12.6 Processor and External Request Protocols

The following sections contain a cycle-by-cycle description of the bus arbitration protocols for each type of processor and external request. Table 12-5 lists the abbreviations and definitions for each of the buses that are used in the timing diagrams that follow.

Table 12-5 System Interface Requests

Scope	Abbreviation	Meaning
Global	Unsd	Unused
SysAD bus	Addr	Physical address
	Data<n>	Data element number n of a block of data
SysCmd bus	Cmd	An unspecified System interface command
	Read	A processor or external read request command
	RwWF	A processor read-with-write-forthcoming request command
	Write	A processor or external write request command
	Null	A processor null request command
	SINull	A System interface release external null request command
	SCNull	A secondary cache release external null request command
	Ivd	A processor or external invalidate request command
	Upd	A processor or external update request command
	Ivtn	An external intervention request command
	Snoop	An external snoop request command
	NData	A noncoherent data identifier for a data element other than the last data element
	NEOD	A noncoherent data identifier for the last data element
	CData	A coherent data identifier for a data element other than the last data element
	CEOD	A coherent data identifier for the last data element

Processor Request Protocols

Processor request protocols described in this section include:

- read
- write
- invalidate and update
- null write
- cluster

NOTE: In the timing diagrams, the two closely spaced, wavy vertical lines (such as those shown in Figure 12-16) indicate one or more identical cycles which are not illustrated due to space constraints.

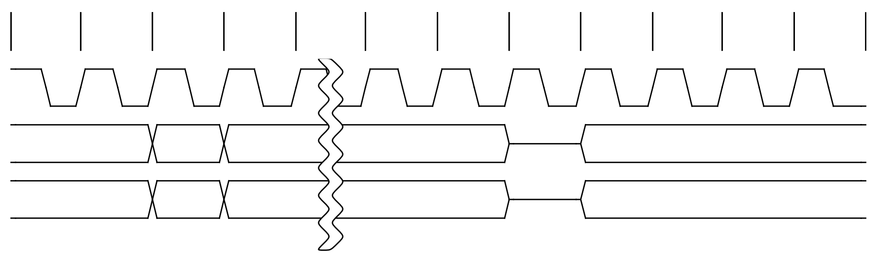

Figure 12-16 Symbol for Undocumented Cycles

Processor Read Request Protocol

The following sequence describes the protocol for a processor read request (the numbered steps below correspond to Figures 12-17 and 12-18).

1. **RdRdy*** is asserted low, indicating the external agent is ready to accept a read request.
2. With the System interface in master state, a processor read request is issued by driving a read command on the **SysCmd** bus and a read address on the **SysAD** bus.
3. At the same time, the processor asserts **ValidOut*** for one cycle, indicating valid data is present on the **SysCmd** and the **SysAD** buses.

   NOTE: Only one processor read request can be pending at a time.
4. The processor makes an uncompelled change to slave state either at the issue cycle of the read request, or sometime after the issue cycle of the read request by asserting the **Release*** signal for one cycle.

NOTE: The external agent must not assert the signal **ExtRqst*** for the purposes of returning a read response, but rather must wait for the uncompelled change to slave state. The signal **ExtRqst*** can be asserted before or during a read response to perform an external request other than a read response.

5. The processor releases the **SysCmd** and the **SysAD** buses one SCycle after the assertion of **Release***.
6. The external agent drives the **SysCmd** and the **SysAD** buses within two cycles after the assertion of **Release***.

Once in slave state (starting at cycle 5 in Figure 12-17), the external agent can return the requested data through a read response. The read response can return the requested data or, if the requested data could not be successfully retrieved, an indication that the returned data is erroneous. If the returned data is erroneous, the processor takes a bus error exception.

Figure 12-17 illustrates a processor read request, coupled with an uncompelled change to slave state, that occurs as the read request is issued. Figure 12-18 illustrates a processor read request, and the subsequent uncompelled change to slave state, that occurs sometime after the read request is issued.

