《片外存储器应用》

[《片外存储器应用》 1](#_Toc184899004)

[第一节：片外存储器 3](#_Toc184899005)

[1.1片外存储器特点 3](#_Toc184899006)

[1.2片外存储器分类 4](#_Toc184899007)

[1.3集成电路存储器 5](#_Toc184899008)

[1.3.4sdram(同步动态ram) 5](#_Toc184899009)

[1.3.4.1sdram特点 6](#_Toc184899010)

[1.3.4.2sdram工作原理 8](#_Toc184899011)

[第二节：sdr sdram 9](#_Toc184899012)

[2.1sdr sdram接口 9](#_Toc184899013)

[2.2sdr sdram命令 10](#_Toc184899014)

[2.3sdr sdram控制器设计 11](#_Toc184899015)

[2.3.1初始化操作电路设计 11](#_Toc184899016)

[2.3.2sdr sdram写操作 13](#_Toc184899017)

[2.3.2.1写操作程序设计 14](#_Toc184899018)

[2.3.3读操作 15](#_Toc184899019)

[2.3.3.1读操作程序设计 17](#_Toc184899020)

[2.3.4.刷新操作 18](#_Toc184899021)

[2.3.4.1刷新操作程序设计 20](#_Toc184899022)

[2.3.5控制程序架构设计 21](#_Toc184899023)

[2.3.5.1sdr控制电路程序设计 22](#_Toc184899024)

[2.3.6仿真验证 24](#_Toc184899025)

[2.3.7在线验证 25](#_Toc184899026)

[第三节：DDRx 26](#_Toc184899027)

[3.1DDRx接口 28](#_Toc184899028)

[3.2ddrx在FPGA中的应用 29](#_Toc184899029)

在数字电路设计中，我们的设计围绕着数据进行的，数据是电路设计开发的核心，数据离不开存储，能够很好的掌握并应用存储对于我们来说属于基本的技能。

数字电路存储从应用的角度有ON-CHIP-MEMERY和片外存储2种，二值最大的区别就是片上存储器存储数据量小，占用设计资源，适合用来作为数据临时存储，片外存储器不占用设计资源，存储数据量较大，适合用来做存储。

# 第一节：片外存储器

片外存储器就是存储器在我们设计电路之外，因此需要专用的存储器接口进行数据读取，比如DDrx存储器有自己通用的专用接口进行数据读取。

由于存储器的读写对数据准确率要求较高，因此不适合异步读写，所以现在的存储器皆采用同步读写，比如SDRAM.

## 1.1片外存储器特点

* 容量大

存储容量可达到G级以上，市面常见的**单芯片存储（存储器颗粒）容量大小有512M，1G，2G，常见的数据位宽有4比特，8比特，16比特**，如果实际需要更大容量，一般是将多个但存储芯片进行串联或者并联。

数据容量扩展有2种方式：

数据位扩展（并联）

如：有2个存储芯片，数据位宽位16位

mem1

mem0

cs\_n

data [31:16]

Data [15:00]

地址位扩展（串联）

如存储器地址32位,数据位宽8位

mem0

mem1

Data[7:0]

addr[31:0]

cs\_n0

cs\_n1

* 独立接口

片外存储器都有自己独立的接口，读写速率越高相对应的操作接口越复杂，尤其是采用同步读写，如果速率在千兆以上，如果没有采用高速接口，对信号质量要求非常高，实际中存储器接口一般采用并行接口传输。

