Nuclei® E603 Integration Guide

Nucleisys

Contents

1 Copyright Notice

The product described herein is subject to continuous development and improvement; information herein is given by Nuclei in good faith but without warranties.

This document is intended only to assist the reader in the use of the product. Nuclei do not assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages. incorrect use of the product.

2 Contact Information

Should you have any problems with the information contained herein or any suggestions, please contact Nuclei System Technology by email support@nucleisys.com, or visit "Nuclei User Center" website http://user.nucleisys.com for supports or online discussion.

List of Figures

List of Tables

3 Preface

3.1 About this specification

This specification describes the Nuclei CPU IP Integrate book.

3.2 Glossary

Table 3.1: Glossary

Abbreviation	Description
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ATB	Advanced Trace Bus
ICB	Nuclei Internal Chip Bus
IDE	Integrated Development Environment
IDEV	Internal Device
JTAG	Joint Test Action Group
NMI	Non Mask Interrupt
SDK	Software Development Kit

4 Revision History

Rev.	Revision Date	Revised Section	Revised Content
1.0.0	2025/8/4	N/A	1.First version

5 Tool Requirement

5.1 Linux Operation Requirement

The release package only can be used in Linux environment, it can't work on Windows environment.

The support Linux we recommend:

Table 5.1: Linux Operation Requirement

Tool	Version
Centos	6/7
Redhat	6/7

5.2 Linux Tool Requirement

Table 5.2: Linux Tool Requirement

Tool	Minimum Version
python2 or python3	Any Please sure the command 'python' exists
make	3.82
tar	1.26
Nuclei Newlib GCC	2025.02-gcc14
Nuclei GLIBC GCC	2025.02-gcc14

Note: Nuclei GCC can be download from the nucleisys website, https://nucleisys.com/download.php

If Core doesn't config MMU, the Nuclei GLIBC GCC is not required.

5.2.1 Nuclei GCC Install

Nuclei GCC has two type:

- Newlib GCC: It is used by baremetal or RTOS.
- GLIBC GCC: It is used by Linux kernel or Linux Application.

5.2.1.1 Newlib GCC Install

For example, you want to install the GCC in /home/$USER/nuclei/newlib_gcc directory.

- Download the GCC tool from the nucleisys website(https://download.nucleisys.com/upload/files/toolchain/gcc/nuclei_riscv_newlibc_prebuilt_linux64_2025.02.tar.bz2) into the /home/$USER/nuclei/newlib_gcc directory
- Extract the compressed file
- Add the tool directory into PATH environment

```
mkdir -p /home/$USER/nuclei/newlib_gcc
cd /home/$USER/nuclei/newlib_gcc
tar -jxvf nuclei_riscv_newlibc_prebuilt_linux64_2025.02.tar.bz2
export PATH=/home/$USER/nuclei/newlib_gcc/gcc/bin:$PATH
```

5.2.1.2 GLIBC GCC Install

For example, you want to install the GCC in /home/$USER/nuclei/glibc_gcc directory.

- Download the GCC tool from the nucleisys website(https://download.nucleisys.com/upload/files/toolchain/linuxgcc/nuclei_riscv_glibc_prebuilt_linux64_2025.02.tar.bz2) into the /home/$USER/nuclei/glibc_gcc directory
- Extract the compressed file
- Add the tool directory into PATH environment

```
mkdir -p /home/$USER/nuclei/glibc_gcc
cd /home/$USER/nuclei/glibc_gcc
tar -jxvf nuclei_riscv_glibc_prebuilt_linux64_2025.02.tar.bz2
export PATH=/home/$USER/nuclei/glibc_gcc/gcc/bin:$PATH
```

5.3 EDA Tool Requirement

Table 5.3: EDA Requirement

Tool	Minimum Version
Synopsys VCS	2019.06-SP1
Synopsys VERDI	2019.06-SP1
Synopsys Design Compiler	2021.06-SP1
Synopsys Library Compiler	2020.09
Synopsys PrimeTime	2020.09-SP5-1
AMD Vivado	2018.03

6 Introduction of Release Package

6.1 Release Package

Release packages includes the Verilog RTL source codes, Evaluation SoC, Simulation Environment, Logic Synthesis and FPGA project.

The release package of Hummingbird E603 Core can be licensed from Nuclei.

> **Warning:** Please make sure the upper tar operation must run on Linux.

6.2 Files in Package

The files in the package are introduced as below.

```
e603
   |--doc              // Directory  for documents
   |--README.md        // Please read this for information
   |--design           // Directory for RTL
      |--core          // Directory for Core
      |--soc           // Directory for subsystem in SoC
      |--tech          // Directory for synthesis tech library related
   |--riscv-tests      // Directory for assemble testcases
   |--tb               // Directory for Verilog TestBench
   |--nuclei-sdk       // Directory for generated nuclei-sdk
   |--nuclei-linux-sdk// Directory for generated nuclei-linux-sdk configuration
   |--sdk-import       // Directory for nuclei-sdk importing files
   |--fpga             // Directory for FPGA project
   |--syn              // Directory for Synthesis project
   |--vsim             // Directory for Simulation
      |--bin           // Directory for functional scripts
      |--Makefile      // Makefile for simulation
```

6.3 Module Hierarchy of Core

For E603,the key points are:

6.3.1 E603 Core

- E603_core_wrapper is the top module of the Core, which include several key sub-modules:
  - e603_core: The Core part.
  - e603_rst_ctrl: The module to sync external async reset signal to synced reset with "Asynchronously assert and synchronously de-assert" style.
  - e603_dbg_top: The module to handle the debug functionalities.
- e603_ucore is under Core hierarchy, it is the main part of Core.
- Besides the e603_ucore, there are several other sub-modules:
  - e603_clic_top: The private interrupt controller.
  - e603_tmr_top: The private timer unit.
  - e603_clk_ctrl: The clock control module.

7 Top Level Integration

Nuclei Hummingbird E603 processor core' delivered RTL have unitive hierarchy and top level.

7.1 Clocks

Clocks to the Core are the baseline of the top level integration.

For the details of the e603 Series Cores' clocks, please refer to Section "Clock Domains" of the document <Nuclei_E603_Databook.pdf>.

7.2 Interfaces

The interfaces of Core need to be carefully checked during the top level integration.

For the details of the e603 Series Cores' interfaces, please refer to Chapter "Interfaces" of the document <Nuclei_E603_Databook.pdf>.

7.3 Memory Map

There are quite several interfaces and private peripherals for the Nuclei Hummingbird E603 Core, the address spaces of them are fixed.

For the details of the e603 Series Cores' memory map, please refer to Section "Address Spaces of Interfaces and Private Peripherals" of the document <Nuclei_E603_Databook.pdf>.

8 SoC for Evaluation

In order to facilitate evaluation of Nuclei Processor Core, the prototype SoC (called EvalSoC) is provided for evaluation purpose.

8.1 Eval_SoC Background

To make it easy for the user to evaluate Nuclei Processor Core, the prototype SoC (called Eval_SoC) is provided for evaluation purposes. This Eval_SoC includes:

- Nuclei Processor Core,
- Memory resources for instruction and data.
  - On-Chip SRAM.
  - External DDR.
- The SoC buses.
- The basic peripherals, such as UART, SPI.

With this prototype SoC, users can run simulations, map it into the FPGA board, and run with real embedded application examples.

Note:

- All modules including the peripherals in the Eval_SoC are for evaluation purpose and without Quality Guarantee/Supports, hence, it is not recommended to be used in any productions.
- If users really need the SoC solution and peripherals IP with Quality Guarantee/Supports, Nuclei provides SoC Subsystem solution which can help user to quickly make the SoC product.

Nuclei can provide a concrete SoC Subsystem based on user's requirements, also we can provide:

- Soc RTL.
- Testbench for simulation.
- FPGA Bitstream.
- Scripts for synthesis.
- SDK with peripheral drivers and test cases.
- Documents.

For more details, please visit https://www.nucleisys.com/subsystem.php or contact Nuclei Support.

8.2 Eval_SoC Diagram

The Eval_SoC diagram is as depicted in *Eval_SoC Diagram with IDEV inside CPU* (page 13).

There are:

- Nuclei Hummingbird E603 CPU
- System Bus
- Peripheral Bus
- Memory resources
- Peripherals

For more information of the peripherals, please refer to *Peripherals of Eval_SoC* (page 16).

In Nuclei CPU, below internal devices(called IDEV) location can config inside or outside cpu.

- Debug
- ECLIC
- TIMER
- PLIC

Warning: In evaluation rtl, IDEV can config to inside or outside CPU.

In official delivery RTL, IDEV is fixed inside CPU.

The CPU module in the figure below contains Debug/ECLIC/TIMER/PLIC module.

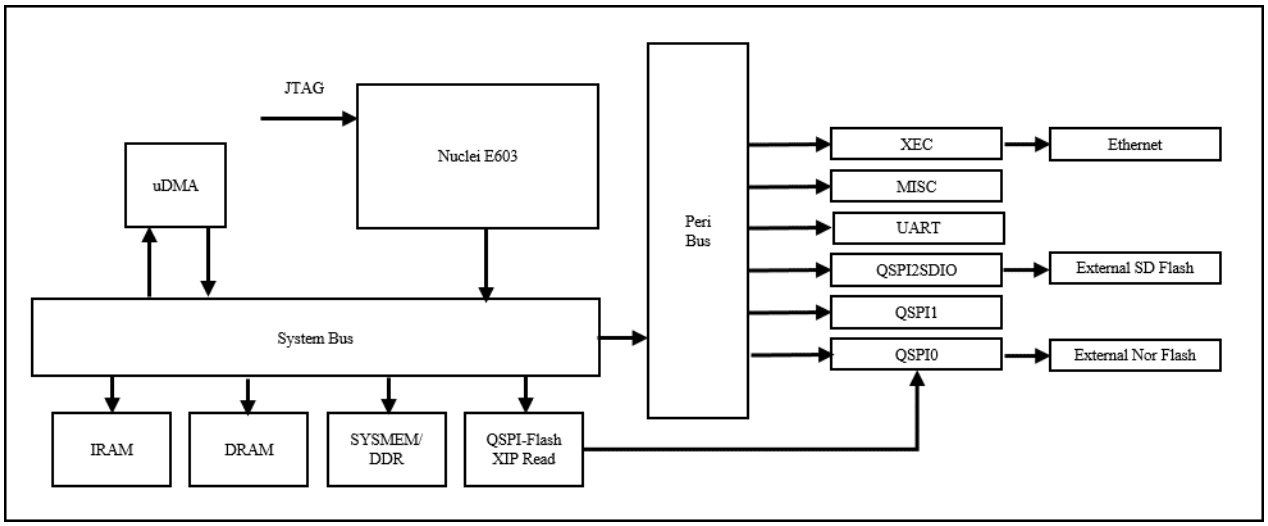


Fig. 8.1: Eval_SoC Diagram with IDEV inside CPU

8.3 Memory Resources of Eval_SoC

The Eval_SoC contains memory resources including on-chip SRAM, External Flash support, and DDR memory. The DDR support is primarily utilized on the FPGA prototyping platform.

