**Modular Decompression System**

**Top Architecture**

**Top Architecture, Ver. 12**

Contents

[TOP Scheme 3](#_Toc291001812)

[General Rules 4](#_Toc291001813)

[Type Register Values 4](#_Toc291001814)

[Individual Blocks 4](#_Toc291001815)

[Memory Block 4](#_Toc291001816)

[Registers in Memory Block 4](#_Toc291001817)

[Write Target 5](#_Toc291001818)

[RAMs in the Memory Block 5](#_Toc291001819)

[SDRAM Bank Select 5](#_Toc291001820)

[Write and Read Counters 5](#_Toc291001821)

[Write to SDRAM 5](#_Toc291001822)

[Read from SDRAM 6](#_Toc291001823)

[SDRAM Arbiter 7](#_Toc291001824)

[Image Block 8](#_Toc291001825)

[Registers in the Image Block 8](#_Toc291001826)

[Receive Data 8](#_Toc291001827)

[Synthetic Picture Generator 9](#_Toc291001828)

[TX Block 10](#_Toc291001829)

[Registers in the TX Block 10](#_Toc291001830)

[Operation 10](#_Toc291001831)

[UART Transmission 11](#_Toc291001832)

[RX Block 12](#_Toc291001833)

[Write to Registers 12](#_Toc291001834)

[Internal Registers in RX Block 12](#_Toc291001835)

[Interconnects 13](#_Toc291001836)

[INTERCON (Z) 13](#_Toc291001837)

[INTERCON (Y) 13](#_Toc291001838)

[INTERCON (X) 13](#_Toc291001839)

[Wishbone Cycles 14](#_Toc291001840)

[Wishbone BURST (Block) Read Transaction 14](#_Toc291001841)

[Burst Read Transaction – Start of Cycle 15](#_Toc291001842)

[Burst Read Transaction – End of Cycle 16](#_Toc291001843)

[Wishbone BURST (Block) Write Transaction 17](#_Toc291001844)

[Burst Write Transaction – Start of Cycle 18](#_Toc291001845)

[Burst Write Transaction – End of Cycle 19](#_Toc291001846)

[Wishbone ERR\_O Signal 19](#_Toc291001847)

# TOP Scheme

* **WBM –** Wishbone Master
* **WBS –** Wishbone Slave

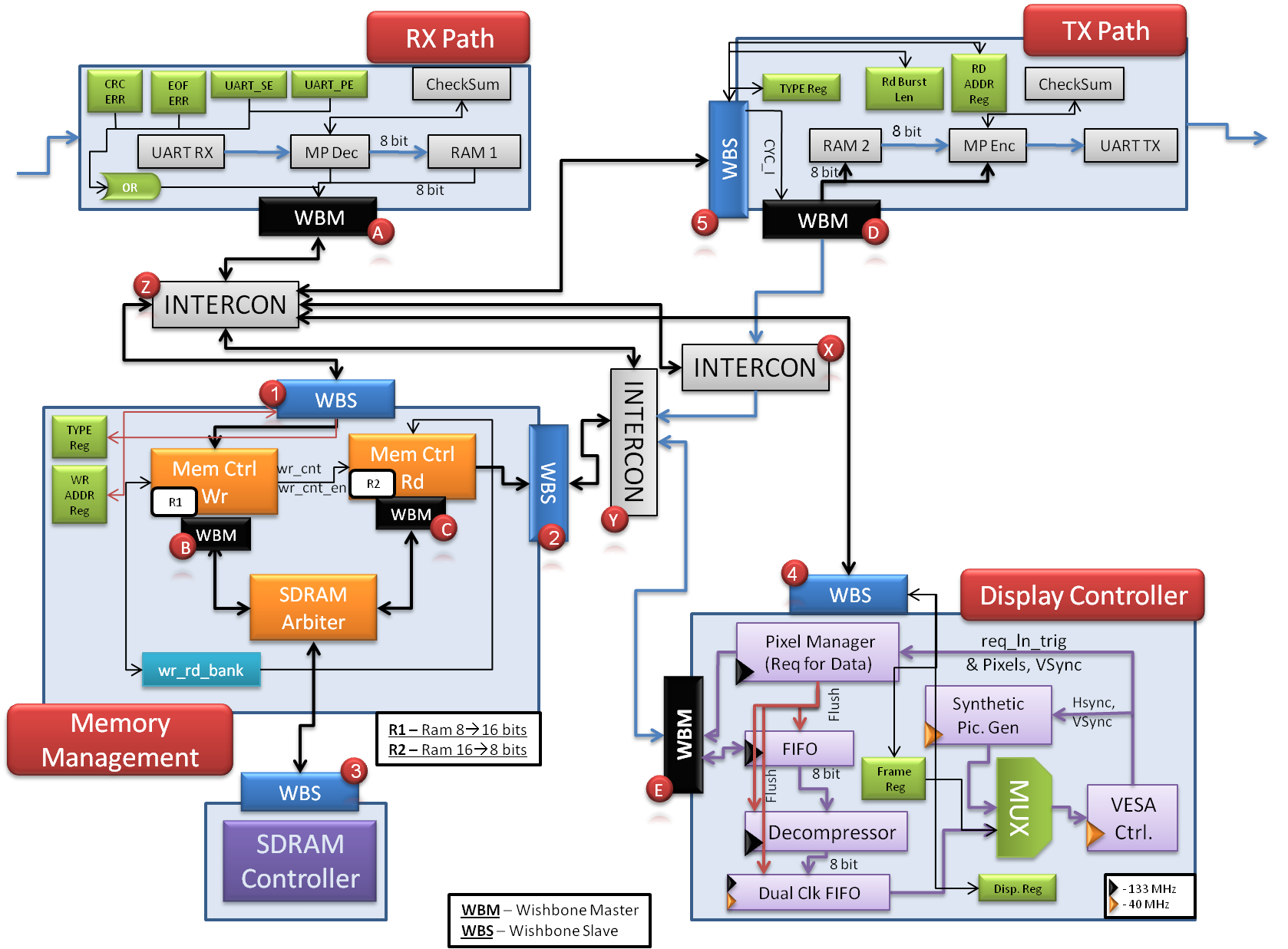


Figure 1– Top Scheme

# General Rules

* All Wishbone Slaves MUST assert STALL\_O, unless they are addressed to.
* TYPE Register is the only register, which exist in more than one block. When writing to them, the main WMB (A) should wait for the ACK\_I signal from only one block, which will be chosen by the INTERCON (Z).

