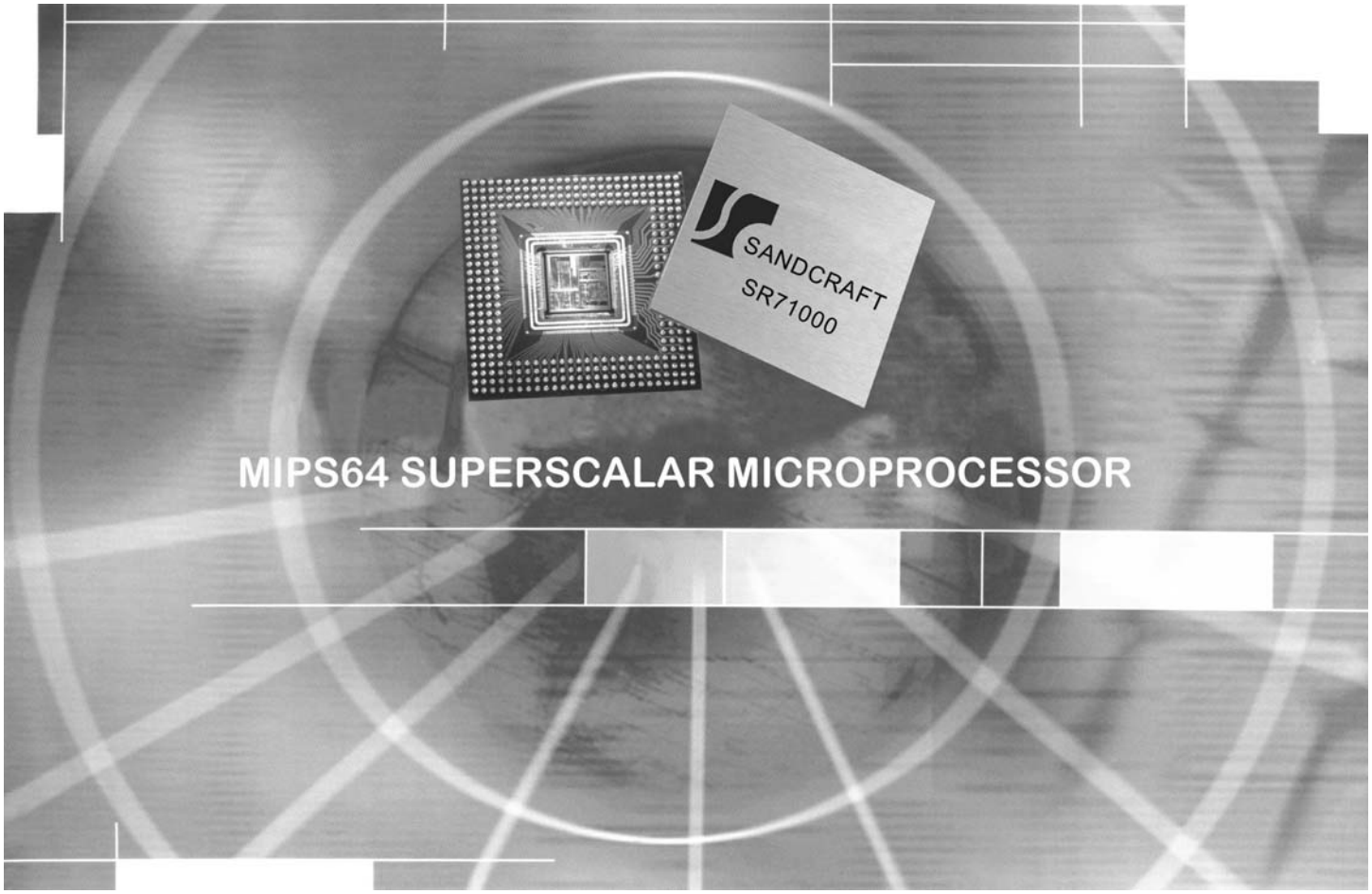
SANDCRAFT
ENGINES FOR THE DIGITAL AGE™

SR71010A-550
SR71010A-600 DATA SHEET



 Publication Number: SR71010A-DS2-1.1

TABLE OF CONTENTS

DESCRIPTION

The SR71010 is a true 2-way superscalar MIPS64™ microprocessor with 9-stage pipeline designed for high performance applications such as networking, image processing and internet servers. The architecture includes dual fetch, dual dispatch, up to 6-issue, up to 6-execute, dual-commit to sustain a maximum performance of 2 instruction execution per cycle. The highly scalable architecture can operate at a frequency up to 600MHz. The SR71010 incorporates intelligent branch prediction logic that enables a performance improvement by keeping the pipeline fully utilized. The SR71010 also maximizes system performance and reduces system cost with integrated L3 cache controller and L3 cache tag.

The SR71010 is a fully static design with dynamic power saving features that minimize power consumption. The high performance system interface, which is fully compatible to R4xxx/5xxx/7xxx SysAD interface, can operate up to 133MHz with split transactions and out-of-order return. The SR71010 includes a high-performance floating point unit (FPU) that is fully MIPS64-compliant. The FPU, which is decoupled from the integer pipeline, can dispatch up to 2 floating-point instructions per cycle.

FEATURES

True 2-way superscalar architecture
- Dual fetch, dual dispatch, up to 6-issue, up to 6-execute, dual-commit
- Maximum operation rate of pipeline: 2 instructions per cycle
- Out-of-order issue and dispatch. In-order retires.
- Fully MIPS64 Instruction Set Architecture (ISA) compliant

9-stage pipeline for high clock frequency
- Optimized pipeline bypass architecture for minimizing instruction inter-dependence stalls

Intelligent dynamic branch prediction
- Bi-mode 3Kbit table, Branch predictor
- Keeps pipeline full and minimizes branch mis-predicts

Speculative execution down predicted paths
- Maximizes sustainable performance

Low power consumption
- Fully static design
- Clock enables on all registers for improved power management
- Dynamic activation of sense amps in caches

High-performance system interface
- Compatible with R4xxx/5xxx/7xxx SysAD interface
- 133 MHz with split transactions and out-of-order return.

High-performance floating point
- Fully MIPS64 compliant
- Decoupled from Integer pipeline
- Dispatch 2 FP instructions per cycle

Cache Hierarchy
- L1
  - **For Instruction**
  - 4-way set associative, Line Locking
  - 32 KB, 32 byte line
  - **For Data**
  - 4-way set associative, Line Locking
  - 32 KB, 32 byte line
  - Write Back, Write Through (Write-Allocate/No-Write-Allocate), Bypass-L2-L3
- L2
  - **Unified Instruction and Data cache**
  - 512 Kbyte on chip
  - 8-way set associative, 32-byte line, line locking, on-chip
- L3
  - **Unified Instruction and Data cache**
  - tag on chip
  - 8-way set associative, 256 byte line with 32 byte sub-blocks, line locking
  - 32, 64, 128, 256 byte lines
  - Supports 2,4,8,16 MB

Extended features:
- 10 interrupts, up to 256MB pages
- Full support of 64bit Virtual Address space 64KB-256MBpages
- 64 dual-entry TLB
- JTAG interface compatible with IEEE1149.1

BLOCK DIAGRAM

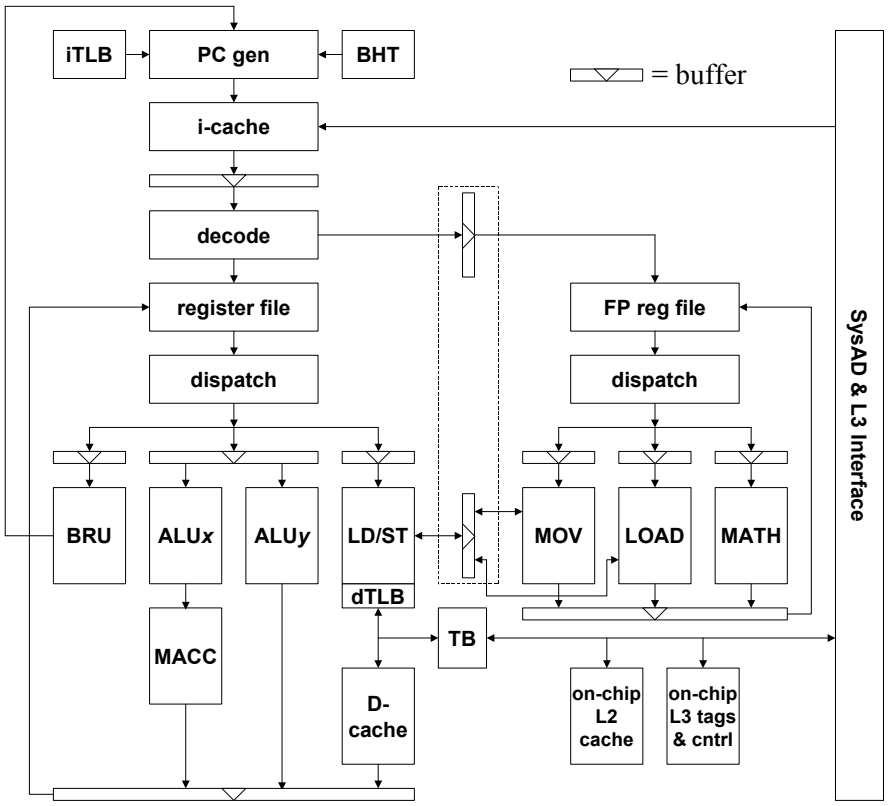


Figure 1. SR71010 block diagram

Instruction Pipeline Stages

The SR71010 has a nine-stage instruction pipeline. Each stage takes one Pcycle (one cycle of Pclock, which runs at a multiple of the frequency of SysClock). Thus, the execution of each instruction takes at least nine Pcycles. An instruction can take longer when the pipe interlocks to resolve instruction dependencies or resource conflicts.

Due to the superscalar and out-of-order nature of the processor, there can be multiple instructions in each stage of the pipeline. Figure 2 hows the nine stages of the instruction pipeline.

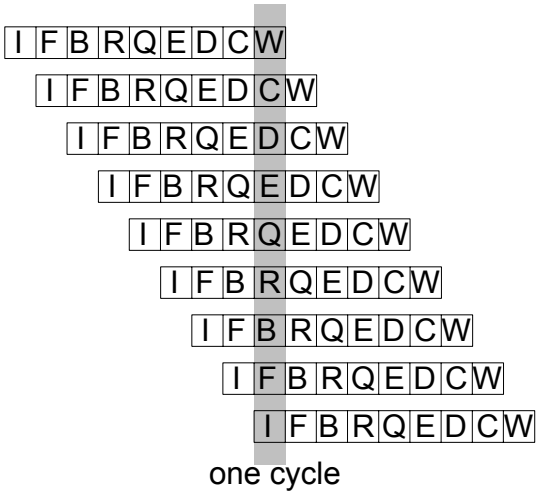


Figure 2. SR71010 pipeline phases

The processor pipe stages are:

I: Instruction Fetch 1
F: Instruction Fetch 2
B: Instruction Buffer
R: Register File Read
Q: Instruction Issue Queue
E: Execution
D: Data Cache Fetch 1
C: Data Cache Fetch 2
W: Register File Write

Two Instructions per cycle are fetched from the Instruction Cache in the F stage and written into a 32-deep instruction buffer. Two instructions per cycle are read out of the buffer, access the register file in the R stage, and are written into a 4-deep issue queue in the Q stage. The instructions wait in the issue queue until their operands and an execution unit is available. Instructions can be dispatched from the issue queue out of order. Instructions are executed by the appropriate unit in the E, D, and C stages. Up to eight instructions executing in these stages can complete execution out of order. In the W stage all exceptions are resolved and instructions commit their results to the register file in order.

