

Introduction to Embedded NIOS and Platform Designer



Intel Programmable Solutions Group

Objectives

Upon completion of this workshop, you should have an understanding of:

1. Platform Designer system development tools - basic features and how to run the tool
2. How to develop applications on the Nios® II processor
3. Eclipse Integrated Development Environment (IDE) and development of “bare metal” applications utilizing Nios and associated FPGA hardware

Bare Metal Programming

Bare-metal programming is a term for programming that operates without various layers of abstraction or, as some experts describe it, "without an operating system supporting it." Bare-metal programming interacts with a system at the hardware level, taking into account the specific build of the hardware.

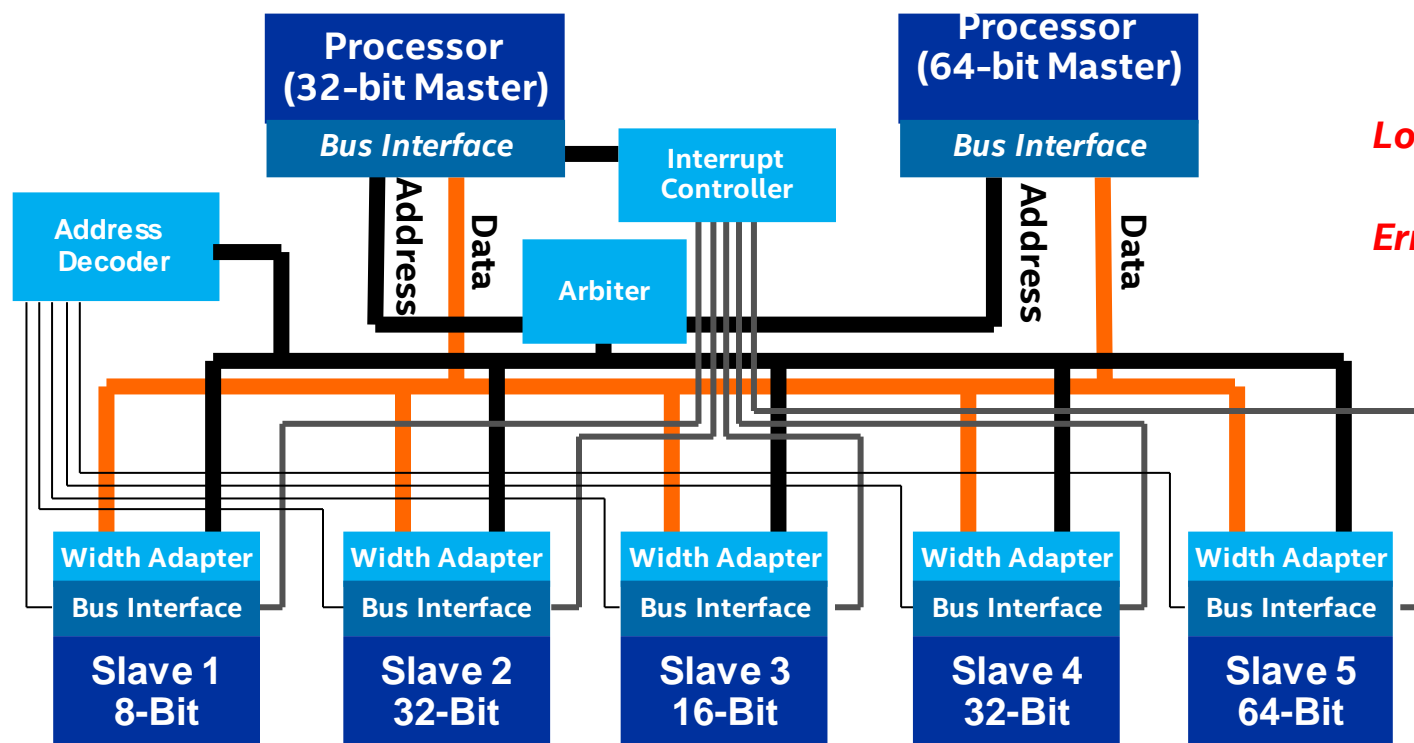
Should you need a Linux operating system for network connectivity, predefined libraries beyond what Nios offers, we suggest using the ARM SoC FPGA products and design flows. The Platform Designer skills you learn in this course will still apply.