NOTE: Timings for the **SysADC** and **SysCmdP** buses are the same as those of the **SysAD** and **SysCmd** buses, respectively.

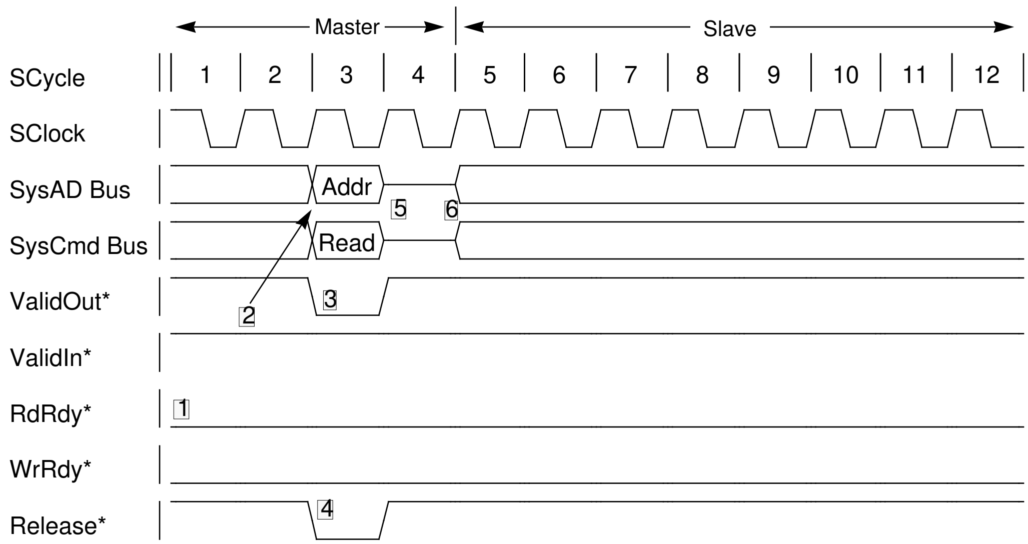

Figure 12-17 Processor Read Request Protocol

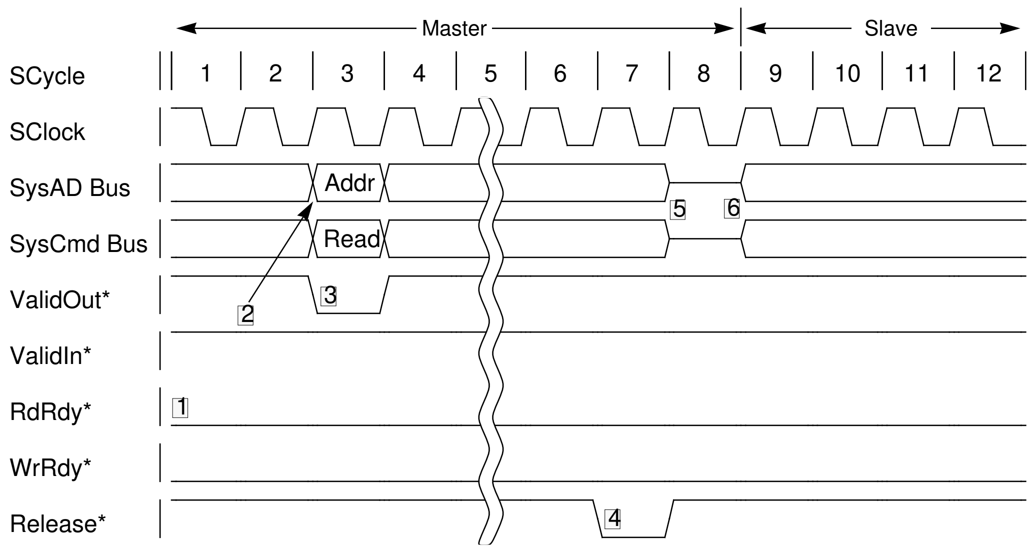

Figure 12-18 Processor Read Request Protocol, Change to Slave State Delayed

When the following three events occur—a read request is pending, **ExtRqst*** is asserted, and **Release*** is asserted for one cycle—it may be unclear if the assertion of **Release*** is in response to **ExtRqst***, or represents an uncompelled change to slave state. The only situation in which the assertion of **Release*** cannot be considered an uncompelled change to slave state is if the following three conditions exist simultaneously:

- the System interface is operating in secondary-cache mode
- the read request was a read-with-write-forthcoming request
- the expected write request has not been issued by the processor.

