# MiniRISC™ CW4010 Superscalar Microprocessor Core

Preliminary Datasheet

## Description

LSI Logic Corporation has developed the MiniRISC CW4010 Superscalar Microprocessor Core, the world's first MIPS-II-compatible superscalar core, using LSI Logic's CoreWare® system-on-a-chip methodology. The CW4010 consists of the superscalar CPU core with an arithmetic logic unit (ALU), a system control coprocessor (CP0), a bus interface unit (BIU), a load store unit (LSU), and an instruction scheduler unit (ISU). Figure 1 shows the core and how it interfaces with LSI Logic's microprocessor building blocks. The following options are available with the basic CPU core: direct-mapped or two-way set associative instruction cache, direct-mapped or two-way set associative data cache, a memory management unit with 64-entry translation lookaside buffer, a standard multiply unit or a high-performance multiply/accumulate unit, and a Writeback Buffer for Writeback cache mode. The cache sizes are selectable up to 16 Kbytes. These options allow the customer to develop a user-defined microprocessor.

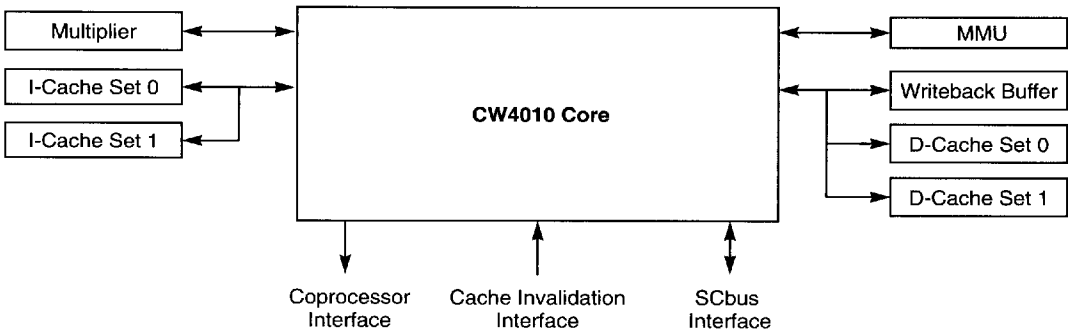


**Figure 1. Microprocessor Core Interface with Building Blocks**

The CPU can issue and retire two instructions per cycle using a combination of five independent execution units. The CW4010 is fully compatible with the MIPS-I and MIPS-II instruction sets, but it uses an updated superscalar architecture to provide higher performance than any other available MIPS solution. With a system clock of 80 MHz, its performance is 150 Dhrystone MIPS.

In addition to the core, the MiniRISC product family includes LSI Logic's MiniSIM™ architectural simulator, Verilog and VHDL models, a system verification environment, a PROM Monitor, third party software support, and core bond-out chip for emulation. The CoreWare program consists of three main elements: a library of cores, a design development and simulation package, and applications support. The large and growing CoreWare library contains a wide range of complex cores based on accepted and emerging industry standards, such as high-speed interconnect, Oak DSP, and image and video compression. LSI Logic provides a complete framework for device and system development and simulation. LSI Logic's advanced ASIC technologies consistently produce Right-First-Time™ silicon. LSI Logic's in-house experts provide design support from system architecture definition through chip layout and test vector generation.



Order Number C15000
November 1994

LSI LOGIC

# MiniRISC™ CW4010 Superscalar Microprocessor Core

Preliminary Datasheet

## Features

- R4000 MIPS-II 32-bit instruction set implementation
- Instruction set extensions to support embedded applications
- Superscalar execution: two instructions issued per clock cycle
- Customer-definable, modular design
- High-performance coprocessor interface for user-definable coprocessors and high-performance hardware FPU
- Integrated cache controllers with separate instruction and data caches
- Optional microprocessor building blocks: Writeback Buffer, multiplier, caches, and MMU
- 64-bit memory and cache interface
- 3.3-volt operation
- Implementation of full scan to achieve 99% fault coverage
- 80-MHz worst case commercial maximum clock rate using high-performance 0.5-micron process
- 150 Dhrystone MIPS performance at 80 MHz
- Verilog and VHDL models available
- MIPS and third party software development tool support, such as compilers, assemblers, debuggers, and real-time operating systems

## Block Diagram

Figure 2 is a block diagram of the Superscalar Microprocessor Core. Descriptions of the internal blocks follow the figure.

