Intel® Itanium® 2 Processor

Hardware Developer's Manual

July 2002

Document Number: 251109-001

Contents

Figures

Tables

Introduction 1

The Intel® Itanium® 2 processor, the second in a family of processors based on the Itanium architecture, is designed to address the needs of high-performance servers and workstations. The Itanium architecture goes beyond RISC and CISC approaches by employing Explicitly Parallel Instruction Computing (EPIC), which pairs extensive 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 multiple operating systems, including versions of Microsoft Windows*, HP-UX* and Linux*. The Itanium 2 processor is designed to support very large scale systems, including those employing thousands of processors, to provide the processing power and performance head room for the most demanding enterprise and technical computing applications. SMBus compatibility and comprehensive reliability, availability and serviceability (RAS) features make the Itanium 2 processor ideal for applications requiring high up-time. For high performance servers and workstations, the Itanium 2 processor offers outstanding performance and reliability for today's applications and the scalability to address the growing e-business needs of tomorrow.

1.1 Itanium® 2 Processor System Bus

Most Itanium 2 processor signals use the Itanium processor's Assisted Gunning Transceiver Logic (AGTL+) signaling technology. The termination voltage, V_{CTERM}, is generated on the baseboard and is the system bus high reference voltage. The buffers that drive most of the system bus signals on the Itanium 2 processor are actively driven to V_{CTERM} during a low-to-high transition to improve rise times and reduce noise. These signals should still be considered open-drain and require termination to V_{CTERM} which provides the high level. When on-die termination is enabled, the Itanium 2 system bus is terminated to V_{CTERM} through active termination within the bus agents at each end of the bus. There is also support of off-die termination in which case the termination is provided by external resistors connected to V_{CTERM}.

AGTL+ inputs use differential receivers which require a reference signal (V_{REF}). V_{REF} is used by the receivers to determine if a signal is a logical 0 or a logical 1. The Itanium 2 processor generates V_{REF} on-die, thereby eliminating the need for an off-chip reference voltage source.

1.2 Processor Abstraction Layer

The Itanium 2 processor requires implementation-specific Processor Abstraction Layer (PAL) firmware. PAL firmware supports processor initialization, error recovery, and other functionality. It provides a consistent interface to system firmware and operating systems across processor hardware implementations. The *Intel® Itanium™ Architecture Software Developer's Manual,* Volume 2: System Architecture, describes PAL. Platforms must provide access to the firmware address space and PAL at reset to allow Itanium 2 processors to initialize.

The System Abstraction Layer (SAL) firmware contains platform-specific firmware to initialize the platform, boot to an operating system, and provide runtime functionality. Further information about SAL is available in the *Itanium Processor Family System Abstraction Layer Specification*.

1.3 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).

In many cases, signals are mapped one-to-one to physical pins with the same names. In other cases, different signals are mapped onto the same pin. For example, this is the case with the address pins A[49:3]#. During the first clock, the address pins are asserted indicating a valid address. The first clock is indicated by the lower case a, or just the pin name itself: Aa[49:3]# or A[49:3]#. During the second clock, other information is asserted on the address pins. These signals are referenced either by their functional signal names, such as DID[9:0]#, or by using a lower case b with the pin name, such as Ab[25:16]#. Note also that several pins have configuration functions at the asserted to deasserted edge of RESET#.

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.

A signal name has all capitalized letters, e.g. VCTERM.

A symbol referring to a voltage level, current level, or a time value carries a plain subscript, e.g. $V_{CC,core}$, or a capitalized abbreviated subscript, e.g. T_{CO}.

1.4 Reference Documents

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

Title	Document Number
Intel® Itanium® 2 Processor at 1.0 GHz and 900 MHz Datasheet	250945
Intel® Itanium® 2 Processor Specification Update	251141
Intel® Itanium™ Architecture Software Developer's Manual	
• Volume 1: Application Architecture	245317
• Volume 2: System Architecture	245318
• Volume 3: Instruction Set Reference	245319
Intel® Itanium® 2 Processor BSDL Model	
Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization	251110
Intel® Itanium™ Processor Family System Abstraction Layer Specification	245359
Intel® Itanium™ Processor Family Error Handling Guide	249278
ITP700 Debug Port Design Guide	249679
System Management Bus Specification	http://www.smbus.org/specs

Contact your Intel representative or check http://developer.intel.com for the latest revision of the reference documents.

1.4.1 Revision History

Version Number	Description	Date
001	Initial release of this document.	July 2002

Itanium® 2 Processor Microarchitecture 2

This chapter provides an introduction to the Itanium 2 processor microarchitecture. For detailed information on Itanium architecture, please refer to the *Intel® Itanium™ Architecture Software Developer's Manual*.

2.1 Overview

The Itanium 2 processor is the second implementation of the Itanium Instruction Set Architecture (ISA). The processor employs EPIC 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 2 processor provides a 6-wide, 8-stage deep pipeline running at either 1.0 GHz or 900 MHz. This provides a combination of both abundant resources to exploit ILP as well as increased frequency for minimizing the latency of each instruction. The resources consist of six integer units, six multimedia units, two load and two store units, three branch units, two extended-precision floating-point units, and two additional single-precision floating-point units. The hardware employs dynamic prefetch, branch prediction, a register scoreboard, and non-blocking caches. Three levels of on-die cache minimize overall memory latency. This includes either a 3 MB or 1.5MB L3 cache, accessed at core speed, providing over 32 GB/cycle of data bandwidth. The system bus is designed for glueless MP support for up to 4 processors per system bus, and can be used as an effective building block for very large systems. The balanced core and memory subsystem provide high performance for a wide range of applications ranging from commercial workloads to high performance technical computing.

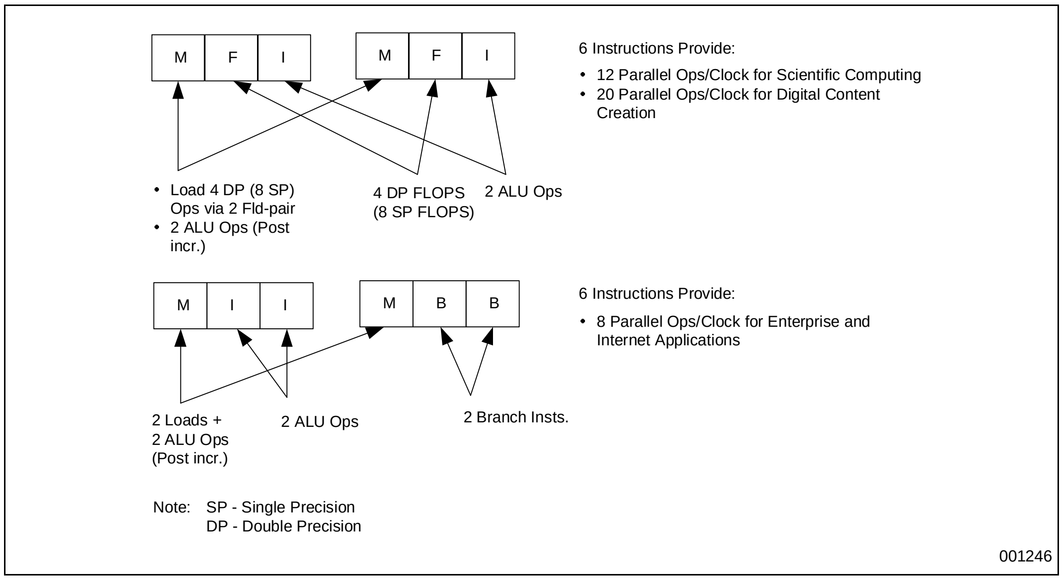
2.1.1 6-Wide EPIC Core

The Itanium 2 processor provides a 6-wide, 8-stage deep pipeline, based on the EPIC design. The pipelines utilize the following execution units: six Integer ALUs, six Multimedia ALUs, two Extended Precision Floating-point Units, two additional Single Precision Floating-point Units, two Load and two Store Units, and three Branch Units. The machine is capable of fetching, issuing, executing, and retiring six instructions, or two instructions bundles, per clock.

An instruction bundle contains three instructions and a template indicator, assigned by the compiler. Each instruction in the bundle is eventually dispersed into one of the execution pipelines according to its type: ALU Integer (A), Non-ALU Integer (I), Memory (M), Floating-point (F), Branch (B), or Extended (L). The Itanium 2 processor's increase in execution units more than triples the dispersal options for the compiler over the Itanium processor. Please refer to the *Intel® Itanium™ Architecture Software Developer's Manual* for more information regarding instructions and bundles, and the *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization* for more information regarding Itanium 2 processor instruction dispersal.

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 six instructions or eight parallel ops per clock (two load/store, two general-purpose ALU ops, two post-increment ALU ops, and two branch instructions). Alternatively, an MIB/MIB pair allows the same mix of operations, but with one branch hint and one branch op, instead of two branch ops. For scientific code, the use of the MFI template in each bundle enables twelve parallel Ops per clock (loading four double-precision operands to the registers, executing four double-precision flops, two integer ALU ops and two 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 twenty parallel ops per clock (loading eight single precision operands, executing eight single precision FLOPs, two integer ALUs, and two post-incrementing ALU operations).

Figure 2-1. Two Examples Illustrating Supported Parallelism

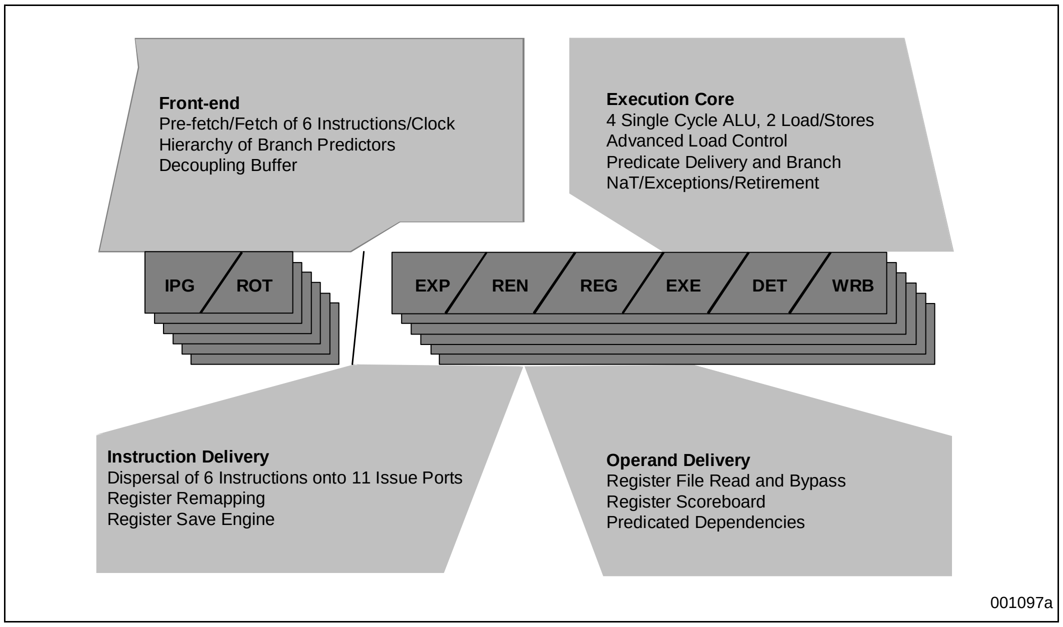

M
F
I
M
F
I
6 Instructions Provide:
• 12 Parallel Ops/Clock for Scientific Computing
• 20 Parallel Ops/Clock for Digital Content Creation
• Load 4 DP (8 SP) Ops via 2 Fld-pair
• 2 ALU Ops (Post incr.)
4 DP FLOPS (8 SP FLOPS)
2 ALU Ops
M
I
I
M
B
B
6 Instructions Provide:
• 8 Parallel Ops/Clock for Enterprise and Internet Applications
2 Loads + 2 ALU Ops (Post incr.)
2 ALU Ops
2 Branch Insts.
Note: SP - Single Precision
DP - Double Precision
001246

2.1.2 Processor Pipeline

The processor hardware is organized into a eight stage core pipeline, shown in Figure 2-2, that can execute up to six instructions in parallel per clock. The first two pipeline stages perform the instruction fetch and deliver the instructions into a decoupling buffer in the instruction rotation (ROT) 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, expand (EXP) and register rename (REN). Operand delivery is accomplished across the register read (REG) stage, 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 exception detection (DET) stage accommodates branch resolution as well as memory exception management and speculation support.

