Device Control Register Bus 3.5 Architecture Specifications

SA14-2706-03

January 27, 2006

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.
IBM IBM Logo PowerPC PowerPC Logo PowerPC Architecture RISCTrace RISCWatch CoreConnect

Other company, product, and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation, life support, space, nuclear, or military applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN “AS IS” BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page can be found at http://www.ibm.com

The IBM Microelectronics Division home page can be found at http://www.ibm.com/chips

title.fm.03
January 27, 2006

Contents

Figures

Tables

About This Book

This book begins with an overview followed by detailed information on device control register bus signals, interfaces, timing and operations.

The device control register bus features:

- 10-bit up to 32-bit address bus
- 32-bit data bus
- 4-cycle minimum read or write transfers extendable by slave or master
- Handshaking supports clocked asynchronous transfers
- Multi-master arbitration
- Slave bus timeout inhibit capability
- Privileged and Non-Privileged transfers
- Slaves may be clocked either faster or slower than master
- Daisy-chain (serial) or distributed-OR (parallel) bus topologies
- A simple but flexible interface

Who Should Use This Book

This book is for hardware, software, and application developers who need to understand Core+ASIC development and system-on-a-chip (SOC) designs.The audience needs to understand embedded system design, operating systems, and the principles of computer organization.

How to Use This Book

This book is organized as follows:

To help readers find material in these sections, the book contains:

Conventions

The following is a list of notational conventions frequently used in this manual.

$\overline{\text{ActiveLow}}$	An overbar indicates an active-low signal.
n	A decimal number
0x*n*	A hexadecimal number
0b*n*	A binary number
=	Assignment
$\wedge$	AND logical operator
$\neg$	NOT logical operator
$\vee$	OR logical operator
$\oplus$	Exclusive-OR (XOR) logical operator
+	Twos complement addition
–	Twos complement subtraction, unary minus
$\times$	Multiplication
$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
||	Concatenation
=, $\neq$	Equal, not equal relations
<, >	Signed comparison relations
$\overset{u}{<}$, $\overset{u}{>}$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* is null, the else does not appear.
do	Do loop. “to” and “by” clauses specify incrementing an iteration variable; “while” and “until” clauses specify terminating conditions. Indenting indicates the scope of a loop.
leave	Leave innermost do loop or do loop specified in a leave statement.
FLD	An instruction or register field
FLD_b	A bit in a named instruction or register field
$FLD_{b:b}$	A range of bits in a named instruction or register field
$FLD_{b,b,\ldots}$	A list of bits, by number or name, in a named instruction or register field
REG_b	A bit in a named register
$REG_{b:b}$	A range of bits in a named register
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register

REG[FLD, FLD . . .]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, . . .
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	The contents of the register RA or 0, if the RA field is 0.
CR_{FLD}	The field in the condition register pointed to by a field of an instruction.
$c_{0:3}$	A 4-bit object used to store condition results in compare instructions.
^{n}b	The bit or bit value *b* is replicated *n* times.
xx	Bit positions which are don't-cares.
CEIL(x)	Least integer $\ge x$.
EXTS(x)	The result of extending *x* on the left with sign bits.
PC	Program counter.
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
MS(addr, n)	The number of bytes represented by *n* at the location in main storage represented by *addr*.
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies a location in main storage.
EA_b	A bit in an effective address.
$EA_{b:b}$	A range of bits in an effective address.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by *n*.
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.

instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.

Related Publications

1. Overview

The device control register (DCR) bus is designed to transfer data between a DCR master, typically a CPU's general purpose registers, and the DCR slave logic's device control registers. The DCR bus removes configuration registers from the memory address map, reduces loading, improving bandwidth of the processor local bus, and reducing latency to registers.

The DCR bus is a rising edge synchronous bus. Therefore, it is assumed that in a system-on-chip (SOC) environment where the DCR master and the DCR slave units are running at different clock speeds, the slower clock's rising edge always corresponds to the faster clock's rising edge. The DCR bus may be implemented in one of two possible configurations. It may be implemented by daisy-chaining the slave devices or by creating a distributed-OR structure out of the slave devices. The daisy chain approach may allow for easier chip-level wiring while the distributed-OR approach allows for easier chip-level timing closure.

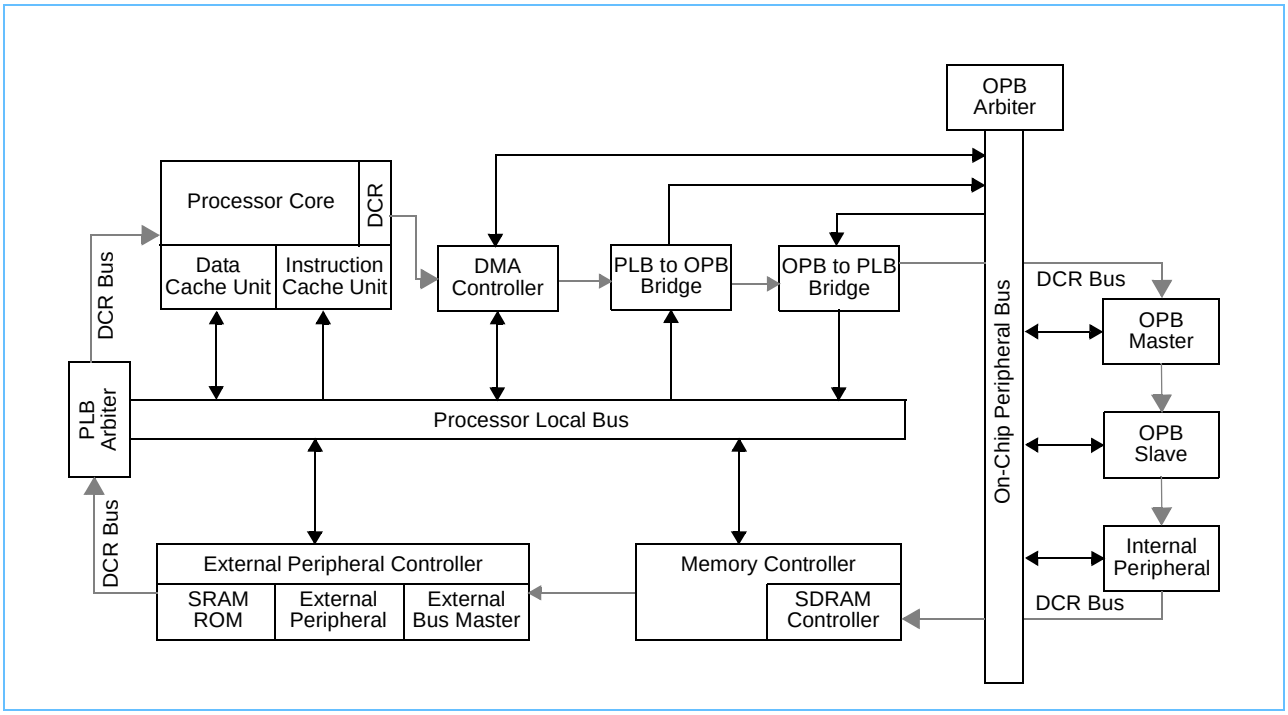
The DCRs are on-chip registers that reside architecturally outside of the DCR master. They are accessed by using DCR read and DCR write operations. For PowerPC CPUs performing DCR accesses, these instructions are known as the "move to device control register" (**mtdcr)** and "move from device control register" (**mfdcr)** instructions. Additional instructions in the PowerPC instruction set may be implemented for automatic handling of the privileged signal and for 32-bit addressing. DCR transactions may control the configuration of on-chip peripherals, such as memory controllers, DMA controllers, arbiters, bridges... etc.

Features of the device control register bus include:

- 10-bit up to 32-bit address bus
- 32-bit data bus
- 4-cycle minimum read or write transfers extendable by slave or master
- Handshaking supports clocked asynchronous transfers
- Multi-master arbitration
- Slave bus timeout inhibit capability
- Privileged and Non-Privileged transfers
- Slaves may be clocked either faster or slower than master
- Daisy-chain (serial) or distributed-OR (parallel) bus topologies
- A simple but flexible interface

Figure 1-1 illustrates how the device control register bus is an integral part of the CoreConnect-based system-on-a-chip designs.

Figure 1-1. Device Control Register Bus Interconnection



As shown in *Figure 1-1*, the embedded on-chip bus structure provides a link between the processor core and other peripherals which consist of Processor Local Bus (PLB) and On-Chip Peripheral (OPB) master and slave devices.

The processor local bus (PLB) is the high performance bus used to access memory through the bus interface units. The two bus interface units shown above: external peripheral controller and memory controller are PLB slaves. The processor core has two PLB master connections, one for instruction cache and one for data cache. Attached to the PLB is also the direct memory access (DMA) controller, which is a PLB master device used in data intensive applications to improve data transfer performance and unburden the CPU.

Lower performance peripherals (such as OPB master, slave, and other internal peripherals) are attached to the on-chip peripheral bus (OPB). A bridge is provided between the PLB and OPB to enable data transfer by PLB masters to and from OPB slaves. In the above example there are two bridges, a PLB to OPB bridge which is a slave on the PLB and a master on the OPB and an OPB to PLB bridge which is a slave on the OPB and a master on the PLB. OPB slaves may also be DMA peripherals.

The device control register (DCR) bus is used primarily for accessing status and control registers within the various PLB and OPB masters and slaves. It is meant to off-load the PLB from the lower performance status and control read and write transfers. It also decreases access latency during periods of high processor bus utilization. The DCR bus architecture allows PLB and OPB data transfers to occur concurrently with device register accesses by the CPU, data transfers between the processor and memory, or among other PLB devices.

2. DCR Bus Master and Slave Interfaces

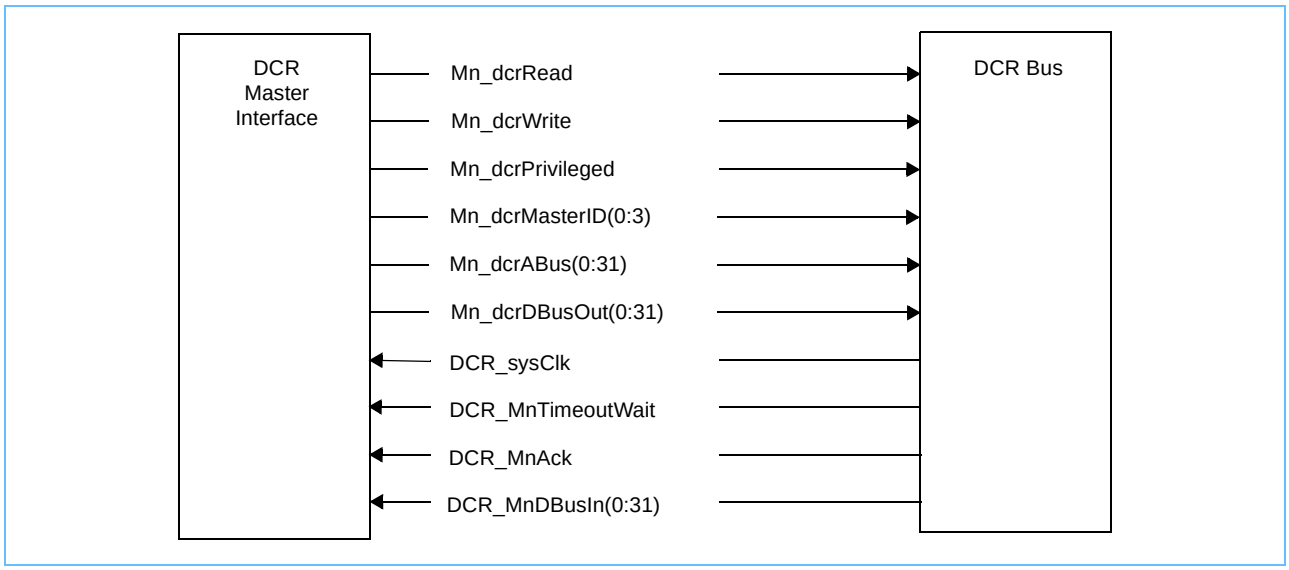
The DCR bus I/O signals are grouped under the following interface categories depending on their function.

