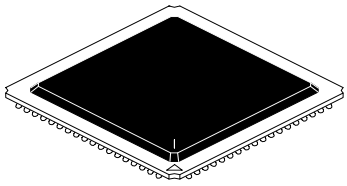
MPC2605/D
Rev. 12, 11/2002

Integrated Secondary Cache for Microprocessors That Implement PowerPC Architecture

MOTOROLA intelligence everywhere™ | digital dna™

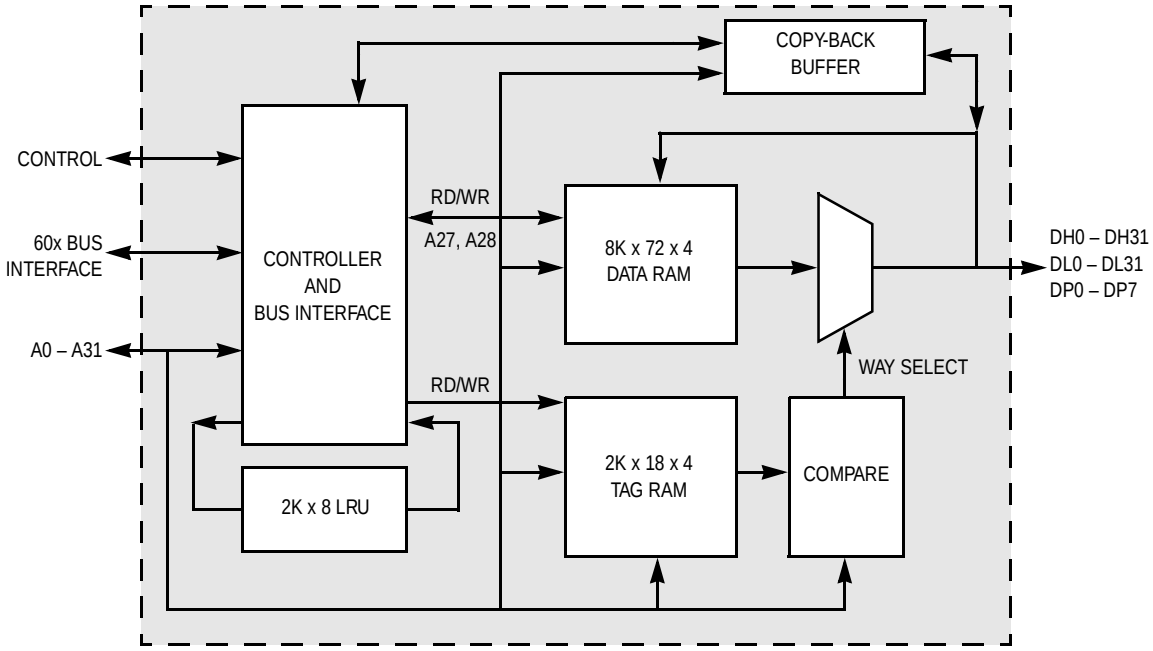
The MPC2605 is a single chip, 256KB integrated look-aside cache with copy-back capability designed for applications using a 60x bus. Using 0.38 µm technology along with standard cell logic technology, the MPC2605 integrates data, tag, host interface, and least recently used (LRU) memory with a cache controller to provide a 256KB, 512KB, or 1MB Level 2 cache with one, two, or four chips on a 64-bit 60x bus.



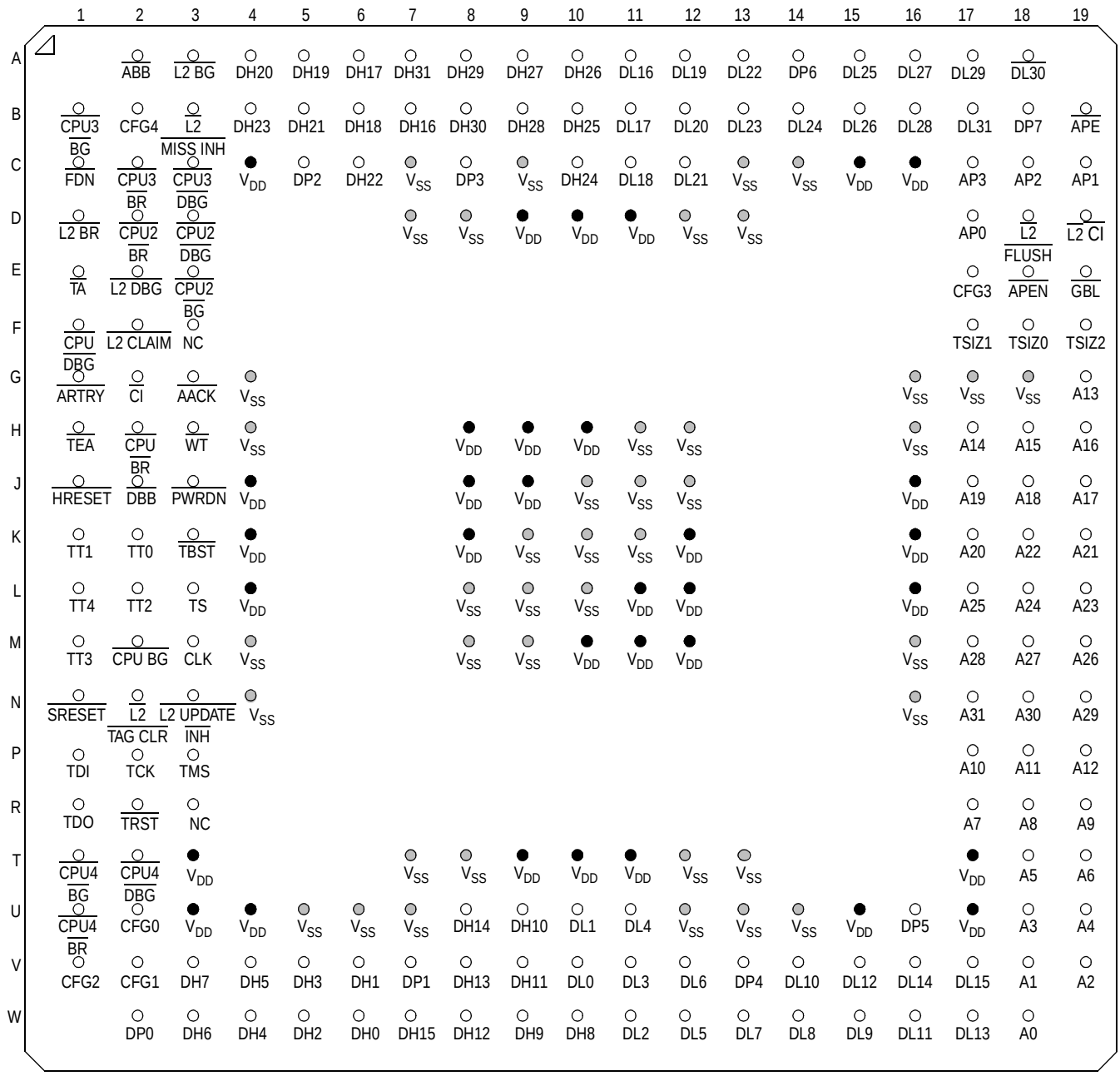
ZP PACKAGE
PBGA
CASE 1138-01

- Single Chip L2 Cache
- 66 or 83 MHz Zero Wait State Performance (2-1-1-1 Burst)
- Four-Way Set Associative Cache Design
- 32K x 72 Data Memory Array
- 8K x 18 Tag Array
- Address Parity Support
- LRU Cache Control Logic
- Copy-Back or Write-Through Modes of Operation
- Copy-Back Buffer for Improved Performance
- Single 3.3 V Power Supply
- 5 V Tolerant I/O
- One-, Two-, or Four-Chip Cache Solution (256KB, 512KB, or 1MB)
- Single Clock Operation
- Compliant with IEEE Standard 1149.1 Test Access Port (JTAG)
- Supports up to Four Processors in a Shared Cache Configuration
- High Board Density 25 mm 241 PBGA Package

BLOCK DIAGRAM



PIN ASSIGNMENT



TOP VIEW

PIN DESCRIPTIONS

Pin Locations	Pin Name	Type	Description
19G, 17H – 19H, 17J – 19J, 17K – 19K, 17L – 19L, 17M – 19M, 17N – 19N, 17P – 19P, 17R – 19R, 18T, 19T, 18U, 19U, 18V, 19V, 18W *	A0 – A31	I/O	Address inputs from processor. Can also be outputs for processor snoop addresses. A0 is the MSB. A31 is the LSB.
3G	$\overline{\text{AACK}}$	I/O	Address acknowledge input/output.
2A	$\overline{\text{ABB}}$	I/O	Used as an input to qualify bus grants. Driven as an output during address tenure initiated by the MPC2605.
17C – 19C, 17D*	AP0 – AP3	I/O	Address parity.
19B	$\overline{\text{APE}}$	O	Address parity error. When an address parity error is detected, $\overline{\text{APE}}$ will be driven low one clock cycle after the assertion of $\overline{\text{TS}}$ then High-Z following clock cycle.
18E	$\overline{\text{APEN}}$	I	Address parity enable. When tied low, enables address parity bits and the address parity error bit.
1G	$\overline{\text{ARTRY}}$	I/O	Address retry status I/O. Generated when a read or write snoop to a dirty processor cache line has occurred.
2U 2V 1V 17E 2B	CFG0 CFG1 CFG2 CFG3 CFG4	I	Configuration inputs. These must be tied to either V_{DD} or V_{SS}. CFG0 CFG1 CFG2 0 0 0 256KB 0 1 0 512KB; A26 = 0 0 1 1 512KB; A26 = 1 1 0 0 1MB; A25 – A26 = 00 1 0 1 1MB; A25 – A26 = 01 1 1 0 1MB; A25 – A26 = 10 1 1 1 1MB; A25 – A26 = 11 CFG3 Snoop Data Tenure Selector 1 Any snoop cycle is address only, even when TT3 = 1 0 TT3 will decide whether a snoop is a data- or address-only tenure CFG4 $\overline{\text{AACK}}$ Driver Enable 0 Disable $\overline{\text{AACK}}$ driver 1 Enable $\overline{\text{AACK}}$ driver
2G	CI	I/O	Cache inhibit I/O.
3M	CLK	I	Clock input. This must be the same as the processor clock input.
2M	$\overline{\text{CPU BG}}$	I	CPU bus grant input.
3E	$\overline{\text{CPU2 BG}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU BG}}$. Used in multiprocessor configuration as the second $\overline{\text{CPU BG}}$.
1B	CPU3 BG	I	MPC2605 logically ORs this signal with $\overline{\text{CPU BG}}$. Used in multiprocessor configuration as the third $\overline{\text{CPU BG}}$.
1T	CPU4 BG	I	MPC2605 logically ORs this signal with $\overline{\text{CPU BG}}$. Used in multiprocessor configuration as the fourth $\overline{\text{CPU BG}}$.
2H	CPU BR	I	CPU bus request input.

PIN DESCRIPTIONS (continued)

Pin Locations	Pin Name	Type	Description
2D	$\overline{\text{CPU2 BR}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU BR}}$. Used in multiprocessor configuration as the second $\overline{\text{CPU BR}}$.
2C	$\overline{\text{CPU3 BR}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU BR}}$. Used in multiprocessor configuration as the third $\overline{\text{CPU BR}}$.
1U	$\overline{\text{CPU4 BR}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU BR}}$. Used in multiprocessor configuration as the fourth $\overline{\text{CPU BR}}$.
1F	$\overline{\text{CPU DBG}}$	I	CPU data bus grant input from arbiter.
3D	$\overline{\text{CPU2 DBG}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU DBG}}$. Used in multiprocessor configuration as the second $\overline{\text{CPU DBG}}$.
3C	$\overline{\text{CPU3 DBG}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU DBG}}$. Used in multiprocessor configuration as the third $\overline{\text{CPU DBG}}$.
2T	$\overline{\text{CPU4 DBG}}$	I	MPC2605 logically ORs this signal with $\overline{\text{CPU DBG}}$. Used in multiprocessor configuration as the fourth $\overline{\text{CPU DBG}}$.
11A – 13A, 15A – 18A, 11B – 17B, 11C, 12C, 10U, 11U, 10V – 12V, 14V – 17V, 11W – 17W *	DL0 – DL31	I/O	Data bus low input and output. DL0 is the MSB. DL31 is the LSB.
4A – 10A, 4B – 10B, 6C, 10C, 8U, 9U, 3V – 6V, 8V, 9V, 3W – 10W*	DH0 – DH31	I/O	Data bus high input and output. DH0 is the MSB. DH31 is the LSB.
2J	$\overline{\text{DBB}}$	I/O	Data bus busy. Used as input when processor is master, driven as an output after a qualified $\overline{\text{L2 DBG}}$ when MPC2605 is the bus master. Note: To operate in Fast L2 mode, this pin must be tied high.
14A, 18B, 5C, 8C, 16U, 7V, 13V, 2W *	DP0 – DP7	I/O	Data bus parity input and output.
1C	$\overline{\text{FDN}}$	I/O	Flush done I/O used for communication between other MPC2605 devices. Must be tied together between all MPC2605 parts along with a pullup resistor.
19E	$\overline{\text{GBL}}$	O	Global transaction. Always negated when MPC2604 is bus master.
1J	$\overline{\text{HRESET}}$	I	Hard reset input from processor bus. This is an asynchronous input that must be low for at least 16 clock cycles to ensure the MPC2605 is properly reset.
3A	$\overline{\text{L2 BG}}$	I	Bus grant input from arbiter.
1D	$\overline{\text{L2 BR}}$	I/O	Bus request I/O. Normally used as an output.
19D	$\overline{\text{L2 CI}}$	I	Secondary cache inhibit sampled, after assertion of $\overline{\text{TS}}$. Assertion prevents linefill.
2F	$\overline{\text{L2 CLAIM}}$	O	L2 cache claim output. Used to claim the bus for processor initiated memory operations that hit the L2 cache. $\overline{\text{L2 CLAIM}}$ goes true (low) before the rising edge of CLK following $\overline{\text{TS}}$ true. Because this output is not always driven, a pullup resistor may be necessary to ensure proper system functioning.
2E	$\overline{\text{L2 DBG}}$	I	Data bus grant input. Comes from system arbiter, used to start data tenure for bus operations where MPC2605 is the bus master.
18D	$\overline{\text{L2 FLUSH}}$	I	Causes cache to write back dirty lines and clears all tag valid bits.