Caches

Attribute	*Instruction cache*	*Data cache*
Size (Kbytes)	32	32
Associativity	4-way	4-way
Line size	32 bytes	32 bytes
Locking	Per line Uninitialized until the line is filled	Per line Uninitialized until the line is filled
Valid bit	Per line Initialized on Cold Reset	Per line Initialized on Cold Reset
Write policy	N/A	Write-back Write-through/No-write-allocate Write-through/Write-allocate Write-back/Bypass-L2-L3
Addressing scheme	Virtual index/Physical Tag	Virtual index/Physical Tag
Replacement policy	Pseudo-LRU	Pseudo-LRU
Error management on tags	Parity (one bit)	Parity (one bit)
Error management on data	8 bits per double word (single error detection)	8 bits per double word (single error detection)

Table 1: Primary Cache parameters

Primary (L1) Instruction Cache

The instruction cache uses a virtually indexed/physically tagged addressing scheme. The instruction cache also supports a set of cache operations using the CACHE instruction. These are used to manage the behavior of the cache for operations such as line locking and flushing.

Primary (L1) Data cache

The data cache is a full-featured local memory used for storing recently used data.

It is also non-blocking and supports up to four outstanding memory transactions. It always returns the critical word first on a cache miss. The data cache uses a virtually indexed/physically tagged addressing scheme. The data cache also supports a set of cache operations using the CACHE instruction. These are used to manage the behavior of the cache for operations such as line locking and flushing.

Secondary (L2) and Tertiary (L3) Caches

Table 2 provides the parameters for the unified, on-chip secondary cache and the unified tertiary cache. Note that the tertiary cache has on-chip tags (in order to provide 8-way associativity) while the data memory is off-chip and accessible via SysAD.

Attribute	***Secondary cache***	***Tertiary cache***
Size (bytes)	512K	2M, 4M, 8M, 16M (data memory is off-chip, accessed via SysAD)
Associativity	8-way	8-way (tags are on-chip)
Line size	32 bytes	32 bytes
Block size	32 bytes	32, 64, 128 and 256 bytes with 32 byte subblocks (lines)
Locking	Per line Uninitialized until the line is filled	Per block Un-initialized until a line is filled
Valid bit	Per line Initialized on Cold Reset	Per block and per line Initialized block on Cold Reset
Write policy	Write-back	Write-through, No-write-allocate of the subblock(line) if it is a (global) block-tag miss
Addressing scheme	Physical address	Physical address
Replacement policy	Pseudo-LRU	Pseudo-LRU
Error management on tags	Parity (one bit)	Parity (one bit)
Error management on data	8 bits per double word (single error detection)	8 bits per double word (single error detection)

Table 2: Secondary and Tertiary Cache parameters

Memory Management Unit

The SR71010 processor provides a full-featured memory management unit (MMU) which uses an on-chip translation lookaside buffer (TLB) to translate virtual addresses into physical addresses.

This section describes the processor's virtual and physical address spaces, the virtual-to-physical address translation, the operation of the TLB in making these translations, and those System Control Coprocessor (CP0) registers that provide the software interface to the TLB.

Translation Lookaside Buffer (TLB)

Mapped virtual addresses are translated into physical addresses using an on-chip TLB. The processor implements a fully associative TLB that holds 64 entries, which provide mapping to 64 odd/even page pairs (128 pages). This TLB holds both instruction and data pages, and is thus also referred to as the Joint TLB (JTLB). When a mapped address is translated, each TLB entry is checked simultaneously for a match with that virtual address. The virtual address is extended with an ASID stored in the CP0 EntryHi register.

Each entry in the TLB can be configured, on a per-entry basis, by the Mask bit-mask field of the entry to contain mapping for different page sizes. Page sizes range from 4 Kbytes to 256 Mbytes, in multiples of 4: that is, 4K, 16K, 64K, 256K, 1M, 4M, 16M, 64M and 256M. The valid values of the Mask field and the effect on the translation are documented in the description of the PageMask Register.

Data Micro TLB

In addition to the double-entry JTLB, the processor also implements a four entry micro DTLB that is dedicated to data address translations. If there is a miss in the DTLB, there will be a pipeline stall while the new TLB entry is transferred from the JTLB to the DTLB. The four-entry DTLB is fully associative with a Pseudo Least Recently Used replacement algorithm. Each DTLB entry contains a mapping of any of the supported page sizes. The processor guarantees that the DTLB is always a subset of the JTLB.

Instruction Micro TLB

In addition to the double entry JTLB, the processor also implements a four entry micro ITLB that is dedicated to instruction address translations. If there is a miss in the ITLB, there will be a pipeline stall while the new TLB entry is transferred from the JTLB to the ITLB. The four-entry ITLB is fully associative with a Pseudo Least Recently Used replacement algorithm. Each ITLB entry contains a mapping of any of the supported page sizes. The processor guarantees that the ITLB is always a subset of the JTLB.

Overview of Address Translation

This section gives an overview of the translation of a virtual address into a physical address.

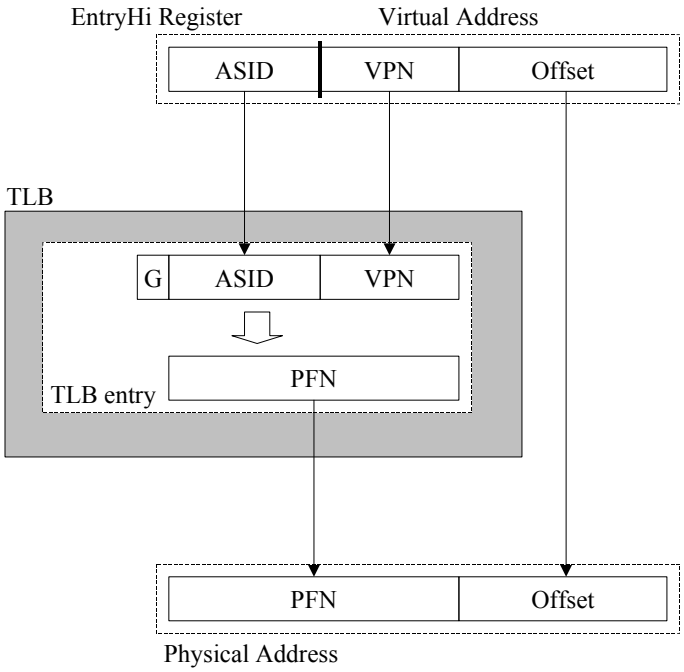


Figure 3. Overview of Virtual-to-Physical Address Translation

The address translation process proceeds as follows:

The Virtual Address (VA) represented by the Virtual Page Number (VPN) and the Address Space Identifier (ASID) is compared with all tags in the TLB.

If there is a match, the Page Frame Number (PFN) representing the upper bits of the Physical Address (PA) is output from the TLB, along with cache attributes for that page.

The Offset (which does not pass through the TLB) is concatenated to the PFN to create the final PA.

As shown, the *logical* virtual address is extended with an 8-bit address space identifier (ASID), which reduces the frequency of TLB flushing when switching contexts. The 8-bit ASID is in the CP0 EntryHi register when writing to the TLB as well as during address translations. The *Global* bit (*G*) comes from the EntryLo0 and EntryLo1 registers.

The processors physical address space is 36-bits, enough to provide 64 gigabytes of physical addressing.

Virtual-to-Physical Address Translation

For each data or instruction fetch, the virtual address is looked up in the TLB to translate into a physical address. Converting a virtual address to a physical address begins by comparing the virtual address from the processor with all TLB entries. A match is determined according to the following pseudo-code:

```
if(TLB[i]R = VAR) and
  ((TLB[i]VPN2 and not(TLB[i]Mask)) = (VAVPN2 and not(TLB[i]Mask)))
then
  if TLB[i]G.EVEN and TLB[i]G.ODD then
      TLB[i]_Matching_VA
    else
      if(TLB[i]ASID = EntryHiASID) then
        TLB[i]_Matching_VA
      else
        TLB[i]_Non_Matching_VA
else
    TLB[i]_Non_Matching_VA
```

This match is referred to as a *TLB hit*. The physical address is output from the TLB and concatenated with the *Offset*, which represents an address within the page frame space. The *Offset* does not pass through the TLB. The physical address and access control bits (*C*, *D*, and *V*) are retrieved from the matching TLB entry. While the *V* bit of the entry must be set for a valid translation to take place, it is not involved in the determination of a matching TLB entry.

If there is no TLB entry that matches the virtual address, a TLB Refill exception occurs, and software is allowed to refill the TLB from a page table of virtual/physical addresses in memory. Software can write over a selected TLB entry or use the hardware mechanism of the Random and Wired registers to write into a random entry.

If the access control bits (*D* and *V*) indicate that the access is not valid, a TLB modification or TLB invalid exception occurs. If the *C* bits equal $x10_2$, the physical address that is retrieved accesses main memory, bypassing the cache. If multiple matches are detected, the information is recorded in the status register (TLB Shutdown) and a Machine Check exception is generated. Multiple matches are considered a programming error.

The following diagram illustrates the TLB address translation process.

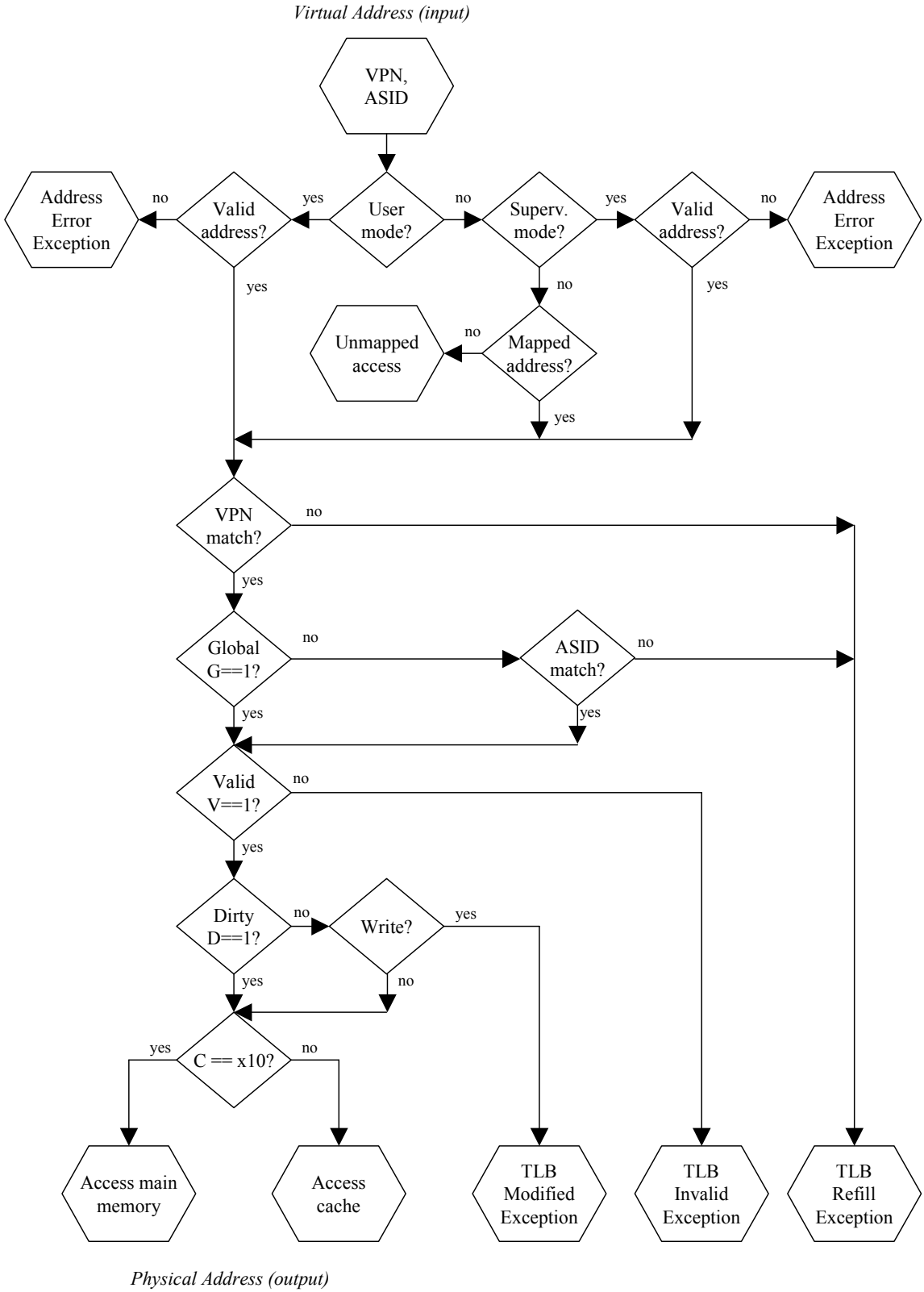


Figure 4. TLB Address Translation

FLOATING POINT UNIT

The SR71010 microprocessor has a fully MIPS64-compliant floating-point unit (FPU). Single-precision and double-precision operations and data formats are supported in this implementation.

