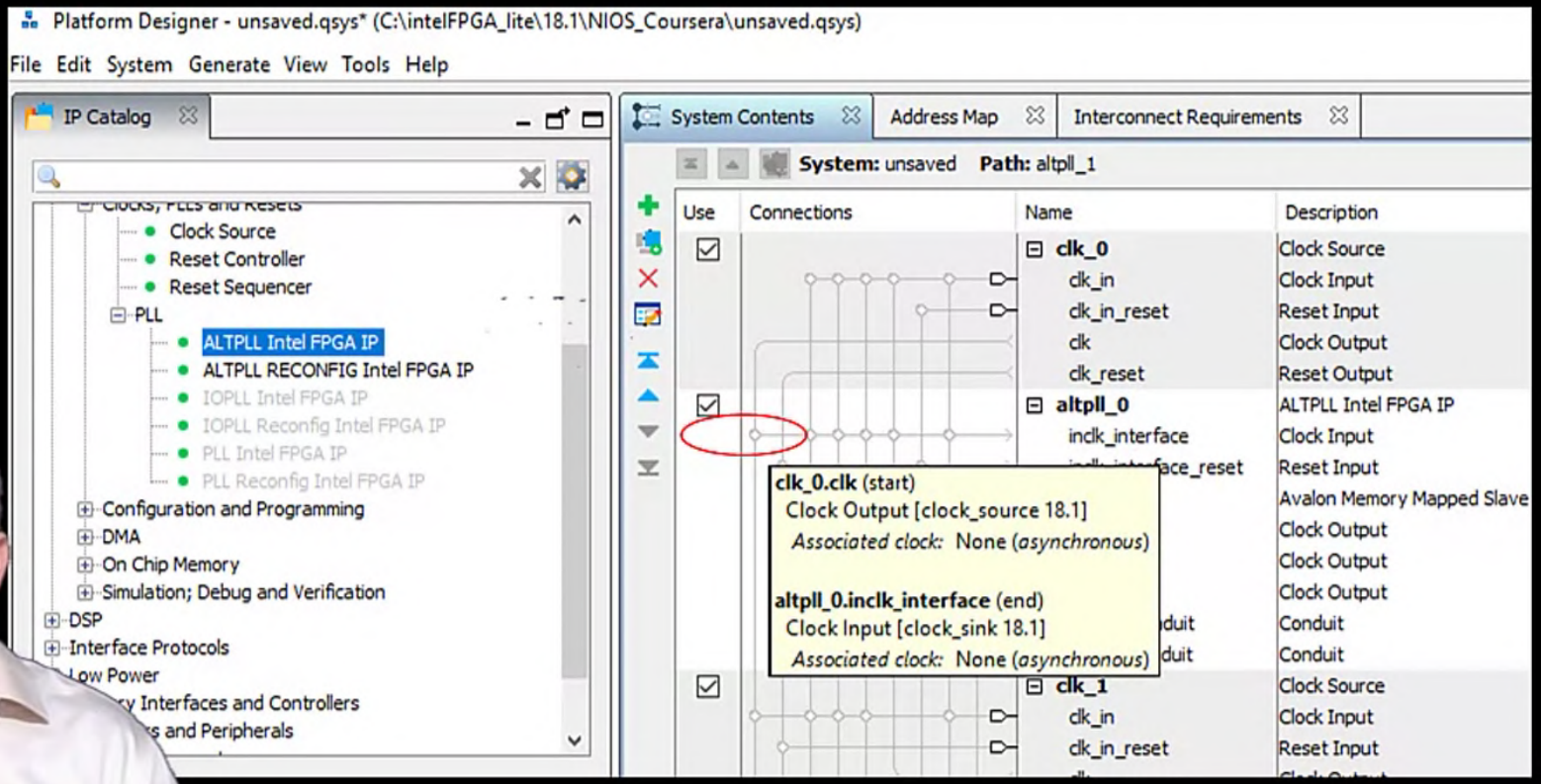
NIOS II Development :

- Platform Designer : add PLL for ADC : 10MHz



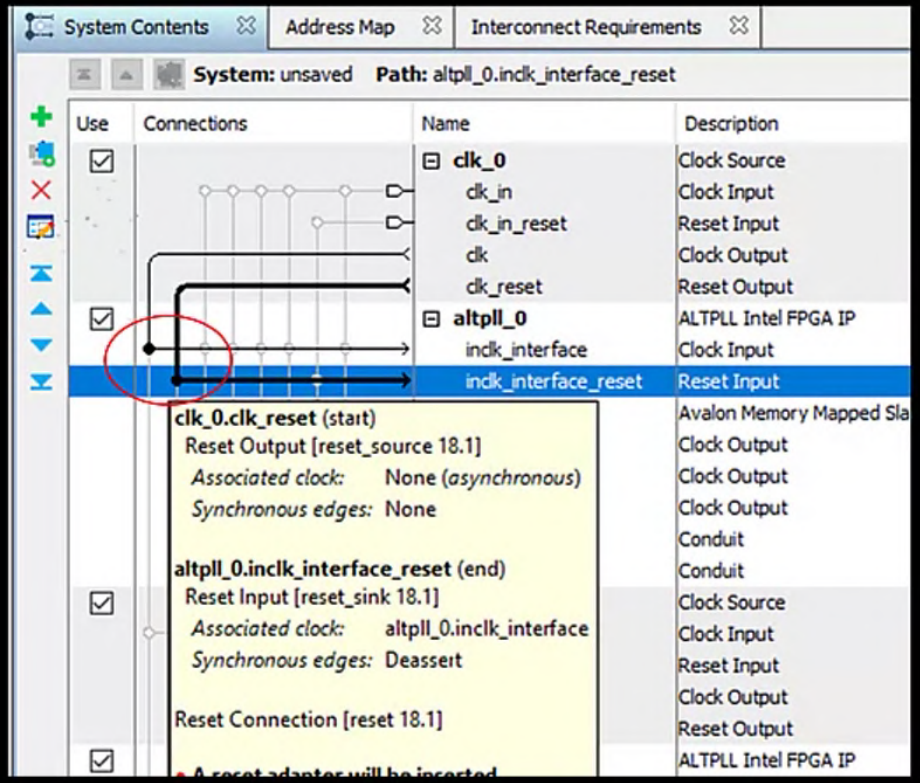
NIOS II Development :

- Platform Designer : Connections : hover over



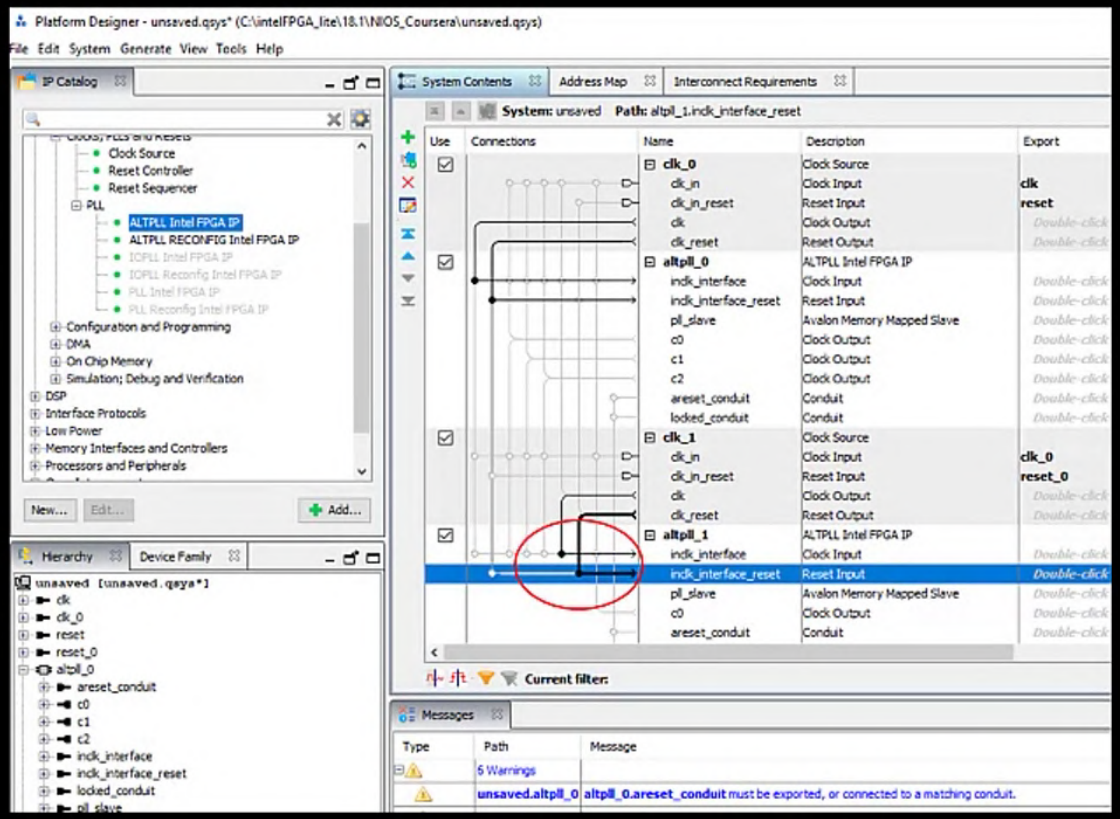
NIOS II Development :

- Platform Designer : Connections clk_0 : click
  - clk = inclk_interface, clk_reset = inclk_interface_reset



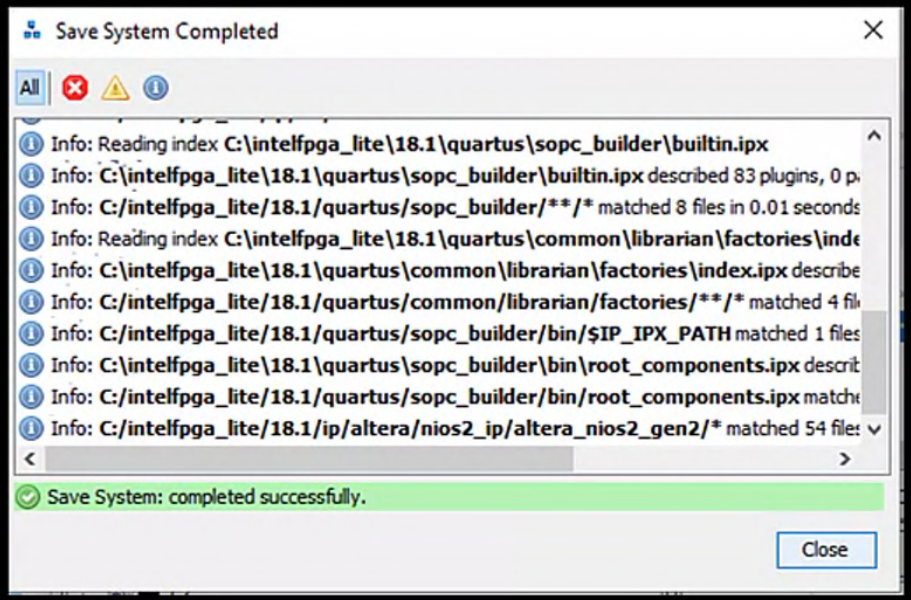
NIOS II Development :

- Platform Designer : Connections clk_1: click
  - clk = inclk_interface, clk_reset = inclk_interface_reset
  - Errors gone now



NIOS II Development :

- Platform Designer :
  - File : Save As core.qsys



Development : Summary

- NIOS II Platform Designer : System Overview

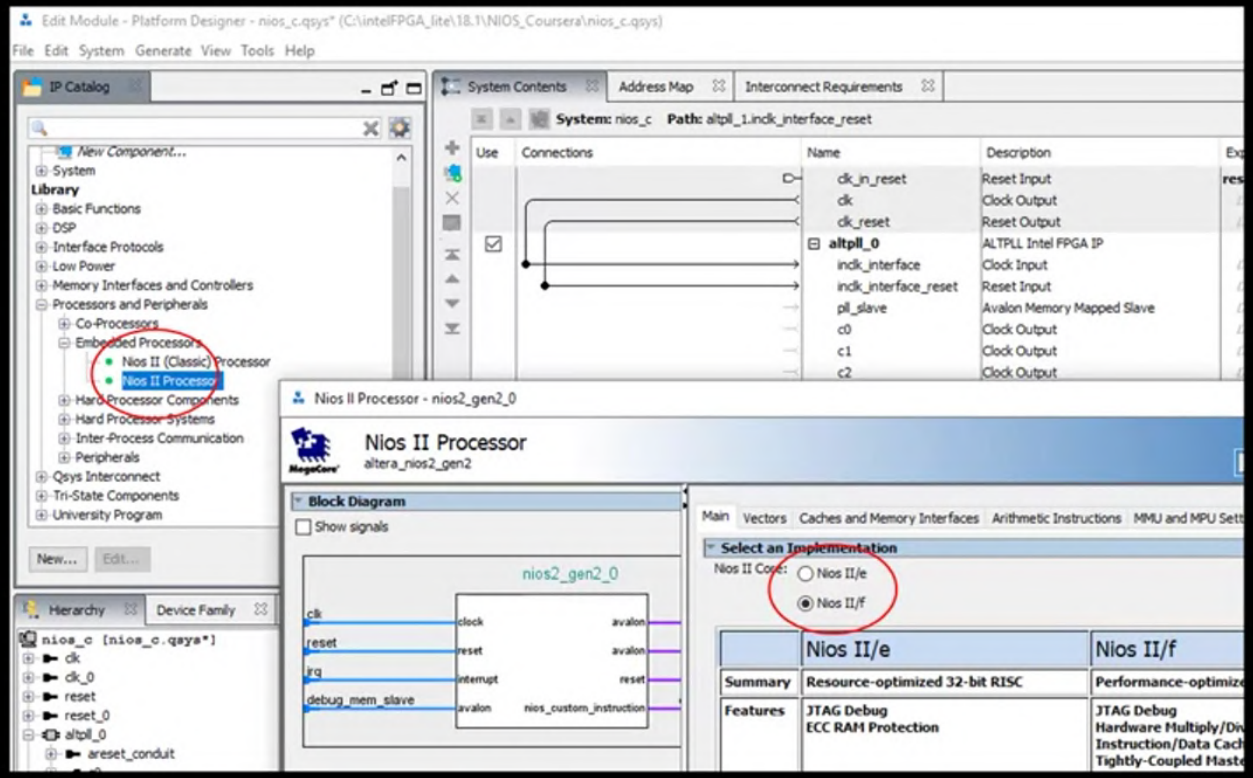
FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

University of Colorado **Boulder**

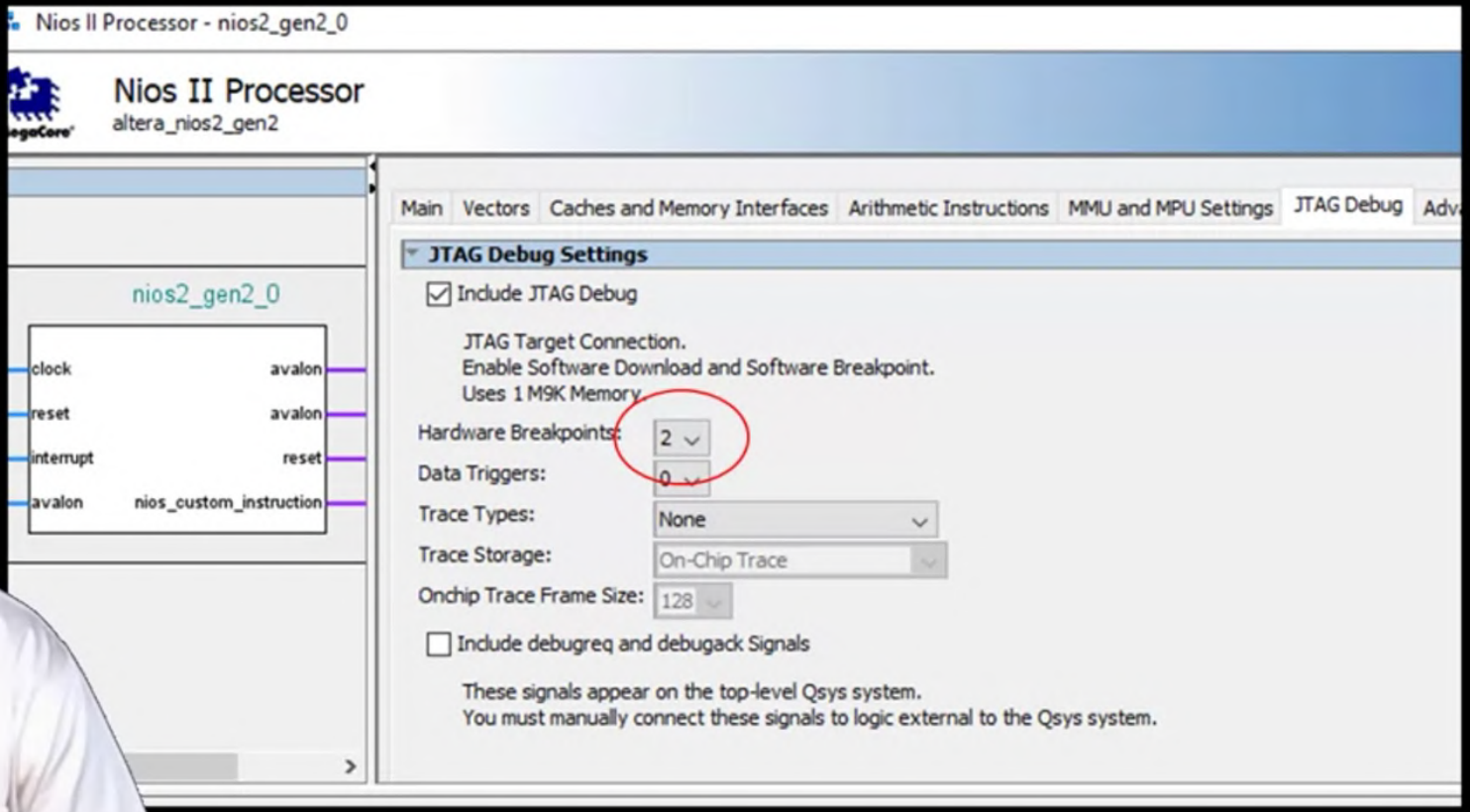
NIOS II Development : Part II

- Platform Designer : add NIOS II



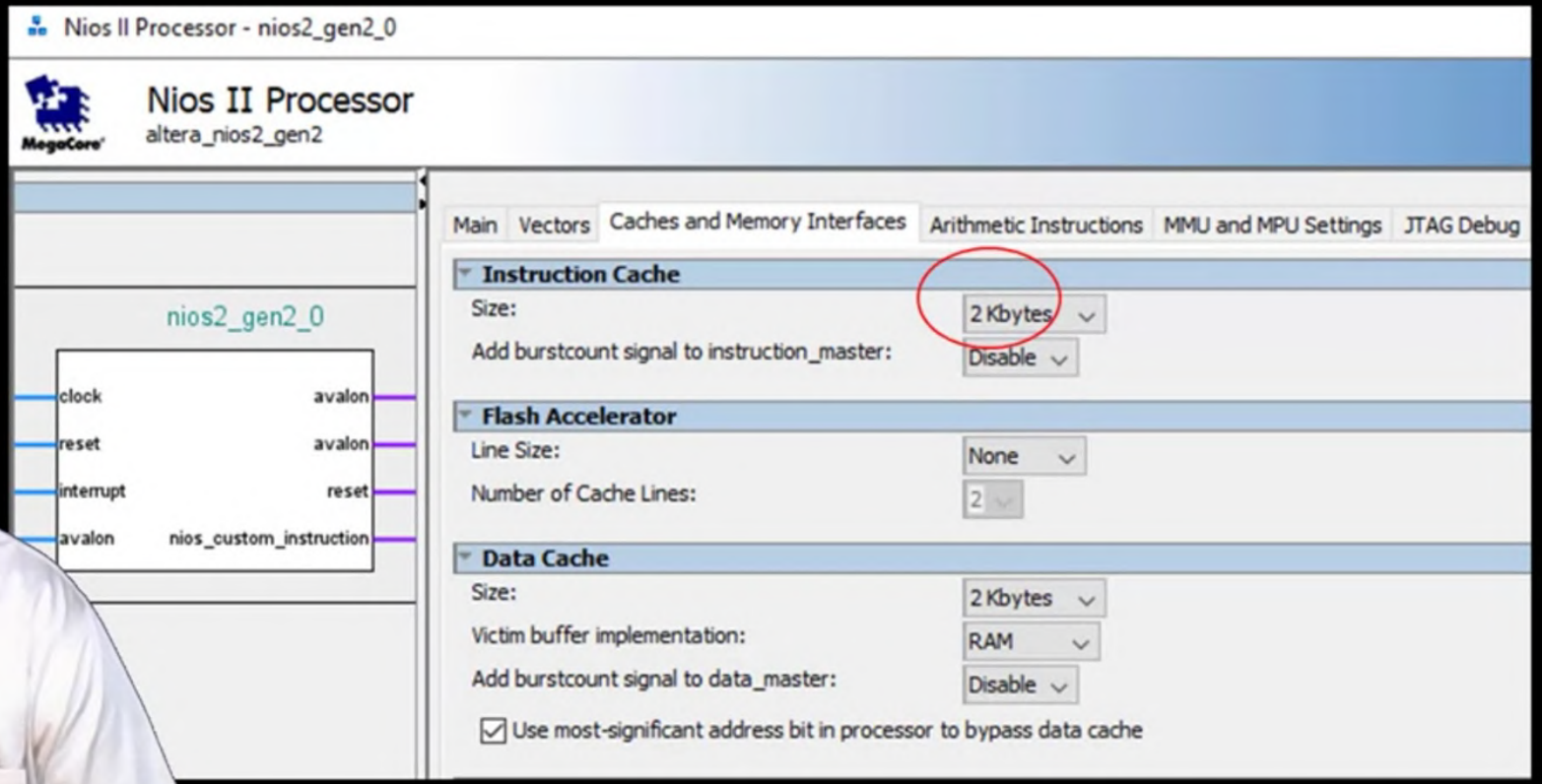
NIOS II Development :