8.3.1 On-Chip SRAM Resource

The Eval_SoC contained on-chip SRAM, detailed as below.

- IRAM with features:
  - Base address is fixed.
  - Size is fixed.
  - Mainly used to save the instructions, but also can be used to save the data.
  - Base address is allocated as shown in *Address Allocation of Eval_SoC Memory* (page 15).
  - The IRAM is connected to system bus, so the Core can access it with system bus interface.
- DRAM with features:
  - Base address is fixed.
  - Size is fixed.d.
  - Mainly used to save the data.
  - Base address is allocated as shown in *Address Allocation of Eval_SoC Memory* (page 15).
  - The SRAM is connected to system bus, so the Core can access it with system bus interface.

8.3.2 External Flash Support

The Eval_SoC supports external SPI Nor Flash with QSPI0 interface, detailed as below.

- The QSPI0 support the eXecute-In-Place (XIP) mode, and it is the default mode after reset.
- With XIP mode, external Flash can be treated as a read-only memory directly accessible
  - Base and Size are fixed.
  - default address space is 0x2000_0000 ~ 0x3FFF_FFFF in this SoC.
- Hence, the instructions program can be saved at external Flash (will not be lost even after power down), and then after power-on reset, the Hummingbird E603 Core can fetch instruction directly from the external Flash with XIP mode through this QSPI master interface, and start to execute the programs.
- Please refer to *QSPI* (page 19) for more info of QSPI XIP mode.

8.3.3 SYSMEM

The Eval_SoC has system memory, detailed as below:

- Base address is 0xa000_0000.
- Size is 2G for E603.
- It is ddr. The ddr uses FPGA Board's DDR IP, can easy customer to boot Linux.
- The DDR only exists in FPGA Board (not in Simulation Environment).

8.4 Address Allocation of Eval_SoC

8.4.1 Memory Address Allocation

The address allocation of the Eval_SoC memory is as shown in *Address Allocation of Eval_SoC Memory* (page 15).

Table 8.1: Address Allocation of Eval_SoC Memory

Component	Address Spaces	Description
IRAM	Base address: EVALSOC_IRAM_BASE_ADDR Size: 2^EVALSOC_IRAM_ADDR_WIDTH Default 32KB for 200/300, 64KB for E603.	IRAM address space.
DRAM	Base address: EVALSOC_DRAM_BASE_ADDR Size: 2^EVALSOC_DRAM_ADDR_WIDTH Default 32KB for 200/300, 64KB for E603.	DRAM address space.
Off-Chip QSPI0 Flash Read	Base address: EVALSOC_FLASH_XIP_BASE_ADDR Size: 2^EVALSOC_FLASH_XIP_ADDR_WIDTH	QSPI0 with XiP mode read-only address space. Default: 0x2000_0000 ~ 0x3FFF_FFFF
SYSMEM	Base address: EVALSOC_SYSMEM_BASE_ADDR Size: 2^EVALSOC_SYSMEM_ADDR_WIDTH	System Memory address space. Note: for E603 series core, the default range is 0x8000_0000 ~ 0xffff_ffff;

The address allocation of the Eval_SoC peripherals is as shown in table-Address_Allocation_of_Eval_SoC_perips.

Peripherals base address is configurable and use EVALSOC_PERIPS_BASE to config, default is 0x1000_0000.

The size is 1MB.

8.5 Reset PC Address

After reset, the reset PC address of the E603 Core could be from external Flash (XIP mode), or from other memory.

- In FPGA board, the reset PC address is from external Flash (XIP)
  - In the SoC, "QSPI Flash XIP Read" bus slave port has been allocated with default address space of 0x2000_0000 ~ 0x3FFF_FFFF at system bus.
  - Please refer to *Eval_SoC Diagram* (page 13) for more information of the Eval_SoC diagram, and refer to *QSPI* (page 19) for more info of QSPI XIP mode.
- In SIMULATION Environment, the reset PC address is default from the ELF file's start address and user can change it to use "BOOT_ADDR" argument.

8.6 Interrupts of Eval_SoC

The interrupts of Eval_SoC are managed by the interrupt controller of the core, the interrupts are directly connected to the external interrupt interface of the E603 processor Core.

Table 8.2: Allocation of External Interrupts in Eval_SoC

IRQ_ID	ECLIC Interrupt ID	Source
0 & 32	19 & 51	uart
1 & 33	20 & 52	uDMA

continues on next page

Table 8.2 – continued from previous page

IRQ_ID	ECLIC Interrupt ID	Source
2 & 34	21 & 53	qspi0
3 & 35	22 & 54	qspi1
4 & 36	23 & 55	qspi2
5 & 37	24 & 56	Counter0 irq
6 & 38	25 & 57	Counter1 irq
7	26	Ethernet irq
8	27	SDIO irq

Note:

- IRQ_ID is the hardware interrupt wire connect number and ECLIC Interrupt ID is software concept.
- To be compatible with interrupt source of the Core is less than 32, some external interrupt are connected to two IRQ inputs.

8.7 Peripherals of Eval_SoC

The Chapter will shortly introduce the peripherals used in the Eval_SoC.

8.7.1 MISC

This component is to:

- Control the input of Core.
- Control some Evalsoc behavior.

Table 8.3: Register Description for MISC

Register	Offset	Description
Counter0_IRQ	0x0	Set the interrupt0 initial counter value. *Interrupt Counter Register* (page 16)
Counter1_IRQ	0x4	Set the interrupt1 initial counter value. *Interrupt Counter Register* (page 16)
MEM_Latency	0x2C	Set the system memory latency. *Memory Latency Register* (page 17)
NMI	0x100	Set the nmi initial counter value. *NMI Counter Register* (page 17)
Event	0x104	Set the event initial counter value. *Event Counter Register* (page 17)
Debug_Ctrl	0x108	Control the input debug of CPU. *Debug Ctrl Register* (page 17)
Misc_ctrl	0x10C	Evalsoc some behavior control. *Misc Ctrl Register* (page 18)
Version	0xF00	Indicate EvalSoC's version. *Version Register* (page 18)

8.7.1.1 Interrupt Counter Register

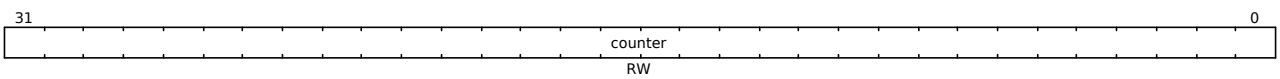
In Misc component, we implement two interrupt counter.

When counter value decrease to 0, it will generate level interrupt. And interrupt will clear when software write 0xffff_ffff into counter.

When the counter value is 0xffff_ffff, it will not decrease.

Table 8.4: Interrupt Counter Register

Field Name	Bits	Reset Value	Description
counter	31:0	0xffff_ffff	counter.



8.7.1.2 Memory Latency Register

This register can config the latency for system memory in evalsoc.

Table 8.5: Memory Latency Register

Field Name	Bits	Reset Value	Description
Reserved	31:9	0	Reserved.
latency	8:0	0	latency.

8.7.1.3 NMI Counter Register

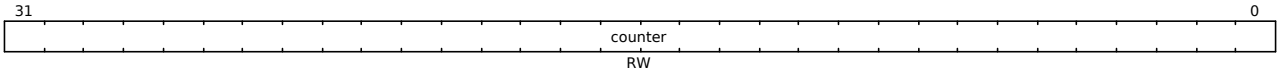
In Misc component, we implement one NMI counter.

When counter value decrease to 0, it will generate NMI. And NMI will clear when software write 0xffff_ffff into counter.

When the counter value is 0xffff_ffff, it will not decrease.

Table 8.6: NMI Counter Register

Field Name	Bits	Reset Value	Description
counter	31:0	0xffff_ffff	NMI Counter.



8.7.1.4 Event Counter Register

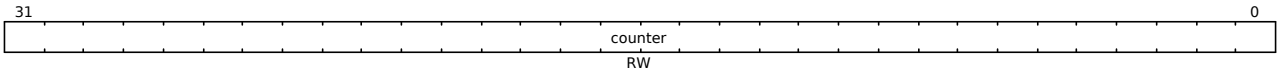
In Misc component, we implement one event counter.

When counter value decrease to 0, it will generate event. And event will clear when software write 0xffff_ffff into counter.

When the counter value is 0xffff_ffff, it will not decrease.

Table 8.7: Event Counter Register

Field Name	Bits	Reset Value	Description
counter	31:0	0xffff_ffff	Event Counter.



8.7.1.5 Debug Ctrl Register

This register control the input debug signal of CPU.

