1 Experiment Introduction

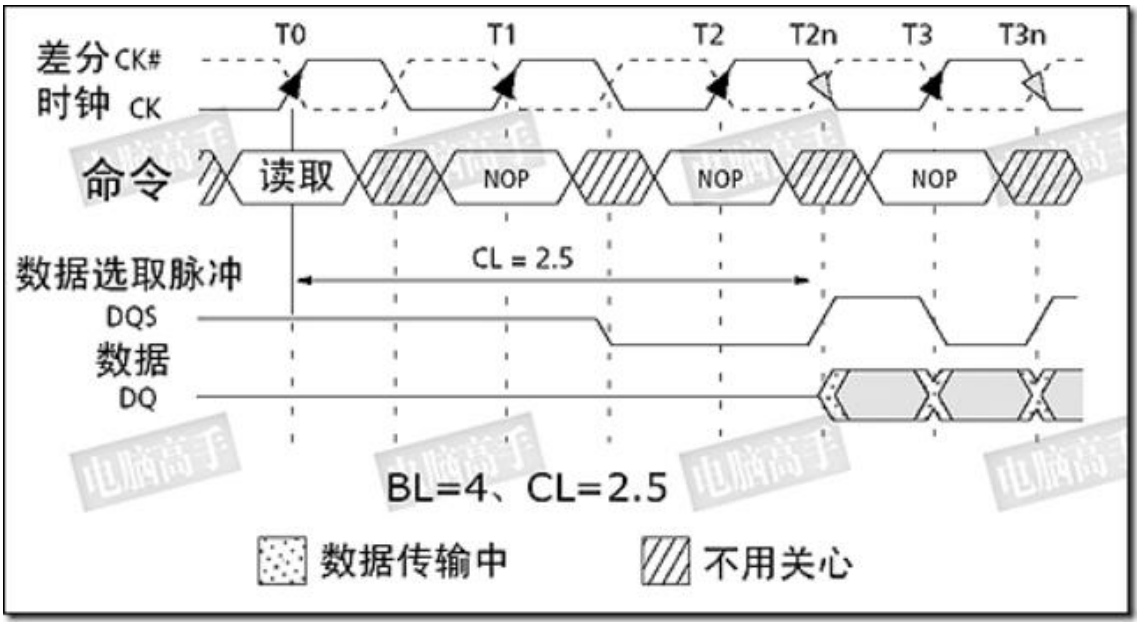
This experiment paves the way for the subsequent experiment of using DDR4 memory. Through loop reading and writing DDR4 memory, understand its working principle and the writing method of DDR4 controller. Due to the complicated control of DDR4, the programming of the controller is difficult. Here, the application of XILINX's MIG controller is introduced, which is the basis for subsequent audio and video experiments that require SDRAM experiments.

2 Experiment Principle

DDR SDRAM is called Double Data Rate SDRAM. DDR SDRAM is improved on the basis of the original SDRAM. Because of this, DDR can defeat the former rival RDRAM by virtue of Conversion cost advantage, becoming the mainstream today. This document only focuses on the DDR principle and the difference between DDR SDRAM and traditional SDRAM (also known as SDR SDRAM).

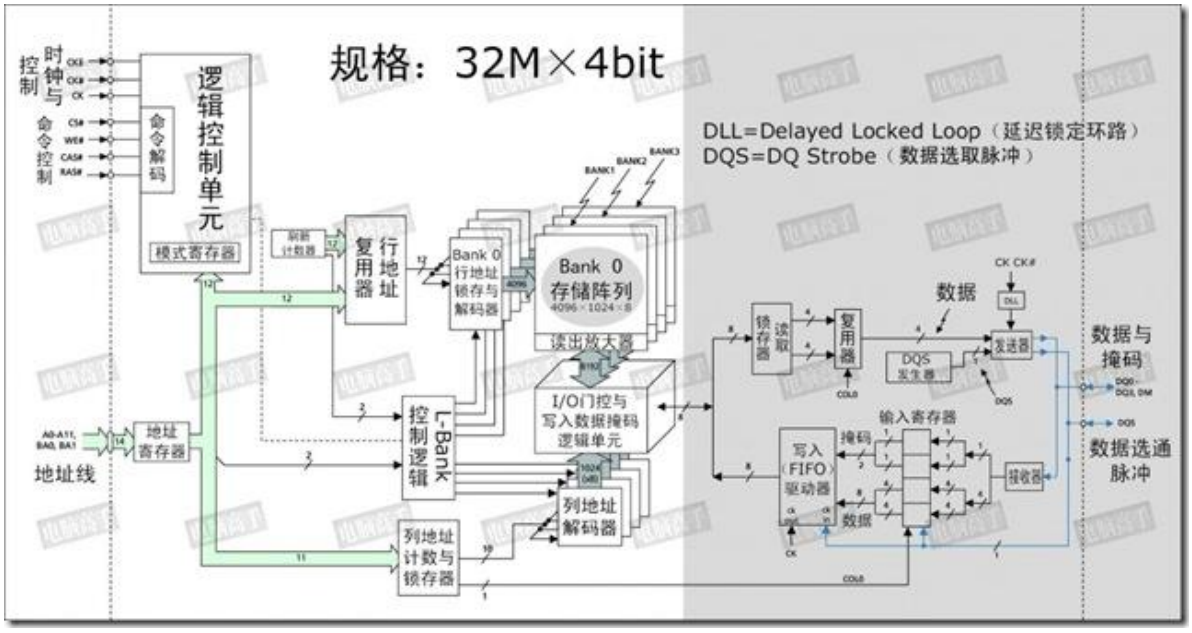
1. The basic principle of DDR

There are many articles discussing the principles of DDR, but some bring some wrong ideas. First let's take a look at a DDR regular timing diagram.



It can be found that it has two more signals: CLK# and DQS, CLK# is opposite to the normal CLK clock phase to form a differential clock signal. The data transmission is performed at the intersection of CLK and CLK#. It can be seen that data is triggered on the rising and falling edges of CLK (this is exactly

the rising edge of CLK#), thus realizing DDR. Here, we can say that the purpose of DDR is achieved through differential signals. Even CLK# helps trigger the second data, but this is only a brief description of performance, which cannot be said from a strict definition. The reason for achieving DDR is to start with its internal improvements.



Internal structure diagram of DDR memory chip

This is a 128Mbit memory chip. As can be seen from the figure, the structure of the DDR white area is basically the same as that the SDRAM. But please pay attention to the gray area, which is different from the SDRAM. The first is the internal L-Bank specification. The capacity of the L-Bank memory cell in SDRAM is the same as the chip bit width, but this is not the case in DDR SDRAM. The capacity of the memory cell is double the bit width of the chip. Therefore, the formula of “Chip bit width = storage unit capacity” is not applied here. Therefore, the actual number of row and column addresses is also different from the SDRAM of the same specification.

Taking this chip as an example, when reading, L-Bank transmits 8 bits of data to the read latch next to the trigger of the internal clock signal, and then divides into two channels of 4 bits of data and transmits them to the multiplexer, which are merged into a 4-bit data stream. Then, the transmitter transmits 4 bits of data to the north bridge twice on the rising and falling edges of the external clock under the control of the DQS. Thus, if the clock frequency is 100MHz, then at the I/O port, since it is triggered by the up and down edge, then the transmission frequency is 200MHz.

Now everyone understands the working principle of DDR SDRAM. This internal memory cell capacity (also known as chip internal bus bit width) = 2 × chip bit width (also known as chip I / O bus bit width) design. It is called “2-bit Prefetch”, and some companies call it “2-n Prefetch” (n stands for chip width).

2. DDR SDRAM is different from SDRAM

The difference between DDR SDRAM and SDRAM is mainly reflected in the following aspects.

The main difference between DDR SDRAM and SDRAM

Memory type / Comparison item	SDRAM	DDR SDRAM
Command		
Full page burst transfer	Support	Not Support
Clock signal hangs	Support	Not Support
Readout signal mask	Support	Not Support
Write signal mask	Support	Support
Features		
Clock	Single clock	Differential clock
Prefetch design	1-bit	2-bit
Data transfer rate	1/Clock cycle	2/Clock cycle
CAS incubation period	2, 3	1.5, 2, 2.5, 3
Write Incubation period	0	variable
Burst length	1, 2, 4, 8, Full Page	2,4,8
Delay locked loop	Optional	Must
Automatic refresh interval period	Fixed	Flexible design (maximum value is the same as fixed value of SDRAM)
Data selection pulse	NA	MUST
Package and Electrical Characteristics		
Package type (≥64Mbit)	TSOP-II 54pin (400 mil)	TSOP-II 66pin (400 mil)
		CSP 60min
Working Voltage	3.3V (LVTTL Interface)	2.5V (SSTL_2 Interface)
Module	168pin DIMM	184pin DIMM
Note: LVTTL=Low Voltage Transistor - Transistor Logit SSTL=Stub Series Terminated Logic		

Note: TSOP-II and CSP

TSOP-II: Refer to the second way of Thin Small Outline Package. The Pins on the both long sides of the package. TOP_I pins are on both sides of the short side

CSP: Chip Scale Package, The package size is basically the same as the core size of the chip, so it is called CSP. The ratio of its core area to package area is approximately 1:1.1, any package that meets this standard is called CSP.

DDR SDRAM, like SDRAM, also performs MRS at boot time. However, due to the increase in operational functions, DDR SDRAM has an EMRS stage (Extended Mode Register Set) before MRS. This extended mode register controls Valid/disabled DLL, output drive strength, QFC valid/invalid, etc.

Tips and Misunderstandings: The Meaning and Function of QFC

QFC refers to the FET Switch Controller, which is active low. That used to control the mutual isolation between the chips on the module by means of an external FET switch. No read/write operation enters the isolation state to ensure that the chips are not interfered with each other. QFC is

a special feature. The manufacturer only adds this function to the chip after receiving the specified requirements from the chip buyer, and needs to make relevant design changes in the mold assembly (such as increasing the pull-up resistor of VddQ). Therefore, the DDR memory that supports this function is rarely seen in the DIY market. In May 2002, JEDEC's latest DDR specification no longer has the definition of QFC, and even if there is a FET switch, it will be controlled by the North Bridge (this is used to isolate the module), so QFC has become a history. However, in many of the introductions, QFC is considered a necessary process, which is obviously wrong.

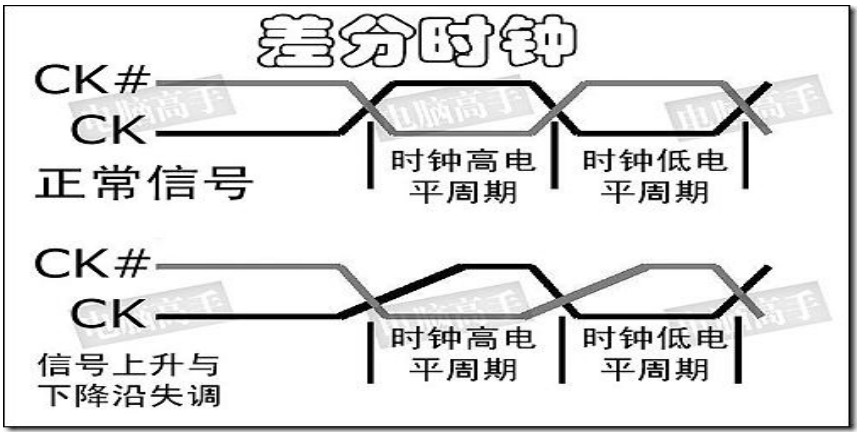
Since the operation methods of EMRS and MRS are similar to the MRS of SDRAM, the specific mode table is no longer listed here. If you are interested, you can view the relevant DDR memory data. Below we focus on the new design and new features of DDR SDRAM.