The FPU, with associated system software, fully conforms to the requirements of ANSI/IEEE Standard 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*. In addition, the MIPS64 architecture fully supports the recommendations of the standard and precise exceptions.

Floating-Point Registers

The FPU has a set of registers that can be accessed in two ways: as 32-bit registers for compatibility with MIPS32 code; and as 64-bit registers. The mode of access of the register file is determined by $Status_{FR}$. Table 3 shows the organization of the floating-point registers in the two modes. In the MIPS32-compatibility mode, double-precision values must be stored in even-odd pairs of registers and are accessed using the identifier of the even register.

Floating-point registers when $Status_{FR}$=1		Floating-point registers when $Status_{FR}$=0	
63	0	63 32	31 0
f0			f0
f1			f1
f2			f2
f3			f3
f4			f4
f5			f5
f6			f6
f7			f7
f8			f8
f9			f9
f10			f10
f11			f11
f12			f12
f13			f13
f14			f14
f15			f15
f16		unpredictable	f16
f17			f17
f18			f18
f19			f19
f20			f20
f21			f21
f22			f22
f23			f23
f24			f24
f25			f25
f26			f26
f27			f27
f28			f28
f29			f29
f30			f30
f31			f31

Table 3: Floating-point registers

Floating-Point Control Registers

The SR71010 processor contains five control registers that can only be accessed by CP1 control register move operations (CTC1, CFC1). The following table lists the assignments of the control registers.

CP1 Register number	*Acronym*	*Name*	*Description*
0	FIR	Floating-point implementation register	Coprocessor implementation and revision register
1-24	*reserved*		
25	FCCR	Floating-point condition codes register	Alternative method for accessing floating-point condition codes.
26	FEXR	Floating-point exceptions register	Alternative method for accessing floating-point cause and flags.
27	*reserved*		
28	FENR	Floating-point enables register	Alternative method of accessing enables and rounding mode
29-30	*reserved*		
31	FCSR	Floating-point control and status register	Rounding mode, cause, trap enables, and flags

Table 4: Floating Point Control Register Assignments

PROCESSOR BUS INTERFACE

The System interface allows the processor to access external resources needed to satisfy cache misses and uncached operations, while permitting an external agent access to some of the processor's internal resources.

The system interface of the processor is compatible with the SysAD bus from the R4000 and R5000 families of MIPS microprocessors. Support includes two outstanding reads and split response on read transactions, but does not include compatibility with tertiary data cache protocols. Please see the chapter on tertiary data cache in this specification if it applies to you.

This document describes the System interface from the point of view of both the processor and the external agent.

Terms Used

The following terms are used in this document:

An external agent is any logic device connected to the processor, over the System interface, that allows the processor to issue requests.
A system event is an event that occurs within the processor and requires access to external system resources.
Sequence refers to the precise series of requests that a processor generates to service a system event.
Protocol refers to the cycle-by-cycle signal transitions that occur on the System interface pins to assert a processor or external request.
Syntax refers to the precise definition of bit patterns on encoded buses, such as the command bus.

Interface Buses

The following figure shows the primary communication paths for the System interface: a 64-bit address and data bus, **SysAD (63:0)**, and a 9-bit command bus, **SysCmd (8:0)**. The **SysAD** and the **SysCmd** buses are bi-directional; that is, they are driven by the processor to issue a processor request and by the external agent to issue an external request.

A request through the System interface consists of:

an address
a System interface command that specifies the precise nature of the request
a series of data elements if the request is for a write
a read response for a read request

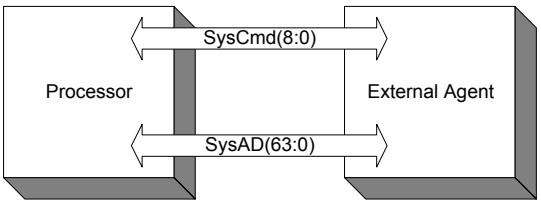


Figure 5. System interface busses

PROCESSOR SIGNAL DESCRIPTION

This chapter describes the signals used by and in conjunction with the processor. The signals include the System interface, the Clock interface, the Interrupt interface, the Joint Test Action Group (JTAG) and Test interface, the Initialization interface, and Power inputs.

Signals are referenced in bold, and low active signals have a trailing asterisk; for instance, the low-active Read Ready signal is **RdRdy***. The arrows indicate whether a signal is an input (the processor receives it), an output (the processor sends it out), or bi-directional.

The following figure illustrates the functional groupings of the processor signals.

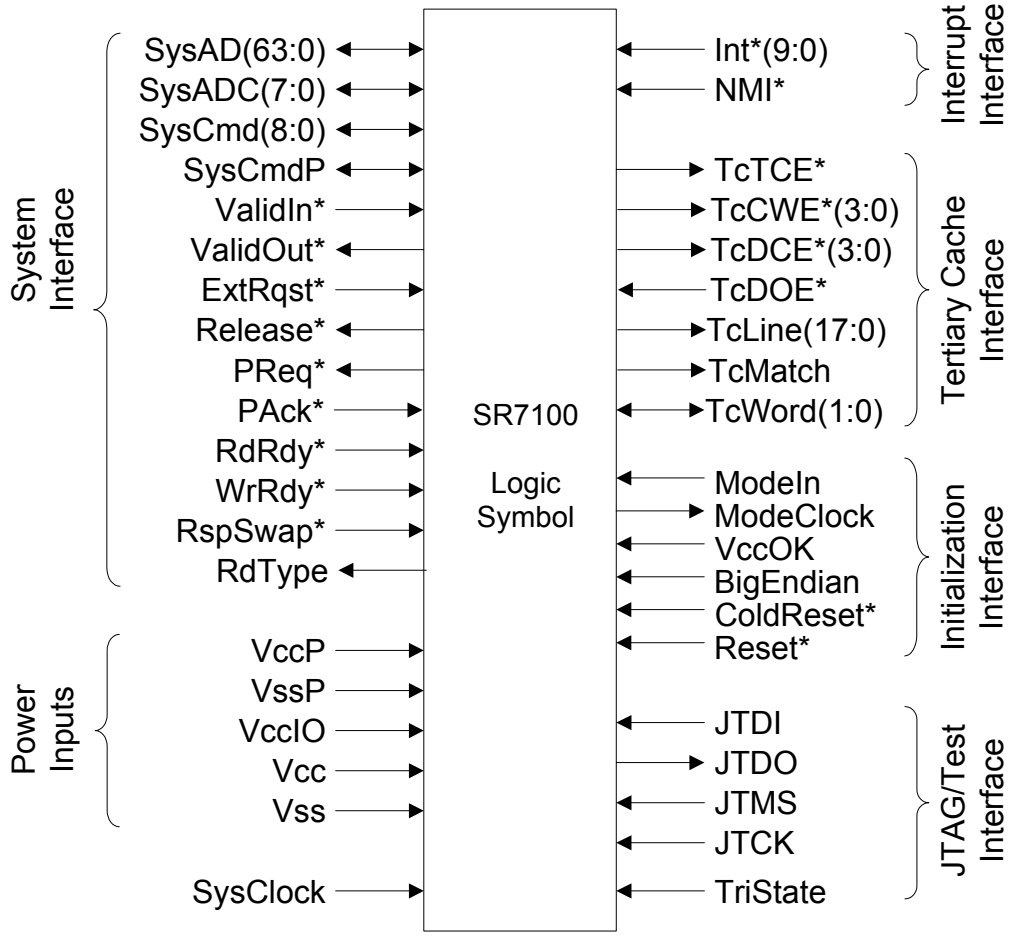


Figure 6. Processor signals

The reset state of pins is marked as subscript in the *Reset* column in the following signal tables. The notations are defined as:

D: **ModeClock** pin is always driven.

CR: All other output-only pins are tri-stated until the **ColdReset*** signal de-asserts (except **JTDO** which remains tri-stated and is controlled by JTAG states).

R: All I/O pins remain tri-stated until the **Reset*** signal de-asserts.

Not applicable to all input-only pins.

System Interface Signals

System interface signals provide the connection between the processor and the other components in the system. The following tables lists the system interface signals.

Name	*Definition*	*Direction*	*Reset*	*Description*
ExtRqst*	External request	Input		An external agent asserts **ExtRqst*** to request use of the System interface. The processor grants the request by asserting **Release***.
Release*	Release interface	Output	CR	In response to the assertion of **ExtRqst***, the processor asserts **Release***, signaling to the requesting device that the System interface is available. **Release*** is also asserted for an uncompelled change to slave state when one or more read requests is outstanding.
PReq*	Processor Request	Output	CR	Indicates that the processor has another request that is pending. This is used to indicate that the processor would like to send another transaction. It is up to the external agent to grant the request by issuing a PAck*
PAck*	Processor Acknowledge	Input		Indicates that the external agent is ready to release the bus back to the processor in response to a PReq*.
RdRdy*	Read ready	Input		The external agent asserts **RdRdy*** to indicate that it can accept processor read requests.
SysAD(63:0)	System address/data bus	Input / Output	R	A 64-bit address and data bus for communication between the processor and an external agent.
SysADC(7:0)	System address / data check bus	Input / Output	R	An 8-bit bus containing parity for the **SysAD(63:0)** bus. **SysADC(7:0)** is valid on data cycles only.
SysCmd(8:0)	System command / data identifier	Input / Output	R	A 9-bit bus for command and data identifier transmission between the processor and an external agent.
SysCmdP	System command parity bit	Input / Output	R	Not used on input, driven to 0 on output.
ValidIn*	Valid input	Input		The external agent asserts **ValidIn*** when it is driving a valid address or data on the **SysAD(63:0)** bus and a valid command or data identifier on the **SysCmd(8:0)** bus.
ValidOut*	Valid output	Output	CR	The processor asserts **ValidOut*** when it is driving a valid address or data on the **SysAD(63:0)** bus and a valid command or data identifier on the **SysCmd(8:0)** bus to the external agent.
RspSwap*	Response swap	Input		This signal indicates to the processor that the data being returned is out-of-order with respect to the two outstanding read requests.
WrRdy*	Write ready	Input		The external agent asserts **WrRdy*** when it can accept a processor write request.
RdType	Read type	Output	CR	During the address cycle of a read request, this signal indicates if the request is an instruction fetch or a data load.

