**CO503: Multi-Processor System-on-Chip (MPSoC) Design**

NAME: KODAGODA K.H.S.P

REG No: E/17/168

**INTRODUCTION**

In this practical there is an implementation of Multi-Processor System-on-Chip. In this practical there are two cpus. Between them three is a shared memory and processors share that common memory address space and communicate with each other via memory. In the part 1 there are simple producer and simple consumer. In this MPSoC there is a software implementation to procced this scenario. In the part 2 there is a dedicated hardware FIFO queue, and this can be used to increase the performance.

Part 1: Producer-Consumer Applications on a Shared Memory Multi-Processor

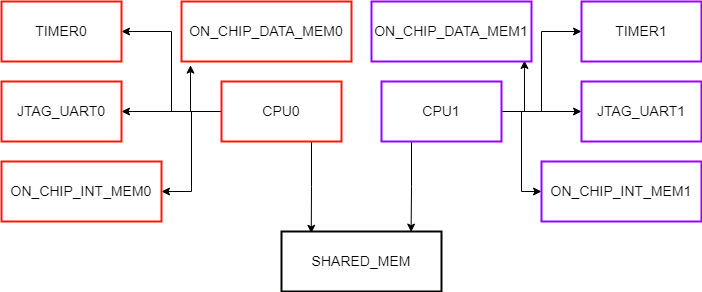
There are two processors they are cpu0 and cpu1. Each CPU has TIMER, JTAG UART and ON-CHIP INSTRUCTION, DATA MEMORIES. Also, both cpus are sharing ON-CHIP MEMORY. That memory is called shared memory and that is common to both cpus. Initially a new project was created, and those hardware peripherals are included to the MPSoC and 50MHz clock signal is used to drive.

CPU0

CPU0

Shared Memory

Figure 1: Abstract hardware design.



CLOCK

Clock Reset

Figure 2: Hardware design of MPSoC with shared memory.

Graphical user interface, application

Description automatically generated

Figure 3: Qsys design of MPSoC hardware.

Then two software projects are created using NIOS II software build tool. After that BSP editor is opened and then there is a tab called “Linker Script” in that window. There is section called “Linker Section Mappings” and it is used to set memory devices for sections. There are three memories for one cpu.

1. On-Chip data memory.

2. On-chip instruction memory.

3. Shared memory.

Graphical user interface, text, application

Description automatically generated

Figure 4: Memory regions

Then “.text” is set to the instruction memory and others are set to data memory.

Graphical user interface, text, application

Description automatically generated

Figure 5: Section mapping

After given skeleton codes were imported and some functions are modified. There is one function for initialization and there are two other functions for writing and reading. There are three pointers, and they are writing pointer, reading pointer and full pointer. Also, there are some important variables, and one variable indicates unit size of the data block and there is another variable to store base address of memory. Then software projects were uploaded and run on both cpus.

**OBSERVATIONS.**

There are two cpus and one is producer and other is consumer. Producer can write data to the shared memory and consumer can read data from memory. Producer was started and it produced 16 items, The capacity of FIFO is 16 and it waited for consumer’s actions. Then consumer started to read and after reading again producer wrote data to the shared memory. From 5 to 495 five by five numbers were read by consumer.

**A picture containing text

Description automatically generatedTable

Description automatically generated with medium confidence**Figure 6: Observations of producer and consumer

**PERFORMANCE.**

Initially producer was started and then consumer was started. The capacity of the FIFO is 16 and it took 22 ms to write and read all items. There can be some additional latency. But somehow time difference between consumers starting and consumer finishing can be considered as the time taken to procced.

Part 2: Hardware FIFOs

Diagram

Description automatically generatedThis part also has same steps with some changes. In the part 1 we used software implementations and in this part, we used hardware FIFO peripheral. The on-chip FIFO memory core is a configurable component used to buffer data and provide flow control in an SOPC Builder system. This FIFI memory can be integrated easily into any SOPC Builder-generated system.

Figure 7: FIFO with Avalon-MM Input and Output Interfaces

This is a simple hardware and here Avalon-MM write master pushes data into the FIFO through the input interface and read master pops data through output interface. Width of the input and output are same.

There are some configurations to do in that hardware peripheral. They were configured and then software

Graphical user interface, application, email

Description automatically generated

Figure 7: Configurations of FIFO

A screenshot of a computer

Description automatically generated with medium confidence  
 Figure 8: Qsys design of MPSoC hardware with FIFO

After generating the hardware design, it was uploaded to the FPGA board. Also, there are two software projects for producer and consumer. There are three functions to complete in the software project and they were coded according to the manual of FIFO. One function was used for initialization and other two functions were used for writing and reading. There are in-built functions and they could be used to that software implementation.

**OBSERVATIONS.**

When producer was started then it produced 16 items to the FIFO and it waited for reading of the consumer. The depth of the FIFO and once it could produce 16 items only. The consumer started to read the data and after reading producer produced rest of the items to the FIFO queue.

Graphical user interface, text, application, email

Description automatically generatedFigure 9: Producer waiting for consumers consuming

Table

Description automatically generatedFinally producer produced all data items and items were consumed by consumer.

Figure 10: Completion of the process.

**DEFFICULTIES.**

Initially IRQ values for FIFO was not set. Then it produced a error and the set IRQ values of FIFO.

Initialization of software implementation was done twice in the producer and consumer. Because of that after consumer again updated the pointer values during the process. So it caused to behavior of the FIFO and then initialization in the consumer software implementation was cleared.

**CONCLUSION.**

Shared memory implementation is a method which can be used to communicate between two processors in the multiprocessor system. To complete the process software implementation needed 22ms and to complete the process hardware implementation only needed 16ms. So hardware implementation of FIFO is better than software implementation of FIFO according to the performance. Also, accuracy of the hardware implementation is high.