芯来科技
NUCLEI

Nuclei™ 600 Series Processor Core Databook

Copyright Notice

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.

Revision History

Rev.	Revision Date	Revised Section	Revised Content
1.0.0	2020/4/1	N/A	1. First version as the full English
1.0.1	2020/5/26	3.9	1. Add MMU TLB SRAM Interface
1.1.0	2020/6/15	1.1, 1.4, 4	1. Add ECC feature on I/D-Cache, I/DLM, TLB
1.1.1	2020/7/9	3.10	1. Add override_dm_sleep to support debug in low-power mode
1.1.2	2020/7/24	3.6	1. Add trace_bjp_taken and trace_dmode in trace interface

Table of Contents

List of Tables

List of Figures

1. Overview

The Nuclei 600 Series Processor Core, or 600 Series Core for short，is a commercial RISC-V Processor Core Series designed by Nuclei System Technology for Real-time or High-performance embedded applications. The 600 Series is a competitive rival to ARM Cortex-R4/R5/R7/M7 Processor Cores, can be used in AIoT edge computing, storage or other real-time scenarios.

1.1. Feature List

The 600 Series have the following features:

- CPU Core
  - N600 series: support RV32I, for the 32-bit micro-controller applications.
  - NX600 series: support RV64I without MMU, for the 64-bit micro-controller applications.
  - UX600 series: support RV64I with MMU, for the 64-bit Linux capable applications. *(MMU can also be turned off by software, then in this mode, UX class core can also be work as micro-controller, UX class core is downward compatible to NX class core.)*
  - 6-pipeline stages, using state-of-the-art processor micro-architecture to deliver the best-of-class performance efficiency and lowest cost.
  - Support dynamic branch predictor.
  - Configurable instruction prefetch logic, which can prefetch subsequent instructions to hide the instruction memory access latency.
- Supported Instruction Set Architecture (ISA)
  - The 600 Series is a configurable RISC-V Processor Core Series, supporting the combination of I/M/A/F/D/C/P instruction extensions.
  - Support misaligned memory access (Load/Store instructions).
- Multiplier and Divider
  - Configurable BOOTH algorithm implemented integer multiplier (1/2/3 stages) for timing optimization.
  - Radix-4 SRT based integer divider with variable cycles implementation for

performance optimization.

- Floating Processing Unit (FPU)
  - Configurable Single-Precision FPU and Double-Precision FPU.
- Digital Signal Processing (DSP)
  - The configurable P extension ISA can support DSP feature, includes SIMD, Partial-SIMD and 64-bit non-SIMD instructions.
- Supported Privileged Modes
  - Support Machine Mode, Supervisor Mode (configurable) and User Mode (configurable).
- Bus Interfaces
  - Support 64-bit wide standard AXI system bus interface with configurable Clock-Ratio feature for accessing system instruction and data.
  - Support 64-bit wide Instruction Local Memory (ILM) bus interface (SRAM interface protocol) for accessing private instruction local memory.
  - Support two 32-bit wide Data Local Memory (DLM) bus interfaces (SRAM interface protocol) for accessing private data local memory: DLM0 and DLM1, only the double-word type will access the DLM0 and DLM1 simultaneously, other types (byte/half-word/word) will access either DLM0 or DLM1 by even/odd banks.
  - Support 32-bit wide Private Peripheral Interface (PPI) bus interface (with standard AHB-Lite interface protocol) for fast accessing private peripherals.
  - Support 64-bit wide AXI slave port, for external device (DMA or other masters) to access internal ILM/DLM0/DLM1.
- Low-Power Management
  - Support WFI (Wait For Interrupt) and WFE (Wait For Event) scheme to enter sleep mode.
  - Support two-level sleep modes: shallow sleep mode, and deep sleep mode.
- Core-Private Timer Unit (TIMER)
  - 64-bits wide real-time counter.

- Support the generation of the timer interrupt defined by the RISC-V standard.
- Support the generation of the precise periodic timer interrupt (can be used as System Tick) with auto clear-to-zero mode.
- Support the generation of software interrupt defined by the RISC-V standard.

■ Enhanced Core Level Interrupt Controller (ECLIC)

- Support the RISC-V architecturally defined software, timer and external interrupts.
- Support configurable number of external interrupt sources.
- Support configurable number of interrupt levels and priorities, and support software dynamically programmable division of interrupt levels and priorities values.
- Support interrupts preemption based on interrupt levels.
- Support vectored interrupt processing mode for extremely fast interrupt response (6 cycles).
- Support fast interrupts tail-chaining mechanism.

■ Support NMI (Non-Maskable Interrupt)

■ Memory Protection

- Support configurable Physical Memory Protection (PMP) to protect the memory.

■ Memory Management Unit (MMU)

- UX class supports configurable MMU unit to manage the Virtual memory system.
- 2-level TLB: Main-TLB and Micro-TLB (i-tlb, d-tlb)
- Configurable ECC feature on Main-TLB

■ Security with Trust Execution Environment

- Support configurable Trust Execution Environment (TEE) feature.

■ Support Instructions Extended by User

- Support Nuclei Instruction Co-unit Extension (NICE) scheme to support user to extend custom instructions according to their applications.

■ Support Instruction Cache (I-Cache)

- The Cache size is configurable.
- N-way associative structure with each way 4KB size.
- Cache Line Size is 32 Bytes.
- Support to LOCK, INVAL cachelines.
- Configurable ECC feature on I-Cache.

■ Support Data Cache (D-Cache)

- The Cache size is configurable.
- 2-way associative structure.
- Cache Line Size is 32 Bytes.
- Support to LOCK, FLUSH and/or INVAL cachelines.
- Configurable ECC feature on D-Cache.

■ Support ECC feature

- ECC on I-Cache, D-Cache, ILM, DLM, TLB.
- ECC mechanism is SECDED *(Single Error Correction, Double Errors Detection)*.
- Please refer to 'Nuclei ISA Spec' for more details of the ECC feature.

■ Debugging System

- Support standard 4-wire JTAG interface.
- Support standard RISC-V debug specification (v0.13).
- Support configurable number of Hardware Breakpoints.
- Support mature interactive debugging software/hardware tools, such as GDB, OpenOCD, Lauterbach TRACE32, Segger J-Link, IAR, etc.

■ Software Development Tools

- The 600 Series support the RISC-V standard compilation toolchain and Integrated Development Environment (IDE) on Linux/Windows systems.

1.2. Supported Instruction Set and Privileged Architecture

The 600 Series is following the Nuclei RISC-V Instruction Set and Privileged Architecture Specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from "Nuclei User

Center" website http://user.nucleisys.com.

1.3. Top Diagram

The top diagram of 600 Series is as depicted in Figure 1-1, the key points of which are:

- Core Wrapper is the top module of the Core, the sub-modules are:
  - Core: The core part. (two same cores will be instantiated if Lock-Step feature configured)
  - Reset Sync: The module to sync external async reset signal to synced reset with "Asynchronously assert and synchronously de-assert" style.
  - DEBUG: The module to handle the debug functionalities interactive with JTAG.
- uCore is under Core hierarchy, it is the main part of Core.
- Besides the uCore, there are several other sub-modules:
  - LM Ctrl: The control module for ILM and DLM.
  - ECLIC: The private interrupt controller.
  - TIMER: The private timer unit.
  - BIU: The bus interface unit.
  - Misc Ctrl: Other miscellaneous modules.

