Multiprocessors design

FPGA
MultiProcessors
Quartus II - NIOSII

Objectives	Design and implementation of a Dual Microprocessors system on FPGA
Tools	FPGA, QuartusII, Qsys and NIOSII SBT
Preliminary	VHDL, C, Embedded System Architecture, bus Avalon, processor NIOS II
Theory	FPGA, Avalon, Mubus, NIOS II
Material	QuartusII, FPGA DE1-SOC, DE0-nano, DE0-nano-SOC, interface RS232, Logic Analyzer
Duration	4h

1 Introduction

In this laboratory a Multiprocessors system is realized on the FPGA system. 2 NIOSII processors (CPU_0 and CPU_1) have to be implemented with QSys on QuartusII. They will have access to a common SDRAM memory or internal SRAM and share a common memory space. To synchronize their access to this common memory, a hardware mutex interface has to be implemented (from the library) as well as a mailbox interface to transfer data safely.

Each processor has its own RTOS and program.

This laboratory is a starting point for the final mini-project, that will be done in the last part of this course.

2 Altera multiprocessor tutorial

To start, study carefully the multiprocessor NIOSII tutorial "Creating Multiprocessor Nios II Systems" from Altera. It is available in:

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/tt/tt_nios2_multiprocessor_tutorial.pdf

This is a multi-processors design to see the methodology to implement a multi processors system with 6 NIOSII processors and common peripherals and SDRAM memory to solve "The Dining Philosophers' Problem".

- Look on the design and adapt the tutorial design to be used on FPGA board with the following components. It will be a simpler design.

3 Multi processor design to realize

Once the study of the tutorial is ok, you have to realize a specific design with:

- 2 NIOS II Processors (/f version), cpu_0 and cpu_1, with for each of them :
  - Timer
  - Performance counter
  - Internal FPGA SRAM 16k bytes, 32 bits width → 4k x 32 for each processor
  - A specific counter for performance evaluation (in VHDL)
  - 8 bits Parallel port for some output with the LEDs
  - JTAG DEBUG Module
- On the common Avalon bus :
  - SDRAM controller
  - An 8 bits bidi parallel port
  - 2x Mutex interfaces
  - Mailbox

- On cpu_0 processor bus :
  - UART serial interface
- On cpu_1 processor bus :
  - JTAG_UART (serial interface)
  -

Manipulation 1 System design

- With QuartusII and QSys realize the system
- Compile and download the hardware
- With NIOSII SBT, write a program for the processor owner of the JTAG UART to test ***printf***
- The same for the other processor to ***printf*** on UART

3.1 Parallel port test

For each processor, write a program to access a parallel port connected to the LEDs and incrementing a counter every 50ms. You can use RTOS uc/OSII on each of them.

3.2 Hardware Mutex

To allow exclusive access to the common parallel port, use a hardware ***mutex*** for access synchronization.

On each processor create a new task to increment a counter with the parallel port:

Cpu_0 increment every 20ms and cpu_1 decrement it every 10ms. The access has to be exclusive.

Evaluate with timer the full time to increment /decrement the parallel port including exclusive access to it.

Manipulation 2

- Develop the software to use the hardware mutex and measure the exclusive access timing.
- How the hardware mutex is working?

3.3 Hardware Mailbox

On one processor, create a process with the RTOS to wait on a queue of message to print. Imagine a mechanism to be able to send a message to print from the second processor to this queue.

Manipulation 3

- Modified the software to use the hardware Mailbox.
- Draw a schematic view of your implementation (hardware and software).

3.4 Hardware Counter

Develop a specific programmable counter that can be modified by simple write of the increment value, and that can be initialized with a specific value.

Both processors have access to this unit.

Manipulation 4

- Replace it in the previous design.
- Modified the software to use the hardware primitive counter in hardware.
- Can we remove the mutex ?