



# P6 Family of Processors

Hardware Developer's Manual

---

*September 1998*



Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Pentium® II processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>

Copyright © Intel Corporation 1998.

\* Third-party brands and names are the property of their respective owners.

|          |   |     |
|----------|---|-----|
| <b>1</b> | <b>Introduction</b> .....   | 1-1 |
|          | 1.1 P6 FAMILY OF PROCESSORS OVERVIEW .....                              | 1-1 |
|          | 1.2 TERMINOLOGY .....   | 1-2 |
|          | 1.3 SPECIFIC PRODUCT REFERENCES .....                                   | 1-2 |
| <b>2</b> | <b>Micro-Architecture Overview</b> .....                                | 2-1 |
|          | 2.1 FULL CORE UTILIZATION .....   | 2-1 |
|          | 2.2 THE P6 FAMILY PROCESSOR PIPELINE .....                              | 2-2 |
|          | 2.2.1 The Fetch/Decode Unit .....                                       | 2-3 |
|          | 2.2.2 The Dispatch/Execute Unit .....                                   | 2-4 |
|          | 2.2.3 The Retire Unit .....   | 2-6 |
|          | 2.2.4 The Bus Interface Unit .....                                      | 2-6 |
|          | 2.3 ARCHITECTURE SUMMARY .....  | 2-7 |
| <b>3</b> | <b>System Bus Overview</b> .....  | 3-1 |
|          | 3.1 SIGNALING ON P6 FAMILY SYSTEM BUS .....                             | 3-1 |
|          | 3.2 SIGNAL OVERVIEW .....   | 3-2 |
|          | 3.2.1 Execution Control Signals .....                                   | 3-2 |
|          | 3.2.2 Arbitration Signals .....   | 3-3 |
|          | 3.2.3 Request Signals .....   | 3-4 |
|          | 3.2.4 Snoop Signals .....   | 3-4 |
|          | 3.2.5 Response Signals .....  | 3-5 |
|          | 3.2.6 Data Response Signals .....                                       | 3-6 |
|          | 3.2.6.1 LINE TRANSFERS .....  | 3-6 |
|          | 3.2.6.2 PART LINE ALIGNED TRANSFERS .....                               | 3-7 |
|          | 3.2.6.3 PARTIAL TRANSFERS .....   | 3-7 |
|          | 3.2.7 Error Signals .....   | 3-7 |
|          | 3.2.8 Compatibility Signals .....                                       | 3-9 |
|          | 3.2.9 Diagnostic Signals .....  | 3-9 |
| <b>4</b> | <b>Data Integrity</b> .....   | 4-1 |
|          | 4.1 ERROR CLASSIFICATION .....  | 4-1 |
|          | 4.2 P6 FAMILY PROCESSOR SYSTEM BUS DATA INTEGRITY<br>ARCHITECTURE ..... | 4-2 |
|          | 4.2.1 Bus Signals Protected Directly .....                              | 4-2 |
|          | 4.2.2 Bus Signals Protected Indirectly .....                            | 4-3 |
|          | 4.2.3 Unprotected Bus Signals .....                                     | 4-3 |
|          | 4.2.4 Hard-Error Response .....   | 4-3 |
|          | 4.2.5 P6 Family Processor System Bus Error Code Algorithms .....        | 4-3 |
|          | 4.2.5.1 PARITY ALGORITHM .....  | 4-3 |
|          | 4.2.5.2 P6 FAMILY SYSTEM BUS ECC ALGORITHM .....                        | 4-4 |
| <b>5</b> | <b>Configuration</b> .....  | 5-1 |
|          | 5.1 DESCRIPTION .....   | 5-1 |
|          | 5.1.1 Output Tristate .....   | 5-2 |
|          | 5.1.2 Built-in Self-test .....  | 5-2 |
|          | 5.1.3 Data Bus Error Checking Policy .....                              | 5-2 |

|          |  |            |
|----------|--|------------|
| 5.1.4    | Response Signal Parity Error Checking Policy .....       | 5-2        |
| 5.1.5    | AERR# Driving Policy .....                               | 5-3        |
| 5.1.6    | AERR# Observation Policy .....                           | 5-3        |
| 5.1.7    | BERR# Driving Policy for Initiator Bus Errors .....      | 5-3        |
| 5.1.8    | BERR# Driving Policy for Target Bus Errors.....          | 5-3        |
| 5.1.9    | BERR# Driving Policy for Initiator Internal Errors ..... | 5-3        |
| 5.1.10   | BINIT# Driving Policy .....                              | 5-3        |
| 5.1.11   | BINIT# Observation Policy.....                           | 5-3        |
| 5.1.12   | In-Order Queue Pipelining .....                          | 5-4        |
| 5.1.13   | Power-On Reset Vector .....                              | 5-4        |
| 5.1.14   | FFRC Mode Enable .....                                   | 5-4        |
| 5.1.15   | APIC Mode .....  | 5-4        |
| 5.1.16   | APIC Cluster ID .....                                    | 5-4        |
| 5.1.17   | Symmetric Agent Arbitration ID.....                      | 5-4        |
| 5.1.18   | Low Power Standby Enable.....                            | 5-7        |
| 5.2      | CLOCK FREQUENCIES AND RATIOS .....                       | 5-8        |
| 5.3      | POWER-ON CONFIGURATION REGISTER .....                    | 5-8        |
| 5.4      | INITIALIZATION PROCESS .....                             | 5-10       |
| <b>6</b> | <b>Test Access Port (TAP).....</b>                       | <b>6-1</b> |
| 6.1      | INTERFACE .....  | 6-1        |
| 6.2      | ACCESSING THE TAP LOGIC .....                            | 6-2        |
| 6.2.1    | Accessing the Instruction Register.....                  | 6-4        |
| 6.2.2    | Accessing the Data Registers.....                        | 6-5        |
| 6.3      | INSTRUCTION SET .....                                    | 6-6        |
| 6.4      | DATA REGISTER SUMMARY .....                              | 6-7        |
| 6.4.1    | Bypass Register.....                                     | 6-7        |
| 6.4.2    | Device ID Register.....                                  | 6-7        |
| 6.4.3    | BIST Result Boundary Scan Register.....                  | 6-8        |
| 6.4.4    | Boundary Scan Register.....                              | 6-8        |
| 6.5      | RESET BEHAVIOR .....                                     | 6-8        |
| <b>7</b> | <b>Integration Tools.....</b>                            | <b>7-1</b> |
| 7.1      | IN-TARGET PROBE (ITP) FOR P6 FAMILY PROCESSORS .....     | 7-1        |
| 7.1.1    | Primary Function.....                                    | 7-1        |
| 7.1.2    | Debug Port Connector Description .....                   | 7-2        |
| 7.1.3    | Debug Port Signal Descriptions .....                     | 7-2        |
| 7.1.4    | Debug Port Signal Notes .....                            | 7-2        |
| 7.1.4.1  | SIGNAL NOTE 1: DBRESET#.....                             | 7-3        |
| 7.1.4.2  | SIGNAL NOTE 5: TDO AND TDI.....                          | 7-3        |
| 7.1.4.3  | SIGNAL NOTE 7: TCK.....                                  | 7-6        |
| 7.1.5    | Debug Port Layout.....                                   | 7-6        |
| 7.1.5.1  | SIGNAL QUALITY NOTES .....                               | 7-8        |
| 7.1.5.2  | DEBUG PORT CONNECTOR .....                               | 7-8        |
| 7.1.6    | Using Boundary Scan to Communicate to the Processor..... | 7-8        |
| <b>A</b> | <b>Signals Reference .....</b>                           | <b>A-1</b> |
| A.1      | ALPHABETICAL SIGNALS LISTING .....                       | A-1        |
| A.1.1    | A[35:3]# (I/O) .....                                     | A-1        |
| A.1.2    | A20M# (I) .....  | A-1        |
| A.1.3    | ADS# (I/O) .....   | A-1        |



|        |                              |      |
|--------|------------------------------|------|
| A.1.4  | AERR# (I/O).....             | A-2  |
| A.1.5  | AP[1:0]# (I/O).....          | A-2  |
| A.1.6  | ASZ[1:0]# (I/O).....         | A-2  |
| A.1.7  | ATTR[7:0]# (I/O).....        | A-2  |
| A.1.8  | BCLK (I).....                | A-3  |
| A.1.9  | BE[7:0]# (I/O).....          | A-3  |
| A.1.10 | BERR# (I/O).....             | A-3  |
| A.1.11 | BINIT# (I/O).....            | A-4  |
| A.1.12 | BNR# (I/O).....              | A-4  |
| A.1.13 | BP[3:2]# (I/O).....          | A-4  |
| A.1.14 | BPM[1:0]# (I/O).....         | A-4  |
| A.1.15 | BPRI# (I).....               | A-4  |
| A.1.16 | BREQ[3:0]# (I/O).....        | A-5  |
| A.1.17 | D[63:0]# (I/O).....          | A-5  |
| A.1.18 | DBSY# (I/O).....             | A-6  |
| A.1.19 | DEFER# (I).....              | A-6  |
| A.1.20 | DEN# (I/O).....              | A-6  |
| A.1.21 | DEP[7:0]# (I/O).....         | A-6  |
| A.1.22 | DID[7:0]# (I/O).....         | A-6  |
| A.1.23 | DRDY# (I/O).....             | A-7  |
| A.1.24 | DSZ[1:0]# (I/O).....         | A-7  |
| A.1.25 | EXF[4:0]# (I/O).....         | A-7  |
| A.1.26 | FERR# (O).....               | A-7  |
| A.1.27 | FLUSH# (I).....              | A-8  |
| A.1.28 | FRCERR (I/O).....            | A-8  |
| A.1.29 | HIT# (I/O), HITM# (I/O)..... | A-8  |
| A.1.30 | IERR# (O).....               | A-8  |
| A.1.31 | IGNNE# (I).....              | A-9  |
| A.1.32 | INIT# (I).....               | A-9  |
| A.1.33 | INTR(I).....                 | A-9  |
| A.1.34 | LEN[1:0]# (I/O).....         | A-9  |
| A.1.35 | LINT[1:0] (I).....           | A-10 |
| A.1.36 | LOCK# (I/O).....             | A-10 |
| A.1.37 | NMI(I).....                  | A-11 |
| A.1.38 | PICCLK (I).....              | A-11 |
| A.1.39 | PICD[1:0] (I/O).....         | A-11 |
| A.1.40 | PRDY# (O).....               | A-11 |
| A.1.41 | PREQ# (I).....               | A-11 |
| A.1.42 | PWRGOOD (I).....             | A-11 |
| A.1.43 | REQ[4:0]# (I/O).....         | A-12 |
| A.1.44 | RESET# (I).....              | A-13 |
| A.1.45 | RP# (I/O).....               | A-13 |
| A.1.46 | RS[2:0]# (I).....            | A-13 |
| A.1.47 | RSP# (I).....                | A-14 |
| A.1.48 | SLP# (I).....                | A-14 |
| A.1.49 | SMI# (I).....                | A-14 |
| A.1.50 | SMMEM# (I/O).....            | A-14 |
| A.1.51 | SPLCK# (I/O).....            | A-14 |
| A.1.52 | STPCLK# (I).....             | A-14 |
| A.1.53 | TCK (I).....                 | A-15 |
| A.1.54 | TDI (I).....                 | A-15 |

|              |                       |                |
|--------------|-----------------------|----------------|
| A.1.55       | TDO (O) .....         | A-15           |
| A.1.56       | THERMTRIP# (O).....   | A-15           |
| A.1.57       | TMS (I).....          | A-15           |
| A.1.58       | TRDY# (I).....        | A-15           |
| A.1.59       | TRST# (I) .....       | A-15           |
| A.2          | SIGNAL SUMMARIES..... | A-16           |
| <b>Index</b> | .....                 | <b>INDEX-1</b> |

## Figures

|     |  |     |
|-----|--|-----|
| 2-1 | Three Engines Communicating Using an Instruction Pool .....                | 2-1 |
| 2-2 | A Typical Pseudo Code Fragment.....  | 2-1 |
| 2-3 | The Three Core Engines Interface with Memory via Unified Caches .....      | 2-3 |
| 2-4 | Inside the Fetch/Decode Unit .....   | 2-4 |
| 2-5 | Inside the Dispatch/Execute Unit.....                                      | 2-5 |
| 2-6 | Inside the Retire Unit .....   | 2-6 |
| 2-7 | Inside the Bus Interface Unit.....   | 2-7 |
| 3-1 | Latched Bus Protocol.....  | 3-1 |
| 5-1 | Hardware Configuration Signal Sampling.....                                | 5-1 |
| 5-2 | BR[1:0]# Physical Interconnection with Two Symmetric Agents .....          | 5-5 |
| 5-3 | BR[1:0]# Physical Interconnection with Four Symmetric Agents .....         | 5-6 |
| 6-1 | Simplified Block Diagram of Processor TAP Logic .....                      | 6-2 |
| 6-2 | TAP Controller Finite State Machine .....                                  | 6-3 |
| 6-3 | Processor TAP Instruction Register.....                                    | 6-4 |
| 6-4 | Operation of the Processor TAP Instruction Register .....                  | 6-5 |
| 6-5 | TAP Instruction Register Access .....                                      | 6-5 |
| 7-1 | Hardware Components of the ITP .....                                       | 7-2 |
| 7-2 | GTL+ Signal Termination.....   | 7-3 |
| 7-3 | Generic DP System Layout for Debug Port Connection .....                   | 7-7 |
| 7-4 | Debug Port Connector on Thermal Plate Site of Circuit Board.....           | 7-8 |
| 7-5 | Hole Positioning for Connector on Thermal Plate Side of Circuit Board..... | 7-8 |
| 7-6 | Processor System Where Boundary Scan Is Not Used.....                      | 7-9 |

## Tables

|      |   |      |
|------|---|------|
| 3-1  | Execution Control Signals.....  | 3-2  |
| 3-2  | Arbitration Signals.....  | 3-3  |
| 3-3  | Request Signals.....  | 3-4  |
| 3-4  | Snoop Signals.....  | 3-5  |
| 3-5  | Response Signals.....   | 3-5  |
| 3-6  | Data Phase Signals .....  | 3-6  |
| 3-7  | Burst Order Used for P6 Family Processor Bus Line Transfers .....             | 3-6  |
| 3-8  | Error Signals .....   | 3-7  |
| 3-9  | PC Compatibility Signals .....  | 3-9  |
| 3-10 | Diagnostic Support Signals.....   | 3-10 |
| 4-1  | Direct Bus Signal Protection .....  | 4-2  |
| 5-1  | APIC Cluster ID Configuration .....   | 5-4  |
| 5-2  | P6 Family Processor Bus BREQ[1:0]# Interconnect<br>(2-Way MP Processors)..... | 5-5  |



|      |   |      |
|------|---|------|
| 5-3  | P6 Family Processor Bus BREQ[3:0]# Interconnect<br>(4-Way MP Processors).....           | 5-6  |
| 5-4  | Arbitration ID Configuration (Two Agents) .....   | 5-7  |
| 5-5  | Arbitration ID Configuration (Four Agents).....   | 5-7  |
| 5-6  | System Bus To Core Frequency Multiplier Configuration .....                             | 5-8  |
| 5-7  | Processor Power-On Configuration Register .....   | 5-9  |
| 5-8  | Power-On Configuration Register APIC Cluster ID Bit Field.....                          | 5-9  |
| 5-9  | Power-On Configuration Register Arbitration ID Configuration.....                       | 5-10 |
| 5-10 | Power-On Configuration Register Bus Frequency to Core Frequency<br>Ratio Bit Field..... | 5-10 |
| 6-1  | 1149.1 Instructions in the Processor TAP.....   | 6-6  |
| 6-2  | TAP Data Registers .....  | 6-7  |
| 6-3  | Device ID Register .....  | 6-7  |
| 6-4  | TAP Reset Actions.....  | 6-8  |
| 7-1  | Debug Port Pinout Description and Requirements .....                                    | 7-4  |
| A-6  | LEN[1:0]# Signals Data Transfer Lengths .....   | A-10 |



## 1.1 P6 FAMILY OF PROCESSORS OVERVIEW

The P6 family of processors is the generation of processors that succeeds the Pentium® line of Intel processors. This processor family implements Intel's dynamic execution microarchitecture, which incorporates a unique combination of multiple branch prediction, data flow analysis, and speculative execution. This enables P6 family processors to deliver higher performance than the Pentium family of processors, while maintaining binary compatibility with all previous Intel Architecture processors.

The first processor designed from the P6 family was the Pentium Pro processor which was followed by the Pentium II processor. As new products are designed, new technologies are utilized. For example, features were added to some P6 family processor products to aid in the design of energy efficient computer systems by offering multiple low-power states such as AutoHALT, Stop-Grant, Sleep and Deep Sleep, to conserve power during idle times.

The targeting of specific markets is another differentiator of products belonging to the P6 family including the Server and Workstation Market, Performance PC Market, Mobile Market and the Basic PC Market. All of these market segments demand specific features and performance. While all P6 family products have the benefits of Intel's dynamic execution microarchitecture, there are also product specific differentiators.

For example, the P6 family offers products with larger cache sizes and support for up to four processors to meet the higher performance demand of the server and workstation markets. Additionally, the Pentium II Xeon™ processor provides manageability requirements of the server and workstation environment by incorporating a System Management Bus (SMBus) interface. This interface can be used in conjunction with system hardware and software to provide more manageability options than any previous P6 family product. Memory is cacheable for 64 GB of addressable memory. This SMBus interface and larger L2 cache sizes, enables these products to provide higher performance and manageability for the server and workstation environment.

For high end desktop and business applications, the Pentium II processor can deliver the necessary computing power. Memory is cacheable for up to 4 GB of addressable memory space, allowing significant headroom for applications. It also incorporates Intel's MMX™ technology for enhanced media and communications performance.

The Intel P6 family also contains processors which are specifically designed and manufactured for the mobile market. These processors can operate under much more restrictive power and size constraints than the previously mentioned products, while still maintaining a high level of performance.

The Intel Celeron™ processor is designed for Basic PC desktops. It provides the same benefits of the P6 family architecture and adds the capabilities of Intel's MMX technology to bring a balanced level of performance and price to Basic PC consumers.

## 1.2 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 FLUSH# is low, a flush has been requested. When NMI is high, a non-maskable interrupt has occurred. In the case of signals where the name does not imply an active state but describes part of a binary sequence (such as address or data), the '#' symbol implies that the signal is inverted. For example, D[3:0] = 'HLHL' refers to a hex 'A', and D#[3:0] = 'LHLH' also refers to a hex 'A' (H= High logic level, L= Low logic level).

The term "system bus" refers to the interface between the processor, system core logic (a.k.a. the core logic components) and other bus agents. The system bus is a multiprocessing interface to processors, memory and I/O. The term "cache bus" refers to the interface between the processor and the L2 cache components. The cache bus does NOT connect to the system bus, and is not visible to other agents on the system bus.

When signal values are referenced in tables, a 0 indicates inactive and a 1 indicates active. 0 and 1 **do not** reflect voltage levels. A # after a signal name indicates active low. An entry of 1 for ADS# means that ADS# is active, with a low voltage level.

## 1.3 SPECIFIC PRODUCT REFERENCES

The reader of this document should also reference product datasheet specific details. Datasheets for Intel processors are located at <http://developer.intel.com>.

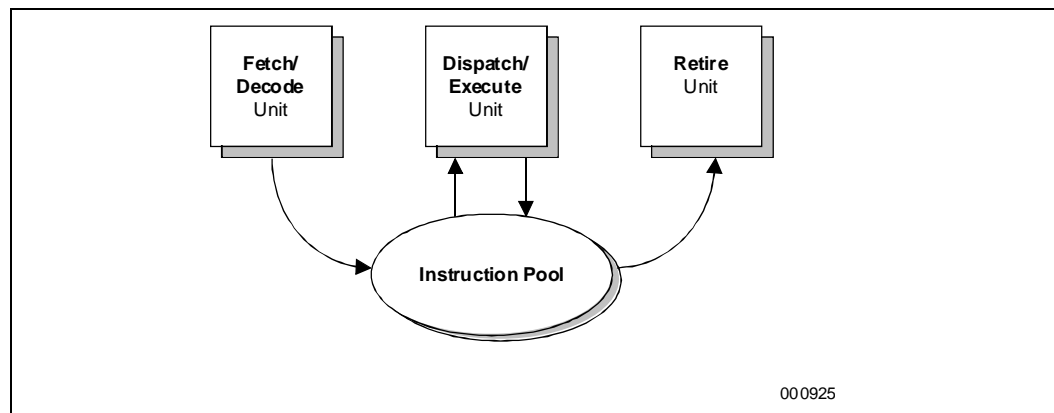
The task of providing all of the technical details of each product in one comprehensive manual is not the goal of this reference manual. The goal of this manual is to provide a reference of commonality between all P6 family products. It is the role each processor's datasheet to provide the specific differentiating details of each product. In the event that the datasheet and this reference manual contradict one another, please use the datasheet as the correct reference.

The P6 family of processor's may contain design defects known as errata. All characterized errata are available on-line at <http://developer.intel.com>.

The P6 family of processors use a dynamic execution micro-architecture. This three-way superscalar, pipelined micro-architecture features a decoupled, multi-stage superpipeline, which trades less work per pipestage for more stages. A P6 family processor, for example, has twelve stages with a pipestage time 33 percent less than the Pentium processor, which helps achieve a higher clock rate on any given manufacturing process.

The approach used in the P6 family micro-architecture removes the constraint of linear instruction sequencing between the traditional “fetch” and “execute” phases, and opens up a wide instruction window using an instruction pool. This approach allows the “execute” phase of the processor to have much more visibility into the program instruction stream so that better scheduling may take place. It requires the instruction “fetch/decode” phase of the processor to be much more efficient in terms of predicting program flow. Optimized scheduling requires the fundamental “execute” phase to be replaced by decoupled “dispatch/execute” and “retire” phases. This allows instructions to be started in any order but always be completed in the original program order. Processors in the P6 family may be thought of as three independent engines coupled with an instruction pool as shown in Figure 2-1.

**Figure 2-1. Three Engines Communicating Using an Instruction Pool**



## 2.1 FULL CORE UTILIZATION

The three independent-engine approach was taken to more fully utilize the processor core. Consider the pseudo code fragment in Figure 2-2:

**Figure 2-2. A Typical Pseudo Code Fragment**

```

    r1 <= mem [r0] /* Instruction 1 */
    r2 <= r1 + r2 /* Instruction 2 */
    r5 <= r5 + 1 /* Instruction 3 */
    r6 <= r6 - r3 /* Instruction 4 */
  
```

000922

The first instruction in this example is a load of r1 that, at run time, causes a cache miss. A traditional processor core must wait for its bus interface unit to read this data from main memory and return it before moving on to instruction 2. This processor stalls while waiting for this data and is thus being under-utilized.

To avoid this memory latency problem, a P6 family processor “looks-ahead” into the instruction pool at subsequent instructions and does useful work rather than stalling. In the example in [Figure 2-2](#), instruction 2 is not executable since it depends upon the result of instruction 1; however both instructions 3 and 4 have no prior dependencies and are therefore executable. The processor executes instructions 3 and 4 out-of-order. The results of this out-of-order execution cannot be committed to permanent machine state (i.e., the programmer-visible registers) immediately since the original program order must be maintained. The results are instead stored back in the instruction pool awaiting in-order retirement. The core executes instructions depending upon their readiness to execute, and not on their original program order, and is therefore a true dataflow engine. This approach has the side effect that instructions are typically executed out-of-order.

