altera
An Intel Company

Nios® V Processor Reference Manual

Updated for Quartus® Prime Design Suite: **24.3**

Online Version
Send Feedback

683632
2024.10.07

Contents

1. Overview

The Nios® V processor is a soft processor core with the following characteristics:

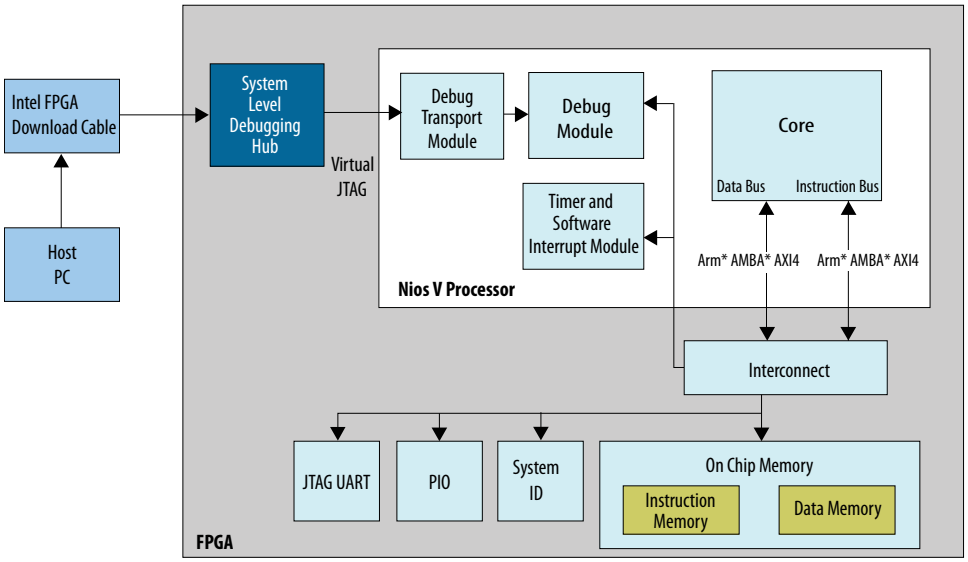
- Designed for Intel® FPGA devices and based on RISC-V* specification
- Does not include peripherals or the connection logic to external devices
- Includes only the circuitry required to implement the Nios V processor architecture

The following table lists the available variants:

Table 1. Nios V Processor Variants

Processor Core	Variant
Nios V/c	Compact Microcontroller
Nios V/m	Microcontroller
Nios V/g	General-purpose processor

Figure 1. Nios V Processor System on Intel FPGA



Related Information

- RISC-V® Specifications
- Nios V/m Processor on page 17
  More information about the Nios V/m processor (Microcontroller).
- Nios V/g Processor on page 43
  More information about the Nios V/g processor (General-purpose processor).

1.1. Quartus® Prime Software Support

Nios V processor build flow is different for Quartus® Prime Pro Edition software and Quartus Prime Standard Edition software. Refer to *AN 980: Nios V Processor Quartus Prime Software Support* for more information about the differences.

Related Information

AN 980: Nios V Processor Quartus Prime Software Support

2. Nios V/c Processor

The Nios V/c processor is a compact microcontroller core developed by Intel based on the RISC-V RV32I instruction set and supports the functional units described in this document.

2.1. Processor Performance Benchmarks

Table 2. Nios V/c Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software

Quartus Prime Edition	FPGA Used	f_{MAX} (MHz)	Logic Size	Architecture Performance	
				DMIPS/MHz Ratio	CoreMark/MHz Ratio
Quartus Prime Pro Edition	Cyclone® 10	315	402 ALM	0.227	0.17
	Arria® 10	347	405 ALM		
	Stratix® 10	361	444 ALM		
	Agilex™ 7	443	438 ALM		
	Agilex 5	389	434 ALM		
Quartus Prime Standard Edition	Cyclone IV E	118	1022 LE	0.268	0.201
	Cyclone V	155	423 ALM		
	Arria V	175	414 ALM		
	Arria V GZ	289	372 ALM		
	Stratix V	332	372 ALM		
	Cyclone 10 LP	137	1025 LE		
	Arria 10	325	355 ALM		
	MAX® 10	137	1022 LE		

Table 3. Benchmark Parameters for Quartus Prime Software

Parameter		Settings/Description	
		Quartus Prime Pro Edition	Quartus Prime Standard Edition
Quartus Prime seed		Maximum performance result are based on 10 seed sweep from Quartus Prime Pro Edition software version 24.3.	Maximum performance result are based on 10 seed sweep from Quartus Prime Standard Edition software version 23.1.
Device speed grade		Fastest speed grade from each Intel FPGA device family.	
Defined peripherals		• Nios V/c processor core (without debug module and internal timer). • 128 KB on-chip memory for the instruction and data bus. • JTAG UART Intel FPGA IP. • Interval Timer Core.	
Toolchain	Version	• riscv32-unknown-elf-gcc (GCC) version 13.2.0 • CMake Version: 3.29.3	• riscv32-unknown-elf-gcc (GCC) version 12.1.0 • CMake Version: 3.27.1
	Compiler configuration	• Compiler flags: `-03` • Assembler options: `-Wa -gdwarf2` • Compile options: `-Wall -Wformat-security -march=rv32i -mabi=ilp32`	

Altera® uses the same Quartus Prime design example for maximum performance benchmark(fMAX) and logic size benchmarks. The compiler settings are:

- **Superior Performance with Maximum Placement Effort** in Quartus Prime Pro Edition software.
- **High Performance Effort** in Quartus Prime Standard Edition software.

Note: Results may vary depending on the version of the Quartus Prime software, the version of the Nios V processor, compiler version, target device and the configuration of the processor. Additionally, any changes to the system logic design can change the performance and LE usage. All results are generated from design built with Platform Designer.

2.2. Processor Pipeline

The Nios V/c processor supports a non-pipelined datapath.

Table 4. Processor Non-pipelined Stages

Stage	Denotation	Function
F	Instruction fetch	Pre-decode for register file read
D	Instruction decode	• Decode the instruction • Register file read data available • Hazard resolution and data forwarding
E	Instruction execute	• ALU operations • Memory address calculation • Branch resolution • CSR read/write
M	Memory	• Memory and multicycle operations • Register file write • Next PC logic • Branch redirection

The Nios V/c processor implements the general-purpose register file using the M20K memory blocks. The processor takes one clock cycle to read from an M20K location. Therefore, the F-stage initiates register file reads so general-purpose register values are available in D-stage.

One instruction is available in the processor datapath at any time. Instructions flow from F-stage to M-stages without any stalls. Instruction and associated control logic are registered during D-stage, E-stage, and M-stage.

The processor requests the next instruction during the M-stage.

- For single cycle instructions, the processor makes the request as soon as the single cycle instruction enters M-stage.
- For multicycle instructions, the processor makes the request as soon as the multicycle instruction completes.

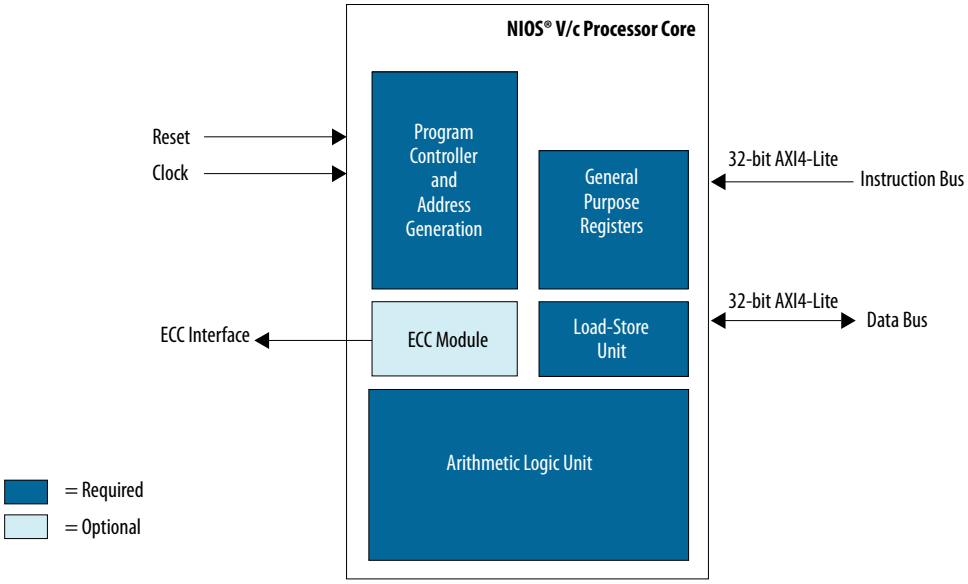
2.3. Processor Architecture

The Nios V/c processor architecture describes an instruction-set architecture (ISA). The ISA in turn necessitates a set of functional units that implement the instructions.

The Nios V processor architecture defines the following functional units:

- General-purpose register file
- Arithmetic logic unit (ALU)
- Instruction bus
- Data bus
- ECC module

Figure 2. Nios V/c Processor Core Block Diagram



2.3.1. General-Purpose Register File

Nios V/c processor implementation supports a flat register file. The register file contains thirty-two 32-bit general-purpose integer registers. Nios V/c processor implements the general-purpose register using M20K memories, which do not support two read ports. Hence, Nios V/c processor duplicates the register files so that two different source registers for an instruction are available in a single cycle. After performing ALU operations, the processor core writes the same result to the destination register in both memories.

2.3.2. Arithmetic Logic Unit

The arithmetic logic unit (ALU) operates on data stored in general-purpose registers. ALU operations take one or two inputs from registers and store the result back into the register.

Table 5. Fundamental Data Operations of the ALU

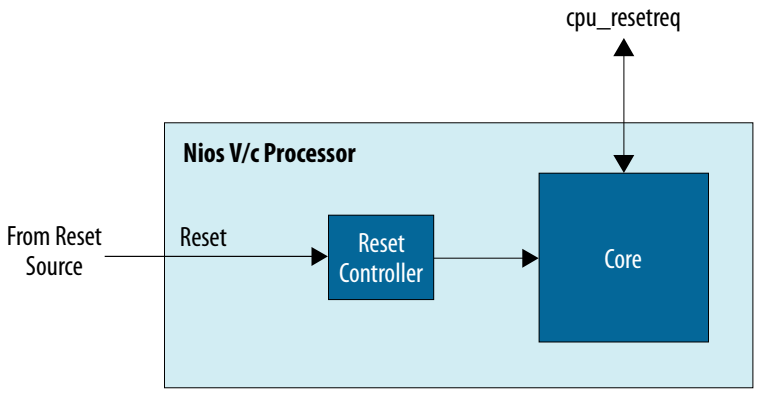
Category	Description
Arithmetic	Addition and subtraction on signed and unsigned operands.
Relational	Equal, not-equal, greater-than-or-equal, and less-than relational operations (==, != >=, <).
Logical	AND, OR, NOR, and XOR logical operations.
Shift	Logical and arithmetic shift operations.

For load and store instructions, the Nios V/c processor uses the ALU to calculate the memory address. For conditional control transfer instructions, Nios V/c processor uses the relational operations in the ALU to determine if the processor takes or leaves the branch.

2.3.3. Reset and Debug Signals

Interface	Type	Description
reset	Reset	A global hardware reset input signal that forces the Nios V processor to reset immediately.
cpu_resetreq	Conduit	An optional local reset ports which appear after you enable **Add Reset Request Interface** parameter. The signal consists of an input resetreq signal and an output ack signal that trigger the Nios V processor to reset without affecting other components in a Nios V processor system. • You can request a reset to the Nios V processor core by asserting the resetreq signal. • The resetreq signal must remain asserted until the processor asserts ack signal. Failure for the signal to remain asserted can cause the processor to be in a non-deterministic state. • Assertion of the resetreq signal in debug mode has no effect on the processor's state. • The Nios V processor responds that the reset is successful by asserting the ack signal. • After the processor is successfully reset, the assertion of the ack signal can happen multiple times periodically until the de-assertion of the resetreq signal.

Figure 3. Nios V/c Processor Reset Network



2.3.4. Memory and I/O Organization

You can configure the Nios V/c processor systems. Consequently, the memory and I/O organization varies from system to system. A Nios V/c processor core uses one or more of the following ports to provide access to memory and I/O:

- Instruction manager port: An Arm* Advanced Microcontroller Bus Architecture (AMBA*) AXI4-Lite Memory-Mapped manager port that connects to instruction memory via system interconnect fabric.
- Data manager port: An AMBA AXI4-Lite Memory-Mapped manager port that connects to data memory and peripherals via the system interconnect fabric.

Nios V/c Processor Core Memory Mapped I/O Access: Both data memory and peripherals are mapped into the address space of the data manager port. Nios V/c processor core uses little-endian byte ordering. Words and half-words are stored in

memory with the more-significant bytes at higher addresses. The Nios V/c processor core does not specify anything about the existence of memory and peripherals. The quantity, type, and connection of memory and peripherals are system dependent.

2.3.4.1. Instruction and Data Buses

2.3.4.1.1. Instruction Manager Port

Nios V/c processor instruction bus is implemented as a 32-bit AMBA 4 AXI-Lite manager port.

The instruction manager port:

- Performs a single function: it fetches instructions to be executed by the processor.
- Does not perform any write operations.
- Can issue successive read requests before data return from prior requests.
- Can prefetch sequential instructions.
- Always retrieves 32-bits of data. Every instruction fetch returns a full instruction word, regardless of the width of the target memory. The widths of memory in the Nios V/c processor system is not applicable to the programs. Instruction address is always aligned to a 32-bit word boundary.
- Does not require any burst adapter because it is non-bursting.

Table 6. Instruction Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	awaddr	Unused	[31:0]	Output
	awprot	Unused	[2:0]	Output
	awvalid	Unused	1	Output
	awready	Unused	1	Input
Write Data Channel	wdata	Unused	[31:0]	Output
	wstrb	Unused	[3:0]	Output
	wvalid	Unused	1	Output
	wready	Unused	1	Input
Write Response Channel	bresp	Unused	[1:0]	Input
	bvalid	Unused	1	Input
	bready	Unused	1	Output
Read Address Channel	araddr	Instruction Address (Program Counter)	[31:0]	Output
	arprot	Unused	[2:0]	Output
	arvalid	Instruction address valid	1	Output
	arready	Instruction address ready (from memory)	1	Input
Read Data Channel	rdata	Instruction	[31:0]	Input

continued...

Interface	Signal	Role	Width	Direction
	rresp	Instruction response: Non-zero value denotes instruction access fault exception	[1:0]	Input
	rvalid	Instruction valid	1	Input
	rready	Constant 1	1	Output

2.3.4.1.2. Data Manager Port

The Nios V/c processor data bus is implemented as a 32-bit AMBA 4 AXI-Lite manager port. The data manager port:

- Performs read data from memory or a peripheral when the processor executes a load instruction.
- Performs write data to memory or a peripheral when the processor executes a store instruction.
- Does not require any burst adapter because it is non-bursting.

axsize signal value indicates the load/store instruction size- byte (LB/SB), halfword (LH/SH) or word (LW/SW). Address on axaddr signal is always aligned to size of the transfer. For store instructions, respective write strobe bits are asserted to indicate bytes being written.

Nios V/c processor core does not support speculative issue of load/store instruction. Hence, a core can issue only one load or store instruction and waits until the issued instruction is complete.

Table 7. Data Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	awaddr	Store address	[31:0]	Output
	awprot	Unused	[2:0]	Output
	awvalid	Store address valid	1	Output
	awready	Store address ready (from memory)	1	Input
Write Data Channel	wdata	Store data	[31:0]	Output
	wstrb	Byte position in word	[3:0]	Output
	wvalid	Store data valid	1	Output
	wready	Store data ready (from memory)	1	Input
Write Response Channel	bresp	Store response: Non-zero value denotes store access fault exception.	[1:0]	Input
	bvalid	Store response valid	1	Input
	bready	Constant 1	1	Output
Read Address Channel	araddr	Load address	[31:0]	Output
	arprot	Unused	[2:0]	Output
	arvalid	Load address valid	1	Output

continued...

Interface	Signal	Role	Width	Direction
	arready	Load address ready (from subordinates)	1	Input
Read Data Channel	rdata	Load data	[31:0]	Input
	rresp	Load response: Non-zero value denotes load access fault exception	[1:0]	Input
	rvalid	Load data valid	1	Input
	rready	Constant 1	1	Output

2.3.4.1.3. Choosing a Suitable Interface

The Nios V/c processor core offers an Avalon® memory-mapped and Advanced eXtensible Interface (AXI) interface for both the instruction manager port and the data manager port. You can choose one of the interfaces in Platform Designer based on your design.

For example, if most IPs in Platform Designer use Avalon memory-mapped interface, you can choose the Avalon memory-mapped interface in the Nios V/c processor core as the bridge interface. Similarly, if most IPs in Platform Designer use AXI Interface, you can choose the AXI Interface in the Nios V/c processor core as the bridge interface.

The benefits of using the appropriate interface are as follows:

- Reduces the amount of bridging logic
- Reduces overhead bandwidth
- Provides better performance

2.3.4.2. Address Map

The address map for memories and peripherals in a Nios V/c processor system is design dependent. The Reset Address is part of the processor. You can specify the Reset Address in Platform Designer during system configuration.

2.3.5. Error Correction Code (ECC)

The Nios V/c processor core has the option to enable error detection and ECC status reporting for the RAM block, that is the Register file. Each RAM block has its own source ID. When an ECC event occurs, the processor transmits the source ID and ECC status to the ECC interface.

- If the ECC event is a correctable error, the processor continues to operate after correcting the error. The correction made is not written back to its memory source.
- If the ECC event is an un-correctable error, the processor halts its current progress and stalls. You need to reset either the processor core alone or the entire system.

Note: To reset only the processor core, apply the **Reset Request Interface** to safely reset the Nios V processor (cleared of any outstanding operations). To reset the entire system, you can use the hard reset interface instead.

The ECC interface allows external logic to monitor ECC errors from the Nios V/c processor. The interface is a conduit, made up of the following output signals.