Misunderstanding: CSP and uBGA, WBGA, TinyBG, FPGA, etc. are different packaging technologies

In fact, CSP is just a package standard/type, and does not involve specific packaging technology. As long as it meets its size standard, it can be called CSP package. In recent years, uBGA, WBGA, TinyBG, and FPGA small chip packaging technology are the specific form of CSP (in fact, it is a kind of BGA packaging technology), it can be seen that CSP does not have a fixed packaging technology.It is not a packaging technology by itself. As long as the manufacturer is willing or capable, it can develop more CSP-compliant packaging technologies.

1) **Differential clock**

The differential clock (see “DDR SDRAM Read Operation Timing Diagram” above) is a necessary design for DDR, but the role of CK# is not to be understood as the second trigger clock (you can simply describe this in the DDR principle)Instead, it acts to trigger clock calibration. Since the data is triggered on the upper and lower edges of CK, the transmission cycle is shortened by half. Therefore, it is necessary to ensure the stability of the transmission period to ensure the correct transmission of data, which requires precise control of the upper and lower edges of the CK. However, due to changes in temperature and resistance performance, the spacing between the upper and lower edges of CK may change. At this time, the inverted CK# plays a corrective role (CK rises slowly and slowly, while CK# rises slowly and falls faster).Due to the triggering of the upper and lower edges, CL=1.5 and 2.5 are also possible and easy to implement. CK# inverted from CK guarantees the accuracy of the trigger timing.



2) Data selection pulse（DQS）

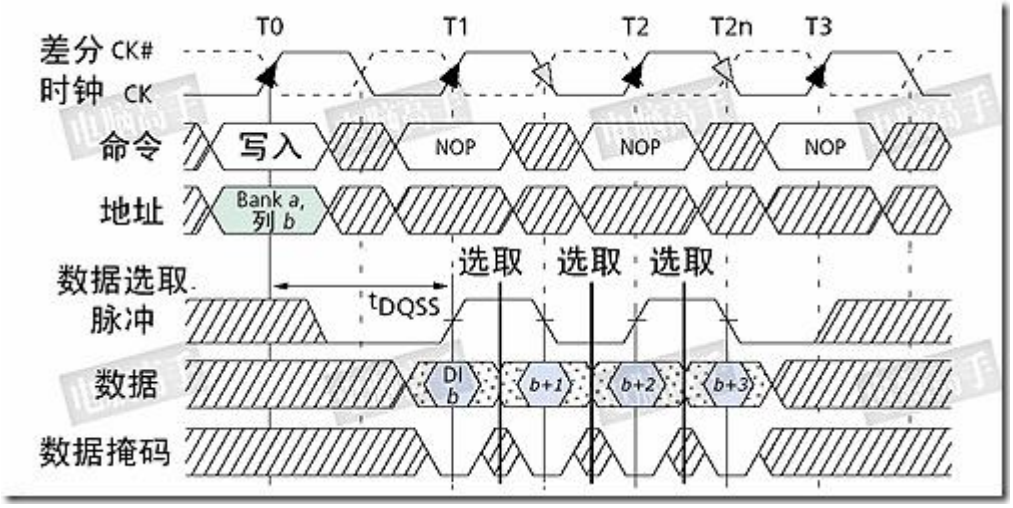
Summarize DQS: It is a bidirectional signal; when reading memory, it is generated by memory, the edge of DQS is aligned with the edge of data; when it is written into memory, it is generated externally, the middle of DQS corresponds to the edge of data, That is, the DQS edge is the most stable intermediate moment of the corresponding data.

DQS is an important function in DDR SDRAM. Its function is mainly used to accurately distinguish each transmission period in one clock cycle, and it is convenient for the receiver to receive data accurately. Each chip has a DQS signal line, which is bidirectional. It is used to transmit the DQS signal sent by the north bridge when writing. When is reading, the chip generates DQS to Transmit to the north bridge. It can be said that it is the synchronization signal of the data.

At the time of reading, “DQS” is generated simultaneously with the data signal (also at the intersection of “CK” and “CK#”). The “CL” in the “DDR” memory is the interval from the “CAS” to the “DQS” generation. The time interval at which the data actually appears on the data I/O bus relative to the DQS trigger is called “tAC”. Note that this is different from “tAC” in “SDRAM. In fact, when DQS is generated, the prefetching inside the chip is completed. “tAC” refers to the data output time of the gray part in the above structure diagram. Due to the prefetching reason, the actual data transmission may be transmitted ahead of the DQS. Due to parallel transmission, DDR memory also has certain requirements for “tAC”. For DDR266, the allowable range of “tAC” is ±0.75 ns, and for DDR433, it is ±0.7 ns. See the previous diagram for their timing diagram, where CL contains a lead-in period for DQS.

As mentioned above, DQS is to ensure the selection data of the receiver, and DQS transmits synchronously with the data when reading. Then, is the reception of DQS as the upper and lower edge? No, if you use the upper and lower edges of DQS to distinguish the data cycle, the danger is great. Since the chip has prefetch operation, the synchronization at the output is difficult to control. It can only be limited to a certain time range. The data may appear faster or slower on each I/O port and will have a certain interval from the DQS. This is why there is a reason for tAC regulations. On the receiving side, everything must be guaranteed to be received synchronously, and there must be no deviations such as tAC. In this way, when writing, the chip no longer generates DQS by itself, but takes the DQS transmitted from the sender as the reference, and delays the corresponding time accordingly. In the middle of the DQS is the selection of the data cycle (the split point is the upper and lower edges when reading), from which two transmission cycles are separated.

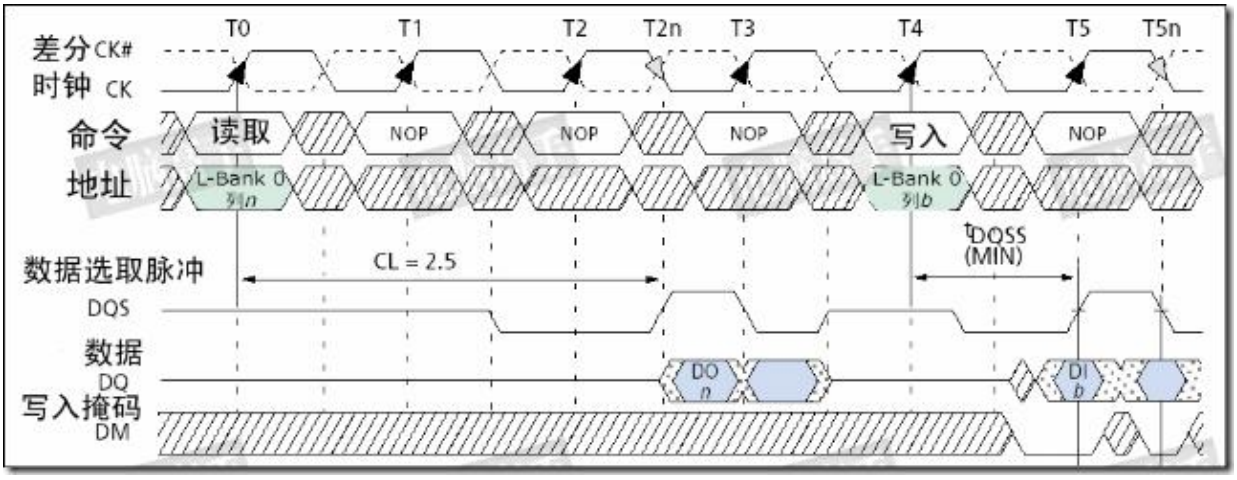
The advantage of this is that since each data signal has a logic level hold period, even if it is not synchronized at the time of transmission, it is in the hold period at the top and bottom of the DQS. At this time, the accuracy of the data reception trigger is undoubtedly the highest. At the time of writing, the middle of the high/low period of DQS is the data cycle division point instead of the up/down edge, but the data reception trigger is still the upper/lower edge of the DQS.



3) Write Delay

In the above DQS write timing diagram, it can be found that the write latency is not already 0. After the write command is issued, the DQS and the write data will wait for a while to be delivered. This cycle is called DQS delay time relative to write command (tDQSS, WRITE Command to the first corresponding rising edge of DQS), for this "time" everyone should be well understood.

Why do you have such a delay design? The reason is also synchronization. After all, one clock cycle is transmitted twice, which requires high control precision. It must wait for the receiver to be fully prepared. “tDQSS” is an important parameter for “DDR” memory write operations. If it is too short, it may be accepted incorrectly. If it is too long, the bus will be idle. The minimum “tDQSS cannot be less than 0.75 clock cycles, and the maximum length cannot exceed 1.25 clock cycles. Some people may say that if this is the case, isn't DQS out of sync with the clock inside the chip? Yes, under normal circumstances, “tDQSS” is a clock cycle, but the receiver's clock is only used to control the synchronization of the command signal when writing, and the data is completely synchronized by DQS, so it does not matter if the DQS is not synchronized with the clock. However, “tDQSS” has an adverse effect - the increased in latency after read and write operations. If “CL = 2.5”, add half a clock cycle to “tDQSS” because the command is issued on the rising edge of CK.



When “CL=2.5”, Delay after reading will be “tDQSS+0.5” clock cycles (BL=2 in the figure)

In addition, the actual write of DDR memory data is subject to more steps, so the writeback time (tWR) is also significantly extended, generally around 3 clock cycles, and In the DDR-II specification, “tWR” is listed as an item of the mode register, which shows its importance.

Misunderstanding: DDR SDRAM is halved in unit time for various delays and latency periods

Some articles believe that DDR doubles the data transfer rate, and the associated latency and Incubation period are also halved. For example, DDR-266 memory, tRCD, CL, and tRP have a unit period of 3.75 ns, which is half the memory of PC133. . This is a serious conceptual error. As can be seen from the timing diagrams we have listed, tRCD, CL, and tRP are defined by clock signals. They cannot be represented by transmission cycles. Otherwise, what is the reference standard of CL=2.5? For DDR-266, the clock frequency is 133MHz and the clock cycle is still 7.5, the same as the PC133 standard. The authors of those articles apparently confuse the clock cycle with the transmission cycle.

4) Burst length and write mask

In DDR SDRAM, the burst length is only 2, 4, and 8 options, without random access operations (burst length is 1) and full page bursts. Why is this? Because L-Bank accesses twice the data width of the chip at a time, the chip must be transmitted at least twice, otherwise how to deal with the internal data? The full-page burst of facts proves to be difficult to use in PC memory, so it is not surprising to be canceled.

However, the definition of burst length is also different from that of SDRAM (see the DDR schematic in the first part of this chapter), which does not refer to the number of consecutively addressed memory cells. It refers to the number of consecutive transmission cycles, each time being a chip bit width data. For burst writes, if there is data that you do not want to store, you can still use the DM signal for masking. The DM signal and the data signal are simultaneously transmitted, and the receiver judges the state of the DM on the rising and falling edges of the DQS. If the DM is high, the data previously selected from the middle of the DQS is masked. One might think that DM is an input signal, meaning that the chip can't send DM signals to the North Bridge as a reference for masking read data. In fact, which data to read is also determined by the Northbridge chip, so the chip does not need to participate in the work of the North Bridge, which data is useful, leave it to “Northbridge” to choose.