Table 5: Signal Descriptions

Power Inputs

Name	*Definition*	*Direction*	*Description*
Vss	Vss for Processor Core and I/O	N/A	Ground for the internal core logic and processor I/O interface.
VccIO	Vcc for Processor I/O		Process dependent power for the I/O pads logic.
VccInt	Vcc for Processor Core		Process dependent power for the internal core logic.
VccP	Quiet Core Vcc for PLL		Quiet Vcc for the internal phase locked loop. Each internal PLL requires a quiet Core voltage Vcc.
VssP	Quiet Vss for PLL		Quiet Vss for the internal phase locked loop. Each internal PLL requires a quiet Vss.

Clock Interface Signals

Name	*Definition*	*Direction*	*Description*
SysClock	System Clock	Input	System clock input that establishes the system interface operating frequency and phase during normal operation.

Interrupt Interface Signals

Name	*Definition*	*Direction*	*Description*
Int*(9:0)	Interrupt (maskable)	Input	Maskable interrupts.
NMI*	Non-maskable interrupt	Input	Non-maskable interrupt, OR'd with bit 6 of the interrupt register.

Initialization Interface Signals

Name	*Definition*	*Direction*	*Reset*	*Description*
ModeIn	Mode Data In	Input		Serial boot-mode data input
ModeClock	Boot Mode Clock	Output	D	Serial boot-mode data clock output at the **SysClock** frequency divided by 256. (130 KHz ~ 520 KHz)
VccOK	Vcc is OK	Input		When asserted, this signal indicates to the processor that the 3.3V power supply has been above 2.4V for more than 100 msec. and will remain stable. The assertion of **VccOK** initiates the reading of the boot-time mode control serial stream.
BigEndian	Endian Mode Select	Input		Sets processor addressing mode to either Big Endian or Little Endian. This input is logically-ORed with Mode ROM bit 8 to become $Config_{BE}$.
ColdReset*	Cold reset	Input		This signal must be asserted for a power on reset or a cold reset. **ColdReset*** must be deasserted synchronously with **SysClock**.
Reset*	Reset	Input		This signal must be asserted for any reset sequence. It can be asserted synchronously or asynchronously for a cold reset, or synchronously to initiate a warm reset. **Reset*** must be deasserted synchronously with **SysClock**.

Tertiary Cache Interface Signals

Name	*Definition*	*Direction*	*Reset*	*Description*
TcTCE*	Tertiary Cache Tag SRAM Chip Enable	Output	CR	This signal is monitored by the external agent and indicates to it that a tertiary cache data RAM access is occurring. In SR71010, this signal is only used to indicate a refilling will follow.
TcCWE*[3:0]	Tertiary Cache Data SRAM Write Enable	Output	CR	Two pairs of two identical write enable signals for the tertiary cache are provided to balance loading.
TcDCE*[3:0]	Tertiary Cache Data SRAM Chip Enable	Output	CR	Two pairs of two identical output enable signals for the tertiary cache are provided to balance loading.
TcDOE*	Tertiary Cache Data SRAM Output Enable	Input		This signal comes from the agent and informs the processor that it is beginning or ending its use of the SysAD bus for refilling the tertiary cache.
TcLine[17:0]	Tertiary Cache Sub-block Index	Output	CR	This bus is the index used to access the data RAMs of the tertiary cache.
TcMatch[a]	Tertiary Cache Sub-block Tag Match	Output	CR	The processor asserts this signal whenever a tertiary cache hit is detected to prevent a refill from the agent.
TcWord[1:0]	Tertiary Cache Double Word Index	Input / Output	R	This signal defines the doubleword in a subblock and is driven by the processor on cache hits and by the external agent on cache miss refills.

Notes:

a. If the SR71010 is used in an existing RM7000 system, the tag RAMs must be removed from the board to prevent drive fights on this signal. For the RM7000, TcMatch is an input driven by the external cache controller. For SR71010, with internal tertiary cache tags, the pin is an output used to signal that a hit has occurred in the L3 cache.

JTAG Interface Signals

Name	*Definition*	*Direction*	*Description*
JTDI	Test Data In	Input	Data is serially scanned in through this pin.
JTCK	Test Clock input	Input	The processor accepts a serial clock on **JTCK**. On the rising edge of **JTCK**, both **JTDI** and **JTMS*** are sampled. The maximum frequency of **JTCK** is 33 MHz, and it runs asynchronously to **SysClock**.
JTDO	Test Data Out	Output	Data is serially scanned out through this pin on the falling edge of **JTCK**.
JTMS	Test Mode Select	Input	JTAG Test Mode Select signal.

Note: See IEEE 1149.1 JTAG Specification for full details of the JTAG interface.

Test Interface Signals

Name	*Definition*	*Direction*	*Description*
Tristate	Tristate all outputs	Input	This signal tristates all Processor outputs to allow board level test to isolate the processor.

SR71010 Extended Interrupt Architecture

A block diagram of the SR71010 Extended Interrupt Architecture (EIA) is given in the following figure. This structure provides two registers that are used to manage up to ten external interrupts by mapping them to the MIPS64 interrupt architecture

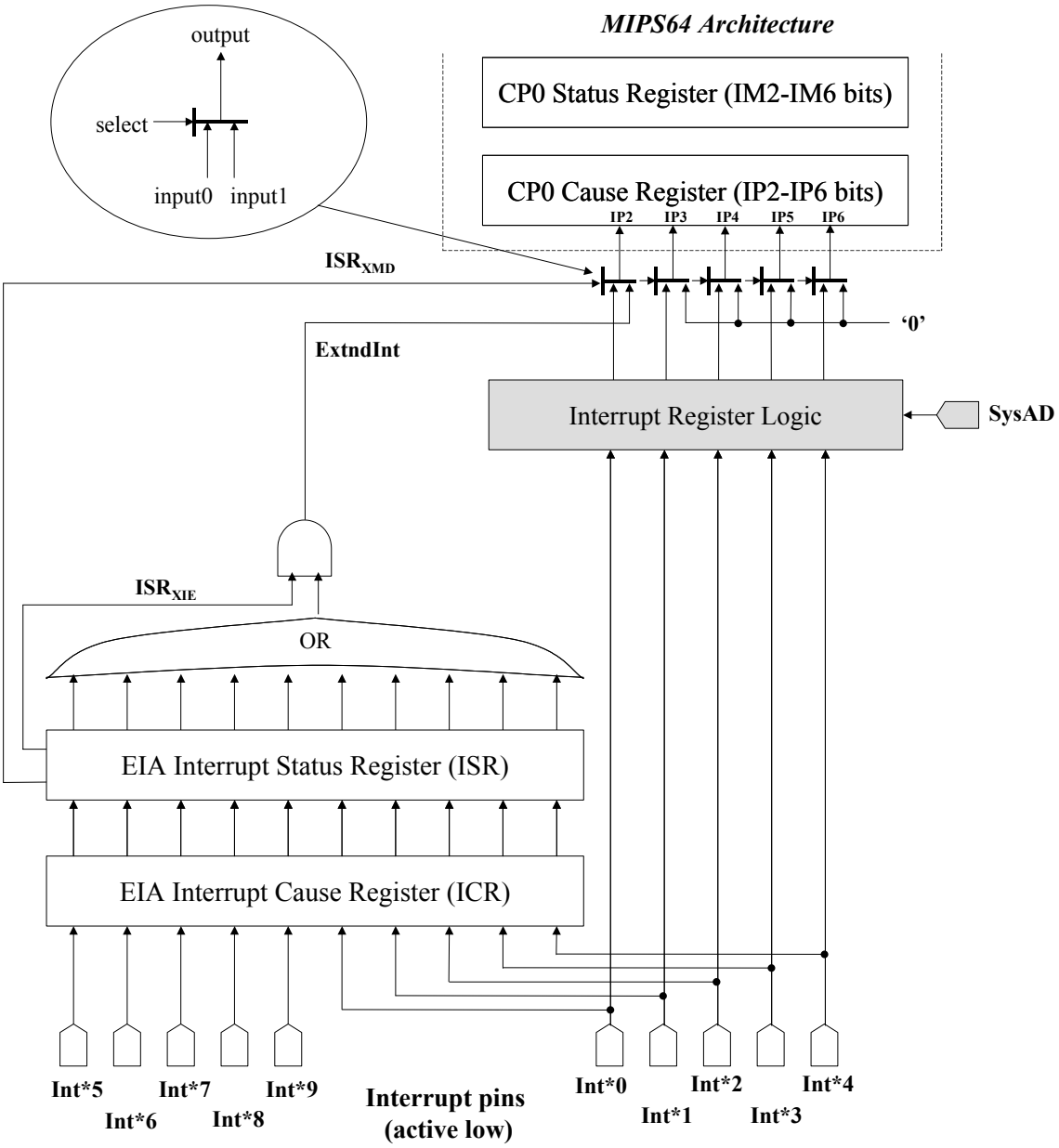


Figure 7. SR71010 Extended Interrupt Architecture Block Diagram

The SR71010 EIA operates in two modes, defined by the EIA Status Register XMD bit, ISR_{XMD}. The objective of the two modes is to provide a fully MIPS64-compatible interrupt interface, and an extended interface with more interrupt pins. The two modes are discussed in the following sections.

Default Mode: ISR_{XMD} = 0.

In default mode, external interrupt pins **Int*4** through **Int*0** bypass the EIA and connect directly to their MIPS64 counterparts. This allows full MIPS64 compliance to be achieved to the pins. External interrupt pins **Int*9** through **Int*5** are ignored (though their state is accessible via the Interrupt Cause Register).

Extended Mode: ISR_{XMD} = 1.

In extended mode, the ten external interrupt pins **Int*9** through **Int*0** are monitored by the EIA Cause Register (ICR) and masked by the EIA Interrupt Status Register (ISR). If an unmasked interrupt occurs, and ISR_{XIE} = 1 (i.e. interrupts are enabled), then the *ExtndInt* signal is asserted and the MIPS64 architecture sees a normal interrupt exception. The interrupt handler software must then examine the state of the EIA registers to determine the cause of the interrupt.

Note that in order to operate correctly, the operating system must enable interrupts using $Status_{IE}$ and ensure that $Cause_{IP2}$ is unmasked (i.e. $Status_{IM2}$=1). Interrupt control should then be managed using the SR71010 EIA registers.

From a MIPS64 architecture perspective, the CP0 Cause Register must be interpreted as follows:

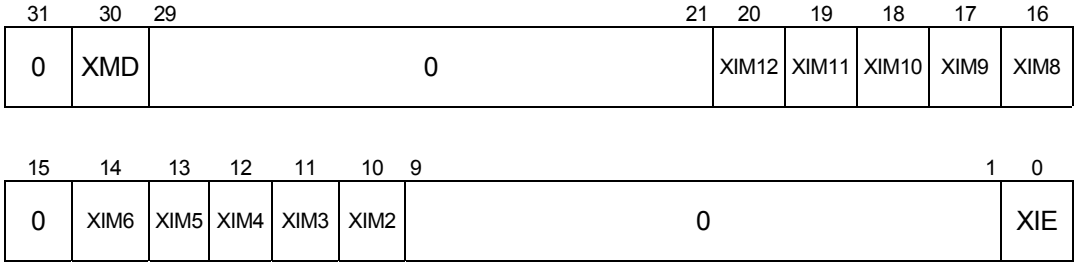
CP0 Cause	***Default Mode ISR_{XMD} = 0***	***Enhanced Mode ISR_{XMD} = 1***
$Cause_{IP7}$	Performance Counter interrupt	
$Cause_{IP6}$	Int*4	Unused – always 0
$Cause_{IP5}$	Int*3	Unused – always 0
$Cause_{IP4}$	Int*2	Unused – always 0
$Cause_{IP3}$	Int*1	Unused – always 0
$Cause_{IP2}$	Int*0	Int*9 through Int*0
$Cause_{IP1}$	Software interrupt 1	
$Cause_{IP0}$	Software interrupt 0	

SR71010 Extended Interrupt Architecture Registers

The EIA registers have been specially organized to allow simple merging with the contents of the MIPS64 CP0 Cause and Status registers. Access to these registers depends on the operating mode and the state of the $Status_{CU0}$ bit.

