



## Features (contd)

- Two bus parking modes selectable via Control Register write:
  - park on selected OPB master (specified in Control Register)
  - park on last OPB master granted OPB access
- Watchdog timer asserts the OPB time-out signal if a slave response is not detected within 16 clock cycles
- Registered or combinational Grant outputs configurable via a design parameter

## Functional Description

The top-level block diagram for the OPB Arbiter is shown in [Figure 1](#). The IPIF block is the OPB bus interface block that handles the OPB bus protocol for reading and writing the Priority Registers and the Control Register within the OPB Arbiter. The ARB2BUS data mux contains the data multiplexor required to output data to the OPB bus during a read cycle. The Arbitration Logic block determines which incoming request has the highest priority and the Park /Lock Logic block determines which master should be granted the bus based on this priority as well as whether the bus is locked or if bus parking is enabled. The Watchdog Timer asserts the OPB timeout signal if a slave response (OPB\_xferAck, OPB\_retry, or OPB\_toutSup) has not been received within 16 clock cycles of the master taking control of the bus.

When C\_NUM\_MASTERS=1, the only logic in the OPB Arbiter is the Watchdog Timer. The OPB Grant signal is set to VCC and all OPB Bus output signals are set to GND.

The modules shown in the block diagram are described in the following sections.

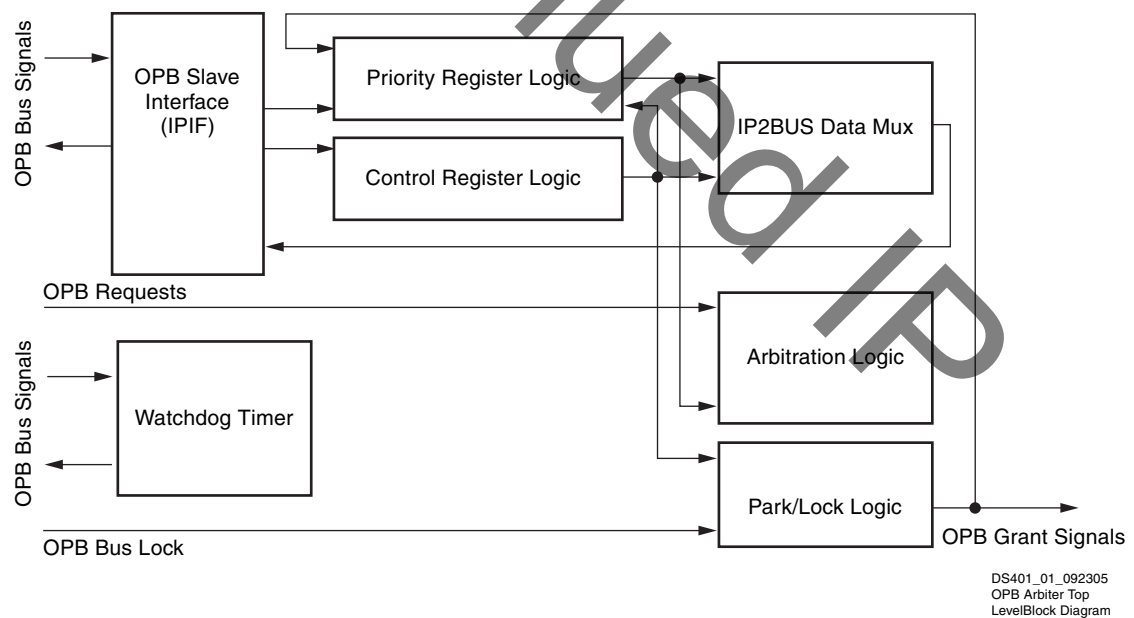


Figure 1: OPB Arbiter Top Level Block Diagram

## OPB Slave Interface (IPIF)

The IPIF block implements a slave interface to the OPB and is only present in the design if `C_PROC_INTRFCE=1` and `C_NUM_MASTERS>1`. Its address in the OPB memory map is determined by setting the parameter `C_BASEADDR`. All registers are addressed by an offset to `C_BASEADDR` as shown in [Table 4](#).

The IPIF block outputs a register write clock enable and a register read clock enable for the register which was addressed depending on the type of data transfer specified by the master. When the data transfer is complete, the IPIF block generates the transfer acknowledge.

## Control Register Logic

The Control Register Logic block simply contains the OPB Arbiter Control Register described in the [OPB Arbiter Control Register](#) section of this document and is present in the design only if `C_NUM_MASTERS > 1`.

## Priority Register Logic

The Priority Register logic contains the Priority Registers and the logic to update the priority of the OPB masters. Descriptions of the OPB Arbiter Priority Registers are found in the [OPB Arbiter Register Descriptions](#) section of this document. The Priority Registers are present in the design only if `C_NUM_MASTERS>1`.

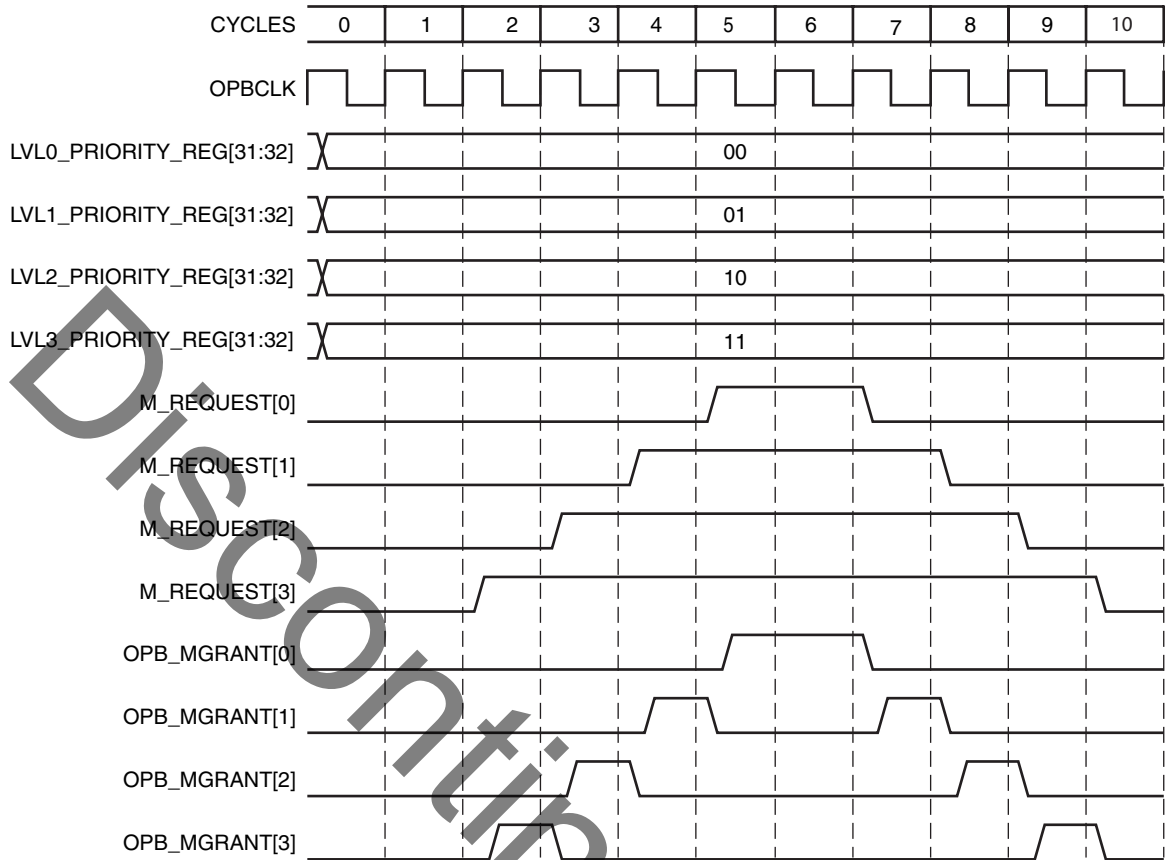
## Priority Register Update Logic

### Fixed Priority Parameterization (`C_DYNAM_PRIORITY=0`)

When the OPB Arbiter is parameterized to support only fixed arbitration, the dynamic priority enable bit in the Control Register is permanently disabled. The Priority Registers are loaded at reset with the value of the Master ID which matches the priority level of the register. For example, the LVL0 Priority Register is loaded with '0' to represent the ID of Master 0. Likewise, the LVLn Priority Register is loaded with the bit encoding of n to represent the ID of Master n as shown in [Table 6](#).

The Priority Registers can be loaded with different Master IDs by writing to the Priority Registers (if `C_PROC_INTRFCE=1`), therefore, the priorities of the OPB Masters can be changed as desired by software. The Priority Registers Valid (PRV) bit in the Control Register is negated whenever the processor modifies the Priority Registers and is asserted whenever the modification is complete. The ID of the masters are used to determine OPB ownership whenever the PRV bit is negated. The relative priorities of the OPB Masters are then determined by the connection of master devices to the request/grant signals. The values of the Priority Registers are used in OPB arbitration whenever the PRV bit is asserted.