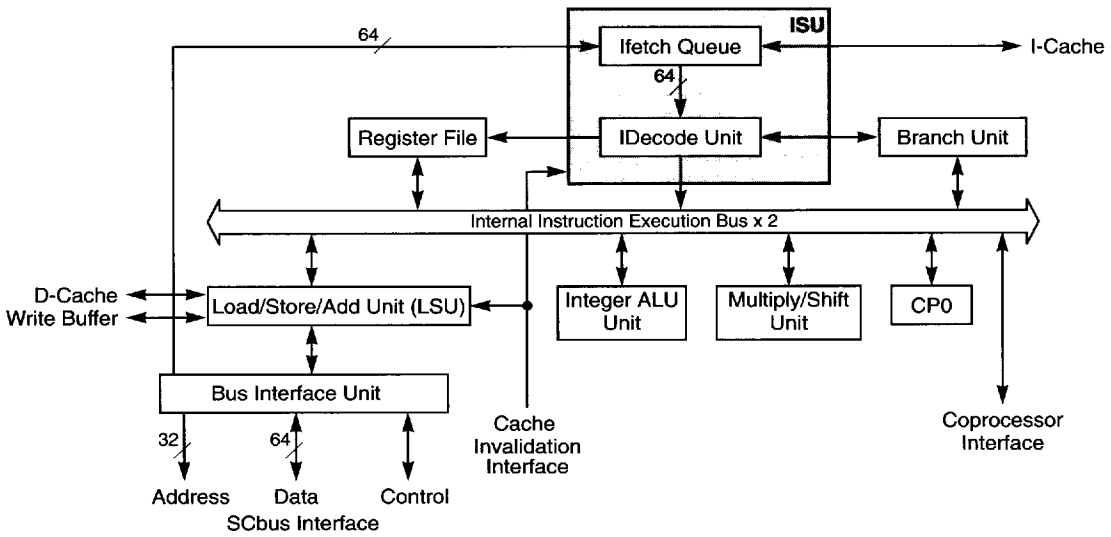


**Figure 2. CW4010 Superscalar Microprocessor Core Block Diagram**

The **Ifetch Queue** optimizes the supply of instructions to the microprocessor, even across breaks in the sequential flow of execution (jumps and branches). The **IDecode Unit** decodes the instructions from the Ifetch Queue, determines the actions required for the instruction execution, and manages the RFile, LSU, ALU, and Multiplier Units accordingly. The **Branch Unit** is used when branch and jump instructions are recognized within the instruction stream.

The **Register File** contains the core's general purpose registers. It supplies source operands to the execution units and handles the storage of results to target registers.

Three units perform logical, arithmetic, and data-movement operations. The **Load/Store/Add Unit (LSU)** manages loads and stores of data values. Loads come from either the D-Cache or the SCbus Interface in the event of a D-Cache miss. Stores pass to the D-Cache and the SCbus Interface through the Write Buffer. The LSU also performs a restricted set of arithmetic operations, including the addition of an immediate offset as required in address calculations. The **Integer ALU Unit** calculates the result of an arithmetic or logical operation. The **Multiplier/Shift Unit** performs multiply and divide operations. The customer has a selection of functional options for this unit, including an option with full multiply/accumulate capability.

The **Bus Interface Unit** manages the flow of instructions and data between the core and the system via the SCbus Interface.

The **SCbus Interface** provides the main channel for communication between the CW4010 core and the other functional blocks in the system. Some blocks may be implemented as CoreWare library functions integrated on the same die as the Microprocessor Core; others may be implemented in separate devices connected via I/O pins at board level.

The **Coprocessor Interface** allows the attachment of tightly coupled special-purpose processing units, to enhance the microprocessor's general-purpose computational power. Using this approach, high-performance application-specific hardware can be made directly accessible to the programmer at the instruction set level. For example, a coprocessor might offer accelerated bit-mapped graphics operations or real-time video decompression.

The **Cache Invalidation Interface** allows supporting hardware outside the Microprocessor Core to maintain the coherency of on-board cache contents for systems that include multiple main-bus masters.

## Pipeline Architecture

Figure 3 shows the CW4010 core's six-stage pipelines. The superscalar CW4010 has two concurrent pipelines—an even and an odd. The first three stages are labelled the instruction fetch phase, and the last three stages are labelled the instruction execution phase.

