开发板购买链接：https://xiaomeige.taobao.com/

推荐 1：高性能 FPGA 开发板 AC6102,：USB3.0+DDR2+千兆以太网：
https://item.taobao.com/item.htm?spm=a1z10.1-c-s.w4004-13687301132.2.GtzY11&id=541620465496

推荐 2：高性价比全功能 FPGA 学习型开发板：以太网+音频+12 位 ADDA
https://item.taobao.com/item.htm?spm=2013.1.w4023-13687301121.4.UTSDWG&id=544830995588

声　　明

本文如有更新或修改，将不另行通知，如有疑问，可以邮件咨询小梅哥，或前往芯航线 FPGA 技术支持群(615381411)寻找最新版本。

如果用户发现本文有任何错误或者难以理解的地方，欢迎发送邮件给小梅哥沟通（xiaomeige_fpga@foxmail.com）

严禁一切未经作者授权的转载或商业使用，如用于培训班、开发板配套资料等。

DDR2 简明教程

SDRAM 基本概念

SDRAM 的全称即同步动态随机存储器（Synchronous Dynamic Random Access Memory），这里的同步是指其时钟频率与对应控制器（CPU/FPGA）的系统时钟频率相同，并且内部命令的发送与数据传输都是以该时钟为基准；动态是指存储阵列需要不断的刷新来保证数据不丢失；随机指数据的读取和写入可以随机指定地址，而不是必须按照严格的线性次序变化。

在 FPGA 和 DSP 系统中，包括现在的 MCU 系统，我们还经常见到一种名为 SRAM 的器件，该器件名字和相比 SDRAM 少了一个 D，也就表明该存储器不存在 SDRAM 的动态刷新特征。同时，这里 SRAM 的 S 也不是代表同步的意思。相反，SRAM 属于异步器件，其在工作时是不需要外部提供时钟的，SRAM 的全称叫做 Asynchronous Static RAM，即异步静态随机存储器。

在存储数据的物理结构方面，SDRAM 使用电容的电荷存储特性存储数据，而 SRAM 使用 CMOS 晶体管存储数据，因此，从这个结构方面也就决定了 SDRAM 的运行功耗要远远低于 SRAM。同时，由于使用晶体管存储数据，要能够正确的存储一位数据，需要最少 6 个晶体管。因此，从芯片面积上来说，单片 SRAM 芯片的容量不可能做到很高。目前常见的有 512K 字节和 1024K 字节容量的，而 SDRAM 则可以最多做到 512Mbit（64M 字节）。
认识动态存储器和静态存储器。

表 1-1. DRAM 和 SRAM 的区别

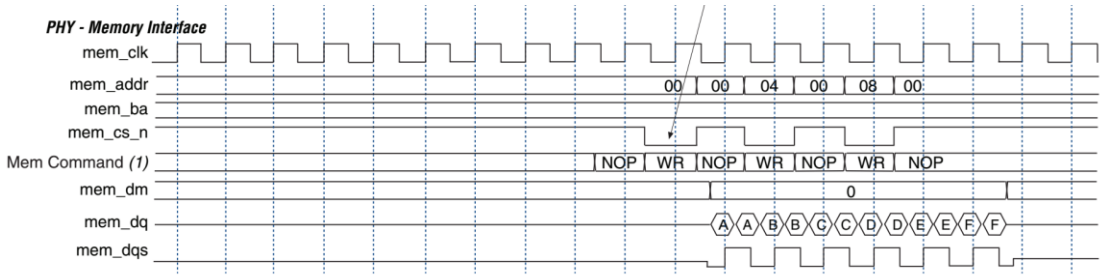
存储器类型	说明	带宽和速度	成本	数据存储容量和能力	功耗	延迟
DRAM	一个动态随机访问存储器（DRAM）单元由一个电容器和一个单一的晶体管组成。DRAM 存储器必须定期进行刷新来保持数据，这导致整体上降低了效率，以及更复杂的控制器。通常，设计师选择 DRAM，其中每个位的成本和容量都是重要的。DRAM 通常用于主要的存储器。	较低的带宽导致较慢的速度	较低的成本	更高的数据存储和能力	较高的功耗	较高的延迟

SRAM	一个静态随机访问存储器（SRAM)单元由六个晶体管组成。SRAM 不需要刷新，因为只要电源未被切断，晶体管就会继续保持数据。通常，设计师选择 SRAM，它的速度比容量更重要。SRAM 通常用于高速缓冲存储器。	较高的带宽导致较快的速度	较高的成本	更低的数据存储和能力	低功耗	低延迟

DDR SDRAM

首先搞清楚，DDR 的全称是 DDR SDRAM，因此其本质也是 SDRAM。

DDR 即双数据速率（Double Data Rates），DDR SDRAM 就应该是“双数据速率同步动态随机存储器”。相较于 SDRAM，DDR SDRAM 多了一个名词叫双数据速率，那么什么是双数据速率？我们通过 DDR 接口的一张时序图进行分析：



SDR、DDR、DDR2 演进之路

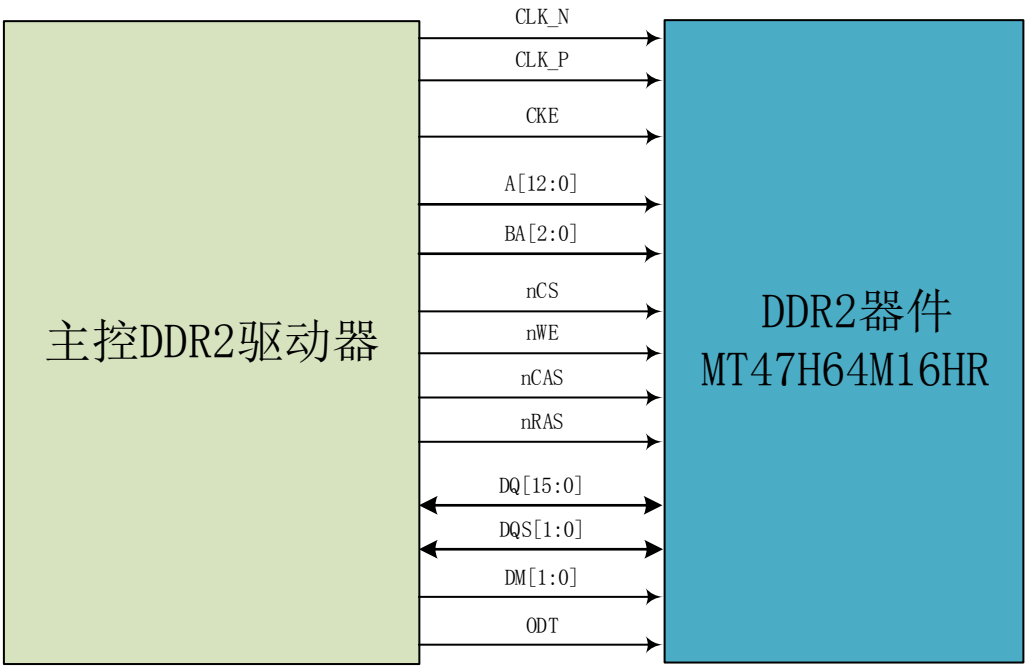
DDR 设备：

1. 功能上实现双速率
2. 增加了差分时钟线（clk，clk_n）
3. 电压等级 SSTL_25（2.5V）
4. 增加了数据线（DQ）的随路时钟（DQS，DQS_n）
5. 速度达到 400M（Mbit/s）

DDR2 设备：

1. 电压等级进一步减低（SSTL_18）
2. 增加了 ODT（On-Die Termination）
3. 内部 4Bit 预读结构
4. 速度最高达到 1066（Mbit/s）

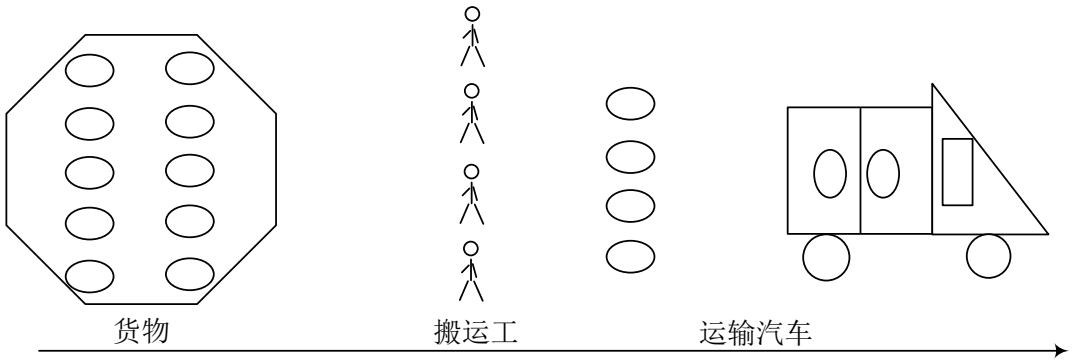
DDR2 器件与主控（CPU、FPGA、DSP）连接



管脚	说明	功能描述
CLK_P 、CLK_N	系统时钟，差分时钟	数据和命令接口的同步时钟，差分时钟信号
CKE	时钟使能	屏蔽系统时钟，以冻结当前操作，当该引脚为高电平时，所有引脚电平才能被正确送入 SDRAM 芯片。
CS_N	片选	用于屏蔽和使能所有输入端口，但不包括 CLK，CKE 和 DQM，低电平有效
RAS_N	行地址选通	该信号为低时，在时钟的上沿锁存行地址，使能行访问和预充电
CAS_N	列地址选通	该信号为低时，在时钟的上沿锁存列地址，使能列访问
WE_N	写使能	该信号为低时，使能写操作和预充电
BA[1:0]	Bank 地址	
A[12:0]	地址总线	关于地址线上的数据，在不通的命令中有不同的意义。详见下文对 A[12:0]的功能描述。

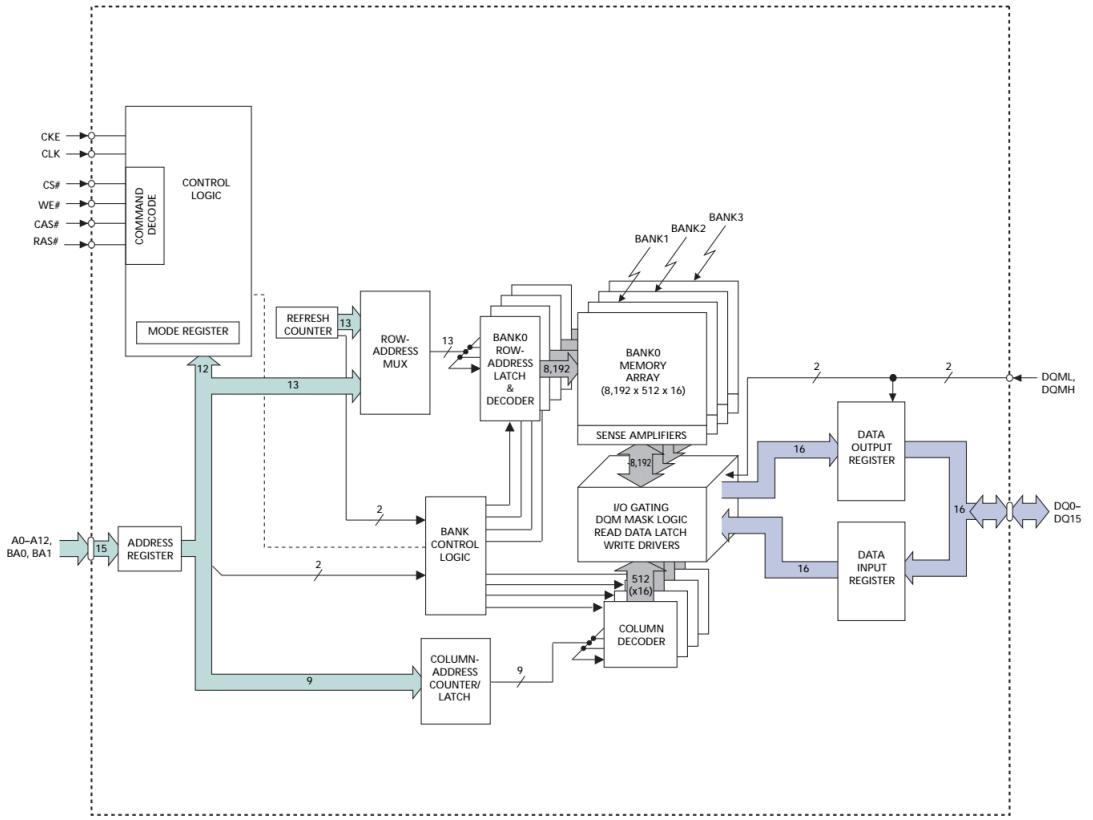
DQM[1:0]	数据掩码	L(U)DQM，低（高）字节掩码，当其为高时，下一个时钟的上沿，数据总线的低（高）字节为高阻态
DQ[15:0]	数据总线	数据输入输出复用
DQS[1:0]	随路时钟	dq 在 dqs 的上沿和下沿翻转，当主控读取 DDR2 存储器时，DQS 由 DDR2 存储器驱动，当主控写入 DDR2 存储器时，DQS 由主控驱动。
ODT	时钟使能	片上终端电阻控制线（控制 dq，dqs，dm，ddr 接收时启动）

