PU-RISCV WIKI

A Processing Unit (PU) is an electronic system within a computer that carries out instructions of a program by performing the basic arithmetic, logic, controlling, and I/O operations specified by instructions. Instruction-level parallelism is a measure of how many instructions in a computer can be executed simultaneously. The PU is contained on a single Metal Oxide semiconductor (MOS) Integrated Circuit (IC).

The RISC-V implementation has a 32/64/128 bit Microarchitecture, 6 stages data pipeline and an Instruction Set Architecture based on Reduced Instruction Set Computer. Compatible with AMBA and Wishbone Buses. For Researching and Developing.

Processing Unit	Module description
riscv_pu	Processing Unit
...riscv_core	Core
...riscv_imem_ctrl	Instruction Memory Access Block
...riscv_biu - imem	Bus Interface Unit (Instruction)
...riscv_dmem_ctrl	Data Memory Access Block
...riscv_biu - dmem	Bus Interface Unit (Data)

Open Source Tools

Verilator

Hardware Description Language SystemVerilog Simulator

```
git clone http://git.veripool.org/git/verilator
```

```
cd verilator
autoconf
./configure
make
sudo make install
```

```
cd sim/verilog/regression/wb/vtor
source SIMULATE-IT
```

```
cd sim/verilog/regression/ahb3/vtor
source SIMULATE-IT
```

Icarus Verilog

Hardware Description Language Verilog Simulator

```
git clone https://github.com/steveicarus/iverilog
```

```
cd iverilog
./configure
make
sh autoconf.sh
sudo make install
```

```
cd sim/verilog/regression/wb/iverilog
source SIMULATE-IT
```

```
cd sim/verilog/regression/ahb3/iverilog
source SIMULATE-IT
```

GHDL

Hardware Description Language GHDL Simulator

```
git clone https://github.com/ghdl/ghdl
```

```
cd ghdl
./configure --prefix=/usr/local
make
sudo make install
```

```
cd sim/vhdl/regression/wb/ghdl
source SIMULATE-IT
```

```
cd sim/vhdl/regression/ahb3/ghdl
source SIMULATE-IT
```

Yosys-ABC

Hardware Description Language Verilog Synthesizer

```
git clone https://github.com/YosysHQ/yosys
```

```
cd yosys
make
sudo make install
```

```
cd synthesis/yosys
source SIMULATE-IT
```

CORE-RISCV

CORE-RISCV Organization

The CORE-RISCV is based on the Harvard architecture, which is a computer architecture with separate storage and signal pathways for instructions and data. A Harvard architecture machine has distinct code and data address spaces: instruction address zero is not the same as data address zero. Instruction address zero might identify a twenty-four-bit value, while data address zero might indicate an eight-bit byte that is not part of that twenty-four-bit value.

Core	Module description
riscv_core	Core
...riscv_if	Instruction Fetch
...riscv_id	Instruction Decoder
...riscv_execution	Execution Unit
.....riscv_alu	Arithmetic & Logical Unit
.....riscv_lsu	Load Store Unit
.....riscv_bu	Branch Unit
.....riscv_mul	Multiplier Unit
.....riscv_div	Division Unit
...riscv_memory	Memory Unit
...riscv_wb	Data Memory Access (Write Back)
...riscv_state	State Unit
...riscv_rf	Register File
...riscv_bp	Correlating Branch Prediction Unit
.....riscv_ram_1r1w	RAM 1RW1
.......riscv_ram_1r1w_generic	RAM 1RW1 Generic
...riscv_du	Debug Unit

In a Harvard architecture, there is no need to make the two memories share characteristics. In particular, the word width, timing, implementation technology, and memory address structure can differ. In some systems, instructions for pre-programmed tasks can be stored in read-only memory while data memory generally requires read-write memory. In some systems, there is much more instruction memory than data memory so instruction addresses are wider than data addresses.