- NIOS II : JTAG Debug (tab)



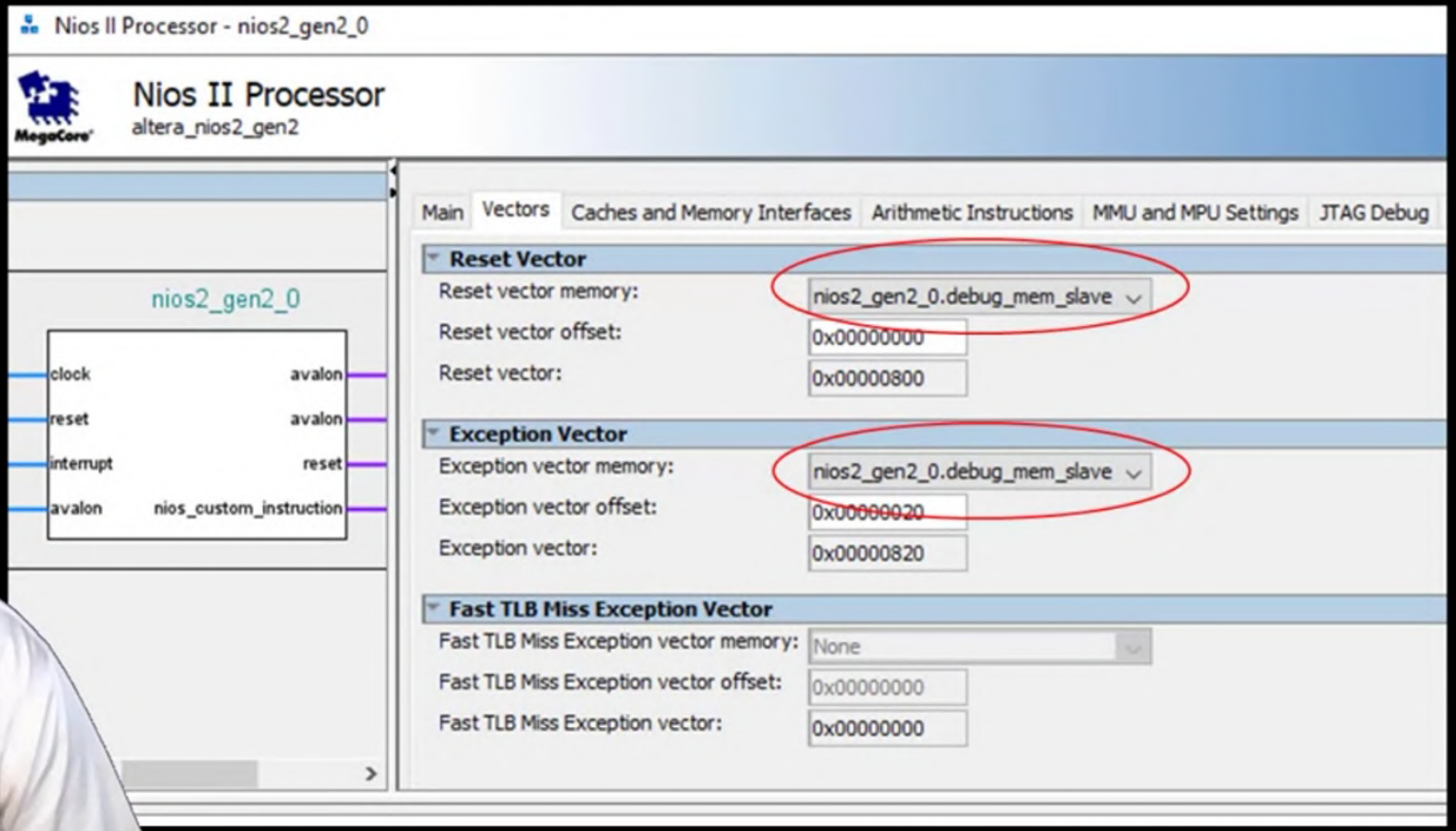
NIOS II Development :

- NIOS II: Caches and Memory (tab)



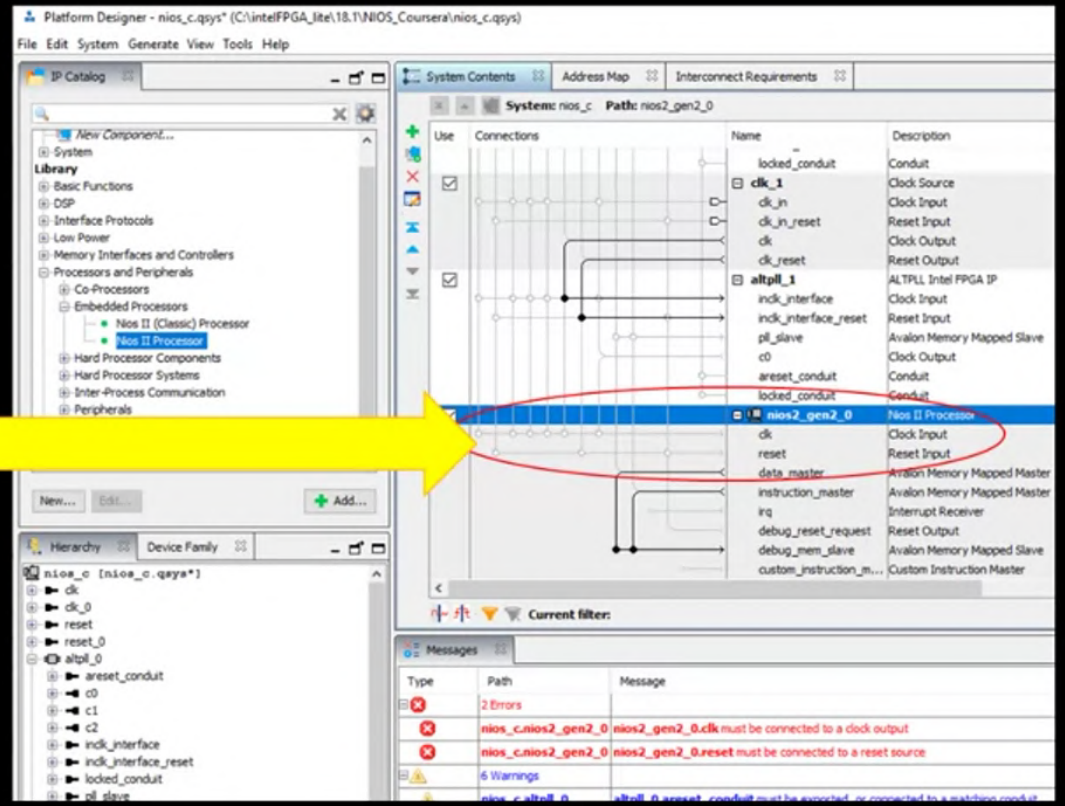
NIOS II Development :

- NIOS II : Vectors (tab)



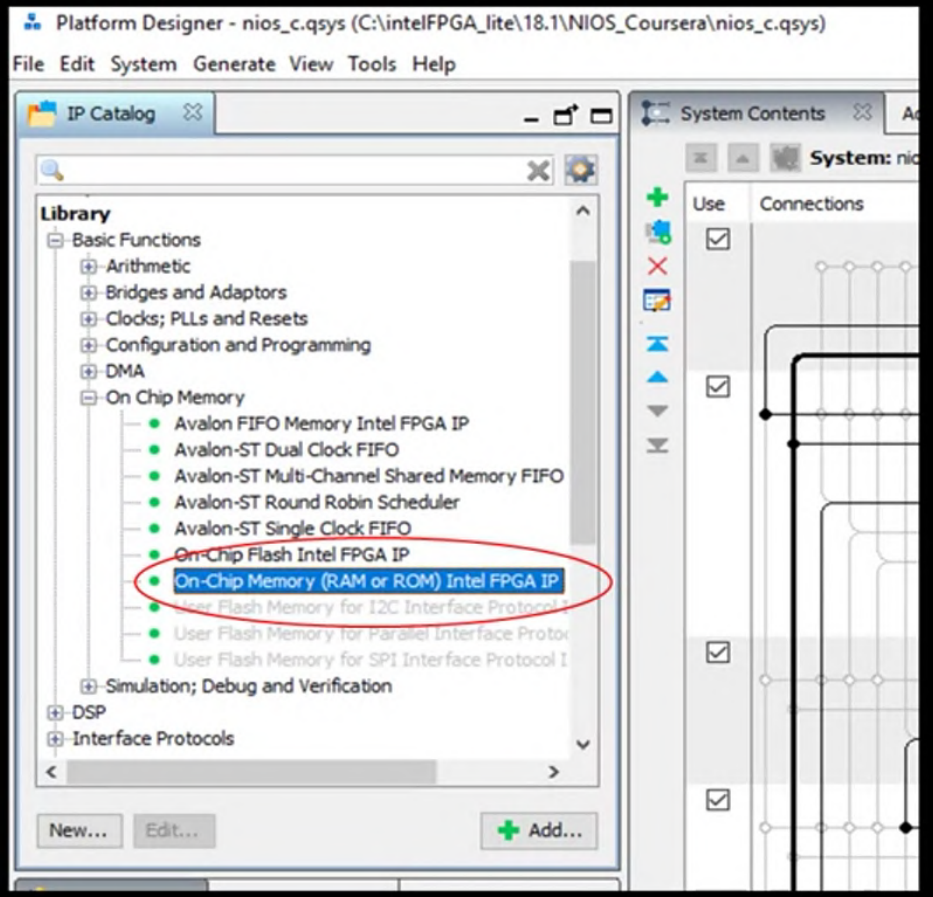
NIOS II Development :

- Platform Designer :
- Finish : Fix Errors
  - Connect the NIOS clock input to the alt pll c0 output
  - Connect the NIOS reset input to the clk_0 reset output
- Easy Right? (Click)
- File (menu) : Save



NIOS II Development :

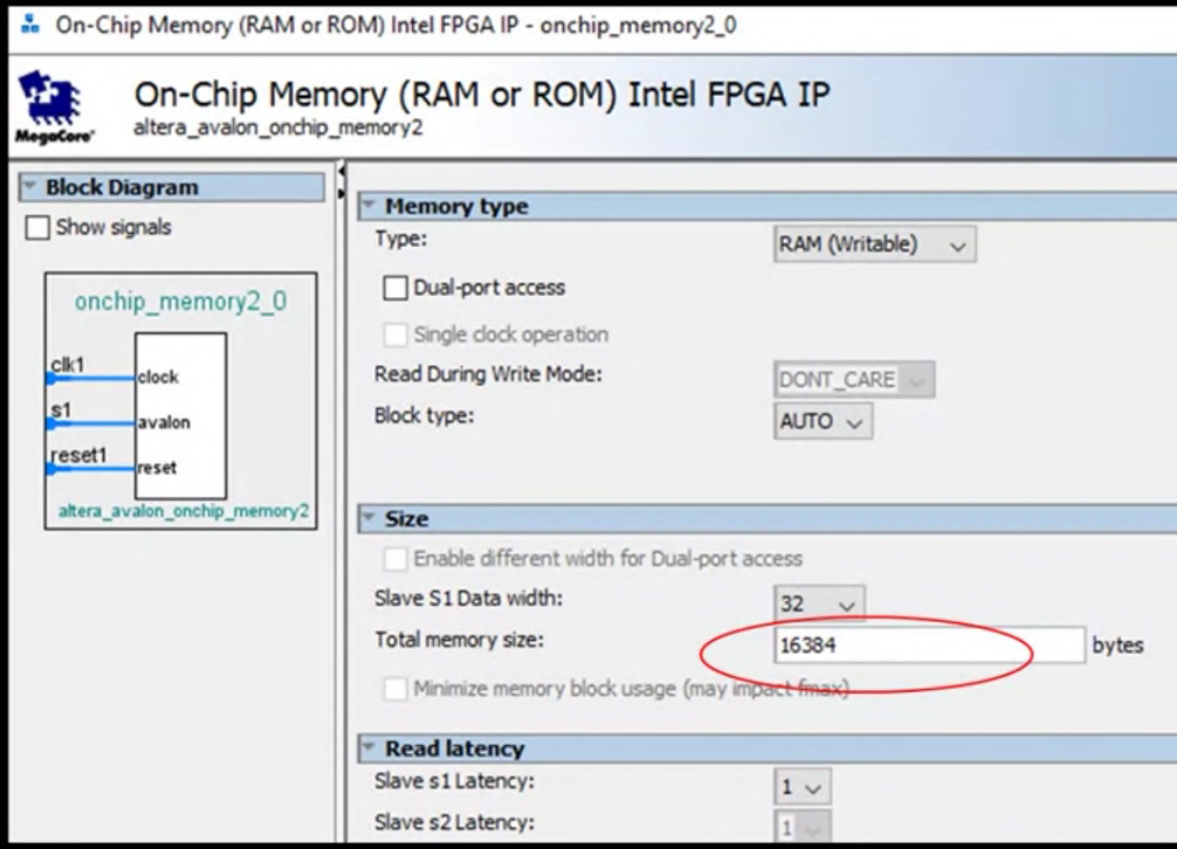
- Platform Designer : add RAM



NIOS II Development :

• Platform Designer : add RAM

- Connect the memory clock input to the alt pll c0 clock
- Connect the memory reset1 signal to the clk_0_reset output
- Using the Connections column, connect the s1 Avalon Memory Mapped Slave interface of the on-chip RAM to the nios2 instruction_master and nios2 data_master
- Set the base address of the on-chip RAM s1 to 0x4000



NIOS II Development :

- Platform Designer : add Flash

On-Chip Flash Intel FPGA IP - onchip_flash_0

On-Chip Flash Intel FPGA IP
altera_onchip_flash

Block Diagram

Show signals

onchip_flash_0
clk — clock
nreset — reset
data — avalon
csr — avalon
altera_onchip_fl

Parameters

Data interface: Parallel
Read burst mode: Incrementing
Read burst count: 8

Configuration Mode

Configuration Scheme: Internal Configuration
Configuration Mode: Single Uncompressed Image with Memory Initialization

Flash Memory

Sector ID	Access Mode	Address Mapping	Type
1	Read and write	0x00000 - 0x07fff	UFM
2	Read and write	0x08000 - 0x0ffff	UFM
3	Read and write	0x10000 - 0x6ffff	CFM
4	Read and write	0x70000 - 0xb7fff	CFM
5	Read and write	0xb8000 - 0x15ffff	CFM

Clock Source

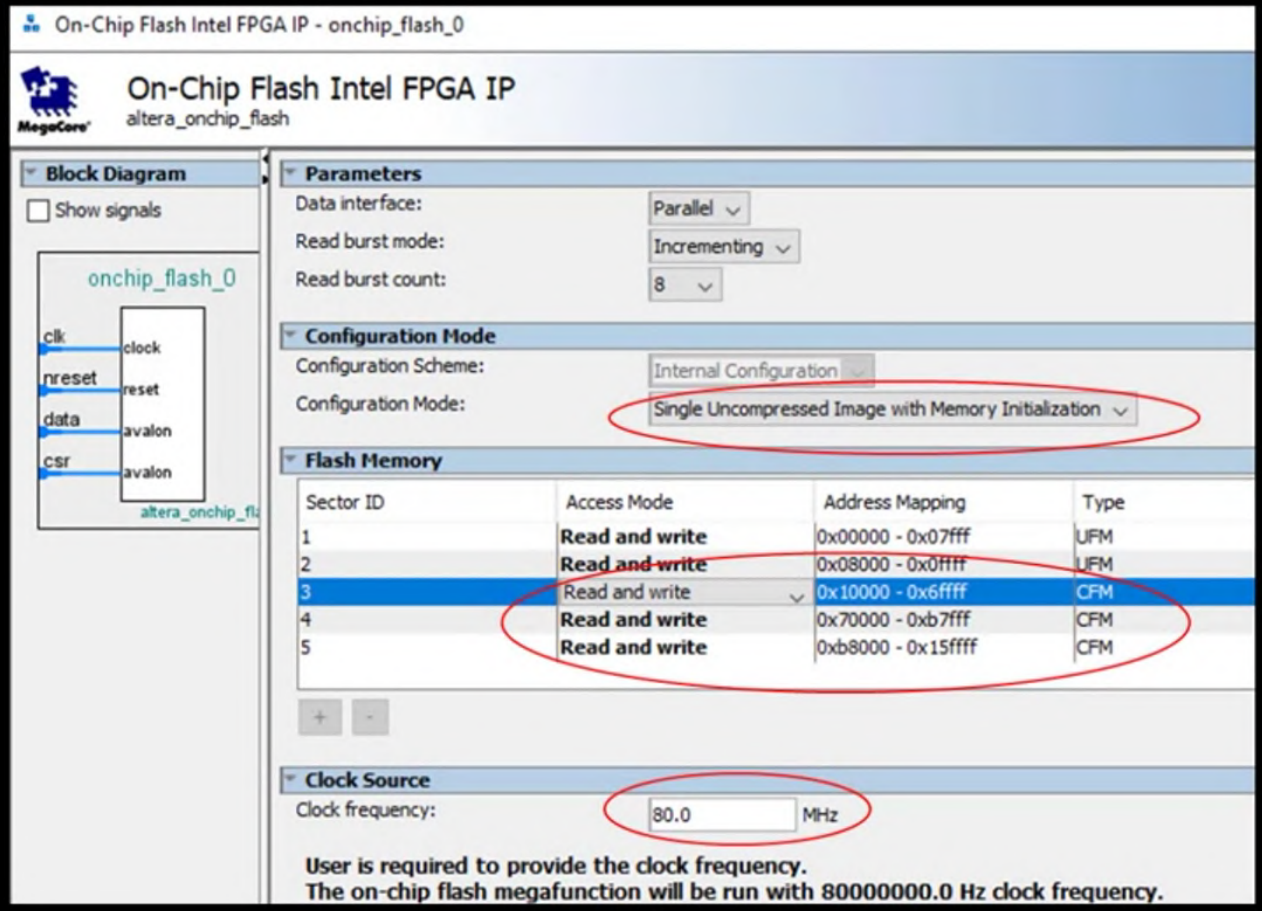
Clock frequency: 80.0 MHz

User is required to provide the clock frequency.
The on-chip flash megafunction will be run with 80000000.0 Hz clock frequency.