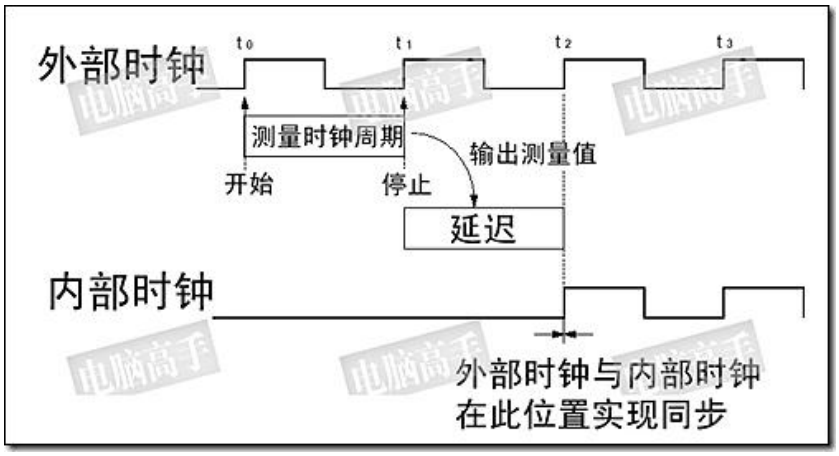
5) Delay locked loop（DLL)

DDR SDRAM has high requirements for clock accuracy. DDR SDRAM has two clocks, one is the external bus clock and the other is the internal working clock. In theory, the two clocks of DDR SDRAM should be synchronized, but for various reasons, delays such as temperature and voltage fluctuations make it difficult to synchronize the two, not to mention the fact that the clock frequency itself is unstable (SDRAM also has an internal clock, but because of its low operating/transmission frequency, internal and external synchronization problems Not prominent). The tAC of DDR SDRAM is caused by the deviation of the internal clock from the external clock, which is likely to cause errors due to data out of sync. In fact, asynchronous is a kind of positive/negative delay. If the delay is unavoidable, if the delay value is set, such as one clock cycle, the rising and falling edges of the internal and external clocks are

synchronized. Since the external clock cycle is not absolutely uniform, it is necessary to dynamically correct the internal clock delay according to the external clock to achieve synchronization with the external clock. This is the task of the DLL.

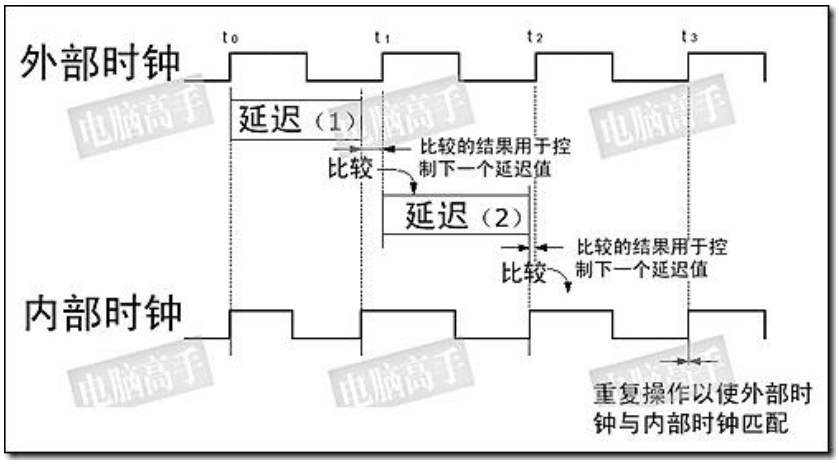
The DLL is different from the PLL on the motherboard. It does not involve frequency and voltage conversion, but instead generates a delay amount to the internal clock. Currently, DLLs have two implementation methods, one is CFM (Clock Frequency Measurement) and the other is Clock Comparator (CC).

CFM measures the frequency period of the external clock, and then controls the internal clock with this period as the delay value, so that the internal and external clocks are exactly one clock cycle apart, thus achieving synchronization. The DLL repeatedly measures the iterative control delay value so that the internal clock is synchronized with the external clock.



CFM DLL working diagram

The CC method compares the length of the internal and external clocks. If the internal clock cycle is short, the less delay is added to the next internal clock cycle, and then compared with the external clock. If the internal clock cycle is long, the extra delay is removed from the next internal clock, and so on, eventually synchronizing the internal and external clocks.



CC type DLL working diagram

CFM and CC have their own advantages and disadvantages. CFM has a fast correction speed of only two clock cycles, but it is susceptible to noise interference, and if the measurement is wrong, the internal delay will be wrong forever. The advantage of CC is that it is more stable and reliable. If the comparison fails, the delay is only affected by one data (and not too serious). It will not involve the delay correction, but its correction time is longer than CFM. The DLL function can be disabled in DDR SDRAM, but only for debugging and evaluation operations, the normal working state is automatically valid.

Misunderstanding: DLL is the key to DDR transmission

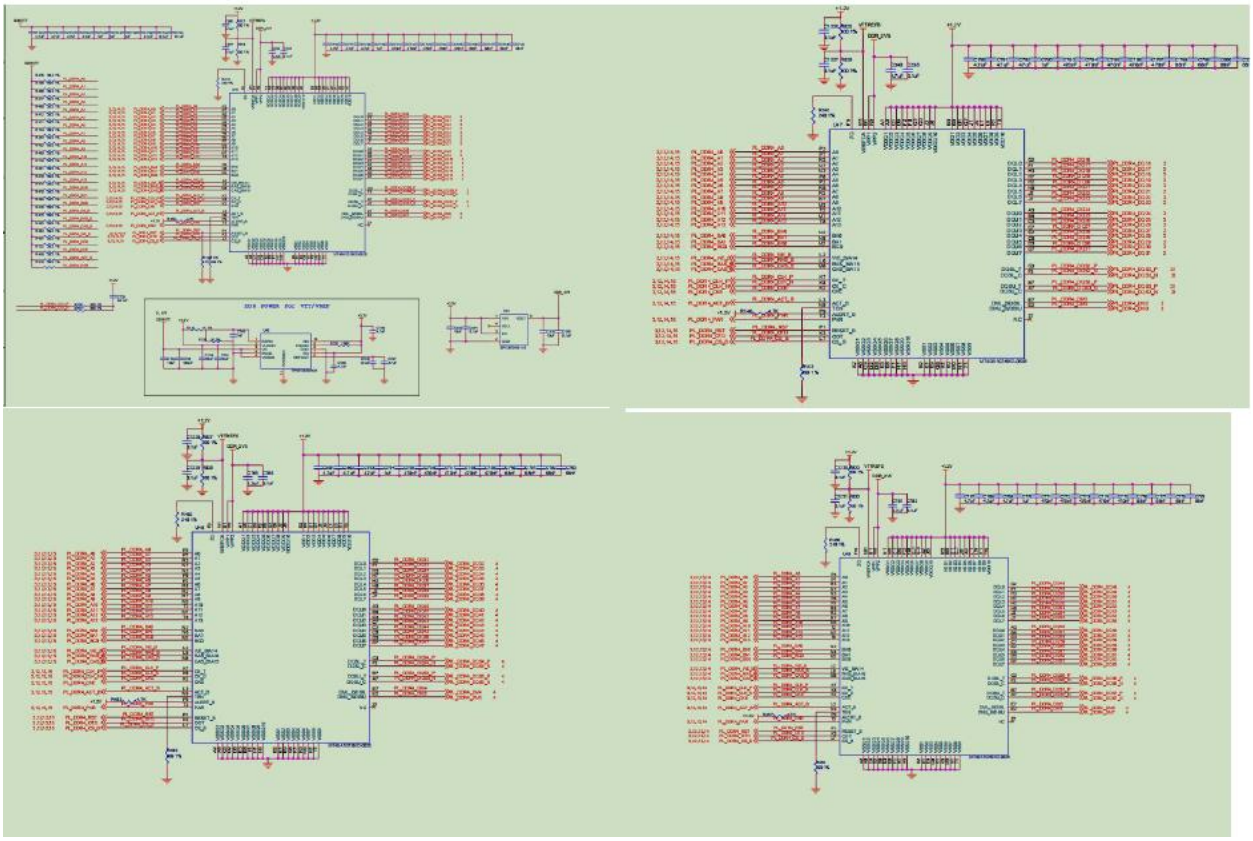
DDR memory provides accurate clock positioning through an internal DLL delay-locked loop, allowing data to be transmitted on the rising and falling edges of each clock cycle. DDR uses a DLL to provide a data filtered signal DQS to select data.

The above are two popular explanations of the working principle of DDR. Can you see the error now? Both of them exaggerate the function of the DLL. The DLL is only an important auxiliary calibration design. It has nothing to do with whether or not the dual edge trigger can be realized. It just guarantees that the data is synchronized with the external clock as much as possible. The latter is a conceptual error. As can be seen from the DDR memory structure diagram, DQS is not generated by the DLL, but is guaranteed by the DLL to synchronize with the external clock. The DQS is generated by a separate DOS generator.

3 Hardware Introduction

The AXKU040 FPGA development board uses four Micron DDR40A1G16KD-062E, each with a capacity of 4Gb. The four DDR4 chips are combined into a 64-bit data bus width to be connected to the FPGA. On the AXKU041 FPGA development board, the DDR address line and control line are made with termination resistors and pulled up to the VTT voltage to ensure the quality of the signal. In the design of the PCB, it fully complies with the DDR design specification of XILINX, and strictly guarantees the isometric design and impedance control. When designing DDR and Kintex-7, the DDR pin assignment of the FPGA should be considered, and cannot be assigned arbitrarily. If the user does not know how to connect, please refer to our schematic diagram to design

In the design of the PCB, considering the reliability of data transmission of high-speed signals, the equidistant design and impedance control are strictly guaranteed on the trace. The schematic diagram of the DDR part of the development board is as follows:



DDR4 Circuit on the AXKU040 FPGA Development Board

4 Programming

4.1 MIG Controller Design

The MIG IP controller is a DDR-controlled IP provided by Xilinx for users, so that users can easily read and write DDR memory through the DDR controller even if they don't understand the control and read/write timing of DDR. The solution for the Kintex UltraScale series DDR controller is as follows:

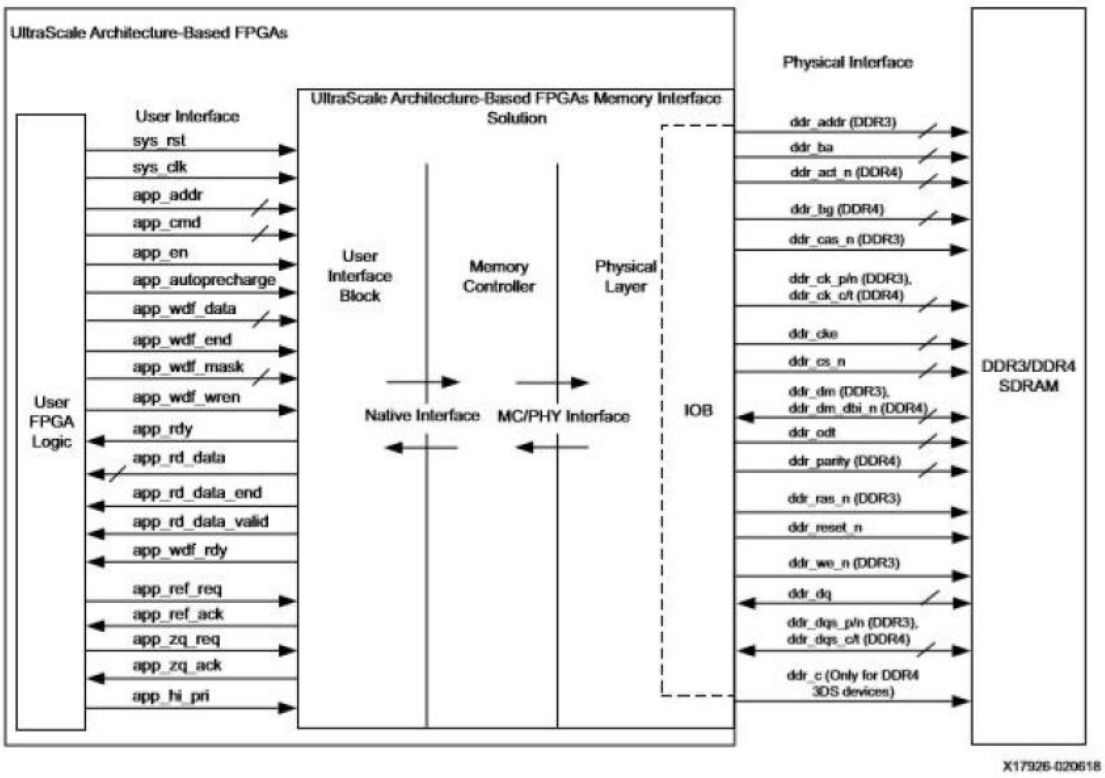
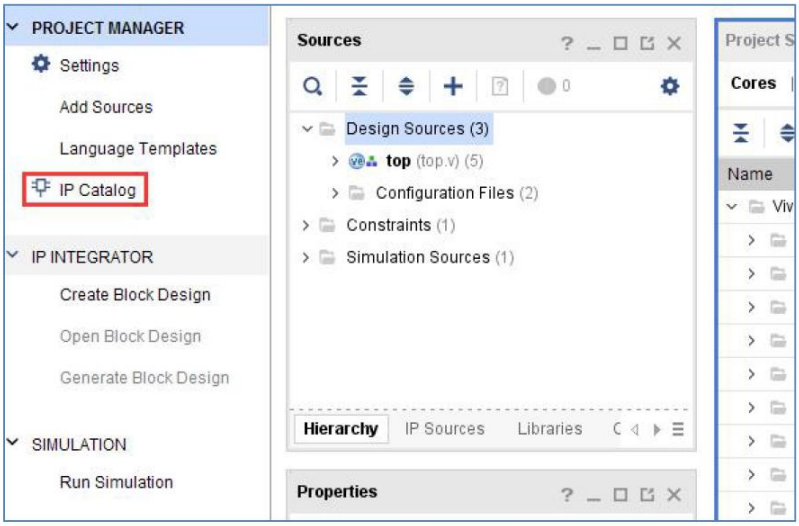


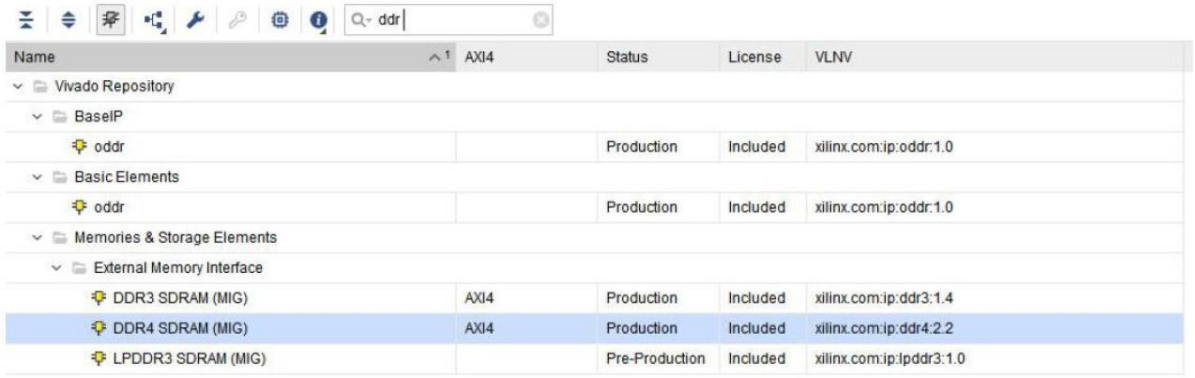
Figure 1-1: **UltraScale Architecture-Based FPGAs DDR3/DDR4 Memory Interface Solution**

The DDR4 controller consists of three parts: the User Interface Block, the Memory Controller and the Physical Layer of DDR4. Developers only need to develop the user's logic design to interface with the DDR controller's user interface to read and write DDR4 data. For more information on the interface definition and timing of the DDR4 controller client, refer to the documentation provided by Xilinx (UG150), and then introduce how to generate and configure the DDR4 controller!

1. First create a new project in the “Vivado” environment, named “ddr4_test”. Then click the “IP Catalog” under the “Project Manager” interface to open the “IP Catalog” interface.

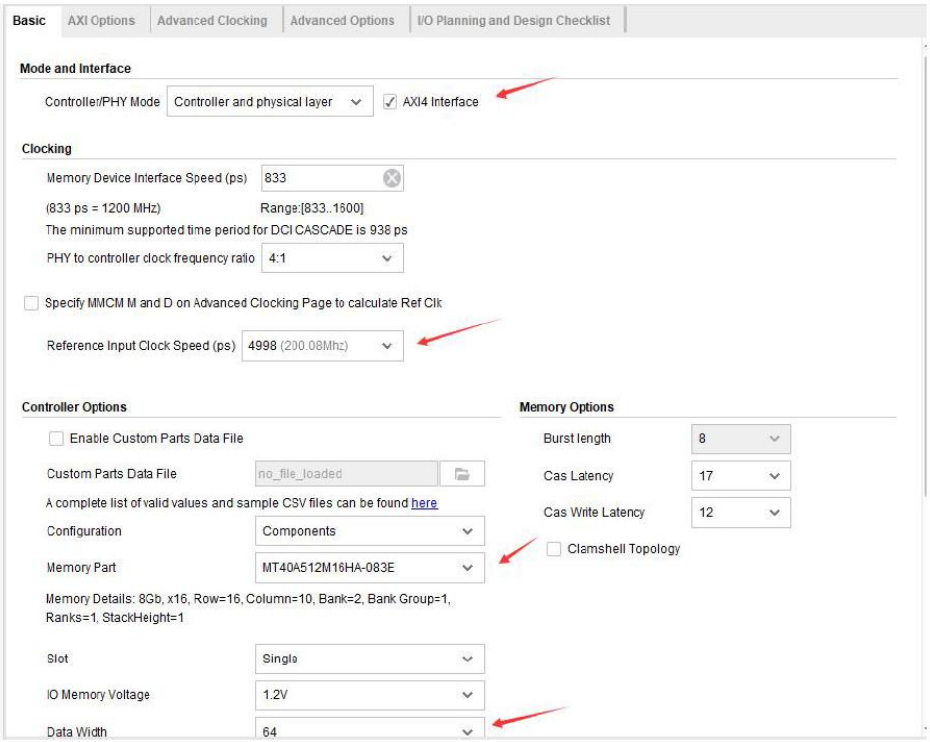


2. Search for DDR4 in the IP Catalog screen and select DDR4 SDRAM

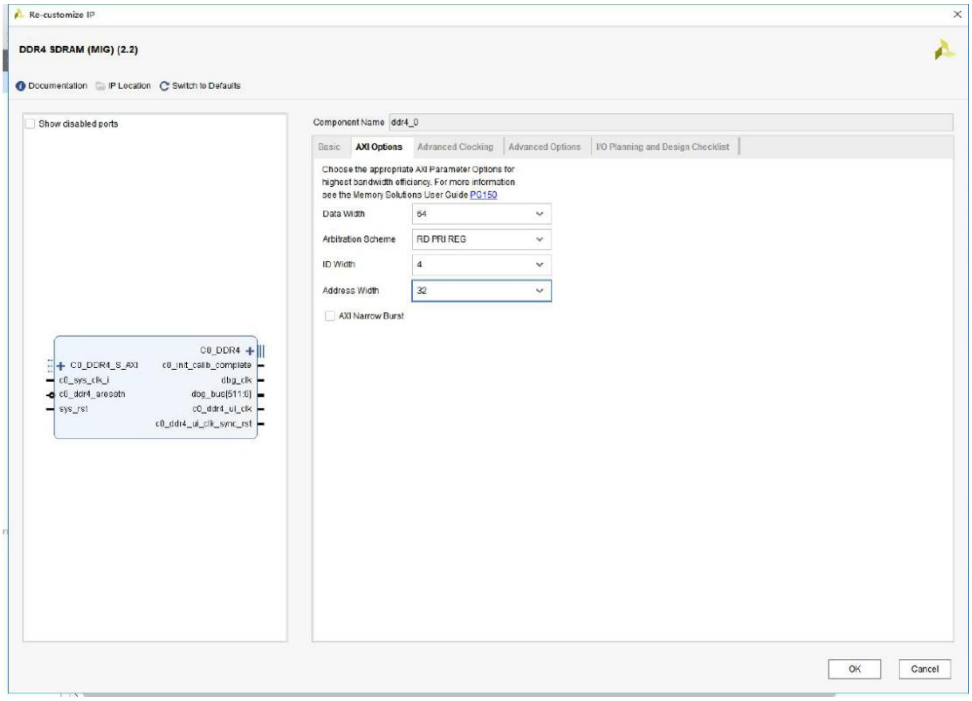


3. Click Next, if you want to know more about MIG, you can click the Product Guide button on the left to open Xilinx related documents to view.