# Type Register Values

* Type\_register [0]: '0' for Normal Mode, '1' for Debug Mode.
* Type\_register [1]: '0' for Image Transaction, '1' for Summary Transaction.
* Type\_register [2]: Displayed image from VESA generator: '0' for Image Transaction (From SDRAM), '1' for Synthetic Pattern Generator.
* Type\_Register [7]: '0' for Data Transmission, '1' for Registers Transmission.

# Individual Blocks

## Memory Block

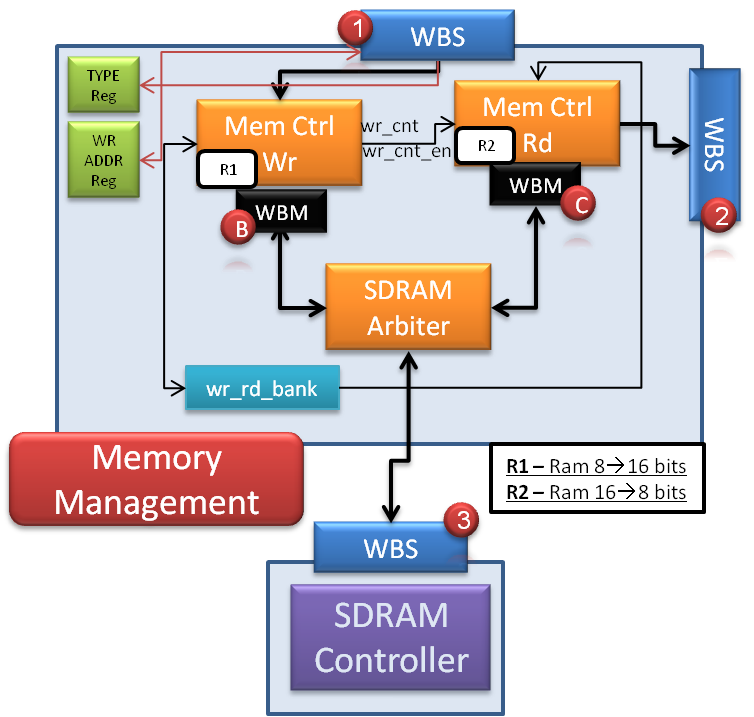


Figure 2– Memory Block

### Registers in Memory Block

The following registers will be placed in the memory block:

(W.O = Write Only, R/W = Read and Write)

* **Type Register** – Message type [W.O]
* **Address Register** – Start SDRAM address to write to / read from [R/W]

### Write Target

WBS (1) receives a TGC\_I signal, which is the data's target:

* **TGC\_I = '1'**: Write to block's registers
* **TGC\_I = '0'**: Write to block's components

### RAMs in the Memory Block

*Mem Ctrl Wr* has an 8 bits input (from RX Path) and 16 bits output (to SDRAM) RAM.

*Mem Ctrl Rd* has a 16 bits input (from SDRAM) and 8 bits output (to TX path) RAM.

### SDRAM Bank Select

The block named *wr\_rd\_bank­* hasan internal register, which its values are '0' or '1'.

* **When '0'**: *Mem Ctrl Wr* will receive the value of '00' (Write to Bank 0 and 1), and *Mem Ctrl Rd* will receive the value of '10' (Read from Bank 2 and 3).
* **When '1'**: *Mem Ctrl Wr* will receive the value of '10' (Write to Bank 2 and 3), and *Mem Ctrl Rd* will receive the value of '00' (Read from Bank 0 and 1).

When the *Type Register*'s value in the memory block indicates that the total sent bytes to the SDRAM is being received, *Mem Ctrl Wr* compares the received amount of received bytes with the number of bytes that it wrote to the SDRAM (Total amount =*wr\_cnt)*. In case *wr\_cnt* is equal to that number, *Mem Ctrl Wr* will signal the *wr\_rd\_bank*, and the above values will switch ('00'↔'01'), so the next image read data from SDRAM will be the last correct received image.

### Write and Read Counters

*Mem Ctrl Wr* shall have an internal register, named *wr\_cnt.*

*Mem Ctrl* Rd shall have an internal register, named *rd\_cnt.* Its initial value will be 0, so as long as the first correct frame will not be received, no data will be read from the SDRAM to the Image Block.

Details about these two register is available in the *Write to SDRAM* and *Read from SDRAM* chapters.

### Write to SDRAM

***Mem Ctrl Wr***: WBM (A) initiate a memory write, with burst length, which is the amount of the current available words in the *Mem Ctrl Wr*'s RAM (which is Message Pack Decoder's LENGTH/2). Burst length will be transmitted to the SDRAM [WBM (B) to WBS (3)] using TGA\_O [8..0], which can represent the numbers 0🡪255 (burst length of 1🡪256 words). This burst will be executed more than one time, when the write burst length is greater than 255 (=256 bytes), which is the default burst value.

*Mem Ctrl Wr* shall contain the following internal registers:

1. ***cur\_wr\_addr[21..0]***, which holds the next address, which data should be written in the SDRAM, and it will write data from that address. This register will increment its value.
2. ***wr\_cnt*** (integer range 0🡪640X480 - 1), which holds the number of written words (16 bits) to the SDRAM. Each time a new image transaction will be written to the SDRAM, *rd\_cnt* will be set to *wr\_cnt,* and *wr\_cnt* counter will be set to 0.