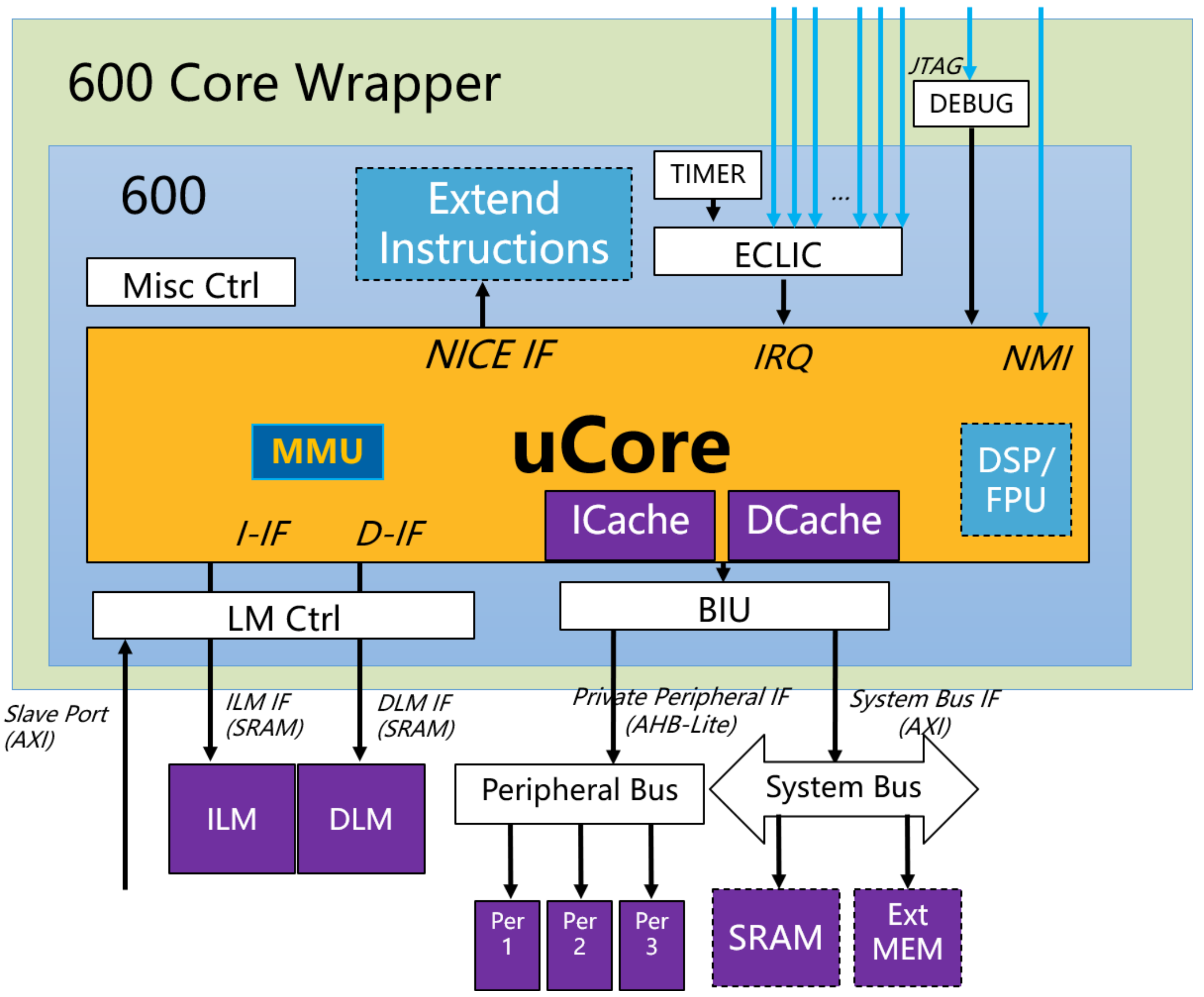

Figure 1-1 The top diagram of 600 Series Core

1.4. Different Cores of 600 Series

The 600 is a series of Cores with different configuration templates. Different Cores have different configurable features, as briefly summarized in Table 1-1.

Table 1-1 Configurable features of different Cores

	605	607	608
Baseline Instruction Set	IMAC/	IMAC/ IMAFC/ IMAFDC/ IMAFDPC	IMAC/ IMAFC/ IMAFDC/ IMAFDPC
Hardware Multiplier/Divider	YES	YES	YES
A (Atomic) Instruction Extension	YES	YES	YES
Unaligned Load/Store Hardware Support	YES	YES	YES
Low Power Features	YES	YES	YES
Interrupts Number	Configurable	Configurable	Configurable
NMI	YES	YES	YES
ILM and DLM Interface	Configurable	Configurable	Configurable
PPI Interface	Configurable	Configurable	Configurable
Instruction Cache (I-Cache)	Configurable	Configurable	Configurable
Data Cache (D-Cache)	Configurable	Configurable	Configurable
User Mode and PMP	Configurable	Configurable	Configurable
ECC	Configurable	Configurable	Configurable
Debugging System	Configurable	Configurable	Configurable
User Instruction Extendibility (NICE)	Configurable	Configurable	Configurable
Timing Enhancing Options	Configurable	Configurable	Configurable
Performance Enhancing Options	Configurable	Configurable	Configurable
Single-Precision Floating point Unit	NO	Configurable	Configurable
Double-Precision Floating point Unit	NO	Configurable	Configurable
Packed-SIMD DSP	NO	Configurable	Configurable
TEE Support	NO	NO	Configurable
MMU Support	UX605/607/608 ONLY		

2. Functional Introductions

2.1. Clock Domains

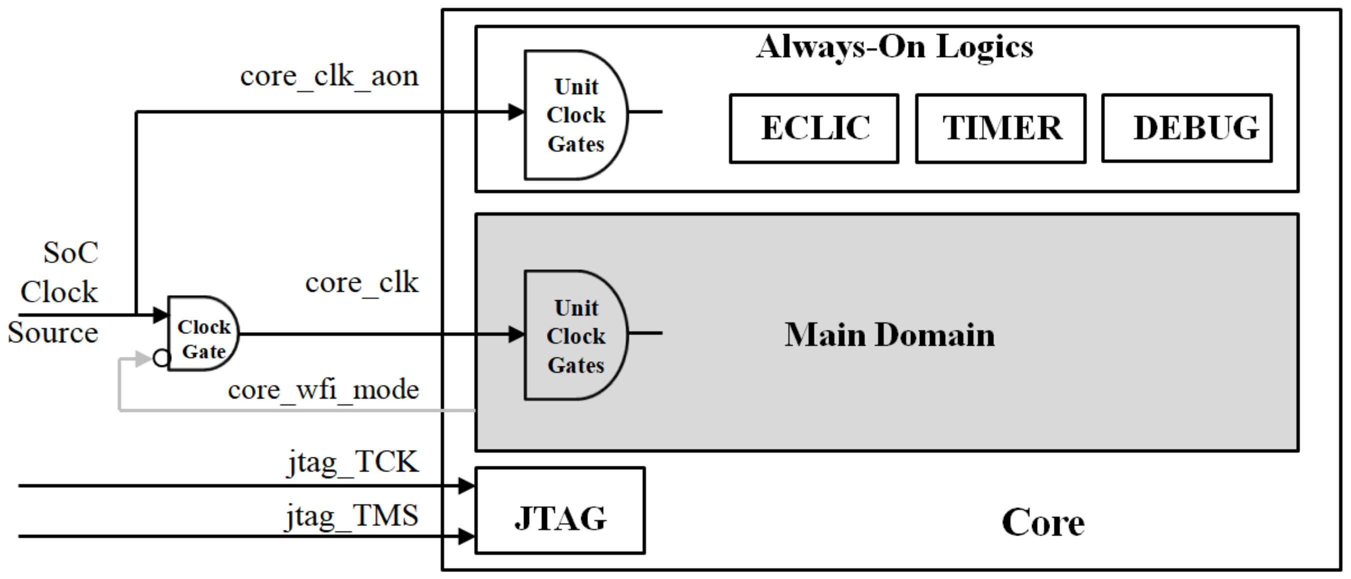

Figure 2-1 Clock domains of the N300 Series Core

The clock domains of the 600 Series Core are shown in Figure 2-1. The entire Core is divided into two asynchronous clock domains:

- The main clock domain (for the core_clk and core_clk_aon), which drive most of the functional logics of the Core. Note:
  - core_clk and core_clk_aon have the same frequency and same phases as they are supposed to be clocks from the same clock source.
  - core_clk is the main working clock that drives the main domain inside the Core, core_clk is supposed to be clock-gated at the SoC level during the sleep mode.
  - core_clk_aon is an always-on clock that drives the always-on logic in the Core, including the ECLIC, TIMER, DEBUG, etc.
- The JTAG_CLK clock domain (for the jtag_TCK), which drives the JTAG logics of DEBUG unit.

The above two clock domains are asynchronous with each other, so asynchronous cross-clock domain processing has been implemented in the Core.

