

SISTEMI EMBEDDED

Building a Nios II Computer
from scratch

Federico Baronti

Last version: 20180326

Introduction

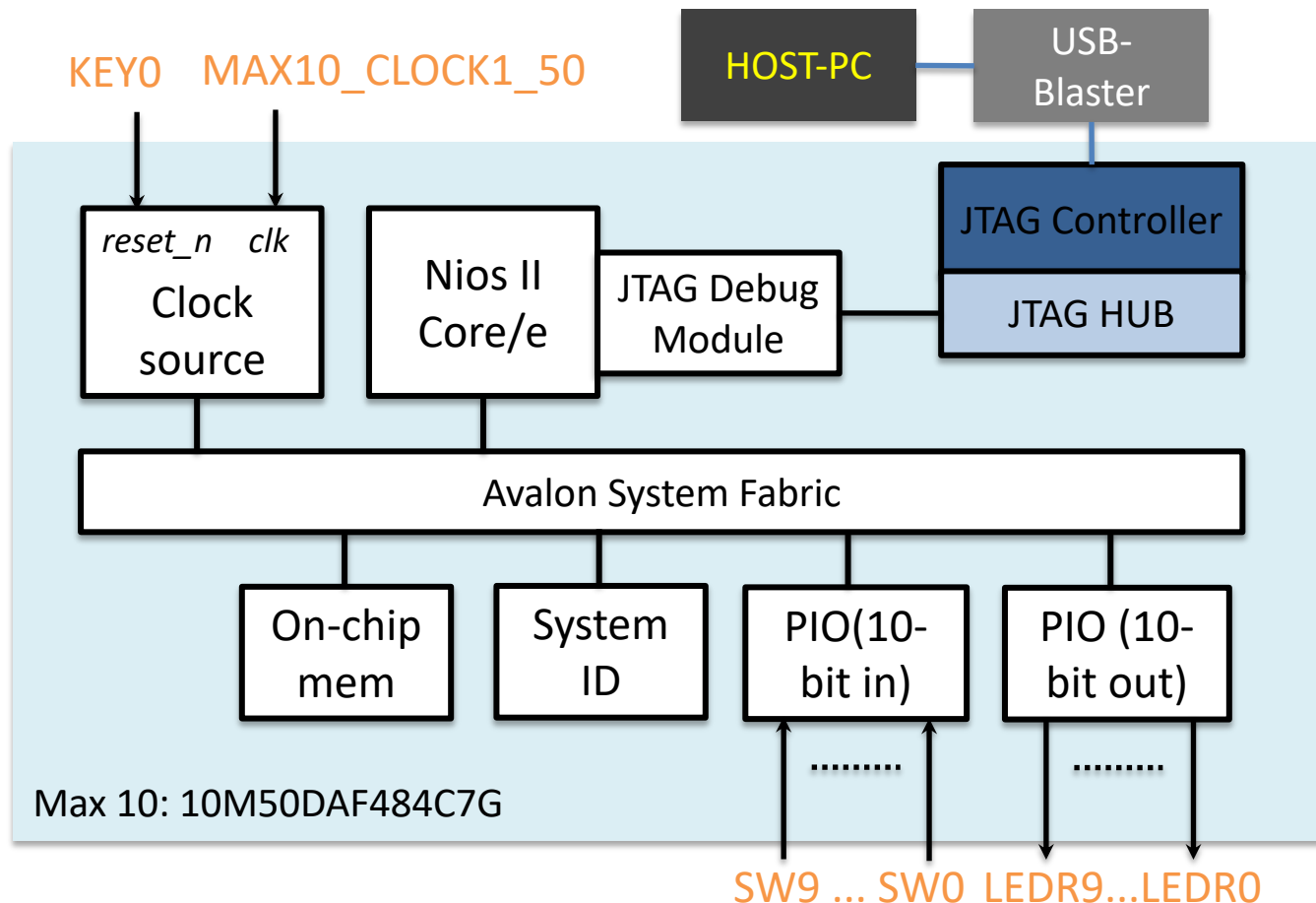
- Problem:
 - Build a (NIOS II) Computer tailored to application needs
- Solutions:
 - Use library cores and custom HDL code
 - **Use specific design tools (Qsys) to help assemble the system**
 - Components (CPUs, memory (controllers), peripherals,...) selected from Altera, other vendors or custom libraries
 - Connections (Avalon System Interconnect Fabric) are generated automatically by the tool
 - **Need for standard interfaces**