For detailed functional description of the signals see *DCR Bus Signals* on page 19

2.1 DCR Master Interface

Figure 2-1 illustrates the master connection to the DCR bus. The master may connect directly to one or more slave or attach to a DCR arbitration unit to allow multiple DCR masters to arbitrate for control of a single slave bus segment. The Mn prefix refers to master number “n” where n can take on value from 0-15. For an implementation with only one master n should be 0.

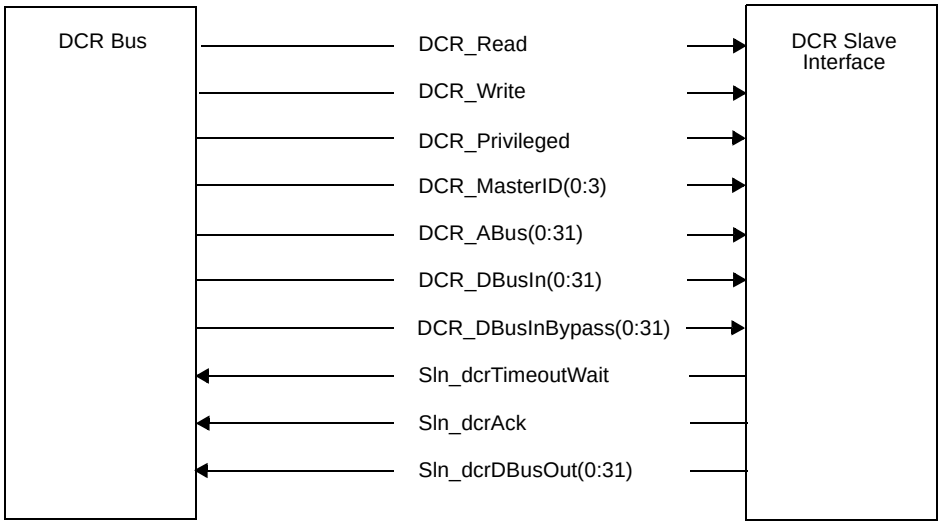
Figure 2-1. DCR Master Interface



2.2 DCR Slave Interface

Figure 2-2 illustrates the slave connection to the DCR bus. The master may connect directly to one or more slave or attach to a DCR arbitration unit to allow multiple DCR masters to arbitrate for control of a single slave bus. The Sln prefix refers to slave number “n” where n can take on any integer value.

Figure 2-2. DCR Slave Interface



3. DCR Bus Signals

This section describes DCR master and slave signals

3.1 DCR Master Signals

Table 3-1 provides a summary of all DCR signals followed by a brief description and page reference for detailed functional description.

Table 3-1. Summary of DCR Master Signals

Signal Name	DCR Master I/O	Description	Implementation	Page
Mn_dcrRead	O	Master DCR read command	Required	19
Mn_dcrWrite	O	Master DCR write command	Required	19
Mn_dcrPrivileged	O	Master DCR privileged operation	Optional	20
Mn_dcrMasterID(0:3)	O	Master DCR master ID indicator	Optional	20
Mn_dcrABus(0:31)	O	Master DCR address bus	(22:31) Required	20
Mn_dcrDBusOut(0:31)	O	Master DCR data bus out	Required	20
DCR_sysClk	I	DCR system clock	Required	20
DCR_MnTimeoutWait	I	DCR timeout counter inhibit	Required	21
DCR_MnAck	I	DCR transfer acknowledge	Required	21
DCR_MnDBusIn(0:31)	I	DCR to master data bus in	Required	21

3.1.1 Mn_dcrRead (Master DCR Read)

This is the control signal for a DCR read operation. This signal is asserted by the master to indicate that a read from a device control register operation is in progress and that the privileged transfer, master ID, and address bus signals are valid. Once the DCR read signal is asserted the master will wait a given number of clock cycles for the DCR_MnAck signal to be asserted indicating that the slave has acknowledged the read operation and is presenting the read data to the master via the DCR_MnDBusIn(0:31) signals. The master deasserts the Mn_dcrRead signal after sampling the assertion of the DCR_MnAck signal. The master must not assert the Mn_dcrRead or Mn_dcrWrite signal for a subsequent operation until it samples the deassertion of the DCR_MnAck signal. Following the assertion of the Mn_dcrRead signal if the DCR_MnAck signal is not asserted by the DCR slave within the time allotted by the master for DCR responses the master must deassert the Mn_dcrRead signal to terminate the read operation. In this case the bus has “timed out” and the master is required to abort the transfer. During DCR master reset this signal must remain deasserted.

3.1.2 Mn_dcrWrite (Master DCR Write)

This is the control signal for a DCR write operation. This signal is asserted by the master to indicate that a write to a device control register operation is in progress and that the privileged transfer, master ID, address bus, and data bus signals are valid. Once the DCR write signal is asserted the master will wait a given number of clock cycles for the DCR_MnAck signal to be asserted indicating that the slave has acknowledged the write operation. The master deasserts the Mn_dcrWrite signal, address bus, and the data bus signals after sampling the assertion of the DCR_MnAck signal. The master must not assert the Mn_dcrWrite or Mn_dcrRead signal for a subsequent operation until it samples the deassertion of the DCR_MnAck signal. Following the assertion of the Mn_dcrWrite signal if the DCR_MnAck signal is not asserted by the DCR slave

within the time allotted by the master for DCR responses the master must deassert the Mn_dcrWrite signal to terminate the write operation. In this case the bus has “timed out” and the master is required to abort the transfer. During DCR master reset this signal must remain deasserted.

3.1.3 Mn_dcrPrivileged (Master DCR Privileged)

This is the privileged transfer qualifier. This signal is asserted by the master to indicate that a read/write to device control register operation is to be considered a privileged operation. Only privileged transfer requests may access “privileged” register locations in DCR slaves. Privileged accesses to non-privileged registers will be allowed by the slave. Non-privileged accesses to privileged registers will be ignored by the slave and result in a bus time-out. When a DCR master attempts a privileged transaction this transfer qualifier is asserted during the entire interval that the Mn_dcrRead signal or Mn_dcrWrite signal is asserted. When a DCR master attempts a non-privileged transaction this transfer qualifier is deasserted during the entire interval that the Mn_dcrRead signal or Mn_dcrWrite signal is asserted. The master should only transition this signal when switching between a privileged and non-privileged transfer and vice versa.

For masters that do not implement the Mn_dcrPrivileged signal, all transfers from those masters should be considered “privileged”. Only when a master is required to differentiate between user and privileged transfers does this signal become required.

3.1.4 Mn_dcrMasterID(0:3) (Master DCR ID Bus)

This is the 4-bit DCR master identifier. These optional signals are valid anytime the Mn_dcrWrite signal or Mn_dcrRead signal is asserted and are considered invalid at all other times.

3.1.5 Mn_dcrABus(0:31) (Master DCR Address Bus)

This is the 32-bit DCR address bus. This bus must be valid and static anytime the Mn_dcrWrite or Mn_dcrRead signal is asserted and is considered invalid at all other times. Each individual address accesses a full 32-bit register location.

Note: DCR masters must implement a minimum of 10 address bits. (22:31) is the minimum implementation.

3.1.6 Mn_dcrDBusOut(0:31) (Master DCR Data Bus Out)

This is the DCR master write data bus. This bus is asserted by the master with valid DCR write data coincident with the assertion of the Mn_dcrWrite signal and must remain unchanged during the entire interval that the Mn_dcrWrite signal is asserted. This bus must be deasserted by the master coincident with the assertion of the Mn_dcrRead signal and must remain unchanged during the entire interval that the Mn_dcrRead signal is asserted. At all other times the bus is considered invalid.

3.1.7 DCR_sysClk (DCR System Clock)

This is the DCR master system clock. This signal provides the reference clock for DCR bus masters to generate read and write transactions. Master are required to assert and deassert all outputs with respect to the rising edge of this clock. For bus masters which fundamentally operate independent of this clock signal they are required to implement the necessary synchronization circuitry. Slave devices may be clocked completely synchronous to this signal or may be rising edge synchronous with this signal and operate at a higher or lower frequency. For reasons of re-use it is strongly recommended that slaves use the interface logic outlined in *DCR Slave Implementation* on page 31.

3.1.8 DCR_MnTimeoutWait (DCR Timeout Wait)

This signal inhibits the master's timeout counter. It is an indication that a DCR slave, or other chip-level logic such as a multi-master arbiter, has decoded and recognized a valid transfer request destined for it but is unable to respond before the DCR master would recognize a bus timeout condition. When a DCR master samples the assertion of the DCR_MnTimeoutWait signal during a DCR read or write operation it must inhibit its internal timeout counter from counting until such time this signal is sampled deasserted.

The DCR_MnTimeoutWait signal may be asserted anytime a valid address and command are on the DCR bus. It is not required to be deasserted before the assertion of the DCR_MnAck signal.

The DCR_MnTimeoutWait signal can be used in a multi-DCR-master system as a means of preventing one DCR master from timing out while attempting a DCR operation coincident with another master's utilization of the DCR bus. See *DCR Arbiter Implementation Example* on page 40 for more details.

3.1.9 DCR_MnAck (DCR Acknowledge)

This is the DCR transfer acknowledge signal. This signal indicates that a DCR slave has recognized the assertion of either the Mn_dcrWrite or Mn_dcrRead signal decoded the transfer qualifiers and claimed the transfer. DCR_MnAck remains asserted as long as the Mn_dcrWrite signal or Mn_dcrRead signal is sampled asserted by the slave. Once the master samples the DCR_MnAck signal asserted it terminates the read or write operation by deasserting the Mn_dcrWrite signal or Mn_dcrRead signal. The master must not re-assert the Mn_dcrWrite signal or Mn_dcrRead signal for a subsequent operation until it samples the DCR_MnAck signal deasserted.

3.1.10 DCR_MnDBusIn(0:31) (DCR Master Data Bus In)

This is the DCR master read data bus. The bus is valid coincident with the assertion of the DCR_MnAck signal and remains valid and static as long as the Mn_dcrRead signal is asserted. This bus is invalid at all other times. The DCR slave logic will drive this bus with the contents of the selected register during a DCR read operation as long as the address, privilege level, and master ID are claimed by the DCR slave and the Mn_dcrRead signal is asserted.

3.2 DCR Slave Signals

Table 3-2 provides a summary of all DCR signals followed by a brief description and page reference for detailed functional description.

Table 3-2. Summary of DCR Slave Signals

Signal Name	DCR Slave I/O	Description	Implementation	Page
DCR_Read	I	DCR read command	Required	22
DCR_Write	I	DCR write command	Required	22
DCR_Privileged	I	DCR privileged operation	Optional	23
DCR_MasterID(0:3)	I	DCR master ID indicator	Optional	23
DCR_ABus(0:31)	I	DCR address bus	Required	23
DCR _DBusIn(0:31)	I	DCR data bus in	Required	23
DCR_DBusInBypass(0:31)	I	DCR data bus bypass	Optional	24
OneToOneMode	I	Slave using DCR_sysClk	Optional	24
Sln_dcrTimeoutWait	O	Slave DCR timeout wait indicator	Optional	24
Sln_dcrAck	O	Slave DCR acknowledge	Required	25
Sln_dcrDBusOut(0:31)	O	Slave DCR data bus out	Required	25

3.2.1 DCR_Read (DCR Read)

This is the control signal for a DCR read operation. This signal is asserted to indicate that a read from device control register operation is in progress and that the privileged transfer, master ID, and address bus signals are valid. Once the DCR read signal is asserted a slave must decode all the transfer qualifiers to determine if the read access is for one of its registers. In response the slave must either assert the Sln_dcrAck signal and coincidently provide the read data on the Sln_dcrDBusOut(0:31) signals or extend the cycle by asserting the Sln_dcrTimeoutWait signal. The DCR_Read signal is deasserted following the assertion of the Sln_dcrAck signal. Slaves may block access to a register if it decodes an unauthorized privilege level or master ID. In this case the slave does not acknowledge the transfer and the bus times out. This signal is deasserted during reset.