2.2. Power Domains

There is no power domains specified inside 600 Series Core. Per different applications, the SoC integrator can choose to divide the power domains according to the convenience of the hierarchies inside the Core.

2.3. Core Interfaces

- Clock and Reset
- JTAG
- Interrupt
- Bus
- NICE
- Misc.

Please refer to Chapter 3 for the details of Core interfaces.

2.4. Memory Resources

600 Series Core supports the following memory resources:

- ILM:
  - The Core supports Instruction Local Memory (ILM) access via an independent SRAM interface if the ILM interface is configured.
  - The address space of the ILM is configurable. Please see Section 2.6 for details.
  - The ILM is implemented by the SoC integrator and can generally be an SRAM or Flash for storing instructions.
  - ECC feature can be configured on ILM.
  - An extend CSR is defined to Enable/Disable the ILM address space, please refer to ‘Nuclei ISA Spec’ for more details.

- DLM:

  - The Core supports Data Local Memory (DLM) access via an independent SRAM interface if the DLM interface is configured.
  - The address space of the DLM is configurable. Please see Section 2.6 for details.
  - The DLM is implemented by the SoC integrator and can generally be an SRAM for storing data.
  - ECC feature can be configured on DLM.
  - An extend CSR is defined to Enable/Disable the DLM address space, please refer to 'Nuclei ISA Spec' for more details.

- I-Cache:

  - The Core supports I-Cache (Instruction Cache) to cache the instruction fetched from MEM interface. Note: the instruction fetched from ILM interface will not be cached, i.e., if the instruction fetch address fall into ILM address space, it will directly access ILM interface with the I-Cache bypassed.
  - I-Cache is N-ways associative structure, with Way size as 4KB and Line Size as 32 Bytes. The size of I-Cache is configurable. Please refer to Chapter 4 for more details of configurations.
  - ECC feature can be configured on I-Cache (Tag and Data Ram).
  - Some extended CSR registers are defined for I-Cache Control and Maintenance (CCM) operations, which can be used to INVAL, LOCK and UNLOCK I-Cache by ADDR or by ALL. Please refer to the "Nuclei_CCM_Mechanism" for more details, user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

- D-Cache:

  - The Core supports D-Cache (Data Cache) to cache the data fetched from MEM interface with cacheable attribute address space. Note: the data fetched from ILM and DLM interface will not be cached.
  - D-Cache uses Write-Back and Write-Allocate mechanism, which is 2-ways associative structure, with Line Size as 32 Bytes. The size of D-Cache is configurable. Please refer to Chapter 4 for more details of configurations.

- ECC feature can be configured on D-Cache (Tag and Data Ram).
- Some extended CSR registers are defined for D-Cache Control and Maintenance (CCM) operations, which can be used to FLUSH, INVAL, LOCK and UNLOCK D-Cache by ADDR or by ALL. Please refer to "Nuclei_CCM_Mechanism" for more details, user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

2.5. Private Peripherals

As shown in Figure 1-1, under the Core hierarchy, in addition to the uCore, the following private peripherals are included:

- DEBUG: handle the JTAG interface and related debugging features.
- ECLIC: the Enhanced Core-Level Interrupt Controller.
- TIMER: the private TIMER unit.

The above peripherals are private to the Core and are accessed using memory mapped address. See Section 2.6 for the details of their specific address space allocation.

2.6. Address Spaces of Interfaces and Private Peripherals

There are quite several interfaces and private peripherals for the Core, the address spaces of them are shown in Table 2-1.

Table 2-1 Address Spaces of the Core

Unit	Base Address	Offset	Description
DEBUG	Configurable	0x000~ 0xFFF	■ Address space of DEBUG unit. ■ Note: DEBUG is private inside the core. And DEBUG is used for debugging functionality. The regular application software should not access this space.
ECLIC	Configurable	0x0000 ~ 0xFFFF	■ Address space of ECLIC unit. ■ Note: ECLIC is private inside the core.
TIMER	Configurable	0x00 ~ 0xFF	■ Address space of TIMER unit. ■ Note: TIMER is private inside the core.

ILM	Configurable	Depends on the configuration of ILM address space	■ Address space of ILM interface.
DLM	Configurable	Depends on the configuration of DLM address space	■ Address space of DLM interface.
PPI	Configurable	Depends on the configuration of PPI address space	■ Address space of PPI interface.
MEM	N/A	N/A	■ The address space other than the above mentioned spaces are all to MEM (System Memory) interface. ■ MEM address space can be: 1. Device, configurable; 2. Non-Cacheable, configurable; 3. Cacheable
Note: please refer to Chapter 4 for more details of configurations.			

There are several key points:

- The Core's instruction fetches will not go to DLM, ECLIC, TIMER, FIO, or PPI anyway.
- Since the Core have instruction fetch and data access paths independent inside the Core, the address space of ILM and DLM can be overlapped.
  - If the Core has not been configured with "600_CFG _LSU_ACCESS_ILM", then the data access cannot access ILM interface anyway.
  - If the Core has been configured with "600_CFG _LSU_ACCESS_ILM", then the data access can access ILM interface if the data access address fall into the ILM address space. Note: if the data access address fall into both ILM and DLM address spaces (since address spaces of ILM and DLM could be overlapped), then the data access still go to ILM interface.
- The total address space of "ILM and DLM" should not overlap with the total address space of "DEBUG, TIMER, ECLIC, FIO and PPI", otherwise it is configuration error.
- The address spaces of DEBUG, TIMER, ECLIC, FIO, and PPI should not overlap; otherwise it is the configuration error.

2.7. Debug Support

600 Series Core supports standard 4-wire JTAG interface, standard RISC-V debug specification (v0.13), configurable number of Hardware Breakpoints, and mature interactive debugging

software/hardware tools, such as GDB, OpenOCD, Lauterbach TRACE32, Segger J-Link, IAR, etc.

600 Series Core defines an input signal i_dbg_stop, which can be controlled by the SoC level to disable the debug functionality or not:

- If the value of the i_dbg_stop signal is 0, the debug functionality of the Core is working properly.
- If the value of the i_dbg_stop signal is 1, the debug functionality of the Core is off, and the external Debug Hardware Probe cannot debug the Core through JTAG interface anymore.

2.8. Interrupt and Exception Mechanism

For a detailed description of the Core's interrupt and exception mechanisms, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.

2.9. NMI Mechanism

NMI (Non-Maskable Interrupt) is a special input signal of the Core, often used to indicate system-level emergency errors (such as external hardware failures, etc.). After encountering the NMI, the Core should abort execution of the current program immediately and process the NMI handler instead. For a detailed description of the NMI mechanism of the 600 Series Core, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.

2.10. Control and Status Registers (CSRs)

Some control and status registers (CSRs) are defined in the RISC-V architecture to configure or record the status of execution. CSR registers are registers internal to the Core that uses their private 12-bit encoding space to access.

Some extended CSR registers are defined for I/D-Cache Control and Maintenance (CCM) operations, which can be used to FLUSH, INVAL, LOCK and UNLOCK I/D-Cache by ADDR or by ALL.

For a detailed description of the Core's CSRs, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.

2.11. Performance Monitor

The RISC-V architecture defines the following two performance counters:

- Cycle Counter:
  - A 64-bit wide clock cycle counter that reflects how many clock cycles the Core has executed. This counter will continuously increment as long as the Core’s core_clk_aon is ON.
  - The CSR mcycle reflect the lower 32 bits of the counter, and the CSR mcycleh reflect the upper 32 bits of the counter. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for the details; user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.
- Instruction Retirement Counter:
  - The RISC-V architecture defines a 64-bit wide instruction completion counter that reflects how many instructions the Core successfully executed. This counter will increment if the processor executes an instruction successfully.
  - The CSR minstret reflect the lower 32 bits of the counter, and the CSR minstreth reflect the upper 32 bits of the counter. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.

The Cycle Counter and the Instruction Retirement Counter are typically used to measure performance.

By default, the counter is zero value after a reset and then increments itself continuously. But in Nuclei 600 Series Core, considering the counter increases the power consumption, there is an extra bit in the customized CSR mcountinhibit that can pause the counter to save power when users don't need to monitor the performance through the counter.

Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

2.12. TIMER Unit

The RISC-V architecture defines a 64-bit Timer Counter which is clocked by the system's low-speed Real Time Clock frequency. The value of this timer is reflected in the register mtime. The RISC-V architecture also defines a 64-bit mtimecmp register that used as a comparison value for the timer. A timer interrupt is generated if the value of mtime is greater than or equal to the value of mtimecmp.

Note: The RISC-V architecture does not define the mtime and mtimecmp registers as CSR registers, but rather as Memory Address Mapped system registers. The specific memory mapped address is not defined by the RISC-V architecture, so it is defined by the implementation. In the 600 Series Core, mtime and mtimecmp are both implemented in the TIMER Unit.

Besides, the TIMER Unit of 600 Series Core can also generate the periodic timer interrupt (normally as System Tick) and the software interrupt.

By default, the counter is zero value after a reset and then increments itself continuously. But in Nuclei 600 Series Core, considering the counter increases the power consumption, there is an extra bit in the customized CSR mcountinhibit that can pause the counter to save power when users don't need to monitor the performance through the counter.

In debug mode, timer will stop counting when executing codes inserted through debugger to reflect the real behaviors of the program under debugging.

Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

2.13. Low-Power Mechanism

The low-power mechanism of the 600 Series Core is as below:

- The clocks of the main units in the Core are automatically gated off when they are in idle state to reduce static power consumption.
- The Core supports different sleep modes (shallow sleep mode or deep sleep mode) through WFI (Wait for Interrupt) and WFE (Wait for Event) mechanisms to achieve lower dynamic and static power consumption. For more details about "Wait for Interrupt" and "Wait for Event" mechanism, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

2.13.1. Clock Control of Entering Sleep Modes

The key points of the clock control (reference scheme) of the core entering sleep mode are as the followings:

- As shown in Figure 2-1, when the WFI/WFE is successfully executed, the output signal core_wfi_mode of the Core is asserted, indicating that the Core has entered to the sleep mode; at the SoC level, the signal core_wfi_mode should be used to control the external gate logic to disable the core_clk.
- If the Core entered the deep sleep mode (core_sleep_value is 1), then SoC can decide whether to disable the always on clock core_clk_aon according to its actual scenario.

2.13.2. Clock Control of Exiting Sleep Mode

The Core can be waked up by interrupt, event, debug or NMI. Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details; user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

The key points of the clock control of the Core exiting the sleep mode are as the followings:

- The output signal core_wfi_mode will be de-asserted immediately after the core being waked up. Assuming the SoC control the gate of core_clk using the signal core_wfi_mode, the working clock of core, core_clk will be enabled as soon as the signal core_wfi_mode is de-asserted.

- For the case that the core is waiting for an interrupt to wake up, because the Core can only handle the interrupt processed and distributed by ECLIC unit, then only the interrupt, which is enabled and has greater interrupt level than the interrupt threshold level, can wake up the core. Furthermore, whether enable the core_clk_aon inside the core needs to be handled carefully:

  - As mentioned in Section 2.1, the TIMER unit is clocked by core_clk_aon, so if the SoC system has disabled the always-on clock core_clk_aon, the TIMER unit cannot generate timer or software interrupt because it has no working clock, and the core cannot be woken up.

  - As mentioned in Section 2.1, the ECLIC unit is clocked by core_clk_aon, so if the SoC system has disabled the always-on clock core_clk_aon, then the external interrupt signal must kept asserted until the SoC enable the core_clk_aon again. Otherwise, the ECLIC unit cannot sample the external interrupt signal because it has no working clock, and the core cannot be woken up.

- For the case that the core is waiting for an event or NMI to wake up, if the core sampled (by the core_clk_aon) the input signal rx_evt (indicate there is one event) or the input signal nmi (indicate there is one NMI), the core will be woken up. Furthermore, whether enable the core_clk_aon inside the core needs to be handled carefully:

  - If the SoC system has disabled the always-on clock core_clk_aon, then the input signal rx_evt or nmi must keep as 1 until the SoC turns on the clock core_clk_aon. Otherwise, the core cannot sample the rx_evt or nmi as the sample logic has no working clock, and the core will not wake up.

2.14. Physical Memory Protection

In order to perform memory access protection and isolation according to memory physical address and execution privilege mode, the RISC-V standard architecture defines a physical memory protection mechanism: Physical Memory Protection (PMP).

600 Series Core can be configured to support the PMP feature. About the programming mode of PMP, please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.

2.15. Memory Management Unit (MMU)

UX class core can have MMU configured for Linux capable applications, which implements the SV39 mode defined in RISC-V privileged specification, to support Page-Based 39-bit Virtual-Memory System, which can be used for converting Virtual-Address to Physical-Address and corresponding Permission Checking.

MMU has two level TLB (Translation Lookaside Buffer) implemented to cache the page tables for fast subsequent accessing: Main-TLB and Micro-TLB (i-tlb, d-tlb); Main-TLB is the shared TLB for both instruction and data page table, Main-TLB can be configured as 32, 64 or 128 entries; Micro-TLB (i/d-tlb) is dedicate for instruction/data page table, each has 8 entries; Micro-TLB (i/d-tlb) will be accessed first, if miss then Main-TLB will be accessed.

MMU supports 4KB, 2MB and 1GB page types, which uses Hardware Translation Table Walk mechanism to fetch page tables from memory when Main-TLB miss without software handling.

ECC feature can be configured on Main-TLB.

Please refer to Nuclei ISA specification (Nuclei_RISCV_ISA_Spec) for more details, user can easily get the specification from “Nuclei User Center” website http://user.nucleisys.com.

2.16. FPU Feature

600 Series Core can be configured to support single-precision (F extension) or double-precision (D extension) floating point instructions. For the details of F/D extension, please refer to Nuclei RISC-V Instruction Set and Privileged Architecture Specification (Nuclei_RISCV_ISA_Spec), user can easily get the specification from "Nuclei User Center" website http://user.nucleisys.com.

2.17. DSP Feature

600 Series Core can be configured to support Packed-SIMD DSP instructions, please refer to another document <Nuclei_DSP_QuickStart> for the details of DSP instructions. User can easily get the copy from "Nuclei User Center" website http://user.nucleisys.com.

2.18. NICE Feature

600 Series Core can be configured to support user to add their custom instructions with NICE (Nuclei Instruction Co-unit Extension) interface. Please refer to another document <Nuclei_NICE_Extension> for the details. User can easily get the copy from "Nuclei User Center" website http://user.nucleisys.com.

2.19. TEE Feature

600 Series Core can be configured to support the TEE (Trust Execution Environment) feature. About the details of TEE, please refer to another document <Nuclei_TEE_Architecture>. User can easily get the copy from "Nuclei User Center" website http://user.nucleisys.com.

3. Core Interfaces

3.1. Clock and Reset Interface

The clock and reset signals of 600 Series Core are as depicted Table 3-1.

Table 3-1 Clock and Reset Signals

Signal Name	Direction	Width	Description
core_clk_aon	Input	1	■ This clock is to drive the Always-On Logics of the Core, please refer to Section 2.1 for more details.
core_clk	Input	1	■ This clock is to drive the Main Logics of the Core, please refer to Section 2.1 for more details.
por_reset_n	Input	1	■ Power on Reset. This signal is active low signal. This reset will reset the entire N300 Core including the JTAG logics. ■ Note: this reset signal will be synced inside N300 Core to make it as "asynchronously asserted and synchronously de-asserted" style.
core_reset_n	Input	1	■ System Reset. This signal is active low signal. This reset will reset the N300 Core except the JTAG logics. ■ Note: this reset signal will be synced inside N300 Core to make it as "asynchronously asserted and synchronously de-asserted" style.
reset_bypass	Input	1	■ If the reset_bypass signal is high, then the internal generated reset will be bypassed, to allow DFT (Design For Test) rules. ■ Note: if the reset_bypass is high, the core_reset_n will be bypassed and disabled, only the por_reset_n will really take effects.
clkgate_bypass	Input	1	■ If the clkgate_bypass is high, the clock gater will be bypassed, to allow DFT (Design For Test) rules.

3.2. JTAG Debug Interface

600 Series Core can be configured to support the standard 4-wire JTAG interface, it is compliant to IEEE 1149.7 T4 Wide protocol.

The JTAG signals of 600 Series Core are as depicted Table 3-2.