If these three conditions exist, the processor cannot accept the read response; rather, it accepts the external request. The write request must be accepted by the external agent before the read response can be issued.

In all other cases, the assertion of **Release*** indicates either an uncompelled change to slave state, or a response to the assertion of **ExtRqst***, whereupon the processor accepts either a read response, or any other external request. If any external request other than a read response is issued, the processor performs another uncompelled change to slave state, asserting **Release***, after processing the external request.

Processor Write Request Protocol

Processor write requests are issued using one of two protocols.

- Doubleword, partial doubleword, word, or partial word writes use a word† write request protocol.
- Block writes use a block write request protocol.

Processor doubleword write requests are issued with the System interface in master state, as described below in the steps below; Figure 12-19 shows a processor noncoherent single word write request cycle.

1. A processor single word write request is issued by driving a write command on the **SysCmd** bus and a write address on the **SysAD** bus.
2. The processor asserts **ValidOut***.
3. The processor drives a data identifier on the **SysCmd** bus and data on the **SysAD** bus.
4. The data identifier associated with the data cycle must contain a last data cycle indication. At the end of the cycle, **ValidOut*** is deasserted.

   NOTE: Timings for the **SysADC** and **SysCmdP** buses are the same as those of the **SysAD** and **SysCmd** buses, respectively.

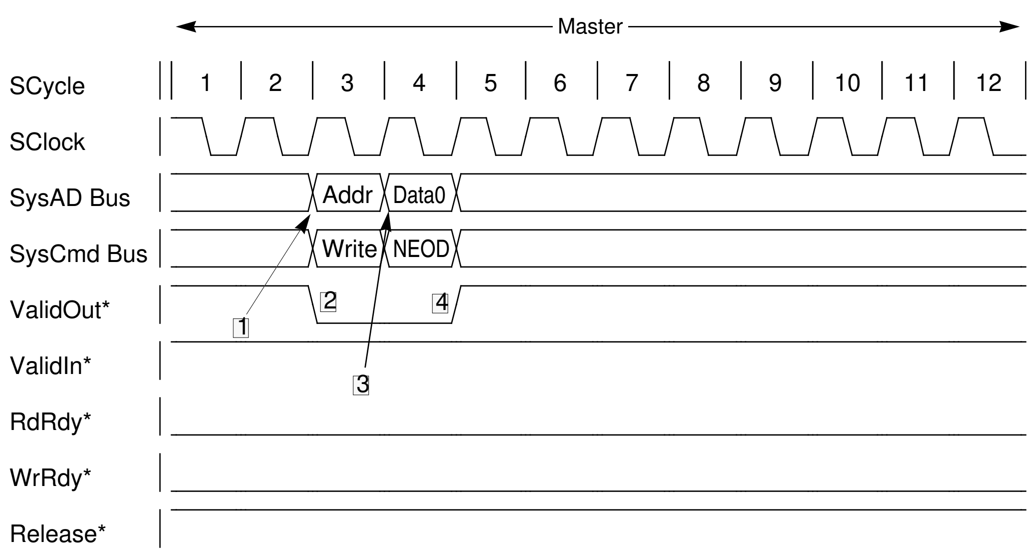

Figure 12-19 Processor Noncoherent Single Word Write Request Protocol

† Called *word* to distinguish it from *block* request protocol. Data transferred can actually be doubleword, partial doubleword, word, or partial word.

Processor block write requests are issued with the System interface in master state, as described below; a processor coherent block request for eight words of data is illustrated in Figures 12-20 and 12-21.