Instruction INPUTS/OUTPUTS Bus

Port	Size	Direction	Description
`ins_stb`	1	Input	Strobe

Port	Size	Direction	Description
`ins_stb_ack`	1	Output	Strobe acknowledge
`ins_d_ack`	1	Output	Data acknowledge
`ins_adri`	`PLEN`	Input	Start address
`ins_adro`	`PLEN`	Output	Response address
`ins_size`	3	Input	Syze
`ins_type`	3	Input	Type
`ins_prot`	3	Input	Protection
`ins_lock`	1	Input	Locked access
`ins_d`	`XLEN`	Input	Write data
`ins_q`	`XLEN`	Output	Read data
`ins_ack`	1	Output	Acknowledge
`ins_err`	1	Output	Error

Data INPUTS/OUTPUTS Bus

Port	Size	Direction	Description
`dat_stb`	1	Input	Strobe
`dat_stb_ack`	1	Output	Strobe acknowledge
`dat_d_ack`	1	Output	Data acknowledge
`dat_adri`	`PLEN`	Input	Start address
`dat_adro`	`PLEN`	Output	Response address
`dat_size`	3	Input	Syze
`dat_type`	3	Input	Type
`dat_prot`	3	Input	Protection
`dat_lock`	1	Input	Locked access
`dat_d`	`XLEN`	Input	Write data
`dat_q`	`XLEN`	Output	Read data
`dat_ack`	1	Output	Acknowledge
`dat_err`	1	Output	Error

RISC PIPELINE

In computer science, instruction pipelining is a technique for implementing instruction-level parallelism within a PU. Pipelining attempts to keep every part of the processor busy with some instruction by dividing incoming instructions into a series of sequential steps performed by different PUs with different parts of instructions processed in parallel. It allows faster PU throughput than would otherwise be possible at a given clock rate.

Typical	Modified	Module
FETCH	FETCH	riscv_if

Typical	Modified	Module
...	PRE-DECODE	riscv_id
DECODE	DECODE	riscv_id
EXECUTE	EXECUTE	riscv_execution
MEMORY	MEMORY	riscv_memory
WRITE-BACK	WRITE-BACK	riscv_wb

- IF – Instruction Fetch Unit : Send out the PC and fetch the instruction from memory into the Instruction Register (IR); increment the PC to address the next sequential instruction. The IR is used to hold the next instruction that will be needed on subsequent clock cycles; likewise the register NPC is used to hold the next sequential PC.
- ID – Instruction Decode Unit : Decode the instruction and access the register file to read the registers. This unit gets instruction from IF, and extracts opcode and operand from that instruction. It also retrieves register values if requested by the operation.
- EX – Execution Unit : The ALU operates on the operands prepared in prior cycle, performing one functions depending on instruction type.
- MEM – Memory Access Unit: Instructions active in this unit are loads, stores and branches.
- WB – WriteBack Unit : Write the result into the register file, whether it comes from the memory system or from the ALU.

INSTRUCTION & DATA CACHE

A PU cache is a hardware cache used by the PU to reduce the average cost (time or energy) to access instruction/data from the main memory. A cache is a smaller, faster memory, closer to a core, which stores copies of the data from frequently used main memory locations. Most CPUs have different independent caches, including instruction and data caches.

Instruction Organization

Instruction Memory	Module description
riscv_imem_ctrl	Instruction Memory Access Block
...riscv_membuf	Memory Access Buffer
.....riscv_ram_queue	Fall-through Queue
...riscv_memmisaligned	Misalignment Check
...riscv_mmu	Memory Management Unit
...riscv_pmachk	Physical Memory Attributes Checker

Instruction Memory	Module description
...riscv_pmpchk	Physical Memory Protection Checker
...riscv_icache_core	Instruction Cache (Write Back)
.....riscv_ram_1rw	RAM 1RW
.......riscv_ram_1rw_generic	RAM 1RW Generic
...riscv_dext	Data External Access Logic
...riscv_ram_queue	Fall-through Queue
...riscv_mux	Bus-Interface-Unit Mux
riscv_biu	Bus Interface Unit