* 带宽高时延大

并行工作在千兆，理论上按照比特计算带宽很高，如600m\*8 = 4800m,由于读写时延大（读写数据需要多个时钟周期），实际带宽看芯片的具体表现。

* 造价低

片外存储器基础记忆元件有可能采用造价较低的半导体或者其他存储介质。

## 1.2片外存储器分类

存储介质

基础记忆元件为非半导体器件

如flash，eeprom，DRAM，sram

特点：体积小，不需要专用的读写设备，直接通过接口读写数据

集成电路存储器

基础记忆元件为半导体器件

如：硬盘，光盘，软盘

特点：体积大，需要专用的读写设备

memery

flash

dram

rom

eeprom

Sram

Sdr sdram

sdram

ram

ddrx sdram

adram

## 1.3集成电路存储器

以半导体为核心，三极管或者mos管，三端器，电容为基本记忆元件构成的存储器，它有如下特点：

* 体积小，易集成
* 相对于存储介质而言存储效率高，速率更快
* 可直接通过接口读写，不需要读写设备（间接读写）。
* 造价低

在实际中集成电路存储器rom大部分情况使用flash,而ram大部分情况下使用sdram,对于数字电路设计者而言，主要书序flash和sdram。

### 1.3.4sdram(同步动态ram)

Sdram是集成电路存储器中非常重要的存储器，它的造价相对于sram(静态ram),二者主要去呗就是内部使用的记忆元件不一样

Sram：mos管作为基本元件

Sdram:内部采用电容作为基本记忆元件

利用**电容的状态**来表示信号的值

0：完全放电状态，

1：完全充电状态

由于电容即使在不掉电的情况下，电荷任然会流失，真是电容本身的特点，所以我们需要**间断的对电容进行充放电，我们这个操作叫做刷新操作**，以维持当前电容状态。

由于电容的本身特性，我们需要不断对电容进行充放电，即刷新位置当前电容状态，所以叫做动态ram,实际中皆采用同步读写，所以叫做sdram.

#### 1.3.4.1sdram特点

* 存储空间表现

存储器常见的空间表现有线性空间，面空间，三维空间，最常见的就是线性空间，sdram采用三维空间

线性空间 ：只需要一个线性地址，如:addr[K-1:0]

|  |
| --- |
| A0 |
| A1 |
| A2 |
| AK-2 |
| AK-1 |

面空间 ： 需要2个地址， 如:col[K0-1:0],row[K1-1:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A00 | A01 | A02 | A… | Ak0-1 k1-1 |
| … | … | ... | ... | ... |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |

三维空间 ：需要3个地址， 如: **col[K0-1:0],row[K1-1:0] ba[K3-1:0]**

L-bank

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A00 | A01 | A02 | A… | Ak0-1 k1-1 |
| … | … | ... | ... | ... |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A00 | A01 | A02 | A… | Ak0-1 k1-1 |
| … | … | ... | ... | ... |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A00 | A01 | A02 | A… | Ak0-1 k1-1 |
| … | … | ... | ... | ... |
| … | … | … | … | … |
| … | … | … | … | … |
| … | … | … | … | … |

* 潜伏期

在读操作操作的时候，当我们读的那一时刻，数据不会立即出来，而是延迟多个时钟周期采出来，我们把这个延迟叫做读潜伏期，sdram都有读潜伏期(又叫做cas延迟)，一般没有写潜伏期，但是ddr3另外。

写命令

读命令

CL=2

CWL=k

数据

CL=k

数据

* 突发

Sdram都支持突发操作，所谓突发就是连续读出或者写入过个数据的操作，一般用BL表示

如BL =4,CL=2

写命令

读命令

D3

D2

D1

D0

D3

D2

D1

D0

* 中断

当sdram正在读写数据的时候，突然发起来其它操作要求，此时读写停止

其它

D3

D2

D1

D0

读命令

CL=2

其它

写命令

D3

D2

D1

D0

* 命令

Sram的所有操作都可以通过命令完成

* 刷新

每间隔一定的时间要对sdram进行刷新操作，电容保持当前状态的极限时间是**64ms**.

