

# **x32 DDR SDRAM Usage Guidance**

For Low Frequency Users

**Product Planning Team**  
**Memory Division**

## **Introduction**

The most of the advanced technology usually starts from where it is needed. It happens at the memory industry as well. For example, the memory used as a frame buffer at the graphic board, which is called graphic memory, has kept tracing the higher performance by both/either adapting a new specific function like 'block write' and/or increasing its operating frequency. The function 'block write' seems not to bring a great advantage any more because it is not advantageous in the area of 3D graphics and the operating frequency of the graphic memory is already high enough to ignore of the advantage of it. Since the 3D graphic function has been adapted to the graphic market, the operating frequency of the graphic memory has been the main focus to boost up the graphic performance. It looks like that it will continue for the time being.

The graphic memory has a x32 interface and is used for a point-to-point interface with high frequency such as SDR(Single Data Rate) SDRAM or DDR(Double Data Rate) SDRAM. The graphic memory has been mainly used at the graphic applications. Other applications like in DSC(Digital Still Camera), Digital STB(Set-Top Box), which need some buffer memories with x32 interface, have used the graphic memory as well. The usage of graphic memory at the consumer applications may be increasing dramatically because of the digitalization in the consumer applications. However, the dependence on the operating frequency in the area of the consumer applications is relatively lower than that in the graphic application. The x32 SDR SDRAM doesn't make any trouble to use it low frequency as long as all the AC parameters are kept. But x32 DDR SDRAM does.

This document is intended to give a solution to use x32 DDR SDRAM low frequency at the consumer applications.

## **DDR SDRAM will replace SDR SDRAM for the graphic memory**

The more graphic functions will be introduced to achieve the realism on display, the more performances of the graphic memory will be required. For a better and more realistic display on 3D applications like 3D games, lots of new rendering technologies have been and will be come out. For example, the more texture data will be required for the better looking and the more data will be needed for the better anti-aliasing, more than twice for FSAA(Full Screen Anti-Aliasing). Even do some graphic controllers already take Transform & Lighting function from CPU which is one of the complicated functions to compute. To make those wonderful technologies happened, the graphic memory should have higher frequency to deal with all the data needed at graphic controllers. The highest frequency of SDR SDRAM is 200MHz, 3.2GB/sec with 128bit data. It is not high enough to keep along with the new or even already-introduced technology for 3D

applications and the resolutions of display. Therefore, it is usually said that DDR SDRAM will gradually take the place of SDR SDRAM market, almost half of it by the year 2001 and all by the year 2004.

### **DDR SDRAM for the graphic memory is targeting 350MHz or even higher in 2002**

Already is there a requirement for 9.6GB/sec of the graphic memory performance with x128bit interface, which means that the graphic memory should work at 300MHz DDR. In 2002, the maximum frequency requirement on x32 DDR SDRAM may be 350MHz or higher to achieve 12GB/sec with 128bit interface. As far as the maximum frequency of the graphic memory is concerned, most of the major memory vendors are targeting 300MHz on x32 DDR SDRAM.

Beyond 2001, DDR SDRAM operating at the higher frequency, such as 400MHz, may be required as long as new graphic technologies continue to be introduced for the better looking display at 3D application.

### **DDR SDRAM targeting 300MHz or higher may not be able to support low frequency**

In order to achieve such a high frequency like 300MHz on x32 DDR SDRAM, one of the key adoptions is to take DLL(Delayed Locked Loop) in it to synchronize DQ & DQS signals to input clock. But, DLL logic has a limitation that it can not cover a wide frequency range. To achieve 300MHz DDR, it should be given up to support the operation at lower frequency, such as 66MHz and lower.

### **DDR SDRAM with non\_DLL will be the solution for low frequency application**

There might be two possible solutions to support the low frequency operation of the future x32 DDR SDRAM. One is to develop different x32 DDR SDRAM running at 66MHz and higher by targeting only for 200MHz. However, it may be a bit risky and the memory vendors may be hesitate to develop it because it's operation is not certain and it can't be a big market. The other is to support non\_DLL operation at the same chip targeting 300MHz. It looks like the most reasonable solution if the receiver on the memory controller can handle the data from DDR SDRAM with non\_DLL.

