

SISTEMI EMBEDDED

Tutorial on creating and using a
custom component in a Qsys system

Federico Baronti

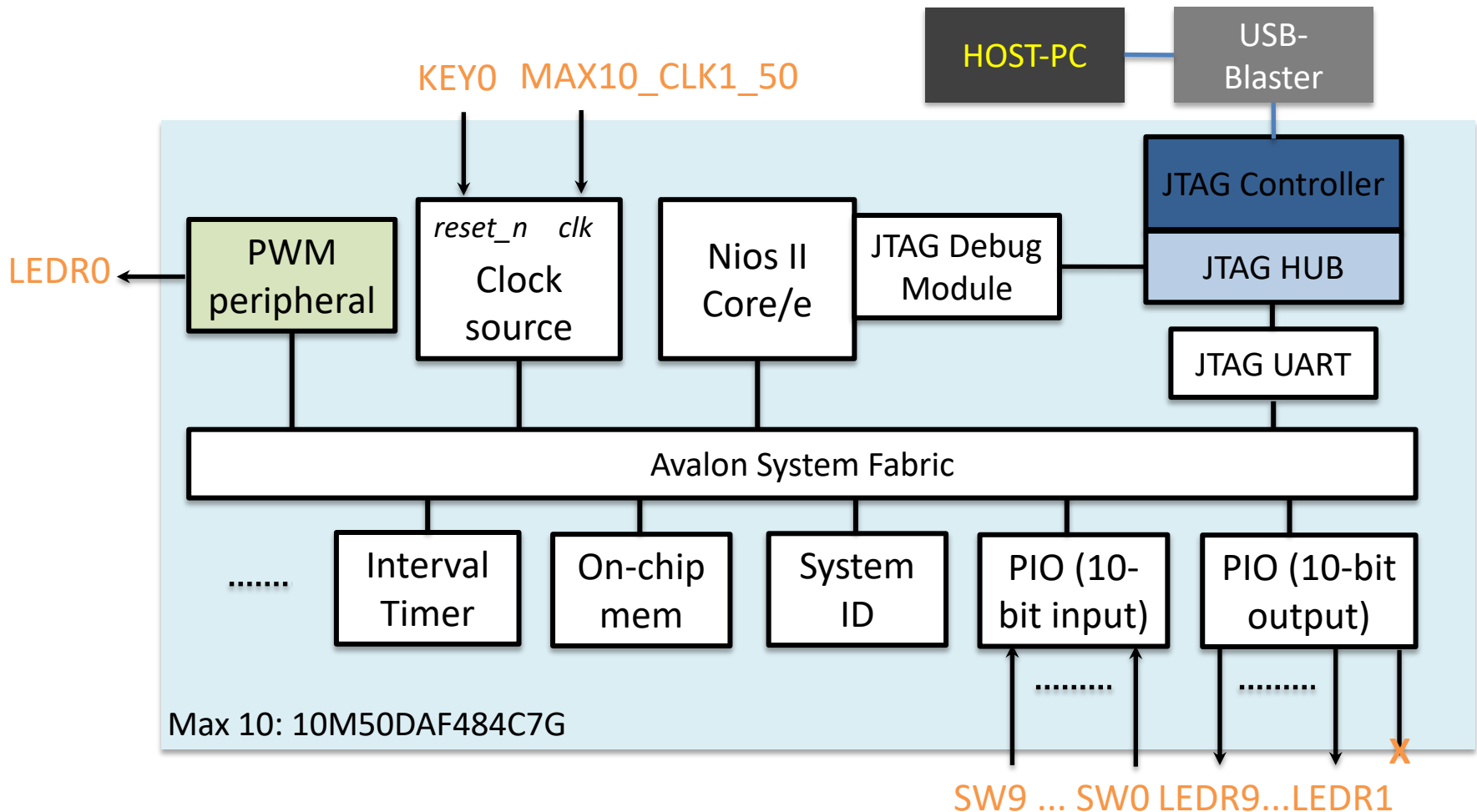
Last version: 20180423

Introduction

- Problem:
 - Use a *custom component* within a Qsys system
- Solution:
 - Provide the *custom component* with standardized interfaces: clock, reset, memory-mapped, ...
 - Add the *custom component* to the Qsys Library
 - Instantiate, configure and connect the *custom component* as any other component in the system
 - Provide an appropriate device driver to control the *custom component* from the software application