Table 3-2 JTAG Signals

Signal Name	Direction	Width	Description

jtag_TCK	Input	1	■ JTAG TCK signal. ■ Note: this signal needs to be constrained as Clock (asynchronous to the Core's main clock).
jtag_TMS	Input	1	■ 4-wire JTAG TMS signal
jtag_TDI	Input	1	■ 4-wire JTAG TDI signal.
jtag_TDO	Output	1	■ 4-wire JTAG TDO signal.
jtag_DRV_TDO	Output	1	■ 4-wire JTAG TDO output enable signal to IO PAD. When the TDO is outputting, this DRV_TDO signal will be high to enable IO PAD as output.

3.3. Interrupt Interface

The interrupt signals of 600 Series Core are as depicted Table 3-3

Table 3-3 Interrupt and NMI signals

Signal Name	Direction	Width	Description
nmi	Input	1	■ NMI (Non-Maskable Interrupt) input. ■ Note: ● nmi_irq signal will not be synced inside the Core, so this signal need to be synced at the SoC level if it is from different clock domain at SoC. ● Please refer to Section 2.9 for the details of NMI.
irq_i	Input	Configurable	■ External Interrupt input, each bit of which can be used to connect an interrupt source. ■ Note: ● irq_i signal will not be synced inside the Core, so these signals need to be synced at the SoC level if it is from different clock domain at SoC. ● Please refer to Section 2.9 for the details of interrupt.

3.4. Bus Interfaces

600 Series Core support several bus interfaces, including:

- ILM Master Interface.
- DLM0/1 Master Interface.
- PPI Interface.

- MEM (System Memory) Interface.
- SLAVE Interface.

3.4.1. ILM and DLM Interface

ILM interface is used to access Instruction Local Memory, and DLM interface is used to access Data Local Memory. Both the ILM and DLM interface can be configured as SRAM protocol interface.

3.4.1.1 ILM Master Interface

ILM interface can be configured as SRAM protocol, the signal list is as shown in Table 3-4.

Table 3-4 ILM SRAM signals

Signal Name	Direction	Width	Description
ilm_cs	output	1	■ SRAM's CS signal.
ilm_addr	output	Depends on configuration	■ SRAM's ADDR signal.
ilm_byte_we	output	8	■ SRAM's WEM signal.
ilm_wdata	output	64	■ SRAM's RAM_IN signal.
ilm_rdata	input	64	■ SRAM's RAM_OUT signal.
clk_ilm_ram	output	1	■ SRAM's CLK signal

3.4.1.2 DLM Master Interface

DLM interface can be configured as SRAM protocol, the signal list is as shown in Table 3-5.

Table 3-5 DLM SRAM signals

Signal Name	Direction	Width	Description
dlmo_cs	output	1	■ SRAM's CS signal.
dlmo_addr	output	Depends on	■ SRAM's ADDR signal.

		configuration	
dlm0_byte_we	output	4	■ SRAM's WEM signal.
dlm0_wdata	output	32	■ SRAM's RAM_IN signal.
dlm0_rdata	input	32	■ SRAM's RAM_OUT signal.
clk_dlm0_ram	output	1	■ SRAM's CLK signal
dlm1_cs	output	1	■ SRAM's CS signal.
dlm1_addr	output	Depends on configuration	■ SRAM's ADDR signal.
dlm1_byte_we	output	4	■ SRAM's WEM signal.
dlm1_wdata	output	32	■ SRAM's RAM_IN signal.
dlm1_rdata	input	32	■ SRAM's RAM_OUT signal.
clk_dlm1_ram	output	1	■ SRAM's CLK signal

3.4.2. MEM Interface

MEM interface is used to access system memory for instruction and data. MEM interface is AXI protocol interface as shown in Table 3-6.

Table 3-6 MEM signals

Signal name	Dir	Bit width	Description
mem_clk_en	Input	1	■ MEM interface clock ratio
mem_arvalid	Output	1	■ AXI protocol ARVALID signal
mem_araddr	Output	64	■ AXI protocol ARADDR signal
mem_arlen	Output	8	■ AXI protocol ARLEN signal (0-255)
mem_arsize	Output	3	■ AXI protocol ARLEN signal (0-7)
mem_arburst	Output	2	■ AXI protocol ARBURST signal (0-2)
mem_arlock	Output	2	■ AXI protocol ARLOCK signal
mem_arcache	Output	4	■ AXI protocol ARCACHE signal
mem_arprot	Output	3	■ AXI protocol ARPROT signal
mem_arready	Input	1	■ AXI protocol ARREADY signal
mem_awvalid	Output	1	■ AXI protocol AWVALID signal
mem_awaddr	Output	64	■ AXI protocol AWADDR signal
mem_awlen	Output	8	■ AXI protocol AWLEN signal (0-255)
mem_awsize	Output	3	■ AXI protocol AWSIZE signal (0-7)
mem_awburst	Output	2	■ AXI protocol AWBURST signal (0-2)

mem_awlock	Output	2	■ AXI protocol AWLOCK signal
mem_awcache	Output	4	■ AXI protocol AWCACHE signal
mem_awprot	Output	3	■ AXI protocol AWPROT signal
mem_awready	Input	1	■ AXI protocol AWREADY signal
mem_wvalid	Output	1	■ AXI protocol WVALID signal
mem_wdata	Output	64	■ AXI protocol WDATA signal
mem_wlast	Output	1	■ AXI protocol WLAST signal
mem_wstrb	Output	8	■ AXI protocol WSTRB signal
mem_wready	Input	1	■ AXI protocol WREADY signal
mem_rvalid	Output	1	■ AXI protocol RVALID signal
mem_rdata	Output	64	■ AXI protocol RDATA signal
mem_rlast	Output	1	■ AXI protocol RLAST signal
mem_rresp	Output	2	■ AXI protocol RRESP signal
mem_rready	Input	1	■ AXI protocol RREADY signal
mem_bvalid	Output	1	■ AXI protocol BVALID signal
mem_bresp	Output	2	■ AXI protocol BRESP signal
mem_bready	Input	1	■ AXI protocol BREADY signal

Note:

- For AR/AW channel:
  - I-Cache Miss and D-Cache Writeback/Eviction will trigger INCR burst type;
  - D-Cache Read/Write Miss will trigger WRAP burst type;
  - if BURST type is WRAP, then LEN must be 3 and SIZE must be 3;
  - if BURST type is FIXED or INCR, then LEN must be 0 and SIZE can be 0/1/2;
  - maximum read outstanding capacity is 8, can be:
    - 4 cacheable instruction fetching, or
    - 8 cacheable load, or
    - 8 noncacheable load, or
    - 8 device read, or
    - Mixed above
  - maximum write outstanding capacity is 8, can be:

➢ 8 cacheable write/eviction, or

➢ 8 noncacheable write, or

➢ 8 device write, or

➢ Mixed above

- Core will keep order between Device Read and Write, which means core will hold Device Read/Write until all previous Device Write/Read completes.

3.4.3. PPI Interface

The configurable PPI (Private Peripheral Interface) is used to access private peripherals. PPI is AHB-Lite protocol interface as shown in Table 3-7.

Table 3-7 PPI signals

Signal name	Dir	Bit width	Description
ppi_ahbl_clk_en	Input	1	■ PPI interface clock ratio
ppi_ahbl_htrans	Output	2	■ AHB-Lite protocol HTRANS signal.
ppi_ahbl_hwrite	Output	1	■ AHB-Lite protocol HWRITE signal.
ppi_ahbl_haddr	Output	32	■ AHB-Lite protocol HADDR signal.
ppi_ahbl_hsize	Output	3	■ AHB-Lite protocol HSIZE signal.
ppi_ahbl_hprot	Output	4	■ AHB-Lite protocol HPROT signal。
ppi_ahbl_hwdata	Output	32	■ AHB-Lite protocol HWDATA signal.
ppi_ahbl_hrdata	Input	32	■ AHB-Lite protocol HRDATA signal.
ppi_ahbl_hresp	Input	2	■ AHB-Lite protocol HRESP signal. ■ Note: support OKAY and ERROR only.
ppi_ahbl_hready	Input	1	■ AHB-Lite protocol HREADY signal.