1. The processor issues a write command on the **SysCmd** bus and a write address on the **SysAD** bus.
2. The processor asserts **ValidOut***.
3. The processor drives a data identifier on the **SysCmd** bus and data on the **SysAD** bus.
4. The processor asserts **ValidOut*** for a number of cycles sufficient to transmit the block of data.
5. The data identifier associated with the last data cycle must contain a last data cycle indication.

   NOTE: As shown in Figure 12-21, however, the first data cycle does not have to immediately follow the address cycle.

Figures 12-20 and 12-21 illustrate a processor coherent block request for eight words of data.

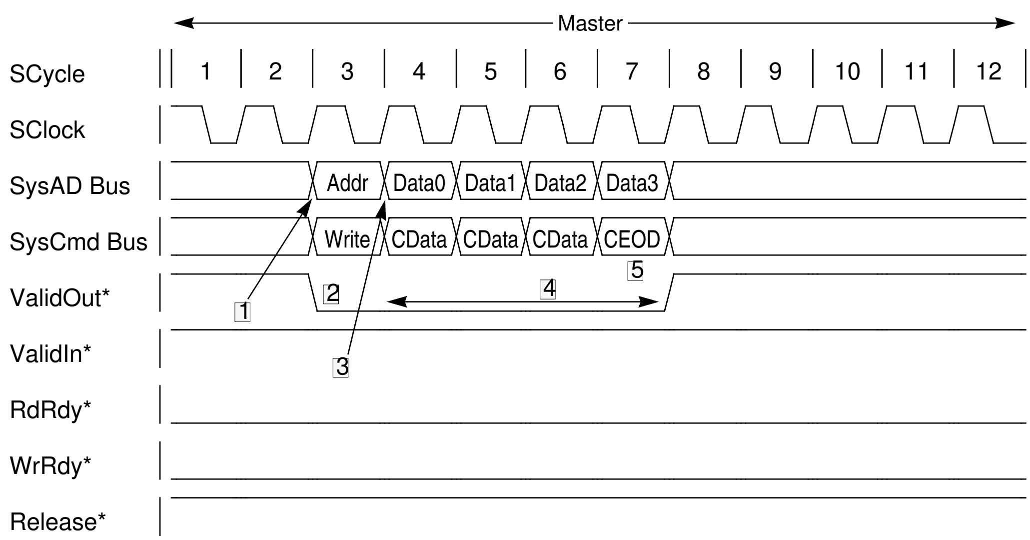

Figure 12-20 Processor Coherent Block Write Request Protocol

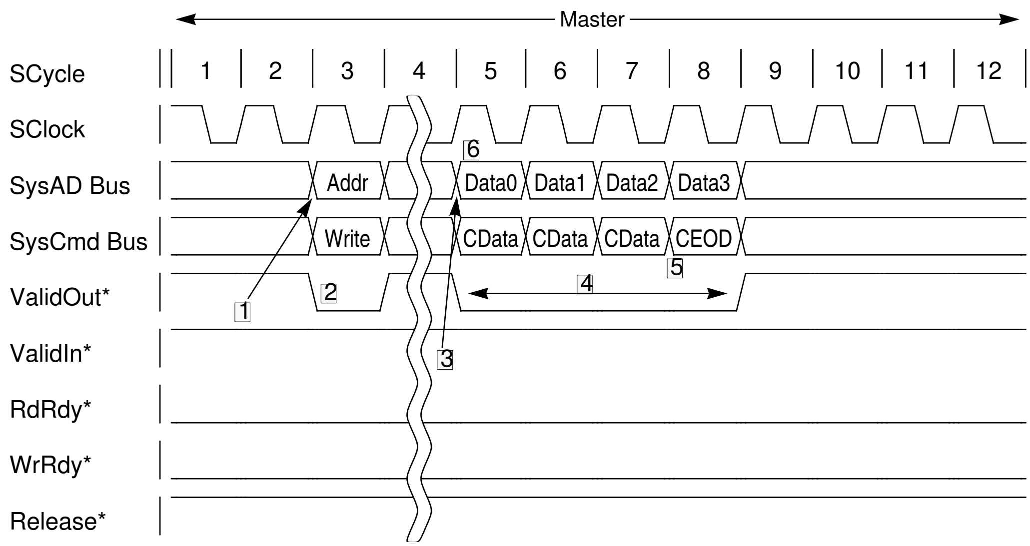

Figure 12-21 Processor Coherent Block Write Request Protocol (Delayed)

