6.4.6 FAST_START register

The FAST_START register permits an initiator to signal a new task to a fetch agent by means of a single block write request (fast start packet) addressed to the register. This write-only register shall support block write requests whose *destination_offset* is equal to the address of the FAST_START register and whose *data_length* is less than or equal to the vendor-dependent size of the register (see 7.6.10), and shall reject all other requests. The format of this register is illustrated below.

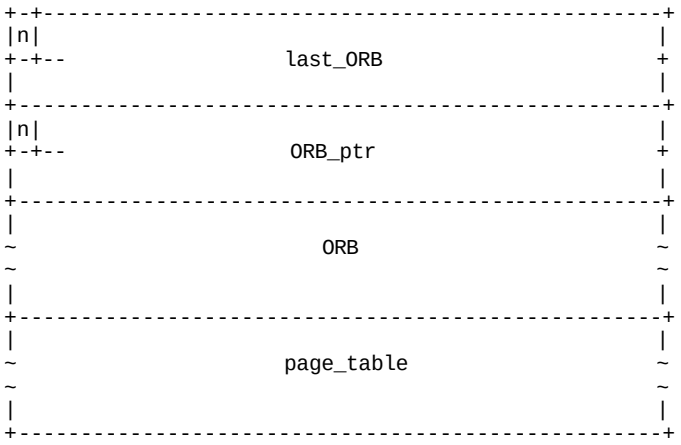

Figure 45 – FAST_START format

The *last_ORB* field contains an ORB pointer as defined in figure 12. If non-null, the *offset_hi* and *offset_lo* fields shall reference the last ORB in the chain signalled by the initiator. The target can then use this field to validate that the fast start packet has been sent at the correct time.

The *ORB_ptr* field also contains an ORB pointer; the *null* bit shall not be set, and the *offset_hi* and *offset_lo* fields shall reference a block of initiator memory whose contents are identical to the ORB field in the fast start packet. The corresponding Serial Bus address shall be formed from the concatenation of the 16-bit node ID of the initiator (or its node handle if the login is bridge aware) and the *ORB_offset* field.

The *ORB* field shall contain an ORB whose format conforms to those specified by 5.1. The length of the ORB is variable and shall be determined by the *ORB_size* field in config ROM. The target shall reject a fast start packet if its *data_length*, in bytes, is less than sixteen plus the size of the ORB field.

The *page_table* field, if present, shall immediately follow the ORB field. If the format of the ORB field includes a nonzero *page_table_present* bit, the *page_table* field shall contain zero or more page table entries whose order and content shall be identical to those contained within the page table referenced by the *data_descriptor* field in the ORB referenced by ORB_ptr. The *page_table* field may be a subset of the page table referenced by the ORB, but no partial page table entries shall be present (see 5.2). The target shall derive the number of immediately available page table entries from the *data_length* of the fast start packet. The maximum number of page table entries that can be included in the fast start packet is limited by the maximum size of the FAST_START register (specified in config ROM)

The effects of a write transaction to the FAST_START register are dependent on the value of *st* in the AGENT_STATE register and the value of the last_*ORB* field in the fast start packet.

If the target is in the ACTIVE state, the target shall treat the transaction as a write to its DOORBELL register, and may ignore the contents of the fast start packet.

If the target is in the RESET or SUSPENDED state, the target shall compare the *last_ORB* field with its own ORB_POINTER register, and accept or reject the transaction according to the following criteria:

- If *last_ORB* is NULL (its *null* bit is set), the target shall accept the transaction.
- If *last_ORB* is valid (non-null), the target shall compare the ORB offset with its own ORB_POINTER register, and only accept the transaction if there is a match between the two.

If the target accepts the transaction, it shall update its ORB_POINTER register with the contents of the ORB_ptr field (or the next_ORB field of the ORB if it is non-null), and pass the ORB and page_table data to the device's working set for execution.

9.1.4 Use of the FAST_START register (informative)

An initiator has the option of signaling new task(s) to the target by means of a block write request addressed to the fetch agent's FAST_START register (see 6.4.6) The block write request contains the address of the ORB to be commenced, a copy of the ORB itself and, optionally, page table data associated with the ORB (It may also contain a previous ORB pointer value to enable the target to validate the ORB chaining mechanism).