3.2.2 DCR_Write (DCR Write)

This is the control signal for a DCR write operation. This signal is asserted to indicate that a write to a device control register operation is in progress and that the privileged transfer, master ID, address bus, and data bus signals are valid. Once the DCR write signal is asserted a slave must decode all the transfer qualifiers to determine if the write access is for one of its registers. In response the slave must either assert the Sln_dcrAck signal and coincidently latch the write data on the DCR_DBusIn(0:31) signals or extend the cycle by asserting the Sln_dcrTimeoutWait signal. The DCR_Write signal is deasserted following the assertion of the Sln_dcrAck signal. Slaves may block access to a register if it decodes an unauthorized privilege level or master ID. In this case the slave does not acknowledge the transfer and the bus times out. This signal is deasserted during reset.

3.2.3 DCR_Privileged (DCR Privileged)

This is the privileged transfer qualifier. This signal is asserted to indicate that the current read or write to device control register operation is to be considered a privileged operation. Only privileged transfer requests may access “privileged” register locations in DCR slaves. Privileged accesses to non-privileged registers must be allowed by the slave. Non-privileged accesses to privileged registers must be ignored by the slave and result in a bus time-out. When a DCR master attempts a privileged transaction this transfer qualifier is asserted during the entire interval that the DCR_Read signal or DCR_Write signal is asserted. When a DCR master attempts a non-privileged transaction this transfer qualifier is deasserted during the entire interval that the DCR_Read signal or DCR_Write signal is asserted. This signal should only transition when switching between a privileged and non-privileged transfer and vice versa.

3.2.4 DCR_MasterID(0:3) (DCR Master ID Bus)

This is the 4-bit DCR master identifier. These signals are valid anytime the DCR_Read signal or DCR_Write signal is asserted and are considered invalid at all other times. Slaves may use these signals to restrict access to registers on a per-master basis. Up to 16 masters are supported.

Table 3-3. Summary of DCR Master IDs

DCR Master ID	Description
0000	Master 0
0001	Master 1
0010	Master 2
0011	Master 3
0100	Master 4
.	.
.	
1110	Master 14
1111	Master 15

3.2.5 DCR_ABus(0:31) (DCR Address Bus)

This is the 32-bit DCR address bus. This bus must be valid and static anytime the DCR_Read or DCR_Write signal is asserted and is considered invalid at all other times. Each individual address accesses a full 32-bit register location. Slaves may implement less than the full 32-bit address bus. A minimum of 10-bits must be supported. (22:31) Slaves with less than 32-bits of address are “right justified” to the lower significant portion of the address bus.

Note: Legacy DCR slaves that implement a 10-bit address bus attach to bits (22:31). See *Implementation with 10-bit and 32-bit Addressable DCR Slaves* on page 46.

3.2.6 DCR_DBusIn(0:31) (DCR Data Bus In)

This is the DCR data bus in to the slaves registers. This bus is asserted with valid and static DCR data during a write to DCR register operation for the entire interval that the DCR_Write signal is asserted. During reset this bus is deasserted and may be used by the DCR slave to initialize its registers.

Note: From the DCR slave's perspective, this bus may be driven directly by the a DCR master, DCR arbiter, or another's unit bypass, as illustrated in *Figure 6-7* on page 40.

3.2.7 DCR_DBusInBypass(0:31) (Data Bus In Bypass)

This is the DCR data bus bypass input to the slave. If the slave implements DCR_DBusInBypass(0:31), then the slave is required to route the value of this data bus to its Sln_dcrDBusOut(0:31) signals at all times unless the slave is selected during a read operation, DCR_Read asserted, and is asserting valid read data. The DCR_DBusInBypass(0:31) is typically connected to the DCR_DBusIn(0:31) data bus or to ground depending on the bus topology implemented.

Note: From the DCR slave's perspective, this input may be driven directly by the a DCR master, DCR arbiter, or another unit's bypass, as illustrated in *Figure 6-7* on page 40. It may also be tied directly to ground.

If the slave does not implement DCR_DBusInBypass(0:31), then it is strongly recommended that the slave should drive the value on the DCR_DBusIn(0:31) input to the Sln_dcrDBusOut(0:31) at all times unless the slave is selected during a read operation. This is considered the legacy method of providing DCR data bypass. When a slave implements this legacy method of data bypassing, the chip integrator does not have the option to tie the bypass data to ground when it is implementing a distributed-OR topology.

Note: All diagrams in this specification will show the use of the optional DCR_DBusInBypass(0:31) since this is considered the preferred implementation.

3.2.8 OneToOneMode (One to One Clock Mode)

This is the one to one clock mode enable input to the slave. This input is strapped by the chip integrator to indicate to the slave whether or not it is being clocked by the DCR system clock. When this signal is strapped asserted the slave is clocked by the DCR system clock signal, DCR_sysClk, and need not synchronize the decoded assertion of the DCR_Read and DCR_Write signals. Instead it may simply use the decoded assertion directly to generate the slave Sln_dcrAck signal assertion. When this signal is strapped deasserted the slave is not clocked by the DCR system clock signal, DCR_sysClk, and must synchronize the decoded assertion of the DCR_Read and DCR_Write signals prior to asserting the Sln_dcrAck signal. See *DCR Slave Implementation* on page 31 for more details. This signal is intended to be static but may switched when the slave or DCR master clock frequency is changed. In this case this signal may only transition when it is guaranteed that no transfers to or from the slave are being performed.

3.2.9 Sln_dcrTimeoutWait (Slave DCR Timeout Wait)

This optional signal is asserted by a DCR slave to indicate that it has recognized the DCR address and transfer qualifiers as a valid access to one of its registers but is unable to respond within the time allotted before the DCR bus master times out. DCR masters must implement an internal timeout counter and must sample the assertion of DCR_MnTimeoutWait signal during a DCR read or write operation and temporarily disable their timeout counter until such time this signal is sampled deasserted.

Note: The Sln_dcrTimeoutWait signal may be asserted anytime a valid address and command are on the DCR bus, and is not required to be negated before the assertion of the Sl_dcrAck signal.

3.2.10 Sln_dcrAck (Slave DCR Acknowledge)

This is the slave DCR transfer acknowledge signal. This signal indicates that the DCR slave has recognized the assertion of either the DCR_Read signal or DCR_Write signal and claimed the address on the DCR_ABus(0:31) signals. Once asserted the slave must continue to assert the Sln_dcrAck signal until it samples the deassertion of the DCR_Read or DCR_Write signal for the transfer. During a read transfer the slave may only assert the Sln_dcrAck signal when it is providing valid read data on the Sln_dcrDBusOut(0:31) signals. During a write transfer the slave should not assert the Sln_dcrAck signal until it has latched the write data present on the DCR_DBusIn(0:31) signals.

3.2.11 Sln_dcrDBusOut(0:31) (Slave DCR Data Bus Out)

This is the DCR data bus out the slave. The slave is required to route the DCR_DBusInBypass(0:31) signals to this data bus at all times unless it is selected during a read operation, DCR_Read asserted, and is asserting valid read data.

Note: From the DCR slave's perspective, this bus may be routed directly to a DCR master, DCR arbiter, or another unit's DCR_DBusIn(0:31) and DCR_DBusInBypass(0:31) signals, as illustrated in *Figure 6-7* on page 40.

The DCR slave may continue to drive valid read data on the Sln_dcrDBusOut(0:31) after the deassertion of DCR_Read up until the slave deasserts Sln_dcrAck, at which time the slave must drive the Sln_dcrDBusOut(0:31) back to the bypass data value DCR_DBusInBypass(0:31). This is allowed because the DCR transaction is not considered completed until the deassertion of Sln_dcrAck. The master device will not consider this data valid after the deassertion of DCR_Read.

4. DCR Bus Topologies

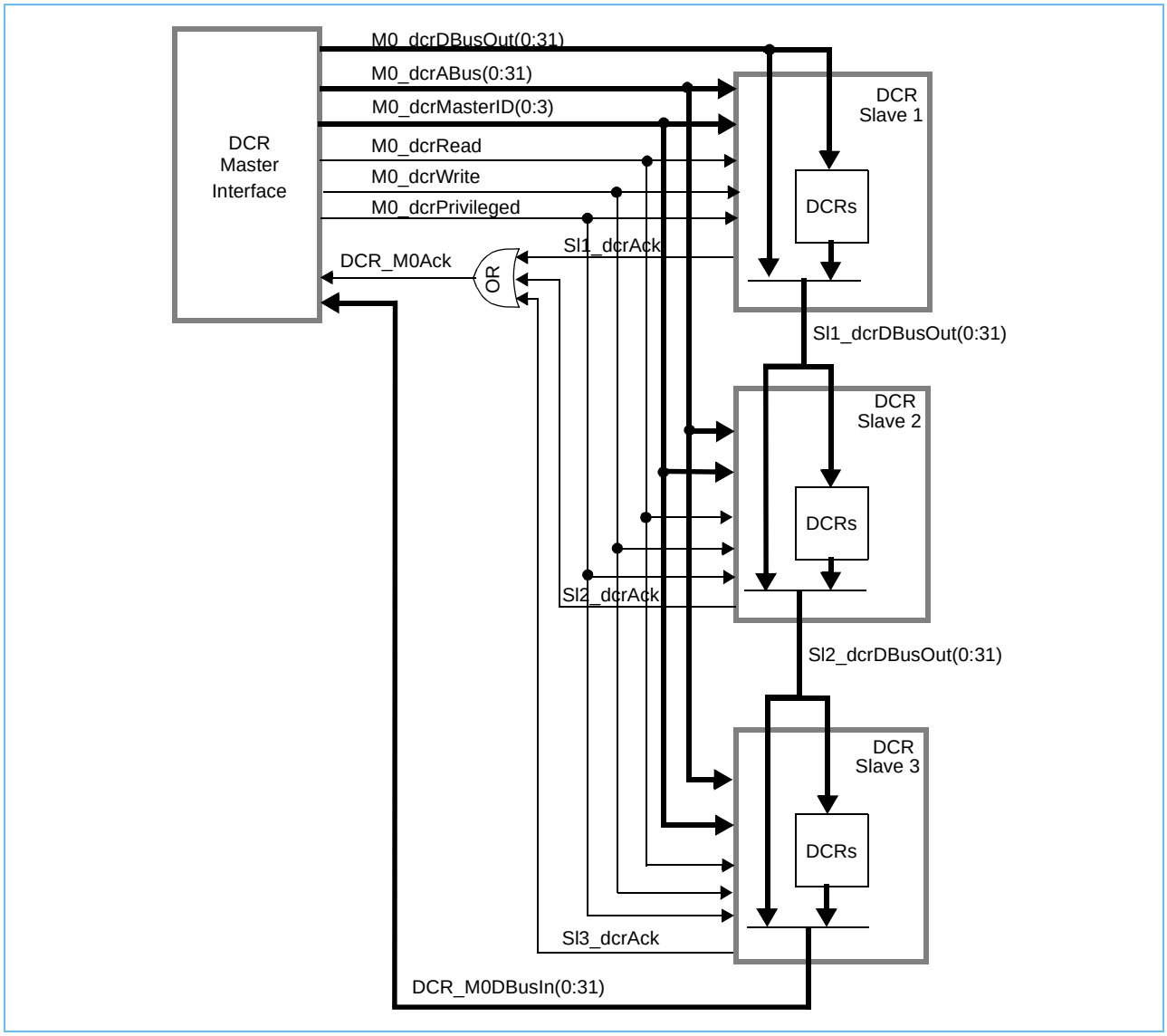
The device control register bus can be implemented in several different structural topologies. Certain topologies are better suited for easy physical design and others for higher frequency timing closure. Several recommended configurations are described in the following sections. It is ultimately the systems-on-a-chip architect's decision which topology, or combination of topologies to implement.

4.1 Daisy-Chain Topology

Figure 4-1 provides a block diagram of a device control register bus topology using a “daisy-chained” approach for the DCR data path. In the daisy-chain topology the master’s outbound data bus signals are received by the first DCR slave unit in the chain. The data flows through the chain network, where each slave unit either passes along the unmodified data bus input to its output or, if it is the slave being read from, places its read data on its data bus output. Each DCR slave implements a bypass mux to keep the path delay to a minimum when the slave is not selected. In this structure the slave DCR_DBusInBypass(0:31) signal inputs must be attached to the preceding master or slave dcrDBusOut(0:31) signals. Further description of the slave implementation is available in *DCR Slave Implementation* on page 31.

Chip integrators may find this topology allows for easier wiring within an SOC design containing slaves “placed around the chip” with limited wiring to a central location, and possibly operating at a slower clock frequency.

