HyperRAM Controller Manual

Xavier Carril

BSC-CNS

April 2020

1 Introduction

To access HyperRAM memory, we will need a controller to make the core interface and HyperRAM understand each other. Below, in Table 1, we have the connections of the HyperRAM controller to the core interface and in Table 2 we have the connections to the HyperRAM module. In the following sections, we will explain how the controller need to be managed to make a correct write/read accesses.

Pin	**Direction**	**Description**
reset	Input	Reset Signal.
clk	Input	Core clock. Actual DRAM clock will be this Div-4.
rd_req	Input	When core not busy, assert 1 clock to make read request.
wr_req	Input	When core not busy, assert 1 clock to make write request.
mem_or_reg	Input	0=DRAM Memory. 1=Configuration Register
wr_byte_en (4 bits)	Input	0xF=Write all 4 bytes. 0xE write Bytes 3-1 but not 0.
rd_num_dwords (22 bits)	Input	Number of dwords to read, example 0x01.
addr (32 bits)	Input	32bit byte (not dword) address for DRAM cell.
wr_d (32 bits)	Input	32bit Write Data to DRAM.
rd_d (32 bits)	Output	32bit Read Data from DRAM.
rd_rdy	Output	Read Ready Strobe. Asserts 1 clock when *rd_d* is valid.
busy	Output	Busy Strobe asserts when Read or Write cycle is busy.
burst_wr_rdy	Output	Asserts when ready to accept next *wr_req* for burst.
latency_1x	Input	Initial latency edges when the memory is configured when depends on RWDS during CA cycles (CR0[3] bit configuration == 0).
latency_2x	Input	Initial latency edges. It should be *latency_1x* * 2.

Table 1: Core interface connections

Pin	Direction	Description
dram_dq_in (8 bits)	Input	Raw data from DRAM.
dram_dq_out (8 bits)	Output	Raw data to DRAM.
dram_dq_oe_l	Output	Data output enable. 0= Select dram_dq_out 1= Select dram_dq_in
dram_rwds_in	Input	Read Write Data Strobe Input.
dram_rwds_out	Output	Read Write Data Strobe Output.
dram_rwds_oe_l	Output	RWDS output enable. 0= Select dram_rwds_out 1= Select dram_rwds_in
dram_ck	Output	Clock to DRAM. Clock core divided by 4
dram_rst_l	Output	Reset signal negated.
dram_cs_l	Output	Chip select signal negated.

Table 2: HyperRAM interface connections

It is necessary to say that to power up the HyperRAM module, the reset signal should be up to 150μs(t_{VCS}) as we can see in Figure 1. About the Chip Select signal (CS#), we should not worry because it is handled by the controller.

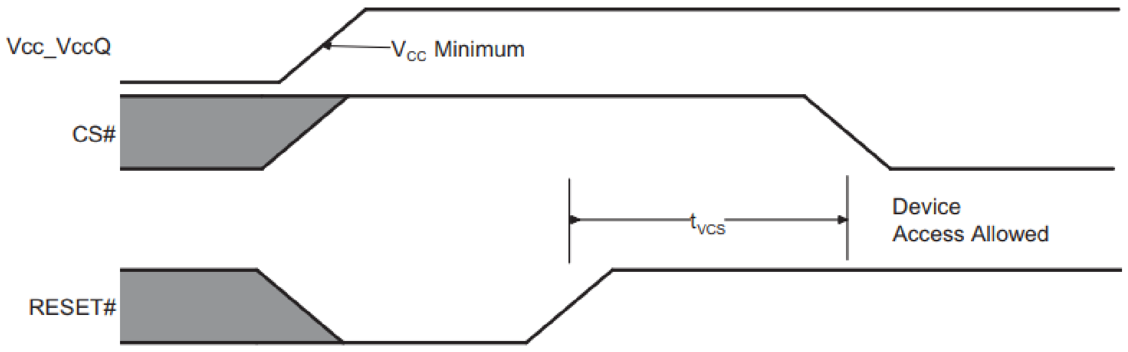

Figure 1: Power-up waveform

2 Single Module

First, we will start making transactions with a single memory module. In this document, we suppose that we are working with the Cypress S27KS0641 model, a 64Mb high-speed CMOS 1.8 Volt Core HyperRAM that operates at 166MHz as maximum. Also, there exists another model of 128Mb, where its range of address is more extensive.