ISR (EIA Status Register, EIA Register 12)

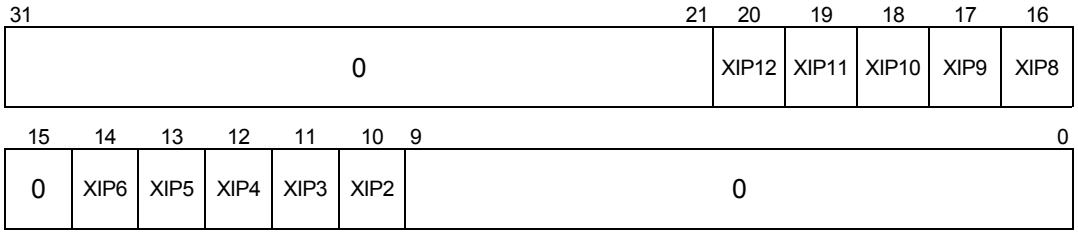
The EIA Status Register is a read/write register that contains the modes, enables, and mask bits used by the Enhanced Interrupt Architecture.



Field	*Bits*	*R/W*	*Description*	*Reset state*
XIE	0	R/W	Master enhanced interrupt enable. 0: Disabled 1: Enabled	*undefined*
0	9:1	R	Must be written as 0, and returns 0 when read	0
XIM2	10	R/W	Enhanced Interrupt Mask bits 0: Corresponding interrupt pending bit in EIA Cause register is masked. 1: Corresponding interrupt pending bit in EIA Cause register is enabled.	*undefined*
XIM3	11	R/W		*undefined*
XIM4	12	R/W		*undefined*
XIM5	13	R/W		*undefined*
XIM6	14	R/W		*undefined*
0	15	R	Must be written as 0, and returns 0 when read	0
XIM8	10	R/W	Enhanced Interrupt Mask bits 0: Corresponding interrupt pending bit in EIA Cause register is masked. 1: Corresponding interrupt pending bit in EIA Cause register is enabled.	*undefined*
XIM9	11	R/W		*undefined*
XIM10	12	R/W		*undefined*
XIM11	13	R/W		*undefined*
XIM12	14	R/W		*undefined*
0	29:21	R	Must be written as 0, and returns 0 when read	0
XMD	30	R/W	Enhanced Interrupt Architecture Mode 0: Map **Int*4** through **Int*0** interrupt pins directly to MIPS64 IP6 through IP2 signals. 1: Map **Int*9** through **Int*0** interrupt pins to the EIA registers which drive MIPS64 IP2 signal.	0
0	31	R	Must be written as 0, and returns 0 when read	0

ICR (EIA Cause Register, EIA Register 13)

The EIA Cause Register is read-only and contains the interrupt pending bits.



Field	*Bits*	*R/W*	*Description*	*Reset state*
0	9:0	R	Must be written as 0, and returns 0 when read	0
XIP2	10	R	Interrupt pending bits 0: Interrupt not pending 1: Interrupt pending	0
XIP3	11	R		0
XIP4	12	R		0
XIP5	13	R		0
XIP6	14	R		0
0	15	R	Must be written as 0, and returns 0 when read	0
XIP8	16	R	Interrupt pending bits 0: Interrupt not pending 1: Interrupt pending	0
XIP9	17	R		0
XIP10	18	R		0
XIP11	19	R		0
XIP12	20	R		0
0	31:21	R	Must be written as 0, and returns 0 when read	0

SR71010 Extended Interrupt Architecture Instructions

Two instructions have been added to the base MIPS64 ISA in the SPECIAL2 region. These instructions allow direct register-to-register access between the general-purpose registers and the EIA registers.

MTER: Move to EIA Register
MFER: Move from EIA Register

SR71010 Extended Interrupt Architecture Code Examples

Switching to Extended Interrupt Mode

The process of switching from normal MIPS64 interrupts to extended interrupts is as follows:

1. Disable interrupts by setting $Status_{IE}$ = 0
2. Enable $Status_{IM2}$=1 so that extended interrupts can assert IP2
3. Ensure no pending interrupts in EIA status setting EIA $Status_{XIE}$=0
4. Switch EIA $Status_{XMD}$=1 to enable extended interrupts
5. Set $Status_{IE}$=1 so that MIPS64 interrupts are enabled
6. Set masks in EIA Status as required
7. Enable extended interrupts by setting EIA $Status_{XIE}$=1

The code fragment below performs this process:

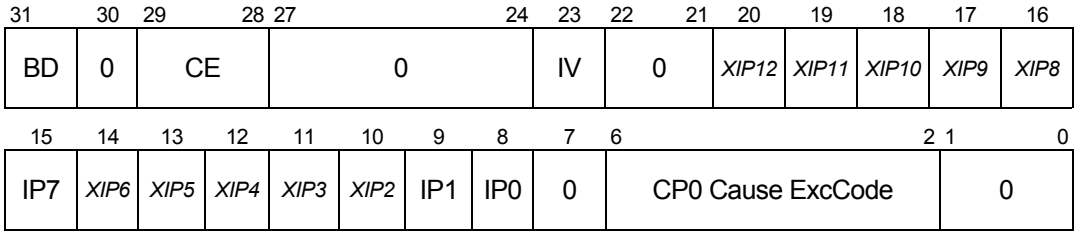
```
# Disable interrupts in CP0 and set IM2
mfc0  $1, $12         # get Status
addiu $2, $0, -2      # set r2 to 11111…10
and   $1, $1, r2      # clear bit 0 (IE)
ori   $1, $1, 0x4000  # set bit 10 (IM2)
mtc0  $1, $12         # write to Status with IE=0, IM2=1

# Disable EIA interrupts and switch mode
mfer  $1, $12         # get EIA Status
and   $1, $1, r2      # clear bit 0 (XIE)
addiu $3, $0, 1       # load r3 with 1
sll   $3, $3, 30      # move to bit 30 or r3
or    $1, $1, r3      # set bit 30 of r1 (XMD)
mter  $1, $12         # write to EIA Status with XIE=0, XMD=1

# Enable Status IE
mfc0  $1, $12         # get Status
ori   $1, $0, 1       # set bit 0 (IE)
mtc0  $1, $12         # write to Status with IE=1
#
# Put code here to set up EIA masks etc.
#
# Enable extended interrupts
mfer  $1, $12         # get EIA Status
ori   $1, $0, 1       # set bit 0 (XIE)
mter  $1, $12         # write to EIA Status with XIE=1

# Extended interrupts are now enabled
```

Merging all pending interrupts

When EIA mode is enabled, all pending interrupts can be amalgamated by merging this register with the CP0 Cause register. Firstly, the EIA interrupt to CP0 Cause must be disabled by setting XIE = 0. This prevents IP2 in CP0 Cause from obscuring the merged result. Then CP0 Cause and EIA Cause can be read and ORed together to give the final result. The code below creates a single register, `r1`, with all pending interrupt bits set and other CP0 Cause fields unaffected:

```
# Note that this code is optimized by assuming that the fields
# defined in CP0 Cause and EIA Cause are as defined (i.e. many are
# zero) so that a simple OR of the registers is possible.
mfer  $1, $12        # get EIA Status Register
xori  $1, 1          # clear XIE bit – disables Cause_IP2
mter  $1, $12        # switch off IP2
mfco  $2, $13        # get CP0 Cause
mfer  $1, $13        # get EIA Cause
or    $1, $1, $2     # merged pending bits in r1
```

The resultant register has the following structure. Note that values from EIA Cause are shown italicized. All other bits come from CP0 Cause.



INITIALIZATION AND TEST INTERFACE

The processor has two types of initialization sequences; both use the **VccOK**, **ColdReset***, and **Reset*** input signals.

Cold reset: restarts all clocks and resets the JTAG logic after the power supply is stable. A cold reset completely reinitializes the internal state machines of the processor without saving any state information. It automatically loads the processor configuration information from the Mode ROM. Assertion of the **VccOK** signal is used to start the configuration sequence.
Warm reset: restarts the processor, but does not affect clocks or JTAG circuitry. A warm reset preserves the processor internal state.

Processor Reset Signals

This section describes the five reset signals **VccOK, ColdReset***, **Reset*, ModeClk** and **ModeIn**.

VccOK: When asserted, this signal indicates that the minimum operating voltage of the processor has been reached for at least 100ms and will remain stable. Assertion of this signal initiates the process to load configuration information from the Mode ROM into the processor. The VccOK signal must remain asserted for the duration of ModeROM load operation. If the VccOK signal is deasserted without ColdReset* also being asserted, then unpredictable behavior will result.

ColdReset*: The **ColdReset*** signal must be asserted (low) for either a power-on reset or a cold reset. **ColdReset*** must be deasserted synchronously with **SysClock**.

Reset*: The **Reset*** signal must be asserted for any reset sequence. It can be asserted synchronously or asynchronously for a cold reset, or synchronously to initiate a warm reset. **Reset*** must be deasserted synchronously with **SysClock**.

ModeClk: This clock signal is used by the Mode ROM for data transfer into the processor. It operates at the **SysClock** frequency divided by 256.

ModeIn: Data in from the Mode ROM.

Power-on Reset

The sequence for a power-on reset is listed below.

1. Power-on reset requires a stable **Vcc** of at least 2.4 volts from the VccIO power source to the processor. It also requires a stable, continuous system clock at the processor operational frequency.

2. After at least 100 ms of stable **Vcc** and **SysClock**, the **VccOK** signal is asserted and the processor configuration is read in from the Mode ROM.

3. **ColdReset*** is asserted for at least 64K (2^{16}) cycles following the assertion of **VccOK**. This gives the processor time to read its configuration from the mode ROM. **ColdReset*** must be deasserted synchronously with **SysClock**.

4. After **ColdReset*** is deasserted synchronously, **Reset*** is deasserted to allow the processor to begin running. (**Reset*** must be held asserted for at least 64 **SysClock** cycles after the deassertion of **ColdReset***.) **Reset*** must be deasserted synchronously with **SysClock**.

5. Upon reset, the processor drives the **SysAD** bus. After **Reset*** is deasserted, the processor branches to the Reset exception vector and begins executing the Cold Reset exception code.

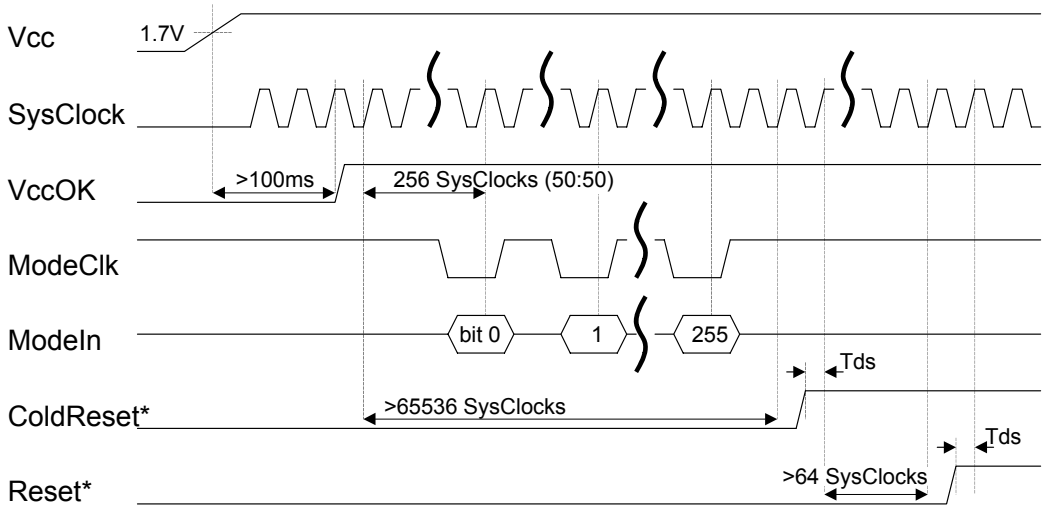


Figure 8. Power-On Reset Timing Diagram

Cold Reset

A cold reset requires the same sequence as a power-on reset except that the power is presumed to be stable before the assertion of the reset inputs. Note that **VccOK** is deasserted for at least 64 clock cycles and then asserted to initiate reading of the Mode ROM.