Please see the *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization* for more information the Itanium 2 processor pipeline.

Figure 2-2. Itanium® 2 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 and Branch
NaT/Exceptions/Retirement
IPG
ROT
EXP
REN
REG
EXE
DET
WRB
Instruction Delivery
Dispersal of 6 Instructions onto 11 Issue Ports
Register Remapping
Register Save Engine
Operand Delivery
Register File Read and Bypass
Register Scoreboard
Predicated Dependencies
001097a

2.1.3 Processor Block Diagram

Figure 2-3 shows a block diagram of the Itanium 2 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 block contains the logic for instruction prefetch, instruction fetch, L1 instruction cache, branch prediction, instruction address generation, instruction buffers, instruction issue, dispersal and rename.

2. **Execution**

   The execution block consists of the multimedia logic, integer ALU execution logic, floating-point (FP) execution logic, integer register file, L1 data cache and FP register file.

3. **Control**

   The control block consists of the exception handler and the pipeline control, as well as the Register Stack Engine (RSE).

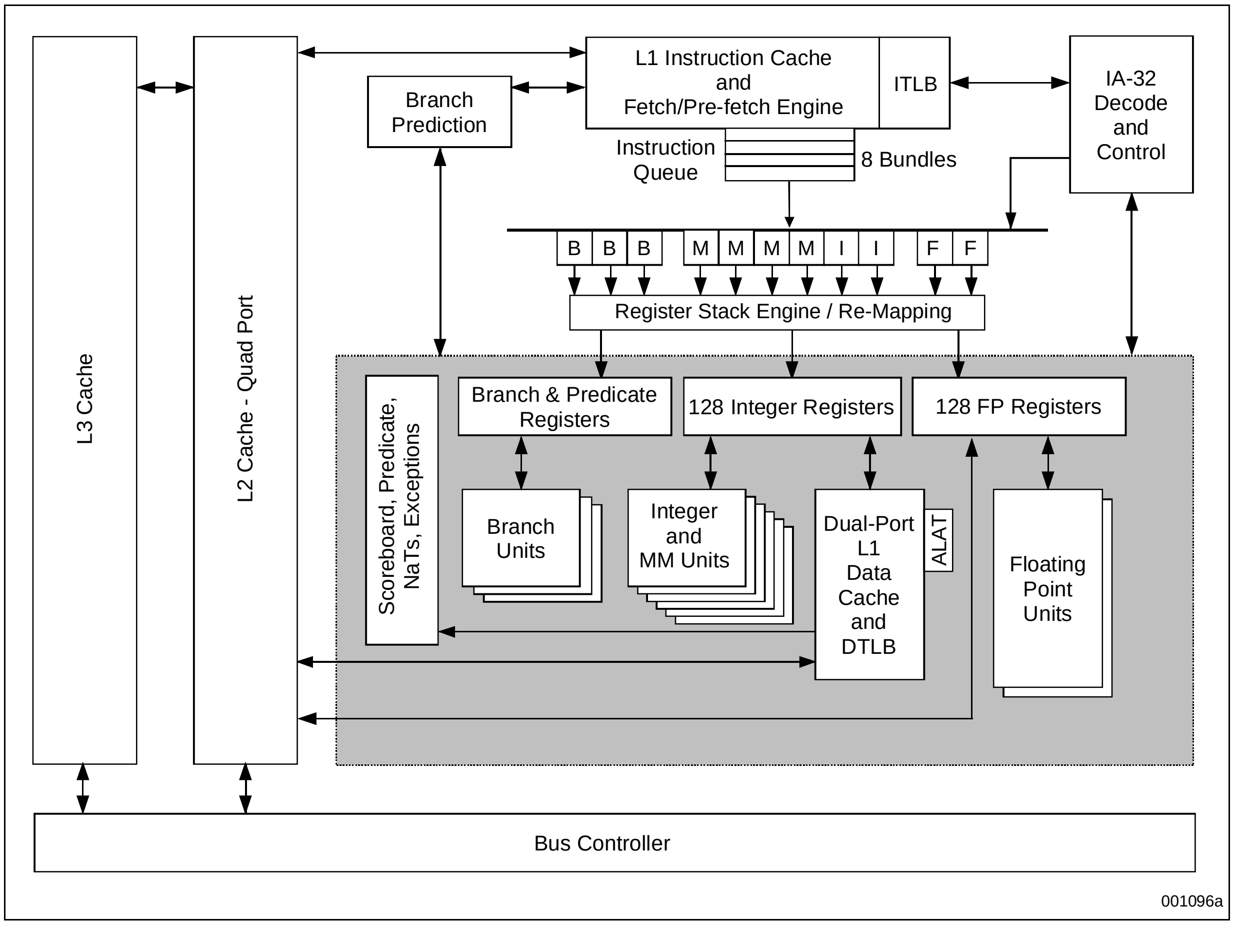
4. **Memory Subsystem**

   The memory subsystem contains the unified L2 cache, on-chip L3 cache, Programmable Interrupt Controller (PIC), instruction and data Translation Lookaside Buffers (TLB), Advanced Load Address Table (ALAT) and external system bus interface logic.

5. **IA-32 Compatibility Execution Engine**

   Instructions for IA-32 applications are fetched, decoded and scheduled for execution by the IA-32 compatibility execution engine.

Figure 2-3. Itanium® 2 Processor Block Diagram



2.2 Instruction Processing

2.2.1 Instruction Prefetch and Fetch

The Itanium 2 processor speculatively prefetches instructions from a pipelined cache into a decoupling buffer. The Itanium 2 uses a sophisticated branch prediction strategy and compiler hints for speculative prefetches. The instruction sequencing portion of the Itanium 2 processor is responsible for fetching and dispersing instructions to the execution units. The instruction address generation unit selects the next instruction pointer (IP). The instruction pointer is selected between the next sequential address, static and dynamic branch prediction addresses, instruction addresses delivered by the compatibility logic, validated target and address to correct for mispredicted branches, or the address of exception handlers.

The Itanium 2 processor reads two instruction bundles (three instructions per bundle) from the L1 instruction cache (L1I) and places them in the instruction buffers. The instruction buffers store bundles of instructions waiting to be consumed by the execution units. To reduce the effect of branch prediction bubbles caused by instruction cache misses, bundles read from the instruction buffers are sent to the instruction issue and rename logic based on the availability of execution resources.

2.2.2 Branch Prediction

The branch prediction logic uses advanced prediction schemes to anticipate the direction and target of each branch read from the instruction cache. The Itanium 2 processor features a 0-bubble branch prediction algorithm and a backup branch prediction table.   Whenever a branch happens, the branch target will be restored to the instruction pointer generation logic.

The instruction prefetch logic serves as the interface between the L1I and L2 cache. It prefetches instructions from L2 before they are needed in order to prevent L1I misses. Prefetching is executed under control of the compiler. If an L1 instruction cache miss does occur, it will stall the instruction address generation logic and retrieve the information from the L2 cache. If the instruction does not reside in L2 cache, it will proceed to check the L3 cache.

2.2.3 Dispersal Logic

There are twelve templates for Itanium instructions. A template contains explicit stop bits to indicate to the hardware to stop parallel issue of subsequent instructions. There are three instructions per bundle and the hardware can handle two bundles (i.e. six instructions) per clock. The dispersal logic sends each instruction to one of the fully pipelined functional units through its issue ports.

The instruction buffer holds a maximum of eight instruction bundles. The buffer can present two bundles to the dispersal logic every cycle. In general, instructions are routed to a supporting execution port on a first available basis.

2.3 Execution

The Itanium 2 processor execution logic consists of six multimedia units, six integer units, two floating-point units, three branch units and four load/store units. The Itanium 2 processor has general registers and FP registers to manage work in progress. Integer loads are processed by the L1 data cache but integer stores will be processed by L2. FP loads and stores are also processed by the L2 cache. Whenever a lookup occurs in L1, a speculative request is sent to the L2 cache.

The multimedia engines treat the 64-bit data as 2 x 32-bit, 4 x 16-bit or 8 x 8-bit packed data types. Three classes of arithmetic operations can be performed on the packed or Single Instruction Multiple Data (SIMD) data types: arithmetic, shift and data arrangement. Meanwhile the integer engines support up to six non-packed integer arithmetic and logical operations. Up to six integer or multimedia operations can be executed each cycle.

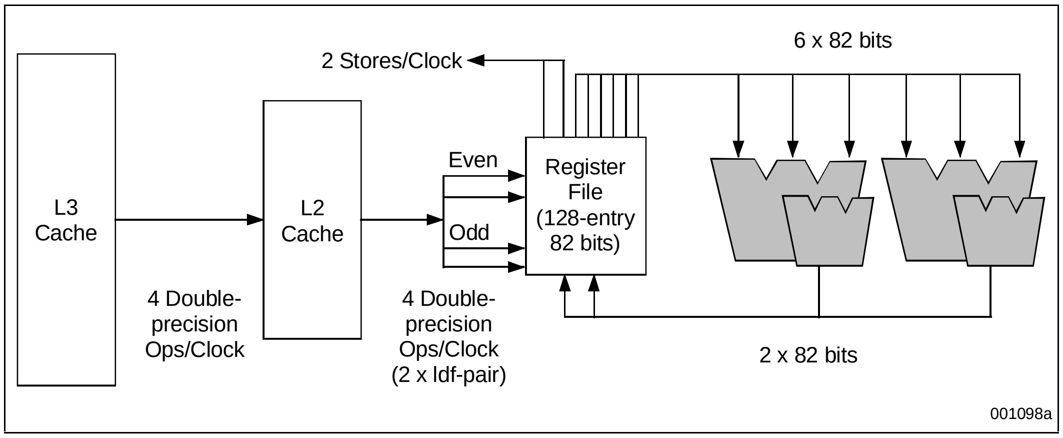
2.3.1 Floating-Point Unit (FPU)

The Itanium 2 processor provides high floating-point execution bandwidth. The Itanium 2 processor FPU has four pipeline stages. Extra bypassing logic allows quick data forwarding from various FP stages to the FP write back stage. The FP logic also includes an FP Multiply Accumulate (FMAC) hardware unit, fast rounding logic and support for SIMD formats. The Itanium 2 processor can issue up to two FP instructions, or two Integer multiplications, plus two FP loads and two FP stores (or four FP loads) instructions every clock cycle.

Numeric operands are checked for possible numeric exceptions before the instruction enters the FP pipeline. Results are written back at the end of the pipeline.

The FPU supports two FMACs that operate on 82-bit values. The FMACs can execute single, double and double-extended precision FP operations. The FPU has a 128-entry FP register file with eight read and at least six write ports. The FP registers can support four double precision loads every clock from memory, two 82-bit writebacks from the FMACs and two store operations for the two parallel extended precision FMACs every clock. Refer to Figure 2-4 for a diagram of the FMAC units.

Figure 2-4. Itanium® 2 Processor FMAC Units


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

2.3.2 Integer Logic

The six integer execution units execute 64-bit arithmetic, logical, shift and bit-field manipulation instructions. Additionally it can execute instructions to accelerate operations on 32-bit pointers. Other operations include computing predicates, linear addresses and flag generation for the IA-32 compatible engine.

The integer logic has six general purpose ALUs and two load and two store ports. The ALUs have full bypassing capability.

2.3.3 Register Files

The Itanium 2 processor implements the massive register resources provided by the Itanium architecture. The large number of registers allow many operations to complete without reading from or writing to memory. The primary execution registers include: 128 general registers, 128 floating-point registers, 64 predicate registers, and 8 branch registers.

2.3.3.1 General Registers

A set of 128 (64-bit) general registers provide the central resource for all integer and integer multimedia computation. They are numbered GR0 through GR127, and are available to all programs at all privilege levels.

The general registers are partitioned into two subsets. General registers 0 through 31 are termed the static general registers. Of these, GR0 is special in that it always reads as zero when sourced as an operand, and attempting to write to GR0 causes an Illegal Operation fault. General registers 32 through 127 are termed the stacked general registers. The stacked registers are made available to a program by allocating a register stack frame consisting of a programmable number of local and output registers.