Example: PWM peripheral

- Provide my_first_computer (enriched with Watchdog, Interval Timer, JTAG UART) with a PWM peripheral to control the brightness of a LED



PWM Peripheral (1)

- Memory-mapped with 3 regs: CONTROL (1 bit), PERIOD (32-bit) and COMPARE (32-bit)
- It is based on a **32-bit up-counter**
 - The counter is forced to 0 when reaches the content of the PERIOD reg
- The PWM output is set when the counter goes from the PERIOD value to 0 and cleared when reaches the COMPARE value

PWM Peripheral (2)

- CONTROL reg allows you to enable/disable the PWM output (when disabled the output is low)
- The PERIOD reg allows you to set the PWM period $T_{\text{PWM}} = T_{\text{clk}} * (\text{PERIOD} + 1)$
 - A write to the PERIOD reg clears the counter
- The COMPARE reg allows you to set duty cycle δ

$$\delta = \frac{\text{COMPARE} + 1}{\text{PERIOD} + 1}$$

- The COMPARE reg is buffered and is updated when the counter goes from *PERIOD* to 0

PWM Peripheral (3)

- Module interface

```
module unipi_se_pwm (  
    // inputs:  
    address,  
    clk,  
    reset_n,  
    read,  
    write,  
    writedata,  
    // outputs:  
    readdata,  
    pwm_out  
);
```

```
// parameters  
parameter RESET_PERIOD = 0;  
parameter RESET_COMPARE = 0;  
parameter RESET_PWM_ENABLE = 0;
```

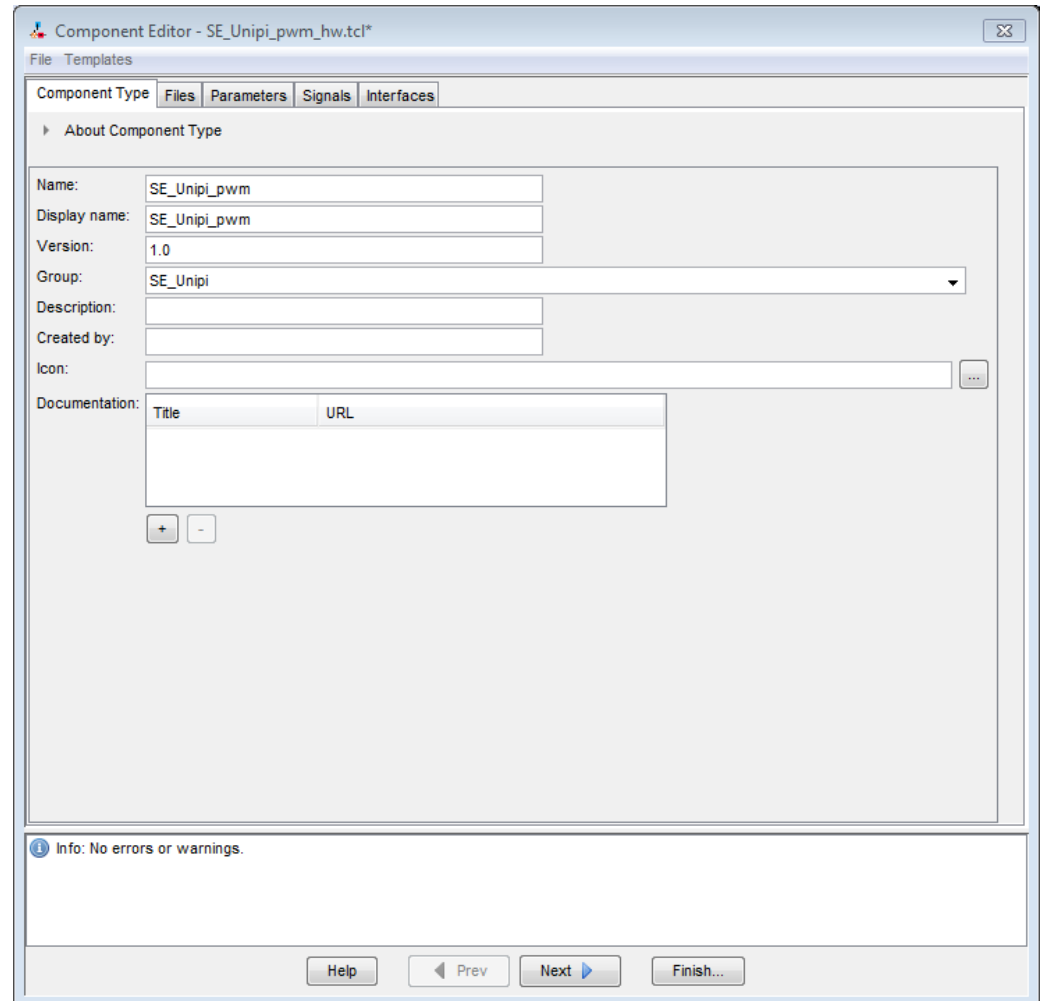
- The HDL code is provided (unipi_se_pwm.v)

