Intel® Itanium™ Processor

Hardware Developer's Manual

August 2001

Document Number: 248701-002

Contents

Figures

Tables

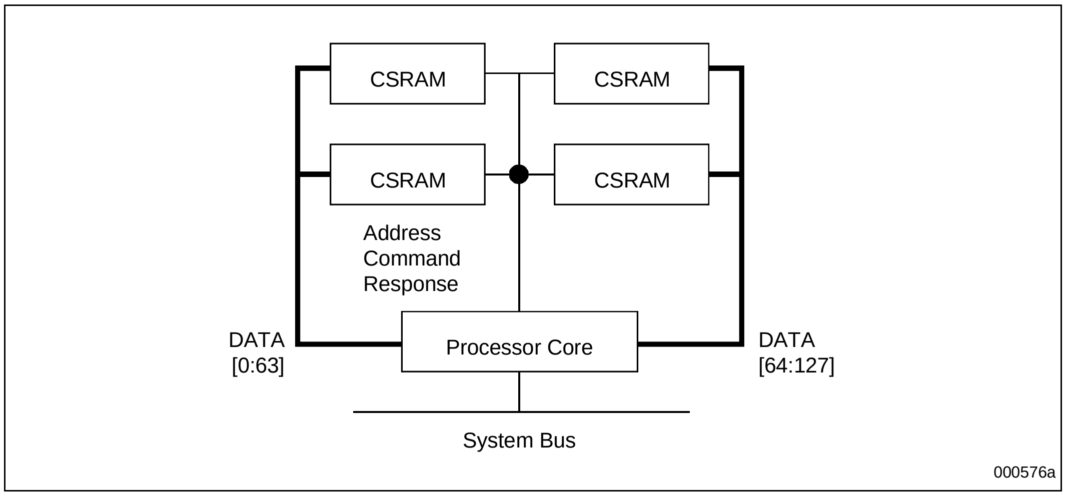
Introduction 1

The Intel® Itanium™ processor, the first in a family of processors based on Itanium architecture, is designed to address the needs of high-performance servers and workstations. The Itanium architecture goes beyond RISC and CISC approaches by pairing massive processing resources with intelligent compilers that enable parallel execution explicit to the processor. Its large internal resources combine with predication and speculation to enable optimization for high performance applications running on Windows* Advanced Server Limited Edition, Windows* XP 64-bit Edition, Linux, HP-UX* 11i v1.5 and other Itanium-based operating systems. The Itanium processor is designed to support very large scale systems, including those employing several thousand processors. It will provide the processing power and performance head room for back-office data-intensive servers, internet servers, and large data set computation intensive applications for high-end workstations. SMBus compatibility and comprehensive RAS features make the Itanium processor ideal for applications that demand continuous operation for maximum reliability and availability. In addition, the Itanium processor is fully compatible, in hardware, with IA-32 instruction binaries to preserve existing software investments. For high performance servers and workstations, the Itanium processor offers outstanding performance and reliability in targeted application segments and the scalability to address the growing e-business needs of tomorrow.

1.1 Intel® Itanium™ Processor 4 MB Cartridge

The Itanium processor 4 MB cartridge contains the processor core and 4 MB of Level 3 (L3) cache (four 1 MB Intel Cache SRAMs). The high speed L3 cache bus is completely isolated in the Itanium processor cartridge and operates at the same frequency as the processor core. Figure 1-1 shows the block diagram for the Itanium processor 4 MB cartridge.

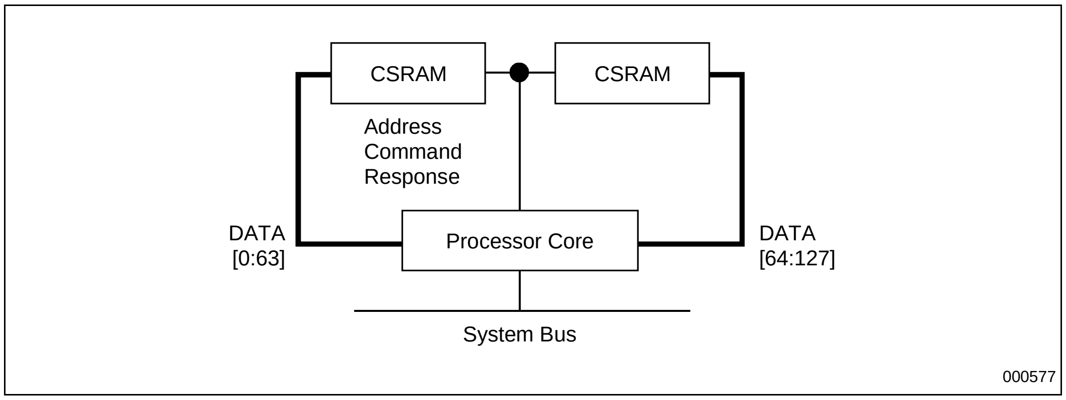
Figure 1-1. Intel® Itanium™ Processor 4 MB Cartridge Block Diagram



1.2 Intel® Itanium™ Processor 2 MB Cartridge

The Itanium processor 2 MB cartridge contains the processor core and 2 MB of L3 cache (two 1 MB Intel Cache SRAMs). The high speed L3 cache bus is completely isolated in the Itanium processor cartridge. Figure 1-2 shows the block diagram for the Itanium processor 2 MB cartridge.

Figure 1-2. Intel® Itanium™ Processor 2 MB Cartridge Block Diagram



1.3 Intel® Itanium™ Processor System Data Bus

The system bus signals use an enhanced version of the low voltage AGTL+ (Advanced Gunning Transceiver Logic) signaling technology used by the Pentium® III and Pentium III Xeon™ processors. For the highest performance, the system bus supports source synchronous data transfers. The system bus signals require external termination on each end of the signal trace to help supply the high signal level and to control reflections on the transmission line. Maximum system data bus throughput is 2.1 GB/sec.

1.4 Processor Abstraction Layer

The Itanium processor cartridge functionality requires the Processor Abstraction Layer (PAL) firmware. The PAL firmware resides in the system flash memory and is part of the Intel Itanium architecture. This firmware provides an abstraction level between the processor hardware implementation and the system platform firmware to maintain a single software interface for multiple implementations of the processor silicon steppings. PAL firmware encapsulates those processor functions that may change from one implementation to another so that the System Abstraction Layer (SAL) can maintain a consistent view of the processor.

The System Abstraction Layer (SAL) consists of the platform dependent firmware. SAL is the Basic Input/Output System (BIOS) required to boot the operating system (OS). The *Intel® Itanium™ Architecture Software Developer's Manual, Vol. 2: System Architecture* describes the PAL interface in detail.

1.5 Terminology

In this document, a '#' symbol after a signal name refers to an active low signal. This means that a signal is in the active state (based on the name of the signal) when driven to a low level. For example, when RESET# is low, a processor reset has been requested. When NMI is high, a non-maskable interrupt has occurred. In the case of lines where the name does not imply an active state but describes part of a binary sequence (such as address or data), the '#' symbol implies that the signal is inverted. For example, D[3:0] = 'HLHL' refers to a hex 'A', and D [3:0] # = 'LHLH' also refers to a hex 'A' (H= High logic level, L= Low logic level).

The term 'system bus' refers to the interface between the processor, system core logic and other bus agents. The system bus is a multiprocessing interface to processors, memory and I/O. The L3 cache does NOT connect to the system bus, and is not accessible by other agents on the system bus. Cache coherency is maintained with other agents on the system bus through the MESI cache protocol as supported by the HIT# and HITM# bus signals.

The term "Intel Itanium processor" refers to the cartridge package which interfaces to a host system board through a PAC418 connector. Intel Itanium processors include a processor core, an L3 cache, and various system management features. The Intel Itanium processor includes a thermal plate for a cooling solution attachment.

1.6 References

The reader of this manual should also be familiar with material and concepts presented in the following documents and tools:

- *Intel® Itanium™ Architecture Software Developer's Manual, Volume 1-4* (Document Number: 245317, 245318, 245319, 245320)
- *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet* (Document Number: 249634)
- *Itanium™ Processor Family Error Handling Guide* (Document Number: 249278)
- *Itanium™ Processor Microarchitecture Reference* (Document Number: 245473)
- *IEEE Standard Test Access Port and Boundary-Scan Architecture*
- *PAC418 VLIF Socket and Cartridge Ejector Design Specification*
- *PAC418 Cartridge/Power Pod Retention Mechanism and Triple Beam Design Guide*
- *Itanium™ Processor Heatsink Guidelines*

Note: Contact your Intel representative for the latest revision of the documents without document numbers.

1.6.1 Revision History

Version Number	Description	Date
001	Initial release of the Intel® Itanium™ Processor Hardware Developer's Manual.	May 2001
002	Updated Section 1: Introduction.	August 2001

Introduction to Microarchitecture 2

This chapter provides an introduction to the Intel Itanium processor microarchitecture. For detailed information on Itanium architecture, please refer to the *Intel® Itanium™ Architecture Software Developer's Manual*. For more detailed description of Itanium processor microarchitecture, please refer to the *Itanium™ Processor Microarchitecture Reference*.

2.1 Overview

The Intel Itanium processor is the first implementation of the Itanium Instruction Set Architecture (ISA). The processor employs EPIC (Explicitly Parallel Instruction Computing) design concepts for a tighter coupling between hardware and software. In this design style, the interface between hardware and software is designed to enable the software to exploit all available compile-time information, and efficiently deliver this information to the hardware. It addresses several fundamental performance bottlenecks in modern computers, such as memory latency, memory address disambiguation, and control flow dependencies. The EPIC constructs provide powerful architectural semantics, and enable the software to make global optimizations across a large scheduling scope, thereby exposing available Instruction Level Parallelism (ILP) to the hardware. The hardware takes advantage of this enhanced ILP, and provides abundant execution resources. Additionally, it focuses on dynamic run-time optimizations to enable the compiled code schedule to flow through at high throughput. This strategy increases the synergy between hardware and software, and leads to higher overall performance.

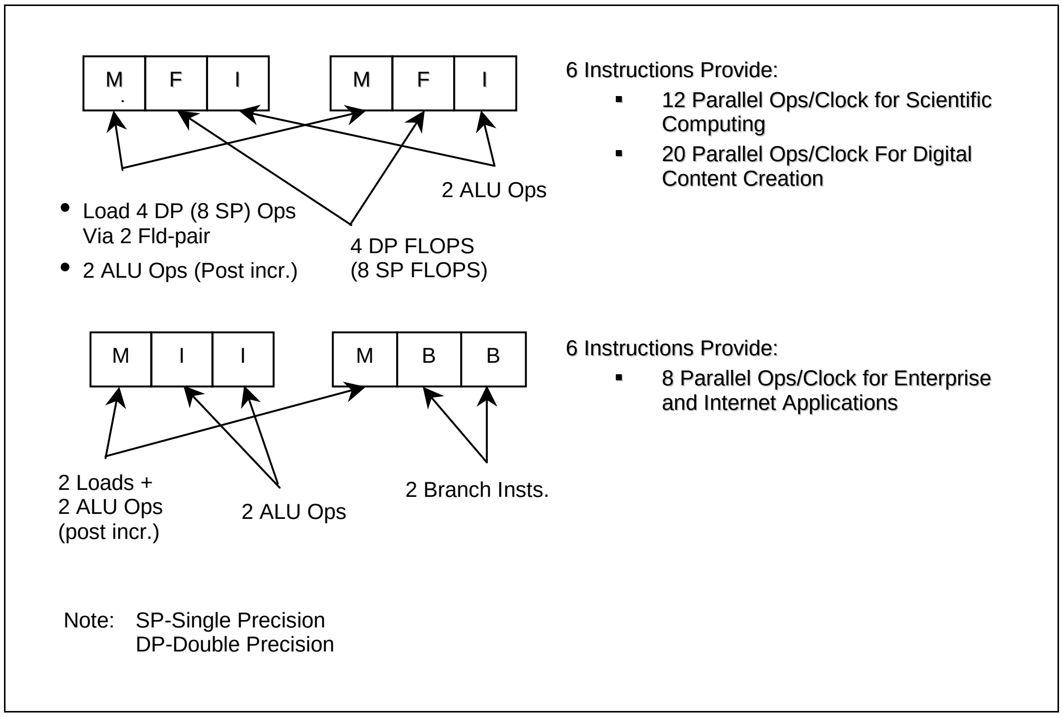
The Itanium processor provides a 6-wide and 10-stage deep pipeline, running at 733 and 800 MHz. This provides a combination of both abundant resources to exploit ILP as well as high frequency for minimizing the latency of each instruction. The resources consist of 4 integer units, 4 multimedia units, 2 load/store units, 3 branch units, 2 extended-precision floating point units, and 2 additional single-precision floating point units. The hardware employs dynamic prefetch, branch prediction, a register scoreboard, and non-blocking caches to optimize for compile-time non-determinism. Three levels of on-package cache minimize overall memory latency. This includes a 4MB L3 cache, accessed at core speed, providing over 12GB/sec of data bandwidth. The system bus is designed for glueless MP support for up to 4-processor systems, and can be used as an effective building block for very large systems. The advanced FPU delivers over 3 GFLOPS of numerics capability (6 GFLOPS for single-precision). The balanced core and memory subsystem provide high performance for a wide range of applications ranging from commercial workloads to high performance technical computing.

2.1.1 6-Wide EPIC Core Delivers a New Level of Parallelism

The processor provides hardware for the following execution units - 4 integer ALUs, 4 Multimedia ALUs, 2 Extended Precision FP Units, 2 additional Single Precision FP Units, 2 load/store units and 3 branch units. The machine can fetch, issue, execute and retire 6 instructions each clock. Given the powerful semantics of the Itanium instructions, this expands out to many more operations being executed each cycle.

Figure 2-1 illustrates two examples demonstrating the level of parallel operation supported for various workloads. For enterprise and commercial codes, the MII/MBB template combination in a bundle pair provides 6 instructions or 8 parallel ops per clock (2 load/store, 2 general-purpose ALU ops, 2 post-increment ALU ops, and 2 branch instructions). Alternatively, an MIB/MIB pair allows the same mix of operations, but with 1 branch hint and 1 branch op, instead of 2 branch ops. For scientific code, the use of the MFI template in each bundle enables 12 parallel Ops per clock (loading 4 double-precision operands to the registers, executing 4 double-precision flops, 2 integer ALU ops and 2 post-increment ALU ops). For digital content creation codes that use single precision floating point, the SIMD features in the machine effectively provide the capability to perform up to 20 parallel ops per clock (loading 8 single precision operands, executing 8 single precision FLOPs, 2 integer ALUs, and 2 post-incrementing ALU operations).

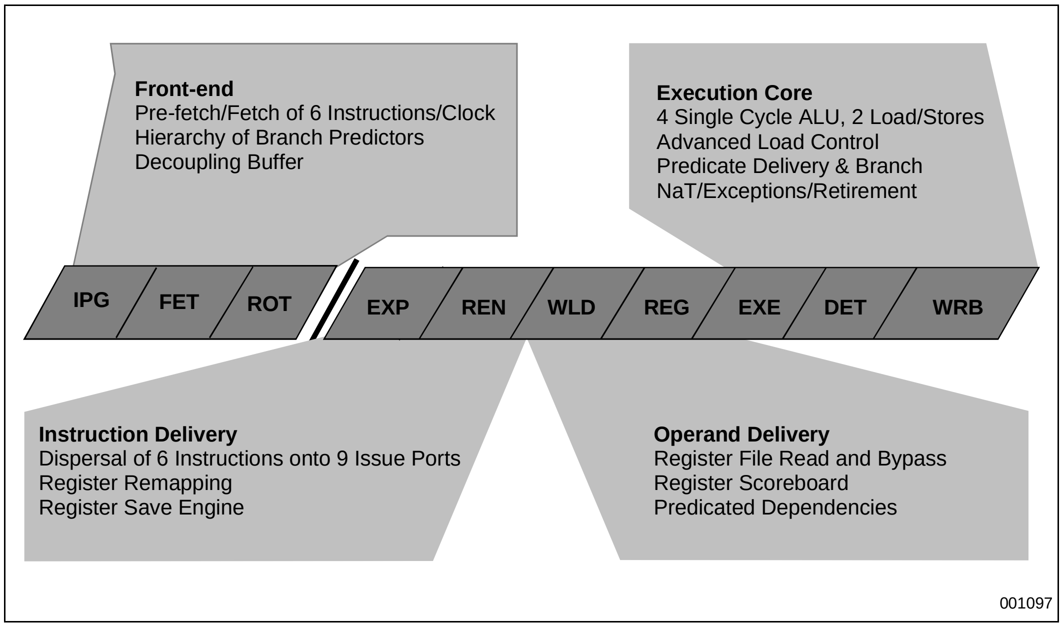
Figure 2-1. Two Examples Illustrating Supported Parallelism


M
F
I
M
F
I
6 Instructions Provide:
▪ 12 Parallel Ops/Clock for Scientific Computing
▪ 20 Parallel Ops/Clock For Digital Content Creation
2 ALU Ops
• Load 4 DP (8 SP) Ops Via 2 Fld-pair
• 2 ALU Ops (Post incr.)
4 DP FLOPS (8 SP FLOPS)
M
I
I
M
B
B
6 Instructions Provide:
▪ 8 Parallel Ops/Clock for Enterprise and Internet Applications
2 Loads + 2 ALU Ops (post incr.)
2 ALU Ops
2 Branch Insts.
Note: SP-Single Precision
DP-Double Precision

2.1.2 Processor Pipeline

The processor hardware is organized into a ten stage core pipeline, shown in Figure 2-2, that can execute up to six instructions in parallel each clock. The first three pipeline stages perform the instruction fetch and deliver the instructions into a decoupling buffer in the ROT (instruction rotation) stage that enables the front-end of the machine to operate independently from the back end. The bold line in the middle of the core pipeline indicates a point of decoupling. Dispersal and register renaming are performed in the next two stages, EXP (expand) and REN (register rename). Operand delivery is accomplished across the WLD (wordline decode) and REG (register read) stages, where the register file is accessed and data is delivered through the bypass network after processing the predicate control. Finally, the last three stages perform the wide parallel execution followed by exception management and retirement. In particular, the DET (exception detection) stage accommodates delayed branch execution as well as memory exception management and speculation support.