[Figure 1](#) shows the fixed priority arbitration for a 4 OPB Master system with combinational grant outputs. [Figure 3](#) shows the same system when configured with registered grant outputs with the master requests separated by an additional clock.



DS401\_02\_092305  
 Fixed Priority Arbitration,  
 Combinational Grant Outputs  
 for 4 OPB Masters

Figure 2: Fixed Priority Arbitration, Combination Grant Outputs for 4 OPB Masters

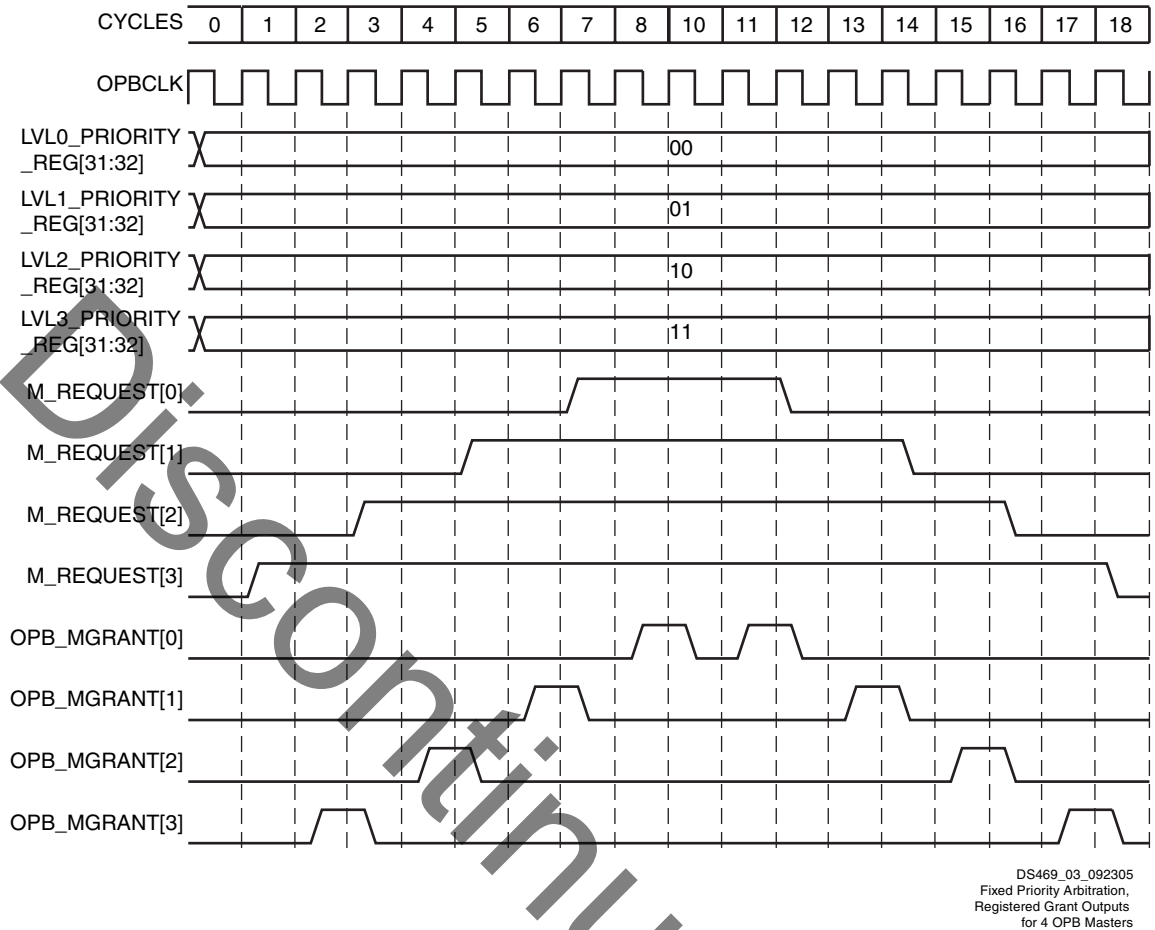


Figure 3: Fixed Priority Arbitration, Registered Grant Outputs for 4 OPB Masters

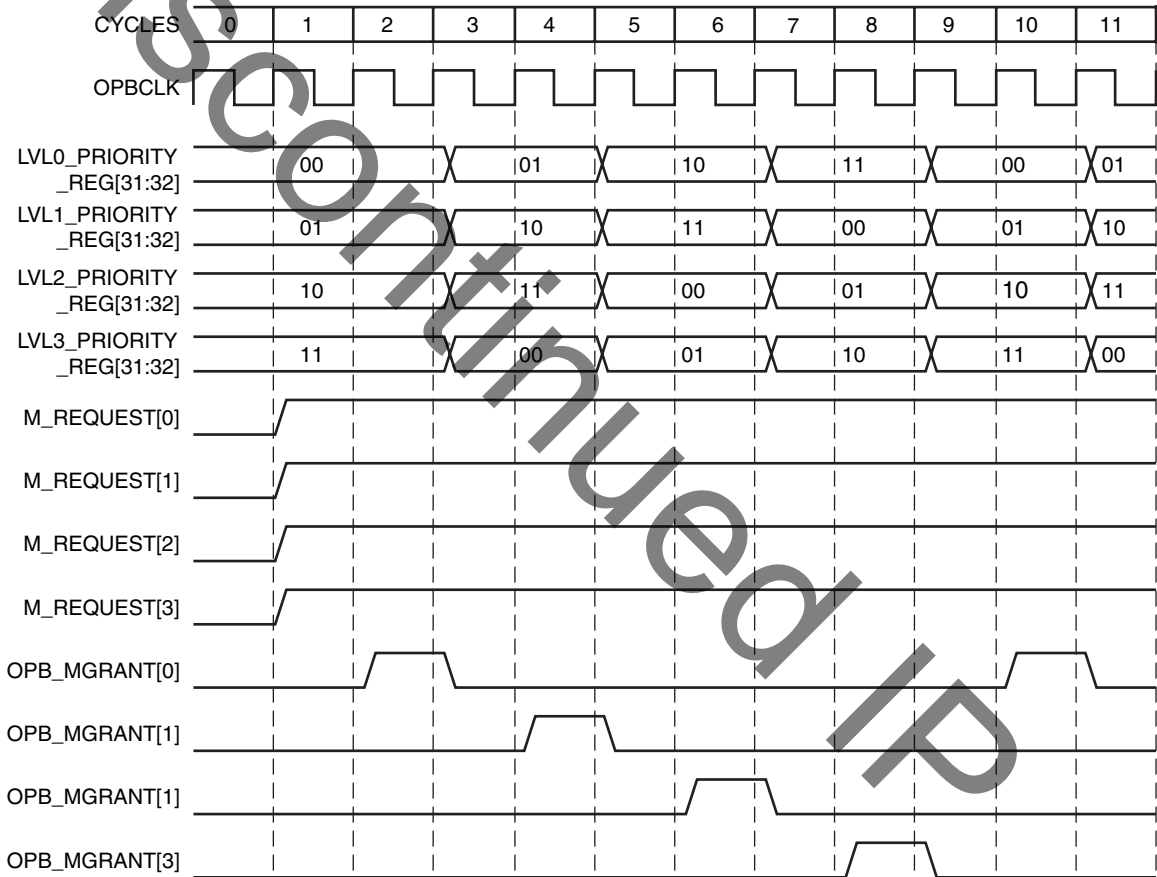
**Dynamic Priority Parameterization (C\_DYNAM\_PRIORITY=1)**

When the OPB Arbiter is parameterized to support dynamic priority arbitration, dynamic priority arbitration mode is enabled at reset or at any other time by writing a “1” to the DPE bit of the Control Register. Disabling dynamic priority arbitration mode by setting the DPE bit to “0” puts the OPB Arbiter into a fixed priority arbitration mode. This effectively freezes the values of the Priority registers (unless updated by an OPB write to the registers) and thus its ordering of arbitration priorities among the attached master devices. Setting the master priorities by software and not allowing them to update results in a static assignment of priority among the OPB masters.

Upon reset, the Priority registers contains the reset values as described in Table 6 and the dynamic priority bit is enabled. When dynamic priority arbitration mode is enabled, the contents of the Priority Registers are reordered after every request-grant cycle, moving the ID of the most recently granted master to the lowest Priority register and moving all other master IDs up one level of priority. The dynamic priority arbitration mode operation results in an implementation of the least recently used (LRU) algorithm. The lowest priority master ID will be the one which was granted the bus most recently, and the highest priority master ID will be the one which was granted the bus the least in the recent past.