Avalon System Interconnect Fabric

- Overview of Avalon standard interfaces:
 - Clock
 - Reset
 - Interrupt
 - Memory-Mapped (master and slave)
 - Streaming (source and sink)
 - Conduit

Example: First Nios System

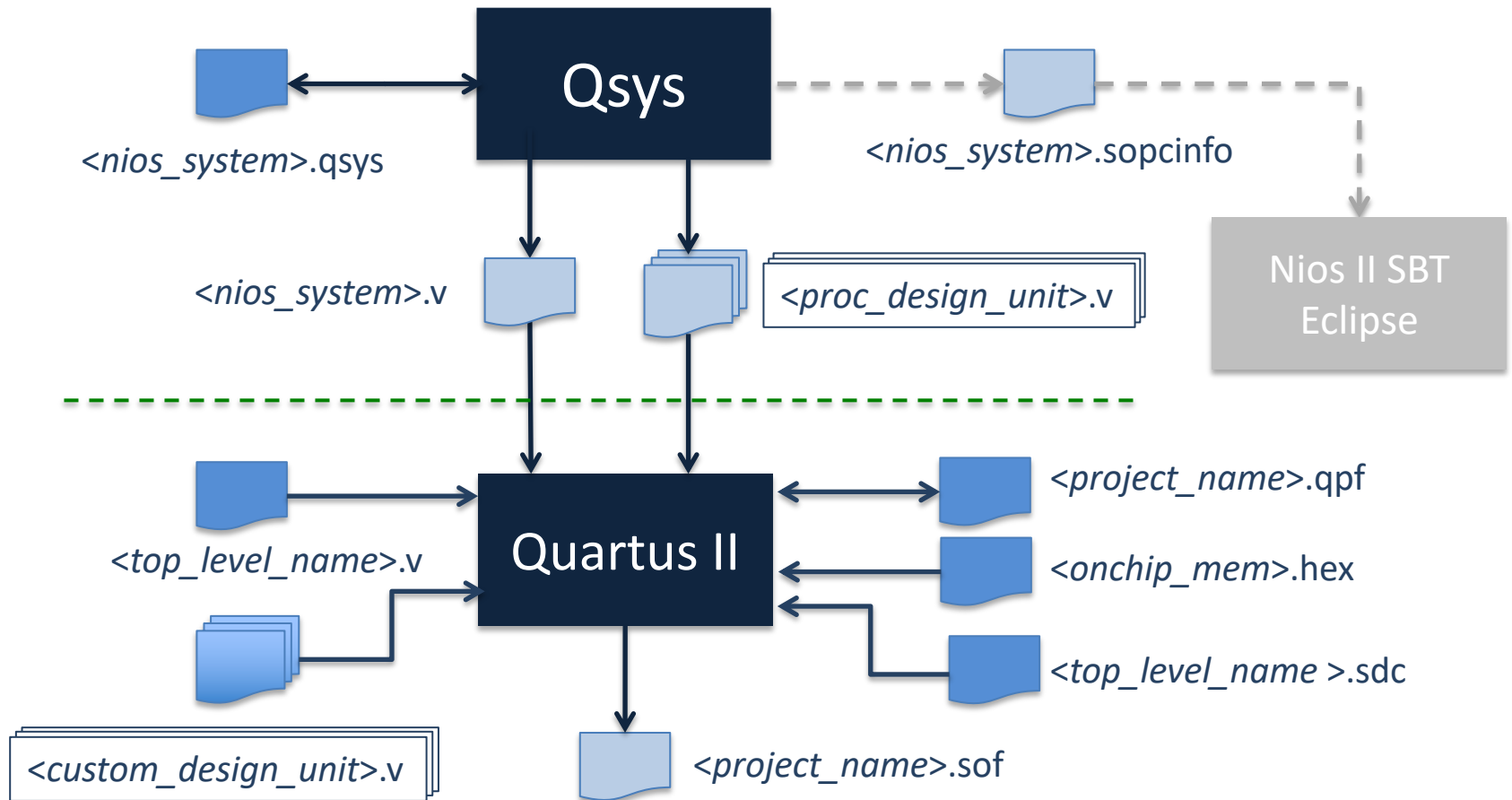
- Handles **slider switches** and **LEDs** through PIO peripherals



