第一节 DDR3 基础知识

1、DDR3 全称第三代双倍速率同步动态随机存储器。

特点：①掉电无法保存数据，需要周期性的刷新。②时钟上升沿和下降沿都会传输数据。③突发传输，突发长度 Burst Length 一般为 8。

2、DDR3 的存储：bank、行地址和列地址

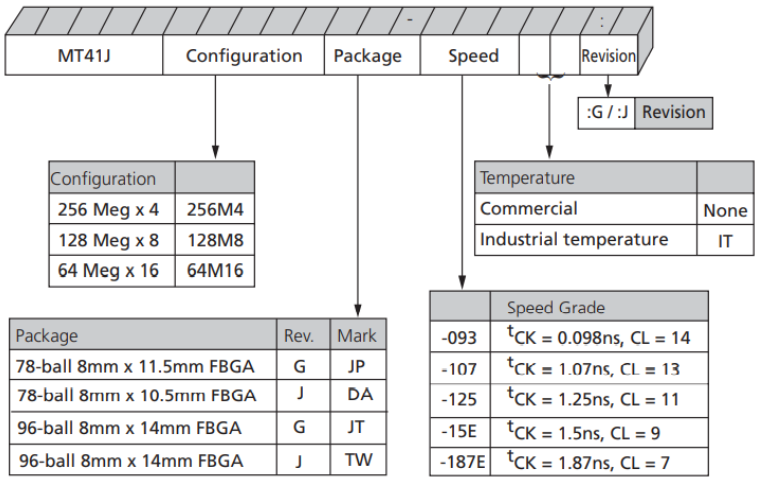
数据怎么存入到 DDR3：先指定一个 Bank 地址，再指定行地址，最后指定列地址。

DDR3 容量计算：bank 数量 × 行数量 ×列数量 × 存储单元容量。

比如 bank address 位宽为 3，Row address 位宽为 14，Column address 为 10，则容量为：2^3 × 2^14 × 2^12 × 16bit

