Memory Controller IP Core

Author: Rudolf Usselmann
rudi@asics.ws

Rev. 1.7
January 21, 2002

Revision History

Rev.	Date	Author	Description
0.1	7/4/01	Rudolf Usselmann	First Draft
0.2	9/4/01	RU	- Added timing diagrams and descriptions. - Modified CSC and TMS registers. - Added IOs for SSRAMS and external masters.
0.3	19/4/01	RU	- Updated core architecture overview drawing. - Fixed several minor syntax errors. - Renamed CSC_MASK to BA_MASK register. - Added Power Down Mode section. - Added External Bus Master Section.
0.4	29/4/01	RU	- Added Bandwidth Section. - Added Table to Power-On Configuration Section. - Added POC register Output - Added Appendix C: IO Buffers - Added Appendix D: Wiring Examples. - Added SSRAM Timing diagrams. - Filled in Memory Organization Section. - Filed in Synchronous Chip Select Devices Section. - Filled in Appendix A: HW Configuration.
1.0	13/5/01	RU	- Added dynamic bus sizing section - Moved Refresh Prescaler and Refresh Early from HW Configuration section in to CSR register. - Added RMW cycle section. - Added Error Signaling Section.
1.1	30/5/01	RU	- Added Section 2.5 Clocks. - Fixed IO names to match the core. - Added WISHBONE Clock and Reset signals.
1.2	25/6/01	RU	- Cleaned up document for references to SRAM and SSRAM. - Add TSM Register specification for SSRAM.
1.3	10/8/01	RU	- Changed IO names to be more clear. - Uniquifyed define names to be core specific. - Removed “Refresh Early” configuration
1.4	27/11/01	RU	- Added Appendix E, “Clocks and Reset”
1.5	28/11/01	RU	- Filled-in details in Chapter 2, “Architecture”
1.6	6/12/01	RU	- Various minor additions and clarifications
1.7	18/1/01	RU	- Added note to point out potential external bus master problem - Added note on EB_ERR_O behavior

1 Introduction

This is a universal Memory Controller core. It supports a variety of memory devices, flexible timing and predefined system startup from a Flash or ROM memory.

Some of the main features are:

1. SDRAM, SSRAM, FLASH, ROM and many other devices supported
2. 8 Chip selects, each uniquely programmable
3. Flexible timing to accommodate a variety of memory devices
4. Burst transfers and burst termination
5. Supports RMW cycles
6. Performance optimization by leaving active rows open
7. Default boot sequence support
8. Dynamic bus sizing for reading from Async. devices
9. Byte parity Generation and Checking
10. Multi Master memory bus support
11. Industry standard WISHBONE SoC host interface
12. Up to 8 * 64 Mbyte memory size
13. Supports Power Down Mode

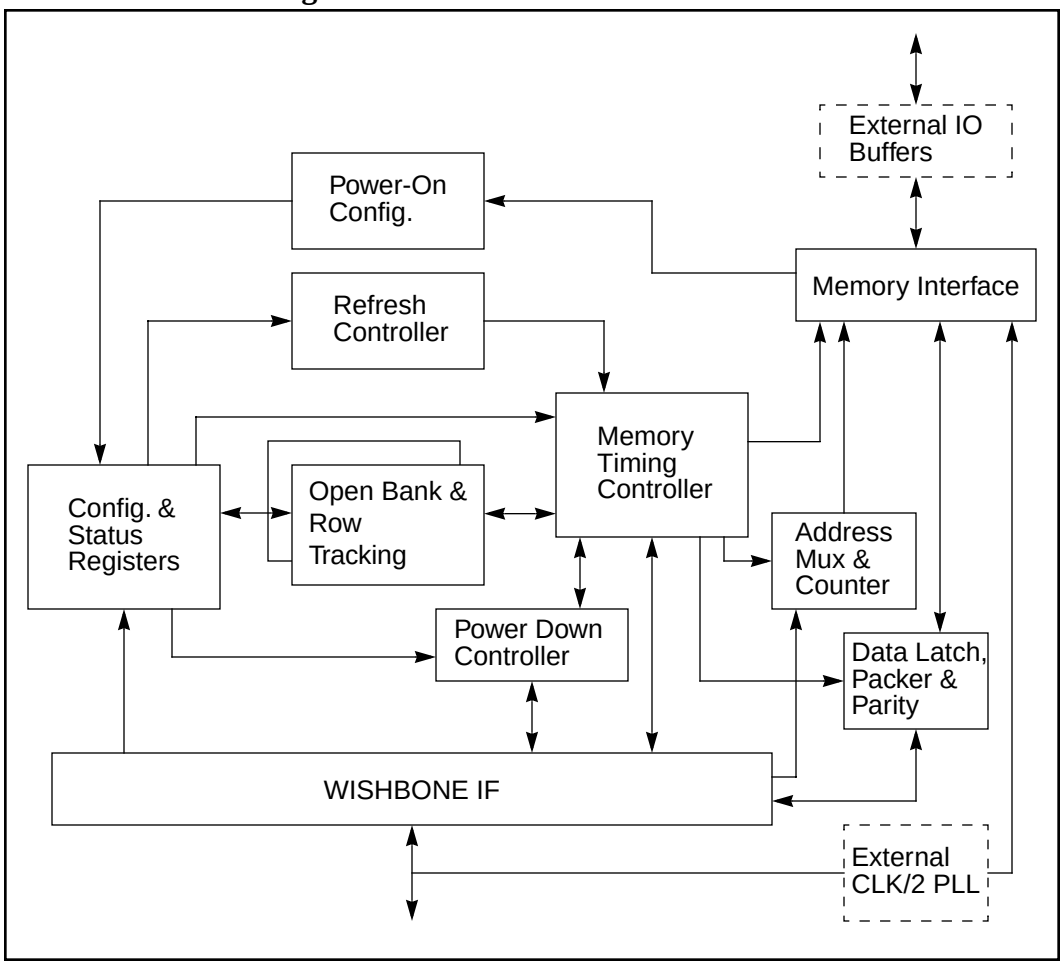
(This page intentionally left blank)

2

Architecture

Below figure illustrates the overall architecture of the core.

Figure 1: Core Architecture Overview



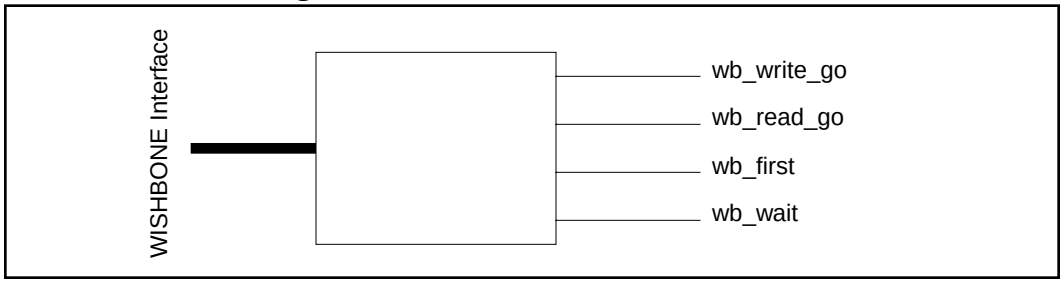
2.1. WISHBONE Interface

The Memory Controller core includes a WISHBONE host interface.This interface is WISHBONE SoC bus specification Rev. B compliant. This implementation implements a 32 bit bus width and does not support other bus widths.

The memory controller slightly deviates from the WISHBONE specification when performing bursts. Please see section 3.5. “Memory Burst Cycles” on page 24 for more details.

The WISHBONE interface block performs very simple decoding of the wishbone signals.

Figure 2: WISHBONE Interface Block

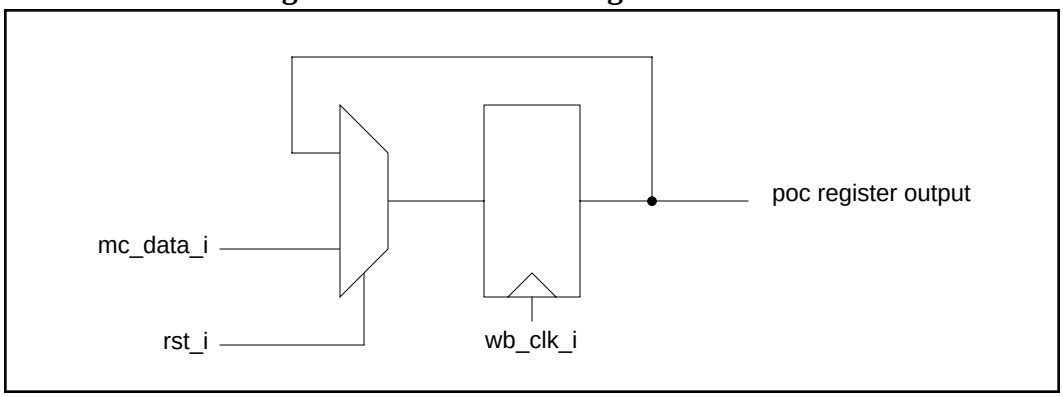


wb_write_go indicates a write cycle, *wb_read_go* indicates a read cycle, *wb_first* indicates the first transfer of a possible burst, and *wb_wait* indicates wait state insertion on the wishbone bus.

2.2. Power-On Configuration

This block is a part of the register file. The Power-On Configuration block, latches the value of the Memory Data Bus during reset.

Figure 3: Power-On Configuration Block

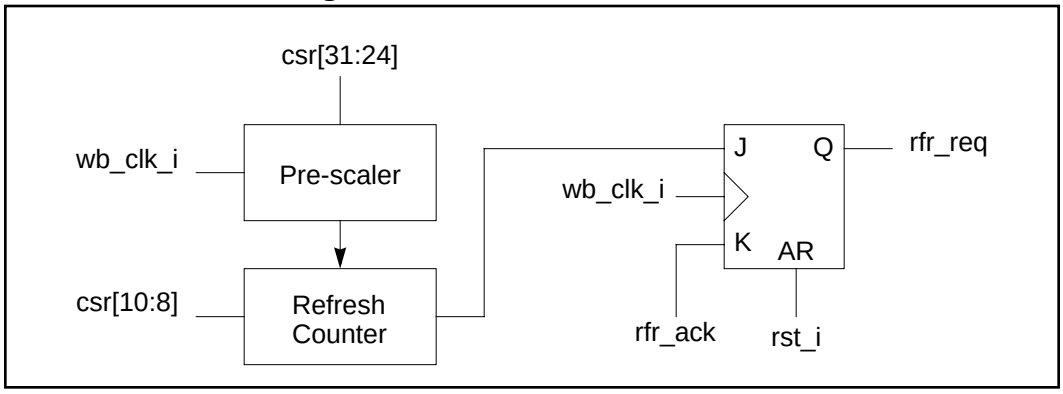


The value read, determines initial configuration of the Memory Controller (bits 3:0), and provides additional configuration bit for the system (bits 31:4).

2.3. Refresh Controller

This block is responsible for generating refresh cycle requests for the attached SDRAMS. All SDRAMs are refreshed at the same time. When SDRAMs with different refresh cycles are configured, the shortest refresh interval is used for all SDRAMs.

Figure 4: Refresh Controller Block

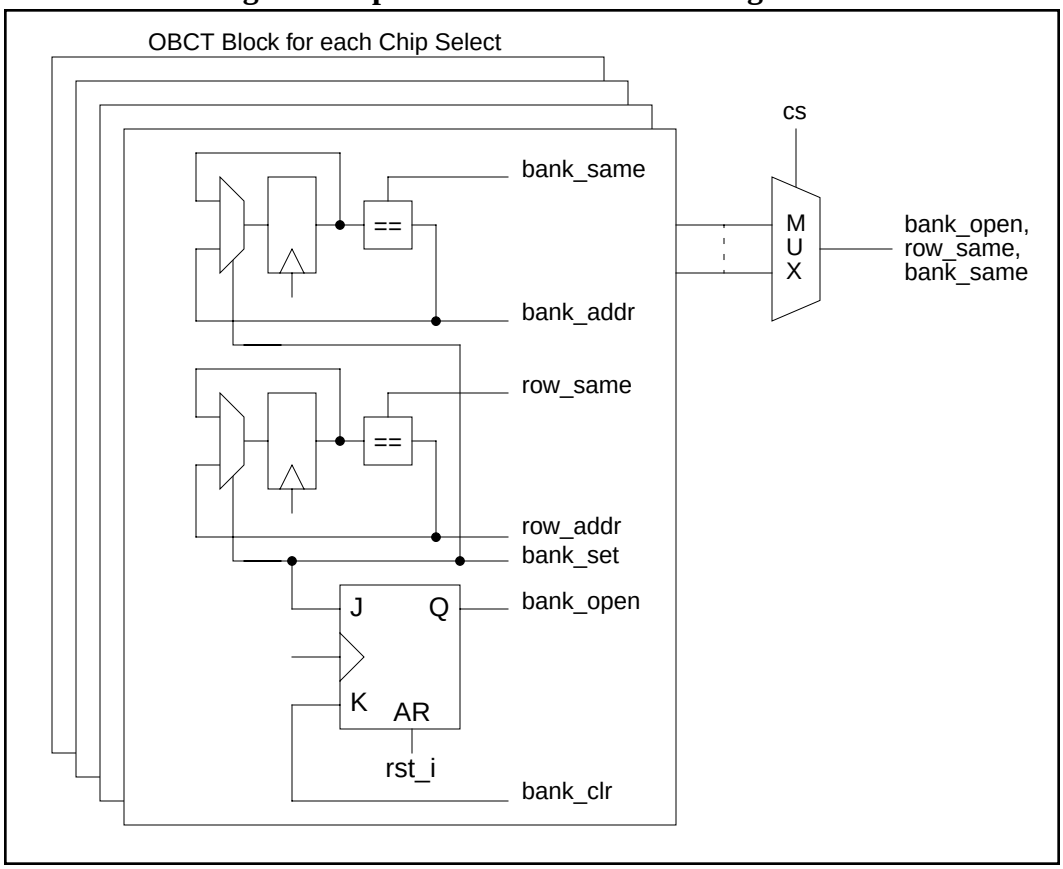


The prescaler divides the system clock to generate a 0.488 uS clock. This clock drives a refresh counter that generates a refresh request on pre-programed intervals. The refresh request remains asserted until the refresh has been performed and refresh acknowledgement asserted.

2.4. Open Bank & Row Tracking

This block remembers which bank and which row within a bank is already open. This feature allows for very fast access to already open rows within a bank and row. This block is only used when the *kro* bit for a chip select is set.