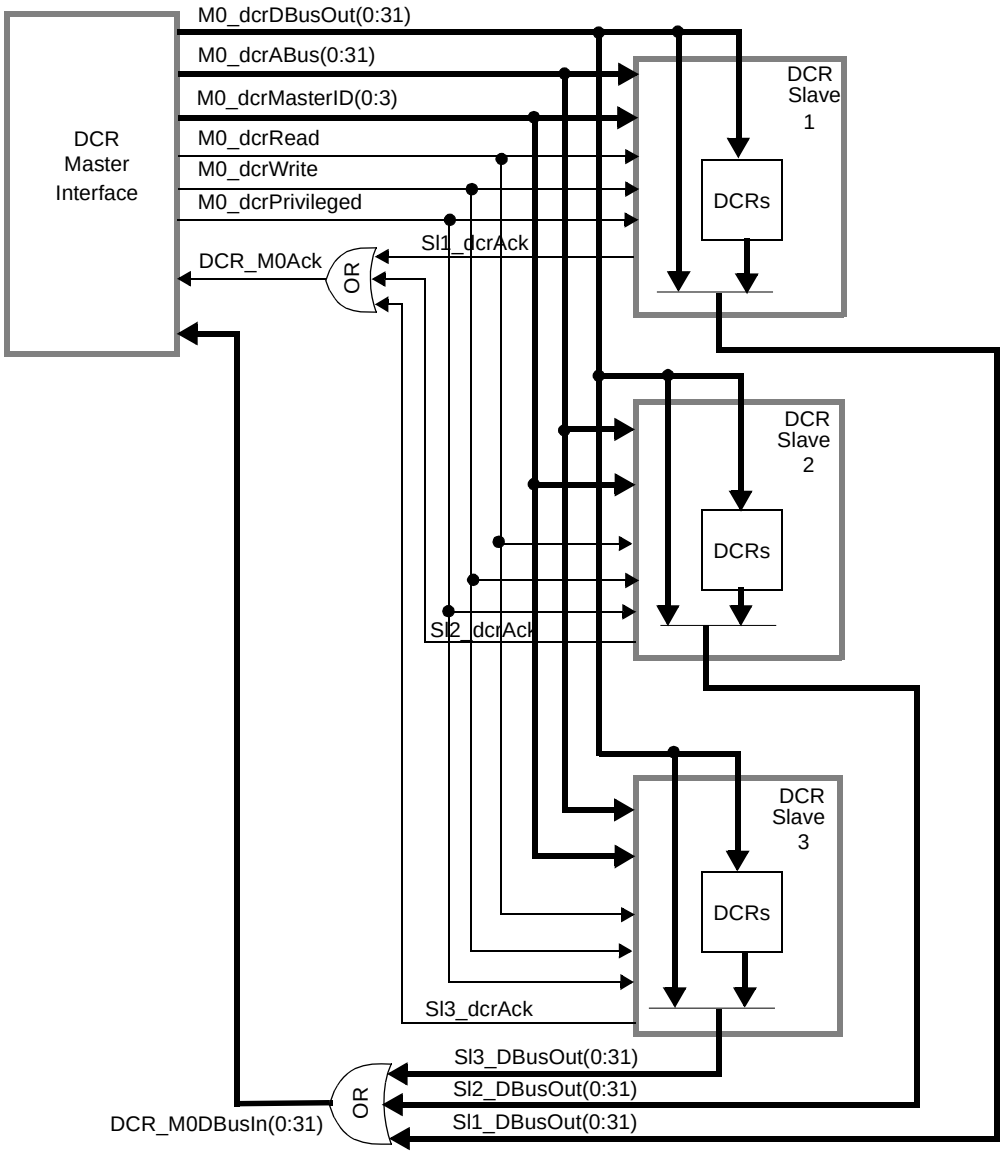
Figure 4-1. Daisy-Chain DCR Bus Topology



4.2 Distributed-OR Topology

Figure 4-2 illustrates another DCR bus configuration referred to as the “Distributed-OR” topology. In this bus scheme the DCR data path daisy chain is split into individual data paths which are then ORed in one or more stages of OR logic based on the geography and wiring constraints of the physical floorplan. The DCR master may only assert the M0_dcrDBusOut(0:31) signals during a write operation and these signals must be deasserted at all other times. This allows the data bus structure to merged with simple OR logic. Each DCR slave receives the DCR data bus output signals directly from the DCR master and the data outputs of all DCR slaves are logically ORed together to achieve the final output data bus, DCR_M0DBusIn(0:31). This DCR bus implementation is better suited for meeting high frequency timing constraints by eliminating the cumulative data path delays associated with the daisy-chained data path topology..

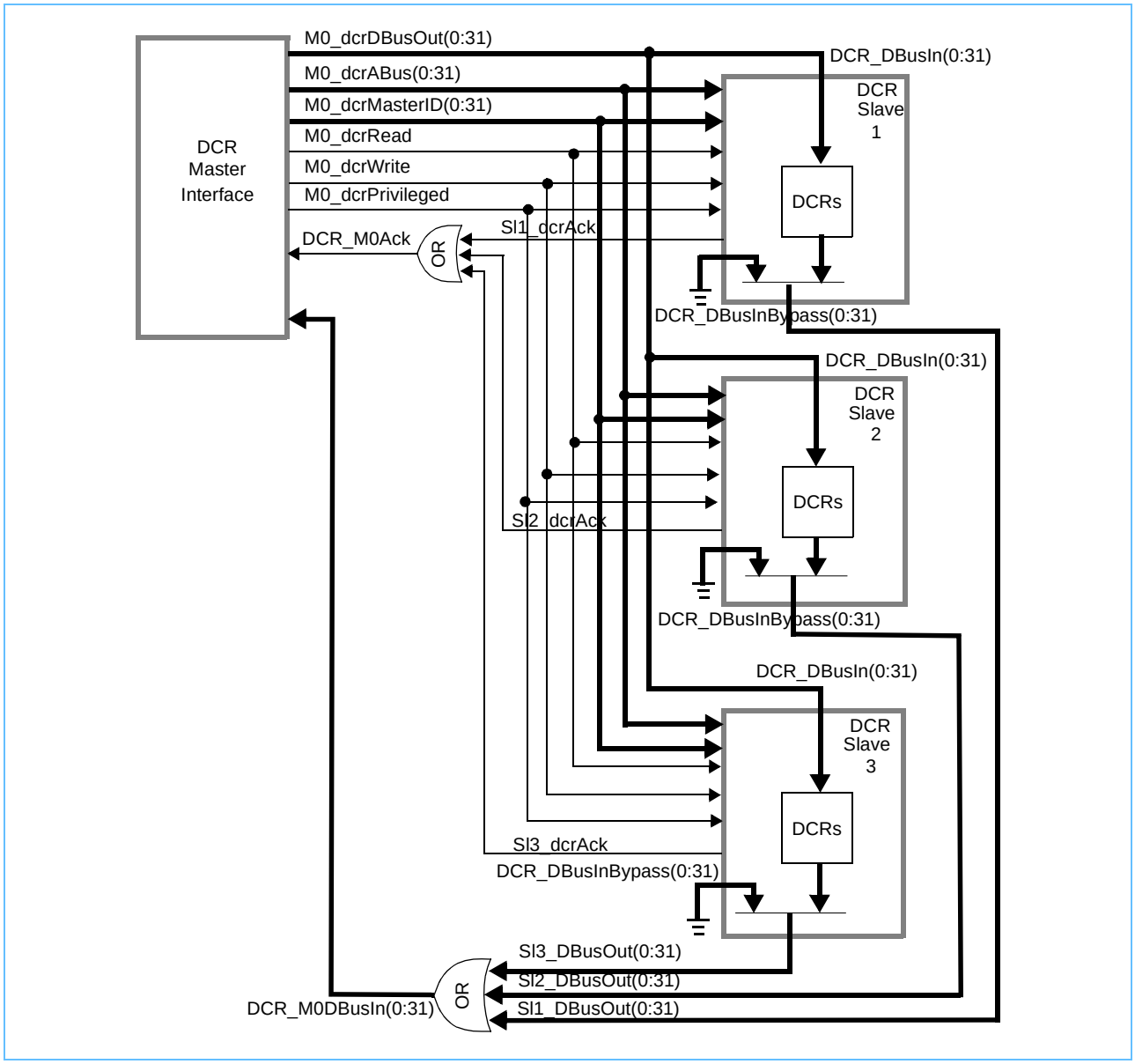
Figure 4-2. Distributed-OR DCR Bus Topology



4.3 Distributed-OR Topology with Bypass Inputs Tied-Off

In most distributed-OR DCR bus implementations, the Mn_dcrDBusOut(0:31) input which feeds the bypass mux input is typically not used. In these implementations, the following chip-level connectivity could be used to improve chip level timing, ease timing analysis, reduce power consumption, and perhaps remove redundant logic. In this arrangement the value of M0_dcrDbusOut(0:31) signals are not relevant when a DCR slave chooses its bypass path and therefore the DCR_DBusInBypass(0:31) inputs must be tied to zero to allow for Sln_DBusOut(0:31) ORing. Only the Sln_dcrDbusOut(0:31) data output of the addressed DCR slave will propagate through the OR logic.

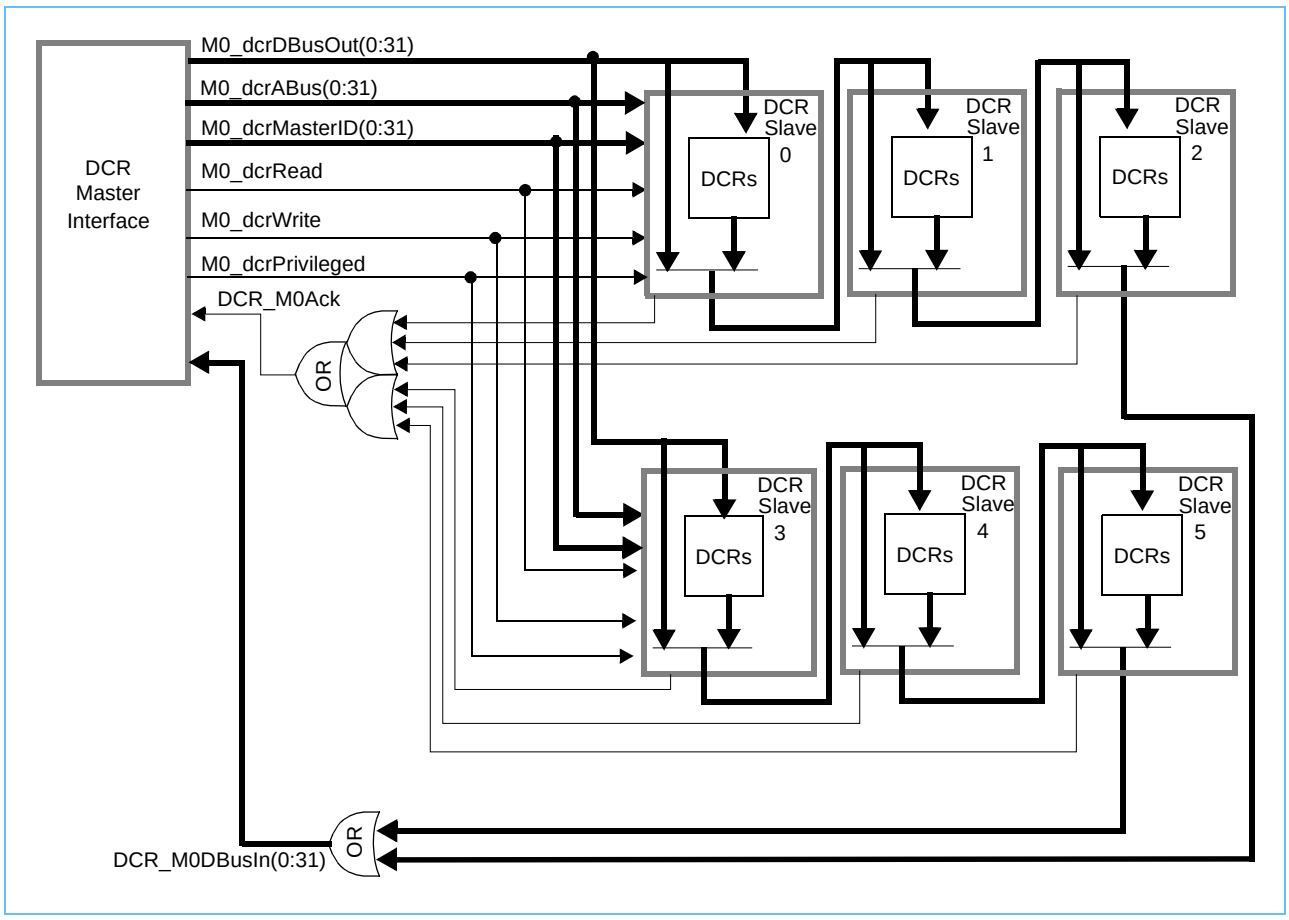
Figure 4-3. Distributed-OR DCR Bus Topology with Tied-off Bypass Inputs



4.4 Mixed Daisy-Chain / Distributed-OR Topology

Figure 4-4 illustrates a bus topology where DCR slaves are split across multiple DCR data path daisy-chains. Each DCR chain contains multiple slaves. All DCR slaves receive address and control signals from the DCR master. The output data buses from each chain are ORed together to create the final output data bus, DCR_M0DBusIn(0:31). Similarly, all slave acknowledge signals are ORed together to create the DCR_M0Ack signal. This structure could be used in balancing chip-level timing paths associated with slower DCR slaves.