Burst operation algorithm:

1. *avail :=* (256 – cur\_wr\_addr[7..0])
2. *blen :=* Required Burst Size, according to current available words in RAM1
3. *Loop while blen > 0*
   1. Execute Write Burst to SDRAM, where burst length is *(blen* ***mod*** *avail*)
   2. *wr\_cnt := wr\_cnt + (blen* ***mod*** *avail*)
   3. *cur\_wr\_addr := cur\_wr\_addr + (blen* ***mod*** *avail*)
   4. *avail :=* 256
   5. *blen :=* (blen **div***avail)*
4. In case the system is working in debug mode, *rd\_cnt* value will be set to *wr\_cnt*.

### Read from SDRAM

***Mem Ctrl Rd***: WBM (D) or WBM (E) initiate a memory read, which will cause WBS (2) to initiate a memory read from SDRAM, through WBM(C) to WBS(3), where TGA\_O [8..0] represents the burst length. This burst will be executed more than one time, when the Read Burst Length Register (See TX Block) is greater than 255 (=256 bytes).

See *Write to SDRAM* for more information.

*Mem Ctrl* Rd shall contain the following internal registers:

1. ***cur\_rd\_addr[21..0]***, which holds the next address, which data should be read from the SDRAM, and it will read data from that address. This register will increment its value at each burst, and will be zeroed when *cur\_rd\_addr*[19..0] = *rd\_cnt*\_i*,* which is the value of *rd\_cnt* at the beginning of the read transaction. *cur\_rd\_addr[21..20]* value will be set to the value, transmitted from *wr\_rd\_bank* when *cur\_rd\_addr*[19..0] = *rd\_cnt*\_i.
2. ***rd\_cnt*** (integer range 0🡪640X480 - 1), which holds the number of written words (16 bits) to the SDRAM in the last completed transaction. When *cur\_rd\_addr*[19..0] = *rd\_cnt*\_i*,* rd\_cnt\_ivalue will be set to *rd\_cnt*.

Burst operation algorithm:

1. *avail :=* (256 – cur\_wr\_addr[7..0])
2. *blen :=* Required Burst Size: Burst Length Register's value
3. *Loop while blen > 0*
   1. Execute Read Burst from SDRAM, where burst length is *(blen* ***mod*** *avail*)
   2. *rd\_cnt := rd\_cnt - (blen* ***mod*** *avail*)
   3. *cur\_rd\_addr := cur\_rd\_addr + (blen* ***mod*** *avail*)
   4. *avail :=* 256
   5. *blen :=* (blen **div***avail)*
4. In case the system is working in debug mode, *rd\_cnt\_i* value will be set to *rd\_cnt* each clock.

### SDRAM Arbiter

Internal MUX will switch between WBM (B) and WBM (C). Two CYC\_I inputs will be received from the *Mem Ctrl Rd* and *Mem Ctrl Wr*.

* When *Mem Ctrl Rd* demands control on the SDRAM (using CYC\_O), in case SDRAM is not busy in a write transaction, *rd\_gnt* will be asserted, until CYC\_I will be negated.
* When *Mem Ctrl* Wr demands control on the SDRAM (using CYC\_O), in case SDRAM is not busy in a read transaction, *wr\_gnt* will be asserted, until CYC\_I will be negated.
* A high priority is given to *Mem Ctrl Rd*. In case both *Mem Ctrl Rd* and *Mem Ctrl Wr* demands control on SDRAM at the same time – *rd\_gnt* will be asserted.

## Image Block

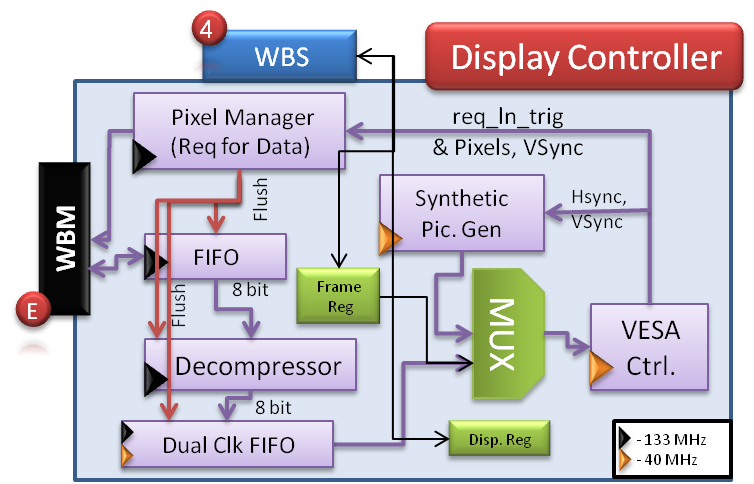


Figure 3– Image Block

The image block transmits the data from the SDRAM / Synthetic Picture Generator, through the DAC on the DE2, to the VGA.

The FIFO's depth is 3,840 (X8 bits).

The Dual Clock FIFO's depth is 640X6 = 3,840 (X8 bits), for 6 lines.

### Registers in the Image Block

The following registers will be placed in the image block:

(W.O = Write Only, R/W = Read and Write)

* **Display Register** – Display Synthetic image / Image from SDRAM [R/W]
* **Frame Register** – X and Y coordinates of the frame [R/W]

### Receive Data

Data transfer is being initialized by the *Pixel Manager*. The burst length will ALWAYS be 256 read cycles. This block has a counter, which holds the current number of received pixels. For example: Suppose that one WORD that has been received is 0xFF03, where 0xFF is the received color and the 0x03 is the repetition + 1. For that transaction, the counter will add '4decimal' to its current value. When the counter's value is *Active Horizontal Pixels X Active Vertical Lines* (i.e.: 640X480 = 307,200 pixels), the WBM is expected to receive [ERR\_I] with [TGD\_I()] from the WBS (3) of the memory block.