3、DDR3 命名

以镁光公司的 DDR3 为例子：**MT41J 64M16 -125**



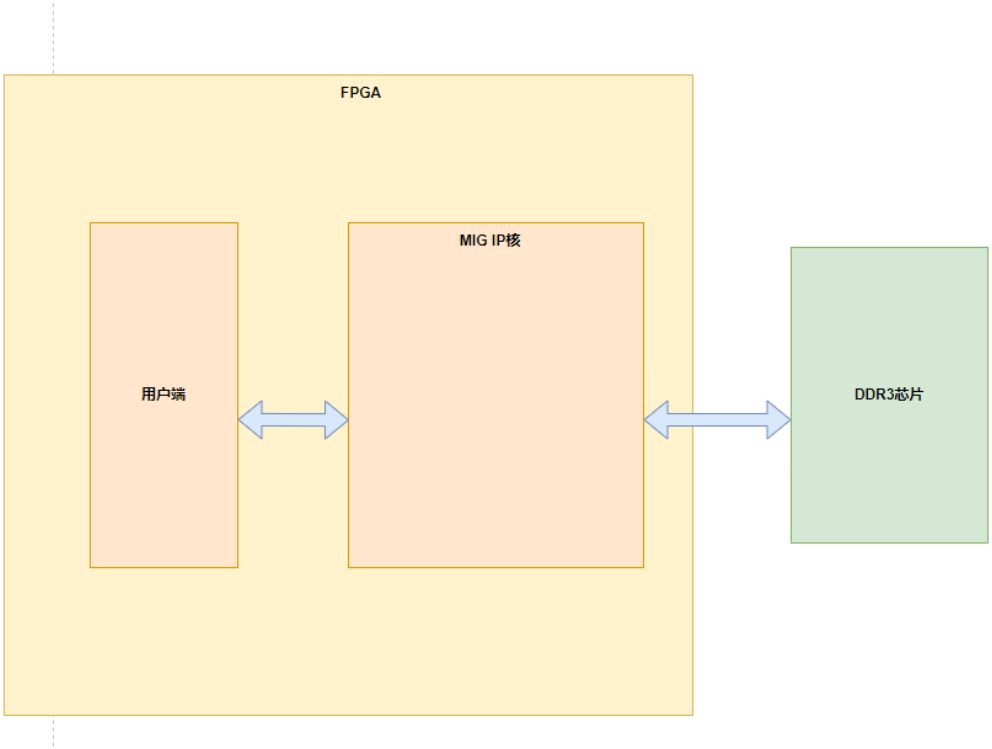
从 DDR3 的命名，我们可以得出几点有用的信息：

① DDR3 传输数据端口的位宽

② DDR3 支持的**最大的**时钟频率

③ DDR3 传输数据的带宽

第二节 时钟



1、时钟类别（不要弄混淆哦）

系统时钟：MIG IP 核工作时钟，一般命名为 sys_clk。

参考时钟：MIG IP 的参考时钟，必须为 200M，命名为 ref_clk

DDR3 芯片工作的时钟：由 FPGA 输入到 DDR3 芯片，为差分时钟

用户端时钟：MIG IP 核输出给用户端的时钟，命名为 ui_clk

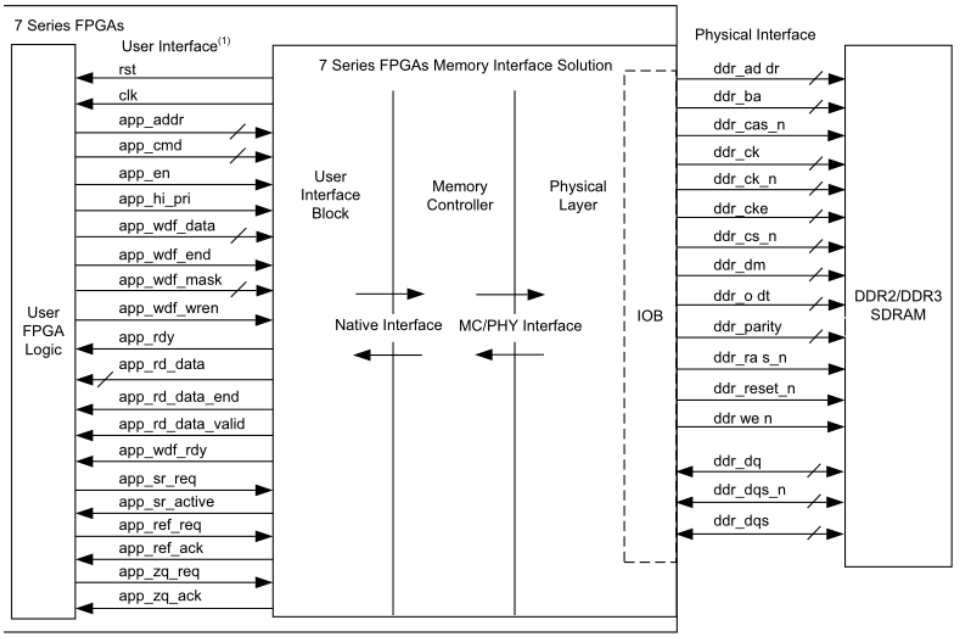
2、DDR3 芯片工作的时钟与用户端时钟有一个比例关系：

DDR3 芯片工作的频率：用户端时钟频率为 4:1 或者 2:1，当 DDR3 芯片工作的时钟为 800M 时候，比例只能为 2:1

第三节 MIG IP 核的配置

1、用户端接口有两种：Native 接口和 AXI4 接口

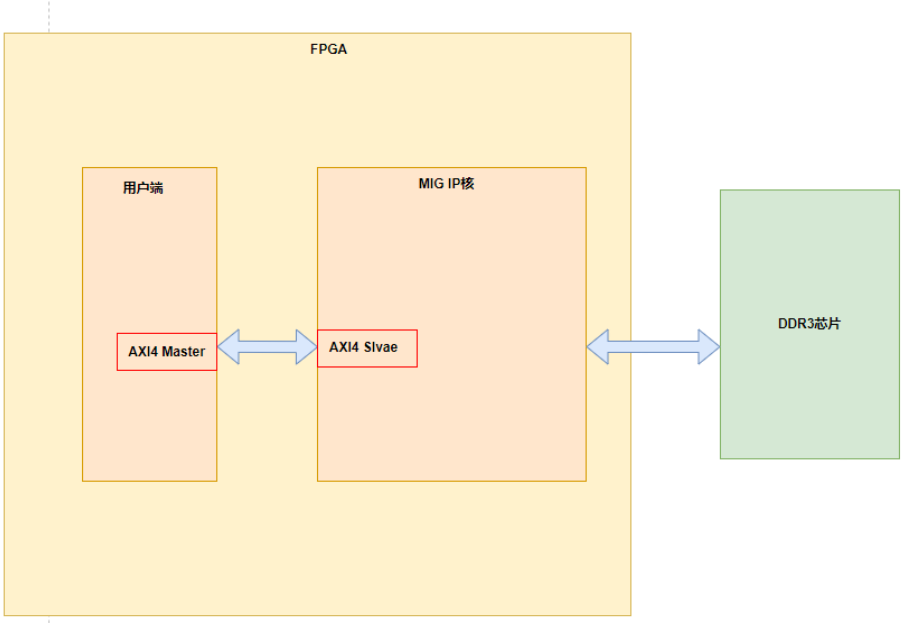
①Native 接口



1. System clock (sys_clk_p and sys_clk_n/sys_clk_i), Reference clock (clk_ref_p and clk_ref_n/clk_ref_i), and system reset (sys_rst_n) port connections are not shown in block diagram.

Figure 1-51: **7 Series FPGAs Memory Interface Solution**