2.1 Read Transaction

In this section, we will explain how to manage the controller in order to make a read transaction. If you want to know how the controller manipulates the HyperRAM controller, please visit the HyperRAM datasheet at: https://www.cypress.com/file/183506/download

2.1.1 Simple Read

If we want to make a read access, the HyperRAM controller will need the address and the number of double words that we want to read, that in this case will be 1 word (32 bits). In all of these accesses, always we will read 4 bytes. Below, we have the waveform that a read access should achieve.

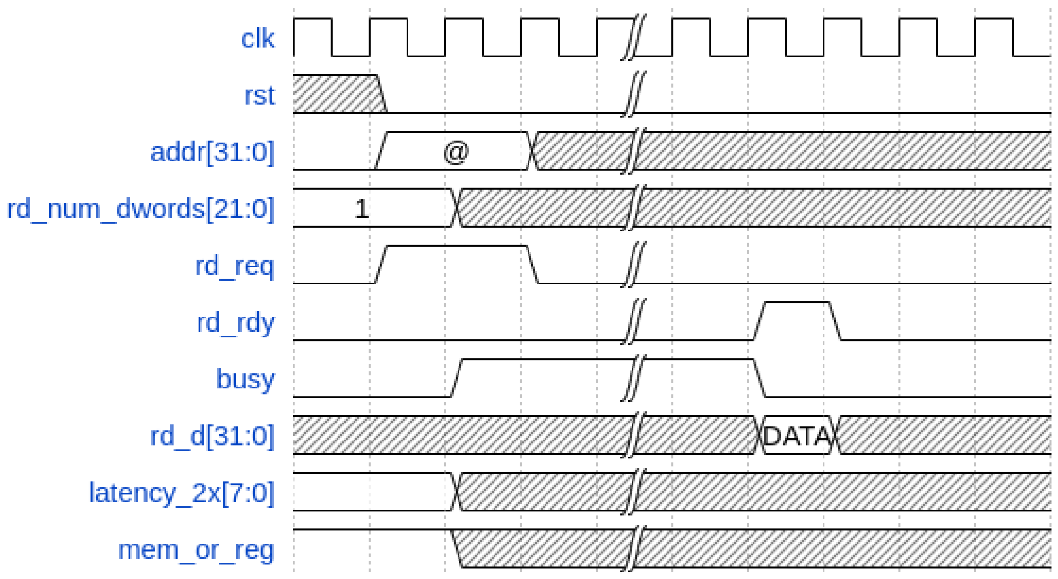

Figure 2: Read transaction waveform

Before to emit a read request, we have to be sure that *busy* signal is low, and the signals *addr*, *rd_num_dwords*, *latency_2x* and *mem_or_reg* are established. The requisites to establish these signals are the following:

- **addr:** The range in a module of 64Mb has to be between 0x00000 and 0x7fffff. In a module of 128Mb has to be between 0x00000 and 0xffffff.
- **rd_num_dwords:** As we are doing a simple read, only we want to read one double word. So the value of this signal will be 1.
- **latency_2x:** In read transactions, it does not matter its value.
- **mem_or_reg:** As we are accessing a memory cell, the value of this signal has to be 1.

Once we are in this situation, we will start to emit the read request, pulling up the *rd_req* signal until the *busy* signal is high. When the signal *rd_rdy* is high, the *busy* signal will be down, and the *rd_d* channel will contain the read data.