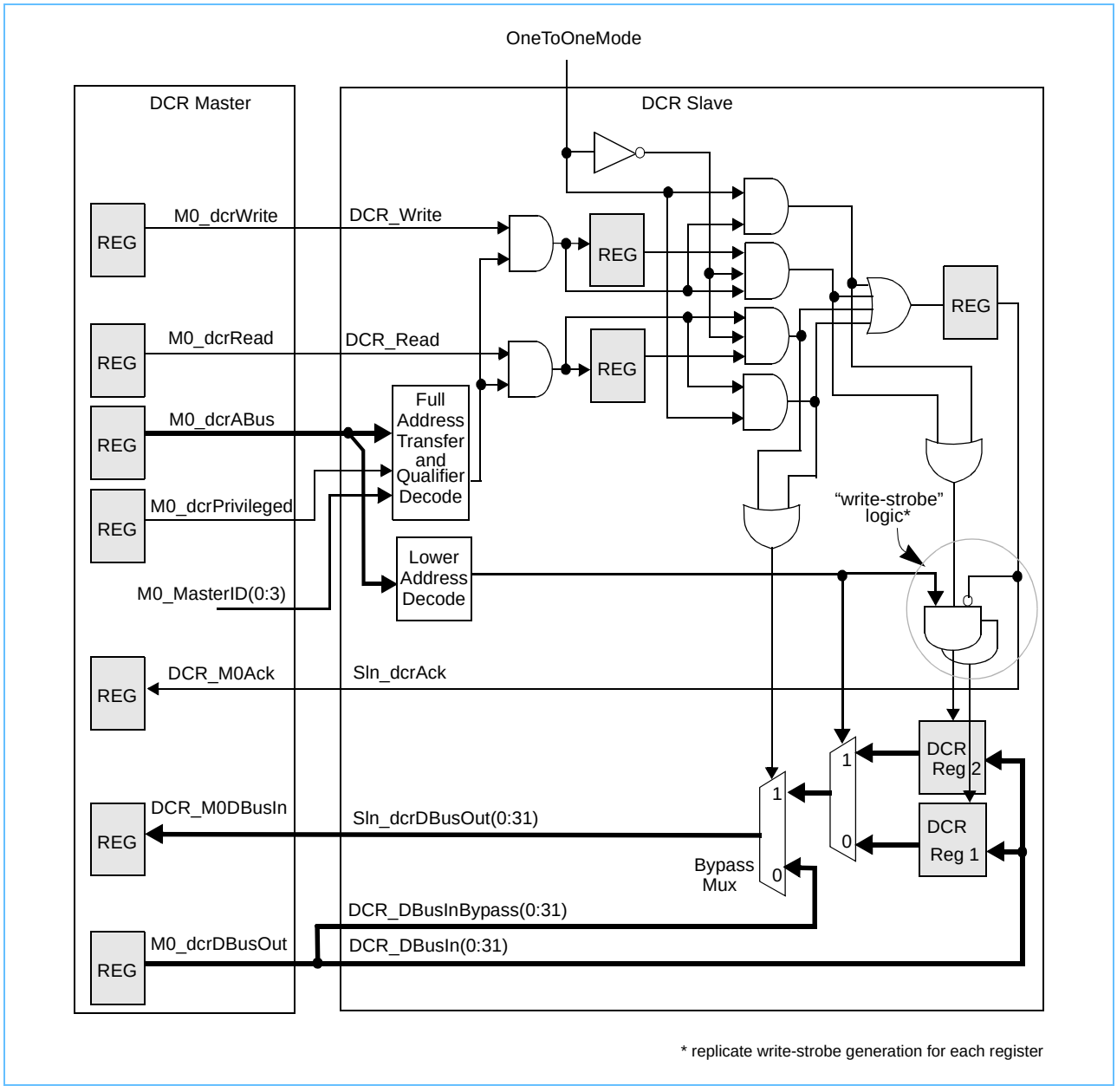
Figure 4-4. Mixed Daisy-Chain/Distributed-OR us Topology



5. DCR Slave Implementation

Figure 5-1 illustrates an implementation of the DCR bus slave interface. For reasons of decreased access latency, re-use amongst different clock domains, and dynamic slave frequency stepping for power conservation it is strongly recommended that all DCR slaves adhere to the implementation specified here.

Figure 5-1. DCR Slave Implementation



5.1 Bypass Muxing

The bypass mux must be implemented to allow for flexibility in chip-level connectivity with legacy DCR ring topologies. This bypass function should be implemented as a 2 to 1 mux or equivalent logic. It is also recommended that the DCR slave implement two independent data bus inputs labeled DCR_DBusInBypass(0:31) and DCR_DBusIn(0:31). The DCR_DBusInBypass(0:31) data input is used to allow data to flow through the DCR slave when not selected and the DCR_DBusIn(0:31) data input provides the path for inbound data to be written into the slave's registers. This allows for improved chip-level timing of implementations where the data bus input feeding the bypass mux may not be not required, and therefore could be tied off. An example of this is shown in *Figure 6-7*. The multi-level multiplexing between the bypass path and the selection of specific slave DCR registers produces minimum flow-through delay for daisy-chain ring topologies.

Note: For the case of chip-specific DCR slaves the chip integrator may choose to not implement data bus bypassing for their unique slave device. In this case, that slave must drive the Sln_dcrDBusOut(0:31) to 32'h00000000 at all times that it is not being addressed (full address + transfer qualifier decodes) with an active DCR_Read asserted. This also means that this chip-specific slave cannot be placed in any form of Daisy-Chain topology and thus will require the Distributed-OR style connection.
All other types of cores that are intended to be re-usable should implement the bypass muxing to allow maximum flexibility.

5.2 One To One Clock Mode

The one to one clock mode enable input to the slave, OneToOneMode, is strapped by the chip integrator to indicate to the slave whether or not it is being clocked by the DCR system clock signal, DCR_sysClk. When this signal is strapped asserted the slave is clocked by the DCR system clock signal and need not synchronize the decoded assertion of the DCR_Read and DCR_Write signals. Instead it may simply use the decoded assertion directly to generate the slave Sln_dcrAck signal assertion. When this signal is strapped deasserted the slave is not clocked by the DCR system clock signal and must synchronize the decoded assertion of the DCR_Read and DCR_Write signals. When one to one clock mode is enabled minimum DCR read and write cycle latency is achieved by eliminating an extra clock of unnecessary synchronization latency.

5.3 Register Write Strobe Generation

The "write strobe" logic causes data to be written into the slave's selected register coincident with the assertion of the slave Sln_dcrAck signal. This guarantees that for slaves operating at clock frequencies lower than the DCR master they latch the contents of the data bus into the register before deassertion of the address, data, and DCR write command.

6. DCR Bus Operations

The DCR bus interface consists of an address and transfer qualifier bus, an input and output data bus, DCR read and DCR write signals, timeout wait, and acknowledge signal.

Transfers are initiated by the DCR master asserting the DCR read or DCR write command signals with the valid transfer qualifiers. Slaves decode the command, address, privilege level, and master ID to determine whether to claim the transfer or not. A command/acknowledge interlock mechanism allows the DCR masters, which are always clocked at the DCR system clock frequency, to be connected to DCR slaves that are clocked at a different frequency. The rising edge of the slower clock, master or slave, must always correspond to the faster clock's rising edge**.**

Following the assertion of a DCR command if no slave claims the transfer by either asserting the timeout wait signal or the acknowledge signal for a predefined number of system clocks the bus is said to have "timed out". Slaves which may cause a bus timeout condition may assert the timeout wait signal to extend the bus cycle. The assertion of the acknowledge signal indicates that a write operation is complete or that the read data has been placed on the bus. When a master detects a bus timeout it simply terminates it bus by deasserting the read or write command.

Data moves over an interface that forms a daisy-chained ring or a distributed-OR topology. In the daisy-chain configuration each slave unit either passes along the unmodified data bus input or places its data onto to its data bus output In the distributed-OR configuration each slave receives the master's data output and provides its data bus output directly to the system OR logic.

Once the master has sampled the assertion of the acknowledge signal it deasserts its command and waits to sample the acknowledge signal deasserted prior to performing another transfer.

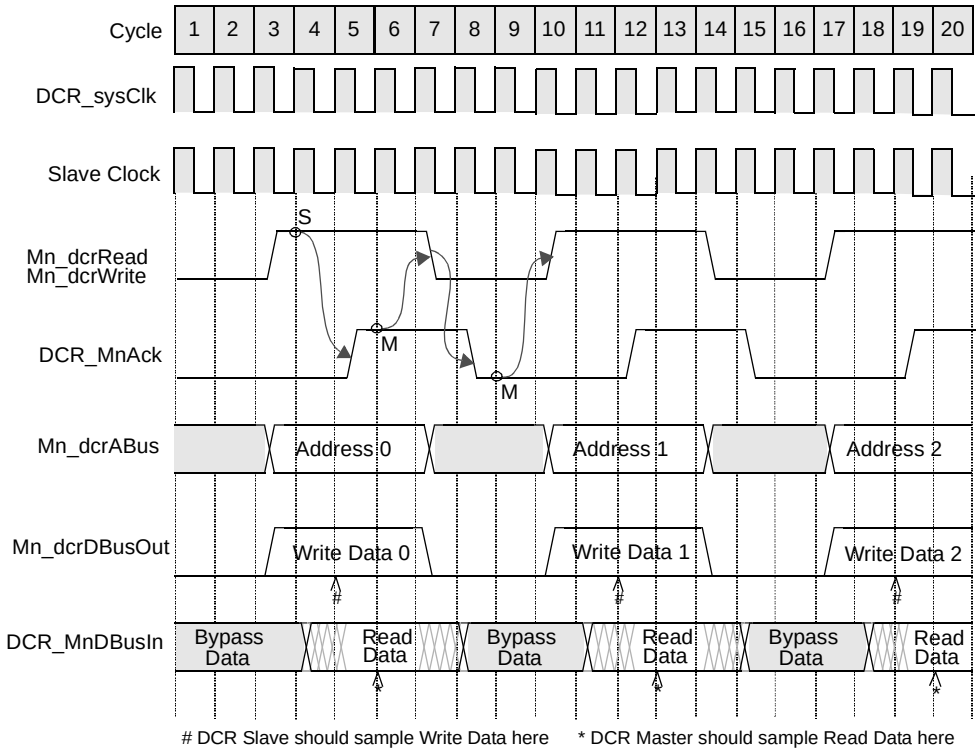
The following waveforms illustrate the operation of the bus.

These waveforms are intended to show the DCR bus running with maximum performance based on the example slave implementation shown in *Figure 5-1*, but it is allowed that either the DCR master, DCR slave, or DCR arbiter can insert extra cycles of latency to the transaction. Slaves that intend to use the DCR bus for run-time functionality (eg. Interrupt controllers, DMA controllers, etc.) should strive for the maximum performance implementation and avoid introducing extra cycles of latency.