②AXI4 接口



2、带宽计算(重点)

①FPGA 写入数据到 DDR3 芯片的带宽为：

800M × 2 × 16bit

②用户端写入数据到 MIG IP 核的带宽为：

200M ×用户端数据位宽

因为 800M × 2 × 16bit = 200M ×用户端数据位宽

所以用户端数据位宽为 128bit

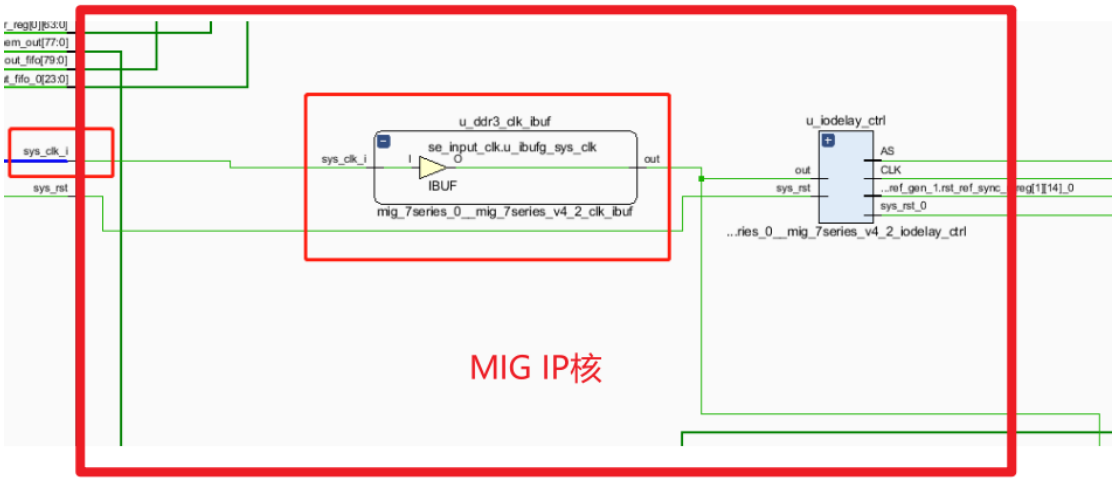
3、系统时钟 system clock 关于 No Buffer、Single-Ended、Differential 怎么选择？

系统时钟由内部时钟产生，比如经过 PLL 后产生的 200M 时钟，选择 No buffer。

系统时钟由 FPGA 外部晶振产生，输入到 FPGA 的管脚，再输入到 MIG IP 核，选择 Single-Ended 或者 Differential。外部晶振产生的时钟为单端时钟，选择 Single-Ended；外部晶振产生的时钟为差分时钟，选择 Differential；

No Buffer 是什么意思？

No Buffer 就是 MIG IP 核内部没有例化 IBUF 原语



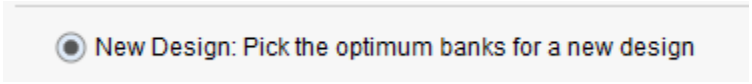
3、参考时钟 Reference Clock 怎么选择？

No Buffer、Single-Ended、Differential 选择的判据同上。

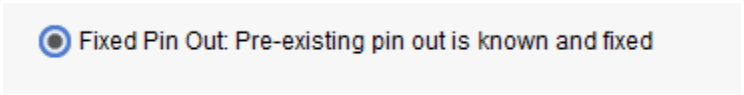
如果系统时钟为 200M，则选择“Use System Clock”。

4、绑定管脚如何选择？

①只需要仿真 DDR3，学习 DDR3，不上板子调试，选择第一个“New Design”



②需要上板调试，选择“Fixed Pin Out”