PIN DESCRIPTIONS (continued)

Pin Locations	Pin Name	Type	Description
3B	$\overline{\text{L2 MISS INH}}$	I	Prevents line fills on misses when asserted.
2N	$\overline{\text{L2 TAG CLR}}$	I	Invalidates all tags and holds cache in a reset condition.
3N	$\overline{\text{L2 UPDATE INH}}$	I	Cache disable. When asserted, the MPC2605 will not respond to signals on the local bus and internal states do not change.
3J	$\overline{\text{PWRDN}}$	I	Provides low power mode. Prevents address and data transitions into the RAM array. MPC2605 becomes active 4 µs after deassertion. Clock must be externally disabled.
1N	$\overline{\text{SRESET}}$	I	Soft reset signal from processor bus. This is an asynchronous signal that takes 8 to 16 clock cycles to synchronize. When asserted, this resets the internal data state machine.
1E	$\overline{\text{TA}}$	I/O	Transfer acknowledge status I/O from processor bus.
3K	$\overline{\text{TBST}}$	I/O	Transfer burst status I/O from processor bus. Used to distinguish between burstable and non-burstable memory operations.
2P	TCK	I	Test clock input for IEEE 1149.1 boundary scan (JTAG).
1P	TDI	I	Test data input for IEEE 1149.1 boundary scan (JTAG).
1R	TDO	O	Test data output for IEEE 1149.1 boundary scan (JTAG).
1H	$\overline{\text{TEA}}$	I	Transfer error acknowledge status input from processor bus.
3P	TMS	I	Test mode select for IEEE 1149.1 boundary scan (JTAG).
2R	$\overline{\text{TRST}}$	I	Test reset input for IEEE 1149.1 boundary scan (JTAG). If JTAG will not be used, $\overline{\text{TRST}}$ should be tied low.
3L	$\overline{\text{TS}}$	I/O	Transfer start I/O from processor bus (can also come from any bus master on the processor bus). Signals the start of either a processor or bus master cycle.
17F – 19F *	TSIZ0 – TSIZ2	I/O	Transfer size I/O from processor bus.
1K, 2K, 1L, 2L, 1M *	TT0 – TT4	I/O	Transfer type I/O from processor bus.
3H	WT	I/O	Write through status input from processor bus. When tied to ground, the MPC2605 will operate in write-through mode only (no copy back).
4C, 15C, 16C, 9D – 11D, 8H – 10H, 4J, 8J, 9J, 16J, 4K, 8K, 12K, 16K, 4L, 11L, 12L, 16L, 10M – 12M, 3T, 9T – 11T, 17T, 3U, 4U, 15U, 17U	V_{DD}	Supply	Power supply: 3.3 V ± 5%.
7C, 9C, 13C, 14C, 7D, 8D, 12D, 13D, 4G, 16G – 18G, 4H, 11H, 12H, 16H, 10J – 12J, 9K – 11K, 8L – 10L, 4M, 8M, 9M, 16M, 4N, 16N, 7T, 8T, 12T, 13T, 5U – 7U, 12U – 14U	V_{SS}	Supply	Ground.
3F, 3R	NC	—	No connection. There is no connection to the chip.

*See Pin Assignment for specific pin assignment of these bus signals.

ABSOLUTE MAXIMUM RATINGS (See Note 1)

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{DD}	–0.5 to 4.6	V
Voltage Relative to V_{SS}	V_{in}, V_{out}	–0.5 to V_{DD} + 0.5	V
Output Current (per I/O)	I_{out}	±20	mA
Power Dissipation (Note 2)	P_D	—	
Temperature Under Bias	T_{bias}	–10 to 85	°C
Junction Temperature	T_J	0 to 125	°C
Storage Temperature	T_{stg}	–55 to 125	°C

NOTES: 1. Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to RECOMMENDED OPERATING CONDITIONS. Exposure to higher than recommended voltages for extended periods of time could affect device reliability.

2. Power dissipation capability is dependent on package characteristics and use environment. See Package Thermal Characteristics.

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to these high-impedance circuits.

This BiCMOS memory circuit has been designed to meet the dc and ac specifications shown in the tables, after thermal equilibrium has been established.

DC OPERATING CONDITIONS AND CHARACTERISTICS

(T_J = 20° to 110°C, Unless Otherwise Noted)

RECOMMENDED OPERATING CONDITIONS (Voltages Referenced to V_{SS} = 0 V)

Parameter	Symbol	Min	Typ	Max	Unit
Supply Voltage (Operating Voltage Range)	V_{DD}	3.135	3.3	3.465	V
Input High Voltage	V_{IH}	2.0	—	5.5	V
Input Low Voltage	V_{IL}	–0.5*	—	0.8	V

*V_{IL} (min) = –2.0 V ac (pulse width ≤ 20 ns).

DC CHARACTERISTICS

Parameter		Symbol	Min	Max	Unit
Input Leakage Current (All Inputs, V_{in} = 0 to V_{DD})		$I_{lkg(I)}$	—	±1.0	mA
Output Leakage Current (High-Z State, V_{out} = 0 to V_{DD})		$I_{lkg(O)}$	—	±1.0	mA
AC Supply Current (I_{out} = 0 mA, All inputs = V_{IL} or V_{IH}, V_{IL} = 0 V, and V_{IH} ≥ 3.0 V, Cycle Time = 15 ns, Max Value Assumes a Constant Burst Read Hit, With 100% Bus Utilization, and 100% Hit Rate)	Cycle Time = 12 ns Cycle Time = 15 ns	I_{CCA}	—	760 720	mA
AC Quiescent Current (I_{out} = 0 mA, All Inputs = V_{IL} or V_{IH}, V_{IL} = 0 V and V_{IH} ≥ 3.0 V, Cycle Time = 15 ns, All Other Inputs DC)	Cycle Time = 12 ns Cycle Time = 15 ns	I_Q	—	205 195	mA
Output Low Voltage (I_{OL} = +8.0 mA)		V_{OL}	—	0.4	V
Output High Voltage (I_{OH} = –4.0 mA)		V_{OH}	2.4	—	V

CAPACITANCE (f = 1.0 MHz, T_A = 25°C, Periodically Sampled Rather Than 100% Tested)

Parameter	Symbol	Typ	Max	Unit
Input Capacitance	C_{in}	4	6	pF
Output Capacitance	C_{out}	6	8	pF
Input/Output Capacitance	$C_{I/O}$	8	10	pF

PACKAGE THERMAL CHARACTERISTICS

Rating	Symbol	Max	Unit
Thermal Resistance Junction to Ambient (Still Air, Test Board with Two Internal Planes)	$R_{\theta JA}$	26.5	°C/W
Thermal Resistance Junction to Ambient (200 lfpm, Test Board with Two Internal Planes)	$R_{\theta JA}$	23.2	°C/W
Thermal Resistance Junction to Board (Bottom)	$R_{\theta JB}$	15.9	°C/W
Thermal Resistance Junction to Case (Top)	$R_{\theta JC}$	6.6	°C/W

AC OPERATING CONDITIONS AND CHARACTERISTICS

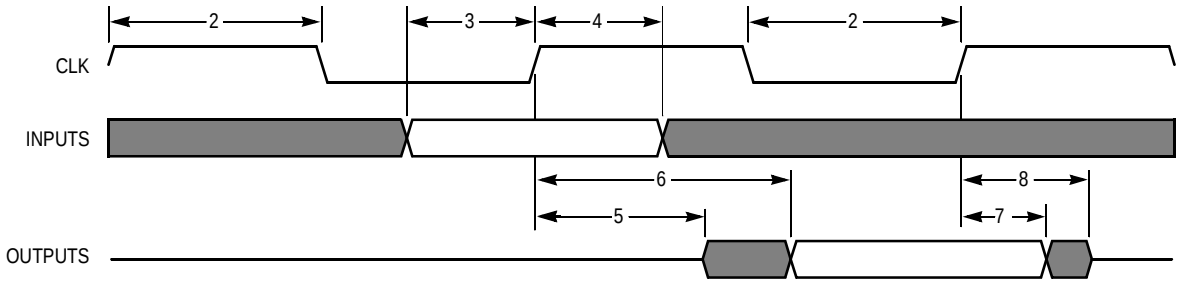
(V_{DD} = 3.3 V ±5%, T_A = 0° to 70°C, Unless Otherwise Noted)

Input Timing Measurement Reference Level . . . . . 1.5 V
Input Pulse Levels . . . . . 0 to 3.0 V
Input Rise/Fall Time . . . . . 2 ns
Output Measurement Timing Level . . . . . 1.5 V
Output Load . . . . . 50 Ω Termination to 1.5 V

AC SPECIFICATIONS

Parameter	Timing Reference	MPC2605-83		MPC2605-66		Unit	Notes
		Min	Max	Min	Max		
Clock Cycle Time	1	12	—	15	—	ns	
Clock High/Low Pulse Width	2	4.8	—	6	—	ns	
Clock Short Term Jitter		—	±200	—	±200	ps	1
Setup for Address, $\overline{\text{ARTRY}}$, $\overline{\text{TA}}$, $\overline{\text{TS}}$, $\overline{\text{TT}}$	3	3.5	—	3.5	—	ns	
Setup for Data and Other Control	3	2.5	—	3	—	ns	
Input Hold Time	4	1	—	1	—	ns	
Clock to Output Low-Z	5	3	—	3	—	ns	1
Clock to Data Output Valid	6	—	8	—	8.5	ns	
Clock to Other Output Valid	6	—	7	—	7.5	ns	
Clock to Output Invalid	7	2	—	2	—	ns	1
Clock to Output High-Z	8	2	10	2	12	ns	1

NOTES: 1. This parameter is sampled and not 100% tested.



MPC2605 RESPONSE TO 60x TRANSFER ATTRIBUTES

TT0 – TT4	$\overline{\text{TBST}}$	$\overline{\text{CI}}$	$\overline{\text{WT}}$	Tag Status	MPC2605 Response	Notes
X1X10	0	1	X	Miss	Line-fill (processor read miss)	1, 2, 3
X1X10	0	1	X	Hit	$\overline{\text{L2 CLAIM}}$, $\overline{\text{AACK}}$, $\overline{\text{TA}}$ (processor read hit)	4
X1010	1	0	X	Hit Clean	Paradox — Invalidate the line (processor n-cacheable read hit clean line)	
X1010	1	0	X	Hit Dirty	Paradox — $\overline{\text{ARTRY}}$, $\overline{\text{L2 BR}}$, then write back data, invalidate the line (processor n-cacheable read hit dirty line)	
00110	0	1	X	Miss	Line-fill except right after a snoop hit to processor (processor write miss)	1, 3, 5, 6
00110	0	1	1	Hit	$\overline{\text{L2 CLAIM}}$, $\overline{\text{AACK}}$, $\overline{\text{TA}}$ except after a snoop hit to processor (processor write hit)	5, 6
00X10	X	1	0	Hit Clean	Cache update (processor write through $\overline{\text{WT}}$ hit clean)	
00110	0	1	0	Hit Dirty	Cache update, clear dirty bit	
00010	1	1	0	Hit Dirty	Paradox — $\overline{\text{ARTRY}}$, $\overline{\text{L2 BR}}$, write back data, keep valid, clear dirty bit	
X0010	1	0	X	Hit Clean	Paradox — Invalidate the line (processor n-cacheable write hit clean line)	
X0010	1	0	X	Hit Dirty	Paradox — $\overline{\text{ARTRY}}$, $\overline{\text{L2 BR}}$, then write back data, invalidate the line (processor n-cacheable SB write hit dirty line)	
00100	X	X	X	Hit Clean	Invalidate tag (flush block address-only)	
00100	X	X	X	Hit Dirty	$\overline{\text{ARTRY}}$, $\overline{\text{L2 BR}}$, write back data, invalidate tag (flush block address-only)	
00000	X	X	X	Hit Clean	No action (clean block address-only)	
00000	X	X	X	Hit Dirty	$\overline{\text{ARTRY}}$, $\overline{\text{L2 BR}}$, write back data, reset dirty bit (clean block address-only)	
01100	X	X	X	Hit	Invalidate tag (kill block address-only)	

NOTES: 1. If a line fill is going to replace a dirty line and the cast out buffer (COB) is full, the line fill will be cancelled. (Unless the line fill is a write that hits in the COB. In this case, the line fill will occur.)

2. If a burst read misses the cache but hits the COB, the MPC2605 will supply the data from the COB, but not perform a line fill.

3. If $\overline{\text{ARTRY}}$ is asserted during a line fill to replace a dirty line, the line fill will be cancelled, the to-be-replaced line will recover its old tag (valid, dirty, tag field), and the COB goes back to an invalid condition, even if the line fill is a burst write to the line in the COB.

4. If $\overline{\text{ARTRY}}$ is asserted during a read hit, the MPC2605 will abort the process.

5. If a processor burst write occurs right after a snoop write that was a cache hit, the MPC2605 will invalidate the line. If the snoop was a cache miss, the MPC2605 will not perform a write allocate.

6. If a processor burst write occurs right after a snoop read that was a cache hit, the MPC2605 will update the cache and clear the dirty bit. If the snoop was a cache miss, the MPC2605 will perform a write allocate.

MPC2605 RESPONSE TO CHIPSET TRANSFER ATTRIBUTES

TT0 – TT4	Tag Status	MPC2605 Response
00100 X0010 X1110	Hit Clean	Invalidate line
00100 X0010 X1110	Hit Dirty	$\overline{\text{ARTRY}}$ and $\overline{\text{L2 BR}}$ write back data, invalidate line (see Note)
00000 X1010	Hit Clean	No action
00000 X1010	Hit Dirty	$\overline{\text{ARTRY}}$ and $\overline{\text{L2 BR}}$, write back data, reset dirty bit (see Note)
0110X 00110	Hit	Invalidate (kill block)

NOTE: In all snoop push cases, $\overline{\text{BR}}$ is sampled the cycle after the ARTRY window. If $\overline{\text{BR}}$ is asserted in this cycle, $\overline{\text{L2 BR}}$ will be immediately negated and an assertion of $\overline{\text{L2 BG}}$ will be ignored

TRANSFER ATTRIBUTES GENERATED FOR L2 COPY BACK

TT0 – TT4	$\overline{\text{TBST}}$	$\overline{\text{CI}}$	$\overline{\text{WT}}$
00010	0	1	1

FUNCTIONAL OPERATION

SYSTEM USAGE AND REQUIREMENTS

The MPC2605 is a high-performance look-aside cache. A look-aside cache is defined as a cache that resides on the same bus as the processor, memory controller, DMA bridge, and arbiter. The advantage of a look-aside cache is that when the processor makes a memory request, the cache adds no delay to the memory controller's response time in the event that the request cannot be satisfied by the cache. However, there are certain system requirements that must be met before a look-aside cache can be used.

Comprehension of $\overline{\text{L2 CLAIM}}$

Because the memory controller sees every memory request that is issued by the processor, there must be a mechanism for the cache to inform the memory controller that it has detected a cache hit and will satisfy the processor's request. The MPC2605 has a signal called $\overline{\text{L2 CLAIM}}$ that is asserted whenever a cache hit is detected. Any memory controller with which the MPC2605 is to be used must have the ability to monitor this signal.