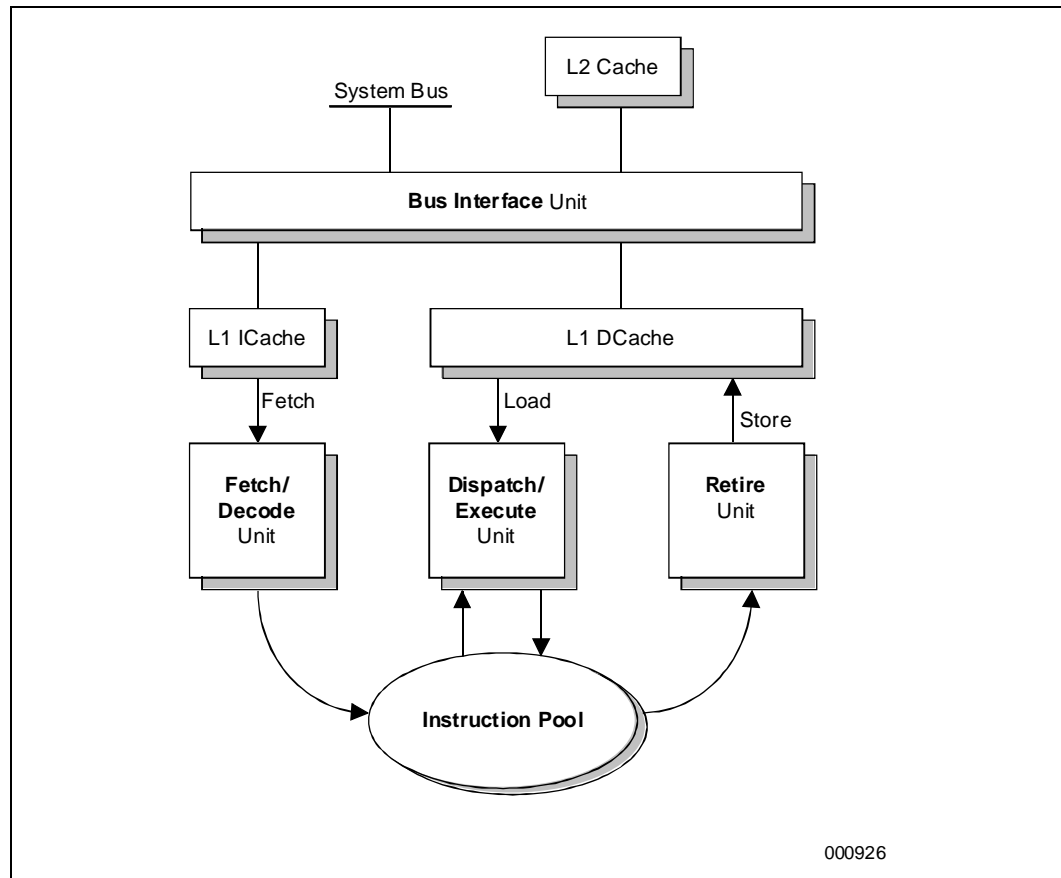
The cache miss on instruction 1 will take many internal clocks, so the core continues to look ahead for other instructions that could be speculatively executed, and is typically looking 20 to 30 instructions in front of the instruction pointer. Within this 20 to 30 instruction window there will be, on average, five branches that the fetch/decode unit must correctly predict if the dispatch/execute unit is to do useful work. The sparse register set of an Intel Architecture (IA) processor will create many false dependencies on registers so the dispatch/execute unit will rename the Intel Architecture registers into a larger register set to enable additional forward progress. The Retire Unit owns the programmer’s Intel Architecture register set and results are only committed to permanent machine state in these registers when it removes completed instructions from the pool in original program order.

Dynamic Execution technology can be summarized as optimally adjusting instruction execution by predicting program flow, having the ability to speculatively execute instructions in any order, and then analyzing the program’s dataflow graph to choose the best order to execute the instructions.

## 2.2 THE P6 FAMILY PROCESSOR PIPELINE

In order to get a closer look at how the P6 family micro-architecture implements Dynamic Execution, [Figure 2-3](#) shows a block diagram of the P6 family of processor products including cache and memory interfaces. The “Units” shown in [Figure 2-3](#) represent stages of the P6 family of processors pipeline.

**Figure 2-3. The Three Core Engines Interface with Memory via Unified Caches**

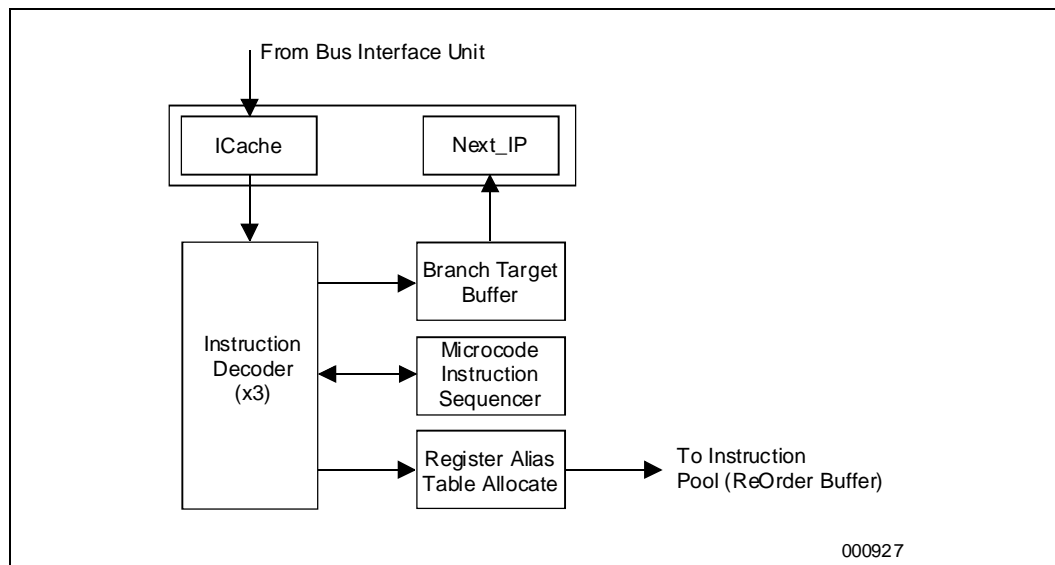


- The **FETCH/DECODE** unit: An in-order unit that takes as input the user program instruction stream from the instruction cache, and decodes them into a series of  $\mu$ operations ( $\mu$ ops) that represent the dataflow of that instruction stream. The pre-fetch is speculative.
- The **DISPATCH/EXECUTE** unit: An out-of-order unit that accepts the dataflow stream, schedules execution of the  $\mu$ ops subject to data dependencies and resource availability and temporarily stores the results of these speculative executions.
- The **RETIRE** unit: An in-order unit that knows how and when to commit (“retire”) the temporary, speculative results to permanent architectural state.
- The **BUS INTERFACE** unit: A partially ordered unit responsible for connecting the three internal units to the real world. The bus interface unit communicates directly with the L2 (second level) cache supporting up to four concurrent cache accesses. The bus interface unit also controls a transaction bus, with MESI snooping protocol, to system memory.

## 2.2.1 The Fetch/Decode Unit

Figure 2-4 shows a more detailed view of the Fetch/Decode unit.

Figure 2-4. Inside the Fetch/Decode Unit



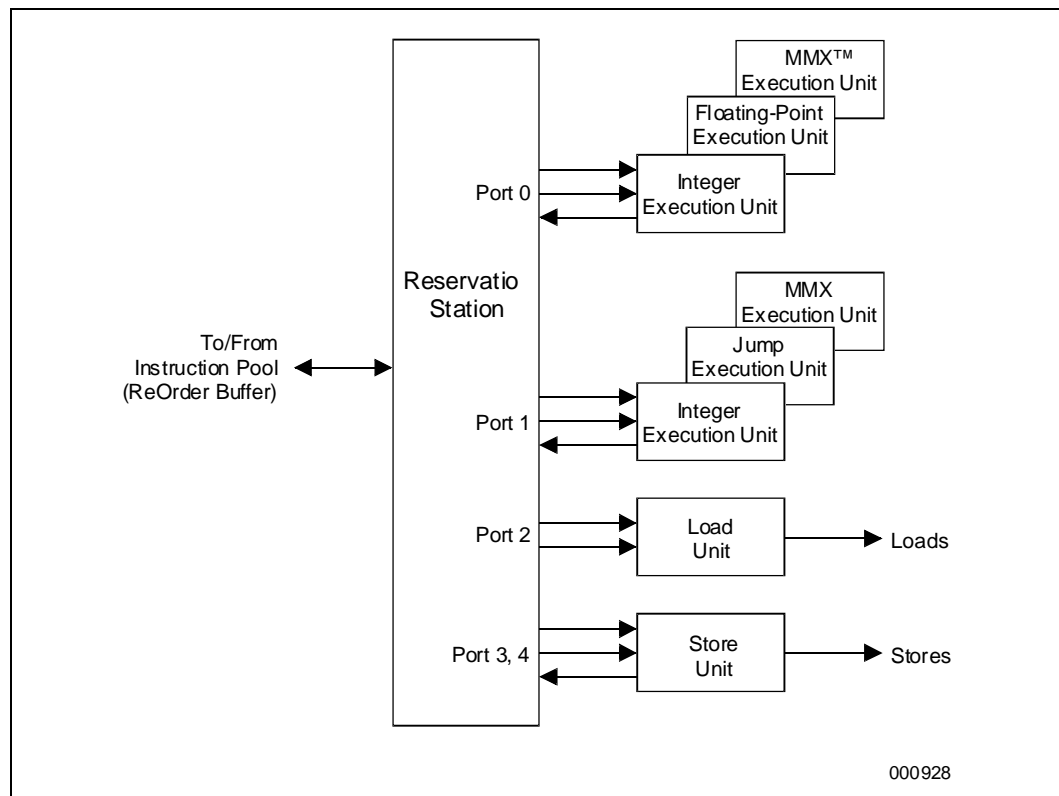
The L1 Instruction Cache is a local instruction cache. The Next\_IP unit provides the L1 Instruction Cache index, based on inputs from the Branch Target Buffer (BTB), trap/interrupt status, and branch-misprediction indications from the integer execution section.

The L1 Instruction Cache fetches the cache line corresponding to the index from the Next\_IP, and the next line, and presents 16 aligned bytes to the decoder. The prefetched bytes are rotated so that they are justified for the instruction decoders (ID). The beginning and end of the Intel Architecture instructions are marked.

Three parallel decoders accept this stream of marked bytes, and proceed to find and decode the Intel Architecture instructions contained therein. The decoder converts the Intel Architecture instructions into triadic  $\mu$ ops (two logical sources, one logical destination per  $\mu$ op). Most Intel Architecture instructions are converted directly into single  $\mu$ ops, some instructions are decoded into one-to-four  $\mu$ ops and the complex instructions require microcode (the box labeled Microcode Instruction Sequencer in Figure 2-4). This microcode is just a set of preprogrammed sequences of normal  $\mu$ ops. The  $\mu$ ops are queued, and sent to the Register Alias Table (RAT) unit, where the logical Intel Architecture-based register references are converted into references to physical registers in P6 family processors physical register references, and to the Allocator stage, which adds status information to the  $\mu$ ops and enters them into the instruction pool. The instruction pool is implemented as an array of Content Addressable Memory called the ReOrder Buffer (ROB).

## 2.2.2 The Dispatch/Execute Unit

The Dispatch unit selects  $\mu$ ops from the instruction pool depending upon their status. If the status indicates that a  $\mu$ op has all of its operands then the dispatch unit checks to see if the execution resource needed by that  $\mu$ op is also available. If both are true, the Reservation Station removes that  $\mu$ op and sends it to the resource where it is executed. The results of the  $\mu$ op are later returned to the pool. There are five ports on the Reservation Station, and the multiple resources are accessed as shown in Figure 2-5.

**Figure 2-5. Inside the Dispatch/Execute Unit**


The P6 family of processors can schedule at a peak rate of 5  $\mu$ ops per clock, one to each resource port, but a sustained rate of 3  $\mu$ ops per clock is more typical. The activity of this scheduling process is the out-of-order process;  $\mu$ ops are dispatched to the execution resources strictly according to dataflow constraints and resource availability, without regard to the original ordering of the program.

Note that the actual algorithm employed by this execution-scheduling process is vitally important to performance. If only one  $\mu$ op per resource becomes data-ready per clock cycle, then there is no choice. But if several are available, it must choose. The P6 family micro-architecture uses a pseudo FIFO scheduling algorithm favoring back-to-back  $\mu$ ops.

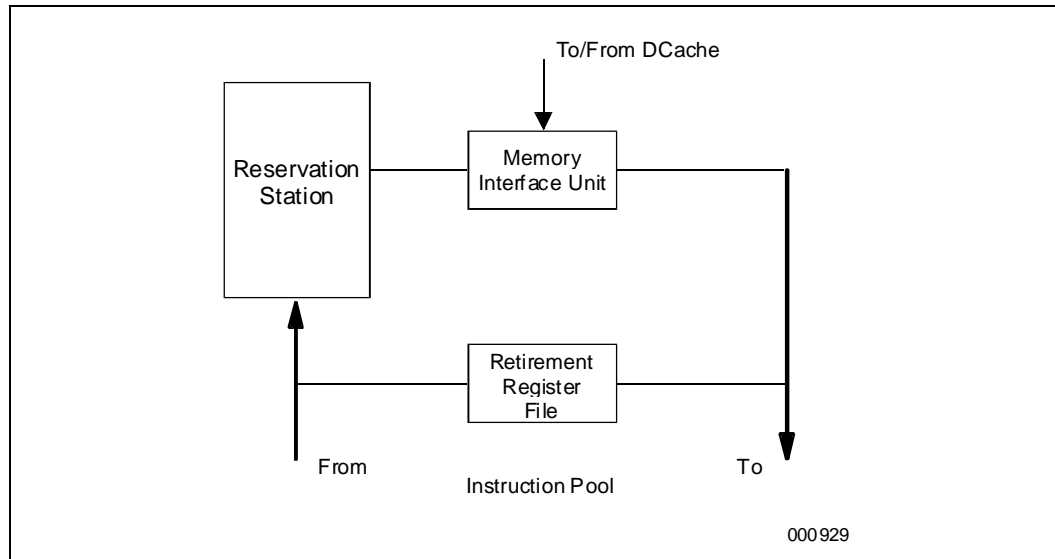
Note that many of the  $\mu$ ops are branches. The Branch Target Buffer (BTB) will correctly predict most of these branches but it can't correctly predict them all. Consider a BTB that is correctly predicting the backward branch at the bottom of a loop; eventually that loop is going to terminate, and when it does, that branch will be mispredicted. Branch  $\mu$ ops are tagged (in the in-order pipeline) with their fall-through address and the destination that was predicted for them. When the branch executes, what the branch actually did is compared against what the prediction hardware said it would do. If those coincide, then the branch eventually retires and the speculatively executed work between it and the next branch instruction in the instruction pool is good.

But if they do not coincide, then the Jump Execution Unit (JEU) changes the status of all of the  $\mu$ ops behind the branch to remove them from the instruction pool. In that case the proper branch destination is provided to the BTB which restarts the whole pipeline from the new target address.

## 2.2.3 The Retire Unit

Figure 2-6 shows a more detailed view of the Retire Unit.

**Figure 2-6. Inside the Retire Unit**



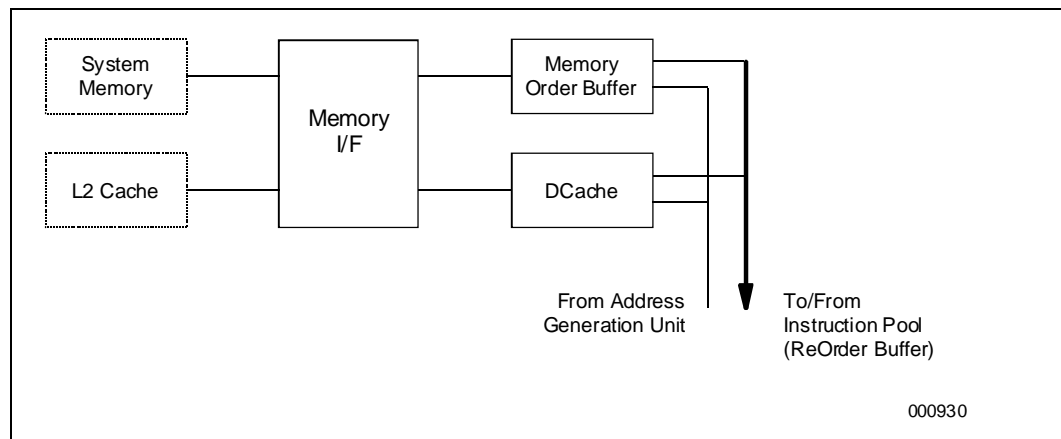
The Retire Unit is also checking the status of  $\mu$ ops in the instruction pool. It is looking for  $\mu$ ops that have executed and can be removed from the pool. Once removed, the original architectural target of the  $\mu$ ops is written as per the original Intel Architecture instruction. The Retire Unit must not only notice which  $\mu$ ops are complete, it must also re-impose the original program order on them. It must also do this in the face of interrupts, traps, faults, breakpoints and mispredictions.

The Retire Unit must first read the instruction pool to find the potential candidates for retirement and determine which of these candidates are next in the original program order. Then it writes the results of this cycle's retirements to the Retirement Register File (RRF). The Retire Unit is capable of retiring 3  $\mu$ ops per clock.

## 2.2.4 The Bus Interface Unit

Figure 2-7 shows a more detailed view of the Bus Interface Unit.

**Figure 2-7. Inside the Bus Interface Unit**



There are two types of memory access: loads and stores. Loads only need to specify the memory address to be accessed, the width of the data being retrieved, and the destination register. Loads are encoded into a single  $\mu$ op.

Stores need to provide a memory address, a data width, and the data to be written. Stores therefore require two  $\mu$ ops, one to generate the address and one to generate the data. These  $\mu$ ops must later re-combine for the store to complete.

Stores are never performed speculatively since there is no transparent way to undo them. Stores are also never re-ordered among themselves. A store is dispatched only when both the address and the data are available and there are no older stores awaiting dispatch.

A study of the importance of memory access reordering concluded:

- Stores must be constrained from passing other stores, for only a small impact on performance.
- Stores can be constrained from passing loads, for an inconsequential performance loss.
- Constraining loads from passing other loads or stores has a significant impact on performance.

The Memory Order Buffer (MOB) allows loads to pass other loads and stores by acting like a reservation station and re-order buffer. It holds suspended loads and stores and re-dispatches them when a blocking condition (dependency or resource) disappears.

## 2.3 ARCHITECTURE SUMMARY

All products within the P6 family of processors are based upon this architectural summary. Dynamic Execution is the combination of improved branch prediction, speculative execution and data flow analysis that enable P6 family processors to deliver superior performance.



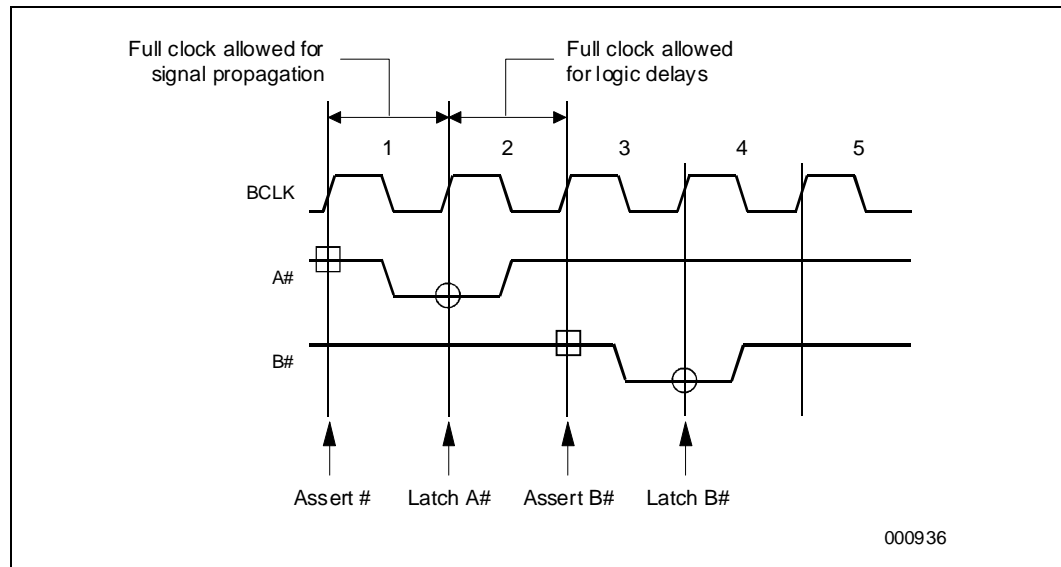
This chapter provides an overview of the P6 family system bus, bus transactions, and bus signals. The P6 family of processor's also support two other synchronous busses (the APIC and the TAP bus), PC compatibility signals, and several implementation specific signals. For a complete signal listing of a specific product, please refer to the relevant datasheet.

## 3.1 SIGNALING ON P6 FAMILY SYSTEM BUS

The P6 family processor system bus supports a synchronous latched protocol. On the rising edge of the bus clock, all agents on the system bus are required to drive their active outputs and sample required inputs. No additional logic is located in the output and input paths between the buffer and the latch stage, thus keeping setup and hold times constant for all bus signals following the latched protocol. The System bus requires that every input be sampled during a valid sampling window on a rising clock edge and its effect be driven out no sooner than the next rising clock edge. This approach allows one full clock for inter-component communication and at least one full clock at the receiver to compute a response.

Figure 3-1 illustrates the latched bus protocol as it appears on the bus. In subsequent descriptions, the protocol is described as “B# is asserted in the clock after A# is observed active,” or “B# is asserted two clocks after A# is asserted.” Note that A# is asserted in T1, but not observed active until T2. The receiving agent uses T2 to determine its response and asserts B# in T3. Other agents observe B# active in T4.

Figure 3-1. Latched Bus Protocol



The square and circle symbols are used in the timing diagrams to indicate the clock in which particular signals of interest are driven and sampled. The square indicates that a signal is driven (asserted, initiated) in that clock. The circle indicates that a signal is sampled (observed, latched) in that clock.

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. The bus signals that must meet this criteria are: BINIT#, HIT#, HITM#, BNR#, AERR#, BERR#.

## 3.2 SIGNAL OVERVIEW

This section describes the function of the System bus signals. In this section, the signals are grouped according to function.

### 3.2.1 Execution Control Signals

Table 3-1 lists the execution control signals, which control the execution and initialization of the processor.

**Table 3-1. Execution Control Signals**

| Pin/Signal Name                             | Pin/Signal Mnemonic           |
|---|-------------------------------|
| Bus Clock                                   | BCLK                          |
| Initialization                              | INIT#, RESET#                 |
| Flush                                       | FLUSH#                        |
| Stop Clock                                  | STPCLK#                       |
| Sleep                                       | SLP#                          |
| Interprocessor Communication and Interrupts | PICCLK, PICD[1:0]#, LINT[1:0] |

The BCLK (Bus Clock) input signal is the System bus clock. All agents drive their outputs and latch their inputs on the BCLK rising edge. Each processor in the P6 family derives its internal clock from BCLK by multiplying the BCLK frequency by a multiplier determined at configuration. See Chapter 5, “Configuration,” for possible clock configuration frequencies.

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

The INIT# input signal resets all processors without affecting their internal (L1 or L2) caches, floating-point registers, or their Machine Check Architecture registers (MCi-CTL). Each processor begins execution at the address vector as defined during power on configuration. INIT# has another meaning on RESET#’s active to inactive transition: if INIT# is sampled active on RESET#’s active to inactive transition, then the processor executes its built-in self-test (BIST).

If the FLUSH# input signal is asserted, the processor writes back all internal cache lines in the Modified state (L1 and L2 caches) and invalidates all internal cache lines (L1 and L2 caches). The flush operation puts all internal cache lines in the Invalid state. All lines are written back and invalidated. The FLUSH# signal has a different meaning when it is sampled asserted on the active to inactive transition of RESET#. If FLUSH# is sampled asserted on the active to inactive transition of RESET#, then the processor tristates all of its outputs. This function is used during board testing.