* 掩码

Sdram支持掩码，sdr sdram支持读写掩码，ddrx支持写掩码。

2’b01

16’h55a5

2’b01

16’h55a5

#### 1.3.4.2sdram工作原理

数据位

列地址位

行地址位

刷新放大器

# 第二节：sdr sdram

Sdr sdram是sdram的最初的形态，有自己通用的接口，所有操作都可以通过命令完成。

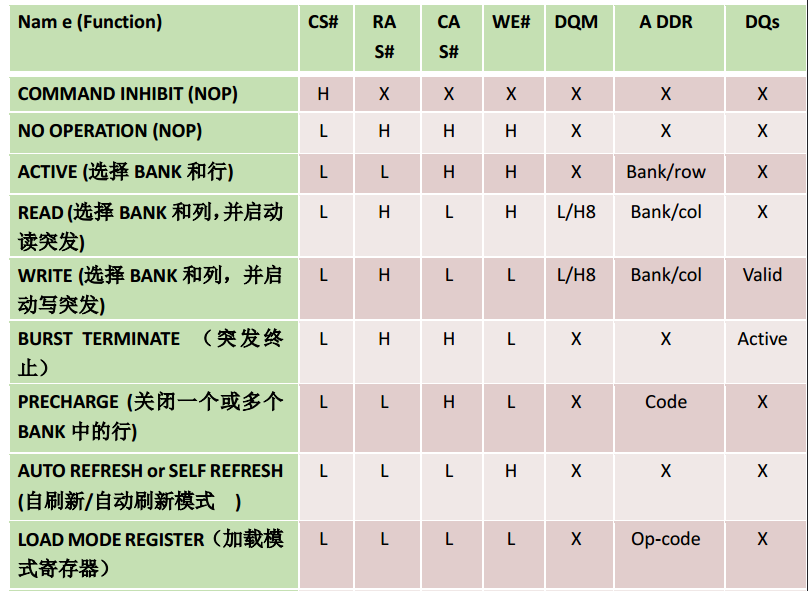
## 2.1sdr sdram接口

Sdr sdram接口通用，任何厂家的芯片都支持该接口。

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ck | Out(控制器)，in(存储器) | 接口同步时钟，上升沿有效 |
| Cke | Out(控制器)，in(存储器) | CK屏蔽信号  1：未屏蔽  0：屏蔽 |
| cs\_n | Out(控制器)，in(存储器) | 总线片选，低电平有效 |
| we | Out(控制器)，in(存储器) | 写使能  1：写  0：对 |
| Ba[B-1:0] | Out(控制器)，in(存储器) | Bank地址 |
| addr[A-1:0] | Out(控制器)，in(存储器) | 行地址，列地址复用 |
| cas\_n | Out(控制器)，in(存储器) | 当cas\_n=0表示addr为列地址 |
| ras\_n | Out(控制器)，in(存储器) | 当ras\_n==0表示addr为行地址 |
| Dq[D-1:0] | Inout | 读写数据信号 |
| Dqm[D/8-1:0] |  | 读写掩码位，按字节掩码 |

## 2.2sdr sdram命令

所有sdr sdram都可以通过命令完成，并且命令都是一样的，具体如下：



|  |  |  |
| --- | --- | --- |
| 命令 | 解析时间 | 描述 |
| CMD\_INIT | 0 | 禁止sdr sdram一切操作 |
| CMD\_NOP | 0 | 没人动作，空操作 |
| CMD\_ACTIVE | **`TRCD** | 打开行所对应bank下行地址所表示的开关电路 |
| CMD\_PR | `TRP | 关闭行所对应bank下行地址所表示的开关电路  ***支持自动预充电，A10==1*** |
| CMD\_RF | `TRFC | 刷新，位置当前电容状态  刷新退出  ***支持自刷新（将时钟禁止一段时间）***  T存储器自身决定  >=120us  刷新进入  cke |
| CMD\_LMR | `TMRD | 初始化时配置工作模式，模式通过行行列复用地址配置 |
| CMD\_WRITE | `BL+`TWR | 写数据，支持写保护 |
| CMD\_READ | `BL+`CL | 读数据，支持潜伏期 |
| CMD\_TM | 0 | 中断当前读写操作 |

## 2.3sdr sdram控制器设计

SDR sdram作为存储器，我们主要是读写数据，它的常规操作是读和写操作，除此之外还有刷新和初始化操作，让其正常工作，我们需要完成4种操作，每一种操作都有自己的独立操作流程。

在设计控制的时候，我们每一种操作设计为一个独立的电路模块，下来我们对这4中操作逐个设计电路，我们以MT48LC32M16A2芯片为例。

### 2.3.1初始化操作电路设计

rst\_n

clk

addr\_init[12:0]

Cmd\_init[4:0]

sdr\_init

done\_init

req\_init

Cke =Cmd\_init[4]

Cs\_n= Cmd\_init[3]

ras\_n= Cmd\_init[2]

cas\_n= Cmd\_init[1]

we\_n= Cmd\_init[0]

If(cnt==`TRFC/20ns)cnt<=0;

Else cnt<=cnt+1;  
IF(cnt==0)cmd\_init<=` CMD\_RF;else

cmd\_nit<=`CMD\_NOP;

addr\_init<=addr\_init;

If(cnt==`TRFC/20ns)cnt<=0;

Else cnt<=cnt+1;  
IF(cnt==0)cmd\_init<=` CMD\_RF;else

cmd\_nit<=`CMD\_NOP;

addr\_init<=addr\_init;