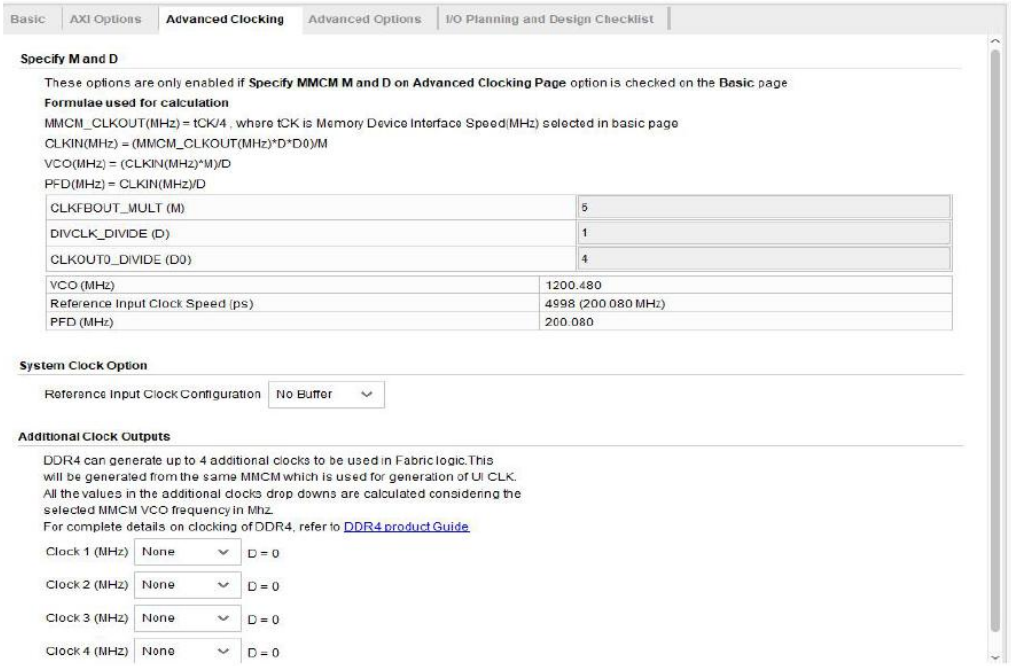
4. Select **AXI4** Interface in the basic column; For reference clock, we choose the system clock 200M corresponding to the development board; Memory chooses the similar DDR model MT40A512M16HA-083E due to version reasons; Data Width select 64 bits



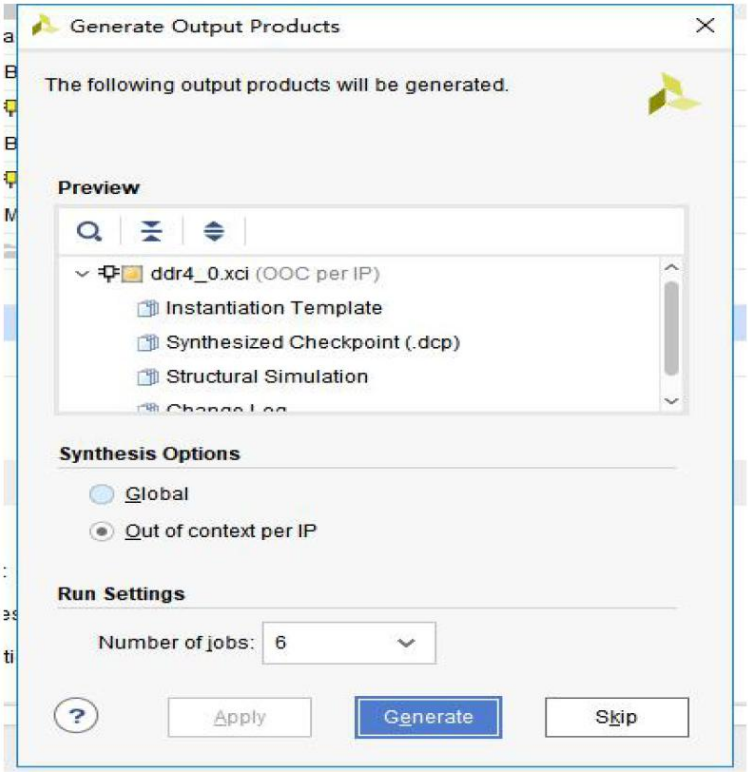
5. The corresponding bit width setting is also required in AXI Options. You can also refer to the official PG150 document for design



6. In Advanced Clocking, we set the system clock to No Buffer. Buffering has already been done in our program, so there is no need to add it here



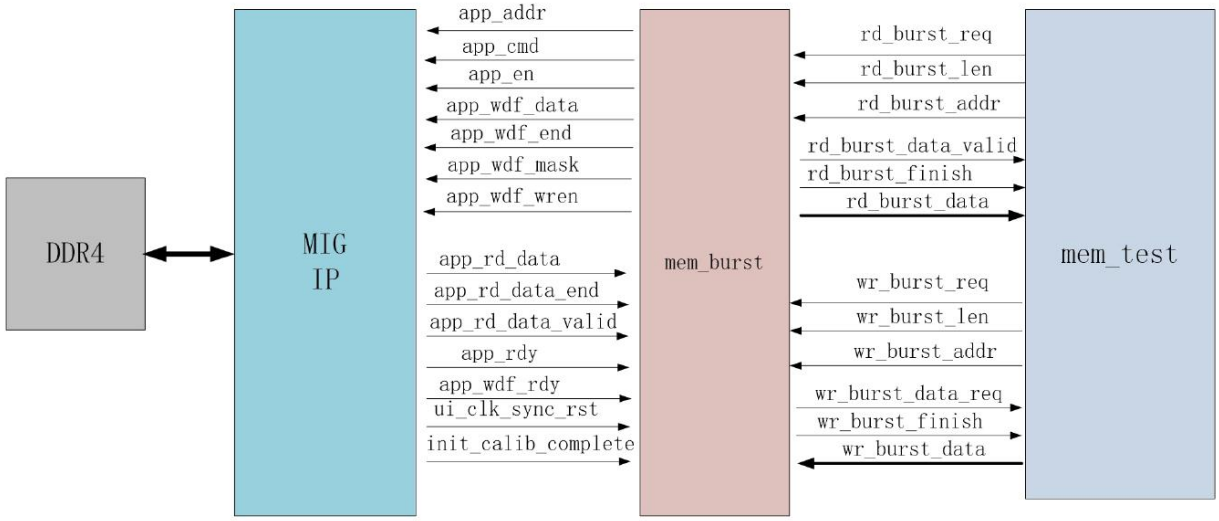
7. The remaining two items select "ok" by default, click Generate to generate ip



For the detailed introduction of MIG 7 Series, please refer to the document "ug586_7Series_MIS.pdf" provided by Xilinx

5 DDR4 test program design

This chapter learns how to write a DDR4 driver that communicates with DDR4 IP to implement DDR4 data reading and writing.



1. First let's write a DDR driver **mem_burst.v**

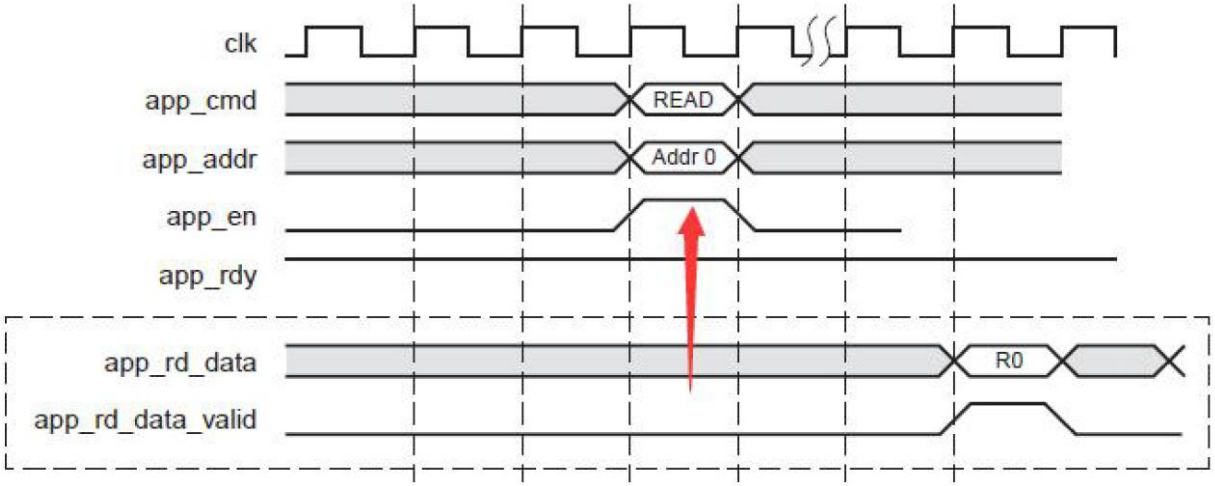
Although the ddr.v program generated by DDR IP has the interface of the user part, the interface timing of this part is still too complicated for us, and there are many interface signals used to read and write data. Therefore, it is necessary to write a more general program with a simpler interface to encapsulate ddr.v, which is the function and purpose of the mem_burst.v program.

Through the **mem_burst** program, users can easily read and write **DDR** data. When writing data, just control the write request signal **wr_burst_req**, write length **wr_burst_len**, write address **wr_burst_add** and write **data wr_burst_data**. Similarly, when reading data, just control the read request signal **rd_burst_req**, read length **rd_burst_len**, read address **rd_burst_add**, read data valid **rd_burst_data_valid** and read data **rd_burst_data**.

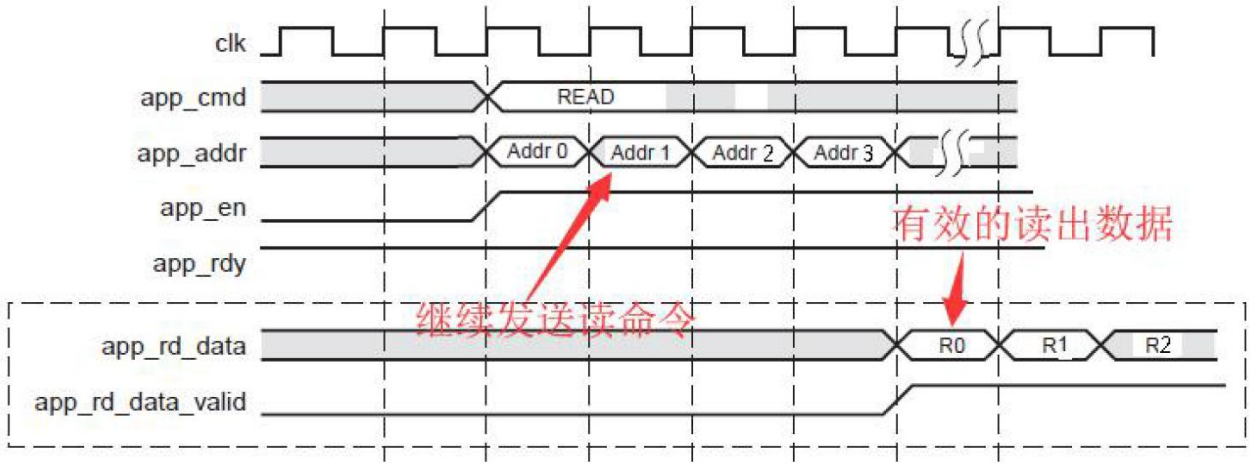
The specific program of **mem_burst.v** refers to the project provided by us. The function of the program is to convert the external **burst** read and write requests into the required signals and timing of the **DDR IP** user interface. The following is the flow of reading and writing:

DDR Burst Read

When the program receives a read request in the **IDLE** state (**rd_burst_req** is high), it will transmit the first data read command (read command, address, command valid) signal to the user interface of the **DDR IP**, and will enter the **MEM_READ** state. The timing of a read command write is as follows:



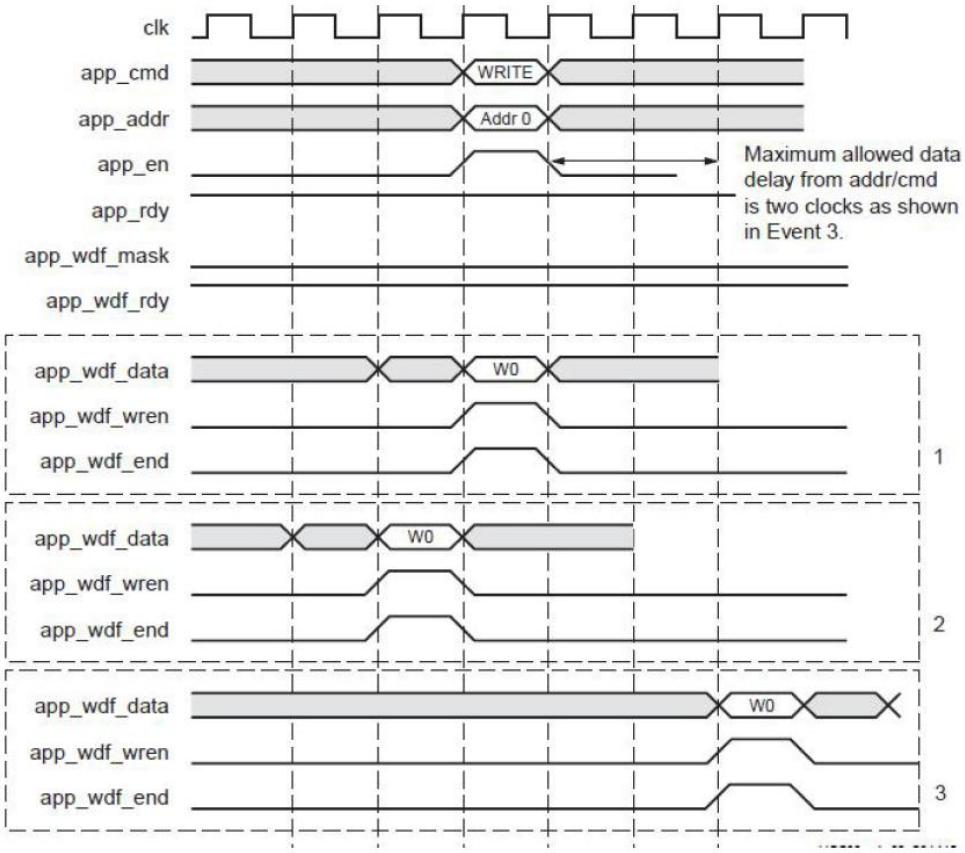
In the **MEM_READ** state, if the user interface of the **DDR IP** is judged to be idle, the remaining data read commands (address increment) will be transmit, After judging that the read address of the burst length is transmitted, go to the **MEM_READ_WAIT** state. In addition, in this **MEM_READ** state, it is also necessary to judge whether the data read by the DDR IP from the DDR is valid (**app_rd_data_valid**), to count whether the read data is the read burst length.



Read burst length DDR data in **MEM_READ_WAIT** state. After reading all the data, it will enter the **READ_END** state, and then return to the **IDLE** state.

DDR Burst Write

When the program receives a write request in the **IDLE** state (**wr_burst_req** is high), it will transmit the first data write command (write command, address, command valid) signal to the user interface of the DDR IP, and will enter the MEM_WRITE state. The following is a timing diagram of writing data. The enable of the write command here is app_en, and the enable of data writing is app_wdf_wren. It is not required to be valid on the same clock, and it can be advanced or delayed.



In the **MEM_WRITE** state, if the user interface of the DDR IP is judged to be idle, the remaining data write commands will be sent (address increase). At the same time, in this **MEM_WRITE** state, the data to be written to the DDR is also written to the data FIFO of the DDR IP. The following figure is the timing diagram of continuous write command and write data.

When the length of the data written to the FIFO of the DDR IP is the length of the write burst, it goes to the **MEM_WRITE_WAIT** state. In the **MEM_WRITE_WAIT** state, it is judged whether the DDR write command (by judging the address register of the write command) is transmit or not. After the transmitting is completed, it enters the **WRITE_END** state, and then returns to the **IDLE** state.

It should be noted here that transmitting a write command to the user interface of the DDR IP and writing data to the DDR are independent, because the data written to the DDR needs to be stored in the FIFO of the DDR IP first. As long as **app_wdf_rdy** is valid, data can be written to the DDR IP. The data written to the DDR needs to be prepared in advance. There is a **wr_burst_data_req** signal to request the user to prepare the DDR data to be written. The data written to the **FIFO** is completed in the **MEM_WRITE** state.

```
assign wr_burst_data_req = (state == MEM_WRITE) & app_wdf_rdy
```

2. Next, let's write a DDR test program **mem_test.v**

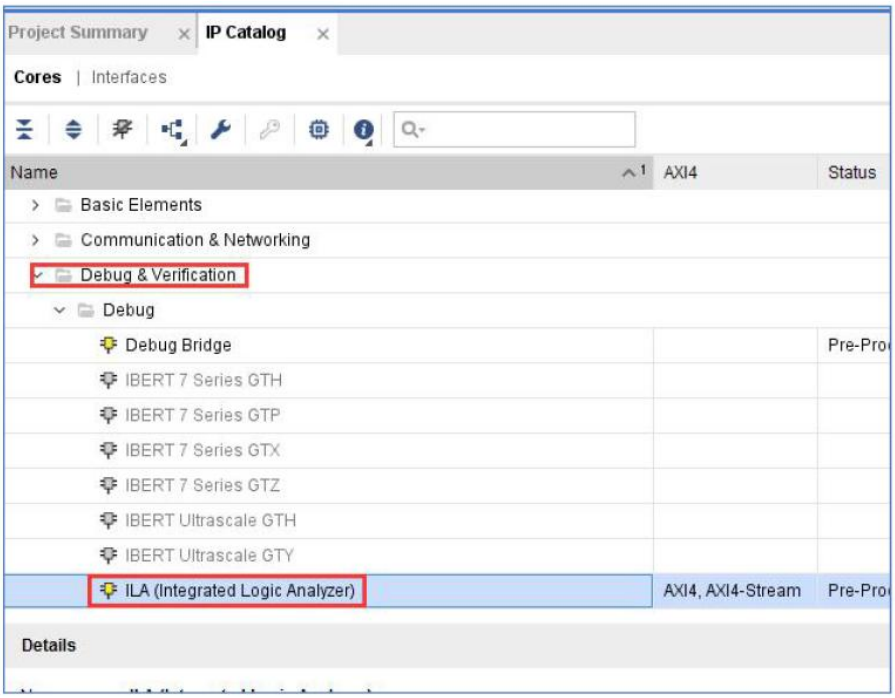
The **mem_test.v** test program mainly implements the burst read and burst write functions of ddr. The program generates read and write request signals, addresses and test data, and verifies whether the read and written data are correct. The data length of burst here is 128. If the read and write data of DDR is wrong (the written data is inconsistent with the read data), the error signal will become high.

The specific read and write timing can be combined with ila to see the specific waveform, which is easier to understand.

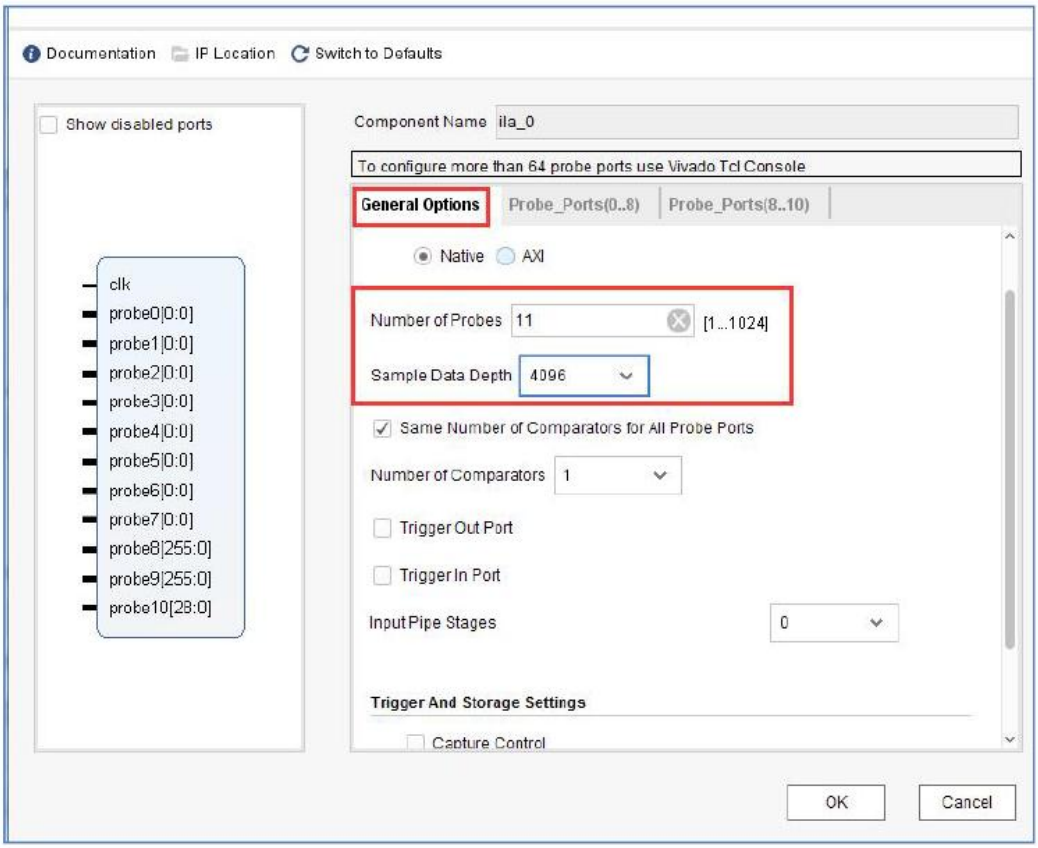
3. Next let's write a top-level program **top.v**

The **top.v** program instantiates the **mem_burst** module, the **mem_test** module and the **DDR IP** module **ddr.v**, the DDR clock generation module, and instantiates an **ila_0 IP** to observe the data, address and control signals of DDR Burst read and Burst write . So we also need to add another **ila IP** to the project.

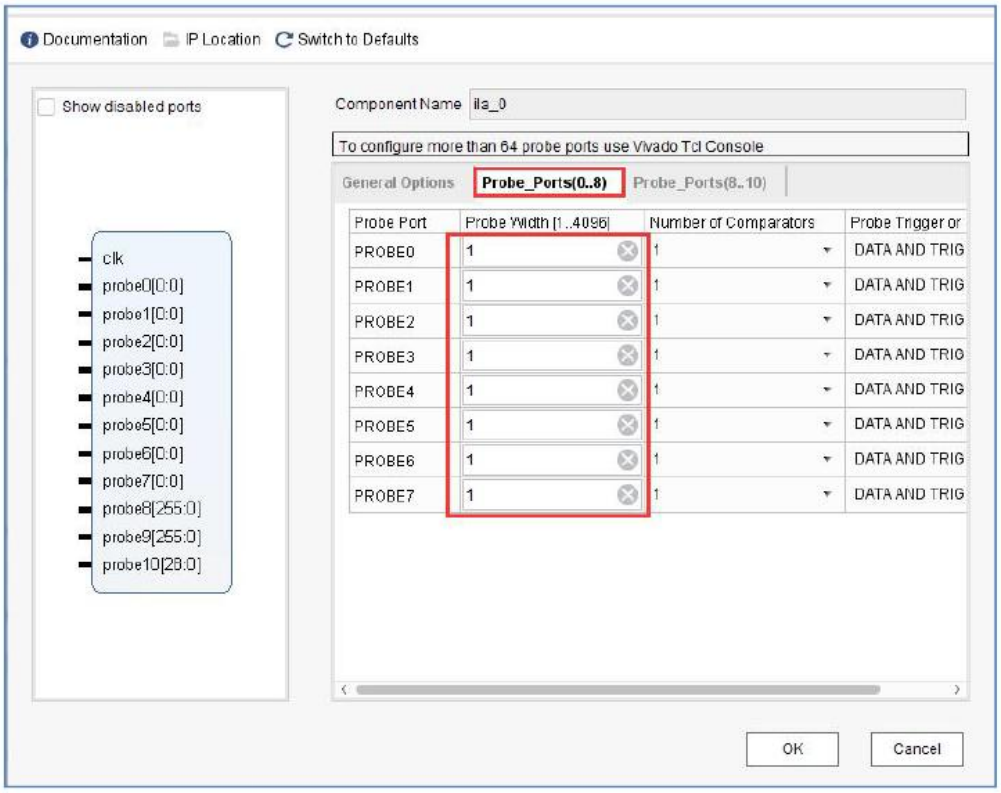
4. Add **ila IP**, open the **IP Catalog** interface, select **ILA (Integrated Logic Analyzer)** under **Debug & Verfication\Debug**, and double-click to open it.