- `cpu_ecc_status` : Indicates the error status
- `cpu_ecc_source` : Indicates the error source.

Table 8. `cpu_ecc_status`

2-bits Encoding	Description	Effects on Software
2'b00	No ECC event	None
2'b01	Reserved	Not Applicable
2'b10	Correctable single bit ECC error	None
2'b11	Un-correctable ECC error	Likely fatal and halts the processor.

Table 9. `cpu_ecc_source`

4-bits Encoding	ECC Source	Available
4'b0000	No ECC event	Always
4'b0001	General Purpose Register (GPR)	Always
4'b0010 ~ 4'b1110	Other RAM Blocks	Not Available
4'b1111	Reserved	Not Applicable

Note: Due to a limitation with embedded memory blocks, the simulation model of Nios V processor does not support ECC on Arria 10 devices.

Related Information

Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide
For more information about ECC on Arria 10 devices.

2.4. Programming Model

2.4.1. Privilege Levels

The privilege levels in Nios V/c processor are designed based on the RISC-V architecture specification. The Nios V/c processor only supports Machine Mode (M-mode), which provides low-level access to the machine implementation.

Related Information

RISC-V® Specifications

2.5. Core Implementation

2.5.1. Instruction Set Reference

The Nios V/c processor is based on the RISC-V RV32I specification. RV32I is a base integer instruction set. RV321 supports 6 types of instruction format. They are R-type, I-type, S-type, B-type, U-type, and J-type.

Table 10. Instruction Formats (R-type)

Bit Field (R-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
funct7							rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			rd			opcode								

Table 11. Instruction Formats (I-type)

Bit Field (I-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[11:0]												rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			rd			opcode								

Table 12. Instruction Formats (S-type)

Bit Field (S-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[11:5]							rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			imm[4:0]			opcode								

Table 13. Instruction Formats (B-type)

Bit Field (B-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[12]	imm[10:5]						rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			imm[4:1]			imm[11]		opcode						

Table 14. Instruction Formats (U-type)

Bit Field (U-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[15:12]				rd			opcode								

Table 15. Instruction Formats (J-type)

Bit Field (J-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[20]	imm[10:1]										imm[11]	imm[19:16]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[15:12]				rd					opcode						

Related Information

RISC-V® Specifications
More information about the instruction set of the RV32I Base Integer.

3. Nios V/m Processor

The Nios V/m processor is a microcontroller core developed by Intel based on the RISC-V instruction set and supports the functional units described in this document.

The Nios V/m processor supports two distinct configurations:

- Pipelined
  - Implements RV32IZicsr instruction set.
  - Supports five-stages pipelined datapath.
- Non-pipelined
  - Implements RV32IZicsr instruction set.
  - Supports non-pipelined datapath.

Related Information

Overview on page 4

3.1. Processor Performance Benchmarks

3.1.1. Pipelined

Table 16. Nios V/m Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software

Quartus Prime Edition	FPGA Used	fMAX (MHz)	Logic Size	Architecture Performance	
				DMIPS/MHz Ratio	CoreMark/MHz Ratio
Quartus Prime Pro Edition	Cyclone 10	276	1064 ALM	0.63	0.49
	Arria 10	281	1038 ALM		
	Stratix 10	324	1341 ALM		
	Agilex 7	410	1252 ALM		
	Agilex 5	311	1288 ALM		
Quartus Prime Standard Edition	Cyclone IV E	107	2831 LE	0.650	0.441
	Cyclone V	145	1253 ALM		
	Arria V	146	1222 ALM		
	Arria V GZ	264	1215 ALM		
	Stratix V	304	1199 ALM		

continued...

Quartus Prime Edition	FPGA Used	fMAX (MHz)	Logic Size	Architecture Performance	
				DMIPS/MHz Ratio	CoreMark/MHz Ratio
	Cyclone 10 LP	135	2848 LE		
	Arria 10	290	1132 ALM		
	MAX 10	123	2864 LE		

Table 17. Benchmark Parameters for Quartus Prime Software

Parameter		Settings/Description	
		Quartus Prime Pro Edition	Quartus Prime Standard Edition
Quartus Prime seed		Maximum performance result are based on 10 seed sweep from Quartus Prime Pro Edition software version 24.3.	Maximum performance result are based on 10 seed sweep from Quartus Prime Standard Edition software version 23.1.
Device speed grade		Fastest speed grade from each Intel FPGA device family.	
Defined peripherals		• Nios V/m processor core (without debug module and internal timer).	• Nios V/m processor core (without debug module).
		• 128 KB on-chip memory for the instruction and data bus. • JTAG UART Intel FPGA IP. • Interval Timer Core.	
Toolchain	Version	• riscv32-unknown-elf-gcc (GCC) version 13.2.0 • CMake Version: 3.29.3	• riscv32-unknown-elf-gcc (GCC) version 12.1.0 • CMake Version: 3.27.1
	Compiler configuration	• Compiler flags: `-03` • Assembler options: `-Wa -gdwarf2` • Compile options: `-Wall -Wformat-security -march=rv32i -mabi=ilp32`	

Intel uses the same Quartus Prime design example for maximum performance benchmark(fMAX) and logic size benchmarks. The compiler settings are:

- **Superior Performance with Maximum Placement Effort** in Quartus Prime Pro Edition software.
- **High Performance Effort** in Quartus Prime Standard Edition software.

Note: Results may vary depending on the version of the Quartus Prime software, the version of the Nios V processor, compiler version, target device and the configuration of the processor. Additionally, any changes to the system logic design might change the performance and LE usage. All results are generated from design built with Platform Designer.

3.1.2. Non-pipelined

Table 18. Nios V/m Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software

Quartus Prime Edition	FPGA Used	f_{MAX} (MHz)	Logic Size	Architecture Performance	
				DMIPS/MHz Ratio	CoreMark/MHz Ratio
Quartus Prime Pro Edition	Cyclone 10	311	724 ALM	0.227	0.170
	Arria 10	337	742 ALM		
	Stratix 10	354	794 ALM		
	Agilex 7	436	826 ALM		
	Agilex 5	336	769 ALM		
Quartus Prime Standard Edition	Cyclone IV E	117	1598 LE	0.268	0.201
	Cyclone V	144	705 ALM		
	Arria V	159	708 ALM		
	Arria V GZ	281	658 ALM		
	Stratix V	330	641 ALM		
	Cyclone 10 LP	135	1604 LE		
	Arria 10	316	559 ALM		
	MAX 10	127	1619 LE		

Table 19. Benchmark Parameters for Quartus Prime Software

Parameter		Settings/Description	
		Quartus Prime Pro Edition	Quartus Prime Standard Edition
Quartus Prime seed		Maximum performance result are based on 10 seed sweep from Quartus Prime Pro Edition software version 24.3.	Maximum performance result are based on 10 seed sweep from Quartus Prime Standard Edition software version 23.1.
Device speed grade		Fastest speed grade from each Intel FPGA device family.	
Defined peripherals		• Nios V/m processor core (without debug module and internal timer). • 128 KB on-chip memory for the instruction and data bus. • JTAG UART Intel FPGA IP. • Interval Timer Core	
Toolchain	Version	• riscv32-unknown-elf-gcc (GCC) version 13.2.0 • CMake Version: 3.29.3	• riscv32-unknown-elf-gcc (GCC) version 12.1.0 • CMake Version: 3.27.1
	Compiler configuration	• Compiler flags: `-O3` • Assembler options: `-Wa -gdwarf2` • Compile options: `-Wall -Wformat-security -march=rv32i -mabi=ilp32`	

Intel uses the same Quartus Prime design example for maximum performance benchmark(fMAX) and logic size benchmarks. However, the compiler settings are different for each benchmarks:

- **Superior Performance with Maximum Placement Effort** in Quartus Prime Pro Edition software.
- **High Performance Effort** in Quartus Prime Standard Edition software.

Note: Results may vary depending on the version of the Quartus Prime software, the version of the Nios V processor, compiler version, target device and the configuration of the processor. Additionally, any changes to the system logic design might change the performance and LE usage. All results are generated from design built with Platform Designer.

3.2. Processor Pipeline

3.2.1. Pipelined Architecture

The Nios V/m processor employs a five-stage datapath.

Table 20. Processor Pipeline Stages

Stage	Denotation	Function
F	Instruction fetch	• PC+4 calculation • Next instruction fetch • Pre-decode for register file read
D	Instruction decode	• Decode the instruction • Register file read data available • Hazard resolution and data forwarding
E	Instruction execute	• ALU operations • Memory address calculation • Branch resolution • CSR read/write
M	Memory	• Memory and multicycle operations • Register file write • Branch redirection
W	Write back	• Facilitates data dependency resolution by providing general-purpose register value.

The Nios V/m processor implements the general-purpose register file using the M20K memory blocks. The processor takes one cycle to read from an M20K location. Therefore, the F-stage initiates register file reads so general-purpose register values are available in D-stage.

Writing to the M20K location takes two cycles. Therefore, the M-stage initiates writes to a general-purpose register. If there is a dependency to resolve, the M-stage carries forward the value to the W-stage.

The core resolves data dependencies in the D-stage. Operands can move from register file read or E-stage, M-stage, or W-stage.

Reasons for the pipeline stalling:

- Data dependency—if the source operand is not available in the D-stage, instruction in the D-stage and F-stage stalls until the operand becomes available. This happens when the destination general-purpose register of a load or multicycle instruction in the E-stage or M-stage is the source for the instruction in the D-stage.
- Resource stall—if a memory operation or multicycle is pending in the M-stage, the instructions in the preceding stages stall until M-stage completes the instruction.

3.2.2. Non-pipelined Architecture

The Nios V/m processor supports a non-pipelined datapath.

Table 21. Processor Non-pipelined Stages

Stage	Denotation	Function
F	Instruction fetch	Pre-decode for register file read
D	Instruction decode	• Decode the instruction • Register file read data available • Hazard resolution and data forwarding
E	Instruction execute	• ALU operations • Memory address calculation • Branch resolution • CSR read/write
M	Memory	• Memory and multicycle operations • Register file write • Next PC logic • Branch redirection

The Nios V/m processor implements the general-purpose register file using the M20K memory blocks. The processor takes one clock cycle to read from an M20K location. Therefore, the F-stage initiates register file reads so general-purpose register values are available in D-stage.

One instruction is available in the processor datapath at any time. Instructions flow from F-stage to M-stages without any stalls. Instruction and associated control logic are registered during D-stage, E-stage, and M-stage.

The processor requests the next instruction during the M-stage.

- For single cycle instructions, the processor makes the request as soon as the single cycle instruction enters M-stage.
- For multicycle instructions, the processor makes the request as soon as the multicycle instruction completes.

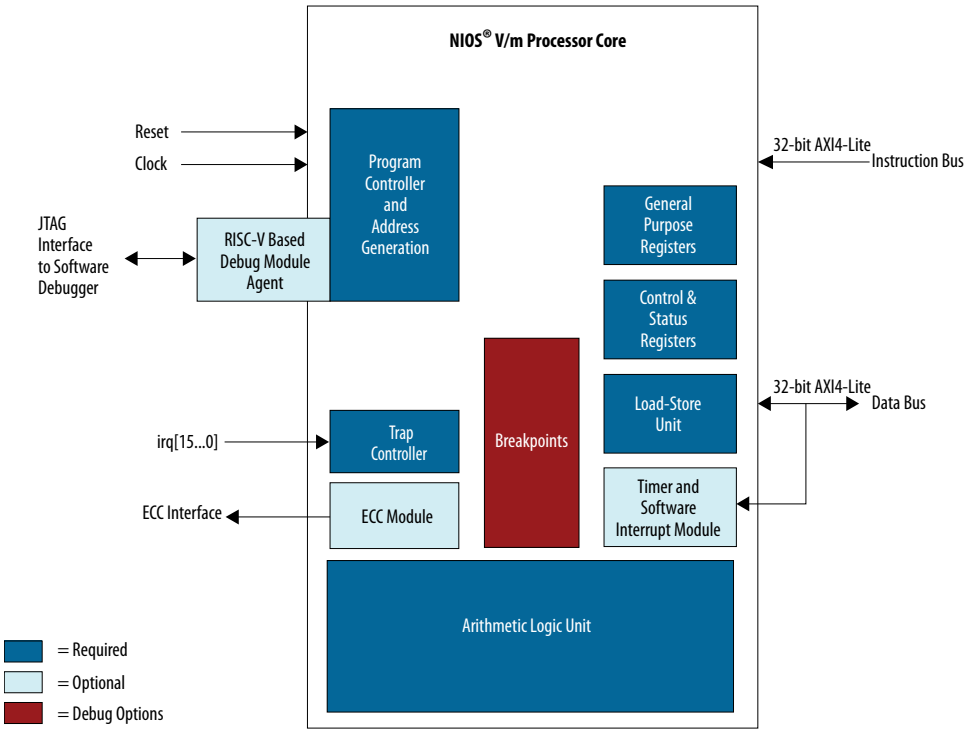
3.3. Processor Architecture

The Nios V/m processor architecture describes an instruction-set architecture (ISA). The ISA in turn necessitates a set of functional units that implement the instructions.

The Nios V/m processor architecture defines the following functional units:

- General-purpose register file
- Arithmetic logic unit (ALU)
- Control and status registers (CSR)
- Trap controller
- Instruction bus
- Data bus
- RISC-V based debug module
- ECC module

Figure 4. Nios V/m Processor Core Block Diagram



3.3.1. General-Purpose Register File

Nios V/m processor implementation supports a flat register file. The register file contains thirty-two 32-bit general-purpose integer registers. Nios V/m processor implements the general-purpose register using M20K memories, which do not support two read ports. Hence, Nios V/m processor duplicates the register files so that two different source registers for an instruction are available in a single cycle. After performing ALU operations, the processor core writes the same result to the destination register in both memories.

3.3.2. Arithmetic Logic Unit

The arithmetic logic unit (ALU) operates on data stored in general-purpose registers. ALU operations take one or two inputs from registers and store the result back into the register.

Table 22. Fundamental Data Operations of the ALU

Category	Description
Arithmetic	Addition and subtraction on signed and unsigned operands.
Relational	Equal, not-equal, greater-than-or-equal, and less-than relational operations (==, != >=, <).
Logical	AND, OR, NOR, and XOR logical operations.
Shift	Logical and arithmetic shift operations.

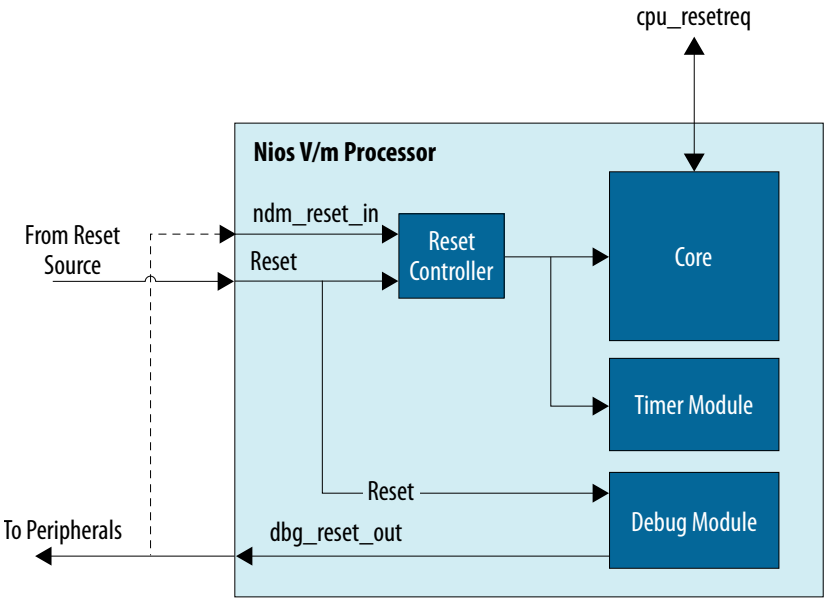
For load and store instructions, the Nios V/m processor uses the ALU to calculate the memory address. For conditional control transfer instructions, Nios V/m processor uses the relational operations in the ALU to determine if the processor takes or leaves the branch.

3.3.3. Reset and Debug Signals

Table 23. Reset and Debug Signals

Interface	Type	Description
`reset`	Reset	A global hardware reset input signal that forces the Nios V processor to reset immediately.
`dbg_reset_out`	Reset	An optional reset output signal which appear after you enable both **Enable Debug** and **Enable Reset from Debug Module** parameters. • This reset output signal is triggered by the JTAG debugger or `niosv-download -r` command. • You can connect this reset output signal to the following input signals: — To the `ndm_reset_in` input to reset the core and the timer module. — To the reset input signal of other components as needed.
`ndm_reset_in`	Reset	An optional reset input signal which appear after you enable both **Enable Debug** and **Enable Reset from Debug Module** parameters. • You can use this signal to trigger reset controller to reset the core and the timer module. • The reset controller synchronizes the reset (hard reset) and `ndm_reset_in` signals.
`cpu_resetreq`	Conduit	An optional local reset ports which appear after you enable **Add Reset Request Interface** parameter. The signal consists of an input `resetreq` signal and an output `ack` signal that trigger the Nios V processor to reset without affecting other components in a Nios V processor system. • You can request a reset to the Nios V processor core by asserting the `resetreq` signal. • The `resetreq` signal must remain asserted until the processor asserts `ack` signal. Failure for the signal to remain asserted can cause the processor to be in a non-deterministic state. • Assertion of the `resetreq` signal in debug mode has no effect on the processor's state. • The Nios V processor responds that the reset is successful by asserting the `ack` signal. • After the processor is successfully reset, the assertion of the ack signal can happen multiple times periodically until the de-assertion of the `resetreq` signal.

Figure 5. Nios V/m Processor Reset Network



3.3.4. Control and Status Registers

Nios V/m processor's Control and Status Registers (CSR) is both readable and writable. Nios V/m updates the CSR during the E-stage of the pipeline.

During the execution of a Nios V processor application, you may observe the following behaviors:

- CSR write instruction (in E-stage) is stalled due to the pending memory or multicycle instructions (in M-stage).
- CSR write instruction (in E-stage) continues after the pending instructions (in Mstage) are complete.
- If the processor generates an exception during the M-stage, the processor flushes the pending instructions in the pipeline (including the CSR write instruction in the E-stage) and initiates the trap handler to service the exception.