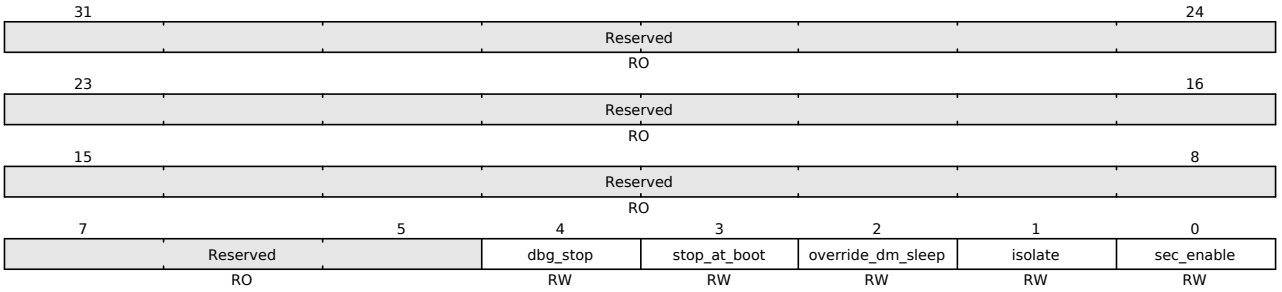
Table 8.8: Debug Ctrl Register

Field Name	Bits	Reset Value	Description
Reserved	31:5	0	Reserved.
dbg_stop	4	0	Control the dbg_i_dbg_stop input.
stop_at_boot	3	0	Control the dbg_stop_at_boot input.

continues on next page

Table 8.8 – continued from previous page

Field Name	Bits	Reset Value	Description
override_dm_sleep	2	0	Control the dbg_override_dm_sleep input.
isolate	1	0	Control the dbg_isolate input.
sec_enable	0	1	Control the dbg_sec_enable input.

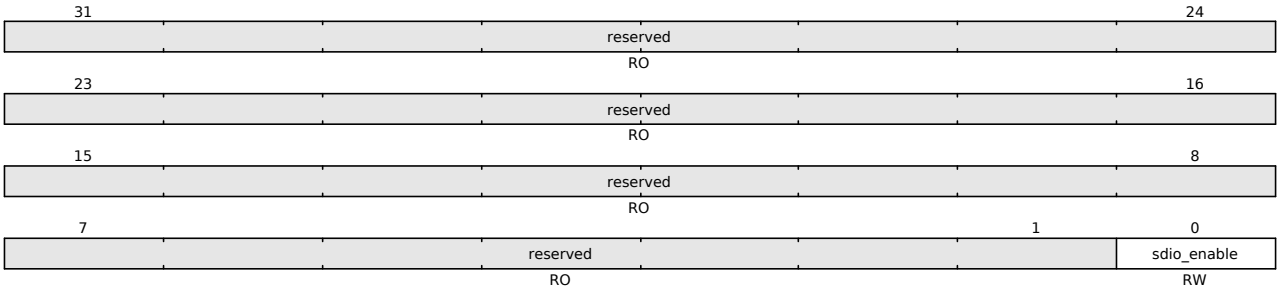


8.7.1.6 Misc Ctrl Register

This register control some evalsoc behavior.

Table 8.9: Misc Ctrl Register

Field Name	Bits	Reset Value	Description
Reserved	31:1	0	Reserved
sdio_enable	0	0	Control the SD card driven source. 0: QSPI2 drive the SD card (default) 1: SDIO drive the SD card

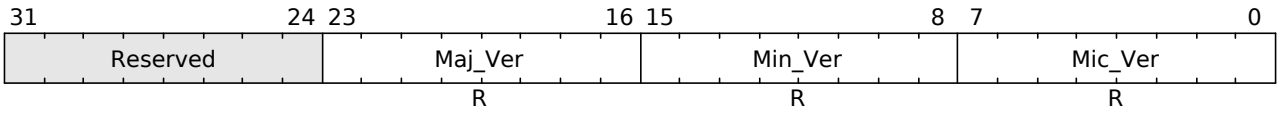


8.7.1.7 Version Register

This register indicate evalsoc version.

Table 8.10: Evalsoc Version Register

Field Name	Bits	Reset Value	Description
Mic_Ver	7:0	X	Micro Version Number
Min_Ver	15:8	X	Minor Version Number
Maj_Ver	23:16	X	Major Version Number
Reserved	31:24	0	Reserved 0



8.7.2 QSPI

There are 2 independent QSPI IP developed by Nucleisys in the SoC which can communicate with outside.

- One QSPI for Flash:
  - There is one QSPI master in the SoC which has dedicated SoC ports to communicate with external FLASH (e.g. Nor Flash with SPI interface).
  - This QSPI support the eXecute-In-Place (XIP) mode, and it is the default mode after reset. With XIP mode, external Flash can be treated as a read-only memory directly accessible. Hence, the instructions program can be saved at external Flash (will not be lost even after power down), and then after power-on reset, the Processor Core can fetch instruction directly from the external Flash with XIP mode through this QSPI master interface, and start to execute the programs.
- Another QSPI for Sd-card:
  - There are one QSPI master (without XIP mode supported) with 1 CS signal.

Note:

- This QSPI in the Eval_SoC is for evaluation purpose and without Quality Guarantee/Supports, hence, it is not recommended to be used in any productions.
- If user really needs the SoC solution or QSPI with Quality Guarantee/Supports, Nuclei provides SoC Subsystem which can help user to quickly make the SoC product. Please visit https://www.nucleisys.com/subsystem.php or contact Nuclei Support.

8.7.3 UART

There is one independent UART (Universal Asynchronous Receiver-Transmitter) IP developed by Nucleisys in the SoC, which can communicate with outside.

Note:

- This UART in the Eval_SoC is for evaluation purpose and without Quality Guarantee/Supports, hence, it is not recommended to be used in any productions.
- If user really needs the SoC solution or UART with Quality Guarantee/Supports, Nuclei provides SoC Subsystem which can help user to quickly make the SoC product. Please visit https://www.nucleisys.com/subsystem.php or contact Nuclei Support.

8.7.4 uDMA

There is one independent uDMA ((Micro Direct Memory Access) IP developed by Nucleisys in the SoC, which can communicate with outside.

Note:

- This uDMA in the Eval_SoC is for evaluation purpose and without Quality Guarantee/Supports, hence, it is not recommended to be used in any productions.
- If user really needs the SoC solution or uDMA with Quality Guarantee/Supports, Nuclei provides SoC Subsystem which can help user to quickly make the SoC product. Please visit https://www.nucleisys.com/subsystem.php or contact Nuclei Support.

8.7.5 SDIO

There is one independent SDIO IP developed by Nucleisys in the SoC, which can communicate with outside.

Note:

- This SDIO in the Eval_SoC is for evaluation purpose and without Quality Guarantee/Supports, hence, it is not recommended to be used in any productions.
- If user really needs the SoC solution or SDIO with Quality Guarantee/Supports, Nuclei provides SoC Subsystem which can help user to quickly make the SoC product. Please visit https://www.nucleisys.com/subsystem.php or contact Nuclei Support.

8.7.6 Ethernet

There is one independent Ethernet IP developed by Nucleisys in the SoC, which can communicate with outside.

Note:

- This Ethernet in the Eval_SoC is for evaluation purpose and without Quality Guarantee/Supports, hence, it is not recommended to be used in any productions.
- If user really needs the SoC solution or Ethernet with Quality Guarantee/Supports, Nuclei provides SoC Subsystem which can help user to quickly make the SoC product. Please visit https://www.nucleisys.com/subsystem.php or contact Nuclei Support.

---

9 Software Development Kit (SDK)

Nuclei has created a "Nuclei Software Development Kit (Nuclei SDK)" which is an open software platform to facilitate the software development for systems based on Nuclei Processor Cores. For more details about Nuclei-SDK, please see it from https://github.com/Nuclei-Software/e603_hbird/tree/main/nuclei-sdk.

Based on the "Nuclei EvalSoC", and with the demo software projects from Nuclei SDK, user can quickly familiarize the software development for Nuclei Processor Cores.

9.1 SDK Test

Nuclei develop many cases based on nuclei-sdk, and these cases can run in simulation env.

The case is in nuclei-sdk/application directory.

9.2 Run SDK Test In Simulation

SDK test can be run in simulation env.

In vsim dir, user can use makefile run_sdk command to run sdk test.

9.3 IDE Import SDK

If you do the importing, you should:

- Using zip tool to compress the nuclei-sdk directory into nuclei-sdk.zip
- Refer to "How to import local Nuclei-sdk into IDE(https://doc.nucleisys.com/nuclei_tools/ide/npk.html#npk-import-local-package)" to do subsequent operations.

10 Nuclei Linux SDK

Nuclei has created a "Nuclei Linux Software Development Kit (Nuclei Linux SDK)" which is an open software platform to facilitate the Linux software development for systems based on Nuclei Hummingbird E603 Processor Cores. For more details about Nuclei-Linux-SDK, please see it from https://github.com/Nuclei-Software/nuclei-linux-sdk/tree/dev_nuclei_6.6_v3_e603.

Please refer to nuclei-linux-sdk/README.md to know more details.

Warning: Only the CPU has MMU feature, can use Nuclei Linux SDK.

11 FPGA and IDE for Evaluation

11.1 FPGA Evaluation Board and JTAG Debugger

Nuclei have customized 3 types FPGA evaluation boards, called While Nuclei offers several evaluation boards, the DDR200T Evaluation Kit is the recommended hardware platform for the Hummingbird E603 core. Its on-board DDR memory is essential for running complex software, including a full Linux operating system, which is one of the key capabilities of the E603.

(please go to Nuclei website for details of the board: https://www.nucleisys.com/developboard.php).

The FPGA boards can be used as the SoC prototype board directly:

- If the FPGA have been pre-burned (programmed) with "Nuclei EvalSoC", this board can be worked as a SoC prototype. The embedded software engineers can use this board without knowing any FPGA hardware knowledge.
- About how to generate the FPGA Bitstream (MCS) with pre-built FPGA project, please refer to *FPGA Prototyping* (page 49).

Nuclei have customized a Debugger hardware (called Hummingbird Debugger Kit), which follows RISC-V Debug Spec and can be used to debug the RISC-V core in FPGA prototype or in real chip.

For the detailed introduction of the "Hummingbird Debugger Kit", please refer to "Development Boards" page of Nuclei website (http://www.nucleisys.com/developboard.php).

11.2 Integrated Development Environment (IDE)

There are many choices for user to select an IDE to develop software based on Nuclei Hummingbird E603 processor core. Here introduces Nuclei Studio first , then third-party IDEs.

11.2.1 Nuclei Studio

Nuclei Studio is an Eclipse based IDE developed and maintained by Nuclei. It has followed advantages for users who want to develop software on the core:

- Free to download and use.
- Support Nuclei Hummingbird E603 core.
- Pre-integrated RISC-V GCC/GDB and Nuclei OpenOCD.
- Pre-integrated Nuclei SDK.
- Support Win10/Ubuntu/Redhat OS.
- One Click for project templates.