第四节 官方源码 example design 解读（Native 接口）

1、MIG IP 核端口信号

① 与 DDR3 物理芯片连接的信号

```
mig_7series_0 u_mig_7series_0 (
   // Memory interface ports
   .ddr3_addr                      (ddr3_addr),  // output [13:0]      ddr3_addr
   .ddr3_ba                        (ddr3_ba),  // output [2:0]     ddr3_ba
   .ddr3_cas_n                     (ddr3_cas_n),  // output            ddr3_cas_n
   .ddr3_ck_n                      (ddr3_ck_n),  // output [0:0]       ddr3_ck_n
   .ddr3_ck_p                      (ddr3_ck_p),  // output [0:0]       ddr3_ck_p
   .ddr3_cke                       (ddr3_cke),  // output [0:0]        ddr3_cke
   .ddr3_ras_n                     (ddr3_ras_n),  // output            ddr3_ras_n
   .ddr3_reset_n                   (ddr3_reset_n),  // output          ddr3_reset_n
   .ddr3_we_n                      (ddr3_we_n),  // output          ddr3_we_n
   .ddr3_dq                        (ddr3_dq),  // inout [15:0]     ddr3_dq
   .ddr3_dqs_n                     (ddr3_dqs_n),  // inout [1:0]       ddr3_dqs_n
   .ddr3_dqs_p                     (ddr3_dqs_p),  // inout [1:0]       ddr3_dqs_p
   .ddr3_cs_n                      (ddr3_cs_n),  // output [0:0]       ddr3_cs_n
   .ddr3_dm                        (ddr3_dm),  // output [1:0]     ddr3_dm
   .ddr3_odt                       (ddr3_odt),  // output [0:0]        ddr3_odt
```

② 用户端读写信号（重要）

```
  // Application interface ports
  .app_addr                    (app_addr),  // input [27:0]    app_addr
  .app_cmd                     (app_cmd),  // input [2:0]      app_cmd
  .app_en                      (app_en),  // input             app_en
  .app_rdy                     (app_rdy),  // output           app_rdy

  .app_wdf_data                (app_wdf_data),  // input [127:0]       app_wdf_data
  .app_wdf_end                 (app_wdf_end),  // input               app_wdf_end
  .app_wdf_wren                (app_wdf_wren),  // input              app_wdf_wren
  .app_wdf_rdy                 (app_wdf_rdy),  // output          app_wdf_rdy
  .app_wdf_mask                (app_wdf_mask),  // input [15:0]       app_wdf_mask

  .app_rd_data                 (app_rd_data),  // output [127:0]       app_rd_data
  .app_rd_data_end             (app_rd_data_end),  // output           app_rd_data_end
  .app_rd_data_valid           (app_rd_data_valid),  // output         app_rd_data_valid
```

③ 用户端时钟与复位、系统时钟与复位

```
//
  .ui_clk                      (ui_clk),  // output            ui_clk
  .ui_clk_sync_rst             (ui_clk_sync_rst),  // output   ui_clk_sync_rst

  // System Clock Ports
  .sys_clk_i                    (sys_clk_i),
  .sys_rst                     (sys_rst) // input sys_rst
```

④ 其他不那么重要的信号

```
  .app_sr_req                  (app_sr_req),  // input         app_sr_req
  .app_ref_req                 (app_ref_req),  // input           app_ref_req
  .app_zq_req                  (app_zq_req),  // input         app_zq_req
  .app_sr_active               (app_sr_active),  // output         app_sr_active
  .app_ref_ack                 (app_ref_ack),  // output           app_ref_ack
  .app_zq_ack                  (app_zq_ack),  // output            app_zq_ack

  .device_temp_i               (device_temp_i),  // input [11:0]           device_temp_i
```

思考一个问题，MIG IP 核用户端的 app_addr 地址信号如何与 DDR3 芯片的地址对应起来？

- **Row Width** – 15
- **Bank Width** – 3
- **Column Width** – 10

Example (1) – When the selected option in the MIG GUI is BANK_ROW_COLUMN and the address to the controller is mapped accordingly.

Original Mapping of the Address Bits																										
BANK Address Bits			**ROW Address Bits**														**COLUMN Address Bits**									
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B2	B1	B0	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
Remapped Address with TG_TEST																											

2、用户端时序

如何学习用户端与 MIG IP 核之间的数据交互时序？

① 阅读 MIG IP 核的 User Guide

② 看官方 design example 源码以及仿真波形

第五节 搭建仿真平台

1、为什么要搭建仿真平台？