NIOS II Development :

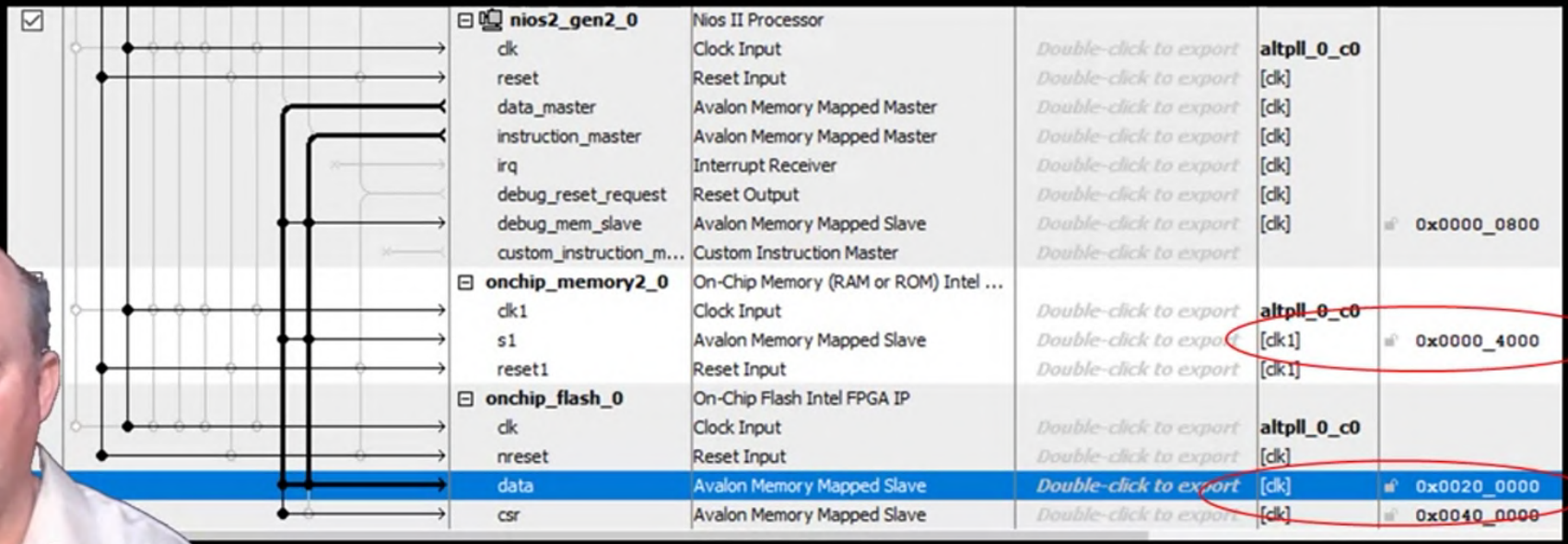
• Platform Designer : add Flash

- Connect altpll_0_c0 as the clock
- Connect the data to both nios2 data_master and nios2 instruction_master
- Connect the csr (control and status registers) to the nios2 data_master only.
- Change the base address of onchip flash data to 0x0020 0000
- Change the base address of onchip flash csr to 0x0040 0000

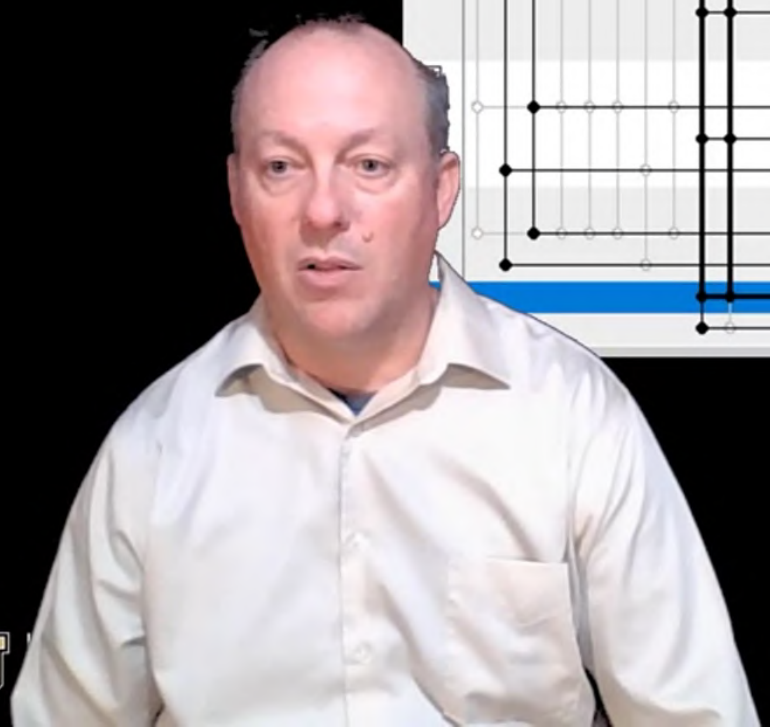


NIOS II Development :

- Platform Designer : Memory and Flash



⊟	nios2_gen2_0	Nios II Processor			
	clk	Clock Input	Double-click to export	altpll_0_c0	
	reset	Reset Input	Double-click to export	[clk]	
	data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
	instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
	irq	Interrupt Receiver	Double-click to export	[clk]	
	debug_reset_request	Reset Output	Double-click to export	[clk]	
	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0800
	custom_instruction_m...	Custom Instruction Master	Double-click to export		
⊟	onchip_memory2_0	On-Chip Memory (RAM or ROM) Intel ...			
	clk1	Clock Input	Double-click to export	altpll_0_c0	
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000_4000
	reset1	Reset Input	Double-click to export	[clk1]	
⊟	onchip_flash_0	On-Chip Flash Intel FPGA IP			
	clk	Clock Input	Double-click to export	altpll_0_c0	
	nreset	Reset Input	Double-click to export	[clk]	
	data	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0020_0000
	csr	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0040_0000

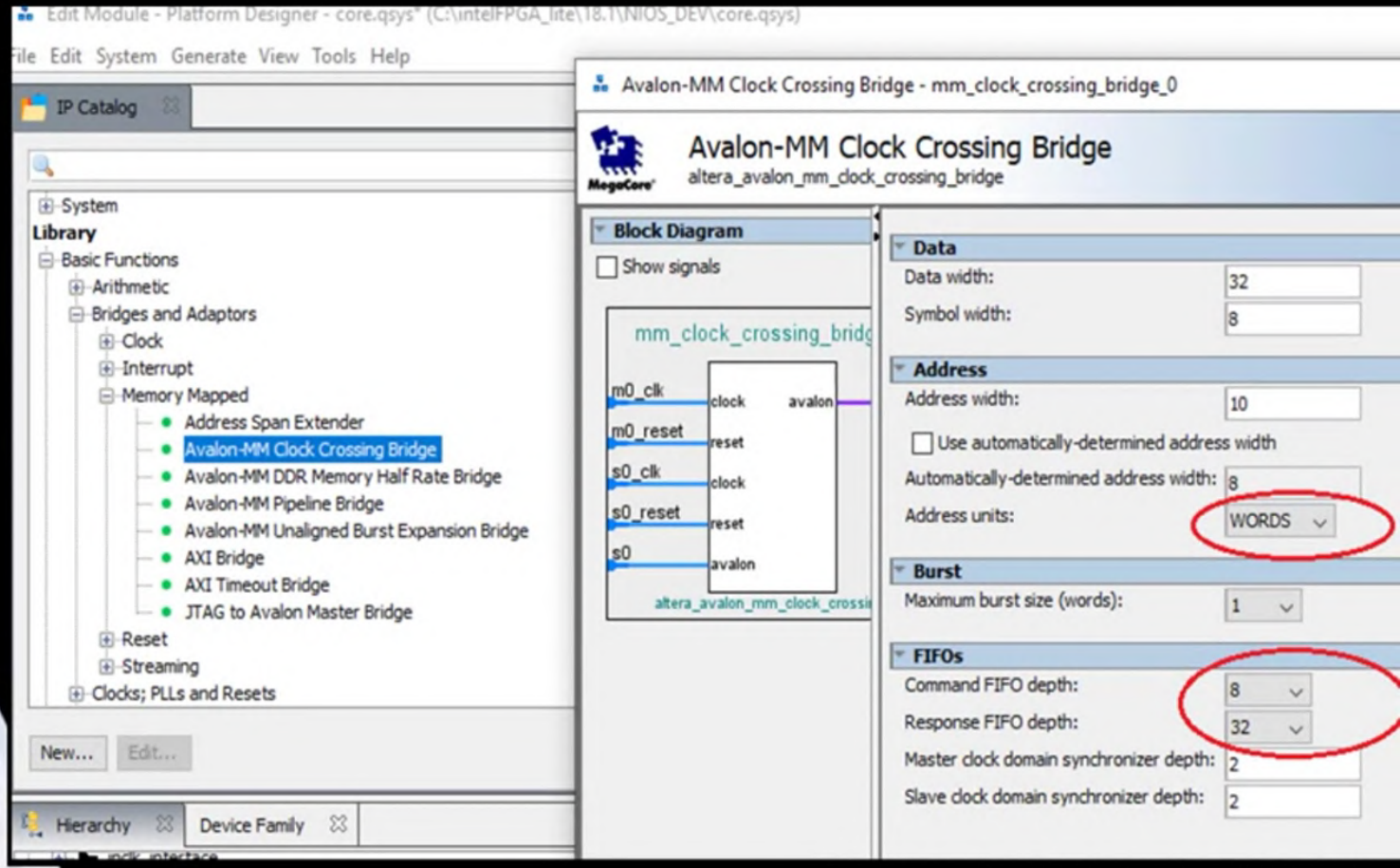


NIOS II Development :

• Platform Designer : add A-MM Clock Crossing Bridge

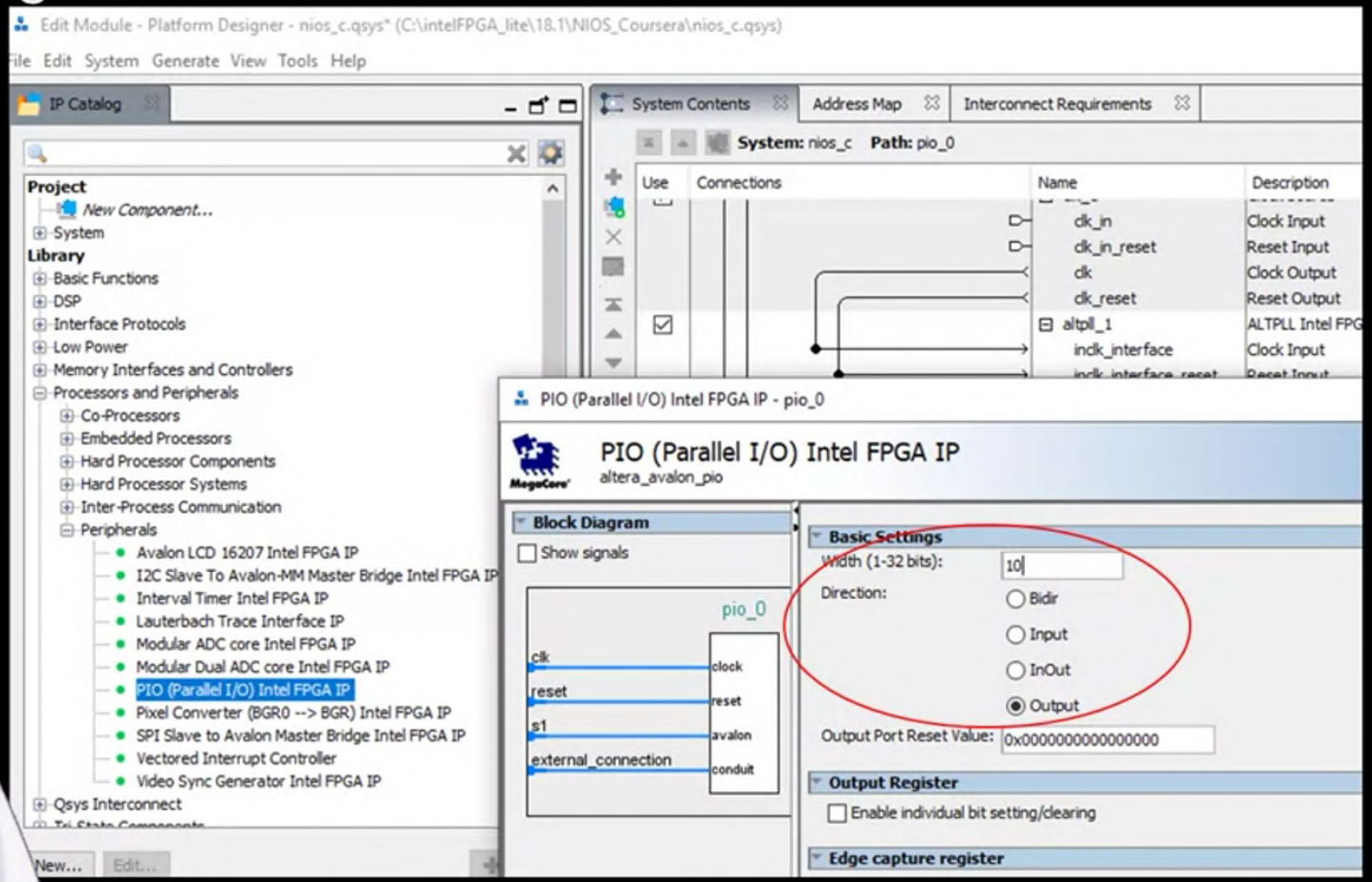
- Address : WORDS
- FIFO :
  - Command 8
  - Response 32

- m0_clk to the pll_0 c2 clock
- s0 clk to the pll_0 c0
- both resets to clk_0



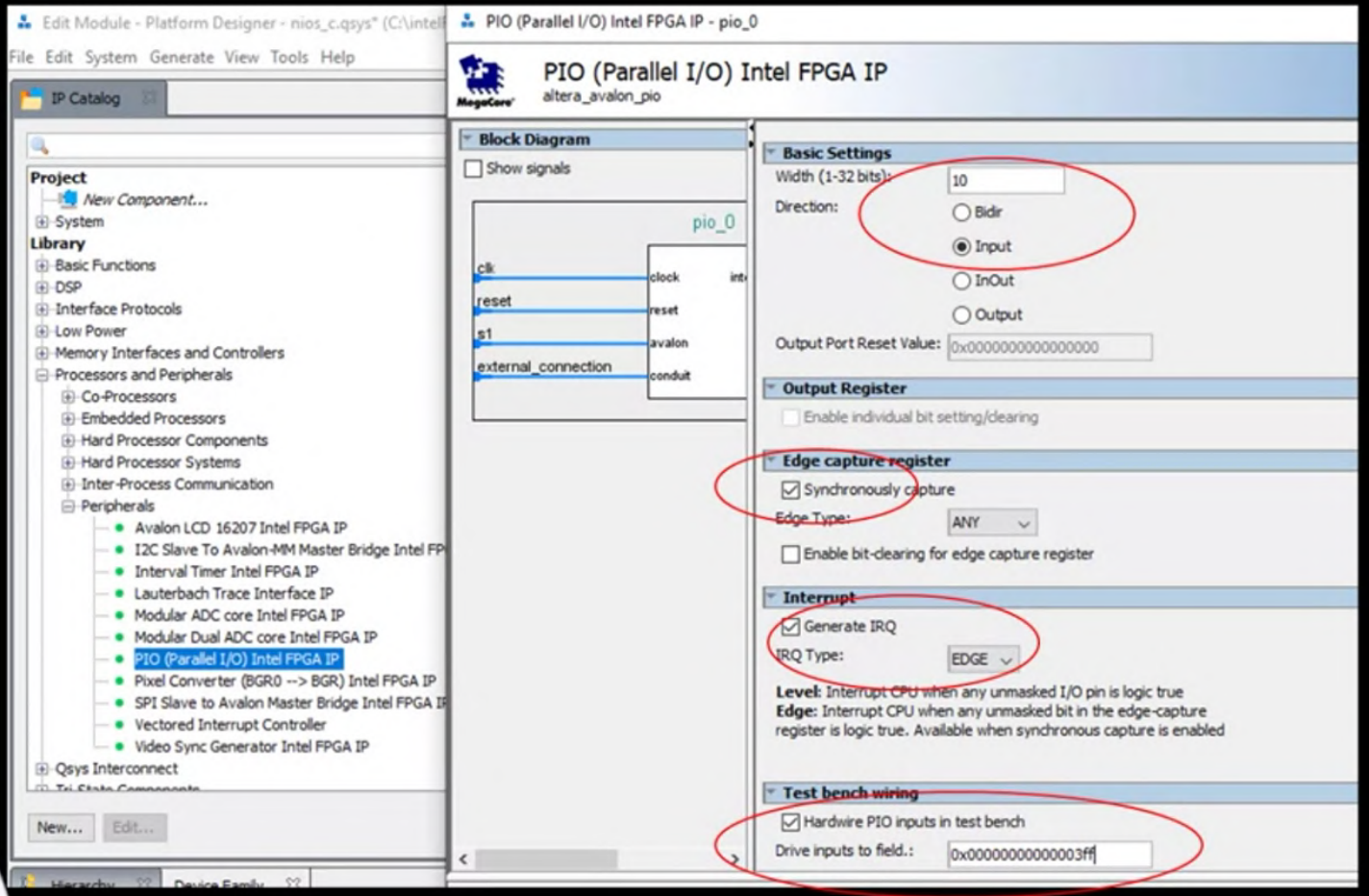
NIOS II Development :