3.3.5. Trap Controller

In the Nios V processor, trap refers to the transfer of control to a trap handler caused by either an exception or an interrupt.

- Exceptions are synchronous events that originate inside the processor. They are commonly caused by an unusual condition occurring at run time associated with an instruction.
- Interrupts are asynchronous events that originate outside of the processor. They are commonly caused by service requests from system peripherals.

3.3.5.1. Exception Handling

The Nios V/m processor architecture provides a simple exception controller to handle all exception types. Each exception, including internal hardware interrupts, causes the processor to transfer execution to an exception address. An exception handler at this address determines the cause of the exception and executes an appropriate exception routine.

You can set the exception address in the **Nios V Processor Board Support Package Editor ➤ BSP Linker Script**. Nios V/m processor stores the address in machine trap vector (`mtvec`) CSR register.

All exceptions are precise. The processor completes all instructions that precede the faulting instruction and does not start the execution of instructions that follow after the faulting instruction.

Table 24. Exceptions

Exception	Description
Instruction Address Misaligned	The core pipeline logic in F-stage detects the exception. This exception is flagged if the core fetched a program counter that is not aligned to a 32-bit word boundary.
Instruction Access Fault	The instruction read response signal detects this exception.
Illegal Instruction	The instruction decoder in the D-stage flags this exception if an instruction word contains encoding for an unimplemented or undefined instruction. The control logic for the CSR read and write flags this exception in the E-stage if a CSR instruction accesses a CSR that is not implemented or undefined.
Breakpoint	The instruction decoder flags the software breakpoint exception EBREAK in the D-stage.
Load Address Misaligned	The core for the load/store unit in the M-stage detects the misalignment. This exception is flagged if the data address is not aligned to the size of the data access.
Store Address Misaligned	
Load Access Fault	The core for the data read and write response signal detects the exception.
Store Access Fault	
Env call from M-mode	The instruction decoder in the D-stage detects the instruction.

3.3.5.2. Interrupt Handling

The Nios V/m processor implementation supports the following interrupts:

- Platform interrupts with 16 level-sensitive interrupt request (IRQ) inputs.
- Internally-generated Timer and Software interrupt. You can access the timer interrupt register using the Timer and Software interrupt module interface by connecting to the data bus.

During an interrupt, the core writes the program counter of the attached instruction into the machine exception program counter (`mepc`) register. An interrupt is usually attached to the instruction in E-stage or in the preceding F-stage or D-stage pipeline. The core is not capable of retracting a memory instruction in the M-stage. If an instruction in M-stage flags an exception while an interrupt is pending and ready to be serviced, the core fetches and executes the exception instruction. If a memory or multicycle instruction is pending in the M-stage, for example, the core is waiting for the response, the core does not flag an interrupt until it receives a response for that instruction. Pending interrupts are flagged by their corresponding bits in Machine Interrupt-Pending (`mip`) register.

An interrupt is taken only when Machine Status Register (`mstatus`) bit 3 is asserted and bits corresponding to its pending interrupt in Machine Interrupt-pending (`mip`) register is asserted.

Table 25. Interrupt Control and Status Registers/Bits

Register	Status Registers/Bits	Description
`mstatus`	`mstatus[3]`/Machine Interrupt-Enable (MIE) field	Global interrupt-enable bit for machine mode
`mie`	`mie[31:16]`/Platform interrupt-enable field	Platform interrupt-enable bit for 16 hardware interrupts
	`mie[7]`/Machine Timer Interrupt-Enable (MTIE) field	Timer interrupt-enable bit for machine mode
	`mie[3]`/Machine Software Interrupt-enable (MSIE) field	Software interrupt-enable bit for machine mode
`mip`	`mip[31:16]`/Platform interrupt-pending field	Platform interrupt-pending bit for 16 hardware interrupts
	`mip[7]`/Machine Timer Interrupt-Pending (MTIP) field	Timer interrupt-pending bit for machine mode
	`mip[3]`/Machine Software Interrupt-Pending (MSIP) field	Software interrupt-pending bit for machine mode

3.3.5.2.1. Timer and Software Interrupt Module

The timer and software interrupt hosts the following registers:

- Machine Time (`mtime`) and Machine Time Compare (`mtimecmp`) registers for timer interrupt.
- Machine Software Interrupt-pending (`msip`) field for the software interrupt.

The value of `mtime` increments after every clock cycle. When the value of `mtime` is greater or equal to the value of `mtimecmp`, the timer posts the interrupt.

3.3.6. Memory and I/O Organization

You can configure the Nios V/m processor systems. Consequently, the memory and I/O organization varies from system to system. A Nios V/m processor core uses one or more of the following ports to provide access to memory and I/O:

- Instruction manager port: An Arm Advanced Microcontroller Bus Architecture (AMBA) AXI4-Lite Memory-Mapped manager port that connects to instruction memory via system interconnect fabric.
- Data manager port: An AMBA AXI4-Lite Memory-Mapped manager port that connects to data memory and peripherals via the system interconnect fabric.

Nios V/m Processor Core Memory Mapped I/O Access: Both data memory and peripherals are mapped into the address space of the data manager port. Nios V/m processor core uses little-endian byte ordering. Words and half-words are stored in memory with the more-significant bytes at higher addresses. The Nios V/m processor core does not specify anything about the existence of memory and peripherals. The quantity, type, and connection of memory and peripherals are system dependent.

3.3.6.1. Instruction and Data Buses

3.3.6.1.1. Instruction Manager Port

Nios V/m processor instruction bus is implemented as a 32-bit AMBA 4 AXI-Lite manager port.

The instruction manager port:

- Performs a single function: it fetches instructions to be executed by the processor.
- Does not perform any write operations.
- Can issue successive read requests before data return from prior requests.
- Can prefetch sequential instructions.
- Always retrieves 32-bit of data. Every instruction fetch returns a full instruction word, regardless of the width of the target memory. The widths of memory in the Nios V/m processor system is not applicable to the programs. Instruction address is always aligned to a 32-bit word boundary.
- Does not require any burst adapter because it is non-bursting.

Table 26. Instruction Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	awaddr	Unused	[31:0]	Output
	awprot	Unused	[2:0]	Output
	awvalid	Unused	1	Output
	awready	Unused	1	Input
Write Data Channel	wdata	Unused	[31:0]	Output
	wstrb	Unused	[3:0]	Output
	wvalid	Unused	1	Output
	wready	Unused	1	Input
Write Response Channel	bresp	Unused	[1:0]	Input
	bvalid	Unused	1	Input
	bready	Unused	1	Output
Read Address Channel	araddr	Instruction Address (Program Counter)	[31:0]	Output
	arprot	Unused	[2:0]	Output
	arvalid	Instruction address valid	1	Output
	arready	Instruction address ready (from memory)	1	Input
Read Data Channel	rdata	Instruction	[31:0]	Input
	rresp	Instruction response: Non-zero value denotes instruction access fault exception	[1:0]	Input
	rvalid	Instruction valid	1	Input
	rready	Constant 1	1	Output

3.3.6.1.2. Data Manager Port

The Nios V/m processor data bus is implemented as a 32-bit AMBA 4 AXI-Lite manager port. The data manager port:

- Performs read data from memory or a peripheral when the processor executes a load instruction.
- Performs write data to memory or a peripheral when the processor executes a store instruction.
- Does not require any burst adapter because it is non-bursting.

`axsize` signal value indicates the load/store instruction size- byte (LB/SB), halfword (LH/SH) or word (LW/SW). Address on `axaddr` signal is always aligned to size of the transfer. For store instructions, respective writes strobe bits are asserted to indicate bytes being written.

Nios V/m processor core does not support speculative issue of load/store instruction. Hence, a core can issue only one load or store instruction and waits until the issued instruction is complete.

Table 27. Data Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	`awaddr`	Store address	[31:0]	Output
	`awprot`	Unused	[2:0]	Output
	`awvalid`	Store address valid	1	Output
	`awready`	Store address ready (from memory)	1	Input
Write Data Channel	`wdata`	Store data	[31:0]	Output
	`wstrb`	Byte position in word	[3:0]	Output
	`wvalid`	Store data valid	1	Output
	`wready`	Store data ready (from memory)	1	Input
Write Response Channel	`bresp`	Store response: Non-zero value denotes store access fault exception.	[1:0]	Input
	`bvalid`	Store response valid	1	Input
	`bready`	Constant 1	1	Output
Read Address Channel	`araddr`	Load address	[31:0]	Output
	`arprot`	Unused	[2:0]	Output
	`arvalid`	Load address valid	1	Output
	`arready`	Load address ready (from subordinates)	1	Input
Read Data Channel	`rdata`	Load data	[31:0]	Input
	`rresp`	Load response: Non-zero value denotes load access fault	[1:0]	Input

continued...

Interface	Signal	Role	Width	Direction
		exception		
	rvalid	Load data valid	1	Input
	rready	Constant 1	1	Output

3.3.6.1.3. Choosing a Suitable Interface

The Nios V/m processor core offers an Avalon memory-mapped and Advanced eXtensible Interface (AXI) interface for both the instruction manager port and the data manager port. You can choose one of the interfaces in Platform Designer based on your design.

For example, if most IPs in Platform Designer use Avalon memory-mapped interface, you can choose the Avalon memory-mapped interface in the Nios V/m processor core as the bridge interface. Similarly, if most IPs in Platform Designer use AXI Interface, you can choose the AXI Interface in the Nios V/m processor core as the bridge interface.

The benefits of using the appropriate interface are as follows:

- Reduces the amount of bridging logic
- Reduces overhead bandwidth
- Provides better performance

3.3.6.2. Address Map

The address map for memories and peripherals in a Nios V/m processor system is design dependent. The following addresses are part of the processor:

1. Reset Address
2. Debug Exception Address
3. Exception Address
4. Timer and Software Interrupt Address

You can specify the Reset Address and Debug Exception Address in Platform Designer during system configuration. You can modify the Exception Address stored in the `mvtec` register. `mvtime` and `mtimecmp` register controls the timer interrupt. The `msip` register bit controls the software interrupt.

3.3.7. RISC-V based Debug Module

The Nios V/m processor architecture supports a RISC-V based debug module that provides on-chip emulation features to control the processor remotely from a host PC. PC-based software debugging tools communicate with the debug module and provide facilities, such as the following features:

- Reset Nios V processor core and timer module
- Download programs to memory
- Start and stop execution
- Set software breakpoints and watchpoints
- Analyze registers and memory

3.3.7.1. Debug Mode

You can enter the Debug Mode, as specified in the RISC-V architecture specification, in the following ways:

1. Halt from Debug Module
2. Software breakpoints
3. Trigger

Upon entering Debug Mode, Nios V processor completes the instruction in W-stage. By the order of priority, instruction in M-stage, E-stage, D-stage or F-stage takes the interrupt.

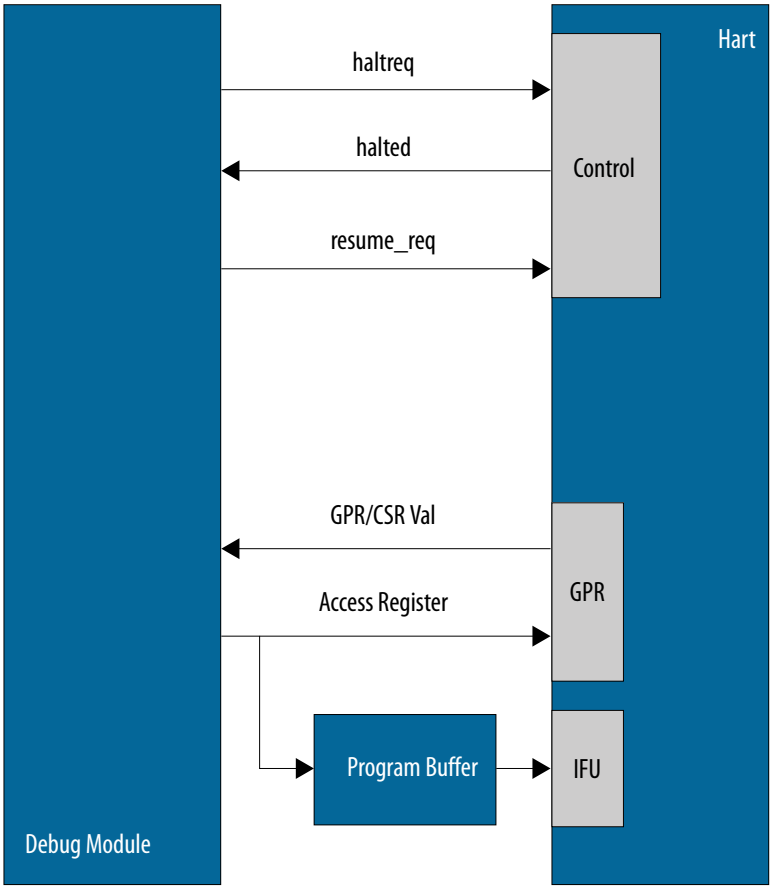
- When there is a valid instruction, Program Counter writes to the Debug Program Counter, `.dpc`.
- If the instruction in M-stage is not valid, then instruction in E-stage takes the interrupt and so on and so forth.
- If there is no valid instruction in the pipeline, the Program Counter for the next instruction writes to the Debug Program Counter, `.dpc`.

Note: For branches, the next Program Counter depends on whether a branch was taken or not taken, and whether the branch prediction (if any) was correct.

Debug Module selects Hardware Thread (Hart); which can be in one of the following states:

1. Non-existent: Debug Module probes a hart which does not exist.
2. Unavailable: Reset or temporary shutdown.
3. Running: Normal operation outside of debug.
4. Halted: Hart is said to be halted when it is in debug mode.

Figure 6. Debug Module Block Diagram



3.3.7.2. Halt from Debug Module

The debugger can write the `haltreq` bit in the Debug Module Control (`dmcontrol`) register, which places the Nios V processor in debug mode. The assertion of `haltreq` sends an asynchronous interrupt to the processor core logic.

3.3.7.3. Trigger

The Nios V processor core supports one address or data match trigger. The trigger registers are accessible using RISC-V `csr` opcodes or abstract debug commands. The firing of trigger can either enter the Debug Mode or raise a breakpoint exception, which depends on the trigger registers.

Table 28. Trigger Registers Implemented in Nios V Processor

Name	Registers	Description
`tselect`	Trigger Select	Nios V processor supports one trigger, therefore the processor selects Trigger 0 at default (value set at 0).
`tdata1`	Trigger Data 1	The value of `type` field is 2 to represent Trigger 0 as an address or data match trigger. Write behavior to `tdata` registers depends on the `dmode` field. The remaining bits acts as `mcontrol`.

continued...

Name	Registers	Description
mcontrol	Match Control	Controls address and data trigger implementation according to action, m, execute, store and load fields. The Nios V processor does not implement other fields.
tdata2	Trigger Data 2	Holds trigger-specific data (virtual address, instruction opcode, data stored or loaded).
tinfo	Trigger Info	The value of info field is 4 at default to signify tdata1.type is 2. This implies selected trigger is an address or data match trigger.

Based on the bit field setting in mcontrol register, Nios V processor can implement different trigger types after the selected trigger matches the tdata2 register.

Table 29. Nios V Processor Triggers Condition and Firing Time

Triggers	Condition	Firing Time	Exception Program Counter
Instruction Address Trigger	Program Counter matches tdata2	Before executing the instruction	The processor sets the Machine Exception Program Counter (mepc) to the instruction address (PC).
Instruction Opcode Trigger	Instruction opcode matches tdata2	Before executing the instruction	
Store Address Trigger	Store address matches tdata2	After executing the store instruction	The processor sets the mepc to the next instruction address (PC + 4).
Store Data Trigger	Store data matches tdata2	After executing the store instruction	
Load Address Trigger	Load address matches tdata2	After executing the load instruction	
Load Data Trigger	Load data matches tdata2	After executing the load instruction	

Table 30. Address or Data Match Trigger Type

Trigger	mcontrol bit fields			
	select	execute	store	load
Instruction Address	0	1	0	0
Instruction Opcode	1	1	0	0
Store Address	0	0	1	0
Store Data	1	0	1	0
Load Address	0	0	0	1
Load Data	1	0	0	1

Note: The trigger is disabled in the following situations:

- The Nios V processor is in debug mode.
- The Nios V processor is in machine mode, while mcontrol.m or mcontrol.type is 0.

3.3.7.4. Abstract Commands in Debug Mode

Nios V/m processor implements Access Register abstract command. The Access Register command allows read-write access to the processor registers including GPRs, CSRs, FP registers and Program Counter. The Access Register also allows program

execution from program buffer. The debugger executes Access Register commands by writing into the Abstract Command (`command`) register using the Access Register command encoding.

Table 31. Access Register Command Encoding

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cmdtype								0	aarsize			aarpostincrement	postexec	transfer	write
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
regno															

Table 32. Fields Descriptions

Field	Role
`cmdtype`	Determine command type 0 : Indicates Access Register command.
`aarsize`	Specifies size of register access 2: Access the lowest 32 bit of register 3: Access the lowest 64 bit of register 4: Access the lowest 128 bit of register
`aarpostincrement`	0: No effect 1: regno is incremented after successful register access
`postexec`	0: No effect 1: Execute program in program buffer
`transfer`	Acts in conjunction with write field. 0: Ignore value in write field 1: Execute operation specified by write field.
`write`	0: Copy data from register 1: Copy data to register
`regno`	Register address to be accessed.