6.1 DCR Master Same Frequency as DCR Slave (OneToOneMode disabled)

Figure 6-1 illustrates the DCR bus operation when the DCR system clock is running at the same frequency as the DCR slave and the slave one to one mode is not enabled or not supported. The DCR master places an address onto Mn_dcrABus and asserts Mn_dcrRead or Mn_dcrWrite. A DCR slave decodes the DCR address and latches this indication at the rising edge of slave clock 4. The slave acknowledges that it is responding to the command at the rising edge of clock 5. After sampling the assertion of DCR_MnAck the DCR master deasserts the Mn_dcrRead or Mn_dcrWrite signal. The DCR slave recognizes the deassertion of the Mn_dcrRead or Mn_dcrWrite signal and the DCR_MnAck signal is deasserted.

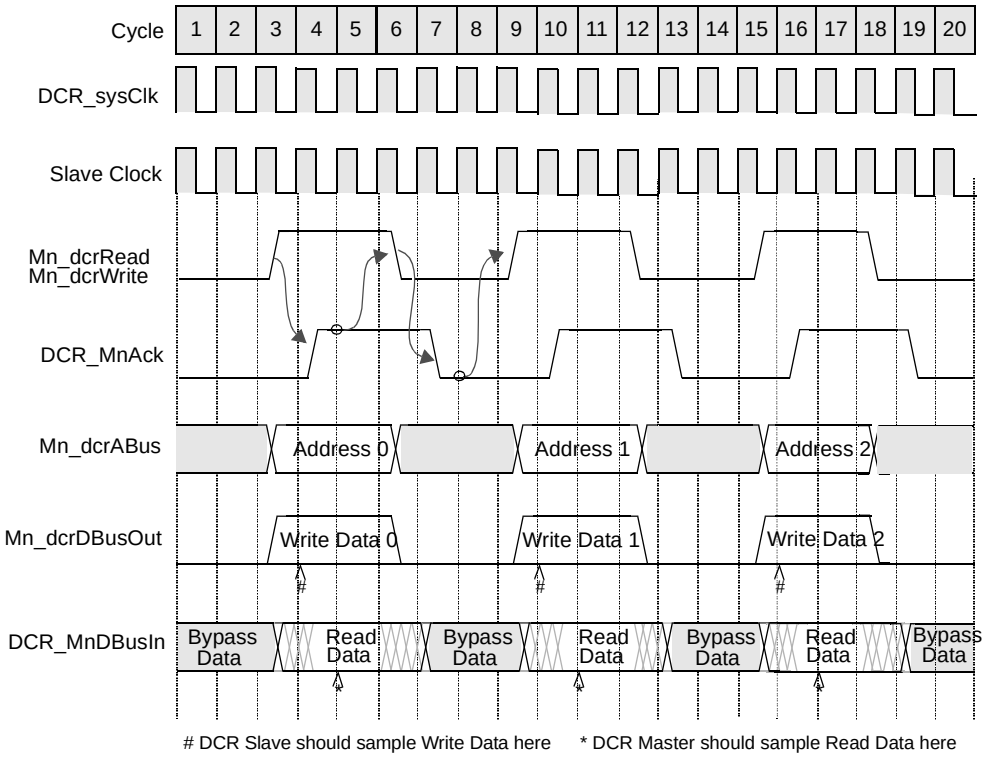
Figure 6-1. DCR Master Clock same as DCR Slaves (One to One Mode Disabled)



6.2 DCR Master Same Frequency as DCR Slave (OneToOneMode enabled)

Figure 6-2 illustrates the DCR bus operation when the DCR system clock is running at the same frequency as the DCR slave and the slave one to one mode is enabled. The DCR master places an address onto M0_dcrABus and asserts Mn_dcrRead or Mn_dcrWrite. A DCR slave decodes the DCR address and because it is in one to one mode acknowledges that it is responding to the command at the rising edge of clock 4. This eliminates one clock of latency on the assertion of DCR_MnAck. After sampling the assertion of DCR_M0Ack the DCR master deasserts the Mn_dcrRead or Mn_dcrWrite signal. The DCR slave recognizes the deassertion of the Mn_dcrRead or Mn_dcrWrite signal and the DCR_M0Ack signal is deasserted.

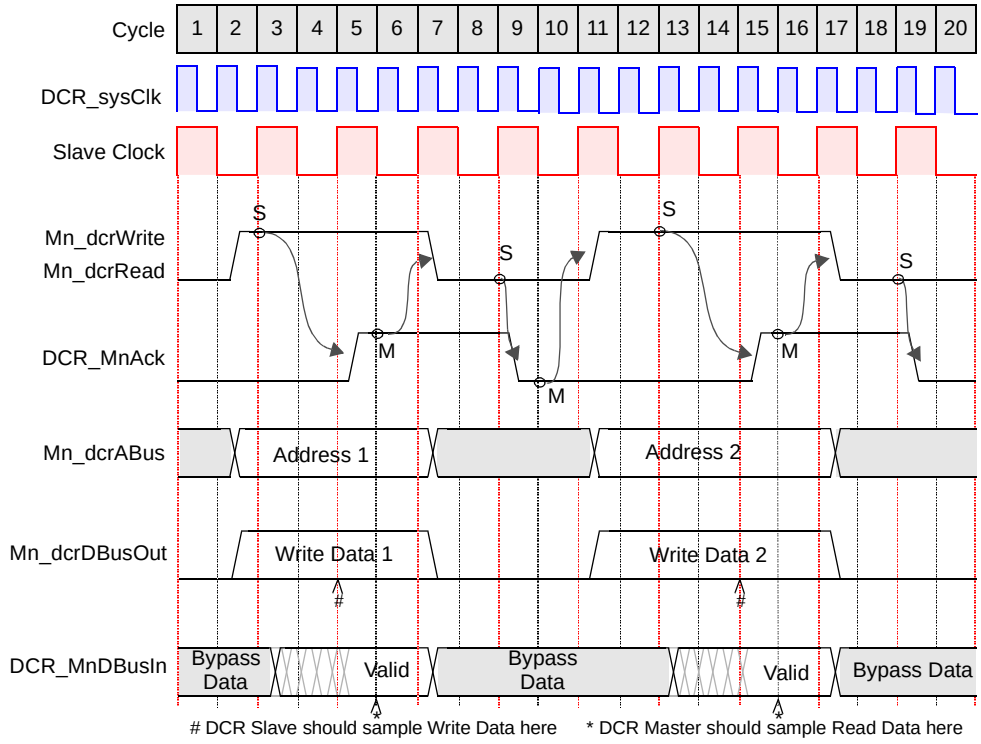
Figure 6-2. DCR Slave One to One Mode Enabled



6.3 Master Clocked at Twice the Frequency of the Slave

Figure 6-3 illustrates DCR bus operation when the DCR system clock is operating at twice the frequency of the DCR slave. (One-to-one mode disabled)

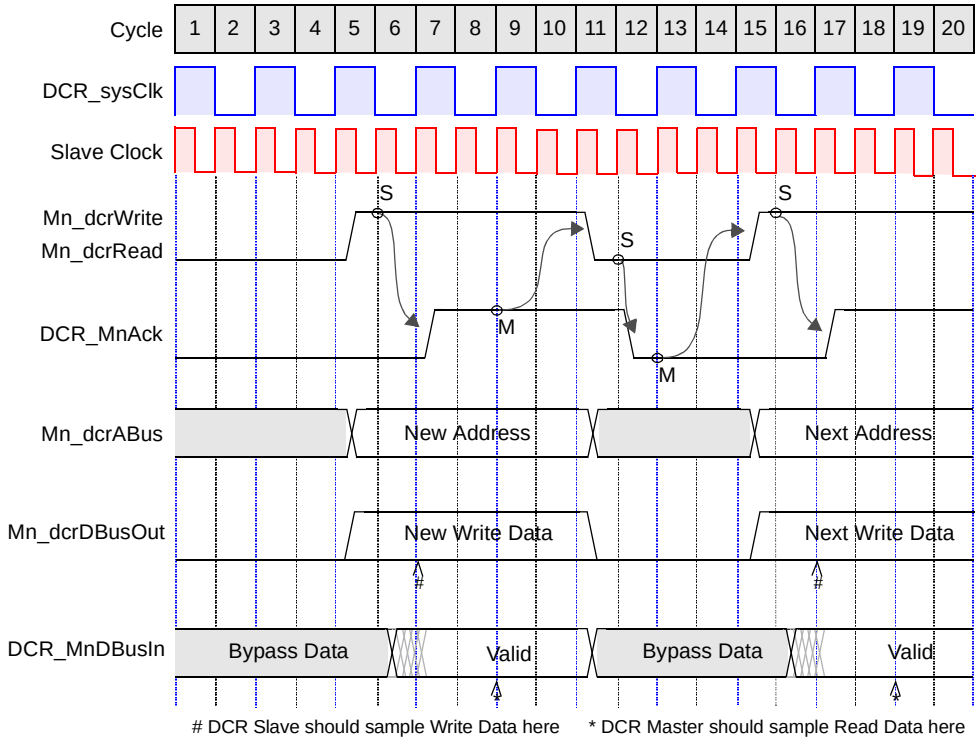
Figure 6-3. DCR Master Two Times Faster than DCR Slave



6.4 DCR Master at half the frequency of the DCR Slave

Figure 6-4 illustrates the DCR bus operation when the master DCR system clock is operating at half the frequency of the slave's clock.(One-to-one mode disabled)

Figure 6-4. DCR Master Slower than DCR Slave

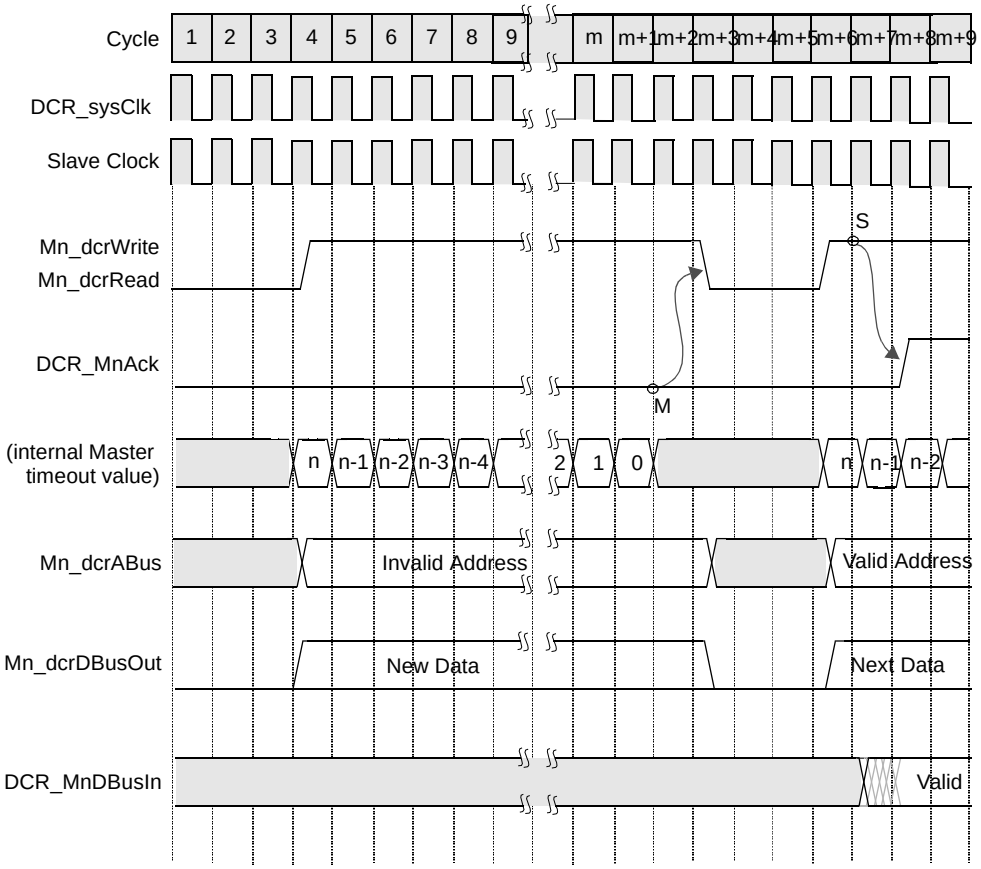


6.5 DCR Bus Time-out

Figure 6-5 illustrates a DCR bus timeout operation. The master asserts a read or write command and waits to sample the assertion of the DCR_MnAck signal. The master waits for a predefined number of clocks and terminates the command if the DCR_MnAck signal is not asserted within the time allotted by the master. The master may then proceed to a subsequent DCR operation.