2.3.3.2 Floating-Point Registers

A set of 128 (82-bit) floating-point registers are used for all floating-point computation. They are numbered FR0 through FR127, and are available to all programs at all privilege levels. The floating-point registers are partitioned into two subsets. Floating-point registers 0 through 31 are termed the static floating-point registers. Of these, FR0 and FR1 are special. FR0 always reads as +0.0 when sourced as an operand, and FR1 always reads as +1.0. When either of these is used as a destination, a fault is raised.

Floating-point registers 32 through 127 are termed the rotating floating-point registers. These registers can be programmatically renamed to accelerate loops.

2.3.3.3 Predicate Registers

A set of 64 (1-bit) predicate registers are used to hold the results of compare instructions. These registers are numbered PR0 through PR63, and are available to all programs at all privilege levels. These registers are used for conditional execution of instructions.

The predicate registers are partitioned into two subsets. Predicate registers 0 through 15 are termed the static predicate registers. Of these, PR0 always reads as ‘1’ when sourced as an operand, and when used as a destination, the result is discarded. The static predicate registers are also used in conditional branching.

Predicate registers 16 through 63 are termed the rotating predicate registers. These rotating registers support efficient software pipeline loops.

2.3.3.4 Branch Registers

A set of 8 (64-bit) branch registers are used to hold branching information. They are numbered BR0 through BR7, and are available to all programs at all privilege levels. The branch registers are used to specify the branch target addresses for indirect branches.

2.3.4 Register Stack Engine (RSE)

The Itanium ISA avoids the spilling and filling of registers at procedure interfaces through a large register file and a mechanism for accessing the registers through an indirection base. The indirection mechanism allows stacking of register frames and sharing of inter-procedure variables through the register file.

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 callers’ registers. The old registers remain in the large on-chip physical register file as long as there is enough physical capacity. When the number of registers needed overflows the available physical capacity, a state machine called the Register Stack Engine (RSE) saves the registers to memory to free up the necessary registers needed for the upcoming call. The RSE maintains the illusion of an infinite number of registers.

On a call return, the base register is restored to the value that the caller was using to access registers prior to the call. Often a return is encountered even before these registers need to be saved, making it unnecessary to restore them. 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 2 processor implements the forced lazy mode of the RSE, as described in the *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization*.

The *Intel® Itanium™ Architecture Software Developer’s Manual* describes the RSE in more detail.

2.4 Control

The control section of the Itanium 2 processor is made up of the exception handler and pipeline control. The exception handler implements exception prioritizing. Pipeline control has a scoreboard to detect register source dependencies and a cache to support data speculation. The machine stalls only when source operands are not yet available. Pipeline control supports predication via predication registers.

The pipeline control section also contains a Performance Monitoring Unit designed to collect data that can be dumped for analyzing Itanium 2 processor performance.

2.5 Memory Subsystem

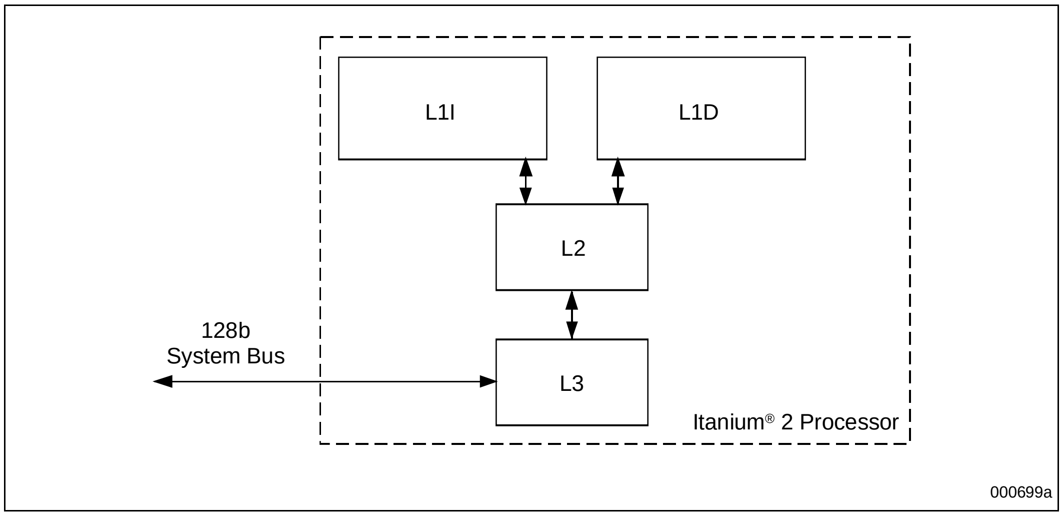
The main system memory is accessed through the 128-bit system bus (refer to Figure 2-5). The system bus is transaction-oriented and pipelined similar to the Itanium processor system bus. The memory subsystem for the Itanium 2 processor contains system bus interface logic, the L1D cache, the L2 cache, the L3 cache, interrupt controller unit, ALAT and TLB.

The Itanium 2 processor supports all non-aligned IA-32 memory accesses. References to memory in Itanium architecture spanning an 8 byte boundary will result in an unaligned fault. To avoid performance degradation associated with unaligned accesses and extra overhead for unaligned data memory fault handlers, aligned memory operands should be used whenever possible.

The L1, L2 and L3 caches are non-blocking. There are separate L1 caches for data and instructions. The L1 data cache is quad ported. The L2 cache is a unified cache and contains both instructions and data. It is quad ported and can be accessed at the full clock speed of the Itanium 2 processor. All ports are used when accessing instructions in L2 cache, but for data requests one can utilize either one, two, three or all of the four ports. When a request to the L2 cache causes a miss, the request is quickly forwarded to the L3 cache.

The integrated external interrupt controller interfaces to the system bus through the external bus logic and receives both external and internal interrupts from the system bus through its memory mapped location.

Figure 2-5. Itanium® 2 Processor Cache Hierarchy



2.5.1 L1 Instruction Cache

The Itanium 2 processor L1 instruction (L1I) cache is 16 KB in size. It is a single cycle, non-blocking, dual ported 4-way set-associative cache memory with a 64 byte line size (there is no way prediction). The tag array is dual ported. One port is for instruction fetches, the other port is shared among prefetches, snoops, fills, and column invalidates. The data array is also dual ported to support simultaneous reads (fetches) and fills. The L1I is fully pipelined and can deliver two instruction bundles (six instructions) every clock.

The L1I cache is physically indexed and tagged.

2.5.2 L1 Data Cache

The L1 data cache is four-ported (two loads and two stores), 16 KB in size and is non-blocking. It is organized as 4-way set-associative (no way prediction) with 64 byte line size. It can support two concurrent loads and two stores. The L1 data cache only caches integer data (does not cache floating-point load or semaphore load data). The L1D cache is write-through with no write allocation. The L1D cache is physically indexed and tagged for loads and stores.

2.5.3 Unified L2 Cache

The unified L2 cache memory is four-ported and supports up to four concurrent accesses via banking. The L2 cache is 256 KB, 8-way set-associative with a 128 byte line size, made of 16 byte banks and is non-blocking and out of order. It has a cache read bandwidth of 64 GB per second. The L2 cache implements a write-back with write-allocate policy. It is physically indexed and physically tagged.

In addition to servicing all L1I and L1D cache misses, the L2 handles all floating-point memory accesses (up to four concurrent floating-point loads per clock). All of the Itanium 2 processor’s semaphore instructions are also handled exclusively by the L2.

2.5.4 Unified L3 Cache

The on chip L3 cache on the Itanium 2 processor is 1.5 MB or 3 MB in size. It is physically indexed and physically tagged. The L3 cache is single ported, fully pipelined non-blocking cache featuring 12 way set-associative with 128 byte line size. It can support 8 outstanding requests, 7 of which are loads/stores and 1 is for fills. The maximum transfer rate from L3 to core/L1I/L1D or L2 is 32 GB/cycle. The L3 protects both tag and data with single bit correction and double bit detection ECC.

2.5.5 The Advanced Load Address Table (ALAT)

A cache structure called the Advanced Load Address Table (ALAT) is used to enable data speculation in the Itanium 2 processor. The ALAT keeps information on speculative data loads issued by the machine and any stores that are aliased with these loads. This structure has 32 entries, is a fully associative array that can handle two loads and two stores per cycle. It can provide aliasing information for the advance load “check” operations.

2.5.6 Translation Lookaside Buffers (TLBs)

There are two types of TLBs on the Itanium 2 processor: Data Translation Lookaside Buffer (DTLB) and the Instruction Translation Lookaside Buffer (ITLB). There are two levels of DTLBs in the Itanium 2 processor: a L1 DTLB and a L2 DTLB. Only L1D cache loads depend on the L1 and L2 DTLB hits. Stores and L2/L3 cache hits only depend on the L2 DTLB hits.

TLB misses in either the DTLB or the ITLB are serviced by the hardware page table walker which supports the Itanium instruction set architecture-defined 8B and 32B Virtual Hash Page Table (VHPT) format. VHPT data is only cached on the L2 and L3 caches, not the L1D.

2.5.6.1 The Data TLB (DTLB)

The first level DTLB (DTLB1) performs virtual to physical address translations for load transactions that hit in the L1 cache. It has two read ports and one write port. The TLB contains 32 entries and is fully associative. It supports 4 KB pages, and can also support subsets of larger caches in 4 KB subsections.

The second level DTLB (DTLB2) handles virtual to physical address translations for data memory references during stores, and protection checking on loads. It contains 128 entries and is fully associative and can support architected page sizes from 4 KB to 4 GB. The DTLB2 contains four ports. Of the 128 entries, 64 can be configured as Translation Registers (TR).

2.5.6.2 The Instruction TLB (ITLB)

The first level ITLB (ITLB1) is responsible for virtual to physical address translations to enable instruction transaction hits in the L1I cache. It is dual ported, contains 32 entries and is fully associative. It supports 4 KB pages only.

The second level ITLB (ITLB2) is responsible for virtual to physical address translations for instruction memory references that miss the ITLB1. It contains 128 entries, is fully associative and supports page sizes from 4 KB to 4 GB. Of the 128 entries, 64 can be configured as TR.

2.5.7 Cache Coherency

The three-level cache system makes it necessary to maintain the consistency of the data in the different caches. Every read access to a memory address must always provide the most up-to-date data at that address. Since the L1 is write-through it maintains a valid bit. The valid bit indicates whether or not the cache line is valid. The L2 and L3 caches use the MESI 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 2 processor, WC loads are performed directly from memory and not from the coalescing buffers.

On the Itanium 2 processor, a separate 2-entry, 128 byte buffer (WCB) is used for WC accesses exclusively. Each byte in the line has a valid bit. If all the valid bits are true, then the line is said to be full and will be evicted (flushed) by the processor. Line evictions are initiated in a “first-written-first-flushed” order even for partially full lines.

For increased performance to cacheable references to frame buffers or graphic controllers, the Itanium 2 processor allows external agents such as a graphics controller to read a line out of the processor's cache without altering the state of the cache line.

2.5.9 Memory Ordering

The Itanium 2 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 2 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

The Itanium 2 processor supports IA-32 application binaries. This includes support for running a mix of IA-32 applications and Itanium-based applications on an Itanium-based operating system (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.

System Bus Overview 3

This chapter provides an overview of the Itanium 2 processor system bus, bus transactions, and bus signals. The Itanium 2 processor also supports signals not discussed in this section. For a complete signal listing, please refer to the *Intel® Itanium® 2 Processor at 1.0 GHz and 900 MHz Datasheet* and Appendix A, "Signals Reference".

3.1 Signaling on the Itanium® 2 Processor System Bus

The Itanium 2 processor system bus supports common clock signaling as well as source synchronous data signaling. Section 3.1.1 and Section 3.1.2 describe in detail the characteristics of each type of signaling. The corresponding timing figures 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. Black bars indicate zero or more clocks are allowed.