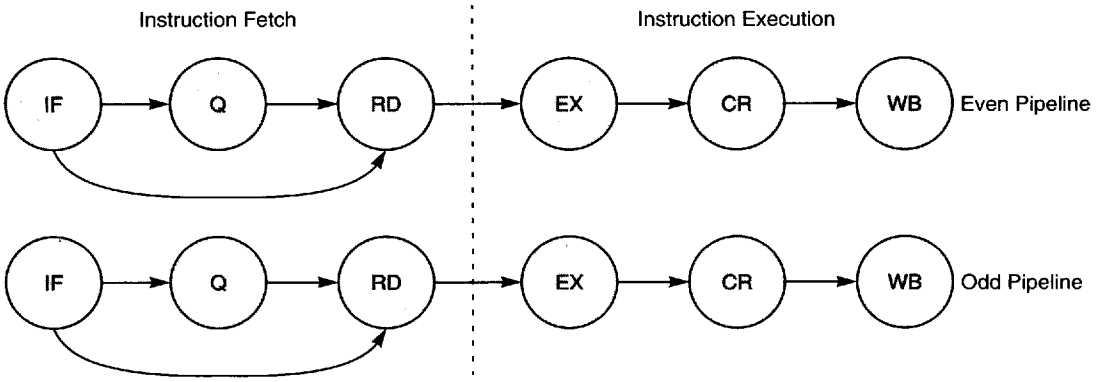


**Figure 3. CW4010 Instruction Pipeline**

In general, the execution of a single CW4010 instruction consists of the following stages:

1. IF (Instruction Fetch) – The CW4010 fetches the instruction during the first stage.
2. Q (Queuing) – Instructions may enter this conditional stage if they deal with branches or register conflicts. An instruction that does not cause a branch or register conflict is fed directly to the RD stage.
3. RD (Read) – During this stage, any required operands are read from the Register File while the instruction is decoded.
4. EX (Execute) – All instructions are executed in this stage. Conditional branches are resolved in this cycle. The address calculation for load and store instructions is performed in this stage.
5. CR (Cache Read) – This stage is used to read the cache for load and store instructions. Data is returned to the register bypass logic at the end of this stage.
6. WB (Writeback) – Results are written into the Register File during this cycle.

Each stage, once it has accepted an instruction from the previous stage, can hold the instruction for re-execution in case the pipeline stalls.

## Instruction Set Summary

Table 1 summarizes the instruction set for the CW4010. The CW4010 supports both MIPS-I and MIPS-II instructions, and also implements some additional CW4010-specific instructions. If the design includes the optional MMU, then the CW4010 supports the TLB instructions. All instructions are 32 bits long. In the table, the MIPS-II, CW4010-specific, and TLB instructions are flagged to distinguish them from the MIPS-I instructions.

## Table 1. CW4010 Instruction Set Summary

| Op | Description |
|---|---|
| **Load/Store Instructions** | |
| LB | Load Byte |
| LBU | Load Byte Unsigned |
| LH | Load Halfword |
| LHU | Load Halfword Unsigned |
| LW | Load Word |
| LWL | Load Word Left |
| LWR | Load Word Right |
| SB | Store Byte |
| SH | Store Halfword |
| SW | Store Word |
| SWL | Store Word Left |
| SWR | Store Word Right |
| LL[1] | Load Linked |
| SC[1] | Store Conditional |
| SYNC[1] | Sync |
| **Arithmetic Instructions: ALU Immediate** | |
| ADDI | Add Immediate |
| ADDIU | Add Immediate Unsigned |
| SLTI | Set on Less Than Immediate |
| SLTIU | Set on Less Than Immediate Unsigned |
| ANDI | AND Immediate |
| ORI | OR Immediate |
| XORI | Exclusive OR Immediate |
| LUI | Load Upper Immediate |
| **Arithmetic Instructions: Three-Operand, Register-Type** | |
| ADD | Add |
| ADDU | Add Unsigned |
| SUB | Subtract |
| SUBU | Subtract Unsigned |
| SLT | Set on Less Than |
| SLTU | Set on Less Than Unsigned |
| AND | AND |
| OR | OR |
| XOR | Exclusive OR |
| NOR | NOR |
| **System Control Coprocessor (CP0) Instructions** | |
| MTC0 | Move To CP0 |
| MFC0 | Move From CP0 |
| RFE | Restore From Exception |
| TLBR[2] | Read Indexed TLB Entry |
| TLBWI[2] | Write Indexed TLB Entry |
| TLBWR[2] | Write Random TLB Entry |
| TLBP[2] | Probe TLB for Matching Entry |
| WAITI[3] | Wait for Interrupt |
| **Multiply/Divide Instructions** | |
| MULT | Multiply |
| MULTU | Multiply Unsigned |
| DIV | Divide |
| DIVU | Divide Unsigned |
| MFHI | Move From HI |
| MTHI | Move To HI |
| MFLO | Move From LO |
| MTLO | Move To LO |
| **Shift Instructions** | |
| SLL | Shift Left Logical |
| SRL | Shift Right Logical |
| SRA | Shift Right Arithmetic |
| SLLV | Shift Left Logical Variable |
| SRLV | Shift Right Logical Variable |
| SRAV | Shift Right Arithmetic Variable |