The definition of how long a master will wait before deasserting the Write/Read command and asserting that the transfer is ‘timed out’ is left up to the individual master implementation. Each master may have a different timeout count value (designated as ‘m’ in the cycle count of *Figure 6-5* below) which should be clearly documented in the master’s documentation so that system integrators can determine if it will be necessary to utilize the DCR_MnTimeoutWait signal for that master given the specific response characteristics of their DCR slave devices.

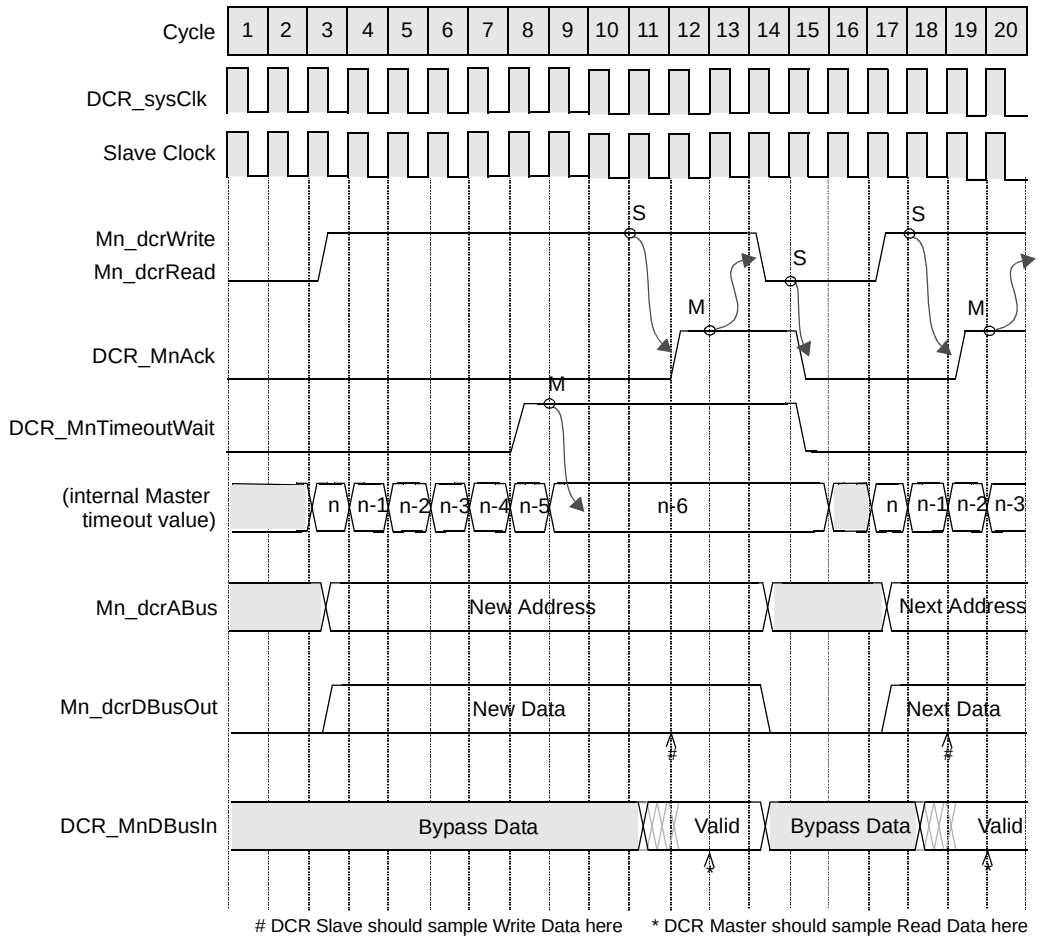
Figure 6-5. DCR Bus Timeout



6.6 DCR Slave utilizing optional “Timeout-Wait” feature

Figure 6-6 describes the operation of the timeout wait feature. The master asserts a read or write command and waits to sample the assertion of the DCR_MnAck signal. The slave subsequently asserts the Sln_dcrTimeoutWait signal which causes the master to sample the DCR_MnTimeoutWait signal asserted. This inhibits the internal master timeout counter from counting any further. The slave eventually asserts the DCR acknowledge signal..

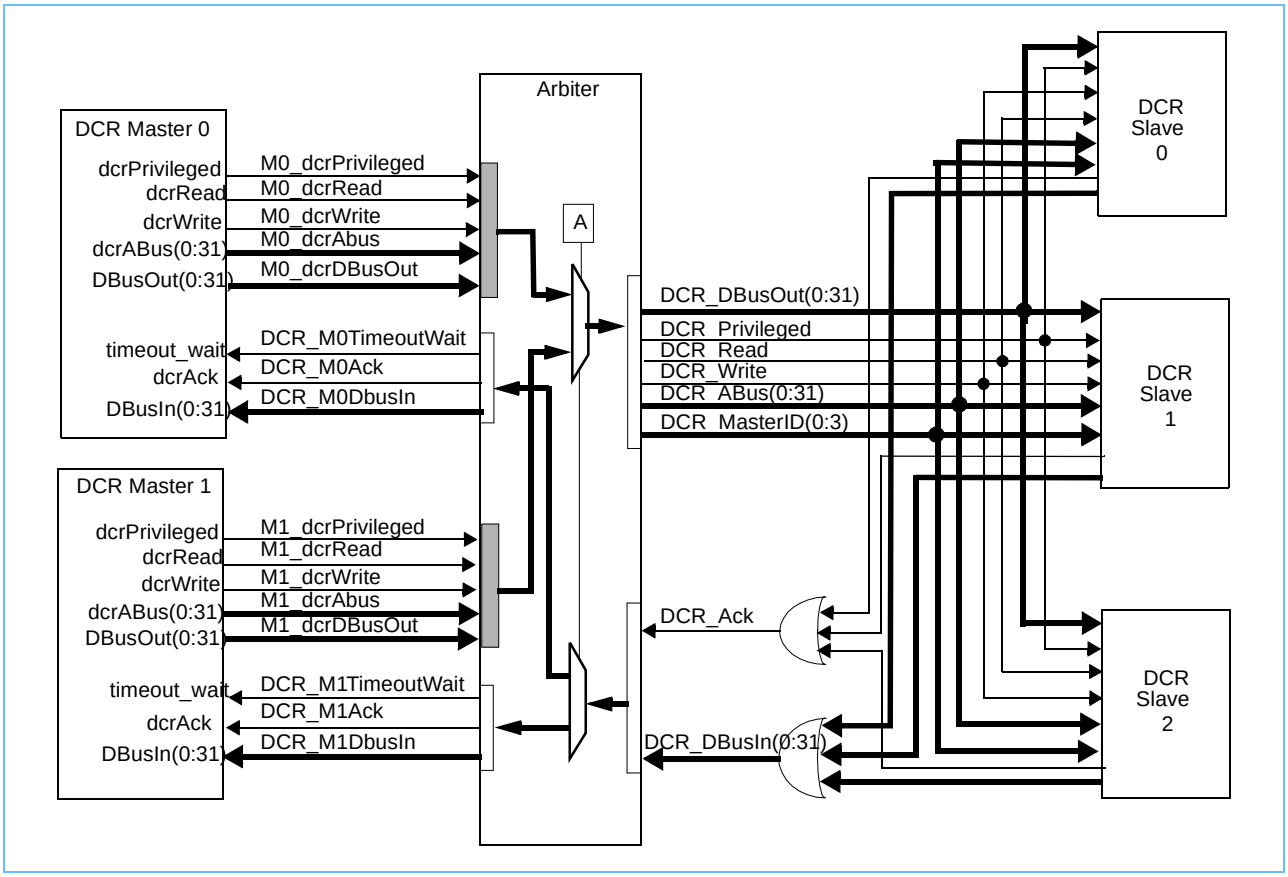
Figure 6-6. DCR Bus Transaction with Optional Timeout Wait Feature



6.7 DCR Arbiter Implementation Example

Figure 6-7 illustrates a structure where two DCR masters share access to the same DCR bus. In this type of implementation an arbiter is used to control DCR bus to negotiate access to the single shared slave bus by utilizing the DCR_MnTimeoutWait signal to temporarily inhibit a given master's DCR timeout functionality. This allows other DCR masters the opportunity to complete DCR cycles without the possibility of timing out an unselected master.

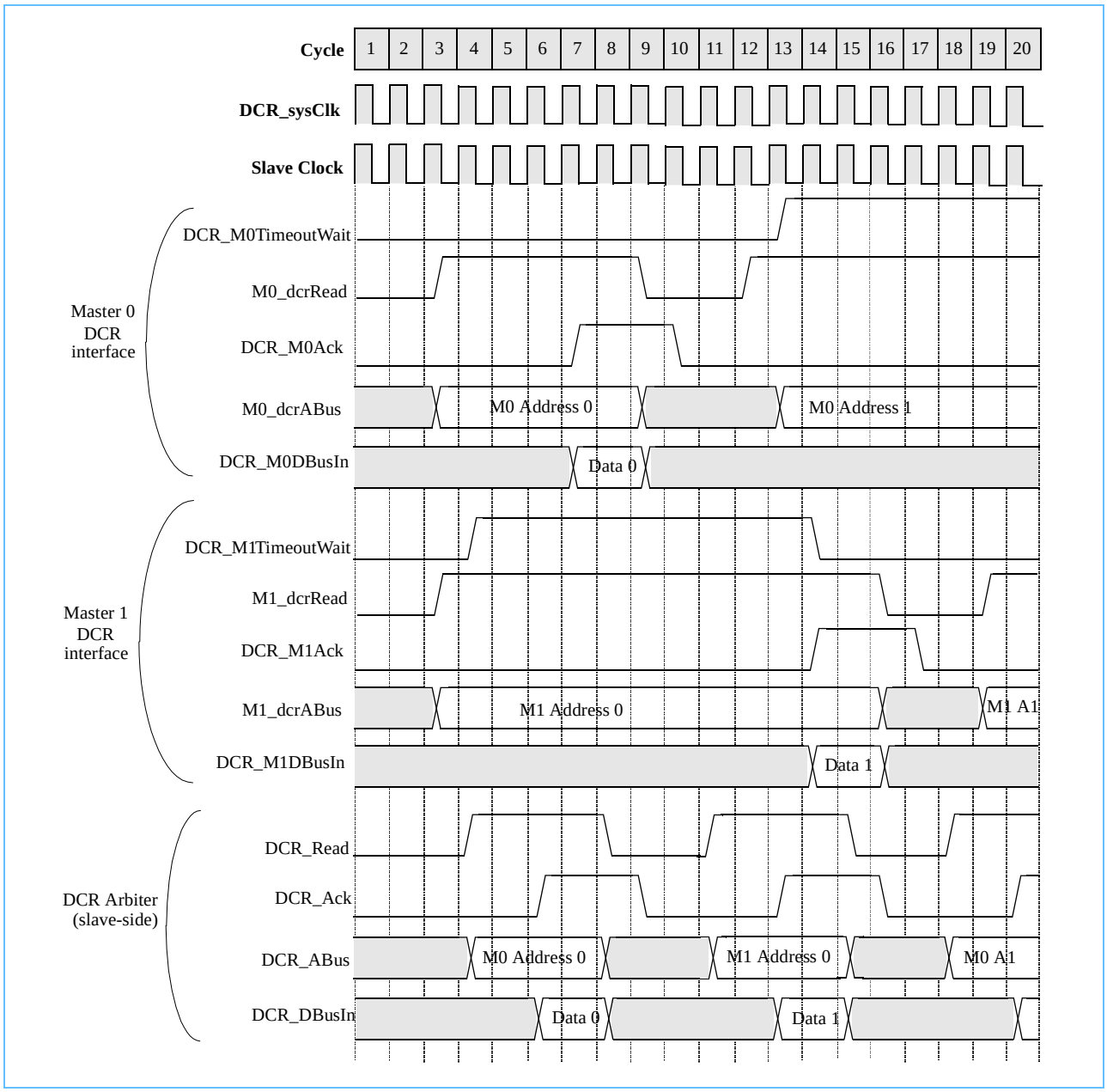
Figure 6-7. DCR Arbiter Example



6.7.1 DCR Arbiter Example

Figure 6-8 illustrates the operation of a two master DCR bus arbiter scheme as described previously in *DCR Arbiter Implementation Example* on page 40. In this arrangement, each master's DCR_MnTimeoutWait signal is used independently to temporarily inhibit its master bus timeout functionality until such time its DCR transaction is completed by the slave. This arbiter illustrates a registered implementation between masters and slaves. Both masters and the arbiter are clocked with the DCR clock. The slave in this case has one-to-one mode disabled.