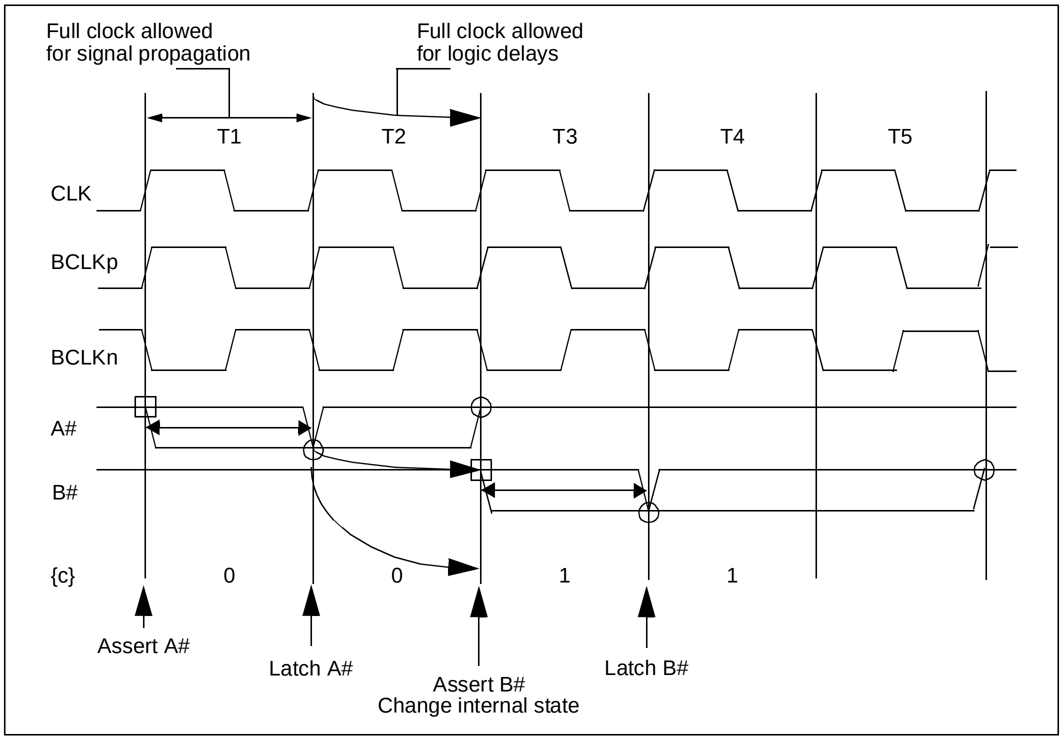
All timing diagrams in this specification show signals as they are asserted or deasserted. There is a one-clock delay in the signal values observed by system bus agents. Any signal names that appear in lowercase letters in brackets {rcnt} are internal signals only, and are not driven to the bus. Internal states change one clock after sampling a bus signal, which is the clock after the bus signal is driven. Uppercase letters that appear in brackets represent a group of signals such as the Request Phase signals [REQUEST]. The timing diagrams sometimes include internal signals to indicate internal states and show how it affects external signals. Internal states change one clock after sampling a bus signal. A bus signal is sampled one clock after the bus signal is driven.

3.1.1 Common Clock Signaling

All signals except the data bus signals on the system bus use a synchronous common clock latched protocol (1x transfer rate). 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 (1) every input be sampled during a valid sampling window on a rising clock edge and, (2) its effect be driven out no sooner than the next rising clock edge. This approach allows one full clock for driving a signal, flight time, and setup as well as 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 later 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 i.e. it 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 {c} 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



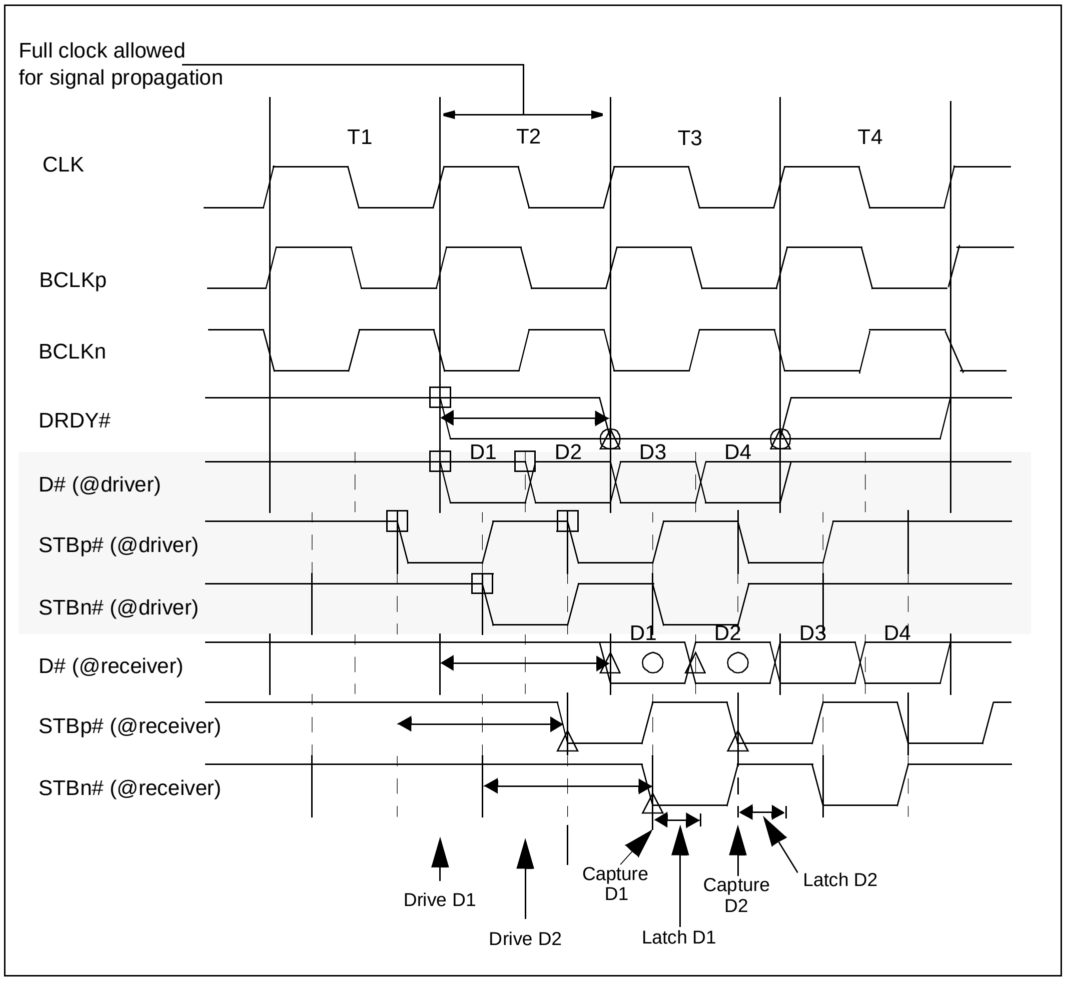
Signals that are driven in the same clock by multiple system bus agents exhibit a "wired-OR glitch" on the electrical low to electrical high transition. To account for this situation, these signal state transitions are specified to have two clocks of settling time when deasserted before they can be safely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

3.1.2 Source Synchronous Signaling

The data bus operates with a source synchronous latched protocol (2x transfer rate). The source synchronous latched protocol (refer to Figure 3-2) sends and latches data with strobes to allow very high transfer rates with reasonable signal flight times. The rest of the system bus always uses the common clock latched protocol.

The source synchronous latched protocol operates the data bus at twice the "frequency" of the common clock. Two chunks of data are driven onto the bus in the time it would normally take to drive one chunk. The worst case flight time is similar to the common clock latched protocol, so the second data transfer may be driven before the first is latched. On both the rising edge and 50% point of the bus clock, drivers send new data. On both the 25% point and the 75% point of the bus clock, drivers send centered differential strobes. The receiver captures the data with the strobes deterministically.

Figure 3-2. Source Synchronous Latched Protocol



The driver pre-drives STBp# before driving data. It sends a rising and falling edge on STBp# and STBn# centered with data. The driver must deassert 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. Data will be latched into the core within one core-cycle after being captured. A signal synchronous to the common clock (DRDY#) indicates to the receiver that valid data has been sent.

3.2 Signal Overview

This section describes the function of various Itanium 2 processor signals. In this section, the signals are grouped according to function. For a complete signal listing, please refer to the *Intel® Itanium® 2 Processor at 1.0 GHz and 900 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 Positive Phase Bus Clock (BCLKp) 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 Negative Phase Bus Clock (BCLKn) input signal is the negative phase of the system bus clock differential pair.

The RESET# signal resets all system bus agents to known states.

Note: The RESET# signal itself does not invalidate the internal caches in the Itanium 2 processor. A subsequent PAL call is used to invalidate all internal caches in the Itanium 2 processor. Modified or dirty cache lines are NOT written back. After RESET# is deasserted, each processor begins execution at the power-on reset vector defined during configuration.

The Power Good (PWRGOOD) input signal must be deasserted during power-on 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]#, BR[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.

LOCK# is never asserted or sampled in the Itanium 2 processor system environment.

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[5:0]#
Address	A[49:3]#
Address Parity	AP[1:0]#
Request Parity	RP#

The assertion of ADS# defines the beginning of the transaction. The REQ[5:0]#, A[49:3]#, AP[1:0]#, and RP# are valid in the clock that ADS# is asserted.

In the clock that ADS# is asserted, the A[49:3]# 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. AP[1]# protects the address signals A[49:27]#. AP[0]# protects the address signals A[26:3]#. 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 protected 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 protects the request pins REQ[5:0]# 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#

Table 3-4. Snoop Signals (Continued)

Signal Function	Signal Names
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 (`PTC.g`) instruction, even after the `PTC.g` transaction completes on the system bus. Software will guarantee that only one `PTC.g` 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 protection 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. All data transfers are at the 2x transfer rate.

Table 3-6. Data Signals

Signal Function	Signal Names
Data Ready	DRDY#, DRDY_C1#, DRDY_C2#
Data Bus Busy	DBSY#, DRDY_C1#, DRDY_C2#
Strobe Bus Busy	SBSY#, SBSY_C1#, SBSY_C2#
Data	D[127:0]#
Data ECC Protection	DEP[15:0]#
Positive phase Data Strobe	STBp[7:0]#
Negative phase Data Strobe	STBn[7:0]#

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 on the bus.

Each of the data bus control signals DBSY#, DRDY#, and SBSY# are replicated on the Itanium 2 processor system bus to enable partitioning of data path chips in the system agents. Two copies of DBSY#, DRDY#, and SBSY# signals are output-only and the third copy serves as both input as well as output.

The D[127:0]# signals provide a 128-bit data path between agents. For partial transfers, BE[7:0]# and A[4:3]# determine which bytes of the data bus contain valid data.

The DEP[15:0]# signals provide optional ECC (error correcting code) protection for D[127:0]#. DEP[15:0]# provides valid ECC protection for the entire data bus on each clock, regardless of which bytes are enabled.

STBp[7:0]# and STBn[7:0]# (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 pair is associated with sixteen data signals and two ECC signals as shown in Table 3-7.

Table 3-7. STBp[7:0]# and STBn[7:0]# Associations

Strobe Signals	Data Signals	ECC Signals
STBp[7]#, STBn[7]#	D[127:112]#	DEP[15:14]#
STBp[6]#, STBn[6]#	D[111:96]#	DEP[13:12]#
STBp[5]#, STBn[5]#	D[95:80]#	DEP[11:10]#
STBp[4]#, STBn[4]#	D[79:64]#	DEP[9:8]#
STBp[3]#, STBn[3]#	D[63:48]#	DEP[7:6]#
STBp[2]#, STBn[2]#	D[47:32]#	DEP[5:4]#
STBp[1]#, STBn[1]#	D[31:16]#	DEP[3:2]#
STBp[0]#, STBn[0]#	D[15:0]#	DEP[1:0]#

3.2.7 Defer Signals

The defer signals, shown in Table 3-8, 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[9:0]#

IDS# is asserted to begin the deferred response. ID[9:0]# returns the ID of the deferred transaction that was sent on DID[9:0]#. 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	THRMTRIP#
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.

THRMTRIP# is the Thermal Trip signal. The Itanium 2 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 THRMTRIP# 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 case temperature drops below specified maximum, a RESET# pulse will reset the processor.

A thermal alert open-drain signal, indicated to the system by the THRMALERT# pin. 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#
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.

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#. LINT[0] is typically software configured as INT, an 8259-compatible maskable interrupt request signal. LINT[1] is typically software configured as NMI, a non-maskable interrupt.