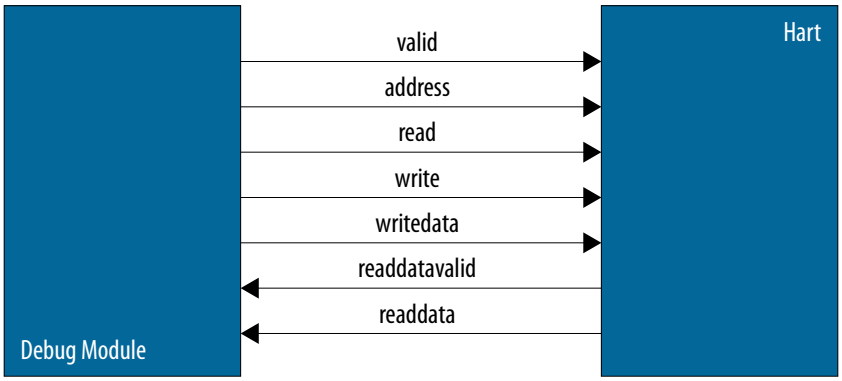
Note: The Nios V/m processor does not support abstract commands when hardware thread is not halted.

Table 33. Software Response to Unimplemented Commands

Field	State	
`cmderr[10:8]`	0	No error
	1	Busy
	2	Command not supported
	3	Exception - from program buffer instruction
	4	Command not executed because hart unavailable, or not in correct state to execute command.
	5	Abstract command failed due to bus error
	6	RSVD
	7	Command failed for other reasons.

Debug Module has the Abstract Control and Status CSR which includes the `cmderr` field. The `cmderr` field represents the current state of abstract command being executed. If the `cmderr` field is non-zero, writes to the `command` register are ignored. To clear the `cmderr` field, write 1 for every bit in the field.

Figure 7. Debug Module Interface Signals



Avalon memory-mapped interface implement the Register Access using request/response bus with Debug Module being the initiator and core being the responder. Address bus carries the register ID.

3.3.7.5. Hardware/Software Interface

The Nios V processor debug module is based on the RISC-V Debug specification. It supports the same Debug Module registers documented in the specification. Each register has a fixed address as specified in the RISC-V Debug Support specification. Debugger can determine the register implementation status by writing or reading from the Debug Module registers. Unimplemented registers return 0 when read.

Debugger can check the status of the system by reading Debug Module Status (`dmstatus`) register. Debug Module Status register is read-only and provides status of the Debug Module and the selected harts.

You can access a specific hart by writing to the `hartsel` field in `dmcontrol`. Other fields in `dmcontrol` specify the action a debugger can take.

Halt Summary 0 (`haltsum0`) register reflects the status of a hart (halted/not halted). The LSB of this register can reflect whether hart is halted or not. Other bits is always 0. This is a read-only register of the debugger.

Related Information

RISC-V® Debug Release
More information about the conceptual view of the states, refer to Section : Overview of States, Figure: Run/Halt Debug State Machine for single hart systems.

3.3.8. Error Correction Code (ECC)

The Nios V/m processor core has the option to enable error detection and ECC status reporting for the RAM block, that is the Register file. Each RAM block has its own source ID. When an ECC event occurs, the processor transmits the source ID and ECC status to the ECC interface.

- If the ECC event is a correctable error, the processor continues to operate after correcting the error. The correction made is not written back to its memory source.
- If the ECC event is an un-correctable error, the processor halts its current progress and stalls. You need to reset either the processor core alone or the entire system.

Note: To reset the processor core alone, you need to apply the **Reset Request Interface** to safely reset the Nios V processor (cleared of any outstanding operations). To reset the entire system, you can use the hard reset interface instead.

The ECC interface allows external logic to monitor ECC errors from the Nios V/m processor. The interface is a conduit, made up of the following output signals.

- `cpu_ecc_status` : Indicates the error status
- `cpu_ecc_source` : Indicates the error source.

Table 34. `cpu_ecc_status`

2-bits Encoding	Description	Effects on Software
2'b00	No ECC event	None
2'b01	Reserved	Not Applicable
2'b10	Correctable single bit ECC error	None
2'b11	Un-correctable ECC error	Likely fatal and halts the processor

Table 35. `cpu_ecc_source`

4-bits Encoding	ECC Source	Available
4'b0000	No ECC event	Always
4'b0001	General Purpose Register (GPR)	Always
4'b0010 ~ 4'b1110	Other RAM Blocks	Not Available
4'b1111	Reserved	Not Applicable

Note: Due to a limitation with embedded memory blocks, the simulation model of Nios V processor does not support ECC on Arria 10 devices.

Related Information

Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide
For more information about ECC on Arria 10 devices.

3.4. Programming Model

3.4.1. Privilege Levels

The privilege levels in Nios V/m processor are designed based on the RISC-V architecture specification. The supported privilege levels available are Machine Mode (M-mode) and Debug Mode (D-mode).

Related Information

RISC-V® Specifications

3.4.1.1. Machine Mode (M-mode)

The default operating mode is Machine mode (M-mode). The core boots up into M-mode after power-on reset. M-mode provides low-level access to the machine implementation and all CSRs.

3.4.1.2. Debug Mode (D-mode)

The Debug mode (D-mode) is an additional privilege level to support off-chip debugging and manufacturing test.

The core can enter D-mode using any of the following methods:

- After reset when bit is set in debug CSR.
- Using debug interrupt.
- Using EBREAK instruction when bit is set in debug CSR.
- Using single step.

3.4.2. Control and Status Registers (CSR) Mapping

Control and status registers report the status and change the behavior of the processor. Since the processor core only supports M-mode and D-mode, Nios V/m processor implements the CSRs supported by these two modes.

Table 36. Control and Status Registers List

Number	Privilege	Name	Description
Machine Trap Setup			
0x300	MRW	`mstatus`	Machine status register. Refer to Machine Status Register Fields table.
0x301	MRW	`misa`	ISA and extensions. Refer to Machine ISA Register Fields table.
0x304	MRW	`mie`	Machine interrupt-enable register. Refer to Machine Interrupt-Enable Register Fields table.
0x305	MRW	`mtvec`	Machine trap-handler base address. Refer to Machine Trap-Handler Base Address Register Fields table.
Machine Trap Handling			
0x341	MRW	`mepc`	Machine exception program counter. Refer to Machine Exception Program Counter Register Fields table.
0x342	MRW	`mcause`	Machine trap cause. Refer to Machine Trap Cause Register Fields table.
0x343	MRW	`mtval`	Machine bad address or instruction. Refer to Machine Trap Value Register Fields table.
0x344	MRW	`mip`	Machine interrupt pending. Refer to Machine Interrupt-Pending Register Fields table.
Trigger Registers			

continued...

Number	Privilege	Name	Description
0x7A0	MRW	tselect	Trigger select. Refer to Trigger Select Register Fields table.
0x7A1	MRW	tdata1 (mcontrol)	Trigger data 1 (Match Control). Refer to Trigger Data 1 (Match Control) Register Fields table.
0x7A2	MRW	tdata2	Trigger data 2. Refer to Trigger Data 2 Register Fields table.
0x7A4	MRO	tinfo	Trigger info. Refer to Trigger Info Register Fields table.
Debug Mode Registers			
0x7B0	DRW	dcsr	Debug control and status register. Refer to Debug Control and Status Register Fields table.
0x7B1	DRW	dpc	Debug Program Counter. Refer to Debug Program Counter Register Fields table.
Machine Information Register			
0xF11	MRO	mvendorid	Vendor ID. Refer to Vendor ID Register Fields table.
0xF12	MRO	marchid	Architecture ID. Refer to Architecture ID Register Fields table.
0xF13	MRO	mimpid	Implementation ID. Refer to Implementation ID Register Fields table.
0xF14	MRO	mhartid	Hardware thread ID. Refer to Hardware Thread ID Register Fields table.

Related Information

The RISC-V® Instruction Set Manual Volume II: Privileged Architecture
More information about M-mode CSR and their respective fields.

3.4.2.1. Control and Status Register Field

The value in the each CSR registers determines the state of the Nios V/m processor. The field descriptions are based on the RISC-V specification.

Table 37. Machine Status Register Fields

The mstatus register is a 32-bit read-write register that keeps track of and controls the hart's current operating state.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD	0								0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0		MPP[1:0]		0		0	MPIE	0	0	0	MIE	0	0	0

The following bitfields are read-only 0 because:

- Bit 22 (TSR = 0): S-mode is not supported
- Bit 21 (TW = 0): There are no modes less privileged than M-mode.
- Bit 20 (TVM = 0): S-mode is not supported
- Bit 19 (MXR = 0): S-mode is not supported
- Bit 18 (SUM = 0): S-mode and U-mode are not supported

- Bit 17 (MPRV = 0): U-mode is not supported
- Bit 16 and Bit 15 (XS[1:0] = 0): S-mode is not supported
- Bit 14 and Bit 13 (FS[1:0] = 0): "F" extension for Single-Precision Floating-Point is not supported
- Bit 10 and Bit 9 (VS[1:0] = 0): S-mode is not supported
- Bit 8 (SPP = 0): S-mode is not supported
- Bit 6 (UBE = 0): U-mode is not supported
- Bit 5 (SPIE = 0): S-mode is not supported
- Bit 1 (SIE = 0): S-mode is not supported

Table 38. Machine ISA Register Fields

The `misa` CSR is a read-write register reporting the ISA supported by the hart.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXL[1:0]		0				Extension[25:0]									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Extension[25:0]															

Table 39. Machine Interrupt-Enable Register Fields

The `mie` register is a 32-bit read-write register that contains interrupt enable bits. This core does not support supervisor or user modes. Thus, any interrupt related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Platform Interrupt[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MEIE	0	0	0	MTIE	0	0	0	MSIE	0	0	0

Table 40. Machine Trap-Handler Base Address Register Fields

The `mtvec` register is a 32-bit read/write register that holds trap vector configuration, consisting of a vector base address (BASE) and a vector mode (MODE).

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Base[31:2]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base[31:2]														Mode	

Table 41. Machine Exception Program Counter Register Fields

The `mepc` register is a 32-bit read-write register that holds the addresses of the instruction that was interrupted or that encountered the exception when a trap is taken into M-mode.

Bit Field													
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
mepc														

Table 42. Machine Trap Cause Register Fields

The `mcause` register is a 32-bit read-write register that hold the code indicating the event that caused the trap when a trap is taken into M-mode.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt	Exception code [30:16]														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Exception code [15:0]															

Table 43. Machine Trap Value Register Fields

The `mtval` register is a 32-bit read-write register that is written with exception-specific information to assist software in handling the trap when a trap is taken into M-mode.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
mtval														

Table 44. Machine Interrupt-Pending Register Fields

The `mip` register is a 32-bit read/write register containing information on pending interrupts. This core does not support supervisor or user modes. Thus, any interrupt related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Platform Interrupt[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MEIP	0	0	0	MTIP	0	0	0	MSIP	0	0	0

Table 45. Trigger Select Register Fields

The `tselect` register is a 32-bit read/write register that selects the current trigger is accessible by other trigger register.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
`tselect`														

Table 46. Trigger Data 1 (Match Control) Register Fields

The `tdata1 (mcontrol)` register is a 32-bit read/write register containing information on the trigger type, tdata registers accessibility, and trigger implementation. This core does not support supervisor or user modes. Thus, any bitfield related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
type=2				dmode	maskmax						hit	select	timing	sizelo	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
action				chain	match				m	0	0	0	execute	store	load

Table 47. Trigger Data 2 Register Fields

The `tdata2` register is a 32-bit read/write register containing the trigger-specific data.

Bit Field														
31	30	29	28	27	26	25	…	6	5	4	3	2	1	0
tdata2														

Table 48. Trigger Info Register Fields

The `tinfo` register is a 32-bit read-only register containing information on each possible tdata1.type.

Bit Field													
31	30	29	…	18	17	16	15	14	13	…	2	1	0
0							info						

Table 49. Debug Control and Status Register Fields

The `dcsr` CSR is a 32-bit read-write register containing information and status during D-mode. This core does not support supervisor or user modes. Thus, any bitfield related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
debugver				0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ebreakm	0	0	0	stepie	stopcount	stoptime	cause			0	mprven	nmip	step	prv	

Table 50. Debug Program Counter Register Fields

Upon entry to D-mode, `dpc` CSR is updated with the virtual address of the next instruction to be executed.

Bit Field														
31	30	29	28	27	26	25	…	6	5	4	3	2	1	0
dpc														

Table 51. Vendor ID Register Fields

The `mvendorid` CSR is a 32-bit read-only register that provides the JEDEC manufacturer ID of the provider of the core.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Bank															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bank									Offset						

Table 52. Architecture ID Register Fields

The `marchid` CSR is a 32-bit read-only register encoding the base microarchitecture of the hart.

Bit Field														
31	30	29	28	27	26	25	…	6	5	4	3	2	1	0
Architecture ID														

Table 53. Implementation ID Register Fields

The `mimpid` CSR provides a unique encoding of the version of the processor implementation.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
Implementation														

Table 54. Hardware Thread ID Register Fields

The `mhartid` CSR is a 32-bit read-only register that contains the integer ID of the hardware thread running the code

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
Hart ID														

3.5. Core Implementation

3.5.1. Instruction Set Reference

The Nios V/m processor is based on the RV32IZicsr specification. RV32IZicsr includes the following instructions sets:

- RV32I base integer instruction set
- Control and status register instructions ("Zicsr" extension).

There are 6 types of instruction formats. They are R-type, I-type, S-type, B-type, U-type, and J-type.

Table 55. Instruction Formats (R-type)

Bit Field (R-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
funct7							rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			rd					opcode						

Table 56. Instruction Formats (I-type)

Bit Field (I-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[11:0]												rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			rd					opcode						

Table 57. Instruction Formats (S-type)

Bit Field (S-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[11:5]							rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			imm[4:0]					opcode						

Table 58. Instruction Formats (B-type)

Bit Field (B-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[12]	imm[10:5]						rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			imm[4:1]			imm[11]		opcode						

Table 59. Instruction Formats (U-type)

Bit Field (U-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[15:12]				rd					opcode						

Table 60. Instruction Formats (J-type)

Bit Field (J-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[20]	imm[10:1]										imm[11]	imm[19:16]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[15:12]				rd					opcode						

Related Information

The RISC-V® Instruction Set Manual Volume I: Unprivileged ISA
More information about the the `fence.i` instruction in RV32I Base Integer.

4. Nios V/g Processor

The Nios V/g processor is a general-purpose CPU core developed by Intel based on the RISC-V RV32IMZicsr_Zicbom instruction set (optionally with "F" extension) and supports the functional units described in this document.

Related Information

Overview on page 4

4.1. Processor Performance Benchmarks

Table 61. Nios V/g Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software

Quartus Prime Edition	FPGA Used	f_{MAX} (MHz)	Logic Size	Architecture Performance	
				DMIPS/MHz Ratio	CoreMark/MHz Ratio
Quartus Prime Pro Edition	Cyclone 10	202	1986 ALM	1.448	2.323
	Arria 10	214	1972 ALM		
	Stratix 10	224	2166 ALM		
	Agilex 7	270	2153 ALM		
	Agilex 5	241	2203 ALM		
Quartus Prime Standard Edition	Cyclone IV E	81	4316 LE	0.942	1.49
	Cyclone V	117	1886 ALM		
	Arria V	121	1917 ALM		
	Arria V GZ	211	1859 ALM		
	Stratix V	231	1853 ALM		
	Cyclone 10 LP	93	4174 LE		
	Arria 10	239	1818 ALM		
	MAX 10	91	4199 LE		

Table 62. Benchmark Parameters for Quartus Prime Software

Parameter		Settings/Description	
		Quartus Prime Pro Edition	Quartus Prime Standard Edition
Quartus Prime seed		Maximum performance result are based on 10 seed sweep from Quartus Prime Pro Edition software version 24.3.	Maximum performance result are based on 10 seed sweep from Quartus Prime Standard Edition software version 23.1.
Device speed grade		Fastest speed grade from each Intel FPGA device family.	
Defined peripherals		• Nios V/g processor core • Without debug module, internal timer, and floating point unit • With 4 KB instruction cache, 4 KB data cache, and branch prediction. • 128 KB on-chip memory for the instruction and data bus. • JTAG UART Intel FPGA IP. • Interval Timer Core.	
Toolchain	Version	• riscv32-unknown-elf-gcc (GCC) version 13.2.0 • CMake Version: 3.29.3	• riscv32-unknown-elf-gcc (GCC) version 12.1.0 • CMake Version: 3.27.1
	Compiler configuration	• Compiler flags: `-03` • Assembler options: `-Wa -gdwarf2` • Compile options: `-Wall -Wformat-security -march=rv32im_zicbom -mabi=ilp32`	

Intel uses the same Quartus Prime design example for maximum performance benchmark(fMAX) and logic size benchmarks. However, the compiler settings are different for each benchmarks:

- **Superior Performance with Maximum Placement Effort** in Quartus Prime Pro Edition software.
- **High Performance Effort** in Quartus Prime Standard Edition software.
- Logic size benchmark: `area_aggressive`

Note: Results may vary depending on the version of the Quartus Prime software, the version of the Nios V processor, compiler version, target device and the configuration of the processor. Additionally, any changes to the system logic design might change the performance and LE usage. All results are generated from design built with Platform Designer.

4.2. Processor Pipeline

The Nios V/g processor employs a five-stage pipeline.

Table 63. Processor Pipeline Stages

Stage	Denotation	Function
F	Instruction fetch	• PC+4 calculation • Next instruction fetch • Pre-decode for register file read
D	Instruction decode	• Decode the instruction • Register file read data available • Hazard resolution and data forwarding

continued...

Stage	Denotation	Function
E	Instruction execute	• ALU operations • Memory address calculation • Branch resolution • CSR read/write
M	Memory	• Memory and multicycle operations • Register file write • Branch redirection
W	Write back	• Facilitates data dependency resolution by providing general-purpose register value.

The Nios V/g processor implements the general-purpose register file using the M20K memory blocks. The processor takes one processing cycle to read from an M20K location. Therefore, the F-stage initiates register file reads so general-purpose register values are available in D-stage.

Writing to the M20K location takes two processing cycles. Therefore, the M-stage initiates writes to a general-purpose register. If there is a dependency to resolve, the M-stage carries forward the value to the W-stage.

The core resolves data dependencies in the D-stage. Operands can move from register file read or E-stage, M-stage, or W-stage.