- Platform Designer : add Parallel IO



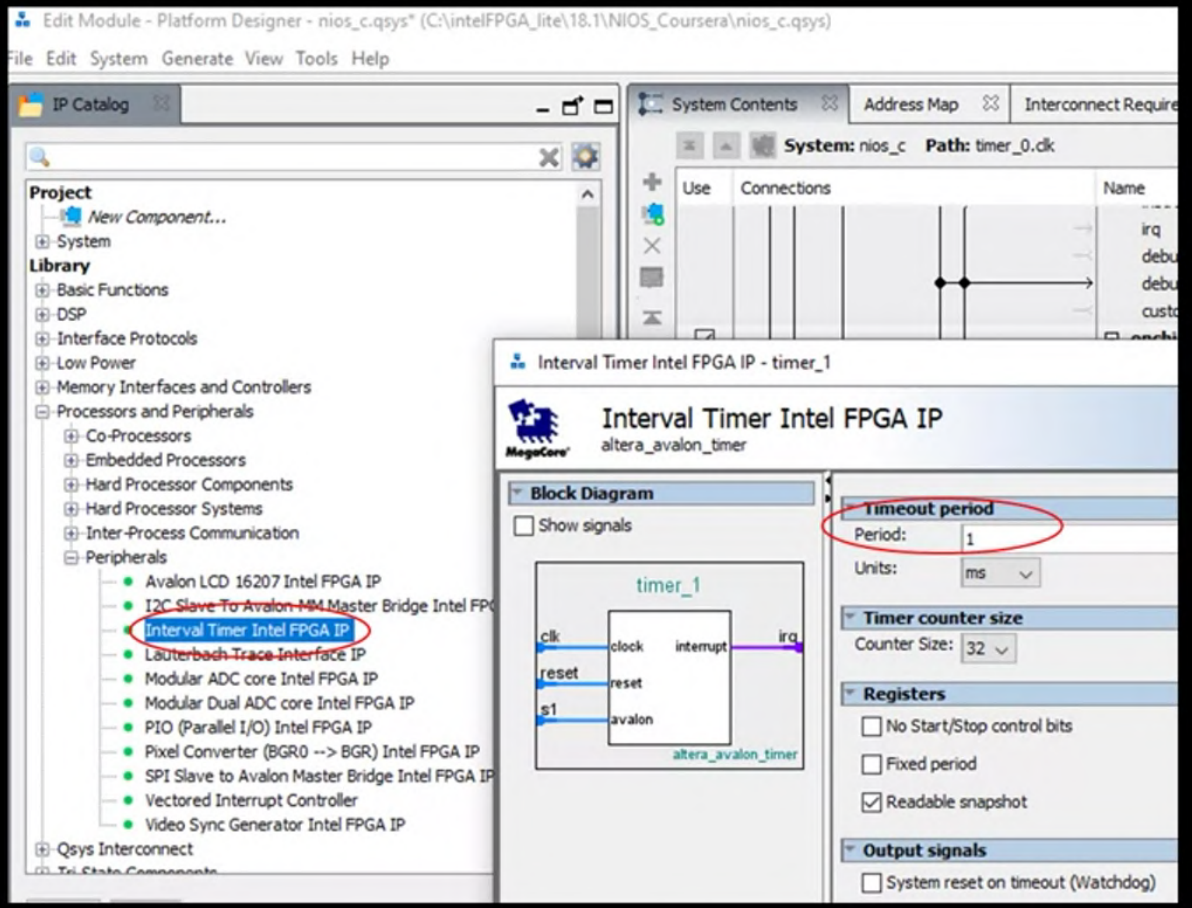
NIOS II Development :

- Platform Designer : add Slide Switches IO



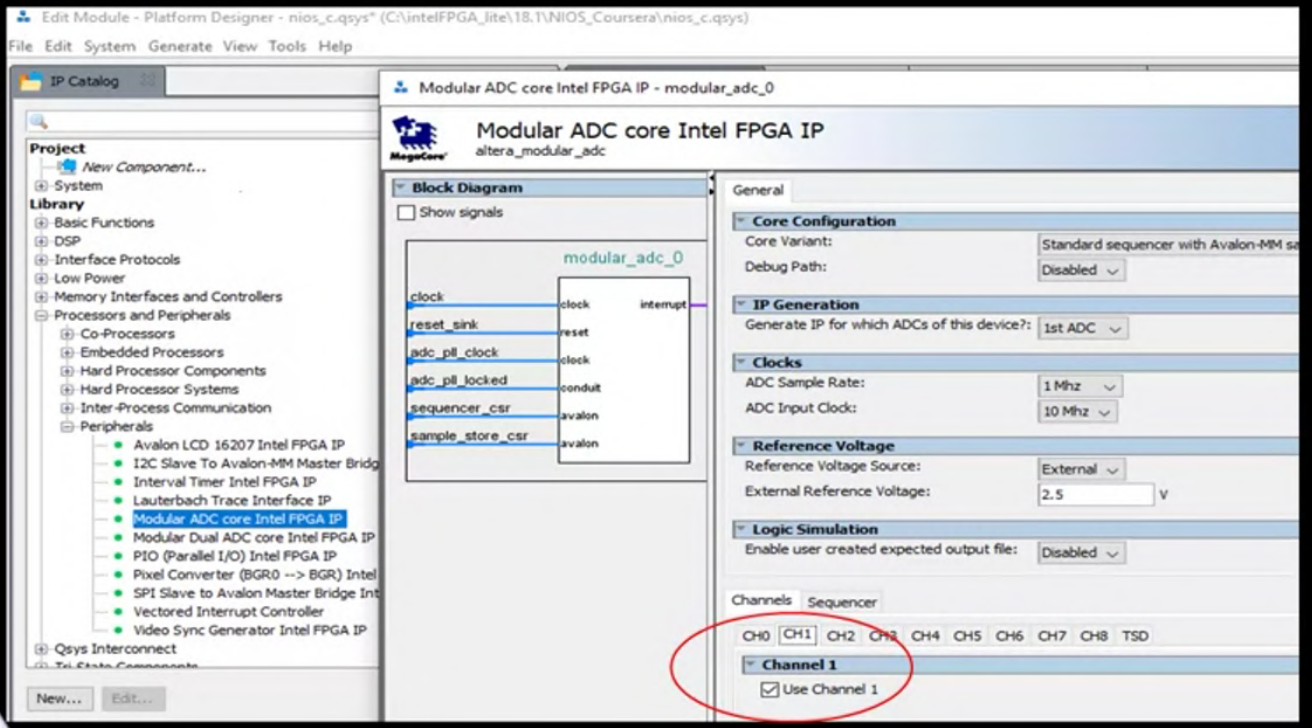
NIOS II Development :

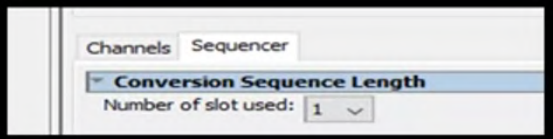
- Platform Designer : add Interval Timer



NIOS II Development :

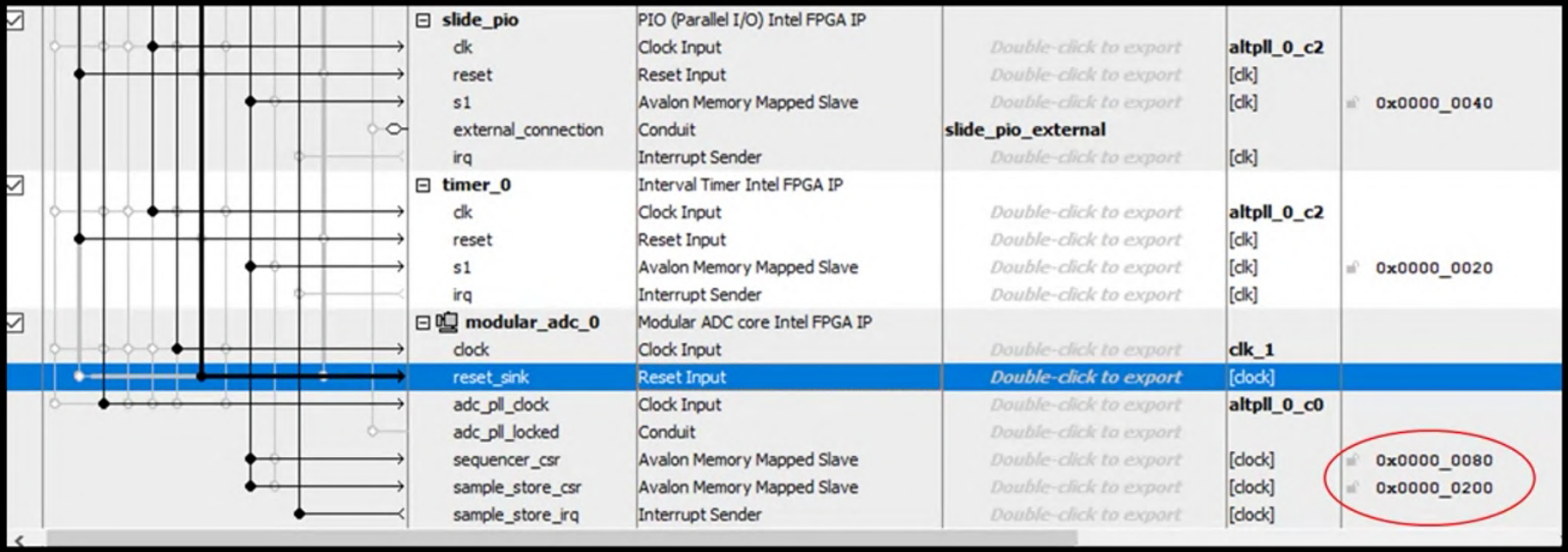
- Platform Designer : add ADC





NIOS II Development :

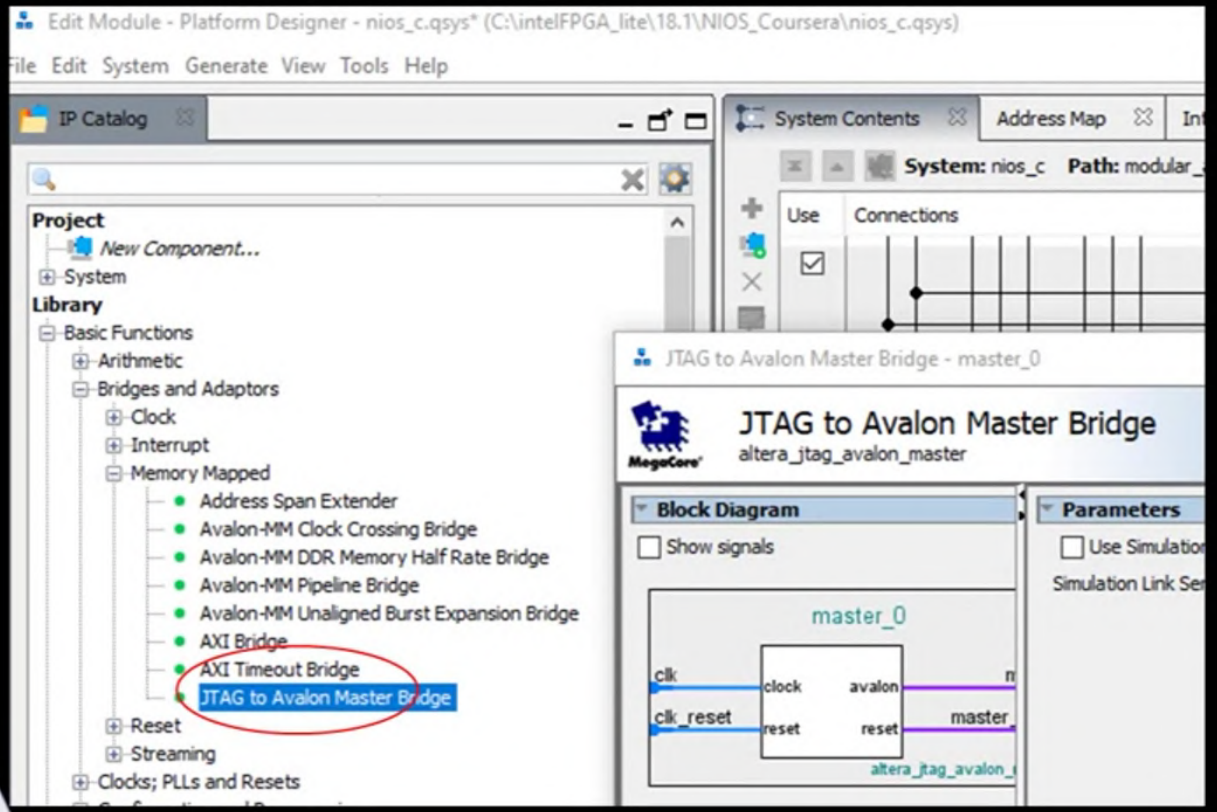
- Platform Designer : add ADC



slide_pio	PIO (Parallel I/O) Intel FPGA IP			
clk	Clock Input	Double-click to export	altpll_0_c2	
reset	Reset Input	Double-click to export	[clk]	
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0040
external_connection	Conduit	slide_pio_external		
irq	Interrupt Sender	Double-click to export	[clk]	
timer_0	Interval Timer Intel FPGA IP			
clk	Clock Input	Double-click to export	altpll_0_c2	
reset	Reset Input	Double-click to export	[clk]	
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0020
irq	Interrupt Sender	Double-click to export	[clk]	
modular_adc_0	Modular ADC core Intel FPGA IP			
clock	Clock Input	Double-click to export	clk_1	
reset_sink	Reset Input	Double-click to export	[clock]	
adc_pll_clock	Clock Input	Double-click to export	altpll_0_c0	
adc_pll_locked	Conduit	Double-click to export		
sequencer_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000_0080
sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000_0200
sample_store_irq	Interrupt Sender	Double-click to export	[clock]	

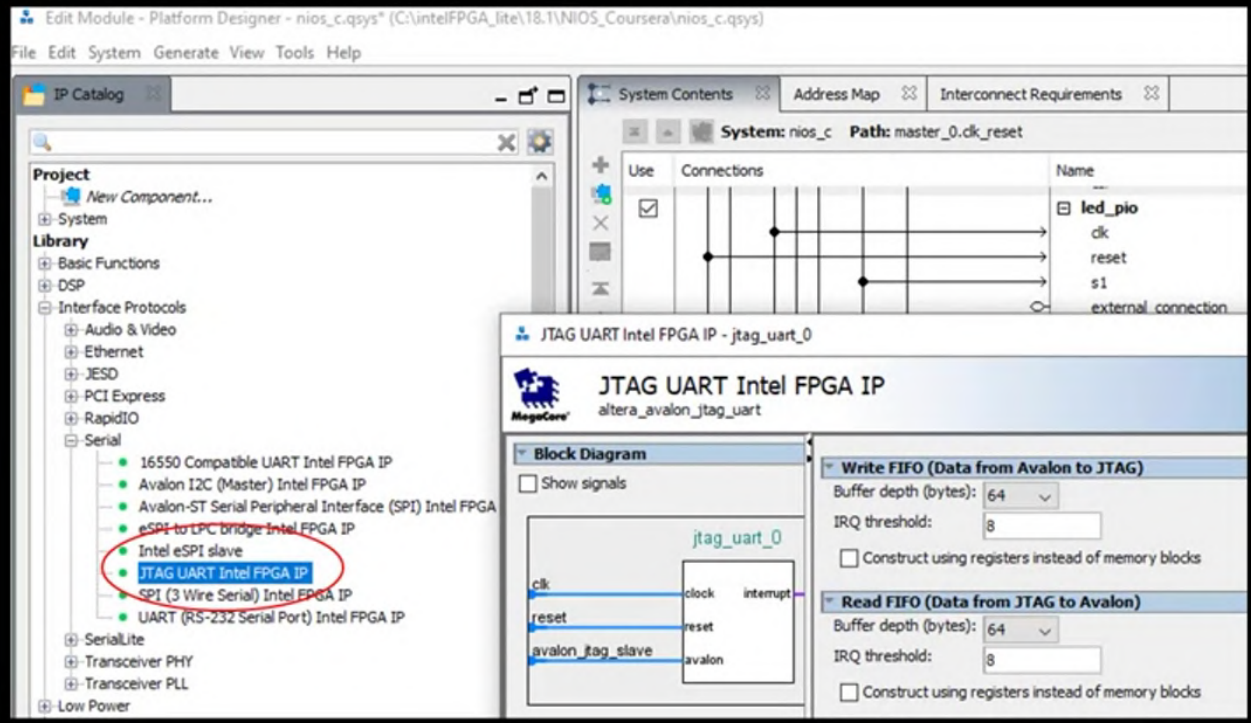
NIOS II Development :

- Platform Designer : add JTAG



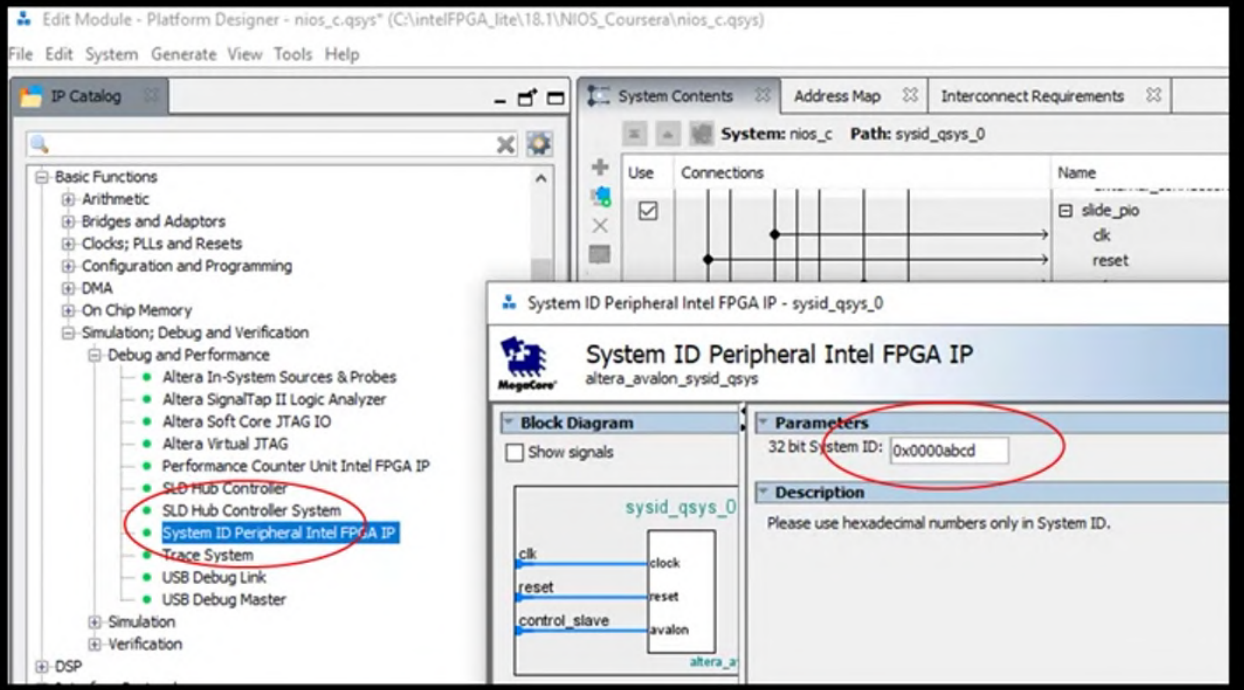
NIOS II Development :