There is one exception: In case the last SDRAM transaction, which contains all the end-of-picture pixels, is exactly 512 Bytes. In that case, the transaction will terminate normally ([ACK\_I]), and the same data tag information [TGD\_I()] will be received.

In case of error, which means that the [TGD\_I()]'s value is not as expected, the Dual Clock FIFO will be flashed and the Runlen Decompressor will return to its initial value. New read transaction will start after the *Pixel Manager* will receive *VSync* signal from the SDRAM Controller.

### Synthetic Picture Generator

According to the Display Register value, an image from the SDRAM will be displayed on the screen, or a synthetic image, generated by the *Synthetic Picture Generator*.

## TX Block

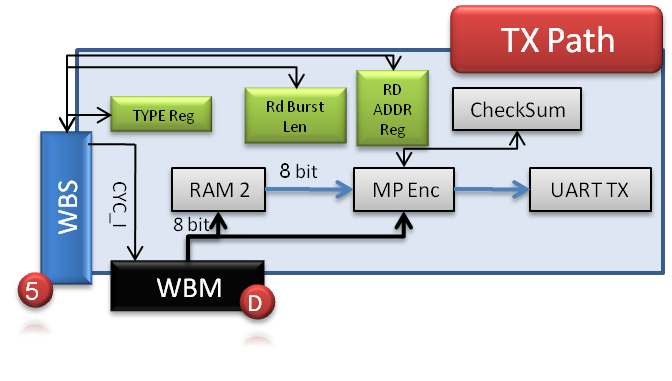


Figure 4– TX Block

The TX Block transmits data for debug purposes, using UART protocol, to the host (Matlab, which is running on a PC).

### Registers in the TX Block

The following registers will be placed in the TX block:

(W.O = Write Only, R/W = Read and Write)

* **Type Register** – Message type [W.O]
* **Read Burst Length Register** – Number of words to read from SDRAM, in debug mode [R/W]
* **Read Address Register** – Read address from SDRAM / Registers [W.O]

### Operation

In case of request, from Matlab, from SDRAM / Register, the WBM (A) of RX block will initiate a write transaction to the TX Block, through WBS (5), to command it to initiate a read transaction, which will be according to the Type Register value: transmit data from the SDRAM or registers. The SDRAM address / register will be determined according to the Read Address Register. INTERCON (Y) will decide who will be the active data path, according to the TGC\_O indication from TX Block, which is derived from the Type Register. When WBS(5) CYC\_I will indicate of end of write to registers operation, and start of data transaction, it will command the WBM(D) to start data transmission. The following steps will be executed:

#### Read from SDRAM

WBS (5) will command WBM (D) to initialize a read transaction. SDRAM read transaction will be initiated, through the INTERCON (X) and INTERCON (Y). Read address will be dictated from Read Address Register. Burst length will be dictated from the Burst Length register.

#### Read from Register Block

WBS (5) will command to WBM (D) to initialize a read transaction. Registers read transaction will be initiated, through the INTERCON (X), and through the INTERCON (Z). The required read register will be determined by the ADR\_O[7..0] signal.

### UART Transmission

The TX Block will initiate a UART transmission to the Matlab, with the following parameters:

1. SOF
2. TYPE – Type of data
3. Address – Read address to SDRAM / Read register address
4. Data Length
5. Data – Data read form SDRAM / Registers.
6. Checksum
7. EOF

## RX Block

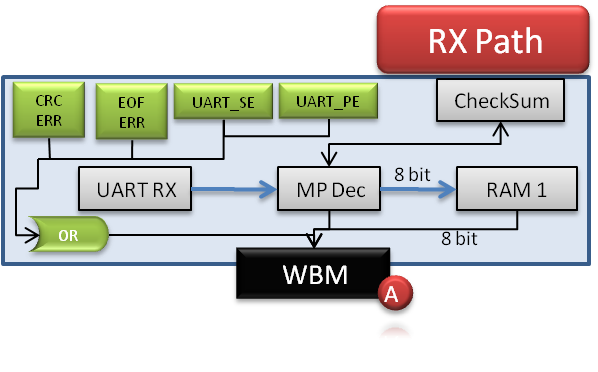


Figure 5– RX Block

When a transmission is being received to the RX block, it is being unwrapped by the Message Pack Decoder. As soon as a full message (EOF has been received) has been received, the *mp\_done* signal will be asserted. Write transaction is being initialized by the RX's WBM (A) through the INTERCON (Z) to the required blocks if the following terms are satisfied:

1. CRC\_ERR = '0' (CRC Error)
2. EOF\_ERR = '0' (EOF has not been received)
3. UART\_SE = '0' (UART Stop Bit Error)
4. UART\_PE = '0' (UART Parity Bit Error)

Two writes modes are possible:

* TGC\_O = '1': Write to Registers
* TGC\_O = '0': Write data to blocks

### Write to Registers

When TGC\_O = '1': Write to Registers, the addressed register, which is determined by ADR\_I [8..0] should receive and save the received DAT\_I [8..0] value.

When TGC\_O = '0': Write data to blocks, the addressed block, which is determined by ADR\_I [8..0] should receive and store the data in the required target (SDRAM, FIFO, etc…)

### Internal Registers in RX Block

The following internal, non readable by other blocks, registers will be placed in the RX Block

* **CRC Error** – CRC error has been detected in the RX path for the current message – Clear on Read Register.
* **EOF Error** – EOF error has been detected in the RX path for the current message – Clear on Read Register.
* **UART Stop Bit Error** – Stop bit has not been received in UART RX.
* **UART Parity Error** – Parity bit error has been detected in UART RX.