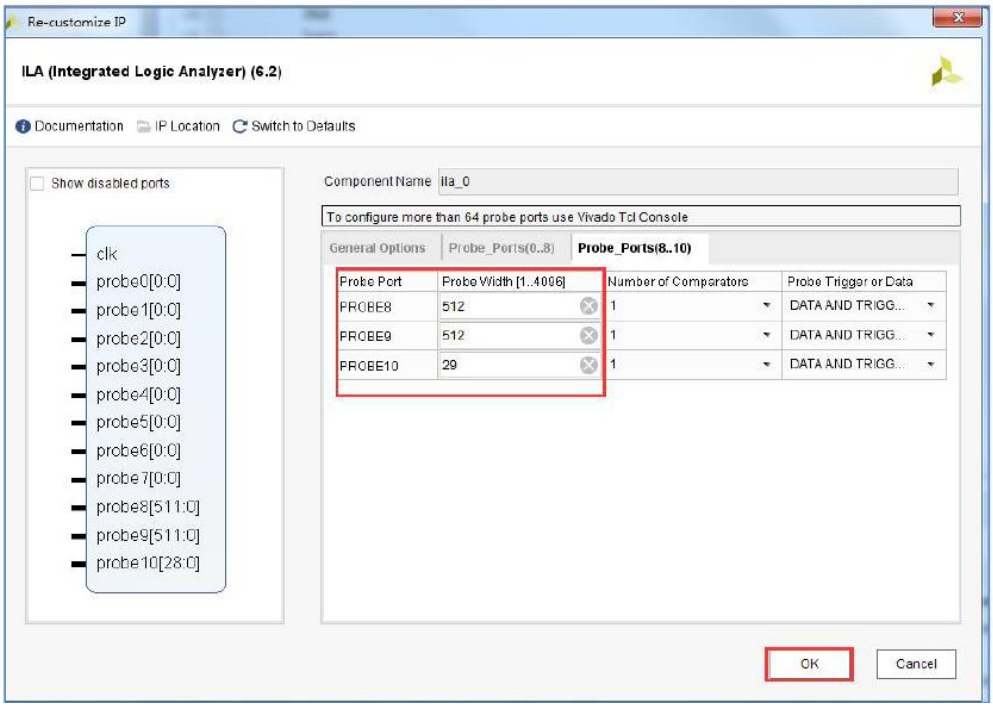
The Component name is **ila_0**, and the name needs to be the same as in the program. The number of probes is 11, that is, 11 sampling channels, and the sampling data depth is **4096**. The deeper the sampling depth, the larger the sampling data volume, but it will consume more FPGA logic resources.



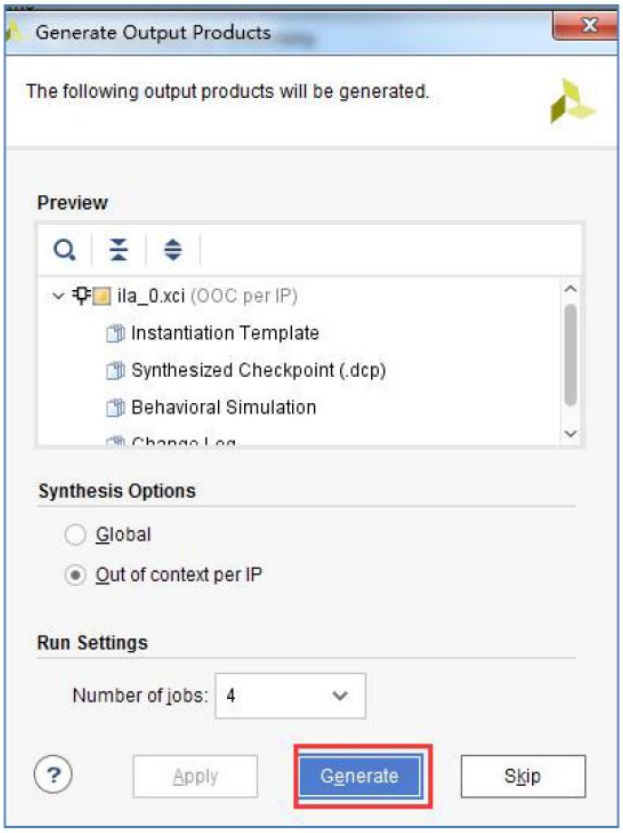
Then set the data width for the first 8 channels. Here, because we only sample one bit signal per channel, the data width is all 1.



Then set the data width of the following 8~10 channels. The 8 channel and 9 channel are set to 256 data width, which is used to sample the DDR read and write data, and the 10 channel is set to 29, which is used to sample the DDR address. Click OK to finish.



Click the “Generate” to generate the IP design file.

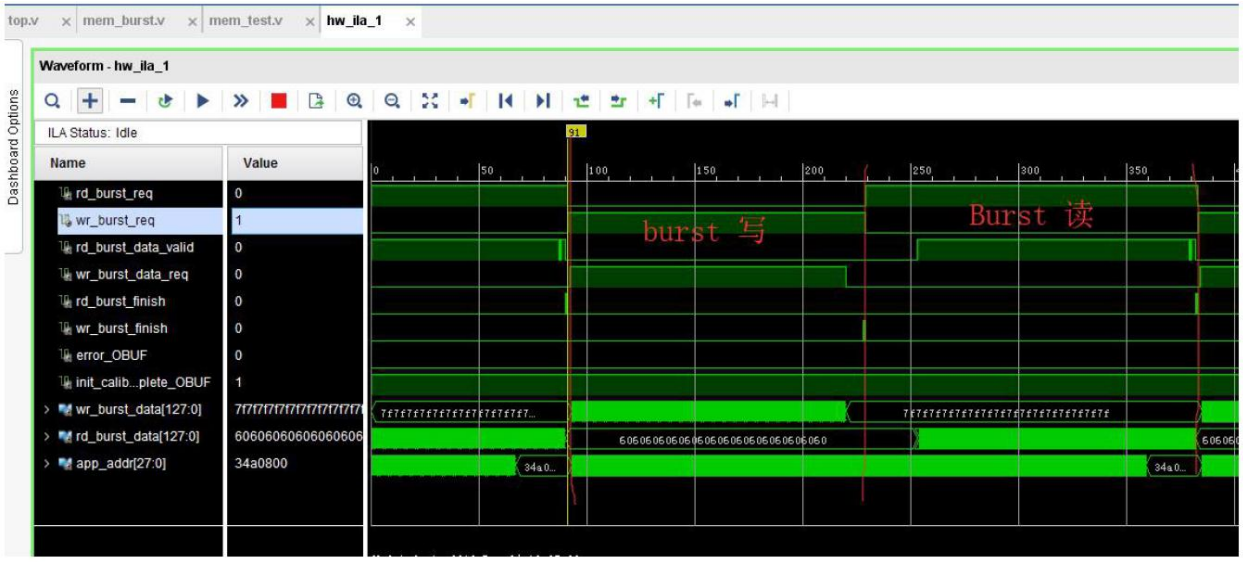


Then add the xdc pin constraint file ddr_test.xdc (see the engineering file), and connect the error signal to the LED1 of the FPGA development board. If the error is high, LED1 will be on. In addition, set the reset signal rst_n of the system and the FPGA clock input sys_clk, and the fan_pwm distribution pin controls the fan to be turned on.

6 Download and test

The compiler generates the “top.bit” file and then downloads the bit file to the FPGA.

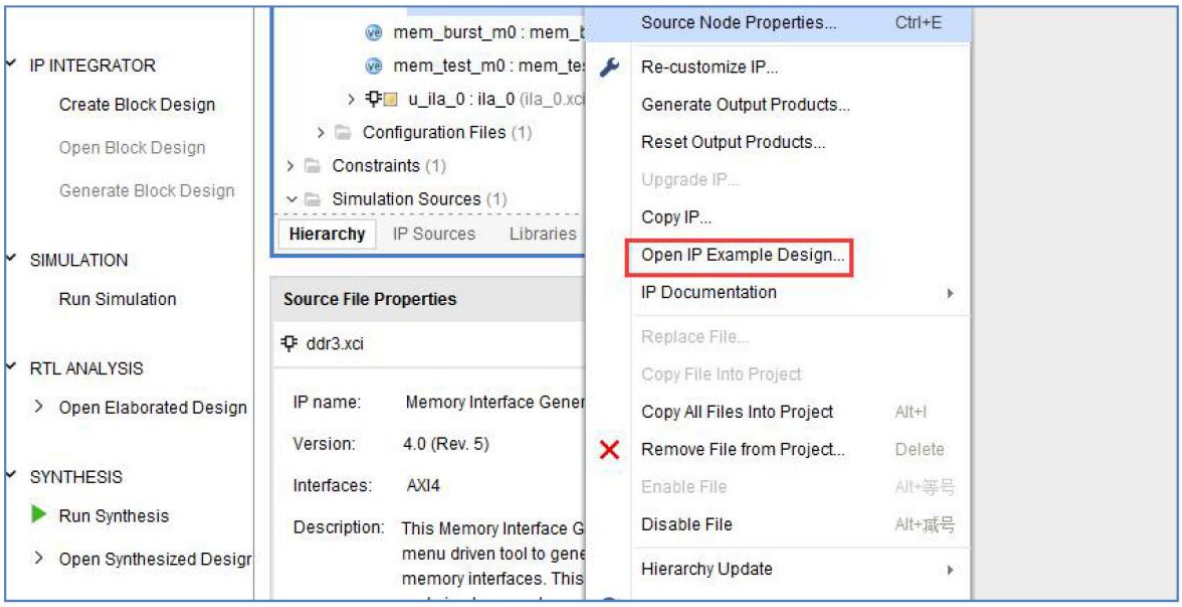
After downloading, the software will automatically open the waveform debugging interface of “ila”, click the “Run trigger for ILA core” button to run ila to collect data.



In addition, we can also check whether the LED1 on the FPGA development board is lit. If it is lit, it means that the error signal is high, and the read and write data of DDR is wrong. If it stays off, it means that the DDR read and write data is correct.