The P6 family processor supplies a STPCLK# pin to enable the processor to enter a low power state. When STPCLK# is asserted, the processor puts itself into the Stop-Grant state. The processor continues to snoop bus transactions while in Stop-Grant state. When STPCLK# is deasserted, the processor restarts its internal clock to all units and resumes execution. The assertion of STPCLK# has no effect on the bus clock.

Some processors in the P6 family support the SLP# signal. The SLP# signal is the sleep signal. When asserted in Stop-Grant state, the processor enters a new low power state, the Sleep state. During Sleep state, the processor stops providing internal clock signals to all units, only leaves PLL still running. Snooping during the Sleep state is not supported. Please see specific processor datasheets for more information on this feature.

The PICCLK and PICD[1:0]# signals support the Advanced Programmable Interrupt Controller (APIC) interface. The PICCLK signal is a clock input for the processor's APIC bus clock. The PICD[1:0]# signals are used for bi-directional serial message passing on the APIC bus.

LINT[1:0] are local interrupt signals, also defined by the APIC interface. In APIC disabled mode, LINT0 defaults to INTR, a maskable interrupt request signal. LINT1 defaults to NMI, a non-maskable interrupt. Both signals are asynchronous inputs. In the APIC enable mode, LINT0 and LINT1 are defined with the local vector table.

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

### 3.2.2 Arbitration Signals

The arbitration signal group (see Table 3-2) is used to arbitrate for the bus. These signals are used to gain ownership of the bus, a requirement for initiating a bus transaction.

Some P6 family processors permit up to five agents to simultaneously arbitrate for the system bus with one to four symmetric agents (on BREQ[3:0]#) and one priority agent (on BPRI#). The following paragraphs describes arbitration from this perspective. P6 family processors arbitrate as symmetric agents. The priority agent normally arbitrates on behalf of the I/O subsystem (I/O agents) and memory subsystem (memory agents). Please note that in systems based on P6 family processors that support only two symmetric agents, the following descriptions are accurate with the exception of the number of symmetric agents, the number of BR# pins and the number of BREQ# signals.

**Table 3-2. Arbitration Signals**

| Pin/Signal Name             | Pin Mnemonic | Signal Mnemonic |
|-----------------------------|--------------|-----------------|
| Symmetric Agent Bus Request | BR[3:0]#     | BREQ[3:0]#      |
| Priority Agent Bus Request  | BPRI#        | BPRI#           |
| Block Next Request          | BNR#         | BNR#            |
| Lock                        | LOCK#        | LOCK#           |

The symmetric agents arbitrate for the bus based on a round-robin rotating priority scheme. The arbitration is fair and symmetric. After reset, agent 0 has the highest priority followed by agent 1. All bus agents track the current bus owner. A symmetric agent requests the bus by asserting its BREQn# 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 active, no symmetric agent issues another unlocked bus transaction until BPRI# is sampled inactive. 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 bus. It is typically asserted when system resources (such as address and/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 bus transaction until all bus agents are initialized.

The assertion of the LOCK# signal indicates that the bus agent is executing an atomic sequence of bus transactions that must not be interrupted. A locked operation cannot be interrupted by another transaction regardless of the assertion of BREQ[3:0]# or BPRI#. LOCK# can be used to implement memory-based semaphores. LOCK# is asserted from the start of the first transaction through the end of the last transaction. The LOCK# signal is always deasserted between two sequences of locked transactions on the System bus.

### 3.2.3 Request Signals

The request signals (see [Table 3-3](#)) initiate a transaction.

**Table 3-3. Request Signals**

| Pin Name        | Pin Mnemonic | Signal Name    | Signal Mnemonic |
|-----------------|--------------|----------------|-----------------|
| Address Strobe  | ADS#         | Address Strobe | ADS#            |
| Request Command | REQ[4:0]#    | Request        | REQ[4:0]#       |
| Address         | A[35:3]#     | Address        | A[35:3]#        |
| Address Parity  | AP[1:0]#     | Address Parity | AP[1:0]#        |
| Request Parity  | RP#          | Request Parity | RP#             |

The assertion of ADS# defines the beginning of the transition. The REQ[4:0]#, A[35:3]#, RP# and AP[1:0]# signals are valid in the clock that ADS# is asserted.

In the clock that ADS# is asserted, the A[35:3]# signals provide a 36-bit, active-low address as part of the request. The P6 family processor maximum physical address space is  $2^{36}$  bytes or 64-Gigabytes (64 GByte). Some P6 family processors may implement less address lines. Address bits 2, 1, and 0 are mapped into byte enable signals for 1 to 8 byte transfers.

The address signals are protected by the AP[1:0]# pins on some processors. AP1# covers A[35:24]#, AP0# covers A[23:3]#. AP[1:0]# must be valid for two clocks beginning when ADS# is asserted. A parity signal on the system bus is correct if there are an even number of electrically low signals in the set consisting of the covered signals plus the parity signal. Parity is computed using voltage levels, regardless of whether the covered signals are active high or active low.

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

### 3.2.4 Snoop Signals

The snoop signal group (see [Table 3-4](#)) provides snoop result information to the System bus agents.

**Table 3-4. Snoop Signals**

| Type                              | Signal Names |
|-----------------------------------|--------------|
| Keeping a Non-Modified Cache Line | HIT#         |
| Hit to a Modified Cache Line      | HITM#        |
| Defer Transaction Completion      | DEFER#       |

On observing a transaction, HIT# and HITM# 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.

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

### 3.2.5 Response Signals

The response signal group (see [Table 3-5](#)) provides response information to the requesting agent.

**Table 3-5. Response Signals**

| Type                      | 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 response agent is the agent responsible for completing the transaction at the top of the In-order Queue. The response agent is the agent addressed by the transaction.

For write transactions, TRDY# is asserted by the response 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 again for the implicit writeback data transfer.

The RSP# signal provides parity for RS[2:0]#. A parity signal on the System bus is correct if there are 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 Response Signals

The data response signals (see Table 3-6) control the transfer of data on the bus and provide the data path.

**Table 3-6. Data Phase Signals**

| Type                | Signal Names |
|---------------------|--------------|
| Data Ready          | DRDY#        |
| Data Bus Busy       | DBSY#        |
| Data                | D[63:0]#     |
| Data ECC Protection | DEP[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 transfer.

DBSY# is used to hold the 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 if no wait states are needed.

The D[63:0]# signals provide a 64-bit data path between bus agents.

Some P6 family processors support data bus error correcting code (ECC). In these processors, the DEP[7:0]# signals provide optional ECC covering D[63:0]#. As described in Chapter 5, “Configuration,” the P6 family data bus can be configured with either no checking or ECC. If ECC is enabled, then DEP[7:0]# provides valid ECC for the entire data bus on each data clock, regardless of which bytes are enabled. The error correcting code can correct single bit errors and detect double bit errors. Please see specific processor datasheets for more information.

#### 3.2.6.1 LINE TRANSFERS

A line transfer reads or writes a cache line, the unit of caching on the P6 family processor system bus. For current products, this is 32 bytes aligned on a 32-byte boundary. While a line is always aligned on a 32-byte boundary, a line transfer need not begin on that boundary. For a line transfer, A[35:3]# carry the upper 33 bits of a 36-bit physical address. Address bits A[4:3]# determine the transfer order, called burst order. A line is transferred in four eight-byte chunks, each of which can be identified by address bits 4:3. The chunk size is 64-bits. Table 3-7 specifies the transfer order used for a 32-byte line, based on address bits A[4:3]# specified in the transaction’s Request Phase.

**Table 3-7. Burst Order Used for P6 Family Processor Bus Line Transfers**

| A[4:3]# (binary) | Requested Address (hex) | 1st Address Transferred (hex) | 2nd Address Transferred (hex) | 3rd Address Transferred (hex) | 4th Address Transferred (hex) |
|------------------|-------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 00               | 0                       | 0                             | 8                             | 10                            | 18                            |
| 01               | 8                       | 8                             | 0                             | 18                            | 10                            |
| 10               | 10                      | 10                            | 18                            | 0                             | 8                             |
| 11               | 18                      | 18                            | 10                            | 8                             | 0                             |

Note that the requested read data is always transferred first. Unlike the Pentium processor, which always transfers writeback data address 0 first, the P6 family transfers writeback data requested address first.

### 3.2.6.2 PART LINE ALIGNED TRANSFERS

A part-line aligned transfer moves a quantity of data smaller than a cache line but an even multiple of the chunk size between a bus agent and memory using the burst order. A part-line transfer affects no more than one line in a cache.

A 16-byte transfer on a 64-bit data bus with a 32-byte cache line size is a part-line transfer, where a chunk is eight bytes aligned on an eight-byte boundary. All chunks in the span of a part-line transfer are moved across the data bus. Address bits A[4:3]# determines the transfer order for the included chunks, using the burst order specified in [Table 3-7](#) for line transfers.

### 3.2.6.3 PARTIAL TRANSFERS

On a 64-bit data bus, a partial transfer moves from 0-8 bytes within an aligned 8-byte span to or from a memory or I/O address.

Processors convert non-cacheable misaligned memory accesses that cross 8-byte boundaries into two partial transfers. For example, a non-cacheable, misaligned 8-byte read requires two Read Data Partial transactions. Similarly, processors convert I/O write accesses that cross 4-byte boundaries into 2 partial transfers. I/O reads are treated the same as memory reads.

I/O Read and I/O Write transactions are 1 to 4 byte partial transactions.

## 3.2.7 Error Signals

[Table 3-8](#) lists the error signals on the system bus.

**Table 3-8. Error Signals**

| Type                 | Signal Names |
|----------------------|--------------|
| Address PArity Error | AERR         |
| Bus Initialization   | BINIT#       |
| Bus Error            | BERR#        |
| Internal Error       | IERR#        |
| FRC Error            | FRCERR       |
| Thermal Overrun      | THERMTRIP#   |

AERR#, can be enabled or disabled as part of the power on configuration (see Chapter 5, “Configuration”). If AERR# is disabled for all system bus agents, request and address parity errors are ignored and no action is taken by bus agents. If AERR# is enabled for at least one bus agent, the agents supporting address parity and observing the start of a transaction check the Address Parity signals (AP[1:0]#) and the RP# parity signal and assert AERR# appropriately if an address parity error is detected.

P6 family processors support two modes of response when the AERR# signal is enabled. This may be configured at power-up with “AERR# observation” mode. AERR# observation configuration must be consistent between all bus agents. If AERR# observation is disabled, AERR# is ignored and no action is taken by the bus agents. If AERR# observation is enabled and AERR# is sampled asserted, the transaction is canceled. In addition, the requesting agent may retry the transaction at a later time up to its retry limit, after which the error becomes a hard error as determined by the initiating processor.

If a transaction is canceled by AERR# assertion, then the transaction is aborted. Snoop results are ignored if they cannot be canceled in time. All agents reset their rotating ID for bus arbitration to the state at reset (such that bus agent 0 has highest priority).

BINIT# is used to signal any bus condition that prevents reliable future operation of the bus. Like the AERR# pin, the BINIT# driver can be enabled or disabled as part of the power-on configuration (see [Chapter 5, “Configuration”](#)). If the BINIT# driver is disabled, BINIT# is never asserted and no action is taken on bus errors.

Regardless of whether the BINIT# driver is enabled, the P6 family processor supports two modes of operation that may be configured at power on. These are the BINIT# observation and driving modes. If BINIT# observation is disabled, BINIT# is ignored and no action is taken by the processor even if BINIT# is sampled asserted. If BINIT# observation is enabled and BINIT# is sampled asserted, all bus state machines are reset. All agents reset their rotating ID for bus arbitration, and internal state information is lost. L1 and L2 cache contents are not affected.

The BERR# pin 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, non-modified snoop error). A bus error that causes the assertion of BERR# can be detected by the processor, or by another bus agent. The BERR# driver can be enabled or disabled at power-on reset. If the BERR# driver is disabled, BERR# is never asserted. If the BERR# driver is enabled, the processor may assert BERR#.

A machine check exception may or may not be taken for each assertion of BERR# as configured at power on. A processor will always disable the machine check exception by default.

If a processor detects an internal error unrelated to bus operation, it asserts IERR#. For example, a parity error in an L1 or L2 cache causes a Pentium Pro processor to assert IERR#. A machine check exception may be taken instead of assertion of IERR# as configured with software.

Two processor agents in the P6 family may be configured as an FRC (functional redundancy checking) pair. In this configuration, one processor acts as the master and the other acts as a checker, and the pair operates as a single processor. If the checker agent detects a mismatch between its internally sampled outputs and the master processor's outputs, the checker asserts FRCERR. FRCERR observation can be enabled at the master processor with software. The master enters machine check on an FRCERR provided that Machine Check Execution is enabled.

The FRCERR signal is also toggled during an FRC checker agent's reset action. FRCERR is asserted one clock after RESET# transitions from its active to inactive state. If the checker processor executes its built-in self-test (BIST), then FRCERR is asserted throughout that test. After BIST completes, the checker processor deasserts FRCERR only if BIST succeeded but continues to assert FRCERR if BIST failed. This feature allows the failure to be externally observed. If the checker processor does not execute its BIST, then it keeps FRCERR asserted for less than 20 clocks and then deasserts it.

The processor protects itself 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. The processor will stop all execution when the junction temperature exceeds the sensor setting. This is signaled to the system by the THERMTRIP# (Thermal Trip) pin. Once activated, the signal remains latched, and the processor stopped, until RESET# goes active. There is no hysteresis built into the thermal sensor itself; as long as the die temperature drops below the trip level, a RESET# pulse will reset the processor and execution will continue. If the temperature has not dropped below the trip level, the processor will continue to drive THERMTRIP# and remain stopped.

### 3.2.8 Compatibility Signals

The compatibility signals group (see [Table 3-9](#)) contains signals defined for compatibility within the Intel Architecture processor family.

**Table 3-9. PC Compatibility Signals**

| Type                        | Signal Names |
|-----------------------------|--------------|
| Floating-Point Error        | FERR#        |
| Ignore Numeric Error        | IGNNE#       |
| Address 20 Mask             | A20M#        |
| System Management Interrupt | SMI#         |

A P6 family agent asserts FERR# when it detects an unmasked floating-point error. FERR# is included for compatibility with systems using MS-DOS\*-type floating-point error reporting.

If the IGNNE# input signal is asserted, the processor ignores a numeric error and continues to execute non-control floating-point instructions. If the IGNNE# input signal is deasserted, the processor freezes on a non-control floating-point instruction if a previous instruction caused an error.

If the A20M# input signal is asserted, the processor masks physical address bit 20 (A20#) before looking up a line in any internal cache and before driving a memory read/write transaction on the bus. Asserting A20M# emulates the 8086 processor's address wraparound at the one Mbyte boundary. A20M# must only be asserted when the processor is in real mode. A20M# is not used to mask external snoop addresses.

The IGNNE# and A20M# signals are valid at all times. These signals are normally not guaranteed recognition at specific boundaries.

The A20M# and IGNNE# signals have different meanings during a reset. A20M# and IGNNE# are sampled on the active to inactive transition of RESET# to determine the multiplier for the internal clock frequency, as described in Chapter 5, "Configuration."

System Management Interrupt (SMI) is asserted asynchronously by system logic. On accepting an SMI, the processor saves the current state and enters SMM mode. It issues an SMI Acknowledge Bus transaction and then begins program execution from the SMM handler.

### 3.2.9 Diagnostic Signals

The diagnostic signals group shown in [Table 3-10](#) provides signals for probing the processor, monitoring processor performance, and implementing an IEEE 1149.1 boundary scan.

The BP[3:2]# signals are the System Support group Breakpoint signals. They are outputs from the processor that indicate the status of breakpoints.

The BPM[1:0]# signals are more System Support group breakpoint and performance monitor signals. They are outputs from the processor that indicate the status of breakpoints and programmable counters used for monitoring performance.

PRDY# and PREQ# are signals used by the debug tools. The PRDY# (Probe Ready) signal is used to determine processor debug readiness. PREQ# (Probe Request) is used by the debug tools to request debug operation of the processors.

**Table 3-10. Diagnostic Support Signals**

| Type                      | Signal Names              |
|---------------------------|---------------------------|
| Breakpoint Signals        | BP[3:2]#                  |
| Performance Monitor       | BPM[1:0]#                 |
| Boundary Scan/Test Access | TCK, TDI, TDO, TMS, TRST# |
| In-Target Probe Access    | PRDY#, PREQ#              |

The P6 family processor system bus architecture incorporates several advanced data integrity features to improve error detection, retry, and correction. The processor system bus architecture includes parity protection for address/request signals, parity or protocol protection on most control signals, and ECC protection for data signals. Some P6 family processors also provide the maximum possible level of error detection by incorporating functional redundancy checking (FRC) support.

The P6 family data integrity features can be categorized as follows:

- Processor internal error detection
- Level 2 (L2) cache and Core-to-L2 cache-interface error detection and limited recovery
- P6 family processor system bus error detection and limited recovery
- P6 family processor system bus FRC support

Some P6 family processors may not support all these data intergroup features.

In addition, the P6 family extends the processor's data integrity features in several ways to form machine check architecture. Several model specific registers are defined for reporting error status. Hardware corrected errors are reported to registers associated with the unit reporting the error. Unrecoverable errors cause the INT 18 machine check exception, as in the Pentium Pro processor.

If machine check is disabled, or an error occurs in a P6 family processor system bus agent without the machine check architecture, the P6 family processor system bus defines a bus error reporting mechanism. The central agent can then be configured to invoke the exception handler via an interrupt (NMI) or soft reset (INIT#).

The terminology used in this chapter is listed below:

- Machine Check Architecture (MCA)
- Machine Check Exception (MCE)
- Machine Check Enable bit (CR4.MCE)
- Machine Check In Progress (MCIP)

For more information on Machine Check Architecture, see the *Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide*.

## 4.1 ERROR CLASSIFICATION

The P6 family processor system bus architecture uses the following error classification. An implementation may always choose to report an error in a more severe category to simplify its logic.

- **Recoverable Error (RE):** The error can be corrected by a retry or by using ECC information. The error is logged in the MCA hardware.

- **Unrecoverable Error (UE):** The error cannot be corrected, but it only affects one agent. The memory interface logic and bus pipeline are intact, and can be used to report the error via an exception handler.
- **Fatal Error (FE):** The error cannot be corrected and may affect more than one agent. The memory interface logic and bus pipeline integrity may have been violated, and cannot be reliably used to report the error via an exception handler. A bus pipeline reset is required of all bus agents before operation can continue. An exception handler may then proceed.

## 4.2 P6 FAMILY PROCESSOR SYSTEM BUS DATA INTEGRITY ARCHITECTURE

The P6 family processor system bus' major address and data paths are protected by ten check bits, providing parity or ECC. Eight ECC bits protect the data bus. Single-bit data ECC errors are automatically corrected. A two-bit parity code protects the address bus. Any address parity error on the address bus when the request is issued can be optionally retried to attempt a correction.

Two control signal groups are explicitly protected by individual parity bits: RP# and RSP#. 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 without the use of FRC mode.

An agent is not required to support all data integrity features, as each feature is individually enabled through the power-on configuration register. See [Chapter 5, "Configuration."](#)

### 4.2.1 Bus Signals Protected Directly

For processors with ECC and parity protection, [Table 4-1](#) shows which signals protect which signals.

**Table 4-1. Direct Bus Signal Protection**

| Signal    | Protects       |
|-----------|----------------|
| RP#       | ADS#,REQ[4:0]# |
| AP[0]#    | A[23:3]#       |
| AP[1]#    | A[35:24]#      |
| RSP#      | RS[2:0]#       |
| DEP[7:0]# | D[63:0]#       |

- **Address/Request Bus Signals.** A parity error detected on AP[1:0]# or RP# is reported or retried based on the following options defined by the power-on configuration:
  - AERR# driver 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.
  - AERR# driver enabled, AERR# observation disabled.  
The agent detecting the parity error asserts the AERR# signal. This signal can be trapped by the central agent and be driven back to one of the processors as NMI.
  - AERR# driver enabled, AERR# observation enabled.

The agent detecting the parity error asserts the AERR# signal. All bus agents must observe AERR# and on the next clock reset bus arbiters and abort the erroneous transaction by removing the transaction from the In-Order Queue and canceling all remaining phases associated with the transaction.

- **Response Signals.** A parity error detected on RSP# should be reported by the agent detecting the error as a fatal error.
- **Data Transfer Signals.** The P6 family processor system bus can be configured with either no data-bus error checking or with ECC. If ECC is selected, single-bit errors can be corrected and double-bit errors can be detected. Corrected single-bit ECC errors are logged as recoverable errors. All other errors are reported as unrecoverable errors. The errors on read data being returned are treated by the requester as unrecoverable errors. The errors on write or writeback data are treated by the target as fatal errors.
- **Snoop Processing.** An error discovered during a snoop lookup may be treated as a recoverable error if the cache state is E,S, or I. If the cache is in the M state, the errors are treated as fatal errors. Any implementation may choose to report all snoop errors as fatal errors.

## 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

Errors on some P6 family processor system bus signals cannot be detected:

- The execution control signals CLK, RESET#, and INIT# are not protected.
- The error signals FRCERR and IERR# are not protected.
- The PC compatibility signals FERR#, IGNNE#, A20M#, and FLUSH# are not protected.
- The system support signals SMI# and STPCLK#, and SLP# are not protected.

## 4.2.4 Hard-Error Response

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

On observing a hard-error response, the initiator may treat it as a unrecoverable or a fatal error.

## 4.2.5 P6 Family Processor System Bus Error Code Algorithms

### 4.2.5.1 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.

#### **4.2.5.2 P6 FAMILY SYSTEM BUS ECC ALGORITHM**

The P6 family architecture system bus uses an ECC code that can correct single-bit errors, detect double-bit errors, and detect all errors confined to one nibble (SEC-DED-S4ED). System designers may choose to detect all these errors, or a subset of these errors.

This chapter describes configuration options for P6 family processor agents.

A system may contain one, two, or four P6 family processors. Processors can also be used in FRC configurations, with two physical processors in each logical FRC unit. All processors are connected to one P6 family processor system bus. FRC capability and Multi-Processor capability is not supported by all P6 family processors. Please see specific processor datasheets for more details.

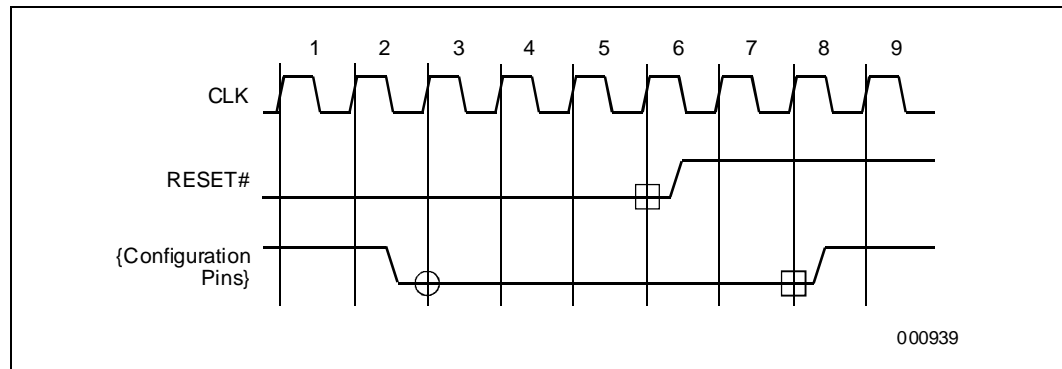
## 5.1 DESCRIPTION

P6 family processors have some configuration options which are determined by hardware, and some which are determined by software.

P6 family processor system bus agents sample their hardware configuration at reset, on the active-to-inactive transition of RESET#. The configuration signals (except IGNNE#, A20M# and LINT[1:0]) must be asserted 4 clocks before the active-to-inactive transition of RESET# and be deasserted two clocks after the active-to-inactive transition of RESET# (see Figure 5-1). The IGNNE#, A20M#, and LINT[1:0] signals must meet a setup time of 1 ms to the active-to-inactive 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. Reset may also reconfigure other system bus agents.