Significant overhead reductions may result from the use of the FAST_START register, since the target need not fetch either the address of the ORB or the ORB itself. In cases where the block write request contains the entire page table, the target need not fetch the page table; even if the entire page table is not written to the FAST_START register (it may be too large), the target may significantly reduce startup latency by fetching the remaining page table entries concurrently with task execution.

The most significant performance gain from using the FAST_START register is obtained when reactivating the fetch agent from the RESET or SUSPENDED state. However the inclusion of the *last_ORB* field in the register allows the initiator to use the FAST_START register with the certainty that the target will only pass ORBs to the device server in the same order as they appear in the ORB chain. This allows a multi-threaded operating system to use the fast start capabilities without incurring the time delays involved making the operation multi-thread safe. To achieve this, a multi-threaded initiator should always maintain a local reference to the last ORB in its chain. When the target is in the SUSPENDED state, the contents of this reference will be equal to the target's ORB_POINTER register. When such an initiator sends a fast start packet, it should set the *last_ORB* field to the contents of its last ORB reference; then the target uses this field to validate that the fast start packet has been sent at the correct point in the chain.

For an initiator to make use of this error-checking capability of the target, it shall perform the following steps (in order) to construct and send a fast start packet:

- Construct the new ORB in its own memory space
- Update its local last_ORB reference (using a suitable atomic swap operation) with the address of the new ORB, but retain the old value for use in the fast start packet.
- The old value points the last ORB in the chain. Modify the *next_ORB* field of this ORB (still in memory) with the address of the new ORB (unless of course the old value is zero, which indicates that there have been no previous ORBs since the last reset).
- Construct the fast start packet as follows: *last_ORB* = old value in last_ORB reference, *ORB_ptr* = new value, *ORB* = new ORB, *page_table* fields set as appropriate to the contents of the ORB.
- Send the fast start packet as a block write request to the target

Using this mechanism, the target can ascertain if a fast start packet has been sent 'out of turn'. If the *last_ORB* field of a fast start packet does not match the target's own ORB_POINTER register, then the target can deduce that there are one or more other ORBs in the queue before the one contained in the fast start packet. Therefore the target shall not execute this ORB at this time, because the ORB will be included further down the chain of ORBs presently signalled by the initiator.

If there is no possibility in the initiator for fast start packets to be sent out of order, it can leave the last_ORB field set to a null ORB pointer value. In this case the target will never check the value, and as long as it is in the RESET or SUSPENDED state, the target will always accept the transaction and pass the ORB to the device server for execution.

There are several ways by which an initiator may securely know that a fetch agent is in the RESET or SUSPENDED state. If the initiator has not written to either of the fetch agent's ORB_POINTER or FAST_START registers since the most recently completed login or reconnect operation or the most recent write to the fetch agent's AGENT_RESET register, the fetch agent is

in the RESET state. Similarly, if the initiator has not written to either of the fetch agent's ORB_POINTER or FAST_START registers since the target last stored a status block with a *src* field equal to one (see 5.3.1), the fetch agent is in the SUSPENDED state.

Either a single ORB or a linked list of ORBs may be signaled in a single block write request to the FAST_START register, dependent upon the value of the *next_ORB* field in the ORB contained within the fast start packet. Once a successful completion response is received for the block write request, the initiator may append to the linked list of ORBs by the methods described in 9.1.2.

Transition F0:F3.
Transition F4:F3.
A block write to the FAST_START register (if the last_ORB field is valid according to the rules in 6.4.6) shall update the ORB_POINTER register with the value of *ORB_ptr* (or the *next_ORB* field of *ORB* if it is non-NULL), and shall cause the fetch agent to transition to state F3. The fetch agent shall also make the ORB and any page table data available to the device server for execution. Once these actions are complete, the target shall confirm the block write request with a response subaction of COMPLETE.

NOTE – A target may interpret a block write addressed to its FAST_START register as if the *ORB_ptr* field had been written to its ORB_POINTER register (although it must still check the *last_ORB* field). Although it is less efficient to elect transition F0:F1, it is functionally equivalent to transition F0:F3.

E.2 ORB_POINTER or FAST_START write request

A consequence of the write operation is that, if successful, the target fetch agent transitions to the ACTIVE state. If no acknowledgement is received by the initiator after a write to either the ORB_POINTER or FAST_START register, the initiator should not retry the write. The recommended method for error recovery is a write to the AGENT_RESET register.