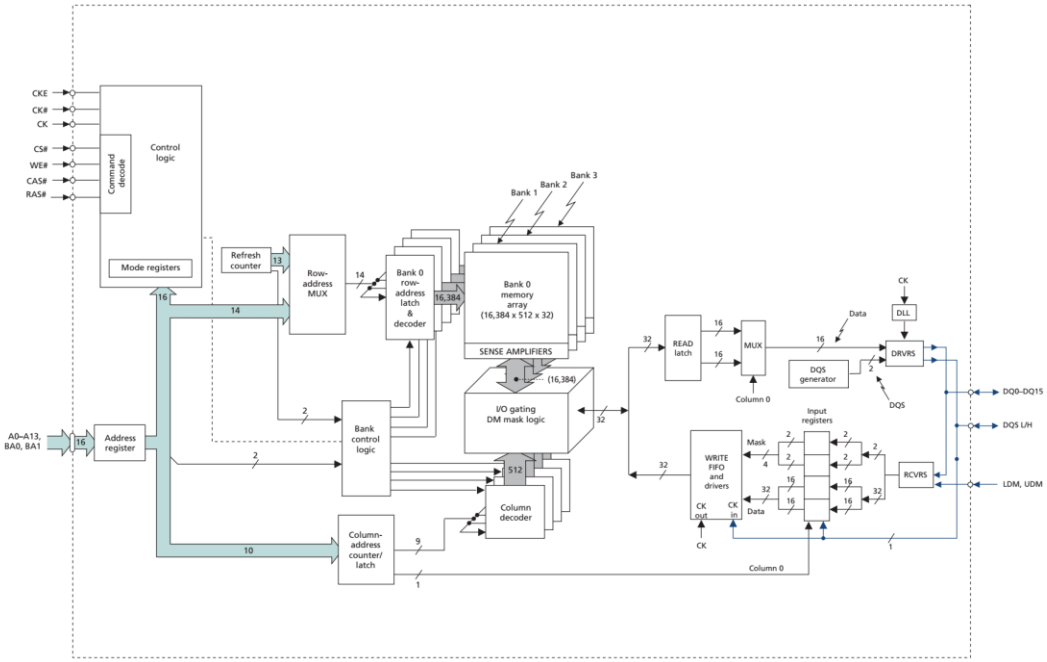
4bit 预读概念举例

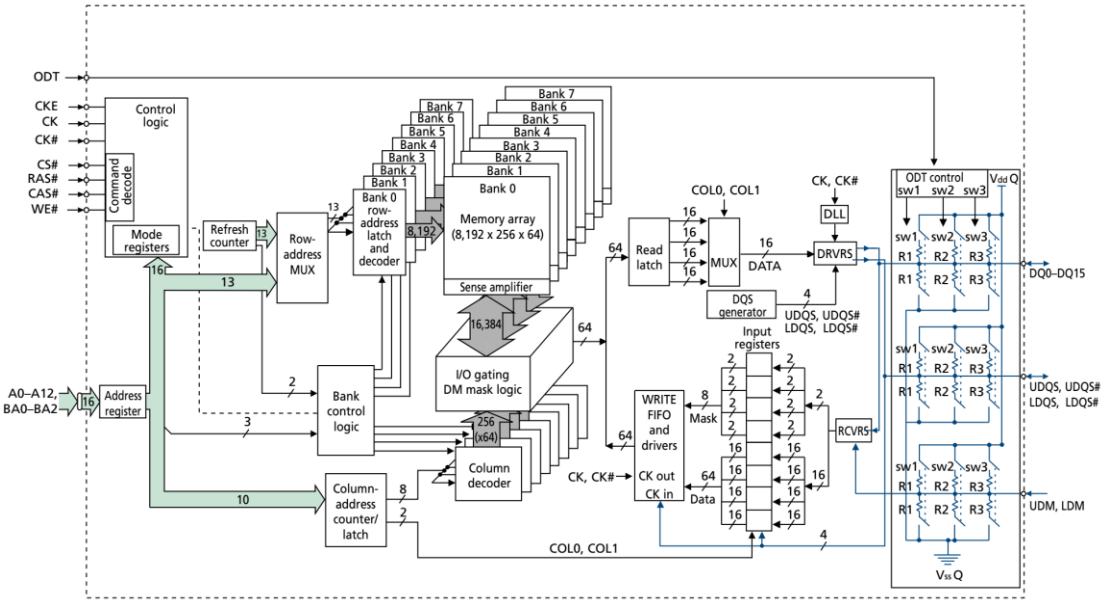


货物：相当于 DDR2 中的存储器单元，每个圆圈货物相当于 1 位存储单元
搬运工：预读取总线，每个搬运工相当于一个通道的预读取总线
运输汽车：接口时钟，汽车每运输一次相当于一个接口时钟周期传输一次数据，那么将 4bit 的预读取数据传输完成，最起码需要该汽车运输两次。

对比 SDR、DDR、DDR2 接口电路







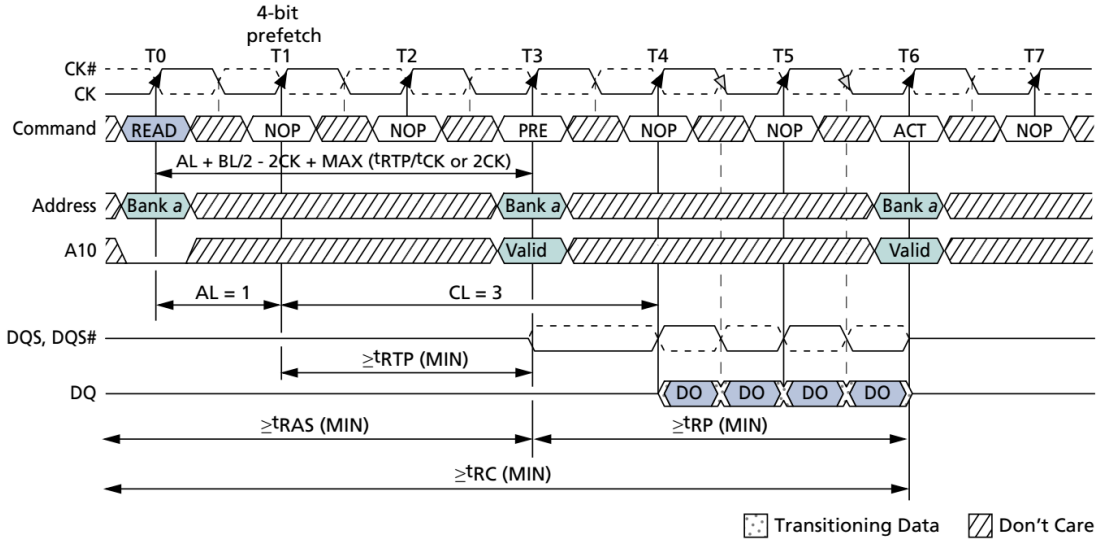
关于突发长度的思考问题：

SDR SDRAM 的突发长度为 1、2、4、8、page
DDR SDRAM 的突发长度为 2、4、8
DDR2 SDRAM 的突发长度为 4、8

典型的 DDR2 传输时序实例：

突发长度为 4 的读操作，不带自动预充电

重点看信号与时钟的对齐关系



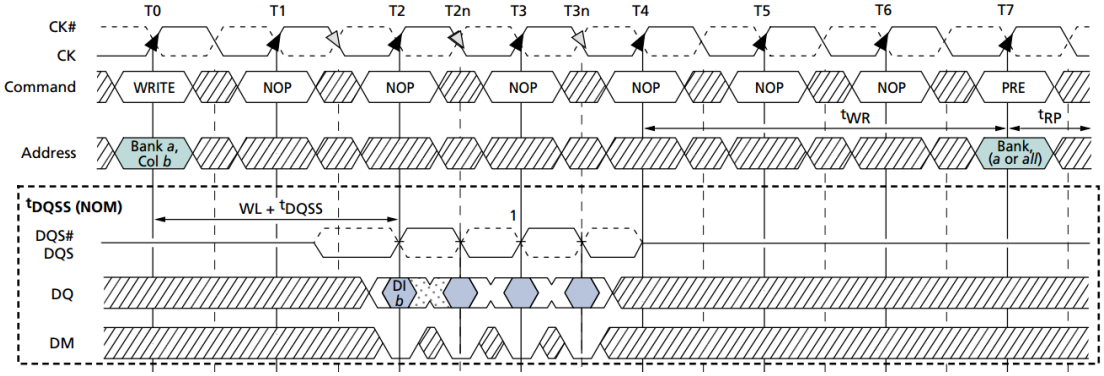
: 1. RL = 4 (AL = 1, CL = 3); BL = 4.
2. ^{t}RTP ≥ 2 clocks.
3. Shown with nominal ^{t}AC, ^{t}DQSCK, and ^{t}DQSQ.

对于读操作：

- DQS 信号由 DDR2 存储器发出
- 控制信号、地址信号与时钟上升沿采用中心对齐方式
- DQ 和 DQS 采用边沿对齐方式

突发长度为 4 的写操作，不带自动预充电

重点看信号与时钟的对齐关系



对于写操作：

- DQS 由主控制器发出
- 控制信号、地址信号与时钟上升沿采用中心对齐方式
- DQ 和 DQS 采用中心对齐方式

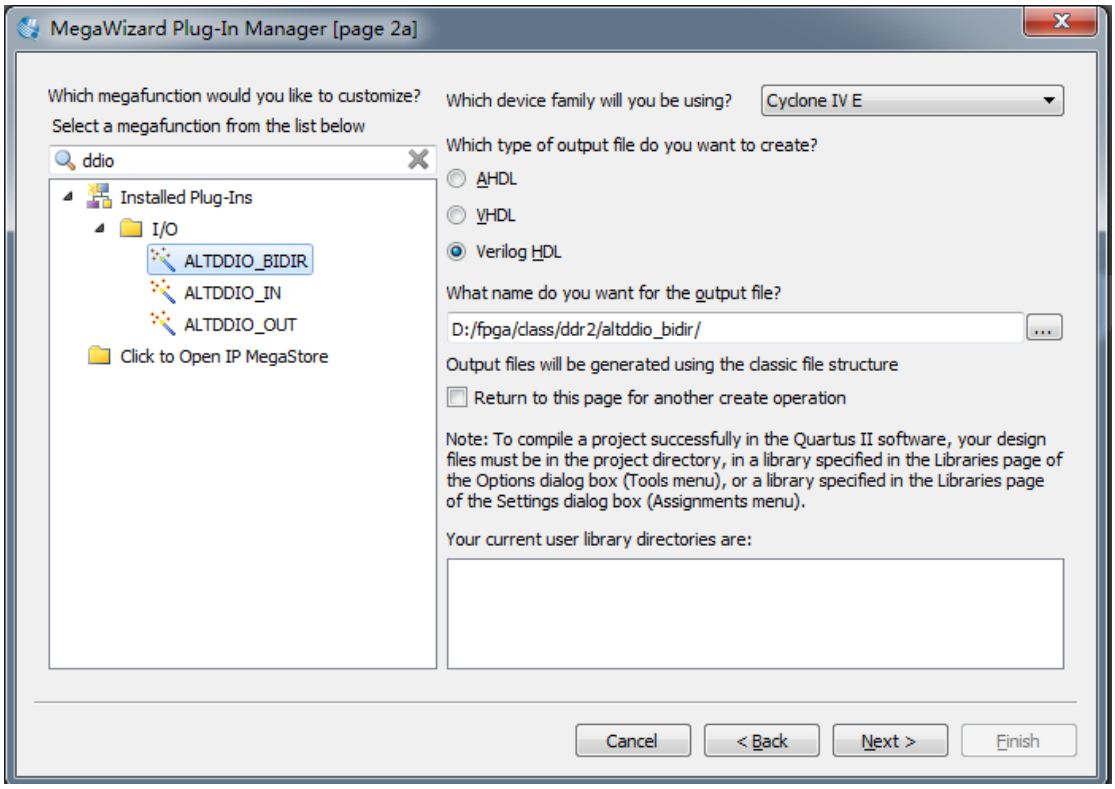
总结：

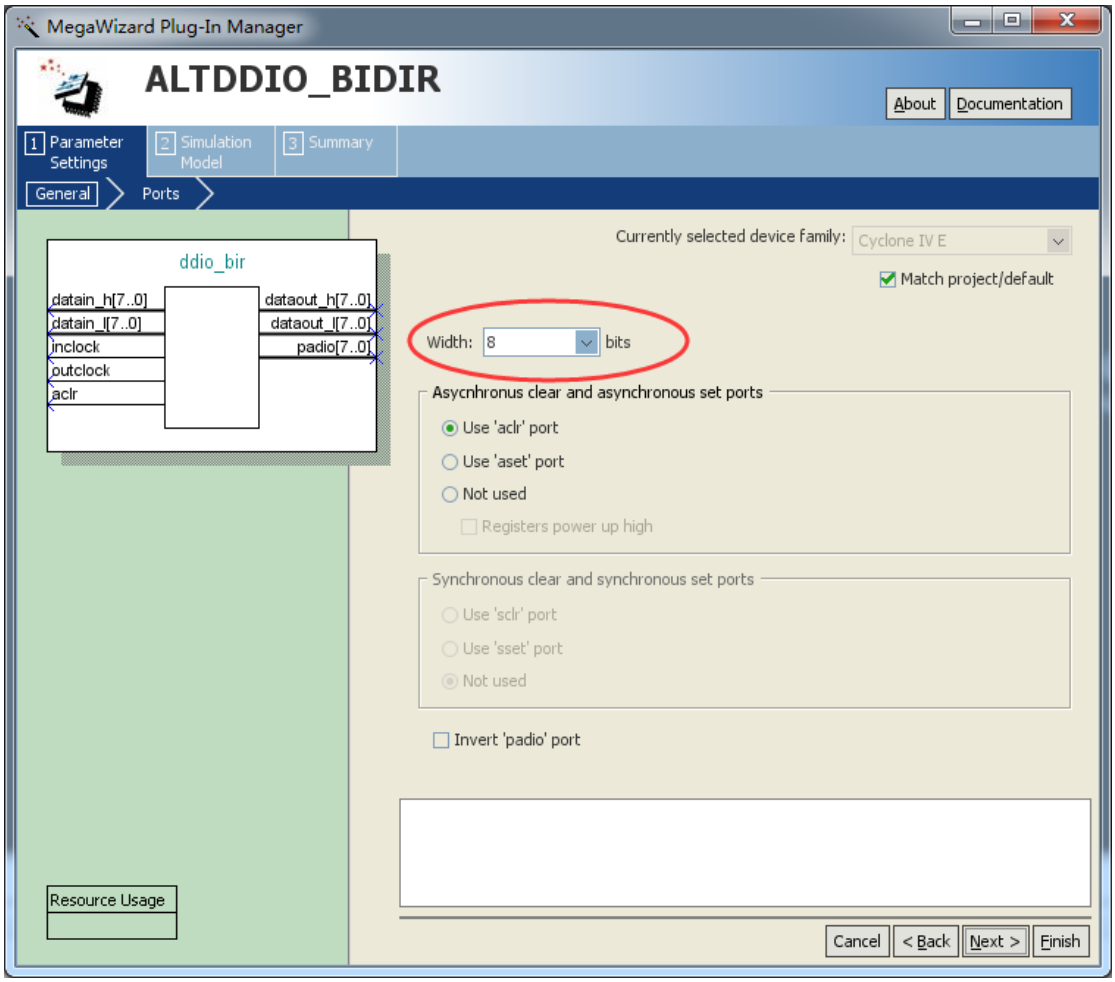
cmd（含 odt，dm）均在时钟上沿被采样（与时钟中心对齐）
cmd（含 odt，dm）有效宽度为半个 clk 周期（sdram 为一个 clk 周期）
dqs 将与 dq 共同走线，构成随路时钟
在写操作时，数据由 DDR2 控制器（FPGA）发出，由于 DDR2 控制器工作时钟与 PHY 时钟的相位差，而理论上 DQS 信号是与 PHY 时钟相位同步的，因此数据呈现出与 DQS 的中心对齐现象。
读操作时，dq 和 dqs 构成边沿对齐，对于 DDR2 控制器，由于控制器工作时钟与 PHY 时钟存在相位差，因此刚好可以实现对于 DDR2 控制器的读逻辑呈现中心对齐。
dqs 和 dq 一样，是双向总线
dqs 从三态转为第一个上沿，经历前导过程，称为前导码（preamble）
dqs 从最后一个上沿转为三态，经历后导过程，称为后导码（postmble）