**Figure 5-1. Hardware Configuration Signal Sampling**



P6 family processor system bus agents can also be configured with some additional software configuration options. These options can be changed by writing to a power-on configuration register which all bus agents must implement. These options should be changed only after taking into account synchronization between multiple P6 family processor system bus agents.

P6 family processor system bus agents have the following configuration options when applicable:

- Output tristate {Hardware}
- Execution of the processor's built-in self-test (BIST) {Hardware}
- Data bus error-checking policy: enabled or disabled {Software}

- Response signal error-checking policy: parity disabled or parity enabled {Software}
- AERR# driving policy: enabled or disabled {Software}
- AERR# observation policy: enabled or disabled {Hardware}
- BERR# driving policy for initiator bus errors: enabled or disabled {Software}
- BERR# driving policy for target bus errors: enabled or disabled {Software}
- BERR# driving policy for initiator internal errors: enabled or disabled {Software}
- BINIT# error-driving policy: enabled or disabled {Software}
- BINIT# error-observation policy: enabled or disabled {Hardware}
- In-order Queue depth: 1 or 8 {Hardware}
- Power-on reset vector: 1M-16 or 4G-16 {Hardware}
- FRC mode: enabled or disabled {Hardware}
- APIC cluster ID: 0 or 1 {Hardware}
- APIC mode: enabled or disabled {Software}
- Symmetric agent arbitration ID: 0, 1, 2, or 3 {Hardware}
- Clock frequencies and ratios {Hardware}

### 5.1.1 Output Tristate

A processor tristates all of its outputs if the FLUSH# signal is sampled active on the RESET# signal's active-to-inactive transition. The only way to exit from Output Tristate mode is with a new activation of RESET# with inactive FLUSH#.

### 5.1.2 Built-in Self-test

A processor executes its built-in self-test (BIST) if the INIT# signal is sampled active on the RESET# signal's active-to-inactive transition. No software control is available to perform built-in self-test (BIST).

### 5.1.3 Data Bus Error Checking Policy

The P6 family data bus error checking can be enabled or disabled. After active RESET#, data bus error checking is always disabled. Data bus error checking can be enabled under software control. For more information on this feature, please refer to the Intel Architecture Software Developer's Manuals for more information.

### 5.1.4 Response Signal Parity Error Checking Policy

The P6 family processor system bus supports parity protection for the response signals, RS[2:0]#. The parity checking on these signals can be enabled or disabled. After active RESET#, response signal parity checking is disabled. It can be enabled under software control.

### 5.1.5 AERR# Driving Policy

The P6 family address bus parity protection on the Request signals, A[35:3]#, ADS# and REQ[4:0]#. However, driving the address parity results on the AERR# pin is optional. After active RESET#, address bus parity error driving is always disabled. It may be enabled under software control.

### 5.1.6 AERR# Observation Policy

The AERR# input receiver is enabled if A8# is observed active on active-to-inactive transition of RESET#. No software control is available to perform this function.

### 5.1.7 BERR# Driving Policy for Initiator Bus Errors

A P6 family processor system bus agent can be enabled to drive the BERR# signal if it detects a bus error. After active RESET#, BERR# signal driving is disabled for detected errors. It may be enabled under software control.

### 5.1.8 BERR# Driving Policy for Target Bus Errors

A P6 family processor system bus agent can be enabled to drive the BERR# signal if the addressed (target) bus agent detects an error. After active RESET#, BERR# signal driving is disabled on target bus errors. It may be enabled under software control. The processor does not drive BERR# on target detected bus errors.

### 5.1.9 BERR# Driving Policy for Initiator Internal Errors

On internal errors, a P6 family processor system bus agent can be enabled to drive the BERR# signal. After active RESET#, BERR# signal driving is disabled on internal errors. It may be enabled under software control.

### 5.1.10 BINIT# Driving Policy

On bus protocol violations, a P6 family processor system bus agent can be enabled to drive the BINIT# signal. After active RESET#, BINIT# signal driving is disabled. It may be enabled under software control. A P6 family processor relies on BINIT# driving to be enabled during normal operation.

### 5.1.11 BINIT# Observation Policy

The BINIT# input receiver is enabled for bus initialization control if A10# is observed active on the active-to-inactive transition of RESET#. A P6 family processor requires BINIT# observation to be enabled during normal operation.

### 5.1.12 In-Order Queue Pipelining

P6 family processor system bus agents are configured to an In-order Queue depth of one if A7# is observed active on RESET#. Otherwise it defaults to an In-order Queue depth of eight. This function cannot be controlled by software. A depth of eight will normally provide the best performance.

### 5.1.13 Power-On Reset Vector

The reset vector at which a processor begins execution after an active RESET# is controlled by sampling A6# on the RESET# signal’s active-to-inactive transition. The reset vector for the processor is 0FFFF0H (1 MB–16) if A6# is sampled active. Otherwise, the reset vector is 0FFFFFFF0H (4 GB–16).

### 5.1.14 FFRC Mode Enable

Some P6 family processor system bus agents can be configured to support a mode in which FRC is disabled or a mode in which FRC is enabled. The processor enters FRC enabled mode if A5# is sampled active on the active-to-inactive transition of RESET#, otherwise it enters FRC disabled mode.

### 5.1.15 APIC Mode

APIC may be enabled or disabled via software. For details, see the *Intel Architecture Software Developer’s Manual, Volume 3*, Chapter 7, “Multiple Processor Management.”

### 5.1.16 APIC Cluster ID

P6 family processors that support multiprocessing provide common APIC bus support for up to four processor-bus clusters, where each cluster contains a P6 family processor bus and up to four P6 family processors. The APIC cluster ID is a 2-bit value that identifies a bus cluster: 0, 1, 2, or 3. The processor determines its APIC cluster ID by sampling A12# and A11# on the RESET# signal’s active-to-inactive transition based on [Table 5-1](#).

**Table 5-1. APIC Cluster ID Configuration**

| APIC Cluster ID | A12# | A11# |
|-----------------|------|------|
| 0               | H    | H    |
| 1               | H    | L    |
| 2               | L    | H    |
| 3               | L    | L    |

**NOTE:** L and H designate electrical levels.

### 5.1.17 Symmetric Agent Arbitration ID

The P6 family processor system bus supports symmetric distributed arbitration among one to four agents. Each processor identifies its initial position in the arbitration priority queue based on an agent ID supplied at configuration. The agent ID can be 0, 1, 2 or 3 for each processor in systems

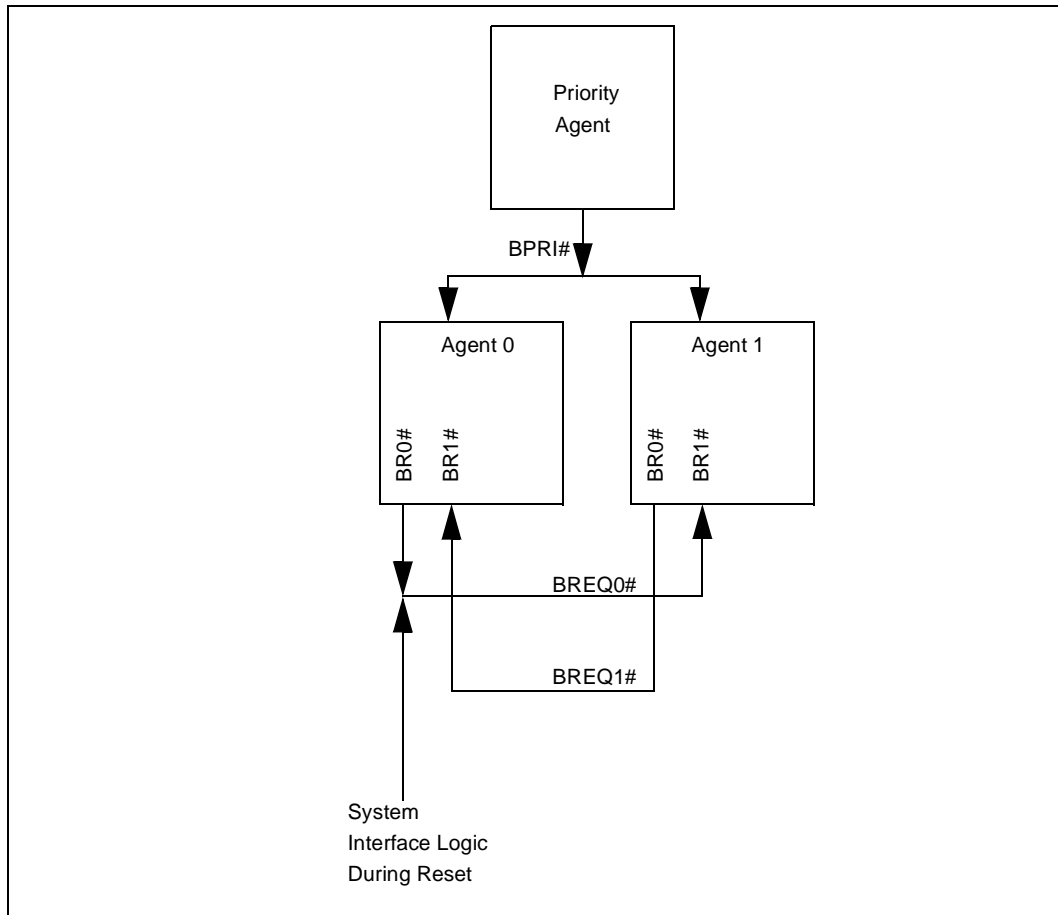
which support four processors, and 0 or 1 in systems that support 2 processors. Each logical processor (not an FRC master/checker pair) on a particular P6 family processor system bus must have a distinct agent ID.

For processors supporting two symmetric agents, the BREQ[1:0]# bus signals are connected to the two symmetric agents as shown in Table 5-2 and in Figure 5-2. Each symmetric agent has one I/O pin (BR0#) and one input only pin (BR1#).

**Table 5-2. P6 Family Processor Bus BREQ[1:0]# Interconnect (2-Way MP Processors)**

| Bus Signal | Agent ID 0 Physical Pin | Agent ID 1 Physical Pin |
|------------|-------------------------|-------------------------|
| BREQ0#     | BR0#                    | BR1#                    |
| BREQ1#     | BR1#                    | BR0#                    |

**Figure 5-2. BR[1:0]# Physical Interconnection with Two Symmetric Agents**

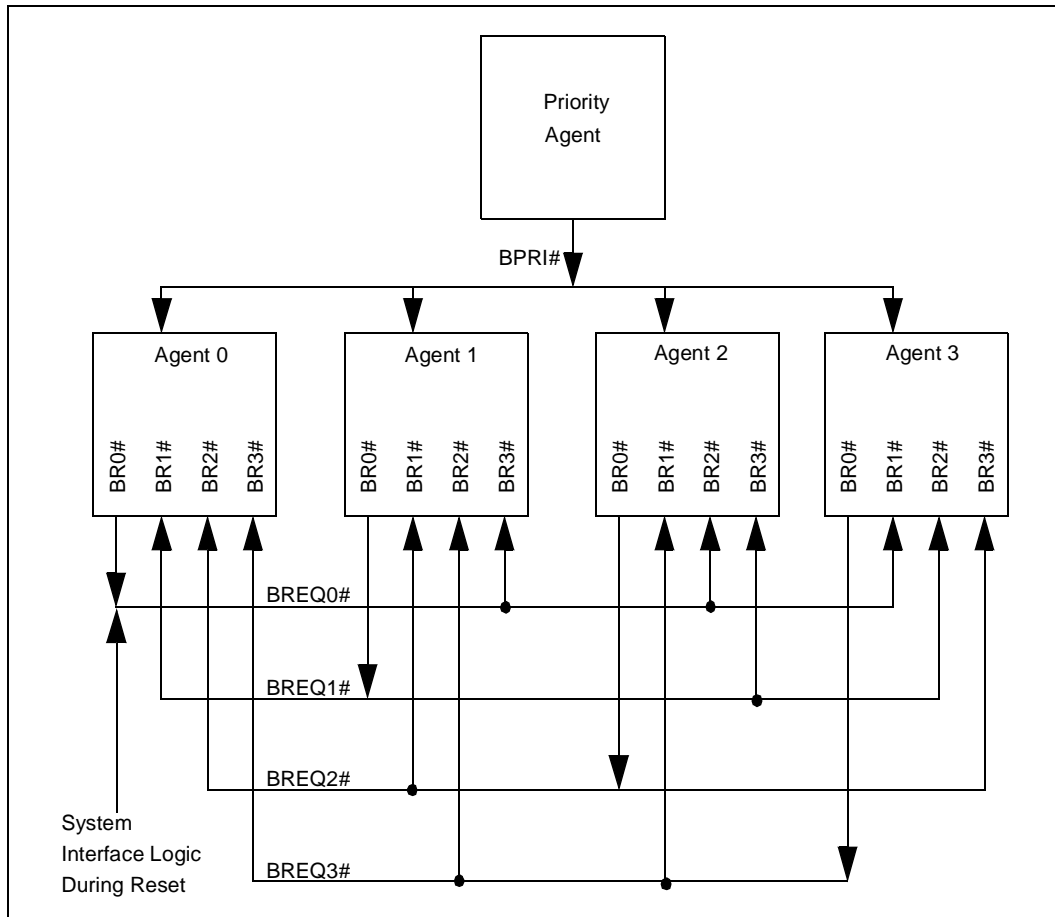


For processors supporting four symmetric agents, the BREQ[3:0]# signals are connected to the four agents as shown in Table 5-3 and Figure 5-3. Each symmetric agent has one I/O pin (BR0#) and three input only pins (BR1#, BR2#, and BR3#).

**Table 5-3. P6 Family Processor Bus BREQ[3:0]# Interconnect (4-Way MP Processors)**

| Bus Signal | Agent ID 0 Physical Pin | Agent ID 1 Physical Pin | Agent ID 2 Physical Pin | Agent ID 3 Physical Pin |
|------------|-------------------------|-------------------------|-------------------------|-------------------------|
| BREQ0#     | BR0#                    | BR3#                    | BR2#                    | BR1#                    |
| BREQ1#     | BR1#                    | BR0#                    | BR3#                    | BR2#                    |
| BREQ2#     | BR2#                    | BR1#                    | BR0#                    | BR3#                    |
| BREQ3#     | BR3#                    | BR2#                    | BR1#                    | BR0#                    |

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



At the RESET# signal's active-to-inactive transition, system interface logic is responsible for assertion of the BREQ0# bus signal. The BREQ1# bus signal remains deasserted in systems with two symmetric agents and BREQ[3:1]# bus signals remain deasserted in systems with four symmetric agents. For systems with two symmetric agents, both processors sample their BR1# pin on the RESET signal's active-to-inactive transition and determine their agent ID from the sampled value. For a system with four symmetric agents all processor's sample their BR[3:1]# pins on the RESET signals's active-to-inactive transition and determine their agent ID from the sampled value.

If FRC is not enabled, then each physical processor is a logical processor. Each processor is designated a non-FRC master and each processor has a distinct agent ID.

If FRC is used, then two physical processors are combined to create a single logical processor. In a two processor system, a processor with pin BR0# driven at reset is designated as an FRC-master and uses agent ID 0. A processor with pin BR1# driven at reset is designated as an FRC checker for processor 0 and assumes the characteristics of its respective master as shown in Table 5-4.

In a four processor system, processors with agent id 0 and 2 are designated as FRC-masters and use their agent ID as their parallel bus arbitration ID. Processors with agent ID 1 and 3 are designated as FRC checkers for processors 0 and 2 respectively and assume the characteristics of their respective masters as shown in Table 5-5.

**Table 5-4. Arbitration ID Configuration (Two Agents)**

| BR0# | BR1# | A5# | Arbitration ID |
|------|------|-----|----------------|
| L    | H    | H   | 0              |
| H    | L    | H   | 1              |
| L    | H    | L   | 0 (master)     |
| H    | L    | L   | 0 (checker)    |

**NOTE:** L and H designate electrical levels.

**Table 5-5. Arbitration ID Configuration (Four Agents)**

| BR0# | BR1# | BR2# | BR3# | A5# | Agent ID | Arbitration ID |
|------|------|------|------|-----|----------|----------------|
| L    | H    | H    | H    | H   | 0        | 0              |
| H    | H    | H    | L    | H   | 1        | 1              |
| H    | H    | L    | H    | H   | 2        | 2              |
| H    | L    | H    | H    | H   | 3        | 3              |
| L    | H    | H    | H    | L   | 0        | 0 (master)     |
| H    | H    | H    | L    | L   | 1        | 0 (checker)    |
| H    | H    | L    | H    | L   | 2        | 2 (master)     |
| H    | L    | H    | H    | L   | 3        | 2 (checker)    |

### 5.1.18 Low Power Standby Enable

A configuration register bit which enables distribution of the core clock during AutoHALT and Stop Grant mode has been included in the power-on configuration register. This register will support bit D26, which can be read and written by software.

- D26=1** (Default for all P6 family processors except the Pentium® Pro processor)

In this mode when the processor enters AutoHALT or Stop Grant, it will not distribute a clock to its core units. This allows the processor to reduce its standby power consumption, but large current transients are produced upon entering and exiting this mode.
- D26=0** (Default for Pentium Pro processor)

In this mode, AutoHALT and Stop Grant will not stop internal clock distribution. The processor will have higher standby power consumption, but will produce smaller current transients on entering and exiting this mode.

## 5.2 CLOCK FREQUENCIES AND RATIOS

The P6 family uses a ratio clock design, in which the bus clock is multiplied by a ratio to produce the processor’s internal (or “core”) clock. The processor begins sampling LINT[1:0], A20M# and IGNNE# on the inactive-to-active transition of RESET# to determine the core-frequency to bus-frequency relationship and immediately begins the internal PLL lock mode. On the active-to-inactive transition of RESET#, the processor internally latches the inputs to allow the pins to be used for normal functionality. Effectively, these pins must meet a large setup time (please refer to your processor datasheet for this value) to the active-to-inactive transition of RESET#.

Table 5-6 describes the settings for relationships between bus frequency and core frequency. For supported settings see the processor datasheet.

**Table 5-6. System Bus To Core Frequency Multiplier Configuration**

| Ratio of Processor Core Frequency to System Bus Frequency | LINT[1] | LINT[0] | IGNNE# | A20M# |
|---|---------|---------|--------|-------|
| Reserved  | H       | L       | H      | H     |
| 3/2   | H       | H       | L      | H     |
| 2   | H       | H       | H      | H     |
| 2   | L       | L       | L      | L     |
| 5/2   | L       | H       | L      | L     |
| 3   | L       | L       | H      | L     |
| 7/2   | L       | H       | H      | L     |
| 4   | L       | L       | L      | H     |
| 9/2   | L       | H       | L      | H     |
| 5   | L       | L       | H      | H     |
| 11/2  | L       | H       | H      | H     |
| 6   | H       | L       | L      | L     |
| 13/2  | H       | H       | L      | L     |
| 7   | H       | L       | H      | L     |
| 15/2  | H       | H       | H      | L     |
| 8   | H       | L       | L      | H     |

## 5.3 POWER-ON CONFIGURATION REGISTER

All bus agents are required to maintain some software read/writeable bits in the power-on configuration register for software-configured options. This register inside P6 family processors is defined in Table 5-7.

**Table 5-7. Processor Power-On Configuration Register**

| Feature   | Processor Active Signals    | Processor Register Bits                                 | Read/Write | Default  |
|---|-----------------------------|---|------------|----------|
| Output tristate enabled                                       | FLUSH#                      | D8=1  | Read       | N/A      |
| Execute BIST  | INIT#                       | D9=1  | Read       | N/A      |
| Data error checking enabled                                   | N/A                         | D1=1  | Read/Write | Disabled |
| Response error checking enabled<br>FRCERR observation enabled | N/A                         | D2=1  | Read/Write | Disabled |
| AERR# driver enabled  | N/A                         | D3=1  | Read/Write | Disabled |
| AERR# observation enabled                                     | A8#                         | D10=1   | Read       | N/A      |
| BERR# driver enabled for initiator bus requests               | N/A                         | D4=1  | Read/Write | Disabled |
| BERR# driver enabled for target bus requests                  | N/A                         | Reserved  | Read/Write | Disabled |
| BERR# driver enabled for initiator internal errors            | N/A                         | D6=1  | Read/Write | Disabled |
| BERR# observation enabled                                     | A9#                         | Reserved  | Read       | N/A      |
| BINIT# driver enabled   | N/A                         | D7=1  | Read/Write | Disabled |
| BINIT# observation enabled                                    | A10#                        | D12=1   | Read       | N/A      |
| In-order queue depth of 1                                     | A7#                         | D13=1   | Read       | N/A      |
| 1 Mbyte power-on reset vector                                 | A6#                         | D14=1   | Read       | N/A      |
| FRC Mode enabled  | A5#                         | D15=1   | Read       | N/A      |
| APIC cluster ID   | A12#, A11#                  | D17, D16<br>see <a href="#">Table 5-8</a>               | Read       | N/A      |
| Reserved  | A14#, A13#                  | D19, D18  | —          | —        |
| Symmetric arbitration ID                                      | BR0#, BR1#,<br>BR2#, BR3#   | D21, D20<br>see <a href="#">Table 5-9</a>               | Read       | N/A      |
| Clock frequency ratios  | LINT[1:0], A20M#,<br>IGNNE# | D25, D24, D23,<br>D22<br>see <a href="#">Table 5-10</a> | Read       | N/A      |
| Low power standby enable                                      | N/A                         | D26   | Read/Write | Enabled  |

**Table 5-8. Power-On Configuration Register APIC Cluster ID Bit Field**

| APIC ID | D[17:16] |
|---------|----------|
| 0       | 00       |
| 1       | 01       |
| 2       | 10       |
| 3       | 11       |

**Table 5-9. Power-On Configuration Register Arbitration ID Configuration**

| Arbitration ID | D[21:20] |
|----------------|----------|
| 0              | 00       |
| 1              | 01       |
| 2              | 10       |
| 3              | 11       |

**Table 5-10. Power-On Configuration Register Bus Frequency to Core Frequency Ratio Bit Field**

| D[25:22] | Ratio of Core Frequency to Bus Frequency |
|----------|--|
| 0000     | 5  |
| 0001     | 3  |
| 0010     | 4  |
| 0011     | 2  |
| 0100     | 11/2                                     |
| 0101     | 7/2                                      |
| 0110     | 9/2                                      |
| 0111     | 5/2                                      |
| 1000     | Reserved                                 |
| 1001     | 7  |
| 1010     | 8  |
| 1011     | 6  |
| 1100     | 2  |
| 1101     | 15/2                                     |
| 1110     | 3/2                                      |
| 1111     | 13/2                                     |

## 5.4 INITIALIZATION PROCESS

After establishing configuration options, a processor executes the following initialization actions:

- Synchronize the internal phase-locked loop (PLL) used to derive the processor clock from the bus clock.
- Configure the parallel bus arbiter based on the processor’s agent ID and FRC enable pin. Configure the APIC bus arbiter ID with additional information available via APIC cluster ID.
- If enabled by the configuration options, begin execution of the built-in self-test (BIST).
- Begin fetching and executing code from the reset address, 00\_FFFF\_FFF0H or 00\_000F\_FFF0.

During initialization, each processor begins active BNR# sequencing from the RESET# signal’s active-to-inactive transition until it is able to accept (though not necessarily issue) bus transactions.

Signals that have special meanings during initialization assume their normal roles for a particular processor when the processor first asserts ADS# after a reset.

Each processor can obtain its power-on-configuration information from a 32-bit register in the MSR space. This register can be read by the initialization software and different processors can then be initialized differently based on their agent ID.

When the reset condition is generated by the activation of RESET#, BPRI# and BNR# must be sampled inactive together on a valid BNR# sampling point, to allow new request generation by a symmetric agent.