- Platform Designer : add JTAG UART



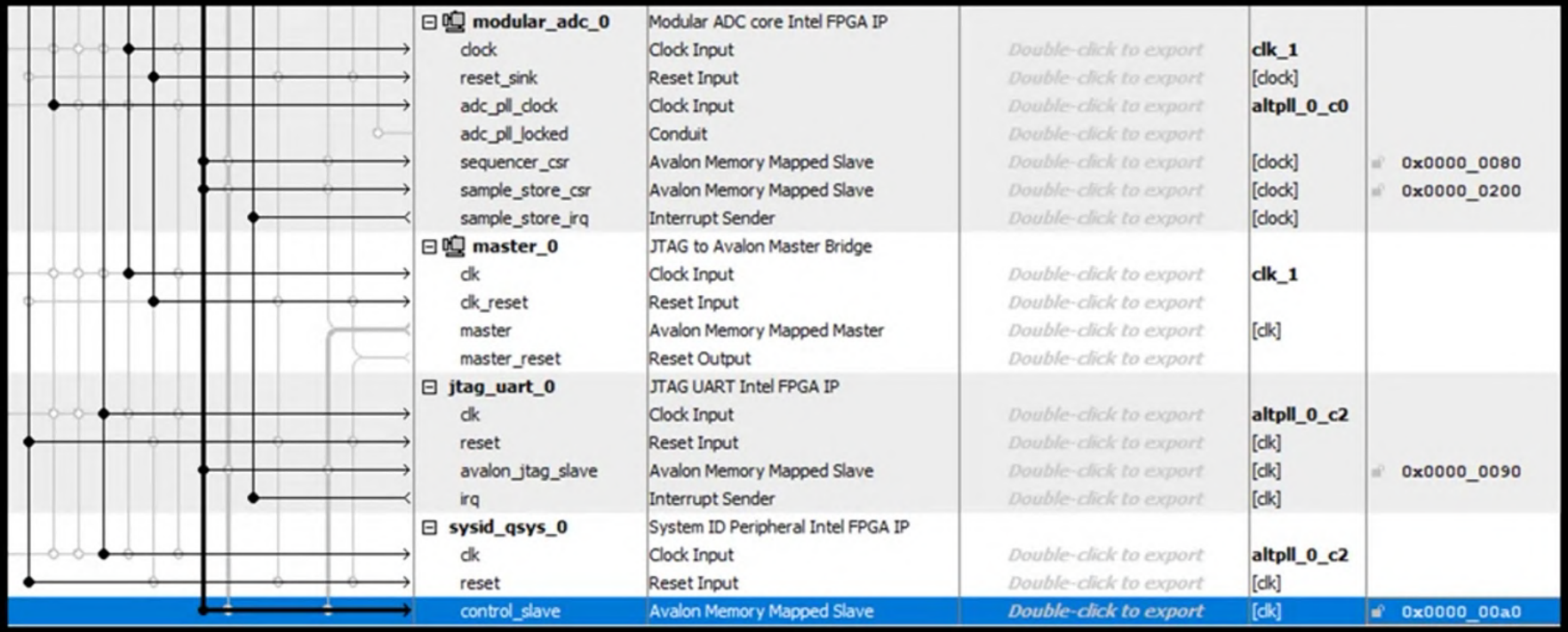
NIOS II Development :

- Platform Designer : add System ID



NIOS II Development :

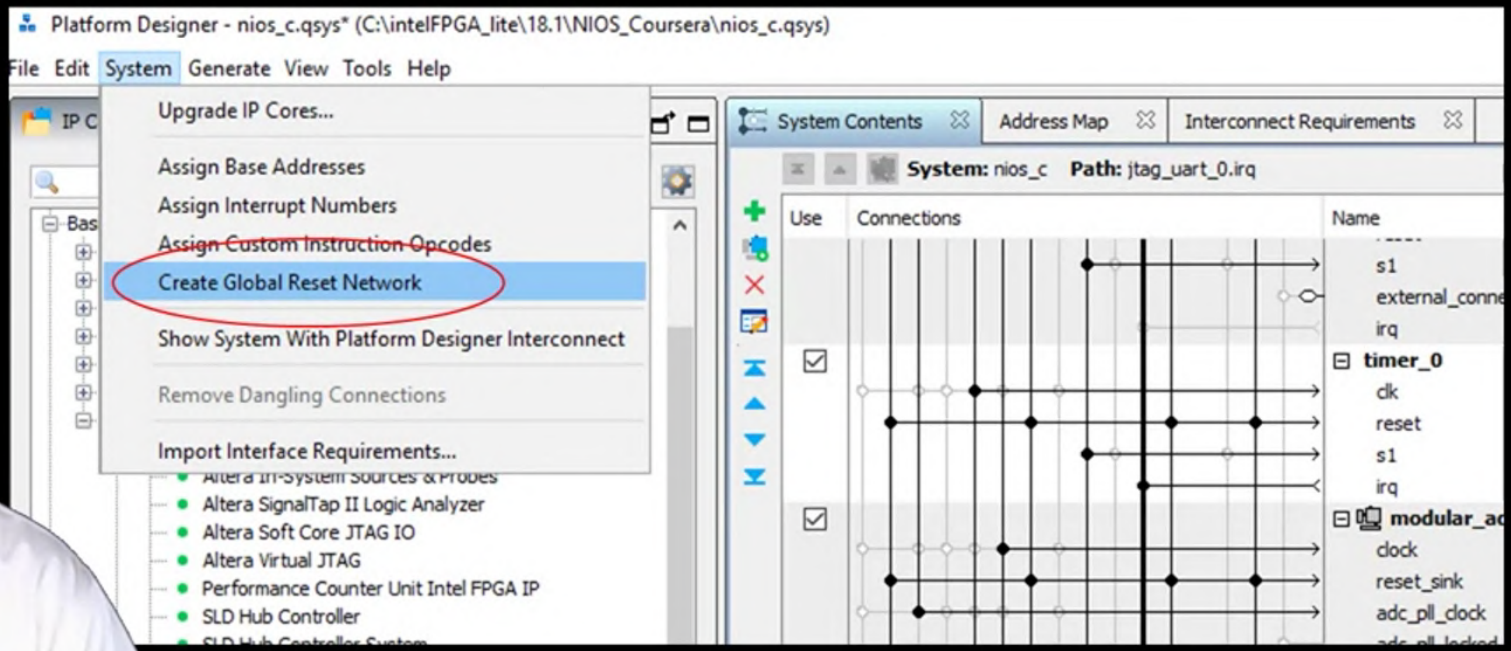
- Platform Designer : add System ID

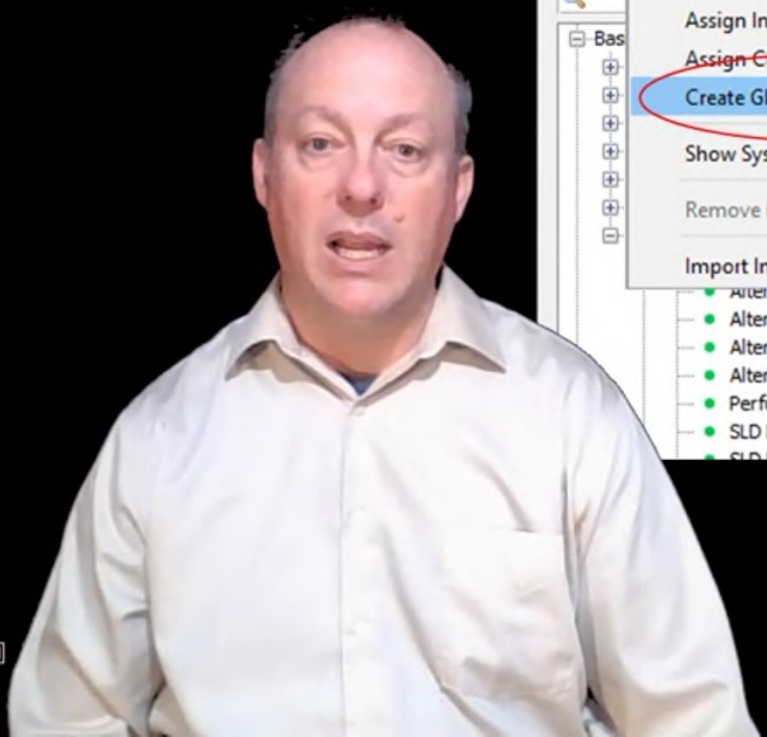


modular_adc_0	Modular ADC core Intel FPGA IP			
clock	Clock Input	Double-click to export	**clk_1**	
reset_sink	Reset Input	Double-click to export	[clock]	
adc_pll_clock	Clock Input	Double-click to export	**altpll_0_c0**	
adc_pll_locked	Conduit	Double-click to export		
sequencer_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000_0080
sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0000_0200
sample_store_irq	Interrupt Sender	Double-click to export	[clock]	
master_0	JTAG to Avalon Master Bridge			
clk	Clock Input	Double-click to export	**clk_1**	
clk_reset	Reset Input	Double-click to export		
master	Avalon Memory Mapped Master	Double-click to export	[clk]	
master_reset	Reset Output	Double-click to export		
jtag_uart_0	JTAG UART Intel FPGA IP			
clk	Clock Input	Double-click to export	**altpll_0_c2**	
reset	Reset Input	Double-click to export	[clk]	
avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0090
irq	Interrupt Sender	Double-click to export	[clk]	
sysid_qsys_0	System ID Peripheral Intel FPGA IP			
clk	Clock Input	Double-click to export	**altpll_0_c2**	
reset	Reset Input	Double-click to export	[clk]	
control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000_00a0

NIOS II Development :

- Platform Designer : Global Reset
  - System (menu) : Create Global Reset Network





Development : Summary

- NIOS II Platform Designer : System Overview

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

FPGA Soft Core processors,
and IP Acquisition.

University of Colorado **Boulder**

NIOS II Development : Compile

- Quartus : Instantiate
- File (menu) Open :
  - (double click) core (dir)
  - (select) core_inst.v
  - (click) Open

- Template to instantiate
- Select All : Copy
- Open : DE10_LITE_Golden_Top.v
- Paste core : change u0 to u3

NIOS II Development : Compile

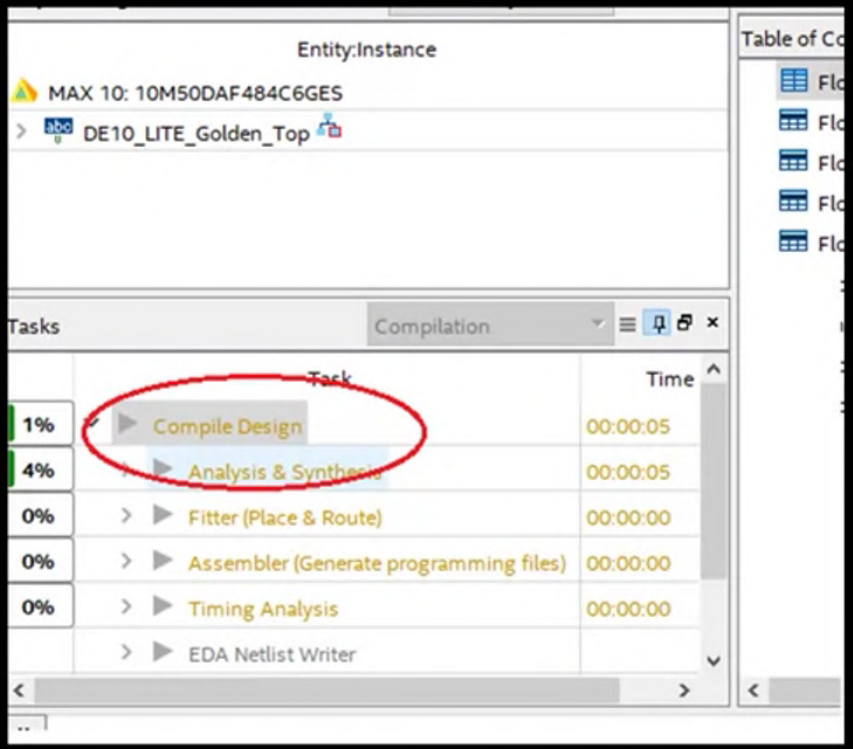
- Quartus : Instantiate

```
//=======================================================
//  Structural coding
//=======================================================
        core u3 (
                .clk_clk                        ( MAX10_CLK1_50),
                .reset_reset_n                  ( ARDUINO_RESET_N ),
                .clk_0_clk                      ( ADC_CLK_10 ),
                .reset_0_reset_n                ( ARDUINO_RESET_N ),
                .pio_0_external_export          ( LEDR  ),
                .pio_1_external_export          ( SW ),
                .altpll_0_c1_clk                (  ),
                .altpll_1_areset_conduit_export ( ARDUINO_IO[2] ),
                .altpll_1_locked_conduit_export ( ARDUINO_IO[3])
```

NIOS II Development : Compile

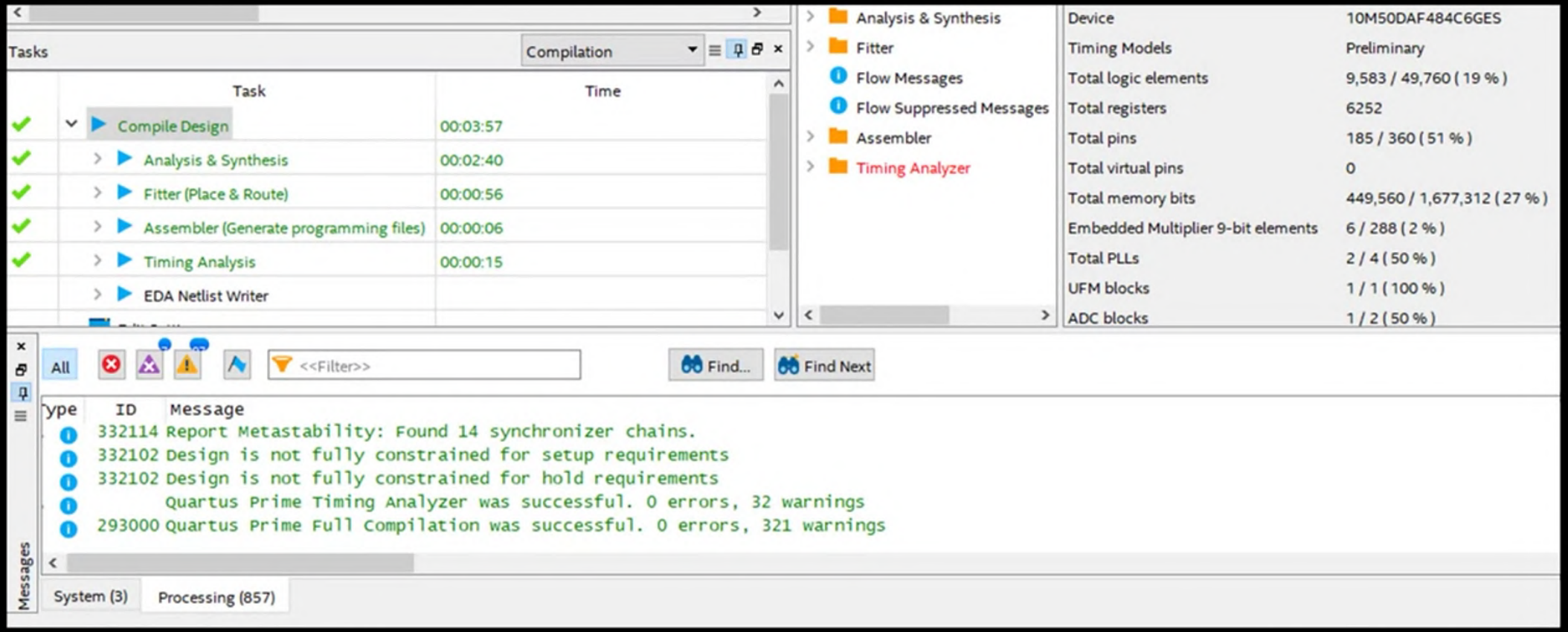
- Quartus : Instantiate
- Run an Analysis and Elaboration
  - (or Analysis & Synthesis).

- Double Click : Compile



NIOS II Development : Compile Done

- Congratulations !



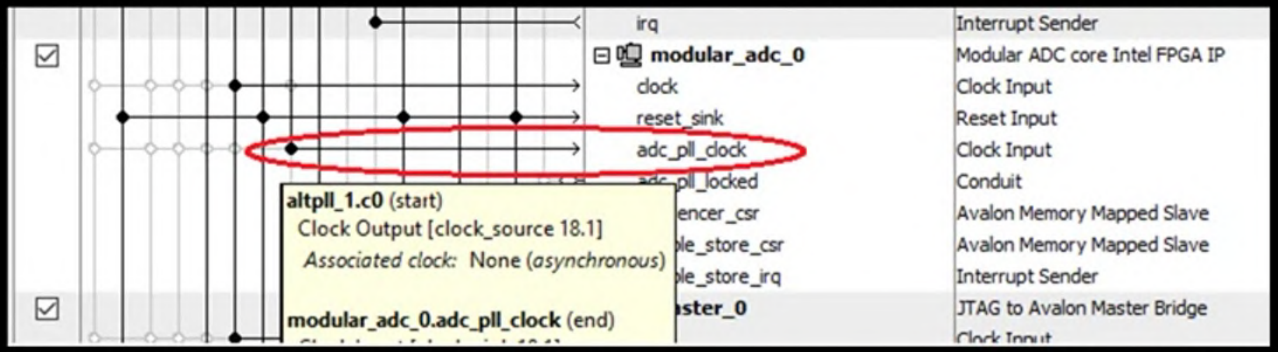
NIOS II Development : Compile

- Potential Issues (This is normal)

Note1 : Quartus 18.1 : If you see Error (14740): Configuration mode on atom "core:u3|altera_onchip_flash:onchip_flash_0|altera_onchip_flash_block:altera_onchip_flash_block|ufm_block" does not match the project setting. Update and regenerate the Qsys system to match the project setting.