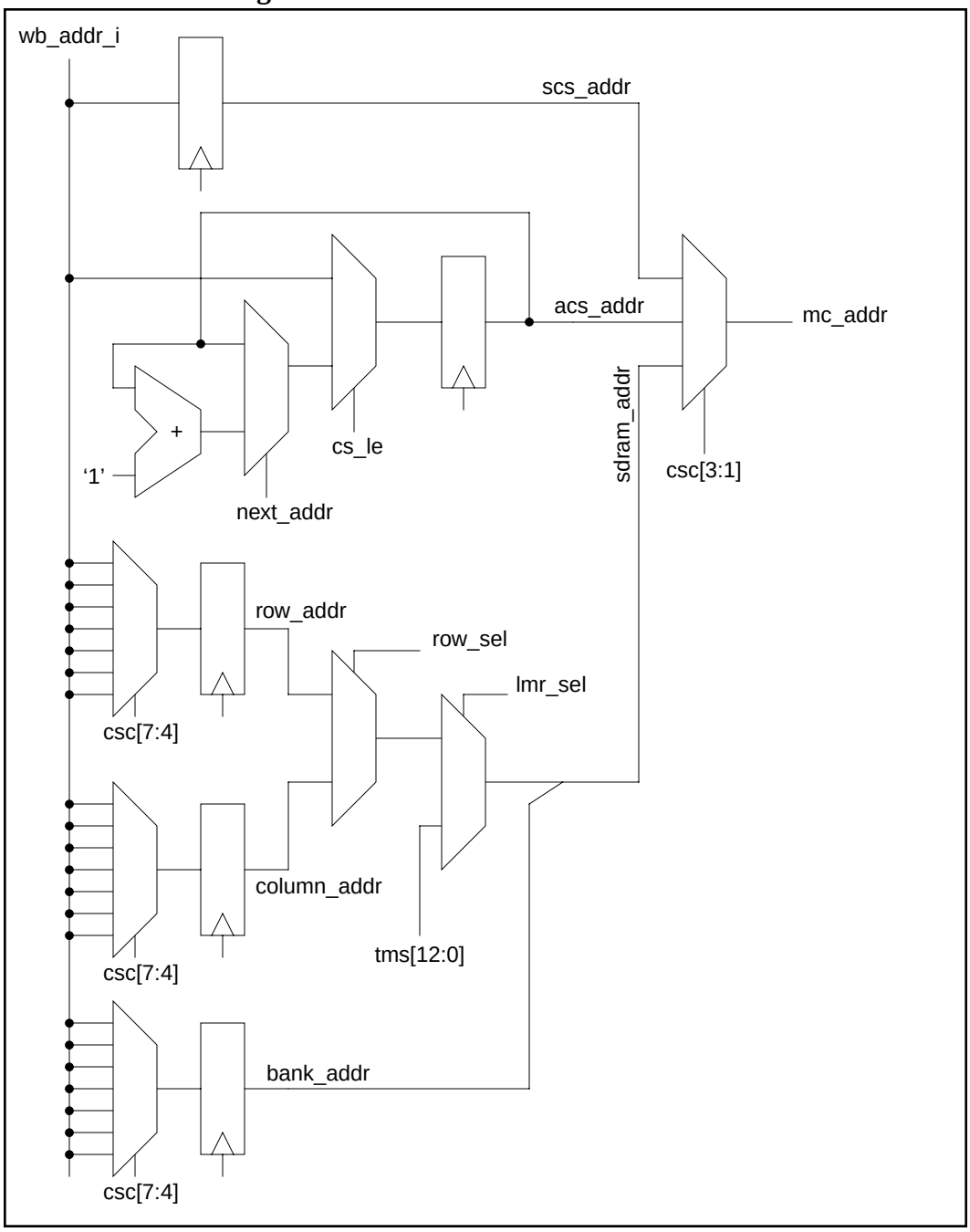
Figure 5: Open Bank and Row Tracking Block



2.5. Address Mux & Counter

This block is responsible for generation of proper addresses. The address generation is divided in to three different blocks: Address for Synchronous Chip Select Devices, address for synchronous chip select devices and address for SDRAMs. Based on the memory type specified in the CSC register the proper address is selected.

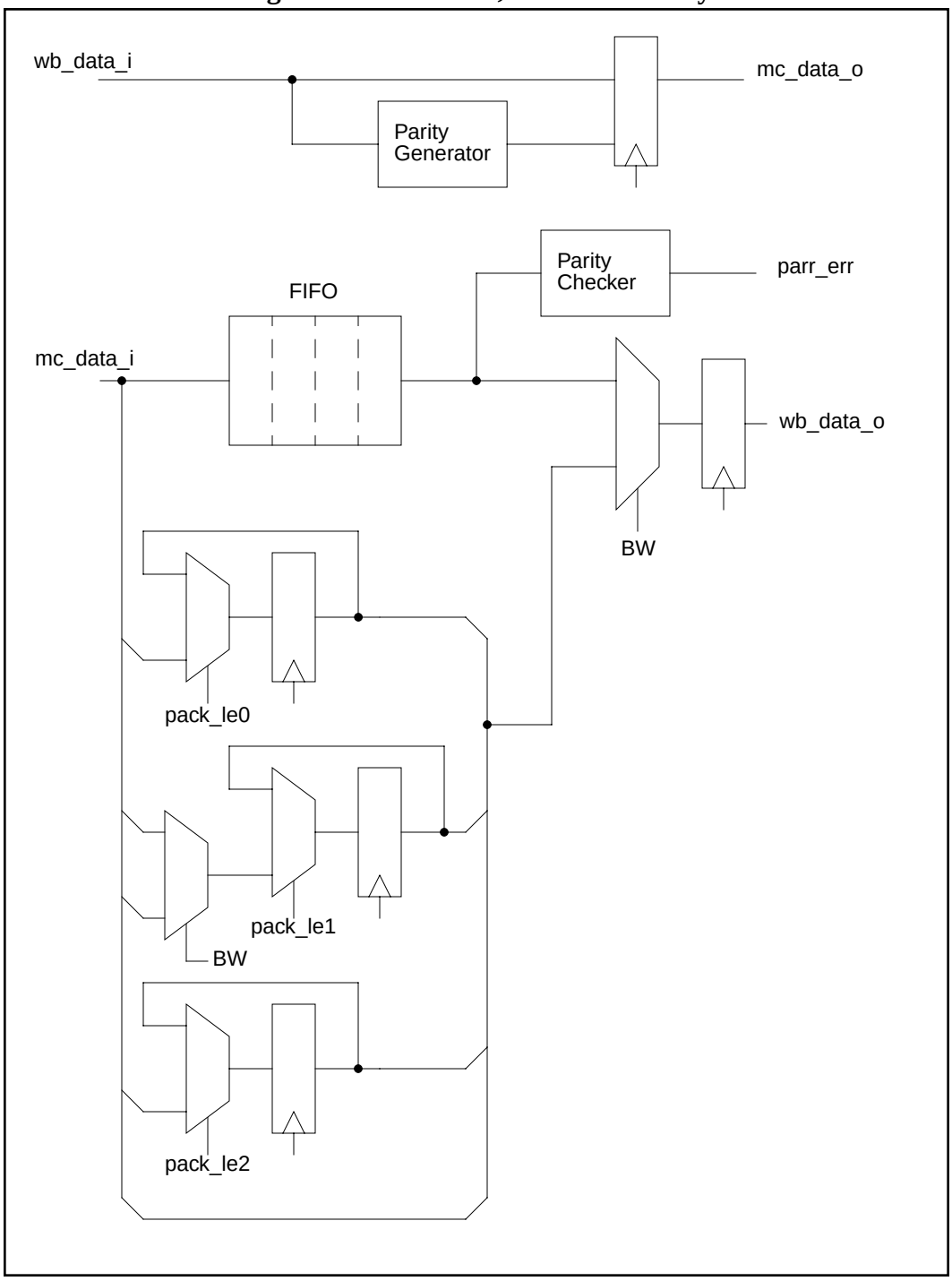
Figure 6: Address Mux & Counter Block



2.6. Data Latch, Packer & Parity

This block is responsible for the data bus connections between the Memory bus and the WISHBONE bus. Data That goes out to the Memory Bus, is simply latched at appropriate times. Data Read from the Memory bus, is either passed through a latch to the WISHBONE bus, or goes through a data packet first. The data packer assembles a 32 bit word from 8 or 16 bit wide devices on the memory bus.

Figure 7: Data Latch, Packer & Parity



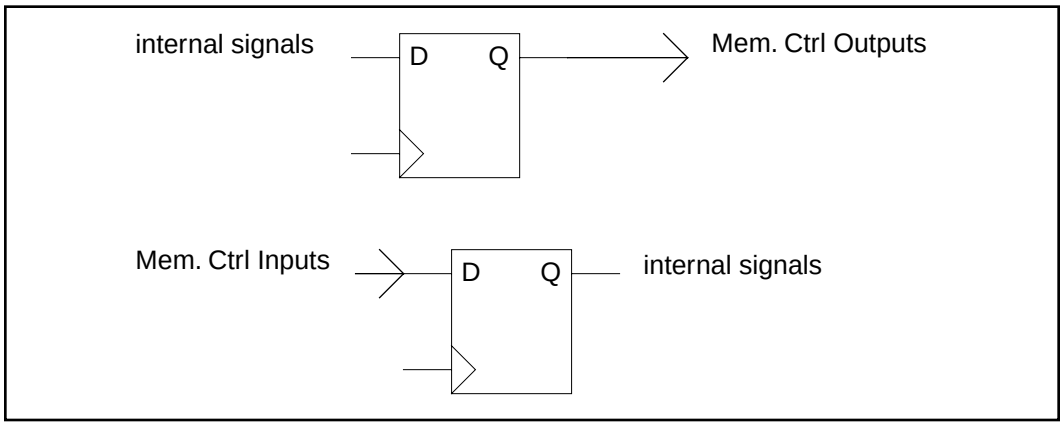
2.7. Memory Timing Controller

This block is responsible for memory timing and control. It performs the appropriate cycles to access various memories, as defined in the memory Controller configuration registers.

This block also generates control signals for most of the other block.

2.8. Memory Interface

The Memory Interface block provides simple synchronization for IOs. All outputs and inputs are registered at the rising edge of the Memory Clock.



2.9. Clocks

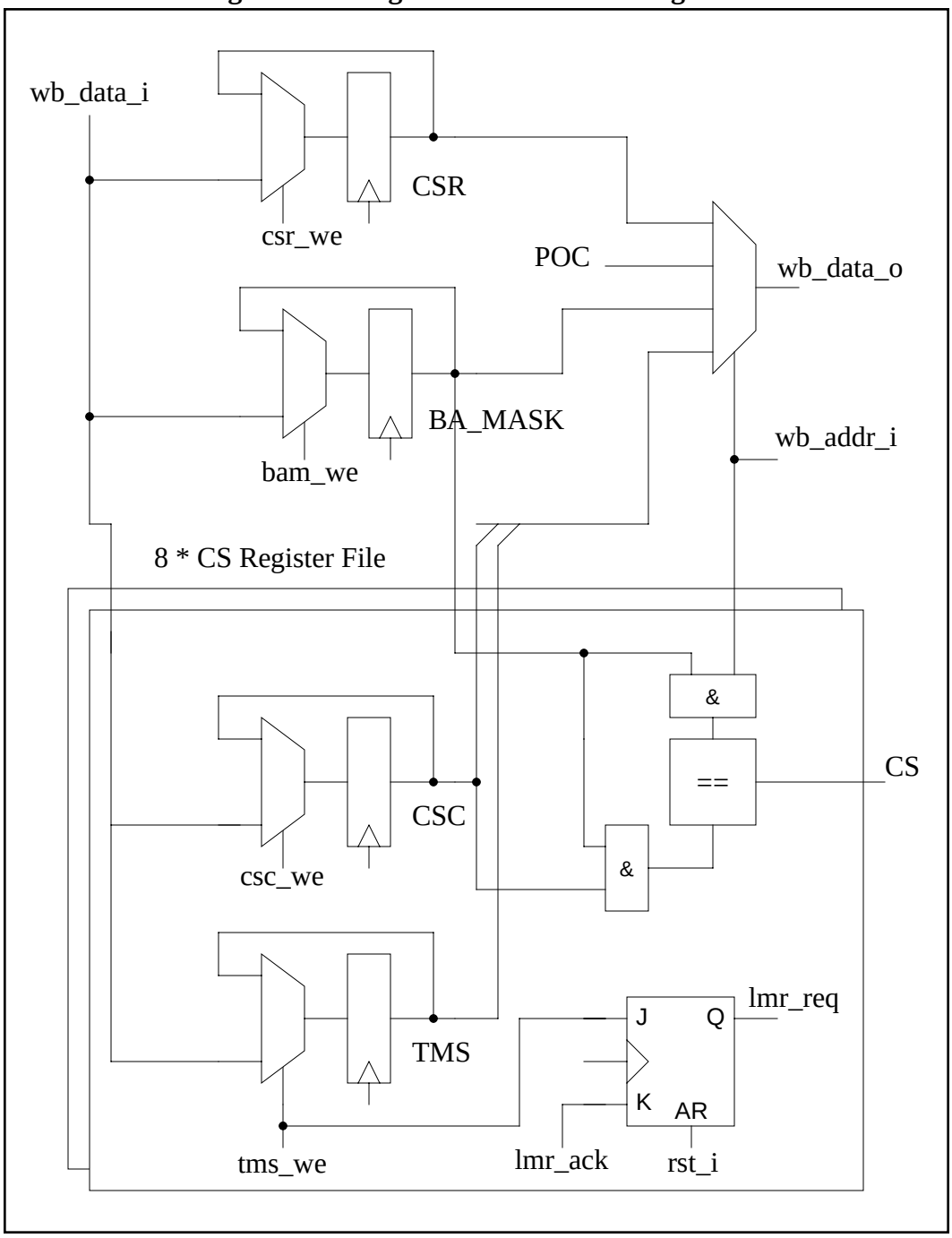
The Memory Controller utilizes two clocks: 1) the main wishbone clock; 2) the memory clock. Both clocks are used by the core internally. The Memory clock is also used by external memory devices.

To avoid synchronization between the clocks and for optimal operations, the memory clock must be derived from the WISHBONE clock by dividing the WISHBONE clock by two and phase synchronizing it to the WISHBONE clock. See Appendix E “Clocks and Reset” on page 51 for more information.

2.10. Configuration & Status Registers

This block holds all the registers in the Memory Controller. It also generate requests to update the Mode Register within SDRAMs whenever the value of the TMS register changes. This block also decodes the actual Chip Select Signals.

Figure 8: Configuration and Status Registers



3

Operation

3.1. Chip Selects

Each chip select is individually fully programmable. The user can chose the address space and memory configuration that a chip select controls.

3.1.1. Address Space

The *BA_MASK* register determines the maximum size that is available to any chip select. This 8 bit register hold a mask that is applied to address bits 21 through 28. This results in a minimum space of 4Mbytes and maximum of 128Mbytes for each chip select.

3.1.2. Memory Device Configuration

Each chip select can be programmed to support a different type of memory and timing parameters. An application may chose to have different size of memories and different speeds of memories attached to each chip select.

3.2. Memory Organization

This section describes the different memory organizations that are supported by the Memory Controller core.

3.2.1. SDRAM Memory Organization

The SDRAM memory may be organized by choosing a variety of different SDRAM devices. All SDRAMS connected to the same chip select, must be of the

same type. The table below summarizes the different SDRAM devices that are supported.

Table 1: Supported SDRAM Organizations

	Bus Width	Number of Chips	64 MBit	128 MBit	256 MBit
Refresh (Interval)	All	-	64mS 4096 cycles (15.625 uS)	64mS 4096 cycles (15.625 uS)	64mS 8192 cycles (7.813 uS)
Bank Addresses	All	-	2	2	2
Total Addr. Lines	All	-	12	12	13
Row/Column Address	8	4	RA 0-11 CA 0-8	RA 0-11 CA 0-9	RA 0-12 CA 0-9
Row/Column Address	16	2	RA 0-11 CA 0-7	RA 0-11 CA 0-8	RA 0-12 CA 0-8
Row/Column Address	32	1	RA 0-10 CA 0-7	RA 0-11 CA 0-7	RA 0-12 CA 0-7
Maximum Memory Size	8	4	32M Bytes	64M Bytes	128M Bytes
Maximum Memory Size	16	2	16M Bytes	32M Bytes	64M Bytes
Maximum Memory Size	32	1	8M Bytes	16M Bytes	32M Bytes[a]

a. 256Mbit by 32 bit devices might not be available at this time.

3.2.2. SSRAM Devices Organization

SSRAMs may be organized as 8, 16 or 32 bit devices. The Memory controller does not distinguish between the different SSRAM organizations. The BW field in the CSC register should always be set to 32. The Memory controller will not perform dynamic bus sizing on accesses to SSRAM.

3.2.3. Asynchronous Chip Select Devices Organization

Asynchronous Chip Select Devices might be organized in three different ways: 8 bit, 16 bit and 32 bit wide data bus. The memory controller will perform multiple reads to assemble one 32 bit word, when reading from Asynchronous Chip Select Devices with a bus width less than 32 bit. When using multiple Asynchronous

Chip Select Devices to assemble a 32 bit word (e.g. 4 devices of 8 bit width), the BW field in the chip selects CSC register must be set to 32 bit bus widths.

The actual size of the Asynchronous Chip Select Devices is irrelevant to the memory controller core. The only restriction is the number of address lines that are provided.

