IOB-SoC

A RISC-V-based System on Chip

IObundle Lda

August 9, 2020

Outline

- Introduction
- Project setup
- Create an IP core to instantiate in your SoC
- Edit the ./system.mk configuration file to declare a new peripheral
- Edit file `firmware.c` to drive the new peripheral
- Run the firmware in internal SRAM
- Run the firmware in external DDR
- Simulate and implement the system
- Implement in FPGA
- Implement in ASIC (WIP)
- Conclusions and future work

Introduction

- Building processor-based systems from scratch is challenging
- The IOB-SoC template eases this task
- Provides a base Verilog SoC equipped with
  - a RISC-V CPU
  - a memory system including boot ROM, RAM and AXI4 interface to DDR
  - a UART communications module
- Users can add IP cores and software to build more complex SoCs
- Here, the addition of a timer IP and its software driver is exemplified

Project setup

- Use a Linux machine or VM
- Install the latest stable version of the open source Icarus Verilog simulator (iverilog.icarus.com)
- Make sure you can access github.com using an ssh key
- Clone the repository github.com/IObundle/iob-soc
- Follow the instructions in its README file

Create an IP core to instantiate in your SoC

- Create a timer IP core repository or, alternatively, use the one at www.github.com/IObundle/iob-timer.git
- An IP core can be integrated into IOb-SoC if it provides the following files:
  1. hardware/hardware.mk
  2. software/software.mk

  - Please refer to the files `hardware/hardware.mk` and `software/software.mk` in the iob-timer submodule to learn how to organize a peripheral core.
- Add the IP core repository as a git submodule of your IOb-SoC repository:

```
git submodule add https://github.com/IObundle/iob-timer.git submodules/TIMER
```

- To configure the system to host the IP core, edit the `./system.mk` file as in the next slide

Edit the ./system.mk configuration file to configure the system with a new peripheral

```
#FIRMWARE
FIRM_ADDR_W:=13

#SRAM
SRAM_ADDR_W=13

#DDR
ifeq ($(USE_DDR),)
        USE_DDR:=0
endif
ifeq ($(RUN_DDR),)
        RUN_DDR:=0
endif

DDR_ADDR_W:=30
CACHE_ADDR_W:=24

#ROM
BOOTROM_ADDR_W:=12

#Init memory (only works in simulation or in FPGA)
ifeq ($(INIT_MEM),)
        INIT_MEM:=0
endif

#Peripheral list (must match respective submodule or folder name in the submodules directory)
PERIPHERALS:=UART
```

Edit the ./system.mk configuration file to configure the system with a new peripheral (continued: 2)

```
#SIMULATION TEST
SIM_LIST="SIMULATOR=icarus" "SIMULATOR=ncsim"
#SIM_LIST="SIMULATOR=icarus"
LOCAL_SIM_LIST=icarus #leave space in the end

ifeq ($(SIMULATOR),ncsim)
       SIM_SERVER=$(SIM_USER)@micro7.lx.it.pt
       SIM_USER=user19
else
#default
       SIMULATOR:=icarus
endif
```

Edit the ./system.mk configuration file to configure the system with a new peripheral (continued: 3)

```
#BOARD TEST
BOARD_LIST="BOARD=CYCLONEV-GT-DK" "BOARD=AES-KU040-DB-G"
#BOARD_LIST="BOARD=AES-KU040-DB-G"
#BOARD_LIST="BOARD=CYCLONEV-GT-DK"
#LOCAL_BOARD_LIST=CYCLONEV-GT-DK #leave space in the end
#LOCAL_COMPILER_LIST=CYCLONEV-GT-DK AES-KU040-DB-G

ifeq ($(BOARD),AES-KU040-DB-G)
	COMPILE_USER=$(USER)
	COMPILE_SERVER=$(COMPILE_USER)@pudim-flan.iobundle.com
	COMPILE_OBJ=synth_system.bit
	BOARD_USER=$(USER)
	BOARD_SERVER=$(BOARD_USER)@baba-de-camelo.iobundle.com
else ifeq ($(BOARD),CYCLONEV-GT-DK)
	COMPILE_SERVER=$(COMPILE_USER)@pudim-flan.iobundle.com
	COMPILE_USER=$(USER)
	COMPILE_OBJ=output_files/top_system.sof
	BOARD_SERVER=$(BOARD_USER)@pudim-flan.iobundle.com
	BOARD_USER=$(USER)
else
#default
	BOARD=CYCLONEV-GT-DK
	COMPILE_OBJ=output_files/top_system.sof
endif
```

