

## List of Common Properties

### Property 1

- **English** : If a snoop hits a modified line in the I1 cache, then the next transaction must be a snoop writeback. (design 1)

### Property 2

- **English**: If signal "enable" rises, then a clock after the fourth transfer signal "pending" must rise. (design 1)

### Property 3

- **English**: If signal "boff" is asserted, then if the first request which is accepted after the assertion of "boff" is not a snoop request, then it is a write request. (design 1)

### Property 4

- **English**: If signal "hit" is active and signal "pending" is not, then next time signal "pending" is active, signal "sel5" is active. (design 1)

### Property 5

- **English** : An urgent request should be the next handled. (design 2)

### Property 6

- **English**: Between a request and its acknowledge the busy signal must remain asserted. (design 2)

### Property 7

- **English**: If a data packet of any size starts and eventually gets a LAST bit, then next data packet must have the FIRST bit asserted. (design 3)

### Property 8

- **English**: If a write command starts and size=N (N=1 through 8), then N assertions of signal "gx\_start" should occur before the LAST bit goes active. (design 3)

### Property 9

- **English**: The address queue ptr increment consecutively (cyclic). In other words, every time that an address is entered into the queue with queue ptr = N, the next time that an address is entered into the queue, the address will be N+1 (cyclically). (design 4)

**Property 10**

- **English:** If data grant received then as soon as dbusy is high for two clocks, take the data bus. (design 5)

**Property 11**

- **English:** The data that returns for read is the last data that was written to the register before the read was issued. (design 5)

**Property 12**

- **English:** Every buffer will be read before it is overwritten. (design 6)

**Property 14**

- **English:** For every write, data transfers must alternate between odd and even entries. In other words, if there is a write, then as long as we are transferring data belonging to this write, consecutive data transfers must alternate between even and odd addresses.

**Property 15**

- **English:** Two consecutive writes cannot be to the same address. Address appears one cycle after the write\_valid.(design 6)

**Property 16**

- **English:** If an address was set valid then the next time the "retire" signal is asserted for this address, the address will be invalidated 3-8 clocks later.

**Property 17**

- **English:** If read req is received, then either the next output req to PLB is read, or the one after that.

**Property 18**

- **English:** If read req is received and then write req is received before read req is output, then read req will be output before write req is output. (Assumption: reqs stay asserted until output.)

**Property 19**

?

**Property 20**

- **English:** "any number of transactions limited by the depth of the queue in the bridge may be split, and may be retried an unlimited number of times provided they are always each retried within a defined timeout period (typically 32,000 clock cycles) of the last attempt. If

any of these transactions is not retried within this time period the bridge must set the Discard Timer Status bit 10 in the Bridge Control register. In addition, the bridge must assert SERR# on the primary interface if enabled to do so by the Discard Timer SERR# Enable bit 11 in the Bridge Control register and the SERR# Enable bit in the Command register"

- **Comments:** many features of the bus are omitted, including error and abort conditions for clarity. note also - all the signals shown are active low, therefore "asserted" means =0.

Transactions can be 'stretched' by either the initiator or the target. The 'good' completion is denoted by irdy\_ and trdy\_ being asserted at the same time.

When a target is not able to complete a transaction quickly enough, it can ask for the transaction to be re-tried at a later time. It does this by asserting the stop\_ signal instead of the trdy\_ signal.

Where the target is a multi-function device or a bus-bridge it is possible for the target to have multiple "open" transactions at any instant. It is also important to understand there is no requirement for the transactions to be re-tried in the order in which they are started, nor for them ultimately to be completed in any particular order.

This property demonstrates the presence of multiple "open" transactions at any time, each transaction being matched by use of a 'tag' that is effectively the concatenation of { address, address\_parity, command, byte\_enables, REQ64# } (a total of about 74 bits if the bus is operating in the 64 bit mode). [[note : this is not strictly correct as the 'byte\_enable' signals are not known until later in the transaction, therefore the concatenations ought to be spilt into two parts, but this explains the principle]] One issue we need to be aware of is that the 'address' and 'command' bits are known only in cycle 2 of the transaction, however we do not know a retry will be requested until between cycle 4 to cycle 17 for any transaction.