FPGA 的 DDR 接口和 DDR 存储器控制器解决方案

Intel（原 Altera）Cyclone IV E FPGA 如何实现双数据速率

为了实现 DDR 接口，Intel（原 Altera）Cyclone IV E FPGA 的部分管脚支持双数据速率传输。在 Quartus II 软件中，我们可以通过调用 altddio IP 核来使用双数据速率 IO。双数据速率 IO 包括 altddio_bidir（双向双速率 IO）、altddio_in（输入型双速率 IO）、altddio_out（输出型双速率 IO）。通过调用双速率数据 IO，就能够实现双数据速率传输了，以下通过仿真的方式来体验 altddio_bidir 的使用方法。

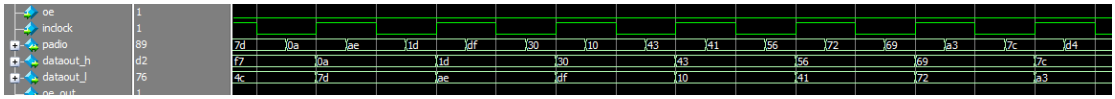

MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
ddio
Installed Plug-Ins
I/O
ALTDDIO_BIDIR
ALTDDIO_IN
ALTDDIO_OUT
Click to Open IP MegaStore
Which device family will you be using?
Cyclone IV E
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
D:/fpga/class/ddr2/altddio_bidir/
Output files will be generated using the classic file structure
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in a library specified in the Libraries page of the Options dialog box (Tools menu), or a library specified in the Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager
ALTDDIO_BIDIR
About
Documentation
1 Parameter Settings
2 Simulation Model
3 Summary
General
Ports
ddio_bir
datain_h[7..0]
datain_l[7..0]
inclock
outclock
aclr
dataout_h[7..0]
dataout_l[7..0]
padio[7..0]
Currently selected device family:
Cyclone IV E
Match project/default
Width: 8 bits
Asycnhronus clear and asynchronous set ports
Use 'aclr' port
Use 'aset' port
Not used
Registers power up high
Synchronous clear and synchronous set ports
Use 'sclr' port
Use 'sset' port
Not used
Invert 'padio' port
Resource Usage
Cancel
< Back
Next >
Finish

IP 端口：

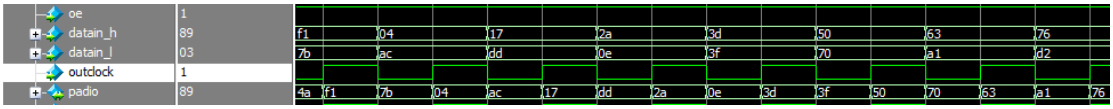
```
module ddio_bir (
    aclr,   //异步清零信号
    datain_h,   //输出数据高段输入端口，用户逻辑
    datain_l,   //输出数据低段输入端口，用户逻辑
    inclock,    //数据输入时钟，由管脚输入
    oe, //数据输入时钟，由管脚输入
    outclock,   //数据输出时钟，由片上电路，如 PLL 产生
    dataout_h,  //输入数据高段输出端口，用户逻辑
    dataout_l,  //输入数据低段输出端口，用户逻辑
    oe_out,
    padio   //  DDR 数据管脚，双向管脚
);

    input     aclr;
    input   [7:0]  datain_h;
    input   [7:0]  datain_l;
    input     inclock;
    input     oe;
    input     outclock;
    output  [7:0]  dataout_h;
    output  [7:0]  dataout_l;
    output    oe_out;
    inout   [7:0]  padio;
```

输入



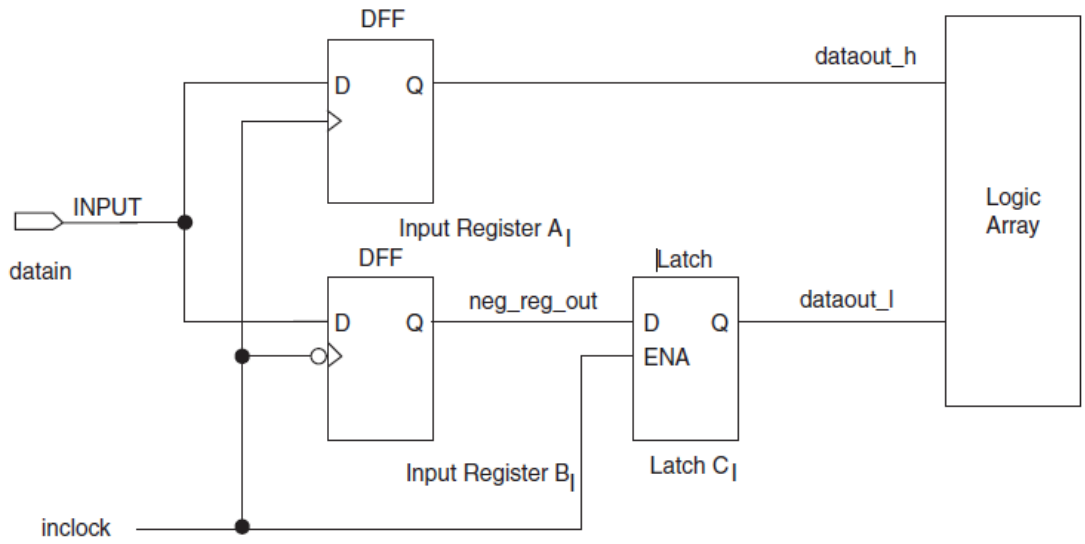
inclk 上升沿锁定高字段
inclk 下降沿锁定低字段

输出



outclk 上升沿输出高字段
outclk 下降沿输出低字段

altddio_in 电路结构如下所示：



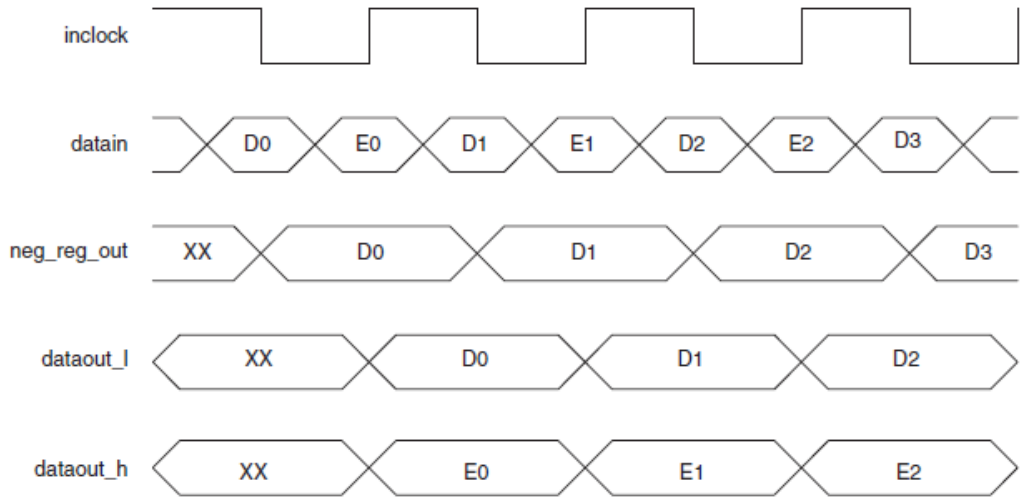
下降沿：BI 寄存数据

低电平：latch 处于非锁存状态，输出 Q 和输入 D 断开，输出依旧保持之前值不变

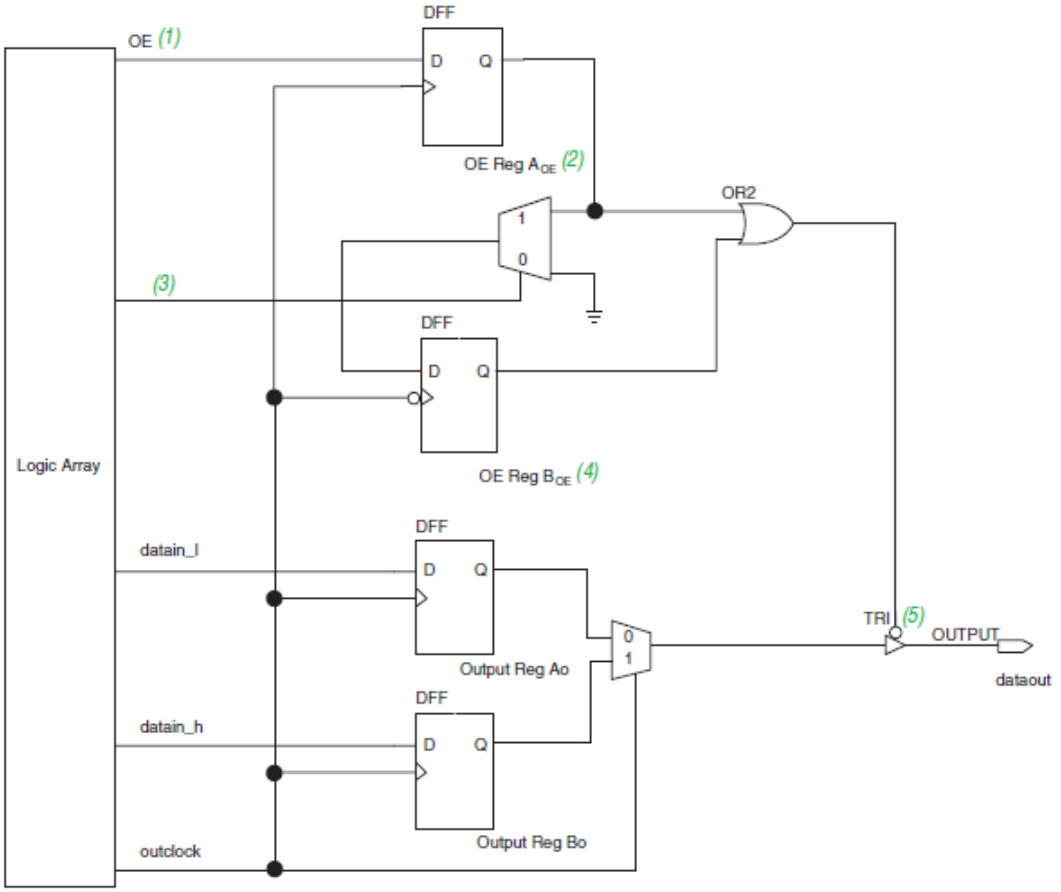
上升沿：AI 寄存数据

高电平：latch 处于锁存状态，输出 Q 和输入 D 直通，输出等于输入，即 BI 的值。

I/O Input Timing Waveform



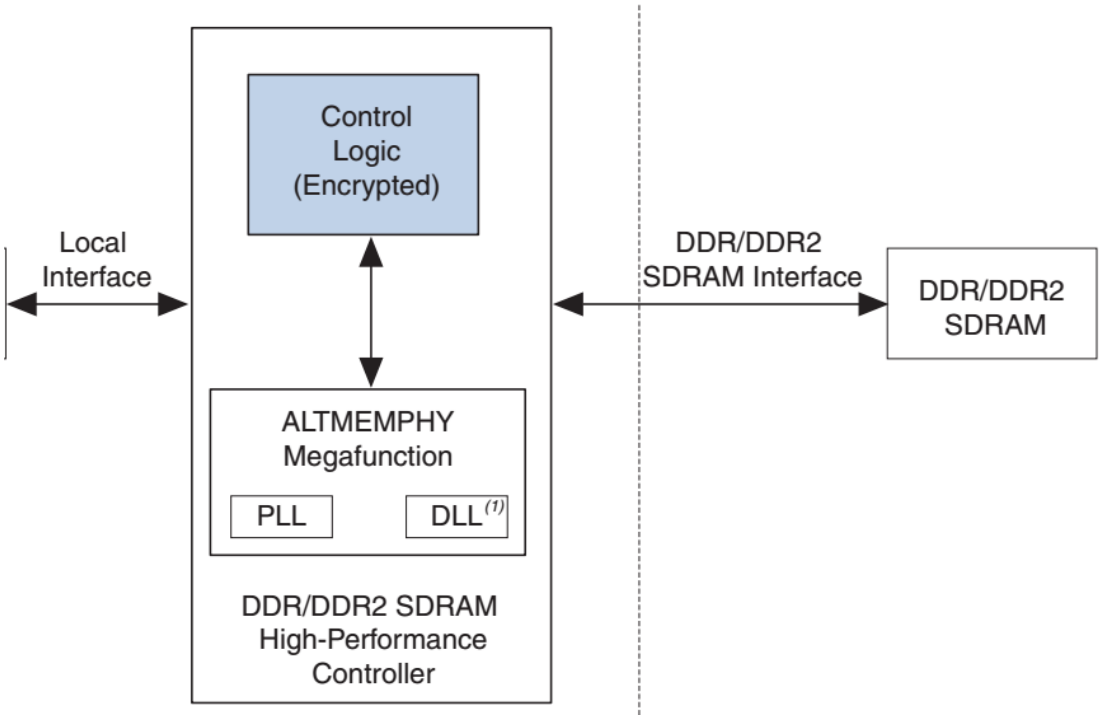
altddio_out 电路结构如下所示：



datain_L 和 datain_H 通过寄存器后接到一个二选一多路器，该二选一多路器的输出选择由 outclk 控制，当 outclk 为高电平，输出 datain_H，当 outclk 为低电平，输出 datain_L，OE 信号通过控制三态门来控制该 pad 是否处于输出状态。

Intel Cyclone IV E FPGA DDR2 存储器控制器解决方案

Intel（原 Altera）Cyclone IV E FPGA 提供了 DDR2 控制器 IP，该 IP 分为两部分，控制逻辑和物理层数据通路。其中物理层数据通路用户可以通过调用 ALTMEMPHY Megafunction IP 来使用，该 IP 核是非加密的，用户使用不需要 license。控制逻辑部分是加密的，用户使用该部分需要有相应的 license。默认的 DDR2 控制器是使用了控制逻辑+ALTMEMPHY 的结构，用户使用该控制器需要获得 license。当然，对于用户，我们也可以完全通过自己编写逻辑代码来实现控制逻辑，并在我们自己的控制逻辑中调用 ALTMEMPHY 来实现完整的 DDR2 控制器。此种方式需要用户对 DDR2 存储器原理和时序有非常深入的了解。



由于 DDR2 工作中需要进行刷新操作，读写切换过程中也会有一定的效率损失。因此 DDR2 的读写效率不可能达到 100%。针对该控制器，Altera 提供了一个效率说明文件。《wp_ddr_sdram_efficiency》，文件中说，该控制器效率在最好的情况下可达 92%。当然，这种情况除非特定应用，否则一般很难达到。

Efficiency—Best-Case Scenario

The best-case scenario arises with long bursts of writes followed by long bursts of reads. For example, in the case of back-to-back writes of length 4 (local side), the efficiency (ignoring refresh) is approaching 100%. However, that is clearly an unrealistic use of the memory. A more typical best case scenario is 10 writes (burst of 4 local side) to the same column followed by 10 reads.