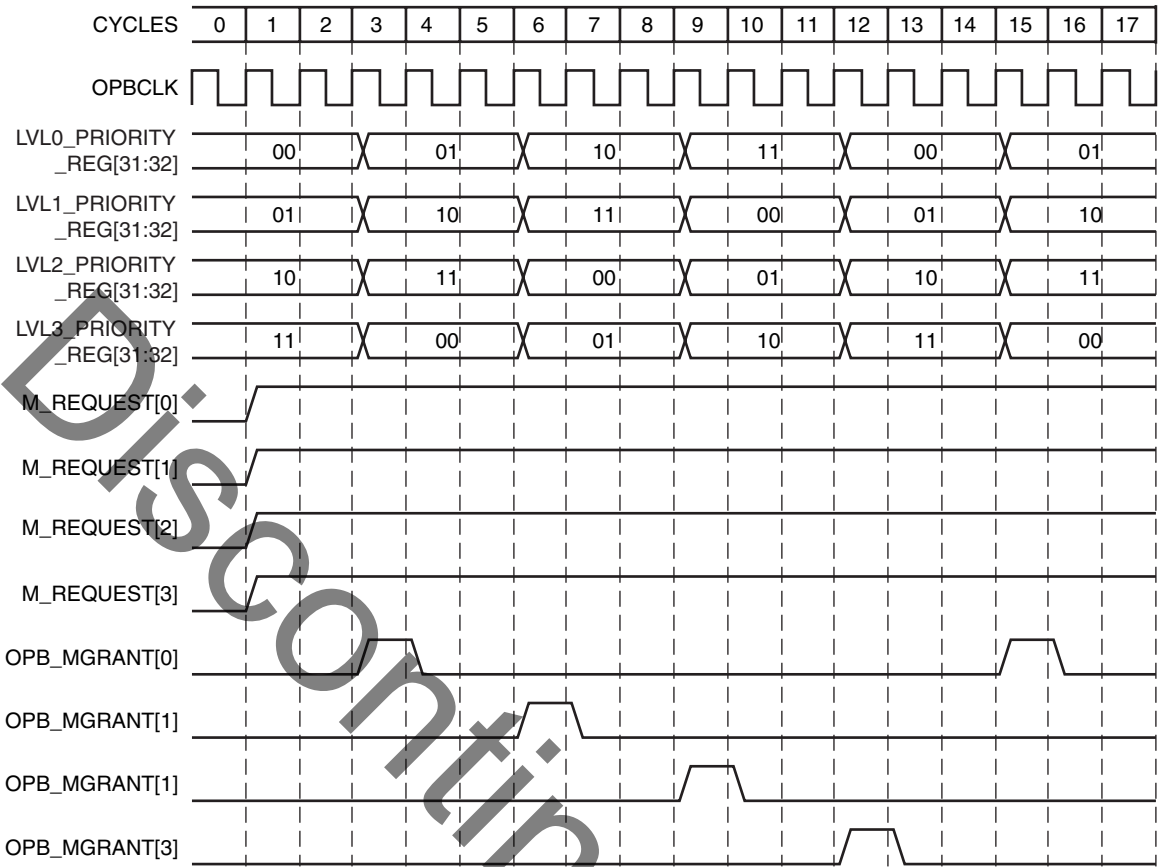
The values to be loaded into the Priority registers are either the values written to the register from the OPB or a shift of the master ID from the next lowest Priority register. In the case of the low Priority register, the ID of the master last granted the bus is loaded into this register. The master ID of the master granted the bus is loaded into the lowest Priority register and the IDs in all other Priority registers up to the priority of the one just granted move up one position in priority. The Priority registers above the one just granted hold their master ID values.

A pipeline register exists between the arbitration logic and Priority Register update logic which delays the master grant signals by an additional clock. Therefore, if the OPB Arbiter is configured for dynamic priority arbitration and registered grant outputs, the master grant signals will be output 2 clocks after a valid arbitration cycle. **Figure 4** shows the dynamic priority arbitration for a 4 OPB master system with combinational grant outputs. **Figure 5** shows dynamic priority arbitration for a 4 master OPB system with registered grant outputs.



DS469\_04\_092305  
Dynamic Priority Arbitration,  
Combinational Grant Outputs  
for 4 OPB Masters

Figure 4: Dynamic Priority Arbitration, Combinational Grant Outputs- 4 OPB Masters



DS469-05\_092305  
Dynamic Priority Arbitration,  
Registered Grant Outputs  
for 4 OPB Masters

Figure 5: Dynamic Priority Arbitration, Registered Grant Outputs- 4 OPB Masters

### Priority Registers and Register Update Logic Block Diagram

The block diagram for the Priority registers and the register update logic is shown in Figure 6. The gray shaded blocks represent the Priority Register update logic which is only present when the OPB Arbiter is parameterized to support Dynamic Priority arbitration.

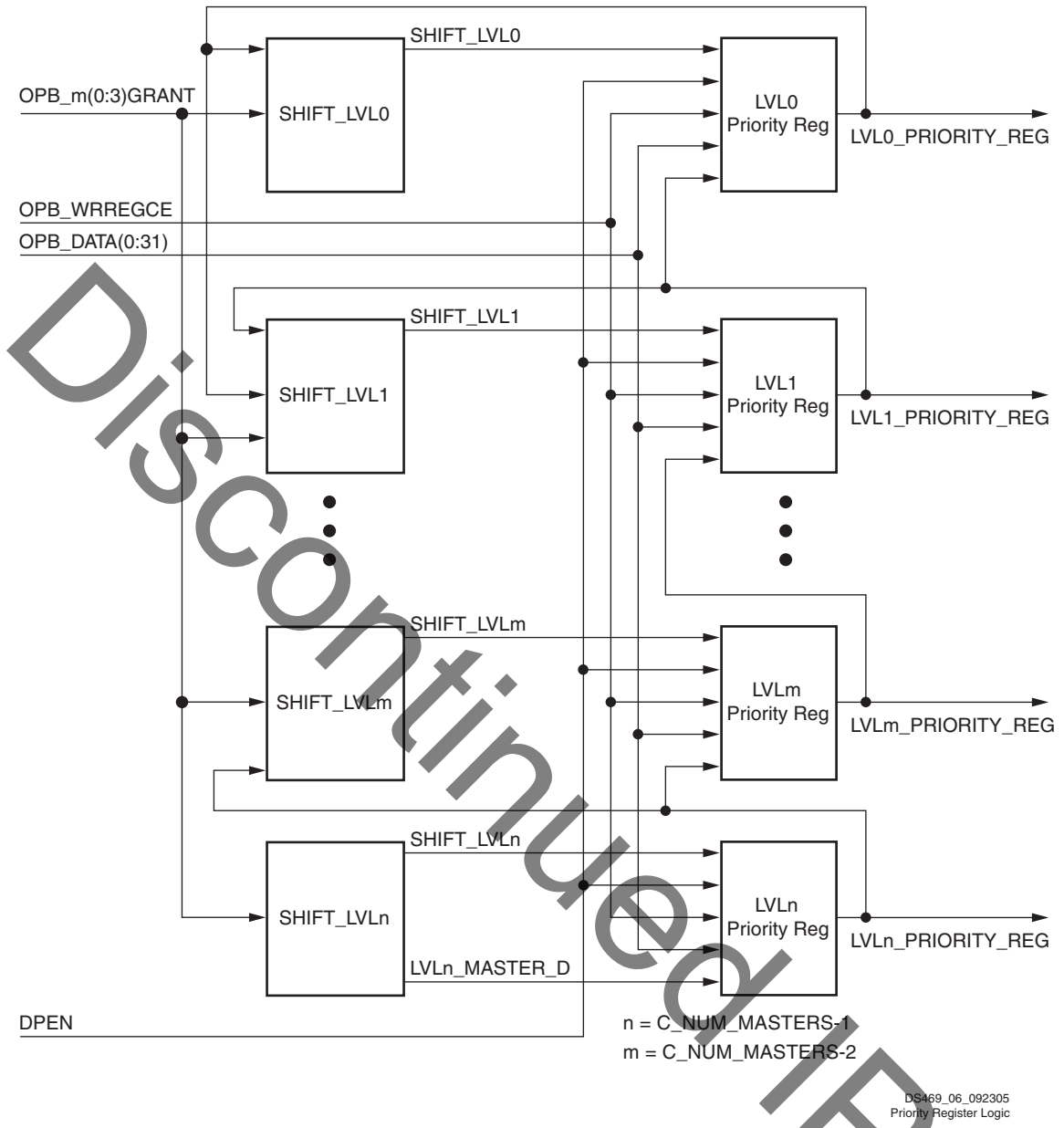


Figure 6: Priority Register Logic

When the OPB Arbiter has been parameterized to support a processor interface (C\_PROC\_INTRFCE=1), the Priority registers can still be loaded by the processor, allowing the processor to change the priorities of the OPB Masters. Also, the arbiter can be set to operate in a fixed priority mode by the processor writing to the Control Register and negating the DPE bit. However, the pipeline registers between the arbitration logic and the register update logic are still present, thereby delaying the grant outputs by an additional clock cycle.