Reasons for the pipeline stalling:

- Data dependency—if the source operand is not available in D-stage, instruction in D-stage and F-stage stalls until the operand becomes available. The scenario can happen if destination general-purpose register of load or multicycle instruction in E-stage or M-stage is the source for instruction in D-stage.
- Resource stall—if a memory operation or multicycle is pending in M-stage, the instructions in preceding stages stalls until M-stage completes the instruction.

4.3. Processor Architecture

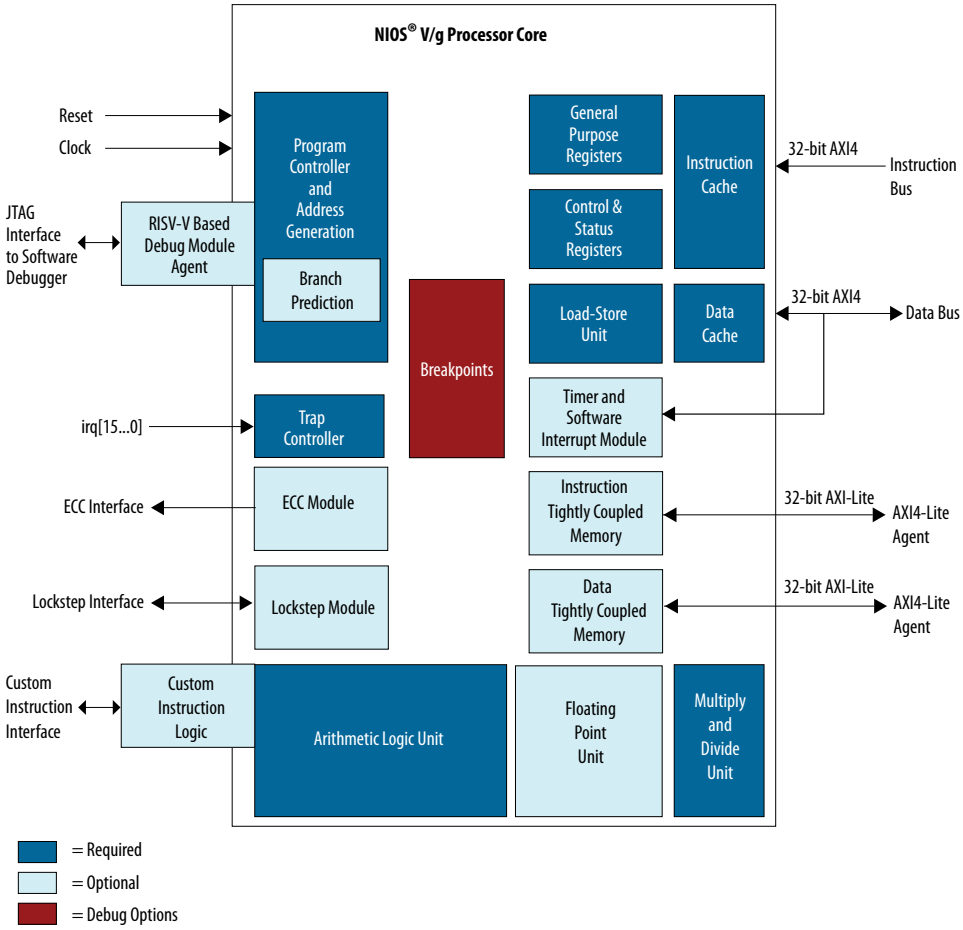
The Nios V/g processor architecture describes an instruction-set architecture (ISA). The ISA in turn necessitates a set of functional units that implement the instructions.

The Nios V/g processor architecture defines the following functional units:

- General-purpose register file
- Arithmetic logic unit (ALU)
- Multiply and divide units
- Floating point unit
- Custom instruction logic
- Control and status registers (CSR)
- Trap controller
- Instruction bus
- Data bus
- Instruction cache
- Data cache

- Tightly coupled memories
- RISC-V based debug module
- ECC module
- Branch prediction
- Lockstep module

Figure 8. Nios V/g Processor Core Block Diagram



4.3.1. General-Purpose Register File

Nios V/g processor implementation supports a flat register file. The register file contains thirty-two 32-bit general-purpose integer registers. Nios V/g processor implements the general-purpose register using M20K memories, which do not support two read ports. Hence, Nios V/g processor duplicates the register files so that two different source registers for an instruction are available in a single cycle. After performing ALU operations, the processor core writes the same result to the destination register in both memories.

4.3.2. Arithmetic Logic Unit

The arithmetic logic unit (ALU) operates on data stored in general-purpose registers. ALU operations take one or two inputs from registers and store the result back into the register.

Table 64. Fundamental Data Operations of the ALU

Category	Description
Arithmetic	Addition and subtraction on signed and unsigned operands.
Relational	Equal, not-equal, greater-than-or-equal, and less-than relational operations (==, != >=, <).
Logical	AND, OR, NOR, and XOR logical operations.
Shift	Logical and arithmetic shift operations.

For load and store instructions, the Nios V/g processor uses the ALU to calculate the memory address. For conditional control transfer instructions, Nios V/g processor uses the relational operations in the ALU to determine if the processor takes or leaves the branch.

4.3.3. Multipy and Divide Units

The multiply and divide units take two inputs from registers, compute, and store the result to the register.

4.3.4. Floating-Point Unit

The floating-point unit (FPU) implements the single precision floating point instructions. The FPU operates on data stored in thirty-two 32-bits floating-point registers, implemented using M20K memories.

Below are the characteristics of the FPU:

- Based on RISC-V "F" Standard Extension for Single-Precision Floating-Point
- Supports floating-point fused multiply-add instructions.
- IEEE 754-2008 compliant except for:
  - Simplified rounding
  - Subnormal supported on a subset of operations
- Consumes resource in a typical system as below(1):
  - 960 ALMs
  - Five M20Ks memories
  - Five DSP blocks

Note: The Nios V/g processor adopts the GNU floating point software emulation for double precision floating point operation.

(1) System using Arria 10 FPGA devices.

4.3.4.1. IEEE 754 Exception Conditions

The FPU offers different IEEE 754 exceptions support when targeting on different Intel FPGA devices. The `fcsr` register holds the accrued exception flags.

Table 65. IEEE 754 Exception Support

IEEE 754 Exception	Arria 10, Cyclone 10 GX	Other Intel FPGA devices
Invalid (NV)	√	√
Division by Zero (DZ)	√	√
Overflow (OF)	√	√
Underflow (UF)	Incomplete	√
Inexact (NX)	Incomplete	√

Related Information

Control and Status Register Field on page 75
Refer to the Floating-Point CSR Register Fields table for more information.

4.3.4.2. Floating Point Operations

The table below provides a detailed summary of the FPU operations.

Table 66. Floating Point Operation Summary

Category	Operation	Cycles[2]	Result	Subnormal	Rounding[3]	GCC Inference
Arithmetic	FDIV.S	14	a ÷ b	Flush-to-0	RNE	a / b
	FSUB.S	1	a - b	Flush-to-0	RNE	a - b
	FADD.S	1	a + b	Flush-to-0	RNE	a + b
	FMUL.S	2	a x b	Flush-to-0	RNE	a * b
	FSQRT.S	12	√a	Flush-to-0	Faithful [4]	sqrt(a)
	FMIN.S	2	(a < b) ? a : b	Supported	RNE	fminf()
	FMAX.S	2	(a < b) ? b : a	Supported	RNE	fmaxf()
Fused Arithmetic[5]	FMADD.S	3	(a x b) + c	Flush-to-0	RNE	(a * b) + c
	FMSUB.S	3	(a x b) – c	Flush-to-0	RNE	(a * b) - c
	FNMSUB.S	3	-(a x b) + c	Flush-to-0	RNE	-(a * b) + c
	FNMADD.S	3	-(a x b) - c	Flush-to-0	RNE	-(a * b) - c

continued...

(2) Preliminary results.

(3) Round-to-Nearest, ties to Even (RNE).

(4) Faithful rounding has a maximum error of 1 Unit of Least Precision (ULP) as compared to the 0.5 ULP in RNE. Faithful rounding is employed to save area and reduce the latency of FSQRT.S.

(5) GCC toolchain infers Fused Arithmetic when the optimization level is -O3 or higher.

Category	Operation	Cycles[2]	Result	Subnormal	Rounding[3]	GCC Inference
Conversion	FCVT.S.W / FCVT.S.WU	3	int_to_float(a)	Supported	None	Casting
	FCVT.W.S / FCVT.WU.S	3	float_to_int(a)	Supported	Round towards Zero	Casting
					Round to Nearest, ties to Max Magnitude	roundf(a)
Compare	FLT.S	1	(a < b) ? 1 : 0	Flush-to-0	RNE	a < b
	FLE.S	1	(a ≤ b) ? 1 : 0	Flush-to-0	RNE	a <= b
	FEQ.S	1	(a = b) ? 1 : 0	Flush-to-0	RNE	a == b
Sign Injection	FSGNJN.S (FNEG.S)	1	-a	Supported	RNE	-a
	FSGNJX.S (FABS.S)	1	|a|	Supported	RNE	fabsf(a)
Classification	FCLASS.S	2	Refer to topic *Floating Point Classification*.	Supported	None	fpclassify(a)

Note: Assume a, b, and c as single-precision floating point values. Nios V Processor Fused Arithmetic has a rounding stage between the multiplier and addition.

The following list describes the header in the table above:

- **Operation** —Provides the name of the floating-point operation. The names match the names of the corresponding RISC-V floating-point instructions.
- **Cycle** —Specifies the number of cycles it takes to execute the instruction.
- **Result**—Describes the computation performed by the operation.
- **Subnormal**—Describes how the operation treats subnormal inputs and subnormal outputs. Subnormals are numbers with a magnitude less than approximately 1.17549435082e-38.
- **Rounding** —Describes how the FPU rounds the result.
- **GCC Inference**—Shows the C code from which GCC infers the instruction operation.

Related Information

Floating Point Classification on page 49

4.3.4.2.1. Floating Point Classification

The FCLASS.S instruction classifies the floating-point value into ten possible classifications specified in the following table and returns an integer value.

(2) Preliminary results.

(3) Round-to-Nearest, ties to Even (RNE).

Table 67. Floating Point Classification

Classification	Description	Single Precision FP Representation			Return Integer Value
		Sign (1 bit)	Exponent (8 bit)	Mantissa (23 bit)	
-infinity	Negative infinity	1	0xFF	0	0
-normal	Negative normalized non-zero value	1	Any value (except values representing other classes)		1
-subnormal	Negative denormalized value	1	0	Non-zero	2
-0	Negative zero value	1	0	0	3
+0	Positive zero value	0	0	0	4
+subnormal	Positive denormalized value	0	0	Non-zero	5
+normal	Positive normalized non-zero value	0	Any value (except values representing other classes)		6
+infinity	Positive infinity	0	0xFF	0	7
Signaling NaN	Signaling NaN	Don't Care	0xFF	0x1 to 0x3FFFFF	8
Quiet NaN	Quiet NaN	Don't Care	0xFF	0x400000 to 0x7FFFFF	9

4.3.5. Custom Instruction

The Nios V/g processor architecture supports user-defined custom instructions. The Nios V/g ALU connects directly to custom instruction logic, enabling you to implement operations in hardware that are accessed and used in the same way as native instructions.

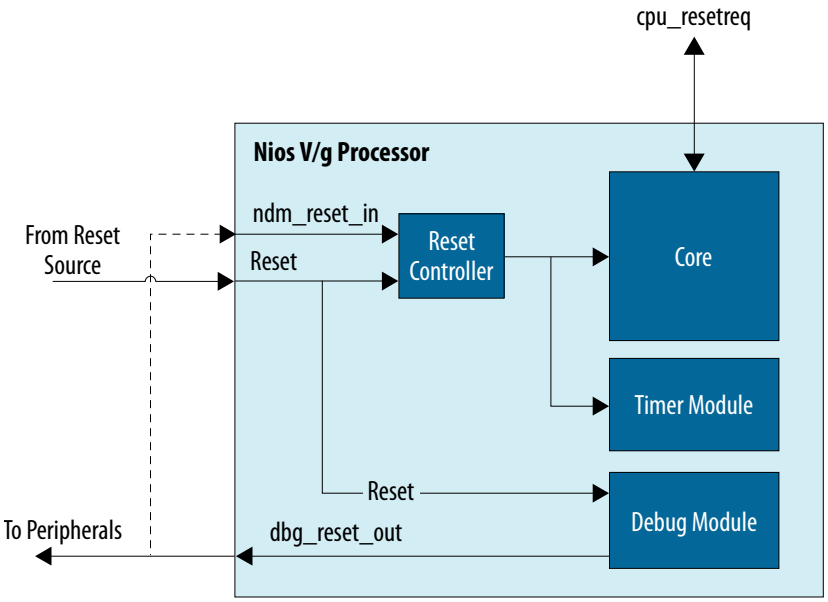
Related Information

AN 977: Nios V Processor Custom Instruction

4.3.6. Reset and Debug Signals

Interface	Type	Description
`reset`	Reset	A global hardware reset input signal that forces the Nios V processor to reset immediately.
`dbg_reset_out`	Reset	An optional reset output signal which appear after you enable both **Enable Debug** and **Enable Reset from Debug Module** parameters. • This reset output signal is triggered by the JTAG debugger or `niosv-download -r` command. • You can connect this reset output signal to the following input signals: — To the `ndm_reset_in` input to reset the core and the timer module. — To the reset input signal of other components as needed.
`ndm_reset_in`	Reset	An optional reset input signal which appear after you enable both **Enable Debug** and **Enable Reset from Debug Module** parameters. • You can use this signal to trigger reset controller to reset the core and the timer module. • The reset controller synchronizes the reset (hard reset) and `ndm_reset_in` signals.
`cpu_resetreq`	Conduit	An optional local reset ports which appear after you enable **Add Reset Request Interface** parameter. The signal consists of an input `resetreq` signal and an output `ack` signal that trigger the Nios V processor to reset without affecting other components in a Nios V processor system. • You can request a reset to the Nios V processor core by asserting the `resetreq` signal. • The `resetreq` signal must remain asserted until the processor asserts `ack` signal. Failure for the signal to remain asserted can cause the processor to be in a non-deterministic state. • Assertion of the `resetreq` signal in debug mode has no effect on the processor's state. • The Nios V processor responds that the reset is successful by asserting the `ack` signal. • After the processor is successfully reset, the assertion of the ack signal can happen multiple times periodically until the de-assertion of the `resetreq` signal.

Figure 9. Nios V/g Processor Reset Network



4.3.7. Control and Status Registers

Nios V/g processor's Control and Status Registers (CSR) are both readable and writable. Nios V/g processor updates the CSR during the E-stage of the pipeline.

During the execution of a Nios V processor application, you may observe the following behaviors:

- CSR write instruction (in E-stage) is stalled due to the pending memory or multicycle instructions (in M-stage)
- CSR write instruction (in E-stage) continues after the pending instructions (in M-stage) are complete.
- If the processor generates an exception during the M-stage, the processor flushes the pending instructions in the pipeline (including the CSR write instruction in the E-stage) and initiates the trap handler to service the exception.

4.3.8. Trap Controller

In the Nios V processor, trap refers to the transfer of control to a trap handler caused by either an exception or an interrupt.

- Exceptions are synchronous events that originate inside the processor. They are commonly caused by an unusual condition occurring at run time associated with an instruction.
- Interrupts are asynchronous events that originate originated outside of the processor. They are commonly caused by service requests from system peripherals.

4.3.8.1. Exception Controller

The Nios V/g processor architecture provides a simple exception controller to handle all exception types. Each exception, including internal hardware interrupts, causes the processor to transfer execution to an exception address. An exception handler at this address determines the cause of the exception and executes an appropriate exception routine.

You can set the exception address in the **Nios V Processor Board Support Package Editor ➤ BSP Linker Script**. Nios V/g processor stores the address in machine trap handler base address (`mtvec`) CSR register.

All exceptions are precise. The processor completes all instructions that precede the faulting instruction and does not start the execution of instructions that follow after the faulting instruction.

Table 68. Exceptions

Exception	Description
Instruction Address Misaligned	The core pipeline logic in F-stage detects the exception. This exception is flagged if the core fetched a program counter that is not aligned to a 32-bit word boundary.
Instruction Access Fault	The instruction read response signal detects this exception.
Illegal Instruction	The instruction decoder in the D-stage flags this exception if an instruction word contains encoding for an unimplemented or undefined instruction. The control logic for the CSR read and write flags this exception in the E-stage if a CSR instruction accesses a CSR that is not implemented or undefined.
Breakpoint	The instruction decoder flags the software breakpoint exception EBREAK in the D-stage.
Load Address Misaligned	The core for the load/store unit in the M-stage detects the misalignment. This exception is flagged if the data address is not aligned to the size of the data access.
Store Address Misaligned	
Load Access Fault	The core for the data read and write response signal detects the exception.
Store Access Fault	
Env call from M-mode	The instruction decoder in the D-stage detects the instruction.

4.3.8.2. Interrupt Controller

The Nios V/g processor implementation supports the following interrupts:

- Platform interrupts with 16 level-sensitive interrupt request (IRQ) inputs.
- Internally-generated Timer and Software interrupt. You can access the timer interrupt register using the Timer and Software interrupt module interface by connecting to the data bus.

During an interrupt, the core writes the program counter of the attached instruction into the machine exception program counter (`mepc`) register. An interrupt is usually attached to the instruction in E-stage or in the preceding F-stage or D-stage pipeline. The core is not capable of retracting a memory instruction in the M-stage. If an instruction in M-stage flags an exception while an interrupt is pending and ready to be serviced, the core fetches and executes the exception instruction. If a memory or multicycle instruction is pending in the M-stage, for example, the core is waiting for the response, the core does not flag an interrupt until it receives a response for that instruction. Pending interrupts are flagged by their corresponding bits in Machine Interrupt-Pending (`mip`) register.

An interrupt is taken only when Machine Status Register (`mstatus`) bit 3 is asserted and bits corresponding to its pending interrupt in Machine Interrupt-pending (`mip`) register is asserted.