Figure 2-2. Itanium™ Processor Core Pipeline

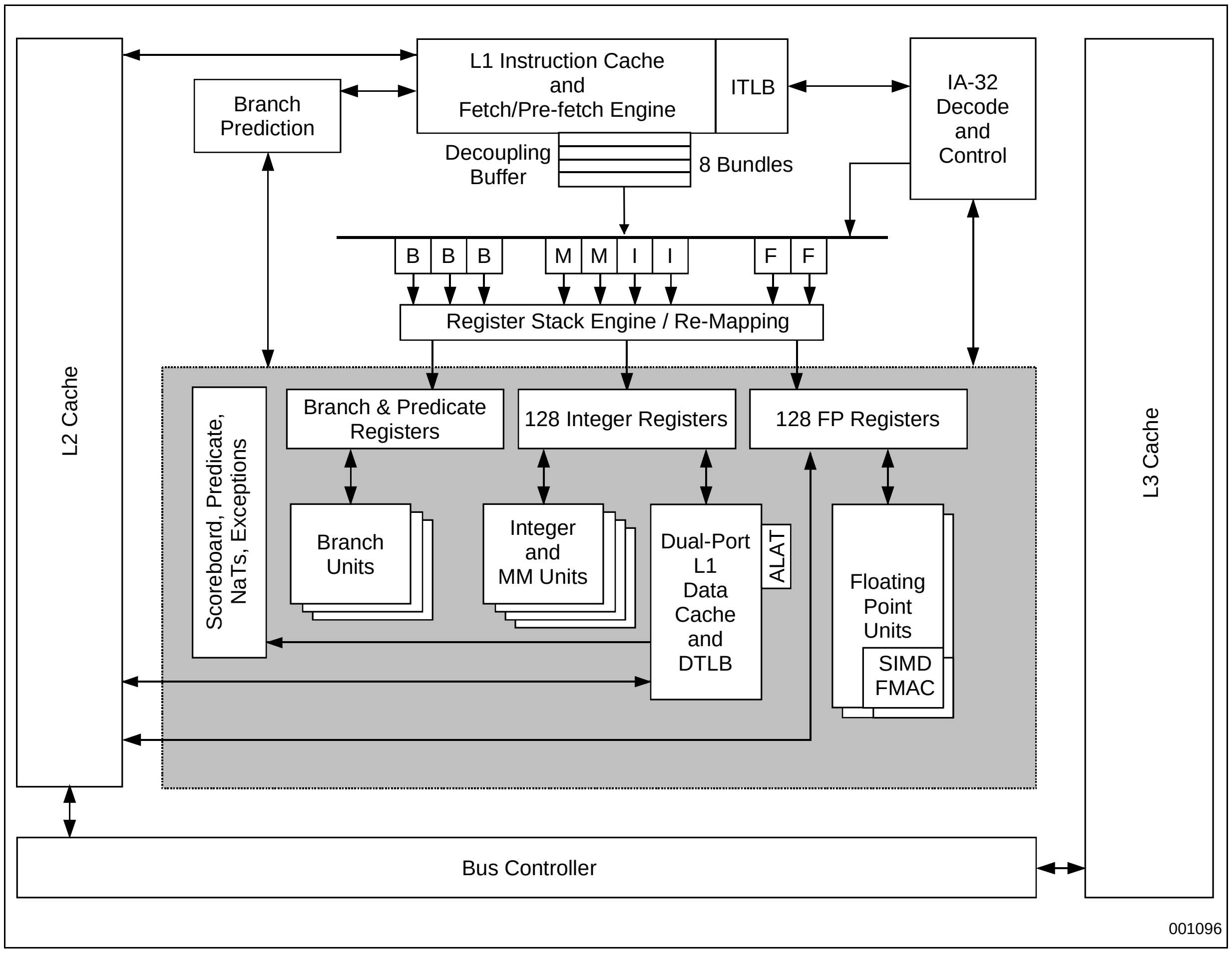

Front-end
Pre-fetch/Fetch of 6 Instructions/Clock
Hierarchy of Branch Predictors
Decoupling Buffer
Execution Core
4 Single Cycle ALU, 2 Load/Stores
Advanced Load Control
Predicate Delivery & Branch
NaT/Exceptions/Retirement
IPG
FET
ROT
EXP
REN
WLD
REG
EXE
DET
WRB
Instruction Delivery
Dispersal of 6 Instructions onto 9 Issue Ports
Register Remapping
Register Save Engine
Operand Delivery
Register File Read and Bypass
Register Scoreboard
Predicated Dependencies
001097

2.1.3 Processor Block Diagram

Figure 2-3 shows the block diagram of the Itanium processor. The function of the processor is divided into five groups, each summarized below. The following sections give a high-level description of the operation of each group.

1. **Instruction Processing**
   The instruction processing group contains the logic for instruction prefetch and fetch, branch prediction, decoupling buffer, register stack engine/remapping.
2. **Execution**
   The execution group contains the logic for integer, floating-point (FP), multimedia, branch execution units, and the integer and FP register files.
3. **Control**
   The control group contains the exception handler and pipeline control.
4. **Memory Subsystem**
   The memory subsystem group contains the L1 instruction cache (L1I), the L1 data cache (L1D), the unified L2 cache, the unified L3 cache, the Programmable Interrupt Controller (PIC), the Advanced Load Address Table (ALAT), and the system bus and backside bus logic.
5. **IA-32 Execution**
   The IA-32 execution group contains hardware for handling IA-32 instructions.

Figure 2-3. Itanium™ Processor Block Diagram



2.2 Instruction Processing

2.2.1 Instruction Prefetch and Fetch

Acting in conjunction with sophisticated branch prediction and correction hardware, the Itanium processor speculatively fetches instructions from a moderately sized and pipelined L1 instruction cache (L1I) into a decoupling buffer. This buffer allows the front end to speculatively fetch ahead and hide the instruction cache and branch prediction latencies. A hierarchy of branch predictors, aided by Branch Hints provided by the compiler, provides up to 4 progressively improving instruction pointer resteers. Software-initiated prefetch probes for future misses in the instruction cache and then prefetches such target code from the L2 cache into a streaming buffer and eventually into the instruction cache.

The 16KB, 4-way set-associative instruction cache is fully pipelined, and can deliver 32B of code (two instruction bundles or 6 instructions) every clock. It is supported by a single-cycle 64-entry Instruction TLB that is fully-associative and backed up by an on-chip hardware page walker. The fetched code is fed into a decoupling buffer that can hold 8 bundles of code. During instruction issue, instructions read from the decoupling buffers are sent to the instruction issue and rename logic based on the availability of execution resources.

2.2.2 Branch Prediction

The processor employs a hierarchy of branch prediction structures to deliver high-accuracy and low-penalty predictions across a wide spectrum of workloads. The branch prediction hardware is assisted by Branch Hint directives provided by the compiler (in the form of explicit BRP instructions, as well as hint specifiers on branch instructions). The directives provide branch target addresses, static hints on branch direction, as well as indications on when to use dynamic prediction. These directives are programmed into the branch prediction structures and used in conjunction with dynamic prediction schemes. The machine provides up to 4 progressive predictions and corrections to the fetch pointer. These 4 resteers are:

Resteer-1: Special single-cycle branch predictor (uses compiler programmed Target Address Registers).

Resteer-2: Adaptive Two-level Multi-way Predictor, & Return Predictor:

Resteers-3,4: Branch Address Calculation & Correction (BAC1, BAC2):

2.2.3 Dispersal Logic

The processor has a total of 9 issue ports capable of issuing up to 2 memory instructions (ports M0 & M1), 2 integer (ports I0 & I1), 2 floating-point (ports F0 & F1), and 3 branch instructions (ports B0, B1, and B2) per clock. The 17 execution units in the processor are fed through the M, I, F, and B groups of issue ports.

The decoupling buffer feeds the dispersal in a bundle granular fashion (up to 2 bundles or 6 instructions per cycle), with a fresh bundle being presented each time one is consumed. Dispersal from the two bundles is instruction granular – the processor disperses as many instruction as can be issued (up to 6), in left-to-right order. The dispersal algorithm is fast and simple, with instructions being dispersed to the first available issue port, subject to two constraints - detection of instruction independence, and detection of resource oversubscription.

2.3 Execution

The Itanium processor has the following execution units: four integer, four multimedia, two extended-precision floating-point and two additional single-precision floating-point, three branch, and two load/store units. The processor implements 128 integer and 128 floating point registers, and eight branch registers.

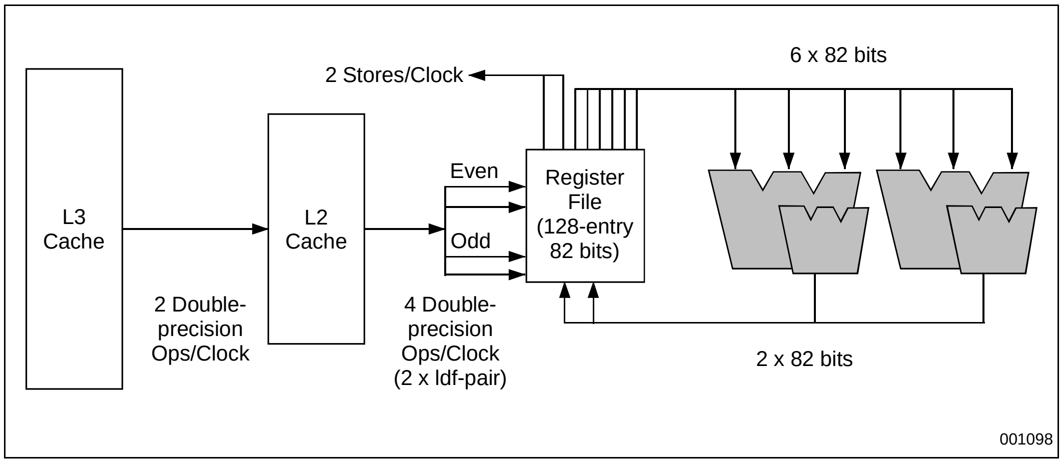
The integer engines support all non-packed integer arithmetic and logical operations. The multimedia engines can treat 64-bit data as either 2 x 32-bit, 4 x 16-bit, or 8 x 8-bit packed data types. Four integer or multimedia operations can be executed each cycle. The floating-point engines support simultaneous multiply-add to provide performance for scientific computation.

2.3.1 The Floating-point Unit (FPU)

The floating-point unit (FPU) delivers up to 6.4 Gflops and provides full support for single, double, extended, and mixed mode precision computations. Parallel FP instructions which operate on pairs of 32-bit single precision numbers increase the single-precision FP computation throughput. The processor also supports multimedia instructions that treat the general registers as vectors of eight 8-bit, four 16-bit, or two 32-bit elements. The FPU execution hardware primarily consists of a low latency floating-point multiply-add (FMAC) primitive with high register and memory bandwidth, which is an effective building block for scientific computation. At peak, two FP instructions, two integer instructions, and two integer or FP loads/stores can be issued every clock.

The FPU has a 128-entry FP register file with eight read and four write ports supporting full bandwidth operation. See Figure 2-4. Every cycle, the eight read ports can feed two extended-precision FMACs (each with three operands) as well as two floating-point stores to memory. The four write ports can accommodate two extended-precision results from the two MAC units and the results from two load instructions each clock. To increase the effective write bandwidth into the FPU from memory, the floating-point registers are divided into odd and even backs. This enables the two physical ports dedicated to load returns to be used to write four values pre clock to the register file (two to each bank), using two ldf-pair instructions. The earliest cache level to feed the FPU is the unified L2 cache. The latency of loads from this cache to the FPU is nine clock cycles. For data beyond the L2 cache, the bandwidth to the L3 cache is two double-precision operations per clock (one 64-byte line every four clock cycles)

Figure 2-4. FMAC Units Deliver 8 Flops/Clock


6 x 82 bits
2 Stores/Clock
L3 Cache
L2 Cache
Even
Odd
Register File (128-entry 82 bits)
2 Double-precision Ops/Clock
4 Double-precision Ops/Clock (2 x ldf-pair)
2 x 82 bits
001098

2.3.2 The Integer Logic

The integer execution engine includes four Arithmetic Logic Units (ALUs) and memory ports used to issue loads or stores. All common operations are fully bypassed.

2.3.3 The Integer Register File

The integer register file is divided into static and stacked subsets. The static subset is visible to all procedures and consists of 32 general registers from GR0 through GR31. GR0 always returns 0. The stacked subset local to each procedure begins at GR32 and may vary in size from zero to 96 registers under software control. The register stack mechanism is implemented by renaming register addresses as a side-effect of procedure calls and returns. The implementation of this rename mechanism is not visible to application programs.

2.3.4 The Register Stack Engine (RSE)

The Itanium processor eliminates overhead associated with call/return by avoiding the spilling and filling of registers at procedure interfaces through a large register file and register stack.When a procedure is called, a new frame of registers is made available to the called procedure without the need for an explicit save of the caller's registers. The old registers remain in the large physical register file as long as there is enough physical capacity. When the number of registers needed overflows the available physical capacity, the Register Stack Engine (RSE) state machine is called to save registers to memory to free up the necessary registers needed for the upcoming call.

On a call return, the processor is reversed to restore access to the state prior to the call. In cases where the RSE has saved some of the callee's registers, the processor stalls on return until the RSE can restore the appropriate number of the callee's registers. The Itanium processor implements the forced lazy mode of the RSE.

2.4 Control

The control group of the Itanium processor is made up of the exception handler and the pipeline control. The exception handler implements exception prioritizing. The pipeline control uses a scoreboard to detect register source dependencies and also special support for control and data speculation as well as predication. For control speculation, the hardware manages the creation and propagation deferred exception tokens (called NaT). For data speculation, the processor provides an Advanced Load Address Table or ALAT (see Section 2.5.5). Predication support includes the management 64 1-bit predicate registers as well as the effects of predicates on instruction execution.

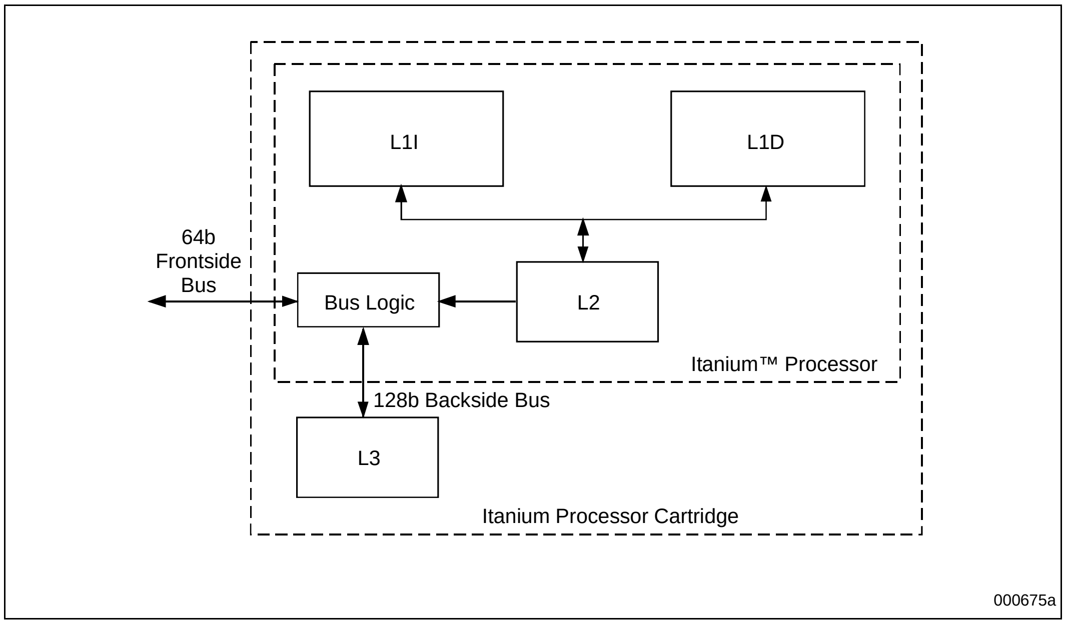
2.5 Memory Subsystem

The Itanium processor accesses main system memory through the system bus described in Chapter 3, "System Bus Overview". The memory subsystem group for the Itanium processor contains the L1 instruction cache (L1I), the L1 data cache (L1D), the unified L2 cache, the unified L3 cache, the Programmable Interrupt Controller (PIC), the Advanced Load Address Table (ALAT), and the system bus and backside bus logic (Figure 2-5).

The L2 cache contains instructions and data accessed at the full clock speed of the processor. The L2 cache can handle two requests per clock via banking if there are no conflict conditions. This cache is unified, allowing it to service both instruction and data side requests from the L1 caches. When a request to the L2 cache causes a miss, the request is automatically forwarded to the L3 cache.

The backside bus logic accesses the L3 cache through a 128-bit backside bus operating at the full clock speed of the processor. L3 cache misses are automatically forwarded to main system memory through the Itanium processor system bus.

Figure 2-5. Itanium™ Processor Memory Subsystem



2.5.1 L1 Instruction Cache

The Itanium processor L1 instruction (L1I) cache is 16 KB in size and organized as 4-way set-associative with a 32B line size. L1I is fully pipelined and can deliver a 32B line containing two instruction bundles (six instructions) every clock. The L1I cache is physically indexed and physically tagged.

2.5.2 L1 Data Cache

The Itanium processor L1 data (L1D) cache is dual-ported 16 KB in size and is organized as 4-way set-associative (with way prediction) with a 32B line size and two-cycle load latency. It can support two concurrent loads or stores. The L1 data cache only caches data for the integer unit, not the floating-point unit. The L1D cache is write-through with no write allocation. The L1D cache is physically indexed and physically tagged.

2.5.3 Unified L2 Cache

The Itanium processor unified L2 cache is pseudo-dual-ported and supports concurrent accesses via banking. The L2 cache is 96 KB, 6-way set-associative with a 64 B line size. The L2 cache uses a write-back with write-allocate policy. The L2 cache is physically indexed and physically tagged.

2.5.4 Unified L3 Cache

The Itanium processor unified L3 cache is available in 2 or 4 MB size and is organized as 4-way set-associative with a 64 B line size. The L3 cache is physically indexed and physically tagged. The L3 cache is accessed via a dedicated 128-bit back-side bus that runs at full processor core speed.

2.5.5 The Advanced Load Address Table (ALAT)

A structure called the Advanced Load Address Table (ALAT) is used to enable data speculation in the Itanium processor. The ALAT keeps information on speculative data loads issued by the processor and any stores that are aliased with these loads. The Itanium processor ALAT has 32 entries and is 2-way set-associative.

2.5.6 Translation Lookaside Buffers (TLBs)

There are three TLBs on the Itanium processor: first level Data Translation Lookaside Buffer (DTLB1), second level Data Translation Lookaside Buffer (DTLB2), and the Instruction Translation Lookaside Buffer (ITLB). TLB misses in either the Data Translation Lookaside Buffers (DTLB1 and DTLB2) or the ITLB are serviced by the hardware page table walker which supports the Itanium-defined 8B and 32B Virtual Hash Page Table (VHPT) format.

2.5.6.1 The Data TLB (DTLB)

The DTLB has a two-level TLB hierarchy (DTLB1 and DTLB2 are not inclusive):

1. The DTLB1 has 32 entries, is fully-associative and maintains cached copies of the main DTLB2. The DTLB1 is not architecturally visible.
2. The DTLB2 has 96 entries, is fully-associative, holds all architecturally defined page sizes, data Translation Register (TR) and data Translation Cache (TC) entries.
3. There are 48 data TRs on the Itanium processor.

2.5.6.2 The Instruction TLB (ITLB)

The ITLB is single-level:

1. The ITLB contains 64 entries and is fully associative.
2. The ITLB holds all architecturally defined page sizes, instruction TRs, and instruction TC entries. The ITLB has only one level of hierarchy.
3. There are eight instruction TRs on the Itanium processor.

2.5.7 Cache Coherency

A three-level cache system requires a mechanism for data consistency between different caches levels. Every read access to a memory address must provide the most current data at that address. The Itanium processor implements the MESI (Modified, Exclusive, Shared, and Invalid) state protocol to maintain cache coherency.

2.5.8 Write Coalescing

For increased performance of uncacheable references to frame buffers, the Write Coalescing (WC) memory type coalesces streams of data writes into a single larger bus write transaction. On the Itanium processor, WC loads are performed directly from memory and not from the coalescing buffers.

On the Itanium processor, a separate two-entry, 64-byte buffer (WCB) is used for WC accesses exclusively. The processor will evict (flush) each buffer if the buffer is full or if specific ordering constraints are met.

2.5.9 Memory Ordering

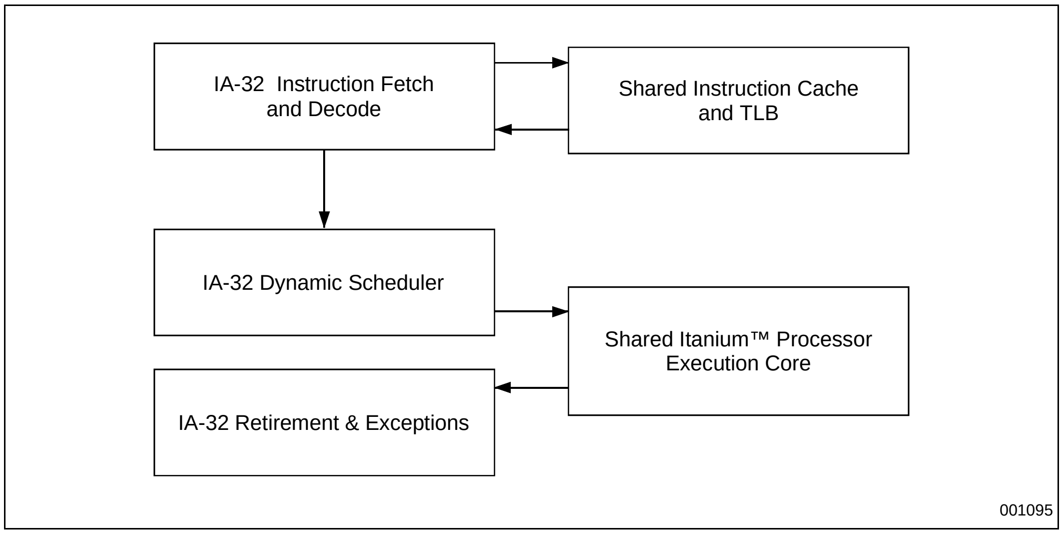
The Itanium processor implements a relaxed memory ordering model to enhance memory system performance. Memory transactions are ordered with respect to visibility whereby visibility of a transaction is defined as a point in time after which no later transactions may affect its operation.