| Op | Description |
|---|---|
| **Other Computational Instructions** | |
| ADDCIU[3] | Add Circular Immediate |
| FFS[3] | Find First Set |
| FFC[3] | Find First Clear |
| SELSR[3] | Select and Shift Right |
| SELSL[3] | Select and Shift Left |
| MADD[3] | Multiply/Add |
| MADDU[3] | Multiply/Add Unsigned |
| MSUB[3] | Multiply/Subtract |
| MSUBU[3] | Multiply/Subtract Unsigned |
| **Jump and Branch Instructions** | |
| J | Jump |
| JAL | Jump And Link |
| JR | Jump Register |
| JALR | Jump And Link Register |
| BEQ | Branch on Equal |
| BNE | Branch on Not Equal |
| BLEZ | Branch on Less than or Equal to Zero |
| BGTZ | Branch on Greater Than Zero |
| BLTZ | Branch on Less Than Zero |
| BGEZ | Branch on Greater than or Equal to Zero |
| BLTZAL | Branch on Less Than Zero And Link |
| BGEZAL | Branch on Greater than or Equal to Zero And Link |
| **Branch Likely Instructions** | |
| BEQL[1] | Branch on Equal Likely |
| BNEL[1] | Branch on Not Equal Likely |
| BLEZL[1] | Branch on Less than or Equal to Zero Likely |
| BGTZL[1] | Branch on Greater Than Zero Likely |
| BLTZL[1] | Branch on Less Than Zero Likely |
| BGEZL[1] | Branch on Greater than or Equal to Zero Likely |
| BLTZALL[1] | Branch on Less Than Zero And Link Likely |
| BGEZALL[1] | Branch on Greater than or Equal to Zero and Link Likely |
| BCzTL[1] | Branch on Coprocessor z True Likely |
| BCzFL[1] | Branch on Coprocessor z False Likely |
| **Coprocessor Instructions** | |
| LWCz | Load Word to Coprocessor z |
| SWCz | Store Word from Coprocessor z |
| MTCz | Move to Coprocessor z |
| MFCz | Move From Coprocessor z |
| CTCz | Move Control to Coprocessor z |
| CFCz | Move Control from Coprocessor z |
| COPz | Coprocessor Operation |
| BCzT | Branch on Coprocessor z True |
| BCzF | Branch on Coprocessor z False |
| **Trap Instructions** | |
| TEQ[1] | Trap on Equal |
| TEQI[1] | Trap on Equal Immediate |
| TGE[1] | Trap on Greater than or Equal |
| TGEI[1] | Trap on Greater than or Equal Immediate |
| TGEU[1] | Trap on Greater than or Equal Unsigned |
| TGEIU[1] | Trap on Greater than or Equal Immediate Unsigned |
| TLT[1] | Trap on Less Than |
| TLTI[1] | Trap on Less Than Immediate |
| TLTU[1] | Trap on Less Than Unsigned |
| TLTIU[1] | Trap on Less Than Immediate Unsigned |
| **Special Instructions** | |
| SYSCALL | System Call |
| BREAK | Breakpoint |

1. MIPS-II instruction.
2. Valid only with implemented MMU building block.
3. CW4010-specific instruction.

## Signal Descriptions

This section describes the CW4010's interface to logic external to the core. This section contains the following subsections:

- Reset and Interrupt Signals
- SCbus Interface Signals
- Cache Invalidation Interface Signals
- Coprocessor Interface Signals
- Miscellaneous Signals

Within each subsection, the signals are described in alphabetical order by mnemonic. Each signal definition contains the mnemonic and the full signal name. The mnemonics for signals that are active LOW end in an "n," and the mnemonics for signals that are active HIGH end in a "p."