# Interconnects

Figure 6– Interconnects

### INTERCON (Z)

Two Wishbone Masters are connected to this interconnection:

1. WBM (A) from RX Path, which responsible to write data (SDRAM / Registers) to Wishbone Slaves.
2. WBM (D) from TX Path, which responsible to read data (SDRAM / Registers) from Wishbone Slaves.

WBM (A) has a higher priority. WBMs take control on the INTERCON (Z) by raising the CYC\_O signal. When one WBM grants control on the INTERCON, the INTERCON transmit STALL\_O to the inactive WBM.

When writing to TYPE register, more than one block will negate the STALL\_O signal. In this case, higher priority will be given to the Memory Block. The consequence will be that only the memory block's ACK\_O will be latched by the WBM (A).

### INTERCON (Y)

This interconnection receives the TYPE register, which is transmitted from the WBM (A). In case of debug mode, WBM (D) will be connected to the path. In case of normal mode, WBM (E) will be connected to the path.

#### Registers in the INTERCON (Y)

The following registers will be placed the INTERCON (Y), and will not be shown in the scheme, because of lack of place in the scheme:

(W.O = Write Only)

* **Type Register** – Message type [W.O]

### INTERCON (X)

According to WBM (D) command (TGC\_I), routes the path toward reading from SDRAM in the Memory Block or from the registers, through INTERCON (Z):

* **TGC\_I = '1'**: Route path to Memory Block
* **TGC\_I = '0'**: Route path to INTERCON (Z)

# Wishbone Cycles

This project uses Read and Write burst, initiated by the Wishbone Master.

* STALL\_I will be use especially for SDRAM (until data is valid: Time of RAS, CAS…) and in the arbiters in the design.
* Wishbone Pipeline Classic Cycle only will be used here.

## Wishbone BURST (Block) Read Transaction

* In case STALL\_I is asserted, WBM will present the same signals' values as the previous clock edge.
* In case of ERR\_I, skip to step 'Clock edge N+1'. TGD\_I tag will contain the error reason.

**Clock Edge 0**

* MASTER negates WE\_O, asserts CYC\_O, STB\_O and places valid value in ADR\_O, TGA\_O, TGC\_O and SEL\_O

**Clock Edge 1, 2, 3…. N-1, where N is burst size**

* MASTER places new valid values in ADR\_O and TGA\_O
* SLAVE presents valid data on DAT\_I, and asserts ACK\_I

**Clock Edge N**

* MASTER places new valid values in ADR\_O and TGA\_O
* SLAVE presents valid data on DAT\_I, and asserts ACK\_I (Which is already asserted)

**Clock Edge N** **+1**

* MASTER negates STB\_O
* SLAVE presents valid data on DAT\_I, and asserts ACK\_I (Which is already asserted)

**Clock Edge N** **+2**

* MASTER negates CYC\_O
* Slave negates ACK\_I

### Burst Read Transaction – Start of Cycle

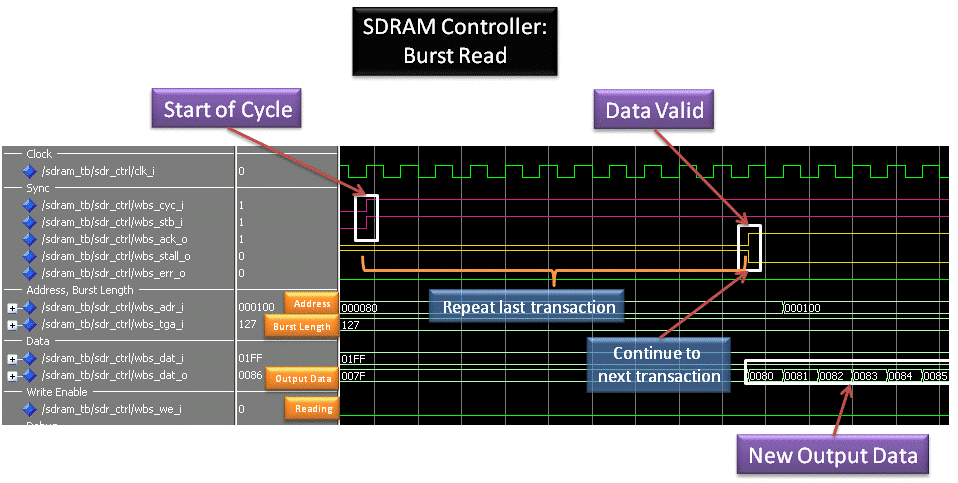


Figure 7 - Start Burst Read Cycle

**Notes:**

* + **WBS\_ADR\_I** is being changed during the transaction. In the above example it has no meaning, since the SDRAM latches the WBA\_ADR\_I at the first WBS\_ACK\_O transaction, and auto-increment it in an internal logic. The WBS\_ADR\_I change here is for the next cycle, which is not shown in this wave. This is the same behavior as WBS\_CTI\_I = '010' (Incrementing burst cycle), where the address is MAY not be incremented, since it is auto-incremented in the SDRAM controller. Note that in the Wishbone SPEC – the address MUST be incremented. Therefore – the SDRAM controller does not fully supports Wishbone Standard.