This chapter describes the implementation of the P6 family test access port (TAP) logic. The TAP complies with the IEEE 1149.1 (“JTAG”) test architecture standard. Basic functionality of the 1149.1-compatible test logic is described here, but this chapter does not describe the IEEE 1149.1 standard in detail. For this information, the reader is referred to the published standard<sup>1</sup>, and to the many books currently available on the subject.

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

For specific boundary scan chain information, see the BSDL file for the specific processor at [developer.intel.com](http://developer.intel.com).

## 6.1 INTERFACE

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

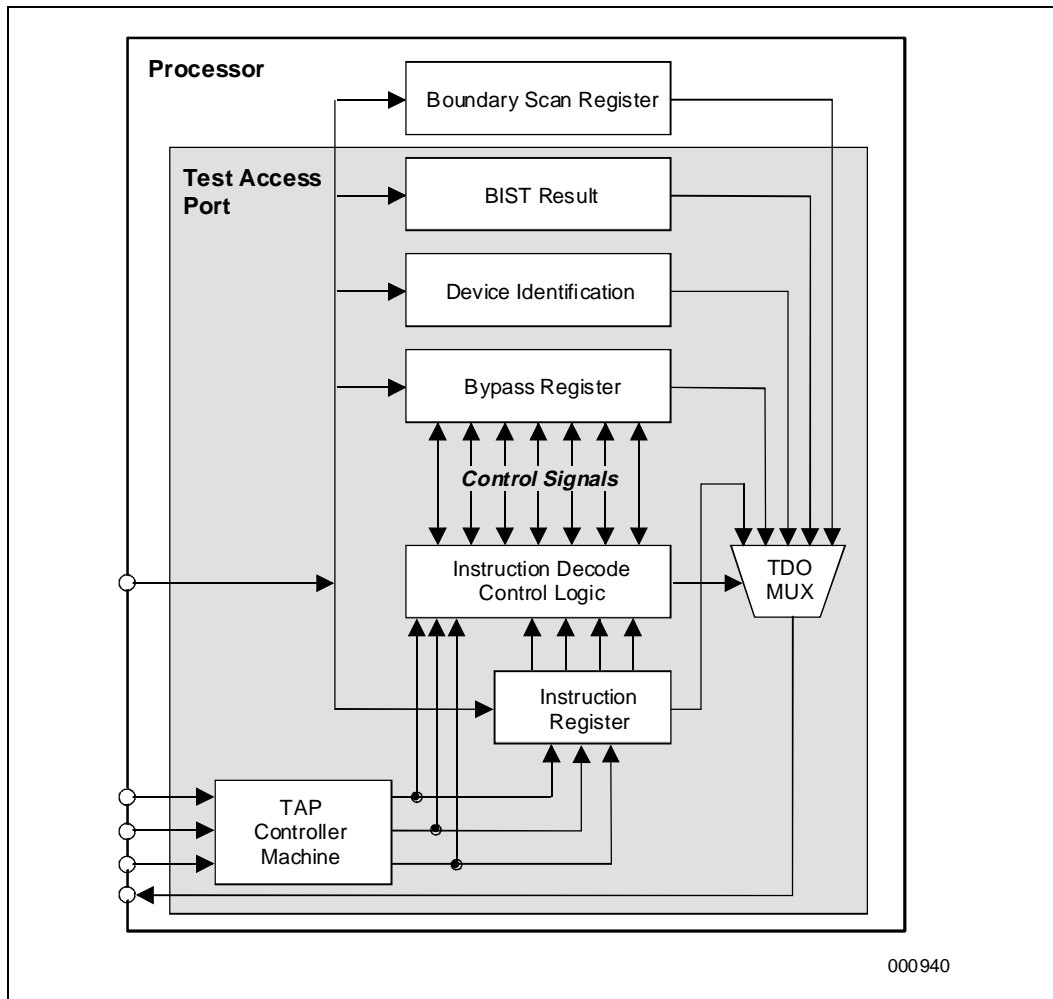
- **TCK:** The TAP clock signal
- **TMS:** “Test mode select,” which controls the TAP finite state machine
- **TDI:** “Test data input,” which inputs test instructions and data serially
- **TRST#:** “Test reset,” for TAP logic reset
- **TDO:** “Test data output,” through which test output is read serially

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

---

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

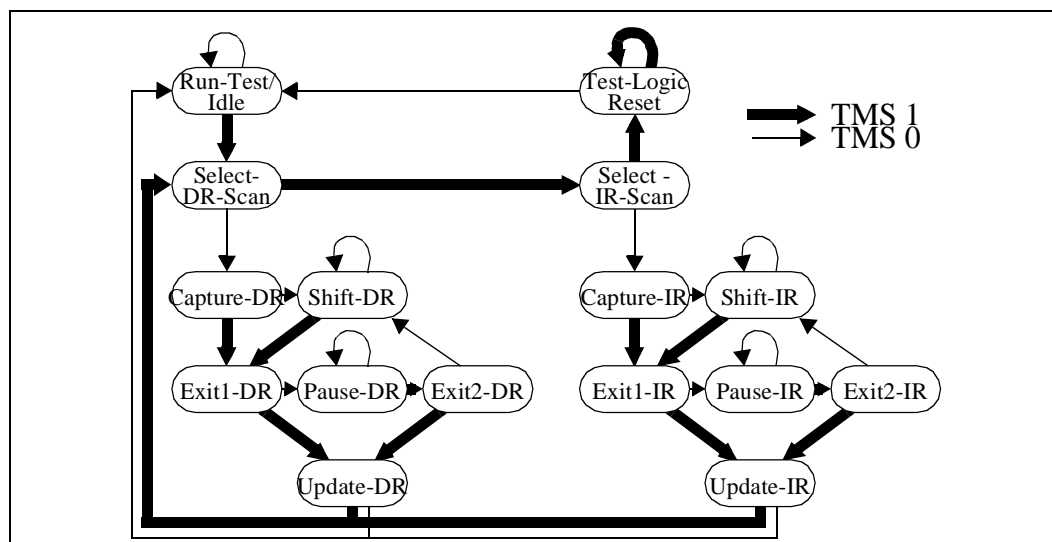
Figure 6-1. Simplified Block Diagram of Processor TAP Logic



## 6.2 ACCESSING THE TAP LOGIC

The TAP is accessed through a 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 Finite State Machine



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 normal operation of the processor can continue. In this state, the instruction in the Instruction Register is forced to IDCODE. The controller is guaranteed to enter Test-Logic-Reset when the TMS input is held active for at least five clocks. The controller also enters this state immediately when TRST# is pulled active, and automatically upon power-up of the processor. The TAP controller cannot leave this state as long as TRST# is held active.
- **Run-Test/Idle:** This is the idle state of the TAP controller. In this state, the contents of all test data registers retain their previous values.
- **Select-IR-Scan:** This is a temporary controller state. All registers retain their previous values.
- **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.
- **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.
- **Exit1-IR:** This is a temporary state. The current instruction does not change.
- **Pause-IR:** Allows shifting of the instruction register to be temporarily halted. The current instruction does not change.
- **Exit2-IR:** This is a temporary state. The current instruction does not change.
- **Update-IR:** The instruction which has been shifted into the Instruction Register is latched onto 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 TAP controller state machine is reset).
- **Select-DR-Scan:** This is a temporary controller state. All registers retain their previous values.
- **Capture-DR:** In this state, the data register selected by the current instruction may capture data at its parallel inputs.

- **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. The parallel, latched output of the selected Data Register does not change while new data is being shifted in.
- **Exit1-DR:** This is a temporary state. All registers retain their previous values.
- **Pause-DR:** Allows shifting of the selected Data Register to be temporarily halted without stopping TCK. All registers retain their previous values.
- **Exit2-DR:** This is a temporary state. All registers retain their previous values.
- **Update-DR:** Data from the shift register path is loaded into the latched parallel outputs of the selected Data Register (if applicable) on the falling edge of TCK. This (and Test-Logic-Reset) is the only state in which the latched paralleled outputs of a data register can change.

## 6.2.1 Accessing the Instruction Register

Figure 6-3 shows the (simplified) physical implementation of the TAP instruction register. This register consists of a 6-bit shift register (connected between TDI and TDO), and the actual instruction register (which is loaded in parallel from the shift register). The parallel output of the TAP instruction register goes to the TAP instruction decoder, shown in Figure 6-1. This architecture conforms to the 1149.1 specification.

Figure 6-3. Processor TAP Instruction Register

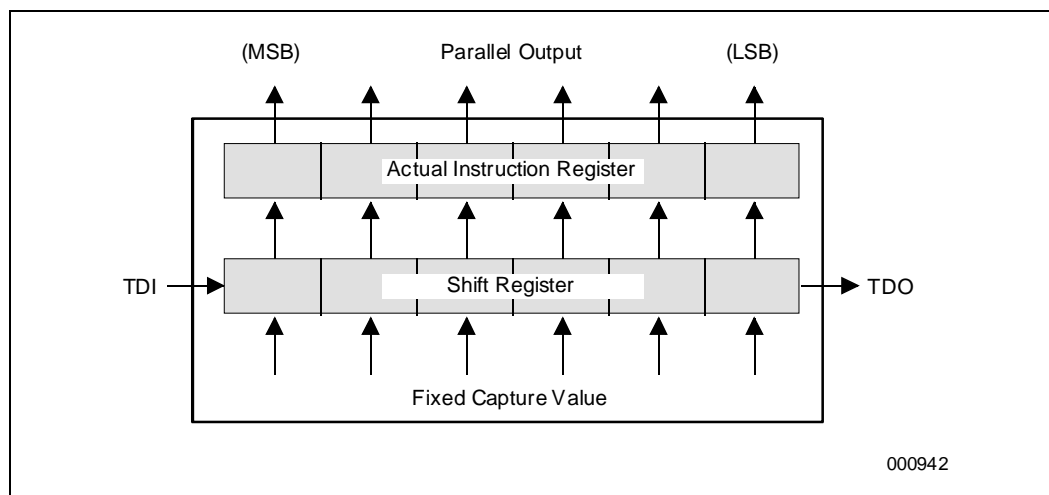
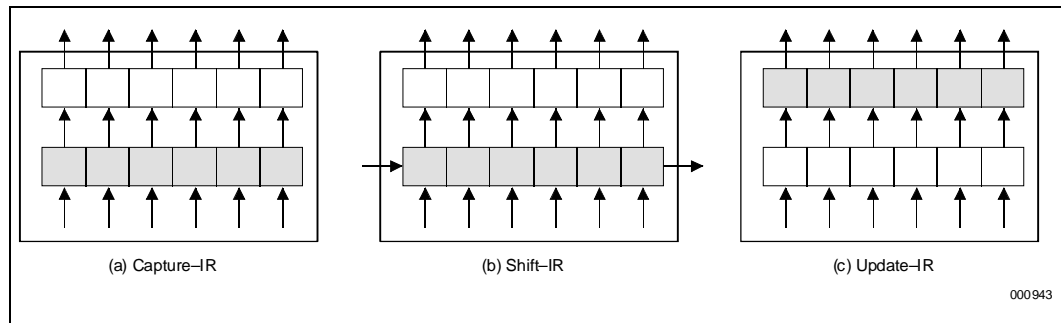


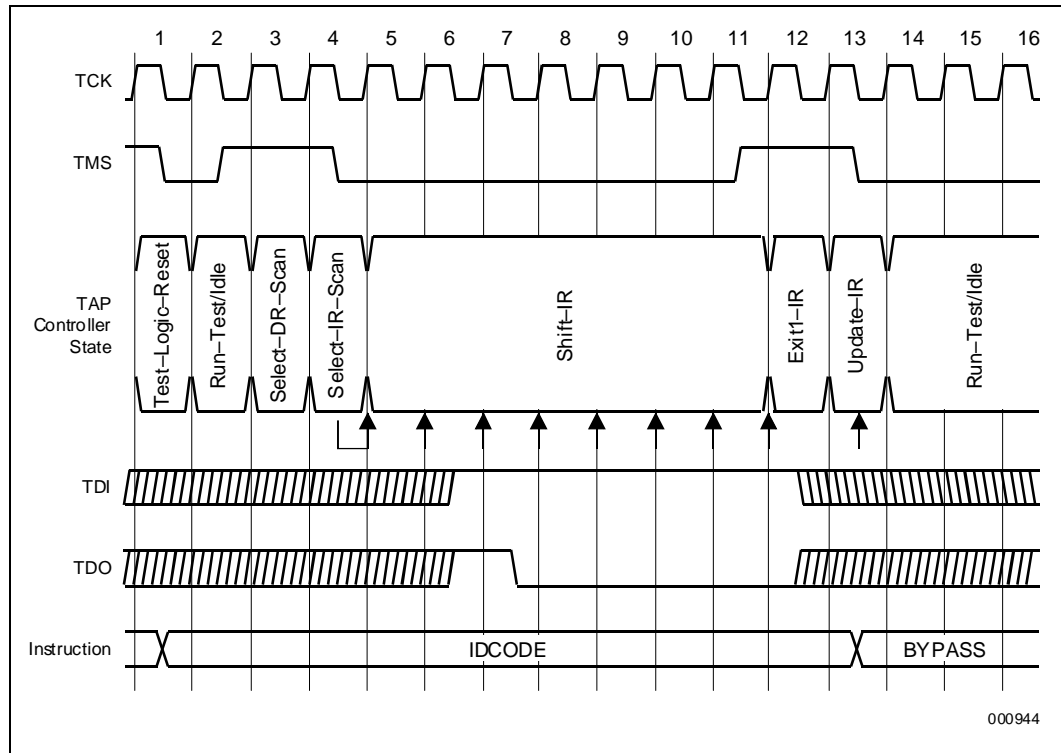
Figure 6-4 shows the operation of the TAP instruction register during the Capture-IR, Shift-IR and Update-IR states of the TAP controller. Flip-flops within the instruction register which are updated in each mode of operation are shaded. In Capture-IR, the shift register portion of the instruction register is loaded in parallel with the fixed value “000001.” In Shift-IR, the shift register portion of the instruction register forms a serial data path between TDI and TDO. In Update-IR, the shift register contents are latched in parallel into the actual instruction register. Note that the only time the outputs of the actual instruction register change is during Update-IR. Therefore, a new instruction shifted into the TAP does not take effect until the Update-IR state of the TAP controller is entered.

Figure 6-4. Operation of the Processor TAP Instruction Register



A timing diagram for loading the BYPASS instruction (op-code “11111”) into the TAP is shown in Figure 6-5. (Note that the LSB of the TAP instruction must be shifted in first.) Vertical arrows on the figure show the specific clock edges on which the Capture-IR, Shift-IR and Update-IR actions actually take place. Capture-IR (which pre-loads the instruction shift register with “000001”) and Shift-IR operate on rising edges of TCK, and Update-IR (which updates the actual instruction register) takes place on the falling edge of TCK.

Figure 6-5. TAP Instruction Register Access



## 6.2.2 Accessing the Data Registers

The test data registers in the processor are designed in the same way as the instruction register, with components (i.e., either the “capture” or “update” functionality) removed from the basic structure as needed. Data registers are accessed just as the instruction register is, only using the “select-DR-scan” branch of the TAP finite state machine in Figure 6-2. A specific data register is selected for access by each TAP instruction. Note that the only controller states in which data register contents

actually change are Capture-DR, Shift-DR, Update-DR and Run-Test/Idle. For each of the TAP instructions described below, therefore, it is noted what operation (if any) occurs in the selected data register in each of these four states.

## 6.3 INSTRUCTION SET

Table 6-1 contains descriptions of the encoding and operation of the TAP instructions. There are seven 1149.1-defined instructions implemented in the TAP. These instructions select from among four different TAP data registers — the boundary scan, BIST result, device ID, and bypass registers.

**Table 6-1. 1149.1 Instructions in the Processor TAP**

| TAP Instruction | Opcode    | Processor Pins Drive From: | Data Register Selected | Action During: |                               |                     |                      |
|-----------------|-----------|----------------------------|------------------------|----------------|-------------------------------|---------------------|----------------------|
|                 |           |                            |                        | RT/Idle        | Capture-DR                    | Shift-DR            | Update-DR            |
| EXTEST          | 000000    | Boundary scan              | Boundary scan          | —              | Sample all processor pins     | Shift data register | Update data register |
| SAMPLE/PRELOAD  | 000001    | —                          | Boundary scan          | —              | Sample all processor pins     | Shift data register | Update data register |
| IDCODE          | 000010    | —                          | Device ID              | —              | Load unique processor ID code | Shift data register | —                    |
| CLAMP           | 000100    | Boundary scan              | Bypass                 | —              | Reset bypass reg              | Shift data register | —                    |
| RUNBIST         | 000111    | Boundary scan              | BIST result            | BIST starts†   | Capture BIST result           | Shift data register | —                    |
| HIGHZ           | 001000    | Floated                    | Bypass                 | —              | Reset bypass reg              | Shift data register | —                    |
| BYPASS          | 111111    | —                          | Bypass                 | —              | Reset bypass reg              | Shift data register | —                    |
| Reserved        | All other | Reserved                   | Reserved               | Reserved       | Reserved                      | Reserved            | Reserved             |

**NOTE:**

† The processor must be reset after this command.

TAP instructions in the P6 family are 6 bits long. For each listed instruction, the table shows the instruction’s encoding, what happens on the processor pins, which TAP data register is selected by the instruction, and the actions which occur in the selected data register in each of the controller states. A single hyphen indicates that no action is taken. Note that not all of the TAP data registers have a latched parallel output (i.e., some are only simple shift registers). For these data registers, nothing happens during the Update-DR controller state.

Full details of the operation of these instructions can be found in the 1149.1 standard.

The only TAP instruction which does not operate exactly as defined in the 1149.1 standard is RUNBIST. In the 1149.1 specification, Rule 7.9.1(b) states that: “Self-test mode(s) of operation accessed through the RUNBIST instruction shall execute only in the Run-Test/Idle controller state.” In the implementation of RUNBIST used in the P6 family, the execution of the BIST routine will not stop if the Run-Test/Idle state is exited before BIST is complete. In all other regards, RUNBIST instruction operates exactly as defined in the 1149.1 specification.

Note that RUNBIST will not function when the processor core clock has been stopped. All other 1149.1-defined instructions operate independently of the processor core clock.

The op-codes are 1149.1-compliant, and are consistent with the Intel-standard op-code encodings and backward-compatible with the Pentium processor 1149.1 instruction op-codes.

## 6.4 DATA REGISTER SUMMARY

Table 6-2 gives the complete list of test data registers which can be accessed through the TAP. The MSB of the register is connected to TDI (for writing), and the LSB of the register is connected to TDO (for reading) when that register is selected.

**Table 6-2. TAP Data Registers**

| TAP Data Register | Size | Selected by Instructions |
|-------------------|------|--------------------------|
| Bypass            | 1    | BYPASS, HIGHZ, CLAMP     |
| Device ID         | 32   | IDCODE                   |
| BIST Result       | 1    | RUNBIST                  |
| Boundary Scan     | 159  | EXTEST, SAMPLE/PRELOAD   |

### 6.4.1 Bypass Register

The Bypass register provides a short path between TDI and TDO. It is loaded with a logical 0 in the Capture-DR state.

### 6.4.2 Device ID Register

The Device ID register contains the processor device identification code in the format shown in Table 6-3. The manufacturer’s identification code is unique to Intel. The part number code is divided into four fields:  $V_{CC}$ , product type (an Intel Architecture compatible processor), generation (sixth generation), and model. The version field is used for stepping information.

**Table 6-3. Device ID Register**

|               | Version | Part Number |              |            |       | Manufacturing ID | “1” | Entire Code                          |
|---------------|---------|-------------|--------------|------------|-------|------------------|-----|--------------------------------------|
|               |         | $V_{CC}$    | Product Type | Generation | Model |                  |     |                                      |
| <b>Size</b>   | 4       | 1           | 6            | 4          | 5     | 11               | 1   | 32                                   |
| <b>Binary</b> | xxxx    | 0           | 000001       | 0110       | 00011 | 00000001001      | 1   | xxxx300000101100<br>0011000000010011 |
| <b>Hex</b>    | x       | 0           | 01           | 6          | 03    | 09               | 1   | x02c3013                             |

### 6.4.3 BIST Result Boundary Scan Register

Holds the results of BIST. It is loaded with a logical 0 on successful BIST completion.

### 6.4.4 Boundary Scan Register

Contains a cell for each defined processor signal pin. For more information on Boundary Scan, refer to the *Boundary Scan Description Language* files specific to each P6 family processor at the Intel developer’s website at [developer.intel.com](http://developer.intel.com).

## 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. Once in this state, all of the reset actions listed in [Table 6-4](#) are performed. 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 the TAP does not receive RESET#.

**Table 6-4. TAP Reset Actions**

| TAP Logic Affected            | TAP Reset State Action     | Related TAP Instructions |
|-------------------------------|----------------------------|--------------------------|
| Instruction Register          | Loaded with IDCODE op-code | —                        |
| Processor boundary scan logic | Disabled                   | CLAMP, HIGHZ, EXTEST     |
| Processor TDO pin             | Tri-stated                 | —                        |

The TAP can be transitioned to the Test-Logic-Reset state in any one of 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 is guaranteed to transition the TAP controller to the Test-Logic-Reset state on a rising edge of TCK.

All P6 family processors support an In-Target Probe (ITP) for program execution control, register/memory/I/O access and breakpoint control. This tool provides functionality commonly associated with debuggers and emulators. An ITP uses on-chip debug features of the processor to provide program execution control. Use of an ITP will not affect the high speed operations of the processor signals, ensuring the system can operate at full speed with an ITP attached.

This chapter describes the ITP as well as a number of technical issues that must be taken into account when including an ITP in a debug strategy. Please note that some information contained in this chapter may not be in exact accordance to your specific product design. Some values (i.e., pull-up and pull-down resistors) are approximations. Please design your system with this in mind and rely on simulations of your design to ensure that signal integrity issues are avoided.

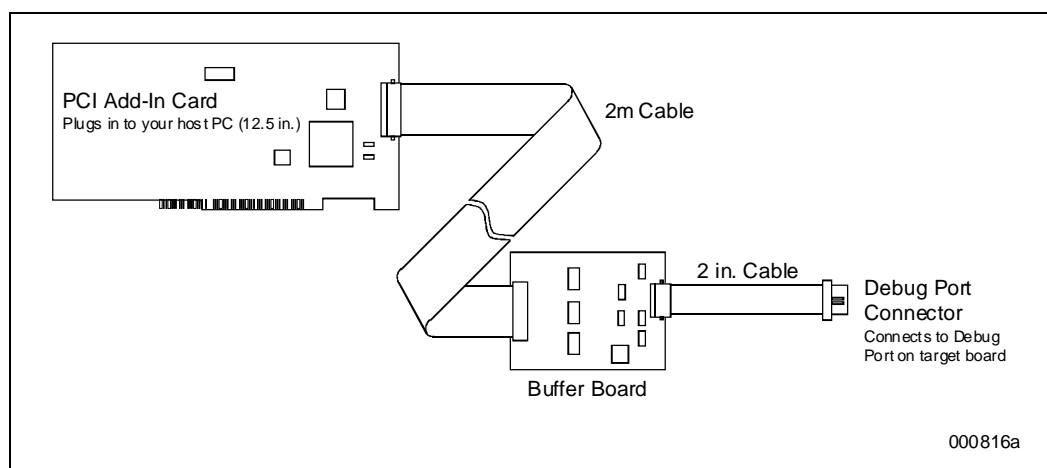
## 7.1 IN-TARGET PROBE (ITP) FOR P6 FAMILY PROCESSORS

An In-Target Probe (ITP) is a debug tool which allows access to on-chip debug features via a small port on the system board called the debug port. For companies offering ITP solutions, please refer to [www.developer.intel.com](http://www.developer.intel.com). The ITP communicates to the processor through the debug port using a combination of hardware and software. The software is an application running on a host PC. The hardware consists of a PCI board in the host PC connected to the signals which make up the processor's debug interface. Due to the nature of the ITP, the processor may be controlled without affecting any high speed signals. This ensures that the system can operate at full speed with the ITP attached. Intel uses the debug port for internal debug and system validation and recommends that all P6 family processor-based system designs include a debug port.