3.2.10 IA-32 Compatibility Signals

The following signals were present for compatibility with IA-32 system environments: FERR#, IGNNE#, and A20M#. As implemented on the Itanium 2 processor, the FERR# signal may be asserted while running an IA-32 application to indicate an unmasked floating point error, and the IGNNE# and A20M# signals are ignored.

3.2.11 Platform Signals

The platform signals, shown in Table 3-11, provides signals which support the platform.

Table 3-11. Platform Signals

Signal Function	Signal Names
Processor Present	CPUPRES#

CPUPRES# can be used to detect the presence of a Itanium 2 processor in a socket. A ground (GND) level indicates that the part is installed while an open indicates no part is installed.

3.2.12 Diagnostic Signals

The diagnostic signals, shown in Table 3-12, provides signals for probing the processor, monitoring processor performance, and implementing IEEE 1149.1 specification for boundary scan.

Table 3-12. 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.

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

Data Integrity 4

The Itanium 2 processor supports an advanced machine check architecture to facilitate error detection, containment, correction and recovery. The system bus includes parity protection for address, request and response signals, parity or protocol protection on most control signals, and ECC protection for data signals.

For more information on Machine Check Architecture, see the *Itanium™ Processor Family Error Handling Guide*.

4.1 Error Classification

The Itanium 2 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.

1. **Hardware Corrected Error**

   The error can be corrected by the processor or the system hardware. The current process continues without interruption.

2. **Firmware Corrected Error**

   The error can be corrected by firmware. The current process continues after Machine Check Abort (MCA) is serviced.

3. **Recoverable Error with Local MCA**

   The error cannot be corrected either by hardware or firmware. Only one agent is affected. Error handling is left to the OS and recovery may not always be possible.

4. **Recoverable Error with Global MCA**

   The error cannot be corrected either by hardware or firmware. Multiple agents on a bus may be affected. Error handling is left to the OS and recovery may not always be possible.

5. **Non-Recoverable Error with Global MCA**

   The error cannot be corrected either by hardware, firmware or OS. Multiple agents on a bus may be affected and the system needs to be restarted.

4.2 Itanium® 2 Processor System Bus Error Detection

The major address and data paths of the Itanium 2 processor system bus are protected by 18 check bits that provide either parity or ECC protection. Sixteen 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. 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[5:0]#
AP[0]#	A[26:3]#
AP[1]#	A[49:27]#
RSP#	RS[2:0]#
IP[0]#	IDS#, IDa[9:0]#
IP[1]#	IDS# (deasserted), IDb[9:2,0]#
DEP[7:0]#	D[63:0]#
DEP[15:8]#	D[127:64]#

- **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# is reported by the agent detecting the error as a non-recoverable error with global MCA if response parity is enabled.

- **Deferred Signals**

  A parity error detected on IP[1:0]# is reported by the agent detecting the error as a non-recoverable error with global MCA.

- **Data Transfer Signals**

  The Itanium 2 processor data 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 cause unrecoverable errors with local MCA.

4.2.2 Bus Signals Protected Indirectly

Some bus signals are not directly protected by parity or ECC. However, they can be indirectly protected due to a requirement to follow a strict protocol. Some processors or other bus agents may enhance error detection or correction for the bus by checking for protocol violations. P6 family processor system bus protocol errors are treated as fatal errors unless specifically stated otherwise.

4.2.3 Unprotected Bus Signals

The following Itanium 2 processor system bus signals are not protected by ECC or parity:

- BCLK, RESET#, PWRGOOD#, LINT[1:0]#, CPUPRES# and INIT# are not protected.
- The error signals THRMTRIP#, THRMALERT# are not protected.

4.2.4 Itanium® 2 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® 2 Processor System Bus ECC Algorithm

The Itanium 2 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.

Configuration and Initialization 5

This chapter describes configuration options and initialization details for the Itanium 2 processor.

A system may contain single or multiple Itanium 2 processors with one to four processors on a single system bus. Multiple system buses on a system are supported.

5.1 Configuration Overview

Itanium 2 processors have some configuration options that are determined by hardware, and some that are determined by PAL.

Itanium 2 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® 2 Processor at 1.0 GHz and 900 MHz Datasheet* for further details.

The Itanium 2 processor can also be configured with additional PAL configuration options. These options can be changed by procedure calls to PAL. These options should be changed only after taking into account synchronization between multiple Itanium 2 processor system bus agents.

5.2 Configuration Features

Table 5-1 specifies the system bus related configuration features on the Itanium 2 processor. These configuration features are supported using fields in implementation specific configuration registers. Some of these features are set by bus signals during reset (at the asserted-to-deasserted transition of RESET# signal) and some can be set by PAL.

The column labelled “Name” indicates the bus signal that affects the configuration field during reset. For a configuration feature, an “N/A” entry in this column indicates that the configuration field cannot be set by any bus signal during reset. The column labelled “Value” shows the recommended bus signal values for the features. For a configuration feature, a “0” entry in this column indicates that the bus signal is deasserted during reset, a “1” entry indicates that the bus signal is asserted during reset, and an “N/A” entry indicates that the configuration feature cannot be set by any bus signal during reset.

The column labelled “PAL Call” indicates the PAL call (if applicable) that allows control over the configuration features. For a configuration feature, an “N/A” entry in this column indicates that the configuration feature cannot be set by a PAL call and no PAL call is defined to read the configuration field.

The “Control” column indicates the PAL read and write control provided for the configuration fields. For a configuration feature, a “Read” entry in the column indicates that it can only be read by PAL, a “Read/Write” indicates that it can be read and modified by the PAL.

The “Default” column indicates the default values for configuration fields after reset. For configuration features that can be set by the bus signals, this column indicates the default set by the corresponding bus signal value indicated in the “Bus Signal Value” column.

Request bus parking feature may have certain performance impact based on the request traffic pattern and implementation of the system agent. A system can chose to set this feature depending on its requirement using A15# signal during reset.

Table 5-1. Power-On Configuration Features

Feature	Bus Signals		PAL Call	Control	Default
	Name	Value			
Data Error Checking Enabled	N/A	N/A	PAL_BUS_SET_FEATURES for Write control, and PAL_BUS_GET_FEATURES for Read control.	Read/Write	Disabled
Response/ID Error Checking Enabled	N/A	N/A			
Address/Request Error Checking Enabled	N/A	N/A			
BERR# Assertion Enabled	N/A	N/A			
BERR# Sampling Enabled	N/A	N/A			
BINIT# Assertion Enabled	N/A	N/A			
Cache Line Replacement Transaction Enabled on Replacement of Line in E State	N/A	N/A			
Cache Line Replacement Transaction Enabled on Replacement of Line in S State	N/A	N/A			
BINIT# Sampling Enabled	A10#	0			
Request Bus Parking Enabled	A15#	0			
In-Order Queue Depth of 1	A7#	0	PAL_BUS_GET_FEATURES	Read	Disabled i.e. default IOQ Depth is 8.
Output Tristate Enabled	A[31:28]#	0000	N/A	Read	Disabled
Symmetric Arbitration ID	BR0#, BR1#, BR2#, BR3#	BREQ0# must be asserted	PAL_FIXED_ADDR	Read	Based on bus mapping between BREQ0# and BR[3:0]#.
Clock Ratios	A[21:17]#	00000	PAL_FREQ_RATIOS	Read	2/8

5.2.1 Data Bus Error Checking

The Itanium 2 processor data bus error checking can be enabled or disabled. After RESET# is asserted, data bus error checking is always disabled. Prior to the transfer of control from PAL to the system, data parity error checking is enabled. Data bus error checking can be enabled through a call to PAL. For more information on this feature, please refer to the *Intel® Itanium™ Architecture Software Developer's Manual*.

5.2.2 Response/ID Signal Parity Error Checking

The Itanium 2 processor system bus supports parity protection for the response signals RS[2:0]# and the transaction ID signals ID[9:0]#. After RESET# is asserted, response signal parity checking is disabled. Prior to the transfer of control from PAL to the system, response parity signal checking is enabled. Response parity signal checking can be enabled or disabled by a call to PAL.

5.2.3 Address/Request Signal Parity Error Checking

The Itanium 2 processor address bus supports parity protection on the Request signals, A[49:3]#, ADS#, and REQ[4:0]#. After RESET# is asserted, request signal parity checking is disabled. Prior to the transfer of control from PAL to the system, address/request parity error checking is enabled. It can be enabled or disabled through a call to PAL.

5.2.4 BERR# Assertion for Initiator Bus Errors

A Itanium 2 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 through a call to PAL.

5.2.5 BERR# Assertion for Target Bus Errors

A Itanium 2 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 through a call to PAL.

5.2.6 BERR# Sampling

If the BERR# sampling policy is enabled, the BERR# input receiver causes a global Machine Check Abort (MCA). It may be enabled through a call to PAL.

5.2.7 BINIT# Error Assertion

If BINIT# error assertion is enabled, then the Itanium 2 processor system bus agent will assert the BINIT# signal in response to a bus protocol violation. After RESET# is asserted, BINIT# signal assertion is disabled. It may be enabled through a call to PAL.

5.2.8 BINIT# Error Sampling

The BINIT# input receiver is enabled for bus initialization control if A[10]# was sampled asserted on the asserted-to-deasserted transition of RESET#.

5.2.9 In-Order Queue Pipelining

Itanium 2 processor system bus agents are configured to an In-Order Queue depth of one if A[7]# is sampled asserted on the asserted to deasserted transition of RESET#. If A[7]# 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 through a call to PAL.

5.2.10 Request Bus Parking Enabled

Itanium 2 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 A[15]# is sampled asserted on the asserted-to-deasserted transition of RESET#. No processor will park on the request bus if A[15]# is sampled deasserted on the asserted-to-deasserted transition of RESET#.

5.2.11 Symmetric Agent Arbitration ID

The Itanium 2 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, 2, 4, or 6. 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. BREQ[3:0]# bus signals are connected to two symmetric agents in a rotating manner as shown in Table 5-2 and in Figure 5-2.Every symmetric agent has one I/O pin (BR0#) and three input only pins (BR1#, BR2#, and BR3#).

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

Bus Signal	Agent 0 Pins	Agent 1 Pins	Agent 2 Pins	Agent 3 Pins
BREQ[0]#	BR[0]#	BR[3]#	BR[2]#	BR[1]#
BREQ[1]#	BR[1]#	BR[0]#	BR[3]#	BR[2]#
BREQ[2]#	BR[2]#	BR[1]#	BR[0]#	BR[3]#
BREQ[3]#	BR[3]#	BR[2]#	BR[1]#	BR[0]#

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

Bus Signal	Agent 0 Pins	Agent 1 Pins
BREQ[0]#	BR[0]#	BR[1]#
BREQ[1]#	BR[1]#	BR[0]#
BREQ[2]#	Not Used	Not Used
BREQ[3]#	Not Used	Not Used