In the descriptions that follow, the verb assert means to drive TRUE or active. The verb deassert means to drive FALSE or inactive.

### Reset and Interrupt Signals

**CRESETn** — **Cold System Reset** — **Input**

The system asserts CRESETn to reset the CW4010. The assertion can be asynchronous to the SCLKp rising edge, but the deassertion must be synchronous to the rising edge of SCLKp. Assertion of this input initializes the Microprocessor Core; no assumptions are made about the previous internal state, and no attempt is made to preserve any part of it. Internally, a reset is handled as a form of exception, and has the highest priority of all such conditions. After CRESETn is deasserted, CP0 generates a cold reset exception (virtual address 0xBFC0.0000).

**EXiNTn[5:0]** — **External Interrupts** — **Input**

External logic asserts these signals to cause the CW4010 to take an interrupt exception. The states of these inputs are reflected in the IP[5:0] field of the Cause Register. Consequently, the interrupting logic should continue to assert the external interrupt input until the exception routine has serviced the interrupt.

To individually disable or mask the interrupt inputs, set the appropriate bit in the Status Register.

External interrupts are not recognized if the interrupt enable bit in the Status Register is cleared. However, the IP bits of the Status Register show the input conditions.

**NMin** — **Non-Maskable Interrupt** — **Input**

When the CW4010 samples this signal's assertion, CP0 generates a non-maskable interrupt exception (virtual address 0xBFC0.0000).

**WRESETn** — **Warm System Reset** — **Input**

The system asserts this input to perform a partial re-initialization of the Microprocessor Core's internal state. This input is used when a cold reset's complete initialization is unnecessary. WRESETn must be asserted and deasserted synchronously to the system clock's rising edge. While WRESETn is asserted, the CW4010 initializes its internal states. After WRESETn is deasserted, CP0 generates a warm reset exception (virtual address 0xBFC0.0000).

## SCbus Interface Signals

**SCAoEn** — **Address Output Enable** — **Output**
The CW4010 asserts this output to indicate the BIU is performing an SCbus transaction and the address output bus bits SCAop[31:0] are valid. It is asserted throughout the entire transaction.

**SCAop[31:0]** — **Address Output Bus** — **Output**
This 32-bit output is the address output bus for instruction fetches and data reads/writes. The address is valid from the beginning to the end of a transaction cycle provided that either SCBRDYn, SCBRTYn, or SCBERRn is asserted. The SCAop bus is valid only when the address output enable signal SCAoEn is asserted.

**SCB32n** — **32-bit Bus Width Sizing** — **Input**
Assertion of this input indicates the external bus slave on the SCbus needs 32-bit bus sizing. The CW4010 samples SCB32n at the same clock rising edge when SCBRDYn is asserted. If SCB32n is asserted for a 64-bit transaction (meaning a doubleword or a part of a burst transaction), the BIU generates a subsequent 32-bit word transaction and either packs data to 64 bits for a read or unpacks data for a write.

**SCBERRn** — **Bus Error** — **Input**
The system asserts this input to indicate that the current transaction must be terminated unsuccessfully. The CW4010 ignores the assertion of either SCBRDYn or SCBRTYn if they are asserted at the same time as SCBERRn. CP0 generates an exception upon detecting the assertion of SCBERRn.

**SCBPWAn** — **Bus In-Page Write Accept** — **Input**
Assertion of this input indicates the external bus slave on the SCbus accepts in-page write transactions. This signal is sampled on the same rising clock edge that SCBRDYn is asserted. SCTPWn must be asserted for the assertion/deassertion of SCBPWAn to be valid.

**SCBRDYn** — **Bus Ready** — **Input**
This input is asserted when the current transaction is terminated. In the case of a read transaction, the system must present valid read data for sampling on the same rising clock edge as the assertion of SCBRDYn.

**SCBRTYn** — **Bus Retry** — **Input**
The system asserts this signal to indicate that the current transaction cannot be performed successfully at the present time, but should be retried later. When this signal is used to signal unsuccessful termination of a transaction, the normal termination signal SBCRDYn is a "don't care" and is ignored.

**SCDip[63:0]** — **Data Input Bus** — **Input**
This bus is the 64-bit data input bus for instruction fetches and data reads. SCDip[63:0] are sampled at the rising edge of the clock when SCBRDYn is asserted.

