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 $T_{PWM} = T_{clk} \times (PERIOD+1)$
  - A write to the PERIOD reg clears the counter
- The COMPARE reg allows you to set duty cycle $\delta$
  \[
  \delta = \frac{COMPARE + 1}{PERIOD + 1}
  \]
  - The COMPARE reg is buffered and is updated when the counter goes from PERIOD to 0
PWM Peripheral (3)

• Module interface

```verilog
module unipi_se_pwm (  // inputs:
    address,  
    clk,      
    reset_n, 
    read,     
    write,    
    writedata,
    // outputs:
    readdata, 
    pwm_out  
);  
// parameters
parameter RESET_PERIOD = 0;
parameter RESETCOMPARE = 0;
parameter RESET_PWM_ENABLE = 0;
```

• The HDL code is provided (unipi_se_pwm.v)
## PWM Peripheral (4)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Avalon interface</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>IN Clock</td>
<td>clock</td>
</tr>
<tr>
<td>reset_n</td>
<td>IN Reset</td>
<td>reset_n</td>
</tr>
<tr>
<td>address</td>
<td>IN Memory-Mapped slave</td>
<td>address</td>
</tr>
<tr>
<td>read</td>
<td>IN Memory-Mapped slave</td>
<td>read</td>
</tr>
<tr>
<td>write</td>
<td>IN Memory-Mapped slave</td>
<td>write</td>
</tr>
<tr>
<td>writedata</td>
<td>IN Memory-Mapped slave</td>
<td>writedata</td>
</tr>
<tr>
<td>readdata</td>
<td>OUT Memory-Mapped slave</td>
<td>readdata</td>
</tr>
<tr>
<td>pwm_out</td>
<td>OUT Conduit</td>
<td>Export</td>
</tr>
</tbody>
</table>
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
Add custom component to Qsys (5)

- Component Library / Project / New Component
  - Parameters
    Set default values,...
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