Pipeline Depth

The 60x bus allows pipelining of transactions such that a new transaction can be initiated before a previous transaction has fully completed. The level of pipelining that exists on the bus is defined by how many new data transactions have been initiated while the original transaction is still being processed. By this definition, the MPC2605 can only work in a one level deep pipeline. In the presence of transactions for which it has asserted $\overline{\text{L2 CLAIM}}$, the MPC2605 can control the level of pipelining by delaying its assertion of $\overline{\text{AACK}}$. However, for transactions that it cannot control, the MPC2605 is dependent on the memory controller to control pipeline depth. Thus, another system requirement for the use of the MPC2605 is the use of a memory controller that only allows one level deep of pipelining on the 60x bus.

Bus Mastering

Bus mastering is a requirement only for systems that seek to use the MPC2605 as a copy-back cache, as opposed to a write-through cache. The requirement is that the system arbiter must have the ability to allow the MPC2605 to become a bus master. Specifically, the system arbiter must be able to recognize assertions of $\overline{\text{L2 BR}}$ and must have the ability to assert $\overline{\text{L2 BG}}$ and $\overline{\text{L2 DBG}}$.

These are the only requirements above and beyond what should already exist. All other necessary control signals are signals that are required for the processor to communicate with the memory controller, DMA bridge, and arbiter.

CONFIGURATION PINS

The MPC2605 has five configuration pins: CFG0, CFG1, CFG2, CFG3, and CFG4.

CFG0 – CFG2

These three configuration pins are used to implement the different cache sizes supported by the MPC2605.

256KB: For a single chip implementation, CFG0, CFG1, and CFG2 should all be tied low.

512KB: This two-chip configuration requires both parts to have CFG0 tied low and CFG1 tied high. CFG2 is used as a chip select when it matches the value of A26. Therefore, one device must have CFG2 tied low and the other device must have CFG2 tied high.

1MB: The four-chip configuration requires all four devices to have CFG0 tied high. The CFG1, CFG2 vector becomes the chip select when it matches the A25, A26 vector. Therefore, each of the four parts must have a unique value of the CFG[1:2] vector.

CFG3

Many core logic chipsets are designed such that the DMA bridge and memory controller are resident in the same device. In such systems there is internal communication between these two functional units. Bus transactions generated by the DMA bridge are solely for the purpose of keeping the system coherent. They are not explicit requests from memory that have data tenures associated with them. However, some chipsets are designed with the memory controller and DMA bridge partitioned into different devices. In

systems such as these, transactions generated by the DMA bridge are true memory requests that have data tenures associated with them. These are called snoop data tenures. Because these two types of systems are fundamentally different, the MPC2605 must know in which type of system it is resident in order to respond properly to the different types of transactions. For systems that do not have snoop data tenures, CFG3 must be tied high. For systems that do use snoop data tenures, CFG3 must be tied low.

CFG4

When the MPC2605 asserts $\overline{\text{L2 CLAIM}}$ to signal to the memory controller that a cache hit has been detected, it is taking control of the address and data tenures of the transaction (see **60x BUS OPERATION** and **MEMORY COHERENCE**). This means that the MPC2605 will assert $\overline{\text{AACK}}$ to end the address tenure, and it will assert $\overline{\text{TA}}$ as needed for the data tenure. If the data bus is idle when a processor request is initiated, the MPC2605 will assert $\overline{\text{AACK}}$ the cycle after $\overline{\text{TS}}$ was asserted. If the data bus is busy when the request is made, the MPC2605 will wait until the outstanding data tenure has completed before asserting $\overline{\text{AACK}}$. By holding off on the assertion of $\overline{\text{AACK}}$, the MPC2605 enforces the policy of, at most, two outstanding data transactions at any one time. Tying CFG4 low prevents the MPC2605 from asserting $\overline{\text{AACK}}$ to end transactions, for which it has asserted $\overline{\text{L2 CLAIM}}$. In systems that tie CFG4 low, it is necessary for the memory controller to assert $\overline{\text{AACK}}$ for all transactions. This allows the DMA bridge to initiate snoop transactions (as defined later) even when there are two outstanding data transactions. If this type of system is implemented, the arbiter must ensure that the processor's bus grant is negated once there are two outstanding data transactions. It is expected that most systems will tie CFG4 high.

RESET/INITIALIZATION

To ensure proper initialization and system functionality, the $\overline{\text{HRESET}}$ pin of the MPC2605 should be connected to the same signal that is used to reset the processor. When $\overline{\text{HRESET}}$ is negated, the MPC2605 commences an internal initialization sequence to clear all of the valid bits in the cache. The sequence takes approximately 4000 clock cycles. During this time the MPC2605 will not participate in any bus transaction that occurs. All transactions are, however, monitored so that, regardless of when the initialization sequence completes, the MPC2605 is prepared to take action on the next transaction initiated by the processor.

At some point after this 4000-cycle sequence, the MPC2605 will detect its first cache hit. At this time the system will experience its first assertion of $\overline{\text{L2 CLAIM}}$. If the memory controller must be configured via software to comprehend assertions of $\overline{\text{L2 CLAIM}}$, this configuration operation must be completed by this time. For systems that cannot guarantee that this requirement is met, it is necessary to disable the MPC2605 until such time as this configuration can be guaranteed. Disabling the MPC2605 can be accomplished by asserting $\overline{\text{L2 UPDATE INH}}$ sometime during reset and negating it when it is deemed safe for caching to commence.

60x BUS OPERATION

All transactions have what is called an address tenure. An address tenure is a set number of bus cycles during which the address bus and its associated control signals are being used for the transaction at hand. In general, there are two types of transactions: those that only have address tenures, called address-only transactions; and those that require the use of the data bus; and therefore, will have a data tenure. These transactions are called data transactions. This section describes how address and data tenures are defined as viewed by the MPC2605.

Address Tenures

Address tenures on the 60x bus are fairly well defined. They start with an assertion of $\overline{\text{TS}}$ by a device that has been granted the bus by the system arbiter. This device is called the bus master for this transaction. At the same time that $\overline{\text{TS}}$ is asserted, the bus master also drives the address and all other relevant control signals that define the transaction. $\overline{\text{TS}}$ is only asserted for one cycle, but all other signals are held valid by the bus master until some other device asserts $\overline{\text{AACK}}$. The device that asserts $\overline{\text{AACK}}$ becomes the slave for this transaction. Typically, the slave is the memory controller, although for transactions that are cache hits, the MPC2605 becomes the slave by driving $\overline{\text{L2 CLAIM}}$.

Transactions can be aborted by any device on the bus by asserting $\overline{\text{ARTRY}}$. $\overline{\text{ARTRY}}$ may be asserted at any time after $\overline{\text{TS}}$ is asserted, but must be held through

the cycle after AACK is asserted. This cycle is referred to as the ARTRY window, since it is the cycle that all devices sample ARTRY to determine if the address tenure has been successfully completed.

If an address tenure is not aborted by an assertion of ARTRY, then the next bus master is free to assert TS the cycle after the ARTRY window to start a new address tenure. If ARTRY is asserted in the ARTRY window, all devices that are not asserting ARTRY must negate their bus request in the following cycle. This next cycle is called the BR window. The purpose of this protocol is to give immediate bus mastership to the device that asserted ARTRY with the expectation that the device will take this opportunity to clean up whatever circumstances caused it to assert ARTRY. Typically, this involves writing data back to memory to maintain coherence in the system.

Data Tenures

Data tenures are more complicated to define than address tenures. They require two conditions to start: an assertion of TS that initiates a data transaction and a qualified assertion of the bus master's data bus grant. For a data bus grant to be considered qualified, no device on the bus may be asserting DBB in the cycle that the data bus grant is asserted.

Data transactions come in two types: single-beat transactions and burst transactions. The type is determined by the state of TBST during the address tenure of the transaction. If the bus master asserts TBST, the transaction is a burst transaction and will require four assertions of TA in order to complete normally. If TBST is negated during the address tenure, the transaction only requires one assertion of TA, thus the name single-beat.

The device that drives the data bus during a data transaction depends on whether the transaction is a read or a write. For a read transaction, the slave device drives the data bus. For a write transaction, the master drives the data bus. In all data transactions, the slave device asserts TA to indicate that either valid data is present on the bus, in the case of a read; or that it is reading data off the data bus, in the case of a write. The master device asserts DBB the cycle after it has been granted the data bus and keeps it asserted until the data tenure has completed.

A data tenure can be aborted in two different ways. The address tenure for the transaction can be aborted by an assertion of ARTRY. Or, the slave device may assert TEA to indicate that some error condition has been detected. Either event will prematurely terminate the data tenure.

Data Streaming

For the majority of data transactions there must be a wait state between the completion of one data tenure and the start of the next. This turnaround cycle avoids the contention on the data bus that would occur if one device starts driving data before another device has had a chance to turn off its data bus drivers. When a cache read hit is pipelined on top of another cache read hit, there is no need for this turnaround cycle since the same device will be driving the data bus for both data tenures. The 60x bus has the ability to remove this unnecessary wait state and allow back-to-back cache read hits to stream together. This ability is only enabled if the system is put into Fast L2 mode. Note that not all processors implementing the PowerPC architecuture support Fast L2 mode.

One of the requirements for taking advantage of this data streaming capability is that the system arbiter must be sophisticated enough to identify situations where streaming may occur. on recognizing these situations, it must assert the processor's data bus grant in the cycle coincident with the fourth assertion of TA of the first cache read, so that the data tenure for the second cache read may commence in the next cycle.

Because it only recognizes qualified assertions of CPU DBG, the MPC2605 must not be aware of the processor’s assertions of DBB. This means that the DBB pin of the MPC2605 must be tied to a pullup resistor rather than connected to the system DBB. This forces the system arbiter to a level of sophistication such that it only supplies qualified data bus grants and thus, the DBB signal is unnecessary to the whole system.

Note: In a multi-chip configuration each MPC2605 device acts as an independent cache. Zero wait state data streaming can only occur if the back-to-back read hits occur in a given device. If the second read hit is not in the device as the first read hit, a wait state will occur between the two data tenures (2-1-1-1-2-1-1-1 timing).

Data Bus Parking

The MPC2605 has the ability to respond to a processor read or write hit starting in the cycle after the processor has asserted TS. This is referred to as a

2-1-1-1 response. However, even though the MPC2605 has this ability, it is dependent on the system to allow this quick of a response to occur. As discussed above, a data tenure cannot start until the master has been given a qualified bus grant. In order for the data tenure to start the cycle after $\overline{TS}$ is asserted, the data bus must be granted in the cycle coinciding with the assertion of $\overline{TS}$. At bus speeds of 66 MHz, it is extremely difficult for an arbiter to detect an assertion of $\overline{TS}$ and itself assert $\overline{CPU\ DBG}$ in the same cycle. In order to realistically allow this situation to occur, $\overline{CPU\ DBG}$ must be asserted independent of the processor's assertion of $\overline{TS}$.

Data bus parking is a system feature, whereby the processor always has a qualified data bus grant when the data bus is idle. It is also a requirement for systems that seek to take advantage of the 2-1-1-1 response time capabilities of the MPC2605. This feature is typically present in arbiters that have the level of sophistication necessary to support data streaming. But it is also a feature of systems that do not even have a data bus arbiter. In these systems, the data bus grant of every device in the system is tied to ground. The assertion of $\overline{DBB}$ by the current data bus master effectively removes the qualified data bus grant of all devices in the system, including its own. Note that in systems that have no data bus arbiter, it is impossible to take advantage of data streaming.

There is another caveat associated with data bus parking. Care must be taken when using data bus parking along with Fast L2 mode. In normal bus mode when the processor reads data off the bus, it will wait one cycle before passing the data on to internal functional units. The purpose of this one cycle waiting period is to check for an assertion of $\overline{DRTRY}$ that invalidates the data that has been previously read. One of the advantages of running the processor in Fast L2 mode is that this internal processor wait state is removed.

A problem will arise, however, if the processor is given data the cycle after $\overline{TS}$ is asserted, as is possible with the MPC2605, and the transaction is aborted by some other device asserting $\overline{ARTRY}$. Because the processor will not sample $\overline{ARTRY}$ until two cycles after the assertion of $\overline{TS}$, the data read off the bus will have already been forwarded to the internal functional units. Thus, incorrect results may occur in the system.

To avoid this situation in a system that seeks to run Fast L2 mode with the data bus parked, there must be a guarantee that $\overline{ARTRY}$ will never be asserted for cache read hits. This is a further requirement to be imposed on the DMA bridge and the memory controller. If this guarantee cannot be made, the data bus cannot be parked when running in Fast L2 mode.

Processor Reads

When the processor issues a read transaction, the MPC2605 does a tag lookup to determine if this data is in the cache. If there is a cache hit and $\overline{CI}$ is not asserted, the MPC2605 will assert $\overline{L2\ CLAIM}$ and supply the data to the processor when the data tenure starts.

If the processor issues a cache-inhibited read ($\overline{CI}$ asserted) and the MPC2605 detects a cache hit to a non-dirty or clean cache line, the line will be marked invalid. If the cache-inhibited read hits a dirty line, the MPC2605 will assert $\overline{ARTRY}$ and write the dirty line back to memory.

If the read misses in the cache, the MPC2605 will perform a linefill only if it is a burst read and it is not marked cache-inhibited. During a linefill, the MPC2605 stores the data present on the bus as it is supplied by the memory controller.

Processor Writes

The conditions for asserting $\overline{L2\ CLAIM}$ for processor writes are almost the same as for processor reads. There must be a cache hit and $\overline{CI}$ must not be asserted. In addition, however, $\overline{WT}$ must not be asserted. Single beat writes that are marked either write-through or cache-inhibited that hit in the cache, cause the MPC2605 to assert $\overline{ARTRY}$ and write the dirty line back to memory.

Transaction Pipelining

As explained in **Pipeline Depth**, the MPC2605 can only handle one level of pipelining on the bus. Since the assertion of $\overline{L2\ CLAIM}$ gives it the ability to assert $\overline{AACK}$, the MPC2605 has the ability to control this pipeline depth for transactions that are cache hits by delaying its assertion of $\overline{AACK}$.

Pipelined cache hits are transactions that hit in the cache but occur while there is still an outstanding data transaction on the bus. The timing of the assertion of $\overline{AACK}$ for a pipelined cache hit is dependent on the completion of the previous transaction. For explanation purposes, the previous transaction will be referred to as transaction one. The pipelined cache hit will be referred to as transaction two.

If transaction one is a cache hit, the MPC2605 will be the slave device for the transaction. Since, for burst operations, the MPC2605 always asserts $\overline{TA}$ for four consecutive clock cycles, the end of the data tenure for transaction one will be at a deterministic clock cycle. In this case, $\overline{AACK}$ for transaction two can be asserted coinciding with the last assertion of $\overline{TA}$ for transaction one. If transaction one is not a cache hit, the MPC2605 will wait until after the data tenure for transaction one has completed before asserting $\overline{AACK}$ to complete the address tenure of transaction two.