The following figure shows the cold reset timing diagram.

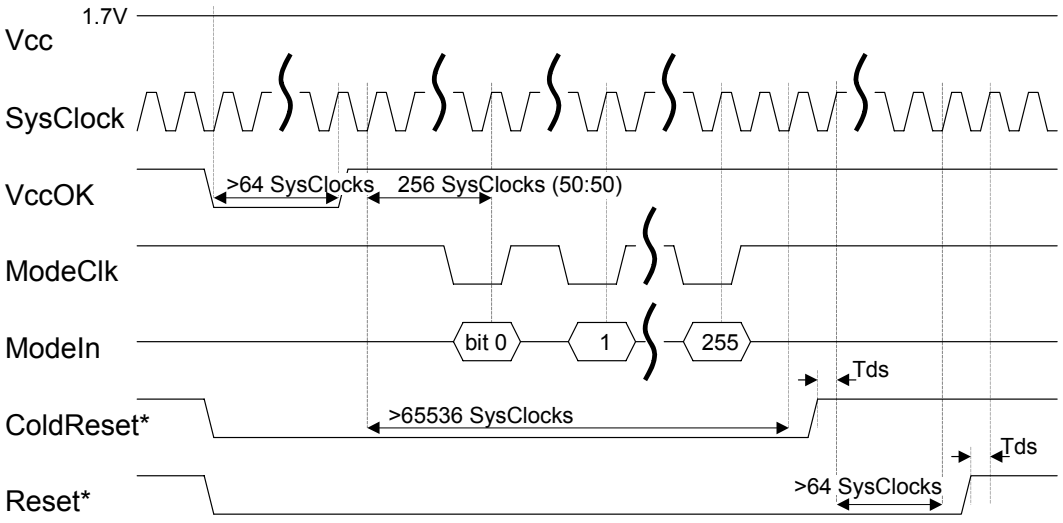


Figure 9. Cold Reset Timing Diagram

Warm Reset

To execute a warm reset, the **Reset*** input is asserted synchronously with **SysClock**. It is then held asserted for at least 64 **SysClock** cycles before being deasserted synchronously with **SysClock**. A warm reset forces the processor to start with a Soft Reset exception

A warm reset is used to reset the processor without affecting the clocks; in other words, a warm reset is a logic reset. This allows the processor to retain as much of its state as possible for debugging. Since a warm reset takes effect immediately upon assertion of the Reset* signal, multicycle operations such as a cache miss or a floating point instruction may be aborted with the result of some loss of data.

Upon reset, the processor drives the **SysAD** bus. After **Reset*** is deasserted, the processor branches to the Reset exception vector and begins executing the reset exception code. If reset is asserted in the middle of a **SysAD** transaction, care must be taken to reset all external agents to avoid **SysAD** bus contention.

The following figure shows the warm reset timing diagram.

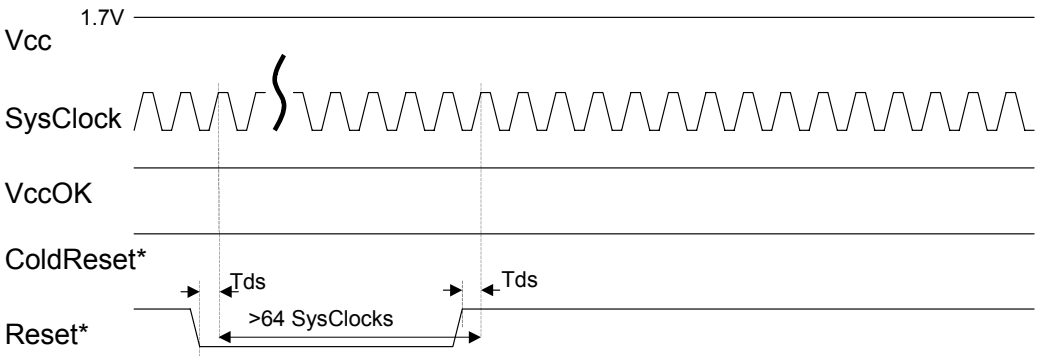


Figure 10. Warm Reset Timing Diagram

Processor Reset State

After a power-on reset, cold reset, or warm reset, all processor internal state machines are reset, and the processor begins execution at the reset vector.

Upon warm reset, all processor state that has already been updated by the time reset was applied is preserved. But the state that is committed, but not updated is lost, since all state-machines abort and return to reset state. Therefore, the precise state of the caches depends on whether or not a cache miss sequence was interrupted by reset. Also, since the processor implements, non-blocking loads, the register update is killed if a pending load has not yet been completed. The Branch Prediction Table is initialized as well.

Processor Initialization and Configuration Signals

The initialization and configuration sequence begins immediately after **VccOK** is asserted. The processor generates the **ModeClk** signal and strobes in data from the mode ROM on rising edges. Data bits are presented on the **ModeIn** pin by the mode ROM. Exactly 256 bits are read by the processor. The processor synchronizes the **ModeClk** output at the time **VccOK** is asserted. The first rising edge of **ModeClk** occurs 256 **SysClock** cycles after **VccOK** is asserted.

Mode ROM Settings

The following table defines the contents of the Mode ROM, which are loaded after a Power-on Reset or Cold Reset sequence.

Bit	*Name*	*Values*			
0	*Reserved*	*Must be 0*			
4:1	*Reserved*[a]	*Must be 0*			
7:5	PClk-to-SClk ratio[b]	*Mode bit 20 = 0*		*Mode bit 20 = 1*	
		000: reserved 001: reserved 010: 4:1 011: 5:1	100: 6:1 101: 7:1 110: 8:1 111: reserved	000: reserved 001: reserved 010: 4.5:1 011: 5.5:1	100: 6.5:1 101: 7.5:1 110: 8.5:1 111: reserved
8	Endian bit (logically Ored with the **BigEndian** pin to define $Config_{BE}$)	0: Little endian ordering 1: Big endian ordering			
14:9	*Reserved*[a]	*Must be 0*			
15	Tertiary data cache SRAM type	0: Dual-cycle deselect 1: Single-cycle deselect			
17:16	*Reserved*[a]	*Must be 0*			
18	Enable extra SR71010 signals[c]	0: RM7000-compatible mode. All do-not-connect pins of RM7000 footprint are truly disconnected on SR71010. Even though the pins are marked as DNC, connections can be made and the SR71010 processor will ignore any signal activity on these lines. The **TcDCE*(3:2)** and **TcCWE*(3:2)** signals are tristated and, internally, the **TriState** pin is deasserted. 1: Extra tertiary cache drivers **TcDCE*(3:2)** and **TcCWE*(3:2)** are enabled, and **TriState** input is enabled. All other do-not-connect pins of customer footprint must be truly disconnected on SR71010[d].			
19	*Reserved*[a]	*Must be 0*			
20	Pclock to SysClock Multipliers[b]	0: Integer (4, 5, 6, 7, 8) 1: Half Integer (4.5, 5.5, 6.5, 7.5, 8.5)			
21	*Reserved*[a]	*Must be 0*			
22	Output drivers slew control[c] (except **TcLine** and **TcWord**)	0: Slow 1: Fast			
23	L3 address drivers slew control[c] (**TcLine** and **TcWord** only)	0: Slow 1: Fast			
255:24	*Reserved*[a]	*Must be 0*			

Table 6: Mode ROM bit definitions

JTAG AND TEST SIGNALS

The processor provides several provisions for chip level and system level test. The core test functionality is the scan interface that is compatible with the Joint Test Action Group (JTAG). The JTAG interface provides boundary scan compatible with the IEEE 1149.1/1149.1a standard, and provides emulation support features through that same interface.

In addition to the JTAG interface, the **Tristate** pin provides the capability of tristating all outputs for system test.

JTAG and Test Pins

JTCK (i) JTAG Test Clock

The processor accepts a serial clock on **JTCK**. At the rising edge of **JTCK**, both **JTDI** and **JTMS** are sampled. The maximum frequency of **JTCK** is 33 MHz, and it will run asynchronously to the processor clock. The **JTCK** clock period to internal processor clock period ratio is required to be at least 4:1 in order to meet the synchronization requirements for the debug module.

JTMS (i) JTAG Mode Select

JTAG command signal decoded by the TAP controller to control test operations. This pin has an internal pull-up so that its level is high when the tool is not connected.

JTDI (i) JTAG Test Data In

Data is serially scanned in through this pin. This pin has an internal pull-up so that its level is high when the tool is not connected.

JTDO (o) JTAG Test Data Out

Data is serially scanned out through this pin on the falling edge of **JTCK**. Per the IEEE 1149.1 standard, the **JTDO** output will be tristated unless data is actively being scanned.

Tristate(i) Tristate all outputs

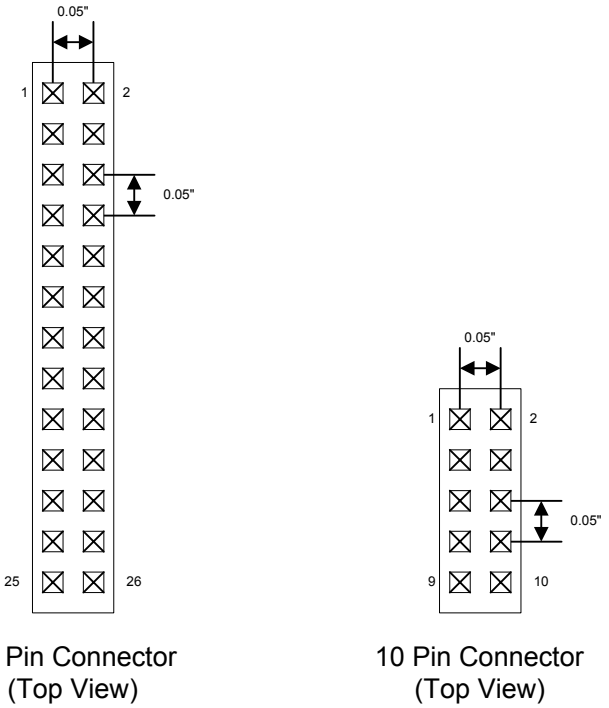
This signal tristates all processor outputs to allow isolation in board level test.

System Design Hooks for Debug Tool Support

Warning: This information is dependent on third party debug tool vendors and subject to change without notice.

System designers are encouraged to incorporate into their board design a 26 pin high-density connector that provides 13 signals and 13 grounds. This assures maximum performance and eliminates noise problems by incorporating a signal-ground type arrangement.

The target connector is a 0.05” pitch 26 pin header connector, Samtec part number FTSH-113-01-L-D (through hole) or FTSH-113-01-L-DV (surface mount) or equivalent. The 26 pins are allocated to 12 signals (and one spare) and 13 GNDs. The connector spacing is convenient 0.05” X 0.05” and provides easy cabling to external equipment. There is another option for a smaller, 10 pin connector that contains only the boundary scan TAP signals. The smaller connector excludes all the real-time trace related signals and contains only the basic JTAG signals. Pin-outs for these connectors are given in the following figure.