3.2.4. Synchronous Chip Select Devices Organization

Synchronous Chip Select Devices may be organized as 8, 16 or 32 bit devices. The Memory controller does not distinguish between the different Synchronous Chip Select Devices organizations. The BW field in the CSC register should always be set to 32. The Memory controller will not perform dynamic bus sizing on accesses to Synchronous Chip Select Devices.

3.3. Memory Timing Configuration

The memory timing is defined by the Timing Select Register (TMSn) for each chip select. Depending on the Memory Type, this register has a different meaning.

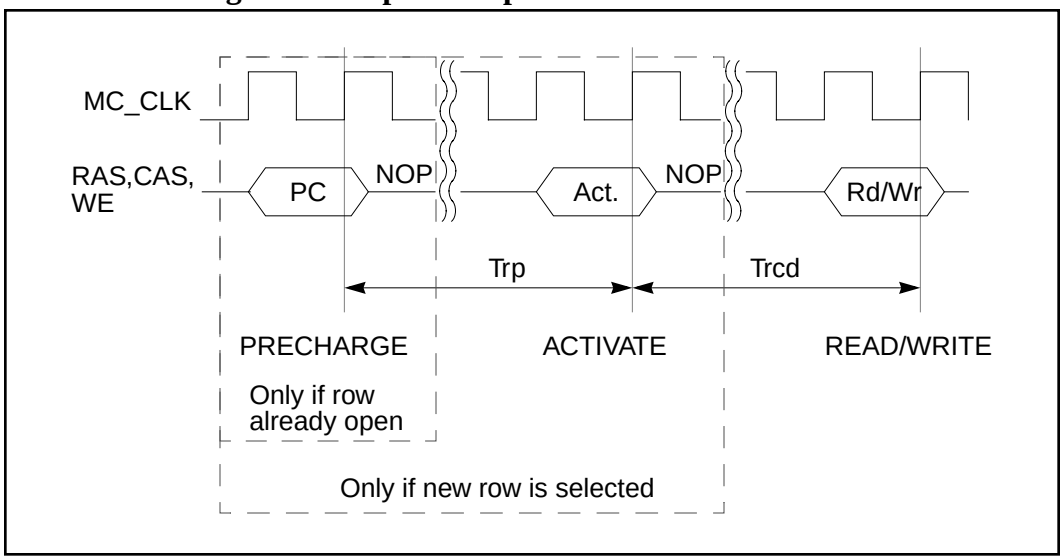
3.3.1. SDRAM Timing Configuration

The SDRAM may be operated in two modes: 1) Keeping the current column open; 2) Closing the column after the Read or Write command.

Both modes have advantages and disadvantages, and it will depend on the overall architecture which mode may be best for a given application.

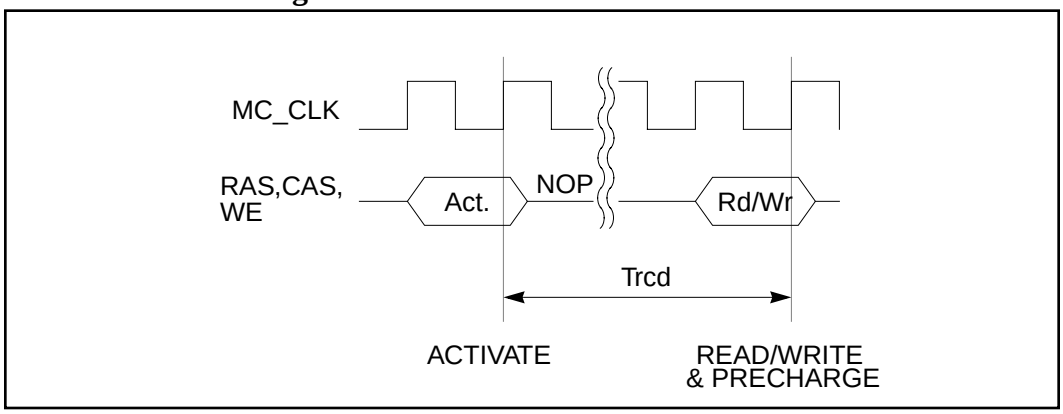
The first mode, is to keep a row open.In this case, after the initial Active command, subsequent accesses to the same row, do not have to go through costly activation cycles. The disadvantage is that a access to a different row, must execute additional Precharge cycles. If a system performs linear access that will most likely hit the same row more than once, this mode will provide better overall performance. The figure below illustrates such accesses.

Figure 9: Keep Row Open Initial SDRAM Access



The second mode, is to close the row immediately after the Read or Write operation. Here, each new access has to execute the Activate command before performing the Read or Write operation. This will help systems that are known to fetch data that is more than one column apart, on every memory access. The figure below illustrates such accesses.

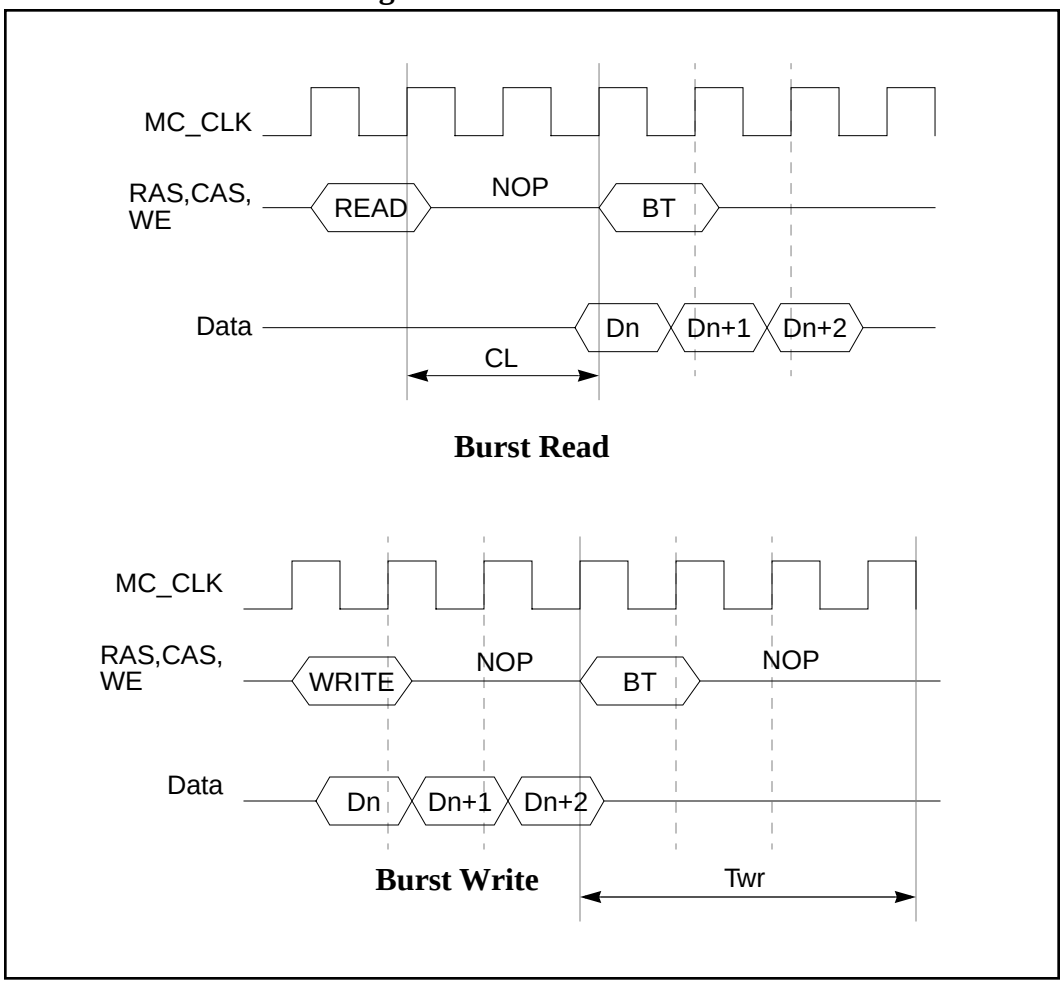
Figure 10: Close Row SDRAM accesses



Either of the above modes supports single word and burst transfers. Depending of the setting in the TMSn register, a 1, 4, 8 word or full page burst may be used.

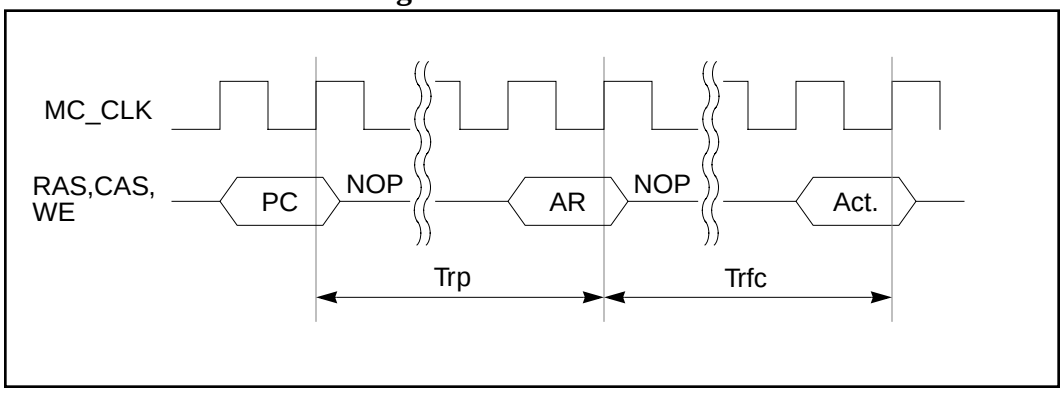
When a transfer is smaller than the programed burst size, a Burst Terminate (BT) command is issued. Below figure illustrate Read and Write burst transfers.

Figure 11: Burst Transfers



The memory controller will also generate Auto Refresh cycles. The frequency of the Auto Refresh counter, is determined by the Trf field in the TMS register.

Figure 12: Auto Refresh



3.3.1.1. TMSn Register configuration for SDRAMs

In SDRAM mode, writes to the TMSn register initiate a write to the SDRAM Mode Register. The Memory Controller Core will interpret the programed values for CAS latency and burst size as well, and perform the proper operations. All delay values are in the form of: ***(desired_delay ÷ mem_clock_period) - 2***. For example, a 60nS delay is desired. The actual value for the TMS register would be: ***(60nS ÷ 10nS) - 2 = 4***

Table 2: TMSn configuration for SDRAMs[a]

Bit #	Access	Description
31:28	RW	RESERVED
27:24	RW	Trfc 4 bit auto refresh period counter value
23:20	RW	Trp 4 bit pre charge period counter value
19:17	RW	Trcd 3 bit Active to Read/Write counter value
16:15	RW	Twr 2 bit Write completion counter value
14:10	RW	RESERVED
9	RW	Write Burst Length 0 - Programmed burst Length 1 - Single Location Access
8:7	RW	Operation Mode 0 - Normal Operation 1-3 - RESERVED
6:4	RW	CL CAS Latency 0 - RESERVED 1 - RESERVED 2 - 2 Cycles 3 - 3 Cycles 4-7 - RESERVED
3	RW	BT Burst Type 0 - Sequential 1 - Interleaved

Table 2: TMSn configuration for SDRAMs[a]

Bit #	Access	Description
2:0	RW	BL Burst Length 0 - 1 1 - 2 2 - 4 3 - 8 4-6 - RESERVED 7 - Full Page[b]

a. For a detailed definition of bits 14:0, please check the SDRAM manufacturers data sheet.
b. Full Page Bursts are not supported by all SDRAMs.

The table below illustrates sample settings for various SDRAM speeds and bus frequencies. Actual values may vary. Please check the SDRAM data sheet for details.

Table 3: TMS Register Example Settings for SDRAMs

Wishbone bus Freq.	SDRAM type	CL	Twr	Trcd	Trp	Trfc
200 Mhz	PC100	3	2	2	2	7
200 Mhz	PC133	2	2	2	2	7
266 Mhz	PC 133	3	2	2	2	9

3.3.2. SSRAM Timing Configuration

Below figures illustrate the SSRAM read and write cycles. Almost all of the SSRAM parameters are in respect to the clock and are not configurable. This Memory Controller supports standard SyncBurst, Pipelined SSRAMs with Double Cycle Deselect.

Figure 13: SSRAM Read Cycle

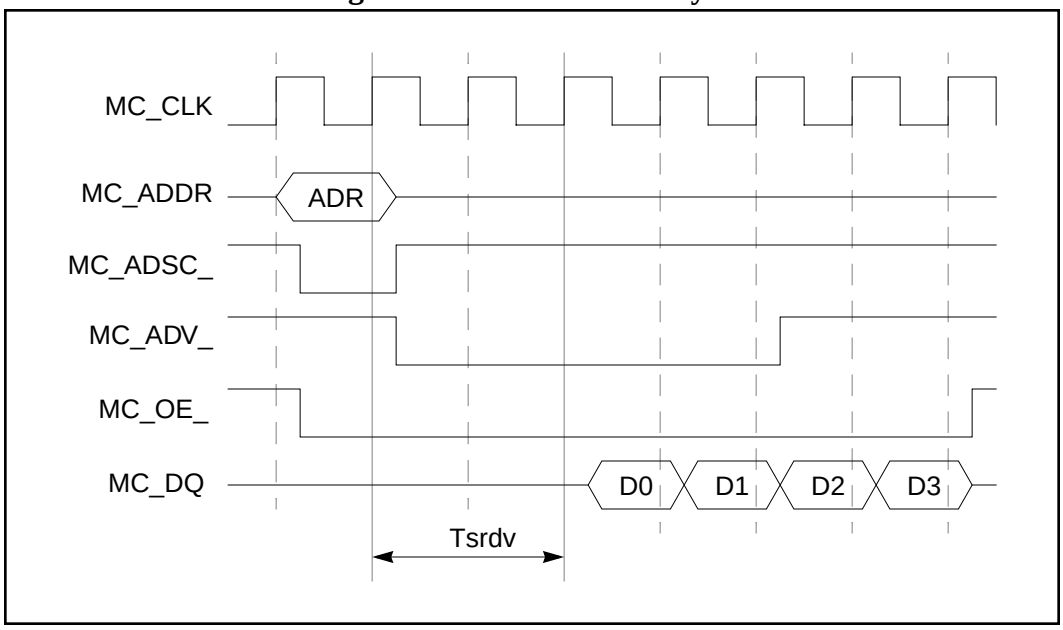
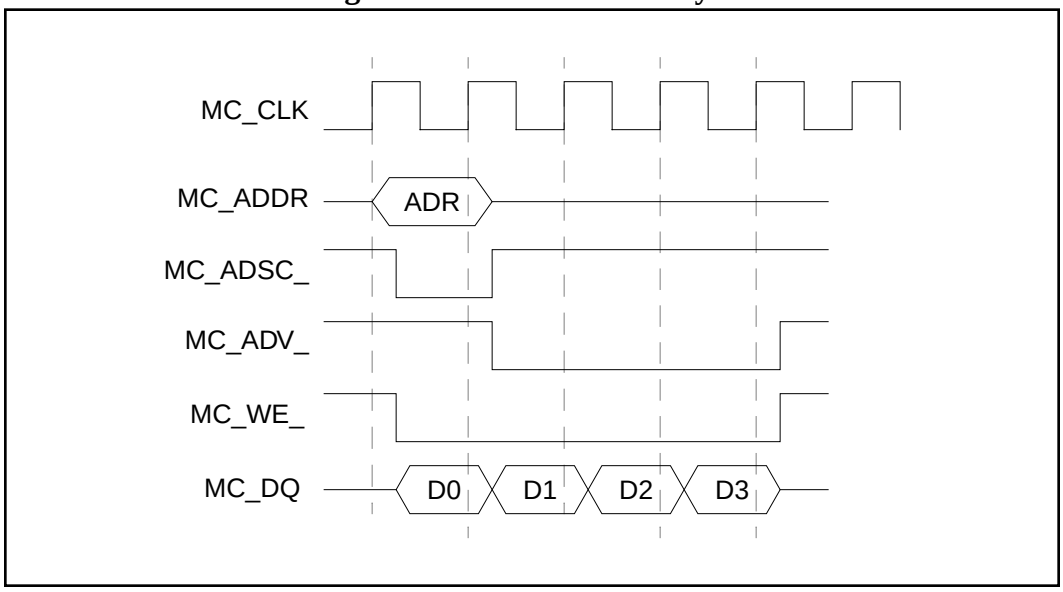