Processor Invalidate and Update Request Protocol

Processor invalidate request and update request protocols are the same as a coherent word write request, except for the following:

- invalidate and update requests are controlled by **RdRdy***, while the write request is controlled by **WrRdy***
- the single data cycle transfer is not used by an invalidate request

Processor invalidate and update requests are acknowledged using the signals **IvdAck*** and **IvdErr***. The external agent drives either **IvdAck*** or **IvdErr*** for one cycle to signal the completion of the current processor update or invalidate request; **IvdAck*** occurs in parallel with requests on the **SysAD** and **SysCmd** buses.

IvdAck* or **IvdErr*** can be driven at any time after a processor update or invalidate request is issued, provided the update request is compulsory.

The processor pipeline stalls until one of the following occurs:

- **IvdAck*** or **IvdErr*** is asserted by the external agent. Assertion of **IvdAck*** indicates a successful invalidation, and the processor continues. **IvdErr*** causes a bus error exception.
- either an intervention, snoop, update, or invalidate request is sent by the external agent, with the Invalidate or Update Cancellation bit set, **SysCmd(4)** = 0, indicating the processor invalidate or update request was cancelled.

If the processor update or invalidate request is cancelled, the instruction that caused the processor request is re-executed. If the external request is sent with **SysCmd(4)** = 1, indicating no cancellation, the processor, after responding to the external request, stalls again until one of the two conditions described above terminate the processor's invalidate or update request.

Processor Null Write Request Protocol

A processor null write request is issued with the System interface in master state; the request consists of a single address cycle. The processor drives a null command on the **SysCmd** bus and asserts **ValidOut*** for one cycle. The **SysAD** bus is unused during the address cycle associated with a null write request, and processor null write requests cannot be controlled with either **RdRdy*** or **WrRdy*** signals. Figure 12-22 illustrates a processor null write request.

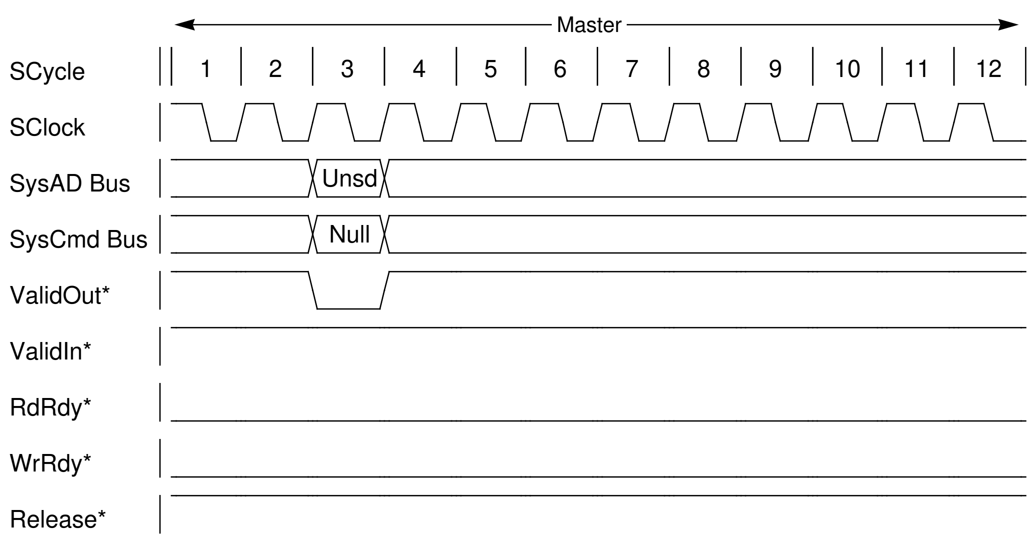

Figure 12-22 Processor Null Write Request Protocol

Processor Cluster Request Protocol

In secondary-cache mode, the processor can issue two types of requests: individual and cluster.