Instruction INPUTS/OUTPUTS AMBA4 AXI-Lite Bus

Signals of the Read and Write Address channels

Write Port	Read Port	Size	Direction	Description
`AWID`	`ARID`	`AXI_ID_WIDTH`	Output	Address ID, to identify multiple streams
`AWADDR`	`ARADDR`	`AXI_ADDR_WIDTH`	Output	Address of the first beat of the burst
`AWLEN`	`ARLEN`	8	Output	Number of beats inside the burst
`AWSIZE`	`ARSIZE`	3	Output	Size of each beat
`AWBURST`	`ARBURST`	2	Output	Type of the burst
`AWLOCK`	`ARLOCK`	1	Output	Lock type, to provide atomic operations
`AWCACHE`	`ARCACHE`	4	Output	Memory type, progress through the system
`AWPROT`	`ARPROT`	3	Output	Protection type
`AWQOS`	`ARQOS`	4	Output	Quality of Service of the transaction
`AWREGION`	`ARREGION`	4	Output	Region identifier, physical to logical
`AWUSER`	`ARUSER`	`AXI_USER_WIDTH`	Output	User-defined data
`AWVALID`	`ARVALID`	1	Output	xVALID handshake signal
`AWREADY`	`ARREADY`	1	Input	xREADY handshake signal

Signals of the Read and Write Data channels

Write Port	Read Port	Size	Direction	Description
`WID`	`RID`	`AXI_ID_WIDTH`	Output	Data ID, to identify multiple streams
`WDATA`	`RDATA`	`AXI_DATA_WIDTH`	Output	Read/Write data
`--`	`RRESP`	2	Output	Read response, current RDATA status
`WSTRB`	`--`	`AXI_STRB_WIDTH`	Output	Byte strobe, WDATA signal
`WLAST`	`RLAST`	1	Output	Last beat identifier
`WUSER`	`RUSER`	`AXI_USER_WIDTH`	Output	User-defined data
`WVALID`	`RVALID`	1	Output	xVALID handshake signal
`WREADY`	`RREADY`	1	Input	xREADY handshake signal

Signals of the Write Response channel

Write Port	Size	Direction	Description
BID	AXI_ID_WIDTH	Input	Write response ID, to identify multiple streams
BRESP	2	Input	Write response, to specify the burst status
BUSER	AXI_USER_WIDTH	Input	User-defined data
BVALID	1	Input	xVALID handshake signal
BREADY	1	Output	xREADY handshake signal

Instruction INPUTS/OUTPUTS AMBA3 AHB-Lite Bus

Port	Size	Direction	Description
HRESETn	1	Input	Asynchronous Active Low Reset
HCLK	1	Input	System Clock Input
IHSEL	1	Output	Instruction Bus Select
IHADDR	PLEN	Output	Instruction Address Bus
IHRDATA	XLEN	Input	Instruction Read Data Bus
IHWDATA	XLEN	Output	Instruction Write Data Bus
IHWRITE	1	Output	Instruction Write Select
IHSIZE	3	Output	Instruction Transfer Size
IHBURST	3	Output	Instruction Transfer Burst Size
IHPROT	4	Output	Instruction Transfer Protection Level
IHTRANS	2	Output	Instruction Transfer Type
IHMASTLOCK	1	Output	Instruction Transfer Master Lock
IHREADY	1	Input	Instruction Slave Ready Indicator
IHRESP	1	Input	Instruction Transfer Response

Instruction INPUTS/OUTPUTS Wishbone Bus

Port	Size	Direction	Description
rst	1	Input	Synchronous Active High Reset
clk	1	Input	System Clock Input
iadr	AW	Input	Instruction Address Bus
idati	DW	Input	Instruction Input Bus
idato	DW	Output	Instruction Output Bus
isel	DW/8	Input	Byte Select Signals
iwe	1	Input	Write Enable Input
istb	1	Input	Strobe Signal/Core Select Input