Platform Designer System Integration

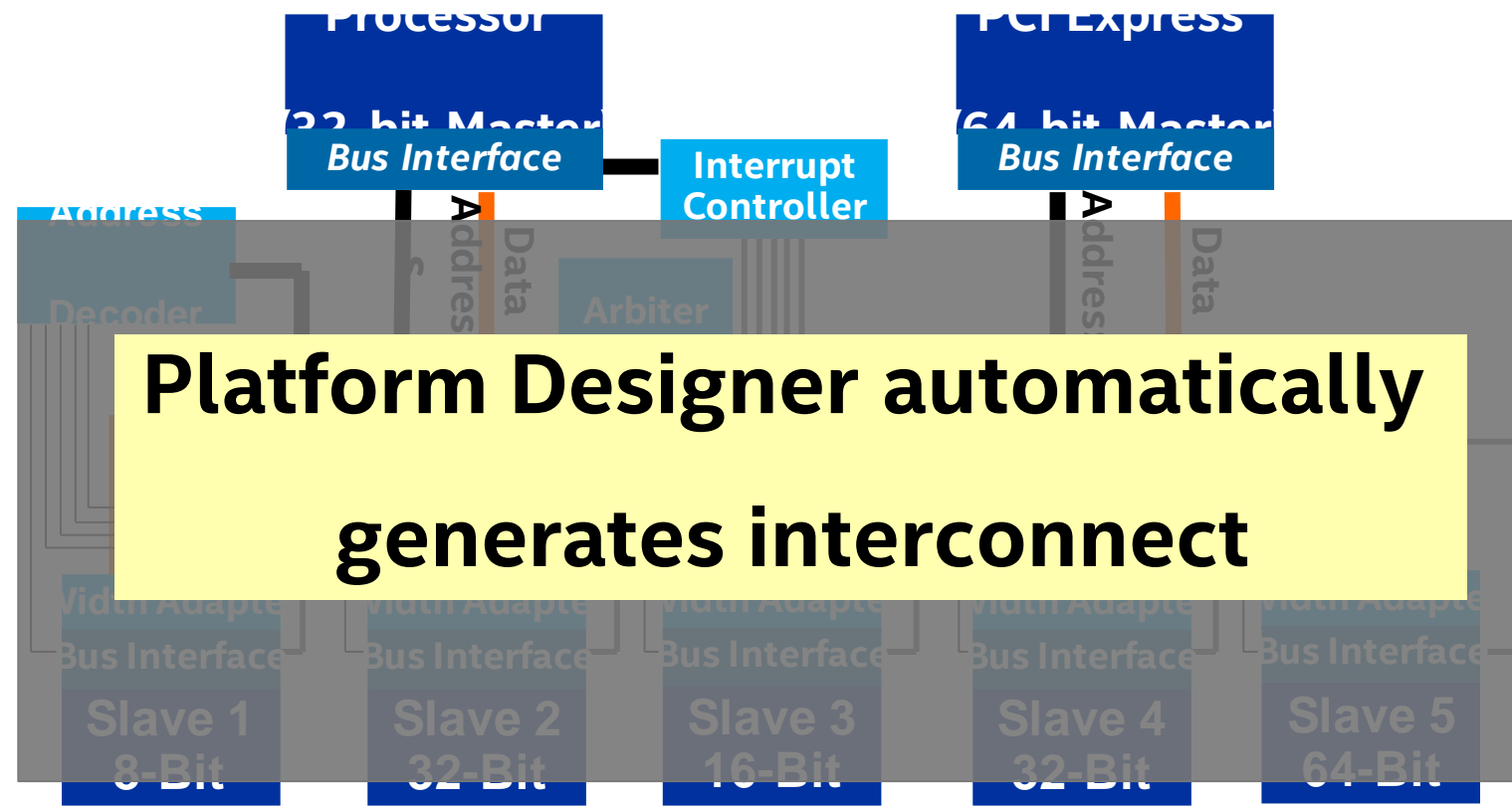
Traditional System Design

- Different interfaces (some standard, some non-standard)
- Significant engineering work required to design custom interface logic
- Integrating design blocks and intellectual property (IP) is tedious and error-prone