Set Following :
Assignments -> Device -> Devise and Pin Options -> Configuration ->Single Uncompressed Image with Memory Initialization (512Kbits UFM)

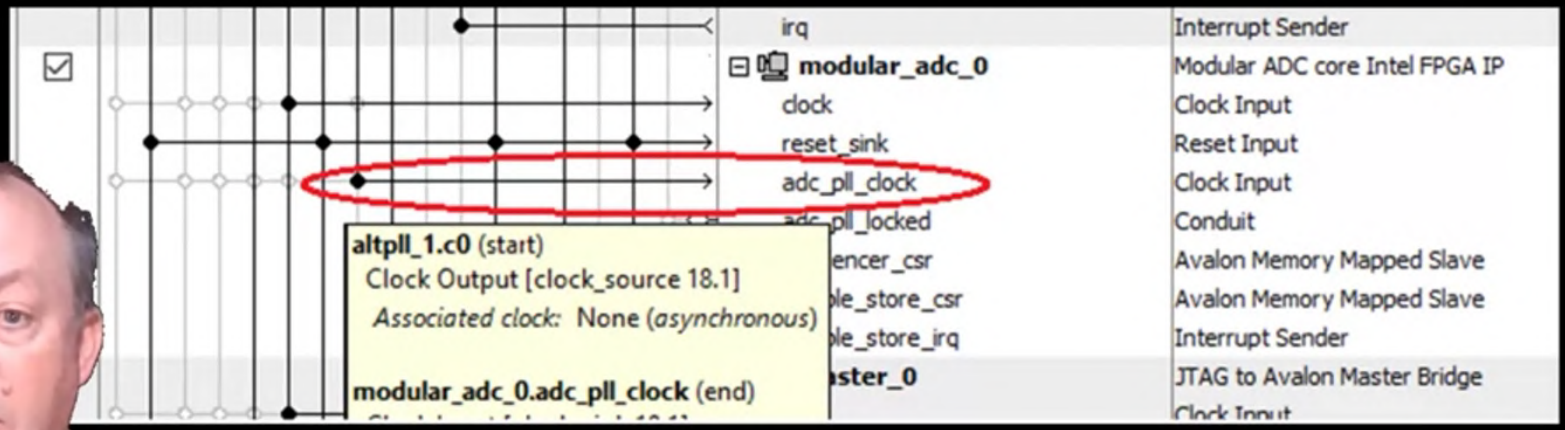
Note2 : If you see and Error due to PLL and ADC ID:170084 Can't route signal "<name>" to atom "<name>" Ensure altpll_1 c0 is connected to adc_pll_clock clock input



NIOS II Development : Compile

- Potential Issues (This is normal)

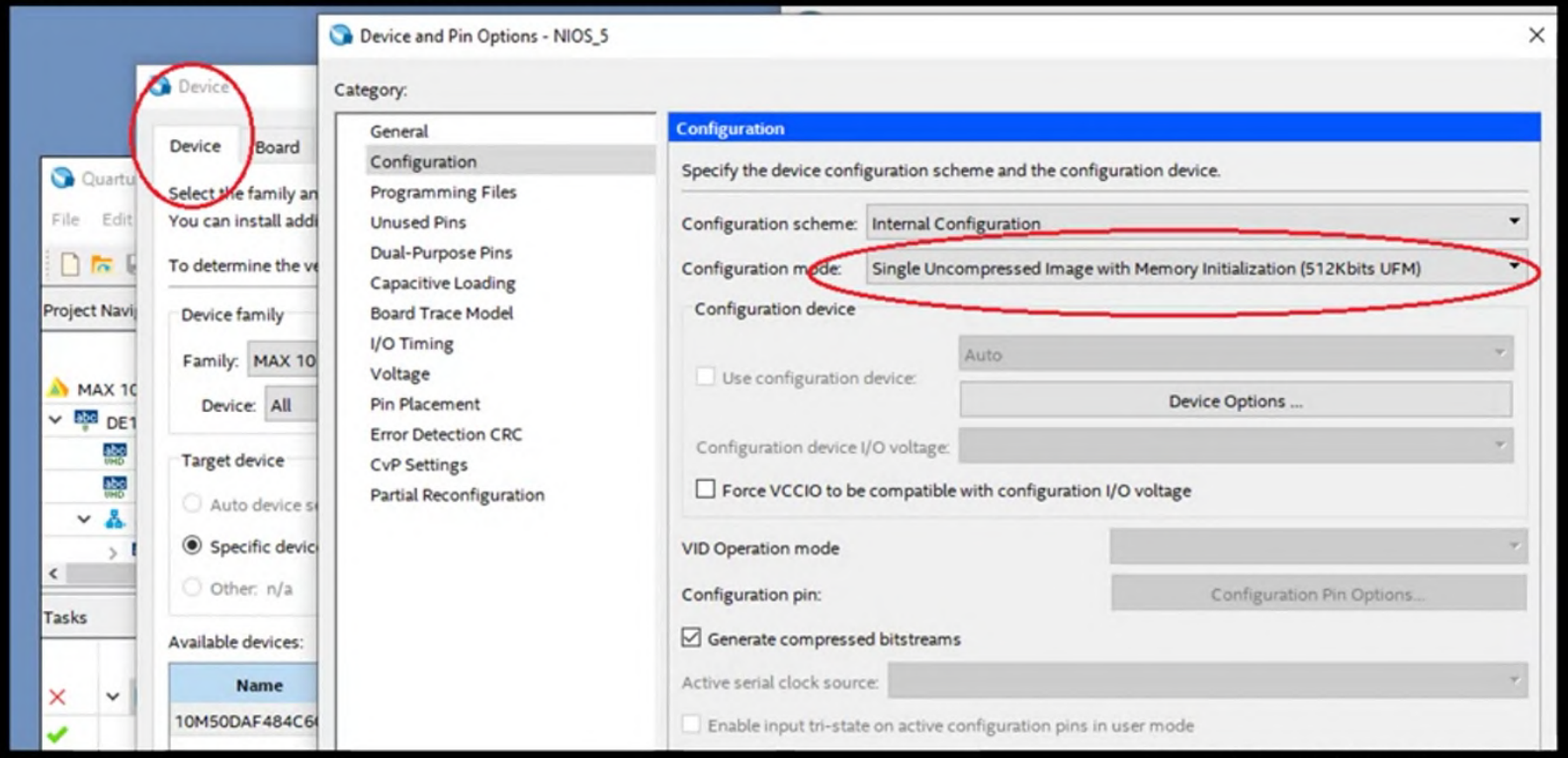
Note2 : If you see and Error due to PLL and ADC ID:170084 Can't route signal "<name>" to atom "<name>" Ensure altpll_1 c0 is connected to adc_pll_clock clock input



NIOS II Development : Compile

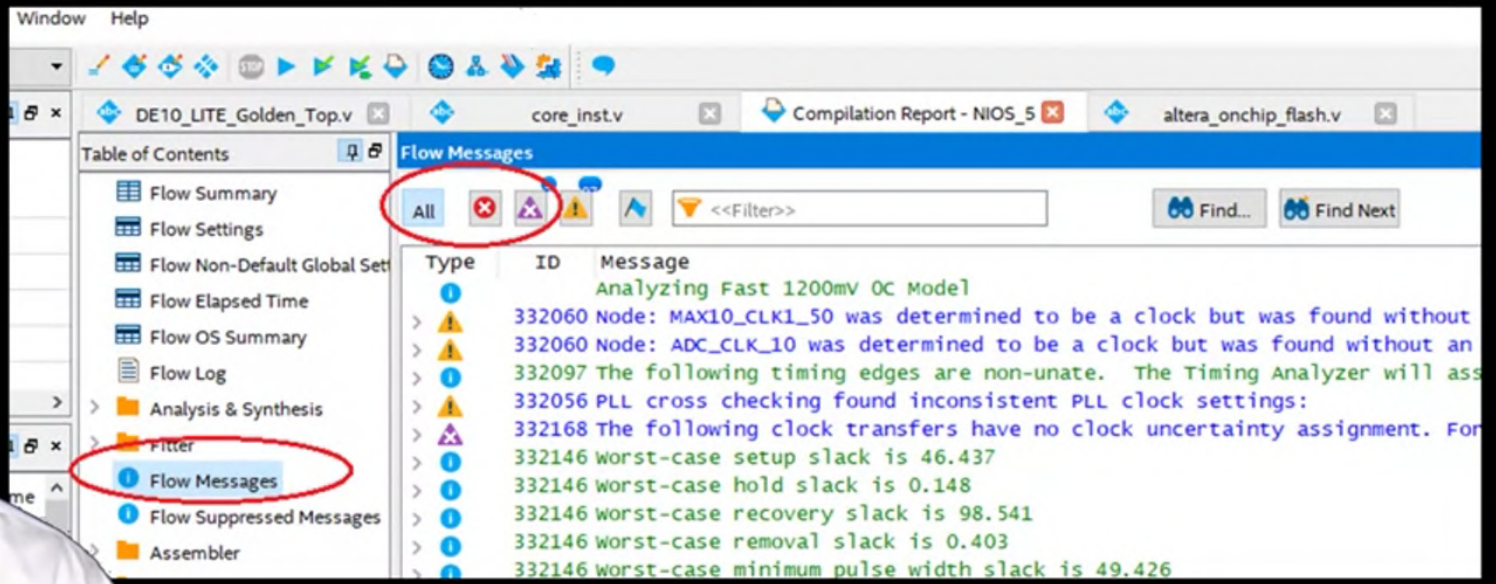
- Potential Issues (This is normal)

Set Following :
Assignments -> Device -> Devise and Pin Options -> Configuration ->Single Uncompressed Image with Memory Initialization (512Kbits UFM)



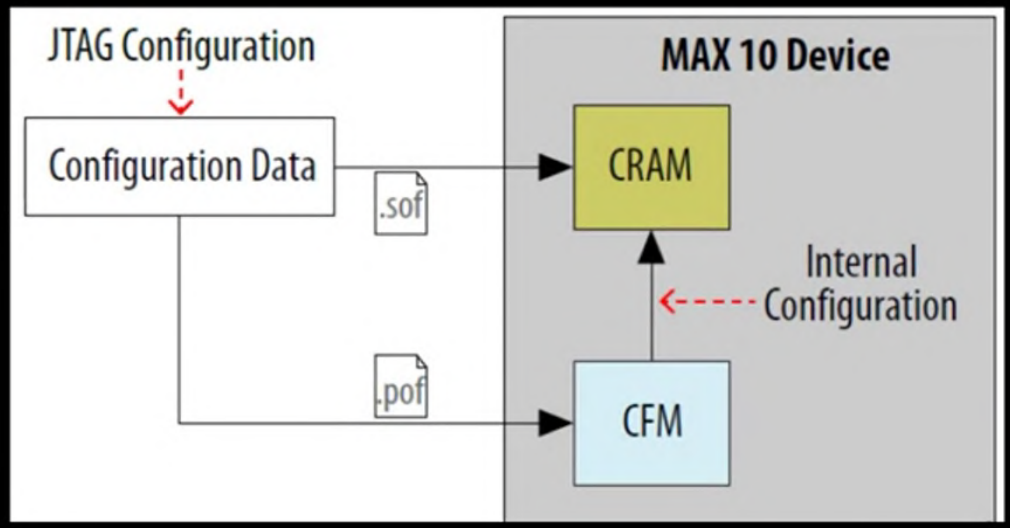
NIOS II Development : Messages

- Quartus : Full Compilation
- SDC : constraints added
- Flow Messages : All, Error, Critical



NIOS II Development : Programming

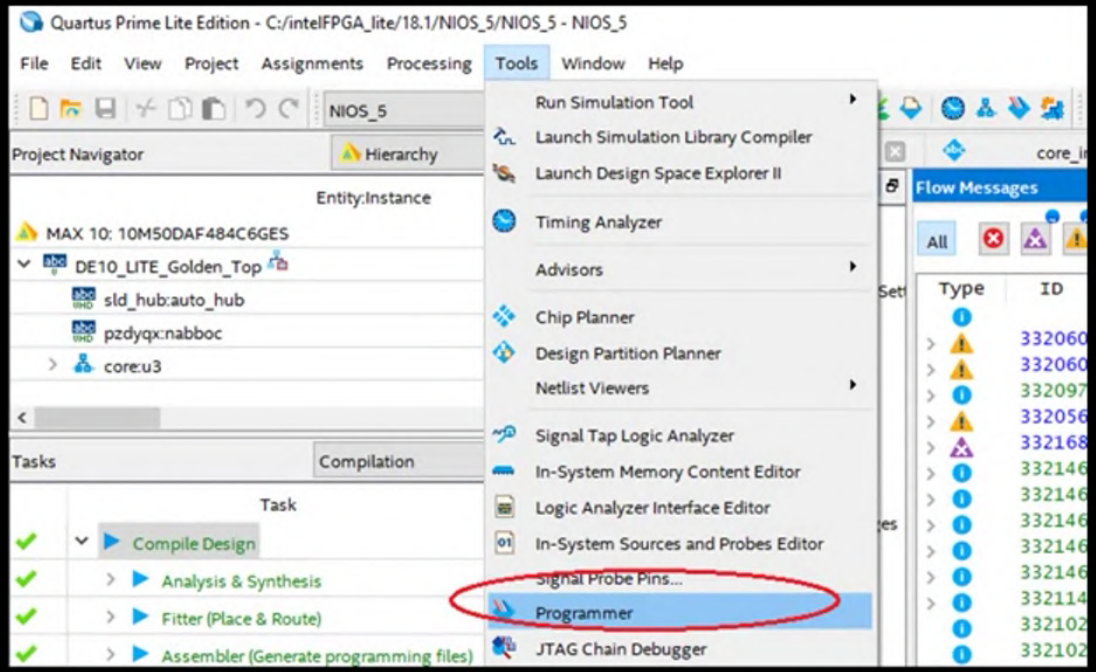
- MAX 10 : Flash
- JTAG : USB Blaster
  - Program .sof file to SRAM configuration cells
    - Temporary
  - Program .pof file to Flash
    - Loads SRAM at each Power ON



NIOS II Development : Programming

- Quartus

1) Connect USB to DE10 Lite Board and Computer
2) Tools (menu) : Programmer



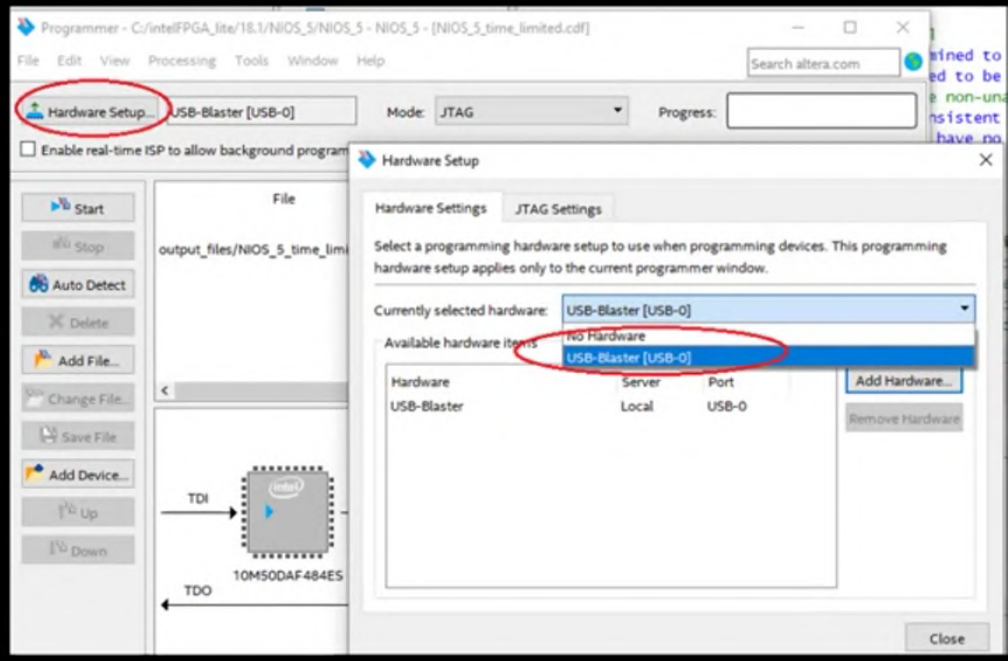
NIOS II Development : Programming

- Quartus

3) Hardware Setup… (Button)

4) USB Blaster [USB-0] : Close

5) Install Drivers if USB-0 not seen

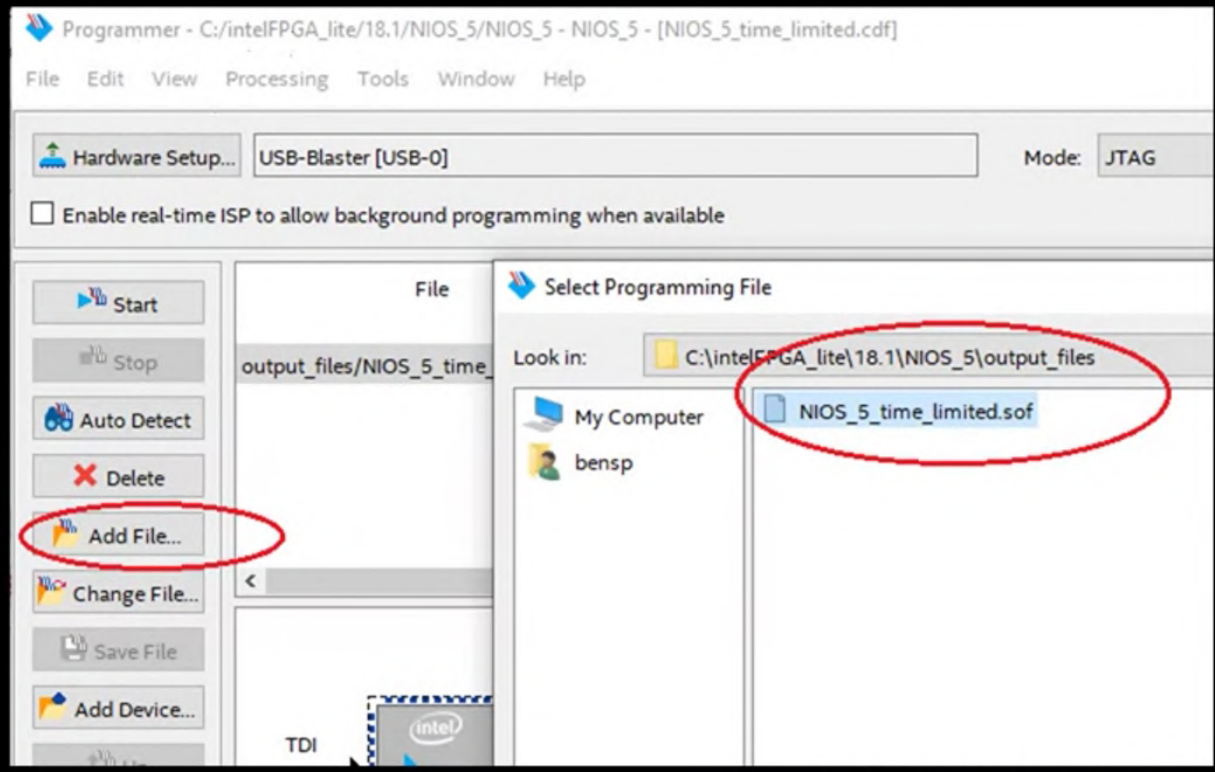


NIOS II Development : Programming

- Quartus

6) Add File… (Button)

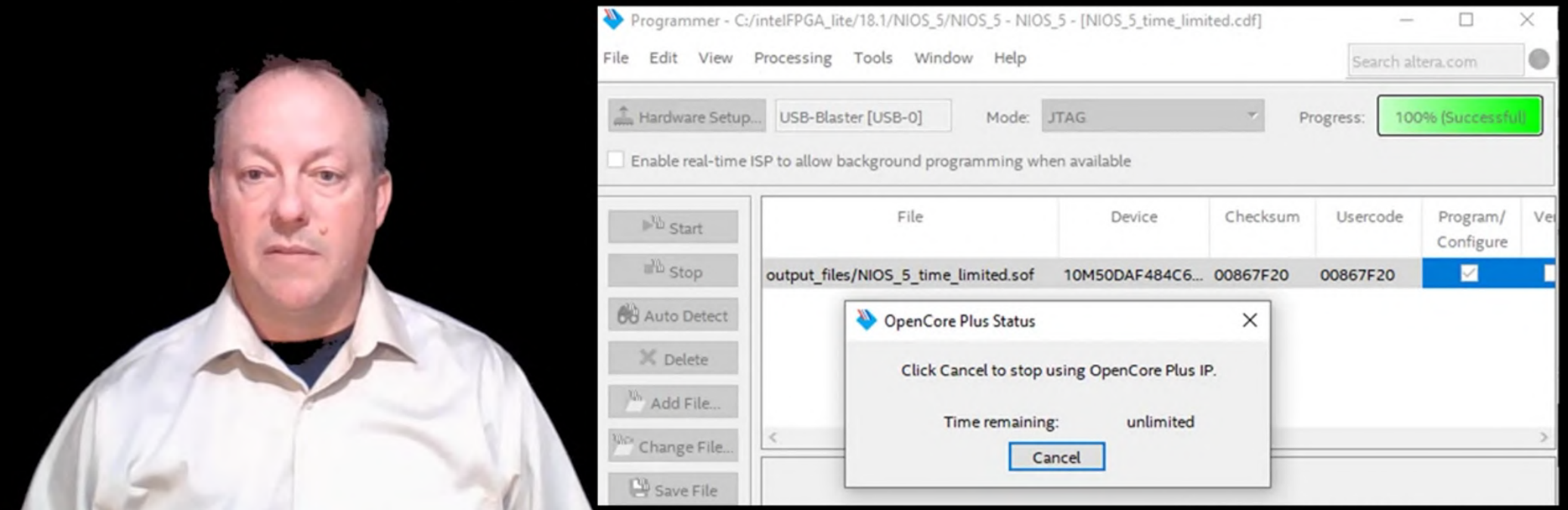
7) Start



NIOS II Development : Programming

7) Start (Button) - - > 100% Successful

Note : do not press “Cancel” until you are done using the NIOS core. Time_limited is .sof file only, no Flash without License