### Burst Read Transaction – End of Cycle

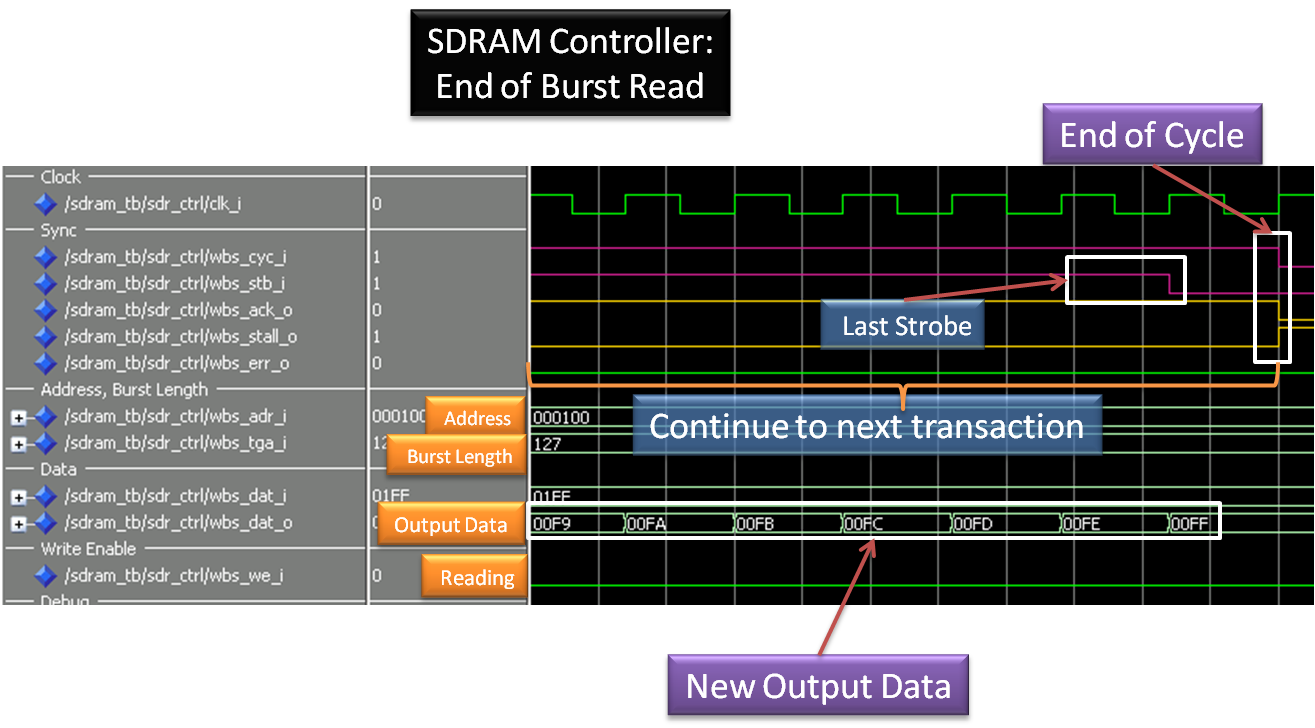


Figure 8 - End Burst Read Cycle

**Notes:**

* Reminder: WBS\_CYC\_I is being negated as soon as the number of transmitted WBS\_ACK\_O is equal to the number of received WBS\_STB\_I, therefore WBS\_CYC\_I negates together with WBS\_ACK\_I negation.

## Wishbone BURST (Block) Write Transaction

* In case STALL\_I is asserted, WBM will present the same signals' values as the previous clock edge.

**Clock Edge 0**

* MASTER asserts WE\_O, asserts CYC\_O, STB\_O and places valid value in DAT\_O, ADR\_O, TGA\_O, TGC\_O and SEL\_O

**Clock Edge 1, 2, 3…. N-2, where N is burst size**

* MASTER places new valid values in DAT\_O, ADR\_O and TGA\_O
* SLAVE asserts ACK\_I

**Clock Edge N** **- 1**

* MASTER places new valid values in DAT\_O, ADR\_O and TGA\_O
* SLAVE asserts ACK\_I (Which is already asserted)

**Clock Edge N**

* MASTER negates STB\_O
* SLAVE asserts ACK\_I (Which is already asserted)