Note also : the scope of the 'tag' variable used in simulation must be local to the instance of the property since there will be multiple, concurrent instances of the property with different tag values, one per "open" transaction.

### Property 21

- **English:** "if an initiator begins a MemoryReadLine transaction, the last implied address of the last data phase of the transaction must not be in a different cache line than the initial address if the target implements the Cacheline Size Register feature. Note, that support of the Cacheline Size Register is optional and that each target may implement a different Cacheline size."

- **Comments:**

The PCI bus allows several forms of 'burst' transfer of data from successive words in memory. One of the burst modes is 'MemoryReadLine' which is recognised from the value on the C/BE# signal group during cycle 2.

Note : for simplicity we will assume the bus is working only in 32-bit mode

If the CachelineSize for this target has been set at, for example, 64 bytes, then the master may not attempt a burst that would cause the auto-incrementing address to rollover from xx0011111100 to xx0100000000.

There are two potential solutions to implement this property, either by watching for the rollover condition or by pre-calculating the maximum number of data transfers allowed in the burst as

$X = (\text{CachelineSize} - (\text{InitialAddress} \& (\text{CachelineSize} - 1))) \gg 2;$

Using the example above

$X = (000001000000 - (\text{xx}0011110100 \& (000001000000 - 000000000001))) \gg 2$   
 $= (000001000000 - (\text{xx}0011110100 \& (000000111111))) \gg 2$   
 $= (000001000000 - (000000110100)) \gg 2$   
 $= (000000001100) \gg 2$   
 $= 3$

Then limiting the non-deterministic burst transfer count to the range [1..X]

### **Property 22**

- **English:** if a target does not implement the Cacheline Size Register feature, the target must respond to a MemoryReadLine or MemoryReadMultiple transaction using a Disconnect With Data during the first data phase or a Disconnect Without Data during the second data phase"

- **Comments:**

For information, the specification explains "This ensures that the transaction will complete (albeit slowly, since each request will complete as a single data phase transaction)."

For the purposes of demonstrating this property we can make a small simplification in the PCI specification and assume:

This property raises the issue of recognizing and reacting to different features of different targets. The property can be implemented by either

a) saving the value of device registers that are typically only visible by monitoring the bus during initialization

or

b) determining the value of the registers by looking into the design during verification, presumably based on some address translation or mapping scheme.

Disconnect With Data is signaled by the target by asserting both the STOP# signal in addition to TRDY# coincident with the initiators IRDY# to effect a data transfer.

Disconnect Without Data is signaled by the target by asserting the STOP# signal without asserting TRDY# coincident with the initiators IRDY#.

### **Property 45**

- **Comments**

PCI compliant devices can behave as master or target agents, or target agents only. The required pins of a PCI agent include Interface Controls FRAME#, TRDY#, IRDY#, STOP#, DEVSEL#, and IDSEL (input), Error Lines PERR# and SERR# Arbitration Lines REQ# (output) and ACK# (input), Clock/Reset CLK and RST# inputs, 32 bit wide Address/Data bus (AD), 4 bit wide Command/Byte Enable line (C/BE). There are also numerous optional signals. The Interface Signals are `asserted' by holding them low. An "address phase" is marked by FRAME# fall; A "data phase" is indicated when TRDY# and IRDY# are high; An "i/o cycle" is a data phase in which the C/BE line indicates an i/o read or write;

After FRAME# has been asserted a target can claim the access cycle by asserting DEVSEL#; a "target abort" is executed by asserting the STOP# line after it has claimed the cycle by asserting DEVSEL#.

A "master abort" is executed by the master asserting (for at least one CLK period if not already asserted) and subsequently deasserting IDRY#; FRAME# is deasserted for at least one CLK

period when IRDY# is asserted; the master abort is finally completed with deassertion of FRAME# and IRDY# lines.

The "last data phase" is indicated when IRDY# is asserted, FRAME# is deasserted and either TRDY# or STOP# is asserted.

The "last data phase" is indicated when IRDY# is asserted, FRAME# is deasserted and either TRDY# or STOP# is asserted.

data phase" completes when IRDY# is asserted with either STOP# or ir TRDY# asserted simultaneously.

The initial data phase is marked by FRAME# fall; subsequent data phases are indicated by FRAME#, IRDY# and TRDY# all being asserted.