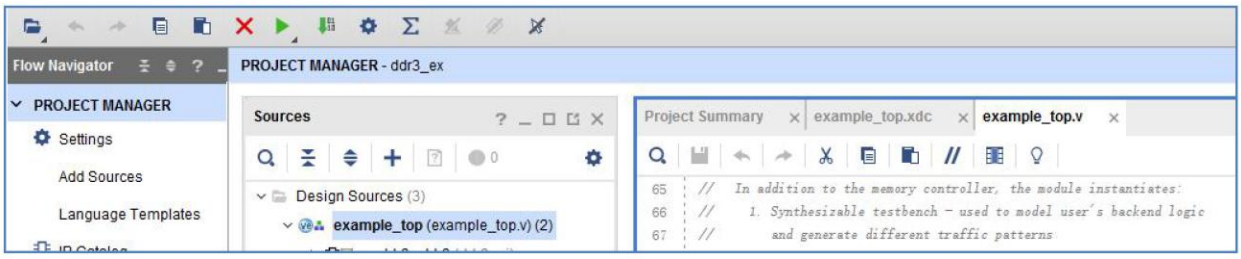
7 DDR4 Simulation

1. In the above example, right click on “ddr IP” and select “Open IP Example Design” from the pop-up menu.



The path generated by the “ddr_ex” project is to select the default, you can do it without modification, click OK.

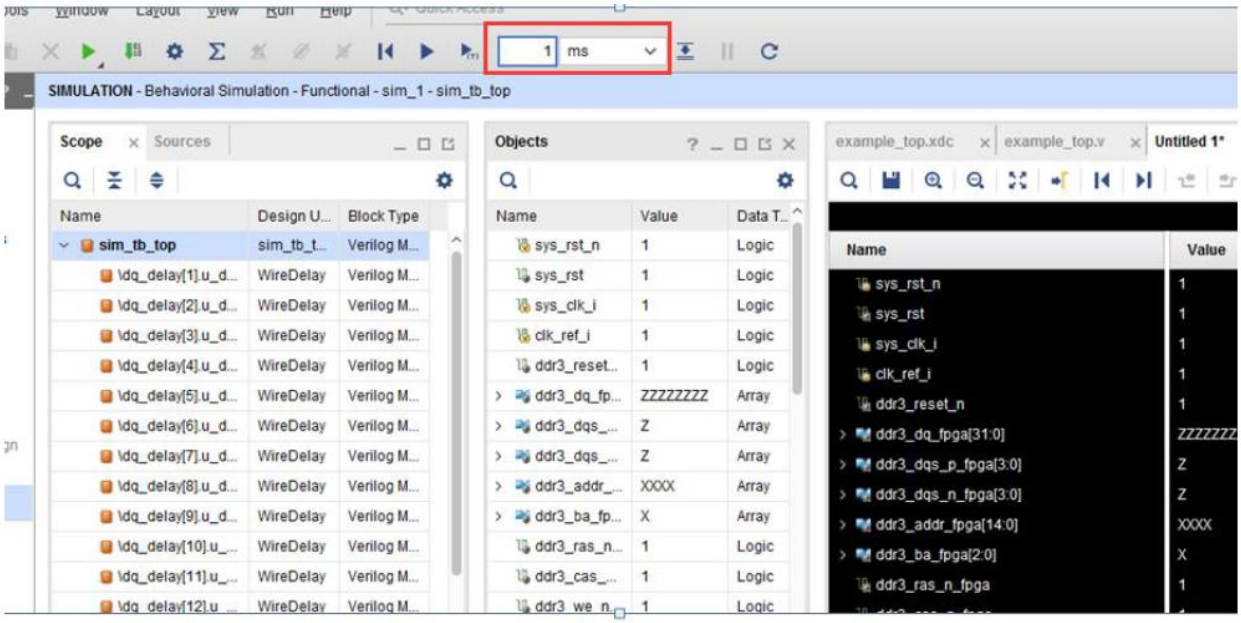
The software will automatically open the generated “ddr4_ex” project, as shown below



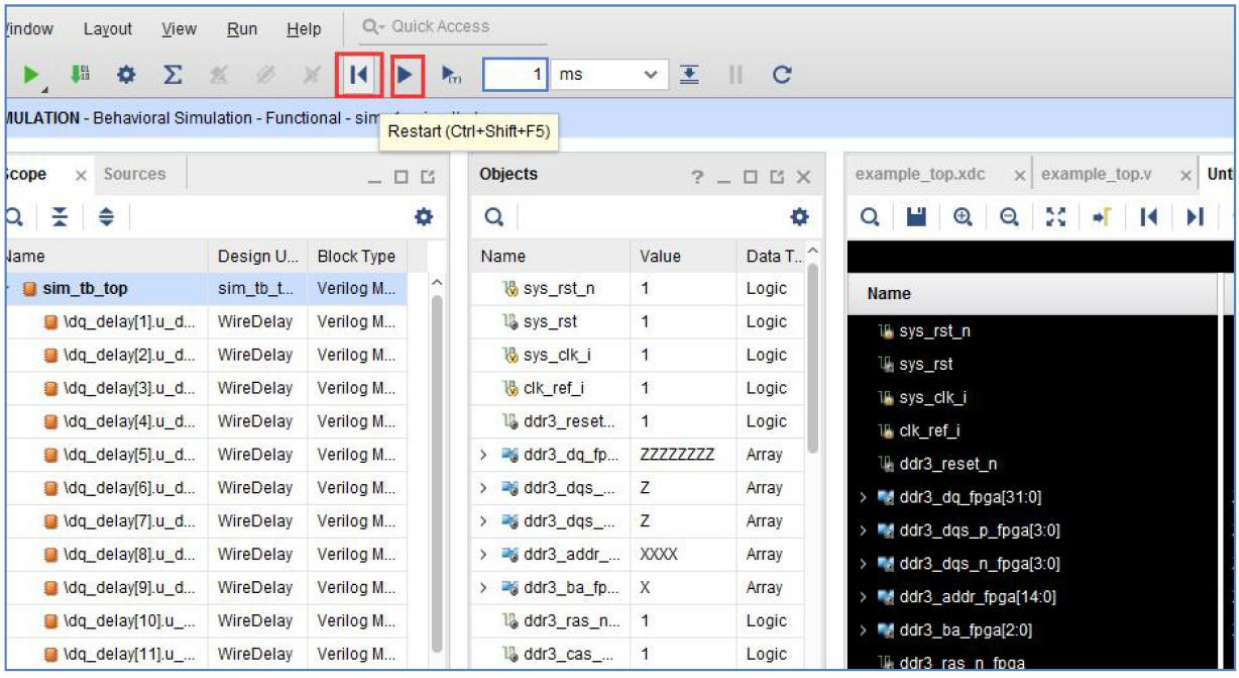
In this “ddr_ex” project, the software has automatically written the “ddr3” test program, the pin definition file “xdc” file and the simulation program.

2. Click Run Simulation button, Then select “Run Behavioral Simulation”, Here we can do a behavioral level simulation.

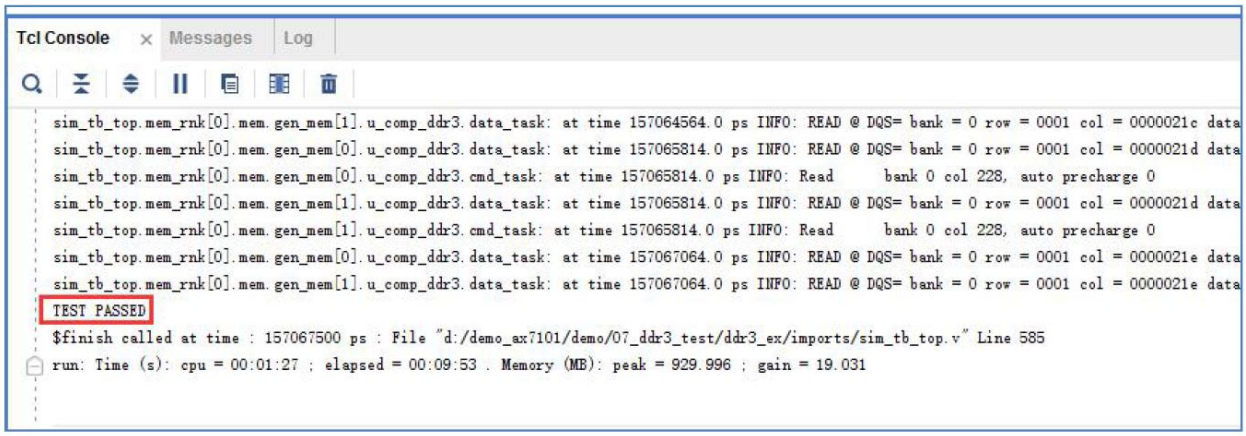
3. Set the simulation time to “1ms” .



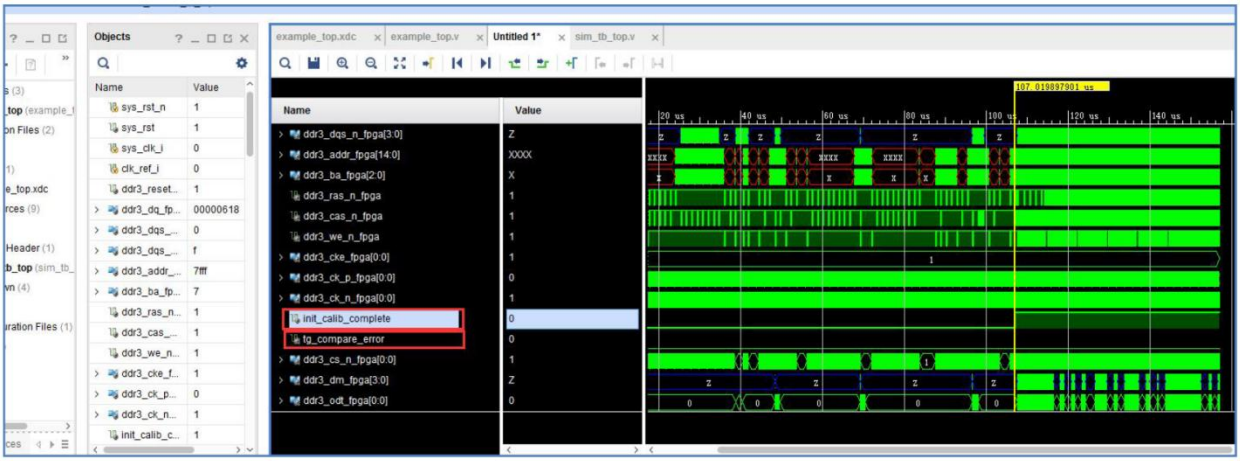
4. Click the "Restart" key and the "Run all" key to start waveform simulation.



5. The simulation time will be longer, and the simulation will be completed in about 155us. The word TEST PASSED will be displayed in the Tcl Console window, indicating that the DDR test is successful and there is no problem with the simulation.



6. Let's take a look at the waveform again. At about 100us, the init_calib_done signal becomes high, indicating that the ddr initialization is successful, and the later time is the waveform simulation of DDR data reading and writing. tg_compare_error is always low, indicating that the data read and written is correct.



If you are interested, you can combine the programs in the project to see each variable in the ddr simulation waveform, which will not be introduced here.

So far, the DDR testing is all done, the **mem_burst** module in this lab is very important, it simplifies the difficulty of our DDR IP interface and timing control, and we will also use it in future routines. In addition, you can refer to the "ug586_7Series_MIS.pdf" document provided by Xilinx for more information on the DDR IP controller.