On the Itanium processor, a transaction is considered visible when it hits the L1D (if the instruction is serviceable by L1D), the L2, or the L3, or when it has reached the visibility point on the system bus.

2.6 IA-32 Execution

Itanium processor supports IA-32 instruction binary compatibility in hardware (Figure 2-6). This includes support for running a mix of IA-32 applications and Itanium-based applications on an Itanium-based OS, as well as IA-32 applications on an IA-32 OS, in both uniprocessor and multiprocessor configurations. The IA-32 engine is designed to make use of the registers, caches, and execution resources of the EPIC machine. To deliver high performance on legacy binaries, the IA-32 engine dynamically schedules instructions.

Figure 2-6. IA-32 Compatibility Microarchitecture


IA-32 Instruction Fetch and Decode
Shared Instruction Cache and TLB
IA-32 Dynamic Scheduler
Shared Itanium™ Processor Execution Core
IA-32 Retirement & Exceptions
001095

System Bus Overview 3

This chapter provides an overview of the Itanium processor system bus, bus transactions, and bus signals. The Itanium processor also supports signals not discussed in this section. For a complete signal listing, please refer to the *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet*.

3.1 Signaling on the Itanium™ Processor System Bus

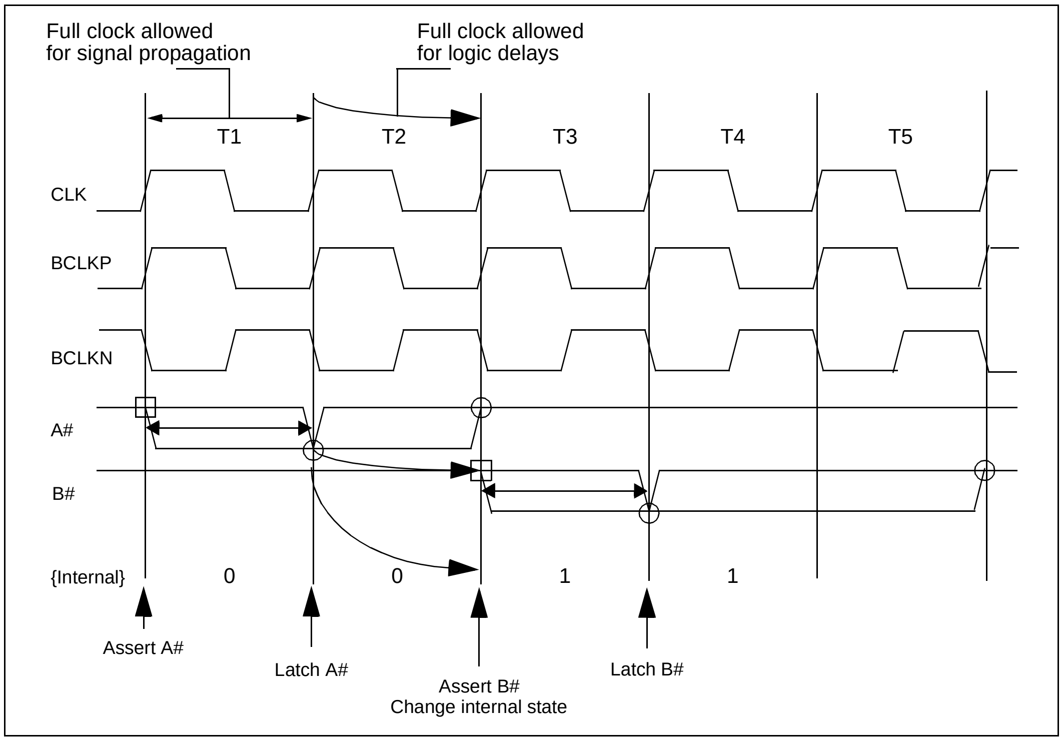
The Itanium processor system bus supports common clock signaling as well as source synchronous signaling for increased performance. Section 3.1.1 and Section 3.1.2 describe in detail the characteristics of each type of signaling. The corresponding timing figures, Figure 3-1 and Figure 3-2, use square, triangle, and circle symbols to indicate the point at which signals are driven, received, and sampled, respectively. The square indicates that a signal is driven (asserted or deasserted) in that clock. The triangle indicates that a signal is received on or before that point. The circle indicates that a signal is sampled (observed, latched, captured) in that clock.

3.1.1 Common Clock Signaling

In this mode, the system bus signaling uses a synchronous common clock latched protocol. On the rising edge of the bus clock, all agents on the system bus are required to drive their active outputs and sample required inputs. No additional logic is located in the output and input paths between the buffer and the latch stage, thus keeping setup and hold times constant for all bus signals following the latched protocol. The system bus requires that every input be sampled during a valid sampling window on a rising clock edge and its effect be driven out no sooner than the next rising clock edge. This approach allows one full clock for communication between system bus agents and at least one full clock at the receiver to compute a response.

Figure 3-1 illustrates the latched bus protocol as it appears on the bus. In subsequent descriptions, the protocol is described as “B# is asserted in the clock after A# is observed asserted,” or “B# is asserted two clocks after A# is asserted.” Note that A# is asserted in T1, but not observed asserted until T2. A# has one full clock to propagate (indicated by the straight line with arrows) before it is observed asserted. The receiving agent uses T2 to determine its response and asserts B# in T3. That is, the receiving agent has one full clock cycle from the time it observes A# asserted (at the rising edge of T2) to the time it computes its response (indicated by the curved line with the single arrow) and drives this response at the rising edge of T3 on B#. Similarly, an agent observes A# asserted at the rising edge of T2, and uses the full T2 clock to compute its response (indicated by the lowermost curved arrow during T2). This response would be driven at the rising edge of T3 (not shown in Figure 3-1) on {internal} signals. Although B# is driven at the rising edge of T3, it has the full clock T3 to propagate. B# is observed asserted in T4.

Figure 3-1. Common Clock Latched Protocol


Full clock allowed for signal propagation
Full clock allowed for logic delays
T1
T2
T3
T4
T5
CLK
BCLKP
BCLKN
A#
B#
{Internal}
0
0
1
1
Assert A#
Latch A#
Assert B# Change internal state
Latch B#

Signals that are driven in the same clock by multiple system bus agents exhibit a “wired-OR glitch” on the electrical low to electrical high transition. To account for this situation, these signal state transitions are specified to have two clocks of settling time when deasserted before they can be safely observed, as shown with B#. The bus signals that must meet this criterion are: BINIT#, HIT#, HITM#, BNR#, TND#, and BERR#.

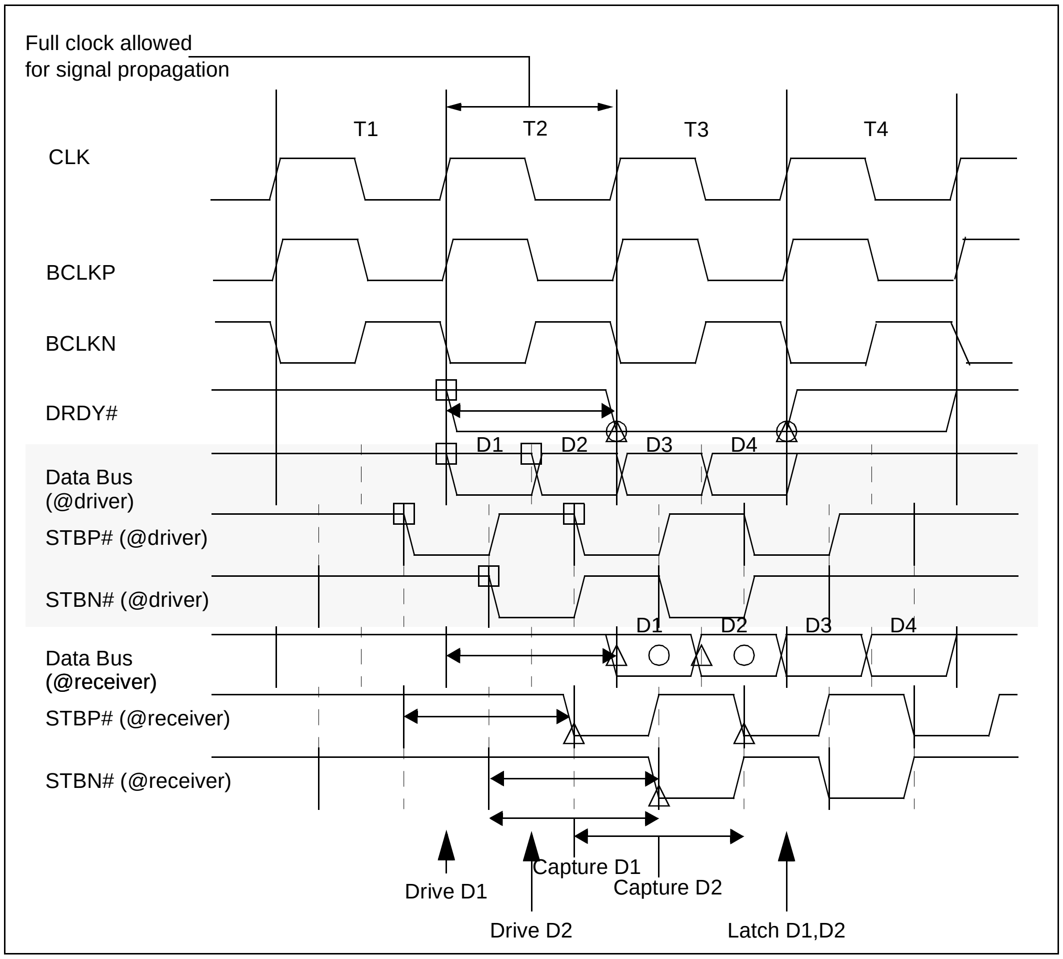
3.1.2 Source Synchronous Signaling

The data bus can also operate in a source synchronous latched protocol to achieve a 2x transfer rate. This source synchronous latched protocol is accomplished by sending and latching data with strobes. The rest of the system bus always uses the common clock latched protocol described in Section 3.1.1.

The source synchronous latched protocol operates the data bus at twice the “frequency” of the common clock. Two source synchronous data transfers are driven onto the bus in the time it would normally take to drive one common clock data transfer. At both the beginning and 50% points of the bus clock period, drivers send new data. At both the 25% point and the 75% point of the bus clock period, drivers send centered differential strobes. The receiver captures the data with the strobes deterministically.

The driver pre-drives STBP# before driving data. It sends a rising and falling edge on STBP# and STBN#, centered with data. The driver deasserts all strobes after the last data is sent. The receiver captures valid data with the difference of both strobe signals, asynchronous to the common clock. A signal synchronous to the common clock (DRDY#) indicates to the receiver that valid data has been sent.

Figure 3-2. Source Synchronous Latched Protocol



3.2 Signal Overview

This section describes the function of various Itanium processor signals. In this section, the signals are grouped according to function. For a complete signal listing, please refer to the *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet*.

3.2.1 Control Signals

The control signals, shown in Table 3-1, are used to control basic operations of the processor.

Table 3-1. Control Signals

Signal Function	Signal Names
Positive Phase Bus Clock	BCLKP
Negative Phase Bus Clock	BCLKN
Reset Processor and System Bus Agents	RESET#
Power Good	PWRGOOD

The BCLKP (Positive Phase Bus Clock) input signal is the positive phase of the system bus clock differential pair. It is also referred to as CLK in some of the waveforms in this overview. It specifies the bus frequency and clock period and is used in the signaling scheme. Each processor derives its internal clock from CLK by multiplying the bus frequency by a multiplier determined at configuration. See Chapter 5, "Configuration and Initialization" for further details.

The BCLKN (Negative Phase Bus Clock) input signal is the negative phase of the system bus clock differential pair.

The RESET# input signal resets all system bus agents to known states and invalidates their internal caches. Modified or dirty cache lines are NOT written back. After RESET# is deasserted, each processor begins execution at the power-on reset vector.

The PWRGOOD (Power Good) input signal must be deasserted during power-up and be asserted after RESET# is first asserted by the system.

3.2.2 Arbitration Signals

The arbitration signals, shown in Table 3-2, are used to arbitrate for ownership of the bus, a requirement for initiating a bus transaction.

Table 3-2. Arbitration Signals

Signal Function	Signal Names
Symmetric Agent Bus Request	BREQ[3:0]#
Priority Agent Bus Request	BPRI#
Block Next Request	BNR#
Lock	LOCK#

BR[3:0]# are the physical pins of the processor. All processors assert only BR0#. BREQ[3:0]# refers to the system bus arbitration signals among four processors. BR0# of each of the four processors is connected to a unique BREQ[3:0]# signal.

Up to five agents can simultaneously arbitrate for the request bus, one to four symmetric agents (on BREQ[3:0]#) and one priority agent (on BPRI#). Processors arbitrate as symmetric agents, while the priority agent normally arbitrates on behalf of the I/O agents and memory agents. Owning the request bus is a necessary pre-condition for initiating a transaction.

The symmetric agents arbitrate for the bus based on a round-robin rotating priority scheme. The arbitration is fair and symmetric. A symmetric agent requests the bus by asserting its BREQ*n*# signal. Based on the values sampled on BREQ[3:0]#, and the last symmetric bus owner, all agents simultaneously determine the next symmetric bus owner.

The priority agent asks for the bus by asserting BPRI#. The assertion of BPRI# temporarily overrides, but does not otherwise alter the symmetric arbitration scheme. When BPRI# is sampled asserted, no symmetric agent issues another unlocked transaction until BPRI# is sampled deasserted. The priority agent is always the next bus owner.

BNR# can be asserted by any bus agent to block further transactions from being issued to the request bus. It is typically asserted when system resources, such as address or data buffers, are about to become temporarily busy or filled and cannot accommodate another transaction. After bus initialization, BNR# can be asserted to delay the first transaction until all bus agents are initialized.

The assertion of the LOCK# signal indicates that the symmetric agent is executing an atomic sequence of transactions that must not be interrupted. A locked sequence cannot be interrupted by another transaction regardless of the assertion of BREQ[3:0]# or BPRI#. LOCK# can be used to implement memory-based semaphores. LOCK# is asserted from the start of the first transaction through the end of the last transaction. When locking is disabled, the LOCK# signal will never be asserted.

3.2.3 Request Signals

The request signals, shown in Table 3-3, are used to initiate a transaction.

Table 3-3. Request Signals

Signal Function	Signal Names
Address Strobe	ADS#
Request	REQ[4:0]#
Address	A[43:3]#
Address Parity	AP[1:0]#
Request Parity	RP#

The assertion of ADS# defines the beginning of the transaction. The REQ[4:0]#, A[43:3]#, AP[1:0]#, and RP# are valid in the clock that ADS# is asserted.

In the clock that ADS# is asserted, the A[43:3]# signals provide an active-low address as part of the request. The low three bits of address are mapped into byte enable signals for 0 to 8 byte transfers. AP1# covers the address signals A[43:24]#. AP0# covers the address signals A[23:3]#. A parity signal on the system bus is correct if there are an even number of electrically low signals in the set consisting of the covered signals plus the parity signal. Parity is computed using voltage levels, regardless of whether the covered signals are active high or active low.

The Request Parity (RP#) signal covers the request pins REQ[4:0]# and the address strobe, ADS#.

3.2.4 Snoop Signals

The snoop signals, shown in Table 3-4, are used to provide snoop results and transaction control to the system bus agents.

Table 3-4. Snoop Signals

Signal Function	Signal Names
Purge Global Translation Cache Not Done	TND#
Keeping a Non-Modified Cache Line	HIT#
Hit to a Modified Cache Line	HITM#
Defer Transaction Completion	DEFER#
Guarantee Sequentiality	GSEQ#

The TND# signal may be asserted by a bus agent to delay completion of a Purge Global Translation Cache (PTCG) instruction, even after the PTCG transaction completes on the system bus. Software will guarantee that only one PTCG instruction is being executed in the system.

The HIT# and HITM# signals are used to indicate that the line is valid or invalid in the snooping agent, whether the line is in the modified (dirty) state in the caching agent, or whether the transaction needs to be extended. The HIT# and HITM# signals are used to maintain cache coherency at the system level.

If the memory agent observes HITM# active, it relinquishes responsibility for the data return and becomes a target for the implicit cache line writeback. The memory agent must merge the cache line being written back with any write data and update memory. The memory agent must also provide the implicit writeback response for the transaction.

If HIT# and HITM# are sampled asserted together, it means that a caching agent is not ready to indicate snoop status, and it needs to extend the transaction.

The DEFER# signal is deasserted to indicate that the transaction can be guaranteed in-order completion. An agent asserting ensures proper removal of the transaction from the In-order Queue by generating the appropriate response.

The assertion of the GSEQ# signal allows the requesting agent to issue the next sequential uncached write even though the transaction is not yet visible. By asserting the GSEQ# signal, the platform also guarantees not to retry the transaction, and accepts responsibility for ensuring the sequentiality of the transaction with respect to other uncached writes from the same agent.

3.2.5 Response Signals

The response signals, shown in Table 3-5, are used to provide response information to the requesting agent.

Table 3-5. Response Signals

Signal Function	Signal Names
Response Status	RS[2:0]#
Response Parity	RSP#
Target Ready (for writes)	TRDY#

Requests initiated in the Request Phase enter the In-order Queue, which is maintained by every agent. The responding agent is responsible for completing the transaction at the top of the In-order Queue. The responding agent is the agent addressed by the transaction.

For write transactions, TRDY# is asserted by the responding agent to indicate that it is ready to accept write or writeback data. For write transactions with an implicit writeback, TRDY# is asserted twice, first for the write data transfer and then for the implicit writeback data transfer.

The RSP# signal provides parity for RS[2:0]#. A parity signal on the system bus is correct if there is an even number of low signals in the set consisting of the covered signals plus the parity signal. Parity is computed using voltage levels, regardless of whether the covered signals are active high or active low.

3.2.6 Data Signals

The data response signals, shown in Table 3-6, control the transfers of data on the bus and provide the data path. Data transfers can be at a 1x or a 2x transfer rate, configured at reset.

Table 3-6. Data Signals

Signal Function	Signal Names
Data Ready	DRDY#
Data Bus Busy	DBSY#
Strobe Bus Busy	SBSY#
Data	D[63:0]#
Data ECC Protection	DEP[7:0]#

Table 3-6. Data Signals (Continued)

Signal Function	Signal Names
Positive phase Data Strobe	STBP[3:0]#
Negative phase Data Strobe	STBN[3:0]#

DRDY# indicates that valid data is on the bus and must be latched. The data bus owner asserts DRDY# for each clock in which valid data is to be transferred. DRDY# can be deasserted to insert wait states in the Data Phase.

DBSY# holds the data bus before the first DRDY# and between DRDY# assertions for a multiple clock data transfer. DBSY# need not be asserted for single clock data transfers.

SBSY# holds the strobe bus before the first DRDY# and between DRDY# assertions for a multiple clock data transfer. SBSY# must be asserted for all data transfers at the 2x transfer rate.

The D[63:0]# signals provide a 64-bit data path between agents. For partial transfers, the byte enable signals BE[7:0]# determine which bytes of the data bus will contain valid data.

The DEP[7:0]# signals provide optional ECC (error correcting code) for D[63:0]#. DEP[7:0]# provides valid ECC for the entire data bus on each clock, regardless of which bytes are enabled.

STBP[3:0]# and STBN[3:0]# (and DRDY#) are used to transfer data at the 2x transfer rate with the source synchronous latched protocol. The agent driving the data transfer drives the strobes with the corresponding data and ECC signals. The agent receiving the data transfer uses the strobes to capture valid data. Each strobe is associated with 16 data bits and 2 ECC signals as shown in Table 3-7.

Table 3-7. STBP[3:0]# and STBN[3:0]# Associations

Strobe Bits	Data Bits	ECC Bits
STBP3#, STBN3#	D[63:48]#	DEP[7:6]#
STBP2#, STBN2#	D[47:32]#	DEP[5:4]#
STBP1#, STBN1#	D[31:16]#	DEP[3:2]#
STBP0#, STBN0#	D[15:0]#	DEP[1:0]#

3.2.7 Defer Signals

The defer signals, shown in Table 3-7, are used by a deferring agent to complete a previously deferred transaction. Any deferrable transaction (DEN# asserted) may use the deferred response signals, provided the requesting agent supports a deferred response (DPS# asserted).