MEMORY COHERENCE

When a processor brings data into its on-chip cache and modifies it, a situation has arisen that the main memory now contains irrelevant or stale data. Given that most systems support some form of DMA, there must exist a means that forces the processor to write this modified or dirty data back to main memory. The DMA bridge is responsible for generating bus transactions to ensure that main memory locations accessed by DMA operations do not contain stale data. These transactions, called snoops, come in three different categories, each will be discussed below.

Snoops cause the processor and the MPC2605 to check to see if they have dirty copies of the memory location specified in the snoop transaction. If either device does have a dirty copy it will assert $\overline{ARTRY}$ and make use of the opportunity presented in the BR window to write this data back to main memory.

Situations can arise where a cache line is dirty in both the processor's L1 cache and in the MPC2605. In cases such as these, snoop transactions should cause the processor to write its data back to memory, since it is by definition more recent than the data in the MPC2605. Since $\overline{ARTRY}$ is a shared signal and it cannot be determined which devices are driving it, the MPC2605 samples $\overline{CPU\ BR}$ in the BR window to determine if the snoop hit a dirty line in the L1 cache. If $\overline{CPU\ BR}$ is asserted during this window, the MPC2605 will defer to the processor.

Snoop Reads

A snoop read causes dirty data to be written back to memory, but allows both the L1 and L2 to keep a valid copy. In cases where the snoop hits a dirty cache line in the processor, the MPC2605 will update its contents as the processor writes the data back to main memory.

Snoop reads can be implemented in two ways. One is that the DMA bridge can issue a clean transaction (TT[0:4] = 00000). The other is that the DMA bridge can do a read transaction (TT[0:4] = x1010). If the DMA bridge does a read transaction, the MPC2605 determines that it is a snoop read rather than a processor read by the state of $\overline{CPU\ BG}$ the cycle before $\overline{TS}$ was asserted. If the processor was not granted the bus, then the transaction had to have been issued by the DMA bridge and is, therefore, a snoop read.

Snoop Writes

Snoop writes also cause dirty data to be written back to main memory. The difference from a snoop read is that the cache line must then be invalidated in both the processor's cache and in the L2 cache. When the processor writes data back to memory in response to a snoop write, the MPC2605 will not cache the data as it appears on the bus. If a valid copy resides in the cache, the MPC2605 will invalidate it.

Again, there are multiple transactions that can be used by the DMA bridge to implement a snoop write. It can issue a flush transaction (TT[0:4] = 00100), a read with intent to modify (TT[0:4] = x1110), or a write with flush (TT[0:4] = 00010). As with snoop reads, the MPC2605 distinguishes between processor issued data transactions and snoop transactions by the state of $\overline{CPU\ BG}$ in the cycle previous to the assertion of $\overline{TS}$.

Snoop Kills

Kills are snoops that cause cache entries to be immediately invalidated, regardless of whether they are dirty. This saves time if the DMA operation is going to modify all the data in the cache line. To implement a snoop kill, the DMA bridge can issue a kill transaction (TT[0:4] = 01100) or a write with kill (TT[0:4] = 00110).

TWO-/FOUR-CHIP IMPLEMENTATION

Multiple Castouts

Because each MPC2605 has its own castout buffer (COB), it is possible for situations to arise where more than one device is needed to do a copy-back operation. Under normal circumstances, each device will enter castout conditions at different times. In these cases, when a device determines that it needs to do a castout,

the $\overline{\text{L2 BR}}$ signal is sampled first. If $\overline{\text{L2 BR}}$ is already asserted, then it is clear that another device is also in a castout situation. The late device will wait until $\overline{\text{L2 BR}}$ is negated before continuing in its attempt to perform a castout.

Because of the BR window protocol associated with assertions of $\overline{\text{ARTRY}}$, it is possible for a situation to arise where device two is waiting for device one to do its castout before asserting $\overline{\text{L2 BR}}$. If there is an assertion of $\overline{\text{ARTRY}}$ by a device other than device one, device one is required to negate $\overline{\text{L2 BR}}$ in the BR window. In order to prevent device two from interpreting device one's negation of $\overline{\text{L2 BR}}$ as an indication that device one has completed its castout, a simple arbitration mechanism is used. All devices have a simple two-bit counter that is synchronized such that all counters always have the same value. For the purposes of performing a castout operation, a given pair can only assert $\overline{\text{L2 BR}}$ if the counter is equal to its value of CFG[1:2]. This simple mechanism prevents more than one device from asserting $\overline{\text{L2 BR}}$ in the same cycle and therefore, not being cognizant of the another device's need to perform a castout.

Snoop Hit Before Castout

The other situation that can cause problems with a shared bus request occurs when a snoop hits a dirty line in one of the MPC2605 devices. If device one has a cache line in its COB, it will assert $\overline{\text{L2 BR}}$ so that it may perform a castout operation. If a snoop hits a dirty line in device two, it will assert both $\overline{\text{ARTRY}}$ and $\overline{\text{L2 BR}}$ so that it can write the snoop data back to main memory. When device one detects that $\overline{\text{ARTRY}}$ has been asserted, it needs to be made aware that device two needs to request the bus. Otherwise, at the same time that device two is asserting $\overline{\text{L2 BR}}$, device one will attempt to conform to the BR window protocol and negate $\overline{\text{L2 BR}}$. This situation is avoided by device one sampling $\overline{\text{FDN}}$ when it detects that $\overline{\text{ARTRY}}$ has been asserted. If $\overline{\text{FDN}}$ is asserted at the same time as $\overline{\text{ARTRY}}$ is asserted, device one will recognize that device two is asserting $\overline{\text{ARTRY}}$. Device one will then High-Z $\overline{\text{L2 BR}}$, so that there will not be contention when device two is asserting $\overline{\text{L2 BR}}$.

MULTIPROCESSING

The MPC2605 can be used as a common cache for up to four processors. For each processor, there is a bus request, bus grant, and data bus grant signal pin on the MPC2605. Each of these pins need to be connected to the respective processor's arbitration signals in the system.

The MPC2605 treats multiple processors as one processor. Thus, the same restrictions on pipelining depth are true with regard to how many processor transactions can be outstanding at any one time. There can only be one data transaction from any processor pipelined on top of a current data transaction that was issued by any processor.

The data tenures for all processors must be performed in the same order as the address tenures on a system-wide basis. If processor one makes a request and then processor two makes a request, processor one's data tenure must precede processor two's data tenure. Note that this is not a 60x bus restriction, but rather a restriction necessary for proper operation of the MPC2605.

The MPC2605 keeps coherent with the L1 caches of multiple processors as defined by the MESI (modified-exclusive-shared-invalid) protocol without actually implementing the protocol. This is possible for two reasons. Since the MPC2605 is a look-aside cache, all transactions are monitored by all devices on the bus. Also, the MPC2605 cannot, on its own, modify data. Thus, if one processor requests exclusive access to a cache line, it is not necessary for the MPC2605 to invalidate its copy of the data, as would be required under the MESI protocol. If a second processor requests the same data, the transaction will cause the first processor to assert $\overline{\text{ARTRY}}$. This will prevent the MPC2605 from supplying stale data to the second processor.

As discussed in **Data Bus Parking**, care must be taken when parking the data bus in Fast L2 mode. By the nature of MP systems running under the MESI protocol, there will be assertions of $\overline{\text{ARTRY}}$ to abort cache read hits. Thus, in an MP system, the data bus cannot be parked to any processor if the system is to be run in Fast L2 mode.

$\overline{\text{PWRDN}}$

An assertion of $\overline{\text{PWRDN}}$ will cause the MPC2605 to go into a low-power sleep state. This state is entered after $\overline{\text{PWRDN}}$ is synchronized and both the address and data buses are idle. All data is retained while in the sleep state.

The behavior of the MPC2605 on negation of $\overline{\text{PWRDN}}$ is dependent on the state of $\overline{\text{WT}}$ at the rising

edge of HRESET. If WT is asserted at reset, the MPC2605 will invalidate all cache entries when PWRDN is negated. If WT is negated at reset, the MPC2605 will leave all cache entries as they were prior to the assertion of PWRDN. However, in this situation, the system designer must ensure that no bus activity occur within 2 microseconds of the negation of PWRDN.

Note: While in the sleep state, the MPC2605 does not disable its internal clock network.

ASYNCHRONOUS SIGNALS

The MPC2605 supports four asynchronous control signals. These signals were originally defined in the PowerPC reference platform (PReP) specification. Because these signals are defined to be asynchronous, the MPC2605 must synchronize them internally. This process takes eight clock cycles. Thus, to guarantee recognition by the MPC2605, assertions of any one of these signals must last a minimum of eight clock cycles.

L2 FLUSH

When L2 FLUSH is asserted, the MPC2605 initiates an internal sequence that steps through every cache line present. Valid lines that are clean, are immediately marked invalid. Valid lines that are dirty, must be written back to main memory.

To keep memory up-to-date, the MPC2605 must still monitor all transactions on the bus. Any transaction that is not a processor burst write will cause the MPC2605 to assert ARTRY. Burst writes cause the MPC2605 to do a lookup on the affected address and mark the line invalid if it is present.

Because the MPC2605 must still monitor all transactions, it cannot use the tag RAM for the flush sequence unless there is a guarantee that no new transactions will be initiated on the bus. The only way to ensure that no new transactions will occur is for the MPC2605 to be granted the bus. Thus, on entering the sequence initiated by the assertion of L2 FLUSH, the MPC2605 will assert L2 BR. As soon as L2 BG is asserted, the MPC2605 can start stepping through the tag RAM entries.

L2 FLUSH need not be held asserted for the flush sequence to complete. Once started, the sequence will run to completion unless overridden by an assertion of HRESET.

L2 MISS INH

When L2 MISS INH is asserted, the MPC2605 will not load any new data into the cache. The data already present will remain valid and the MPC2605 will respond to cache hits. This condition only lasts as long as L2 MISS INH is asserted. When L2 MISS INH is negated, the MPC2605 will start to bring new data into the cache when there are cache misses.

L2 TAG CLR

When L2 TAG CLR is asserted, the MPC2605 will invalidate all entries in the cache. This internal sequence is the same as the one initiated by an assertion of HRESET. During this sequence, the MPC2605 will not participate in any bus transaction. However, it will keep track of all bus transactions so that when the sequence is finished, the MPC2605 can immediately participate in the next bus transaction.

As is the case with assertions of L2 FLUSH, an assertion of L2 TAG CLR need not be held for the duration of the sequence. Once asserted, the sequence will run to completion regardless of the state of L2 TAG CLR.

L2 UPDATE INH

When L2 UPDATE INH is asserted, the MPC2605 is disabled from responding to cacheable transactions. Bus transactions continue to be monitored so that as soon as L2 UPDATE INH is negated, the MPC204GA can participate in the next transaction.

READ HIT/WRITE HIT

Figure 1 shows a read hit from an idle bus state. The MPC2605 asserts $\overline{\text{L2 CLAIM}}$ the cycle after $\overline{\text{TS}}$ to inform the memory controller that there is a cache hit and the cache will control the rest of the transaction. $\overline{\text{L2 CLAIM}}$ is held through the cycle after $\overline{\text{AACK}}$ is asserted. Since there are no active data tenures from previous transactions, the MPC2605 asserts $\overline{\text{AACK}}$ the cycle after $\overline{\text{TS}}$ is asserted. Note that there must be a qualified assertion of $\overline{\text{CPU DBG}}$ in the same cycle as the assertion of $\overline{\text{TS}}$ for the MPC2605 to respond with $\overline{\text{TA}}$ in the next cycle. $\overline{\text{CPU DBG}}$ does not affect the timing of $\overline{\text{L2 CLAIM}}$ or $\overline{\text{AACK}}$.

The write hit timing is virtually the same. The only difference is the processor drives the data instead of the MPC2605.

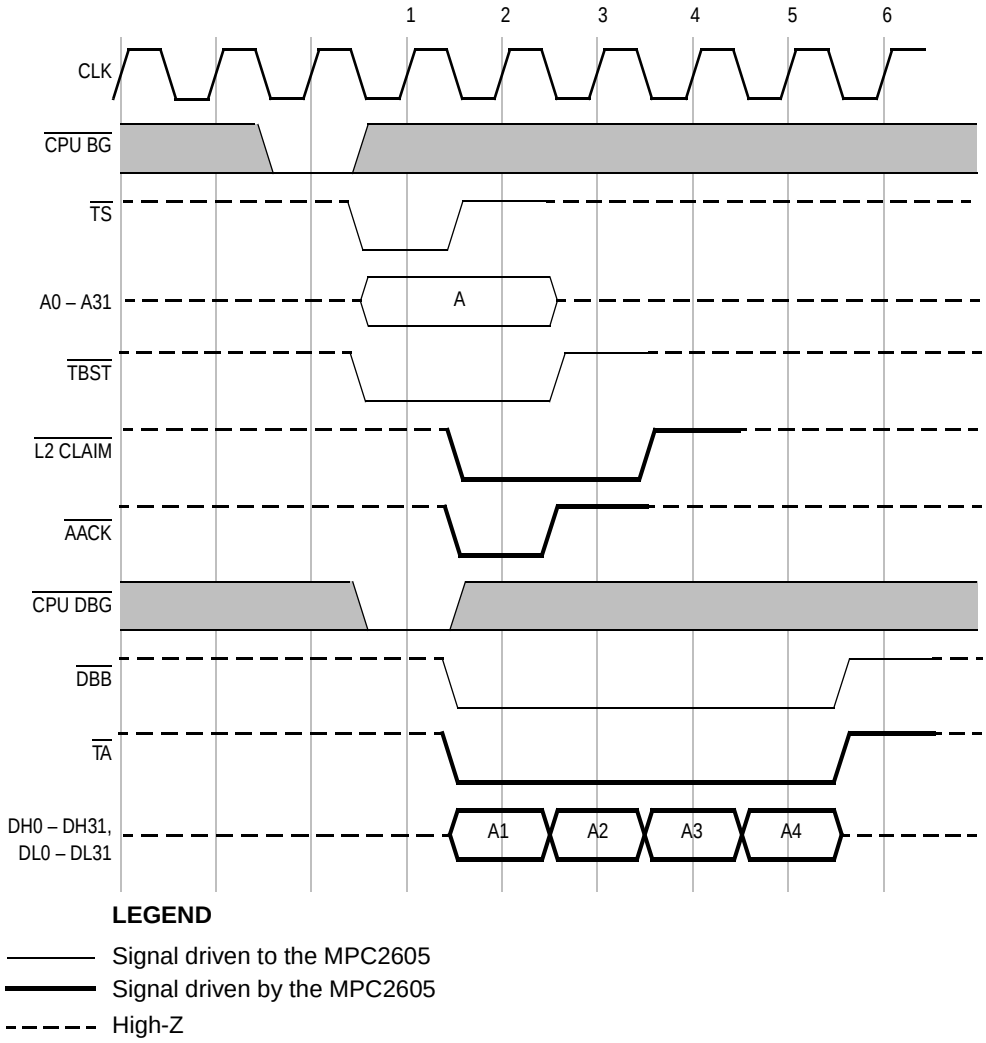


Figure 1. Burst Read (or Write) Hit

MULTIPLE READ/WRITE HITS (NORMAL BUS MODE)

Figure 2 is an illustration of MPC2605 pipeline depth limit with multiple read hits. The MPC2605 supports only one level of address pipelining for data transfer. Therefore, it must hold off on its assertion of $\overline{\text{AACK}}$ for a pipelined $\overline{\text{TS}}$ until the data tenure for the first $\overline{\text{TS}}$ is done. The MPC2605 asserts $\overline{\text{AACK}}$ at the same time as the fourth $\overline{\text{TA}}$ for data tenures that it controls.