## ARB2BUS Data Mux

When a read of the OPB Arbiter Priority Registers or the OPB Arbiter Control Register is requested on the OPB, the ARB2BUS Data Mux outputs the requested data to the IPIF which sends this data to the OPB with the required protocol. This logic is only present in the design if  $C\_PROC\_INTRFCE=1$  and  $C\_NUM\_MASTERS>1$ .

## Arbitration Logic

Figure 7 depicts the functional block diagram of arbitration logic for the OPB arbiter. This logic is only present in the design if  $C\_NUM\_MASTERS>1$ . The pipeline registers are only present in the arbitration logic if  $C\_DYNAM\_PRIORITY = 1$ .

All master request signals are input to the Prioritize Request block which consists of multiplexors which prioritize the master's requests into the signals  $lv10\_req$ ,  $lv11\_req$ ,  $lv1m\_req$ , and  $lv1n\_req$  based on the requesting masters' priorities if  $PRV = 1$  or the master IDs if  $PRV = 0$ . ( $n = C\_NUM\_MASTERS-1$ ,  $m = C\_NUM\_MASTERS-2$ ).

The prioritized request signals are then input into the priority encoder which determines which priority grant is asserted, i.e.,  $grant\_lv10$ ,  $grant\_lv11$ ,  $grant\_lv1m$ , and  $grant\_lv1n$ .

The prioritized grant signals are then input to the Assign Grants block to determine which master's grant signal is asserted based on the priority of that master, again determined by examination of the master IDs in the Priority Registers if  $PRV = 1$ , or the master IDs if  $PRV = 0$ . The master's priority code selects the appropriate prioritized grant signal to be output to that master. These intermediate grant signals are then registered if the OPB Arbiter is configured to support dynamic priority arbitration to reduce the number of logic levels between the arbitration logic and the Priority Register update logic. Because the Priority Register update logic is not present when the OPB Arbiter is not configured to support dynamic priority arbitration, these pipeline registers are not necessary and therefore are not present in the design.

The arbitration logic also contains the logic for detecting valid arbitration cycles which is input to the Park/Lock logic. Valid arbitration cycles are defined as when either the  $OPB\_select$  signal is deasserted indicating no data transfer is in progress or when  $OPB\_XferAck$  is asserted, indicating the final cycle in a data transfer and  $OPB\_busLock$  is not asserted.

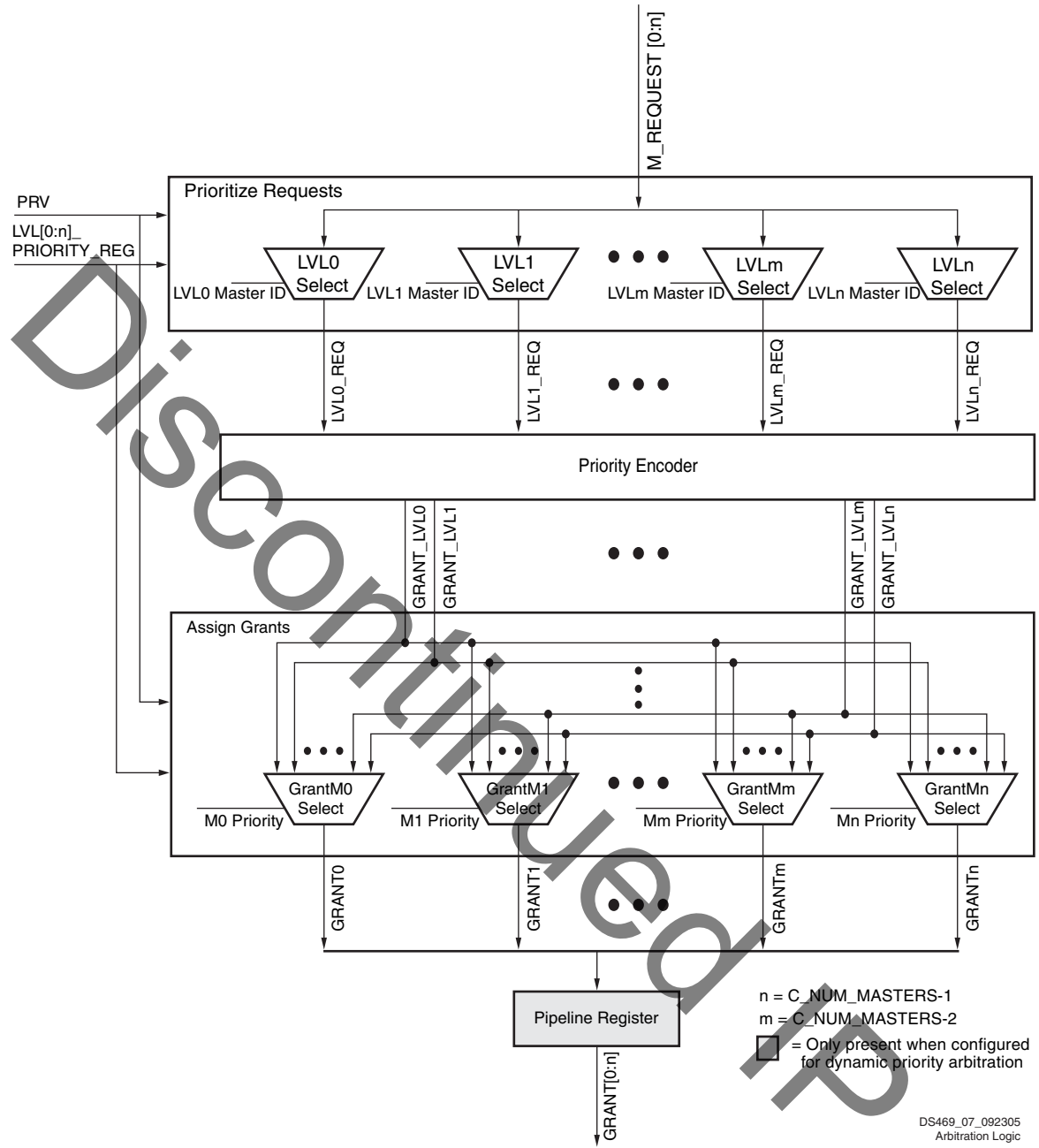
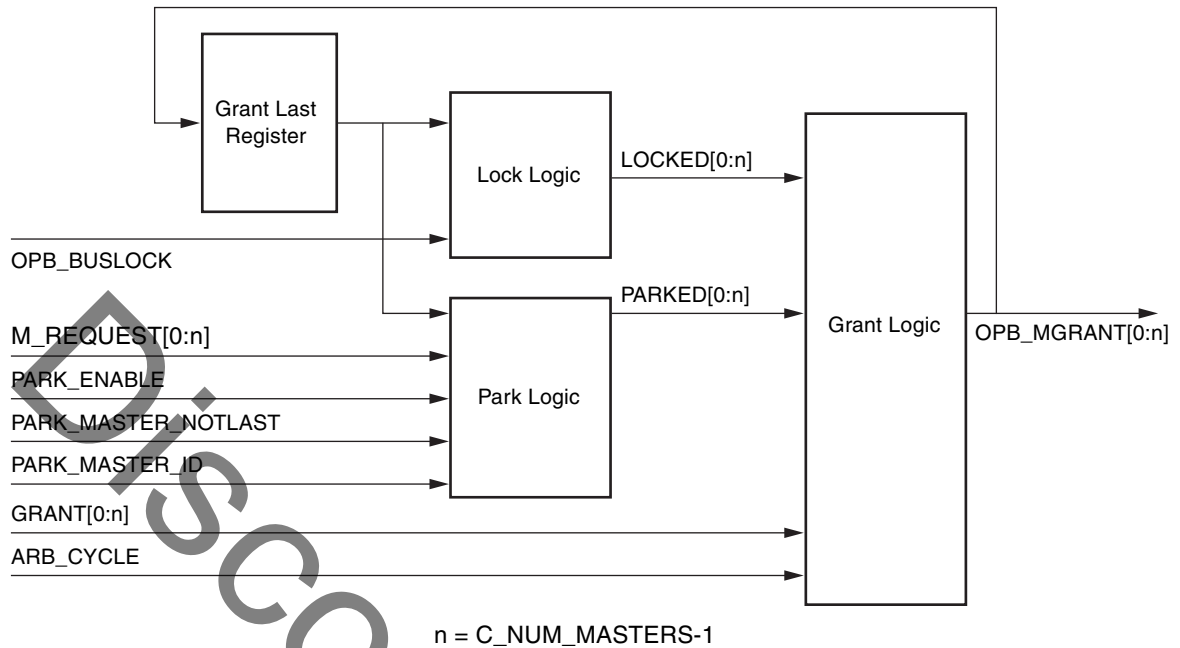


Figure 7: Arbitration Logic

### Park/Lock Logic

The Park/Lock logic processes the raw or registered grant outputs from the Arbitration Logic and is only present in the design if  $C\_NUM\_MASTERS > 1$ . It provides for the parking (if  $C\_PARK = 1$ ) and locking functionality of the OPB Arbiter and generates the final grant signals which are sent to the OPB master devices during valid arbitration cycles as shown in Figure 8. The OPB Arbiter can be parameterized to register the master grant signals by setting the parameter  $C\_REG\_GRANTS$  to 1.