### **Nothing different between non\_DLL and DLL except $tAC^1/tDQSCK^2$**

To provide the easiest way for memory controller to support both DLL and non\_DLL on the same chip, all the AC timings except  $tAC/tDQSCK$  will be kept. See the following table, which

---

<sup>1</sup> .  $tAC$  : Output Access Time from Clock

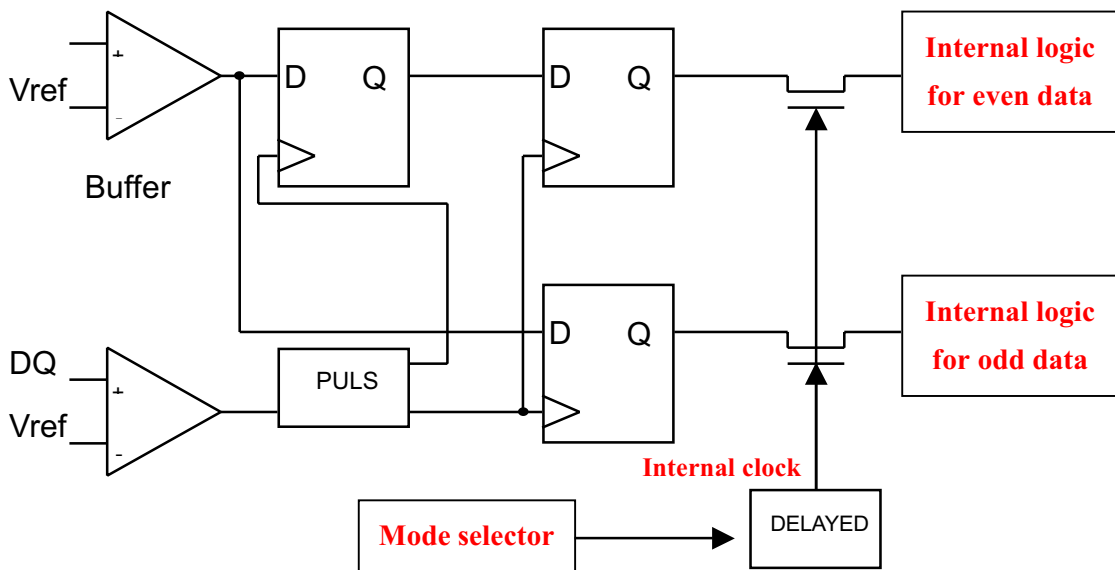
<sup>2</sup> .  $tDQSCK$  : DQS Output Access Time from Clock

is based on 2Mx32 DDR A-die 166MHz.

Symbol	Parameter	DLL mode	Non_DLL mode
TAC	Output access time from CK	+/- 0.75ns	2.5 to 5.5ns from clock
TDQSCK	DQS access time from CK	+/- 0.75ns	2.5 to 5.5ns from clock
TDQSQ	DQS edge to output data edge	+/- 0.5ns	+/- 0.5ns
TQH	Output DQS valid window	THP - 0.5ns	THP - 0.5ns
tRPST	Read post-amble	Half clock	Half clock
TDQSS	CK to valid DQS-in	+/- 0.25tCK	+/- 0.25tCK
TDS/tDH	DQ setup/hold time	0.5ns	0.5ns
TWPRES	Write pre-amble	Half clock	Half clock
TIS/tIH	Input setup/hold time	1.1ns	1.1ns

### Additional design consideration for receiver to support DDR with non\_DLL

For the time being until the x32 DDR SDRAM supporting up to 300MHz is available, there will be no available issue to get x32 DDR SDRAM with DLL working at lower frequency. But, maybe from 2002, there will be. So, it may be better to consider supporting both DLL and non\_DLL from the beginning of the memory controller design. If you are familiar with DDR SDRAM, it may not be so difficult to have another option to support non\_DLL. The only difference between DLL and non\_DLL is the time when the data-out comes out from the memory at READ timing(tAC & tDQSCK). So, the turn-on timing of the input buffer(or DQS enable logic) has to be shifted with reference by tDQSCK on non\_DLL and a reference **internal clock** used to fetch the data from the input buffer to the internal logic has to be optimized for non\_DLL as well.