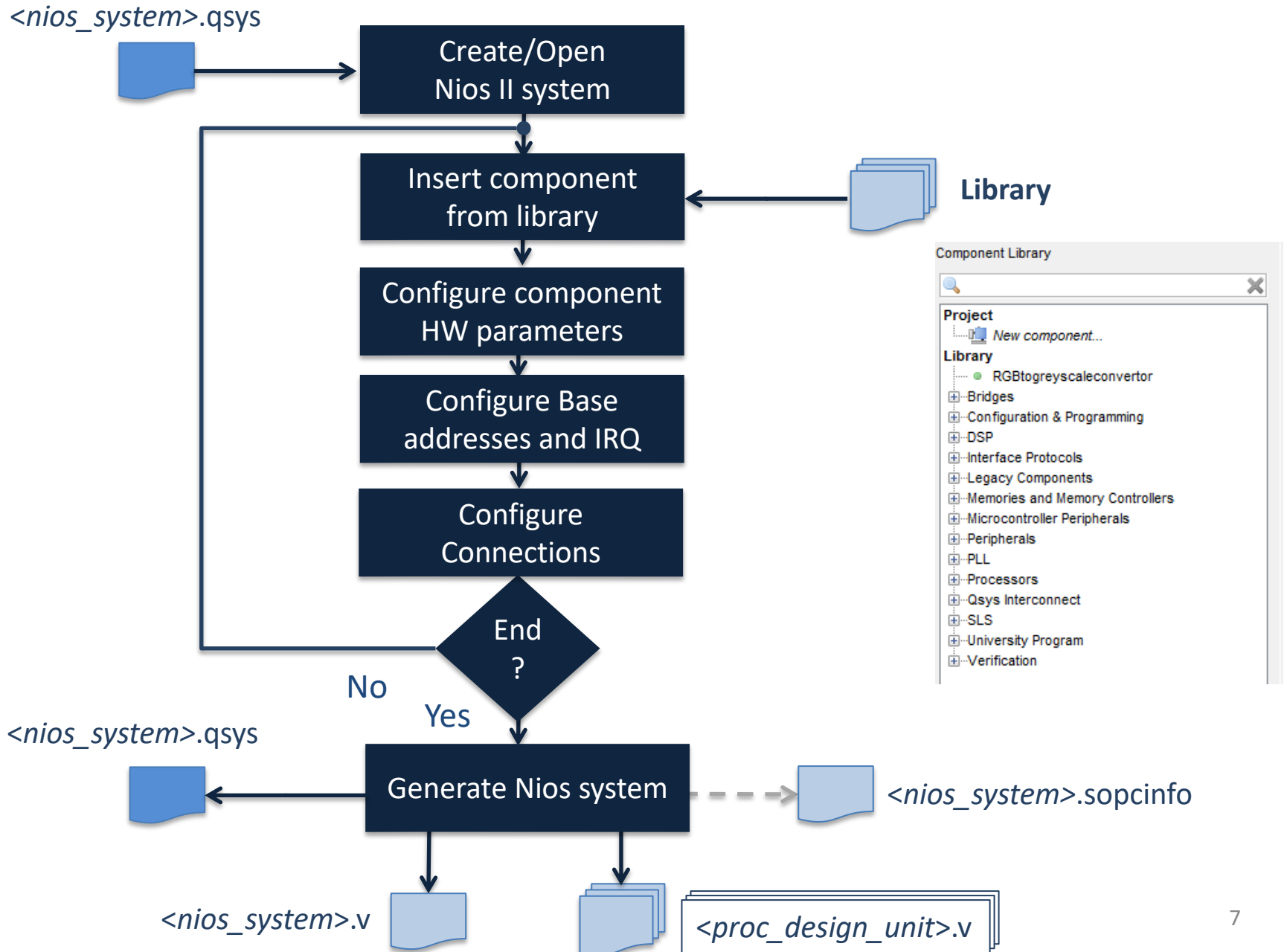
First Nios Computer components

- CPU (simplest, *i.e.*, *economy* version) with JTAG Debug Module
- On-chip memory for program and data (64 KB)
- 2 PIOs
 - Input for reading slider switches (10 bit)
 - Output for driving red LEDs (10 bit)
- System ID Peripheral for computer identification

Nios II Hardware Flow



Qsys Flow



Guided example (1)

- Open the template Quartus project: DE10_Lite_First_Computer
- Launch Qsys tool
- Define the Nios_system components
 - Clock source: *clk* (it is added automatically)
 - Nios II Proc.: *nios2_proc*
 - Choose the economy version of the NiosII proc. (NiosII/e) and the Level 1 for the JTAG Debug Module
 - On-chip Memory: *onchip_memory* (dual-port configuration, single clock)
 - PIO: *leds*
 - Output for driving LEDs
 - PIO: *sliders*
 - Input for reading slider switches status
 - System ID Peripheral: *sysid* (ID = 1!)