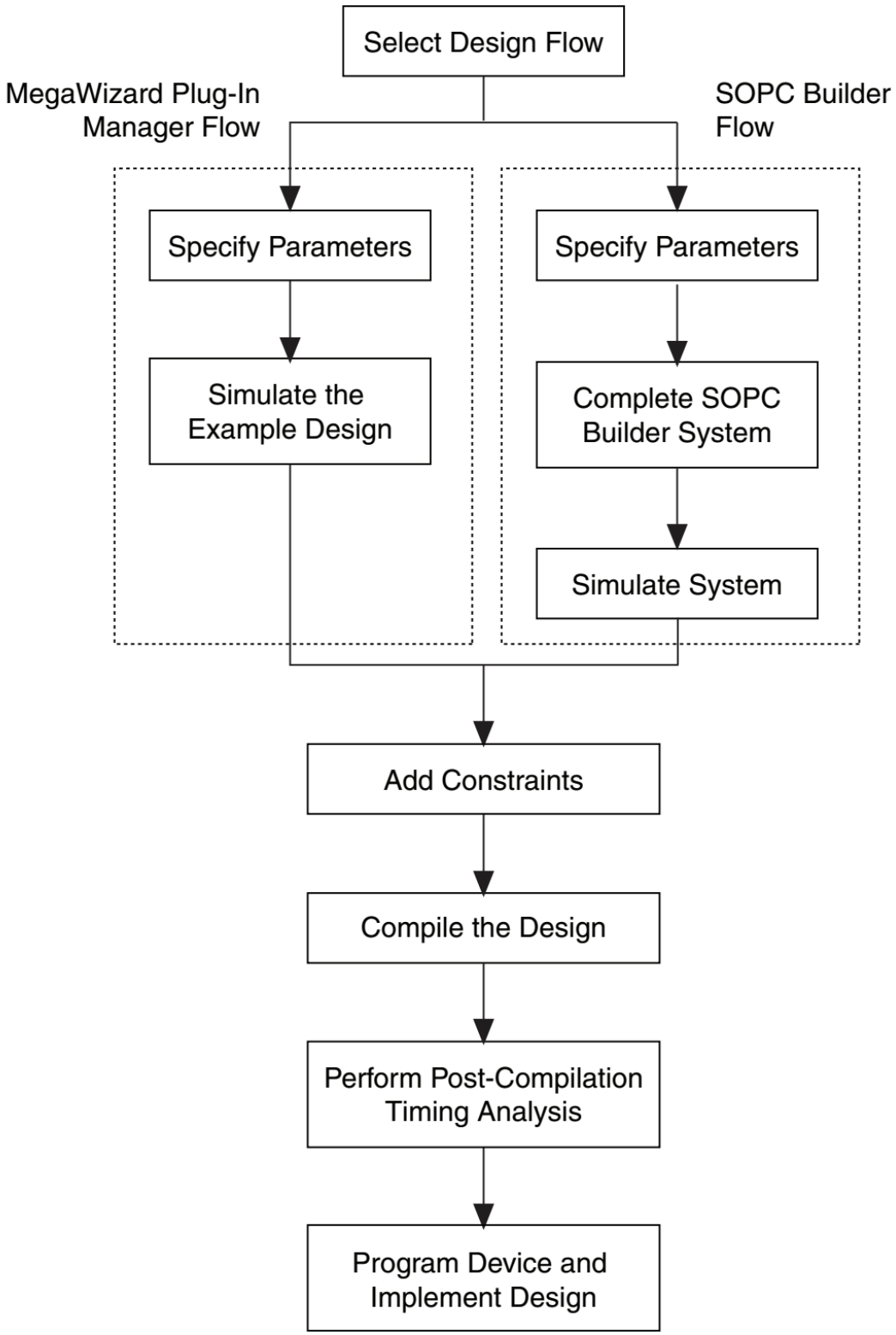
The efficiency in this best-case scenario is around 92%.

The Effect of Refresh

DDR2 控制器使用方式

DDR2 SDRAM High-Performance Controller 可用于 SOPC 系统或者用户逻辑。当该存储器控制器用于 SOPC 系统中时，使用 Avalon MM 接口，当该存储器控制器用于用户逻辑时，使用 local interface。当使用用户逻辑时，可以得到较高的使用频率。下图为 DDR2 控制

器的使用流程图。


Select Design Flow
MegaWizard Plug-In Manager Flow
SOPC Builder Flow
Specify Parameters
Simulate the Example Design
Specify Parameters
Complete SOPC Builder System
Simulate System
Add Constraints
Compile the Design
Perform Post-Compilation Timing Analysis
Program Device and Implement Design

当将 DDR2 控制器用于 SOPC 系统时，我们可以通过 Avalon MM master 主机对 DDR2 存储器进行读写，该主机可以是处理器如 NIOS II CPU，也可以是其他带有 Avalon MM 主机接口的逻辑，如 SGDMA IP、FrameReader、自定义 Avalon MM 主机 IP。在接下来的课程中，我们首先体验 DDR2 控制器的 SOPC 开发流程，该流程我们主要完成三个实验：

1. 实验 1 使用 DDR2 作为 NIOS II 的主运行内存，使用 NIOS II 完成对 DDR2 存储器的读写测试。
2. 实验 2 使用 DDR2 作为 NIOS II 的主运行内存，并共享为 5 寸 800*480 分辨率显示屏的显示缓存，通过带 Avalon MM 主机读接口的自定义 VGA_TFT 控制器，直接从 DDR2 中实时读取需要显示的数据并显示在 TFT 屏上。
3. 实验 3 使用 DDR2 作为 NIOS II 的主运行内存，并共享为 5 寸 800*480 分辨率显示屏的显示缓存。同时支持一个带 Avalon MM 主机写接口的自定义摄像头采集控制器，实现摄像头数据的采集并存储在 DDR2 中，然后再由 5 寸屏将图像显示在屏幕上。