26 Pin Connector
(Top View)

10 Pin Connector
(Top View)

Figure 11. JTAG connectors

Pin	*Signal*	*Target Input/Output*	*Target Termination*
1	*reserved*		
3	JTDI	Input	1 Kohm pull-up resistor
5	JTDO	Output	33 Ohm series resister
7	JTMS	Input	1 Kohm pull-up resistor
9	JTCK	Input	1 Kohm pull-up resistor
11	Tristate	Input	1 Kohm pull-down
13,15,17, 19,21,23, 25	*reserved*	Output	33 Ohm series resistor

Table 7: 26-pin connector pin assignment

Pin	*Signal*	*Target Input/Output*	*Target Termination*
1	*reserved*		
3	JTDI	Input	1 Kohm pull-up resistor
5	JTDO	Output	33 Ohm series resister
7	JTMS	Input	1 Kohm pull-up resistor
9	JTCK	Input	1 Kohm pull-up resistor

Table 8: 10-pin connector pin assignment

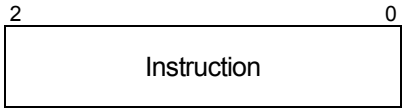
In addition to the above debug port connector, system designers may want to add similar connectors to allow probe access to additional signals. The socket/connectors should be placed somewhere adjacent to the CPU chip to minimize trace length. This extra socket/connectors will enable the user to connect a logic analyzer pre-processor to the target board without having to remove (i.e. de-solder) the CPU to insert the pre-processor between the board and CPU. The pre-processor provides full visibility to all external CPU signals and real-time trace and inverse assembly with a standard logic analyzer.

JTAG Instructions and Register

The JTAG TAP is controlled via JTAG instructions and registers.

Instruction Register

The Instruction Register provides the operation code for the JTAG control logic. Instructions are entered into the test logic during an instruction register scan sequence in the TAP controller. Table 9 is a summary of the instructions supported in the processor JTAG logic.



Instruction	*Code*	*Data Register*	*Function*
EXTEST	000	Boundary Scan	For testing circuitry external to the chip.
reserved	001-101	*reserved*	
SAMPLE/ PRELOAD	110	Boundary Scan	Allows data to be preloaded into the parallel outputs of the boundary scan register prior to another instruction such as EXTEST.
BYPASS	111	Bypass	Connect JTDI to JTDO through the one bit Bypass Registers,without interfere the normal operation of the chip and the running code.

Table 9: JTAG Instructions

ELECTRICAL SPECIFICATIONS

Symbol	*Parameter*	*Test Conditions*	*Minimum*	*Maximum*	*Units*
T_{TERM}	Terminal Voltage with respect to V_{SS}		-0.5	+3.9	V
T_{STG}	Storage Temperature		-55	125	^{o}C
T_{CASE}	Operating Temperature	Case Temperature		95	^{o}C
I_{IN}	DC Input Current			20	mA
I_{OUT}	DC Output Current			50	mA

Notes:

1. Operating conditions will be further refined in subsequent revisions of this document.

Recommended Operating Conditions

Symbol	*Parameter (V)*	*Core Frequency*	*Minimum, Maximum*	*Bus Freq*
VccInt VccP	Core Voltage	CPU speed 550-600MHz	1.80V ± 80mV	
VccIO	I/O Supply Voltage		3.3V ± 5% or 2.5V ± 3%	≤ 133 MHz[1] ≤ 100 MHz[1]

Notes:
1. Assumes slew rate control bits (ModeROM bits 22, 23) are set to high slew rate. This is highly recommended for 2.5V I/O and is required to meet the designated maximum bus frequency. For 3.3V I/O the designated bus frequency is guaranteed only for the high slew rate setting, but the slow slew rate setting is provided for lightly loaded transmission lines to reduce potential over and under shoots.

DC Electrical Characteristics

Symbol	*Parameter*	*Conditions*	*Minimum*	*Maximum*	*Units*
V_{OH}	Output HIGH Voltage	I_{OH}= 20 μA	VccIO - 0.1		V
		I_{OH}= 4 mA	2.4		
V_{OL}	Output LOW Voltage	I_{OH}= 20 μA		0.1	
		I_{OH}= 4 mA		0.4	
V_{IH}	Input HIGH Voltage		0.7 x VccIO	VccIO + 0.3[1]	
V_{IL}	Input LOW Voltage		-0.3[1]	0.2 x VccIO	
I_{IN}	Input Current	V_{IN} = 0		± 20	μA
		V_{IN} = VccIO		± 20	
I_{LEAK}	Input Leakage			20	
I_{IOLEAK}	Input/Output Leakage			20	
C_{IN}	Input Capacitance			10	pF
C_{OUT}	Output Capacitance			10	

Notes:
1. The undershoot and overshoot condition must not exceed 15ns.

Table 10: 3.3 V Operating Parameters

Power Consumption

Parameter	*Conditions*[2]	*Clock Frequency*	
		550MHz	*600MHz*
Maximum Power	Stand-by (No SysAD activity)	TBD	TBD
	Maximum (No FPU operation)	TBD	TBD
	Maximum (Worst case instruction mix)	6W[1]	7W[1]

Notes:
1. These numbers are preliminary estimates and are liable to change.
2. Operating conditions will be further refined in subsequent revisions of this document.

TIMING SPECIFICATION

The following figure shows the **SysClock** timing parameters. The **SysClock** input must meet the maximum rise time t_{CR}, maximum fall time t_{CF}, minimum t_{CH} time, minimum t_{CL} time, and t_{JI} input jitter parameters for proper operation of the PLL.

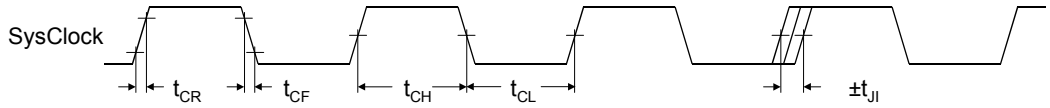


Figure 12. SysClock

Symbol	*Parameter*	*CPU Speed*				*Units*
		550MHz		*600MHz*		
		Min	*Max*	*Min*	*Max*	
t_{CYCLE}	SysClock cycle time	8.0	30.0	7.5	30.0	ns
t_{CR}	SysClock rise time		2.1		2.0	ns
t_{CF}	SysClock fall time		2.1		2.0	ns
t_{CH}	SysClock high time	3.2		3.0		ns
t_{CL}	SysClock low time	3.2		3.0		ns
t_{JI}	SysClock jitter in		± 0.19		± 0.18	ns

Table 11: SysClock Parameters

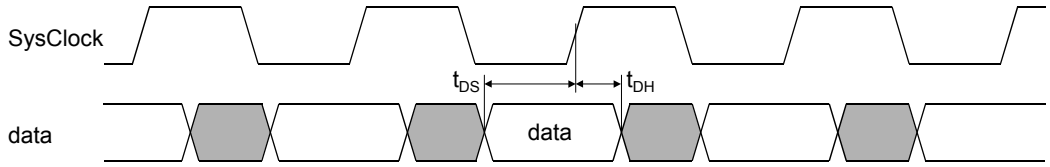


Figure 13. Input timing

Symbol	*Parameter*	*CPU Speed*				*Units*
		550MHz		*600MHz*		
		Min	*Max*	*Min*	*Max*	
t_{DS}	Data set up time (3.3V)[1]	2.1		2.0		ns
	Data set up time (2.5V)[1]	2.7		2.5		ns
t_{DH}	Data hold time (3.3V)[1]	0.9		0.8		ns
	Data hold time (2.5V)[1]	1.1		1.0		ns

Notes:
1. Timings are measured from 1.5V of clock to 1.5V of signal.
2. Capacitive load for all outputs is 30pF, except for **TcWord** and **TcLine** with 50pF.
3. Data output times are for all outputs including both tristate I/O and output only pins.

The following figure shows the output timing parameters measured at the midpoint of the rising clock edge.

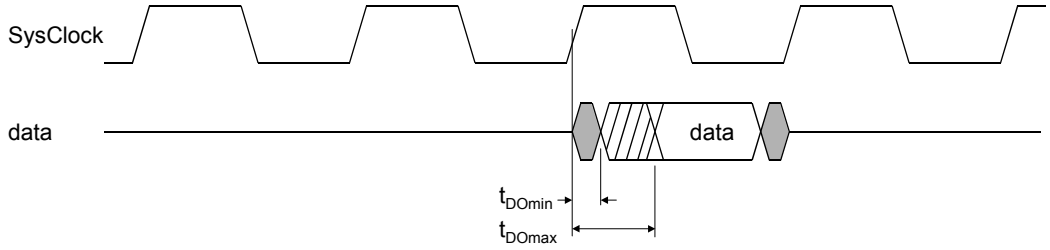


Figure 14. Output timing

Symbol	*Parameter*	*CPU Speed*				*Units*
		550MHz		*600MHz*		
		Min	*Max*	*Min*	*Max*	
t_{DO}	Data out time (3.3V, slow mode[4])	1.6	5.3	1.5	5.0	ns
	Data out time (3.3V, fast mode[4])	1.6	4.3	1.5	4.0	ns
	Data out time (2.5V, slow mode[4])	1.6	6.4	1.5	6.0	ns
	Data out time (2.5V, fast mode[4])	1.6	5.5	1.5	5.0	ns

Notes:
1. Timings are measured from 1.5V of clock to 1.5V of signal.
2. Capacitive load for all outputs is 30pF, except for **TcWord** and **TcLine** with 50pF.
3. Data output times are for all outputs including both tristate I/O and output only pins.
4. Output drive strengths are determined by bits 22 and 23 of the Mode ROM.

PINOUT AND PACKAGE

The following table gives the pin out for the SR71010 processor. DNC means 'Do Not Connect'.

Pin	*Name*	*Pin*	*Name*	*Pin*	*Name*	*Pin*	*Name*
A01	VccIO	B16	SysAD(30)	D08	VccInt	G22	SysAD(59)
A02	Vss	B17	SysAD(29)	D09	VccIO	G23	SysAD(58)
A03	Vss	B18	SysAD(28)	D10	SysAD(1)	H01	Vss
A04	TcLine(11)	B19	TcLine(5)	D11	VccInt	H02	SysAD(37)
A05	DNC[c]	B20	Vss	D12	VccIO	H03	SysAD(5)
A06	Vss	B21	Vss	D13	VccInt	H04	DNC[c]
A07	DNC[c]	B22	VccIO	D14	SysAD(31)	H20	VccInt
A08	Vss	B23	Vss	D15	VccIO	H21	SysAD(27)
A09	SysAD(32)	C01	Vss	D16	VccInt	H22	SysAD(26)
A10	SysADC(1)	C02	Vss	D17	TcLine(7)	H23	Vss
A11	TriState[b]	C03	VccIO	D18	VccIO	J01	SysAD(7)
A12	Vss	C04	VccIO	D19	VccIO	J02	SysAD(6)
A13	VccInt	C05	DNC[c]	D20	VccIO	J03	VccInt
A14	VccInt	C06	TcLine(9)	D21	VccIO	J04	VccIO
A15	SysAD(63)	C07	SysAD(3)	D22	Vss	J20	VccIO
A16	Vss	C08	SysAD(2)	D23	TcDCE(3) [a]	J21	VccInt
A17	SysAD(61)	C09	VccInt	E01	VccInt	J22	SysAD(57)
A18	Vss	C10	SysAD(0)	E02	TcLine(14)	J23	SysAD(56)
A19	DNC[c]	C11	SysADC(4)	E03	TcLine(12)	K01	SysAD(40)
A20	TcLine(4)	C12	VccInt	E04	VccIO	K02	SysAD(8)
A21	Vss	C13	SysADC(3)	E20	VccIO	K03	SysAD(39)
A22	Vss	C14	SysADC(2)	E21	TcCWE(2) [a]	K04	SysAD(38)
A23	VccIO	C15	SysAD(62)	E22	TcCWE(3) [a]	K20	SysAD(25)
B01	Vss	C16	VccInt	E23	TcLIne(1)	K21	SysAD(24)
B02	VccIO	C17	SysAD(60)	F01	Vss	K22	SysAD(55)
B03	Vss	C18	TcLine(6)	F02	TcLine(16)	K23	SysAD(23)
B04	Vss	C19	TcDCE(2) [a]	F03	TcLine(15)	L01	SysAD(10)
B05	TcLine(10)	C20	VccIO	F04	VccIO	L02	SysAD(41)
B06	SysAD(35)	C21	VccIO	F20	VccIO	L03	SysAD(9)
B07	SysAD(34)	C22	Vss	F21	TcLine(3)	L04	VccInt
B08	VccInt	C23	Vss	F22	TcLine(0)	L20	VccInt
B09	SysAD(33)	D01	TcLine(13)	F23	Vss	L21	SysAD(54)
B10	SysADC(5)	D02	Vss	G01	SysAD(36)	L22	SysAD(22)
B11	SysADC(0)	D03	VccIO	G02	SysAD(4)	L23	SysAD(53)
B12	DNC[c]	D04	VccIO	G03	TcLine(17)	M01	Vss
B13	SysADC(7)	D05	VccIO	G04	VccInt	M02	SysAD(11)
B14	SysADC(6)	D06	VccIO	G20	TcLine(2)	M03	SysAD(42)
B15	DNC[c]	D07	TcLine(8)	G21	VccInt	M04	VccIO