Table 3-8. Defer Signals

Signal Function	Signal Names
ID Strobe	IDS#
Transaction ID	ID[7:0]#

IDS# is asserted to begin the deferred response. ID[7:0]# returns the ID of the deferred transaction that was sent on DID[7:0]#. Please refer to Appendix A, "Signals Reference" for further details.

3.2.8 Error Signals

Table 3-9 lists the error signals on the system bus.

Table 3-9. Error Signals

Signal Function	Signal Names
Bus Initialization	BINIT#
Bus Error	BERR#
Thermal Trip	THERMTRIP#
Thermal Alert	THRMALERT#

BINIT# is used to signal any bus condition that prevents reliable future operation of the bus. BINIT# assertion can be enabled or disabled as part of the power-on configuration register (see Chapter 5, "Configuration and Initialization"). If BINIT# assertion is disabled, BINIT# is never asserted and the error recovery action is taken only by the processor detecting the error.

BINIT# sampling can be enabled or disabled at power-on reset. If BINIT# sampling is disabled, BINIT# is ignored and no action is taken by the processor even if BINIT# is sampled asserted. If BINIT# sampling is enabled and BINIT# is sampled asserted, all processor bus state machines are reset. All agents reset their rotating ID for bus arbitration, and internal state information is lost. Cache contents are not affected. BINIT# sampling and assertion must be enabled for proper processor error recovery.

A machine-check abort is taken for each BINIT# assertion, configurable at power-on.

BERR# is used to signal any error condition caused by a bus transaction that will not impact the reliable operation of the bus protocol (for example, memory data error or non-modified snoop error). A bus error that causes the assertion of BERR# can be detected by the processor or by another bus agent. BERR# assertion can be enabled or disabled at power-on reset. If BERR# assertion is disabled, BERR# is never asserted. If BERR# assertion is enabled, the processor supports two modes of operation, configurable at power-on (refer to section 5.2.6 and 5.2.7 for further details). If BERR# sampling is disabled, BERR# assertion is ignored and no action is taken by the processor. If BERR# sampling is enabled, and BERR# is sampled asserted, the processor core is signaled with the machine check exception.

A machine check exception is taken for each BERR# assertion, configurable at power-on.

THERMTRIP# is the Thermal Trip signal. The Itanium processor protects itself from catastrophic overheating by using an internal thermal sensor. This sensor is set well above the normal operating temperature to ensure that there are no false trips. Data will be lost if the processor goes into thermal trip. This is signaled to the system by the assertion of the THERMTRIP# pin. Once asserted, the signal remains asserted until RESET# is asserted by the platform. There is no hysteresis built into the thermal sensor itself; as long as the die temperature drops below the trip level, a RESET# pulse will reset the processor. If the temperature has not dropped below the trip level, the processor will continue to assert THERMTRIP# and remain stopped.

A thermal alert open-drain signal, indicated to the system by the THRMALERT# pin, brings the ALERT# interrupt output from the thermal sensor located on the Itanium processor to the platform. The signal is asserted when the measured temperature from the processor thermal diode equals or exceeds the temperature threshold data programmed in the high-temp or low-temp registers on the sensor. This signal can be used by the platform to implement thermal regulation features such as generating an external interrupt to tell the operating system that the processor core is heating up.

3.2.9 Execution Control Signals

The execution control signals, shown in Table 3-10, contains signals that change the execution flow of the processor.

Table 3-10. Execution Control Signals

Signal Function	Signal Names
Initialize Processor	INIT#
Tristate outputs during reset configuration	TRISTATE#
Platform Management Interrupt	PMI#
Programmable Local Interrupts	LINT[1:0]

INIT# triggers an unmaskable interrupt to the processor. Semantics required for platform compatibility are supplied in the PAL firmware interrupt service routine. INIT# is usually used to break into Hanging or Idle Processor states.

INIT# has another meaning during reset configuration. If INIT# is sampled asserted on the asserted to deasserted transition of RESET#, then the processor executes its Built-In Self Test (BIST).

If TRISTATE# is sampled asserted on the asserted to deasserted transition of RESET#, then the processor tristates all of its outputs. This function is used during board level testing.

PMI# is the platform management interrupt pin. It triggers the highest priority interrupt to the processor. PMI# is usually used by the system to trigger system events that will be handled by platform specific firmware.

LINT[1:0] are programmable local interrupt pins defined by the interrupt interface.These pins are disabled after RESET#. LINT0 is typically software configured as INT, an 8259-compatible maskable interrupt request signal. LINT1 is typically software configured as NMI, a non-maskable interrupt.

LINT[1:0] are also used, along with the A20M# and IGNNE# signals, to determine the multiplier for the internal clock frequency as described in Chapter 5, “Configuration and Initialization”.

3.2.10 IA-32 Compatibility Signals

The compatibility signals, shown in Table 3-11, contains signals defined for compatibility within the Intel Architecture processor family.

Table 3-11. IA-32 Compatibility Signals

Signal Function	Signal Names
Floating-Point Error	FERR#
Ignore Numeric Error	IGNNE#
Address 20 Mask	A20M#

The Itanium processor asserts FERR# when it detects an unmasked floating-point error. FERR# is included for compatibility and is never asserted in the Itanium-based system environment.

If IGNNE# is asserted, it allows execution of floating-point instructions in the presence of prior unmasked floating-point exceptions. If IGNNE# is deasserted, the processor will wait for interrupts in the presence of unmasked floating-point exceptions. IGNNE# is included for compatibility and is never asserted in the Itanium-based system environment.

If A20M# is asserted, the processor is interrupted. Semantics required for platform compatibility are supplied in the PAL firmware interrupt service routine. A20M# is included for compatibility and is never asserted in the Itanium-based system environment.

A20M# and IGNNE# are also used, along with the LINT[1:0] signals, to determine the multiplier for the internal clock frequency as described in Chapter 5, "Configuration and Initialization".

3.2.11 Platform Signals

The platform signals, shown in Table 3-12, provides signals which support the platform.

Table 3-12. Platform Signals

Signal Function	Signal Names
Processor Present	CPUPRES#
Data Transfer Rate	DRATE#

CPUPRES# can be used to detect the presence of an Itanium processor in a socket. A ground (GND) level indicates that the part is installed while an open indicates no part is installed.

DRATE# configures the system bus data transfer rate. If asserted, the system bus operates at a 1x data transfer rate. If deasserted, the system bus operates at a 2x data transfer rate. DRATE# must be valid at the asserted to deasserted transition of RESET# and must not change its value while RESET# is deasserted.

3.2.12 Diagnostic Signals

The diagnostic signals, shown in Table 3-13, provides signals for probing the processor, monitoring processor performance, and implementing IEEE 1149.1 specification for boundary scan.

Table 3-13. Diagnostic Signals

Signal Function	Signal Names
Breakpoint / Performance Monitor	BPM[5:0]#
Boundary Scan/Test Access	TCK, TDI, TDO, TMS, TRST#

BPM[5:0]# are the Breakpoint and Performance Monitor signals. These signals can be configured as outputs from the processor that indicate the status of breakpoints and programmable counters for monitoring processor events. These signals can be configured as inputs to break program execution.

TCK is used to clock activity on the five-signal Test Access Port (TAP). TDI is used to transfer serial test data into the processor. TDO is used to transfer serial test data out of the processor. TMS is used to control the sequence of TAP controller state changes. TRST# is used to asynchronously initialize the TAP controller.

Data Integrity 4

The Itanium processor system bus incorporates several advanced data integrity features to improve error detection and correction. The system bus includes parity protection for address/request signals, parity or protocol protection on most control signals and Error Correcting Code (ECC) protection for data signals.

The terminology used in this chapter is listed below:

- Machine Check Abort (MCA)

For more information on Machine Check Architecture, see the *Intel® Itanium™ Architecture Software Developer's Manual*, Vol. 2: System Architecture.

4.1 Error Classification

The Itanium processor classifies errors in the following categories, listed with increasing severity. An implementation may always choose to report an error in a more severe category to simplify its logic.

Continuable Error:	The error can be corrected by hardware or firmware.
Local MCA:	The error cannot be corrected, but the memory image is intact. Only one agent is affected and may report the error via machine check handler.
Recoverable Error:	The error cannot be corrected and only a specific memory range is corrupted. Any agent using the data in that range is affected and may report the error via machine check handler.
Global MCA:	The error cannot be corrected and the memory image may be corrupted. All agents are affected and may report the error via machine check handler.
BINIT Global MCA:	The error cannot be corrected. All agents are affected and cannot reliably report the error via an exception handler without a bus reset.

4.2 Itanium™ Processor System Bus Error Detection

The major address and data paths of the Itanium processor system bus are protected by 10 check bits that provide either parity or ECC. Eight ECC bits protect the data bus. Single-bit data errors are automatically corrected. A two-bit parity code protects the address bus.

Three control signal groups are explicitly protected by individual parity bits RP#, RSP#, and IP[1:0]#. Errors on most remaining bus signals can be detected indirectly due to a well-defined bus protocol specification that enables detection of protocol violation errors. Errors on a few bus signals cannot be detected.

An agent is not required to enable all data integrity features since each feature is individually enabled through the power-on configuration register. See Chapter 5, "Configuration and Initialization".

4.2.1 Bus Signals Protected Directly

Most system bus signals are protected either by parity or by ECC. Table 4-1 shows the parity and ECC signals and the signals protected by these parity and ECC signals.

Table 4-1. Direct Bus Signal Protection

Signal(s)	Protect(s)
RP#	ADS#,REQ[4:0]#
AP0#	A[23:3]#
AP1#	A[43:24]#
RSP#	RS[2:0]#
IP0#	IDS#, IDa[7:0]#
IP1#	IDS#, IDb[7:2,0]#
DEP[7:0]#	D[63:0]#

- **Address/Request Bus Signals**: A parity error detected on AP[1:0]# or RP# is reported based on the option defined by the power-on configuration.
- **Address/Request parity disabled**: The agent detecting the parity error ignores it and continues normal operation. This option is normally used in power-on system initialization and system diagnostics.
- **Response Signals**: A parity error detected on RSP# should be reported by the agent detecting the error as a protocol error.
- **Deferred Signals**: A parity error detected on IP[1:0]# should be reported by the agent detecting the error as a protocol error.
- **Data Transfer Signals**: The Itanium processor system bus can be configured with either no data bus error checking or ECC. If ECC is selected, single-bit errors can be corrected and double-bit errors and poisoned data can be detected. Corrected single-bit ECC errors are continuable errors. Double-bit errors and poisoned data may be recoverable or non-recoverable. The errors on read data being returned are treated by the requestor as local errors. The errors on write or writeback data are treated by the target as recoverable errors.

4.2.2 Unprotected Bus Signals

The following Itanium processor system bus signals are not protected by ECC or parity:

- The execution control signals BCLK, RESET#, and INIT# are not protected.
- The IA-32 compatibility signals FERR#, IGNNE#, and A20M# are not protected.
- The system support signal PMI# is not protected.

4.2.3 Hard-fail Response

The target can assert a hard-fail response to a transaction that has generated an error. The central agent can also claim responsibility for a transaction after a response time-out expiration and terminate the transaction with a hard-fail response.

On observing a hard-fail response, the initiator may treat it as a local or a global machine check.

4.2.4 Itanium™ Processor System Bus Error Code Algorithms

4.2.4.1 Parity Algorithm

All bus parity signals use the same algorithm to compute correct parity. A correct parity signal is high if all covered signals are high or if an even number of covered signals are low. A correct parity signal is low if an odd number of covered signals are low. Parity is computed using voltage levels, regardless of whether the covered signals are active-high or active-low. Depending on the number of covered signals, a parity signal can be viewed as providing "even" or "odd" parity; this specification does not use either term.

4.2.4.2 Itanium™ Processor System Bus ECC Algorithm

The Itanium processor system bus uses an ECC code that can correct single-bit errors, detect double-bit errors, send poisoned data, and detect all errors confined to one nibble. System designers may choose to detect all these errors or a subset of these errors. They may also choose to use the same ECC code in additional system level caches, main memory arrays, or I/O subsystem buffers. For more information, see the *Intel® Itanium™ Architecture Software Developer's Manual*.

Configuration and Initialization 5

This chapter describes configuration options and initialization details for the Itanium processor.

A system may contain single or multiple Itanium processors with one to four processors on a single system bus. All processors are connected to one system bus unless the description specifically states otherwise.

5.1 Configuration Overview

Itanium processors have some configuration options that are determined by hardware, and some that are determined by software.

Itanium processors sample their hardware configuration on the asserted-to-deasserted transition of RESET#. The sampled information configures the processor and other bus agents for subsequent operation. These configuration options cannot be changed except by another reset. All resets reconfigure the bus agents. Refer to the *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet* for further details.

The Itanium processor can also be configured with software configuration options. These options can be changed by writing to a power-on configuration register that all bus agents must implement. These options should be changed only after taking into account synchronization between multiple Itanium processor system bus agents.

5.2 Configuration Features

Table 5-1 lists the features provided by the power-on configuration register and the signals that are reserved during reset configuration. The table also includes the PAL calls corresponding to the features that are under user control during power-on configuration. For details on the PAL calls, refer to the *Intel® Itanium™ Architecture Software Developer's Manual, Volume 2*. All bus agents are required to maintain some software read/writable bits to control some of the features listed in the table.

Table 5-1. Processor Power-on Configuration Features

Feature	Select Signals	PAL Call	Read/Write	Default/Inactive Value
Data Error Checking Enabled	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
Response/ID Error checking enabled	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
Address/Request Error checking enabled	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
BERR# Assertion Enabled	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
BERR# Sampling Enabled	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
BERR# Assertion Enabled for Initiator Internal Errors	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
BINIT# Assertion Enabled	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled
Output Tristate Enabled	TRISTATE#	N/A	Read	Disabled
Execute BIST	INIT#	N/A	Read	Disabled
BINIT# Sampling Enabled	A10#	PAL_BUS_SET_FEATURES	Read/Write	Disabled
In-order Queue Depth of 1	A7#	PAL_BUS_GET_FEATURES	Read	Disabled i.e. IOQ depth is 8
Reserved[1]	A31#, A30#, A14#, A13#, A12#, A11#, A8#, A6#, A5#	N/A	N/A	N/A
Unused Address bits during reset configuration[2]	A43#–A32#, A29#–A16#, A9#, A4#, A3#	N/A	N/A	N/A
Symmetric Arbitration ID	BR3#–BR0#	PAL_FIXED_ADDR	Read	Based on system bus mapping w.r.t. BREQ0#
Clock Ratios	A20M#, IGNNE#, LINT[1:0]	PAL_FREQ_RATIOS	Read	2/8
Request Bus Parking Enabled	A15#	PAL_BUS_SET_FEATURES	Read/Write	Disabled
Data Transfer Rate	DRATE#	PAL_BUS_GET_FEATURES	Read	Disabled i.e. Source Synchronous data transfer rate is selected
LOCK# Assertion Faulted	N/A	DCR.lc = 1[3]	Read/Write	Disabled (dcr.lc=0) i.e. No fault on bus lock operations
LOCK# Assertion Masked	N/A	PAL_BUS_SET_FEATURES	Read/Write	Disabled (default from PAL) i.e. LOCK# assertion enabled

1. The reserved bits must either be un-driven or driven inactive during reset configuration.
2. These bits are not used during reset configuration, and are don't care.
3. This bit is used for platforms which do not support the LOCK# pin and have to rely on the DCR.lc bit to cause a fault. When the locked data reference fault handler is not installed, this bit is set to prevent the LOCK# pin from being asserted.

5.2.1 Output Tristate

The Itanium processor tristates all of its tristatable outputs if the TRISTATE# signal is sampled asserted on the asserted-to-deasserted transition of the RESET# signal. The only way to exit out of the Output Tristate mode is by assertion of RESET# with TRISTATE# deasserted. Refer to the *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet* for further details.

5.2.2 Built-in Self Test (BIST)

The Itanium processor executes its Built-In Self Test (BIST) if the INIT# signal is sampled asserted on the asserted-to-deasserted transition of the RESET# signal. In a multi-processor cluster-based system architecture, the INIT# pin of different processors may or may not be shared. No software control is available to perform this function.

5.2.3 Data Bus Error Checking

The Itanium processor data bus error checking can be enabled or disabled. After RESET# is asserted, data bus error checking is always disabled. Data bus error checking can be enabled under software control. For more information on this feature, please refer to the *Intel® Itanium™ Architecture Software Developer's Manual*.

5.2.4 Response/ID Signal Parity Error Checking

The Itanium processor system bus supports parity protection for the response signals RS[2:0]# and the transaction ID signals ID[7:0]#. The parity checking on these signals can be enabled or disabled. After RESET# is asserted, response signal parity checking is disabled. It can be enabled under software control.

5.2.5 Address/Request Signal Parity Error Checking

The Itanium processor address bus supports parity protection on the Request signals, A[43:3]#, ADS#, and REQ[4:0]#. The parity checking on these signals can be enabled or disabled. After RESET# is asserted, request signal parity checking is disabled. It can be enabled under software control.

5.2.6 BERR# Assertion for Initiator Bus Errors

An Itanium processor system bus agent can be enabled to assert the BERR# signal if it detects a bus error. After RESET# is asserted, BERR# signal assertion is disabled for detected errors. It may be enabled under software control.

5.2.7 BERR# Assertion for Target Bus Errors

An Itanium processor system bus agent can be enabled to assert the BERR# signal if the addressed (target) bus agent detects an error. After RESET# is asserted, BERR# signal assertion is disabled on target bus errors. It may be enabled under software control.

5.2.8 BERR# Sampling

If the BERR# sampling policy is enabled, the BERR# input receiver causes a global Machine Check Abort (MCA). It may be enabled under software control.

5.2.9 BINIT# Error Assertion

When the bus protocol is violated, an Itanium processor system bus agent can be enabled to assert the BINIT# signal. After RESET# is asserted, BINIT# signal assertion is disabled. It may be enabled under software control.

5.2.10 BINIT# Error Sampling

The BINIT# input receiver is enabled for bus initialization control if A10# was sampled asserted on the asserted-to-deasserted transition of RESET#.

5.2.11 LOCK# Assertion

For requests which require bus locked sequences, Itanium processors can be configured to lock the system bus, fault, or complete the request non-atomically. It may be enabled under software control.

5.2.12 In-order Queue Pipelining

Itanium processor system bus agents are configured to an In-order Queue depth of one if A7# is sampled asserted on the asserted to deasserted transition of RESET#. If A7# is sampled deasserted on the asserted to deasserted transition of RESET#, the processors default to an In-Order Queue depth of eight. This function cannot be controlled by software.

5.2.13 Request Bus Parking Enabled

Itanium processor system bus agents can be configured to park on the request bus when idle. The last processor to own the request bus will park on an idle request bus if A15# is sampled asserted on the asserted-to-deasserted transition of RESET#. No processor will park on the request bus if A15# is sampled deasserted on the asserted-to-deasserted transition of RESET#.

5.2.14 Data Transfer Rate

The transfer rate of an Itanium processor system bus agent is configured by sampling DRATE# at the asserted-to-deasserted transition of RESET#. If DRATE# is sampled asserted, the data bus will be configured at the 1x data transfer rate. If DRATE# is sampled deasserted, the data bus will be configured at the 2x data transfer rate.

5.2.15 Symmetric Agent Arbitration ID

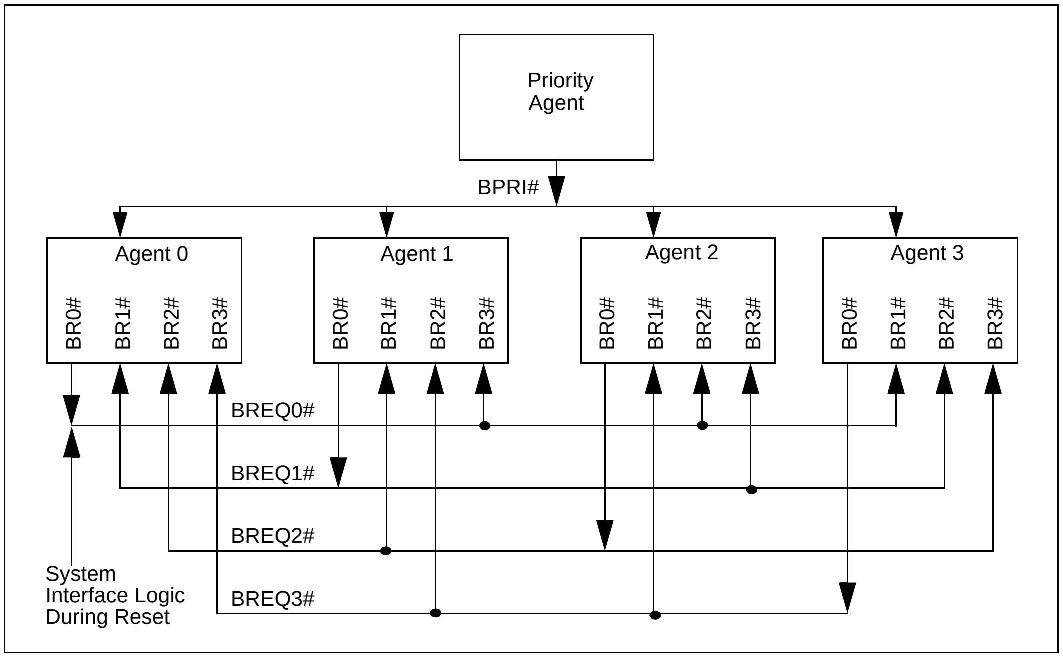
The Itanium processor system bus supports symmetric distributed arbitration among one to four bus agents. Each processor identifies its initial position in the arbitration priority queue based on an agent ID supplied at configuration. The agent ID can be 0, 1, 2, or 3. Each logical processor on a particular Itanium processor system bus must have a distinct agent ID.