3.4.4. SLAVE Interface

The Slave interface is used for external masters (DMA or other Masters) to access the internal ILM/DLM0/1, the Slave interface can be configured as AXI protocol, which lists in Table 3-8.

Table 3-8 Slave signals

Signal name	Dir	Bit width	Description
slv_arready	Output	1	■ AXI protocol ARREADY signal.
slv_arvalid	Input	1	■ AXI protocol ARVALID signal.
slv_arid	Input	4	■ AXI protocol ARID signal.
slv_araddr	Input	32	■ AXI protocol ARADDR signal.
slv_arlen	Input	8	■ AXI protocol ARLEN signal.
slv_arsize	Input	3	■ AXI protocol ARSIZE signal.
slv_arburst	Input	2	■ AXI protocol ARBURST signal.

slv_arregion	Input	4	■ AXI protocol ARREGION signal.
			arregion[0]:
			0 specifies ILM region;
			1 specifies DLM region;

slv_awready	Output	1	■ AXI protocol AWREADY signal.
slv_awvalid	Input	1	■ AXI protocol AWVALID signal.
slv_awid	Input	4	■ AXI protocol AWID signal.
slv_awaddr	Input	32	■ AXI protocol AWADDR signal.
slv_awlen	Input	8	■ AXI protocol AWLEN signal.
slv_awsize	Input	1	■ AXI protocol AWSIZE signal.
slv_awburst	Input	2	■ AXI protocol AWBURST signal.

slv_awregion	Input	4	■ AXI protocol AWREGION signal.
			awregion[0]:
			0 specifies ILM region;
			1 specifies DLM region;

slv_wready	Output	1	■ AXI protocol WREADY signal.
slv_wvalid	Input	1	■ AXI protocol WVALID signal.
slv_wdata	Input	64	■ AXI protocol WDATA signal.
slv_wstrb	Input	8	■ AXI protocol WSTRB signal.
slv_wlast	Input	1	■ AXI protocol WLAST signal.
slv_rready	Input	1	■ AXI protocol RREADY signal.
slv_rvalid	Output	1	■ AXI protocol RVALID signal.
slv_rid	Output	4	■ AXI protocol RID signal.
slv_rdata	Output	64	■ AXI protocol RDATA signal.
slv_rresp	Output	2	■ AXI protocol RRSP signal.
slv_rlast	Output	1	■ AXI protocol RLAST signal.
slv_bready	Input	1	■ AXI protocol BREADY signal.
slv_bvalid	Output	1	■ AXI protocol BVALID signal.
slv_bid	Output	4	■ AXI protocol BID signal.
slv_bresp	Output	2	■ AXI protocol BRESP signal.

slv_bus_clk_en	Input	1	■ Slave interface clock ratio.

Note:

- ■ Do not support AR/AW channel LOCK/CACHE/PROT/QOS/USR operations.
- ■ For AR/AW channel:
  - if BURST type is WRAP, then LEN must be 4 and SIZE must be 3;
  - if BURST type is FIXED or INCR, then LEN must be 0 and SIZE can be 0/1/2;

3.5. NICE Interface

The configurable NICE interface is used to allow user to extend the custom instructions according to their applications. Please refer to another document <Nuclei_NICE_Extension> for the details. User can easily get the copy from "Nuclei User Center" website http://user.nucleisys.com.

3.6. Trace Interface

The Trace interface is used to output the internal key information from the Core, as shown in Table 3-9.

Table 3-9 Trace Interface signals

Signal Name	Direction	Width	Description
trace_ivalid	Output	1	■ Indicate there is a valid instruction is committing or entering trap (exception, NMI, and interrupts).
trace_iexception	Output	1	■ Indicating the Core is entering exception or NMI mode.
trace_interrupt	Output	1	■ Indicating the Core is entering interrupt mode.
trace_cause	Output	64	■ Indicating the cause of trap (same as mcause).
trace_tval	Output	64	■ Indicating the value of exception (same as mtval).
trace_iaddr	Output	64	■ Indicating the PC of current instruction.
trace_instr	Output	64	■ Indicating the instruction code of current instruction.
trace_priv	Output	2	■ Indicating the current privilege mode.

trace_bjp_taken	Output	1	■ Indicating the final result of branch or jump instruction is taken.
trace_dmode	Output	1	■ Indicating whether current instruction is in debug mode.

3.7. I-Cache SRAM Interface

The I-Cache SRAM interface is the interface of Data RAM and Tag RAM used in Instruction Cache, as shown in Table 3-10.

Table 3-10 I-Cache SRAM Interface

Signal Name	Direction	Width	Description
icache_disable_init	Input	1	■ If this signal is 0, normally after reset, the I-Cache's Tag RAM will be cleared to zero with hundreds or thousands of cycles (depends on the cache size). During this period, all the instruction fetches in the I-Cache/System-Bus path will be blocked. ■ If this signal is 1, then the I-Cache's Tag RAM clear-to-zero operations will be skipped.
clk_icache_tag*n*	Output	1	■ Tag RAM*n*'s CLK signal
icache_tag*n*_cs	Output	1	■ CS signal of Tag RAM*n*
icache_tag*n*_we	Output	1	■ Write enable of Tag RAM*n*
icache_tag*n*_addr	Output	IM	■ Address of Tag RAM*n*
icache_tag*n*_wdata	Output	IM	■ Write data of Tag RAM*n*
icache_tag*n*_rdata	Input	IM	■ Read data of Tag RAM*n*
clk_icache_data*n*	Output	1	■ Data RAM*n*'s CLK signal
icache_data*n*_cs	Output	1	■ CS signal of Data RAM*n*
icache_data*n*_wem	Output	IM	■ Write enable of Data RAM*n*
icache_data*n*_addr	Output	IM	■ Address of Data RAM*n*
icache_data*n*_wdata	Output	IM	■ Write data of Data RAM*n*
icache_data*n*_rdata	Input	IM	■ Read data of Data RAM*n*

Note:

- IM indicates Implementation dependent, replies on configurations;
- I-Cache can be configured with 1/2/4/8 ways, so *n* can be "0/1/2/3/4/5/6/7" accordingly;

3.8. D-Cache SRAM Interface

The D-Cache SRAM interface is the interface of Data RAM and Tag RAM used in Data Cache, as shown below.

Table 3-11 D-Cache SRAM Interface

Signal Name	**Direction**	**Width**	**Description**
dcache_disable_init	Input	1	■ If this signal is 0, normally after reset, the D-Cache's Tag RAM will be cleared to zero with hundreds or thousands of cycles (depends on the cache size). During this period, all the instruction fetches in the D-Cache/System-Bus path will be blocked. ■ If this signal is 1, then the D-Cache's Tag RAM clear-to-zero operations will be skipped.
clk_dcache_w*n*_tram	Output	1	■ Tag RAM*n*'s CLK signal
dcache_w*n*_tram_cs	Output	1	■ CS signal of Tag RAM*n*
dcache_w*n*_tram _we	Output	1	■ Write enable of Tag RAM*n*
dcache_w*n*_tram _addr	Output	IM	■ Address of Tag RAM*n*
dcache_w*n*_tram_din	Output	32	■ Write data of Tag RAM*n*
dcache_w*n*_tram_dout	Input	32	■ Read data of Tag RAM*n*
clk_dcache_dram_b*n*	Output	1	■ Data RAM*n*'s CLK signal
dcache_dram_b*n*_cs	Output	1	■ CS signal of Data RAM*n*
dcache_dram_b*n*_wem	Output	4	■ Write enable of Data RAM*n*
dcache_dram_b*n*_addr	Output	IM	■ Address of Data RAM*n*
dcache_dram_b*n*_din	Output	32	■ Write data of Data RAM*n*
dcache_dram_b*n*_dout	Input	32	■ Read data of Data RAM*n*

Note:

- IM indicates Implementation dependent, replies on configurations;
- D-Cache is 2 ways, so *n* can be "0/1" accordingly;

3.9. MMU TLB SRAM Interface

The TLB SRAM interface is the interface of RAM used in TLB, as shown below.