当将 DDR2 控制器用于用户逻辑时，我们可以通过 local interface 接口对 DDR2 存储器进行读写操作，该流程我们主要完成两个实验：

1. DDR2 存储器读写测试。
2. 使用 DDR2 存储器配合 CMOS 摄像头和 VGA 控制器实现实时图像采集显示系统。

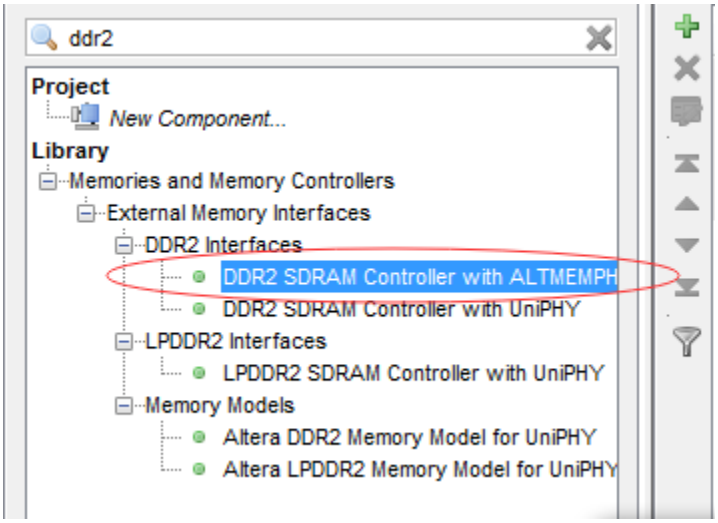
DDR2 在 SOPC 系统中的应用

创建 quartus ii 工程

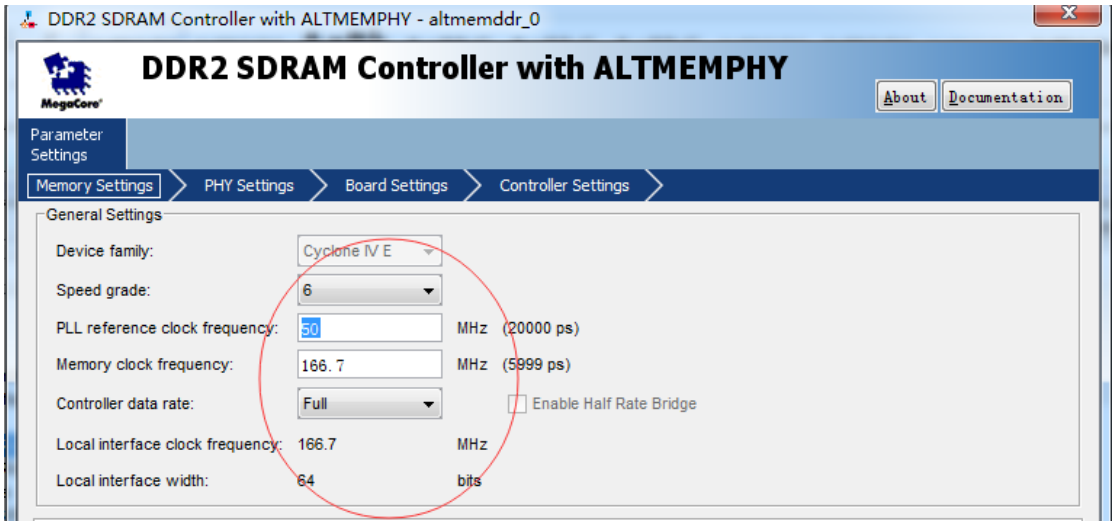
新建 qsys 项目

添加 nios CPU

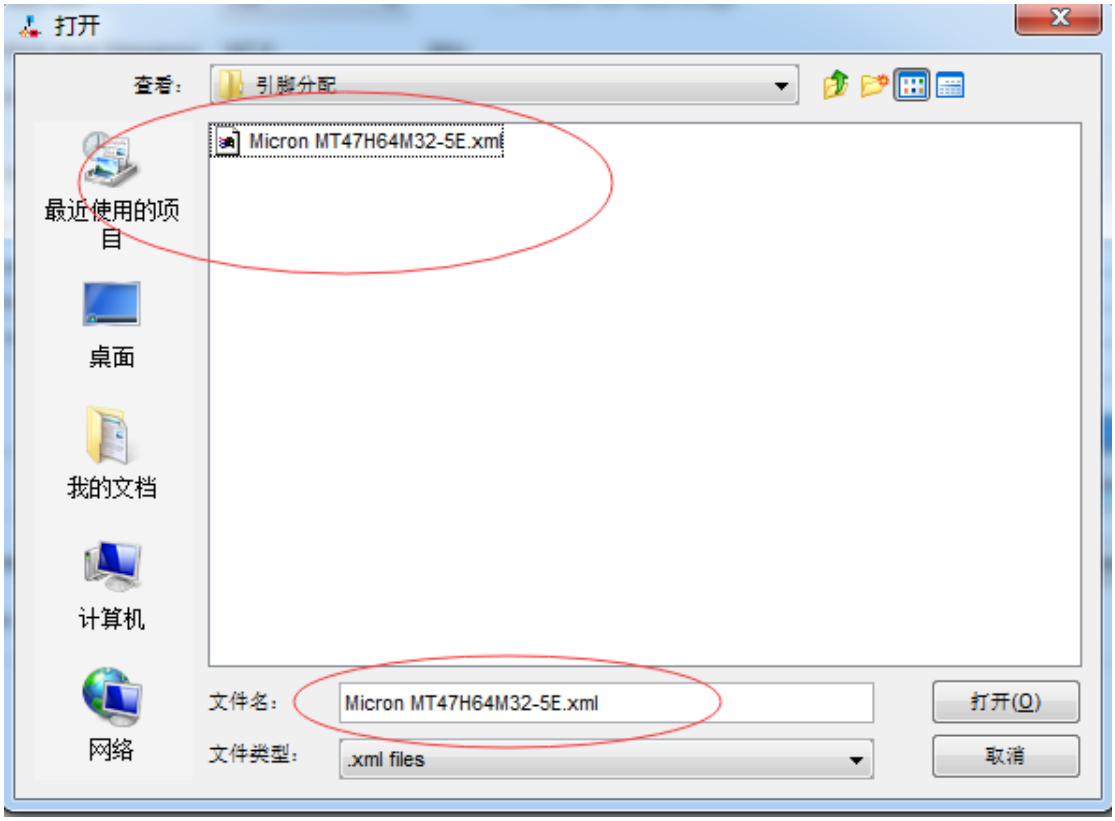
添加 DDR2 控制器

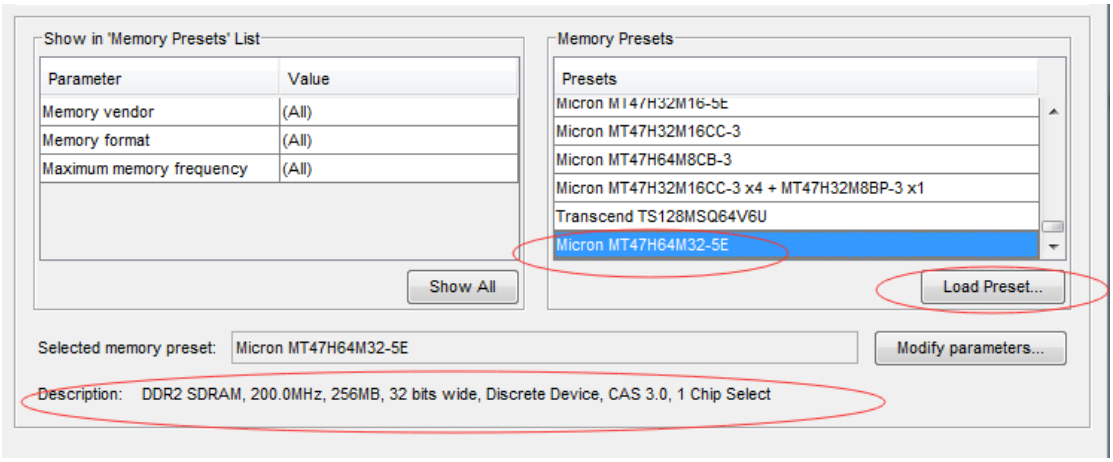


设置控制器参数：
PLL 参考时钟：50M
存储器时钟：166.7M
控制器数据速率：full

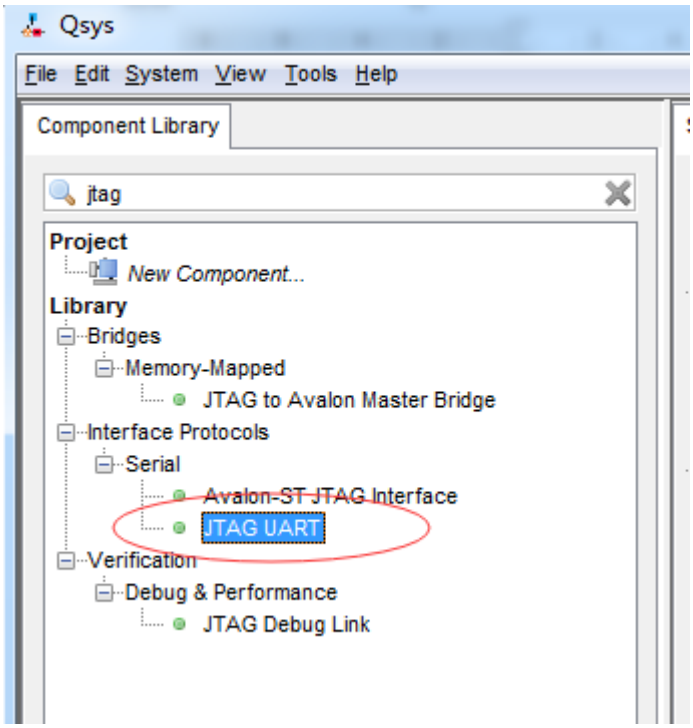


加载硬件 32bit DDR2 存储器（两片 16bit 存储器组合而成）参数：Micron MT47H64M32-5E.xml

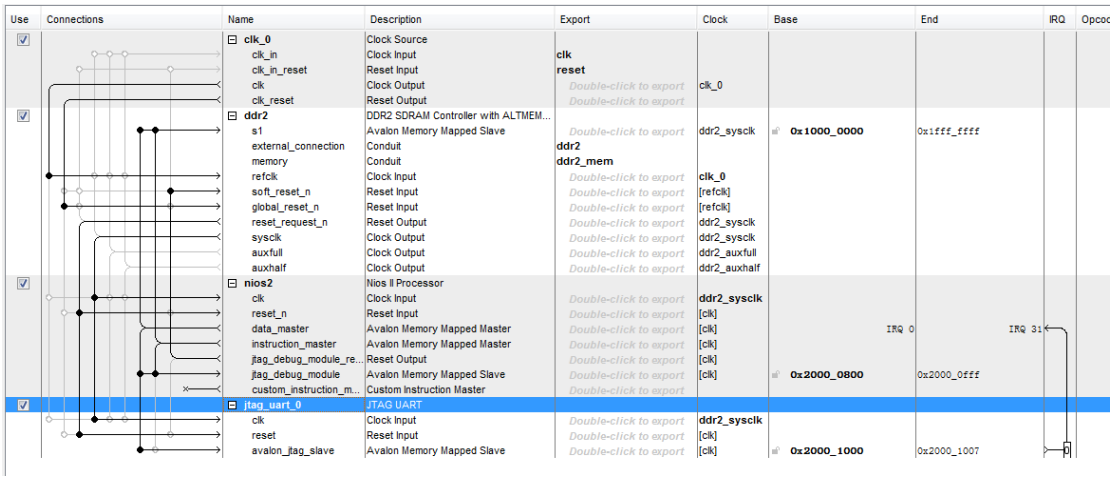




添加 jtag uart



连接时钟和复位信号



连接要点：
Ddr2 的 reset_request_n 作为全局复位。
Soft_reset_n 连接 debug 模块或者 watchdog 模块的复位输出
Global_reset_n 连接外部复位输入逻辑。

External_connection 和 memory 导出。

Sysclk：DDR2 主时钟，本例中为 166.7M
Auxfull：全速模式下和 sysclk 频率相同，半速模式下是 sysclk 的两倍
Auxhalf：全速模式下频率是 sysclk 频率一半，半速模式下和 sysclk 相同

设置 NIOS II 复位地址和异常地址

External Memory Interface Width	Reset Vector Offset		Exception Vector Offset	
	AFI	Non-AFI	AFI	Non-AFI
8	0×40	0×20	0×60	0×40
16	0×80	0×40	0×A0	0×60
32	0×100	0×80	0×120	0×A0
64	0×200	0×100	0×220	0×120

编辑工程顶层。将 DDR 的引脚全部以 ddr2_mem 打头

```
    mysystem u0 (
        .clk_clk                (clk),                //      clk.clk
        .reset_reset_n          (reset_n),          //    reset.reset_n
        .ddr2_local_refresh_ack (), //     ddr2.local_refresh_ack
        .ddr2_local_init_done   (),   //         .local_init_done
        .ddr2_reset_phy_clk_n   (),   //         .reset_phy_clk_n
        .ddr2_mem_mem_odt       (ddr2_mem_odt),       // ddr2_mem.mem_odt
        .ddr2_mem_mem_clk       (ddr2_mem_clk),       //         .mem_clk
        .ddr2_mem_mem_clk_n     (ddr2_mem_clk_n),     //         .mem_clk_n
        .ddr2_mem_mem_cs_n      (ddr2_mem_cs_n),      //         .mem_cs_n
        .ddr2_mem_mem_cke       (ddr2_mem_cke),       //         .mem_cke
        .ddr2_mem_mem_addr      (ddr2_mem_addr),      //         .mem_addr
        .ddr2_mem_mem_ba        (ddr2_mem_ba),        //         .mem_ba
        .ddr2_mem_mem_ras_n     (ddr2_mem_ras_n),     //         .mem_ras_n
        .ddr2_mem_mem_cas_n     (ddr2_mem_cas_n),     //         .mem_cas_n
        .ddr2_mem_mem_we_n      (ddr2_mem_we_n),      //         .mem_we_n
        .ddr2_mem_mem_dq        (ddr2_mem_dq),        //         .mem_dq
        .ddr2_mem_mem_dqs       (ddr2_mem_dqs),       //         .mem_dqs
        .ddr2_mem_mem_dm        (ddr2_mem_dm)         //         .mem_dm
    );
```

添加 mysystem.qsys

对工程进行分析和综合

设置 SDC 时序约束
添加<*variation name*>_phy_ddr_timing.sdc 文件

设置引脚分配：
Clk50：G1
Reset_n：L8
DDR2：
可以使用我们提供的引脚分配脚本文件：AC6102_ddr2_pin_assignments.tcl：
也可以手动分配，分配时需要注意以下参数：

1. 电平标准：所有信号接口电平都为 SSTL-18 Class I

2. 驱动电流：
   a) 地址和控制信号全部选择最大驱动电流 Maximum Current（addr、ba、cas、cke、csn、ras、odt、wen）
   b) 数据线、时钟线选择 12mA。（clk、dq、dqm、dqs）

3. 输出使能：
   dq、dqm、dqs 分配在同一个输出使能组里。（如何打开输出使能组？）（如何设置同一个输出使能组？）（不设置会出现什么问题）

全编译工程

下载 sof 到开发板中

打开 eclipse 创建 memtest small 工程，运行并调试。

编写 DDR2 读写校验程序并验证。

```
#include <stdio.h>
#include "stdint.h"
#include "system.h"
#include "unistd.h"

unsigned short *DDR2_USER = (unsigned short *)(DDR2_BASE+0x100000);

int main()
{
    uint8_t  DDR2_flag=0;
    uint32_t  i=0;

//  printf("Hello from Nios II!\n");

    printf(" Xiaomeige AC6102  ");
    printf(" DDR2 32bit test   \n");

//  memset(DDR2_BASE,0,0xFFFF);

    while(1)
    {
        printf("DDR2 write data.");

       for(i=0;i<0x09FFFFF;i++)
       {
           *(DDR2_USER+i)=i&0xFFFF;
       }
```

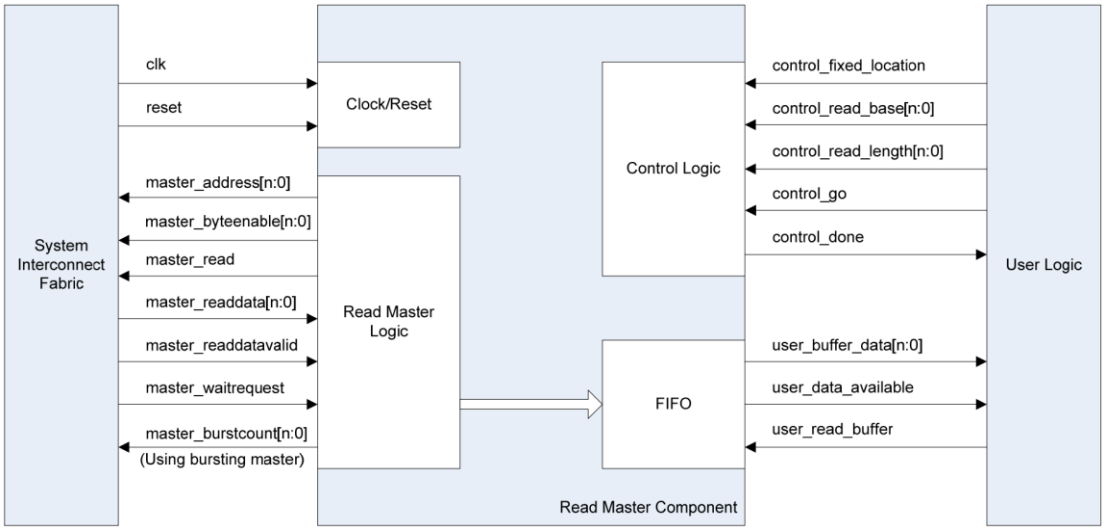
```
        *(DDR2_USER+0x0A5A5A5-0x10000) = 0xA5A5;
        *(DDR2_USER+0x15A5A5A-0x10000) = 0x5A5A;
        *(DDR2_USER+0x2A5A5A5-0x10000) = 0xA5A5;
        *(DDR2_USER+0x35A5A5A-0x10000) = 0x5A5A;

         usleep(10000);

         printf("DDR2 read data..");

        for(i=0;i<0x09FFFFF;i++)
       {
           if((i&0xFFFF)!=*(DDR2_USER+i))
           {
               DDR2_flag = 1;
           }
       }

      if(0xA5A5!=*(DDR2_USER+0x0A5A5A5-0x10000))
           DDR2_flag = 1;
      if(0x5A5A!=*(DDR2_USER+0x15A5A5A-0x10000))
           DDR2_flag = 1;
      if(0xA5A5!=*(DDR2_USER+0x2A5A5A5-0x10000))
           DDR2_flag = 1;
      if(0x5A5A!=*(DDR2_USER+0x35A5A5A-0x10000))
           DDR2_flag = 1;

      usleep(10000);

        if(!DDR2_flag)
        {
            printf("DDR2 test: OK!!!\n");
        }
        else
        {
            printf("DDR2 test: fail\n");
        }

         usleep(2000000);
    }

    return 0;
}
```

```
{
    printf("DDR2 write data.");

    for(i=0;i<0x09FFFFF;i++)
    {
        *(DDR2_USER+i)=i&0xFFFF;
    }
```

Problems | Tasks | Console | Properties | Nios II Console

```
New_configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
DDR2 write data.DDR2 read data..DDR2 test: OK!!!
```

基于 Avalon mm master 接口的逻辑分析仪系统演示与介绍。



Quartus II 13.0 中仿真 DDR2 IP 核

芯航线 2016 年最新产品 USB3.0 FPGA 开发板 AC6102 上提供了有 2Gb 的 DDR2 存储器。要想能够正确的对 DDR2 存储器进行操作，需要复杂的控制逻辑，这些控制逻辑我们称之为 DDR2 控制器。DDR2 控制器由两部分组成，一部分是 Altera

FPGA 芯片里面提供的硬线电路，该部分电路主要实现 DDR2 存储器的接口控制，由于 DDR2 的接口时钟较高，最高可达 400MHz（DDR2 800），使用 FPGA 的通用搭建的电路无法支持这么高的接口速率，所以，要想使用 DDR2 存储器，该部分电路必须使用，该电路称为（high_performance_controller）。另一部分为 DDR2 的读写控制逻辑，该部分主要实现对 DDR2 高效的读写操作，此部分可以使用通用的 FPGA 逻辑实现。我们可以自己编写 Verilog 代码实现，也可以使用 Altera 或第三方提供的 IP 核。该部分 IP 一般是收费的。一个可靠高效的 DDR2 控制器需要非常深厚的设计功底才能实现，因此一般情况下我们都使用 Altera 提供的 IP 核实现。使用该 IP 核需要获得授权，获得授权的方式有很多种，如找 altara 的代理购买，或者通过其他途径得到，这里不做过多介绍。

我们曾经在《FPGA 设计思想与验证方法视频教程》中提到过，对一个 IP 核最直观的学习方式之一就是对该 IP 核进行仿真，观察其工作时序。本节，我们不做过多的理论介绍，仅以一种 step by step 的方式代领大家学习下 DDR2 IP 核的仿真。至于对 IP 核参数的设置和仿真结果的具体分析，本节不做详细介绍。