BREQ[3:0]# bus signals are connected to the four symmetric agents in a rotating manner as shown in Table 5-2 and in Figure 5-1. Every symmetric agent has one I/O pin (BR0#) and three input only pins (BR1#, BR2#, and BR3#).

Table 5-2. Itanium™ Processor Bus BREQ[3:0]# Interconnect (4-Way Processors)

Bus Signal	Agent 0 Pins	Agent 1 Pins	Agent 2 Pins	Agent 3 Pins
BREQ0#	BR0#	BR3#	BR2#	BR1#
BREQ1#	BR1#	BR0#	BR3#	BR2#
BREQ2#	BR2#	BR1#	BR0#	BR3#
BREQ3#	BR3#	BR2#	BR1#	BR0#

Figure 5-1. BR[3:0]# Physical Interconnection with Four Symmetric Agents



At the asserted-to-deasserted transition of RESET#, system interface logic is responsible for asserting the BREQ0# bus signal. The BREQ[3:1]# bus signals remain deasserted. All processors sample their BR[3:1]# pins on the asserted-to-deasserted transition of RESET# and determine their agent ID from the sampled value.

Each physical processor is a logical processor with a distinct agent ID.

Table 5-3. Arbitration ID Configuration

BR0#	BR1#	BR2#	BR3#	Arbitration ID
L[1]	H	H	H	0
H	H	H	L	1
H	H	L	H	2
H	L	H	H	3

1. L and H designate electrical levels.

5.2.16 Clock Frequency Ratios

The following system bus ratio configurations are defined for the Itanium processor.

Table 5-4. Itanium™ Processor System Bus to Core Frequency Multiplier Configuration

Ratio of Bus Frequency to Core Frequency	LINT[1]	LINT[0]	IGNNE#	A20M#
2/11	0(L)	0(L)	0(H)	0(H)
2/12	0(L)	1(H)	1(L)	1(L)

5.3 Initialization Overview

The Itanium processor initializes a minimum set of internal states upon RESET#. The processor and PAL firmware initialize and test the processor on reset.

5.3.1 Initialization with RESET#

The Itanium processor begins initialization upon detection of RESET# signal active. RESET# signal assertion is not maskable and ignores all instruction boundaries including both IA-32 and Itanium instructions.

Table 5-5 shows the architectural state initialized by the processor hardware and PAL firmware at reset. All other architectural states are undefined at hardware reset. Refer to the *Intel® Itanium™ Architecture Software Developer's Manual* for a detailed description of the registers.

Table 5-5. Itanium™ Processor Reset State (after PAL)

Processor Resource	Symbol	Value	Description
Instruction Pointer	IP	Refer to the *Itanium™ Architecture Software Developer's Manual* for details	SALE_RESET entry point for the Itanium™ processor.
Register Stack Configuration Register	RSC	mode=0	Enforced lazy mode.
Current Frame Marker	CFM	sof=96, sol=0, sor=0, rrbs=0	All physical general purpose registers are available, register state is undefined, no locals in the general register frame, no rotation in the general register frame, rename base for FR, GR and PR registers is set to 0.
Translation Register	TR	Invalid	All TLBs are cleared.
Translation Cache	TC	Invalid	All TLBs are cleared.
Caches	—	Invalid	All caches are disabled.

5.3.2 Initialization with INIT#

The Itanium processor supports an INIT interrupt, also referred to as “warm boot” or “soft boot”. INIT can be initiated by either asserting the INIT# signal or an INIT interrupt message. INIT cannot be masked except when a Machine Check (MC) is in progress. In this case, the INIT interrupt is held pending. INIT is recognized at instruction boundaries. An INIT interrupt does not disturb any processor architectural states, the state of the caches, model specific registers, or any integer or floating-point states.

Table 5-6 shows the processor state modified by INIT. Refer to the *Intel® Itanium™ Architecture Software Developer’s Manual* for a detailed description of the registers.

Table 5-6. Itanium™ Processor INIT State

Processor Resource	Symbol	Value	Description
Instruction Pointer	IP	Refer to the *Itanium™ Architecture Software Developer’s Manual* for details	PALE_INIT entry point for the Itanium™ processor.
Interruption Instruction Bundle Pointer	IIP	Original value of IP	Value of IP at the time of INIT.
Interruption Processor Status Register	IPSR	Original value of PSR	Value of PSR at the time of INIT.
Interruption Function State	IFS	v=0	Invalidate IFS.

Test Access Port (TAP) 6

This chapter describes the implementation of the Itanium processor Test Access Port (TAP) logic. The TAP complies with the IEEE 1149.1 (JTAG) Specification. Basic functionality of the 1149.1-compatible test logic is described here. For details of the IEEE 1149.1 Specification, the reader is referred to the published standard[1], and to the many books currently available on the subject.

A simplified block diagram of the TAP is shown in Figure 6-1. The TAP logic consists of a finite state machine controller, a serially-accessible instruction register, instruction decode logic and data registers. The set of data registers includes those described in the 1149.1 standard (the bypass register, device ID register, BIST result register, and boundary scan register).

For specific boundary scan chain information, please reference the Itanium processor BSDL file available at developer.intel.com.

6.1 Interface

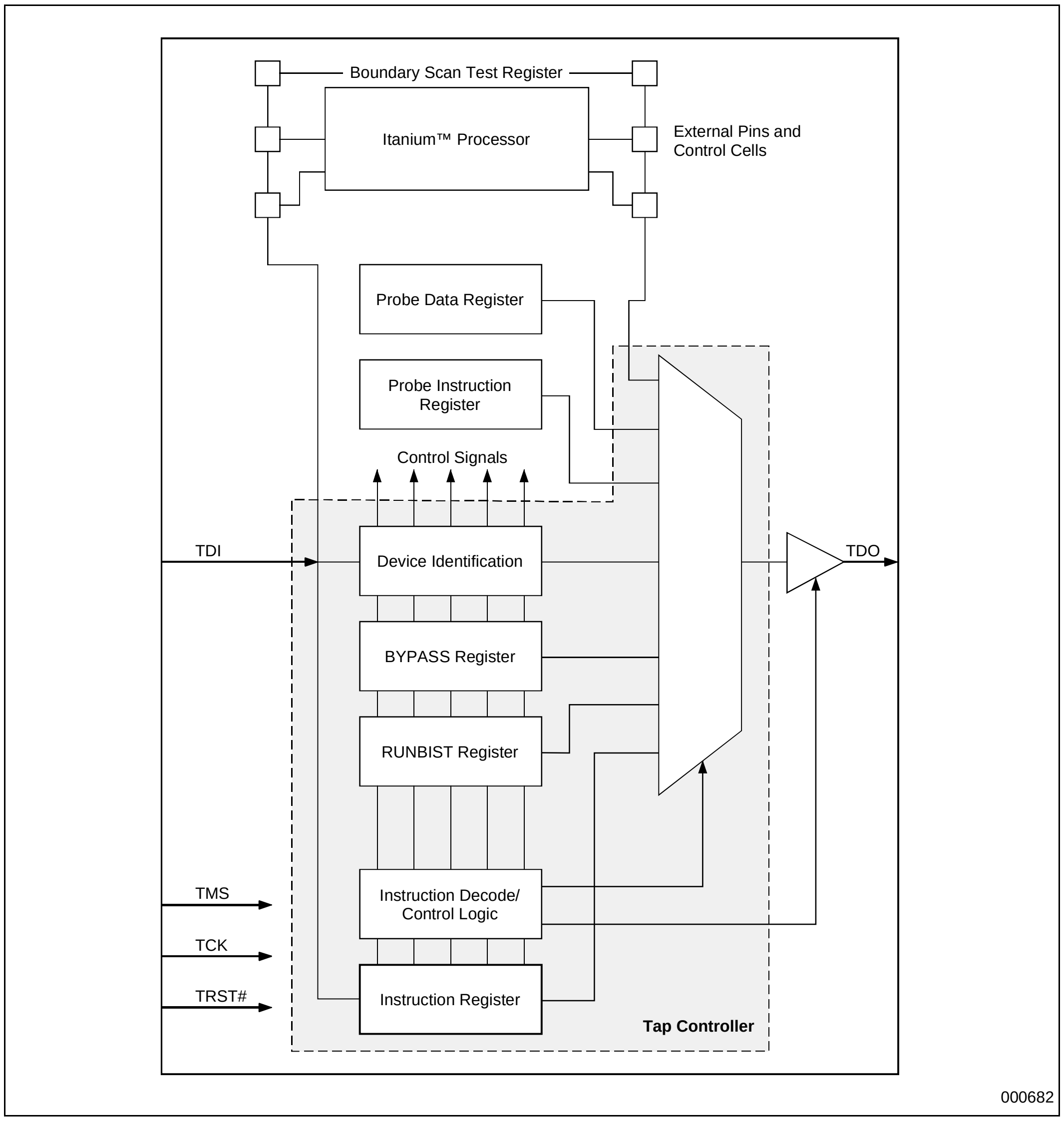
The TAP logic is accessed serially through 5 dedicated pins on the processor package:

- **TCK:** The TAP clock signal
- **TMS:** “Test mode select,” which controls the TAP finite state machine
- **TDI:** “Test data input,” which inputs test instructions and data serially
- **TRST#:** “Test reset,” for TAP logic reset
- **TDO:** “Test data output,” through which test output is read serially

TMS, TDI and TDO operate synchronously with TCK (which is independent of any other processor clock). TRST# is an asynchronous input signal.

1. ANSI/IEEE Std. 1149.1-1990 (including IEEE Std. 1149.1a-1993), “IEEE Standard Test Access Port and Boundary Scan Architecture,” IEEE Press, Piscataway NJ, 1993.

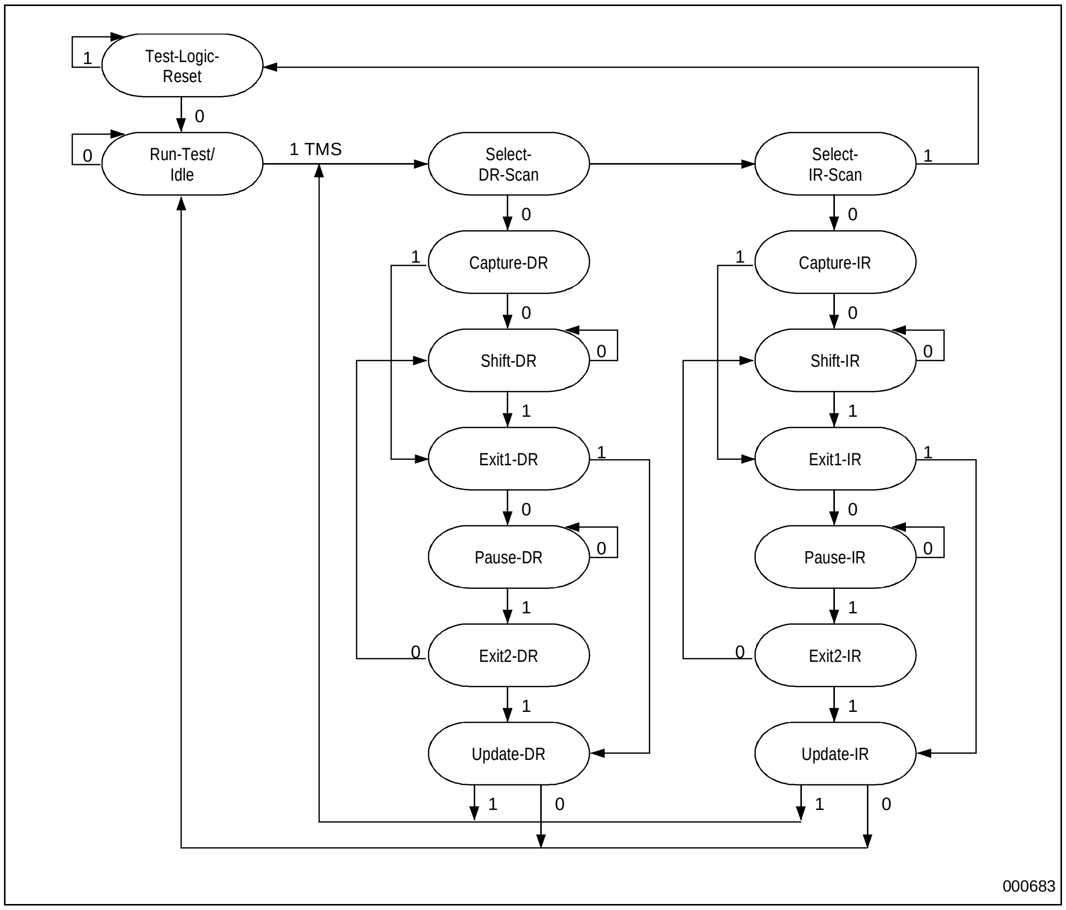
Figure 6-1. Test Access Port Block Diagram



6.2 Accessing The TAP Logic

The TAP is accessed through an IEEE 1149.1-compliant TAP controller finite state machine. This finite state machine, shown in Figure 6-2, contains a reset state, a run-test/idle state, and two major branches. These branches allow access either to the TAP Instruction Register or to one of the data registers. The TMS pin is used as the controlling input to traverse this finite state machine. TAP instructions and test data are loaded serially (in the Shift-IR and Shift-DR states, respectively) using the TDI pin. State transitions are made on the rising edge of TCK.

Figure 6-2. TAP Controller State Diagram



The following is a brief description of each of the states of the TAP controller state machine. Refer to the IEEE 1149.1 standard for detailed descriptions of the states and their operation.

- **Test-Logic-Reset:** In this state, the test logic is disabled so that the processor operates normally. In this state, the instruction in the Instruction Register is forced to IDCODE. Regardless of the original state of the TAP Finite State Machine (TAPFSM), it always enters Test-Logic-Reset when the TMS input is held asserted for at least five clocks. The controller also enters this state immediately when the TRST# pin is asserted, and automatically upon power-up. The TAPFSM cannot leave this state as long as the TRST# pin is held asserted.
- **Run-Test/Idle:** A controller state between scan operations. Once entered the controller will remain in this state as long as TMS is held low. In this state, activity in selected test logic occurs only in the presence of certain instructions. For instructions that do not cause functions to execute in this state, all test data registers selected by the current instructions retain their previous state.
- **Select-IR-Scan:** This is a temporary controller state in which all test data registers selected by the current instruction retain their previous state.
- **Capture-IR:** In this state, the shift register contained in the Instruction Register loads a fixed value (of which the two least significant bits are “01”) on the rising edge of TCK. The parallel, latched output of the Instruction Register (current instruction) does not change in this state.

- **Shift-IR:** The shift register contained in the Instruction Register is connected between TDI and TDO and is shifted one stage toward its serial output on each rising edge of TCK. The output arrives at TDO on the falling edge of TCK. The current instruction does not change in this state.
- **Exit-IR:** This is a temporary state and the current instruction does not change in this state.
- **Pause-IR:** Allows shifting of the Instruction Register to be temporarily halted. The current instruction does not change in this state.
- **Exit2-IR:** This is a temporary state and the current instruction does not change in this state.
- **Update-IR:** The instruction which has been shifted into the Instruction Register is latched into the parallel output of the Instruction Register on the falling edge of TCK. Once the new instruction has been latched, it remains the current instruction until the next Update-IR (or until the TAPFSM is reset).
- **Select-DR-Scan:** This is a temporary controller state and all test data registers selected by the current instruction retain their previous values.
- **Capture-DR:** In this state, data may be parallel-loaded into test data registers selected by the current instruction on the rising edge of TCK. If a test data register selected by the current instruction does not have a parallel input, or if capturing is not required for the selected test, then the register retains its previous state.
- **Shift-DR:** The data register connected between TDI and TDO as a result of selection by the current instruction is shifted one stage toward its serial output on each rising edge of TCK. The output arrives at TDO on the falling edge of TCK. If the data register has a latched parallel output then the latch value does not change while new data is being shifted in.
- **Exit1-DR:** This is a temporary state and all data registers selected by the current instruction retain their previous values.
- **Pause-DR:** Allows shifting of the selected data register to be temporarily halted without stopping TCK. All registers selected by the current instruction retain their previous values.
- **Exit2-DR:** This is a temporary state and all registers selected by the current instruction retain their previous values.
- **Update-DR:** Some test data registers may be provided with latched parallel outputs to prevent changes in the parallel output while data is being shifted in the associated shift register path in response to certain instructions. Data is latched into the parallel output of these registers from the shift-register path on the falling edge of TCK.

6.3 TAP Registers

The following is a list of all test registers which can be accessed through the TAP.

Boundary Scan Register
The Boundary Scan register consists of several single-bit shift registers. The boundary scan register provides a shift register path from all the input to the output pins on the Itanium processor. Data is transferred from TDI to TDO through the boundary scan register.

Bypass Register
The bypass register is a one-bit shift register that provides the minimal path length between TDI and TDO. The bypass register is selected when no test operation is being performed by a component on the board. The bypass register loads a logic zero at the start of a scan cycle.

Device Identification (ID) Register
The device ID register contains the manufacturer's identification code, version number, and part number. The device ID register has a fixed length of 32 bits, as defined by the IEEE 1149.1 specification.

RUNBIST Register
The runbist register is a one-bit register that is loaded with logic zero when BIST is successfully completed. The register reports the result of the Itanium processor BIST.

Instruction Register
The instruction register contains a four-bit command field to indicate one of the following instructions: BYPASS, EXTEST, SAMPLE/PRELOAD, IDCODE, RUNBIST, HIGHZ, and CLAMP. The most significant bit of the Instruction register is connected to TDI and the least significant bit is connected to TDO.

6.4 TAP Instructions

Table 6-1 shows the IEEE 1149.1 Standard defined instructions for the TAP controller. Please note that except for BYPASS, which is all 1s, all public instructions as defined by the IEEE 1149.1 must have an instruction code of 0000 xxxx.

- **BYPASS:** The bypass register contains a single shift-register stage and is used to provide a minimum length serial path between the TDI and TDO pins. This bypass enables the rapid movement of test data to and from other components on a system board.
- **EXTEST:** This instruction allows data to be serially loaded into the boundary scan chain through TDI, and forces the output buffers to drive the data contained in the boundary scan register. This instruction can be used in conjunction with SAMPLE/PRELOAD to test the board-level interconnect between components.
- **SAMPLE/PRELOAD:** This instruction allows data to be sampled from the input buffers to be captured in the boundary scan register and serially unloaded from the TDO pin. This instruction also allows data to be pre-loaded into the boundary scan chain prior to selecting another boundary scan instruction. This instruction can be used in conjunction with the EXTEST instruction to test the board-level interconnect between components.
- **IDCODE:** This instruction places the device ID register between TDI and TDO to allow the device identification value to be shifted out to TDO. The register contains the manufacturer's identity, part number, and version number. This instruction is the default instruction after the TAP has been reset.
- **RUNBIST:** This instruction invokes BIST and places the BIST result register between TDI and TDO. The BIST result can then be examined. BIST will continue executing even if the FSM leaves the Run-Test/Idle state.
- **HIGHZ:** This instruction places all of the output buffers of the component in an inactive drive state. In this state, board-level testing can be performed without incurring the risk of damage to the component. During the execution of the HIGHZ instruction, the bypass register is placed between TDI and TDO.