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

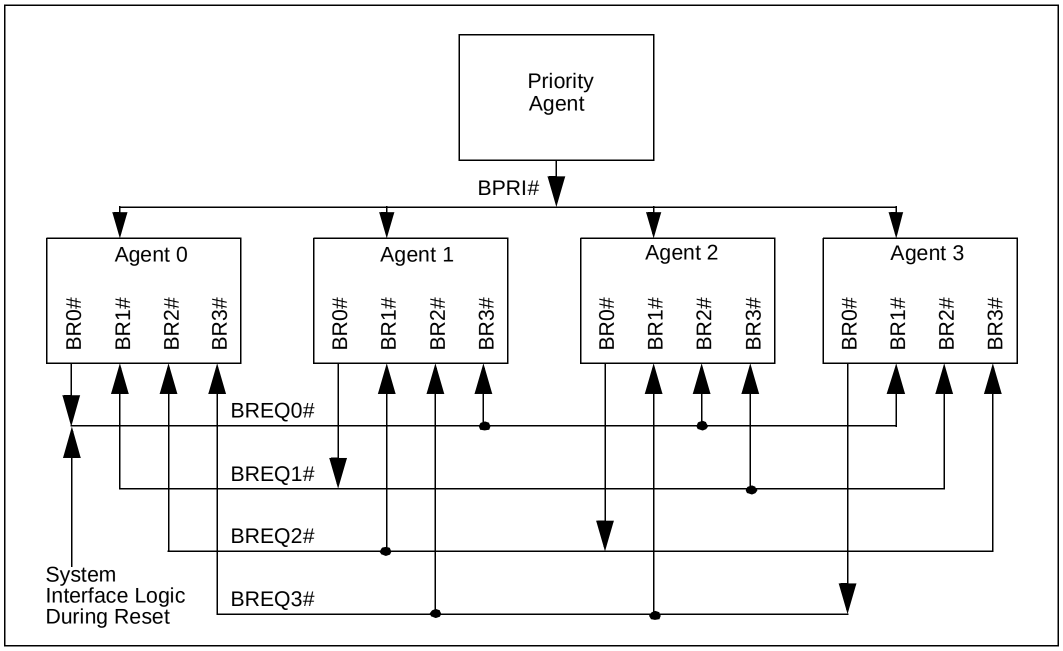
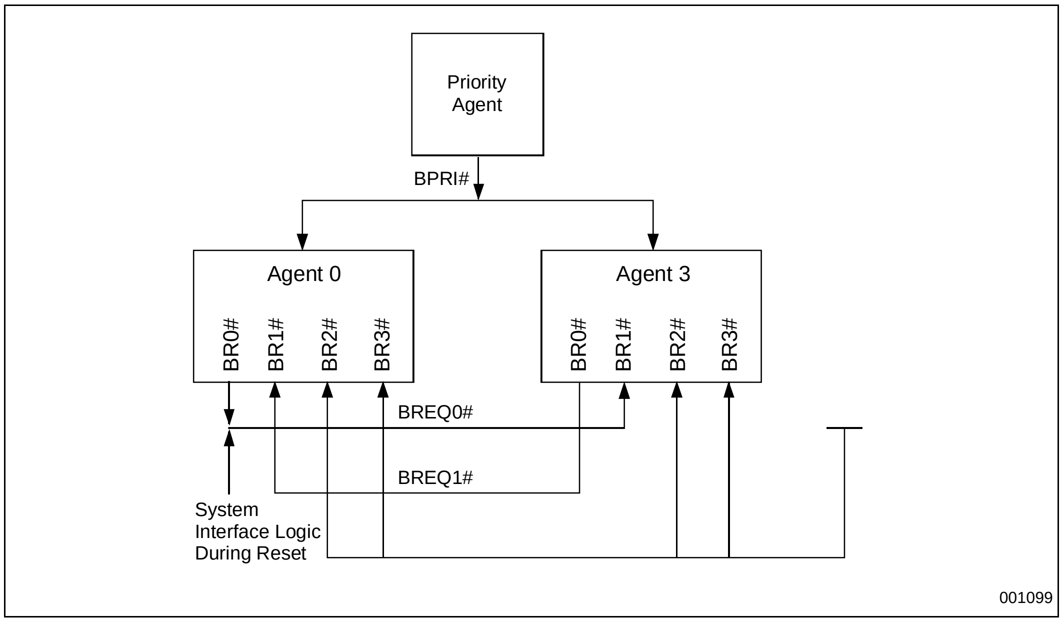


Figure 5-2. BR[3:0]# Physical Interconnection with Two Symmetric Agents



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 arbitration ID from the sampled value.

Each physical processor is a logical processor with a distinct arbitration ID and agent ID (refer to Table 5-4).

Table 5-4. Arbitration ID Configuration[1]

BR0#	BR1#	BR2#	BR3#	Arbitration ID	Agent ID Reported
L	H	H	H	0	0
H	H	H	L	1	2
H	H	L	H	2	4
H	L	H	H	3	6

1. L and H designate electrical levels.

5.2.12 Clock Frequency Ratios

Table 5-5 defines the system bus ratio configurations for the Itanium 2 processor.

Table 5-5. Itanium® 2 Processor System Bus to Core Frequency Multiplier Configuration

Ratio of Bus Frequency to Processor Frequency	A[21]#	A[20]#	A[19]#	A[18]#	A[17]#
2/9	1	0	1	1	0
2/10	1	0	1	0	1

5.3 Initialization Overview

The processor and PAL firmware initialize and test the processor on reset.

5.3.1 Initialization with RESET#

The Itanium 2 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 5-6 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 5-6. Itanium® 2 Processor Reset State (after PAL)

Processor Resource	Symbol	Value	Description
Instruction Pointer	IP	Refer to the *Intel® Itanium™ Architecture Software Developer's Manual* for details.	SALE_RESET entry point for the Itanium® 2 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 2 processor supports an INIT interrupt. 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 5-7 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 5-7. Itanium® Processor INIT State

Processor Resource	Symbol	Value	Description
Instruction Pointer	IP	Refer to the *Intel® Itanium™ Architecture Software Developer's Manual* for details.	PALE_INIT entry point for the Itanium® 2 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.

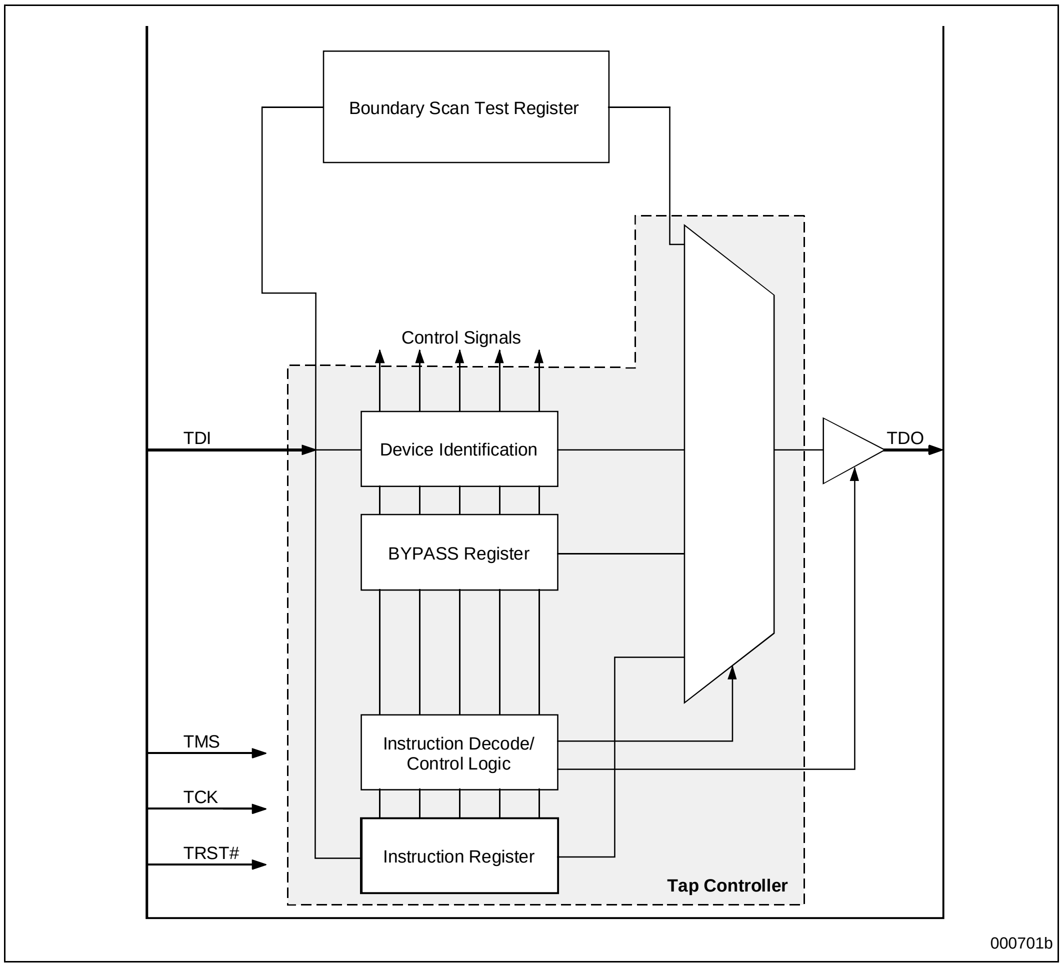
Test Access Port (TAP) 6

This chapter describes the implementation of the Itanium 2 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 other industry standard material on the subject.

A simplified block diagram of the TAP is shown in Figure 6-1. The Itanium 2 processor contains an integrated TAP controller, a Boundary Scan register, four input pins (TDI, TCK, TMS and TRST#) and one output pin (TDO). The integrated TAP controller consists of an Instruction Register, a Device ID Register, a Bypass Register and control logic.

For specific boundary scan chain information, please reference the *Intel® Itanium® 2 Processor Boundary Scan Description Language (BSDL) Model.*

Figure 6-1. Test Access Port Block Diagram



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.

6.1 Interface

The TAP scan chain is accessed serially through five dedicated pins on the processor package:

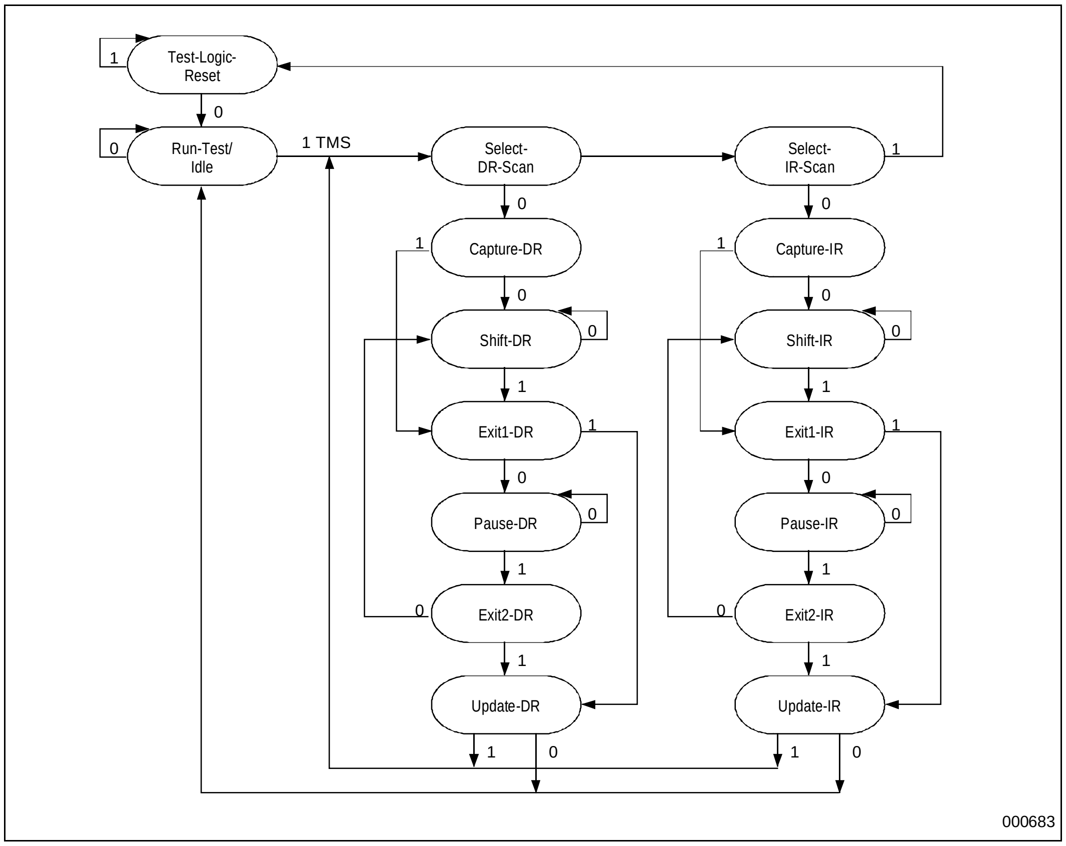
- **TCK**: The TAP clock signal.
- **TMS**: “Test Mode Select,” which controls the TAP finite state machine.
- **TDI**: “Test Data Input,” which inputs test instructions and data serially.
- **TRST#**: “Test Reset,” for TAP logic reset.
- **TDO**: “Test Data Output,” through which test output is read serially.

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

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-on. 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.

1. **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 2 processor. Data is transferred from TDI to TDO through the boundary scan register.

2. **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.

3. **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.

4. **Instruction Register**

   The instruction register contains a four-bit command field to indicate one of the following instructions: BYPASS, EXTEST, SAMPLE/PRELOAD, IDCODE, 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. Except for BYPASS, which is all 1s, all instructions as defined by the IEEE 1149.1 must have an instruction code of 0000 xxxx.