Table 69. Interrupt Control and Status Registers/Bits

Register	Status Registers/Bits	Description
`mstatus`	`mstatus[3]`/Machine Interrupt-Enable (MIE) field	Global interrupt-enable bit for machine mode
`mie`	`mie[31:16]`/Platform interrupt-enable field	Platform interrupt-enable bit for 16 hardware interrupts
	`mie[7]`/Machine Timer Interrupt-Enable (MTIE) field	Timer interrupt-enable bit for machine mode
	`mie[3]`/Machine Software Interrupt-enable (MSIE) field	Software interrupt-enable bit for machine mode
`mip`	`mip[31:16]`/Platform interrupt-pending field	Platform interrupt-pending bit for 16 hardware interrupts
	`mip[7]`/Machine Timer Interrupt-Pending (MTIP) field	Timer interrupt-pending bit for machine mode
	`mip[3]`/Machine Software Interrupt-Pending (MSIP) field	Software interrupt-pending bit for machine mode

4.3.8.2.1. Timer and Software Interrupt Module

The timer and software interrupt hosts the following registers:

- Machine Time (`mtime`) and Machine Time Compare (`mtimecmp`) registers for timer interrupt.
- Machine Software Interrupt-pending (`msip`) field for the software interrupt.

The value of `mtime` increments after every clock cycle. When the value of `mtime` is greater or equal to the value of `mtimecmp`, the timer posts the interrupt.

4.3.9. Memory and I/O Organization

You can configure the Nios V/g processor systems. Consequently, the memory and I/O organization varies from system to system. A Nios V/g processor core uses one or more of the following ports to provide access to memory and I/O:

- Instruction manager port: An Arm Advanced Microcontroller Bus Architecture (AMBA) 4 AXI Memory-Mapped manager port that connects to instruction memory via system interconnect fabric.
- Data manager port: An AMBA 4 AXI Memory-Mapped manager port that connects to data memory and peripherals via the system interconnect fabric.
- Instruction Cache: Fast cache memory internal to the Nios V/g processor core.
- Data Cache: Fast cache memory internal to the Nios V/g processor core.

Nios V/g Processor Core Memory Mapped I/O Access: Both data memory and peripherals are mapped into the address space of the data manager port. Nios V/g processor core uses little-endian byte ordering. Words and half-words are stored in memory with the more-significant bytes at higher addresses. The Nios V/g processor core does not specify anything about the existence of memory and peripherals. The quantity, type, and connection of memory and peripherals are system dependent.

4.3.9.1. Instruction and Data Buses

4.3.9.1.1. Instruction Manager Port

Nios V/g processor instruction bus is implemented as a 32-bit AMBA 4 AXI manager port.

The instruction manager port:

- Performs a single function: it fetches instructions to be executed by the processor.
- Does not perform any write operations.
- Can issue successive read requests before data return from prior requests.
- Can prefetch sequential instructions.
- Always retrieves 32-bit of data. Every instruction fetch returns a full instruction word, regardless of the width of the target memory. The widths of memory in the Nios V/g processor system is not applicable to the programs. Instruction address is always aligned to a 32-bit word boundary.
- Implements a burst adapter because it is bursting capable and can issue wrapping burst.

Table 70. Instruction Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	awaddr	Unused	[31:0]	Output
	awlen	Unused	[7:0]	Output
	awsize	Unused	[2:0]	Output
	awburst	Unused	[1:0]	Output
	awprot	Unused	[2:0]	Output
	awvalid	Unused	1	Output
	awready	Unused	1	Input
Write Data Channel	wdata	Unused	[31:0]	Output
	wstrb	Unused	[3:0]	Output
	wlast	Unused	1	Output
	wvalid	Unused	1	Output
	wready	Unused	1	Input
Write Response Channel	bresp	Unused	[1:0]	Input
	bvalid	Unused	1	Input
	bready	Unused	1	Output
Read Address Channel	araddr	Instruction Address (Program Counter)	[31:0]	Output
	arlen	Read burst length • 0 for peripheral region access • 7 for cacheable region access	[7:0]	Output
	arsize	Constant 2 (4 bytes)	[2:0]	Output

continued...

Interface	Signal	Role	Width	Direction
	arburst	Constant 2 (WRAP)	[1:0]	Output
	arprot	Unused	[2:0]	Output
	arvalid	Instruction address valid	1	Output
	arready	Instruction address ready (from memory)	1	Input
Read Data Channel	rdata	Instruction	[31:0]	Input
	rresp	Instruction response: Non-zero value denotes instruction access fault exception	[1:0]	Input
	rlast	Last transfer in a read burst	1	Input
	rvalid	Instruction valid	1	Input
	rready	Constant 1	1	Output

4.3.9.1.2. Data Manager Port

The Nios V/g processor data bus is implemented as a 32-bit AMBA 4 AXI manager port. The data manager port:

- Performs read data from memory or a peripheral when the processor executes a load instruction.
- Performs write data to memory or a peripheral when the processor executes a store instruction.
- Implements a burst adapter because it is bursting capable and can only issue incremental bursts.

axsize signal value indicates the load/store instruction size- byte (LB/SB), halfword (LH/SH) or word (LW/SW). Address on axaddr signal is always aligned to size of the transfer. For store instructions, respective writes strobe bits are asserted to indicate bytes being written.

Nios V/g processor core does not support speculative issue of load/store instructions. Therefore the core can issue only one load or store instruction and it waits until the issued instruction is complete.

Table 71. Data Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	awaddr	Store address	[31:0]	Output
	awlen	Write burst length • 0 for peripheral region access • 7 for cacheable region access	[7:0]	Output
	awsize	Store size: SB, SH, SW	[2:0]	Output
	awprot	Unused	[2:0]	Output
	awvalid	Store address valid	1	Output
	awready	Store address ready (from memory)	1	Input
Write Data Channel	wdata	Store data	[31:0]	Output

continued...

Interface	Signal	Role	Width	Direction
	wstrb	Byte position in word	[3:0]	Output
	wlast	Last transfer in write burst	1	Output
	wvalid	Store data valid	1	Output
	wready	Store data ready (from memory)	1	Input
Write Response Channel	bresp	Store response: Non-zero value denotes store access fault exception.	[1:0]	Input
	bvalid	Store response valid	1	Input
	bready	Constant 1	1	Output
Read Address Channel	araddr	Load address	[31:0]	Output
	arlen	Read burst length • 0 for peripheral region access • 7 for cacheable region access	[7:0]	Output
	arsize	Load size: LB, LH, LW	[2:0]	Output
	arprot	Unused	[2:0]	Output
	arvalid	Load address valid	1	Output
	arready	Load address ready (from subordinates)	1	Input
Read Data Channel	rdata	Load data	[31:0]	Input
	rresp	Load response: Non-zero value denotes load access fault exception	[1:0]	Input
	rlast	Last transfer in a read burst	1	Input
	rvalid	Load data valid	1	Input
	rready	Constant 1	1	Output

4.3.9.2. Address Map

The address map for memories and peripherals in a Nios V/g processor system is design dependent. The following addresses are part of the processor:

1. Reset Address
2. Debug Exception Address
3. Peripheral Region Base Address
4. Exception Address
5. Timer and Software Interrupt Address

You can specify the Reset Address, Debug Exception Address and Peripheral Region Base Address in Platform Designer during system configuration. You can modify the Exception Address stored in the `mvtec` register. `mvtime` and `mtimecmp` register controls the timer interrupt. The `msip` register bit controls the software interrupt.

4.3.9.3. Cache Memory

The Nios V/g processor architecture supports cache memories on both the instruction manager port (instruction cache) and the data manager port (data cache). The cache memories can improve the average memory access time for Nios V/g processor systems that use slow off-chip memory such as SDRAM for program and data storage.

The processor core connects the caches through a 32-bit AXI-4 interface, thus bursting is enabled. The instruction and data caches are always enabled at run-time, but software can bypass the data cache so that peripheral accesses do not return cached data. Software handles cache management and cache coherency. The Nios V/g instruction set provides instructions for cache management.

Table 72. Cache Byte Address Field

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
tag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tag		tag/index				index					offset				

Related Information

The RISC-V® Instruction Set Manual Volume I: Unprivileged ISA
More information on RISC-V "CMO" Extension for Base Cache Management Operation ISA.

4.3.9.3.1. Instruction Cache

The instruction cache memory has the following characteristics:

- Direct-mapped cache implementation
- 32 bytes (8 words) per cache line
- Configurable size of 1, 2, 4, 8, and 16 KBytes
- The instruction manager port reads an entire cache line at a time from memory, and issues one read per clock cycle.
- Critical word first
- Burst length of 8

The instruction byte address size is 32-bit. The size of the tag and index field depends only on the size of the cache memory. The offset field is always five bits (i.e., a 32-byte line). The instruction fetch fence (`fence.i`) instruction flushes and invalidates all instruction cache lines.

Related Information

The RISC-V® Instruction Set Manual Volume I: Unprivileged ISA
More information about the the `fence.i` instruction in RV32I Base Integer.

4.3.9.3.2. Data Cache

The data cache memory has the following characteristics:

- Direct-mapped cache implementation
- 32 bytes (8 words) per cache line
- Configurable size of 1, 2, 4, 8, and 16 KBytes
- The data manager port reads an entire cache line at a time from memory, and issues one read per clock cycle.
- Write-back
- Write-allocate (i.e., on a store instruction, a cache miss allocates the line for that address)
- Burst length of 8

The data byte address size is 32 bit. The size of the `tag` and `index` field depends only on the size of the cache memory. The `offset` field is always five bits (i.e., a 32-byte line).

The Nios V/g processor instruction set provides cache block management instructions for the data cache.

Table 73. RISC-V Zicbom Cache Block Management

Instruction	Name	Operation	Encoding
`cbo.clean <rs1>`[6][7]	Clean Data Cache Address	• Identifies the cache line with `tag` and `index` field. • If there is a cache hit, performs the following operations: — Clear the cache line's dirty state. — If the cache line is valid and dirty, data is written back to the memory.	Refer to *RISC-V Base Cache Management Operation ISA Extension*.
`cbo.flush <rs1>`[6][7]	Flush Data Cache Address	• Identifies the cache line with `tag` and `index` field. • If there is a cache hit, performs the following operations: — Invalidate the cache line. — Write the data back to the memory if the cache line is valid and dirty; otherwise, ignore the data.	
`cbo.inval <rs1>`[6][7]	Invalidate Data Cache Address	• Identifies the cache line with `tag` and `index` field. • If there is a cache hit, performs the following operations: — Invalidate the cache line. — Do not write the data back to the memory.	

(6) Source register 1 (`rs1`) holds the 32-bit cache line address (tag, index, and offset).

(7) If the specified cache line address is not found (cache miss), these instruction are implemented as `nop`, and does no changes to any cache lines.

Table 74. Custom Cache Block Management Instructions

Instruction	Operation	Encoding
`cbo.clean.ix <rs1>`[(8)	• Identifies the cache line with `index` field, • Clears the cache line's dirty state. • Keeps the cache line's valid state. • If the cache line is valid and dirty, data is written back to the memory.	Refer to Encoding for `cbo.clean.ix`
`cbo.flush.ix <rs1>`(8)	• Identifies the cache line with `index` • Invalidates the cache line. • If the cache line is valid and dirty, data is written back to the memory. • If the cache line is not dirty, data is not written back to the memory field,	Refer to Encoding for `cbo.flush.ix`
`cbo.inval.ix <rs1>`[(8)	• Identifies the cache line with index field, • Invalidates the cache line. • Data is not written back to the memory.	Refer to Encoding for `cbo.inval.ix`

Table 75. Encoding for `cbo.clean.ix`

Bit Field				
31:20	19:15	14:12	11:7	6:0
imm	rs1	cbo	Reserved	misc-mem
000010000001	rs1	010	00000	0001111

Table 76. Encoding for `cbo.flush.ix`

Bit Field				
31:20	19:15	14:12	11:7	6:0
imm	rs1	cbo	Reserved	misc-mem
000010000010	rs1	010	00000	0001111

Table 77. Encoding for `cbo.inval.ix`

Bit Field				
31:20	19:15	14:12	11:7	6:0
imm	rs1	cbo	Reserved	misc-mem
000010000000	rs1	010	00000	0001111

4.3.9.3.3. Configurable Cache Memory

You can configure the size of the cache memory. The cache memory has no effect on program functionality but affects the speed with which the processor fetches instructions and reads/writes data.

(8) Source register 1 (`rs1`) holds the cache line index, which can be 5 to 9-bits depending on the cache size.

4.3.9.3.4. Bypassing Cache (Peripheral Region)

The Nios V/g architecture has two peripheral regions for bypassing the caches. Nios V/g cores optionally support the peripheral region mechanism to indicate cacheability. In the Platform Designer, the peripheral region cache-ability mechanism allows you to specify a region of address space that is non-cacheable. The peripheral region is any integer power of 2 bytes, from a minimum of 64 kilobytes up to a maximum of 2 gigabytes, and must be located at a base address aligned to the size of the peripheral region. The peripheral region is available provided that an MMU is not present. The peripheral regions are non-bursting.

Note: Any accesses to the Nios V processor's debug module or timer module are non-cacheable.

Note: You must place the peripherals driven by the Nios V/g processor within a defined peripheral region to achieve cache bypass, which is required by a standard design implementation.

4.3.9.3.5. Using Cache Memory Effectively

The effectiveness of cache memory to improve performance is based on the following conditions:

- Regular memory is located off-chip and has a longer access time than on-chip memory.
- The largest, performance-critical instruction loop is smaller than the instruction cache.
- The largest block of performance-critical data is smaller than the data cache.

The optimal cache configuration is application-specific, but you can define a configuration that works for a variety of applications. Refer to the following examples:

- If a Nios V/g processor system only has fast on-chip memory and never accesses slow off-chip memory, an instruction or data cache is unlikely to boost the performance.
- If a program's critical loop is 2 KB but the instruction cache is 1 KB, an instruction cache does not improve execution speed. In this case, an instruction cache can actually degrade performance.

4.3.9.4. Tightly Coupled Memory

Tightly coupled memory (TCM) guarantees fixed low-latency memory access for performance-critical applications. The advantages of TCM over cache memory are as follows:

- Performance similar to cache memory
- Software can guarantee that performance-critical code or data is located in TCM
- No real-time caching overhead, such as loading, invalidating, or flushing memory

Physically, a TCM is a dedicated on-chip memory within the Nios V processor core. Intel implements TCM using the M20K memory blocks. The Nios V processor architecture supports four TCMs for instruction access and data access (two per access type). Each TCM provide an AXI4-Lite interface for connection with an external master, or manager. The interconnect allows you to connect Avalon or other supported manager, gaining access to read or write the respective TCMs.

4.3.9.4.1. Instruction and Data Tightly Coupled Memory

Nios V/g processor supports the following TCMs:

- Two instruction TCMs
- Two data TCMs

Table 78. TCM Characteristics

Characteristics	TCM
4 bytes (1 words) per address line	• Instruction TCM • Data TCM
Fixed memory latency of 1 cycle	• Instruction TCM • Data TCM
Configurable size of 0 (Disabled) to 512MBytes	• Instruction TCM • Data TCM
Configurable 32-bits base address	• Instruction TCM • Data TCM
Supports memory initialization using MIF or HEX file	• Instruction TCM • Data TCM
Read-only permission for processor core	Instruction TCM
Read/Write permission for external AXI4-Lite manager	Instruction TCM
Read/Write permission for processor core and external AXI4-Lite manager	Data TCM

Table 79. Tightly Coupled Memory Interface Signals

Interface	Signal	Role	Width	Direction
Write Address Channel	awaddr	Write address	Width = log (tcm_size)/log 2[9]	Input
	awprot	Unused	[2:0]	Input
	awvalid	Write address valid	1	Input
	awready	Write address ready (from TCM)	1	Output
Write Data Channel	wvalid	Write data valid	1	Input
	wdata	Write data	[31:0]	Input
	wstrb	Byte position in word	[3:0]	Input
	wready	Write data ready (from TCM)	1	Output
Write Response Channel	bvalid	Write response valid	1	Output
	bresp	Write response: Non-zero value denotes store access fault exception	[1:0]	Output
	bready	Write response ready (from external manager)	1	Input
Read Address Channel	araddr	Read address	Width = log (tcm_size)/log 2[9]	Input

continued...

(9) The width of the Write and Read Addresses depends on the selected TCM size.

Interface	Signal	Role	Width	Direction
	arprot	Unused	[2:0]	Input
	arvalid	Read address valid	1	Input
	arready	Read address ready (from TCM)	1	Output
Read Data Channel	rdata	Read data	[31:0]	Output
	rvalid	Read data valid	1	Output
	rresp	Read data response: Non-zero value denotes load access fault exception	[1:0]	Output
	rready	Read data ready (from TCM)	1	Input

4.3.9.4.2. Accessing Tightly Coupled Memory

TCM occupies standard address space like other memory devices connected via system interconnect fabric. The processor configures the address ranges for TCM (if any).

Access from Processor Core

When TCM is present, the Nios V/g processor core decodes addresses internally to determine if the requested addresses reside in the TCMs. If the address resides in the TCM, the processor core fetches the instruction from instruction TCM or loads the data from the data TCM. The software accesses TCM using regular load and store instructions. From the software's perspective, there is no difference in accessing a TCM compared to other memory.

Accessing TCM bypasses cache memory. The processor core functions as if the cache were not present for the address span of the TCM. Instructions for managing the cache do not affect the TCM, even if the instruction specifies an address in TCM.

Access from External AXI4-Lite Manager

Any external AXI4-Lite manager can access any TCMs as RAM in the system if it is connected to the TCM's AXI4-Lite interface. The TCMs support dual-port access, which allows two hosts (processor core and external AXI4-Lite manager) to access the memory simultaneously.

You can access the TCM memories of a Nios V processor core (Core 1) from another core (Core 2) in a multi-core system. Core 2 acts as the external AXI4-Lite Manager in this setup. Enable this feature by connecting the data bus of Core 2 to the TCM AXI4-Lite subordinate port of Core 1.

4.3.9.4.3. Using Tightly Coupled Memory Effectively

A system can use TCM to achieve maximum performance for accessing a specific code or data section. For example, interrupt-intensive applications can place exception handler code into a TCM to minimize interrupt latency. Similarly, compute-intensive digital signal processing (DSP) applications can place data buffers into TCM for the fastest possible data access.