### 7.1.1 Primary Function

The primary function of an ITP is to provide a control and query interface for multiple processors. With an ITP, one can control program execution and have the ability to access processor registers, system memory and I/O. Thus, one can start and stop program execution using a variety of breakpoints, single-step the program at the assembly code level, as well as read and write registers, memory and I/O. The on-chip debug features are controlled from an application running on an Intel processor-based PC with a PCI card slot. (See [Figure 7-1](#).)

Figure 7-1. Hardware Components of the ITP



## 7.1.2 Debug Port Connector Description

The ITP connects to the system through the debug port. Recommended connectors, to mate the ITP cable with the debug port on the board, are available in either a vertical or right-angle configuration. Both configurations fit into the same board footprint. The connectors are manufactured by AMP Incorporated and are in the AMPMODU System 50 line. Following are the AMP part numbers for the two connectors:

- Amp 30-pin shrouded vertical header: 104068-3
- Amp 30-pin shrouded right-angle header: 104069-5

**Note:** These are high density through-hole connectors with pins on 0.050 in. by 0.100 in. centers. Do not confuse these with the more common 0.100 in. by 0.100 in. center headers.

The debug port must be mounted on the system motherboard; the processor does not contain a debug port.

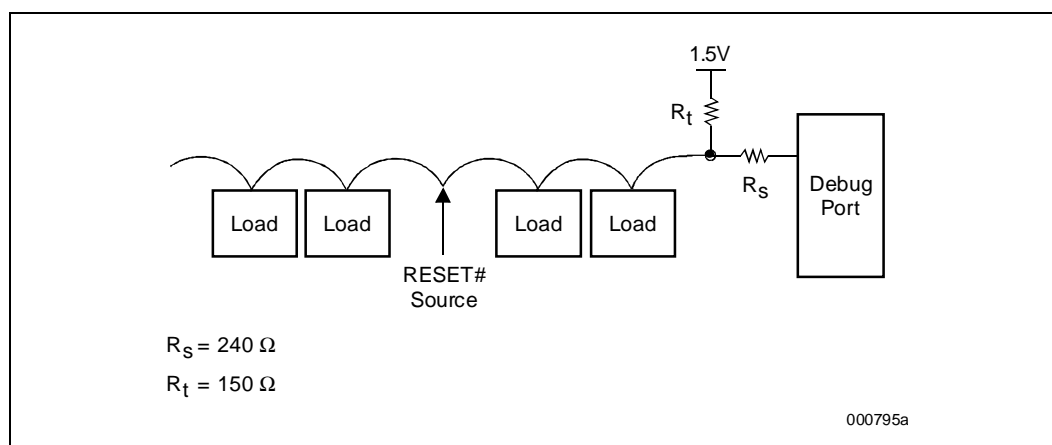
## 7.1.3 Debug Port Signal Descriptions

Table 7-1 describes the debug port signals and provides the pin assignment. The mechanical pinout is shown in Section 7.1.5.2.  $V_{CCMOS}$  is used to indicate the CMOS voltage level of signals in the system. This value is typically 2.5 V or 3.3 V. Please refer to your product specific datasheet for actual values.

## 7.1.4 Debug Port Signal Notes

In general, all open drain GTL+ outputs from the system must be retained at a proper logic level, whether or not the debug port is installed. GTL+ signals from the processor system (RESET#, PRDY#) should be terminated at the debug port, as shown in Figure 7-2.

Figure 7-2. GTL+ Signal Termination



#### 7.1.4.1 SIGNAL NOTE 1: DBRESET#

The DBRESET# output signal from the ITP is an open drain with about 5Ω of resistance. The usual implementation is to connect it to the PWROK open drain signal on the PCIset components as an OR input to initiate a system reset. In order for the DBRESET# signal to work properly, it must actually reset the entire target system. The signal should be pulled up (Intel recommends a 240Ω resistor, but system designers will need to fine tune specific system designs) to meet two considerations: (1) the signal must be able to meet  $V_{ol}$  of the system, and (2) it must allow the signal to meet the specified rise time. When asserted by the ITP, the DBRESET# signal will remain asserted for 100 ms. A large capacitance should not be present on this signal as it may prevent a full charge from building up within 100 ms.

#### 7.1.4.2 SIGNAL NOTE 5: TDO AND TDI

The TDO signal coming out of each processor has a 25Ω driver and should be pulled up to  $V_{CC_{CMOS}}$  using a 150Ω resistor.

**Note:** When designing the circuitry to reroute the scan chain around empty processor slots, care should be taken so that the multiple TDO pull-up resistors do not end up in parallel (see Figure 7-3). The TDI line coming out of the debug port should be pulled up to  $V_{CC_{CMOS}}$  using approximately a 10 KΩ resistor value.

Table 7-1. Debug Port Pinout Description and Requirements <sup>1</sup>

| Name     | Pin | Description   | Specification Requirement   | Notes   |
|----------|-----|---|---|---|
| RESET#   | 1   | Reset signal from MP cluster to ITP.  | <ul style="list-style-type: none"> <li>• Terminate<sup>2</sup> signal properly at the debug port</li> <li>• Debug port must be at the end of the signal trace</li> </ul>  | Connected to high speed comparator (biased at 2/3 of the level found at the POWERON pin) on the ITP buffer board. Additional load does not change timing calculations for the processor bus agents if routed properly.  |
| DBRESET# | 3   | Allows ITP to reset entire target system.   | <ul style="list-style-type: none"> <li>• Tie signal to target system reset (recommendation: PWR OK signal on PCIsset as an ORed input)</li> <li>• Pulled-up signal with the proper resistor (see notes)</li> </ul>  | Open drain output from ITP to the target system. It will be held asserted for 100 ms; capacitance needs to be small enough to recognize assert. A 240Ω pull-up resistor is recommended; signal should be fine tuned to (1) meet $V_{OL}$ of target system and (2) meet specified rise time. |
| TCK      | 5   | TAP (Test Access Port) clock from ITP to processor connectors.  | <ul style="list-style-type: none"> <li>• Place pull-up resistor (to <math>V_{CCCMOS}</math>) near the ITP.</li> </ul>   | Poor routing can cause multiple clocking problems, usually on falling edge of TCK. Should be routed to all components in the boundary scan chain <sup>3</sup> . For MP systems, each processor should receive a separately buffered TCK signal.   |
| TMS      | 7   | Test mode select signal from ITP to processor connectors, controls the TAP finite state machine   | <ul style="list-style-type: none"> <li>• Daisy chain signal to the processor(s)<sup>3</sup></li> <li>• Add ~150Ω pull-up resistor to <math>V_{CCCMOS}</math></li> <li>• Place pull-up resistor (to <math>V_{CCCMOS}</math>) at the point furthest from the debug port.</li> </ul> | Operates synchronously with TCK. Should be routed to all components in the boundary scan chain <sup>3</sup> . For MP systems, each processor should receive a separately buffered TMS signal.   |
| TDI      | 8   | Test data input signal from ITP to first component in boundary scan chain of MP cluster; inputs test instructions and data serially.                                      | <ul style="list-style-type: none"> <li>• Add ~150 to 330Ω pull-up resistor (to <math>V_{CCCMOS}</math>)</li> </ul>  | Operates synchronously with TCK.  |
| POWERON  | 9   | Used by ITP to determine when target system power is ON and, once target system is ON, enables all debug port electrical interface activity. From target $V_{TT}$ to ITP. | <ul style="list-style-type: none"> <li>• Add ~1KΩ pull-up resistor (to <math>V_{TT}</math>)</li> </ul>  | If no power is applied, the ITP will not drive any signals; isolation provided using isolation gates. Voltage applied is internally used to set GTL+ threshold (or reference) at 2/3 $V_{TT}$ .   |

**Table 7-1. Debug Port Pinout Description and Requirements (Continued)<sup>1</sup>**

| Name    | Pin | Description  | Specification Requirement   | Notes  |
|---------|-----|--|---|--|
| TDO     | 10  | Test data output signal from last component in boundary scan chain of MP cluster to ITP; test output is read serially. | <ul style="list-style-type: none"> <li>Add ~150Ω pull-up resistor (to V<sub>CC</sub><sub>CMOS</sub>)</li> <li>Design pull-ups to route around empty processor sockets (so resistors are not in parallel)</li> </ul>   | Operates synchronously with TCK. Each P6 family processor has a 25Ω driver.  |
| DBINST# | 11  | Indicates to target system that the ITP is installed.  | <ul style="list-style-type: none"> <li>Add ~10KΩ pull-up resistor</li> </ul>  | Not required if boundary scan is not used in target system.  |
| TRST#   | 12  | Test reset signal from ITP to MP cluster, used to reset TAP logic.   | <ul style="list-style-type: none"> <li>Add ~680Ω pull-down resistor</li> </ul>  | Asynchronous input signal.   |
| BSEN#   | 14  | Informs target system that ITP is using boundary scan.   |   | Not required if boundary scan is not used in target system.  |
| PREQ0#  | 16  | PREQ0# signal, driven by ITP, makes requests to P0 to enter debug.   | <ul style="list-style-type: none"> <li>Add 150 to 330Ω pull-up resistor (to V<sub>CC</sub><sub>CMOS</sub>)</li> </ul>   |  |
| PRDY0#  | 18  | PRDY0# signal, driven by P0, informs ITP that P0 is ready for debug.   | <ul style="list-style-type: none"> <li>Terminate<sup>2</sup> signal properly at the debug port</li> <li>Debug port must be at the end of the signal trace</li> </ul>  | Connected to high speed comparator (biased at 2/3 of the level found at the POWERON pin) on the ITP buffer board. Additional load does not change timing calculations for the processor bus agents if routed properly. |
| PREQ1#  | 20  | PREQ1# signal from ITP to P1.  | For MP design: <ul style="list-style-type: none"> <li>Add 150 to 330Ω pull-up resistor (to V<sub>CC</sub><sub>CMOS</sub>)</li> </ul> For uni-processor design: <ul style="list-style-type: none"> <li>Leave unconnected</li> </ul>  |  |
| PRDY1#  | 22  | PRDY1# signal from P1 to ITP.  | For MP design: <ul style="list-style-type: none"> <li>Terminate<sup>2</sup> signal properly at the debug port</li> <li>Debug port must be at the end of the signal trace</li> </ul> For uni-processor design: <ul style="list-style-type: none"> <li>Leave unconnected</li> </ul> | Connected to high speed comparator (biased at 2/3 of the level found at the POWERON pin) on the ITP buffer board. Additional load does not change timing calculations for the processor bus agents.                    |
| PREQ2#  | 24  | Signal not used.   | <ul style="list-style-type: none"> <li>Leave unconnected</li> </ul>   |  |
| PRDY2#  |     | Signal not used.   | <ul style="list-style-type: none"> <li>Leave unconnected</li> </ul>   |  |
| PREQ3#  |     | Signal not used.   | <ul style="list-style-type: none"> <li>Leave unconnected</li> </ul>   |  |
| BCLK    | 29  | Bus clock from the MP cluster.   | <ul style="list-style-type: none"> <li>Use a separate driver to drive signal to the debug port</li> <li>Must be connected to GND if it is not connected to BCLK</li> </ul>  | Separate driver is used to avoid loading issues associated with having the ITP either installed or uninstalled.  |

**Table 7-1. Debug Port Pinout Description and Requirements (Continued)<sup>1</sup>**

| Name   | Pin                                     | Description      | Specification Requirement   | Notes |
|--------|---|------------------|---|-------|
| PRDY3# | 30                                      | Signal not used. | <ul style="list-style-type: none"> <li>Leave unconnected</li> </ul>                 |       |
| GND    | 2, 4, 6, 13, 15, 17, 19, 21, 23, 25, 27 | Signal ground.   | <ul style="list-style-type: none"> <li>Connect all pins to signal ground</li> </ul> |       |

**NOTES:**

1. Resistor values with "~" preceding them can vary from the specified value; use resistors as close as possible to the value specified.
2. Termination should include a series (~240Ω) and pull-up (connected to 1.5 V) resistors.
3. Signal should be at end of daisy chain and the boundary scan chain should be partitioned into two distinct sections to assist in debugging the system: one partition with only the processor(s) for system debug (i.e., used with the ITP) and another with all other components for manufacturing or system test.

**7.1.4.3 SIGNAL NOTE 7: TCK**

TCK is a high speed signal and should be routed accordingly. Follow all guidelines to assure the quality of the signal when beginning use of the ITP to debug the target.

Due to the number of loads on the TCK signal, special care should be taken when routing it. Poor routing can lead to multiple clocking of some agents on the debug chain, usually on the falling edge of TCK. This causes information to be lost through the chain and can result in bad commands being issued to some agents on the bus. Systems using other TCK routing schemes, particularly those with 'T' or 'Y' configurations where the trace from the source to the 'T' is long, could have signal integrity problems.

For detailed routing information, please refer to the applicable product datasheet.

**7.1.5 Debug Port Layout**

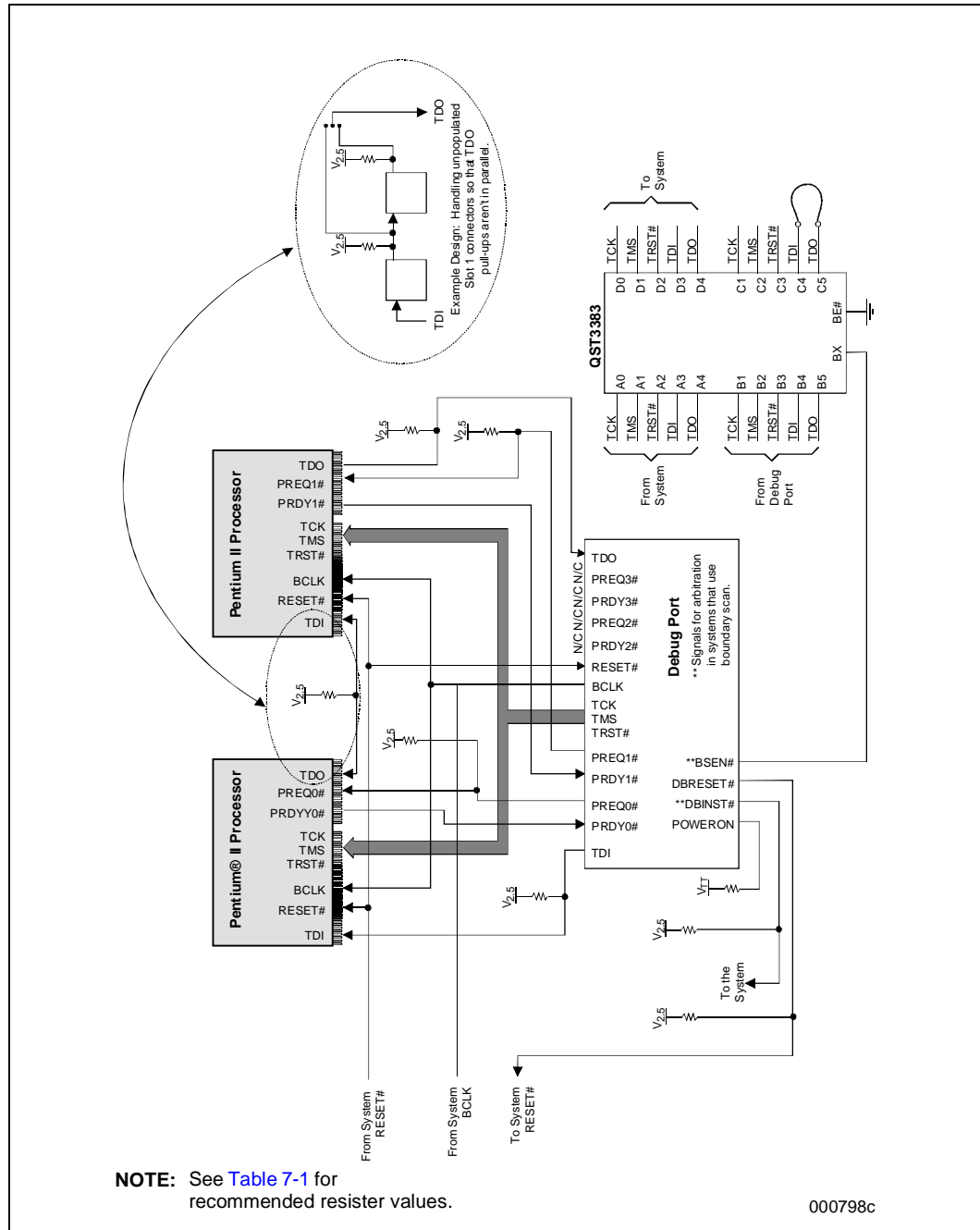
Figure 7-3 shows the simplest way to layout the debug port in a multiprocessor system. In this example, two processors are the only components in the system boundary scan chain. Systems incorporating boundary scan for use other than for the ITP should consider providing a method to partition the boundary scan chain in two distinct sections; one for system debug using the ITP, the other for manufacturing or system test.

System debug using the ITP requires only that the processors be in the boundary scan chain. During system debug, routing boundary scan signals (particularly TCK and TMS) solely to the processors enhances the likelihood that the boundary scan signals can be clocked at high speed. This will improve the performance of debug tools that must access large amounts of data via boundary scan. Additionally, removing all but the processors from the boundary scan chain reduces the possibility for errors in the chain when using the ITP for system debug.

If the system includes the use of boundary scan for test during normal system operation, then one should consider including a component such as the QST3383 in the layout. This component is used to multiplex the boundary scan lines in order to avoid contention between the system and the ITP. Using the QST3383, the system boundary scan lines are routed directly to the system when the ITP is not installed. However, if the ITP is installed and is communicating with the processors, the BSEN# signal will enable the multiplexer to pass the boundary scan lines from the debug port to the system.

**Note:** When the ITP is installed and communicating with the processors, the TDI line from the system boundary scan control logic does not pass through to the system, but is instead tied back into the TDO line. Thus, while the ITP is communicating with the processors, it is not possible for the system boundary scan control logic to access a processor via the boundary scan chain.

**Figure 7-3. Generic DP System Layout for Debug Port Connection**



### 7.1.5.1 SIGNAL QUALITY NOTES

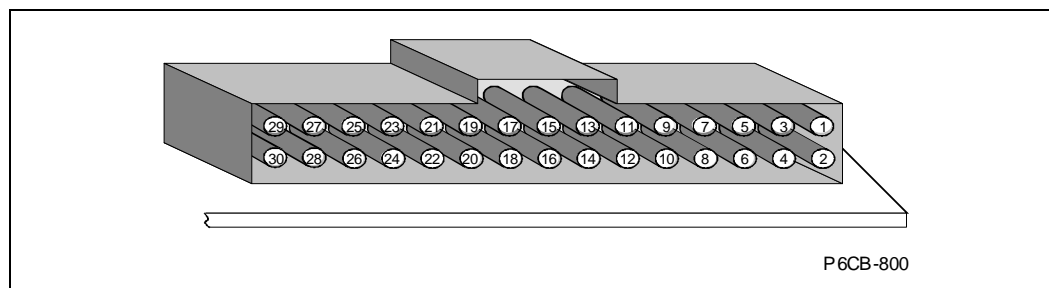
Signals from the debug port are fed to the system from an ITP via a buffer board and a cable. If system signals routed to the debug port (i.e., TDO, PRDY[x]# and RESET#) are used elsewhere in the system, then dedicated drivers should be used to isolate the signals from reflections coming from the end of this cable. If the P6 family processor boundary scan signals are used elsewhere in the system, then the TDI, TMS, TCK, and TRST# signals from the debug port should be isolated from the system signals with multiplexers as discussed earlier.

In general, no signals should be left floating. Thus, signals going from the debug port to the processor system should not be left floating. If they are left floating, there may be problems when the ITP is not plugged into the connector.

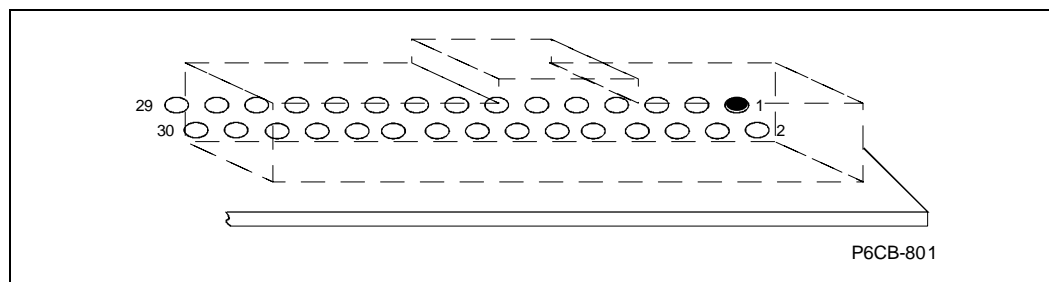
### 7.1.5.2 DEBUG PORT CONNECTOR

Figure 7-4 and Figure 7-5 illustrate how the debug port connector should be installed on a circuit board. Note the way the pins are numbered on the connector and how the through holes are positioned on the board. Figure 7-5 also shows a dotted line representation of the connector and behind it the through holes as seen from the top side of the circuit board (the side on which the connector will be placed). The through holes are shown so that the pin numbers of the connector leads will fall on the circuit board. Although this may appear very simple, it is surprising how often mistakes are made in this aspect of the debug port layout.

**Figure 7-4. Debug Port Connector on Thermal Plate Site of Circuit Board**



**Figure 7-5. Hole Positioning for Connector on Thermal Plate Side of Circuit Board**

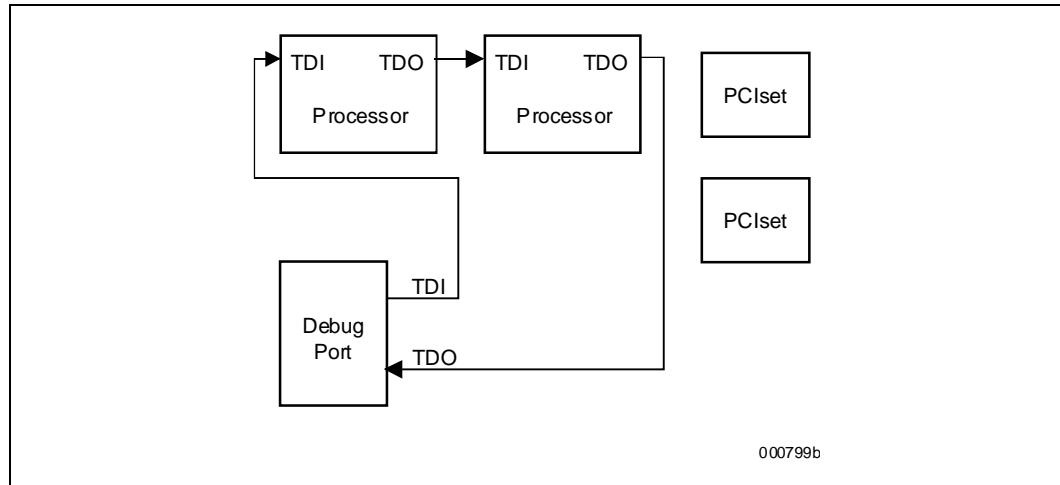


## 7.1.6 Using Boundary Scan to Communicate to the Processor

An ITP communicates to the P6 family processor by stopping its execution and sending/receiving messages over boundary scan pins. As long as each processor is tied into the system boundary scan chain, the ITP can communicate with it. In the simplest case, the processors are back to back in the

scan chain, with the boundary scan input (TDI) of the first processor connected up directly to the pin labeled TDI on the debug port and the boundary scan output of the last processor connected up to the pin labeled TDO on the debug port as shown in [Figure 7-6](#).

**Figure 7-6. Processor System Where Boundary Scan Is Not Used**





This appendix provides an alphabetical listing of all standard P6 family processor 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 applicable product datasheet.