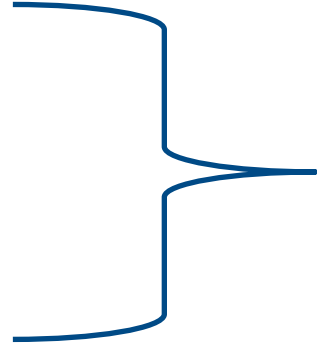


Automatic Interconnect Generation

- Avoids error-prone integration
- Saves development time with automatic logic & HDL generation
- Enables you to focus on value-add blocks



Platform Designer Features

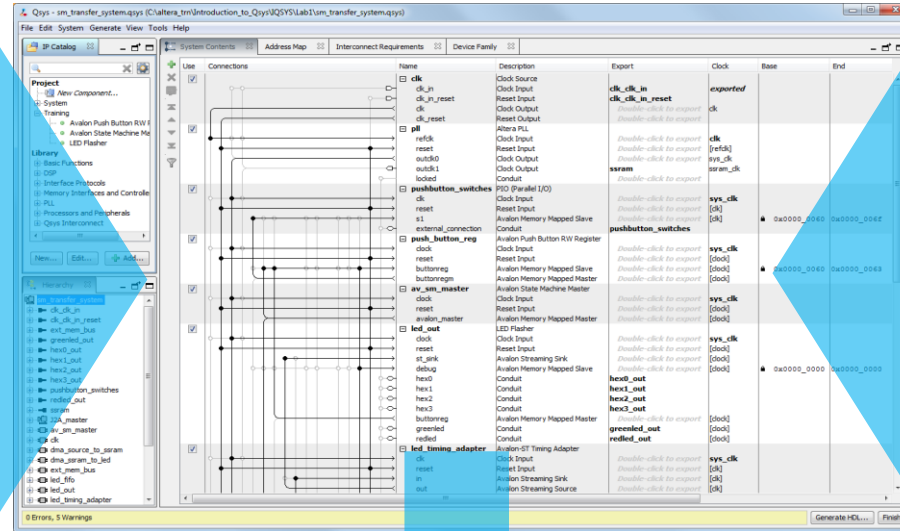
- Easy IP integration with switch fabric connectivity
 - Dynamic system generation
 - High-level system visualization
 - Custom IP authoring
 - IP verification and bus functional models (BFMs)
 - Simulation support for ModelSim[®] and other 3rd-party simulation tools
 - Real-time system debug through tools like System Console
- 
- Today's Lab