**SCDoEn** — **Data Output Enable** — **Output**
Assertion of this output indicates the data output bus bits SCDop[63:0] are valid and that the current transaction is a write transaction. This signal is asserted from the beginning to the end of a write transaction cycle.

**SCDop[63:0]** — **Data Output Bus** — **Output**
This bus is the 64-bit data output bus for data writes and the writeback data of the D-Cache. This bus is valid from the beginning to the end of a write transaction cycle.

**SCHGTn** **Bus Hold Grant** **Output**
The bus hold request is the highest priority during the arbitration. The BIU enters a hold state and asserts the grant signal SCHGTn to indicate the BIU has released SCbus ownership.

**SCHRQn** **Bus Hold Request** **Input**
Assertion of this input indicates an external bus master wants to own the bus. The bus hold request has the highest priority during the arbitration.

**SCiFETn** **Instruction Fetch** **Output**
The CW4010 asserts this output to indicate the BIU is fetching an instruction. This output is provided for monitoring purposes.

**SCLoCKn** **Bus Lock** **Output**
Assertion of this output indicates that the bus is requesting exclusive access to the current target. This signal is asserted when a Load Linked instruction is executed, and stays at the active level until a Store Conditional instruction is executed.

**SCTBEn[7:0]** **Byte Enable** **Output**
Assertion of these signals indicate which bytes are valid on the SCDip[63:0] and SCDop[63:0] data buses. The correspondence between byte enables and the data bus bytes depends on whether the byte ordering is big endian or little endian, as shown in the following table.

| Byte Enable | Corresponding Data Bus Byte (Big Endian) | Corresponding Data Bus Byte (Little Endian) |
|---|---|---|
| SCTBEn7 | [7:0] | [63:56] |
| SCTBEn6 | [15:8] | [55:48] |
| SCTBEn5 | [23:16] | [47:40] |
| SCTBEn4 | [31:24] | [39:32] |
| SCTBEn3 | [39:32] | [31:24] |
| SCTBEn2 | [47:40] | [23:16] |
| SCTBEn1 | [55:48] | [15:8] |
| SCTBEn0 | [63:56] | [7:0] |

**SCTBSTn** **Single/Burst Transaction** **Output**
A HIGH on this signal indicates that the current transaction is either a byte, halfword, tribyte, word, or doubleword operation. A LOW on this output indicates that the transaction is a burst operation (four doublewords).

**SCTPWn** **Next Transaction is In-Page Write** **Output**
The CW4010 asserts this output to indicate that the next transaction will be in the same DRAM page that is defined in the Cache Configuration Register. This signal is asserted throughout any individual write transaction. In-page writes may be performed back-to-back up to a maximum of four transactions, in which case SCTPWn is asserted from the first through the third transactions and deasserted in the last.

**SCTSEn** **Transaction Start Enable** **Input**
Assertion of this input acknowledges the start of a new SCbus transaction. Within the Microprocessor Core, transaction requests are arbitrated only when SCTSEn is

asserted. This signal may be used by the system to insert idle cycles between transactions.

**SCTSSn** **Transaction Start Strobe** **Output**
The core asserts this output to indicate that a transaction has started. The core asserts SCTSSn for one clock cycle at the beginning of the transaction. If a single-cycle transaction is followed immediately by the start of another transaction, SCTSSn is held asserted for two cycles.

### Cache Invalidation Interface Signals

**DCiNVAp[31:5]** **D-Cache Invalidation Address Bus** **Input**
This input bus is the address bus for D-Cache Invalidation. The CW4010 samples this bus when DCiNVSn is asserted.

**DCiNVSn** **D-Cache Invalidation Strobe** **Input**
Assertion of this input indicates the D-Cache Invalidation Address Bus is valid, and the CW4010 needs to start a snooping sequence. If the D-Cache tag is identical to the appropriate upper address bits, the CW4010 invalidates the line.

**iCiNVAp[31:5]** **I-Cache Invalidation Address Bus** **Input**
This input bus is the address bus for I-Cache Invalidation. The CW4010 samples this bus when iCiNVSn is asserted.

**iCiNVSn** **I-Cache Invalidation Strobe** **Input**
Assertion of this input indicates the I-Cache Invalidation Address Bus is valid, and the CW4010 needs to start a snooping sequence. If the I-Cache tag is identical to the appropriate upper address bits, the CW4010 invalidates the line.