PWM Peripheral (4)

Signal name		Avalon interface	Role
clk	IN	Clock	clock
reset_n	IN	Reset	reset_n
address	IN	Memory-Mapped slave	address
read	IN	Memory-Mapped slave	read
write	IN	Memory-Mapped slave	write
writedata	IN	Memory-Mapped slave	writedata
readdata	OUT	Memory-Mapped slave	readdata
pwm_out	OUT	Conduit	Export

Add custom component to Qsys (1)

- Component Library / Project / *New Component*
 - *Component Type*
Name your custom component



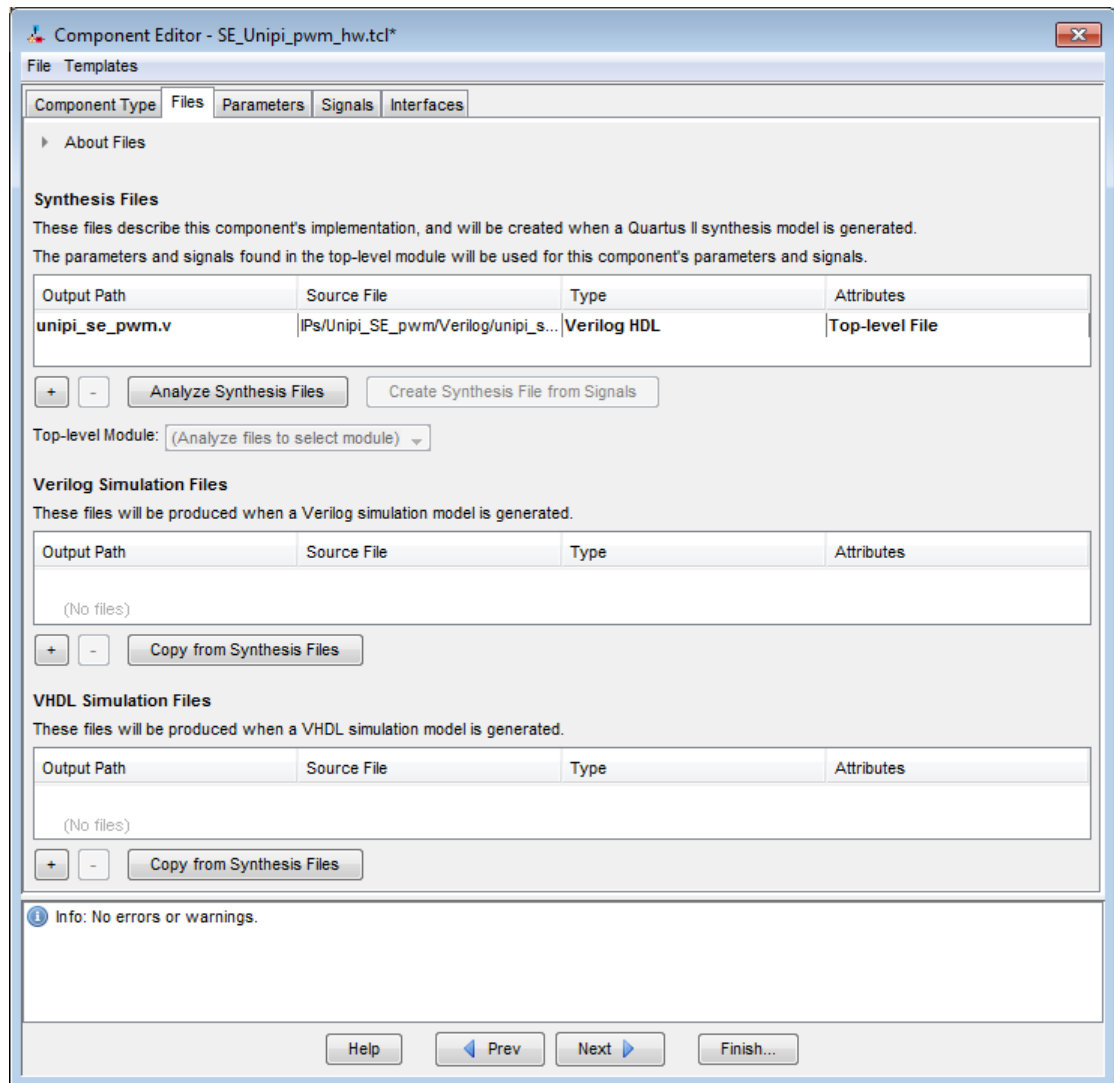
Add custom component to Qsys (2)