Figure 14: SSRAM Write Cycle



3.3.2.1. TMSn Register Configuration for SSRAM Devices

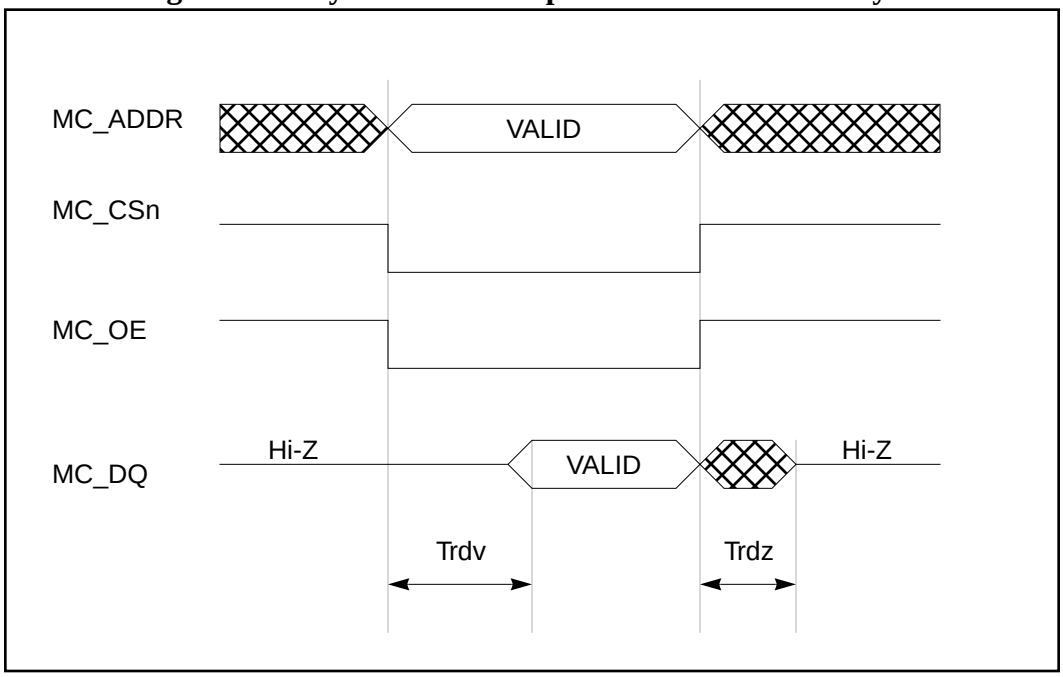
For SSRAM Devices, the TMSn register has no meaning. All timing values are fixed as illustrated in the above timing diagrams. Tsrdv is fixed for 2 clock cycles.

3.3.3. Asynchronous Chip Select Devices Timing Configuration

The Asynchronous Chip Select Devices timing is divided in to two areas: Read and Write. The Memory Controller core supports very basic access to the Asyn-

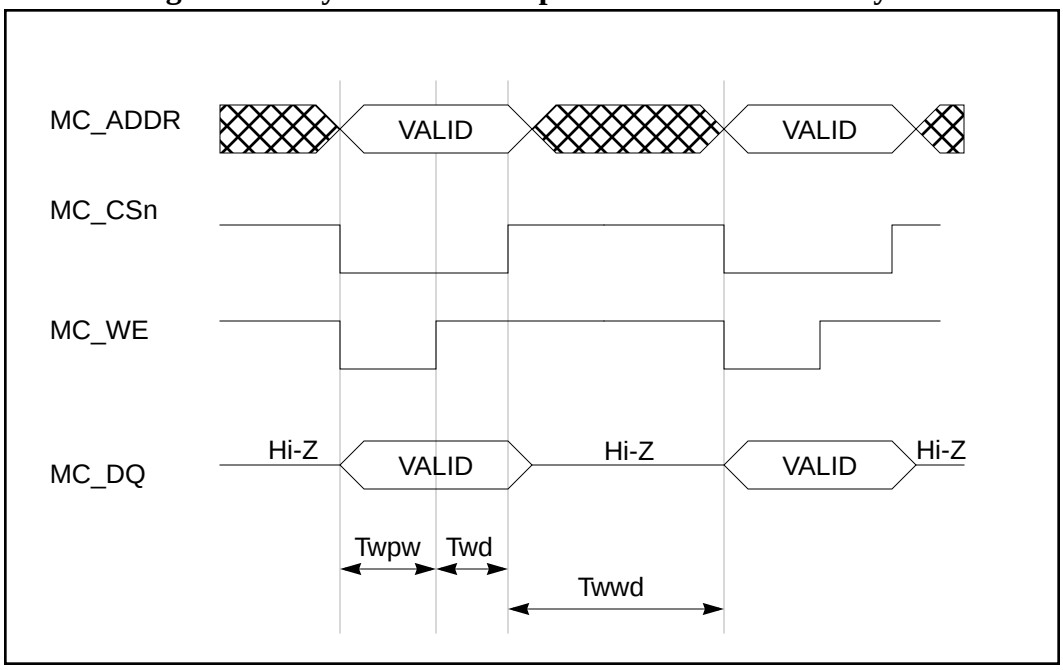
chronous Chip Select Devices, which should be sufficient to use most of the industry standard devices such as Flash, ROM and EEPROM.

Figure 15: Asynchronous Chip Select Devices Read Cycle



The figure above illustrates the Asynchronous Chip Select Devices Read timing. The Memory Controller Core will assert the address, chip select and output enable at the same time. It will wait Trdv nS for the data to be valid on the memory interface, and latch the data internally. Then it will de-assert the address, chip select and output enable to the Asynchronous Chip Select Devices. It will now wait Trdz nS for the data actually reach high impedance state, before allowing any other accesses to occur on the Memory Interface.

Figure 16: Asynchronous Chip Select Devices Write Cycle



The figure above illustrates a Asynchronous Chip Select Device write cycle. The memory Controller core will assert the address, data, chip select and write enable at the same time. After Twpw nS, it will de-assert write enable. After Twd nS it will de-assert the address, data and chip select. The next write will not be performed until Twwd nS have elapsed.

The actual internal write operation of a attached Flash might take much longer than the actual write cycle. The host is responsible to either time write operations or sample the FRDY bit in the main CSR. The FRDY bit is the *mc_sts* signal provided by some Flash devices. It indicates when a Flash is busy or ready for the next operation.

In order for write operation to be performed, the *WP* bit in CSCn register must be cleared. If this bit is not cleared, a write operation will not be performed.

The host is also responsible for setting the program enable signal (*mc_vpen*) when writing to Flash devices. This is accomplished by writing a one in to the *FVPEN* bit in the main CSR.

3.3.3.1. TMSn Register Configuration for Asynchronous Chip Select Devices

For Asynchronous Chip Select Devices, the TMSn register holds values for internal counters that time the various access parameters. This internal counters are clocked by the Memory Bus clock. All delay values, except for Trdv, are in the form of: ***(desired_delay ÷ mem_clock_period) - 2***. For example, a 60nS delay is desired. The actual value for the TMS register would be: ***(60nS ÷ 10nS) - 2 = 4.*** For Trdv the delay value is in the form of: ***(desired_delay ÷ mem_clock_period) - 3***.

For example, a 60nS delay is desired. The actual value for the TMS register would be: ***(60nS ÷ 10nS) - 3 = 3***

Table 4: TMSn parameters for Asynchronous Chip Select Devices

Bit #	Access	Description
25:20	RW	Twwd 6 bit write pulse high counter value
19:16	RW	Twd 4 bit write disable to chip disable counter value
15:12	RW	Twpw 4 bit write pulse with counter value
11:8	RW	Trdz 4 bit read to high z counter value
7:0	RW	Trdv 8 bit read to data valid counter value

3.3.4. Synchronous Chip Select Devices Timing Configuration

Synchronous Chip Select Devices are very similar to the Asynchronous Chip select devices. The differentiate in two areas: 1) Synchronous Chip Select devices must be synchronous to the Memory Controllers clock; 2) Synchronous Chip Select Devices must Assert ACK to indicate when they have accepted data, or provide valid data. Below two diagrams Illustrate the timing of Synchronous Chip Select Devices.

Figure 17: Synchronous Chip Select Devices Read Cycle

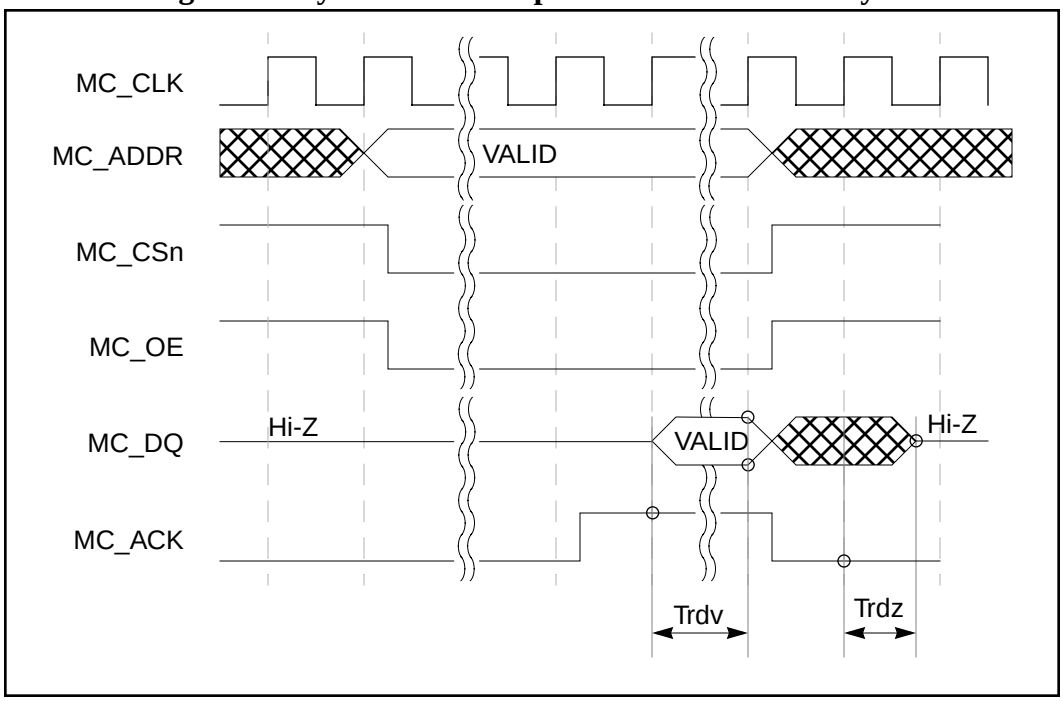
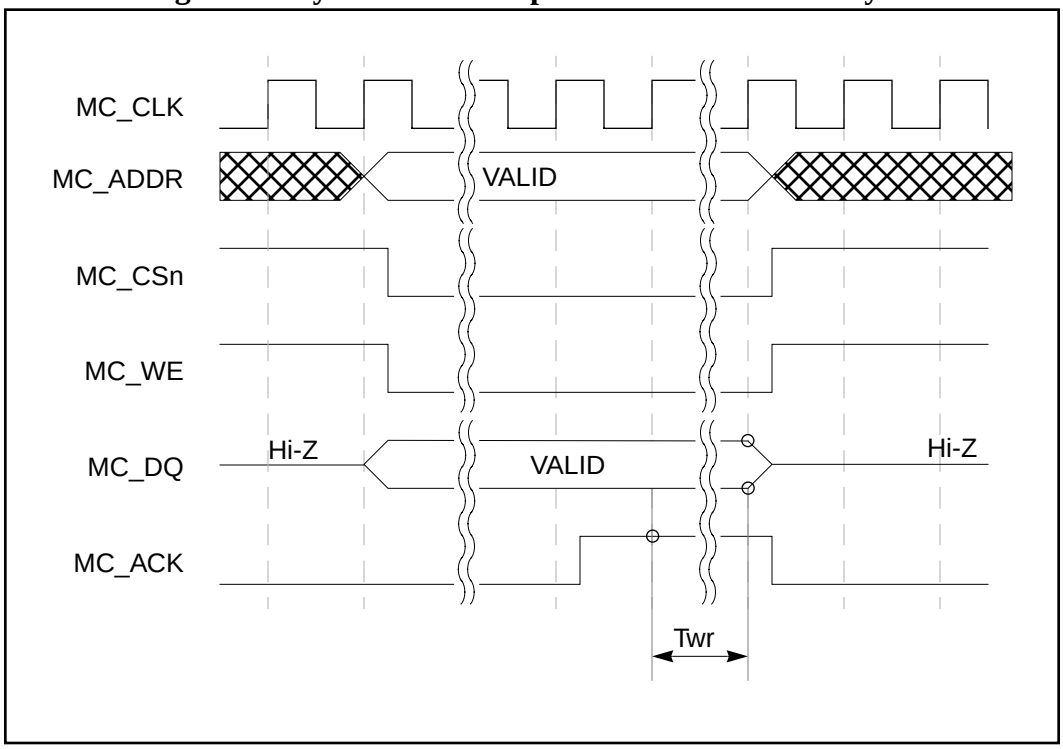


Figure 18: Synchronous Chip Select Devices Write Cycle



3.3.4.1. TMSn Register Configuration for Synchronous Chip Select Devices

For Synchronous Chip Select Devices, the TMSn register holds values for internal counters that time the various access parameters. This internal counters are clocked by the Memory Bus clock.

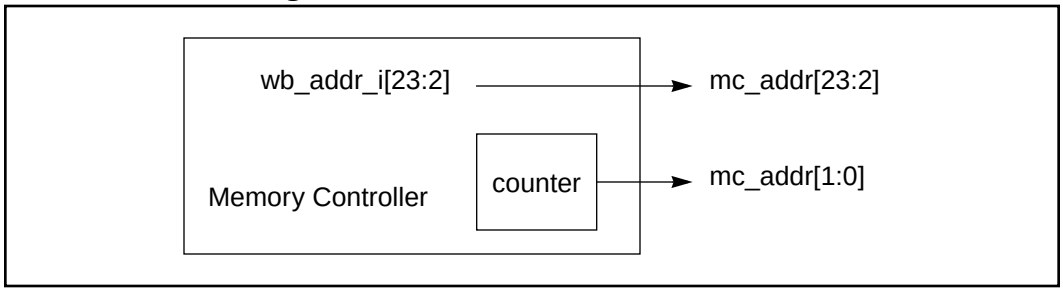
Table 5: TMSn parameters for Synchronous Chip Select Devices

Bit #	Access	Description
24:16	RW	Tto 9 bit time out counter value Determines the maximum time the Memory Controller will wait for *mc_ack* to be asserted.
15:12	RW	Twr 4 bit write pulse width counter value
11:8	RW	Trdz 4 bit read to high z counter value
7:0	RW	Trdv 8 bit read to data valid counter value

3.4. Dynamic Bus Sizing

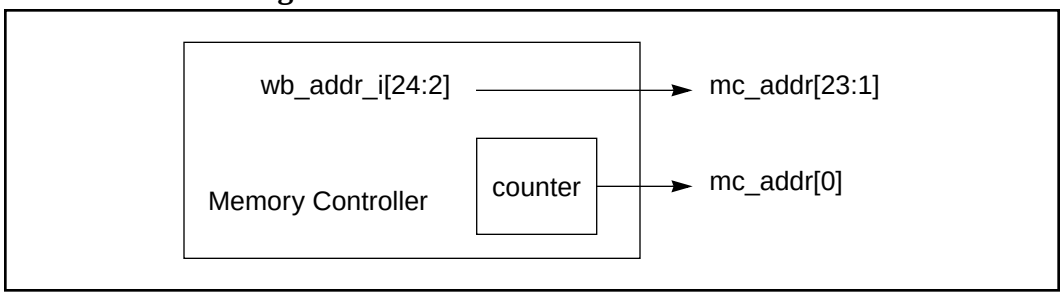
The memory controller will perform dynamic bus sizing for attached Asynchronous Chip Select devises, whenever the bus width (BW field in the CSC register) is set to 8 or 6 bits. When the bus width is 8 bits, the memory controller will perform 4 individual reads to assemble one 32 bit word. It will also automatically generate the address to select the appropriate byte.