Please see the Nuclei Website https://www.nucleisys.com/download.php to get the install package and user guide. Please refer follow picture:

Fig. 11.1: Nuclei Studio IDE Install Package and User Guide

11.2.2 Third-Party IDEs

Nuclei collaborates with many famous IDEs vendors ,such as Segger, Lauterbach, IAR, COMPILER,etc. Their IDEs and relate tools can fully support Nuclei Hummingbird E603 Core based chips and DDR200T.

Please see the Nuclei Website https://www.nucleisys.com/download.php to get the details. For the quick-start of Segger Embedded Studio (SES) for Nuclei Core, there are two documents which can be downloaded from this page.

12 Simulation

Nuclei Hummingbird E603 release package provide one simulation environment, and the simulation environment based on VCS eda tool.

We suggest the VCS version is newer than 2019.06-SP1.

12.1 Introduction of Testbench

The Verilog Testbench source codes are under the "tb" directory as below.

```
e603
    |----tb
       |----tb_*.v //Verilog TestBench source codes
```

The functionality of Testbench is briefly introduced as below:

- Instantiated DUT.
- Generate the clock and reset.
- According to the run options, to recognize the Testcase name, and then use elf2mem to read the elf file and then initialize the memory in SoC
- At the end of the simulation, check the value of X3, if the X3 value is 1, then the test is passed, print the "PASS" on the terminal, otherwise it is failed and printed as "FAIL", as shown in *Print the PASS or FAIL in Testbench* (page 26)

Note:

- User can also integrate these tb_*.v files into their SoC environment, such that in the user's SoC, the Testcase can also be as the sanity Testcases.
- However, these above Testcases are very basic tests, which cannot guarantee the full coverage. If users have modified the Core's RTL code, should not assume the functional correctness can be verified by running the above Testcases.

```
@(pc_write_to_host_cnt == 32'd8)

    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~ Test Result Summary ~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~TESTCASE: %s ~~~~~~~~~~~~~~", testcase);
    $display("~~~~~~~~~~~~~~~Total cycle_count value: %d ~~~~~~~~~~~~~~", cycle_count);
    $display("~~~~~~~~~~~The valid Instruction Count: %d ~~~~~~~~~~~~~~", valid_ir_cycle);
    $display("~~~~~The test ending reached at cycle: %d ~~~~~~~~~~~~~~", pc_write_to_host_cycle);
    $display("~~~~~~~~~~~~~~~~The final x3 Reg value: %d ~~~~~~~~~~~~~~", x3);
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
if (x3 == 1) begin
    $display("~~~~~~~~~~~~~~~~~ TEST_PASS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~ #####     ##     ####    #### ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~ #    #   #  #   #       #     ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~ #    #  #    #   ####    #### ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~ #####   ######       #       #~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~ #       #    #  #    #  #    #~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~ #       #    #   ####    #### ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
end
else begin
    $display("~~~~~~~~~~~~~~~~~ TEST_FAIL ~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~######    ##       #    #     ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~#        #  #      #    #     ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~#####   #    #     #    #     ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~#       ######     #    #     ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~#       #    #     #    #     ~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~#       #    #     #    ######~~~~~~~~~~~~~~~~");
    $display("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
end
```

Fig. 12.1: Print the PASS or FAIL in Testbench

12.2 Steps to Run Simulation

The steps to run simulation are as below (use e603 for example):

```
// Note: Before operation, it is required to install the “RISC-V
GNU Toolchain”. The toolchain can be downloaded from Nuclei website
(https://www.nucleisys.com/download.php).

// After the “RISC-V GNU Toolchain” package downloaded and
decompressed, there will be a “bin” directory under GCC folder. User
needs to add this “bin” path into the Linux $PATH environment
variable.

// Step 1: Generate the tests.

cd e603/vsim
    // Enter into the vsim directory.

make clean
    // Clean up the directory.

make install
    // Use this command to generate the Testbench and compile the riscv-tests


// Step 2: Compile RTL.

make compile
    // Compile the Verilog source codes.
```

(continues on next page)

(continued from previous page)

```
// Step 3: If want to run one single testcase, use the following commands.

make run_test TESTNAME=rv64ui-p-add
    // This command will run the simulation for one Testcase “rv64ui-p-add”
    // from riscv-tests/isa/generated directory.

make wave TESTNAME=rv64ui-p-add
    // This command will check the generated waveform.


// Step 4: If want to run the regression, use the following commands.

make regress_run
    // This command will run the regression for all the tests from
    // riscv-tests/isa/generated directory.

make regress_collect
    // This command will collect the simulation result for regression. It
    // will print a summary result, with each line for each testcase. For
    // each line, if the Testcase is passed then marked as “PASS”, otherwise
    // as “FAIL”.

// Step 5: If want to run dhrystone/coremark, use the following commands.

make run_sdk TESTNAME=dhrystone
    // This command will run the dhrystone program in nuclei-sdk.

make run_sdk TESTNAME=coremark
    // This command will run the coremark program in nuclei-sdk.
```

Note:

- make help will list all commands supported.
- In Step1, make install will check if the riscv gcc toolchain is installed or not, then compile the riscv-test or not. And if there is multi version riscv gcc toolchain including gcc 13, it will use gcc 13.
- In Step2, make compile needs the environment has VCS tool, we have verified vcs2019-06-SP1 and later version. If your VCS version is older, please contact Nuclei Support.
- In Step 3, our EvalSoC default set the reset_vector of core to flash xip base address, and in simulation env, the reset_vector will be forced to elf start address. if user want to set the reset_vector to other address, please use BOOT_ADDR argument just like:

  make run_test TESTNAME=rv64ui-p-add BOOT_ADDR=a0000000

  it means the reset vector is 0xa000_00000, the value of BOOT_ADDR is always hex number without “0x” prefix.
- In Step5, if the dhrystone or coremark result is not as expected, please contact Nuclei Support.

13 Simulation with Simple Assembly Testcase

13.1 Overview of Self-Check Testcase

The "Self-Check Testcase" is a kind of assembly Testcase which can self-check if it is "passed" or "failed". The Self-Check Testcase are under the following directory.

```
e603
   |----riscv-tests
      |----isa_origs // The directory for the source codes of Self-Check Testcases.
```

The "Self-Check Testcase" will set some "expected value" at the check-point, if the "real result" is not as the expected, then it will jump to the label of TEST_FAIL, otherwise it will continue to run until it reach the final ending label of TEST_PASS.

For example, as shown in *The code segment of add.S test* (page 29), the Testcase (source code under isa_origs/rv32ui/add.S) is to test the "add" instruction to compute two operands' addition (e.g., 0x00000003 and 0x00000007) , and then set its expected value (e.g., 0x0000000a). And then use the "compare" instruction to compare the "real result" is as expected or not, if not matched, then the test will jump to TEST_FAIL.

At the label of TEST_PASS, the test will set the value of general register X3 to 1; while at the label of TEST_FAIL, the test will set the value of general register X3 to "not 1". Hence, the testbench can monitor the final X3 value to check the Testcase is passed or failed.

```
RVTEST_CODE_BEGIN

  #-------------------------------------------------------------
  # Arithmetic tests
  #-------------------------------------------------------------

  TEST_RR_OP( 2,  add, 0x00000000, 0x00000000, 0x00000000 );
  TEST_RR_OP( 3,  add, 0x00000002, 0x00000001, 0x00000001 );
  TEST_RR_OP( 4,  add, 0x0000000a, 0x00000003, 0x00000007 );

  TEST_RR_OP( 5,  add, 0xffffffffffff8000, 0x0000000000000000, 0xffffffffffff8000 );
  TEST_RR_OP( 6,  add, 0xffffffff80000000, 0xffffffff80000000, 0x00000000 );
  TEST_RR_OP( 7,  add, 0xffffffff7fff8000, 0xffffffff80000000, 0xffffffffffff8000 );

  TEST_RR_OP( 8,  add, 0x0000000000007fff, 0x0000000000000000, 0x0000000000007fff );
  TEST_RR_OP( 9,  add, 0x000000007fffffff, 0x000000007fffffff, 0x0000000000000000 );
  TEST_RR_OP( 10, add, 0x0000000080007ffe, 0x000000007fffffff, 0x0000000000007fff );

  TEST_RR_OP( 11, add, 0xffffffff80007fff, 0xffffffff80000000, 0x0000000000007fff );
  TEST_RR_OP( 12, add, 0x000000007fff7fff, 0x000000007fffffff, 0xffffffffffff8000 );

  TEST_RR_OP( 13, add, 0xffffffffffffffff, 0x0000000000000000, 0xffffffffffffffff );
  TEST_RR_OP( 14, add, 0x0000000000000000, 0xffffffffffffffff, 0x0000000000000001 );
  TEST_RR_OP( 15, add, 0xfffffffffffffffe, 0xffffffffffffffff, 0xfffffffffffffffd );

  TEST_RR_OP( 16, add, 0x0000000080000000, 0x0000000000000001, 0x000000007fffffff );

  #-------------------------------------------------------------
  # Source/Destination tests
  #-------------------------------------------------------------

  TEST_RR_SRC1_EQ_DEST( 17, add, 24, 13, 11 );
  TEST_RR_SRC2_EQ_DEST( 18, add, 25, 14, 11 );
  TEST_RR_SRC12_EQ_DEST( 19, add, 26, 13 );
```

Fig. 13.1: The code segment of add.S test

13.2 Testbench to Initialize Self-Check Testcase

In order to have the Self-Check Testcase simulated in the Verilog Testbench, it is needed to convert the Testcase into the binary file with the format which can be initialized by Verilog Testbench.