### Coprocessor Interface Signals

**CPBUSYn[3:1]** **Coprocessor Busy** **Input**
A coprocessor asserts its respective Coprocessor Busy input to indicate it is temporarily unable to accept new coprocessor operations (for example, because a complex internal operation is in progress). The CW4010 stalls until the coprocessor deasserts the Coprocessor Busy signal.

**CPCoDEp[31:0]** **Coprocessor Instruction Code Bus** **Output**
This bus outputs the instruction opcode to the coprocessor. It is valid when one of the CPXSTBn signals is asserted.

**CPCoNDn[3:0]** **Coprocessor Condition** **Input**
The CW4010 samples these inputs when executing Coprocessor Conditional Branch instructions. CPCoNDn3 is associated with Coprocessor 3 instructions, CPCoNDn2 is associated with Coprocessor 2 instructions, and CPCoNDn1 is associated with Coprocessor 1 instructions. CPCoNDn0 is available for use as a general-purpose input; it is not pre-allocated for CP0 as in various other MIPS implementations.

**CPFRCDp[31:0]** **Data from Coprocessor** **Input**
This bus inputs data from a coprocessor register to a CPU general-purpose register or memory. It is valid when the data enable signal CPFRCEn is asserted.

**CPFRCEn** **Data from Coprocessor Enable** **Output**
This signal indicates when the data input bus CPFRCDp[31:0] is valid.

**CPRSTn[3:1]** **Coprocessor Reset** **Output**
These outputs indicate the condition of CU bits [3:1] of the Status Register in the CP0. If the CU bit is zero, the corresponding CPRSn output is asserted LOW. These outputs are asserted LOW when the cold reset is asserted as the CU bits are cleared. The CU bits are not cleared when the warm reset is asserted. Software uses these CPRSTn outputs as indicators to reset the coprocessors.

**CPToCDp[31:0]** **Data to Coprocessor** **Output**
This bus outputs data to a coprocessor register from a CPU general-purpose register or memory. It is valid when the data enable signal CPToCEn is asserted.

**CPToCEn** **Data to Coprocessor Enable** **Output**
This signal indicates when the data output bus CPToCDp[31:0] is valid.

**CPXoDDn** **Coprocessor Instruction at Odd Slot** **Input**
Coprocessors use this signal in conjunction with PCANCRn and PCANoDDn to determine the correct action for exception handling.

**CPXSTBn[3:1]** **Coprocessor Instruction Execution Strobe** **Output**
The core asserts one of these signals to indicate to the respective coprocessor that it should begin an operation.

**FPEoDDn** **FPU Error Exception in Odd Slot** **Input**
This input is used in the handling of Floating-Point Coprocessor exceptions. It is only sampled if FPERRXn is asserted.

**FPERRXn** **Floating-Point Unit Error Exception** **Input**
Assertion of this input indicates a Floating-Point Coprocessor error.

**PCANCRn** **Pipeline Cancel at CR stage** **Output**
The CW4010 asserts this signal to indicate to a coprocessor that an exception has been detected, and may require cancellation of a previously issued instruction.

**PCANoDDn** **Pipeline Cancel is for Odd Slot** **Output**
Coprocessors use this output in conjunction with PCANCRn to determine whether cancellation of an instruction is necessary. This output is valid only when PCANCRn is asserted.

**PSTALLn** **Pipeline Stall Broadcasting Signal** **Output**
The CW4010 asserts this signal to indicate that coprocessors should stall any operations currently in progress.

## Miscellaneous Signals

**BENDn** — **Big Endian** — **Input**

This input must be tied LOW for big-endian byte ordering and HIGH for little-endian. byte ordering.

**FRCMn** — **Force Cache Miss** — **Input**

The system asserts this signal to force a cache miss for either I-Cache or D-Cache references. Cache misses forced in this way behave exactly the same as accesses to the uncached memory area.

**SCLKp** — **System Clock** — **Input**

This is the processor system clock input, which determines the instruction cycle time of the microprocessor. All internal logic is synchronized to the rising edge of this signal. The relationship of input clock frequency to core clock frequency is 1:1, so full speed operation requires an 80-MHz input clock.

**WSTALLn** — **Wait Interrupt Stall** — **Output**

The CW4010 asserts this signal to indicate that software through execution of the WAITI instruction has placed the core in the "Wait-for-Interrupt" stall condition, which reduces system power requirements. The core remains stalled until it receives an external interrupt, NMI, cold reset, or warm reset.