Port	Size	Direction	Description
icyc	1	Input	Valid Bus Cycle Input
iack	1	Output	Bus Cycle Acknowledge Output
ierr	1	Output	Bus Cycle Error Output
iint	1	Output	Interrupt Signal Output

Data Organization

Data Memory	Module description
riscv_dmem_ctrl	Data Memory Access Block
...riscv_membuf	Memory Access Buffer
.....riscv_ram_queue	Fall-through Queue
...riscv_memmisaligned	Misalignment Check
...riscv_mmu	Memory Management Unit
...riscv_pmachk	Physical Memory Attributes Checker
...riscv_pmpchk	Physical Memory Protection Checker
...riscv_dcache_core	Data Cache (Write Back)
.....riscv_ram_1rw	RAM 1RW
.......riscv_ram_1rw_generic	RAM 1RW Generic
...riscv_dext	Data External Access Logic
...riscv_mux	Bus-Interface-Unit Mux
riscv_biu	Bus Interface Unit

Data INPUTS/OUTPUTS AMBA4 AXI-Lite Bus

Signals of the Read and Write Address channels

Write Port	Read Port	Size	Direction	Description
AWID	ARID	AXI_ID_WIDTH	Output	Address ID, to identify multiple streams
AWADDR	ARADDR	AXI_ADDR_WIDTH	Output	Address of the first beat of the burst
AWLEN	ARLEN	8	Output	Number of beats inside the burst
AWSIZE	ARSIZE	3	Output	Size of each beat
AWBURST	ARBURST	2	Output	Type of the burst
AWLOCK	ARLOCK	1	Output	Lock type, to provide atomic operations
AWCACHE	ARCACHE	4	Output	Memory type, progress through the system
AWPROT	ARPROT	3	Output	Protection type
AWQOS	ARQOS	4	Output	Quality of Service of the transaction
AWREGION	ARREGION	4	Output	Region identifier, physical to logical
AWUSER	ARUSER	AXI_USER_WIDTH	Output	User-defined data
AWVALID	ARVALID	1	Output	xVALID handshake signal
AWREADY	ARREADY	1	Input	xREADY handshake signal

Signals of the Read and Write Data channels

Write Port	Read Port	Size	Direction	Description
`WID`	`RID`	`AXI_ID_WIDTH`	Output	Data ID, to identify multiple streams
`WDATA`	`RDATA`	`AXI_DATA_WIDTH`	Output	Read/Write data
`--`	`RRESP`	2	Output	Read response, current RDATA status
`WSTRB`	`--`	`AXI_STRB_WIDTH`	Output	Byte strobe, WDATA signal
`WLAST`	`RLAST`	1	Output	Last beat identifier
`WUSER`	`RUSER`	`AXI_USER_WIDTH`	Output	User-defined data
`WVALID`	`RVALID`	1	Output	xVALID handshake signal
`WREADY`	`RREADY`	1	Input	xREADY handshake signal

Signals of the Write Response channel

Write Port	Size	Direction	Description
`BID`	`AXI_ID_WIDTH`	Input	Write response ID, to identify multiple streams
`BRESP`	2	Input	Write response, to specify the burst status
`BUSER`	`AXI_USER_WIDTH`	Input	User-defined data
`BVALID`	1	Input	xVALID handshake signal
`BREADY`	1	Output	xREADY handshake signal

Data INPUTS/OUTPUTS AMBA3 AHB-Lite Bus

Port	Size	Direction	Description
`HRESETn`	1	Input	Asynchronous Active Low Reset
`HCLK`	1	Input	System Clock Input
`DHSEL`	1	Output	Data Bus Select
`DHADDR`	`PLEN`	Output	Data Address Bus
`DHRDATA`	`XLEN`	Input	Data Read Data Bus
`DHWDATA`	`XLEN`	Output	Data Write Data Bus
`DHWRITE`	1	Output	Data Write Select
`DHSIZE`	3	Output	Data Transfer Size
`DHBURST`	3	Output	Data Transfer Burst Size
`DHPROT`	4	Output	Data Transfer Protection Level
`DHTRANS`	2	Output	Data Transfer Type
`DHMASTLOCK`	1	Output	Data Transfer Master Lock
`DHREADY`	1	Input	Data Slave Ready Indicator
`DHRESP`	1	Input	Data Transfer Response