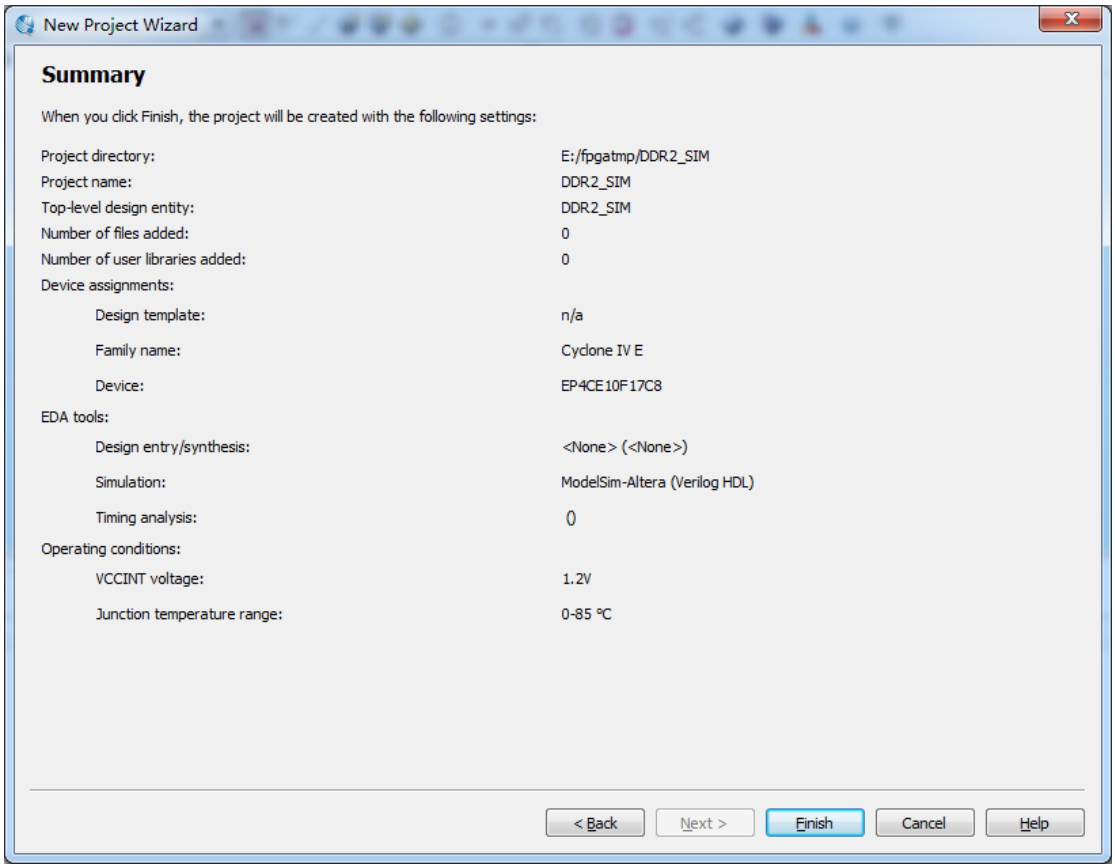
仿真还是在我们最熟悉的 Altera 开发套件 QuartusII 13.0 中进行。首先创建工程，我将工程命名为 DDR2_SIM，器件选择任意一个器件即可，比如我们最熟悉的 EP4CE10F17C8，也可以 AC6102 上使用的 EP4CE30F23C7。选择仿真工具选择 modelsim - altera，语言为 Verilog。最后，整个工程建立完毕后的 Summary 如下图所示：

注意：

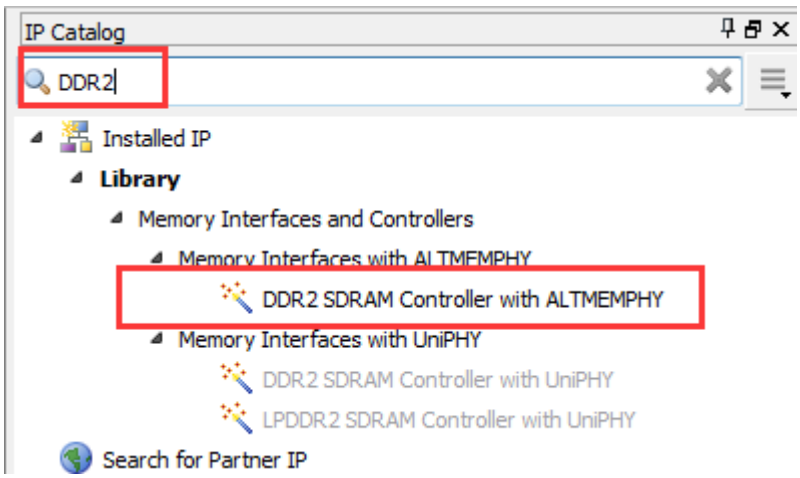
1、打开 Quartus II 软件时，请右键选择以管理员方式运行，切记，否则可能导致 IP 生成不成功。

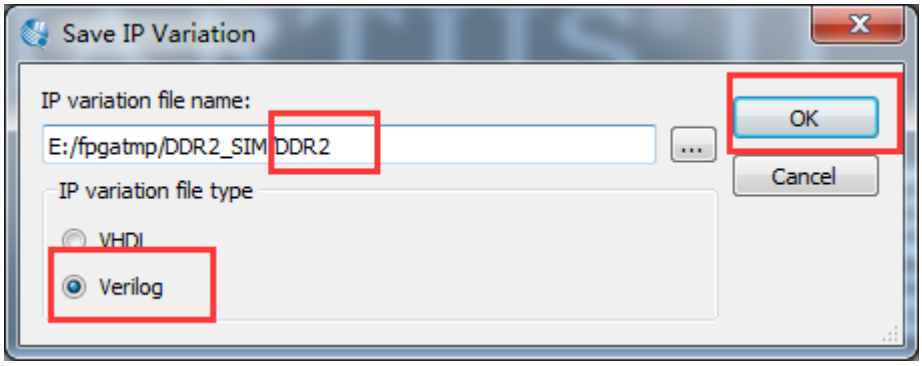
2、创建工程时不要将工程创建在和 Quartus II 安装目录相同的盘符下，否则可能导致生产 IP 失败。

3、如果在生产 IP 界面长时间卡住，无法生成 IP，请关闭掉 Windows 系统的自动更新和防火墙后重新尝试。



工程创建好后，打开 IP 创建向导，在搜索栏处输入 DDR2，然后在搜索结果中选择 DDR2 SDRAM Controller with ALTMEMPHY，如下图所示：

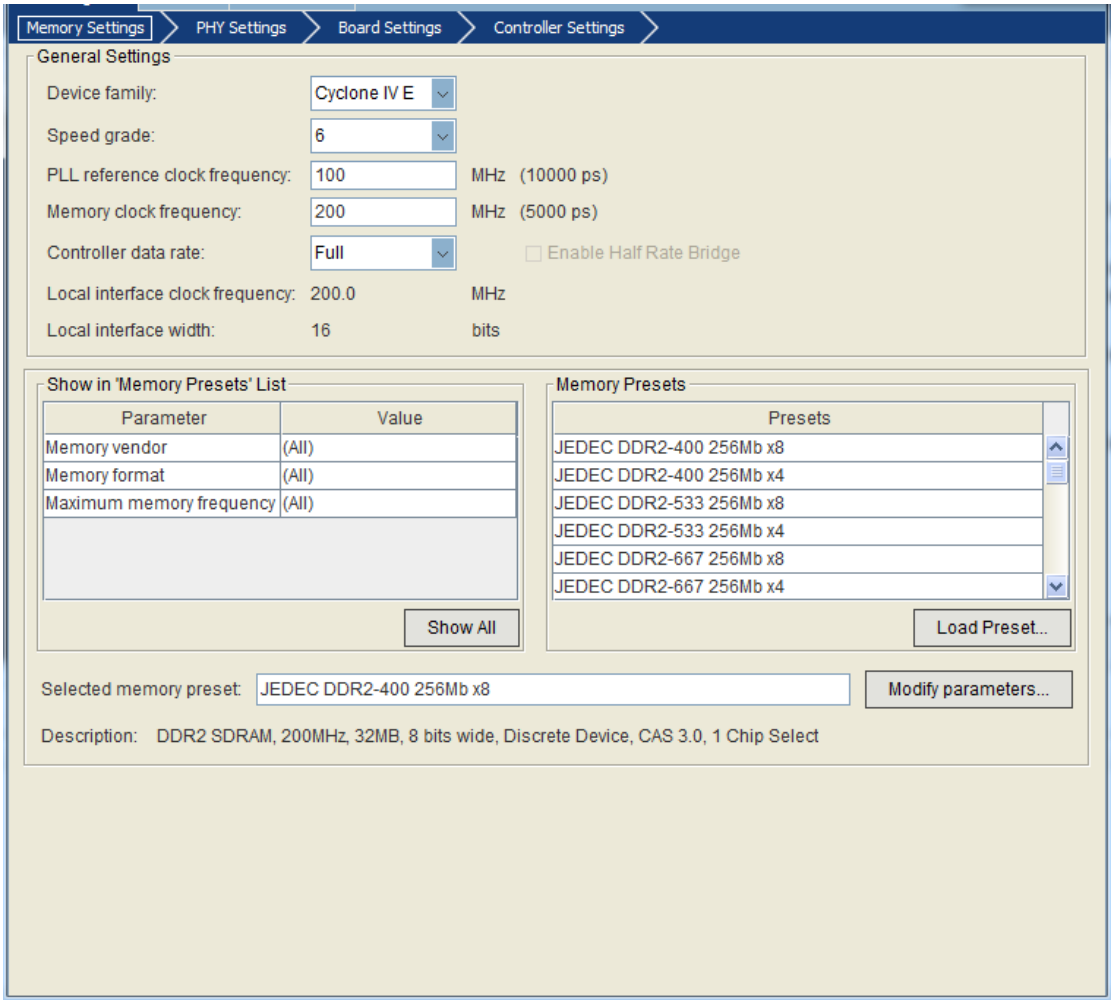




将该 IP 命名为 DDR2，语言选择 Verilog，然后点击 OK，就会开始加载参数设置对话框，整个加载过程大约需要等待 20 到 30 秒左右才会弹出 GUI 界面，请大家耐心等待。

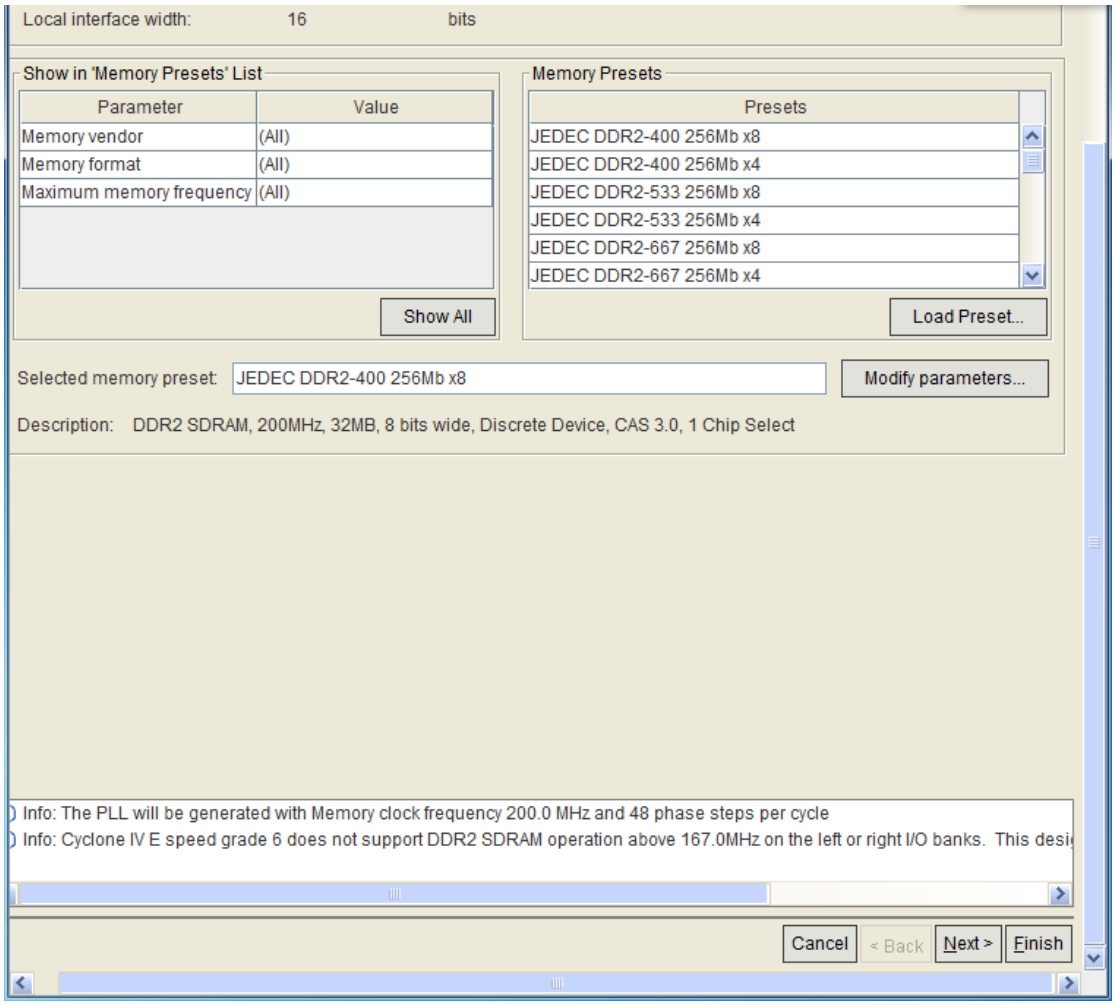
注意：IP 请直接创建在 Quartus 的工程根目录下，不要再创建子文件夹，否则仿真的时候会报错。提示找不到文件，实际是因为文件太多，IP 中有些文件路径包含时候处理不好导致的，属于 IP 核设计的小 bug。

弹出的 GUI 界面如下所示：



很遗憾，对于分辨率较低的电脑，如 1366*768。整个界面会显示不全，上半部分无法看到，也无法拖动窗口，原本右下侧的 finish 和 cancel 按钮也没有显露出来，这个问题我从使用 Quartus II 11.0 的时候就发现了（没有用过 10.x 版本，据说是从 10.x 版本开始出现这个问题的），到了 Quartus II 15.1 中这个问题依然没有得到解决，在 1080p 的屏幕上则能够正常显示。

虽然界面默认没有完全显示，但是我们还是有办法来让他显示的。如果你不需要移动这个配置窗口的位置，或者说对界面最上方未显示的部分内容已知或者不关心（实际我们也真的不用去关心）那么简单的解决方法就是单击系统右下角的显示桌面按钮（win7），然后再在任务栏中点击该配置界面，就能够成功加载右下角的 finish 和 cancel 按钮了，但是界面上半部分依旧无法看到。如下图所示：