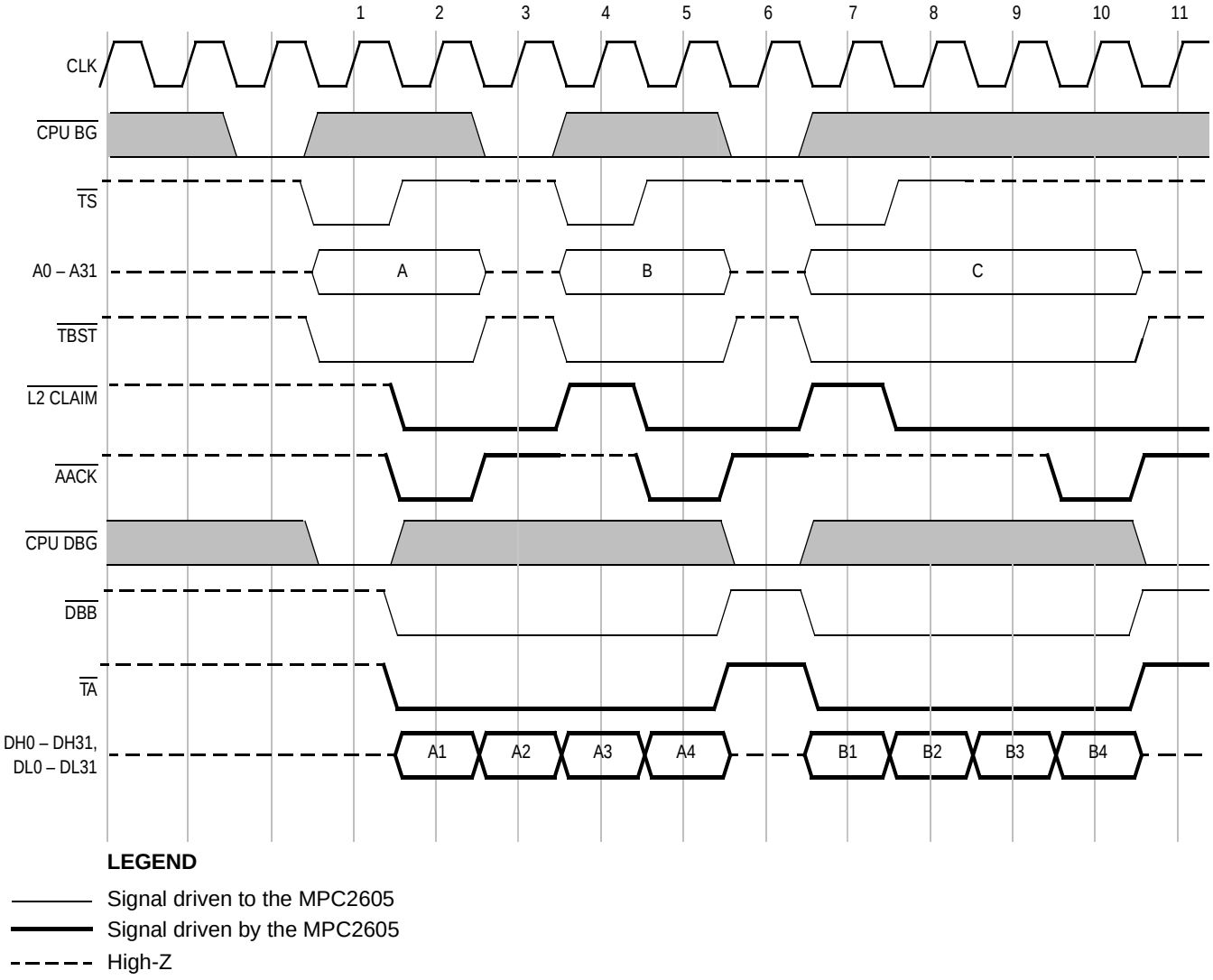


Figure 2. Multiple Burst Read (or Write) Hits

READ MISS (NORMAL BUS MODE)

Figure 3 is an illustration of the MPC2605 pipeline depth with a read miss followed by a read hit.

For illustration purposes, the read miss is shown as a 3-1-1-1 response from memory. $\overline{\text{AACK}}$ for the second access is not driven true until the cycle after the fourth $\overline{\text{TA}}$ of the read miss. This is because the MPC2605 is not in control of $\overline{\text{TA}}$ for the first access and must, therefore, wait until the first access' data tenure is complete before it can drive $\overline{\text{AACK}}$ true for the read hit.

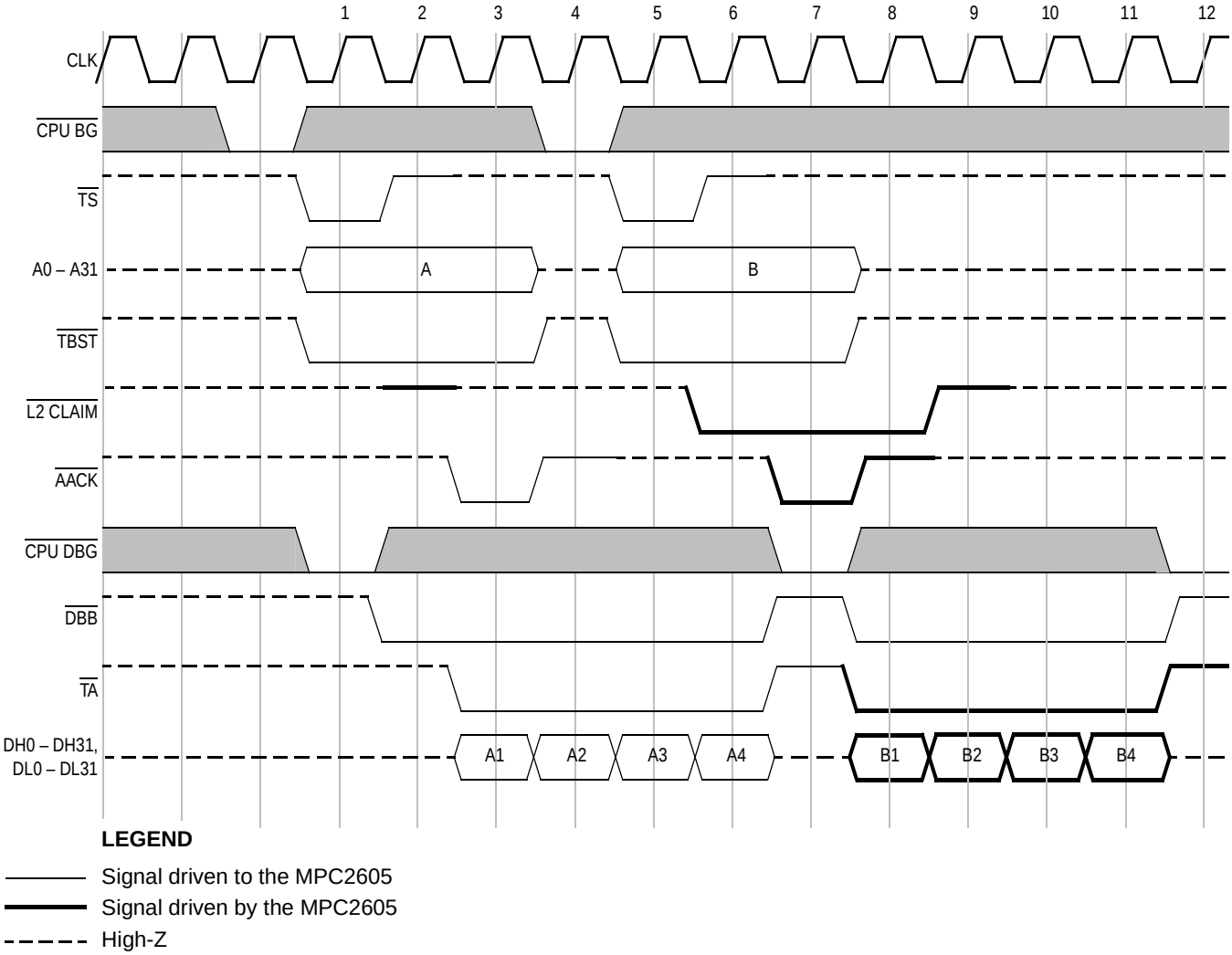


Figure 3. Read Miss Followed by a Burst Hit

MULTIPLE READ HITS (FAST L2 MODE)

Back-to-back pipelined burst read hits for the MPC2605 in Fast L2 mode, also called data streaming mode, are shown in Figure 4. Note that $\overline{\text{CPU DBG}}$ is negated, except for the cycles coincident with the fourth $\overline{\text{TA}}$ of each data tenure. This is a requirement for data streaming. Note also that $\overline{\text{DBB}}$ is not shown. For proper operation in Fast L2 mode, the $\overline{\text{DBB}}$ pin of the MPC2605 must be tied to a pull-up resistor.

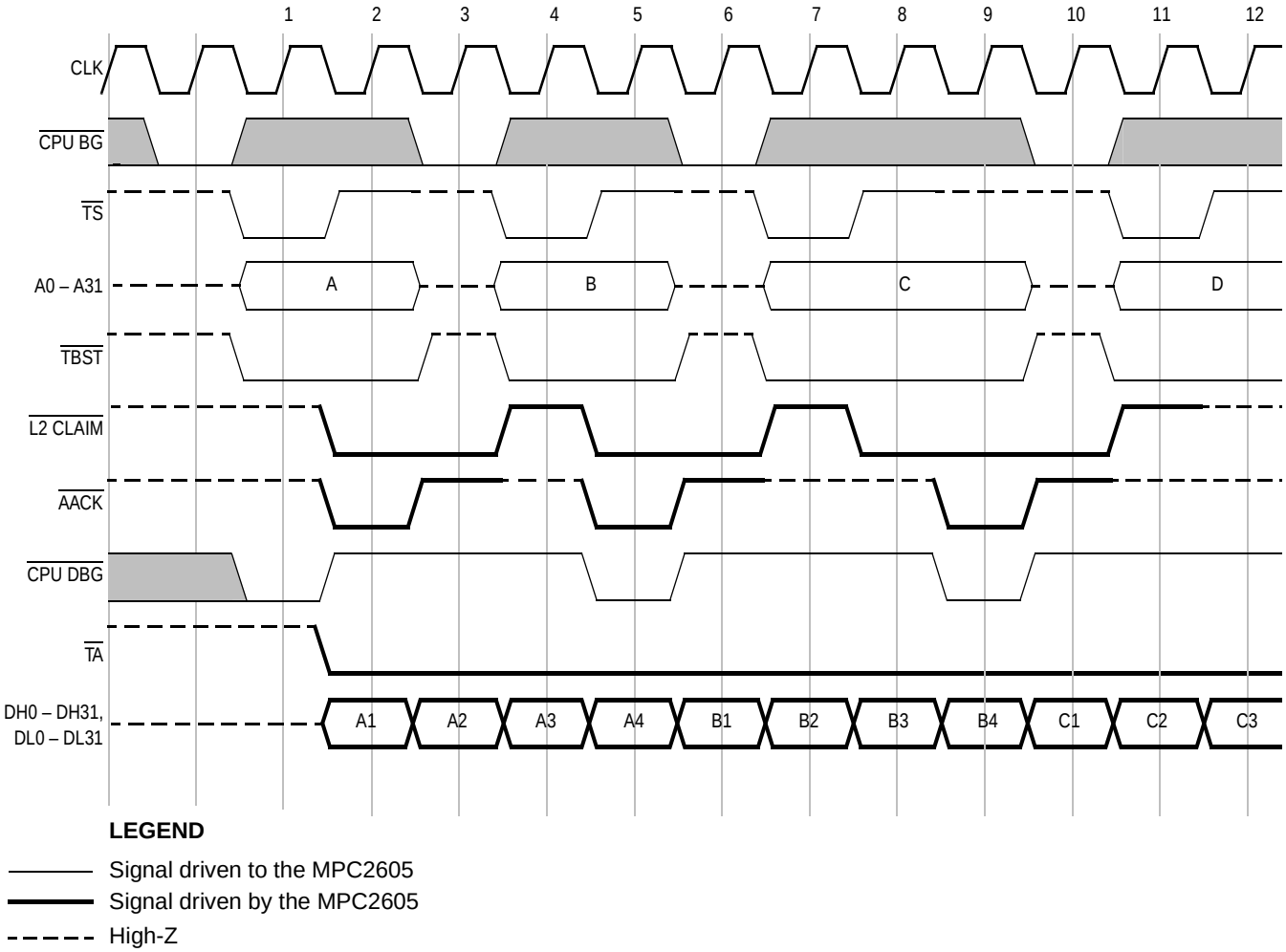


Figure 4. Multiple Burst Read Hits in Fast L2 Mode

WRITE THROUGH BURST WRITE HIT

Figure 5 shows the fastest possible burst write hit to a write-through mode L2 cache line, read miss or write miss processing that replaces a clean line. For these operations, the MPC2605 will not assert any signals on the 60x bus. A cache line is considered write-through if $\overline{WT}$ is asserted by the processor when it asserts $\overline{TS}$.

The speed that a write-through operation completes, is solely dependent on the memory controller. The timing shown here assumes that the memory controller has a write buffer that can accept data this quickly.