- **CLAMP:** This instruction selects the bypass register while the output buffers drive the data contained in the boundary scan chain. This instruction protects the receivers from the values in the boundary scan chain while data is being shifted out.

Table 6-1. Instructions for the Itanium™ Processor TAP Controller

Format	Encoding (Binary)	Encoding (Hex)
Public Instructions (IEEE 1149.1 Standard Mandatory)		
BYPASS	1111 1111	FFh
EXTEST	0000 0000	00h
SAMPLE/PRELOAD	0000 0001	01h
Public Instructions (Not Mandatory)		
IDCODE	0000 0010	02h
RUNBIST	0000 0111	07h
HIGHZ	0000 1000	08h
CLAMP	0000 1011	0Bh

6.5 Reset Behavior

The TAP and its related hardware are reset by transitioning the TAP controller finite state machine into the Test-Logic-Reset state. The TAP is completely disabled upon reset (i.e., by resetting the TAP, the processor will function as though the TAP did not exist). Note that there is no logic in the TAP which responds to the normal processor reset signal. The TAP can be transitioned to the Test-Logic-Reset state by any one of the following three ways:

- Power on the processor. This automatically (asynchronously) resets the TAP controller.
- Assert the TRST# pin at any time. This asynchronously resets the TAP controller.
- Hold the TMS pin high for 5 consecutive cycles of TCK. This transitions the TAP controller to the Test-Logic-Reset state.

6.6 Scan Chain Order

Figure 6-3 and Figure 6-4 illustrate the order of the scan chain of the Itanium processor cartridge of varying L3 cache sizes.

Figure 6-3. Intel® Itanium™ Processor 2MB Cartridge Scan Chain Order

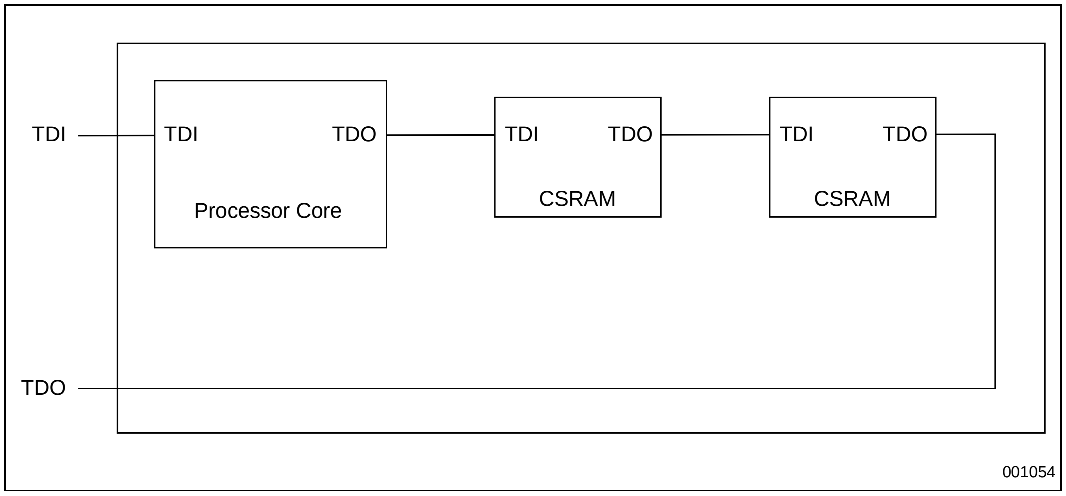
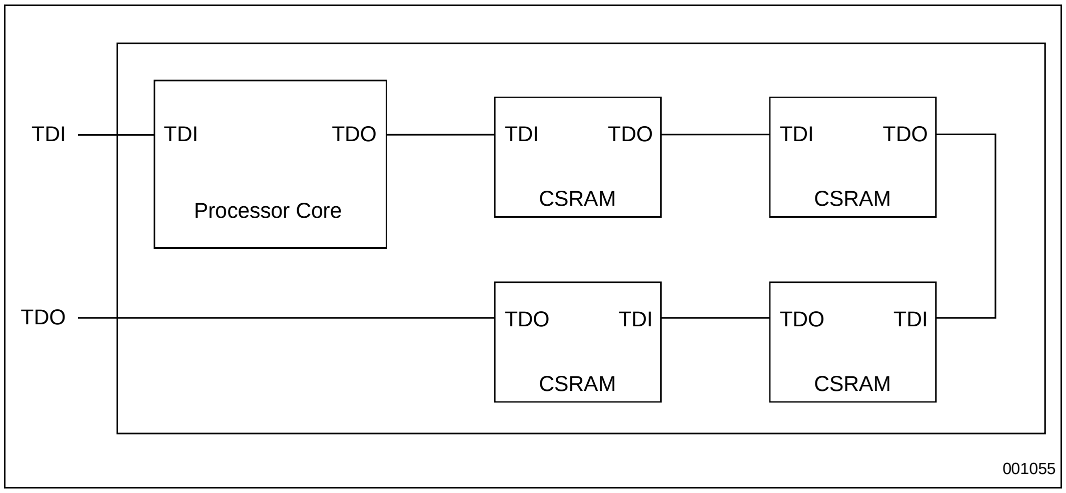


Figure 6-4. Intel® Itanium™ Processor 4MB Cartridge Scan Chain Order



Integration Tools 7

The Itanium processor supports an In-target Probe (ITP) for program execution control, register/memory/IO access, and breakpoint control. This tool provides functionality commonly associated with debuggers and emulators. Use of an ITP will not affect high speed operations of the processor signals thereby allowing the system to maintain full operating speed.

The Itanium processor also supports a Logic Analyzer Interface (LAI) module to connect a logic analyzer to signals on the board. Third party logic analyzer vendors offer a variety of products with bus monitoring capability.

This chapter describes the ITP and the LAI, as well as a number of technical issues that must be taken into account when including these tools in a debug strategy. Please note that simulation of your design is required to ensure that signal integrity issues are avoided.

7.1 In-target Probe (ITP) for the Itanium™ Processor

The Itanium processor debug port (DP) is the command and control interface for the ITP debugger. The ITP enables run-time control of the Itanium processors for system debug. Support of the ITP is a critical requirement for effective debug of the processor board.

The debug port, which is connected to the system bus, is a combination of the system, TAP, and execution signals. There are certain electrical, functional, and mechanical constraints on the debug port which must be followed. The electrical constraint requires the debug port to operate at the speed of the Itanium processor system bus and use the TAP signals at high speed. The functional constraint requires the debug port to use the TAP system via a handshake and multiplexing scheme. The mechanical constraint requires the ITP associated hardware to fit within a specified volume (see Section 7.1.2).

7.1.1 Primary Function

The primary function of an ITP is to provide a control and query interface for multiple processors. With an ITP, one can control program execution and have the ability to access processor registers, system memory and I/O. Thus, one can start and stop program execution using a variety of breakpoints, single-step the program at the assembly code level, as well as read and write registers, memory and I/O. The processors are controlled from an application running on an Intel processor-based PC with a PCI card slot.

7.1.2 Mechanical Requirements

The debug port cable's egress imposes mechanical requirements on the debug port that must be followed. Figure 7-1 and Figure 7-2 show the mechanical volume occupied by the ITP buffer board hardware when connected to the DP connector. All dimensions for the diagrams that follow are in units of inches.

Figure 7-1. Front View of the Mechanical Volume Occupied by ITP Hardware

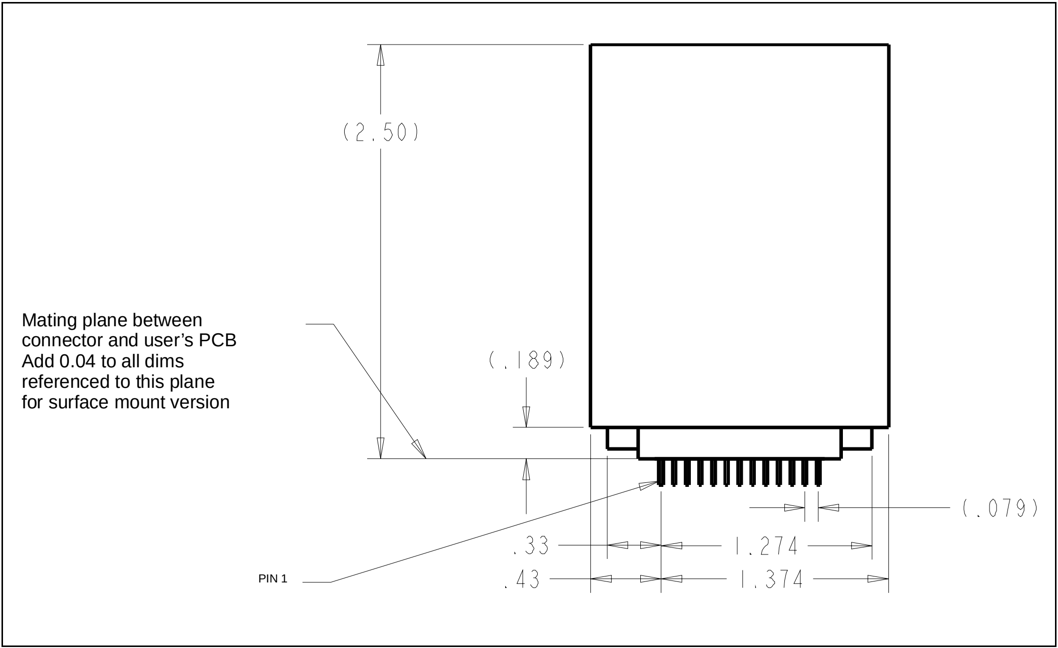
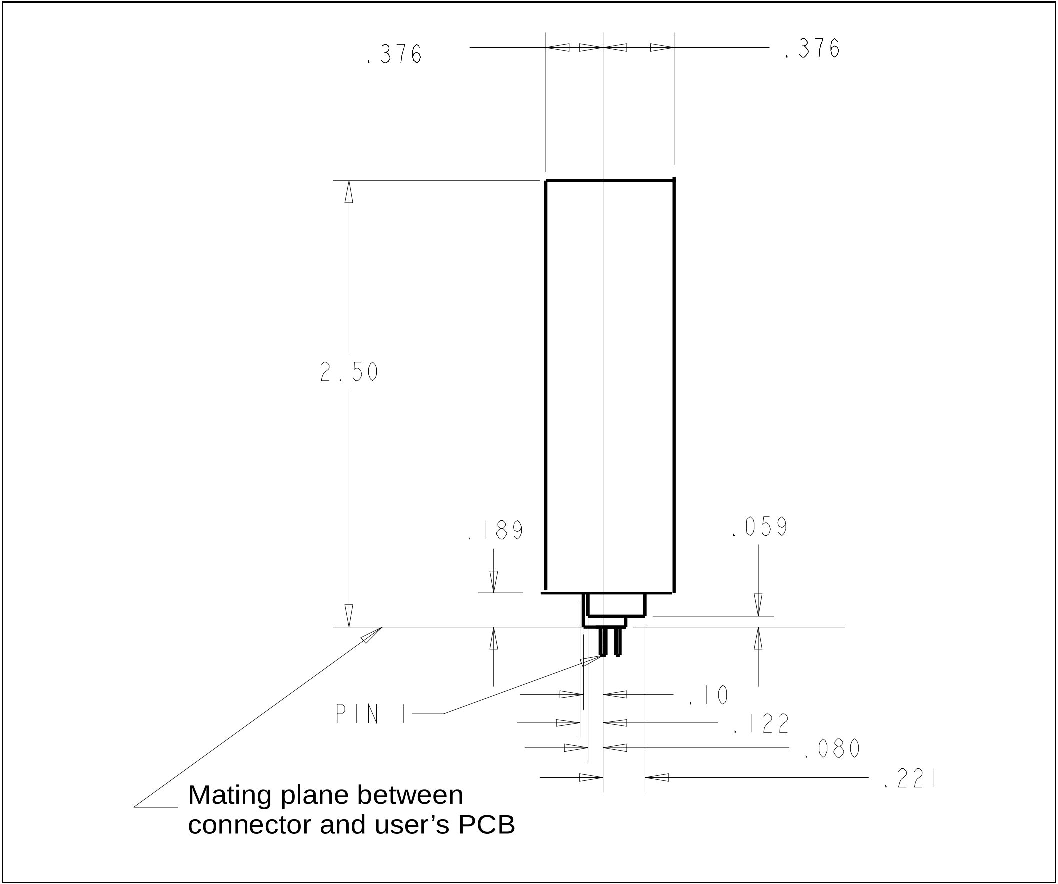


Figure 7-2. Side View of the Mechanical Volume Occupied by ITP Hardware



7.1.3 Connector and Signals

The “standard” JTAG IEEE 1149.1 Specification signal levels are not followed for the Itanium processor debug port. Additionally, the Itanium processor is capable of a much higher speed TAP clock, TCK, than a normal 1149.1 Specification environment. It is highly recommended that system designers conduct platform simulations to avoid signal integrity issues.

To remove any possible confusion over the connectivity of the ITP DP the connector pinout is shown below in Figure 7-3 along with a connectivity table, Table 7-1. The recommended DP connector is manufactured by Berg* under part number 61641-303, a 25 pin through-hole mount header (pin 26 removed for keying). Also available from Berg is a surface mount version of this connector under part number 61698-302TR. The through-hole mount version is recommended for durability reasons. Full specification of the connectors, including PCB layout guidelines, are available from Berg Electronics.

Figure 7-3. Debug Port Connector Pinout Bottom View

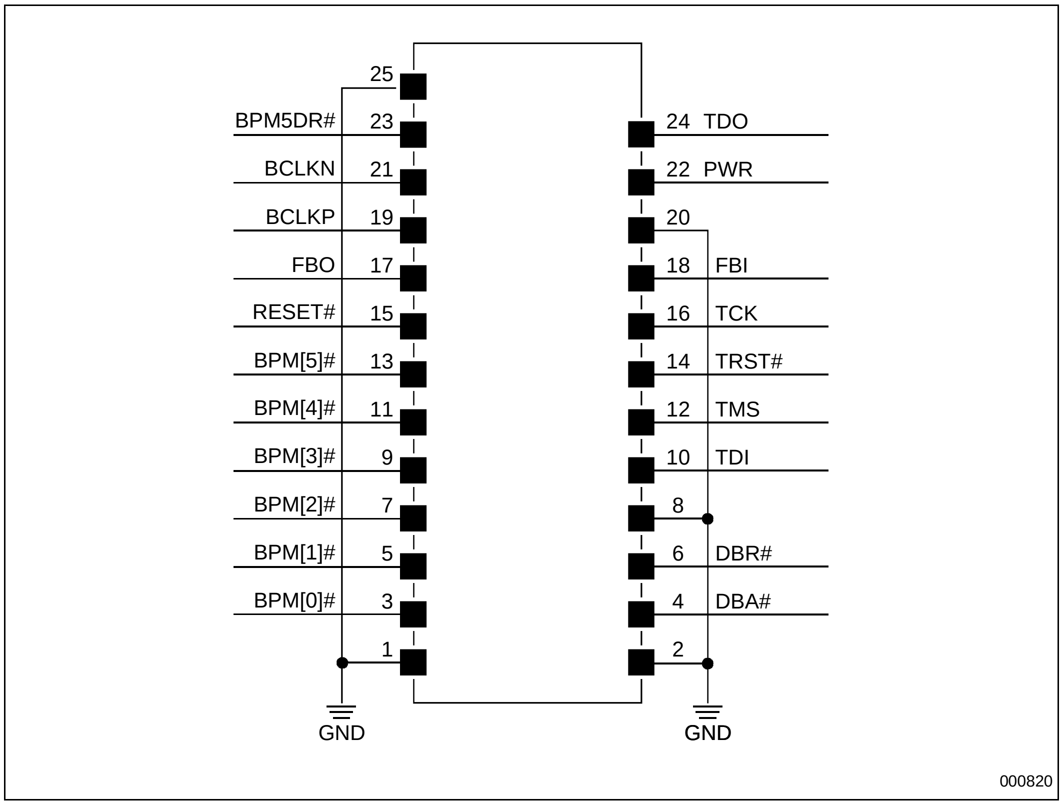


Table 7-1. Recommended Debug Port Signal Connectivity

Pin	Pin Connection
BCLK(P,N)	Differential bus clock driven by the main system bus clock driver. Signals require termination.
BPM[5:0]#	Used by DP to force break at reset and to monitor events. Connected to all the Itanium™ processors and the chipset. Signals require termination.
BPM5DR#	Signals assertion of pin BPM5# and is connected to that signal for on-board isolation.
DBA#	Driven from the DP to indicate active TAP access of system or active event monitoring while running; if not used can be left as a no connect.

Table 7-1. Recommended Debug Port Signal Connectivity (Continued)

Pin	Pin Connection
DBR#	This signals the target to initiate a reset. It is an open drain and should be pulled up on the target. Connect to the system reset (not the same as RESET#).
FBI	If using an additional clock buffer to drive TCK, use FBI as the clock source; otherwise it is a no connect.
FBO	This pin should be connected to the TCK signal of the Itanium Processor. Depending on which topology is used, the specific connectivity will differ.
GND	Ground.
PWR	Connect to 1.5KΩ pull-up to $V_{TT.}$
RESET#	Input from target system to DP port indicating the system is completely inactive. This can be connected to the main reset line shared between the processors and chipset.
TCK	Connect to all devices in scan chain and debug port connector, unless using a TCK clock buffer. If using a TCK clock buffer, termination resistor is not required and should be no connect.
TDI	TAP data in. Connected to the TDI of the first device in scan chain.
TDO	TAP data out. Connected to TDO of the last device in scan chain.
TMS	TAP state management signal. Connected all devices in scan chain and debug port connector. May be individually buffered from debug port connector.
TRST#	TAP reset signal. Connected to all devices in scan chain and debug port connector. May be individually buffered from debug port connector.

In the past, standard practice for cost reduction of system platforms is to first remove the debug header to improve board throughput, and then eventually remove the signal traces entirely to save board space. This was an acceptable solution previously since an interposer style debug connector could be used on the processor to gain access to the TAP chain. This is no longer acceptable since, in a multiprocessor system using the Itanium processor with its tight timing margins, there is no capability to use an interposer debug connector. This necessitates, at an absolute minimum, leaving the hardware (without the header) for the debug port on the system board. Without this, the ability to debug a multiprocessor system through a TAP compliant interface is very limited.

7.2 Logic Analyzer Interface (LAI) for the Itanium™ Processor

A Logic Analyzer Interface (LAI) module provides a way to connect a logic analyzer to signals on the board. Third party logic analyzer vendors offer a variety of products with bus monitoring capability. Consult the vendor for specific details and product offerings.

The Itanium processor system bus can be monitored with logic analyzer equipment. Due to the complexity of Itanium multiprocessor systems, the LAI is critical in providing the ability to probe and capture system bus signals using a logic analyzer for use in system debug and validation. Therefore, the guidelines for the LAI keepout volume must be strictly adhered to in order for a logic analyzer to interface with the target system.

An LAI probe adapter is installed between the socket and the Itanium processor cartridge. The LAI adapter pins plug into the socket, while the Itanium processor cartridge pins plug into a socket on the LAI adapter. Cabling that is part of the LAI probe adapter egresses the system to allow an electrical connection between the Itanium processor and a logic analyzer or debug tool. The maximum volume occupied by the LAI adapter, known as the keep-out volume, as well as the cable egress restrictions, are illustrated in Figure 7-4, Figure 7-5, and Figure 7-6. Please contact the logic analyzer vendor to get the actual keep-out volume for their respective LAI implementation.