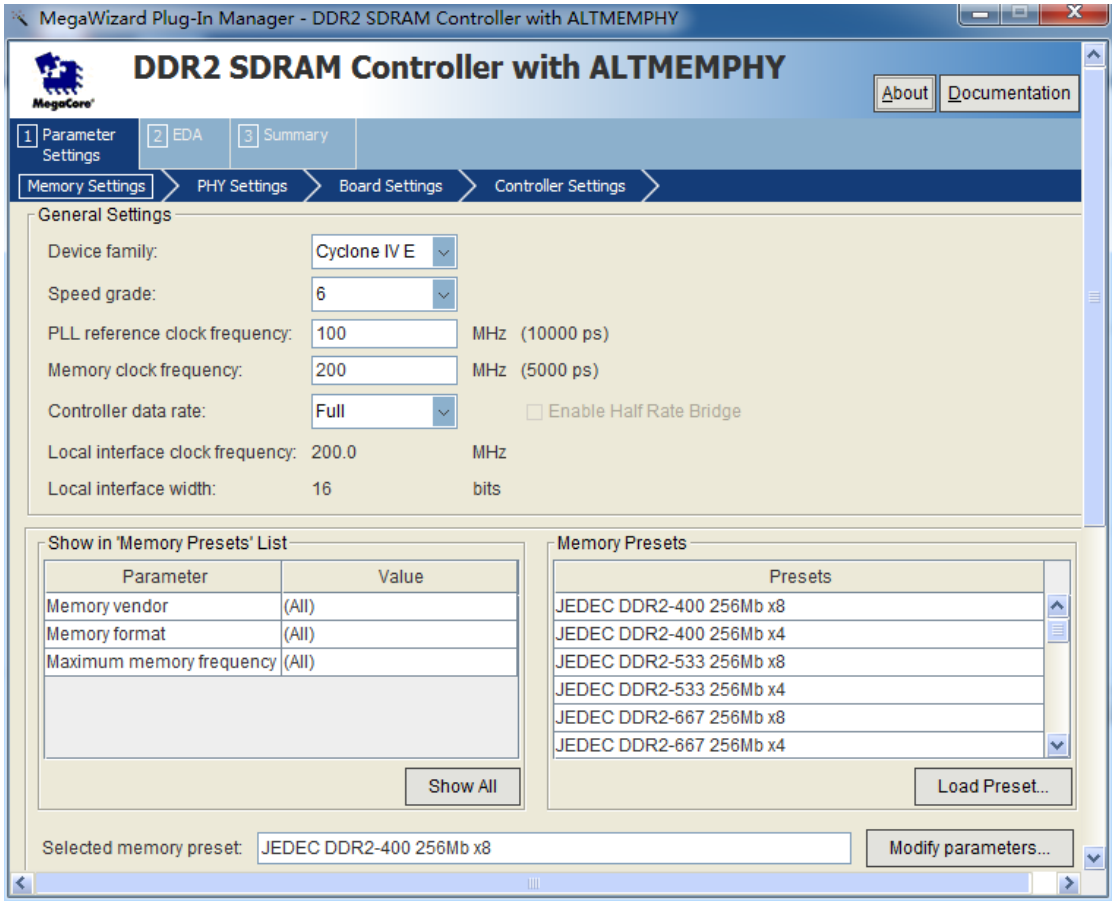
但是此时依旧无法看到上半部分，终极解决方案就是，

1、打开这个配置界面的 GUI

2、点击系统右下角的显示桌面按钮（win7），回到桌面。

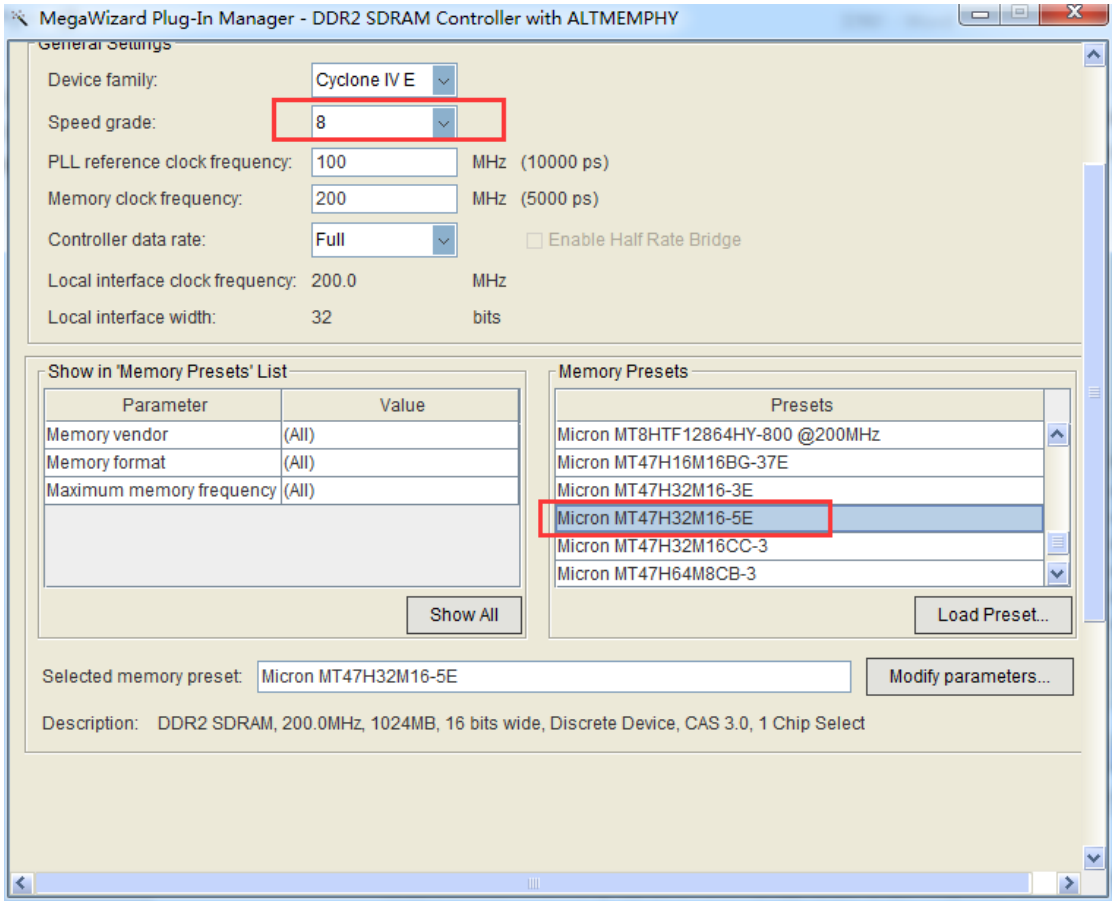
3、设置电脑屏幕分辨率为最小值（我是这么做的，不清楚设置其他分辨率是否也能奏效）

第三步：将屏幕分辨率修改回正常值。这时候再看，整个界面就能够正常显示了，如下图：

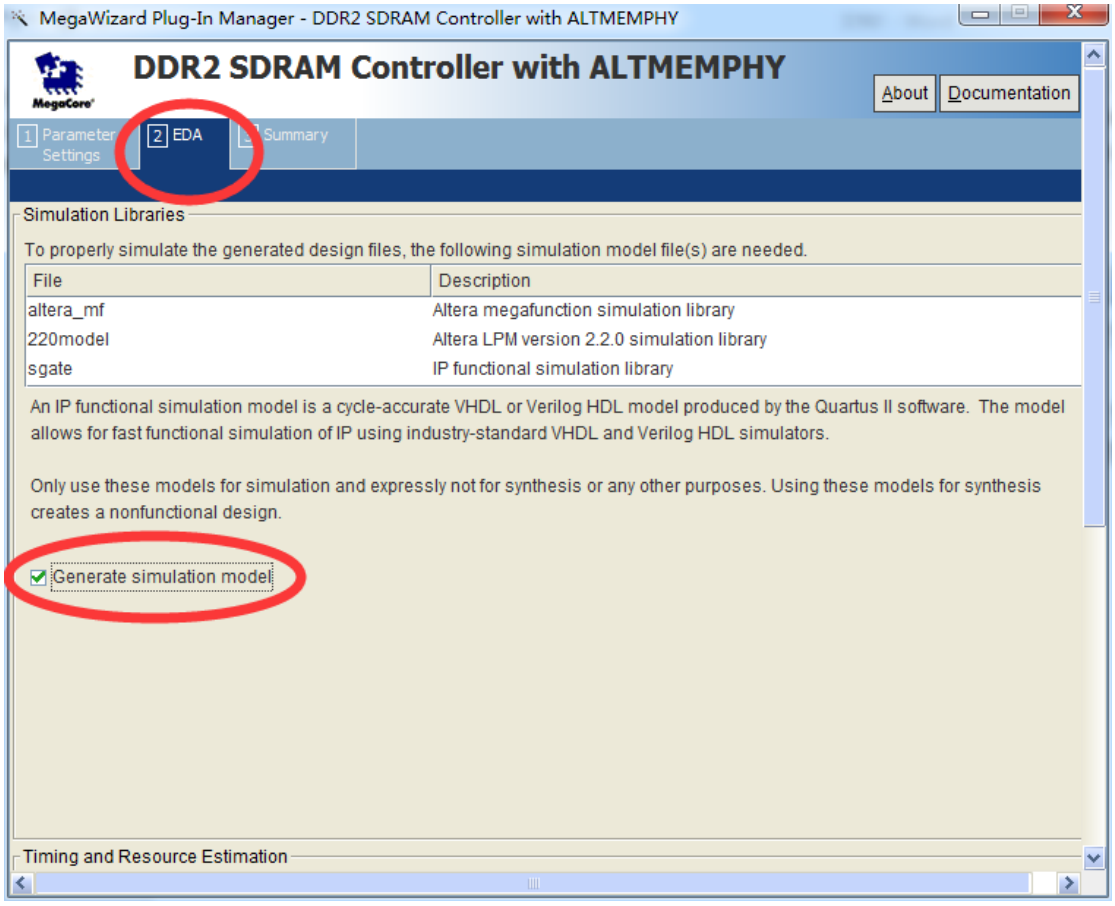


拖着右侧的进度条往下划就能看到 finish 和 cancel 按钮了。同时页面最上方的内容也能看到了。

这里我们在 Memory 选项卡中，设置速度等级（Speed Grade）为 8，与实际芯片保持一致。Memory Presets 为“Micron MT47H32M16-5E “，如下图所示：



此页中其他选项保持默认，接下来的若干项均保持默认即可，直到切换到 EDA 选项卡处，勾选 Generate Simulation model，这里是为了仿真时生成仿真模型，以配合 modelsim 进行仿真。



点击右下角的 finish，软件则开始生成 IP 核控制器的相关文件和示例内容。生成完毕大约 2 分钟。

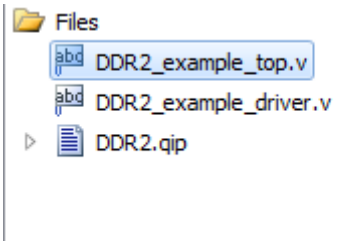
生成完毕后，设置 DDR2.qip 为设计顶层模块，然后执行分析和综合（快捷键是 Ctrl + K）。

分析和综合完成后，整个 IP 核占用资源如下图所示（图为我之前使用 Quartus II 15.0 截得）：

Flow Summary	
Flow Status	Successful - Sat Jun 20 11:40:55 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	DDR2_SIM
Top-level Entity Name	DDR2
Family	Cyclone IV E
Device	EP4CE10F17C8
Timing Models	Final
Total logic elements	5,635
Total combinational functions	4,612
Dedicated logic registers	2,844
Total registers	2942
Total pins	161
Total virtual pins	0
Total memory bits	11,232
Embedded Multiplier 9-bit elements	0
Total PLLs	1

话说这个控制器还是挺耗费资源的啊。

接下来设置仿真，DDR2 IP 默认就提供了一个 DDR2_example_top 文件来作为例子的顶层，该例子可综合可仿真，可以作为我们使用 DDR2 控制器的设计参考。查看 DDR2_example_top 文件，其中例化了 DDR2 控制器和一个 DDR2_example_driver 文件。于是我回到 Quartus II 中，手动添加 DDR2_example_top.V 和 DDR2_example_driver.V 文件到工程中来，并更改 DDR2_example_top.V 为设计顶层文件，然后分析和综合。

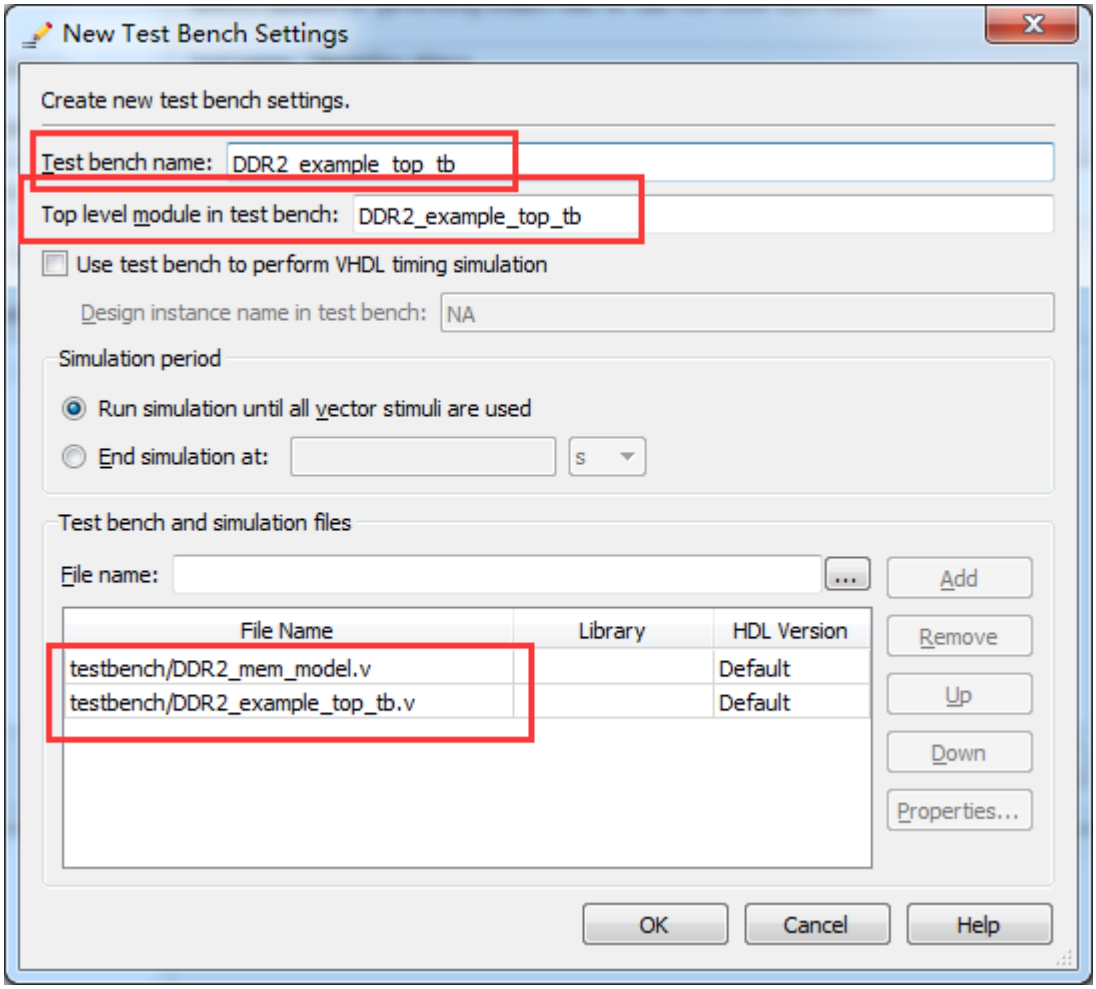


接下来设置 NativeLink 以将设计工程和 Modelsim-altera 关联起来。

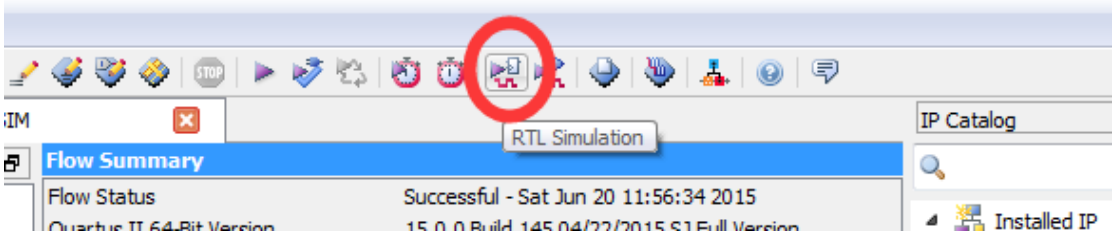
设置 testbench 为这里具体怎么添加文件我就不多说了，不会的请参看我们的视频教程或相关文档教程。