You can implement one or more of the following functions or modules using TCMs to enhance the performance of your system:

- Constant access time with no arbitration delays.
- Separate exception stack for use only while handling interrupts.
- Fast data buffers
- Fast sections of code:
  - Fast interrupt handler
  - Critical loop

If the application's memory requirements are small enough to fit entirely on-chip, it is possible to use tightly coupled memory exclusively for code and data. Larger applications must selectively choose what to include in tightly coupled memory to maximize the cost-performance trade-off.

Consider the following design when using TCMs:

- Strike a balance between the speed enhancement from caches with the higher speed enhancement from TCMs.
- Divide the on-chip memory resources equitably for the best combination of TCMs and cache.
- Locating any information (code or data) within TCMs eliminates cache overhead such as cache flushing, loading, or invalidating.

Note: You can connect Nios V processor data manager to instruction TCM subordinate port. Like Nios II processor, the Instruction TCM can map into the processor's data region for memory debugging.

Related Information

AN 978: Nios® V Processor Migration Guidelines
Refer to the Nios V V Processor Migration Guide for the difference between Nios V andNios II processors TCMs.

4.3.10. RISC-V based Debug Module

The Nios V/g processor architecture supports a RISC-V based debug module that provides on-chip emulation features to control the processor remotely from a host PC. PC-based software debugging tools communicate with the debug module and provide facilities, such as the following features:

- Reset Nios V processor core and timer module
- Download programs to memory
- Start and stop execution
- Set software breakpoints and watchpoints
- Analyze registers and memory

4.3.10.1. Debug Mode

You can enter the Debug Mode, as specified in the RISC-V architecture specification, in the following ways:

1. Halt from Debug Module
2. Software breakpoints
3. Trigger

Upon entering Debug Mode, Nios V processor completes the instruction in W-stage. By the order of priority, instruction in M-stage, E-stage, D-stage or F-stage takes the interrupt.

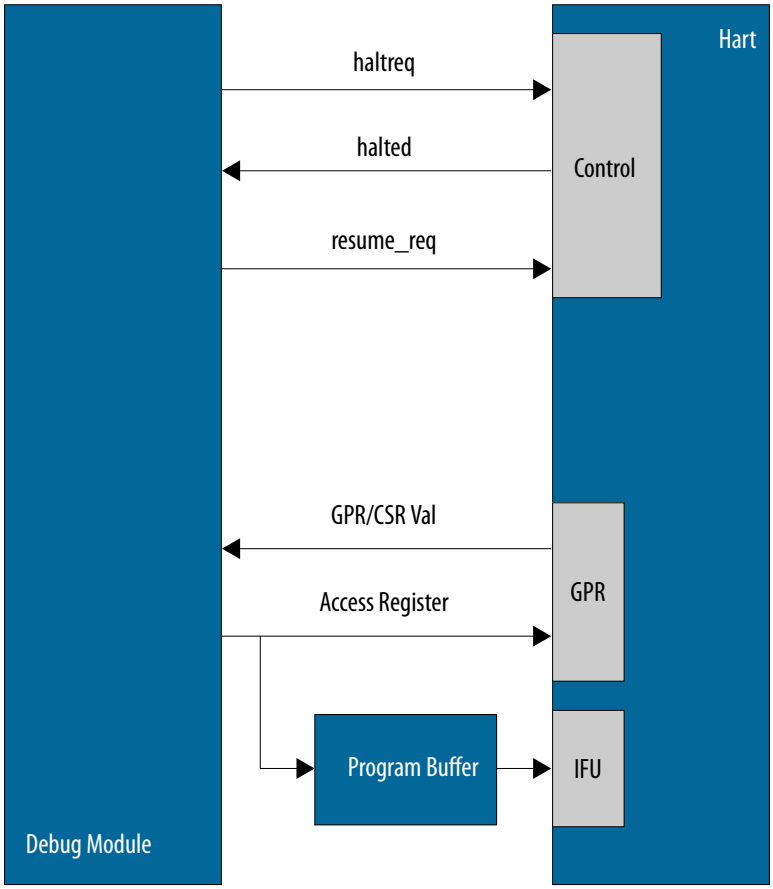
- When there is a valid instruction, Program Counter writes to the Debug Program Counter, `.dpc`.
- If the instruction in M-stage is not valid, then instruction in E-stage takes the interrupt and so on and so forth.
- If there is no valid instruction in the pipeline, the Program Counter for the next instruction writes to the Debug Program Counter, `.dpc`.

Note: For branches, the next Program Counter depends on whether a branch was taken or not taken, and whether the branch prediction (if any) was correct.

Debug Module selects Hardware Thread (Hart); which can be in one of the following states:

1. Non-existent: Debug Module probes a hart which does not exist.
2. Unavailable: Reset or temporary shutdown.
3. Running: Normal operation outside of debug.
4. Halted: Hart is said to be halted when it is in debug mode.

Figure 10. Debug Module Block Diagram



4.3.10.2. Halt from Debug Module

The debugger can write the `haltreq` bit in the Debug Module Control (`dmcontrol`) register, which places the Nios V processor in debug mode. The assertion of `haltreq` sends an asynchronous interrupt to the processor core logic.

4.3.10.3. Trigger

The Nios V processor core supports one address or data match trigger. The trigger registers are accessible using RISC-V `csr` opcodes or abstract debug commands. The firing of trigger can either enter the Debug Mode or raise a breakpoint exception, which depends on the trigger registers.

Table 80. Trigger Registers Implemented in Nios V Processor

Name	Registers	Description
`tselect`	Trigger Select	Nios V processor supports one trigger, therefore the processor selects Trigger 0 at default (value set at 0).
`tdata1`	Trigger Data 1	The value of `type` field is 2 to represent Trigger 0 as an address or data match trigger. Write behavior to `tdata` registers depends on the `dmode` field. The remaining bits acts as `mcontrol`.

continued...

Name	Registers	Description
mcontrol	Match Control	Controls address and data trigger implementation according to action, m, execute, store and load fields. The Nios V processor does not implement other fields.
tdata2	Trigger Data 2	Holds trigger-specific data (virtual address, instruction opcode, data stored or loaded).
tinfo	Trigger Info	The value of info field is 4 at default to signify tdata1.type is 2. This implies selected trigger is an address or data match trigger.

Based on the bit field setting in mcontrol register, Nios V processor can implement different trigger types after the selected trigger matches the tdata2 register.

Table 81. Nios V Processor Triggers Condition and Firing Time

Triggers	Condition	Firing Time	Exception Program Counter
Instruction Address Trigger	Program Counter matches tdata2	Before executing the instruction	The processor sets the Machine Exception Program Counter (mepc) to the instruction address (PC).
Instruction Opcode Trigger	Instruction opcode matches tdata2	Before executing the instruction	
Store Address Trigger	Store address matches tdata2	After executing the store instruction	The processor sets the mepc to the next instruction address (PC + 4).
Store Data Trigger	Store data matches tdata2	After executing the store instruction	
Load Address Trigger	Load address matches tdata2	After executing the load instruction	
Load Data Trigger	Load data matches tdata2	After executing the load instruction	

Table 82. Address or Data Match Trigger Type

Trigger	mcontrol bit fields			
	select	execute	store	load
Instruction Address	0	1	0	0
Instruction Opcode	1	1	0	0
Store Address	0	0	1	0
Store Data	1	0	1	0
Load Address	0	0	0	1
Load Data	1	0	0	1

Note: The trigger is disabled in the following situations:

- The Nios V processor is in debug mode.
- The Nios V processor is in machine mode, while mcontrol.m or mcontrol.type is 0.

4.3.10.4. Abstract Commands in Debug Mode

Nios V/g processor implements Access Register abstract command. The Access Register command allows read-write access to the processor registers including GPRs, CSRs, FP registers and Program Counter. The Access Register also allows program

execution from program buffer. The debugger executes Access Register commands by writing into Abstract Command (`command`) register using the Access Register command encoding.

Table 83. Access Register Command Encoding

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cmdtype								0	aarsize			aarpostincrement	postexec	transfer	write
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
regno															

Table 84. Fields Descriptions

Field	Role
`cmdtype`	Determine command type 0 : Indicates Access Register command.
`aarsize`	Specifies size of register access 2: Access the lowest 32 bit of register 3: Access the lowest 64 bit of register 4: Access the lowest 128 bit of register
`aarpostincrement`	0: No effect 1: regno is incremented after successful register access
`postexec`	0: No effect 1: Execute program in program buffer
`transfer`	Acts in conjunction with write field. 0: Ignore value in write field 1: Execute operation specified by write field.
`write`	0: Copy data from register 1: Copy data to register
`regno`	Register address to be accessed.

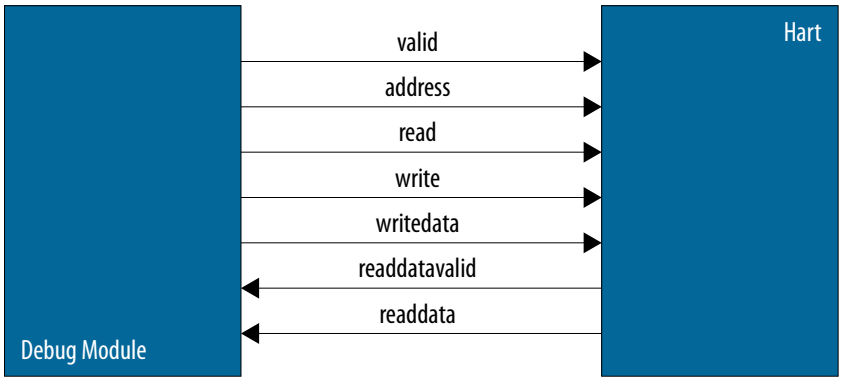
Note: The Nios V/g processor does not support abstract commands when hardware thread is not halted.

Table 85. Software Response to Unimplemented Commands

Field	State	
`cmderr[10:8]`	0	No error
	1	Busy
	2	Command not supported
	3	Exception - from program buffer instruction
	4	Command not executed because hart unavailable, or not in correct state to execute command.
	5	Abstract command failed due to bus error
	6	RSVD
	7	Command failed for other reasons.

Debug Module has the Abstract Control and Status CSR which includes the `cmderr` field. The `cmderr` field represents the current state of abstract command being executed. If the `cmderr` field is non-zero, writes to the `command` register are ignored. To clear the `cmderr` field, write 1 for every bit in the field.

Figure 11. Debug Module Interface Signals



Avalon memory-mapped interface implement the Register Access using request/response bus with Debug Module being the initiator and core being the responder. Address bus carries the register ID.

4.3.10.5. Hardware/Software Interface

The Nios V processor debug module is based on the RISC-V Debug specification. It supports the same Debug Module registers documented in the specification. Each register has a fixed address as specified in the RISC-V Debug Support specification. Debugger can determine the register implementation status by writing or reading from the Debug Module registers. Unimplemented registers return 0 when read.

Debugger can check the status of the system by reading Debug Module Status (`dmstatus`) register. Debug Module Status register is read-only and provides status of the Debug Module and the selected harts.

You can access a specific hart by writing to the `hartsel` field in `dmcontrol`. Other fields in `dmcontrol` specify the action a debugger can take.

Halt Summary 0 (`haltsum0`) register reflects the status of a hart (halted/not halted). The LSB of this register can reflect whether hart is halted or not. Other bits is always 0. This is a read-only register of the debugger.

Related Information

RISC-V® Debug Release
More information about the conceptual view of the states, refer to Section : Overview of States, Figure: Run/Halt Debug State Machine for single hart systems.

4.3.11. Error Correction Code (ECC)

The Nios V/g processor core has the option to enable error detection and ECC status for the following internal RAM blocks:

- Register file
- Instruction cache
- Data cache
- Tightly coupled memories

Each RAM block has its source ID. When an ECC event occurs, the processor transmits the source ID and ECC status to the ECC interface.

- If the ECC event is a correctable error, the processor continues to operate without correcting the error.
- If the ECC event is an un-correctable error, the processor halts its current progress and stall. You need to reset either the processor core alone, or entire system

Note: To reset the processor core alone, you need to apply the **Reset Request Interface** to safely reset the Nios V processor (Cleared of any outstanding operations). To reset the entire system, you can use the hard reset interface instead.

Related Information

Embedded Memory (RAM: 1-PORT, RAM: 2-PORT, ROM: 1-PORT, and ROM: 2-PORT) User Guide
For more information about ECC on Arria 10 devices.

4.3.11.1. ECC Interface

The ECC interface allows external logic to monitor ECC errors in the Nios V/g processor. The interface is a conduit and made up of two output signals.

- `cpu_ecc_status`: Indicates the error status.
- `cpu_ecc_source`: Indicates the error source

Table 86. `cpu_ecc_status`

2-bits Encoding	ECC Status	Effects on Software	Next Action
2'b00	No ECC event	None	None
2'b01	Reserved	Not Applicable	Not Applicable
2'b10	Correctable single bit ECC error	None	None
2'b11	Un-correctable ECC error	Likely fatal and halts processor	Reset either processor or entire system

Table 87. `cpu_ecc_source`

4-bits Encoding	ECC Source	Available
4'b0000	No ECC event	Always
4'b0001	General Purpose Register (GPR)	Always

continued...

4-bits Encoding	ECC Source	Available
4'b0010	Instruction Cache Data RAM	Always
4'b0011	Instruction Cache Tag RAM	Always
4'b0100	Data Cache Data RAM	Always
4'b0101	Data Cache Tag RAM	Always
4'b0110	Instruction TCM1	When Instruction TCM1 is present
4'b0111	Instruction TCM2	When Instruction TCM2 is present
4'b1000	Data TCM1	When Data TCM1 is present
4'b1001	Data TCM2	When Data TCM2 is present
4'b1010	Floating Point Register (FPR)	When FPU is present
4'b1111	Reserved	Not Applicable

Note: Due to a limitation with embedded memory blocks, the simulation model of Nios V processor does not support ECC on Arria 10 devices.

4.3.11.2. ECC Error Injection

ECC error injection allows you to inject ECC errors into the processor. This ECC error injection tests the hardware features, firmware, and driver development. You can trigger an ECC exception by writing an error code into the ECC Error Inject module based on the `mtval2` table.

Note: To prevent the hardware from continuing to trigger the ECC error, write a zero to turn off the ECC error injection.

Altera provides an error injection API for users to inject an ECC error into the processor based on the Table *List of ECC Sources based on `mtval2` Bit Field*. The source header file is in <BSP `Project`>`\HAL\inc\sys \alt_ecc_error_inject.h`.

```
alt_ecc_error_inject(ALT_ECC_ERROR_TYPE type)I
```

For example, if you write 24 to the module, the `cpu_ecc_status` and `cpu_ecc_source` of Data TCM1 become 2'b10 and 4'b1000 respectively. If you write 25 to the module, the `cpu_ecc_status` and `cpu_ecc_source` of Data TCM1 become 2'b11 and 4'b1000, respectively.

Table 88. How ECC Error Injection Works

ECC Error Inject	Module based on mtval2	`cpu_ecc_status`	`cpu_ecc_source`
24	Data TCM1 Correctable Error	2'b10 (Correctable)	4'b1000 (Data TCM1)
25	Data TCM1 Uncorrectable Error	2'b11 (Uncorrectable)	4'b1000 (Data TCM1)

4.3.11.3. Affected CSR during ECC Event

The CSRs are involved in the ECC: `mcause`, `mtval` and `mtval2`. For more information, refer to the *Control and Status Register Field*.

Table 89. List of Affected CSR

Affected CSR	Description
mcause	An exception code 19 (0x13 – Hardware Error Exception) is written to the mcause.
mtval	Contains the violating address when an ECC error occurs during a load/store or instruction fetch error.
mtval2	Provides more information for an ECC event. Refer to the table below.

Table 90. List of ECC Sources based on mtval2 Bit Field

mtval2 Value	ECC Error Source
0 (32'h0)	No Error
1 (32'h1)	GPR ECC Uncorrectable Error
3 (32'h3)	FPR ECC Uncorrectable Error
16 (32'h10)	Instruction TCM1 Correctable Error
17 (32'h11)	Instruction TCM1 Uncorrectable Error
18 (32'h12)	Instruction TCM2 Correctable Error
19 (32'h13)	Instruction TCM2 Uncorrectable Error
24 (32'h18)	Data TCM1 Correctable Error
25 (32'h19)	Data TCM1 Uncorrectable Error
26 (32'h1A)	Data TCM2 Correctable Error
27 (32'h1B)	Data TMC2 Uncorrectable Error
33 (32'h21)	Instruction Cache TAG RAM Uncorrectable Error
35 (32'h23)	Instruction Cache Data RAM Uncorrectable Error
41 (32'h29)	Data Cache TAG RAM Uncorrectable Error
43 (32'h2B)	Data Cache Data RAM Uncorrectable Error
Others	Reserved

4.3.12. Branch Prediction

Nios V/g processor core supports Static Branch Prediction. This core implements Backward Taken, Forward Not Taken (BTFN) mechanism. This technique is a simple form of branch prediction because it does not rely on the history of the branches. It predicts the outcome of a branch instruction as follows:

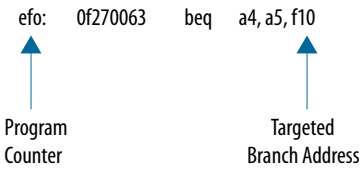
- Always taken, when backward branches, or
- Always not taken when forward branches.

The processor core determines the type of branches by comparing the Program Counter and targeted branch address (branch PC).

- Backward branch - When the target branch address is less than the Program Counter.
- Forward branch - When the target branch address is greater than the Program Counter.

For example, the assembly code below shows a forward branch. ef0 is the Program Counter, and f10 is the targeted branch address. Since the targeted branch address is greater than the Program Counter, the processor core does not take the branch.

Figure 12. Branch Prediction



Note: Enabling branch prediction can improve the core performance (such as the Dhrystone and CoreMark benchmark), but it may also increase logic resource utilization.