2.1.2 Burst Read

In order to read more than four consecutive bytes in one transaction, we have to specify the number of double words (4 bytes) in the *rd_num_dword* signal. Below, in Figure 3, we have an example waveform of a read burst transaction of 2 double words so that it will be ignored after the initial cycles. The address specified in *addr* will be incremented by 4 in each double read word and the number of pulses of the signal *rd_rdy* would be equal to the value of *rd_num_dword* signal.

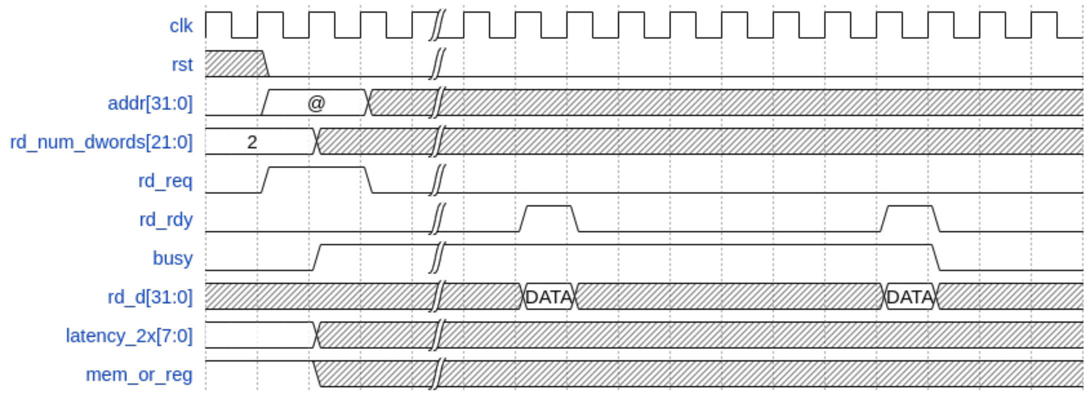

Figure 3: Read burst transaction waveform

2.1.3 Read access timing

First, we have to take into account that the maximum time that we can have the Chip Select down (t_{CSM}) is about 4μs, and the minimum time between transactions with Chip Select high (t_{CSHI}), it has to be 6μs if the clock goes to 166MHz[1]. So the time that we can achieve in one access[2], either simple or burst read is as follows:

$$Time\ Access = 7\ edges\ of\ send\ CA\ commands\ *\ t_{\mathrm{CRAM}}/2\ +$$

$$+12\ latency\ cycles * t_{\mathrm{CRAM}} +\ (2\ bytes * t_{\mathrm{CRAM}}) * nBurst \leq t_{\mathrm{CSM}}$$

t_{CRAM} is the time cycle of the DRAM clock (Core Clock Div-4).

2.2 Write Transaction

In this section, we will explain how to manage the controller in order to make a write transaction. If you want to know how the controller manipulates the HyperRAM controller, please visit the HyperRAM datasheet at : `https://www.cypress.com/file/183506/download`

[1]For other clock frequencies, please visit the Cypress HyperRAM datasheet.
[2]Time at *busy* signal is HIGH.

2.2.1 Simple Write

If we want to do a write access, the HyperRAM controller will want the address to access, the number of bytes to write, and the data to write. Below, we have the waveform that a write access should be accomplished.