Easy to use System-Integration UI

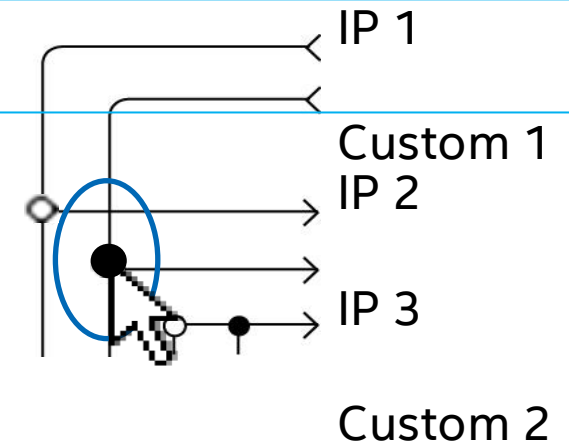


Catalog of available IP

- Interface protocols
- Memory
- DSP
- Embedded
- Bridges
- PLL
- Custom systems



Connect custom IP and systems



**Accelerate
development**

**Simplify
integration**

Automate error-prone integration tasks

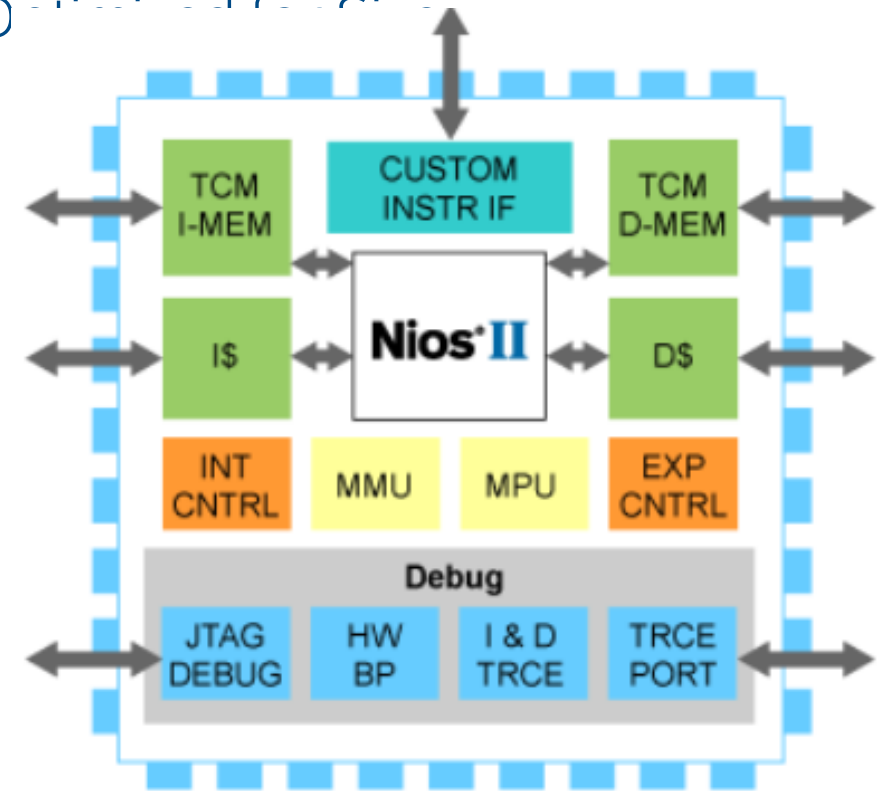
HDL



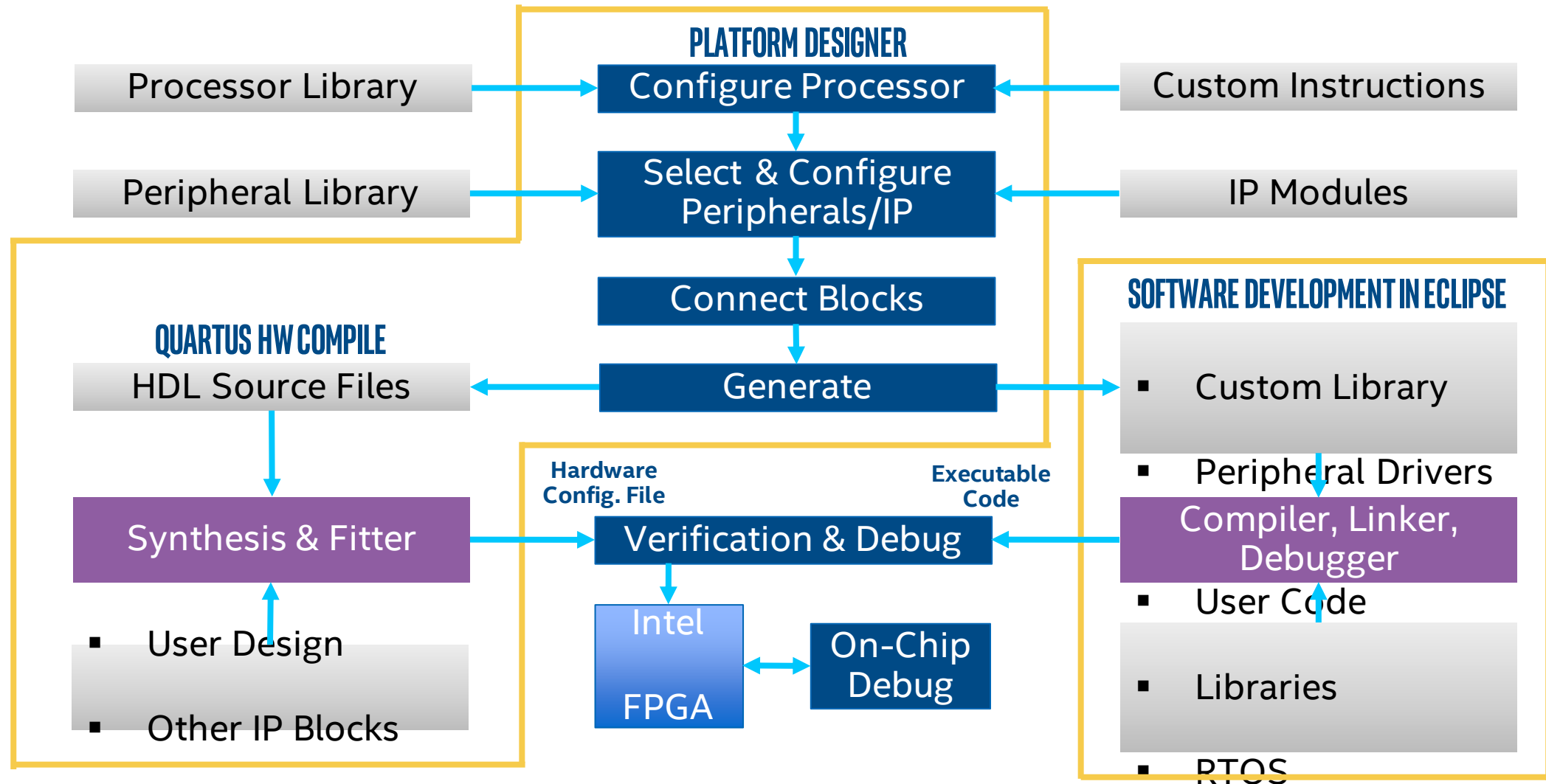
NIOS® II Processor

Nios II Processor

- Intel's 2nd Generation 32-Bit RISC Soft Microprocessor
- Two Variants: Fast - Optimized for Speed; Economy - Optimized for Cost
- Configurable features
 - Cache size
 - Tightly coupled memories
 - Arithmetic implementation
 - Interrupt controller
 - MMU/MPU
- Custom instructions and peripherals
- Compatible with all Intel FPGAs



Nios II Design Flow



Nios II Embedded Design Suite Features

Software Build Tools (SBT)