需要添加的 testbench 文件有两个，位于 testbench 文件夹下，分别为 DDR2_example_top_tb.v 和 DDR2_mem_model.V。其中 DDR2_mem_model.V 是一个 DDR2 的仿真模型，该模型直接用行为语言描述了一个虚拟的 DDR2 器件，这样，通过 DDR2 控制器来操作这个虚拟的器件，就能够保证控制器得到正常的操作相应，从而使仿真正常的进行下去。

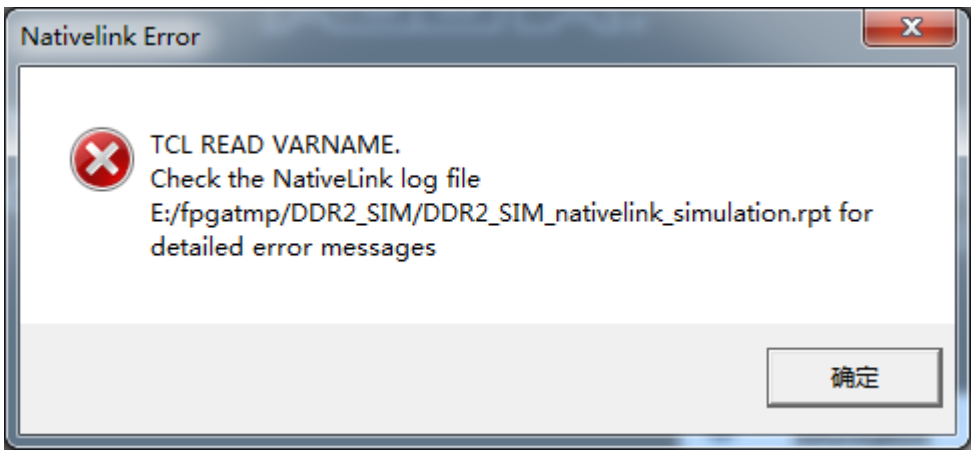
添加完成后，设置 Test bench name 和 top level module in test bench 为 DDR2_example_top_tb。然后一路点击 OK 下去，直到设置完毕。 如下图所示：



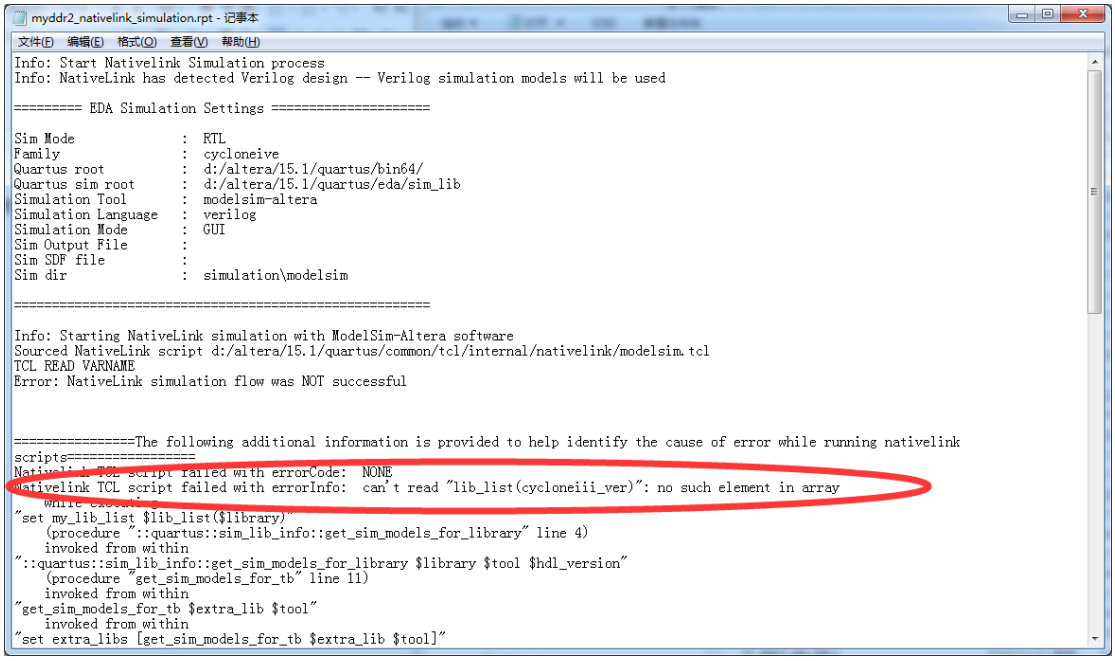
接下来，就可以直接点击 RTL Simulation 按钮执行仿真了：



注意，如果用户使用高于 Quartus II 13.0 版本的软件（如 Quartus II 15.1）生成 IP 并仿真，在这一步会报如下错误。



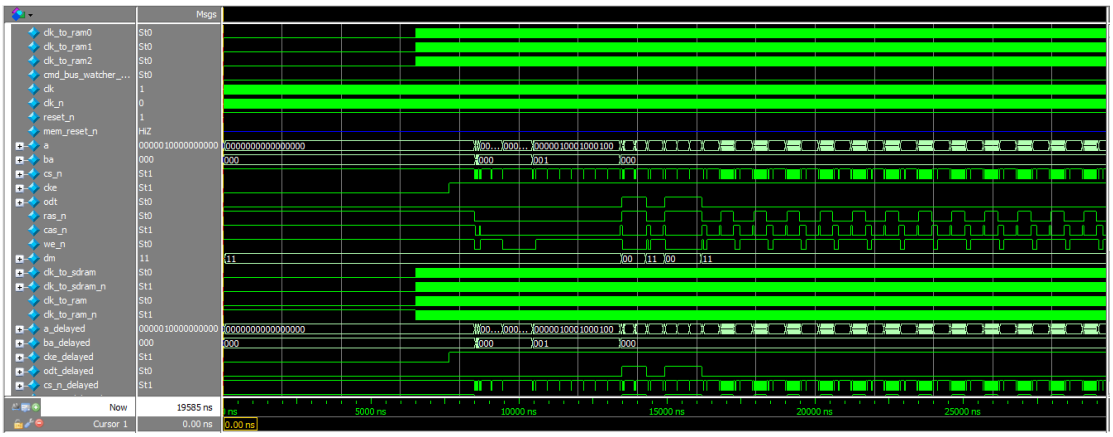
根据信息提示，查看对应的文件内容，DDR2_SIM_nativelink_simulation.rpt，发现如下信息：

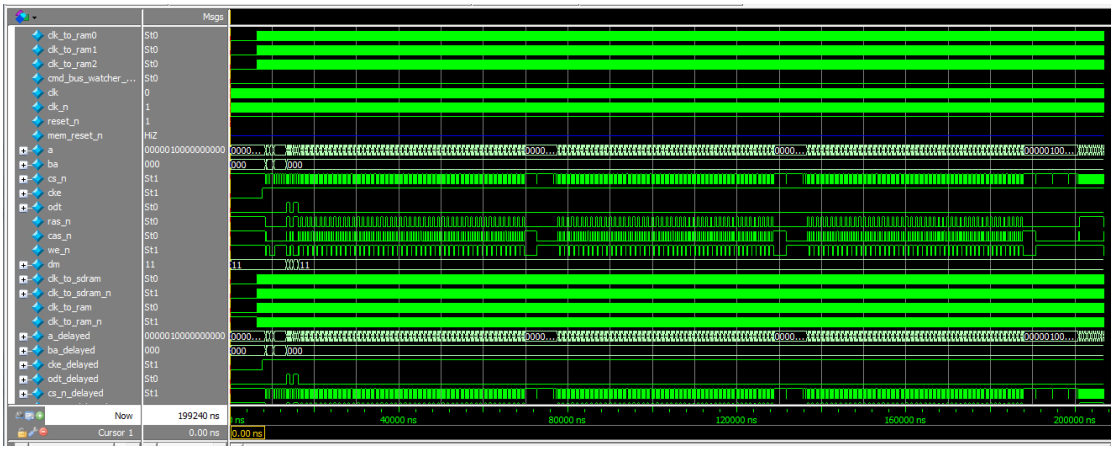


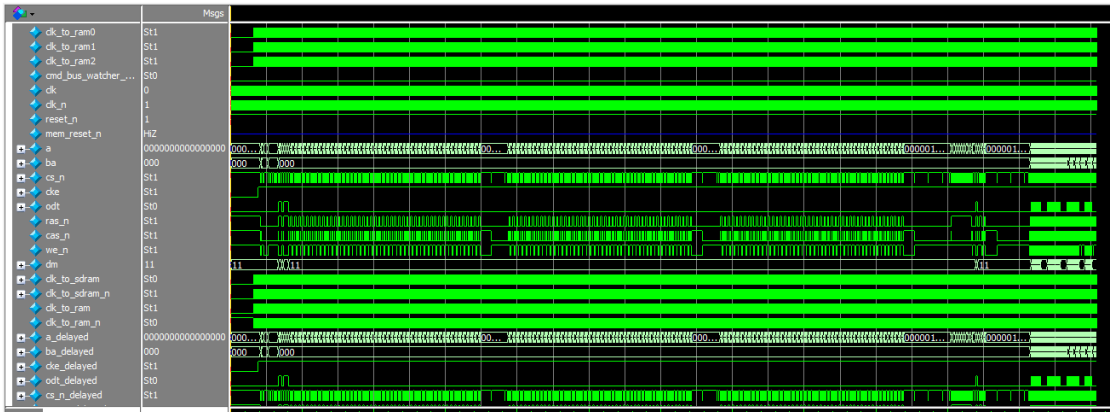
也就是说，仿真需要调用一个叫 cycloneiii_ver 的库，结果找不到。这个库。为啥找不到呢？可能这个 IP 核的工程模型是基于 cycloneiii 器件设计的，仿真的时候默认还是调用 cyclone iii 的器件模型，但是从 Quartus II 13.0 以后，Quartus II 软件已经不再支持 cyclone iii。软件里面也不再提供 cyclone iii 的器件模型，因此仿真时候无法找到这个模型。当然无法开始仿真了。具体可以参照 Altera 官方网站的解释

https://www.altera.com/support/support-resources/knowledge-base/solutions/fb210109.html

使用 Quartus II 13.0 仿真运行完成后，截得仿真波形如下图所示：








```
#               91570000          PCH all banks
#               91575000          ARF
#               91580000          ARF
```

```
#              112615000          ARF
#              112620000          ACT     row address 0000 bank 0
#              112625000          ACT     row address 0000 bank 0
#              112630000          ACT     row address 0000 bank 0
#              112635000          ACT     row address 0000 bank 0
#              112640000          ACT     row address 0000 bank 0
#              112645000          ACT     row address 0000 bank 0
#              112650000          ACT     row address 0000 bank 0
#              112655000          ACT     row address 0000 bank 0
#              112660000          ACT     row address 0000 bank 0
#              112665000          ACT     row address 0000 bank 0
#              112670000          ACT     row address 0000 bank 0
#              112675000          RD from col address 0028 bank 0
#              112680000          RD from col address 002c bank 0
#              112685000          RD from col address 002c bank 0
#              112690000          RD from col address 0018 bank 0
#              112695000          RD from col address 0018 bank 0
```

```
#               240985000          RD from col address 07f4 bank 7
#               241025000          ACT     row address 7f7f bank 7
#               241040000          RD from col address 07ec bank 7
#               241080000          ACT     row address 9eff bank 7
#               241095000          RD from col address 07dc bank 7
#               241135000          ACT     row address bdff bank 7
#               241150000          RD from col address 07bc bank 7
#               241190000          ACT     row address dbff bank 7
#               241205000          RD from col address 077c bank 7
#               241245000          ACT     row address f7ff bank 7
#               241260000          RD from col address 06fc bank 7
#               241300000          ACT     row address 0fff bank 7
#               241315000          RD from col address 05fc bank 7
#               241355000          ACT     row address 3fff bank 7
#               241370000          RD from col address 03fc bank 7
```

通过本流程，相信大家已经能够完成 DDR2 控制器的仿真了，本节内容就到这里，由于最近事务繁多，后续内容择机发布。

小梅哥
2016 年 11 月 17 日于成都市电子科技大学

附录：

Quartus II 仿真 DDR2 失败的原因
https://www.altera.com/support/support-resources/knowledge-base/solutions/fb210109.html

Last Modified: June 11, 2015
Version Found: v14.0
Simulation of ALTMEMPHY IP with Cyclone IV Devices Fails

Description
This problem affects all ALTMEMPHY-based external memory interfaces targeting Cyclone IV devices.

Simulation of an ALTMEMPHY-based external memory interface fails with the following error message:

can't read "lib_list(cycloneiii_ver)": no such element in array

Cyclone IV simulations require Cyclone III libraries, because many atoms and modules from the earlier family are reused. The problem is that Cyclone III has reached end-of-life, and its libraries have been removed from the Quartus II

software version 14.0. Consequently, simulations of ALTMEMPHY-based IP on Cyclone IV devices in the Quatus II software version 14.0 or later, will fail.

Workaround/Fix

The workaround for this issue is to use a version of the Quartus II software earlier than 14.0.

This issue will not be fixed.