Development : Summary

- NIOS II Platform Designer : Compile
- NIOS II Platform Designer : Program

FPGA Design for Embedded Systems

FPGA Softcore Processors and IP Acquisition

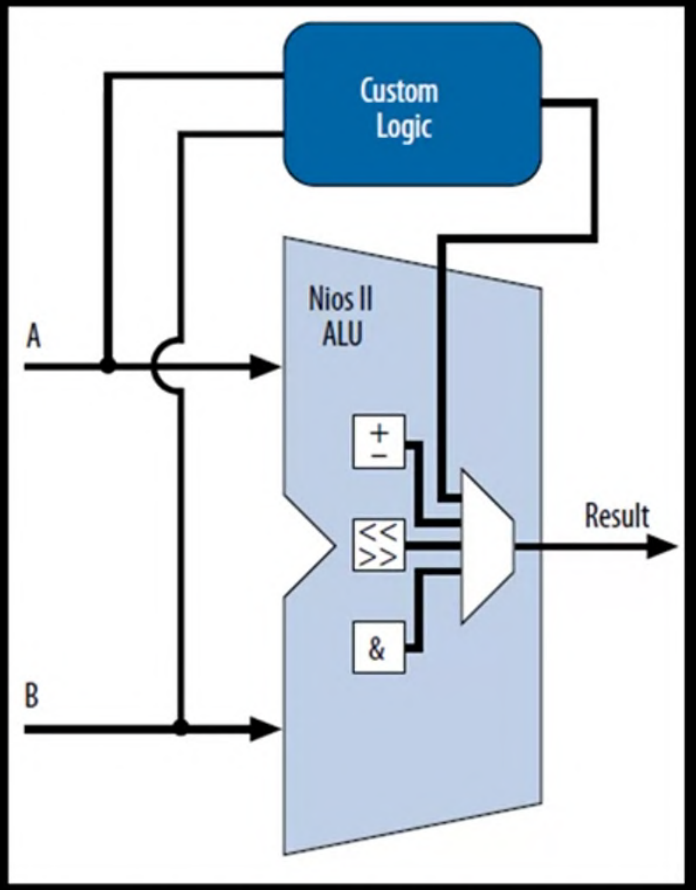
University of Colorado **Boulder**

Soft Processor : Customization

- Custom Instruction Implementation
- Build the Custom Example Hardware
  - Component Editor
  - Platform Designer

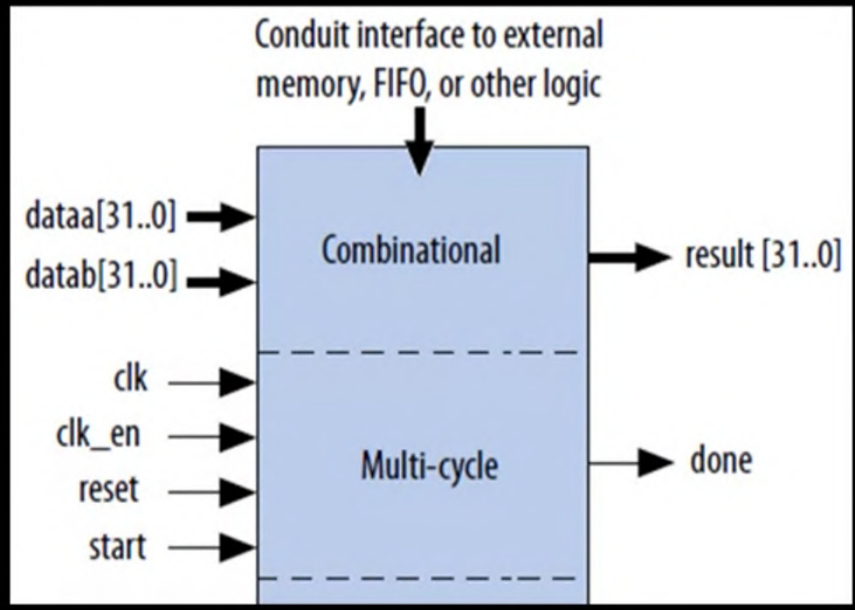
Soft Processor : Customization

- NIOS II ALU
  - Accelerate time critical software algorithms by converting them to custom hardware logic blocks
  - Provide an easy way to experiment with hardware-software tradeoffs



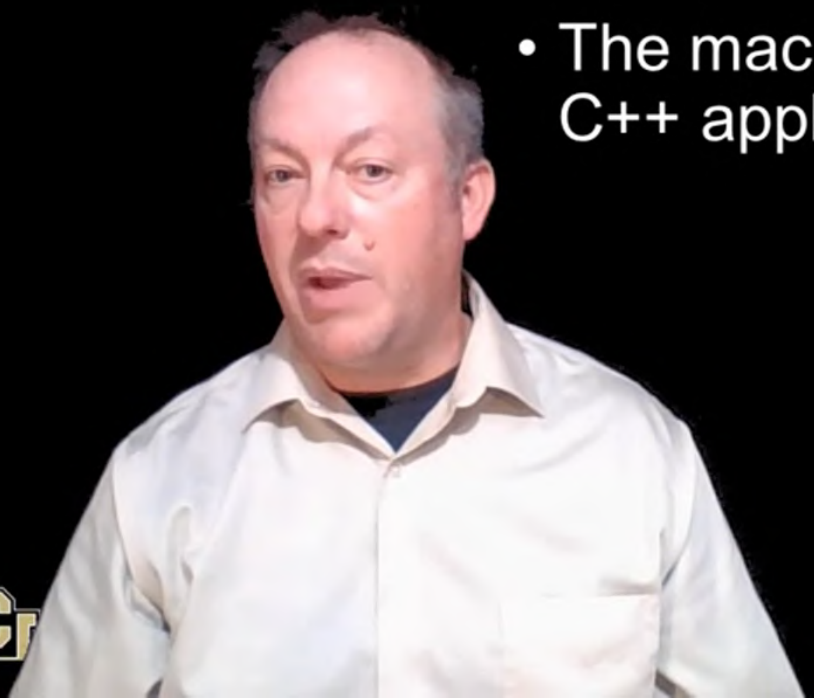
Soft Processor : Customization

- NIOS II ALU
  - Custom instruction logic block interfaces with the Nios II processor through three ports:
    - dataa
    - datab
    - result



Soft Processor : Customization

- NIOS II ALU
- Each custom instruction in the Embedded Design Suite (EDS) generates a macro in the system header file system.h
- The macro can be used directly in the C or C++ application code



Soft Processor : Customization

The Nios II custom instruction hardware tasks:

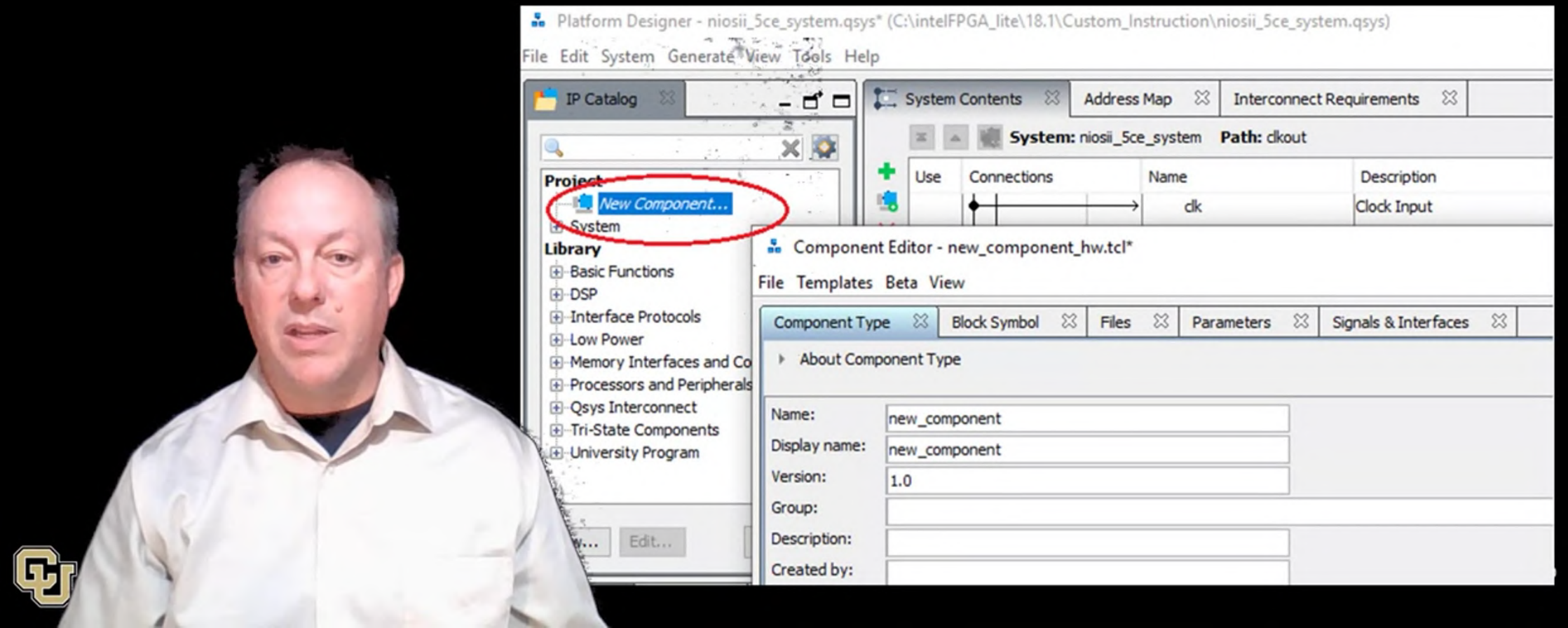
1. Open the Component Editor
2. Specify the custom instruction component type
3. Display the custom instruction block symbol
4. Add the HDL files
5. Configure the custom instruction parameter type
6. Set up the custom instruction interfaces
7. Configure the custom instruction signal type
8. Save and add the custom instruction
9. Generate the system and compile in the Quartus software

Soft Processor : Customization

- Tools (menu): Platform Designer
- Platform Designer :
  - File (menu) : Open <nios_file>.qsys

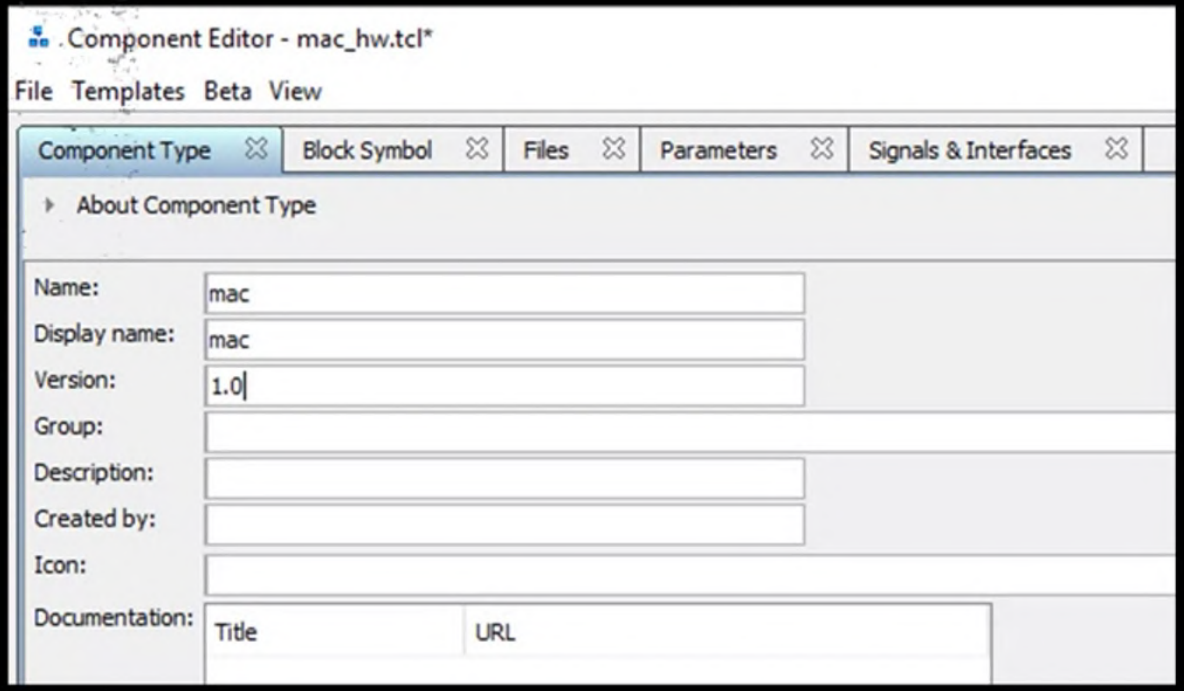
Soft Processor : Customization

- Platform Designer :
  - New Component (double click) : Component Editor



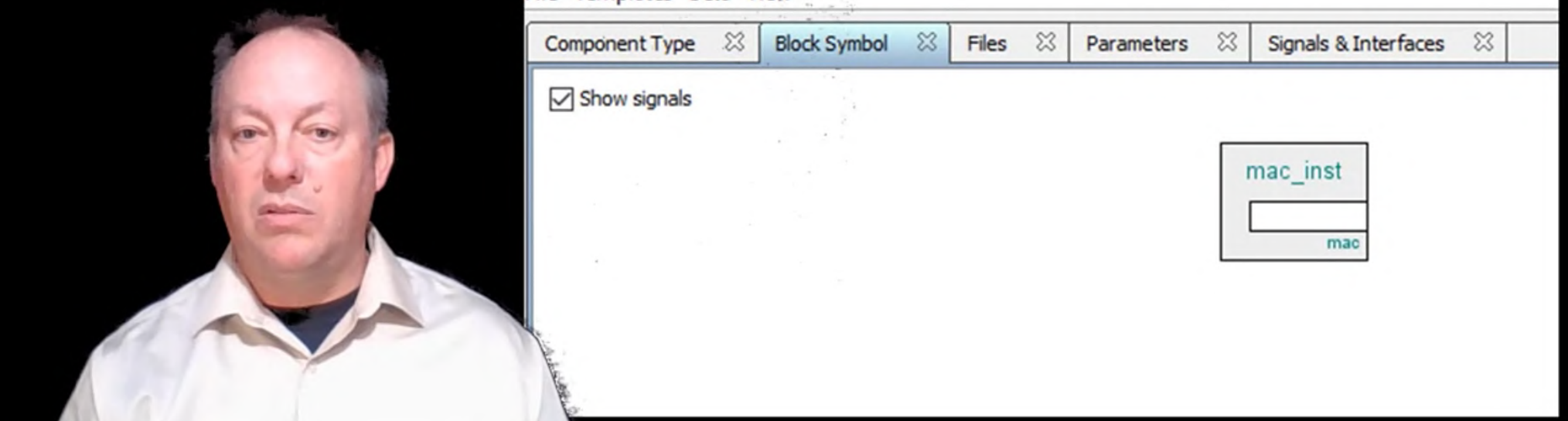
Soft Processor : Custom Instruction

- Component Editor : Component Type (tab)
  - Name : mac
  - Display name : mac
  - Version : 1.0