4.3.13. Lockstep Module

The Nios V Processor Lockstep feature utilizes fRSmartComp technology to implement a smart comparator in RTL. Altera utilizes the Dual-Core Lock Step (DCLS) safety architecture to implement the smart comparator and integrates fRSmartComp technology into the Nios V/g processor, allowing for the design of fail-safe applications.

Related Information

Nios V Processor: Lockstep Implementation

4.4. Programming Model

4.4.1. Privilege Levels

The privilege levels in Nios V/g processor are designed based on the RISC-V architecture specification. The privilege levels available are Machine Mode (M-mode) and Debug Mode (D-mode).

Related Information

RISC-V® Specifications

4.4.1.1. Machine Mode (M-mode)

The default operating mode is Machine mode (M-mode). The core boots up into M-mode after power-on reset. M-mode provides low-level access to the machine implementation and all CSRs.

4.4.1.2. Debug Mode (D-mode)

The Debug mode (D-mode) is an additional privilege level to support off-chip debugging and manufacturing test.

The core can enter D-mode using any of the following methods:

- After reset when bit is set in debug CSR.
- Using debug interrupt.
- Using EBREAK instruction when bit is set in debug CSR.
- Using single step.

4.4.2. Control and Status Registers (CSR) Mapping

Control and status registers report the status and change the behavior of the processor. Since the processor core only supports M-mode and D-mode, Nios V/g processor implements the CSRs supported by these two modes.

Table 91. Control and Status Registers List

Number	Privilege	Name	Description
Floating-Point CSRs			
0x001	MRW	`fflags`	Floating-Point Accrued Exceptions. Refer to Floating-Point CSR Register Fields table.
0x002	MRW	`frm`	Floating-Point Dynamic Rounding Mode. Refer to Floating-Point CSR Register Fields table.
0x003	MRW	`fcsr`	Floating-Point Control and Status Register (`frm` and `fflags`). Refer to Floating-Point CSR Register Fields table.
Machine Trap Setup			
0x300	MRW	`mstatus`	Machine status register. Refer to Machine Status Register Fields table.
0x301	MRW	`misa`	ISA and extensions. Refer to Machine ISA Register Fields table.
0x304	MRW	`mie`	Machine interrupt-enable register. Refer to Machine Interrupt-Enable Register Fields table.
0x305	MRW	`mtvec`	Machine trap-handler base address. Refer to .Machine Trap-Handler Base Address Register Fields table.
Machine Trap Handling			
0x341	MRW	`mepc`	Machine exception program counter. Refer to Machine Exception Program Counter Register Fields table.
0x342	MRW	`mcause`	Machine trap cause. Refer to Machine Trap Cause Register Fields table.
0x343	MRW	`mtval`	Machine bad address or instruction. Refer to Machine Trap Value Register Fields table.
0x344	MRW	`mip`	Machine interrupt pending. Refer to Machine Interrupt-Pending Register Fields table.
0x348	MRW	`mtval2`	Machine bad guest physical address. Refer to Machine Trap Value 2 Register Fields table.
Trigger Registers			
0x7A0	MRW	`tselect`	Trigger select. Refer to Trigger Select Register Fields table.
0x7A1	MRW	`tdata1 (mcontrol)`	Trigger data 1 (Match Control). Refer to Trigger Data 1 (Match Control) Register Fields table.

continued...

Number	Privilege	Name	Description
0x7A2	MRW	tdata2	Trigger data 2. Refer to Trigger Data 2 Register Fields table.
0x7A4	MRO	tinfo	Trigger info. Refer to Trigger Info Register Fields table.
Debug Mode Registers			
0x7B0	DRW	dcsr	Debug control and status register. Refer to Debug Control and Status Register Fields table.
0x7B1	DRW	dpc	Debug Program Counter. Refer to Debug Program Counter Register Fields table.
Machine Information Register			
0xF11	MRO	mvendorid	Vendor ID. Refer to Vendor ID Register Fields table.
0xF12	MRO	marchid	Architecture ID. Refer to Architecture ID Register Fields table.
0xF13	MRO	mimpid	Implementation ID. Refer to Implementation ID Register Fields table.
0xF14	MRO	mhartid	Hardware thread ID. Refer to Hardware Thread ID Register Fields table.

Related Information

The RISC-V® Instruction Set Manual Volume II: Privileged Architecture
More information about M-mode CSR and their respective fields.

4.4.2.1. Control and Status Register Field

The value in the each CSR registers determines the state of the Nios V/g processor. The field descriptions are based on the RISC-V specification.

Table 92. Floating-Point CSR Register Fields

The `fcsr` CSR is a 32-bit read/write register that holds the accrued exception flags.

RISC-V CSR instructions can access `fflags` and `frm` field individually by specifying the CSR address (0x001 and 0x002) respectively.

When accessing `frm` field individually using the CSR address 0x002, the rounding mode is transferred as bits [2:0] of the destination register.

RNE (000) is the only supporting rounding mode. Writing other rounding mode into `frm` generates an illegal instruction exception.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Rounding Mode (`frm`)			Accrued Exceptions (`fflags`)				
								000			NV	DZ	OF	UF	NX

Table 93. Machine Status Register Fields

The `mstatus` register is a 32-bit read-write register that keeps track of and controls the hart's current operating state.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD	0								0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FS[1:0]		MPP[1:0]		0		0	MPIE	0	0	0	MIE	0	0	0

The following bitfields are read-only 0:

- Bit 22 (TSR = 0): S-mode is not supported
- Bit 21 (TW = 0): There are no modes less privileged than M-mode.
- Bit 20 (TVM = 0): S-mode is not supported
- Bit 19 (MXR = 0): S-mode is not supported
- Bit 18 (SUM = 0): S-mode and U-mode are not supported
- Bit 17 (MPRV = 0): U-mode is not supported
- Bit 16 and Bit 15 (XS[1:0] = 0): S-mode is not supported
- Bit 10 and Bit 9 (VS[1:0] = 0): S-mode is not supported
- Bit 8 (SPP = 0): S-mode is not supported
- Bit 6 (UBE = 0): U-mode is not supported
- Bit 5 (SPIE = 0): S-mode is not supported
- Bit 1 (SIE = 0): S-mode is not supported

Table 94. Machine ISA Register Fields

The `misa` CSR is a read-write register reporting the ISA supported by the hart.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXL[1:0]		0				Extension[25:0]									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Extension[25:0]															

Table 95. Machine Interrupt-Enable Register Fields

The `mie` register is a 32-bit read-write register that contains interrupt enable bits. This core does not support supervisor or user modes. Thus, any interrupt related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Platform Interrupt[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MEIE	0	0	0	MTIE	0	0	0	MSIE	0	0	0

Table 96. Machine Trap-Handler Base Address Register Fields

The `mtvec` register is a 32-bit read/write register that holds trap vector configuration, consisting of a vector base address (BASE) and a vector mode (MODE).

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Base[31:2]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base[31:2]														Mode	

Table 97. Machine Exception Program Counter Register Fields

The `mepc` register is a 32-bit read-write register that holds the addresses of the instruction that was interrupted or that encountered the exception when a trap is taken into M-mode.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
mepc														

Table 98. Machine Trap Cause Register Fields

The `mcause` register is a 32-bit read-write register that hold the code indicating the event that caused the trap when a trap is taken into M-mode.

Bit Field														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt	Exception code [30:16]														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Exception code [15:0]															

Table 99. Machine Trap Value Register Fields

The `mtval` register is a 32-bit read-write register that is written with exception-specific information to assist software in handling the trap when a trap is taken into M-mode.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
mtval														

Table 100. Machine Interrupt-Pending Register Fields

The `mip` register is a 32-bit read/write register containing information on pending interrupts. This core does not support supervisor or user modes. Thus, any interrupt related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Platform Interrupt[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MEIP	0	0	0	MTIP	0	0	0	MSIP	0	0	0

Table 101. Trigger Select Register Fields

The `tselect` register is a 32-bit read/write register that selects the current trigger is accessible by other trigger register.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
tselect														

Table 102. Machine Trap Value 2 Register Fields

The mtval2 register is a 32-bit read-write register that is written with exception-specific information to assist software in handling the trap when a trap is taken into M-mode.

This core does not support hypervisor mode. Thus, the Nios V/g processor implements this register for ECC exception only (No implementation on guest-page faults).

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
mtval2														

Table 103. Trigger Data 1 (Match Control) Register Fields

The `tdata1 (mcontrol)` register is a 32-bit read/write register containing information on the trigger type, tdata registers accessibility, and trigger implementation. This core does not support supervisor or user modes. Thus, any bitfield related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
type=2				dmode	maskmax						hit	select	timing	sizelo	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
action				chain	match				m	0	0	0	execute	store	load

Table 104. Trigger Data 2 Register Fields

The `tdata2` register is a 32-bit read/write register containing the trigger-specific data.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
tdata2														

Table 105. Trigger Info Register Fields

The `tinfo` register is a 32-bit read-only register containing information on each possible tdata1.type.

Bit Field												
31	30	29	...	18	17	16	15	14	13	...	2	1	0
0							info						

Table 106. Debug Control and Status Register Fields

The `dcsr` CSR is a 32-bit read-write register containing information and status during D-mode. This core does not support supervisor or user modes. Thus, any bitfield related to these modes is read-only 0.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
debugver				0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ebreakm	0	0	0	stepie	stopcount	stoptime	cause			0	mprven	nmip	step	prv	

Table 107. Debug Program Counter Register Fields

Upon entry to D-mode, `dpc` CSR is updated with the virtual address of the next instruction to be executed.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
dpc														

Table 108. Vendor ID Register Fields

The `mvendorid` CSR is a 32-bit read-only register that provides the JEDEC manufacturer ID of the provider of the core.

Bit Field															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Bank															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bank									Offset						

Table 109. Architecture ID Register Fields

The `marchid` CSR is a 32-bit read-only register encoding the base microarchitecture of the hart.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
Architecture ID														

Table 110. Implementation ID Register Fields

The `mimpid` CSR provides a unique encoding of the version of the processor implementation.

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
Implementation														

Table 111. Hardware Thread ID Register Fields

The `mhartid` CSR is a 32-bit read-only register that contains the integer ID of the hardware thread running the code

Bit Field														
31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
Hart ID														

4.5. Core Implementation

4.5.1. Instruction Set Reference

The Nios V/g processor is based on the RV32IMZicsr_Zicbom specification, optionally "F" extension. RV32IMZicsr_Zicbom includes:

- RV32I base integer instruction ("I" extension)
- Integer multiplication and division instructions ("M" extension)
- Control and status register instructions ("Zicsr" extension)
- Base cache management operation ISA ("CMO" extension)
- Optionally, single-precision floating-point instruction ("F" extension)

There are 6 types of instruction formats. They are R-type, I-type, S-type, B-type, U-type, and J-type.

Besides the 'CMO' extension from RISC-V, Altera also provides customized cache management instructions. Refer to *Data Cache* for more information.

Table 112. Instruction Formats (R-type)

Bit Field (R-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
funct7							rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			rd					opcode						

Table 113. Instruction Formats (I-type)

Bit Field (I-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[11:0]												rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			rd					opcode						

Table 114. Instruction Formats (S-type)

Bit Field (S-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[11:5]							rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			imm[4:0]					opcode						

Table 115. Instruction Formats (B-type)

Bit Field (B-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[12]	imm[10:5]						rs2					rs1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rs1	funct3			imm[4:1]			imm[11]		opcode						

Table 116. Instruction Formats (U-type)

Bit Field (U-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[15:12]				rd					opcode						

Table 117. Instruction Formats (J-type)

Bit Field (J-type)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
imm[20]	imm[10:1]										imm[11]	imm[19:16]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[15:12]				rd					opcode						

Related Information

- The RISC-V® Instruction Set Manual Volume I: Unprivileged ISA
  More information about the instruction set of the RV32I Base Integer, the 'M' standard extension and the Zicsr extension.
- Data Cache on page 59

5. Nios V Processor Reference Manual Archives

For the latest and previous versions of this user guide, refer to Nios® V Processor Reference Manual. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

IP versions are the same as the Quartus Prime Design Suite software versions up to v19.1. From Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

6. Document Revision History for the Nios V Processor Reference Manual

Document Version	Quartus Prime Version	Changes
2024.10.07	24.3	• Updated the fMAX and Logic Size in the table *Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Pro Edition* for each Nios V/c, Nios V/m, and Nios V/g processors. • Updated the table *Benchmark Parameters for Quartus Prime Software* for each Nios V/c, Nios V/m, and Nios V/g processors. • Added a new topic *Choosing A Suitable Interface* for Nios V/c Processor and Nios V/m processors. • Updated the topic *Nios V/g: Processor Architecture*: — Updated the figure *Nios V/g Processor Core Block Diagram* to add Branch Prediction and Lockstep Module. — Added new topics: • *Branch Prediction* • *Lockstep Modules* • Edited the topic *Error Correction Code* and split int into a new topic *ECC Interface*. — Added new topics: • *ECC Error Injection* • *Affected CSR during ECC Event* • *Branch Prediction* — Added `mtval2` as a new control and status register to the table *Control and Status Registers List* and the topic *Control and Status Register Field*. • Edited the following topics to add more clarity: — *Nios V/m: Processor: Trap Controller* — *Nios V/m: Halt from Debug Module* — *Nios V/m: Hardware/Software Interface* — *Nios V/g: Control and Status Registers*
2024.07.26	24.2	• Updated the following topics for each Nios V/c, Nios V/m, and Nios V/g chapters: — Updated the fMAX and Logic Size in the Table: *Processor Performance Benchmarks* for Quartus Prime Pro Edition. — Updated the toolchain version in the Table: *Benchmark Parameters forQuartus Prime Software*. — *Instruction Manager Port*. — *Data Manager Port*. — *Error Correction Code*. — *Instruction Set Reference*.

continued...

Document Version	Quartus Prime Version	Changes
2024.04.01	24.1	• Updated the following tables: — *Nios V/c Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software.* — *Benchmark Parameters for Quartus Prime Software* for the Nios V/c, Nios V/m, and Nios V/g processors. — *Instruction Interface Signals* for the Nios V/c, Nios V/m, and Nios V/g processors. — *Data Interface Signals* for the Nios V/c, Nios V/m, and Nios V/g processors. — *Nios V/m Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software*. — *Nios V/g Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Software*.
2023.12.11	23.4	• Updated all tables related to *Processor Performance Benchmarks*. • Edited *Nios V/c Processor Core Block Diagram*. • Revised instruction set from RV32IAZicsr to RV32IZicsr. • Updated *Accessing Tightly Coupled Memory*. • Updated the note in *Using Tightly Coupled Memory Effectively* to revise the support for Instruction TCM.

continued...

Document Version	Quartus Prime Version	Changes
2023.10.02	23.3	• Added Nios V/c processor in the table *Nios V Processor Variants*. • Added new section *Nios V/c Processor*. • Updated section *Nios V/m Processor*: — Added new sub-topics *Pipelined* and *Non-pipelined* in the topics *Processor Performance Benchmarks* and *Processor Pipeline*. — Updated table *Nios V/m Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Pro Edition Software*. — Added ECC module in figure *Nios V/m Processor Core Block Diagram*. — Updated tables *Instruction Interface Signals* and *Data Interface Signals*. — Added new topic *Error Correction Code (ECC)*. • Updated section *Nios V/g Processor*: — Updated table *Nios V/g Processor Performance Benchmarks in Intel FPGA Devices for Quartus Prime Pro Edition Software*. — Added ECC module, Instruction Tightly Coupled Memory, Data Tightly Coupled Memory, and Floating Point Unit in figure *Nios V/g Processor Core Block Diagram*. — Added the following topics: • *Floating-Point Unit* • *IEEE 754 Exception Conditions* • *Floating Point Operations* • *Tightly Coupled Memory* • *Instruction and Data Tightly-Coupled Memory* • *Accessing Tightly-Coupled Memory* • *Using Tightly-Coupled Memory Effectively* • *Error Correction Code (ECC)* — Updated tables *Instruction Interface Signals* and *Data Interface Signals*. — Added Floating-Point CSRs in table *Control and Status Registers List*. — Added *Floating-Point CSR Register Fields* in *Control and Status Register Field*.
2023.05.26	23.1	Added a link to *AN 980: Nios V Processor Quartus Prime Software Support.*
2023.04.14	23.1	• Updated *Nios V/m Processor Performance Benchmarks*. • Added a new section *Nios V/g Processor*. • Updated product family name to "Intel Agilex® 7".

Document Version	Quartus Prime Version	IP Version	Changes
2022.10.31	22.1std	1.0.0	• Updated references from *Quartus Prime Pro Edition* to Quartus Prime to indicate support for both Pro and Standard Edition. • Added Table: *Nios V/m Processor Performance Benchmarks in Intel FPGA Devices for Intel Quartus Prime Standard Edition* in *Processor Performance Benchmarks* section.
2022.09.26	22.3	22.3.0	• Updated the values in Table: *Nios V/m Processor Performance Benchmarks in Intel FPGA Devices*. • Replaced Table: *Reset and Debug Signals* with new signals, types, and descriptions. • Updated the step to set exception address in section *Exception Controller*.

continued...

Document Version	Quartus Prime Version	IP Version	Changes
2022.08.01	22.2	21.3.0	• Edited the performance metric value in Table: *Architecture Performance*. • Added new sections — *Trigger* — *Typical Use Cases* • Edited the following sections: — *Reset and Debug Signals* — *RISC-V based Debug Module* — *Debug Mode* — *Halt from Debug Module* — *Control and Status Registers (CSR) Mapping* — *Control and Status Register Field*
2022.06.30	22.1	21.2.0	Added new section *Reset and Debug Signals*.
2022.03.28	21.4	21.1.1	Updated *RISC-V based Debug Module* section with details for Nios V processor.
2021.12.13	21.4	21.1.1	Updated IP version and Quartus Prime version.
2021.11.15	21.3	21.1.0	Edited Table: *Architecture Performance* in Section: *Processor Performance Benchmarks*. • Change *CoreMark* to *CoreMark/MHz Ratio* and updated the value to *0.32148*.
2021.10.04	21.3	21.1.0	Initial release.