### Non\_DLL gives more power saving

To minimize the power consumption is one of the key items at the mobile applications like PDA & IMT-2000. Also does the power consumption of a memory need to be reduced. Especially, the standby current of the memory is the most important one to be reduced. With non\_DLL, huge standby current reduction can be achieved. See the following table, which is based on Samsung 2Mx32 DDR SDRAM A-die.

Parameter	Symbol	DLL mode		Non_DLL mode	
		-6	-7	-6	-7
Operating current	ICC1	292mA	263mA	243mA	222mA
Precharge Standby Current in Power-down mode	ICC2P	42mA	42mA	0.94mA	0.94mA
Precharge standby current in Non Power-down mode	ICC2N	128mA	110mA	95mA	83mA
Active standby current power-down mode	ICC3P	72mA	72mA	3.24nA	3.24mA
Active standby current Non power-down mode	ICC3N	161mA	141mA	100mA	86mA
Operating current	ICC4	367mA	327mA	328mA	303mA
Refresh current	ICC5	366mA	307mA	301mA	262mA
Self refresh current	ICC6	2.0mA	2.0mA	1.8mA	1.8mA

### Samsung will support non\_DLL mode on future x32 DDR SDRAM

To give a solution to another application using x32 DDR SDRAM at 100MHz below frequency, all of Samsung x32 DDR SDRAM will support non\_DLL mode. See the table below for details. The tDQSCK is the target value based on 66Mhz with non\_DLL and no different by frequency. Even on future x32 DDR like 2Mx32 DDR C-die and 4Mx32 DDR B-die may have same tDQSCK window to keep backward compatibility

Device	Generation	MP	Min Frequency on DLL	Max Frequency On non-DLL	<b>TDQSCK on non-DLL</b>
2Mx32 DDR	A-die	2Q00	125Mhz	183Mhz	<b>2.5 ~ 5.5ns</b>
	B-die	3Q01	100Mhz	<100Mhz	<b>4.5 ~ 7.5ns</b>
	C-die	1H02	100Mhz	<100Mhz	<b>4.5 ~ 7.5ns</b>
4Mx32 DDR	M-die	4Q00	100Mhz	<100Mhz	<b>4.5 ~ 7.5ns</b>
	A-die	3Q01	100Mhz	<100Mhz	<b>4.5 ~ 7.5ns</b>
	B-die	2H02	100Mhz	<100Mhz	<b>4.5 ~ 7.5ns</b>

### Better to take non\_DLL mode on the application using lower than 100Mhz

Simply saying, why not non\_DLL if you use x32 DDR SDRAM at lower than 100MHz which will last longer than x32 SDR SDRAM or x32 DDR SDRAM with DLL. Again, it will give you

a huge power saving as well.

### **Conclusion**

As mentioned before, it looks like that most of the consumer products do not require such a high frequency as what the graphic applications do. The frequency gap required between the graphic and the consumer applications will be getting wider and wider. If the graphic memory could alive with SDR SDRAM in the future as well, there will be no concern for the applications requiring relatively lower performance. But unfortunately, not for the graphic applications but for the consumer applications, the availability of x32 SDR SDRAM will be worse and worse by being replaced by x32 DDR SDRAM in the graphic market. It implies that adapting x32 DDR needs to be considered even though the required performance can be achieved with x32 SDR SDRAM.

For the application having a plan to adapt x32 DDR SDRAM at 100MHz below, it is highly recommended supporting non\_DLL mode instead of using DLL because of the better availability and the huge power saving mentioned above.