Figure 7-4. Front View of LAI Adapter Keepout Volume

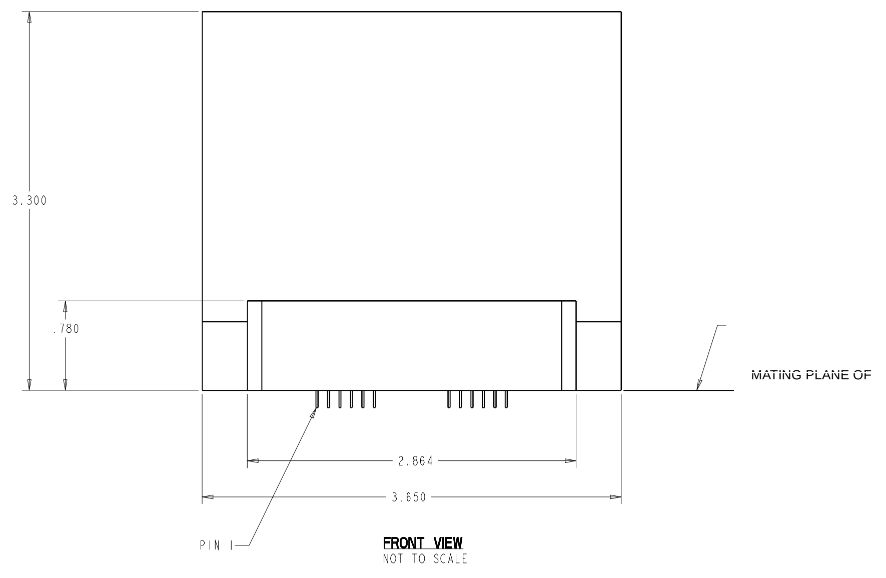


Figure 7-5. Side View of LAI Adapter Keepout Volume

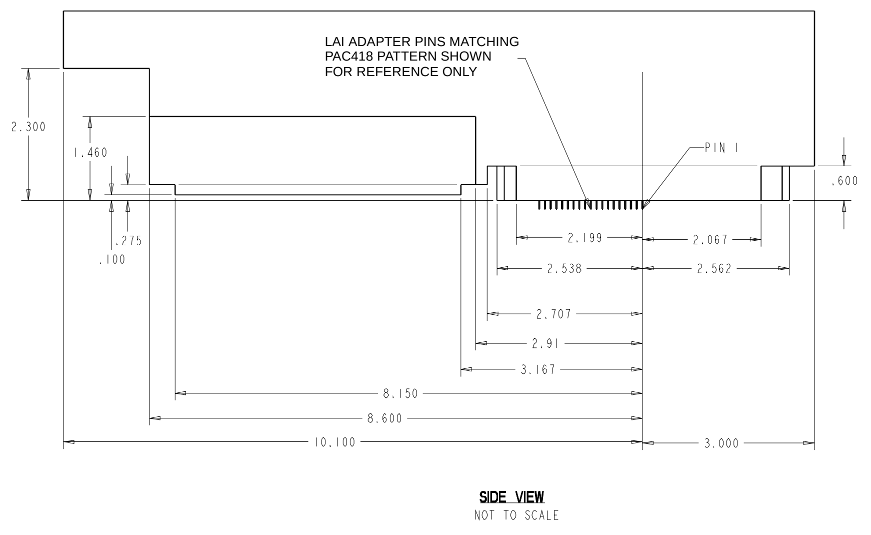
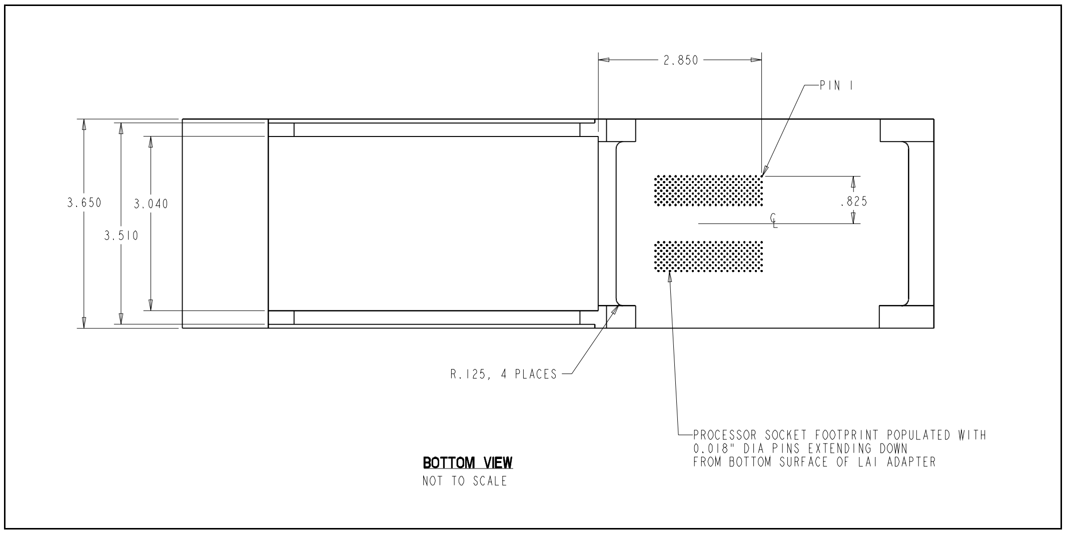


Figure 7-6. Bottom View of LAI Adapter Keepout Volume


2.850
PIN 1
3.650
3.040
3.510
.825
℄
R.125, 4 PLACES
PROCESSOR SOCKET FOOTPRINT POPULATED WITH
0.018" DIA PINS EXTENDING DOWN
FROM BOTTOM SURFACE OF LAI ADAPTER
BOTTOM VIEW
NOT TO SCALE

In addition to the system bus logic analyzer connection, consideration should be given to the other buses in the system. For initial debug boards, logic analyzer connections should be provided for monitoring the critical buses, including the LPC and PCI buses. If a PCI slot is present, logic analyzer vendors provide a plug-in card for easy connectivity. Contact the logic analyzer vendor for connector recommendations and part numbers.

Signals Reference A

This appendix provides an alphabetical listing of all Itanium processor system bus signals. The tables at the end of this appendix summarize the signals by direction: output, input, and I/O.

For a complete pinout listing including processor specific pins, please refer to the *Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet*.

A.1 Alphabetical Signals Reference

A.1.1 A[43:3]# (I/O)

The A[43:3]# (Address) signals define a 2^{44} Byte physical memory address space. When ADS# is active, these pins transmit the address of a transaction; when ADS# is inactive, these pins transmit transaction type information. These signals must connect the appropriate pins of all agents on the Itanium processor system bus. The A[43:24]# signals are parity-protected by the AP1# parity signal, and the A[23:3]# signals are parity-protected by the AP0# parity signal.

On the active-to-inactive transition of RESET#, the processors sample the A[43:3]# pins to determine their power-on configuration.

A.1.2 A20M# (I)

The A20M# (Address-20 Mask) signal may be asserted as an intended side effect of a particular I/O write originating from an OUT instruction. The platform must guarantee that the OUT instruction does not complete until this pin is asserted.

A20M# is ignored in the Itanium-based system environment.

A.1.3 ADS# (I/O)

The ADS# (Address Strobe) signal is asserted to indicate the validity of the transaction address on the A[43:3]# pins. All bus agents observe the ADS# activation to begin parity checking, protocol checking, address decode, internal snoop, or deferred reply ID match operations associated with the new transaction.

A.1.4 AP[1:0]# (I/O)

The AP[1:0]# (Address Parity) signals can be driven by the request initiator along with ADS#, A[45:3]#, REQ[4:0]#, and RP#. AP1# covers A[43:24]#, and AP0# covers A[23:3]#. A correct parity signal is high if an even number of covered signals are low and low if an odd number of covered signals are low. This allows parity to be high when all the covered signals are high.

A.1.5 ASZ[1:0]# (I/O)

The ASZ[1:0]# signals are the memory address-space size signals. They are driven by the request initiator during the first Request Phase clock on the REQa[4:3]# pins. The ASZ[1:0]# signals are valid only when REQa[1:0]# signals equal 01B, 10B, or 11B, indicating a memory access transaction. The ASZ[1:0]# decode is defined in Table A-1.

Table A-1. Address Space Size

ASZ[1:0]#		Memory Address Space	Memory Access Range
0	0	32-bit	0 to (4 Gbyte –1)
0	1	36-bit	4 Gbyte to (64 Gbyte –1)
1	0	44-bit	64 Gbyte to (16 Tbyte –1)
1	1	Reserved	Reserved

All observing bus agents that support the 4 GByte (32-bit) address space must respond to the transaction only when ASZ[1:0]# equals 00. All observing bus agents that support the 64 GByte (36-bit) address space must respond to the transaction when ASZ[1:0]# equals 00B or 01B. All observing bus agents that support the 16 TByte (44-bit) address space must respond to the transaction when ASZ[1:0]# equals 00B, 01B, or 10B.

A.1.6 ATTR[7:0]# (I/O)

The ATTR[7:0]# signals are the attribute signals. They are driven by the request initiator during the second clock of the Request Phase on the Ab[31:24]# pins. The ATTR[7:0]# signals are valid for all transactions. The ATTR[7:3]# are reserved and undefined. The ATTR[2:0]# are driven based on the memory type. Please refer to Table A-2.

Table A-2. Effective Memory Type Signal Encoding

ATTR[2:0]#	Effective Memory Type	Description
000	UC	Uncacheable
100	WC	Write Coalescing
101	WT	Write-Through
110	WP	Write-Protect
111	WB	Writeback

A.1.7 BCLKP / BCLKN (I)

The BCLKP and BCLKN differential clock signals determine the bus frequency. All agents drive their outputs and latch their inputs on the differential crossing of BCLKP and BCLKN when they are using the common clock latched protocol.

BCLKP and BCLKN indirectly determine the internal clock frequency of the Itanium processor. Each Itanium processor derives its internal clock by multiplying the BCLKP and BCLKN frequency by a ratio that is defined and allowed by the power-on configuration.

A.1.8 BE[7:0]# (I/O)

The BE[7:0]# signals are the byte-enable signals for partial transactions. They are driven by the request initiator during the second Request Phase clock of on the Ab[15:8]# pins.

For memory or I/O transactions (REQa[4:0]# = {10000B, 10001B, XX01XB, XX10XB, XX11XB}) the byte-enable signals indicate that valid data is requested or being transferred on the corresponding byte on the 64-bit data bus. BE0# indicates D[7:0]# is valid, BE1# indicates D[15:8]# is valid,..., BE7# indicates D[63:56]# is valid.

For Special transactions ((REQa[4:0]# = 01000B) and (REQb[1:0]# = 01B)), the BE[7:0]# signals carry special cycle encodings as defined in Table A-3. All other encodings are reserved.

Table A-3. Special Transaction Encoding on Byte Enables

Special Transaction	Byte Enables[7:0]#
NOP	0000 0000
Shutdown	0000 0001
Tristate#	0000 0010
Halt	0000 0011
Sync (WBINVD)	0000 0100
Reserved	0000 0101
StopGrant Acknowledge	0000 0110
Reserved	0000 0111
xTPR Update	0000 1000

For Deferred Reply transactions, BE[7:0]# signals are reserved. The Defer Phase transfer length is always the same length as that specified in the Request Phase. A Bus Invalidate Line (BIL) transaction is the only exception to this rule.

On the Itanium processor, a BIL transaction may return one cache line (64 Bytes); however, the length of the data returned on a BIL transaction may change for future Itanium processor family members.

A.1.9 BERR# (I/O)

The BERR# (Bus Error) signal can be asserted to indicate an unrecoverable error without a bus protocol violation. BERR# assertion conditions are configurable at the system level. Configuration options enable BERR# to be driven as follows:

- Asserted by the requesting agent of a bus transaction after it observes an error
- Asserted by any bus agent when it observes an error in a bus transaction

When the bus agent samples an asserted BERR# signal, the processor enters a Machine Check Handler.

A.1.10 BINIT# (I/O)

The BINIT# (Bus Initialization) signal is asserted to signal any bus condition that prevents reliable future operation if enabled during the power-on configuration.

If BINIT# observation is enabled during power-on configuration, and BINIT# is sampled asserted, all bus state machines are reset. All agents reset their rotating IDs for bus arbitration to the same state as that after reset, and internal count information is lost. The L2 and L3 caches are not affected.

If BINIT# observation is disabled during power-on configuration, BINIT# is ignored by all bus agents with the exception of the central agent. The central agent must handle the error in a manner that is appropriate to the system architecture.

A.1.11 BNR# (I/O)

The BNR# (Block Next Request) signal is used to assert a bus stall by any bus agent that is unable to accept new bus transactions to avoid an internal transaction queue overflow. During a bus stall, the current bus owner cannot issue any new transactions.

Since multiple agents might need to request a bus stall at the same time, BNR# is a wire-OR signal. In order to avoid wire-OR glitches associated with simultaneous edge transitions driven by multiple drivers, BNR# is asserted and sampled on specific clock edges.

A.1.12 BPM[5:0]# (I/O)

The BPM[5:0]# signals are system support signals mainly used for inserting breakpoints and for performance monitoring. They can be configured as outputs from the processor that indicate the status of breakpoints (as inputs) and programmable counters (as outputs) used for monitoring performance.

A.1.13 BPRI# (I)

The BPRI# (Bus Priority-agent Request) signal is used to arbitrate for ownership of the system bus. Observing BPRI# asserted (as asserted by the priority agent) causes all other agents to stop issuing new requests, unless such requests are part of an ongoing locked operation.The priority agent keeps BPRI# asserted until all of its requests are completed, then releases the bus by deasserting BPRI#.

A.1.14 BR0# (I/O) and BR[3:1]# (I)

BR[3:0]# are the physical bus request pins that drive the BREQ[3:0]# signals in the system. The BREQ[3:0]# signals are interconnected in a rotating manner to individual processor pins. Table A-4 gives the rotating interconnection between the processor and bus signals.

Table A-4. BR0# (I/O), BR1#, BR2#, BR3# Signals Rotating Interconnect

Bus Signal	Agent 0 Pins	Agent 1 Pins	Agent 2 Pins	Agent 3 Pins
BREQ0#	BR0#	BR3#	BR2#	BR1#
BREQ1#	BR1#	BR0#	BR3#	BR2#
BREQ2#	BR2#	BR1#	BR0#	BR3#
BREQ3#	BR3#	BR2#	BR1#	BR0#

During power-up configuration, the central agent must assert the BR0# bus signal. All symmetric agents sample their BR[3:0]# pins on asserted-to-deasserted transition of RESET#. The pin on which the agent samples an asserted level determines its agent ID. All agents then configure their pins to match the appropriate bus signal protocol as shown in Table A-5.

Table A-5. BR[3:0]# Signals and Agent IDs

Pin Sampled Asserted on RESET#	Agent ID
BR0#	0
BR3#	1
BR2#	2
BR1#	3

A.1.15 BREQ[3:0]# (I/O)

The BREQ[3:0]# signals are the Symmetric-agent Arbitration Bus signals (called bus request). A symmetric agent *n* arbitrates for the bus by asserting its BREQ*n*# signal. Agent *n* drives BREQ*n*# as an output and receives the remaining BREQ[3:0]# signals as inputs.

The symmetric agents support distributed arbitration based on a round-robin mechanism. The rotating ID is an internal state used by all symmetric agents to track the agent with the lowest priority at the next arbitration event. At power-on, the rotating ID is initialized to three, allowing agent 0 to be the highest priority symmetric agent. After a new arbitration event, the rotating ID of all symmetric agents is updated to the agent ID of the symmetric owner. This update gives the new symmetric owner lowest priority in the next arbitration event.

A new arbitration event occurs either when a symmetric agent asserts its BREQ*n*# on an Idle bus (all BREQ[3:0]# previously deasserted), or the current symmetric owner deasserts BREQn# to release the bus ownership to a new bus owner *n*. On a new arbitration event, all symmetric agents simultaneously determine the new symmetric owner using BREQ[3:0]# and the rotating ID. The symmetric owner can park on the bus (hold the bus) provided that no other symmetric agent is requesting its use. The symmetric owner parks by keeping its BREQ*n*# signal asserted. On sampling BREQn# asserted by another symmetric agent, the symmetric owner deasserts BREQ*n*# as soon as possible to release the bus. A symmetric owner stops issuing new requests that are not part of an existing locked operation on observing BPRI# asserted.

A symmetric agent can deassert BREQ*n*# before it becomes a symmetric owner. A symmetric agent can reassert BREQ*n*# after keeping it deasserted for one clock.

A.1.16 CPUPRES# (O)

CPUPRES# can be used to detect the presence of an Itanium processor in a socket. A ground indicates that an Itanium processor is installed, while an open indicates that an Itanium processor is not installed.

A.1.17 D[63:0]# (I/O)

The D[63:0]# (Data) signals provide a 64-bit data path between various system bus agents. Partial transfers require one data transfer clock with valid data on the byte(s) indicated by asserted byte enables BE[7:0]#. Data signals that are not valid for a particular transfer must still have correct ECC (if data bus ECC is selected). If BE0# is asserted, D[7:0]# transfers the least significant byte.

If BE7# is asserted, D[63:56]# transfers the most significant byte. The data driver asserts DRDY# to indicate a valid data transfer.

A.1.18 D/C# (I/O)

The D/C# (Data/Code) signal is used to indicate data (1) or code (0), only during Memory Read transactions on REQa1#.

A.1.19 DBSY# (I/O)

The DBSY# (Data Bus Busy) signal is asserted by the agent that is responsible for driving data on the system bus to indicate that the data bus is in use. The data bus is released after DBSY# is deasserted.

A.1.20 DEFER# (I)

The DEFER# signal is asserted by an agent to indicate that the transaction cannot be guaranteed in-order completion. Assertion of DEFER# is normally the responsibility of the addressed memory agent or I/O agent.

A.1.21 DEN# (I/0)

The DEN# (Defer Enable) signal is driven on the bus on the second clock of the Request Phase on the Ab4# pin. DEN# is asserted to indicate that the transaction can be deferred by the responding agent.

A.1.22 DEP[7:0]# (I/O)

The DEP[7:0]# (Data Bus ECC Protection) signals provide optional ECC protection for D[63:0]# (Data Bus). They are driven by the agent responsible for driving D[63:0]#. During power-on configuration, DEP[7:0]# signals can be enabled for either ECC checking or no checking.

The ECC error correcting code can detect and correct single-bit errors and detect double-bit or nibble errors. Chapter 4, “Data Integrity”, provides more information about ECC.

A.1.23 DHIT# (I)

The DHIT# (Deferred Hit) signal is driven during the Deferred Phase by the deferring agent. For Bus Read Line (BRL) transactions on the bus DHIT# returns the final cache status that would have been indicated on HIT# for a transaction which was not deferred. The Itanium processor will ignore DHIT# for Bus Read Invalidate Line (BRIL), Bus Read Partial (BRP), and Bus Write Partial (BWP) transactions on the bus.

A.1.24 DID[7:0]# (I/O)

DID[7:0]# are Deferred Identifier signals. The requesting agent transfers these signals by using A[23:16]#. They are transferred on Ab[23:16]# during the second clock of the Request Phase on all transactions, but Ab[19:16]# is only defined for deferrable transactions (DEN# asserted).

DID[7:0]# is also transferred on Aa[23:16]# during the first clock of the Request Phase for Deferred Reply transactions.

The deferred identifier defines the token supplied by the requesting agent. DID[7:4]# carry the agent identifiers of the requesting agents (always valid) and DID[3:0]# carry a transaction identifier associated with the request (valid only with DEN# asserted). This configuration limits the bus specification to 16 bus masters with each one of the bus masters capable of making up to sixteen requests. Table A-6 shows the DID encodings.

Table A-6. DID[7:0]# Encoding

DID7#	DID[6]#	DID[5:4]#	DID[3:0]#
Agent Type	Reserved	Agent ID	Transaction ID

DID[7]# indicates the agent type. Symmetric agents use 0. Priority agents use 1. DID[5:4]# indicates the agent ID. Symmetric agents use their arbitration ID. DID[6]# is reserved. DID[3:0]# indicates the transaction ID for an agent. The transaction ID must be unique for all transactions issued by an agent which have not reported their snoop results.

The Deferred Reply agent transmits the DID[7:0]# (Ab[23:16]#) signals received during the original transaction on the Aa[23:16]# signals during the Deferred Reply transaction. This process enables the original requesting agent to make an identifier match and wake up the original request that is awaiting completion.

A.1.25 DPS# (I/O)

The DPS# (Deferred Phase Enable) signal is driven to the bus on the second clock of the Request Phase on the Ab3# pin. DPS# is asserted if a requesting agent supports transaction completion using the Deferred Phase. A requesting agent that supports the Deferred Phase will always assert DPS#. A requesting agent that does not support the Deferred Phase will always deassert DPS#.

A.1.26 DRATE# (I)

The DRATE# (Data Transfer Rate) signal configures the system bus data transfer rate. If DRATE# is asserted, the system bus operates at the 1x data transfer rate. If DRATE# is deasserted, the system bus operates at the 2x data transfer rate. DRATE# must be valid at the asserted-to-deasserted transition of RESET# and must not change value while RESET# is deasserted.

A.1.27 DRDY# (I/O)

The DRDY# (Data Ready) signal is asserted by the data driver on each data transfer, indicating valid data on the data bus. In a multi-cycle data transfer, DRDY# can be deasserted to insert idle clocks.