- Component Library / Project / *New Component*

– Files

Select the HDL
file(s)

Analyze Synthesis
Files

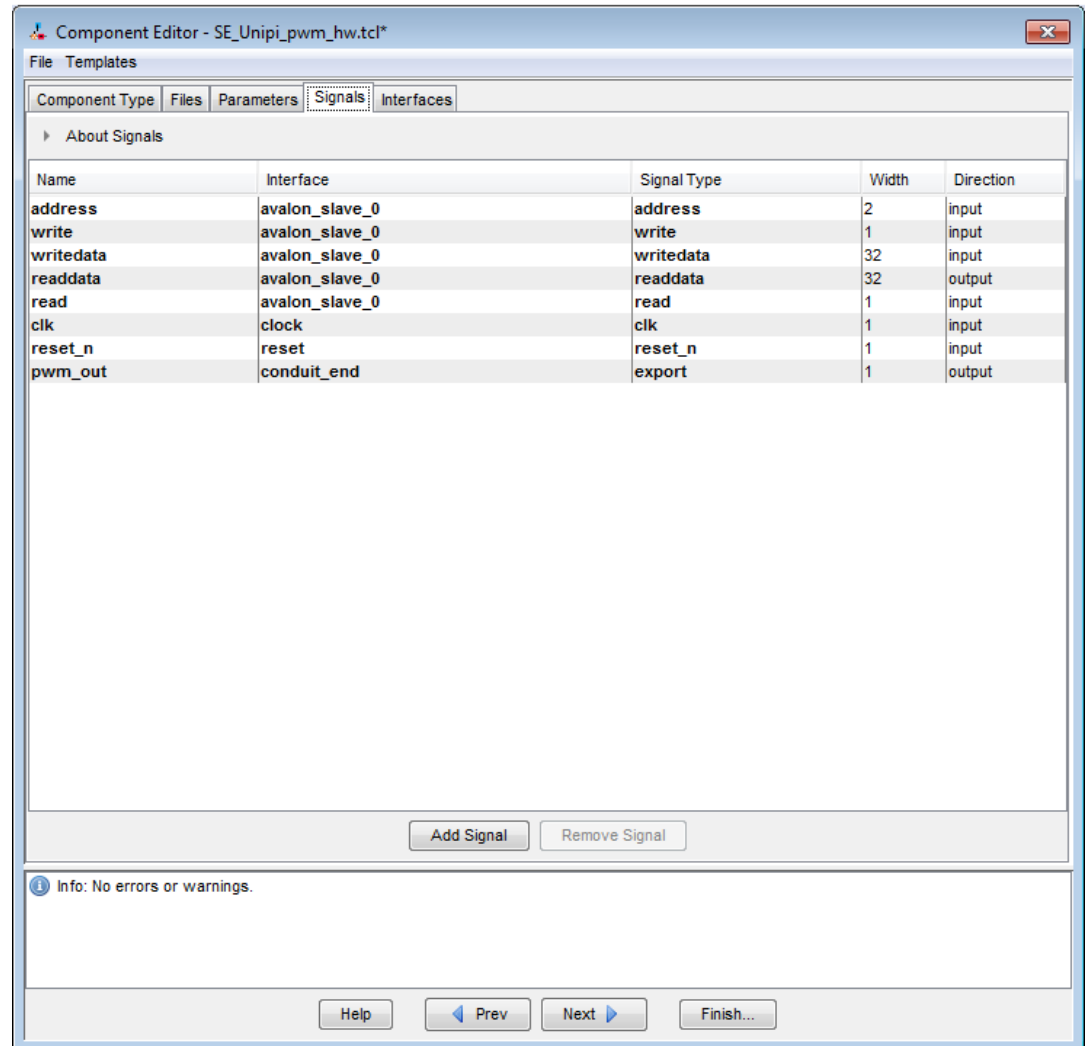


Add custom component to Qsys (3)