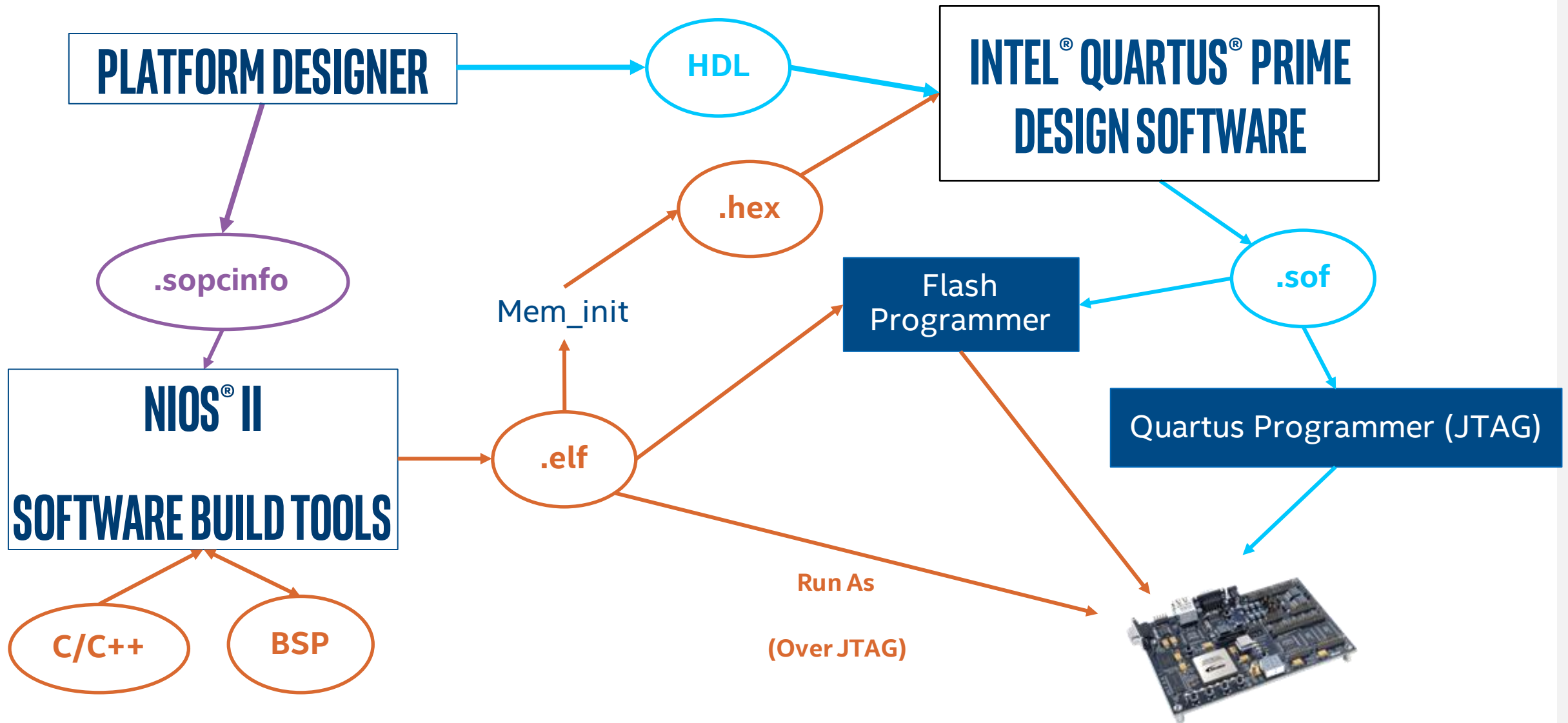
- Proprietary and open-source tools for creating Nios® II programs
- Commands, utilities & scripts
- GUI for developing and debugging software
- Plug-ins to industry standard Eclipse IDE
- Automated board support package (BSP) creation

Hardware Abstraction Layer (HAL)

- Unix-style API
- Interface with peripherals, timers, interrupts, etc.
- Device access and initialization
- All hardware related software libraries automatically updated

<code>_exit()</code>	<code>gettimeofday()</code>	<code>sbrk()</code>	<code>open()</code>
<code>_close()</code>	<code>ioctl()</code>	<code>stat()</code>	<code>opendir()</code>
<code>closedir()</code>	<code>isatty()</code>	<code>usleep()</code>	<code>read()</code>
<code>fstat()</code>	<code>kill()</code>	<code>wait()</code>	<code>readdir()</code>
<code>getpid()</code>	<code>lseek()</code>	<code>write()</code>	<code>rewinddir()</code>

Nios II SW Flow





Ecosystem and Support

Debugging

- Quartus Prime debugging tools available for all Nios systems
- System Console
 - Perform register/address level transactions through a scriptable Tcl environment without software
- External memory interface (EMIF) debug toolkit
- Transceiver toolkit
- SignalTap™ II logic analyzer
 - Signal and logic level FPGA hardware debug