DS469\_08\_092305  
Park/Lock Logic

Figure 8: Park/Lock Logic

### Grant Last Register

The Grant Last Register holds the state of the grant outputs from the last request/grant arbitration cycle. This information is used to determine which master currently has control of the bus to implement bus locking and park on last master parking.

### Lock Logic

If an OPB master asserts the bus lock signal upon assuming control of the bus, the OPB arbiter will continue to grant the OPB to the master which locked the bus. Bus lock signals from all attached masters are ORed together to form OPB\_busLock, which is an input to the OPB Arbiter. When OPB\_busLock is asserted, bus arbitration is locked to the last granted master (as indicated by the Grant Last register). All other master Grant outputs are gated off and will not be asserted, regardless of the state of the Request inputs or the programmed priorities.

When the OPB bus is locked, bus request and grant signals have no effect on bus arbitration. The OPB master may proceed with data transfer cycles while asserting bus lock without engaging in bus arbitration and without regard to the state of the request and grant signals. Grant signals will be generated if the master asserts its request signal. The locked master's grant signal will be asserted in response to its request signal during valid arbitration cycles. However, the locked master need not assert its request or receive an asserted grant signal to control the bus. The master owns the bus by virtue of asserting its bus lock signal after being granted the bus and before another valid arbitration cycle. The master which asserted bus lock will retain control of the bus until bus lock is deasserted for at least one complete cycle. The OPB arbiter will detect the bus lock signal and will continue to grant the bus to the current master, regardless of other (higher priority) requests. **Figure 9** shows the OPB bus lock operation when the OPB Arbiter is configured for fixed priority arbitration and combinational grant outputs. **Figure 10** shows the OPB bus lock operation when the OPB Arbiter is configured for

fixed priority arbitration and registered grant outputs. Note that the bus grant signal is asserted one clock later.

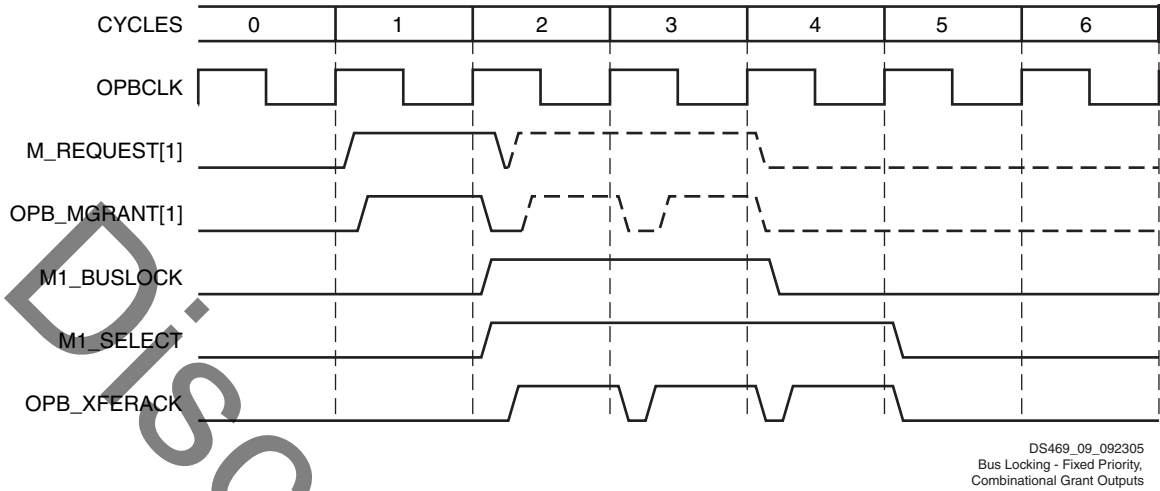


Figure 9: Bus Locking - Fixed Priority, Combinational Grant Outputs

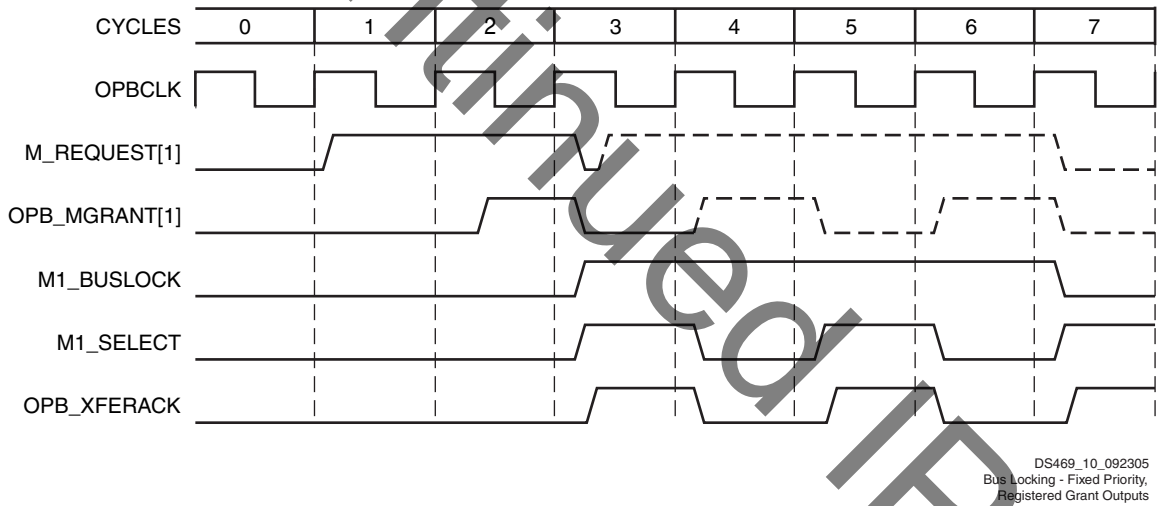


Figure 10: Bus Locking - Fixed Priority, Registered Grant Outputs

### Park Logic

When C\_PARK=1, the OPB Arbiter supports bus parking. Bus parking is when a master’s grant signal is asserted during valid arbitration cycles when no other master devices are requesting. This reduces latency for the parked master eliminating the need for a request/grant cycle when initiating a new OPB transfer.

Asserting a grant signal for parking is considered an arbitration, because it determines which device controls the bus. If dynamic priority mode is enabled, the ID of the parked master will be shifted to the lowest priority slot of the Priority Register. The master will remain parked (its grant signal asserted) so

long as no other master asserts a request signal. If the parked master and another master assert request at the same time, the parked master will control the bus because the bus was parked on this master even though this master is at a lowest priority. **Figure 11** illustrates this behavior when the OPB Arbiter is configured to support fixed priority arbitration and combinational grant outputs. **Figure 12** illustrates this behavior when the OPB Arbiter is configured to support dynamic priority arbitration when the OPB Arbiter is configured to support dynamic priority arbitration and registered grant outputs.