Qsys main window

The screenshot shows the Qsys main window with the Address Map tab selected. The window title is "Qsys" and the menu bar includes "File", "Edit", "System", "View", "Tools", and "Help". The "Component Library" on the left lists various components under "Project" and "Library". The "System Contents" table is the central focus, with annotations:

- Component instance name:** A blue box highlights the "clk_0" instance name in the "Name" column.
- Base address:** A blue box highlights the "Base" column header.
- Configure internal connections:** A green box highlights the "Conn..." column.
- Decide signals to be routed (exported) to the Qsys system boundary:** A green box highlights the "Export" column.

The "System Contents" table has the following data:

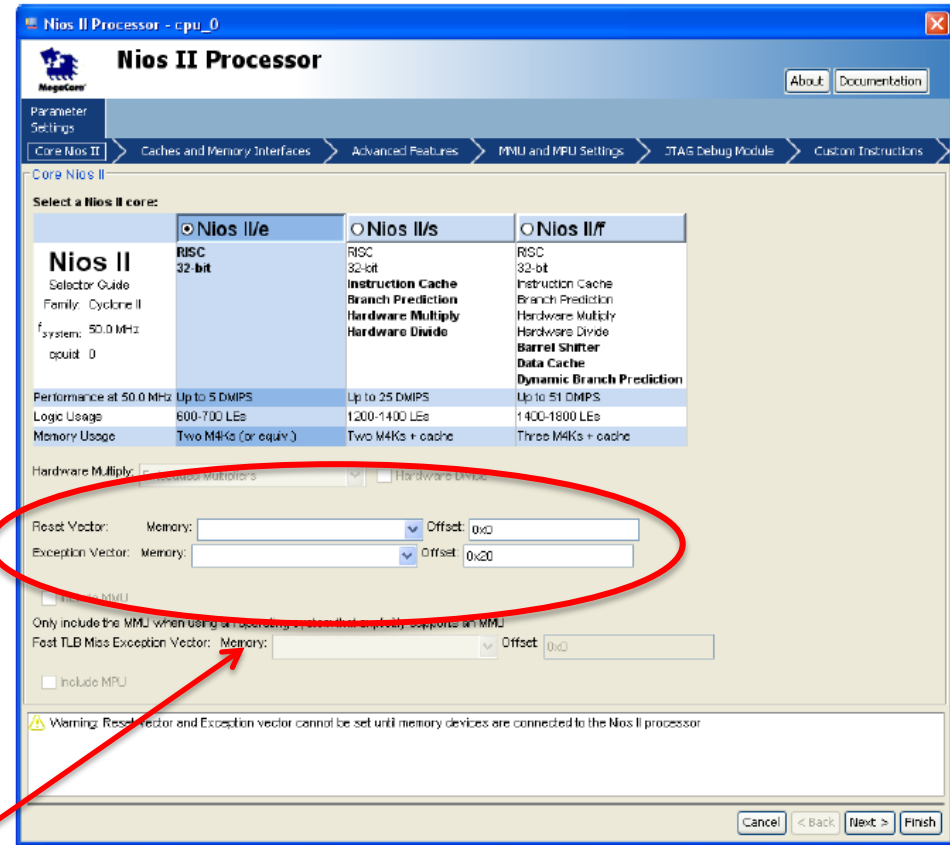
Use	Conn...	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk_0	Clock Source				
		clk_in	Clock Input	clk			
		clk_in_reset	Reset Input	reset			
		clk	Clock Output		clk_0		
		clk_reset	Reset Output				

The "Export" column contains the text: "Double-click to export" and "Double-click to export".

At the bottom, the "Messages" pane shows a table with columns "Description" and "Path".