The binary file (.verilog file) for each Testcase will be generated under riscv-tests/isa/generated directory, exampled as below.

```
e603
    |----riscv-tests
       |----isa
          |----generated
             |----rv64ui-p-addi         // The generated Elf file
             |----rv64ui-p-addi.dump    // The disassembly code
                    ...               // more cases ...
```

The content of disassembly code (e.g., rv64ui-p-addi.dump) is as shown in *The content of rv32ui-p-addi.dump file* (page 30).

```
rv32ui-p-add:     file format elf32-littleriscv


Disassembly of section .text.init:

80000000 <_start>:
80000000:   a081                    j   80000040 <reset_vector>
80000002:   0001                    nop

80000004 <trap_vector>:
80000004:   34202f73            csrr    t5,mcause
80000008:   4fa1                    li  t6,8
8000000a:   03ff0663            beq t5,t6,80000036 <write_tohost>
8000000e:   4fa5                    li  t6,9
80000010:   03ff0363            beq t5,t6,80000036 <write_tohost>
80000014:   4fad                    li  t6,11
80000016:   03ff0063            beq t5,t6,80000036 <write_tohost>
8000001a:   80000f17            auipc   t5,0x80000
8000001e:   fe6f0f13            addi    t5,t5,-26 # 0 <_start-0x80000000>
80000022:   000f0363            beqz    t5,80000028 <trap_vector+0x24>
80000026:   8f02                    jr  t5
80000028:   34202f73            csrr    t5,mcause
8000002c:   000f5363            bgez    t5,80000032 <handle_exception>
80000030:   a009                    j   80000032 <handle_exception>

80000032 <handle_exception>:
80000032:   5391e193            ori gp,gp,1337

80000036 <write_tohost>:
80000036:   00001f17            auipc   t5,0x1
8000003a:   fc3f2523            sw  gp,-54(t5) # 80001000 <tohost>
8000003e:   bfe5                    j   80000036 <write_tohost>

80000040 <reset_vector>:
80000040:   f1402573            csrr    a0,mhartid
80000044:   e101                    bnez    a0,80000044 <reset_vector+0x4>
80000046:   4181                    li  gp,0
80000048:   00000297            auipc   t0,0x0
8000004c:   fbc28293            addi    t0,t0,-68 # 80000004 <trap_vector>
80000050:   30529073            csrw    mtvec,t0
80000054:   80000297            auipc   t0,0x80000
80000058:   fac28293            addi    t0,t0,-84 # 0 <_start-0x80000000>
8000005c:   00028e63            beqz    t0,80000078 <reset_vector+0x38>
80000060:   10529073            csrw    stvec,t0
```

Fig. 13.2: The content of rv32ui-p-addi.dump file

14 Simulation with Comprehensive C Program

If user wants to run simulation with comprehensive C program, then the "Nuclei-SDK" is needed. For more details about Nuclei-SDK, please see its online doc from http://doc.nucleisys.com/nuclei_sdk.

14.1 Run SDK Test

Take "helloworld" from Nuclei-SDK as example, user can use the following steps to make it running under simulation environment.

```
// Step 1: install tb

cd e603/vsim

make install

// Step 2: run nuclei-sdk helloworld test

make run_sdk TESTNAME=helloworld

// Step 3: open waveform

make wave_sdk TESTNAME=helloworld
```

14.2 Run ELF Test

If you want to run elf immediately, use below command to run

```
// Step 1: install tb

cd e603/vsim

make install

// Step 2: run your test

make run_test TESTCASE=your_elf_path

// Step 3: open waveform

make wave TESTCASE=your_elf_path
```

15 Simulation with Benchmark

Nuclei-sdk provide three benchmark test, and these test can run in simulation env.

15.1 Run Dhrystone

Note: Here we use e603 Core as example case.

```
// Step 1: cd vsim dir

cd e603/vsim

// Step 2: run nuclei-sdk dhrystone test

make run_sdk TESTNAME=dhrystone

// Step 3: open waveform

make wave_sdk TESTNAME=dhrystone
```

15.2 Run Coremark

Note: Here we use E603 Core as example case.

```
// Step 1: cd vsim dir

cd e603/vsim

// Step 2: run nuclei-sdk coremark test

make run_sdk TESTNAME=coremark

// Step 3: open waveform

make wave_sdk TESTNAME=coremark
```

16 Spyglass

Core RTL code has been full checked by spyglass tools.

16.1 RTL Lint

User can ignore below spyglass lint:

- FlopEConst
- FlopSRConst
- W34: Macro ‘xxx’ is never used
- W111: Not all elements of array ‘xxx’ are ‘read’.
- W120: Variable ‘xxx’ declared but not used.
- W146: Explicit named association is recommended in instance references.
- W175: Parameter ‘xxx’ declared but not used.
- W240: Input “xxx” declared but not read.
- W287b: Instance output port “xxx” is not connected.
- W310: Converting integer to reg.
- W313: Converting integer to single bit.
- W341: Constant xxx is extended by 0.
- W415a: Signal “xxx” is being assinged multiple times (assignment within same for-loop) in same always block.
- W446: Output port ‘xxx’ is being read inside module.
- W527: Potential dangling ‘else’ statement of ‘if’ statement.
- W528: Variable “xxx” set but not read.
- W563: Use of unary reduction operator “x” on a 1-bit expression.
- Notab: Use spaces rather than tabs for indentation.
- ExprParen: Use parenthesis in complex expressions.
- RegOutputs: Port ‘xxx’ is not driven by a register.
- Indent: Statement xxx.
- ClkName: Clock signal name ‘xxx’ should conform to the regular expression.
- ConstName: Constant ‘xxx’ does not follow recommended naming convention
- InstName: Instance name ‘xxx’ does not follow naming convention.

- SigName: Signal xxx does not follow naming convention.
- VarName: Variable xxx does not follow naming convention.
- ParamName: Parameter name xxx does not follow naming convention
- PortComment: Port declaration does not have comments.
- SignalComment: Signal declaration does not have comments.
- VariableComment: Variable declaration does not have comments.
- LineLength: Line length xxx is more than yyy characters.
- STARC05-1.1.1.1: Filename 'xxx' does not indicate module and should be 'yyy'
- STARC05-1.1.4.6a: Port 'xxx' is 'tied-high'.
- STARC05-2.3.1.3: Insert delay in flip-flop data path
- STARC05-2.3.5.1: Flip-flop 'xxx' has fixed input value 'y'
- STARC05-2.10.5.1: Logic 'xxx' with operand bit-width greater than 8 is common with conditional expression.
  - xxx_gnrl.v
- STARC05-2.10.6.6:n The expression 'xxx' has more than one arithmetic operators
- STARC05-3.1.3.1: Order of specification of port 'xxx' not consistent with port declaratio.
- ConstDrivenNet-ML: Wire xxx is assigned a constant value 1'b0/1'b1.
- UnloadedInPort-ML: Detected unloaded(unconnected)input port "xxx".
- UnloadedNet-ML: Detected unloaded(unconnected) net.
- UnloadedOutTerm-ML: etected unloaded(unconnected) output terminal.
- UndrivenNUnloaded-ML: Detected undriven and unloaded(unconnected) net "xxx".
- UnInitParam-ML: Instance xxx instantiated but parameter not specified
- NoConstSourceInAlways-ML: Signal 'xxx' is assigned a constant value 'yyy' in implicit always block.
- NoExprInPort-ML: Expressions "xxx" used in port connection.
- ConstantInput-ML: Input 'xxx' to Instance 'yyy' is assigned a constant.
- ParamOverrideMismatch-ML: Mismatch in number of parameter over-rides and number of parameters.
- GenvarUsage-ML: Genvar variable 'i' declared but not used.
- SelfAssignment-ML: Same operand 'xxx' used on both sides of an assignment.
- OneModule-ML: File 'xxx' contains more than one module
- ModuleName-ML: Module 'xxx' does not follow the recommended naming convention
- HangingFlopOutput-ML: Flop output "xxx" connected to module output "yyy" is unused.

16.2 CDC Lint

User can ignore below spyglass cdc lint for some module or signals:

- Ac_cdc01a
  - Fast(core_clk_aon) to slow(jtag_tck) clock
- Ac_datahold01a
  - Destination flop "***.u_dbg_top.u_dbg.***"
- Ac_datahold01a
  - Designation flot "***.dtm_tap_aon_top.***"
- Ac_glitch03

  - Destination flop "***.dtm_tap_aon_top.***"
- Clock_glitch05
  - Asynchronous source '***.reset_bypass'
  - Asynchronous source '***.clkgate_bypass'
- Clock_sync06
  - Primary output signal "ram_sync_reset_n"
- Reset_check12
  - flop "***" is not asynchronously asserted
- Ac_conv03
  - synchronizers ***.dtm_tap_aon_top.***
- Ac_conv04
  - At least two bits (0, 1) of source bus "***.u_dbg_top.u_dbg.***"
- Ac_datahold01a
  - Synchronized crossing: destination flop "***.u_dbg_top.***"
- Reset_sync04
  - Asynchronous reset signal '***.por_reset_n' is synchronized at least twice

17 SRAM Memory Integration

17.1 About SRAM Memory integration

When RTL is generated, the SRAM wrapper blocks of the CPU model contain correctly sized generic SRAM models.

Although these SRAM modes are suitable for logical simulation, they cannot be implemented without change. You must substitute your physical SRAM for the generic SRAM models to correctly floorplan, synthesize, and time the design.

17.2 SRAM Requirements

You must ensure that physical SRAMs incorporated in the Nuclei CPU substituted for the generic SRAM satisfy these requirements:

- All timings must be synchronous to the master rising clock edge.
- There is single-cycle access at the main clock frequency.
- SRAM accesses must be suppressed during production test, preferable by inhibiting chip select.
- The clock can be held static during inactive cycles because there are no minimum frequency requirements.
- Because SRAM outputs must always be driven, you must not use tristate logic. However, if the SRAM is disabled for a given cycle:
  - The outputs are not required to retain the value read in the last level access.
  - The SRAM outputs are not required to be meaningful during a write cycle.
- Variable bit-write control is required for some SRAM blocks.
- Verilog simulation models are required to run the SRAM integration testbench.

These first two requirements mean that control, address, and write data inputs are set up before the rising clock edge.

This setup time must be of some percentage value that is less than 100% of the period of the master clock. There can be significant logic external to the SRAM within the setup cycle. To maximize frequency, you must select SRAM architectures with minimal setup time.

The output read data is launched from the same rising clock edge, and the clock-to-output delay must be of some percentage value that is less than 100% of the period of the main clock.