- Component Library / Project / *New Component*

– *Signals*

Assign each signal of the module to the appropriate Avalon Interface and the relevant signal role



Add custom component to Qsys (4)

- Component Library / Project / *New Component*

– Interfaces

Configure each Avalon Interface

The screenshot shows the Component Editor window for a custom component named "SE_Unipi_pwm_hw.tcl". The "Interfaces" tab is active, showing the configuration for an "avalon_slave_0" (Avalon Memory Mapped Slave) interface. The configuration includes:

- Name: avalon_slave_0
- Type: Avalon Memory Mapped Slave
- Associated Clock: clock
- Associated Reset: reset
- Assignments: Edit...

The "Block Diagram" shows the component's internal structure, including the "avalon_slave_0" block and its connections to the "address", "write", "writedata", "readdata", and "read" signals. An orange arrow points from the "Read wait" and "Write wait" fields in the "Timing" section to the "read" signal in the block diagram.

The "Parameters" section shows the following settings:

- Address units: WORDS
- Associated clock: clock
- Associated reset: reset
- Bits per symbol: 8
- Burstcount units: WORDS
- Explicit address span: 00000000000000000000

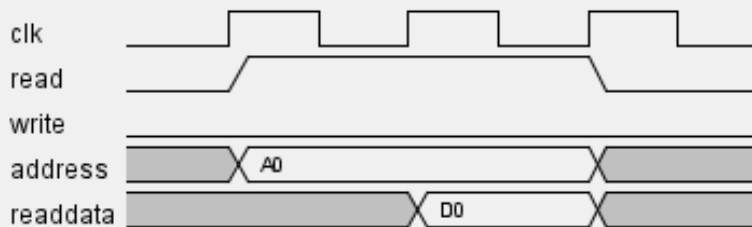
The "Timing" section shows the following settings:

- Read wait: 1
- Write wait: 0
- Timing units: Cycles

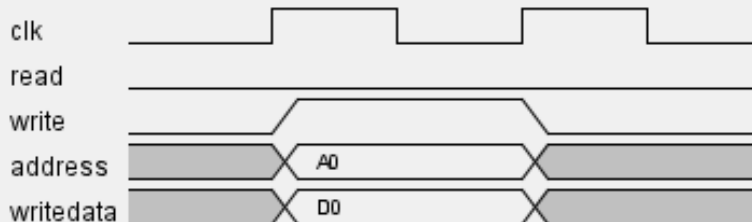
The "Pipelined Transfers" section shows the following settings:

- Read latency: 0
- Maximum pending read transactions: 0
- Burst on burst boundaries only:

Read Waveforms



Write Waveforms



Add custom component to Qsys (5)

- Component Library / Project / *New Component*

– *Parameters*

Set default values,...

Component Editor - SE_Unipi_pwm_hw.tcl*

File Templates

Component Type Files Parameters Signals Interfaces

▶ About Parameters

Name	Default Value	Edita...	Type	Group	Tooltip
RESET_PERIOD	499999	<input checked="" type="checkbox"/>	integer		
RESET_COMPARE	249999	<input checked="" type="checkbox"/>	integer		
RESET_PWM_ENABLE	1	<input checked="" type="checkbox"/>	integer		
AUTO_CLOCK_C...	-1	<input checked="" type="checkbox"/>	inteq...		clock_rate for inter...

Parameters:

Remove Parameter

Preview - SE_Unipi_pwm

SE_Unipi_pwm

SE_Unipi_pwm

Block Diagram

Show signals

SE_Unipi_pwm_0

avalon_slave_0 avalon

clock clock

reset reset

conduit_end conduit

SE_Unipi_pwm

Parameters

RESET_PERIOD: 499999

RESET_COMPARE: 249999

RESET_PWM_ENABLE: 1

Putting into practice (1)

- Instantiate, configure and connect the SE_unipi_pwm component
- Generate the Qsys system
- Back to Quartus II
 - Update the nios_system instance to include the pwm output port and connect it to LEDR[0]
- Compile the project and configure the FPGA
 - If you have used the default values for the PWM Peripheral, the LEDR[0] should be on with average brightness

Putting into practice (2)

- Time to start an Eclipse project
- Write a program that allows you to control the brightness of LEDR[0] by means of the SW7-Sw0
 - Try to use the provided device drivers:
unipi_se_pwm.c, unipi_se_pwm.h