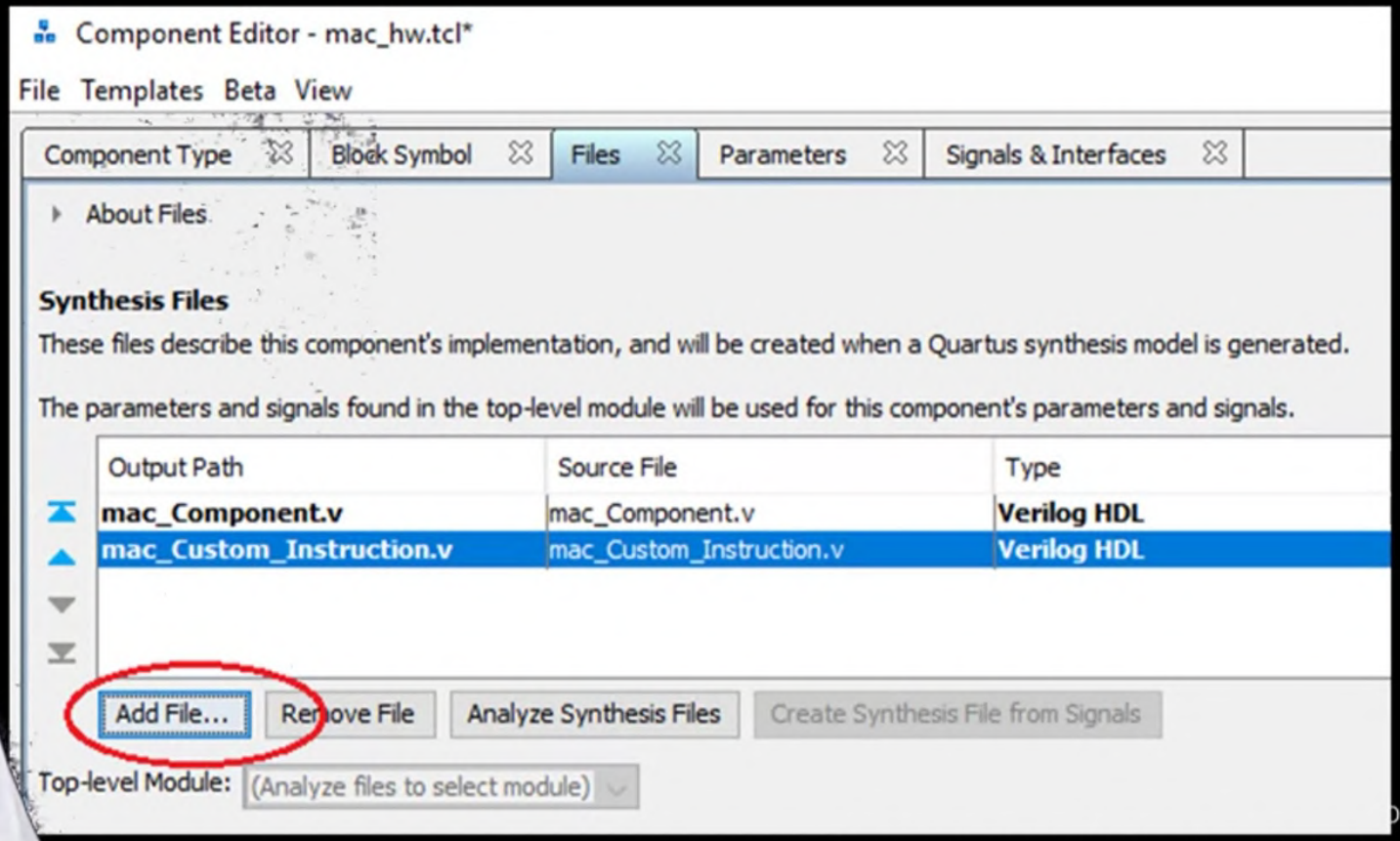
Soft Processor : Custom Instruction

- Component Editor : Block Symbol (tab)
  - View



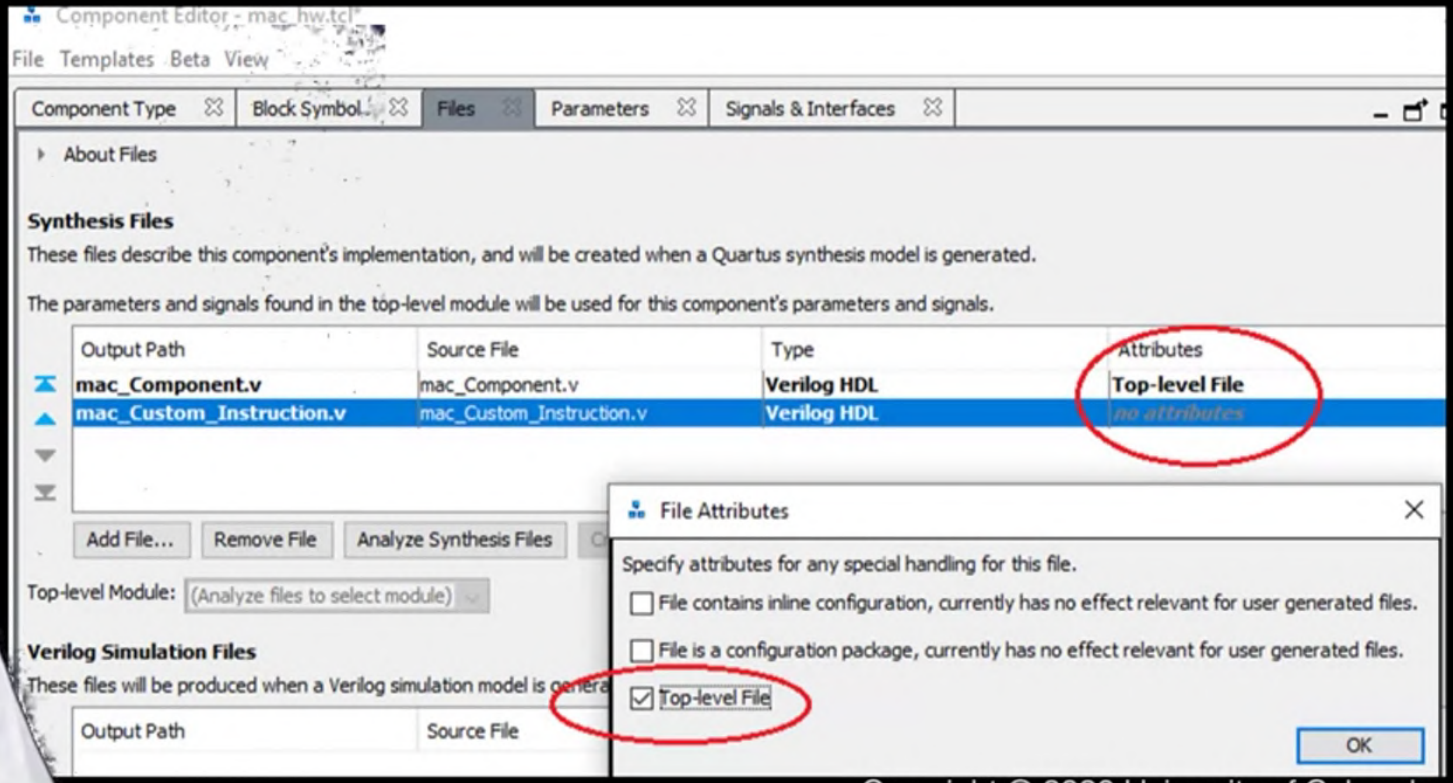
Soft Processor : Custom Instruction

- Component Editor : Files (tab)
  - Add



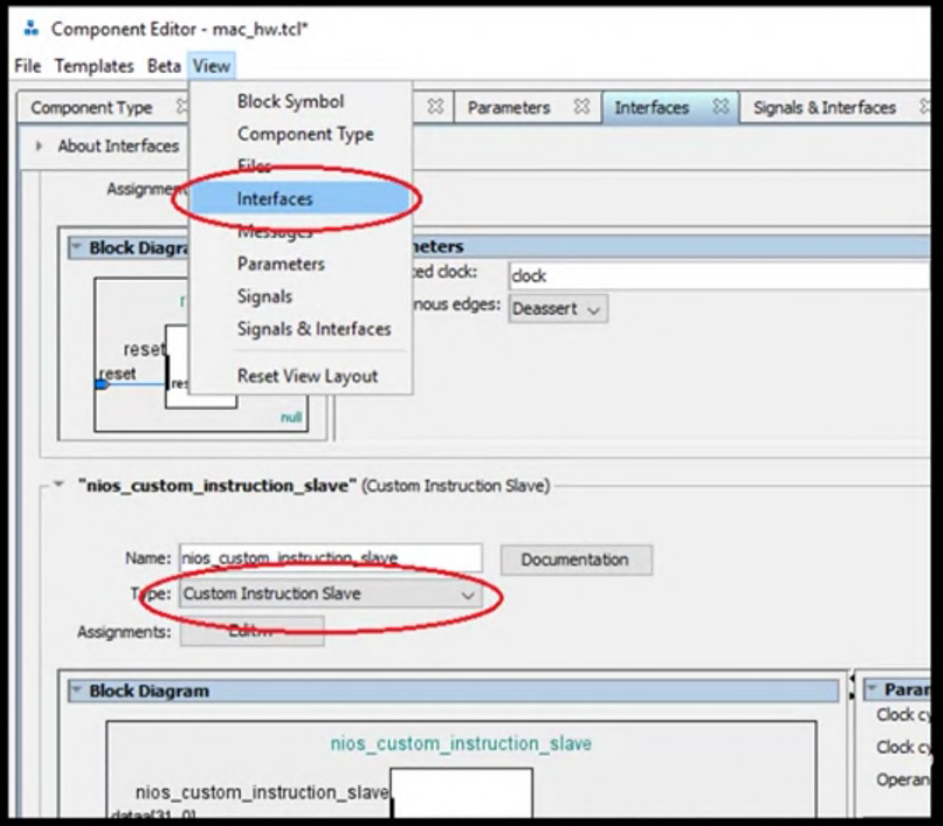
Soft Processor : Custom Instruction

- Component Editor : Files (tab) : Attributes (column)
  - Select : Top-level File : OK



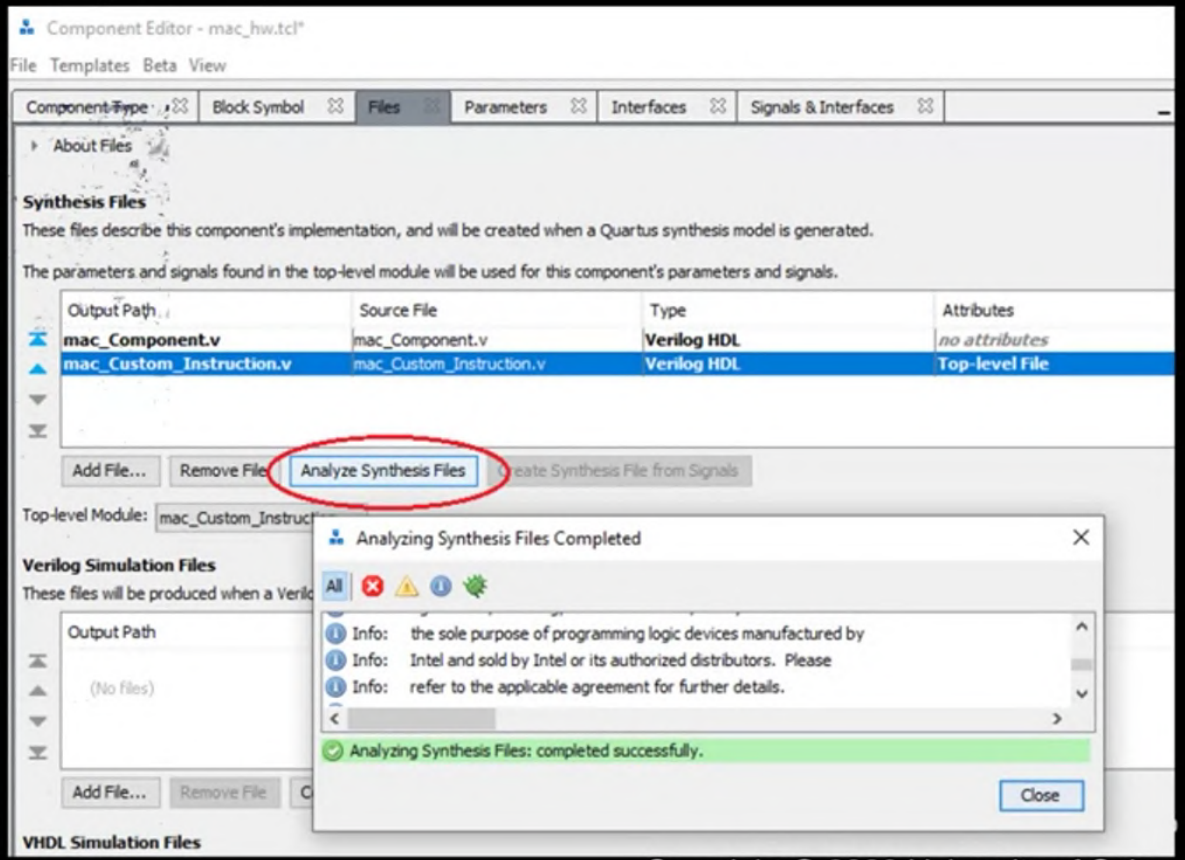
Soft Processor : Custom Instruction

- Component Editor : Interface Type
  - View (menu) : Interfaces
  - Type : Custom Instruction Slave



Soft Processor : Custom Instruction

- Component Editor :
  - Analyze Synthesis Files (button)

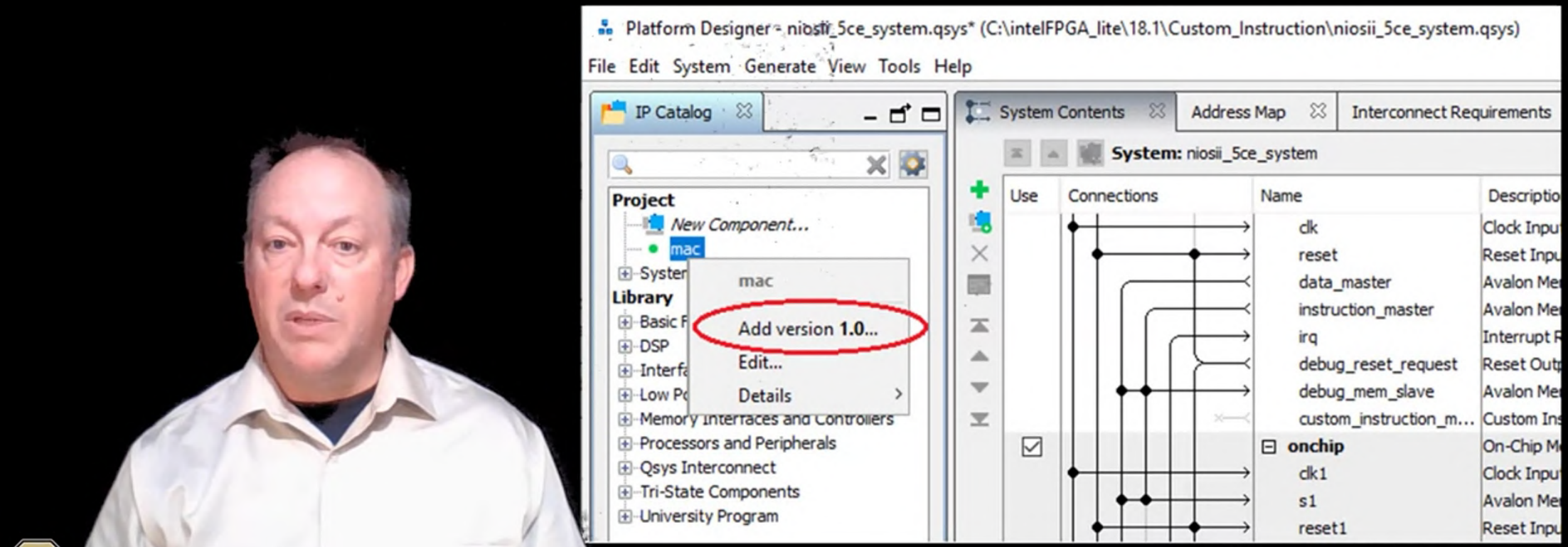


Soft Processor : Custom Instruction

- Component Editor :
  - Finish : Yes, Save.

Soft Processor : Custom Instruction

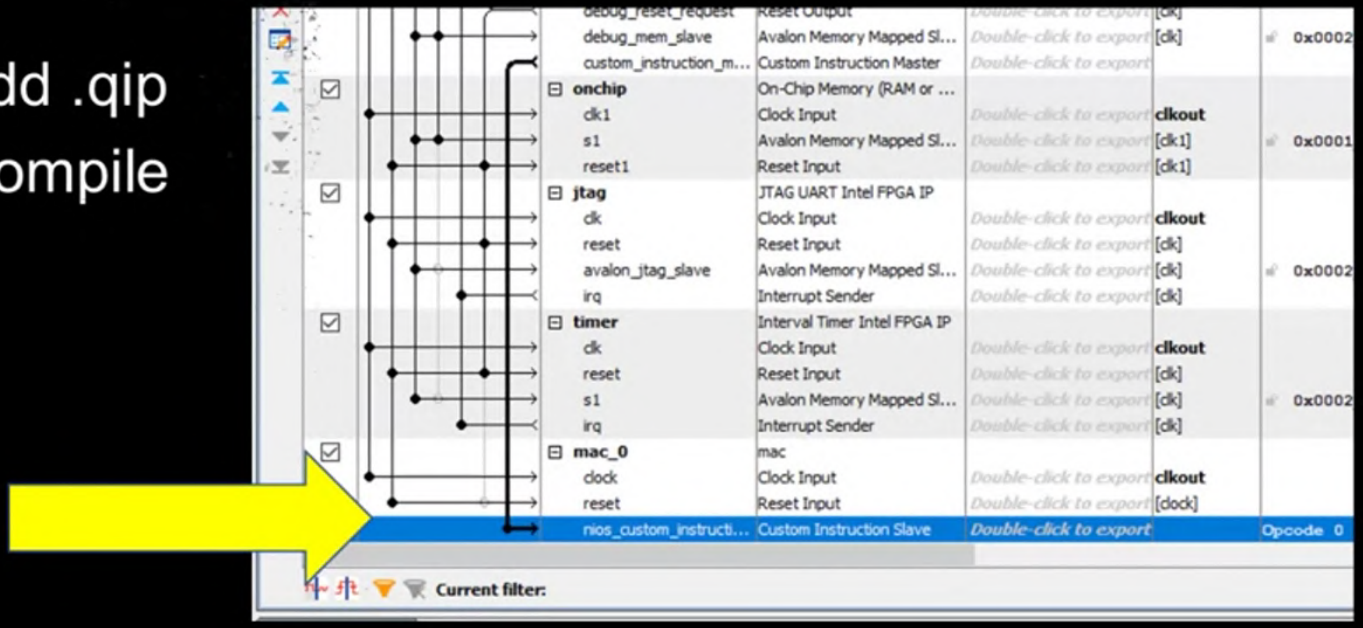
- Platform Designer : IP Catalog : mac
  - mac : (right click) Add version 1.0…



Soft Processor : Custom Instruction

- Platform Designer : mac_0
  - Connect the clock, reset, and instruction nodes
- Generate (menu) : Generate HDL

- Quartus : Add .qip
- Quartus : Compile



Soft Processor : Custom Instruction

- Software Build : References

Custom Instruction User Guides :

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_nios2_custom_instruction.pdf

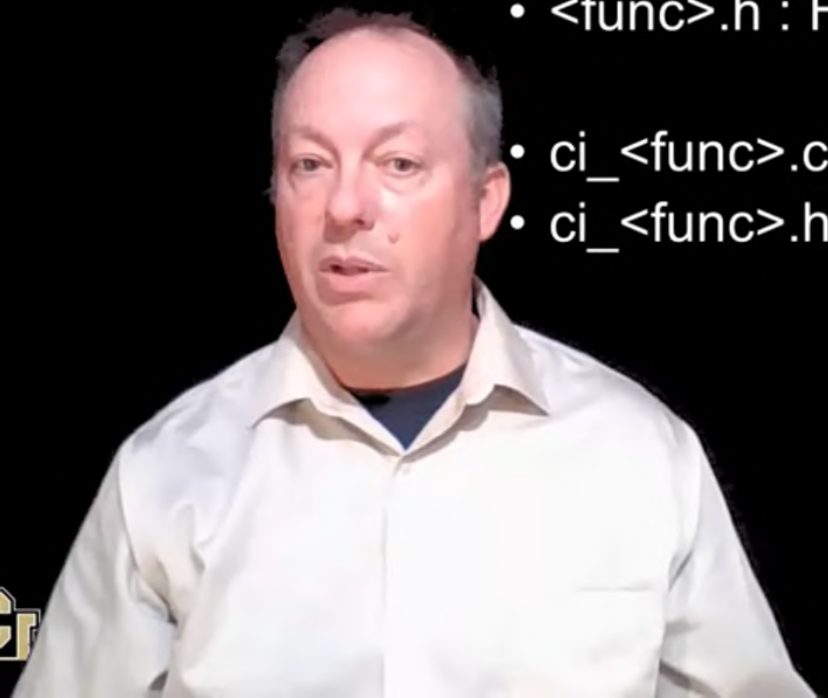
https://www.intel.com/content/www/us/en/programmable/documentation/cru1439932898327.html

Design Example :
https://www.intel.com/content/dam/altera-www/global/en_US/others/support/examples/download/ug_custom_instruction_files.zip

Soft Processor : Custom Instruction

- Software Build : files
  - <func>_main.c : Main program that populates random test data, executes the <func> in software with the custom instruction, validates the output, and reports the processing time.
  - <func>.c : Software <func> algorithm run by the Nios II processor
  - <func>.h : Header file for <func>.c

- ci_<func>.c : Program that accesses <func> custom instruction
- ci_<func>.h : Header file for ci_<func>.c



Soft Processor : Custom Instruction

- Software Build : Run
  - To run the application software, create an Executable and Linking Format File (.elf) first.
  - To create the .elf file, follow the instructions in the "Nios II Software Build Flow" section in the readme_qsys.txt file in the extracted design files.

  - Macro that is defined in the ci_<func>.c file.

    ```
    #define <func>_CI_MACRO(n, A) \
    __builtin_custom_ini(ALT_CI_<func>_CUSTOM_COMPONENT_0_N + (n & 0x7), (A))
    ```

  - This macro accepts a single int type input operand and returns an int type value.

  - Upcoming series Step into the Software process in more depth

Soft Processor : Customization

- Custom Instruction Implementation
- Build the Custom Example Hardware
  - Component Editor
  - Platform Designer

- Begin Software Flow