Notes:

a. These output signals are tristated according to bit 18 of the Mode ROM.

b. These inputs will be disconnected and default internally to their deasserted value according to bit 18 of the Mode ROM

c. These signals are *Do-Not-Connect* pins that should not be connected externally.

Pin	Name	Pin	Name	Pin	Name	Pin	Name
M20	VccIO	U03	ModeClock	Y17	Int*(2)	AB09	VccInt
M21	SysAD(52)	U04	JTCK	Y18	VccIO	AB10	TcCWE*(0)
M22	SysAD(21)	U20	VccInt	Y19	VccIO	AB11	TcDCE*(0)
M23	Vss	U21	NMI*	Y20	VccIO	AB12	SysCmd(1)
N01	SysAD(43)	U22	Reset*	Y21	VccIO	AB13	SysCmd(3)
N02	VccInt	U23	ColdReset*	Y22	Vss	AB14	SysCmd(7)
N03	SysAD(12)	V01	Vss	Y23	Int*(7)	AB15	DNC[c]
N04	SysAD(44)	V02	JTDO	AA01	Vss	AB16	DNC[c]
N20	SysAD(19)	V03	JTMS	AA02	Vss	AB17	TcDOE*
N21	SysAD(51)	V04	VccIO	AA03	VccIO	AB18	Int*(0)
N22	VccInt	V20	VccIO	AA04	VccIO	AB19	Int*(4)
N23	SysAD(20)	V21	Int*(9)	AA05	DNC[c]	AB20	Vss
P01	SysAD(13)	V22	VccInt	AA06	TcMatch[d]	AB21	Vss
P02	SysAD(45)	V23	Vss	AA07	ValidOut*	AB22	VccIO
P03	SysAD(14)	W01	JTDI	AA08	SysClock	AB23	Vss
P04	VccInt	W02	VccIO	AA09	VccInt	AC01	VccIO
P20	VccInt	W03	DNC[c]	AA10	DNC[c]	AC02	Vss
P21	SysAD(49)	W04	VccIO	AA11	DNC[c]	AC03	Vss
P22	SysAD(18)	W20	VccIO	AA12	SysCmd(0)	AC04	RdType
P23	SysAD(50)	W21	Int*(6)	AA13	SysCmd(4)	AC05	WrRdy*
R01	SysAD(46)	W22	Int*(8)	AA14	SysCmd(8)	AC06	Vss
R02	SysAD(15)	W23	VccInt	AA15	TcTCE*	AC07	VssP
R03	SysAD(47)	Y01	DNC[c]	AA16	DNC[c]	AC08	Vss
R04	VccIO	Y02	Vss	AA17	VccInt	AC09	TcWord(1)
R20	VccIO	Y03	VccIO	AA18	Int*(3)	AC10	TcCWE*(1)
R21	SysAD(16)	Y04	VccIO	AA19	DNC[c]	AC11	TcDCE*(1)
R22	SysAD(48)	Y05	VccIO	AA20	VccIO	AC12	Vss
R23	SysAD(17)	Y06	VccIO	AA21	VccIO	AC13	SysCmd(2)
T01	Vss	Y07	RdRdy*	AA22	Vss	AC14	SysCmd(6)
T02	RspSwap*	Y08	Release*	AA23	Vss	AC15	SysCmdP
T03	PReq*	Y09	VccIO	AB01	Vss	AC16	Vss
T04	VccInt	Y10	TcWord(0)	AB02	VccIO	AC17	DNC[c]
T20	ExtRqst*	Y11	VccInt	AB03	Vss	AC18	Vss
T21	VccOK	Y12	VccIO	AB04	Vss	AC19	Int*(1)
T22	BigEndian	Y13	SysCmd(5)	AB05	ModeIn	AC20	Int*(5)
T23	Vss	Y14	VccInt	AB06	ValidIn*	AC21	Vss
U01	PAck*	Y15	VccIO	AB07	VccP	AC22	Vss
U02	VccInt	Y16	VccInt	AB08	VccInt	AC23	VccIO

Notes:

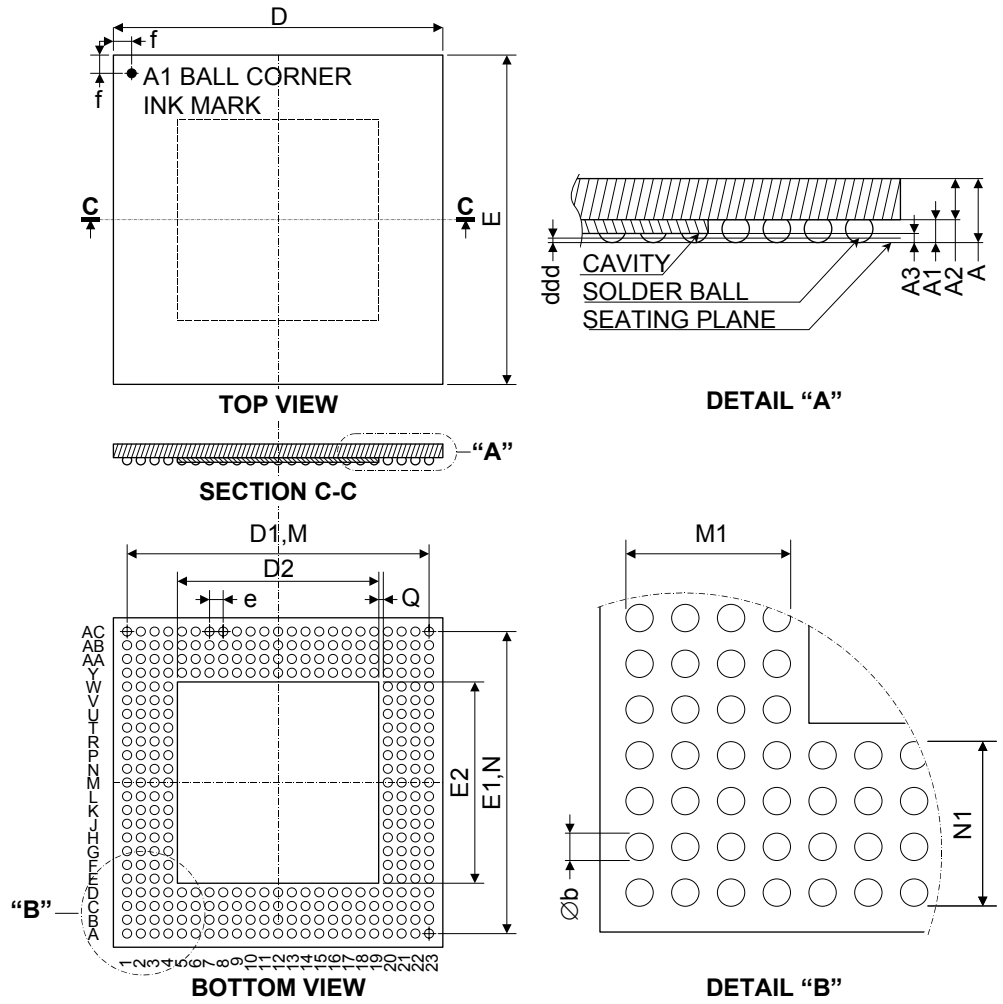
c. These signals are *Do-Not-Connect* pins that should not be connected externally.

d. This signal is an input on the RM7000 and is derived from the external tertiary cache tag RAMs. It is used to indicate tertiary cache hits to both the RM7000 and memory controller On SR71010, this signal is an output, due to the internal tertiary cache tag RAMs. It is used to drive the memory controller input.

IMPORTANT NOTE: If the SR71010 is to be used in an existing RM7000 socket, any external tertiary cache RAMs must be removed from the system board to prevent drive fights on this signal.

Table 12: Package Pinout

Pin orientation for the 304-ball BGA package.

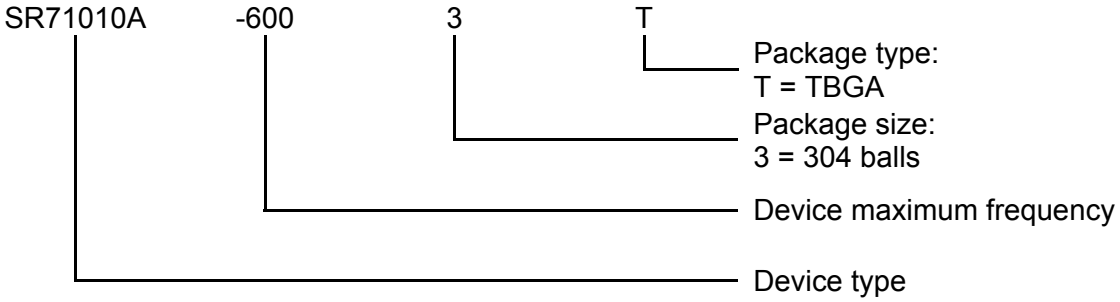


Symbol	*Min*	*Nominal*	*Max*	*Note*
A	1.33	1.50	1.67	Overall thickness
A1	0.50	0.60	0.70	Ball height
A2	0.83	0.90	0.97	Body thickness
A3	0.20			Encapsulation clearance
D,E		31.00 BSC		Body size
D1,E1		27.94		Ball footprint
D2,E2			18.57	Encapsulation size
b	0.60	0.75	0.90	Ball diameter
e		1.27		Ball pitch
ddd		0.20		Coplanarity
f		1.27		Pin A1 reference ink mark
M,N		23		Ball matrix
M1,N1		4		Number of rows deep
θ_{JC}		0.3		°C/Watt
θ_{JA}		11[4]		°C/Watt @ 0 cfm air flow

Notes:
1. All dimensions in millimeters unless otherwise indicated.
2. There is a minimum clearance of 0.25mm between the edge of the solder ball and the body edge.
3. Reference document: JEDEC MO-192.
4. Estimated - will be further refined in subsequent revisions of this document.

Figure 15. 304 Pin Ball Grid Array

ORDERING INFORMATION



This data sheet covers the following part numbers:

SR71010A-550-3T
SR71010A-600-3T