All of the requests that are part of a cluster must be accepted by the external agent before a response to the read request, that began the cluster, can be returned to the processor. A cluster consists of:

- a processor read with write forthcoming request followed by a write request
- a processor read request followed by a potential update request
- a processor read with write forthcoming request followed by a potential update request, followed by a write request.

Figure 12-23 illustrates a cluster consisting of a read with write forthcoming request, followed by a potential update request, followed by a coherent block write request for eight words of data (with minimum spacing between the requests that form the cluster), followed by an uncompelled change to slave state at the earliest opportunity.

NOTE: Timings for the **SysADC** and **SysCmdP** buses are the same as those of the **SysAD** and **SysCmd** buses, respectively. There may be unused cycles between the requests that form a cluster.

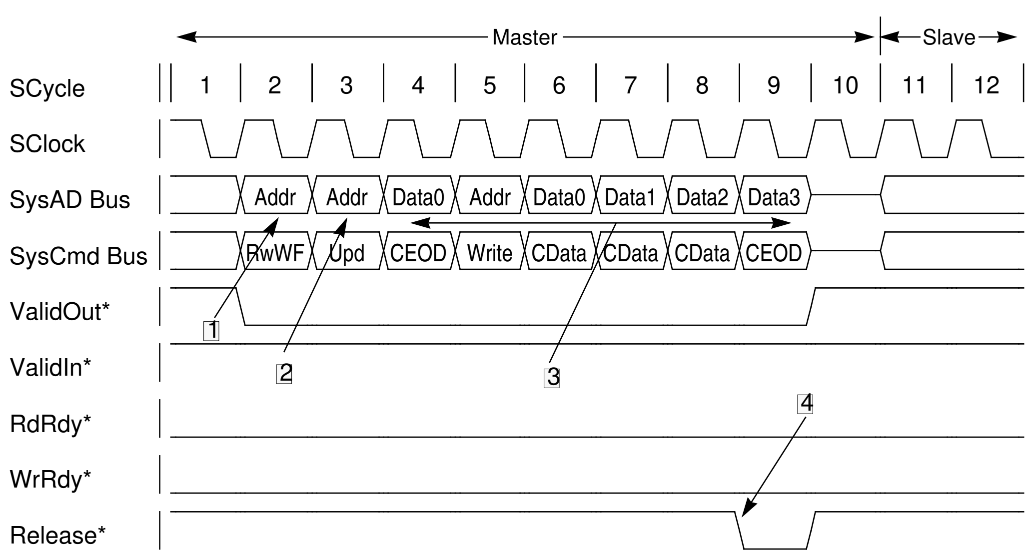

Figure 12-23 Processor Cluster Request Protocol

Processor Request and Cluster Flow Control

The external agent uses **RdRdy*** to control the flow of the following processes:

- processor read request
- processor invalidate request
- processor update request
- processor read request, followed by a potential update request within a cluster.

Figures 12-24 through 12-27 illustrate this flow control, as described in the steps below.

1. The processor samples the signal **RdRdy*** to determine if the external agent is capable of accepting a read, invalidate, update request, or a read request followed by a potential update request.
2. The signal **WrRdy*** controls the flow of a processor write request.
3. The processor does not complete the issue of a read, invalidate, update request, or a read request followed by a potential update request, until it issues an address cycle in response to the request for which the signal **RdRdy*** was asserted two cycles earlier.
4. The processor does not complete the issue of a write request until it issues an address cycle in response to the write request for which the signal **WrRdy*** was asserted two cycles earlier.

Figure 12-24 illustrates two processor write requests in which the issue of the second is delayed for the assertion of **WrRdy***.