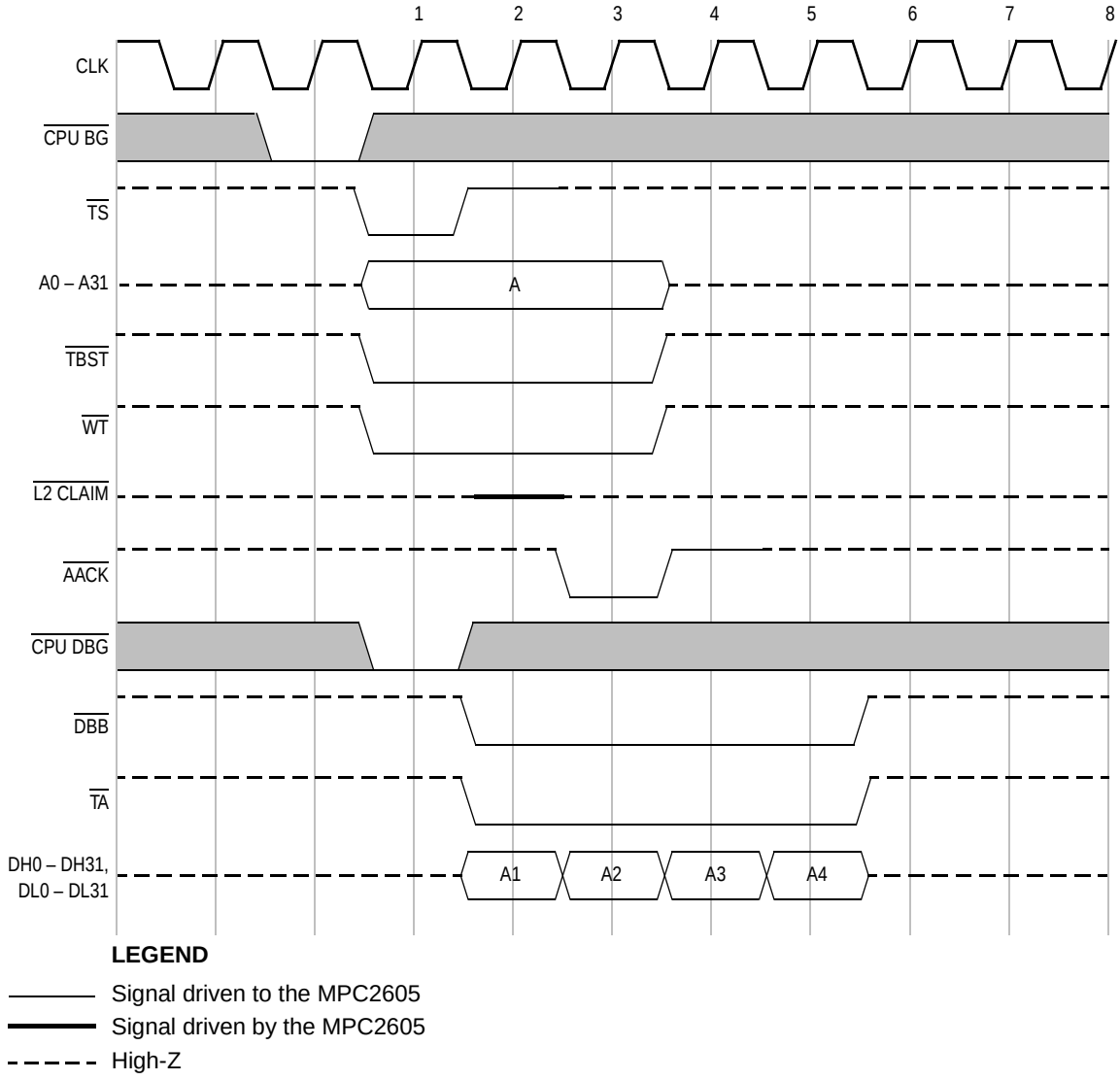


Figure 5. Fastest Possible Write Through Burst Write

READ/WRITE MISS

Figure 6 is an illustration of a processor read or write miss that causes the MPC2605 to replace a dirty line. $\overline{\text{L2 BR}}$ is asserted two clocks after $\overline{\text{TS}}$. The dirty data to be replaced is moved into the internal COB at the same time the new data is written into the cache. Note that the copy-back operation occurs after the processor request is satisfied. In addition, no delay is added to the processor transaction. It proceeds as fast as the memory controller will allow.

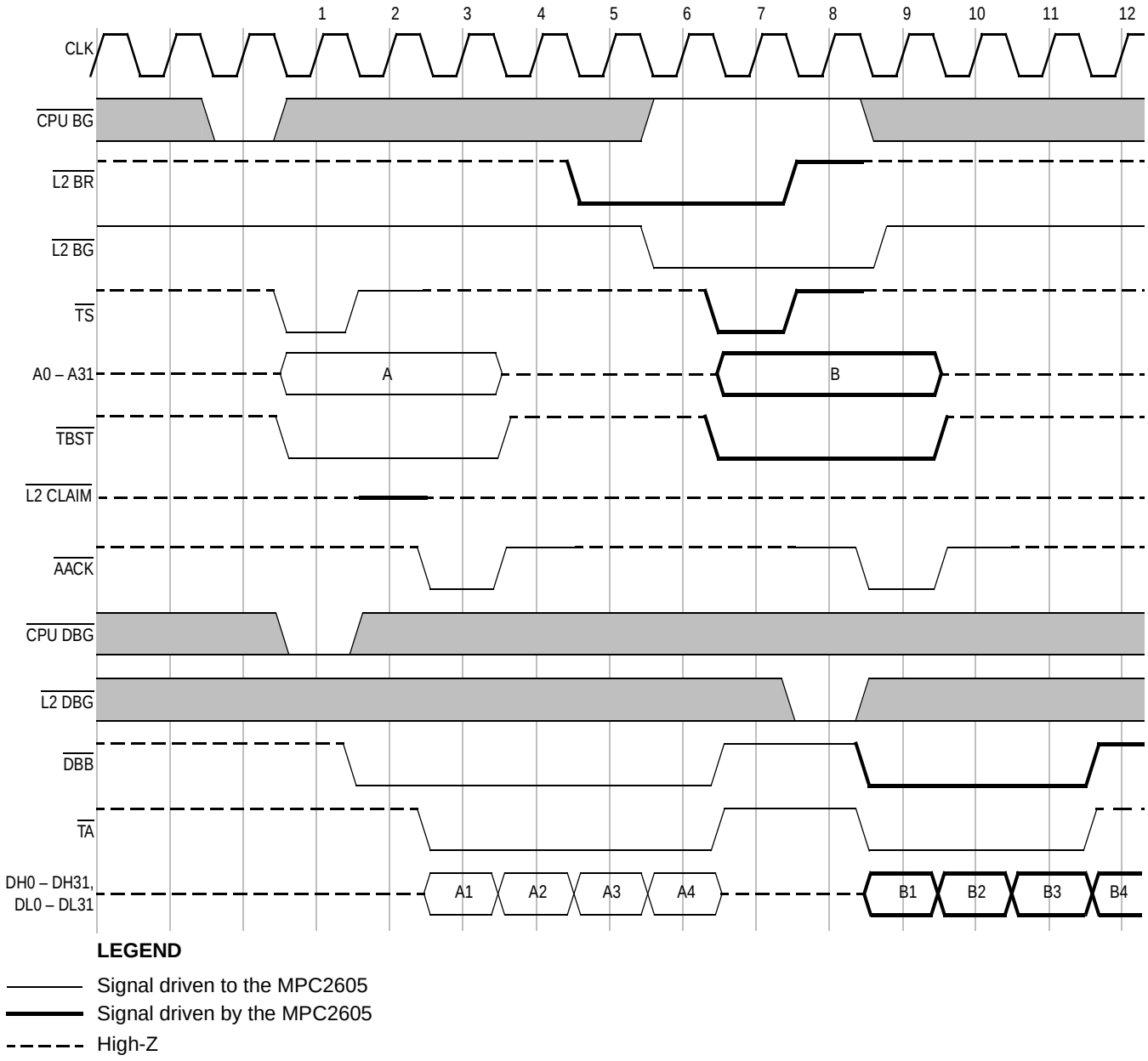


Figure 6. Read or Write Miss Followed by Castout

READ/WRITE SNOOP HIT (DIRTY L2 LINE)

Figure 7 is an illustration of a read or write snoop to a cache line that is dirty in the L2, but is not dirty in the processor's cache. When a snoop hits a dirty line, the MPC2605 will assert $\overline{\text{ARTRY}}$ through the cycle following the assertion of $\overline{\text{AACK}}$. This cycle is called the ARTRY window. Note that the MPC2605 also asserts $\overline{\text{L2 BR}}$ at the same time it asserts $\overline{\text{ARTRY}}$. Because the snoop could also have hit a dirty line in the processor's cache, the MPC2605 samples the processor's $\overline{\text{BR}}$ signal, the cycle following the ARTRY window. This cycle is called the BR window. If the processor's $\overline{\text{BR}}$ signal is not asserted, the MPC2605 will start sampling $\overline{\text{L2 BG}}$, the cycle after the BR window.

Note that the MPC2605 cannot do a 2-1-1-1 copy back burst. The earliest that it can handle the first assertion of $\overline{\text{TA}}$ is two cycles after its assertion of $\overline{\text{TS}}$.

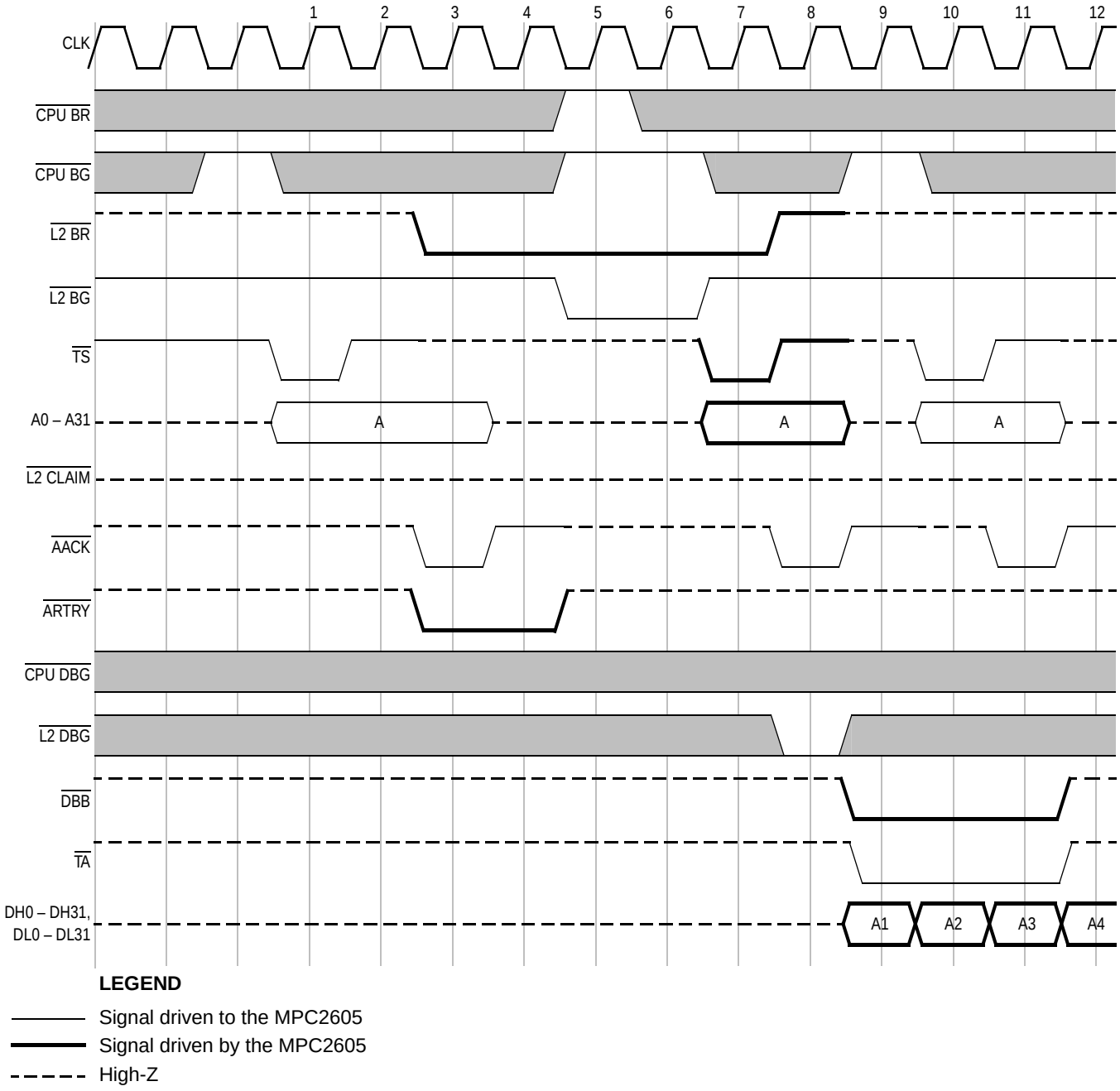


Figure 7. Read or Write Snoop Hit to Dirty L2 Cache Line and Clean Processor Cache Line

READ/WRITE SNOOP HIT (DIRTY L2 AND PROCESSOR LINE)

An illustration of a read or write snoop hit to a dirty L2 cache line is shown in Figure 8. The processor has a dirty copy of the cache line. In this case, both the processor and MPC2605 assert $\overline{\text{ARTRY}}$. This situation is detected by sampling $\overline{\text{CPU BR}}$ in the BR window, as described in the previous example. If $\overline{\text{CPU BR}}$ is asserted in the BR window, the MPC2605 will negate $\overline{\text{L2 BR}}$. It will also ignore assertions of $\overline{\text{L2 BG}}$. This allows the processor to write back its dirty cache line. At this time, the MPC2605 will either update or invalidate its copy depending on whether it is a snoop read or snoop write.