17.3 SRAM Pipeline Timing

Based on the SRAM data out timing, SRAM include below:

- one cycle sram pipeline
- two cycle sram pipeline
- three cycle sram pipeline
- four cycle sram pipeline

17.3.1 One Cycle SRAM Pipeline

One Cycle SRAM pipeline timing is as below:

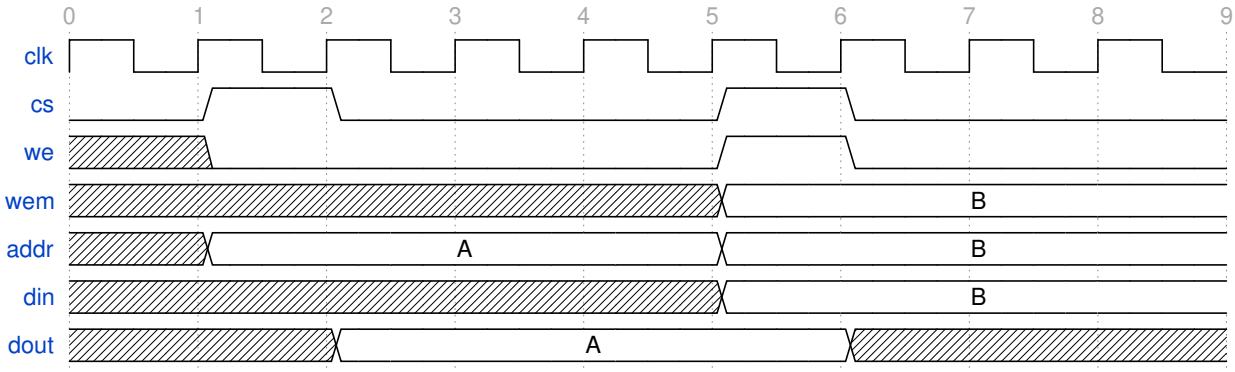


Fig. 17.1: One Cycle SRAM

- @cycle 1: cs is 1 and we is 0, SRAM read request, read address is A
- @cycle 2: SRAM return read data A in dout signal.
- @cycle 3: SRAM dout keep previous read value.
- @cycle 4: SRAM dout keep previous read value.
- @cycle 5: cs is 1 and we is 1, SRAM write request, write address is B, write data is B
- @cycle 6: SRAM return unknown value in dout signal.
- @cycle 7: SRAM dout keep previous value.
- @cycle 8: SRAM dout keep previous value.

17.3.2 Two Cycle SRAM Pipeline

Two Cycle SRAM pipeline timing is as below:

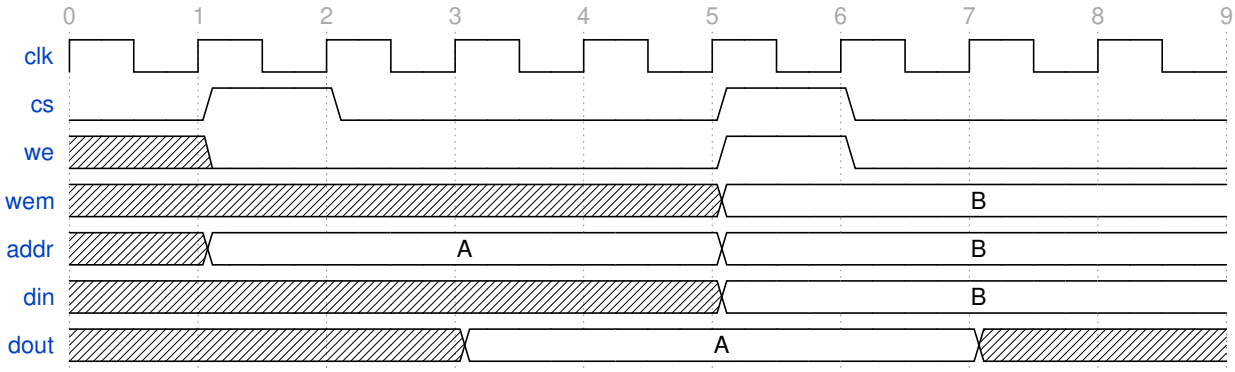


Fig. 17.2: Two Cycle SRAM

- @cycle 1: cs is 1 and we is 0, SRAM read request, read address is A
- @cycle 2: SRAM are processing the read request
- @cycle 3: SRAM return read data A in dout signal.
- @cycle 4: SRAM dout keep previous read value.
- @cycle 5: cs is 1 and we is 1, SRAM write request, write address is B, write data is B
- @cycle 6: SRAM are processing the write request
- @cycle 7: SRAM return unknown data in dout signal.
- @cycle 8: SRAM dout keep previous value.

17.3.3 Three Cycle SRAM Pipeline

Three Cycle SRAM pipeline timing is as below:

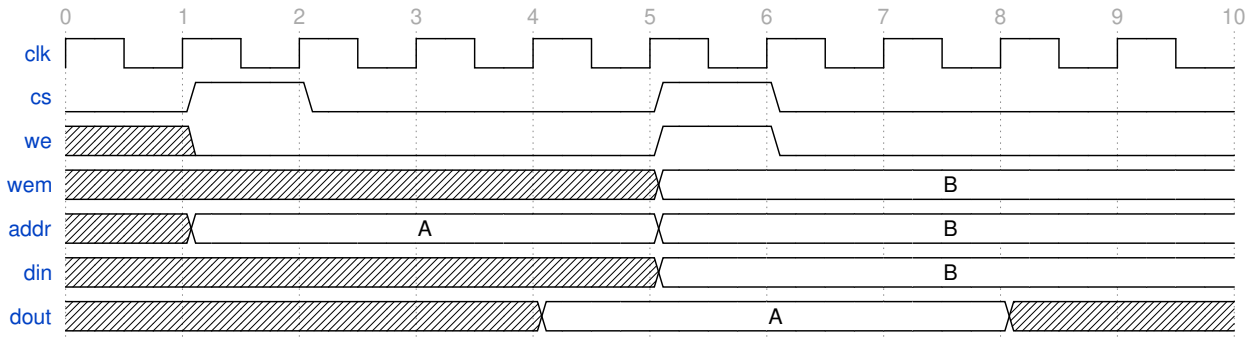


Fig. 17.3: Three Cycle SRAM

- @cycle 1: cs is 1 and we is 0, SRAM read request, read address is A
- @cycle 2: SRAM are processing the read request
- @cycle 3: SRAM are processing the read request
- @cycle 4: SRAM return read data A in dout signal.
- @cycle 5: cs is 1 and we is 1, SRAM write request, write address is B, write data is B
- @cycle 6: SRAM are processing the write request
- @cycle 7: SRAM are processing the write request
- @cycle 8: SRAM return unknown data in dout signal.

17.3.4 Four Cycle SRAM Pipeline

Four Cycle SRAM pipeline timing is as below:

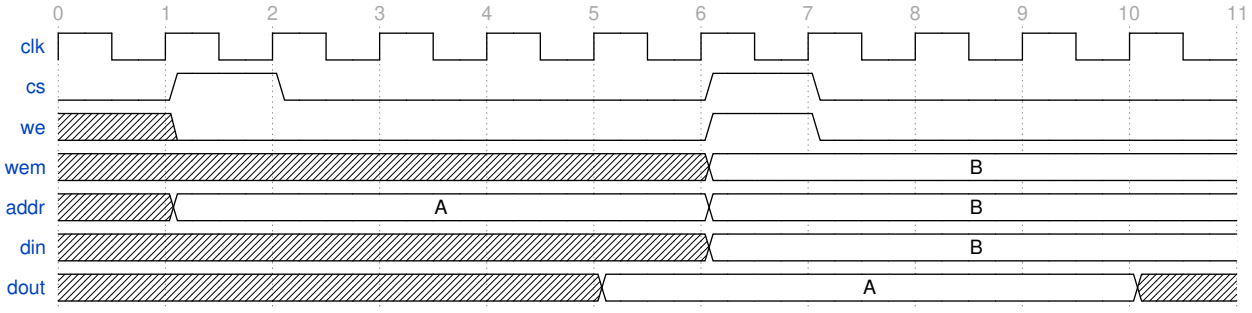


Fig. 17.4: Four Cycle SRAM

- @cycle 1: cs is 1 and we is 0, SRAM read request, read address is A
- @cycle 2: SRAM are processing the read request
- @cycle 3: SRAM are processing the read request
- @cycle 4: SRAM are processing the read request
- @cycle 5: SRAM return read data A in dout signal.
- @cycle 6: cs is 1 and we is 1, SRAM write request, write address is B, write data is B
- @cycle 7: SRAM are processing the write request
- @cycle 8: SRAM are processing the write request
- @cycle 9: SRAM are processing the write request
- @cycle 10: SRAM return unknown data in dout signal.

17.4 Generic SRAM model

Nuclei CPU provide two generic ram model:

- e603_gnrl_ram: Generic ram model without dft signal.
- e603_gnrl_dft_ram: Generic ram model with dft signal.

17.4.1 e603_gnrl_ram

The interface and parameter of e603_gnrl_ram is as below:

```
module e603_gnrl_ram
#(parameter DP = 32,           // Depth
  parameter DW = 32,           // Data Width
  parameter FORCE_X2ZERO = 0, // Read data X force to 0
  parameter GATE_CLK = 1,      // clock gating enable
  parameter MW = 4,            // WRITE MASK Width
  parameter AW = 15            // Address Width
  ) (
  input           sd,          // shutdown mode
  input           ds,          // deep sleep mode
  input           ls,          // light sleep mode

  input           rst_n,       // Reset
  input           clk,         // Clock
  input [DW-1:0] din,          // sram write data
  input [AW-1:0] addr,         // sram write address
  input           cs,          // sram chip select
  input           we,          // sram write enable
  input [MW-1:0] wem,          // sram write mask
  output[DW-1:0] dout          // sram read data
);
```

This module provide the sram simulation model, and the sram is one cycle sram. Please refer to *One Cycle SRAM Pipeline* (page 37)

When SYNTHESIS config, this module will instance e603_gnrl_tech_ram module. Please refer to *Generic Technology SRAM model* (page 40).