Figure 19: 8 bit Bus Address Generation



When 16 bit bus width is selected, the memory controller will perform two reads to assemble one 32 bit word. It will also automatically generate the address to select the appropriate half word.

Figure 20: 16 bit Bus Address Generation



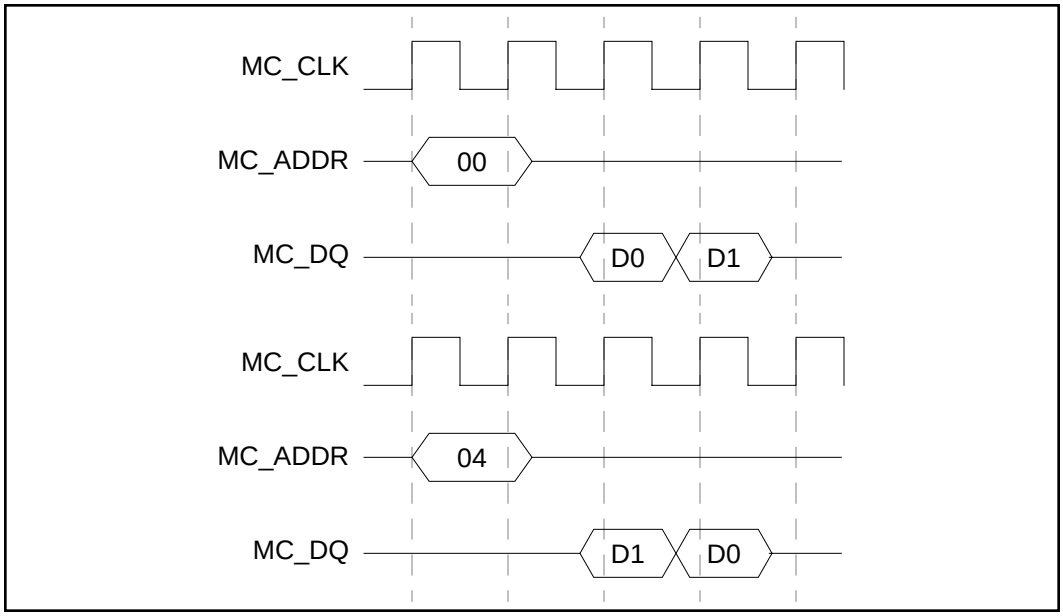
3.5. Memory Burst Cycles

Since this Memory Controller utilizes the burst capabilities of memory devices, all bursts have to be performed to sequential addresses. Depending on the memory device and selected burst features, this may vary from 16 to 4096 byte transfers. For SDRAMS this applies to both read and write cycles (unless single writes are chosen), for SSRAMs this applies to read cycles only. On all other devices bursts are performed by concatenating individual read and write cycles, and the bursts address may be in any sequence.

All devices that support bursts, will wrap around on the bursts boundaries. If the starting address is not aligned to the burst boundary, it may wrap around to the beginning while performing a burst.

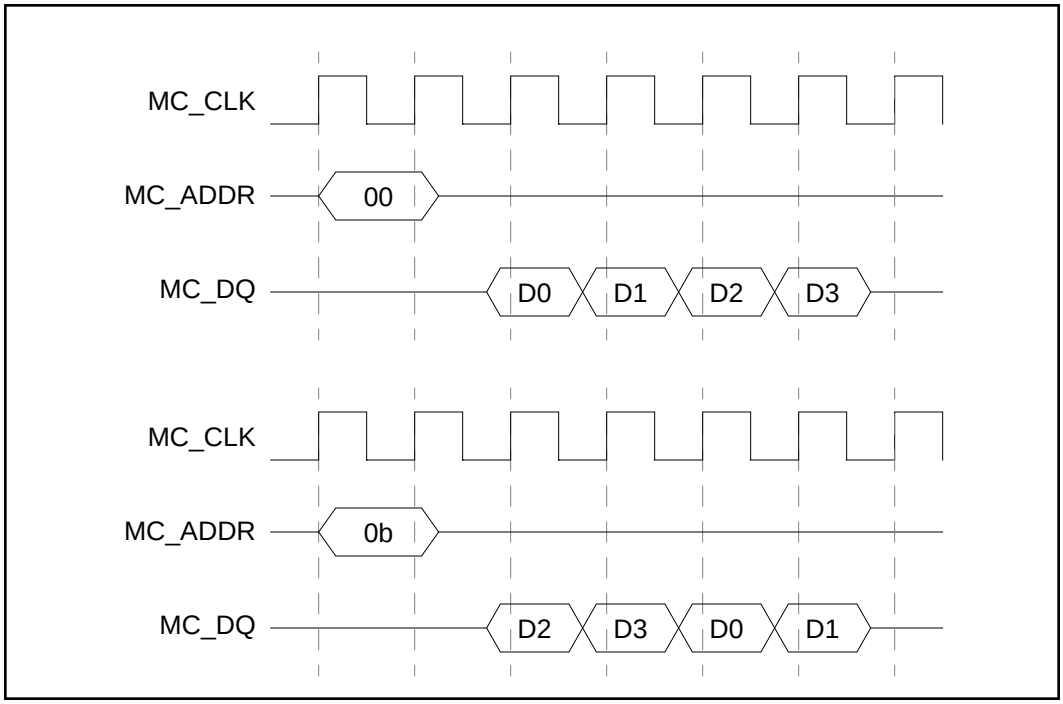
In other words a device that supports a burst size of four (16 bytes), will internally increment the two lower address bits. If the starting address is 2, the sequence will be 2-3-0-1. For 8 byte bursts, *MC_ADDR[2]* selects which word in the burst is accessed first, for 16 byte bursts, *MC_ADDR[3:2]* select which word inside the burst is accessed first, and for 32 byte bursts, *MC_ADDR[4:2]* select which word is addressed first. In all cases if the first word is not 0 (zero), the burst will wrap around. Please consult the memory devices data sheets for more information.

Figure 21: Burst Address Cycling (8 byte burst)



The next figure illustrates two burst examples for 16 byte bursts.

Figure 22: Burst Address Cycling (16 byte burst)



This slightly deviates from the way the WISHBONE specification defines burst. However, to support truly random burst, would mean a major performance impact, as burst capabilities of memory devices could not be utilized.

3.6. RMW Cycles

The Memory Controller supports read-modify-write cycles as described by the WISHBONE specification. The RMW cycles can be any number of reads followed by any number of writes. All burst rules apply, except that it is guaranteed that the memory controller will fetch a new address when the data flow direction changes. In other words the Memory Controller will definitely fetch a new address for the write portion.

3.7. Error Signaling

The Memory Controller will assert the WISHBONE *WB_ERR_O* output when one of the following conditions occurs:

1. Parity error is detected.
2. A write to write protected chip select is attempted.
3. An access to synchronous chip select device times out.

Please note, when the *WB_ERR_O* is asserted the *WB_ACK_O* will not be asserted, according to the WISHBONE specification.

3.8. Power-On Configuration

The Power On configuration allows for the core to be externally configured to a default state that will allow a host CPU to boot from an external device such as a Flash.

During a reset, all chip selects will be de-asserted, and non of the external devices should be driving the Memory Controller data bus (*MC_DATA*). External pull up and pull down resistors on the data bus will provide for an configuration value to be provided to the Memory Controller and the rest of the system. The *POC* register, latches the value on the data bus during a reset.

The memory controller interprets bits 3 through 0 for it's internal configuration. The remaining bits may be used by other system components. The table below outlines the memory controller power on configuration options.

Table 6: Power-On Boot Configuration

POC[3:2]	POC[1:0]	Description
		Data Bus Width
----	00	8 Bit Data Bus
----	01	16 Bit Data Bus
----	10	32 Bit Data Bus
----	11	RESERVED
		This bits correspond to the CSCn register bits 5:4
		Device Type
00	----	DISABLED
01	----	SSRAM
10	----	Async. Device
11	----	Sync. Device
		This bits correspond to the CSCn register bits 3:1, except "00" is interpreted as Power-On Boot configuration is disabled.

3.9. Performance Notes

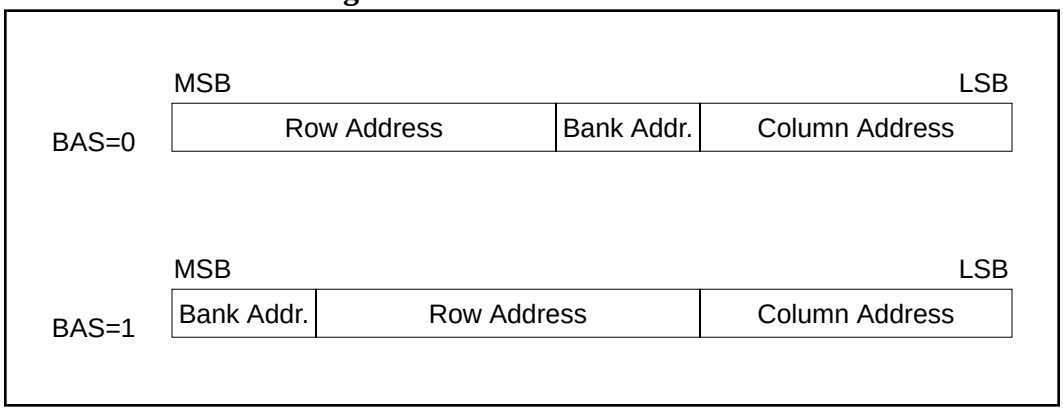
This memory controller has been optimized to achieve the highest bandwidth possible with SDRAMs. Most SDRAMs available today support theoretical maximum burst speeds between 228 Mbytes/s (burst=4, CL=3, F=100Mhz) and 528 Mbytes/s (burst=256, CL=2, F=133Mhz) for row accesses, depending on burst size and memory speed. Even though this numbers are almost impossible to achieve in real life application, one should always try to get the most from the memory system.

This high throughput applies only to accesses in the same row. Each time a different row is selected huge access penalties are imposed. To sustain high bandwidth over longer periods of time, it is therefore desirable to have very large row sizes. This can be achieved by selecting SDRAM configurations that naturally pro-

vide larger rows sizes (e.g. 8 bit wide SDRAM devices) or by extending the row accesses across multiple banks. Most SDRAMS provide 4 banks.

This memory controller utilizes this feature. The address bus is divided in to column address row address and bank address. The goal is to minimize the changes in the row address, as costly closing and reactivation of row cycles must be performed. Therefore this memory controller allows for the address to be distributed in two ways (from MSB to LSB): 1) Row Address, Bank Address, Column Address; 2) Bank Address, Row Address, Column Address.

Figure 23: Address Distribution

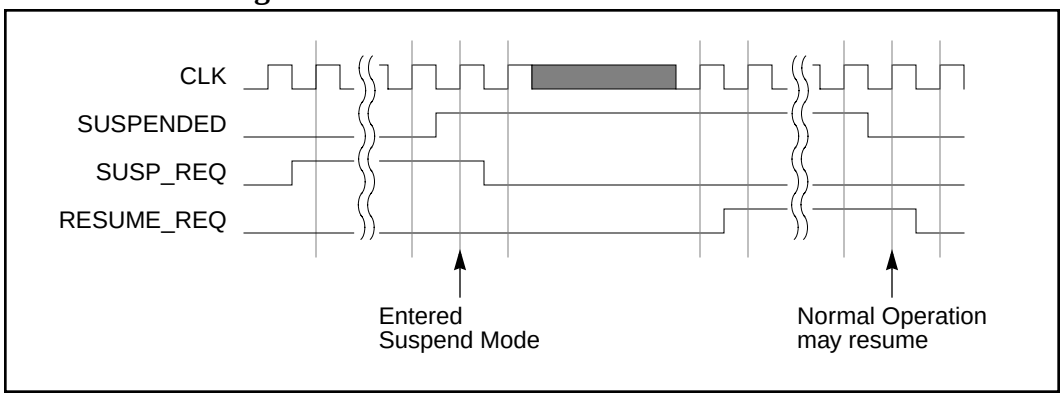


This trick effectively either extends the row size four fold (BAS=0), or creates four smaller memories that can be randomly accessed (BAS=1). Even though each bank has to be individually activated, it can remain activated as long as the row address for the bank does not change.

3.10. Power Down (Sleep) Mode

The Memory Controller supports suspending of attached memory devices and the controller itself. To suspend the memories and the Memory Controller, the susp_req signal must be asserted and kept asserted until the suspended output is asserted. Once in the power down mode, both the Memory Controller and Memory Clocks can be turned off. All memory contents will be preserved. To resume, the clocks must be turned on first. After the clocks are stable, the resume_req signal must be asserted. Once the suspended output has been de-asserted, normal operation may resume. The memory controller must not be reset during suspend or normal operations, as all contents of attached SDRAMS may be lost.

Figure 24: Power Down Mode Enter/Resume



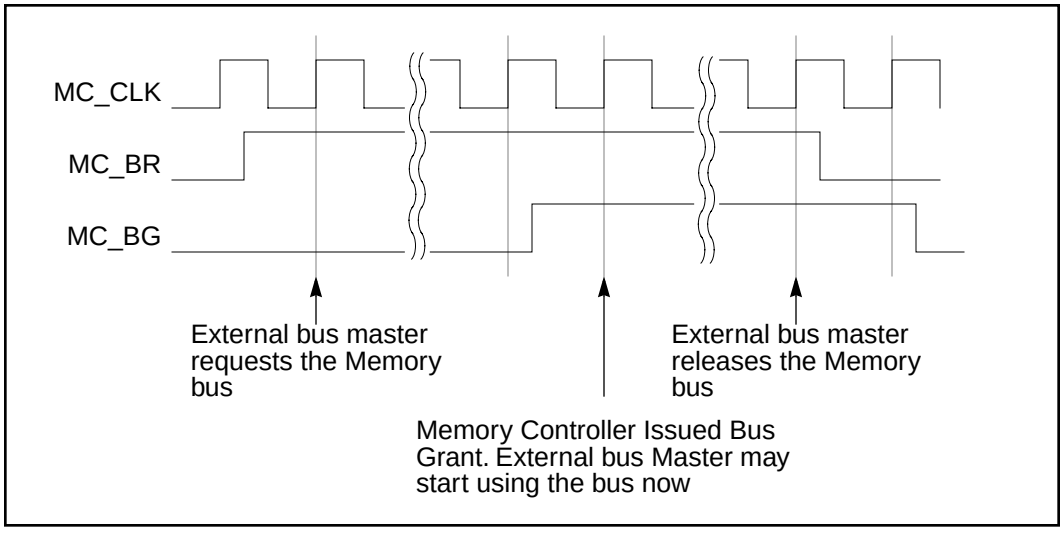
The susp_req signal must be de-asserted, once suspended output is activated. The resume_req input may be asserted at any time (even with susp_req). The Memory Controller will sample resume_req only once it has entered suspended state (suspended output active).

Before entering suspended state, the Memory Controller will place all attached SDRAMs in to “Self Refresh” mode. In this mode the SDRAMs consume very little power, do not require an external clock, and maintain memory contents by performing internal refresh cycles.

