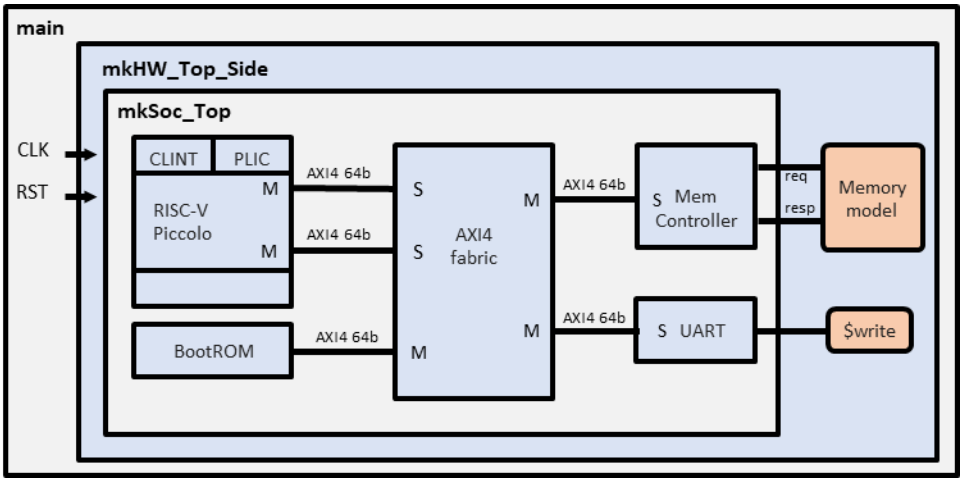
Demonstration of Piccolo running FreeRTOS in a Questa simulation Jun-23

This director contains the files and scripts for:

1. Compiling a Questa simulation executable from source files for the hardware system shown below
2. Running the simulation with a FreeRTOS “main_blinky” ELF file loaded into the Memory model and executed by the Piccolo processor. (The FreeRTOS C source code and Makefile is not packaged in this demo but is available by request).



mkSoC_Top: Connects the Piccolo processor is a BootROM, memory controller and UART through a 64-bit AXI4 fabric in module mkSoC_Top. The Piccolo Debug Module is not connected. To improve simulation speed, programs are loaded directly into the Memory model. The UART is not a fully functional UART for this demo. I’s used only in output mode to transmit program output through the $write simulation function.

mkHW_Top_side: Connects the simulation SoC to the Memory model. Also plumbs the CLK and RST signals form the level above.

main: Top level simulation wrapper providing a clock and reset and other simulation controls like verbosity.

Directory structure:

bin	scripts to compile and run the Questa simulation
build	Directory where Questa executable is compiled and run
C_VPI	VPI C files (e.g. for UART input when used)
lib/Verilog	Common Verilog modules (e.g. FIFO, BRAM)
lib/VPI	VPI library files
Piccolo_src	Snapshot of Piccolo output generated from BSV
programs	FreeRTOS main_blinky assembler and ELF files

tools	elf_to_hex tool for converting ELF files into hex files to be loaded into simulation memory
Verilog_RTL	Verilog files for Piccolo processor and SoC

Questa notes:

1. Version 2019.1 linux_x86_64 Jan 1 2019 run on Debian
2. The following was necessary to run Questa:
   export MTI_VCO_MODE=64

Running the demo:

```
$ cd build                    #from directory Bluespec_Questa_FreeRTOS

$ ../bin/compile_Questa.sh    #to build the Questa executable

$ ../bin/run_Questa.sh        #to load FreeRTOS and run the executable
```

After some initial simulation output, FreeRTOS will start running in about 90 second and until terminated with Ctrl-C will continue printing lines of the form:

```
# [0]: Hello from RX
#
# [0]: Hello from TX
#
# [1] TX: awoken
#
# [1] RX: received value
#
# Blink !!!
```

Here is a larger snapshot of the simulation output:

```
# 2019.1

# vsim -quiet -lib work_mkTop_HW_Side -pli "./directc_mkTop_HW_Side.so" -do "run -all;
quit" -c main
# Start time: 17:11:04 on Jun 23,2020
# ** Note: (vsim-8009) Loading existing optimized design _opt
# //  Questa Sim-64
# //  Version 2019.1 linux_x86_64 Jan  1 2019
# //
# //  Copyright 1991-2019 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  QuestaSim and its associated documentation contain trade
# //  secrets and commercial or financial information that are the property of
# //  Mentor Graphics Corporation and are privileged, confidential,
# //  and exempt from disclosure under the Freedom of Information Act,
# //  5 U.S.C. Section 552. Furthermore, this information
# //  is prohibited from disclosure under the Trade Secrets Act,
```

```
# //  18 U.S.C. Section 1905.
# //
# run -all
# ** Warning: (vsim-PLI-3412) Memory address read on line 33348 of file "Mem.hex" is
out of bounds specified for readmem command.  (Current address [33554431], address
range [0:8388607])    : ../lib/Verilog/RegFileLoad.v(80)
#    Time: 0 ns  Iteration: 0  Instance: /main/top/mem_model/rf
# ================================================================
# Bluespec RISC-V standalone system simulation v1.2
# Copyright (c) 2017-2019 Bluespec, Inc. All Rights Reserved.
# ================================================================
# 0: main.top.soc_top.boot_rom_axi4_deburster::AXI4_Deburster.rl_reset
# 0: main.top.soc_top.mem0_controller_axi4_deburster::AXI4_Deburster.rl_reset
# INFO: watch_tohost = 0, tohost_addr = 0xc00c3780
# 1:main.top.soc_top.rl_reset_start_initial ...
# 2: Core.rl_cpu_hart0_reset_from_soc_start
# ================================================================
# CPU: Bluespec  RISC-V  Piccolo  v3.0 (RV32)
# Copyright (c) 2016-2019 Bluespec, Inc. All Rights Reserved.
# ================================================================
# 5: D_MMU_Cache: cache size 4 KB, associativity 1, line size 32 bytes (= 8 XLEN
words)
# 5: I_MMU_Cache: cache size 4 KB, associativity 1, line size 32 bytes (= 8 XLEN
words)
# 135: main.top.soc_top.core.cpu.rl_reset_complete: restart at PC = 0x70000000
# 136: Near_Mem_IO_AXI4.set_addr_map: addr_base 0x10000000 addr_lim 0x10010000
# 136: Core.rl_cpu_hart0_reset_complete
# 137: Mem_Controller.set_addr_map: addr_base 0xc0000000 addr_lim 0x100000000
# 137:main.top.soc_top.rl_reset_complete_initial
# [0]: Hello from RX
#
# [0]: Hello from TX
#
# [1] TX: awoken
#
# [1] RX: received value
#
# Blink !!!
#
# [1]: Hello from RX
#
# [1] TX: sent
#
# [1]: Hello from TX
#
# [2] TX: awoken
#
# [2] RX: received value
#
# Blink !!!
#
# [2]: Hello from RX
#
# [2] TX: sent
#
# [2]: Hello from TX
#
# [3] TX: awoken
#
# [3] RX: received value
#
# Blink !!!
#
```

```
# [3]: Hello from RX
#
# [3] TX: sent
#
# [3]: Hello from TX
#
# [4] TX: awoken
#
# [4] RX: received value
#
# Blink !!!
...
```