#### **Property 45.1**

- **English:** A target must assert DEVSEL# before any other response within 1 to 3 clocks following the address phase. If no target responds, a Master-Abort should be performed after 15 cycles.

Note that this property deviates from the actual PCI spec by requiring that the Master-Abort \*must\* be completed in 15 cycles.

#### **Property 45.2**

- **English:** For an IO cycle, if the initiator asserts byte-enables of lesser significance than what is indicated by AD[1:0] the target must terminate the transaction with target abort

#### **Property 45.3**

- **English:** During read transactions, no target may assert TRDY# in the first cycle following the assertion of FRAME#.

#### **Property 45.4**

- **English:** Once a target has asserted TRDY# or STOP# it cannot change DEVSEL#, TRDY#, or STOP# until the current data phase completes.

#### **Property 45.5**

- **English:** Once DEVSEL# has been asserted, it cannot be deasserted until the last data phase has been completed, except to signal Target-Abort.

#### **Property 45.6**

- **English :** All targets are required to complete the initial data phase of a transaction (read or write) within 16 cycles from the assertion of FRAME#.

Note that this property deviates from the actual PCI spec by ignoring the additional number of cycles a bus bridge is allowed.

#### **Property 45.7**

- **English:** PERR# has a turnaround cycle on the 4th clock after the last data phase, which is three clocks after the turnaround for AD# lines.

#### **Property 45.8**

- **English:** Once a master has asserted IRDY#, it cannot change IRDY# or FRAME# until the current data phase completes. (Note: DEVSEL# and IRDY# can go low in either order.)

#### **Property 45.9**

- **English:** A master is required to assert its IRDY# within 8 clocks from any given data phase (initial and subsequent).

#### **Property 45.10**

- **English:** For a Special Cycle transaction, if the initiator inserted one or more wait states before asserting IRDY# with the message, the master must extend the master abort timeout period (that is to say the time before it terminates the transaction) by at least the same number of wait states.

#### **Property 46**

- **English:** Whenever a “read” signal is asserted, “busy” has to be asserted for 3 cycles and then de-asserted. If an additional “read” arrives before “busy” has been de-asserted, then “busy” has to stay high for 3 cycles from the last “read”. The value of “busy” at the same cycle in which “read” is asserted is not important.

#### **Property 47**

- **English :** A bit vector “x[7:0]” is not allowed to contains more than one bit asserted. Moreover, if bit p ( $0 \leq p \leq 7$ ) of x is asserted at time N and bit q of x is asserted at time M, then  $|M-N| > 4$ .

#### **Property 48**

- **English:** Put an assumption on the environment such that the run is initiated by 5 clk cycles of rst followed by rst staying low forever.

#### **Property 49 (global variables)**

- **English :**  
datain - control signal, indicates that there is valid data on the data\_input\_bus.  
data\_input\_bus - 32 bit data input bus.  
dataout - control signal, indicates that there is valid data on the data\_output\_bus.  
data\_output\_bus - 32 bit data output bus.

Write a specification of a fifo, where when datain rises data is entered into the fifo (thru the data\_input\_bus port) and when dataout rises data exits the fifo. The specification needs to check that for each data that enters the fifo, that data comes out at the anticipated dataout rising edge. That is, data exits in order of arrival.

data: uint(bits:32);

#### **Property 50 (suspend)**

- **English:** Same property as above with the addition of the 'wait' signal. When the 'wait' signal is asserted, all specification threads are suspended, then when the 'wait' signal is deasserted the specifications continue from the point they left off (before suspension).

#### **Property 53**

- **English:** Write an assumption that specifies that once the design enters a power-down mode it stays in this mode and the design may choose to enter the power-down mode only during initialization period, that is, only until new\_cycle is set for the first time.

#### **Property 54**

- **English:** Write an assumption that constrains the behavior of the “op-code instruction markers” (i.e., vectors first\_opcode\_input and last\_byte\_input). The required behavior is as follows: if bit j in vector instr\_val\_input is zero then both markers need to be zero in index j.

Remark: first\_opcode\_input is a vector such that its element j is set iff j is the first byte of an opcode. Similarly, last\_byte\_input is a vector such that its element j is set iff j is the last byte of an opcode.