Figure 12-25 illustrates a processor cluster in which the issue of the read and a potential update request are delayed for the assertion of **RdRdy***.

Figure 12-26 illustrates a processor cluster in which the issue of the write request is delayed for the assertion of **WrRdy***.

Figure 12-27 illustrates the issue of a processor write request delayed for the assertion of **WrRdy*** and the completion of an external invalidate request.

NOTE: Timings for the **SysADC** and **SysCmdP** buses are the same as those of the **SysAD** and **SysCmd** buses, respectively.

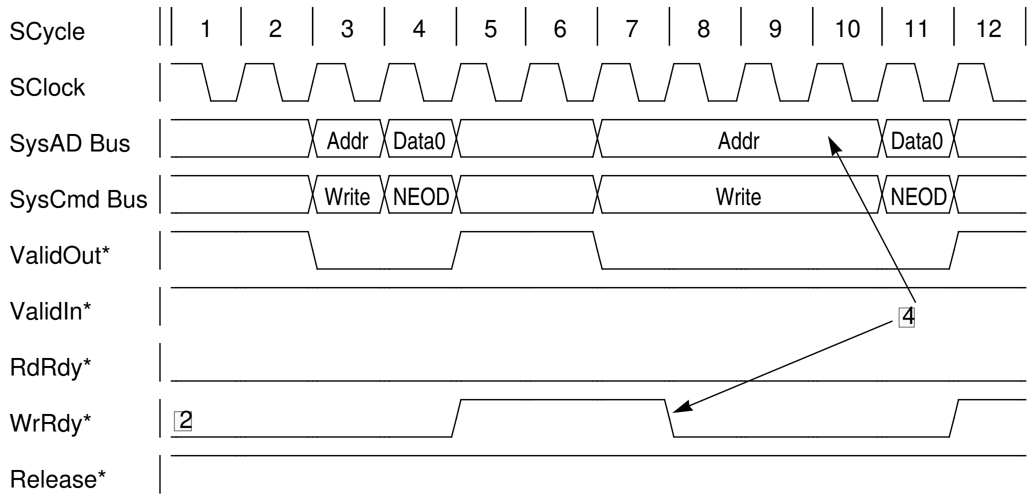

Figure 12-24 Two Processor Write Requests, Second Write Delayed for the Assertion of WrRdy*

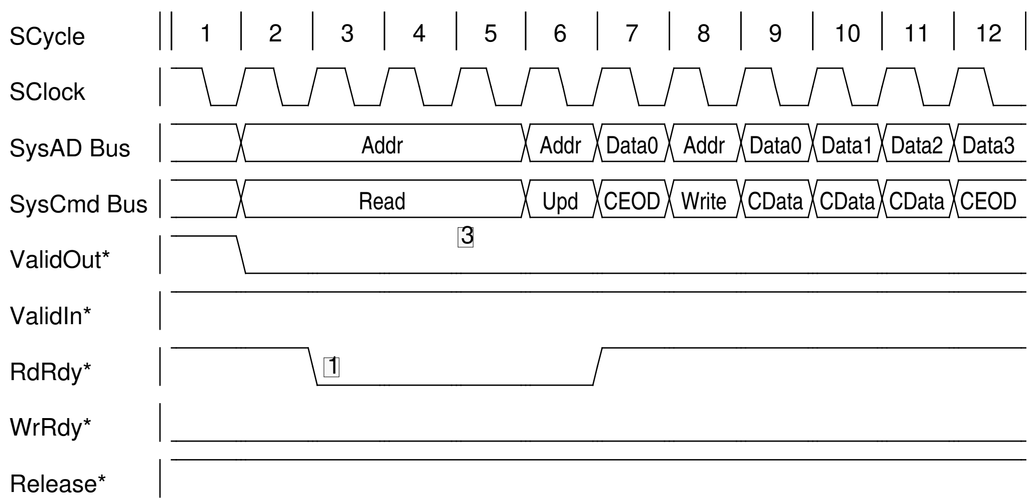

Figure 12-25 Processor Read Request within a Cluster Delayed for the Assertion of RdRdy*