3.11. Memory Bus Arbitration

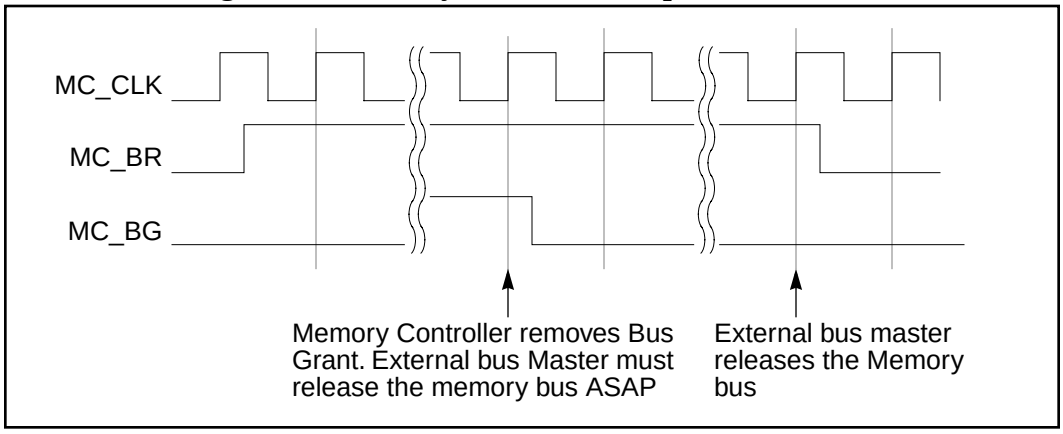
The Memory Controller includes and Arbiter for External Bus Masters. External Bus masters may acquire the bus by asserting *mc_br.* Once grant has been given (*mc_gnt* asserted) the external bus master may use the memory bus. External bus masters must keep the *mc_br* signal asserted for the entire duration they are on the bus. Once they have released the bus they must de-assert *mc_br.*

Figure 25: External Bus Master bus Acquisition and Release



If the memory controller de-asserted *mc_gnt* before the external master releases *mc_br*, this indicates to the external bus master to get of the bus as soon as possible, and indicate that the bus has been released by de-asserting *mc_br*.

Figure 26: Memory Controller Requests Bus back



Note:

When an external bus master takes over the memory bus, SDRAMs are NOT placed in to a sleep/suspend mode. Therefore, external bus masters must not block the bus for more than the maximum SDRAM refresh rate (15-31 uS). The memory controller will request the Memory bus back when it needs to perform a refresh or another internal device tries to access a device on the memory bus. External bus masters should release the bus as soon as possible, when the Memory Controller requests the bus back. ***Failure to do so, might result in loss of data due to refresh violation.***

(This page intentionally left blank)

4

Core Registers

This section describes all control and status registers inside the memory Controller core. The *Address* field indicates a relative address in hexadecimal. *Width* specifies the number of bits in the register, and *Access* specifies the valid access types to that register. Where RW stands for read and write access, RO for read only access. A ‘C’ appended to RW or RO, indicates that some or all of the bits are cleared after a read.

All RESERVED bits should always be written with zero. Reading RESERVED bits will return undefined values. Software should follow this model to be compatible to future releases of this core.

Table 7: Control/Status Registers

Name	Addr.	Width	Access	Description
CSR	0	32	RW	Main Configuration/Status Register
POC	4	32	RO	Power-On Configuration Register
BA_MASK	8	32	RW	Base Address Mask Register
CSC0	10	32	RW	Chip Select Configuration Register 0
TMS0	14	32	RW	Timing Select Register 0
CSC1	18	32	RW	Chip Select Configuration Register 1
TMS1	1c	32	RW	Timing Select Register 1
CSC2	20	32	RW	Chip Select Configuration Register 2
TMS2	24	32	RW	Timing Select Register 2
CSC3	28	32	RW	Chip Select Configuration Register 3
TMS3	2c	32	RW	Timing Select Register 3
CSC4	30	32	RW	Chip Select Configuration Register 4
TMS4	34	32	RW	Timing Select Register 4
CSC5	38	32	RW	Chip Select Configuration Register 5

Table 7: Control/Status Registers

Name	Addr.	Width	Access	Description
TMS5	3c	32	RW	Timing Select Register 5
CSC6	40	32	RW	Chip Select Configuration Register 6
TMS6	44	32	RW	Timing Select Register 6
CSC7	48	32	RW	Chip Select Configuration Register 7
TMS7	4c	32	RW	Timing Select Register 7

4.1. Control Status Register (CSR)

This is the main control and status register.

Table 8: CSR Register

Bit #	Access	Description
31:24	RW	Refresh Prescaler The refresh Prescaler is responsible for generating a 488.28nS time reference for the Refresh Generator. This field indicates how many bus cycles it would take to generate the closest (lower) time interval. For a WISHBONE bus frequency of 200 Mhz this would be 97 cycles to generate a 485 nS time reference. This represents a 0.63% error. The value written in to this field is the actual calculated value minus 1.
23:11	RO	RESERVED
10:8	RW	REF_INT Refresh Interval 0: 0.976 uS 1: 1.953 uS 2: 3.906 uS 3: 7.812 uS 4: 15.625 uS 5: 32.250 uS 6: 62.500 uS 7: 125.00 uS Actual refresh interval might vary slightly. It might be up 1% shorter than indicated, but it will never exceed the indicated values.
7:3	RO	RESERVED
2	RW	FS Flash Sleep. Writing a 1 to this bit will place all attached Flash devices in to a sleep (power-down) mode. This is achieved by forcing the *mc_rp_* signal to a logical zero.
1	RW	F VPEN This bit is output on the *mc_vpen* output

Table 8: CSR Register

Bit #	Access	Description
0	RO	FRDY Value of the *mc_sts* pin

Value after reset:

CSR: 0000h

4.2. Power On Configuration Register (POC)

This register latches the value of the Memory Interface data bus during reset.

Value after reset:

POC: <Value of the Memory Interface Data Bus during reset>

4.3. Base Address Mask Register (BA_MASK)

This register holds the address mask for all chip selects.

Table 9: COR Register

Bit #	Access	Description
31:16	RO	RESERVED
7:0	RW	MASK Base Address Mask

Value after reset:

BA_MASK: 0000 h

4.4. Chip Select Configuration Register (CSCn)

The Chip Select Configuration registers, determine the address range and attached device type for each chip select.

Table 10: Chip Select Configuration Register

Bit #	Access	Description
31:24	RO	RESERVED

Table 10: Chip Select Configuration Register

Bit #	Access	Description
23:16	RW	SEL Base Address. This field is ANDed with address mask and compared to address input (address 28-21). If there is a match this chip select is asserted.
15:12	RO	RESERVED
11	RW	PEN Parity Enable 1- Parity Generation and Checking is enabled 0 - Parity Generation and Checking is disabled Parity Generation and Checking is supported for SSRAM and SDRAM memories only.
10	RW	KRO 1- Keep Row Open 0 - Close Row after Read/Write operation This bit has only meaning for SDRAMs
9	RW	BAS Bank Address Select 0 - Bank Address follows Column Address 1 - Bank Address follows Row Address This bit has only meaning for SDRAMs
8	RW	WP Write Protect. This bit is used to write protect an area. 1- Write Protected 0 - Writes are enabled
7:6	RW	MS Memory Size 0 - 64 Mb 1 -128 Mb 2 - 256 Mb 3 - RESERVED This bit has only meaning for SDRAMs
5:4	RW	BW Bus With. For Asynchronous Chip Select Devices it indicates the data bus width of the attached device. The memory controller will perform multiple reads and assemble a full 32 bit word when reading from these devices with less than 32 bit data bus. For SDRAM it indicates the data bus width of a single SDRAM device. The total bus width for SDRAM must be 32 bits wide. 0 - 8 bit bus 1 - 16 bit bus 2 - 32 bit bus 3 - RESERVED

Table 10: Chip Select Configuration Register

Bit #	Access	Description
3:1	RW	MEM_TPYE This field indicates the Memory Type. 0 - SDRAM 1 - SSRAM 2 - Asynchronous Chip Select Devices 3 - Synchronous Chip Select Devices, external ack 4-7 - RESERVED
0	RW	EN Chip Select Enable 1 - This chip select is enabled 0 - This chip select is disabled

Value after reset:

CSCn: 0000h

4.5. Timing Select Register (TMSn)

The timing select register, defines various timing parameters for the attached memories.

Value after reset:

TMSn: 0000h

(This page intentionally left blank)

5

Core IOs

5.1. Interface IOs

The SoC interface is WISHBONE Rev B compliant.

Table 11: Host Interface (WISHBONE)

Name	Width	Direction	Description
clk_i	1	I	WISHBONE Clock Input
rst_i	1	I	WISHBONE Reset Input The reset is asynchronous and an active low signal.
wb_addr_i	32	I	Address Input
wb_data_i	32	I	Data Input
wb_data_o	32	O	Data Output
wb_sel_i	4	I	Indicates which bytes are valid on the data bus.
wb_we_i	1	I	Input for slave. Indicates a Write Cycle when asserted high.
wb_cyc_i	1	I	Input for slave. Encapsulates a valid transfer cycle.
wb_stb_i	1	I	Input for slave. Indicates a valid transfer.
wb_ack_o	1	O	Acknowledgment Output. Indicates a normal Cycle termination.
wb_err_o	1	O	Error acknowledgment output. Indicates an abnormal cycle termination.

In addition, the Memory Controller has a power management interface, that allow the core to be placed in to a power saving mode and turn off the clocks.

Table 12: Power Management Interface

Name	Width	Direction	Description
susp_req_i	1	I	Request to enter suspend state
resume_req_i	1	I	Request to exit suspend state and enter normal operation mode
suspended_o	1	O	Indicates that the memory Controller has entered suspended mode. When this signal is asserted, all clocks may be turned off.
poc_o	32	O	This is the POC register output. It may be used by other system modules for power on configuration. See also section 3.8. “Power-On Configuration” on page 26.

5.2. Memory Interface IOs

This section describes the memory interface signals.

Table 13: Memory Interface IOs

Name	Width	Direction	Description
mc_clk_i	1	I	Memory Clock Input
mc_br_pad_i	1	I	External Master Bus request
mc_bg_pad_o	1	O	External Master Bus Grant
mc_ack_pad_i	1	I	Memory Controller Acknowledgement
mc_data_pad_o	32	O	Memory Data Bus Output
mc_data_pad_i	32	I	Memory Data Bus Input
mc_dp_pad_o	4	O	Memory Data Byte Parity Output
mc_dp_pad_i	4	I	Memory Data Byte Parity Input
mc_doe_pad_doe_o	1	O	Memory Data Bus Output Enable
mc_coe_pad_coe_o	1	O	Memory Address and Control Signals Output Enable
mc_addr_pad_o	24	O	Memory Address Bus (including Bank Address) mc_addr[13] is used as BA0 and mc_addr[14] is used as BA1 for SDRAMs.
mc_dqm_pad_o	4	O	Memory Byte Enables
mc_oe_pad_o_	1	O	Memory Output Enable
mc_we_pad_o_	1	O	Memory Write Enable

Table 13: Memory Interface IOs

Name	Width	Direction	Description
mc_cas_pad_o_	1	O	Memory Column Address Select
mc_ras_pad_o_	1	O	memory Row Address Select
mc_cke_pad_o_	1	O	Memory Clock Enable
mc_cs_pad_o_	8	O	Memory Chip Select Outputs
mc_rp_pad_o_	1	O	Flash Reset/Power-Down
mc_sts_pad_i	1	I	Flash Ready/Busy Status
mc_vpen_pad_o	1	O	Flash Erase/Program enable
mc_adsc_pad_o_	1	O	SSRAM ADSC signal
mc_adv_pad_o_	1	O	SSRAM Address Advance
mc_zz_pad_o	1	O	SSRAM Snooze Enable

Total IOs (after inserting Bi-Dir buffers for Data Lines): 87

(This page intentionally left blank)

Appendix A

Core HW Configuration

This Appendix describes the configuration of the core. This step is performed before final Synthesis and tape-out of the core.

A.1. Address Space Selection

The Memory Controller includes configuration registers and memory address spaces. The address range for each of them can be individual selected by editing the “mc_defines.v” file. The define statement below, specifies the address decoding for registers.

```
`define     MC_REG_SEL  (wb_addr_i[31:29] == 3'h7)
```

In this case address lines 31:29 must be all ones to access the configuration registers.

The next define statement specifies the address decoding for all chip selects and the attached devices.

```
`define     MC_MEM_SEL  (wb_addr_i[31:29] == 3'h0)
```

In this case address lines 31:29 must be all zeros to select the attached memories.

A.2. Chip Selects

This core supports up to 8 Chip Selects. The actual number of chip selects supported by any given implementation may be defined by editing the “mc_defines.v” file. Where Chip Select 0 is always supported and can not be disabled.

To select how many Chip Selects are supported, look for the following lines:

```
`define     MC_HAVE_CS1 1     // Chip Select 1 Present
`define     MC_HAVE_CS2 1     // Chip Select 2 Present
`define     MC_HAVE_CS3 1     // Chip Select 3 Present
`define     MC_HAVE_CS4 1     // Chip Select 4 Present
//`define   MC_HAVE_CS5 1     // Chip Select 5 Not Implemented
//`define   MC_HAVE_CS6 1     // Chip Select 6 Not Implemented
//`define   MC_HAVE_CS7 1     // Chip Select 7 Not Implemented
```

In this example, Chip Selects 4:0 are supported, and Chip Selects 7:5 are omitted.

A.3. Power On Boot Selection

One of the Chip Selects may be used for Power-On Boot. The following define statement in the “mc_defines.v” file specifies which chip select is used for Power-On Boot.

```
`define      MC_DEF_SEL         0
```

In addition, one may chose the value for the TMS register for Power-On Boot. It is recommended to set it all to ones, selecting the slowest possible mode for Memory Device access. This will guarantee that any device may be used. Software may adjust this values during booting, to increase boot speed.

```
`define      MC_DEF_POR_TMS     32'hffff_ffff
```

The define statement above specifies the Power-On Boot value for the TMS register.

A.4. Refresh Counter Configuration

When SDRAMs are initialized, they require for several Refresh cycles to be performed before normal operations may be performed. Typically the requirement is anywhere between two and eight refresh cycles. It is recommended to always chose at least eight refresh cycles during initialization and set the below define statement to eight.

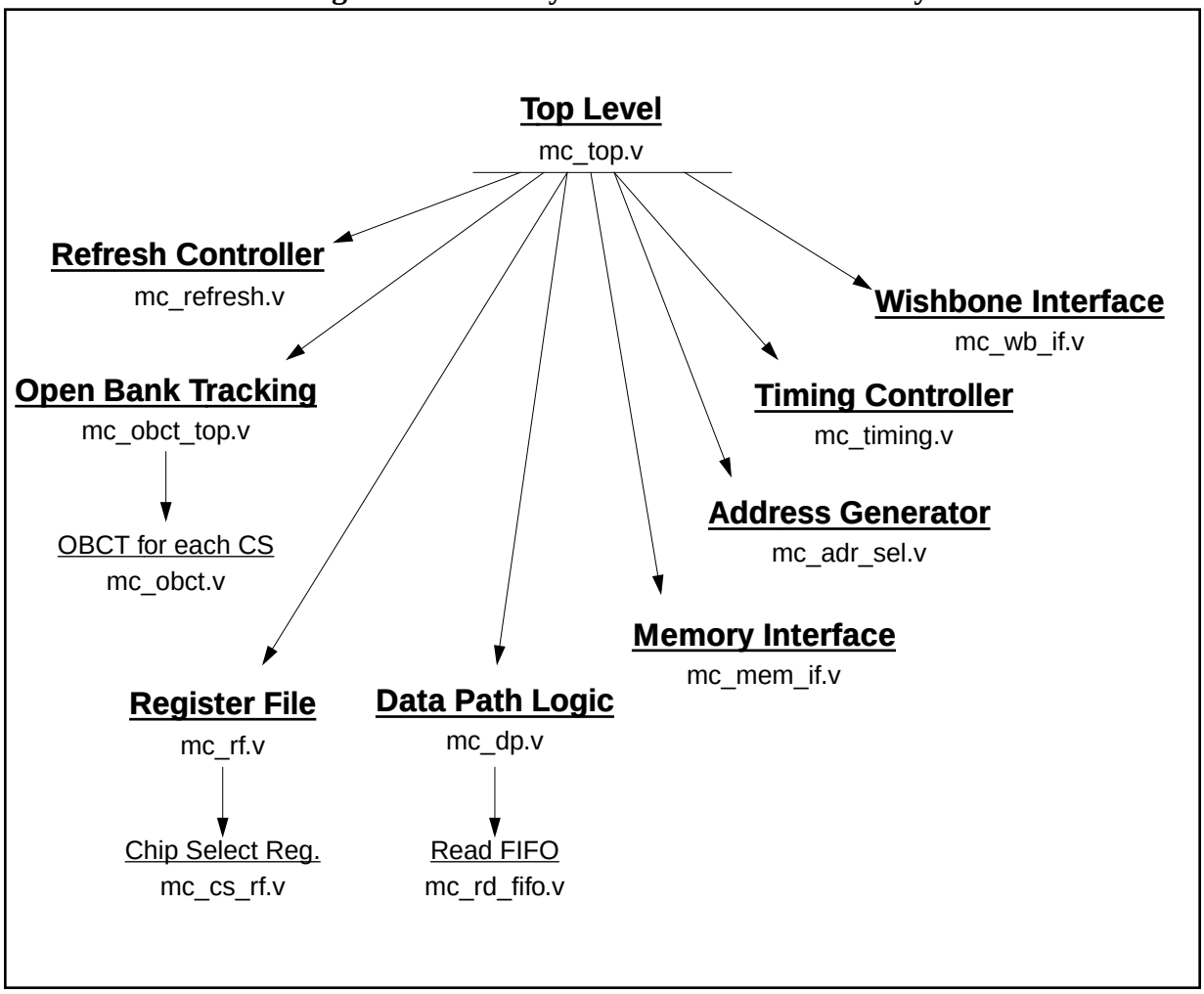
```
`define      MC_INIT_RFRC_CNT   8
```

Appendix B

File Structure

This section outlines the hierarchy structure of the Memory Controller core Verilog Source files.