#### **Property 55**

- **English:** Write a parametric assertion that states that every byte that gets into the queue will eventually be taken out of the queue. The parameter represents the width of the address of elements in the queue. This behavior is interrupted once the signals reset or raprep\_cycle83l are set.

#### **Property 56**

- **English:** write an assumption that restricts the behavior of the lin\_addr\_input vector such that its 4 low level bits (called align) and its higher level bit (rest 28 bits, called addr) behave as follows:  
In case of a reset in previous cycle addr is set to zero and align gets an arbitrary value.  
In case of a valid new cycle addr and align are advanced to the next cache line.  
In case of a branch advance to the branch address.

#### **Property 57**

- **English:** Every request which is acknowledged (signal req must stay asserted until ack is received) must be followed at some point in the future with a valid grant (a grant which is not aborted the following cycle)

**Property 58**

- **English:** Every request which is not retried (a retry happens or not two cycles after assertion of signal req) must be followed by a sequence in which busy is asserted until and is asserted (if end is never asserted, then busy must stay asserted forever).

**Property 59**

- **English:** Every request which is not retried (a retry happens or not two cycles after assertion of signal req) must eventually receive an ack

**Property 60**

- **English:** On every assertion of hi\_pri\_req, one of the next two assertions of gnt must be to a high priority requestor (dst=hi\_pri).

**Property 61**

- **English:** Every transaction must complete, and within every transaction, a full data transfer must occur.

**Property 62**

- **English:** A sequence beginning with the assertion of signal go, containing eight not necessarily consecutive assertions of signal get, during which kill is not asserted, must be followed by a sequence of eight assertions of signal put before signal end can be asserted.

**Property 63**

- **English:** Lack of deadlock.

**Property 64**

- **English:** Lack of dependency: In one version of PCI, TRDY must not be dependent on IRDY

**Property 65**

- **English:** Reset can rise asynchronously, but falls on the rising edge of clk. Once asserted reset stays high at least 6 full clk cycles, where clk cycles are of indeterminate length. It is possible that reset eventually asserts forever.

**Property 66**

- **English:** When clk is high, if port1 has a vl15 request to port2 then eventually port2 is granted to port1. The property holds after reset. Also write an assumption that for all 16 ports and for all 16 virtual-lanes, requests must keep pending until they are granted.

**Property 67**

**English: Write an assumption that for each of the sixteen input ports, a port cannot request the same output port in both vl0 and vl15 simultaneously.**

**Property 68**

- **English:** Put an assumption on the environment such that the run is initiated by rst cycle of at least 5 clk cycles, followed by rst staying low forever.

**Property 69**

- **English:** If within 8 cycles from the beginning of the transaction, 'p\_start\_reg' and 'discard\_rx\_' both appear, then the valid signal should be deasserted in the next cycle.

**Property 70**

- **English:** transaction sequence with two data-transfers, during which an error occurs, should be retried.

**Property 71**

- **English:** The number of retries cannot be greater than 20, if the number of retries is greater than 20 a special error flag will be asserted.

**Property 72**

- **English:** In this example, the behavior of a simple cache is specified. In this cache each address has a given number of lives. Each time an address is read, the lives of all other addresses in the cache is reduced by one. When the life of an address reaches zero, the address is removed from the cache's memory. When an address is reread it is returned all of its lives.

**Property 73 (strong liveness)**

- **English:** The finite state machine LockDet cannot stay in state UNSAT forever if the oldest uop keeps replaying.

**Property 74**

- **English:** We model a bus with a Bus No Request (BNR) signal. Generally, the bus is either: (a) free, requests may be sent. (b) stalled, requests may not be sent. (c) throttled, one request may be sent.  
A request is signaled by asserting ADS. In state free ADS may be asserted freely (but at least 3 clks apart). In state stalled ADS may not be asserted. In state throttled ADS may be asserted once.  
If the bus is stalled or throttled, BNR is sampled 2 clocks after the previous sampling. If the bus is free, BNR is sampled 3 clocks after ADS is asserted.

Transition from state to state:

In state free, if BNR is asserted move to state stalled.

In state stalled, if BNR is not asserted move to state throttled.

In state throttled, if BNR is asserted return to state stalled, otherwise move to state free.