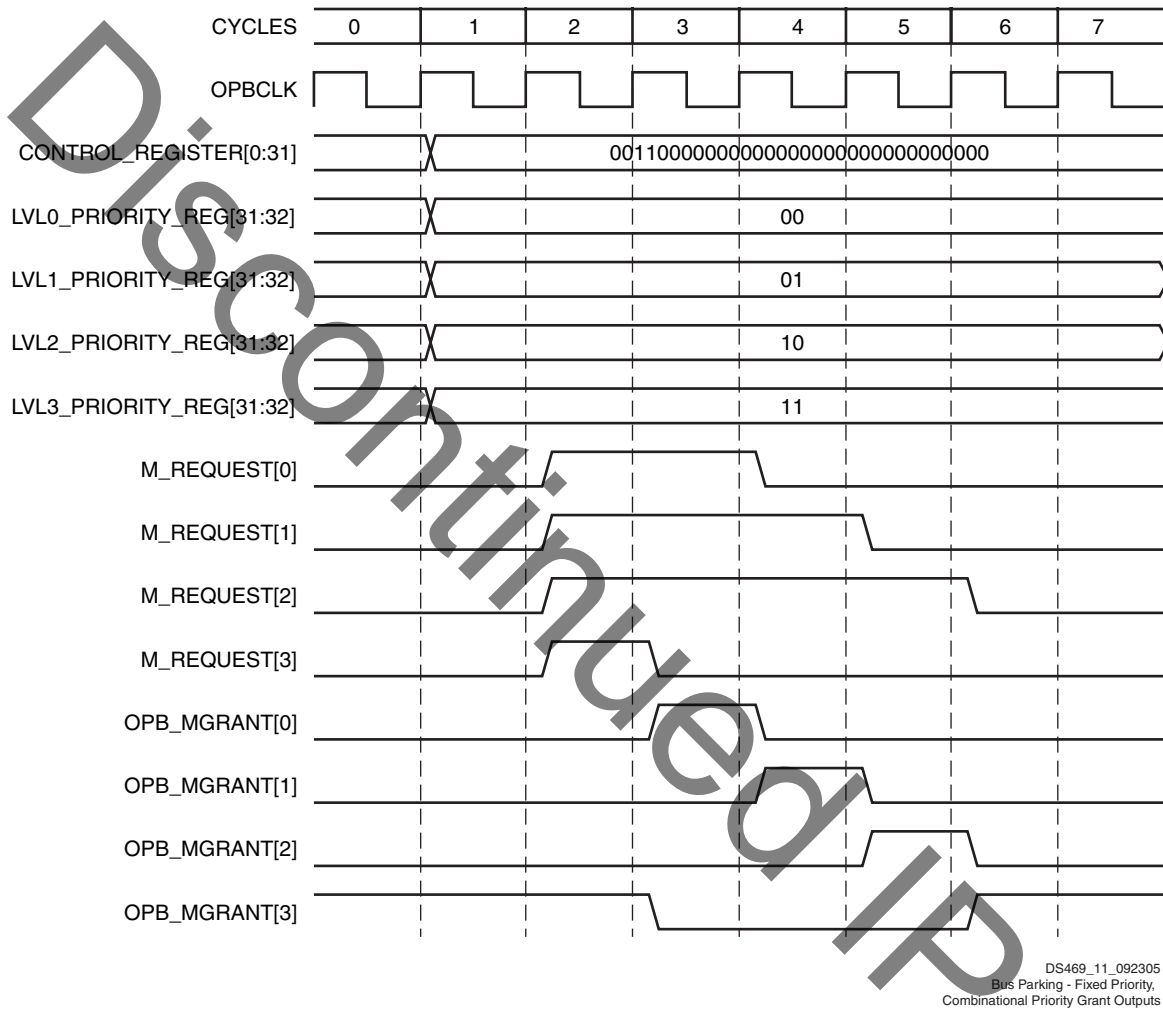


Figure 11: Bus Parking - Fixed Priority Arbitration, Combinational Grant Outputs

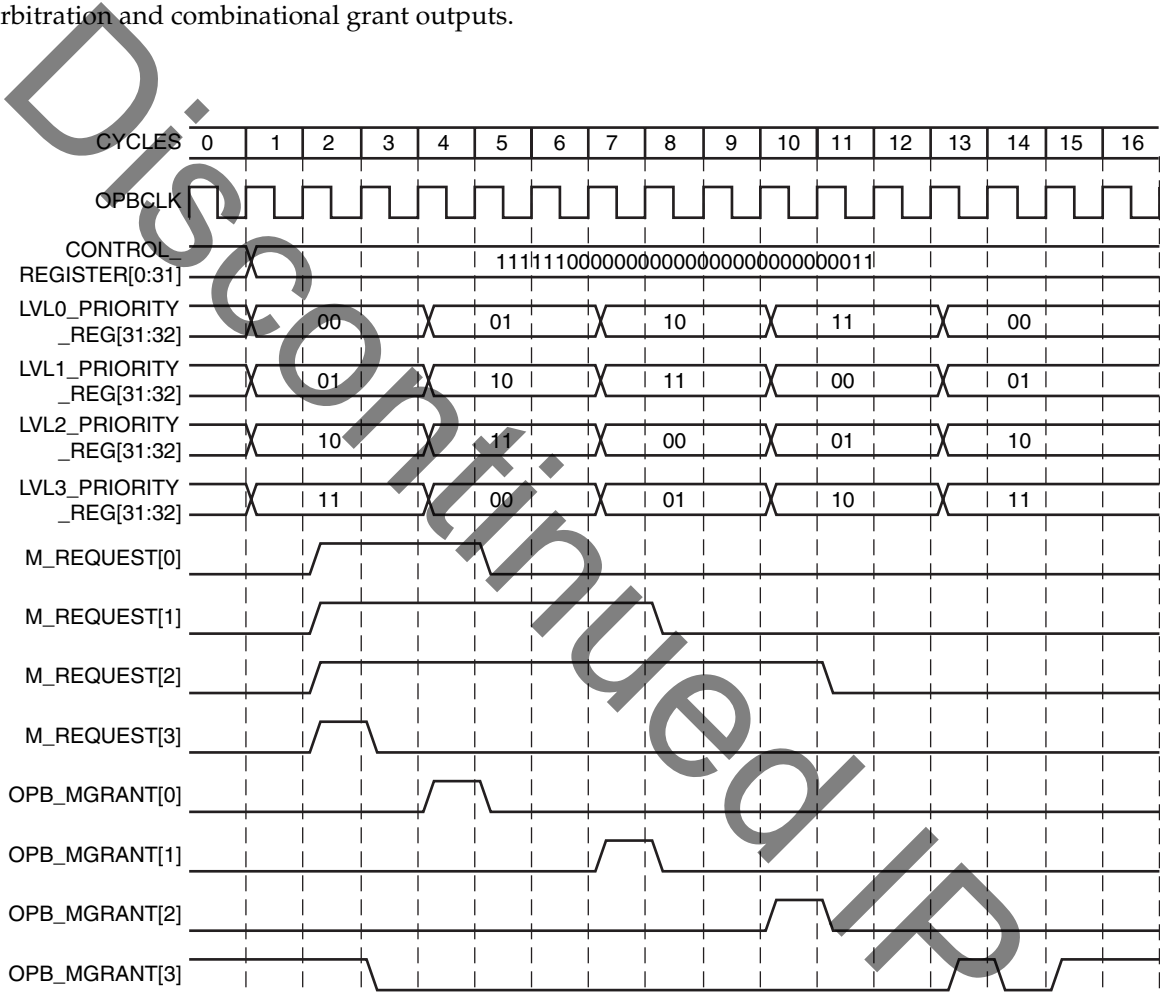
When the grant outputs are registered, the parked master's grant will assert once, negate, and then stay asserted. This allows the internal arbitration state machine a clock cycle to check if OPB\_select is asserted before parking. This case occurs when a request from a master is aborted, but the grant is in the internal pipeline.

When C\_PARK = 1, bus parking is enabled or disabled by the value of the Park Enable control bit in the OPB Arbiter Control Register (see **Table 5**). The bus can either be parked on the master who was last

granted the bus, or on a specified master as indicated by the Park Master ID bits in the OPB Arbiter Control Register. If bus parking is not desired, the park logic can be eliminated by setting C\_PARK=0.

**Park on Master Not Last**

When bus parking is enabled (bit PEN = 1 in the OPB Arbiter Control Register and C\_PARK=1) and Park on Master Not Last is selected (bit PMNL = 1 in the OPB Arbiter Control Register), the bus will be parked on the master whose ID is contained in the Park Master ID (PMID) field of the OPB Arbiter Control Register. This master’s grant signal will be asserted during valid arbitration cycles when no other master’s request signal is asserted. **Figure 13** shows bus parking on the master specified in the Control Register for a 4 OPB master system when the OPB Arbiter is parameterized for fixed priority arbitration and combinational grant outputs.



DS469\_12\_092305  
 Bus Parking - Dynamic Priority  
 Arbitration, Registered Grant Outputs