Figure 27: Memory Controller Core Hierarchy Structure



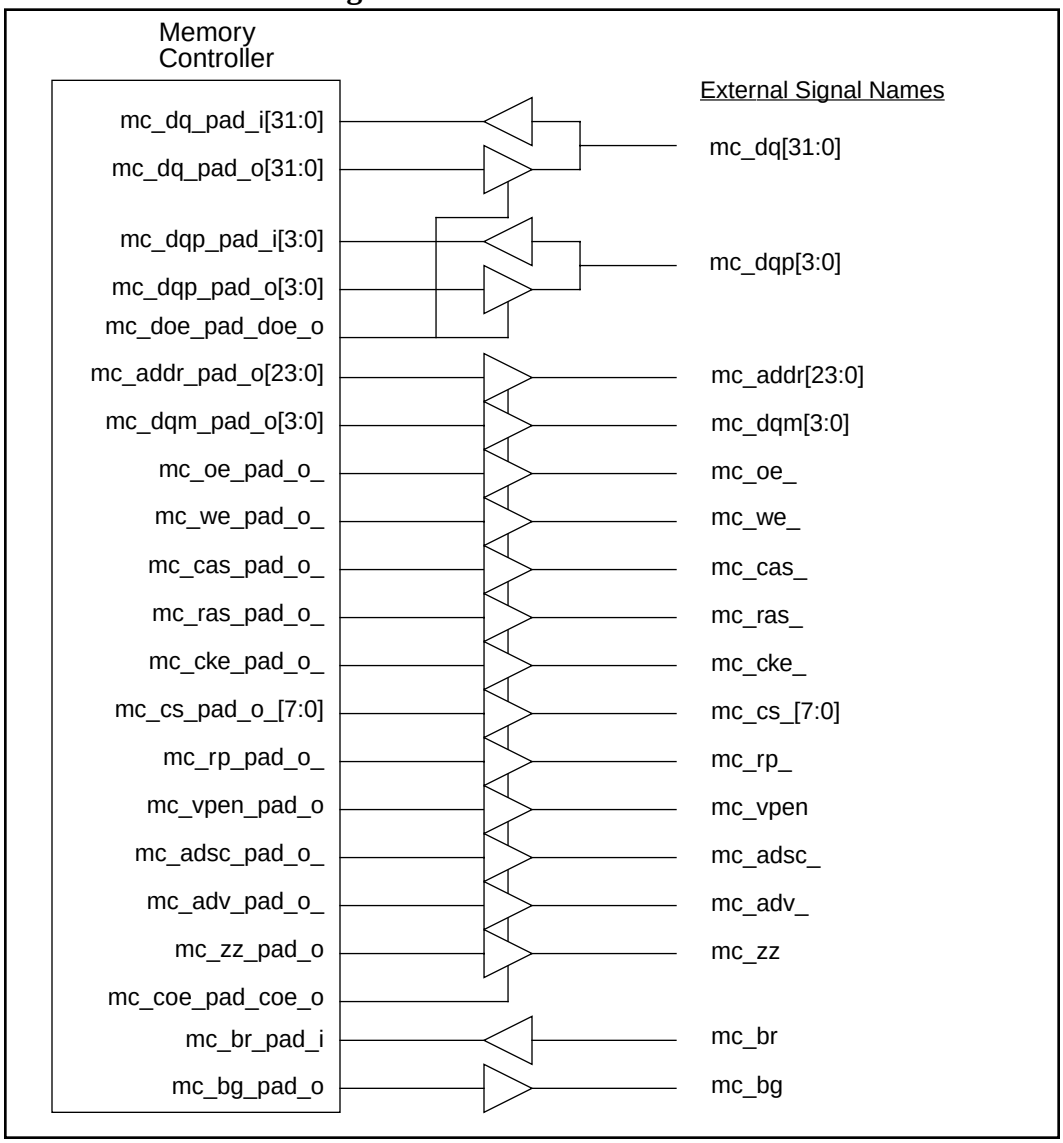
(This page intentionally left blank)

Appendix C

IO Buffers

For proper operation, IO buffers must be inserted on the Memory Bus. The Figure below illustrates the connection and wiring of all IO buffers

Figure 28: IO Buffers Insertion



(This page intentionally left blank)

Appendix D

Wiring Examples

This Section demonstrates how to connect various Memory Devices to the Memory Controller. This are just examples, and actual connections might vary. Specifically which chip select is used, is entirely up to the system designer.

Figure 29: Connecting FLASH Memories

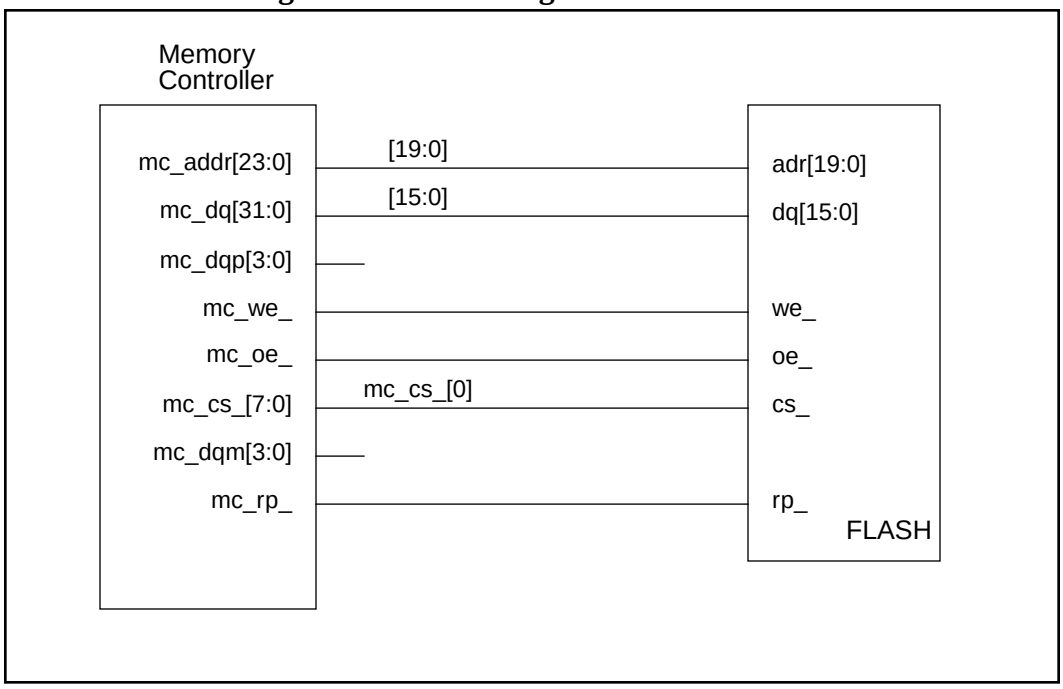


Figure 30: Connecting (x32) SDRAM Memories

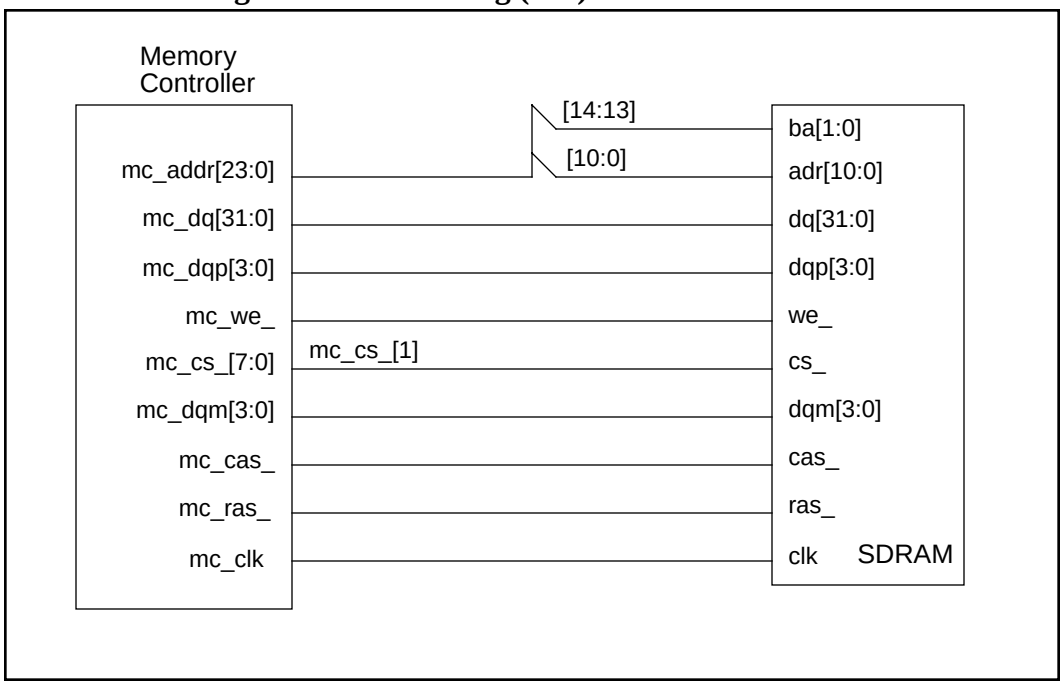


Figure 31: Connecting (x8) SDRAM Memories

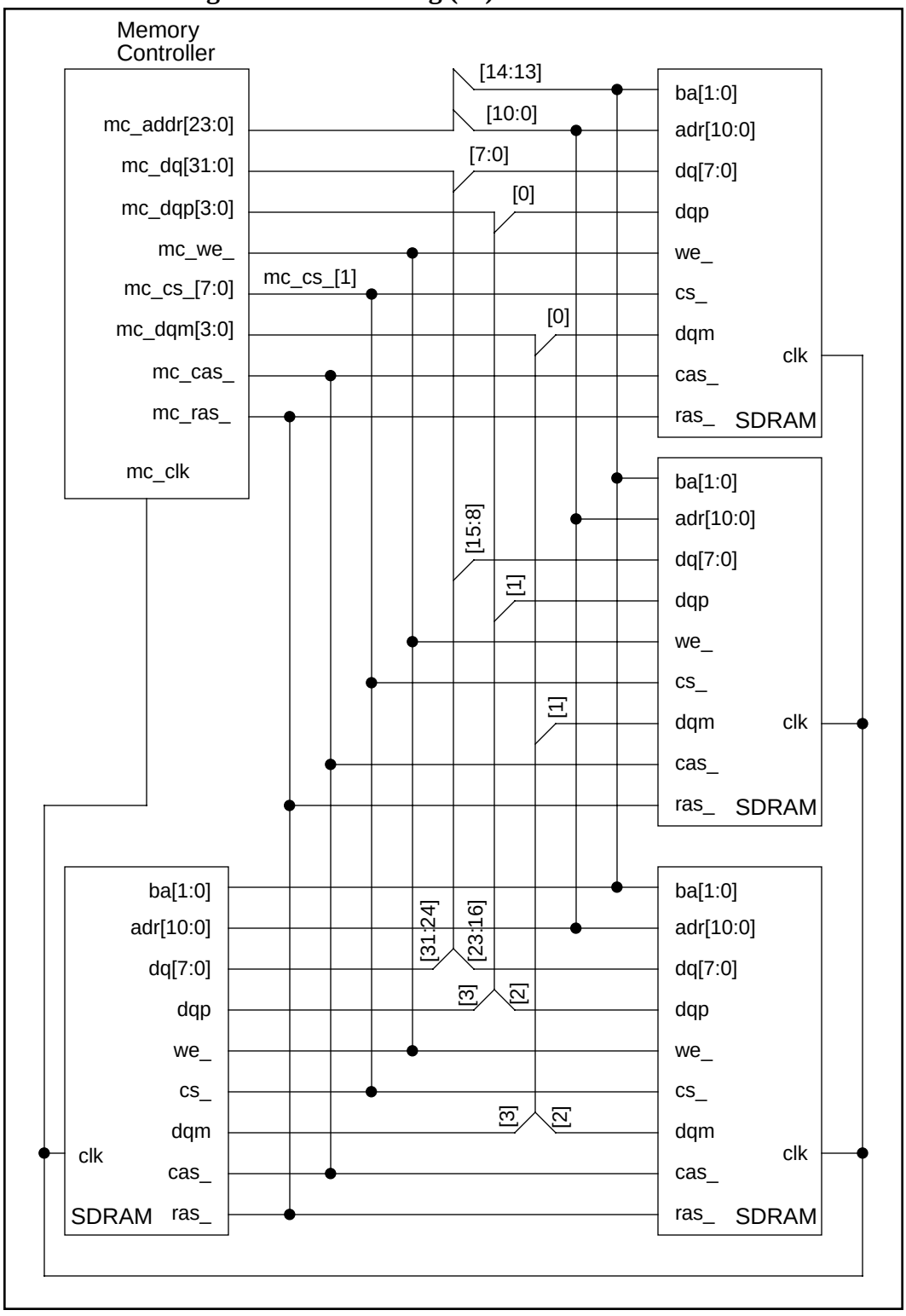
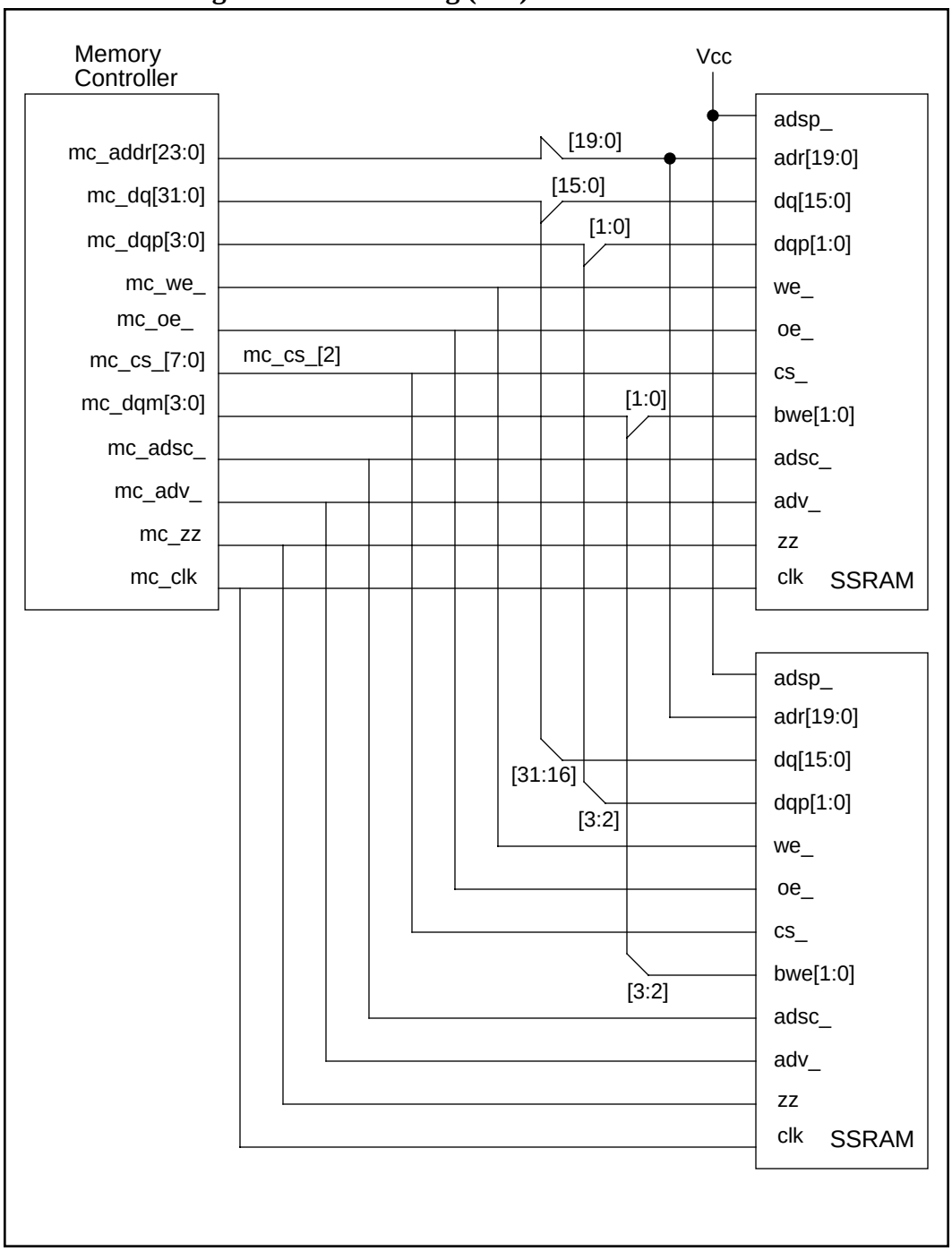