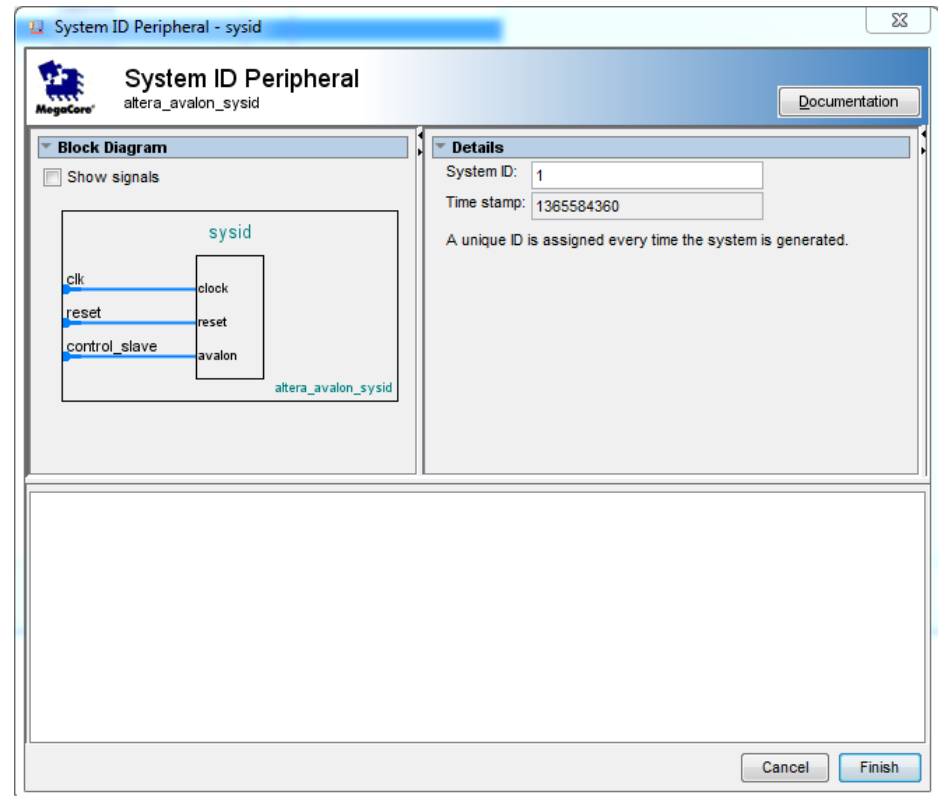
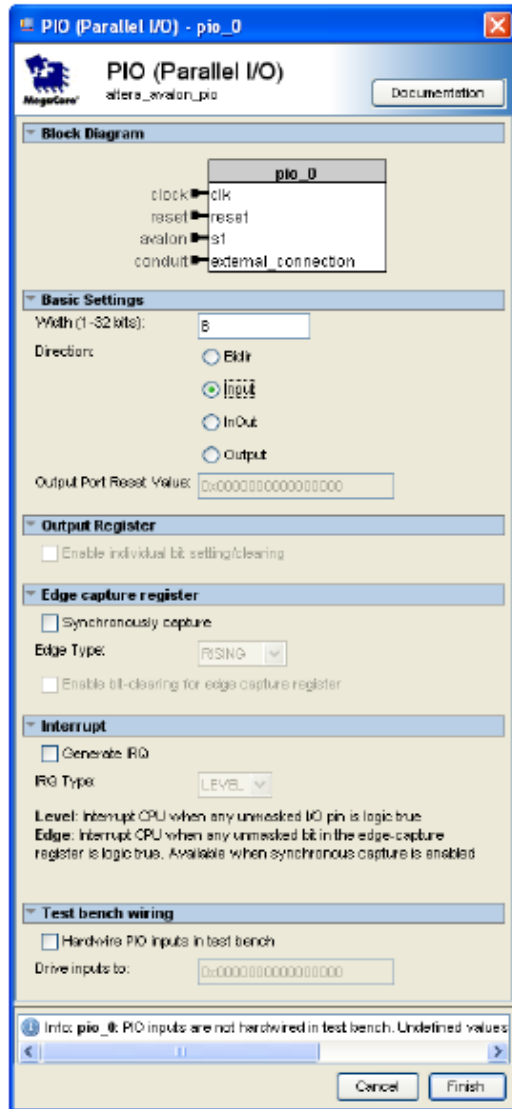
CPU choice

- Choose the most suited processor core
- 3 variants:
 - Economy
 - Standard
 - Fast
- Different features
 - Trade-off performance-cost