**Clock Edge N** **+1**

* MASTER negates CYC\_O
* SLAVE negates ACK\_I

### Burst Write Transaction – Start of Cycle

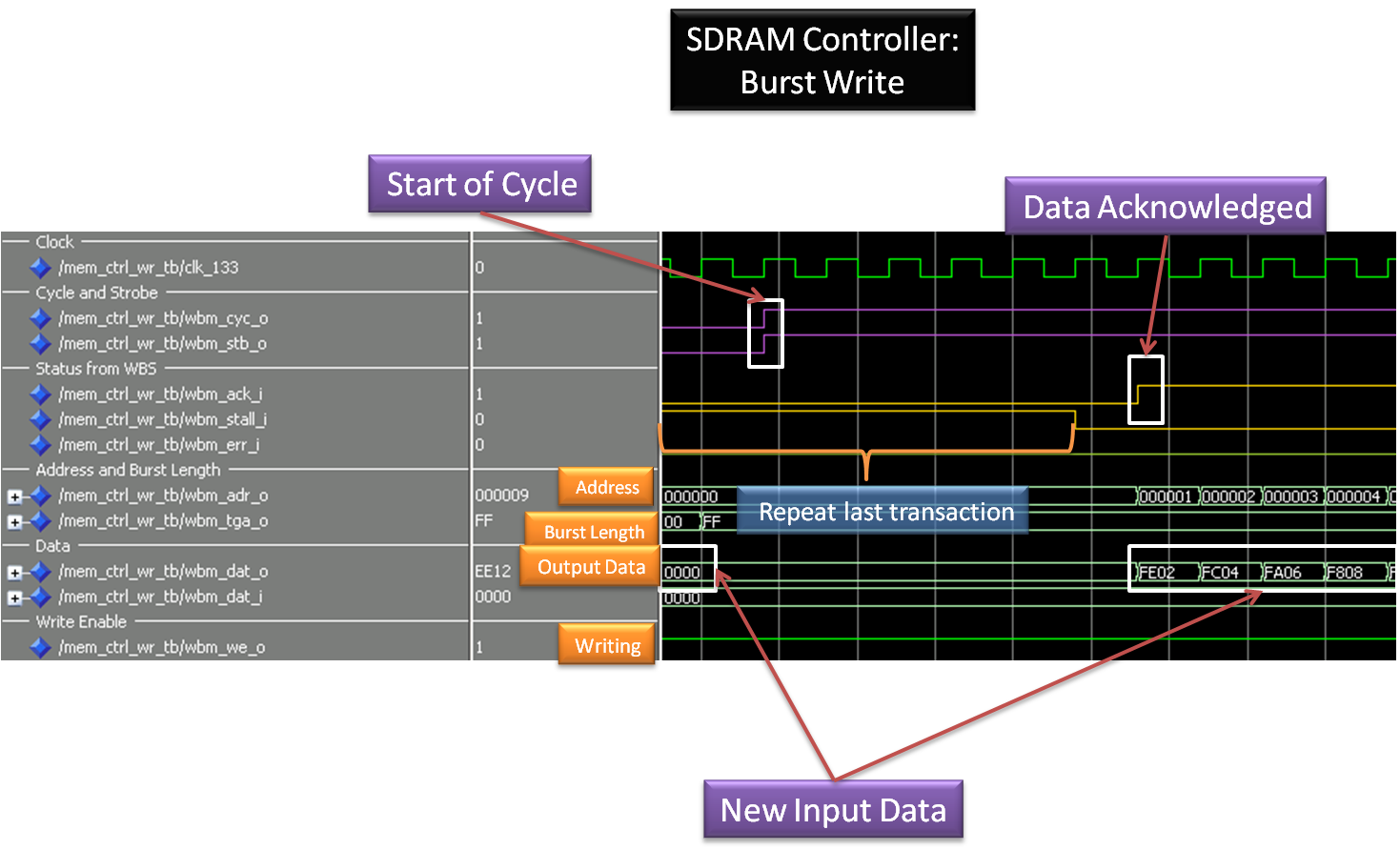


Figure 9 - Start Burst Write Cycle

### Burst Write Transaction – End of Cycle

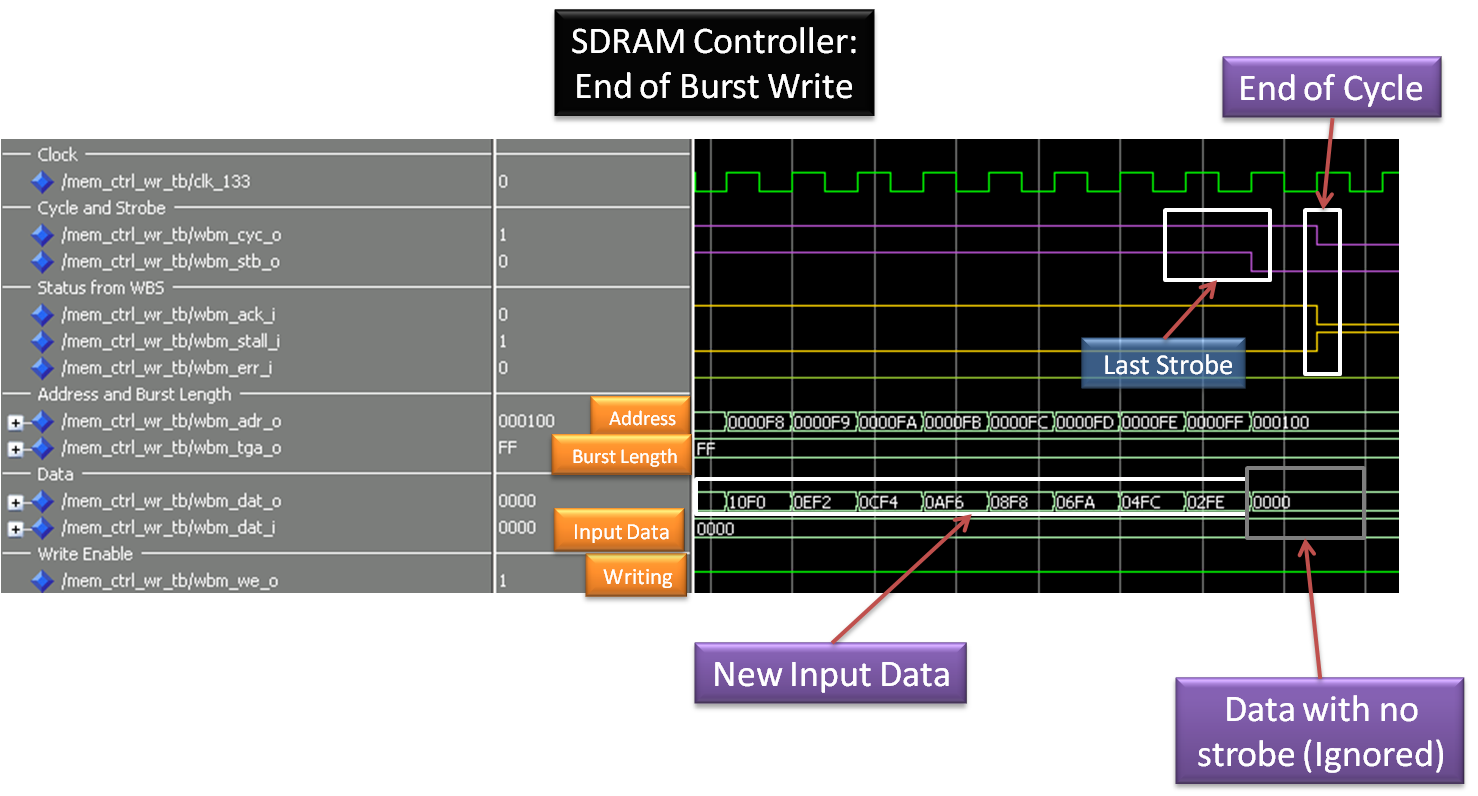


Figure 10 – End Burst Write Cycle

## Wishbone ERR\_O Signal

When CYC\_I and STB\_I are asserted, but there is no more data to be supplied by the Wishbone Slave, the Wishbone Slave negated ACK\_O and asserts ERR\_O, to signal the Wishbone Master that an error has occurred.