Figure 32: Connecting (x16) SSRAM Memories



Appendix E

Clocks and Reset

This Appendix describes the Clocks and Reset inputs of the Memory Controller IP core.

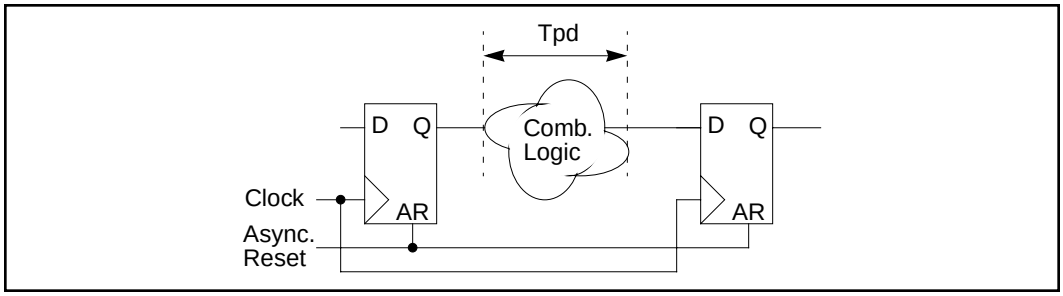
E.1. Reset

The Memory Controller utilizes a global Asynchronous Reset.

Asynchronous Reset when applied incorrectly may cause the system to come up in an undefined state caused by Metastability.

Lets analyze how asynchronous reset can be harmful to any system and how to apply it correctly. First, lets look at a typical path using asynchronous reset. The figure below shows an example.

Figure 33: Example Design with Async. Reset

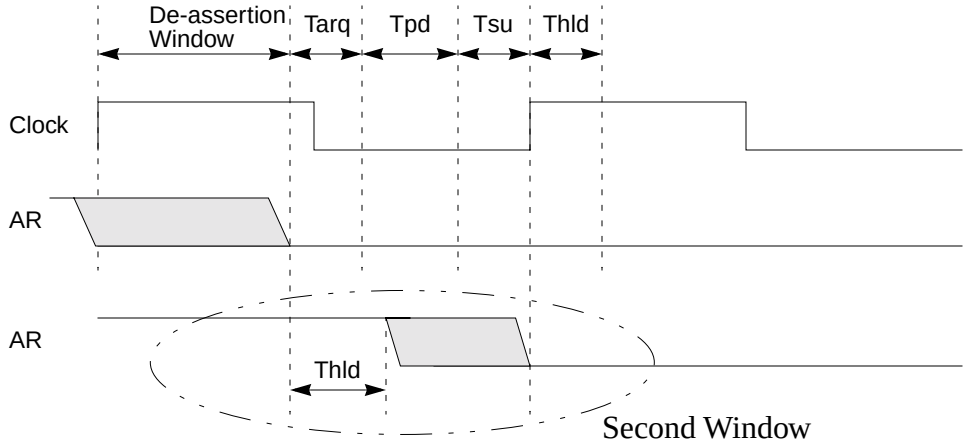


Remember that Flip-Flops may become metastable when their Setup (*Tsu*) or Hold (*Thld*) times are violated. Below timing diagram identifies the windows for correct de-assertion of Asynchronous Reset. The main window may be expressed as the time between the rising edge of the clock and *Tcycle-(Tarq+Tpd+Tsu)*. Where *Tcycle* is the clock period and *Tarq* is the propagation delay between Asynchronous Reset input and Flip-Flop Q Output.

A second window might be identified as the time between *Tcycle-(Tarq+Tpd+Tsu-Thld)* and the next rising edge of the clock. However this window might be more difficult to meet as it’s starting point is arbitrary within a Clock Period.

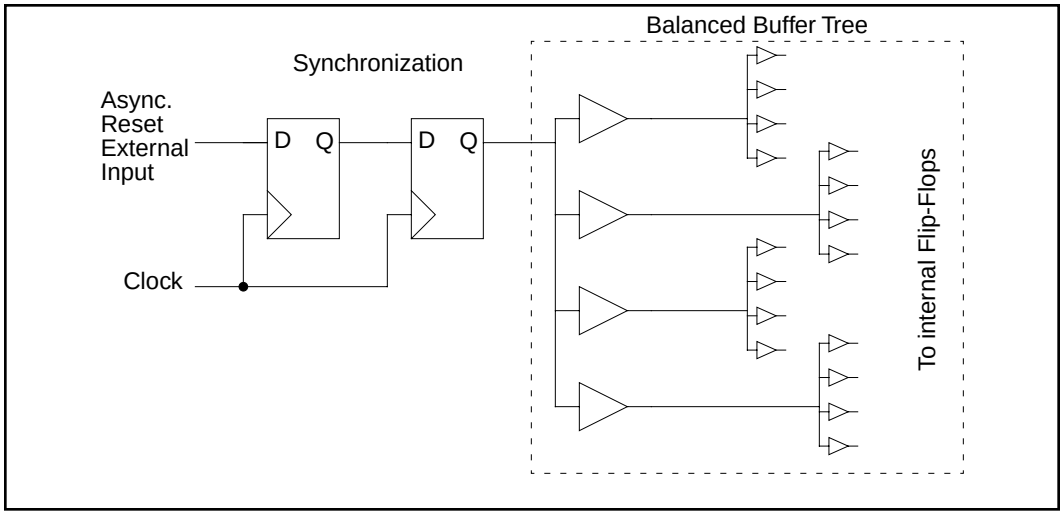
Another consideration is the skew between Asynchronous Reset throughout the chip. The designer should ensure that the skew is kept to an absolute minimum and take the skew in consideration when calculating the window for Async. Reset de-assertion.

Figure 34: Async Reset de-assertion Example



Now we can design a circuit that will properly assert Asynchronous Reset meeting the above requirements. The Figure below illustrates a sample circuit that should be sufficient for asserting and de-asserting Asynchronous reset.

Figure 35: Asynchronous reset generation circuit



E.2. Clocks

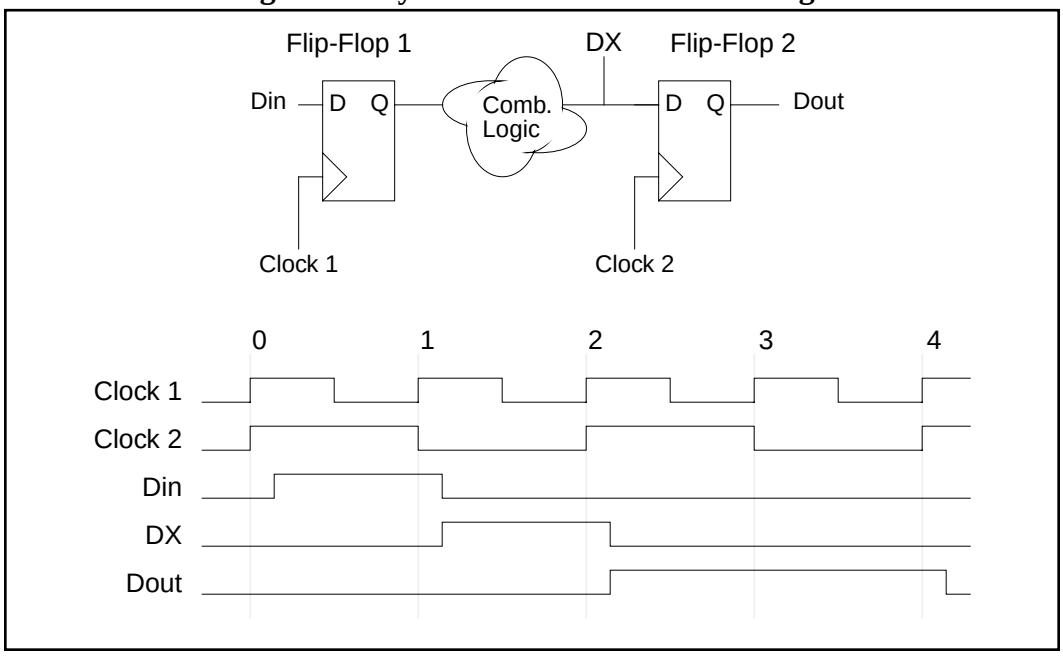
The Memory Controller IP core requires two clocks: 1) WISHBONE Clock; 2) Memory Clock. The goal was to run the WISHBONE bus at twice the speed of the Memory Bus. So for 100Mhz SDRAMs (PC100), the WISHBONE bus would run

at 200Mhz; for 133Mhz SDRAMS (PC133), the WISHBONE bus would run at 266Mhz and with 166Mhz SDRAMs, the WISHBONE bus would run at 332 Mhz.

To simplify the design and avoid having to implement synchronization between the two clock domains, the requirement was made that both clocks are 100%[1] phase synchronous. Lets try to understand what this *really* means:

Lets review the design choice first. The assumption was made that the clocks are synchronous and no expensive synchronizers are needed. That means signals may cross clock domains any time without being synchronized.

Figure 36: Synchronous Multi-Clock Design



In this example it becomes clear how clock phase shift will brake the design. Look at clock edge 2 in the figure above. If there is a delay of clock 2 relative to clock 1, Flip -Flop 2 might miss the active stage of DX, or enter a metastable state if it's setup or hold times are violated.

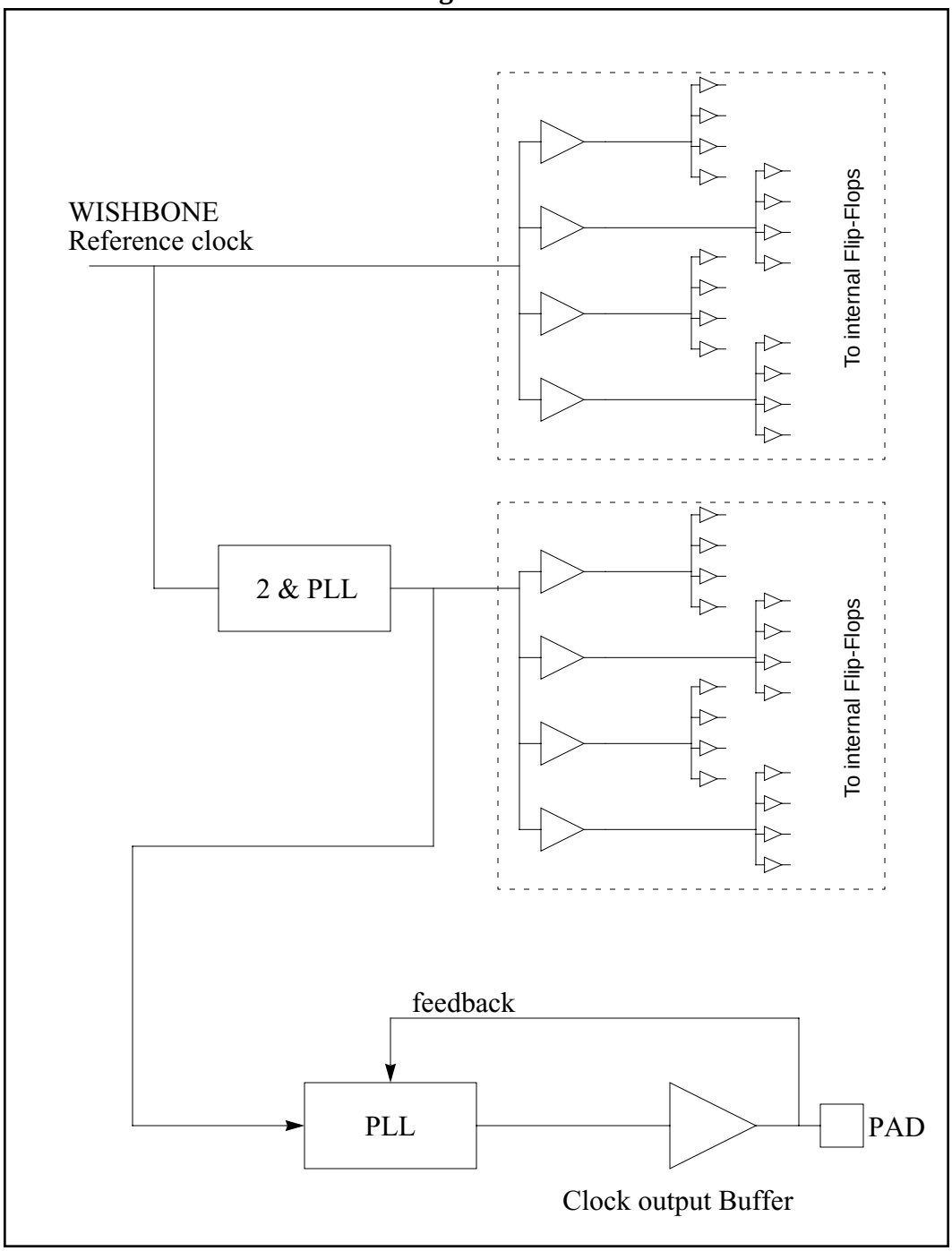
The same scenario applies for signals travelling the other direction (from clock domain 2 to clock domain 1). In that case, the delay between clock 1 relative to clock 2 starts to play an important role.

The same applies to the External Memory Clock. It must be phase synchronized with the internal Memory Clock.

Now that we understand the problem, we can design a proper clocking circuit for the memory controller.

1. Realistically a skew of less than Tcq-Tsu (of a typical Flip-Flop) should be sufficient. This should also include any clock jitters and clock tree imbalances.

Figure 37:



The figure above shows a conceptual view of the clock generation and distribution. In this example the Memory Clock is derived from the WISHBONE clock, divided by two and then phase synchronized with the WISHBONE clock. The clock output for external memories utilizes a PLL to compensate for the Output Buffer delay and synchronizing the phase of the external clock to the internal memory clock.