Figure 12: Bus Parking - Dynamic Priority Arbitration, Registered Grant Outputs

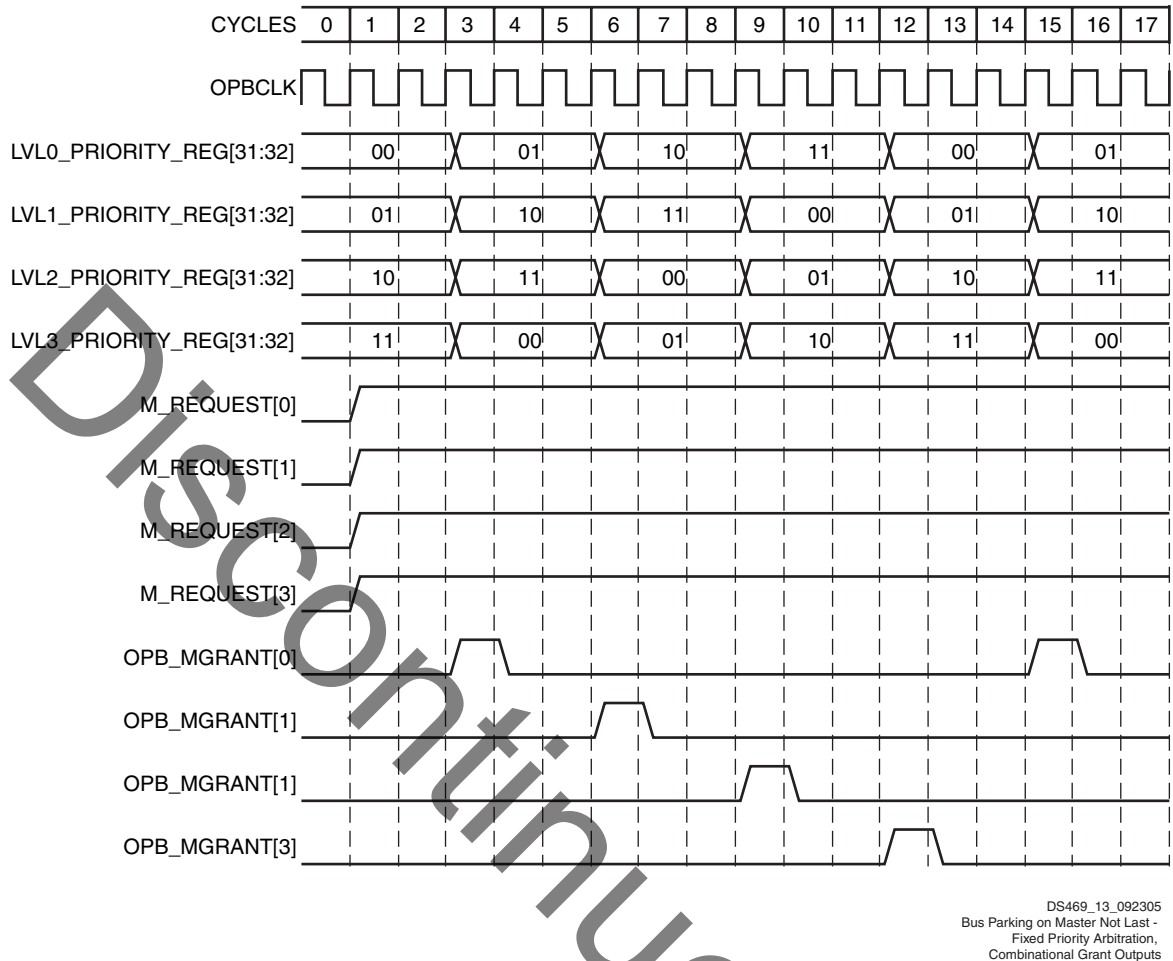
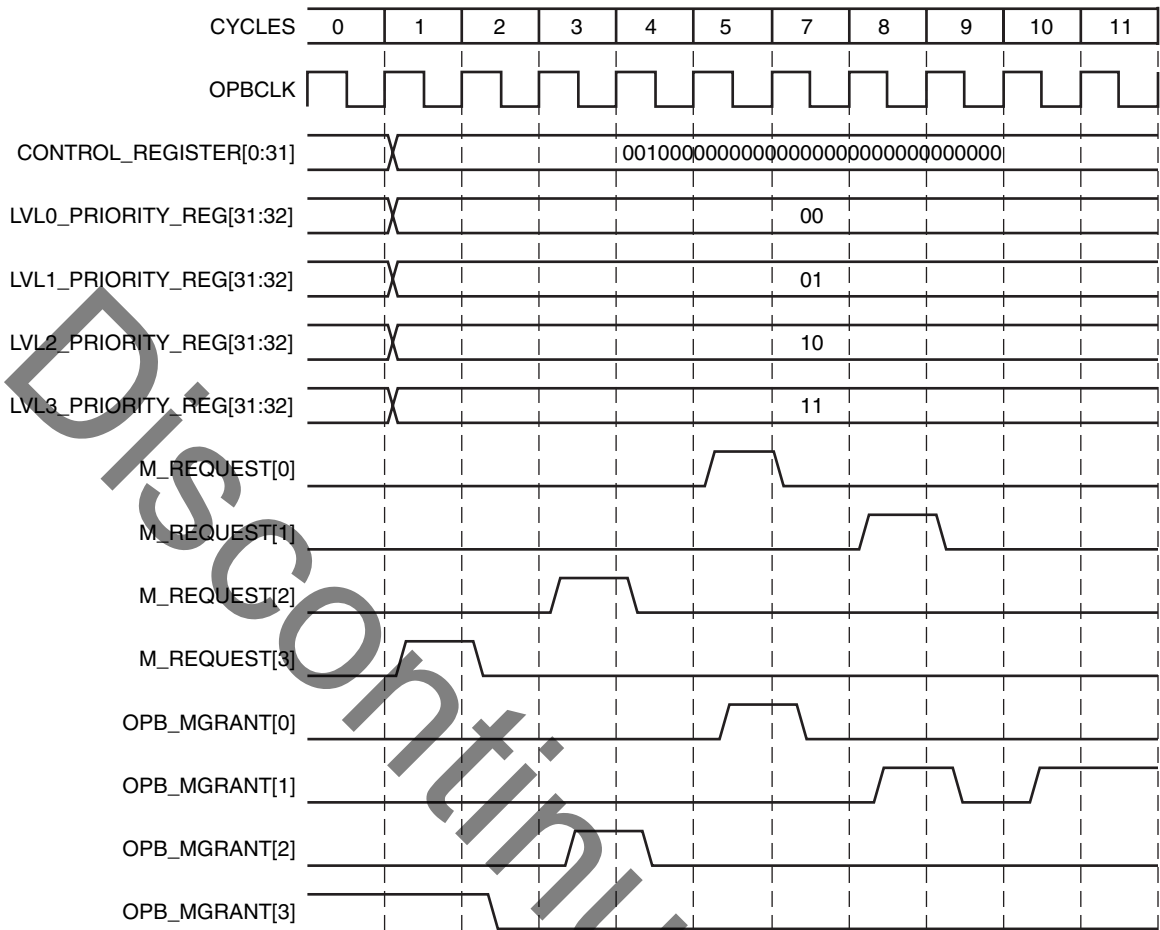


Figure 13: Bus Parking on Master Not Last - Fixed Priority Arbitration, Combinational Grant Outputs

**Park on Last Master**

When bus parking is enabled (bit PEN = 1 in the OPB Arbiter Control Register and C\_PARK=1) and Park on Master Not Last is negated (bit PMNL = 0 in the OPB Arbiter Control Register), the bus will be parked on the master which was most recently granted the bus as indicated by the Grant Last register. This master's grant signal will be asserted during valid arbitration cycles when no other master's request signal is asserted. Figure 14 shows bus parking on the last master with the OPB Arbiter parameterized for fixed priority arbitration and combinational grant outputs for a 4 OPB Master system.



DS469\_14\_092305  
 Bus Parking on Last Master -  
 Fixed Priority Arbitration,  
 Combinational Grant Outputs

Figure 14: Bus Parking on Last Master - Fixed Priority Arbitration, Combinational Grant Outputs

### Grant Logic

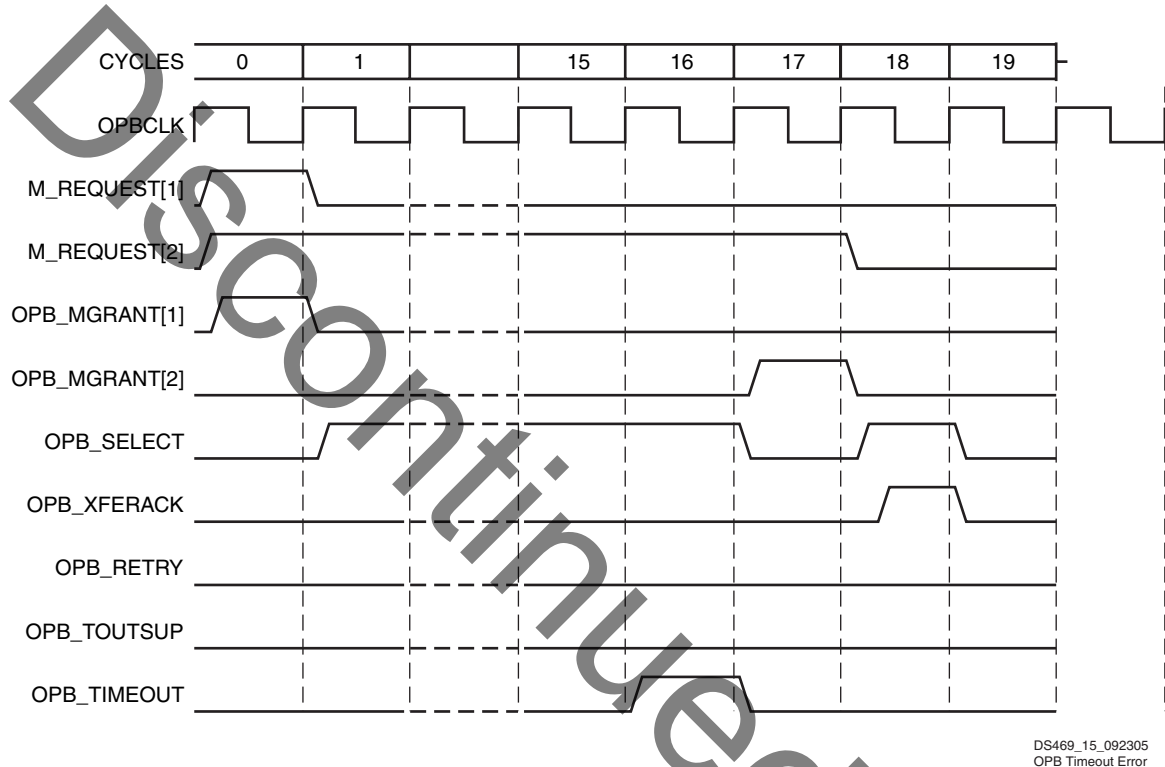
The Grant Logic block determines the final grant signals to the OPB Masters based on the intermediate grant signals from the arbitration logic and the results of the park and lock logic. The OPB Arbiter can be parameterized so that the grant signals are either registered outputs or combinational outputs. Registering the grant signals allows for higher OPB clock frequencies at the cost of a 1-cycle arbitration latency. Combinational grant outputs allow the grant signals to be asserted within the same clock cycle as the Master request signals, however, the overall clock frequency of the OPB will be affected. Basic OPB arbitration using registered grant outputs is shown in Figure 18.