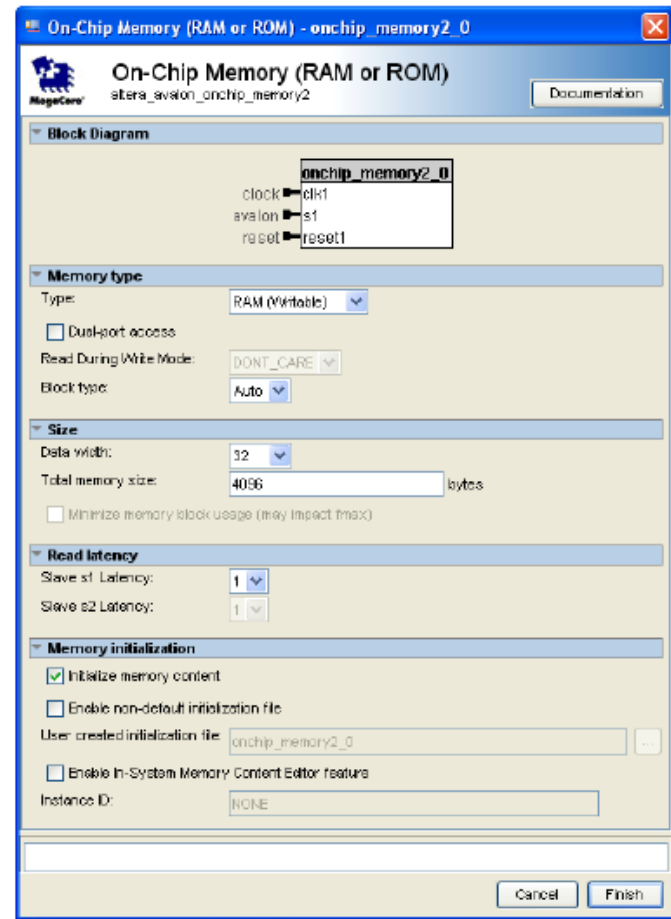
At least one memory must be present in the Qsys system in order to configure the **Reset** and **Exception** addresses

Additional peripherals



On-chip memory

- Define the organization of the on chip-memory
 - Type = RAM
 - Dual-port access
 - Single clock operation
 - Size = 65536 bytes
 - Word length = 32
- Initialization file:
<*nios_system*>_onchip_memory.hex



Guided example (2)

- **Configure internal connections**

- Route *clk* from Clock Source component to the other components

- Create *reset* network

- “Route” reset signals from Clock Source and JTAG Debug Module (within the Nios II proc.) components to the other components
- Can be done automatically using Create Global Reset Network command (System menu)

- Link the Avalon Memory-Mapped Interfaces:

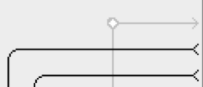
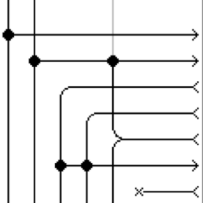
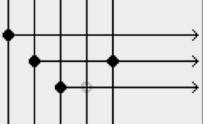
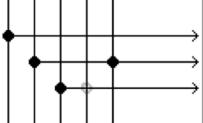
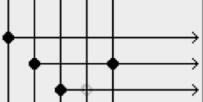
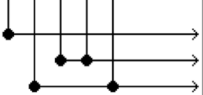
- *data_master* (Nios II proc.), *jtag_debug_module* (Nios II proc.), *s1* (*onchip_memory*), *s1* (PIO: *sliders*, *leds*), *control_slave* (*sysid*)
- *instruction_master* (Nios II proc.), *jtag_debug_module* (Nios II proc.), *s2* (*onchip_memory*)

Guided example (3)

- **Export external connections**
 - *Sliders* and *leds* PIOs have conduit interfaces, the related signals (`external_connection`) must be routed to the Qsys system boundary
- **Assign base addresses**
 - Manually to each component with slave Memory-Mapped Interfaces (pay attention to avoid overlaps!)
 - Assign Base Addresses (from the System menu)

Guided example (4)