17.4.2 e603_gnrl_dft_ram

The interface and parameter of e603_gnrl_dft_ram is as below:

```
module e603_gnrl_dft_ram
#(parameter DP = 32,           // Depth
  parameter DW = 32,           // Data Width
  parameter FORCE_X2ZERO = 0, // Read data X force to 0
  parameter GATE_CLK = 1,      // Clock Gating Enable
  parameter MW = 4,            // WRITE MASK Width
  parameter AW = 15,           // Address Width
  parameter DFT_IN = 1,        // DFT Input Signal Width
  parameter DFT_OUT = 1,       // DFT Output Signal Width
  ) (
  input           sd,          // shutdown mode
  input           ds,          // deep sleep mode
  input           ls,          // light sleep mode
  input [DFT_IN-1:0] dft_in,  // dft input signal

  input           rst_n,       // Reset
  input           clk,         // Clock
  input [DW-1:0] din,          // sram write data
  input [AW-1:0] addr,         // sram write address
  input           cs,          // sram chip select
  input           we,          // sram write enable
  input [MW-1:0] wem,          // sram write mask
  output[DW-1:0] dout,         // sram read data
  output[DFT_OUT-1:0] dft_out // dft output signal
);
```

This module provide the sram simulation model, and the sram is one cycle sram. Please refer to *One Cycle SRAM Pipeline* (page 37).

When SYNTHESIS config, this module will instance e603_gnrl_tech_ram module. Please refer to *Generic Technology SRAM model* (page 40).

17.5 Generic Technology SRAM model

Nuclei CPU provide one generic technology ram model named e603_gnrl_tech_ram.

```
module e603_gnrl_tech_ram
#(parameter DP = 32,           // Depth
  parameter DW = 32,           // Data Width
  parameter FORCE_X2ZERO = 0, // Read data X force to 0
  parameter MW = 4,            // WRITE MASK Width
  parameter AW = 15            // Address Width
  ) (
  input           sd,          // shutdown mode
  input           ds,          // deep sleep mode
  input           ls,          // light sleep mode

  input           rst_n,       // Reset
  input           clk,         // Clock
  input [DW-1:0] din,          // sram write data
  input [AW-1:0] addr,         // sram write address
  input           cs,          // sram chip select
  input           we,          // sram write enable
  input [MW-1:0] wem,          // sram write mask
```

(continues on next page)

(continued from previous page)

```
  output[DW-1:0] dout          // sram read data
);
```

This module provide the sram simulation model, and the sram is one cycle sram. Please refer to *One Cycle SRAM Pipeline* (page 37)

To facilitate customers in replacing the actual SRAMs, we provide instantiation coe for each SRAM configuration used in the CPU. Users only need to replace the corresponding instantiation code.

In the e603_tech_ram module, it include other file depends on the config.

- if TECH_NUCLEI_DUMMY_RAM and TECH_NUCLEI_DUMMY_RAM_SPLIT config, it will include spram_macros_wrapper_splited.v
- if only TECH_NUCLEI_DUMMY_PATH config, it will include spram_macros_wrapper.v

The files spram_macros_wrapper_splited.v and spram_macros_wrapper.v are all auto generated in the synthesis flow. Please refer to Logic Synthesis chapter.

17.5.1 spram_macros_wrapper.v

In the spram_macros_wrapper.v, it instance the origin SRAM configuration.

For example, one 64KB ILM, the code is as below:

```
if (DP == 8192 && DW == 64 && MW == 8)
   begin: u_ram_8192x64_mw8
   wire [63:0]    bwen_pre = ~{
                           {8{wem[7]}},
                           {8{wem[6]}},
                           {8{wem[5]}},
                           {8{wem[4]}},
                           {8{wem[3]}},
                           {8{wem[2]}},
                           {8{wem[1]}},
                           {8{wem[0]}}};
   wire [63:0]    bwen = bwen_pre[63:0];

   NUCLEI_SPRAM_8192X64_MW8    u_ram (
                .A          (addr      ),
                .D          (din       ),
                .Q          (dout      ),
                .CEN        (cen       ),//ACTIVE LOW
                .WEN        (wen       ),//ACTIVE LOW
                .BWEN       (bwen      ),//ACTIVE LOW
                .CLK        (clk       ));
   end
```

So user only need to implement the module NUCLEI_SPRAM_8192X64_MW8 and the ILM SRAM replacement has been completed.

17.6 Single Core SRAM Wrapper

Nuclei CPU will auto generate the e603_core_rams module for single Core.

The e603_core_rams structure is as below:

```
- e603_core_rams:
  - u_core (e603_core_wrapper)
  - u_rams (e603_rams_wrapper)
```

The e603_rams_wrapper includes.

- ILM SRAM
- DLM SRAM
- ICACHE Tag SRAM
- ICACHE Data SRAM
- DCACHE Tag SRAM
- DCACHE Data SRAM
- BPU SRAM
- MMU SRAM

In the e603_rams_wrapper, RTL will instance e603_gnrl_ram for each SRAM.

For example, ILM SRAM:

```
e603_gnrl_ram
 #(
   .FORCE_X2ZERO(0),
   .GATE_CLK(0),
   .DP(8192),
   .DW(64),
   .MW(8),
   .AW(13)
 )u_ilm_ram (
   .sd   (1'b0   ),
   .ds   (1'b0   ),
   .ls   (1'b0   ),
   .clkgate_bypass (clkgate_bypass),

   .cs   (ilm_cs   ),
   .we   (ilm_we   ),
   .wem  (ilm_wem   ),
   .addr (ilm_addr ),
   .din  (ilm_din  ),
   .dout (ilm_dout ),
   .rst_n(rst_n),
   .clk  (clk_ilm  )
   );
```

17.7 SMP Core SRAM Wrapper

Nuclei CPU will auto generate the e603_cluster_top module for SMP Core.

The e603_cluster_top structure is as below:

```
- e603_cluster_top:
   - u_cc_top (e603_cc_rams)
     - u_cc (e603_l2_top)
     - u_rams (e603_cc_rams_wrapper)
   - u_core0 (e603_core_rams)
     - u_core (e603_core_wrapper)
     - u_rams (e603_rams_wrapper)
   - u_core1 (e603_core_rams)
     - u_core (e603_core_wrapper)
     - u_rams (e603_rams_wrapper)
   - u_core(n) (e603_core_rams)
     - u_core (e603_core_wrapper)
     - u_rams (e603_rams_wrapper)
```

The e603_cc_rams_wrapper includes:

- L2Cache Attribute SRAM
- L2Cache Tag SRAM
- L2Cache Data SRAM
- Dcache Shadow tag SRAM.

The e603_rams_wrapper includes.

- ILM SRAM
- DLM SRAM
- ICACHE Tag SRAM
- ICACHE Data SRAM
- DCACHE Tag SRAM
- DCACHE Data SRAM
- BPU SRAM
- MMU SRAM

In the e603_cc_rams_wrapper, RTL will instance e603_gnrl_ram for each SRAM.

For example, l2c_b0_aram SRAM:

```
e603_gnrl_ram
#(
  .FORCE_X2ZERO(0),
  .GATE_CLK(0),
  .DP(512),
  .DW(15),
  .MW(1),
  .AW(9)
)u_l2c_b0_aram (
  .sd   (1'b0   ),
  .ds   (1'b0   ),
  .ls   (1'b0   ),
  .clkgate_bypass (clkgate_bypass),

  .cs   (b0_aram_cs   ),
  .we   (b0_aram_we   ),
```

(continues on next page)

(continued from previous page)

```
  .wem  (b0_aram_we    ),
  .addr (b0_aram_addr ),
  .din  (b0_aram_din  ),
  .dout (b0_aram_dout ),
  .rst_n(rst_n),
  .clk  (clk_b0_aram  )
  );
```

In the e603_rams_wrapper, RTL will instance e603_gnrl_ram for each SRAM.

17.8 Flow for memory integration

The following figure shows the supported SRAM integration for the Nuclei CPU. This integration flow enables you to:

- Identify the SRAM organization required for each SRAM block.
- Integrate your library SRAM.
- Test that SRAM integration is successful.

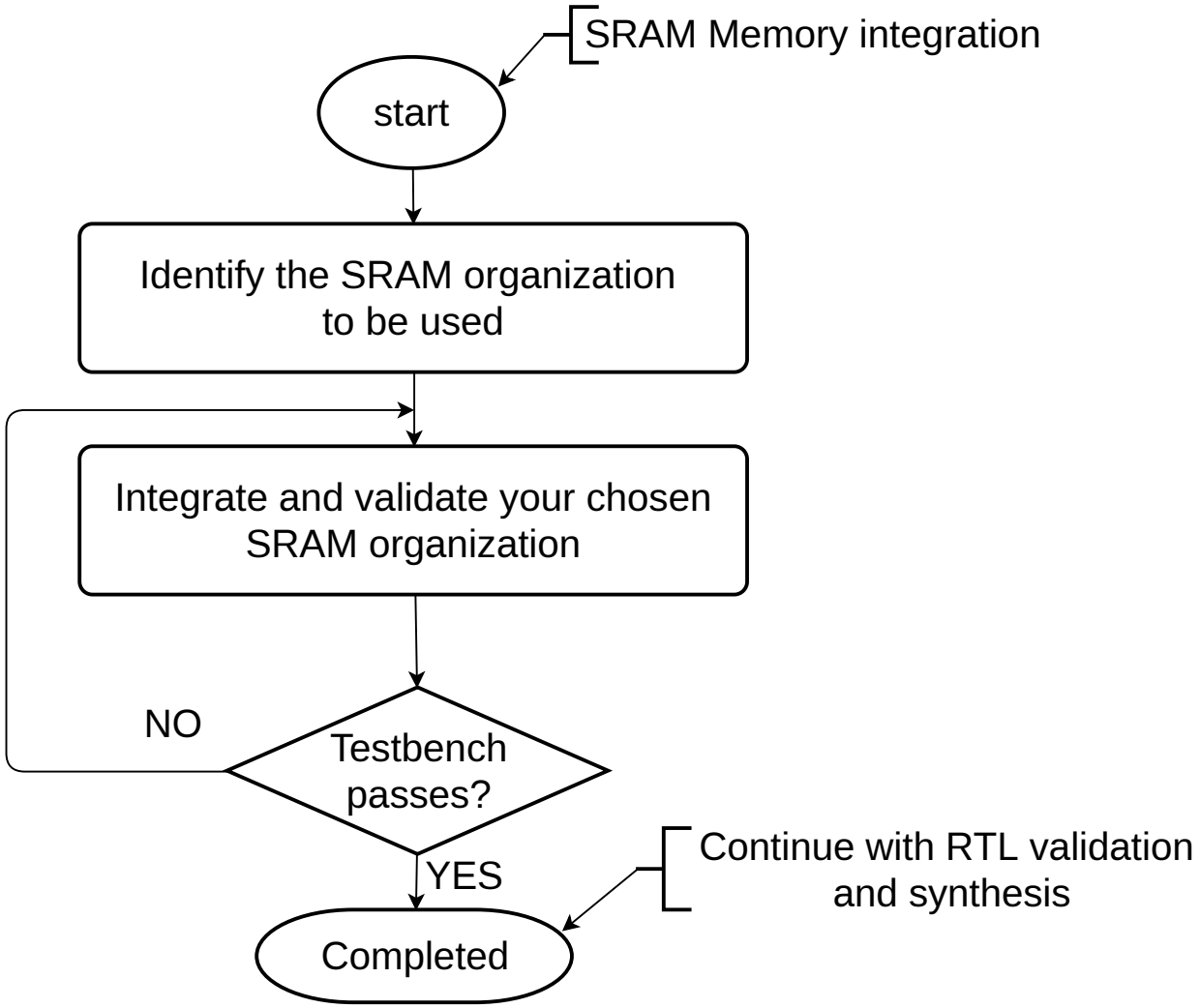


Fig. 17.5: SRAM Integration Flow

SRAM integration is the process of substituting a physical SRAM for the existing generic e603_gnrl_tech_ram model. You must edit the SRAM wrapper module to accommodate your SRAM instance.