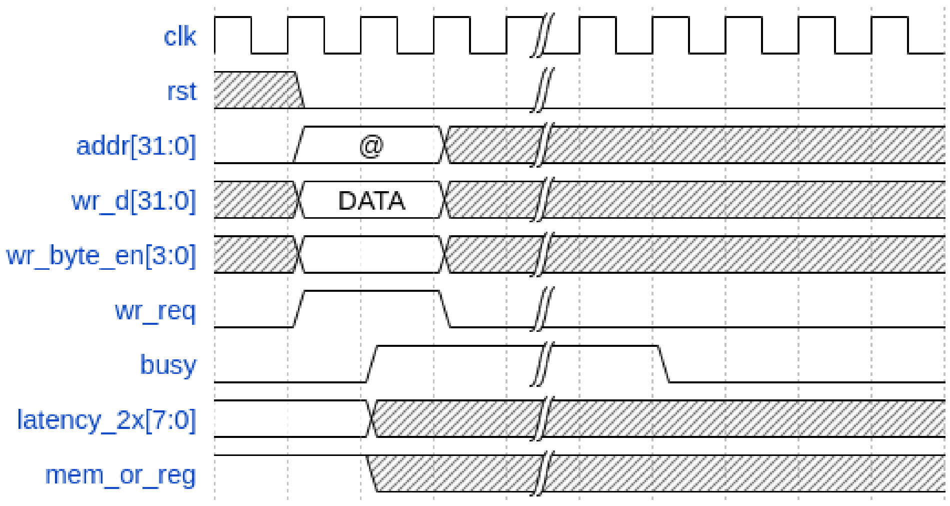

Figure 4: Write transaction waveform

Before to emit a write request, we have to be sure that *busy* signal is low, and the signals *addr*, *wr_d*, *wr_byte_en*, *latency_2x* and *mem_or_reg* are established. The requisites to establish these signals are the following:

- **addr:** The range in a module of 64Mb has to be between 0x00000 and 0x7fffff. In a module of 128Mb has to be between 0x00000 and 0xffffff.
- **wr_d:** This is a 32bit data channel.
- **wr_byte_en:** This is a mask to know which are the bytes to be written.
- **latency_2x:** If the memory is configured by default, the double initial latency should be: 6 cycles if set at 166MHz * (2 *latency_2x*) * (2 edges by each cycle) - 2 data alignment - 1 edge added in CA command timing = 21 edges.
- **mem_or_reg:** As we are accessing to a memory cell, the value of this signal has to be 1.

Once we are in this situation, we will start to emit the write request, pulling up the *wr_req* signal until the *busy* signal is high.

2.2.2 Burst Write

To write more than four consecutive bytes in one transaction, we have to send a write request with their respective data and mask, each time that the *burst_wr_rdy* signal is high. Below, in Figure 5, we have an example waveform of a write burst transaction of 2 double words. The address specified in *addr* signal will be incremented by 4 in each doubleword wrote so that it will be ignored after the initial cycles. Between the *burst_wr_rdy* pulse and the write request, a maximum of 5 cycles can pass.

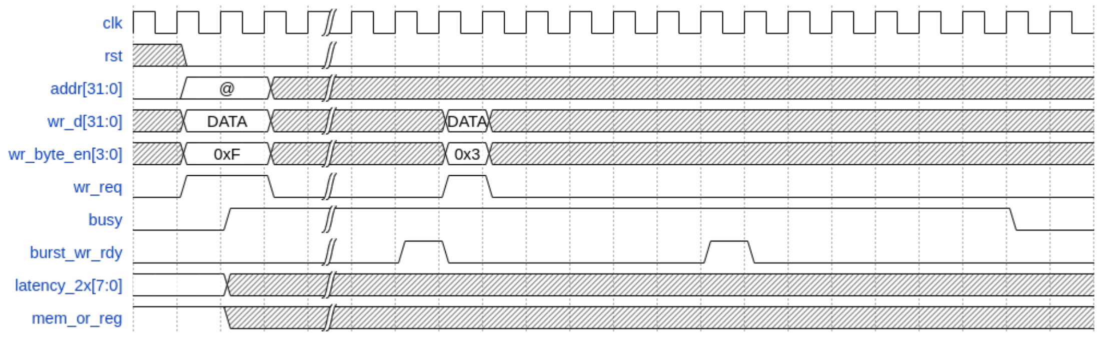

Figure 5: Read burst transaction waveform

2.2.3 Write access timing

First, we have to take into account that the maximum time that we can have the Chip Select down (t_{CSM}) is about 4μs, and the minimum time between transactions with Chip Select high (t_{CSHI}), it has to be 6μs if the clock goes to 166MHz[3]. So the time that we can achieve in one access[4], either simple or burst write is as follows:

$$Time\ Access = 7\ edges\ of\ send\ CA\ commands\ *\ t_{\mathrm{CRAM}}/2\ +$$
$$+\ latency\_2x\ edges * t_{\mathrm{CRAM}}/2 + \ (2\ bytes * t_{\mathrm{CRAM}}) * nBurst \leq t_{\mathrm{CSM}}$$

t_{CRAM} is the time cycle of the DRAM clock (Core Clock Div-4).