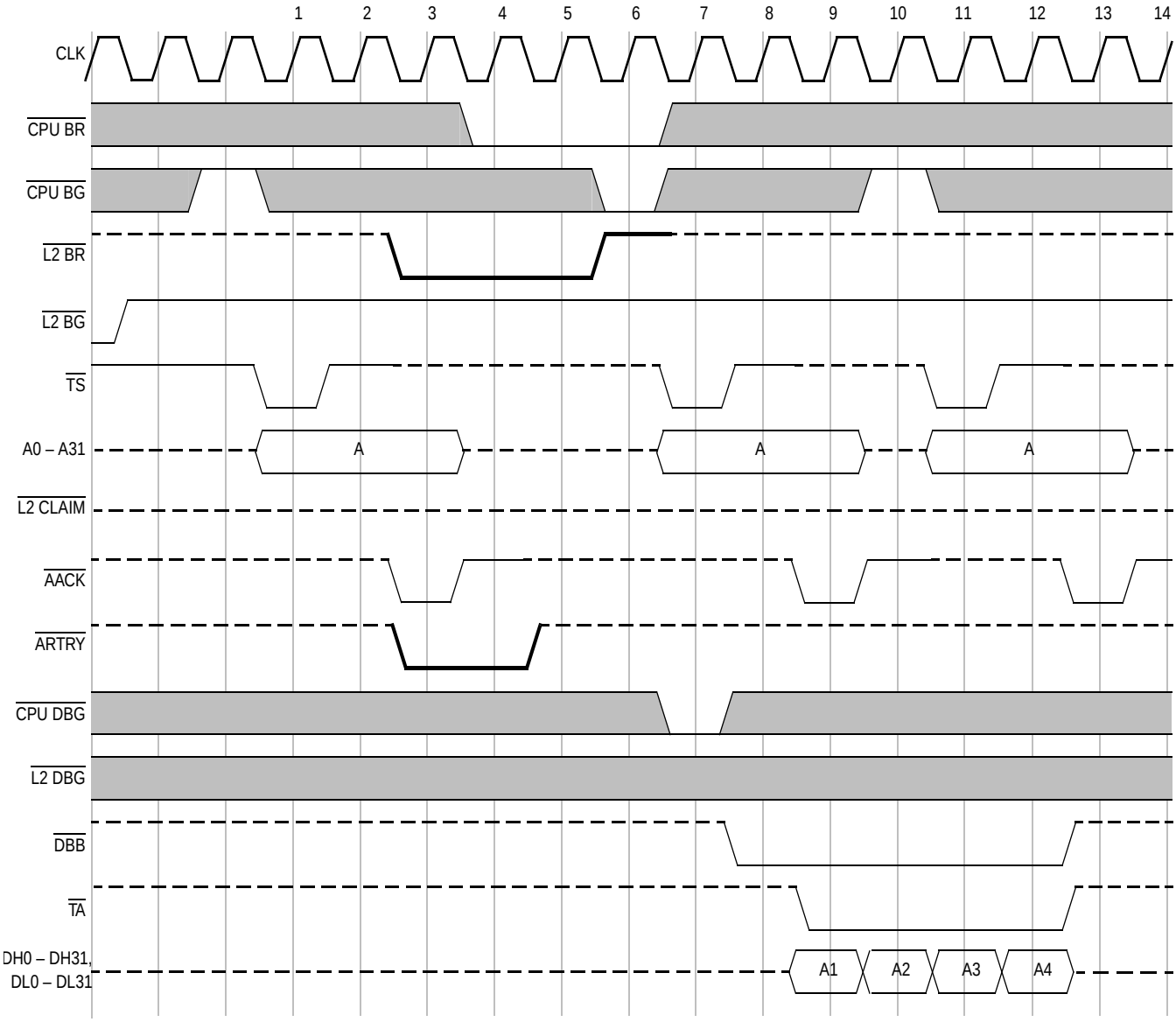


Figure 8. Read or Write Snoop Hit to Dirty L2 Cache Line and Dirty Processor Cache Line

READ HIT/WRITE HIT (WITHOUT $\overline{\text{CPU DBG}}$ PARKED)

Most of the previous examples have assumed $\overline{\text{CPU DBG}}$ is asserted in the same cycle that the processor asserts $\overline{\text{TS}}$. This implies $\overline{\text{CPU DBG}}$ is parked. In some systems it may not be desirable or possible to park $\overline{\text{CPU DBG}}$. Figure 9 shows the response for a read hit from the MPC2605, as gated by the assertion of $\overline{\text{CPU DBG}}$. The fastest response possible in a system that does not park $\overline{\text{CPU DBG}}$ is 3-1-1-1.

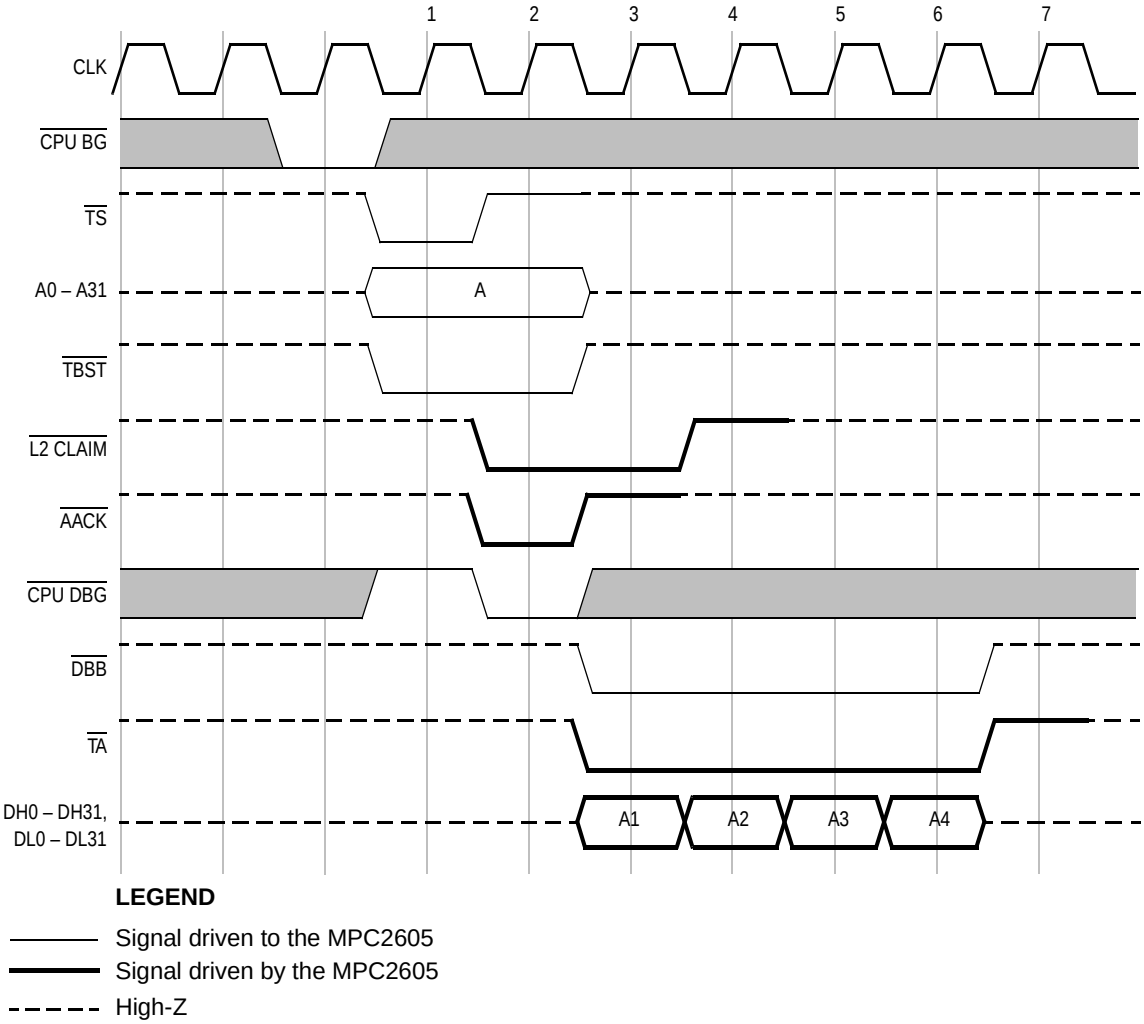


Figure 9. Burst Read (or Write) Hit Without $\overline{\text{CPU DBG}}$ Parked

JTAG

AC OPERATING CONDITIONS AND CHARACTERISTICS FOR THE TEST ACCESS PORT (IEEE 1149.1)

(T_J = 20° to 110°C, Unless Otherwise Noted)

Input Timing Measurement Reference Level . . . . . 1.5 V
Input Pulse Levels . . . . . 0 to 3.0 V
Input Rise/Fall Time . . . . . 3 ns

Output Measurement Timing Level . . . . . 1.5 V
Output Load . . . . . 50 Ω Termination to 1.5 V

TAP CONTROLLER TIMING

Parameter	Symbol	MPC2605-66		MPC2605-83		Unit	Notes
		Min	Max	Min	Max		
Cycle Time	t_{CK}	30	—	30	—	ns	
Clock High Time	t_{CKH}	12	—	12	—	ns	1
Clock Low Time	t_{CKL}	12	—	12	—	ns	1
Clock Low to Output Valid	t_A	5	9	5	9	ns	
Clock Low to Output High-Z	t_{CKZ}	0	9	0	9	ns	2
Clock Low to Output Active	t_{CKX}	0	9	0	9	ns	3, 4
Setup Times: TMS TDI	t_s t_{sd}	2	—	2	—	ns	1
Hold Times: TMS TDI	t_h t_{hd}	2	—	2	—	ns	1

NOTES: **1.** This parameter is sampled and not 100% tested.
2. TDO will High-Z from a clock low edge depending on the current state of the TAP state machine.
3. TDO is active only in the SHIFT-IR and SHIFT-DR state of the TAP state machine.
4. Transition is measured ±500 mV from steady-state voltage. This parameter is sampled and not 100% tested.

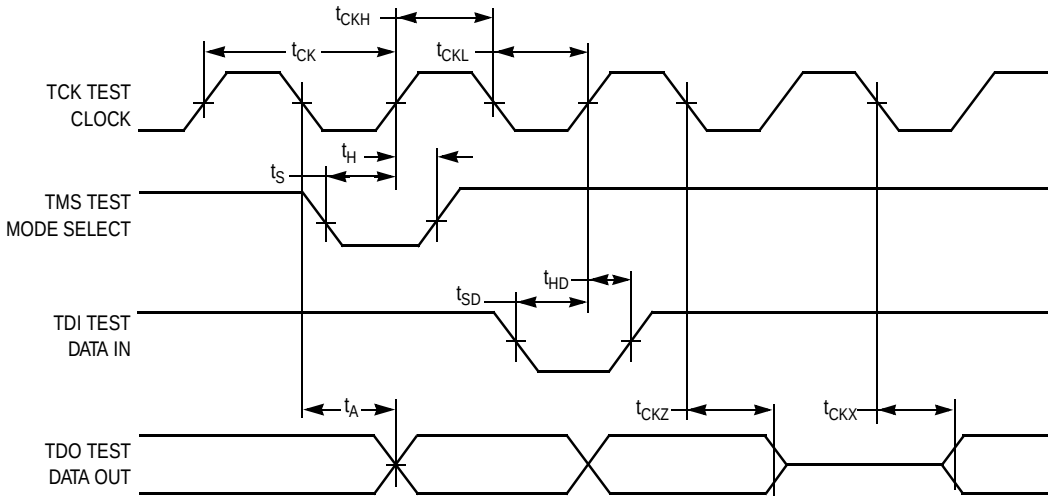


Figure 10. TAP Controller Timing

TEST ACCESS PORT DESCRIPTION

INSTRUCTION SET

A five-pin IEEE Standard 1149.1 Test Port (JTAG) is included on this device. When the TAP (Test Access Port) controller is in the SHIFT-IR state, the instruction register is placed between TDI and TDO. In this state, the desired instruction would be serially loaded through the TDI input. $\overline{\text{TRST}}$ resets the TAP controller to the test-logic reset state. The TAP instruction set for this device is as follows.

STANDARD INSTRUCTIONS

Instruction	Code (Binary)	Description
BYPASS	1111*	Bypass instruction
SAMPLE/PRELOAD	0010	Sample and/or preload instruction
EXTEST	0000	Extest instruction
HIGHZ	1001	High-Z all output pins while bypass register is between TDI and TDO
CLAMP	1100	Clamp output pins while bypass register is between TDI and TDO

*Default state at power up.

SAMPLE/PRELOAD TAP INSTRUCTION

The SAMPLE/PRELOAD TAP instruction is used to allow scanning of the boundary scan register without causing interference to the normal operation of the chip logic. The 169-bit boundary scan register contains bits for all device signal and clock pins and associated control signals. This register is accessible when the SAMPLE/PRELOAD TAP instruction is loaded into the TAP instruction register in the SHIFT-IR state. When the TAP controller is then moved to the SHIFT-DR state, the boundary scan register is placed between TDI and TDO. This scan register can then be used prior to the EXTEST instruction to preload the output pins with desired values, so that these pins will drive the desired state when the EXTEST instruction is loaded. As data is written into TDI, data also streams out of TDO, where it can be used to pre-sample the inputs and outputs.

SAMPLE/PRELOAD would also be used prior to the CLAMP instruction to preload the values on the output pins that will be driven out when the CLAMP instruction is loaded.

EXTEST TAP INSTRUCTION

The EXTEST instruction is intended to be used in conjunction with the SAMPLE/PRELOAD instruction to assist in testing board level connectivity. Normally, the SAMPLE/PRELOAD instruction would be used to preload all output pins. The EXTEST instruction would then be loaded. During EXTEST, the boundary scan register is placed between TDI and TDO in the SHIFT-DR state of the TAP controller. Once the EXTEST instruction is loaded, the TAP controller would then be moved to the run-test/idle state. In this state, one cycle of TCK would cause the preloaded data on the output pins to be driven while the values on the input pins would be sampled. Note the TCK, not the clock pin (CLK), is used as the clock input while CLK is only sampled during EXTEST. After one clock cycle of TCK, the TAP controller would then be moved to the SHIFT-DR state where the sampled values would be shifted out of TDO (and new values would be shifted in TDI). These values would normally be compared to expected values to test for board connectivity.

CLAMP TAP INSTRUCTION

The CLAMP instruction is provided to allow the state of the signals driven from the output pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. The signals driven from the output pins will not change while the CLAMP instruction is selected. EXTEST could also be used for this purpose, but CLAMP shortens the board scan path by inserting only the bypass register between TDI and TDO. To use CLAMP, the SAMPLE/PRELOAD instruction would be used first to scan in the values that will be driven on the output pins when the CLAMP instruction is active.

HIGHZ TAP INSTRUCTION

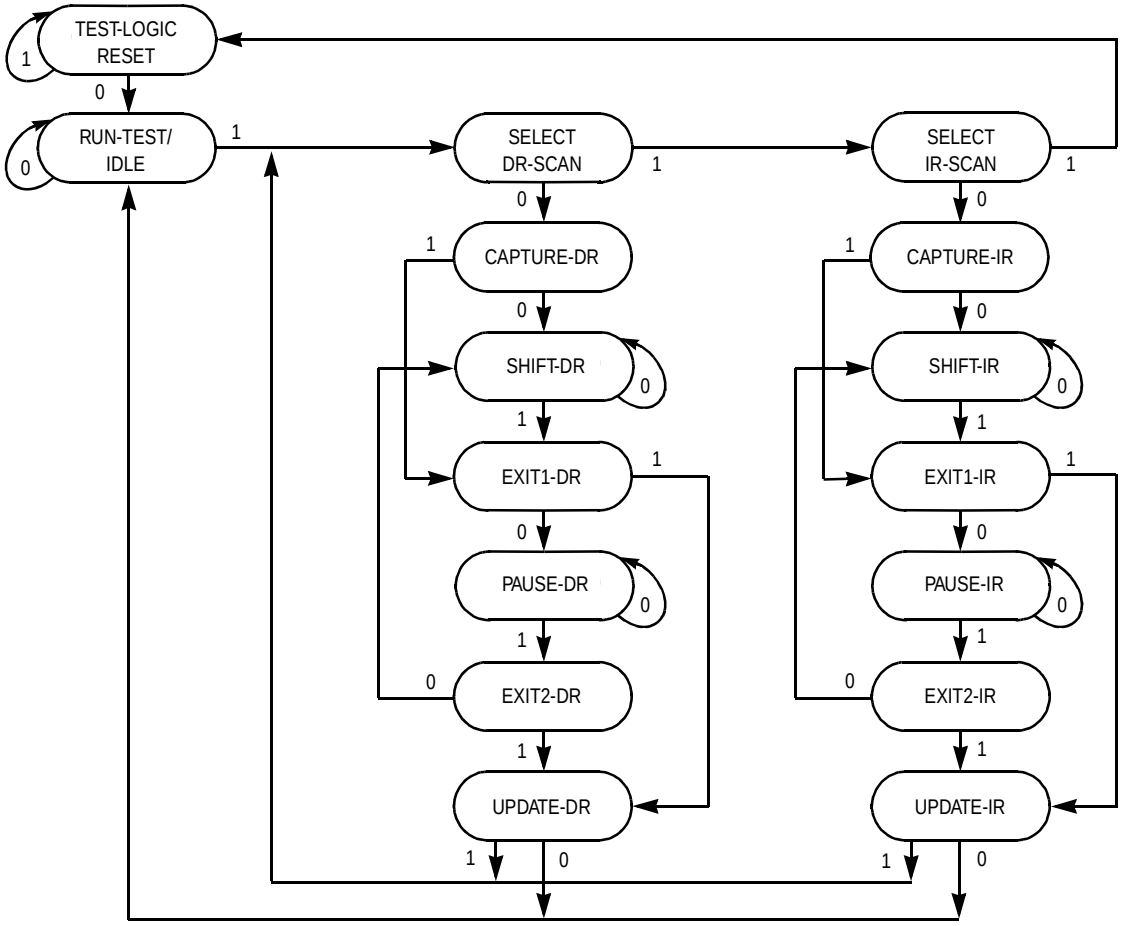
The HIGHZ instruction is provided to allow all the outputs to be placed in an inactive drive state (High-Z). During the HIGHZ instruction, the bypass register is connected between TDI and TDO.