## A.1 ALPHABETICAL SIGNALS LISTING

### A.1.1 A[35:3]# (I/O)

The A[35:3]# (Address) signals define a  $2^{36}$ -byte physical memory address space. See datasheets for actual address lines implemented. When ADS# is active, these pins transmit the address of a transaction; when ADS# is inactive, these pins transmit transaction type information. These signals must connect the appropriate pins of all agents on the P6 family processor system bus. The A[35:24]# signals are parity-protected by the AP1# parity signal, and the A[23:3]# signals are parity-protected by the AP0# parity signal.

On the active-to-inactive transition of RESET#, the processors sample the A[35:3]# pins to determine their power-on configuration.

### A.1.2 A20M# (I)

If the A20M# (Address-20 Mask) input signal is asserted, the P6 family processor masks physical address bit 20 (A20#) before looking up a line in any internal cache and before driving a read/write transaction on the bus. Asserting A20M# emulates the 8086 processor's address wrap-around at the 1-Mbyte boundary. Assertion of A20M# is only supported in real mode.

A20M# is an asynchronous signal. However, to ensure recognition of this signal following an I/O write instruction, it must be valid along with the TRDY# assertion of the corresponding I/O Write bus transaction.

During active RESET#, each processor begins sampling the A20M#, IGNNE#, and LINT[1:0] values to determine the ratio of core-clock frequency to bus-clock frequency. (See Table 5-6.) On the active-to-inactive transition of RESET#, each processor latches these signals and freezes the frequency ratio internally. System logic must then release these signals for normal operation.

### A.1.3 ADS# (I/O)

The ADS# (Address Strobe) signal is asserted to indicate the validity of the transaction address on the A[35:3]# pins. All bus agents observe the ADS# activation to begin parity checking, protocol checking, address decode, internal snoop, or deferred reply ID match operations associated with the new transaction.

### A.1.4 AERR# (I/O)

The AERR# (Address Parity Error) signal can be observed and driven by all system bus agents, and if used, must connect the appropriate pins on all bus agents. AERR# observation is optionally enabled during power-on configuration; if enabled, a valid assertion of AERR# aborts the current transaction.

If AERR# observation is disabled during power-on configuration, a central agent may handle an assertion of AERR# as appropriate to the Machine Check Architecture (MCA) of the system.

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

The AP[1:0]# (Address Parity) signals can be driven by the request initiator along with ADS#, A[35:3]#, REQ[4:0]#, and RP#. AP1# covers A[35:24]#, and AP0# covers A[23:3]#. A correct parity signal is high if an even number of covered signals are low and low if an odd number of covered signals are low. This allows parity to be high when all the covered signals are high. AP[1:0]# should connect the appropriate pins of all P6 family processor system bus agents.

### A.1.6 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 REQ[4:3]# pins. The ASZ[1:0]# signals are valid only when REQ[1:0]# signals equal 01B, 10B, or 11B, indicating a memory access transaction. The ASZ[1:0]# decode is defined in Table A-1

**Table A-1. ASZ[1:0]# Signal Decode**

| ASZ[1:0]# |   | Description              |
|-----------|---|--------------------------|
| 0         | 0 | 0 <= A[35:3]# < 4 GB     |
| 0         | 1 | 4 GB <= A[35:3]# < 64 GB |
| 1         | x | Reserved                 |

If the memory access is within the 0-to-(4 GByte -1) address space, ASZ[1:0]# must be 00B. If the memory access is within the 4 Gbyte-to-(64 GByte -1) address space, ASZ[1:0]# must be 01B. All observing bus agents that support the 4 Gbyte (32 bit) address space must respond to the transaction only when ASZ[1:0]# equals 00. All observing bus agents that support the 64 GByte (36 bit) address space must respond to the transaction when ASZ[1:0]# equals 00B or 01B

### A.1.7 ATTR[7:0]# (I/O)

The ATTR[7:0]# signals are the attribute signals. They are driven by the request initiator during the second Request Phase clock on the Ab[31:24]# pins. The ATTR[7:0]# signals are valid for all transactions. The ATTR[7:3]# are reserved and undefined. The ATTR[2:0]# are driven based on the Memory Range Register attributes and the Page Table attributes. Table A-2 defines ATTR[3:0]# signals.

**Table A-2. ATTR[7:0]# Field Descriptions**

| ATTR[7:3]#   | ATTR[2]#                 | ATTR[1:0]# |              |              |             |
|--------------|--------------------------|------------|--------------|--------------|-------------|
| Reserved (0) | Potentially Speculatable | 11         | 10           | 01           | 00          |
|              |                          | WriteBack  | WriteProtect | WriteThrough | UnCacheable |

### A.1.8 BCLK (I)

The BCLK (Bus Clock) signal determines the bus frequency. All P6 family processor system bus agents must receive this signal to drive their outputs and latch their inputs on the BCLK rising edge.

Most external timing parameters are specified with respect to the BCLK signal.

### A.1.9 BE[7:0]# (I/O)

The BE[7:0]# signals are the byte-enable signals. They are driven by the request initiator during the **second** Request Phase clock on the A[15:8]# pins. These signals carry various information depending on the REQ[4:0]# value.

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

For Special transactions ((REQ[4:0]# = 01000B) and (REQ[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 BE[7:0]#**

| BE[7:0]#                    | Special Cycle          |
|-----------------------------|------------------------|
| 0000 0000                   | Reserved               |
| 0000 0001                   | Shutdown               |
| 0000 0010                   | Flush                  |
| 0000 0011                   | Halt                   |
| 0000 0100                   | Sync                   |
| 0000 0101                   | Flush Acknowledge      |
| 0000 0110                   | Stop Clock Acknowledge |
| 0000 0111                   | SMI Acknowledge        |
| 0000 1000 through 1111 1111 | Reserved               |

### A.1.10 BERR# (I/O)

The BERR# (Bus Error) signal can be asserted to indicate an unrecoverable error without a bus protocol violation. It may be driven by any system bus agents, and must connect the appropriate pins of all such agents, if used. However, P6 family processors do not observe assertions of the BERR# signal. This function is an expected chipset function.

BERR# assertion conditions are configurable at a system level. Assertion options are defined by the following options:

- Enabled or disabled.
- Asserted optionally for internal errors along with IERR#.
- Asserted optionally by the request initiator of a bus transaction after it observes an error.
- Asserted by any bus agent when it observes an error in a bus transaction.

### A.1.11 **BINIT# (I/O)**

The BINIT# (Bus Initialization) signal may be observed and driven by all P6 family processor system bus agents, and if used must connect the appropriate pins of all such agents. If the BINIT# driver is enabled during power on configuration, BINIT# is asserted to signal any bus condition that prevents reliable future information.

If BINIT# observation is enabled during power-on configuration, and BINIT# is sampled asserted, all bus state machines are reset and any data which was in transit is lost. All agents reset their rotating ID for bus arbitration to the state after reset, and internal count information is lost. The L1 and L2 caches are not affected.

If BINIT# observation is disabled during power-on configuration, a central agent may handle an assertion of BINIT# as appropriate to the Machine Check Architecture (MCA) of the system.

### A.1.12 **BNR# (I/O)**

The BNR# (Block Next Request) signal is used to assert a bus stall by any bus agent that is unable to accept new bus transactions. 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 activated on specific clock edges and sampled on specific clock edges.

### A.1.13 **BP[3:2]# (I/O)**

The BP[3:2]# (Breakpoint) signals are outputs from the processor that indicate the status of breakpoints.

### A.1.14 **BPM[1:0]# (I/O)**

The BPM[1:0]# (Breakpoint Monitor) signals are breakpoint and performance monitor signals. They are outputs from the processor which indicate the status of breakpoints and programmable counters used for monitoring processor performance. For more information, please refer to the *Intel Architecture Software Developer Manual, Volume 3: System Programming Guide*.

### A.1.15 **BPRI# (I)**

The BPRI# (Bus Priority Request) signal is used to arbitrate for ownership of the P6 family processor system bus. It must connect the appropriate pins of all P6 family processor system bus agents. Observing BPRI# active (as asserted by the priority agent) causes all other agents to stop

issuing new requests, unless such requests are part of an ongoing locked operation. The priority agent keeps BPRI# asserted until all of its requests are completed, then releases the bus by deasserting BPRI#.

### A.1.16 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.

For a two processor system, the BR0# and BR1# (Bus Request) pins drive the BREQ[1:0]# signals in the system. The BREQ[1:0]# signals are interconnected in a rotating manner to individual processor pins.

During power-up configuration, the central agent must assert the BR0# bus signal. All symmetric agents sample their BR[1:0]# pins on active-to-inactive transition of RESET#. The pin on which the agent samples an active level determines its agent ID. All agents then configure their pins to match the appropriate bus signal protocol, as shown in [Table 5-4](#).

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 inactive), or the current symmetric owner de-asserts BREQ $m$ # to release the bus ownership to a new bus owner  $n$ . On a new arbitration event, based on BREQ[3:0]#, and the rotating ID, all symmetric agents simultaneously determine the new symmetric owner. 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 active. On sampling active BREQ $m$ # asserted by another symmetric agent, the symmetric owner de-asserts 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 upon observing BPRI# active.

A symmetric agent can not deassert BREQ $n$ # until it becomes a symmetric owner. A symmetric agent can reassert BREQ $n$ # after keeping it inactive for one clock.

On observation of active AERR#, RESET#, or BINIT#, the BREQ[3:0]# signals must be deasserted in the next clock. BREQ[3:0]# can be reasserted in the clock after sampling the RESET# active-to-inactive transition or three clocks after sampling BINIT# active and RESET# inactive. On AERR# assertion, if bus agent  $n$  is in the middle of a bus-locked operation, BREQ $n$ # must be re-asserted after two clocks, otherwise BREQ[3:0]# must stay inactive for at least 4 clocks.

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

The D[63:0]# (Data) signals are the data signals. These signals provide a 64-bit data path between the P6 family processor system bus agents, and must connect the appropriate pins on all such agents. The data driver asserts DRDY# to indicate a valid data transfer.

### A.1.18 DBSY# (I/O)

The DBSY# (Data Bus Busy) signal is asserted by the agent responsible for driving data on the P6 family processor system bus to indicate that the data bus is in use. The data bus is released after DBSY# is deasserted. This signal must connect the appropriate pins on all P6 family processor system bus agents.

### A.1.19 DEFER# (I)

The DEFER# signal is asserted by an agent to indicate that a transaction must be retried. Assertion of DEFER# is normally the responsibility of the addressed memory or I/O agent. This signal must connect the appropriate pins of all P6 family processor system bus agents.

### A.1.20 DEN# (I/O)

The DEN# signal is the defer-enable signal. It is driven to the bus on the **second** clock of the Request Phase on the A[4]# pin. DEN# is asserted to indicate that the transaction can be deferred by the responding agent.

### A.1.21 DEP[7:0]# (I/O)

The DEP[7:0]# (Data Bus ECC Protection) signals provide optional ECC protection for the data bus. They are driven by the agent responsible for driving D[63:0]#, and must connect the appropriate pins of all P6 family processor system bus agents which use them. The DEP[7:0]# signals are enabled or disabled for ECC protection during power on configuration.

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

The DID[7:0]# signals are Deferred Identifier signals. They are transferred using the A[23:16]# pins by the request initiator. They are transferred on during the **second** clock of the Request Phase on all transactions, but only defined for deferrable transactions (DEN# asserted). DID[7:0]# is also transferred on A[23:16]# during the **first** clock of the Request Phase for Deferred Reply transactions.

The deferred identifier defines the token supplied by the request initiator. DID[7:4]# carry the request initiators' agent identifier and DID[3:0]# carry a transaction identifier associated with the request. This configuration limits the bus specification to 16 bus masters with each one of the bus masters capable of making up to sixteen requests.

Every deferrable transaction issued on the P6 family processor bus which has not been guaranteed completion (has not successfully passed its Snoop Result Phase) will have a unique Deferred ID. This includes all outstanding transactions which have not had their snoop result reported, or have had their snoop results deferred. After a deferrable transaction passes its Snoop Result Phase without DEFER# asserted, its Deferred ID may be reused. Similarly, the deferred ID of a transaction which was deferred may be reused after the completion of the snoop window of the deferred reply.

DID[7]# indicates the agent type. Symmetric agents use 0. Priority agents use 1. DID[6:4]# indicates the agent ID. Symmetric agents use their arbitration ID. P6 family processors have at most four symmetric agents, and do not assert DID[6]#. DID[3:0]# indicates the transaction ID for

an agent. The transaction ID must be unique for all transactions issued by an agent which have not reported their snoop results. P6 family processors create at most four open transactions per processor.

**Table A-4. DID[7:0]# Encoding**

| DID[7]#    | DID[6:4]# | DID[3:0]#      |
|------------|-----------|----------------|
| Agent Type | Agent ID  | Transaction ID |

The Deferred Reply agent transmits the DID[7:0]# signals received during the second clock of the original transaction on the A[23:16]# signals during the first clock of the Deferred Reply transaction. This process enables the original request initiator to make an identifier match and wake up the original request waiting for completion.

### A.1.23 DRDY# (I/O)

The DRDY# (Data Ready) signal is asserted by the data driver on each data transfer, indicating valid data on the data bus. In a multi-cycle data transfer, DRDY# may be deasserted to insert idle clocks. This signal must connect the appropriate pins of all P6 family processor system bus agents.

### A.1.24 DSZ[1:0]# (I/O)

The DSZ[1:0]# signals are the data-size signals. They are transferred on REQ[4:3]# pins in the **second** clock of Request Phase by the requesting agent. The DSZ[1:0]# signals define the data transfer capability of the requesting agent. For the P6 family processor, DSZ#= 00, always.

### A.1.25 EXF[4:0]# (I/O)

The EXF[4:0]# signals are the Extended Function signals. They are transferred on the A[7:3]# pins by the request initiator 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-5](#).

**Table A-5. EFX[4:0]# Signal Definitions**

| EXF   | Name     | Extended Functionality | When Activated  |
|-------|----------|------------------------|---|
| EXF#4 | SMMEM#   | SMM Mode               | After entering SMM mode   |
| EXF#3 | SPLCK#   | Split Lock             | The first transaction of a bus lock operation                     |
| EXF#2 | Reserved | Reserved               |   |
| EXF#1 | DEN#     | Defer Enable           | The transactions for which Defer or Retry Response is acceptable. |
| EXF0# | Reserved | Reserved               |   |

### A.1.26 FERR# (O)

The FERR# (Floating-point Error) signal is asserted when the processor detects an unmasked floating-point error. FERR# is similar to the ERROR# signal on the Intel 387 coprocessor, and is included for compatibility with systems using MS-DOS\*-type floating-point error reporting.

### A.1.27 FLUSH# (I)

When the FLUSH# input signal is asserted, processors write back all data in the Modified state from their internal caches and invalidate all internal cache lines. At the completion of this operation, the processor issues a Flush Acknowledge transaction. The processor does not cache any new data while the FLUSH# signal remains asserted.

FLUSH# is an asynchronous signal. However, to ensure recognition of this signal following an I/O write instruction, it must be valid along with the TRDY# assertion of the corresponding I/O Write bus transaction.

On the active-to-inactive transition of RESET#, each processor samples FLUSH# to determine its power-on configuration.

### A.1.28 FRCERR (I/O)

If two processors are configured in a Functional Redundancy Checking (FRC) master/checker pair, as a single “logical” processor, the FRCERR (Functional Redundancy Checking Error) signal is asserted by the checker if a mismatch is detected between the internally sampled outputs and the master’s outputs. The checker’s FRCERR output pin must be connected with the master’s FRCERR input pin in this configuration.

For point-to-point connections, the checker always compares against the master’s outputs. For bussed single-driver signals, the checker compares against the signal when the master is the only allowed driver. For bussed multiple-driver wired-OR signals, the checker compares against the signal only if the master is expected to drive the signal low.

When a processor is configured as an FRC checker, FRCERR is toggled during its reset action. A checker asserts FRCERR for approximately 1 second after the active-to-inactive transition of RESET# if it executes its Built-In Self-test (BIST). When BIST execution completes, the checker processor deasserts FRCERR if BIST completed successfully, and continues to assert FRCERR if BIST fails. If the checker processor does not execute the BIST action, then it keeps FRCERR asserted for approximately 20 clocks and then deasserts it.

All asynchronous signals must be externally synchronized to BCLK by system logic during FRC mode operation.

### A.1.29 HIT# (I/O), HITM# (I/O)

The HIT# (Snoop Hit) and HITM# (Hit Modified) signals convey transaction snoop operation results, and must connect the appropriate pins of all P6 family processor system bus agents. Any such agent may assert both HIT# and HITM# together to indicate that it requires a snoop stall, which can be continued by reasserting HIT# and HITM# together.

### A.1.30 IERR# (O)

The IERR# (Internal Error) signal is asserted by a processor as the result of an internal error. Assertion of IERR# is usually accompanied by a SHUTDOWN transaction on the P6 family processor system bus. This transaction may optionally be converted to an external error signal (e.g., NMI) by system core logic. The processor will keep IERR# asserted until it is handled in software, or with the assertion of RESET#, BINIT#, or INIT#.

### A.1.31 IGNNE# (I)

The IGNNE# (Ignore Numeric Error) signal is asserted to force the processor to ignore a numeric error and continue to execute non-control floating-point instructions. If IGNNE# is deasserted, the processor generates an exception on a non-control floating-point instruction if a previous floating-point instruction caused an error. IGNNE# has no effect when the NE bit in control register 0 is set.

IGNNE# is an asynchronous signal. However, to ensure recognition of this signal following an I/O write instruction, it must be valid along with the TRDY# assertion of the corresponding I/O Write bus transaction.

During active RESET#, the P6 family processor begins sampling the A20M#, IGNNE#, and LINT[1:0] values to determine the ratio of core-clock frequency to bus-clock frequency. (See [Table 5-7](#).) On the active-to-inactive transition of RESET#, the P6 family processor latches these signals and freezes the frequency ratio internally. System logic must then release these signals for normal operation.

### A.1.32 INIT# (I)

The INIT# (Initialization) signal, when asserted, resets integer registers inside all processors without affecting their internal (L1 or L2) caches or floating-point registers. Each processor then begins execution at the power-on reset vector configured during power-on configuration. The processor continues to handle snoop requests during INIT# assertion. INIT# is an asynchronous signal and must connect the appropriate pins of all P6 family processor system bus agents.

If INIT# is sampled active on the active to inactive transition of RESET#, then the processor executes its Built-In Self-test (BIST).

### A.1.33 INTR(I)

The INTR signal is the Interrupt Request signal. The INTR input indicates that an external interrupt has been generated. The interrupt is maskable using the IF bit in the EFLAGS register. If the IF bit is set, the P6 family processor vectors to the interrupt handler after the current instruction execution is completed. Upon recognizing the interrupt request, the P6 family processor issues a single Interrupt Acknowledge (INTA) bus transaction. INTR must remain active until the INTA bus transaction to guarantee its recognition.

INTR is sampled on every rising BCLK edge. INTR is an asynchronous input but recognition of INTR is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. INTR must also be deasserted for a minimum of two clocks to guarantee its inactive recognition. In FRC mode, INTR must be synchronous to BCLK. On power-up the LINT[1:0] signals are used for power-on-configuration of clock ratios. Both these signals must be software configured by programming the APIC register space to be used either as NMI/INTR or LINT[1:0] in the BIOS. Because APIC is enabled after reset, LINT[1:0] is the default configuration.

### A.1.34 LEN[1:0]# (I/O)

The LEN[1:0]# signals are data-length signals. They are transmitted using REQb[1:0]# signals by the request initiator in the second clock of Request Phase. LEN[1:0]# define the length of the data transfer requested by the initiator as defined in [Table A-8](#). The LEN[1:0]#, HITM#, and RS[2:0]# signals together define the length of the actual data transfer.

**Table A-6. LEN[1:0]# Signals Data Transfer Lengths**

| LEN[1:0]# | Request Initiator's Data Transfer Length |
|-----------|--|
| 00        | 0-8 Bytes                                |
| 01        | 16 Bytes                                 |
| 10        | 32 Bytes                                 |
| 11        | Reserved                                 |

### A.1.35 LINT[1:0] (I)

The LINT[1:0] (Local APIC Interrupt) signals must connect the appropriate pins of all APIC Bus agents, including all processors and the core logic or I/O APIC component. When the APIC is disabled, the LINT0 signal becomes INTR, a maskable interrupt request signal, and LINT1 becomes NMI, a non-maskable interrupt. INTR and NMI are backward compatible with the signals of those names on the Pentium processor. Both signals are asynchronous.

Both of these signals must be software configured via BIOS programming of the APIC register space to be used either as NMI/INTR or LINT[1:0]. Because the APIC is enabled by default after reset, operation of these pins as LINT[1:0] is the default configuration.

During active RESET#, the P6 family processor begins sampling the A20M#, IGNNE#, and LINT[1:0] values to determine the ratio of core-clock frequency to bus-clock frequency. (See Table 5-6.) On the active-to-inactive transition of RESET#, the P6 family processor latches these signals and freezes the frequency ratio internally. System logic must then release these signals for normal operation; see Figure 7-4 for an example implementation of this logic.

### A.1.36 LOCK# (I/O)

The LOCK# signal indicates to the system that a transaction must occur atomically. This signal must connect the appropriate pins of all processor system bus agents. For a locked sequence of transactions, LOCK# is asserted from the beginning of the first transaction to the end of the last transaction. The LOCK# signal is always deasserted between the sequences of locked transactions on the P6 family processor system bus.

When the priority agent asserts BPRI# to arbitrate for ownership of the P6 family processor system bus, it will wait until it observes LOCK# deasserted. This enables symmetric agents to retain ownership of the P6 family processor system bus throughout the bus locked operation and ensure the atomicity of lock.

When the lock prefix is prefixed to an instruction and the memory area being accessed is writeback memory and is completely contained in the cache line in the processor, the LOCK# signal is generally not asserted. Instead, only the processor's cache is locked. Here, the processor's cache coherency mechanism insures that the operation is carried out atomically with regards to memory.

Bit 31 of the Model Specific Register (MSR) at address 33h to 1 will prevent LOCK# from being asserted when locked transactions, which are split across a cache line boundary, are issued from the processor. See the *Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide*, for more details on the LOCK# function.

### A.1.37 NMI(I)

The NMI signal is the Non-maskable Interrupt signal. It is the state of the LINT1 signal when APIC is disabled. 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 IRET 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, active and inactive pulse widths must be a minimum of two clocks. In FRC mode, NMI must be synchronous to BCLK.

### A.1.38 PICCLK (I)

The PICCLK (APIC Clock) signal is an input clock to the processor and core logic or I/O APIC which is required for operation of all processors, core logic, and I/O APIC components on the APIC bus. During FRC mode operation, PICCLK must be 1/4 of (and synchronous to) BCLK.

### A.1.39 PICD[1:0] (I/O)

The PICD[1:0] (APIC Data) signals are used for bi-directional serial message passing on the APIC bus, and must connect the appropriate pins of all processors and core logic or I/O APIC components on the APIC bus.

### A.1.40 PRDY# (O)

The PRDY# (Probe Ready) signal is a processor output used by debug tools to determine processor debug readiness.

### A.1.41 PREQ# (I)

The PREQ# (Probe Request) signal is used by debug tools to request debug operation of the processors.

### A.1.42 PWRGOOD (I)

The PWRGOOD (Power Good) signal is a 2.5 V tolerant processor input. The processor requires this signal to be a clean indication that the clocks and power supplies ( $V_{CCCORE}$ , etc.) are stable and within their specifications. Clean implies that the signal will remain low (capable of sinking leakage current), without glitches, from the time that the power supplies are turned on until they come within specification. The signal must then transition monotonically to a high (2.5 V) state. [Figure A-1](#) illustrates the relationship of PWRGOOD to other system signals. PWRGOOD can be driven inactive at any time, but clocks and power must again be stable before a subsequent rising edge of PWRGOOD. It must also meet the minimum pulse width specification and be followed by a 1 ms RESET# pulse.