Table 6-1. Instructions for the Itanium® 2 Processor TAP Controller

Instructions	Encoding (Binary)	Encoding (Hex)
IEEE 1149.1 Standard		
BYPASS	1111 1111	FFh
EXTEST	0000 0000	00h
SAMPLE/PRELOAD	0000 0001	01h
Additional		
IDCODE	0000 0010	02h
HIGHZ	0000 1000	08h
CLAMP	0000 1011	0Bh

- **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.
- **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.

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.

Integration Tools 7

The Itanium 2 processor supports In-Target Probe (ITP) devices, and Logic Analyzer devices through a Logic Analyzer Interface (LAI), to allow monitoring of processor and system bus activity. Each device has its' own considerations for design and use.

7.1 In-Target Probe (ITP)

The Itanium 2 processor supports the ITP for program execution control, register/memory/IO access, and breakpoint control. This tool provides some of the functions commonly associated with debuggers and emulators. Use of an ITP will not affect high speed operations of the processor signals thereby allowing the system bus to maintain full operating speed.

Please refer to the *ITP700 Debug Port Design Guide* for more information on the ITP.

7.2 Logic Analyzer Interface (LAI)

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.

The Itanium 2 processor system bus can be monitored with logic analyzer equipment. Due to the complexity of Itanium 2 multiprocessor systems, the LAI is critical in providing the ability to probe and capture system bus signals for use in system debug and validation. There are two sets of considerations to keep in mind when designing an Itanium 2 processor-based system that can make use of an LAI: mechanical considerations and electrical considerations. Please consult your Logic Analyzer vendor for specific details.

Signals Reference A

This appendix provides an alphabetical listing of all Itanium 2 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® 2 Processor at 1.0 GHz and 900 MHz Datasheet*.

A.1 Alphabetical Signals Reference

A.1.1 A[49:3]# (I/O)

The Address (A[49:3]#) signals, with byte enables, define a 2^{50} Byte physical memory address space. When ADS# is active, these pins transmit the address of a transaction. These pins are also used to transmit other transaction related information such as transaction identifiers and external functions in the cycle following ADS# assertion. These signals must connect the appropriate pins of all agents on the Itanium 2 processor system bus. The A[49:27]# signals are parity-protected by the AP1# parity signal, and the A[26:3]# signals are parity-protected by the AP0# parity signal.

On the active-to-inactive transition of RESET#, the processors sample the A[49:3]# pins to determine their power-on configuration.

A.1.2 A20M# (I)

A20M# is ignored in the Itanium 2 processor system environment.

A.1.3 ADS# (I/O)

The Address Strobe (ADS#) signal is asserted to indicate the validity of the transaction address on the A[49:3]#, REQ[5:0]#, AP[1:0]# and RP#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 Address Parity (AP[1:0]#) signals can be driven by the request initiator along with ADS# and A[49:3]#. AP[1]# covers A[49:27]#, and AP[0]# covers A[26:3]#. 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[2:1]# 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	Reserved	Reserved
0	1	36-bit	0 to (64 GByte - 1)
1	0	50-bit	64 GByte to (1 Pbyte –1)
1	1	Reserved	Reserved

Any memory access transaction addressing a memory region that is less than 64 GB (i.e. Aa[49:36]# are all zeroes) must set ASZ[1:0]# to 01. Any memory access transaction addressing a memory region that is equal to or greater than 64 GB (i.e. Aa[49:36]# are not all zeroes) must set ASZ[1:0]# to 10. All observing bus agents that support the 64 GByte (36-bit) address space must respond to the transaction when ASZ[1:0]# equals 01. All observing bus agents that support larger than the 64 GByte (36-bit) address space must respond to the transaction when ASZ[1:0]# equals 01 or 10.

A.1.6 ATTR[3:0]# (I/O)

The ATTR[3:0]# signals are the attribute signals. They are driven by the request initiator during the second clock of the Request Phase on the Ab[35:32]# pins. The ATTR[3:0]# signals are valid for all transactions. The ATTR[3]# signal is reserved. 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]#	Description
000	Uncacheable
100	Write Coalescing
101	Write-Through
110	Write-Protect
111	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 on the signals that are using the common clock latched protocol.

BCLKp and BCLKn indirectly determine the internal clock frequency of the Itanium 2 processor. Each Itanium 2 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 on the Ab[15:8]# pins.

For memory or I/O transactions, the byte-enable signals indicate that valid data is requested or being transferred on the corresponding byte on the 128-bit data bus. BE[0]# indicates that the least significant byte is valid, and BE[7]# indicates that the most significant byte is valid. Since BE[7:0]# specifies the validity of only 8 bytes on the 16 byte wide bus, A[3]# is used to determine which half of the data bus is validated by BE[7:0]#.

For special transactions ((REQa[5:0]# = 001000B) 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
Flush (INVD)	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 except the Bus Invalidate Line (BIL) transaction.

A BIL transaction may return one cache line (128 bytes).

A.1.9 BERR# (I/O)

The Bus Error (BERR#) signal can be asserted to indicate a recoverable error with global MCA. 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 internal error.
- Asserted by any bus agent when it observes an error in a bus transaction.

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

BERR# is a wired-OR signal to allow multiple bus agents to drive it at the same time.

A.1.10 BINIT# (I/O)

If enabled by configuration, the Bus Initialization (BINIT#) signal is asserted to signal any bus condition that prevents reliable future operation.

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 priority agent. The priority agent must handle the error in a manner that is appropriate to the system architecture.

BINIT# is a wired-OR signal.

A.1.11 BNR# (I/O)

The Block Next Request (BNR#) 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 used for inserting breakpoints and for performance monitoring. They can be configured as outputs from the processor that indicate programmable counters used for monitoring performance, or inputs from the processor to indicate the status of breakpoints.

A.1.13 BPRI# (I)

The Bus Priority-agent Request (BPRI#) signal is used by the priority agent to arbitrate for ownership of the system bus. Observing BPRI# asserted 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 BR[0]# (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 and Table A-4 give the rotating interconnection between the processor and bus signals for both the 4P and 2P system bus topologies.

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

Bus Signal	Agent 0 Pins	Agent 1 Pins	Agent 2 Pins	Agent 3 Pins
BREQ[0]#	BR[0]#	BR[3]#	BR[2]#	BR[1]#
BREQ[1]#	BR[1]#	BR[0]#	BR[3]#	BR[2]#
BREQ[2]#	BR[2]#	BR[1]#	BR[0]#	BR[3]#
BREQ[3]#	BR[3]#	BR[2]#	BR[1]#	BR[0]#

Table A-5. BR0# (I/O), BR1#, BR2#, BR3# Signals for 2P Rotating Interconnect

Bus Signal	Agent 0 Pins	Agent 3 Pins
BREQ[0]#	BR[0]#	BR[1]#
BREQ[1]#	BR[1]#	BR[0]#
BREQ[2]#	Not Used	Not Used
BREQ[3]#	Not Used	Not Used

During power-on configuration, the priority agent must assert the BR[0]# 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-6.

Table A-6. BR[3:0]# Signals and Agent IDs

Pin Sampled Asserted on RESET#	Arbitration ID	Agent ID Reported
BR[0]#	0	0
BR[3]#	1	2
BR[2]#	2	4
BR[1]#	3	6

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 CCL# (I/O)

CCL# is the Cache Cleanse signal. It is driven on the second clock of the Request Phase on the EXF[2]#/Ab[5]# pin. CCL# is asserted for Memory Write transaction to indicate that a modified line in a processor may be written to memory without being invalidated in its caches.

A.1.17 CPUPRES# (O)

CPUPRES# can be used to detect the presence of a Itanium 2 processor in a socket. A ground indicates that a Itanium 2 processor is installed, while an open indicates that a Itanium 2 processor is not installed.

A.1.18 D[127:0]# (I/O)

The Data (D[127:0]#) signals provide a 128-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]# and A[3]#. Data signals that are not valid for a particular transfer must still have correct ECC (if data bus error checking is enabled). The data driver asserts DRDY# to indicate a valid data transfer.

A.1.19 D/C# (I/O)

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

A.1.20 DBSY# (I/O)

The Data Bus Busy (DBSY#) 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.

DBSY# is replicated three times to enable partitioning of the data paths in the system agents. This copy of the Data Bus Busy signal (DBSY#) is an input as well as an output.

A.1.21 DBSY_C1# (O)

DBSY# is a copy of the Data Bus Busy signal. This copy of the Data Bus Busy signal (DBSY_C1#) is an output only.

A.1.22 DBSY_C2# (O)

DBSY# is a copy of the Data Bus Busy signal. This copy of the Data Bus Busy signal (DBSY_C2#) is an output only.

A.1.23 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 priority agent.

A.1.24 DEN# (I/O)

The Defer Enable (DEN#) signal is driven on the bus on the second clock of the Request Phase on the Ab[4]# pin. DEN# is asserted to indicate that the transaction can be deferred by the responding agent.

A.1.25 DEP[15:0]# (I/O)

The Data Bus ECC Protection (DEP[15:0]#) signals provide optional ECC protection for Data Bus (D[127:0]#). They are driven by the agent responsible for driving D[127:0]#. During power-on configuration, bus agents 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.1.26 DHIT# (I)

The Deferred Hit (DHIT#) signal is driven during the Deferred Phase by the deferring agent. For read transactions on the bus DHIT# returns the final cache status that would have been indicated on HIT# for a transaction which was not deferred. DID[9:0]# (I/O)

DID[9:0]# are Deferred Identifier signals. The requesting agent transfers these signals by using A[25:16]#. They are transferred on Ab[25:16]# during the second clock of the Request Phase on all transactions, but Ab[20:16]# is only defined for deferrable transactions (DEN# asserted). DID[9:0]# is also transferred on Aa[25:16]# during the first clock of the Request Phase for Deferred Reply transactions.

The deferred identifier defines the token supplied by the requesting agent. DID[9]# and DID[8:5]# carry the agent identifiers of the requesting agents (always valid) and DID[4:0]# carry a transaction identifier associated with the request (valid only with DEN# asserted). This configuration limits the bus specification to 32 logical bus agents with each one of the bus agents capable of making up to 32 requests. Table A-7 shows the DID encodings.

Table A-7. DID[9:0]# Encoding

DID[9]#	DID[8:5]#	DID[4:0]#
Agent Type	Agent ID[3:0]	Transaction ID[4:0]

DID[9]# indicates the agent type. Symmetric agents use 0. Priority agents use 1. DID[8:5]# indicates the agent ID. Symmetric agents use their arbitration ID. DID[4:0]# indicates the transaction ID for an agent. The transaction ID must be unique for all deferrable transactions issued by an agent which have not reported their snoop results.

The Deferred Reply agent transmits the DID[9:0]# (Ab[25:16]#) signals received during the original transaction on the Aa[25:16]# signals during the Deferred Reply transaction. This process enables the original requesting agent to make an identifier match with the original request that is awaiting completion.

A.1.27 DPS# (I/O)

The Deferred Phase Enable (DPS#) signal is driven to the bus on the second clock of the Request Phase on the Ab[3]# 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.1.28 DRDY# (I/O)

The Data Ready (DRDY#) 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.

DRDY# is replicated three times to enable partitioning of data paths in the system agents. This copy of the Data Ready signal (DRDY#) is an input as well as an output.

A.1.29 DRDY_C1# (O)

DRDY# is a copy of the Data Ready signal. This copy of the Data Phase data-ready signal (DRDY_C1#) is an output only.

A.1.30 DRDY_C2# (O)

DRDY# is a copy of the Data Ready signal. This copy of the Data Phase data-ready signal (DRDY_C2#) is an output only.

A.1.31 DSZ[1:0]# (I/O)

The Data Size (DSZ[1:0]#) 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. For the Itanium 2 processor, DSZ# = 01, always.

A.1.32 EXF[4:0]# (I/O)

The Extended Function (EXF[4:0]#) signals are transferred on the A[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
EXF[4]#	Reserved	Reserved
EXF[3]#	SPLCK#/FCL#	Split Lock / Flush Cache Line
EXF[2]#	OWN#/CCL#	Memory Update Not Needed / Cache Cleanse
EXF[1]#	DEN#	Defer Enable
EXF[0]#	DPS#	Deferred Phase Supported

A.1.33 FCL# (I/O)

The Flush Cache Line (FCL#) signal is driven to the bus on the second clock of the Request Phase on the A[6]# pin. FCL# is asserted to indicate that the memory transaction is initiated by the global Flush Cache (FC) instruction.