Figure 6-8. Two Master Arbitration Example



7. DCR Bus Timing Guidelines

The DCR signal timing guidelines are described in this section.

7.1 Master Timing Definitions

Begin Signal is valid within 8% of the clock cycle from the rise of the DCR_sysClk signal.

Early Signal is valid within 18% of the clock cycle from the rise of the DCR_sysClk signal.

Middle Signal is valid within 43% of the clock cycle from the rise of the DCR_sysClk signal.

Late Signal is valid within 68% of the clock cycle from the rise of the DCR_sysClk signal.

End Signal is valid within 78% of the clock cycle from the rise of the DCR_sysClk signal.

These definitions assume that there is zero clock delay. For outputs, these delays represent the total logic delay from the rising edge of the DCR system clock at the input to a register to the DCR Master output. For inputs, these delays represent the arrival time of the input relative to a zero delayed DCR system clock.

DCR Masters are required to provide all signal handshaking synchronous to the DCR_sysClk signal. Slave devices may be clocked completely synchronous to this signal or may be rising edge synchronous with the DCR_sysClk signal and operate at a higher or lower frequency. *Table 7-1* describes the DCR Master timing guidelines

Table 7-1. DCR Master Timing Guidelines

Signal Name	Driven By	Output Valid	Received By
Mn_dcrRead	DCR Master	Begin	DCR Bus
Mn_dcrWrite	DCR Master	Begin	DCR Bus
Mn_dcrPrivileged	DCR Master	Begin	DCR Bus
Mn_dcrMasterID(0:3)	DCR Master	Begin	DCR Bus
Mn_dcrABus(0:31)	DCR Master	Begin	DCR Bus
Mn_dcrDBusOut(0:31)	DCR Master	Begin	DCR Bus
DCR_MnTimeoutWait	DCR Bus	End	DCR Master
DCR_MnAck	DCR Bus	End	DCR Master
DCR_MnDBusIn(0:31)	DCR Bus	End	DCR Master

7.2 Slave Timing Definitions

Begin Signal is valid within 8% of the clock cycle from the rise of the slave clock signal.

Early Signal is valid within 18% of the clock cycle from the rise of the slave clock signal.

Middle Signal is valid within 43% of the clock cycle from the rise of the slave clock signal.

Late Signal is valid within 68% of the clock cycle from the rise of the slave clock signal.

End Signal is valid within 78% of the clock cycle from the rise of the slave clock signal.

These definitions assume that there is zero clock delay. For outputs, these delays represent the total logic delay from the rising edge of the slave clock at the input to a register to the output of the core. For inputs, these delays represent the arrival time of the input relative to a zero delayed slave clock.

Slave devices may be clocked completely synchronous to the DCR_sysClk or may be rising edge synchronous with the DCR_sysClk signal and operate at a higher or lower frequency. For reasons of latency, re-use, and power conservation techniques it is strongly recommended that slaves use the interface logic outlined in *DCR Slave Implementation* on page 31.

Table 7-2 describes the DCR Slave timing guidelines and all signals are defined with respect to the *slave's clock*. If the slave interface logic is clocked with DCR_sysClk then the master and slave timing guidelines are complimentary

Table 7-2. DCR Slave Timing Guidelines

Signal Name	Driven By	Output Valid	Received By
DCR_Read	DCR Bus	Early	DCR Slave
DCR_Write	DCR Bus	Early	DCR Slave
DCR_Privileged	DCR Bus	Early	DCR Slave
DCR_MasterID(0:3)	DCR Bus	Early	DCR Slave
DCR_ABus(0:31)	DCR Bus	Early	DCR Slave
DCR _DBusIn(0:31)	DCR Bus	Early	DCR Slave
DCR_DBusInBypass(0:31)	DCR Bus	Early	DCR Slave
OneToOneMode	Strapping	Always	DCR Slave
Sln_dcrTimeoutWait	DCR Slave	Begin	DCR Bus
Sln_dcrAck	DCR Slave	Begin	DCR Bus
Sln_dcrDBusOut(0:31)	DCR Slave	Middle	DCR Bus

The Sln_dcrDBusOut(0:31) signals are specified as "Middle" because the DCR_DBusIn(0:31) signals don't arrive until early in the cycle and the slave needs time to propagate the data through the bypass mux. The bypass mux logic should be implemented to keep the path delay to a minimum when the slave is not selected. This delay is critical to achieving chip timing in a ring topology.

The delay from a DCR slave's register to the Sln_dcrDBusOut(0:31) signals should also be kept to a minimum. Slaves should target an "Early" timing guideline in this case.

There will be always be additional delay when more than one slave is attached to the bus between the output of a DCR slave and the input to the DCR master in both the ring and distributed-OR topologies. This additional delay will vary depending upon where the slave is placed in the DCR ring or the size and implementation of the distributed-OR logic.

Note: If the DCR ring becomes too long to make chip timing the ring may be broken up into smaller rings or into a distributed-OR structure with “rings of one” as described in the *Mixed Daisy-Chain / Distributed-OR Topology* on page 30. Additional logic may be required to control and mux the signals back to the DCR master.

8. Legacy DCR Slaves

The legacy mode of the DCR bus architecture consists of a 10-bit only address bus, a “move to DCR” mtdcr signal for writes, a “move from DCR” mfdcr signal for reads, a 32-bit data path, and a DCR acknowledge signal. The DCR_privileged signal, DCR_MasterID(0:3) signals, Sln_dcrTimeoutWait signal, and the upper 22-bits of the address bus are not implemented. These slave are compatible with the current definition of the DCR bus architecture, however because of the additional transfer qualifiers and address bus bits the decoding of the slave requires external logic to correctly place the legacy device in the address map.

As always the systems-on-a-chip architect integrates the DCR slave cores and defines the address mapping of them. To retain software compatibility it may be advantageous to simply place legacy components in the 1K of addressing and decode the higher order 22 address bit as a “zero”. Also since legacy slaves don’t implement the privileged transfer qualifier, and when they were developed all transfers were privileged, it is suggested that they are mapped as a privileged slave via read/write command signal decoding. If access to the slave is to allowed only on a per-master basis then the DCR_MasterID(0:3) signals may decoded in the address or command decoding logic. An example illustrating legacy slave attachment follows.

8.1 Address Decoding for Mixed 32-bit/10-bit DCR Bus Implementations

Figure 8-1 illustrates an example of address decoding that can be used in a multi-slave implementation containing 32-bit addressable DCR slaves and legacy 10-bit addressable DCR slaves. In this example, the 10-bit DCR slave is mapped to DCR addresses that contain zeros in the twenty-two most-significant bits of the address, M0_dcrAbus(0:21). The 32-bit addressable DCR slave must be configured to not share the same address region that is supported by the 10-bit DCR slave to prevent “address aliasing” from occurring.

Figure 8-1. Implementation with 10-bit and 32-bit Addressable DCR Slaves

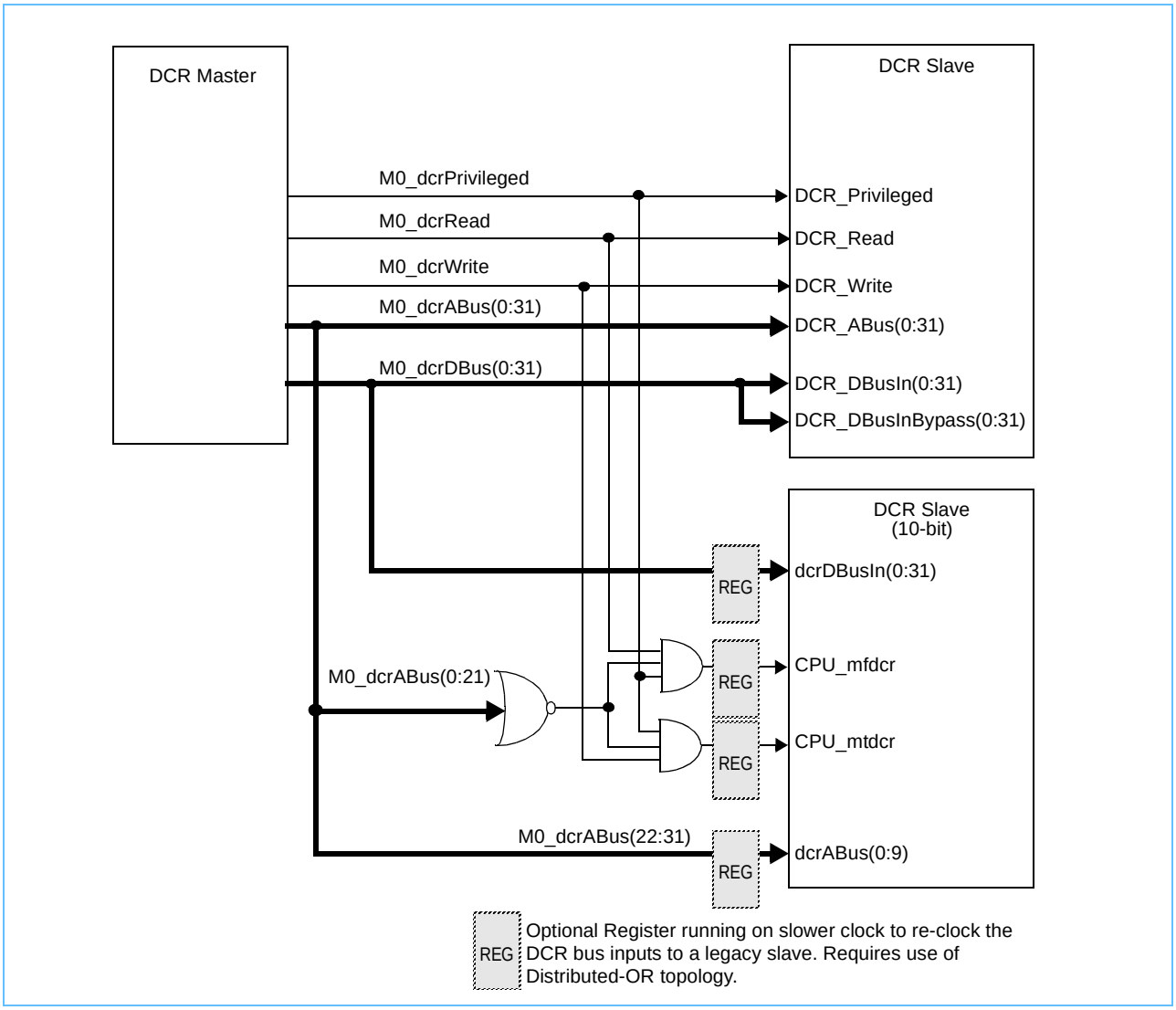


Figure 8-1 also illustrates an optional technique to handle legacy slave devices that may not be able to operate at the higher clock speeds of the newer DCR slave or DCR master devices. In this case it is acceptable to re-latch the DCR bus inputs to the legacy slave at the slower slave clock frequency, effectively making the legacy slave run with extra cycle(s) of latency due to the additional registers. If this is utilized, it implies that this portion of the design is configured in a Distributed-OR topology as described in *Section 4.2 Distributed-OR Topology* and *Section 4.4 Mixed Daisy-Chain / Distributed-OR Topology*.

Index

Revision Log

Revision Date	Contents of Modification
01/27/06	GA version of DCR bus architecture specifications
06/03/05	Preliminary version of DCR bus architecture specifications