The PWRGOOD signal must be supplied to the processor as it is used to protect internal circuits against voltage sequencing issues. The PWRGOOD signal does not need to be synchronized for FRC operation. It should be driven high throughout boundary scan operation.

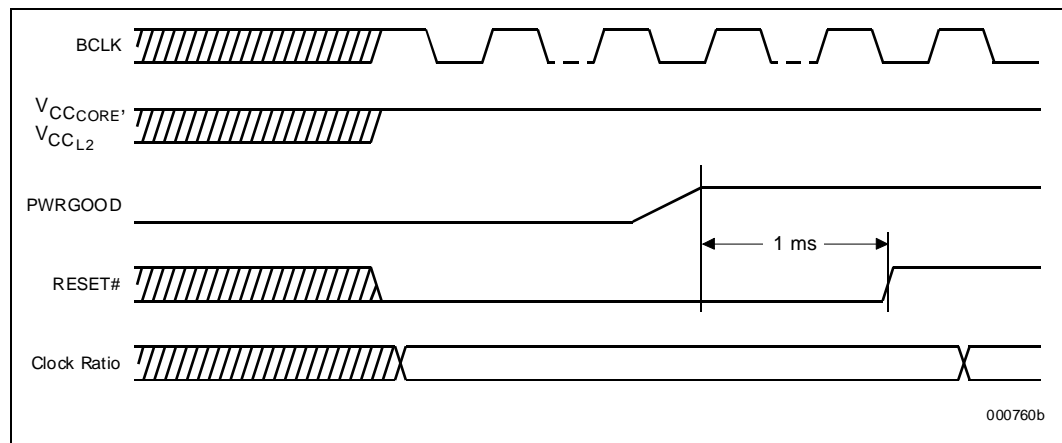
### A.1.43 REQ[4:0]# (I/O)

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

All receiving agents observe the REQ[4:0]# signals to determine the transaction type and participate in the transaction as necessary, as shown in Table A-7.

**Table A-7. Transaction Types Defined by REQ# Signals**

| Transaction                            | REQ[4:0]# (First Clock) |   |   |         |   | REQ[4:0]# (Second Clock) |   |   |      |   |
|--|-------------------------|---|---|---------|---|--------------------------|---|---|------|---|
|  | 4                       | 3 | 2 | 1       | 0 | 4                        | 3 | 2 | 1    | 0 |
| Deferred Reply                         | 0                       | 0 | 0 | 0       | 0 | x                        | x | x | x    | x |
| Rsvd ( <i>Ignore</i> )                 | 0                       | 0 | 0 | 0       | 1 | x                        | x | x | x    | x |
| Interrupt Acknowledge                  | 0                       | 1 | 0 | 0       | 0 | DSZ#                     |   | x | 0    | 0 |
| Special Transactions                   | 0                       | 1 | 0 | 0       | 0 | DSZ#                     |   | x | 0    | 1 |
| Rsvd ( <i>Central agent response</i> ) | 0                       | 1 | 0 | 0       | 0 | DSZ#                     |   | x | 1    | x |
| Branch Trace Message                   | 0                       | 1 | 0 | 0       | 1 | DSZ#                     |   | x | 0    | 0 |
| Rsvd ( <i>Central agent response</i> ) | 0                       | 1 | 0 | 0       | 1 | DSZ#                     |   | x | 0    | 1 |
| Rsvd ( <i>Central agent response</i> ) | 0                       | 1 | 0 | 0       | 1 | DSZ#                     |   | x | 1    | x |
| I/O Read                               | 1                       | 0 | 0 | 0       | 0 | DSZ#                     |   | x | LEN# |   |
| I/O Write                              | 1                       | 0 | 0 | 0       | 1 | DSZ#                     |   | x | LEN# |   |
| Rsvd ( <i>Ignore</i> )                 | 1                       | 1 | 0 | 0       | x | DSZ#                     |   | x | x    | x |
| Memory Read & Invalidate               | ASZ#                    |   | 0 | 1       | 0 | DSZ#                     |   | x | LEN# |   |
| Rsvd ( <i>Memory Write</i> )           | ASZ#                    |   | 0 | 1       | 1 | DSZ#                     |   | x | LEN# |   |
| Memory Code Read                       | ASZ#                    |   | 1 | D/C#=0  | 0 | DSZ#                     |   | x | LEN# |   |
| Memory Data Read                       | ASZ#                    |   | 1 | D/C#=1  | 0 | DSZ#                     |   | x | LEN# |   |
| Memory Write (may not be retried)      | ASZ#                    |   | 1 | W/WB#=0 | 1 | DSZ#                     |   | x | LEN# |   |
| Memory Write (may be retried)          | ASZ#                    |   | 1 | W/WB#=1 | 1 | DSZ#                     |   | x | LEN# |   |

**Figure A-1. PWRGOOD Relationship at Power-On**


#### A.1.44 RESET# (I)

Asserting the RESET# signal resets all processors to known states and invalidates their L1 and L2 caches without writing back any of their contents. RESET# must remain active for one microsecond for a “warm” reset; for a power-on reset, RESET# must stay active for at least one millisecond after V<sub>CC</sub>CORE and CLK have reached their proper specifications. On observing active RESET#, all P6 family processor system bus agents will deassert their outputs within two clocks.

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

The processor may have its outputs tristated via power-on configuration. Otherwise, if INIT# is sampled active during the active-to-inactive transition of RESET#, the processor will execute its Built-In Self-test (BIST). Whether or not BIST is executed, the processor will begin program execution at the reset-vector (default 0\_FFFF\_FFF0h). RESET# must connect the appropriate pins of all P6 family processor system bus agents.

#### A.1.45 RP# (I/O)

On processor's that support bus parity, the RP# (Request Parity) signal is driven by the request initiator, and provides parity protection on ADS# and REQ[4:0]#. It must connect the appropriate pins of all P6 family processor system bus agents.

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

The RS[2:0]# (Response Status) signals are driven by the response agent (the agent responsible for completion of the current transaction), and must connect the appropriate pins of all P6 family processor system bus agents.

### A.1.47 RSP# (I)

On processor's that support bus parity, the RSP# (Response Parity) signal is driven by the response agent (the agent responsible for completion of the current transaction) during assertion of RS[2:0]#, the signals for which RSP# provides parity protection. It must connect the appropriate pins of all P6 family processor system bus agents.

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. While RS[2:0]# = 000, RSP# is also high, since this indicates it is not being driven by any agent guaranteeing correct parity.

### A.1.48 SLP# (I)

This signal exists on processor's that support sleep state. The SLP# (Sleep) signal, when asserted in Stop Grant state, causes processors to enter the Sleep state. During Sleep state, the processor stops providing internal clock signals to all units, leaving only the Phase-Locked Loop (PLL) still operating. Processors in this state will not recognize snoops or interrupts. The processor will recognize only assertions of the SLP#, STPCLK#, and RESET# signals while in Sleep state. If SLP# is deasserted, the processor exits Sleep state and returns to Stop Grant state, restarting its internal clock signals to the bus and APIC processor core units.

### A.1.49 SMI# (I)

The SMI# (System Management Interrupt) signal is asserted asynchronously by system logic. On accepting a System Management Interrupt, processors save the current state and enter System Management Mode (SMM). An SMI Acknowledge transaction is issued, and the processor begins program execution from the SMM handler.

### A.1.50 SMMEM# (I/O)

The SMMEM# signal is the System Management Mode Memory signal. It is driven on the second clock of the Request Phase on the EXF4#/Ab7# signal. It is asserted by the processor to indicate that the processor is in System Management Mode and is executing out of SMRAM space.

### A.1.51 SPLCK# (I/O)

The SPLCK# signal is the Split Lock signal. It is driven in the second clock of the Request Phase of the EXF[3]#/Ab[6]# signal of the first transaction of a locked operation. It is driven to indicate that the locked operation will consist of four locked transactions. Note that SPLCLK# is asserted only for locked operations and only in the first transaction of the locked operation.

### A.1.52 STPCLK# (I)

The STPCLK# (Stop Clock) signal, when asserted, causes processors to enter a low power Stop Grant state. The processor issues a Stop Grant Acknowledge transaction, and stops providing internal clock signals to all processor core units except the bus and APIC units. The processor continues to snoop bus transactions and service interrupts while in Stop Grant state. When STPCLK# is deasserted, the processor restarts its internal clock to all units and resumes execution. The assertion of STPCLK# has no effect on the bus clock.

STPCLK# is an asynchronous input. In FRC mode, STPCLK# must be synchronous to BCLK.

**A.1.53 TCK (I)**

The TCK (Test Clock) signal provides the clock input for the P6 family processor Test Bus (also known as the Test Access Port).

**A.1.54 TDI (I)**

The TDI (Test Data In) signal transfers serial test data into the P6 family processor. TDI provides the serial input needed for JTAG specification support.

**A.1.55 TDO (O)**

The TDO (Test Data Out) signal transfers serial test data out of the P6 family processor. TDO provides the serial output needed for JTAG specification support.

**A.1.56 THERMTRIP# (O)**

The processor protects itself 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. The processor will stop all execution when the junction temperature exceeds the sensor setting. This is signaled to the system by the THERMTRIP# (Thermal Trip) pin. Once activated, the signal remains latched, and the processor stopped, until RESET# goes active. There is no hysteresis built into the thermal sensor itself; as long as the die temperature drops below the trip level, a RESET# pulse will reset the processor and execution will continue. If the temperature has not dropped below the trip level, the processor will continue to drive THERMTRIP# and remain stopped.

**A.1.57 TMS (I)**

The TMS (Test Mode Select) signal is a JTAG specification support signal used by debug tools.

**A.1.58 TRDY# (I)**

The TRDY# (Target Ready) signal is asserted by the target to indicate that it is ready to receive a write or implicit writeback data transfer. TRDY# must connect the appropriate pins of all P6 family processor system bus agents.

**A.1.59 TRST# (I)**

TRST# (TAP Reset) must be driven low during power on reset. This can be accomplished with a pull-down resistor whose value varies from processor to processor.

## A.2 SIGNAL SUMMARIES

The following tables list attributes of the P6 family processor output, input and I/O signals.

**Table A-8. Output Signals**

| Name       | Active Level | Clock  | Signal Group |
|------------|--------------|--------|--------------|
| FERR#      | Low          | Asynch | CMOS Output  |
| IERR#      | Low          | Asynch | CMOS Output  |
| PRDY#      | Low          | BCLK   | GTL+ Output  |
| TDO        | High         | TCK    | JTAG Output  |
| THERMTRIP# | Low          | Asynch | CMOS Output  |

**NOTE:** Outputs are not checked in FRC mode.

**Table A-9. Input Signals**

| Name      | Active Level | Clock  | Signal Group1 | Qualified               |
|-----------|--------------|--------|---------------|-------------------------|
| A20M#     | Low          | Asynch | CMOS Input    | Always <sup>2</sup>     |
| BCLK      | High         | —      | Clock         | Always                  |
| BPRI#     | Low          | BCLK   | GTL+ Input    | Always                  |
| DEFER#    | Low          | BCLK   | GTL+ Input    | Always                  |
| FLUSH#    | Low          | Asynch | CMOS Input    | Always <sup>2</sup>     |
| IGNNE#    | Low          | Asynch | CMOS Input    | Always <sup>2</sup>     |
| INIT#     | Low          | Asynch | CMOS Input    | Always <sup>2</sup>     |
| INTR      | High         | Asynch | CMOS Input    | APIC disabled mode      |
| LINT[1:0] | High         | Asynch | CMOS Input    | APIC enabled mode       |
| NMI       | High         | Asynch | CMOS Input    | APIC disabled mode      |
| PICCLK    | High         | —      | APIC Clock    | Always                  |
| PREQ#     | Low          | Asynch | CMOS Input    | Always                  |
| PWRGOOD   | High         | Asynch | CMOS Input    | Always                  |
| RESET#    | Low          | BCLK   | GTL+ Input    | Always                  |
| RS[2:0]#  | Low          | BCLK   | GTL+ Input    | Always                  |
| RSP#      | Low          | BCLK   | GTL+ Input    | Always                  |
| SLP#      | Low          | Asynch | CMOS Input    | During Stop Grant state |
| SMI#      | Low          | Asynch | CMOS Input    |                         |
| STPCLK#   | Low          | Asynch | CMOS Input    |                         |
| TCK       | High         | —      | JTAG Input    |                         |
| TDI       | High         | TCK    | JTAG Input    |                         |
| TMS       | High         | TCK    | JTAG Input    |                         |
| TRDY#     | Low          | BCLK   | GTL+ Input    |                         |
| TRST#     | Low          | Asynch | JTAG Input    |                         |

**NOTES:**

1. All asynchronous input signals except PWRGOOD must be synchronous in FRC.
2. Synchronous assertion with active TRDY# ensures synchronization.

**Table A-10. Input/Output Signals (Single Driver)**

| Name       | Active Level | Clock | Signal Group | Qualified     |
|------------|--------------|-------|--------------|---------------|
| A[35:3]#   | Low          | BCLK  | GTL+ I/O     | ADS#, ADS##+1 |
| ADS#       | Low          | BCLK  | GTL+ I/O     | Always        |
| AP[1:0]#   | Low          | BCLK  | GTL+ I/O     | ADS#, ADS##+1 |
| ASZ[1:0]#  | Low          | BCLK  | GTL+ I/O     | ADS#          |
| ATTR[7:0]  | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| BE[7:0]#   | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| BP[3:2]#   | Low          | BCLK  | GTL+ I/O     | Always        |
| BPM[1:0]#  | Low          | BCLK  | GTL+ I/O     | Always        |
| BR0#       | Low          | BCLK  | GTL+ I/O     | Always        |
| BREQ[3:1]# | Low          | BCLK  | GTL+ I/O     | Always        |
| D[63:0]#   | Low          | BCLK  | GTL+ I/O     | DRDY#         |
| DBSY#      | Low          | BCLK  | GTL+ I/O     | Always        |
| DEN#       | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| DEP[7:0]#  | Low          | BCLK  | GTL+ I/O     | DRDY#         |
| DID[7:0]#  | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| DRDY#      | Low          | BCLK  | GTL+ I/O     | Always        |
| DSZ[1:0]#  | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| EXF[4:0]#  | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| FRCERR     | High         | BCLK  | GTL+ I/O     | Always        |
| LEN[1:0]#  | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| LOCK#      | Low          | BCLK  | GTL+ I/O     | Always        |
| REQ[4:0]#  | Low          | BCLK  | GTL+ I/O     | ADS#, ADS##+1 |
| RP#        | Low          | BCLK  | GTL+ I/O     | ADS#, ADS##+1 |
| SMMEM#     | Low          | BCLK  | GTL+ I/O     | ADS##+1       |
| SPLCK#     | Low          | BCLK  | GTL+ I/O     | ADS##+1       |

**Table A-11. Input/Output Signals (Multiple Drivers)**

| Name      | Active Level | Clock  | Signal Group | Qualified |
|-----------|--------------|--------|--------------|-----------|
| AERR#     | Low          | BCLK   | GTL+ I/O     | ADS##+3   |
| BERR#     | Low          | BCLK   | GTL+ I/O     | Always    |
| BINIT#    | Low          | BCLK   | GTL+ I/O     | Always    |
| BNR#      | Low          | BCLK   | GTL+ I/O     | Always    |
| HIT#      | Low          | BCLK   | GTL+ I/O     | Always    |
| HITM      | Low          | BCLK   | GTL+ I/O     | Always    |
| PICD[1:0] | High         | PICCLK | APIC I/O     | Always    |





# Index

## A

|   |                    |
|---|--------------------|
| Active Signal (definition of)                     | 1-2                |
| ADS#  | 3-4                |
| Advanced Programmable Interrupt Controller (APIC) | 3-3                |
| APIC Cluster                                      | 5-2                |
| APIC Cluster ID                                   | 5-4                |
| APIC Mode   | 5-4                |
| AERR#   | 3-7, 4-2, 5-2, 5-3 |
| AP[1:0]#  | 3-4                |
| Arbitration ID                                    |                    |
| Configuration (Four Agents)                       | 5-7                |
| Configuration (Two Agents)                        | 5-7                |
| Arbitration Signals                               | 3-3                |
| ASZ[1:0]# signals                                 | A-2                |
| Attribute signals                                 | A-2                |
| ATTR[7:0]# signals                                | A-2                |
| A20M#   | 3-9                |
| A[35:3]#  | 3-4                |

## B

|                                    |               |
|------------------------------------|---------------|
| BCLK                               | 3-2           |
| BERR#                              | 3-8, 5-2, 5-3 |
| BE[7:0]# signals                   | A-3           |
| BINIT#                             | 3-8, 5-2, 5-3 |
| BNR#                               | 3-4, 5-10     |
| Boundary Scan Chain                | 6-1, 7-8      |
| Boundary Scan Register             | 6-8           |
| BPM[1:0]#                          | 3-9           |
| BPRI#                              | 3-4           |
| BP[3:2]#                           | 3-9           |
| Branch Target Buffer (BTB)         | 2-5           |
| BREQ0#                             | 5-6           |
| BREQ1#                             | 5-6           |
| BREQ[3:0]#                         | 3-3, 5-5      |
| BREQ[3:0]# signals                 | A-5           |
| Built-In Self-test (BIST)          | 5-2           |
| BIST Result Boundary Scan Register | 6-8           |
| FRCERR                             | 3-8           |
| Burst Order                        | 3-6           |
| Bus                                |               |
| Request signals                    | A-5           |
| Bus Interface Unit                 | 2-6           |
| Bus Signal Protection              | 4-2           |
| BYPASS                             | 6-5           |
| Bypass Register                    | 6-7           |
| Byte Enable                        |               |
| Signals                            | A-3           |
| Special transaction encoding       | A-7           |

## C

|                       |     |
|-----------------------|-----|
| Clock Frequencies     | 5-8 |
| Compatibility Signals | 3-9 |

## D

|                             |          |
|-----------------------------|----------|
| Data Bus Error Checking     | 5-2      |
| Data Register Summary       | 6-7      |
| Data Registers              | 6-5      |
| Data Response Signals       | 3-6      |
| Data Transfer Signals       | 4-3      |
| Datasheets                  | 1-2      |
| DBRESET#                    | 7-3      |
| DBSY#                       | 3-6      |
| Defer-enable signal         | A-6      |
| Deferred identifier signals | A-6      |
| DEFER#                      | 3-5      |
| DEN# signal                 | A-6      |
| DEP[7:0]#                   | 3-6      |
| Device ID Register          | 6-7      |
| Diagnostic Signals          | 3-9      |
| Diagnostic Support Signals  | 3-10     |
| DID[7:0]# signals           | A-6      |
| Dispatch/Execute Unit       | 2-4      |
| DRDY#                       | 3-6      |
| Dynamic Execution           | 1-1, 2-2 |
| D[63:0]#                    | 3-6      |

## E

|                             |     |
|-----------------------------|-----|
| Errata                      | 1-2 |
| Error Classification        | 4-1 |
| Error Code Algorithms       | 4-3 |
| Error Correcting Code (ECC) | 3-6 |
| Error Signals               | 3-7 |
| Execution Control Signals   | 3-2 |

## F

|                                      |     |
|--------------------------------------|-----|
| Fatal Error (FE)                     | 4-2 |
| FERR#                                | 3-9 |
| Fetch/Decode Unit                    | 2-3 |
| FLUSH#                               | 3-2 |
| Functional Redundancy Checking (FRC) | 3-8 |
| Configurations                       | 5-1 |
| Data Integrity                       | 4-1 |
| FRC Mode                             | 5-2 |
| FRCERR                               | 3-8 |

## H

|                     |     |
|---------------------|-----|
| Hard-Error Response | 4-3 |
| HITM#               | 3-5 |
| HIT#                | 3-5 |

## I

|       |     |
|-------|-----|
| IERR# | 3-8 |
|-------|-----|

|                                |          |
|--------------------------------|----------|
| IGNNE# .....                   | 3-9      |
| INIT# .....                    | 3-2      |
| In-Order Queue (IOQ) .....     | 5-2      |
| Pipelining .....               | 5-4      |
| Instruction Register .....     | 6-4      |
| Instruction Set .....          | 6-6      |
| In-Target Probe (ITP) .....    | 7-1      |
| Debug Port Connector .....     | 7-1, 7-2 |
| Debug Port Layout .....        | 7-6      |
| Debug Port Signal .....        | 7-2      |
| Primary Function .....         | 7-1      |
| Integration Tools .....        | 7-1      |
| Intel Architecture .....       | 2-2      |
| Interrupt Request signal ..... | A-9      |
| INTR signal .....              | A-9      |

## J

|                                 |     |
|---------------------------------|-----|
| Jump Execution Unit (JEU) ..... | 2-5 |
|---------------------------------|-----|

## L

|                                |     |
|--------------------------------|-----|
| Latched Bus Protocol .....     | 3-1 |
| Line Transfers .....           | 3-6 |
| LINT[1:0] .....                | 3-3 |
| LOCK# .....                    | 3-4 |
| Low Power Standby Enable ..... | 5-7 |

## M

|  |     |
|--|-----|
| Machine Check Architecture (MCA) ..... | 4-1 |
| Machine Check Exception (MCE) .....    | 3-8 |
| Memory .....                           |     |
| Address-space size signals .....       | A-2 |
| Micro-Architecture .....               | 2-1 |

## N

|   |      |
|---|------|
| NMI signal .....                          | A-11 |
| Non-maskable Interrupt (NMI) signal ..... | A-11 |
| Nujac .....                               | 3-2  |

## O

|                              |     |
|------------------------------|-----|
| Out-of-order Execution ..... | 2-2 |
| Output Tristate .....        | 5-2 |

## P

|                                  |     |
|----------------------------------|-----|
| Parity Protection .....          | 4-1 |
| Partial Transfers .....          | 3-7 |
| Part-Line Aligned Transfer ..... | 3-7 |
| Pentium II processor .....       | 1-1 |

|                                       |          |
|---------------------------------------|----------|
| Pentium Pro processor .....           | 1-1      |
| Pentium processor .....               | 2-1, 3-6 |
| Phase-Locked Loop (PLL) .....         | 5-10     |
| PICCLK .....                          | 3-3      |
| PICD[1:0]# .....                      | 3-3      |
| Power-On Configuration Register ..... | 5-9      |

## R

|                               |          |
|-------------------------------|----------|
| Recoverable Error (RE) .....  | 4-1      |
| Request command signals ..... | A-12     |
| Request Signals .....         | 3-4      |
| REQ[4:0]# .....               | 3-4      |
| Signals .....                 | A-12     |
| Reservation Station .....     | 2-4      |
| RESET# .....                  | 3-2, 5-1 |
| Behavior .....                | 6-8      |
| Response Signals .....        | 3-5, 4-3 |
| Retire Unit .....             | 2-6      |
| RP# .....                     | 3-4, 4-2 |
| RSP# .....                    | 3-5, 4-2 |
| RS[2:0]# .....                | 3-5      |

## S

|   |      |
|---|------|
| SLP# .....                                    | 3-3  |
| SMMEM signal .....                            | A-14 |
| Snoop Processing .....                        | 4-3  |
| Snoop Signals .....                           | 3-4  |
| STPCLK# .....                                 | 3-3  |
| Symmetric-agent arbitration bus signals ..... | A-5  |
| System Bus (definition of) .....              | 1-2  |
| System Management Bus (SMBus) .....           | 1-1  |
| System Management Interrupt (SMI) .....       | 3-9  |
| System Management Mode Memory signal .....    | A-14 |

## T

|                              |     |
|------------------------------|-----|
| Test Access Port (TAP) ..... | 6-1 |
| Instructions .....           | 6-6 |
| TCK .....                    | 6-1 |
| TDI .....                    | 6-1 |
| TDO .....                    | 6-1 |
| TMS .....                    | 6-1 |
| TRST# .....                  | 6-1 |
| THERMTRIP# .....             | 3-8 |
| TRDY# .....                  | 3-5 |

## U

|                                |     |
|--------------------------------|-----|
| Unrecoverable Error (UE) ..... | 4-2 |
|--------------------------------|-----|