3 Multiple Modules

In case of having more than one memory module, the way to access them is not as tricky as we told before. The only thing to take into account is in which module we have to put down the signal Chip Select (CS). As we do sequential accesses, the best option is as shown in Figure 6. Supposing that we have four chip modules, the chip selector module works as a binary decoder, in which the signal *selCS* will have the value of 24 and 23 bit of the address to choose the corresponding module to access. All the signals, except the chip selector, are connected to all memory modules. Each one will not heed these signals if its corresponding Chip Select signal is not set to low.

[3]For other clock frequencies, please visit the Cypress HyperRAM datasheet.
[4]Time at *busy* signal is HIGH.

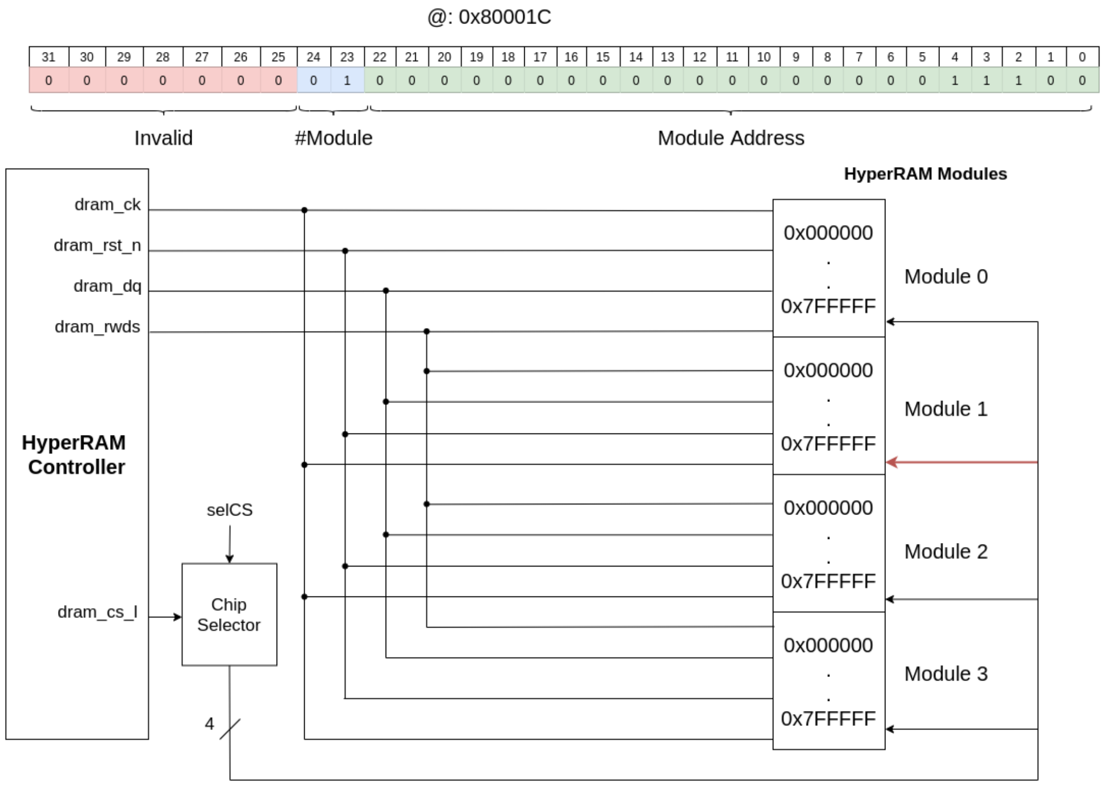

Figure 6: Access in address 0x80001C in multiple modules chips