Additional Nios® II Resources

- [Nios® II Documentation](#) on intel.com
 - Including One-Click Download zip file of all available documentation
 - Nios® II Hardware and Software Developer's Handbooks
- [Intel Forum](#) - Thousands of users and topics
- [Design Store](#) - Lots of designs (search for Nios, Qsys, or Platform Designer)
- Eclipse Help Content and Welcome Page



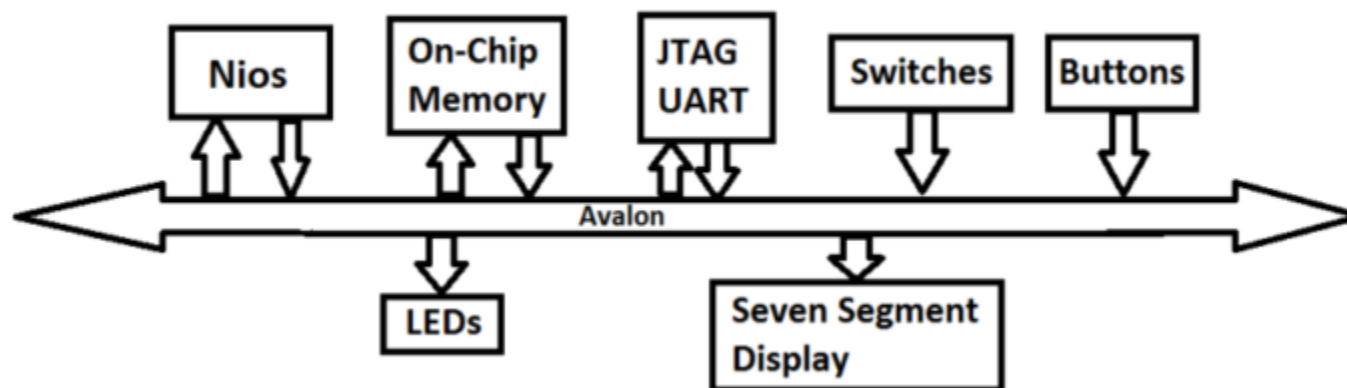
Demo



Embedded Nios Lab

Lab

1. Build system block diagram in Platform Designer
2. Connect to a top level Verilog module
3. Export memory map into Eclipse environment
4. Build project and compile code
5. Download hardware.
6. Download software
7. Try some other cool code and download it.



Tips

1. Follow connections in exactly! Double check inputs vs. outputs.
2. Name things exactly! No spaces in files OR folder names.
3. The instruction and data master is connected to the on chip memory (not just data master)!
4. Remember **Platform Designer** → **System** → **Assign Base Addresses** after edits
5. When connecting Eclipse to the board, “Refresh Target” is hidden on the right.
6. To reprogram HW, first stop eclipse from running software (use “stop” button) or the USB blaster will be hung!
7. If HW fails to program, hit Start a second time!
8. Have fun and try more embedded projects on your own!

Using the Hands On Lab Environment

1. The instructor will start up the Hands-On lab environment. Select a machine. Start with the N1 machines. Your name will show up below the machine name. Everyone should select their own machine.
2. Login student. PW=QPrime.1
3. Make working directories under the download folder.
4. Close any open Quartus windows.
5. DO NOT double click the .qar file. You must open from Quartus else the wrong version of Quartus is invoked.
6. You have no means to change switch values or view LEDs/7-segments. However you can see console I/O in Eclipse. Have the instructor demo board interaction (viewing "Hello world from Nios!"). You can send the instructor the .sof and .elf files to demo.

Legal Disclaimers/Acknowledgements

- Intel technologies may require enabled hardware, software or service activation
- No product or component can be absolutely secure
- Your costs and results may vary
- Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others
- OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos
- *Other names and brands may be claimed as the property of others
- This document is intended for personal use only.
- Unauthorized distribution, modification, public performance, public display, or copying of this material via any medium is strictly prohibited.