A.1.28 DSZ[1:0]# (I/O)

The DSZ[1:0]# (Data Size) signals are transferred on REQb[4:3]# signals in the second clock of the Request Phase by the requesting agent. The DSZ[1:0]# signals define the data transfer capability of the requesting agent as shown in Table A-7.

Table A-7. Data Transfer Rates of Requesting Agent (DSZ[1:0]#)

DSZ[1:0]#		Supported Rates
0	0	1x
0	1	2x
1	x	Reserved

A.1.29 EXF[4:0]# (I/O)

The EXF[4:0]# (Extended Function) signals are transferred on the Ab[7:3]# pins by the requesting agent during the second clock of the Request Phase. The signals specify any special functional requirement associated with the transaction based on the requestor mode or capability. The signals are defined in Table A-8.

Table A-8. Extended Function Signals

Extended Function Signal	Signal Name Alias	Function
EXF4#	Reserved	Reserved
EXF3#	SPLCK#/FCL#	Split Lock / Flush Cache Line
EXF2#	OWN#	Modified State Guaranteed
EXF1#	DEN#	Defer Enable
EXF0#	DPS#	Deferred Phase Supported

A.1.30 FCL# (I/O)

The FCL# (Flush Cache Line) signal is driven to the bus on the second clock of the Request Phase on the /Ab6# pin. FCL# is asserted to indicate that the memory transaction is initiated by the new global Flush Cache (FC) instruction.

A.1.31 FERR# (O)

The FERR# (Floating-point Error) signal is asserted when the Itanium processor detects an unmasked floating-point error. FERR# is included for compatibility and is never asserted in the Itanium-based system environment.

A.1.32 GSEQ# (I)

Assertion of the GSEQ# (Guaranteed Sequentiality) signal implies that the platform guarantees completion of the transaction without a retry while maintaining sequentiality.

A.1.33 HIT# (I/O) and HITM# (I/O)

The HIT# (Snoop Hit) and HITM# (Hit Modified) signals convey transaction snoop operation results. Any bus agent can assert both HIT# and HITM# together to indicate that it requires a snoop stall. The stall can be continued by reasserting HIT# and HITM# together.

A.1.34 ID[7:0]# (I)

The ID[7:0]# (Transaction ID) signals are driven by the deferring agent. The signals in the two clocks are referenced IDa[7:0]# and IDb[7:0]#. During both clocks, ID[7:0]# signals are protected by the IP0# parity signal for the first clock, and by the IP1# parity signal on the second clock.

IDa[7:0]# returns the ID of the deferred transaction which was sent on Ab[23:16]# (DID[7:0]#).

A.1.35 IDS# (I)

The IDS# (ID Strobe) signal is asserted to indicate the validity of ID[7:0]# in that clock and the validity of DHIT# and IP[1:0]# in the next clock.

A.1.36 IGNNE# (I)

The IGNNE# (Ignore Numeric Error) signal is asserted as an intended side effect of an I/O write originating from an OUT instruction. The platform must guarantee that the OUT instruction does not complete until this pin is asserted. IGNNE# is ignored in the Itanium-based system environment.

During active RESET#, each processor begins sampling the A20M#, IGNNE#, and LINT[1:0] values to determine the ratio of core-clock frequency to bus-clock frequency. On the active-to-inactive transition of RESET#, each processor latches these signals and freezes the frequency ratio internally. System logic must then release these signals for normal operation.

A.1.37 INIT# (I)

The INIT# (Initialization) signal triggers an unmasked interrupt to the processor. INIT# is usually used to break into hanging or idle processor states. Semantics required for platform compatibility are supplied in the PAL firmware interrupt service routine.

A.1.38 INT (I) (LINTx configured as INT)

INT is the 8259 Interrupt Request signal which indicates that an external interrupt has been generated. The interrupt is maskable. The processor vectors to the interrupt handler after the current instruction execution has been completed.

The LINT0 pin must be software configured to be used either as the INT signal or another local interrupt.

A.1.39 IP[1:0]# (I)

The IP[1:0]# (ID Parity) signals are driven on the second clock of the Deferred Phase by the deferring agent. IP0# protects the IDa[7:0]# and IDS# signals for the first clock, and IP1# protects the IDb[7:2, 0]# and IDS# signals on the second clock.

A.1.40 LEN[1:0]# (I/O)

The LEN[1:0]# (Data Length) signals are transmitted using REQb[1:0]# signals by the requesting agent in the second clock of Request Phase. LEN[1:0]# define the length of the data transfer requested by the requesting agent as shown in Table A-9. The LEN[1:0]#, HITM#, and RS[2:0]# signals together define the length of the actual data transfer.

Table A-9. Length of Data Transfer

LEN[1:0]#		Length
0	0	0 - 8 bytes
0	1	16 bytes
1	0	Reserved
1	1	64 bytes

A.1.41 LINT[1:0] (I)

LINT[1:0] are local interrupt signals. These pins are disabled after RESET#. LINT0 is typically software configured as INT, an 8259-compatible maskable interrupt request signal. LINT1 is typically software configured as NMI, a non-maskable interrupt. Both signals are asynchronous inputs.

During active RESET#, each processor begins sampling the A20M#, IGNNE#, and LINT[1:0] values to determine the ratio of core-clock frequency to bus-clock frequency. On the active-to-inactive transition of RESET#, each processor latches these signals and freezes the frequency ratio internally. System logic must then release these signals for normal operation.

A.1.42 LOCK# (I/O)

The LOCK# (Bus Lock) signal indicates to the system that a transaction must occur atomically. For a locked sequence of transactions, LOCK# is asserted from the beginning of the first transaction through to the end of the last transaction. A locked operation can be prematurely aborted (and LOCK# deasserted) if DEFER# is asserted during the first bus transaction of the sequence. The sequence can also be prematurely aborted if a hard error (such as a hard failure response) occurs on any one of the transactions during the locked operation.

For requests which would lock the system bus, the Itanium processor may optionally lock the system bus, fault, or complete the request non-atomically.

When the priority agent asserts BPRI# to arbitrate for bus ownership, it waits until it observes LOCK# deasserted. This enables the symmetric agents to retain bus ownership throughout the bus locked operation and guarantee the atomicity of the locked transaction.

A.1.43 NMI (I) (LINTX configured as NMI)

The NMI signal is the Non-maskable Interrupt signal. Asserting NMI causes an interrupt with an internally supplied vector value of 2. An external interrupt-acknowledge transaction is not generated. If NMI is asserted during the execution of an NMI service routine, it remains pending and is recognized after the EOI is executed by the NMI service routine. At most, one assertion of NMI is held pending.

NMI is rising-edge sensitive. Recognition of NMI is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. If asserted asynchronously, asserted and deasserted pulse widths of NMI must be a minimum of two clocks. This signal must be software configured to be used either as NMI or as another local interrupt (LINT1 pin).

A.1.44 OWN# (I/O)

The OWN# (Guaranteed Cache Line Ownership) signal is driven to the bus on the second clock of the Request Phase on the Ab5# pin. OWN# is asserted if cache line ownership is guaranteed. This allows a memory controller to ignore implicit writebacks.

A.1.45 PMI# (I)

The PMI# (Platform Management Interrupt) signal triggers the highest priority interrupt to the processor. PMI# is usually used by the system to trigger system events that will be handled by platform specific firmware.

A.1.46 PWRGOOD (I)

The PWRGOOD (Power Good) signal must be deasserted (L) during power-up, and must be asserted (H) after RESET# is first asserted by the system.

A.1.47 REQ[4:0]# (I/O)

The REQ[4:0]# are the Request Command signals. They are asserted by the current bus owner in both clocks of the Request Phase. In the first clock, the REQa[4:0]# signals define the transaction type to a level of detail that is sufficient to begin a snoop request. In the second clock, REQb[4:0]# signals carry additional information to define the complete transaction type. REQb[4:3]# signals transmit DSZ[1:0]# or the data transfer width rate information of the requestor for transactions that involve data transfer. REQb[2]# is reserved. REQb[1:0]# signals transmit LEN[1:0]# (the data transfer length information). In both clocks, REQ[4:0]# and ADS# are protected by parity RP#.

All receiving agents observe the REQ[4:0]# signals to determine the transaction type and participate in the transaction as necessary, as shown in Table A-10.

Table A-10.Transaction Types Defined by REQa# / REQb# Signals

Transaction	REQa[4:0]#					REQb[4:0]#				
	4	3	2	1	0	4	3	2	1	0
Deferred Reply	0	0	0	0	0	x	x	x	x	x
Reserved (Ignore)	0	0	0	0	1	x	x	x	x	x
Interrupt Acknowledge	0	1	0	0	0	DSZ[1:0]#		0	0	0
Special Transactions	0	1	0	0	0	DSZ[1:0]#		0	0	1
Reserved (Central Agent Response)	0	1	0	0	0	DSZ[1:0]#		0	1	x
Reserved (Central Agent Response)	0	1	0	0	1	DSZ[1:0]#		0	x	x
Interrupt	0	1	0	0	1	DSZ[1:0]#		1	0	0
Purge TC	0	1	0	0	1	DSZ[1:0]#		1	0	1
Reserved (Central Agent Response)	0	1	0	0	1	DSZ[1:0]#		1	1	x
I/O Read	1	0	0	0	0	DSZ[1:0]#		x	x	x
I/O Write	1	0	0	0	1	DSZ[1:0]#		x	x	x
Reserved (Ignore)	1	1	0	0	x	DSZ[1:0]#		x	x	x

Table A-10.Transaction Types Defined by REQa# / REQb# Signals (Continued)

Transaction	REQa[4:0]#					REQb[4:0]#				
	4	3	2	1	0	4	3	2	1	0
Memory Read & Invalidate	ASZ[1:0]#		0	1	0	DSZ[1:0]#		x	LEN[1:0]#	
Reserved (Memory Write)	ASZ[1:0]#		0	1	1	DSZ[1:0]#		x	LEN[1:0]#	
Memory Read	ASZ[1:0]#		1	D/C#	0	DSZ[1:0]#		x	LEN[1:0]#	
Memory Write	ASZ[1:0]#		1	WSNP#	1	DSZ[1:0]#		x	LEN[1:0]#	

A.1.48 RESET# (I)

Asserting the RESET# signal resets all processors to known states and invalidates their L2 and L3 caches without writing back Modified (M state) lines. RESET# must remain asserted for one microsecond for a “warm” reset; for a power-on reset, RESET# must stay asserted for at least one millisecond after V_{CC} and BCLKP have reached their proper specifications. On observing asserted RESET#, all system bus agents must deassert their outputs within two clocks.

A number of bus signals are sampled at the asserted-to-deasserted transition of RESET# for the power-on configuration.

Unless its outputs are tristated during power-on configuration, after asserted-to-deasserted transition of RESET#, the processor optionally executes its Built-In Self-Test (BIST) and begins program execution at the reset-vector

A.1.49 RP# (I/O)

The RP# (Request Parity) signal is driven by the requesting agent, and provides parity protection on ADS# and REQ[4:0]#.

A correct parity signal is high if an even number of covered signals are low and low if an odd number of covered signals are low. This definition allows parity to be high when all covered signals are high.

A.1.50 RS[2:0]# (I)

The RS[2:0]# (Response Status) signals are driven by the responding agent (the agent responsible for completion of the transaction).RSP# (I)

A.1.51 RSP# (I)

The RSP# (Response Parity) signal is driven by the responding agent (the agent responsible for completion of the current transaction) during assertion of RS[2:0]#, the signals for which RSP# provides parity protection.

A correct parity signal is high if an even number of covered signals are low and low if an odd number of covered signals are low. During the Idle state of RS[2:0]# (RS[2:0]#=000), RSP# is also high since it is not driven by any agent guaranteeing correct parity.

A.1.52 SBSY# (I/O)

The SBSY# (Strobe Bus Busy) signal is driven by the agent transferring data when it owns the strobe bus. SBSY# holds the strobe bus before the first DRDY# and between DRDY# assertions for a multiple clock data transfer. SBSY# is deasserted before DBSY# to allow the next data transfer agent to predrive the strobes before the data bus is released.

A.1.53 SPLCK# (I/O)

The SPLCK# (Split Lock) signal is driven in the second clock of the Request Phase on the Ab6# pin of the first transaction of a locked operation. It is driven to indicate that the locked operation will consist of four locked transactions.

A.1.54 STBN[3:0]# and STBP[3:0]# (I/O)

STBP[3:0]# and STBN[3:0]# (and DRDY#) are used to transfer data at the 2x transfer rate in lieu of BCLKP. They are driven by the data transfer agent with a tight skew relationship with respect to its corresponding bus signals, and are used by the receiving agent to capture valid data in its latches before use in its core. This functions like an independent double frequency clock constructed from a falling edge of either STBP[3:0]# or STBN[3:0]#. The data is synchronized by DRDY#. Each strobe is associated with 16 data bus signals and 2 ECC signals as shown in Table A-11.

Table A-11.STBp[3:0]# and STBn[3:0]# Associations

Strobe Bits	Data Bits	ECC Bits
STBP3#, STBN3#	D[63:48]#	DEP[7:6]#
STBP2#, STBN2#	D[47:32]#	DEP[5:4]#
STBP1#, STBN1#	D[31:16]#	DEP[3:2]#
STBP0#, STBN0#	D[15:0]#	DEP[1:0]#

A.1.55 TCK (I)

The TCK (Test Clock) signal provides the clock input for the IEEE 1149.1 compliant Test Access Port (TAP).

A.1.56 TDI (I)

The TDI (Test Data In) signal transfers serial test data into the Itanium processor. TDI provides the serial input needed for IEEE 1149.1 compliant Test Access Port (TAP).

A.1.57 TDO (O)

The TDO (Test Data Out) signal transfers serial test data out from the Itanium processor. TDO provides the serial output needed for IEEE 1149.1 compliant Test Access Port (TAP).

A.1.58 THERMTRIP# (O)

The THERMTRIP# (Thermal Trip) signal protects the Itanium processor from catastrophic overheating by use of an internal thermal sensor. This sensor is set well above the normal operating temperature to ensure that there are no false trips. Data will be lost if the processor goes into thermal trip (signaled to the system by the assertion of the THERMTRIP# signal). Once THERMTRIP# is asserted, the platform must assert RESET# to protect the physical integrity of the processor.

A.1.59 THRMALERT# (O)

A thermal alert open-drain signal (indicated to the system by the THRMALERT# pin) brings the ALERT# interrupt output from the thermal sensor located on the Itanium processor cartridge out to the front side bus. The signal is asserted when the measured temperature from the processor thermal diode equals or exceeds the temperature threshold data programmed in the high-temp (THIGH) or low-temp (TLOW) registers on the sensor. This signal can be used by the platform to implement thermal regulation features.

A.1.60 TMS (I)

The TMS (Test Mode Status) signal is an IEEE 1149.1 compliant Test Access Port (TAP) specification support signal used by debug tools.

A.1.61 TND# (I/O)

The TND# (TLB Purge Not Done) signal is asserted to delay completion of a TLB Purge instruction, even after the TLB Purge transaction completes on the system bus.

A.1.62 TRDY# (I)

The TRDY# (Target Ready) signal is asserted by the target to indicate that it is ready to receive a write or implicit writeback data transfer.

A.1.63 TRISTATE# (I)

The TRISTATE# pin, sampled during the reset configuration, notifies the Itanium processor to tristate output pins.

A.1.64 TRST# (I)

The TRST# (TAP Reset) signal is an IEEE 1149.1 compliant Test Access Port (TAP) support signal used by debug tools.

A.1.65 WSNP# (I/O)

The WSNP# (Write Snoop) signal indicates that snooping agents will snoop the memory write transaction

A.2 Signal Summaries

The following tables list attributes of the Itanium processor output, input, and I/O signals.

Table A-12.Output Signals

Name	Active Level	Clock	Signal Group
CPUPRES#	Low	—	Platform
FERR#	Low	Asynchronous	IA-32 compatibility
TDO	High	TCK	Diagnostic
THERMTRIP#	Low	Asynchronous	Error
THRMALERT#	Low	Asynchronous	Error

Table A-13.Input Signals

Name	Active Level	Clock	Signal Group	Qualified/Valid
A20M#	Low	Asynch	IA-32 compatibility	Always[1]
BPRI#	Low	BCLKP	Arbitration	Always
BR1#	Low	BCLKP	Arbitration	Always
BR2#	Low	BCLKP	Arbitration	Always
BR3#	Low	BCLKP	Arbitration	Always
BCLKP	High	—	Control	Always
BCLKN	High	—	Control	Always
D/C#	Low	BCLKP	Request	Request Phase (Mem Rd)
DEFER#	Low	BCLKP	Snoop	Snoop Phase
DHIT#	Low	BCLKP	Defer	IDS#+1
DRATE#	Low	BCLKP	Platform	Always
ID[7:0]#	Low	BCLKP	Defer	IDS#, IDS#+1
IDS#	Low	BCLKP	Defer	Always
IGNNE#	Low	Asynch	IA-32 compatibility	Always[1]
INIT#	Low	Asynch	Exec Control	Always[1]
INT (LINT0)	High	Asynch	Exec Control	—
IP[1:0]#	Low	BCLKP	Defer	IDS#+1
NMI (LINT1)	High	Asynch	Exec Control	—
RESET#	Low	BCLKP	Control	Always
RS[2:0]#	Low	BCLKP	Response	Always
RSP#	Low	BCLKP	Response	Always
PMI#	Low	Asynch	Exec Control	—
PWRGOOD	High	Asynch	Control	—
TCK	High	—	Diagnostic	Always
TDI	High	TCK	Diagnostic	Always
TMS	High	TCK	Diagnostic	Always
TRISTATE#	Low	Asynch	Exec Control	Only during reset configuration
TRST#	Low	Asynch	Diagnostic	Always
TRDY#	Low	BCLKP	Response	Response Phase
GSEQ#	Low	BCLKP	Snoop	Snoop Phase

1. Synchronous assertion with asserted RS[2:0]# guarantees synchronization.

Table A-14.Input/Output Signals (Single Driver)

Name	Active Level	Clock	Signal Group	Qualified/Valid
A[43:3]#	Low	BCLKP	Request	ADS#, ADS#+1
ADS#	Low	BCLKP	Request	Always
AP[1:0]#	Low	BCLKP	Request	ADS#, ADS#+1
ASZ[1:0]#	Low	BCLKP	Request	ADS#
ATTR[7:0]#	Low	BCLKP	Address	ADS#+1
BE[7:0]#	Low	BCLKP	Address	ADS#+1
BR0#	Low	BCLKP	Arbitration	Always
BPM[5:0]#	Low	BCLKP	Diagnostic	Always
D[63:0]#	Low	BCLKP	Data	DRDY#
D/C#	Low	BCLKP	Request	ADS#
DBSY#	Low	BCLKP	Data	Always
DEN#	Low	BCLKP	Address	ADS#+1
DEP[7:0]#	Low	BCLKP	Data	DRDY#
DID[7:0]#	Low	BCLKP	Address	ADS#+1
DPS#	Low	BCLKP	Address	ADS#+1
DSZ[1:0]#	Low	BCLKP	Request	ADS#+1
DRDY#	Low	BCLKP	Data	Always
EXF[4:0]#	Low	BCLKP	Address	ADS#+1
FCL#	Low	BCLKP	Address	ADS#+1
LEN[1:0]#	Low	BCLKP	Request	ADS#+1
LOCK#	Low	BCLKP	Arbitration	Always
OWN#	Low	BCLKP	Address	ADS#+1
REQ[4:0]#	Low	BCLKP	Request	ADS#, ADS#+1
RP#	Low	BCLKP	Request	ADS#, ADS#+1
SBSY#	Low	BCLKP	Data	Always
SPLCK#	Low	BCLKP	Address	ADS#+1
STBN[7:0]#	Low	—	Data	Always
STBP[7:0]#	Low	—	Data	Always
WSNP#	Low	BCLKP	Request	ADS#

Table A-15.Input/Output Signals (Multiple Driver)

Name	Active Level	Clock	Signal Group	Qualified/Valid
BNR#	Low	BCLKP	Arbitration	Always
BERR#	Low	BCLKP	Error	Always
BINIT#	Low	BCLKP	Error	Always
HIT#	Low	BCLKP	Snoop	Snoop Phase
HITM#	Low	BCLKP	Snoop	Snoop Phase
TND#	Low	BCLKP	Snoop	Always

Index