We are now ready to generate the Qsys system and go back to Quartus II

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk	Clock Source				
		clk_in	Clock Input	clk			
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	<i>Double-click to export</i>	clk		
		clk_reset	Reset Output	<i>Double-click to export</i>			
<input checked="" type="checkbox"/>		nios2_proc	Nios II Processor				
		clk	Clock Input	<i>Double-click to export</i>	clk		
		reset_n	Reset Input	<i>Double-click to export</i>	[clk]		
		data_master	Avalon Memory Mapped Master	<i>Double-click to export</i>	[clk]		IRQ 0
		instruction_master	Avalon Memory Mapped Master	<i>Double-click to export</i>	[clk]		IRQ 31
		jtag_debug_module_reset	Reset Output	<i>Double-click to export</i>	[clk]		
		jtag_debug_module	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	0x4800	0x4fff
		custom_instruction_master	Custom Instruction Master	<i>Double-click to export</i>			
<input checked="" type="checkbox"/>		green_leds	PIO (Parallel I/O)				
		clk	Clock Input	<i>Double-click to export</i>	clk		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		s1	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	0x5000	0x500f
		external_connection	Conduit	green_leds_external_connection			
<input checked="" type="checkbox"/>		sliders	PIO (Parallel I/O)				
		clk	Clock Input	<i>Double-click to export</i>	clk		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		s1	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	0x5010	0x501f
		external_connection	Conduit	sliders_external_connection			
<input checked="" type="checkbox"/>		sysid	System ID Peripheral				
		clk	Clock Input	<i>Double-click to export</i>	clk		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		control_slave	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	0x5020	0x5027
<input checked="" type="checkbox"/>		onchip_memory	On-Chip Memory (RAM or ROM)				
		clk1	Clock Input	<i>Double-click to export</i>	clk		
		s1	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk1]	0x2000	0x3fff
		reset1	Reset Input	<i>Double-click to export</i>	[clk1]		

Guided example (6)

- Back to Quartus II
 - Import Qsys system into Quartus project. Do **one** of the followings:
 - Method I: Add the .qip file stored in *nios_system>/synthesis* to the project
 - Method II: Add the .qsys file to the project
 - Create/Edit the root module of the project
 - Include the Nios_system module as hierarchical block (Verilog)
 - Compile the project to make the hardware ready

Guided example (7)

- **Integrating Qsys system into Quartus II project**
 - Method I: Add the (Quartus II file) .qip file stored in `<nios_system>/synthesis` to the project
 - .qip file is created when generating the Qsys system together with the .sopcinfo and the HDL files
 - It lists all the files necessary for compilation in Quartus II, including the references to the HDL files generated by Qsys

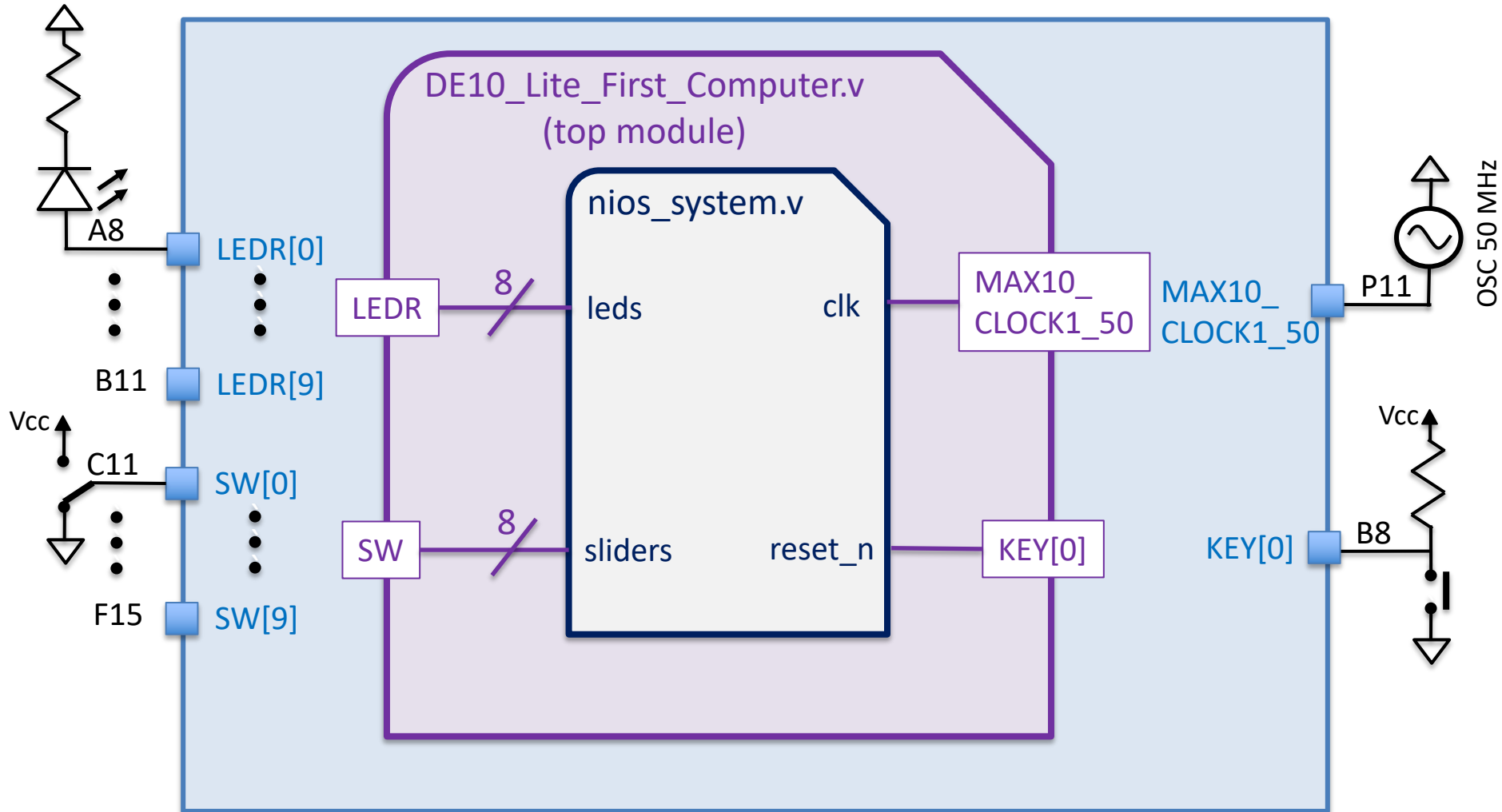
Guided example (8)