Table 3-12 D-Cache SRAM Interface

Signal Name	**Direction**	**Width**	**Description**

mmu_tlb_disable_init	Input	1	■ If this signal is 0, normally after reset, the TLB RAM will be cleared to zero automatically in hundreds of cycles (depends on the TLB size). ■ If this signal is 1, then the TLB RAM clear-to-zero operations will be skipped.
clk_mmu_tlb_way*n*	Output	1	■ TLB way*n*'s RAM CLK signal
tlb_tram_way*n*_cs	Output	1	■ CS signal of TLB way*n* Tag RAM
tlb_dram_way*n*_cs	Output	1	■ CS signal of TLB way*n* Data RAM
tlb_tram_way*n*_we	Output	1	■ Write enable of TLB way*n* Tag RAM
tlb_dram_way*n*_we	Output	1	■ Write enable of TLB way*n* Data RAM
tlb_tram_way*n*_addr	Output	IM	■ Address of TLB way*n* Tag RAM
tlb_dram_way*n*_addr	Output	IM	■ Address of TLB way*n* Data RAM
tlb_tram_way*n*_wdata	Output	IM	■ Write data of TLB way*n* Tag RAM
tlb_dram_way*n*_wdata	Output	IM	■ Write data of TLB way*n* Data RAM
tlb_tram_way*n*_dout	Input	IM	■ Read data of TLB way*n* Tag RAM
tlb_dram_way*n*_dout	Input	IM	■ Read data of TLB way*n* Data RAM

Note:

- IM indicates Implementation dependent, replies on configurations;
- TLB is 2 ways, so *n* can be "0/1" accordingly;

3.10. Other Functional Interface

Table 3-13 Other Interface signals

Signal Name	Direction	Width	Description
tx_evt	Output	1	■ 600 Series Core use this txevt output a pulse signal as the transmitting Event signal. ■ Please refer to "Nuclei_RISCV_ISA_Spec" for more details of CSR register txevt.
rx_evt	Input	1	■ The receiving Event as the event to wake up Core from WFE mode. ■ Please refer to Section 2.13 for more details of WFE.
mtime_toggle_a	Input	1	■ The mtime_toggle_a is a periodic pulse signal (normally as System Tick) from the SoC system, and used to drive the counter of the internal TIMER unit inside the Core. ■ Note: ● This signal is treated as an asynchronous input signal, and is synchronized within the Core (using

			several DFF synchronizers). ● After the synchronization, both the rising edge and falling edge of the signal are sampled by the core_clk_aon of the Core, and any detected edge will trigger the TIMER counter to increment. ● It is recommended that use the output of the register driven by the "slow clock" (e.g., rtc_clk, whose frequency is the divided frequency of core_clk_aon) as the input of this signal. Then the self-increment frequency is equal to the frequency of the "slow clock", as shown in the Figure 3-1. Hence, the lower the frequency of the slow clock, the lower the self-increment frequency of the internal timer, the lower the dynamic power consumption.
dbg_toggle_a	Input	1	■ The dbg_toggle_a is a periodic pulse signal from the SoC system, and used to drive the time-out counter of the DEBUG unit inside the Core. ■ The time-out counter of DEBUG unit is used to protect the case that if the JTAG Debugger Probe is unexpectedly pulled out which leave the DEBUG unit in an uncertain state. ■ Note: ● This signal is treated as an asynchronous input signal, and is synchronized within the Core (using several DFF synchronizers). ■ In order to make the time-out upper limit to around 170-320ms, it is recommended to use 25kHz~50kHz system real-time-clock as the slow clock to generate this dbg_toggle_a signal, the generation scheme of which is similar to Figure 3-1.
hart_id	Input	32	■ The Core's HART ID indication signal, when integration in SoC, this input can be tied to a specific constant value, and the value of it will be reflected in CSR register mhartid inside the Core. ■ In single Core case, this signal should be tied as zero.
reset_vector	Input	64	■ This signal is to indicate the PC value of the first instruction to be fetched after reset. ■ User can use this signal at SoC level to control the PC address of first instruction executed after reset.
hart_halted	Output	1	■ If this output signal is 1, it is indicating the Core is under debug mode.
i_dbg_stop	Input	1	■ If this input signal is 1, then the Core's Debug functionality will be disabled, and the external Debug Hardware Probe cannot debug the Core through JTAG interface.
sysrstreq	Output	1	■ This output signal is the System Reset request generated from the JTAG. The SoC integrator can use this signal to trigger the Core's core_reset_n to reset the Core except the JTAG logics Note: 1.please do not trigger por_reset_n; 2.please do not disable core_clk_aon;

			3.please delay it one cycle on SoC;
core_wfi_mode	Output	1	■ If this signal is 1, then indicating the Core is under sleep mode. ■ Please refer to Section 2.13 for more details of sleep modes.
core_sleep_value	Output	1	■ This signal is to indicate the shallow sleep or deep sleep mode. ■ Please refer to Section 2.13 for more details of sleep modes.
override_dm_sleep	Input	1	■ This signal can be configured in SoC to enable the clock of debug module when deep sleep mode (core_sleep_value=1), to support debug in low-power.
icache_ecc_sberr_pulse	Output	1	■ To indicate that I-Cache has 1-bit ECC error, if I-Cache configured
dcache_ecc_sberr_pulse	Output	1	■ To indicate that D-Cache has 1-bit ECC error, if D-Cache configured
ilm_ecc_sberr_pulse	Output	1	■ To indicate that ILM has 1-bit ECC error, if ILM configured
dlm_ecc_sberr_pulse	Output	1	■ To indicate that DLM has 1-bit ECC error, if DLM configured
tlb_ecc_sberr_pulse	Output	1	■ To indicate that TLB has 1-bit ECC error, if TLB configured
icache_ecc_dberr_pulse	Output	1	■ To indicate that I-Cache has 2-bit ECC error, if I-Cache configured
dcache_ecc_dberr_pulse	Output	1	■ To indicate that D-Cache has 2-bit ECC error, if D-Cache configured
ilm_ecc_dberr_pulse	Output	1	■ To indicate that ILM has 2-bit ECC error, if ILM configured
dlm_ecc_dberr_pulse	Output	1	■ To indicate that DLM has 2-bit ECC error, if DLM configured
tlb_ecc_dberr_pulse	Output	1	■ To indicate that TLB has 2-bit ECC error, if TLB configured

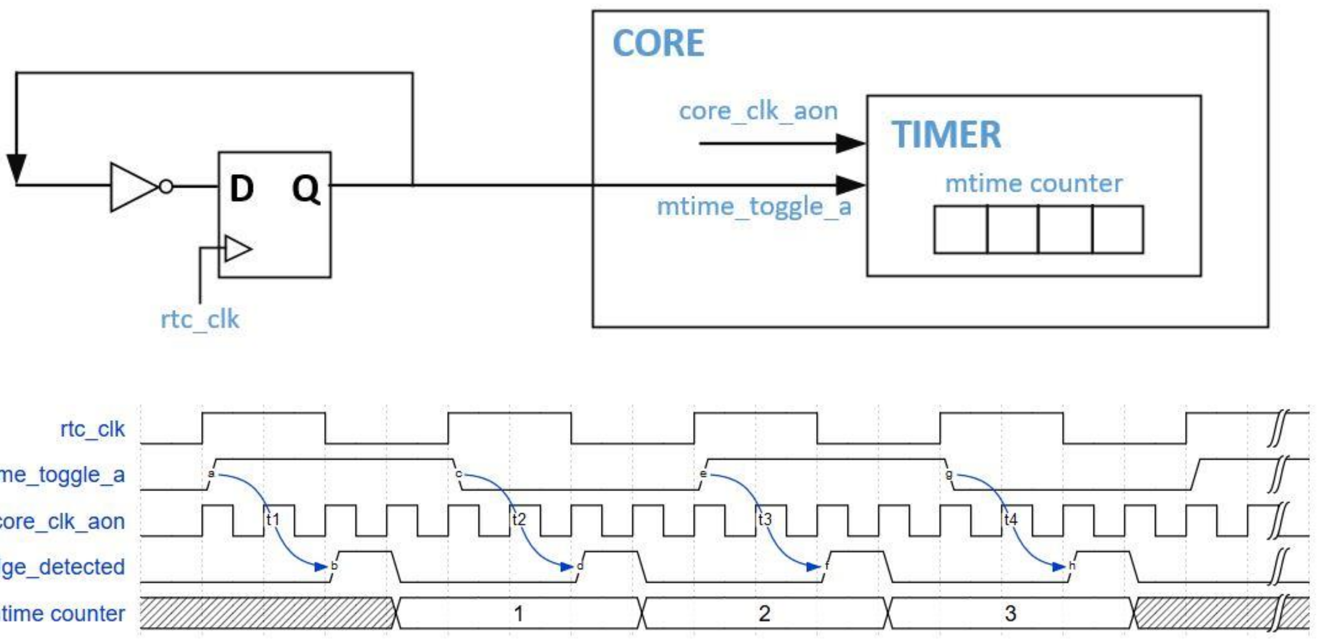

Figure 3-1 mtime_toggle_a signal generation

4. Configurable Options

600 Series Core is fully configurable. The configurable options are as shown in Table 4-1.

Note: about how to configure the processor IP package with GUI tools, please refer to another document <Nuclei_Processor_Integration_Guide> for the details. User can easily get the copy from "Nuclei User Center" website http://user.nucleisys.com.