Edit the ./system.mk configuration file to configure the system with a new peripheral (continued: 4)

```
#ROOT DIR ON REMOTE MACHINES
REMOTE_ROOT_DIR=./sandbox/iob-soc

#ASIC
ASIC_NODE:=umc130

#DOC TYPE
DOC_TYPE:=presentation
```

Edit the `firmware.c` file to drive the new peripheral

`./software/firmware/firmware.c`

```
#include "system.h"
#include "periphs.h"
#include "iob-uart.h"
#include "iob_timer.h"

int main()
{
  unsigned long long elapsed;
  unsigned int elapsedu;

  //read current timer count, compute elapsed time
  elapsed = timer_get_count(TIMER_BASE);
  elapsedu = timer_time_us(TIMER_BASE);

  //init uart
  uart_init(UART_BASE, FREQ/BAUD);

  uart_printf("\nHello world!\n");

  uart_txwait();

  uart_printf("\nExecution time: %d clocks in %dus @%dMHz\n\n",
              (unsigned int)elapsed, elapsedu, FREQ/1000000);

  uart_txwait();
  return 0;
}
```

Run the firmware in internal SRAM

1. Initialize the SRAM with the firmware
   - Define `INIT_MEM=1`
   - Works in simulation and FPGA
   - Firmware may be recompiled and reloaded via UART afterwords
2. Do not initialize the SRAM with the firmware (default)
   - Works in simulation, FPGA and ASIC
   - The firmware is (re)compiled and (re)loaded via UART

Run the firmware in external DDR

1. Initialize the DDR with the firmware
    - Define `USE_DDR=1` and `INIT_MEM=1`
    - Works in simulation only
    - In FPGA or ASIC the external DDR cannot be initialized
2. Do not initialize the DDR with the firmware (default)
    - Define `USE_DDR=1`
    - This option is valid for simulation, FPGA and ASIC
    - The firmware is (re)compiled and (re)loaded via UART
    - In FPGA or ASIC a third party DDR controller IP core is required

Simulate IOb-SoC

- Simulation runs in directory ./hardware/simulation pointed by the SIMULATOR variable in system.mk
- To add your simulation directory in ./hardware/simulation for your simulator use the files in the existing simulation directories as examples.
- In file ./system.mk find and insert in the following section as given:

```
ifeq ($(SIMULATOR),ncsim)
        SIM_SERVER=$(SIM_USER)@micro7.lx.it.pt
        SIM_USER=user19
else ifeq ($(SIMULATOR),your_simulator)
        SIM_SERVER=$(SIM_USER)@your.simulation.server
        SIM_USER=your_username
else
#default
        SIMULATOR:=icarus
endif
```

Simulate IOb-SoC (continued)

- To run the simulation type `make sim INIT_MEM=1`
- The firmware, bootloader and system verilog description are compiled as you can see from the printed messages
- During simulation the following is printed:

```
IOb-SoC Bootloader:

Reboot CPU and run program...

Hello world!

Execution time: 6583 clocks in 66us @100MHz
```

Implement in FPGA

- Add your FPGA folder in `./hardware/fpga` using the other folders in there as examples
- In file `./system.mk`:
  1. Define `FPGA_BOARD` with the name of your FPGA folder
  2. Define `FPGA_BOARD_SERVER` with the URL or IP address of the computer connected to the board
  3. Define `FPGA_COMPILE_SERVER` with the URL or IP address of the computer containing the FPGA compiler and tools
  4. Define `FPGA_BOARD_ROOT_DIR` and `FPGA_COMPILE_ROOT_DIR` with the name of the remote root directories for the repository files
- To compile and load the hardware design in the FPGA, type `make fpga-load`
- To load and run your firmware in the FPGA, type `make run-firmware`

Implement in ASIC (WIP)

- Add your ASIC folder in ./hardware/asic using the other folders in there as examples
- In the file ./system.mk:
  1. Define ASIC_NODE with the name of your ASIC folder
  2. Define ASIC_COMPILE_SERVER with the URL or IP address of the computer containing the ASIC design tools
  3. Define ASIC_COMPILER_ROOT_DIR with the name of the remote root directory for the repository files
- To compile the ASIC, type make asic

Conclusions and future work

- Conclusions
  - A tutorial on SoC creation using IOb-SoC is presented
  - The addition of a peripheral IP core (timer) is illustrated
  - A simple software driver for the IP core is exemplified
  - How to compile and run the system is explained
  - Options for implementing the main memory are presented
  - Implementation of FPGA and ASIC is explained
- Future work
  - Non-volatile (flash) external memory support
  - Real Time Operating System (RTOS)