- **Integrating Qsys system into Quartus II project**
 - Method II: Add the .qsys file to project
 - The Qsys system is **now** (re)generated by Quartus II at each compilation
 - The generated HDL files are stored at a different path than those generated directly by Qsys
 - db/ip/<nios_system>
 - Note that the *sysid* timestamp changes at each compilation in Quartus II
 - **The BSP must be regenerated using the new sopcinfo file after each compilation, even if we have not made any change to the Qsys system!**

Guided example (7a)

- Project root module

Max 10: 10M50DAF484C7G



Guided example (7b)

- Project root module

```
// DE10_Lite_First_Computer.v
```

```
module DE10_Lite_first_computer(
```

```
    //input
```

```
    MAX10_CLOCK1_50,
```

```
    KEY,
```

```
    SW,
```

```
    //output
```

```
    LEDR
```

```
);
```

```
input MAX10_CLOCK1_50;
```

```
input [1:0] KEY;
```

```
input [9:0] SW;
```

```
output [9:0] LEDR;
```

```
// Add the nios_system instance
```

```
// The instance template can be copied from Qsys HDL example tab
```

```
endmodule
```

Guided example (7c)

- Project root module
(using Verilog-2001 C-style port declaration)

```
// DE10_Lite_First_Computer.v
```

```
module DE10_Lite_first_computer(  
    input MAX10_CLOCK1_50,  
    input [1:0] KEY,  
    input [9:0] SW,  
  
    output [9:0] LEDR  
);
```

```
// Add the nios_system instance
```

```
// The instance template can be copied from Qsys HDL example tab
```

```
endmodule
```

Testing First Nios System (1)

- Write a program that makes the RED LEDs to be controlled by the SLIDERS SWITCHES
- **If successful**, generate the hex file to initialize the on-chip memory. Recompile the Quartus project and reprogram the FPGA. Your program should run automatically!
- To generate the hex file from elf. Open the Nios 2 Command Shell and navigate to the Eclipse project folder. Customize the following command:

```
elf2hex --record=4 --width=32 --base=<onchip_memory base address>  
--end=<onchip_memory end address> --input=<eclipse_project_name.elf>  
--output=../../Hardware/onchip_mem.hex
```

Testing First Nios System (1a)

- Enrich the *First Nios System* w/ 2 additional PIOs properly configured to control the **push buttons** (w/ edge capture capability) and the **HEX3-HEX0 7-seg displays** available on the DE10-Lite board.
 - Make the ID of this new computer equal to 2
 - **Test the computer running** the **LED rotation**, the **Fast Click** and the **Week day** programs
 - Recall that the push button signal is low when the switch is pressed and that a led of the 7-seg display is ON when driven low

Testing First Nios System (2)

- Go back to Qsys, add the JTAG-UART peripheral (Library/Interface Protocols/Serial), regenerate the Nios system and compile the design again (top level entry does not need to be changed)
- Write a program that say Hello to the host together w/ the system ID and timestamp