A.1.34 FERR# (O)

The FERR# signal may be asserted to indicate an unmasked floating point error generated by an IA-32 application.

A.1.35 GSEQ# (I)

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

A.1.36 HIT# (I/O) and HITM# (I/O)

The Snoop Hit (HIT#) and Hit Modified (HITM#) 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.1.37 ID[9:0]# (I)

The Transaction ID (ID[9:0]#) signals are driven by the deferring agent. The signals in the two clocks are referenced IDa[9:0]# and IDb[9:0]#. During both clocks, ID[9:0]# signals are protected by the IP0# parity signal for the first clock, and by the IP[1]# parity signal on the second clock.

IDa[9:0]# returns the ID of the deferred transaction which was sent on Ab[25:16]# (DID[9:0]#).

A.1.38 IDS# (I)

The ID Strobe (IDS#) signal is asserted to indicate the validity of ID[9:0]# in that clock and the validity of DHIT# and IP[1:0]# in the next clock.

A.1.39 IGNNE# (I)

IGNNE# is ignored in the Itanium 2 processor system environment.

A.1.40 INIT# (I)

The Initialization (INIT#) 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.1.41 INT (I)

INT is the 8259-compatible 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. An interrupt acknowledge transaction is generated by the processor to obtain the interrupt vector from the interrupt controller.

The LINT[0] pin can be software configured to be used either as the INT signal or another local interrupt.

A.1.42 IP[1:0]# (I)

The ID Parity (IP[1:0]#) signals are driven on the second clock of the Deferred Phase by the deferring agent. IP0# protects the IDa[9:0]# and IDS# signals for the first clock, and IP[1]# protects the IDb[9:2, 0]# and IDS# signals on the second clock.

A.1.43 LEN[2:0]# (I/O)

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

Table A-9. Length of Data Transfers

LEN[2:0]#	Length
000	0 – 8 bytes
001	16 bytes
010	32 bytes
011	64 bytes
100	128 bytes
101	Reserved
110	Reserved
111	Reserved

A.1.44 LINT[1:0] (I)

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

A.1.45 LOCK# (I/O)

LOCK# is never asserted or sampled in the Itanium 2 processor system environment.

A.1.46 NMI (I)

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.1.47 OWN# (I/O)

The Guaranteed Cache Line Ownership (OWN#) signal is driven to the bus on the second clock of the Request Phase on the Ab[5]# pin. OWN# is asserted if cache line ownership is guaranteed. This allows a memory controller to ignore memory updates due to implicit writebacks.

A.1.48 PMI# (I)

The Platform Management Interrupt (PMI#) 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.1.49 PWRGOOD (I)

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

A.1.50 REQ[5:0]# (I/O)

The REQ[5:0]# 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[5:0]# signals define the transaction type to a level of detail that is sufficient to begin a snoop request. In the second clock, REQb[5:0]# signals carry additional information to define the complete transaction type. REQb[4:3]# signals transmit DSZ[1:0]# or the data transfer rate information of the requestor for transactions that involve data transfer. REQb[2:0]# signals transmit LEN[2:0]# (the data transfer length information). In both clocks, REQ[5:0]# and ADS# are protected by parity RP#.

All receiving agents observe the REQ[5:0]# 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[5:0]#						REQb[5:0]#					
	5	4	3	2	1	0	5	4	3	2	1	0
Deferred Reply	0	0	0	0	0	0	0	x	x	x	x	x
Reserved	0	0	0	0	0	1	0	x	x	x	x	x
Interrupt Acknowledge	0	0	1	0	0	0	0	DSZ[1:0]#		0	0	0
Special Transactions	0	0	1	0	0	0	0	DSZ[1:0]#		0	0	1
Reserved	0	0	1	0	0	0	0	DSZ[1:0]#		0	1	x
Reserved	0	0	1	0	0	1	0	DSZ[1:0]#		0	x	x
Interrupt	0	0	1	0	0	1	0	DSZ[1:0]#		1	0	0
Purge TC	0	0	1	0	0	1	0	DSZ[1:0]#		1	0	1
Reserved	0	0	1	0	0	1	0	DSZ[1:0]#		1	1	x
I/O Read	0	1	0	0	0	0	0	DSZ[1:0]#		x	x	x
I/O Write	0	1	0	0	0	1	0	DSZ[1:0]#		x	x	x
Reserved	0	1	1	0	0	x	0	DSZ[1:0]#		x	x	x
Memory Read & Invalidate	0	ASZ[1:0]#		0	1	0	0	DSZ[1:0]#		LEN[2:0]#		
Reserved	0	ASZ[1:0]#		0	1	1	0	DSZ[1:0]#		LEN[2:0]#		
Memory Read	0	ASZ[1:0]#		1	D/C#	0	0	DSZ[1:0]#		LEN[2:0]#		
Memory Read Current	1	ASZ[1:0]#		1	0	0	0	DSZ[1:0]#		LEN[2:0]#		
Reserved	1	ASZ[1:0]#		1	1	0	0	DSZ[1:0]#		LEN[2:0]#		
Memory Write	0	ASZ[1:0]#		1	WSNP#	1	0	DSZ[1:0]#		LEN[2:0]#		
Cache Line Replacement	1	ASZ[1:0]#		1	WSNP#	1	0	DSZ[1:0]#		0	0	0

A.1.51 RESET# (I)

Asserting the RESET# signal resets all processors to known states and invalidates all 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 begins program execution at the reset-vector.

A.1.52 RP# (I/O)

The Request Parity (RP#) signal is driven by the requesting agent, and provides parity protection for ADS# and REQ[5:0]#.

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.1.53 RS[2:0]# (I)

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

A.1.54 RSP# (I)

The Response Parity (RSP#) 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.1.55 SBSY# (I/O)

The Strobe Bus Busy (SBSY#) 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.

SBSY# is replicated three times to enable partitioning of data paths in the system agents. This copy of the Strobe Bus Busy signal (SBSY#) is an input as well as an output.

A.1.56 SBSY_C1# (O)

SBSY# is a copy of the Strobe Bus Busy signal. This copy of the Strobe Bus Busy signal (SBSY_C1#) is an output only.

A.1.57 SBSY_C2# (O)

SBSY# is a copy of the Strobe Bus Busy signal. This copy of the Strobe Bus Busy signal (SBSY_C2#) is an output only.

A.1.58 SPLCK# (I/O)

The Split Lock (SPLCK#) signal is driven in the second clock of the Request Phase on the Ab[6]# 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.1.59 STBn[7:0]# and STBp[7:0]# (I/O)

STBp[7:0]# and STBn[7:0]# (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. This functions like an independent double frequency clock constructed from a falling edge of either STBp[7:0]# or STBn[7:0]#. The data is synchronized by DRDY#. Each strobe pair is associated with 16 data bus signals and 2 ECC signals as shown in Table A-11.

Table A-11. STBp[7:0]# and STBn[7:0]# Associations

Strobe Bits	Data Bits	ECC Bits
STBp[7]#, STBn[7]#	D[127:112]#	DEP[15:14]#
STBp[6]#, STBn[6]#	D[111:96]#	DEP[13:12]#
STBp[5]#, STBn[5]#	D[95:80]#	DEP[11:10]#
STBp[4]#, STBn[4]#	D[79:64]#	DEP[9:8]#
STBp[3]#, STBn[3]#	D[63:48]#	DEP[7:6]#
STBp[2]#, STBn[2]#	D[47:32]#	DEP[5:4]#
STBp[1]#, STBn[1]#	D[31:16]#	DEP[3:2]#
STBp[0]#, STBn[0]#	D[15:0]#	DEP[1:0]#

A.1.60 TCK (I)

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

A.1.61 TDI (I)

The Test Data In (TDI) signal transfers serial test data into the Itanium 2 processor. TDI provides the serial input needed for IEEE 1149.1 compliant Test Access Port (TAP).

A.1.62 TDO (O)

The Test Data Out (TDO) signal transfers serial test data out from the Itanium 2 processor. TDO provides the serial output needed for IEEE 1149.1 compliant Test Access Port (TAP).

A.1.63 THRMTRIP# (O)

The Thermal Trip (THRMTRIP#) signal protects the Itanium 2 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 THRMTRIP# signal). Once THRMTRIP# is asserted, the platform must assert RESET# to protect the physical integrity of the processor.

A.1.64 THRMALERT# (O)

THRMALERT# 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.1.65 TMS (I)

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

A.1.66 TND# (I/O)

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

A.1.67 TRDY# (I)

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

A.1.68 TRST# (I)

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

A.1.69 WSNP# (I/O)

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

A.2 Signal Summaries

Table A-12 through Table A-15 list attributes of the Itanium 2 processor output, input, and I/O signals.

Table A-12. Output Signals

Name	Active Level	Clock	Signal Group
CPUPRES#	Low	—	Platform
DBSY_C1#	Low	BCLKp	Data
DBSY_C2#	Low	BCLKp	Data
DRDY_C1#	Low	BCLKp	Data
DRDY_C2#	Low	BCLKp	Data
FERR#	Low	Asynchronous	PC Compatibility

Table A-12. Output Signals (Continued)

Name	Active Level	Clock	Signal Group
SBSY_C1#	Low	BCLKp	Data
SBSY_C2#	Low	BCLKp	Data
TDO	High	TCK	TAP
THRMTRIP#	Low	Asynchronous	Error
THRMALERT#	Low	Asynchronous	Error

Table A-13. Input Signals

Name	Active Level	Clock	Signal Group	Qualified
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	System Bus	Request Phase (Mem Rd)
DEFER#	Low	BCLKp	Snoop	Snoop Phase
DHIT#	Low	BCLKp	System Bus	IDS#+1
GSEQ#	Low	BCLKp	Snoop	Snoop Phase
ID[9:0]#	Low	BCLKp	Defer	IDS#, IDS#+1
IDS#	Low	BCLKp	Defer	Always
INIT#	Low	Asynch	Exec Control	Always[1]
INT (LINT0)	High	Asynch	Exec Control	
IP[1:0]#	Low	BCLKp	System Bus	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
TRST#	Low	Asynch	Diagnostic	Always
TRDY#	Low	BCLKp	Response	Response 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
A[49:3]#	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	System Bus	ADS#
ATTR[3:0]#	Low	BCLKp	System Bus	ADS#+1
BE[7:0]#	Low	BCLKp	System Bus	ADS#+1
BR0#	Low	BCLKp	System Bus	Always
BPM[5:0]#	Low	BCLKp	Diagnostic	Always
CCL#	Low	BCLKp	System Bus	ADS#+1
D[127:0]#	Low	BCLKp	Data	DRDY#
DBSY#	Low	BCLKp	Data	Always
D/C#	Low	BCLKp	System Bus	ADS#
DEN#	Low	BCLKp	System Bus	ADS#+1
DEP[15:0]#	Low	BCLKp	System Bus	DRDY#
DID[9:0]#	Low	BCLKp	System Bus	ADS#+1
DRDY#	Low	BCLKp	Data	Always
DPS#	Low	BCLKp	System Bus	ADS#+1
DSZ[1:0]#	Low	BCLKp	System Bus	ADS#+1
EXF[4:0]#	Low	BCLKp	System Bus	ADS#+1
FCL#	Low	BCLKp	System Bus	ADS#+1
LEN[2:0]#	Low	BCLKp	System Bus	ADS#+1
LOCK#	Low	BCLKp	Arbitration	Always
OWN#	Low	BCLKp	System Bus	ADS#+1
REQ[5:0]#	Low	BCLKp	Request	ADS#, ADS#+1
RP#	Low	BCLKp	Request	ADS#, ADS#+1
SBSY#	Low	BCLKp	Data	Always
SPLCK#	Low	BCLKp	System Bus	ADS#+1
STBn[7:0]#	Low	—	Data	Always
STBp[7:0]#	Low	—	Data	Always
WSNP#	Low	BCLKp	System Bus	ADS#

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

Name	Active Level	Clock	Signal Group	Qualified
BNR#	Low	BCLKp	System Bus	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