Table 4-1 The configurable option of 600 Series

Categories	Macro	Description
ISA	600_CFG_ISA_RV64I	■ To configure RV64 or RV32
PA_SIZE	600_CFG_PA_SIZE	■ To configure the PA width: 32-56bits. PA is fixed to be 32bits when RV32.
User Mode	600_CFG_HAS_UMODE	■ This Macro configures to have the user mode.
TEE	600_CFG_HAS_TEE	■ This Macro configures to have the TEE feature.
Integer Multiplier	600_CFG_BOOTH_MUL_nCYC	■ Configure MUL stages: 1/2/3
PMP	600_CFG_HAS_PMP	■ This Macro configures to have the PMP feature. ■ Note: this option only appeared if the User Mode have been configured.
	600_CFG_PMP_ENTRY_NUM	■ This Macro configures the PMP entries number. ● 8: Means PMP has 8 entries. ● 16: Means PMP has 16 entries.
Debug Features	600_CFG_HAS_DEBUG	■ This Macro configures to have the Debug functionalities. ■ Note: the Debug unit cost about 4K Gates resource.
	600_CFG_DEBUG_BASE_ADDR	■ This Macro to configure the base address of the Debug unit. ■ Note: the Debug unit will occupy 4K address space starting from its base address.
	600_CFG_DEBUG_TRIGM_NUM	■ This Macro to configure the number of Hardware Trigger (2/4/8). ■ Note: each Trigger cost about 64bits DFFs resource.
	600_CFG_DEBUG_COUNTLEN	■ This Macro to configure the time-out counter's width of "DEBUG time-out protection" feature. ■ Note: the timer-out upper limit is calculated by 2^600_CFG_DEBUG_COUNTLEN/(

		2*FREQ_dbg_toggle_a). User should configure this Macro to make the upper limit to around 170-320ms.
I-Cache	600_CFG_HAS_ICACHE	■ This Macro configures to have I-Cache.
	600_CFG_ICACHE_WAY	■ This Macro to configure Icache way num: 2/4/8;
D-Cache	600_CFG_HAS_DCACHE	■ This Macro configures to have D-Cache.
	600_CFG_DCACHE_ADDR_WIDTH	■ This Macro to configure the address space of Dcache. ■ For example, if the ADDR_WIDTH is 10, then Dcache size is 1KB;
	600_CFG_DCACHE_HAS_TWO_STAGES	■ To configure one more register flop stage to easy timing in D-Cache stage, but add 1 more cycle latency.
	600_CFG_DEVICE_REGION_NUM	■ To configure the num of Device regions.
	600_CFG_DEVICE_REGION*n*_BASE	■ Configure Device Region *n* base addr (*n* can be 0-7);
	600_CFG_DEVICE_REGION*n*_MASK	■ Configure Device Region *n* addr mask (*n* can be 0-7); ■ For example, if BASE is 0x10000000, MASK is 0xfffff000, then this REGION will be 0x10000000 – 0x10000fff;
	600_CFG_NC_REGION_NUM	■ To configure the num of Non-Cacheable regions.
	600_CFG_NC_REGION*n*_BASE	■ Configure Non-Cacheable Region *n* base addr (*n* can be 0-7);
	600_CFG_NC_REGION*n*_MASK	■ Configure Non-Cacheable Region *n* addr mask (*n* can be 0-7); ■ For example, if BASE is 0x10000000, MASK is 0xfffff000, then this REGION will be 0x10000000 – 0x10000fff;
Local Memory	600_CFG_HAS_ILM	■ This Macro configures to have ILM.
	600_CFG_ILM_BASE_ADDR	■ This Macro to configure the base address of the ILM.
	600_CFG_ILM_ADDR_WIDTH	■ This Macro to configure the address space of ILM. ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address space of ILM is 0x1000_0000 ~0x100F_FFFF.
	600_CFG_HAS_DLM	■ This Macro configures to have DLM.
	600_CFG_DLM_BASE_ADDR	■ This Macro to configure the base address of the DLM.
	600_CFG_DLM_ADDR_WIDTH	■ This Macro to configure the address space of DLM. ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address space of DLM is 0x1000_0000

		~0x100F_FFFF.
	600_CFG_HAS_TWO_STAGES	■ To configure one more register flop stage in DLM, but add 1 more cycle latency.
	600_CFG_HAS_SLAVE_PORT	■ Configure Slave interface for external masters to access internal ILM/DLM;
	600_CFG_SLV_BID_W	■ Configure the slave interface AXI ID width.
PPI	600_CFG_HAS_PPI	■ This Macro configures to have PPI.
	600_CFG_PPI_BASE_ADDR	■ This Macro to configure the base address of the PPI interface.
	600_CFG_PPI_ADDR_WIDTH	■ This Macro to configure the address space of PPI. ■ For example, if the ADDR_WIDTH is 20, and the BASE_ADDR is 0x1000_0000, then the address space of PPI is 0x1000_0000 ~0x100F_FFFF.
Interrupt	600_CFG_IRQ_NUM	■ To configure the number of external interrupts, can be up to 1023 (PLIC) or 1005 (ECLIC).
	600_CFG_CUT_TIMING	■ To add one more register flop stage to easy the timing of interrupt arbitrations when big num of interrupts configured, but add 1 more cycle for interrupt latency.
PLIC	600_CFG_HAS_PLIC	■ To configure PLIC
	600_CFG_PLIC_BASE_ADDR	■ To configure the base address of the PLIC.
	600_CFG_PLIC_PRIO_WIDTH	■ To configure the bit width of priority.
ECLIC	600_CFG_HAS_CLIC	■ To configure CLIC.
	600_CFG_CLIC_BASE_ADDR	■ This Macro to configure the base address of the ECLIC. Please refer to Section 2.6 for more details.
	600_CFG_CLICINTCTLBITS	■ This Macro to configure the bits width (range from 1 to 8) of level registers in ECLIC. ■ For example, if this Macro is configured as 3, then ECLIC can support 8 levels; if this Macro is configured as 8, then ECLIC can support 256 levels.
TIMER	600_CFG_TMR_BASE_ADDR	■ This Macro to configure the base address of the TIMER. Please refer to Section 2.6 for more details.
NICE	600_CFG_HAS_NICE	■ This Macro configures to have NICE feature, to allow user to extend their custom instructions.
DSP	600_CFG_HAS_DSP	■ This Macro configures to have packed-SIMD DSP support.
FPU	600_CFG_FPU_SINGLE	■ This Macro configures to have single-precision floating point support.

	600_CFG_FPU_DOUBLE	■ This Macro configures to have both single-precision and double-precision floating point support.
	600_CFG_4CYC_FPU	■ This Macro configures to have floating point MAC unit as 4 cycles latency (better timing and worse performance), otherwise it is 3cycles.
MMU	UX600_CFG_HAS_MMU	■ This Macro is to configure the MMU.
	UX600_CFG_TLB_INDEX_WIDTH	■ To configure the TLB index width for entry num: 5/6/7 (32/64/128 entries)
	UX600_CFG_ASID_WIDTH	■ To configure the ASID bit-width: 1-16
ECC	600_CFG_HAS_ECC	■ To configure the ECC feature on ILM, DLM, I-Cache, D-Cache, TLB, if configured.