BYPASS TAP INSTRUCTION

The BYPASS instruction is the default instruction loaded at power up. This instruction will place a single shift register between TDI and TDO during the SHIFT-DR state of the TAP controller. This allows the board level scan path to be shortened to facilitate testing of other devices in the scan path.

DISABLING THE TEST ACCESS PORT AND BOUNDARY SCAN

It is possible to use this device without utilizing the four pins used for the test access port. To circuit disable the device, TCK must be tied to V_{SS} to preclude mid-level inputs. $\overline{\text{TRST}}$ should be tied to V_{SS} to ensure proper $\overline{\text{HRESET}}$ operation. Although TDI and TMS are designed in such a way that an undriven input will produce a response equivalent to the application of a logic 1, it is still advisable to tie these inputs to V_{DD} through a 1K resistor. TDO should remain unconnected.



NOTE: The value adjacent to each state transition represents the signal present at TMS at the rising edge of TCK.

Figure 11. TAP Controller State Diagram

BOUNDARY SCAN ORDER

BIT NUMBER

The order of the boundary scan chain. Bit 0 is the closest to TDO.

BIT/PIN NAME

The name of the physical pin. For an output enable cell, this is the name of the corresponding output enable.

BIT/PIN TYPE

Input — Input only pin.
I/O — Bi-directional pin that can be put into a High-Z state.
Output — Output only pin.
Output Enable — Boundary scan cell to hold the output enable state of other I/O pads. Output enable does not correspond to a physical pin. To set an I/O to an input, the output enable must have a 1. To set an I/O to an output, the output enable must have a 0. Note that these internal output enables are active low.
Reserved — This signal is reserved and must always be a 1.

OUTPUT ENABLE

The name of the output enable cell that determines if the cell is enabled or in the High-Z state. If the pin type is input or output enable, this entry will be empty.

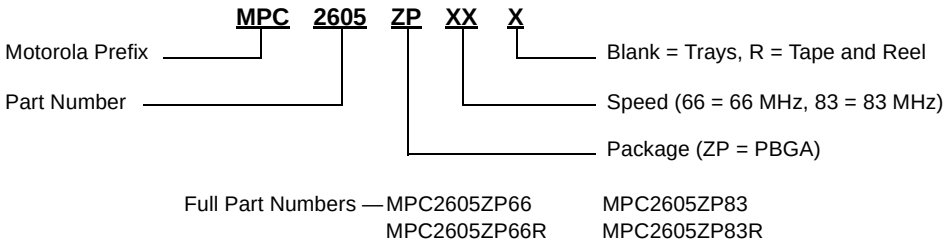
Bit No.	Bit/Pin Name	Bit/Pin Type	Output Enable
0	Reserved	Reserved	
1	DL16	I/O	DOE
2	DL17	I/O	DOE
3	DL18	I/O	DOE
4	DL19	I/O	DOE
5	DL20	I/O	DOE
6	DL21	I/O	DOE
7	DL22	I/O	DOE
8	DL23	I/O	DOE
9	DP6	I/O	DOE
10	DL24	I/O	DOE
11	DL25	I/O	DOE
12	DL26	I/O	DOE
13	DL27	I/O	DOE
14	DL28	I/O	DOE
15	DL29	I/O	DOE
16	DL30	I/O	DOE
17	DL31	I/O	DOE
18	DP7	I/O	DOE
19	DH24	I/O	DOE
20	DH25	I/O	DOE
21	DH26	I/O	DOE
22	DH27	I/O	DOE
23	DH28	I/O	DOE
24	DH29	I/O	DOE
25	DH30	I/O	DOE
26	DH31	I/O	DOE
27	DP3	I/O	DOE
28	DH16	I/O	DOE
29	DH17	I/O	DOE
30	DH18	I/O	DOE
31	DH19	I/O	DOE
32	DH20	I/O	DOE
33	DH21	I/O	DOE
34	DH22	I/O	DOE
35	DH23	I/O	DOE
36	DP2	I/O	DOE
37	L2 BG	Input	
38	L2 MISS INH	Input	
39	ABB	I/O	ABBOE
40	CPU3 DBG	Input	
41	CPU3 BG	Input	
42	CPU3 BR	Input	
43	CPU2 DBG	Input	
44	CPU2 BG	Input	
45	CPU2 BR	Input	

Bit No.	Bit/Pin Name	Bit/Pin Type	Output Enable
46	FDN	I/O	FDNOE
47	L2 DBG	Input	
48	L2 BR	I/O	L2BROE
49	TA	I/O	TAOE
50	L2 CLAIM	Output	L2CLAIMOE
51	CPU DBG	Input	
52	AACK	I/O	AACKOE
53	CI	I/O	AOE
54	ARTRY	I/O	ARTRYOE
55	WT	I/O	AOE
56	CPU BR	Input	
57	TEA	Input	
58	PWRDN	Input	
59	DBB	I/O	DBBOE
60	HRESET	Input	
61	TBST	I/O	AOE
62	TT0	I/O	AOE
63	TS	I/O	AOE
64	TT1	I/O	AOE
65	TT2	I/O	AOE
66	TT4	I/O	AOE
67	TT3	I/O	AOE
68	CPU BG	Input	
69	SRESET	Input	
70	L2 TAG CLR	Input	
71	L2 UPDATE INH	Input	
72	CPU4 BG	Input	
73	CPU4 DBG	Input	
74	CPU4 BR	Input	
75	CFG0	Input	
76	CFG2	Input	
77	CFG1	Input	
79	DH8	I/O	DOE
79	DH9	I/O	DOE
80	DH10	I/O	DOE
81	DH11	I/O	DOE

Bit No.	Bit/Pin Name	Bit/Pin Type	Output Enable
82	DH12	I/O	DOE
83	DH13	I/O	DOE
84	DH14	I/O	DOE
85	DH15	I/O	DOE
86	DP1	I/O	DOE
87	DH0	I/O	DOE
88	DH1	I/O	DOE
89	DH2	I/O	DOE
90	DH3	I/O	DOE
91	DH4	I/O	DOE
92	DH5	I/O	DOE
93	DH6	I/O	DOE
94	DH7	I/O	DOE
95	DP0	I/O	DOE
96	DL0	I/O	DOE
97	DL1	I/O	DOE
98	DL2	I/O	DOE
99	DL3	I/O	DOE
100	DL4	I/O	DOE
101	DL5	I/O	DOE
102	DL6	I/O	DOE
103	DL7	I/O	DOE
104	DP4	I/O	DOE
105	DL8	I/O	DOE
106	DL9	I/O	DOE
107	DL10	I/O	DOE
108	DL11	I/O	DOE
109	DL12	I/O	DOE
110	DL13	I/O	DOE
111	DL14	I/O	DOE
112	DL15	I/O	DOE
113	DP5	I/O	DOE
114	A0	I/O	AOE
115	A1	I/O	AOE
116	A2	I/O	AOE
117	A3	I/O	AOE

Bit No.	Bit/Pin Name	Bit/Pin Type	Output Enable
118	A4	I/O	AOE
119	A5	I/O	AOE
120	A6	I/O	AOE
121	A7	I/O	AOE
122	A8	I/O	AOE
123	A9	I/O	AOE
124	A10	I/O	AOE
125	A11	I/O	AOE
126	A12	I/O	AOE
127	A31	I/O	AOE
128	A30	I/O	AOE
129	A29	I/O	AOE
130	A28	I/O	AOE
131	A27	I/O	AOE
132	A26	I/O	AOE
133	A25	I/O	AOE
134	A24	I/O	AOE
135	A23	I/O	AOE
136	A22	I/O	AOE
137	A21	I/O	AOE
138	A20	I/O	AOE
139	A19	I/O	AOE
140	A18	I/O	AOE
141	A17	I/O	AOE
142	A16	I/O	AOE
143	A15	I/O	AOE
144	A14	I/O	AOE
145	A13	I/O	AOE
146	TSIZ2	I/O	AOE
147	TSIZ0	I/O	AOE
148	TSIZ1	I/O	AOE
149	GBL	Output	AOE
150	CFG3	Input	
151	L2 CI	Input	
152	L2 FLUSH	Input	
153	AP0	I/O	AOE
154	AP1	I/O	AOE
155	AP2	I/O	AOE
156	AP3	I/O	AOE
157	APE	Output	APEOE
158	TAOE	Output Enable	
159	L2CLAIMOE	Output Enable	
160	L2BROE	Output Enable	
161	FDNOE	Output Enable	
162	DBBOE	Output Enable	
163	DOE	Output Enable	
164	ARTRYOE	Output Enable	
165	APEOE	Output Enable	
166	ABBOE	Output Enable	
167	AACKOE	Output Enable	
168	AOE	Output Enable	

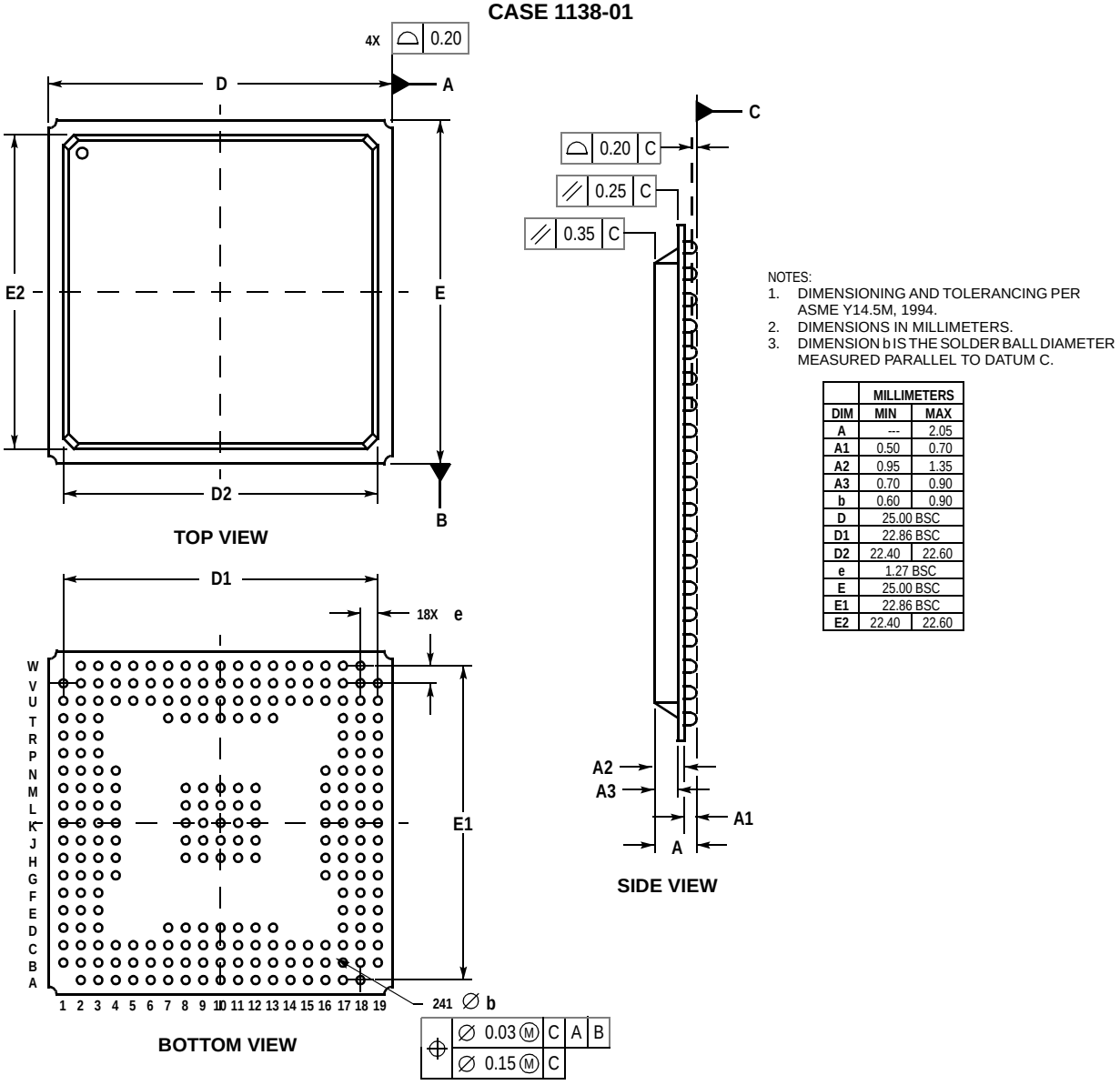
ORDERING INFORMATION
(Order by Full Part Number)



Full Part Numbers — MPC2605ZP66 MPC2605ZP83
MPC2605ZP66R MPC2605ZP83R

PACKAGE DIMENSIONS

ZP PACKAGE
PBGA
CASE 1138-01



NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS IN MILLIMETERS.
3. DIMENSION b IS THE SOLDER BALL DIAMETER MEASURED PARALLEL TO DATUM C.

	MILLIMETERS	
DIM	MIN	MAX
A	---	2.05
A1	0.50	0.70
A2	0.95	1.35
A3	0.70	0.90
b	0.60	0.90
D	25.00 BSC	
D1	22.86 BSC	
D2	22.40	22.60
e	1.27 BSC	
E	25.00 BSC	
E1	22.86 BSC	
E2	22.40	22.60

THIS PAGE INTENTIONALLY LEFT BLANK

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140
(800) 441-2447

JAPAN:

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre, 2 Dai King Street
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

(800) 521-6274

HOME PAGE:

www.motorola.com/semiconductors

MPC2605/D