Data INPUTS/OUTPUTS Wishbone Bus

Port	Size	Direction	Description
`rst`	1	Input	Synchronous Active High Reset
`clk`	1	Input	System Clock Input
`dadr`	`AW`	Input	Data Address Bus
`ddati`	`DW`	Input	Data Input Bus
`ddato`	`DW`	Output	Data Output Bus
`dsel`	`DW/8`	Input	Byte Select Signals
`dwe`	1	Input	Write Enable Input
`dstb`	1	Input	Strobe Signal/Core Select Input
`dcyc`	1	Input	Valid Bus Cycle Input
`dack`	1	Output	Bus Cycle Acknowledge Output
`derr`	1	Output	Bus Cycle Error Output
`dint`	1	Output	Interrupt Signal Output

Parameters

Parameter	Type	Default	Description
`JEDEC_BANK`	`Integer`	0x0A	JEDEC Bank
`JEDEC_MANUFACTURER_ID`	`Integer`	0x6E	JEDEC Manufacturer ID
`XLEN`	`Integer`	64	Data Path Width
`PLEN`	`Integer`	64	Physical Memory Address Size
`PMP_CNT`	`Integer`	16	Physical Memory Protection Entries
`PMA_CNT`	`Integer`	16	Physical Menory Attribute Entries
`HAS_USER`	`Integer`	1	User Mode Enable
`HAS_SUPER`	`Integer`	1	Supervisor Mode Enable
`HAS_HYPER`	`Integer`	1	Hypervisor Mode Enable
`HAS_RVM`	`Integer`	1	“M” Extension Enable
`HAS_RVA`	`Integer`	1	“A” Extension Enable
`HAS_RVC`	`Integer`	1	“C” Extension Enable
`HAS_BPU`	`Integer`	1	Branch Prediction Unit Control Enable
`IS_RV32E`	`Integer`	0	Base Integer Instruction Set Enable
`MULT_LATENCY`	`Integer`	1	Hardware Multiplier Latency
`ICACHE_SIZE`	`Integer`	16	Instruction Cache size
`ICACHE_BLOCK_SIZE`	`Integer`	64	Instruction Cache block length
`ICACHE_WAYS`	`Integer`	2	Instruction Cache associativity
`ICACHE_REPLACE_ALG`	`Integer`	0	Instruction Cache replacement
`DCACHE_SIZE`	`Integer`	16	Data Cache size
`DCACHE_BLOCK_SIZE`	`Integer`	64	Data Cache block length
`DCACHE_WAYS`	`Integer`	2	Data Cache associativity
`DCACHE_REPLACE_ALG`	`Integer`	0	Data Cache replacement algorithm

Parameter	Type	Default	Description
HARTID	Integer	0	Hart Identifier
PC_INIT	Address	'h200	Program Counter Initialisation Vector
MNMIVEC_DEFAULT	Address	PC_INIT-'h004	Machine Mode Non-Maskable
MTVEC_DEFAULT	Address	PC_INIT-'h040	Machine Mode Interrupt Address
HTVEC_DEFAULT	Address	PC_INIT-'h080	Hypervisor Mode Interrupt Address
STVEC_DEFAULT	Address	PC_INIT-'h0C0	Supervisor Mode Interrupt Address
UTVEC_DEFAULT	Address	PC_INIT-'h100	User Mode Interrupt Address
BP_LOCAL_BITS	Integer	10	Number of local predictor bits
BP_GLOBAL_BITS	Integer	2	Number of global predictor bits
BREAKPOINTS	Integer	3	Number of hardware breakpoints
TECHNOLOGY	String	GENERIC	Target Silicon Technology