Note: We strongly recommend that users replace the SRAM instantiation in spram_macros_wrapper.v or spram_macros_wrapper_split.v.

18 Logic Synthesis

18.1 Logic Synthesis for Verilog RTL

We have provided a set of reference synthesis flow. This flow use one open source technology library.

Note: User can replace the open source technology library to real asic technology library to synthesis.

We provide three synthesis flow:

- Mode1: Core logic synthesis. Apply on single core.
- Mode2: Core logic with sram synthesis. Apply on single core.
- Mode3: Hierarchical synthesis. Apply on smp core
  - Synthesis e603_core_rams module with sram first.
  - Synthesis e603_cluster_top module last.

18.2 Tool required

This flow require three tools:

- Synopsys Design Compiler tool.
  - We recommend using a tool version no lower than 2021.06-SP1.
  - Flow mode1/mode2/mode3 required.
- Synopsys Library Compiler tool.
  - We recommend using a tool version no lower than 2020.09.
  - Flow mode2/mode3 required.
- Synopsys PrimeTime tool.
  - We recommend using a tool version no lower than 2020.09-SP5-1.
  - Flow mode3 required.

18.3 Flow Mode

Run the flow steps:

```
// Step1: load the dc tool into your PATH
export PATH={your_dc_bin_path}:$PATH

// Step2: Enter into e603/syn

cd e603/syn

// Step3: generate the synthesis flow

make gen

// Step4: Enter into generated demo dir
cd demo_logic_only

// Step5: Run synthesis
make dc
```

Note: For more detail information, you can see syn/README.md to get more details.

18.4 Tech Module Replace

Core will use some special std cell module, and these module files are put in design/tech dir.

Take e603 as example:

Table 18.1: Technology module

Tech_file	Tech_mode	Description	ASIC Replace
e603_hand_buf.v	e603_hand_buf	Buffer function. Using to keep signal.	Recommend replace
e603_hand_mux_bit.v	e603_hand_mux_bit	Mux function. Using to jtag clock mux.	Must replace.
e603_gnrl_tech_clkgate.v	e603_gnrl_tech_clkgate	Clocking gating function.	Must replace.
e603_gnrl_tech_sync.v	e603_gnrl_tech_sync	Sync function. Using to sync some input signal. Using to asynchronous reset, synchronous release.	Optional replace.
e603_gnrl_tech_sram.v	e603_gnrl_tech_sram	Sram function.	Must replace.

18.5 Notes for Attentions

The example synthesis project above is just for reference, if the user want to get more precise result, it is suggested with following notes:

- It is strongly recommended to use the "**Flatten**" synthesis mode to flatten the hierarchy during synthesis optimization, to achieve better result of timing and areas.
- The "clock gating module" in the design source files, need to be replaced to the real "clock gating cell" from the ASIC process library. And please keep the comment unchanged during replacing.
  - Take e603 as example, the "clock gating module" is module "e603_gnrl_tech_clkgate" in , which is in file e603/design/tech/e603_gnrl_tech_clkgate.v.

- The “mux module” in the design source files, need to be replaced to the real “mux cell” from the ASIC process library (this module is for clock source switch, user can chose specific cell if the ASIC process library has). And please keep the comment unchanged during replacing.
  - Take e603 as example, the “mux module” is module “e603_hand_mux_bit”, which is in file

    e603/design/tech/e603_hand_mux_bit.v
- The “hand buf module” in the design source files, need to be replaced to the real “hand buf cell” from the ASIC process library. And please keep the comment unchanged during replacing.
  - Take e603 as example, the “hand buf module” is module “e603_hand_buf”, which is in file

    e603/design/tech/e603_hand_buf.v
- The “sync module” in the design source files, can be replaced to the real “sync cell” from the ASIC process library. And please keep the comment unchanged during replacing.
  - Take e603 as example, the “sync module” is module “e603_gnrl_tech_sync”, which is in file

    e603/design/tech/e603_gnrl_tech_sync.v
- The “sram module” in the design source files, need to be replaced to the real “sram” from the ASIC process library. And please keep the comment unchanged during replacing.
  - Take e603 as example, The “sram” is module “e603_gnrl_tech_ram”, which is in file

    e603/design/tech/e603_gnrl_tech_ram.v
  - We provide one sram wrapper file in file

    e603/syn/mem/wrapper/spram_macros_wrapper.v

    User can replace the module name with prefix “NUCLEI_SPRAM_” to real ASIC sram.

18.6 SRAM Partition

User can split a SRAM with a large number of rows into multiple SRAMs if the number of rows cannot be supported, but user are responsible for adding the proper multiplexers to combine the data outputs of the multiple SRAMS.

To easily the SRAM split, We provide one tool to split big sram into some small sram, and this tool generate spram_macros_wrapper_splited.v file.

For example one sram 4096*32, we can split this sram to

- Two 2048*32 sram
- Four 1024*32 sram

Note: Please see syn/README.md to know the usage.

When you have split the sram, you can use evalsoc to verify the correctness of SRAM partitioning.

Below are the steps:

- CD to vsim dir.
- **Modify Makefile to add your SRAM verilog code into testbench.**
  - Change the USER_COMPILE_OPTIONS, add your sram tech define.
  - Change the USER_INCDIRS to the directory of spram_macros_wrapper_splited.v file.
  - Change the USER_RTLDIRS to the directory of sram model file.
- Execute ‘make compile’ to compile rtl and your sram models.
- Execute ‘make verilog’ to launch verdi to see rtl.

19 FPGA Prototyping

19.1 Files in FPGA Project

The files in the FPGA project are introduced as below.

```
e603
  |----fpga                    // Directory for the FPGA project
    |----boards                // Directory for the FPGA boards
      |----ddr200t                          // Directory for DDR200T Evaluation Kit
        |----nuclei-master.xdc  // The main .xdc constraint files
        |----xdc                            // Directory for the .xdc constraint files
        |----script                         // Directory for TCL script
        |----src                            // Directory for Verilog code for fpga top
    |----Makefile       // Makefile
  |----Makefile         // Top Makefile
  |----common.mk        // Common.mk
```

Note: FPGA board (mcu200t, ddr200t, ku060) as the real-time clock (32.768KHz).

- The JTAG Pads of SoC are constrained by nuclei-master.xdc, and map them to the pins of MCU_JTAG connecter on FPGA board (ddr200t).

19.2 Generate Bitstream (Bit format)

In *SoC for Evaluation* (page 12), it introduces the Nuclei Evaluation SoC, the SoC can be generated as FPGA Bitstream, and program into FPGA board (ddr200t), such that, the FPGA board can be worked as a prototype board.

The steps to generate the Bistream for FPGA board are as below (take e603 as example):

```
// Enter into fpga directory
cd e603/fpga

// DDR200T Evaluation Kit (ddr200t):
make BOARD_NAME=ddr200t bit

// The generated MCS format Bitstream will be under
// e603/fpga/gen/$FPGA_NAME/obj/system.bit
```

19.3 Generate Bitstream (MCS format)

The steps to generate the bistream for FPGA board are as below:

```
// Enter into fpga directory
cd e603/fpga

// DDR200T Evaluation Kit (ddr200t):
make BOARD_NAME=ddr200t mcs

// The generated MCS format Bitstream will be under
// e603/fpga/gen/$FPGA_NAME/obj/system.mcs
```

19.4 CPU Clock Setting

The default CPU clock frequency is 16M.

User can set the CORE_FREQ arg when generate bitstream.

For example, change the frequency to 50M

```
// Enter into fpga directory
cd e603/fpga

// DDR200T Evaluation Kit (ddr200t):
make BOARD_NAME=ddr200t CORE_FREQ=50 bit
```

19.5 Program Bitstream into FPGA

About how to program the Bitstream (BIT or MCS format) into the FPGA board, please refer to the document <Nuclei_FPGA_Evaluation_Boards_and_Debugger_Kit_Introduction.pdf> which can be downloaded from "Development Boards" page of Nuclei website (https://www.nucleisys.com/upload/files/fpga/doc/Nuclei_FPGA_DebugKit_Intro.pdf).

20 Appendix

- **Nuclei RISC-V IP Products**: https://www.nucleisys.com/product.php
- **Nuclei Spec Documentation**: https://nucleisys.com/download.php#spec
- **Nuclei RISCV Tools and Documents**: https://nucleisys.com/download.php
- **Nuclei Prebuilt Toolchain and IDE**: https://nucleisys.com/download.php#tools
- **NMSIS**: https://github.com/Nuclei-Software/NMSIS
- **Nuclei SDK**: https://github.com/Nuclei-Software/nuclei-sdk
- **Nuclei Linux SDK**: https://github.com/Nuclei-Software/nuclei-linux-sdk
- **Nuclei Software Organization in Github**: https://github.com/Nuclei-Software/
- **RISC-V MCU Organization in Github**: https://github.com/riscv-mcu/
- **RISC-V MCU Community Website**: https://www.rvmcu.com/
- **Nuclei riscv-openocd**: https://github.com/riscv-mcu/riscv-openocd
- **Nuclei riscv-gnu-toolchain**: https://github.com/riscv-mcu/riscv-gnu-toolchain