### Watchdog Timer

The watchdog timer generates the OPB\_timeout signal within the 16th cycle following the assertion of OPB\_select if there is no response from a slave (OPB\_xferAck or OPB\_retry) and if toutSup is not asserted by an addressed slave device to suppress the timeout. This logic is always present in the OPB Arbiter design.

Upon assertion of OPB\_timeout, the master device which initiated the transfer cycle must terminate the transfer by deasserting Mn\_select in the cycle following the assertion of OPB\_timeout as shown in Figure 15. If OPB\_busLock is not asserted, the OPB Arbiter will perform a bus arbitration in the cycle in which OPB\_select is deasserted. If OPB\_busLock is asserted, the requesting master retains control of the OPB, but must still deassert Mn\_select following the assertion of OPB\_timeout for at least one cycle.

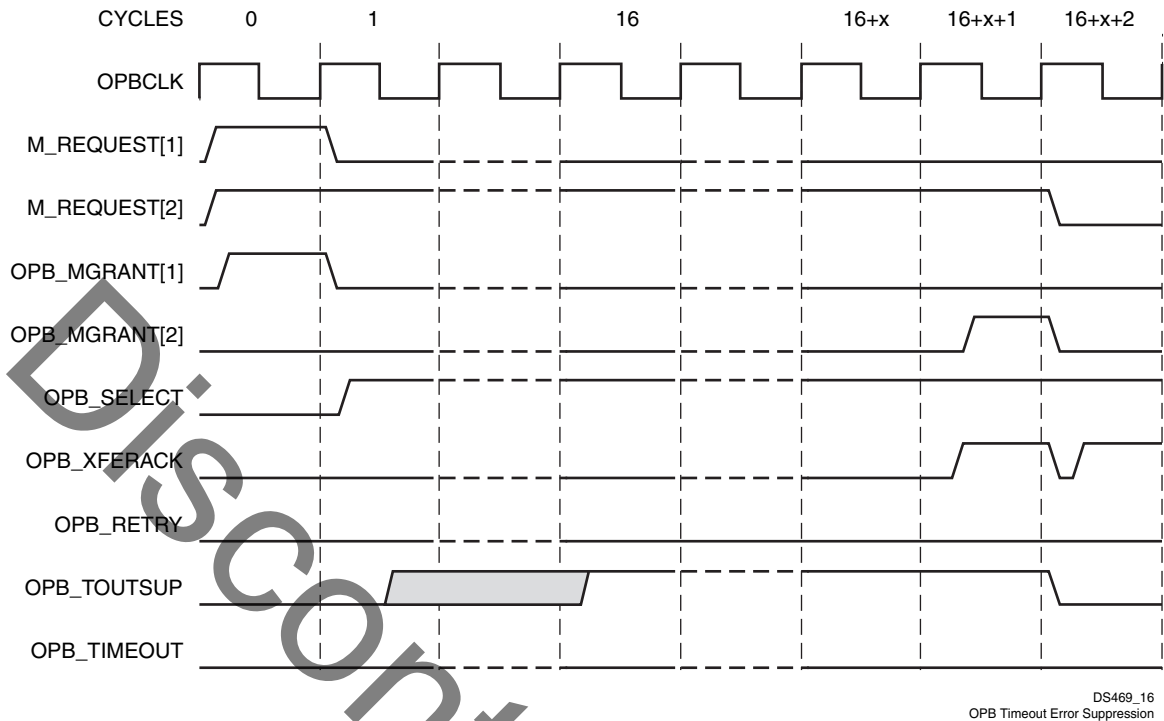
If OPB\_xferAck or OPB\_retry are asserted in the 16th cycle following the assertion of Mn\_select coincident to the assertion of OPB\_timeout, the master device should ignore OPB\_timeout, and respond to the slave's OPB\_xferAck or OPB\_retry signal.



DS469\_15\_092305  
OPB Timeout Error

Figure 15: OPB Timeout Error

To prevent a bus timeout, an OPB slave must assert OPB\_toutSup (OR of all slave's Sln\_ToutSup) within 16 cycles from the assertion of OPB\_select. OPB\_toutSup will be used by the OPB Arbiter to suppress the assertion of OPB\_timeout and to suspend the timeout counter. When OPB\_toutSup is asserted, the timeout counter holds its current value. When OPB\_toutSup is negated, the timeout counter resumes counting. OPB timeout error suppression is shown in Figure 16.



DS469\_16  
OPB Timeout Error Suppression

Figure 16: OPB Timeout Error Suppression

## OPB Arbiter I/O Signals

The I/O signals for the OPB Arbiter are listed in Table 1. The interfaces referenced in this table are shown in Figure 1 in the OPB Arbiter block diagram.

Table 1: OPB Arbiter I/O Signals

Port	Signal Name	Interface	I/O	Initial State	Description
<b>OPB Slave Signals</b>					
P1	ARB_DBus(0:C_OPB_DWIDTH-1)	IPIF	O	0	Arbiter output data bus
P2	ARB_xferAck	IPIF	O	0	Arbiter transfer acknowledge
P3	ARB_Retry	IPIF	O	0	Arbiter retry
P4	ARB_ToutSup	IPIF	O	0	Arbiter timeout suppress
P5	ARB_ErrAck	IPIF	O	0	Arbiter error acknowledge
P6	OPB_ABus(0:C_OPB_AWIDTH-1)	IPIF	I		OPB address bus
P7	OPB_BE(0:C_OPB_DWIDTH/8-1)	IPIF	I		OPB byte enables
P8	OPB_DBus(0:C_OPB_DWIDTH-1)	IPIF	I		OPB data bus
P9	OPB_RNW	IPIF	I		Read not Write (OR of all master RNW signals)
P10	OPB_seqAddr	IPIF	I		OPB sequential address

Table 1: OPB Arbiter I/O Signals (Contd)

Port	Signal Name	Interface	I/O	Initial State	Description
<b>Arbitration Signals</b>					
P11	M_request[0:C_NUM_MASTERS-1] <sup>(1)</sup>	Arbitration Logic	I		Request from OPB Masters
P12	OPB_xferAck	Arbitration Logic	I		Transfer Acknowledge indicating end of data transfer cycle (OR of all slave xferAcks)
P13	OPB_select	Arbitration Logic Watchdog Timer	I		Master has taken control of the bus (OR of all master selects)
P14	OPB_retry	Watchdog Timer	I		Bus cycle retry (OR of all slave retries)
P15	OPB_toutSup	Watchdog Timer	I		Suppress timeout (OR of all slave toutSup)
P16	OPB_timeout	Watchdog Timer	O	0	Timeout signal for OPB
P17	OPB_busLock	Park/Lock Logic	I		Bus lock (OR of all master buslocks)
P18	OPB_MGrant[0:C_NUM_MASTERS-1] <sup>(1)</sup>	Park/Lock Logic	O	0	Grant to OPB Masters
<b>System</b>					
P19	OPB_Clk	System	I		System clock
P20	OPB_Rst	System	I		System Reset (active high)
<b>Notes:</b>					
1. Name has been modified slightly from that in the IBM OPB Arbiter specification to support parameterization of the number of masters					

The signal, ARB\_DBusEn, is not an output of the Xilinx OPB Arbiter as the IPIF module internally gates the OPB Arbiter data bus with the enable signal.

The following signals listed in the IBM OPB Arbiter core are not supported:

- ARB\_sleepReq -the FPGA implementation of the OPB bus will not support sleep modes
- LSSD\_AClk - FPGA implementation does not support scan
- LSSD\_BClk - FPGA implementation does not support scan
- LSSD\_CClk - FPGA implementation does not support scan
- LSSD\_scanGate - FPGA implementation does not support scan
- LSSD\_scanIn - FPGA implementation does not support scan
- LSSD\_scanOut - FPGA implementation does not support scan

