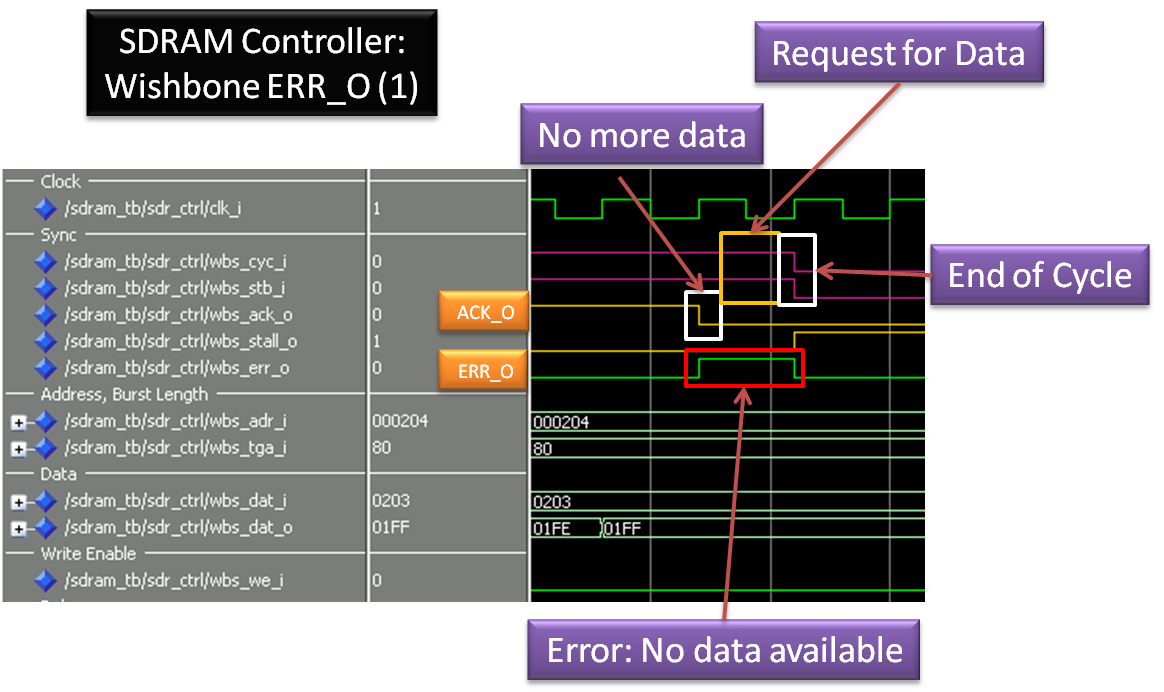


Figure 11 – ERR\_O (1) – Read Cycle

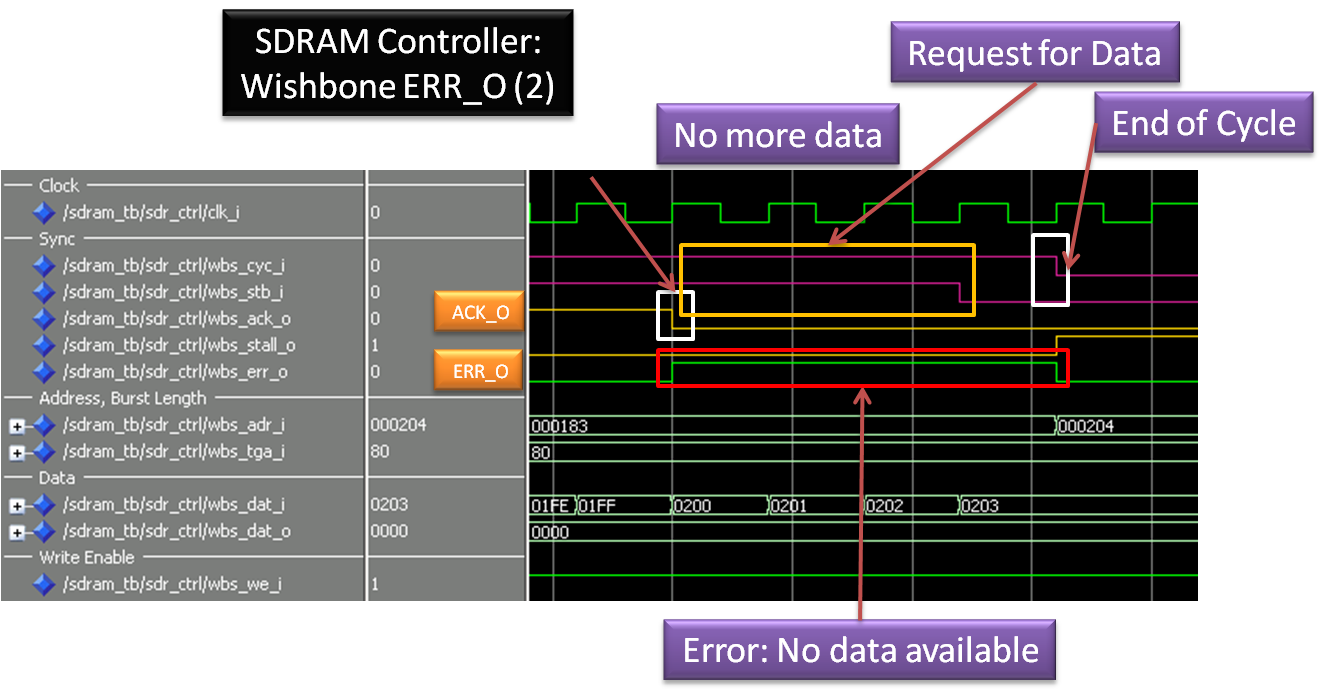


Figure 12 – ERR\_O (2) – Write Cycle