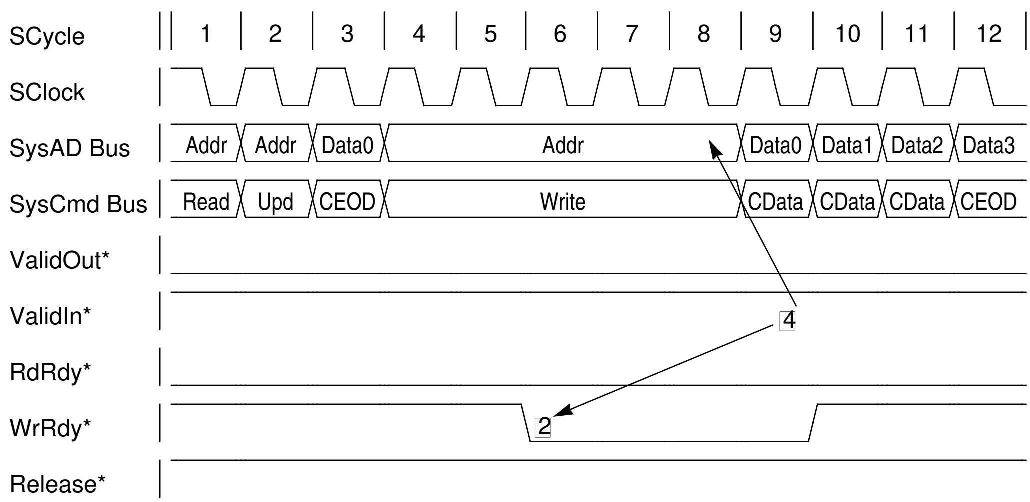

Figure 12-26 Processor Write Request within a Cluster Delayed for the Assertion of WrRdy*

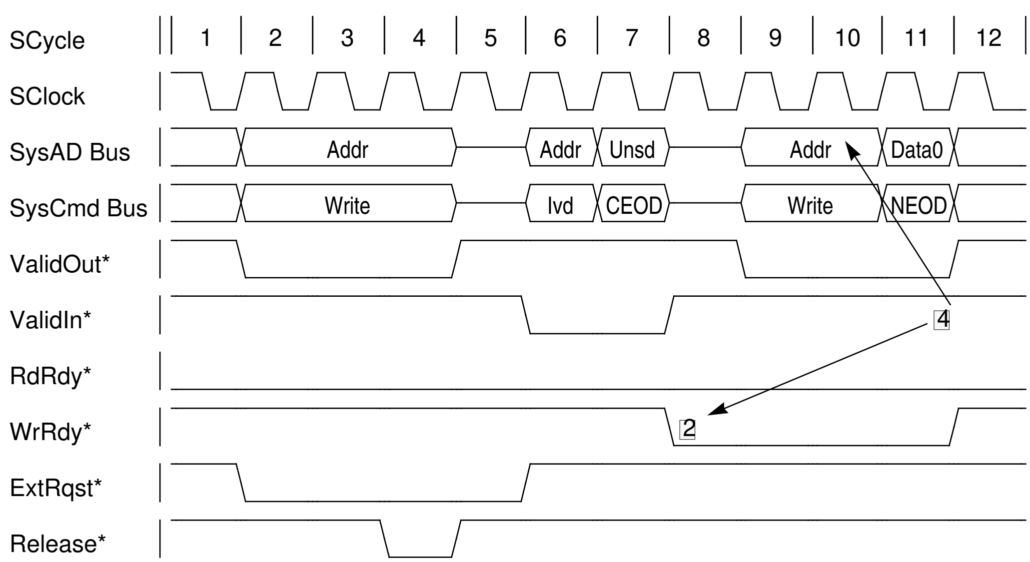

Figure 12-27 Processor Write Request Delayed for the Assertion of WrRdy* *and the Completion of an External Invalidate Request*