If(cnt==`TRP/20ns)cnt<=0;

Else cnt<=cnt+1;  
IF(cnt==0)cmd\_init<=` CMD\_PR;else

cmd\_nit<=`CMD\_NOP;

addr\_init<=13’h1FFF;

If(cnt==100us/20ns)cnt<=0;

Else cnt<=cnt+1;

rst\_n

req\_init

If(cnt==`TRFC/20ns)cnt<=0;

Else cnt<=cnt+1;  
IF(cnt==0)cmd\_init<=` CMD\_LMR;else

cmd\_nit<=`CMD\_NOP;

addr\_init<=addr\_init;

{

3’b000,

1’b0,//支持突发

2’b00,

3’b.11//CL=3

1’b0,//顺序突发

3b010//BL=4

};

Cnt==`TMRD/20ns

Cnt==`TRFC/20ns

LMR\_RF0

PR\_RF1

Cnt==`TRFC/20ns

PR\_RF0

Cnt==`TRP/20ns

PR\_SNT

CHECK\_MYSELF

Cnt==100us/20ns-1

IDLE

端口说明

|  |  |
| --- | --- |
| 信号名 | 描述 |
| 系统相关 | |
| clk | 系统时钟，也是CK的时钟，默认50mhz  目前sdr sdram的ck是频率瓶颈为200mhz |
| rst\_n | 系统复位 |
| 用户接口 | |
| req\_init | 当req\_init==1表示初始化请求 |
| done\_init | 当done\_init==1表示当前初始化完成 |
| Sdr sdram初始化相关的sdr sdram接口 | |
| Cmd\_init[4:0] | Cke =Cmd\_init[4]  Cs\_n= Cmd\_init[3]  ras\_n= Cmd\_init[2]  cas\_n= Cmd\_init[1]  we\_n= Cmd\_init[0] |
| addr\_init[12:0] | Sdr sdram接口行列复用地址信号 |

### 2.3.2sdr sdram写操作

Sdr sdram工作有4种独立的操作，写操作就是突发（已设置支持突发）的写入数据，没有写潜伏期。

DBL-1

D…

D1

D0

写命令

Write rRecovery

`BL+TWR

Sdr sdram写操作按照标准的流程写，我们对写操作流程进行分析

给出写数据bank地址和行地址

给出写数据bank地址和行地址

给出写数据bank地址和行地址

给出bank地址

**多行操作流程**

行内操作流程

习惯性的操作流程

非sdr sdram真实操作流程

关闭当前行

写数据，给出bank地址和列地址

关闭当前行

写数据，给出bank地址和列地址

如果往当前bank下的行内写数据

写数据，给出bank地址和列地址

写数据，给出bank地址和列地址

写数据同时给出列地址

给出行地址

给出bank地址

写数据同时给出列地址

给出行地址

通过以上的分析，sdr sdram的操作流程分情况，多行操作流程，行内操作流程，但是**行内操作也可以按照多行操作流程操作**，在实际应用中，一般我们都是按照多行操作流程设计，这样提高了涉及到 通用性。

#### 2.3.2.1写操作程序设计

每种操作是独立的，这里我们把写操作设计为一个独立的模块，设计思路和初始化一样。

Cke =Cmd\_w[4]

Cs\_n= Cmd\_w[3]

ras\_n= Cmd\_w[2]

cas\_n= Cmd\_w[1]

we\_n= Cmd\_w[0]

clk

rst\_n

ba\_w[1:0]

req\_w

dqm\_w[15:0]

dq\_w[15:0]

addr\_w[12:0]

cmd\_w[4:0]

req\_w

sdr\_write

done\_w

wrdata[ 4\*16-1:0]

addr\_w[ 24:0]

cnt==`BL+`TWR+`TRP

WRITE\_COL\_OFF\_ROW

cnt==`TRCD

ON\_ROW

rst\_n

IDLE

If(cnt ==`TWR+`BL+`TRP)cnt<=0;

Else cnt<=cnt+1;

If(cnt==0)begin

Cmd\_w<= ` CMD\_WRITE;

Addr<={2’bxx,1’b1,laddr[09:00]};

Ba<= laddr[24:23];end

If(cnt ==`TRCD)cnt<=0;

Else cnt<=cnt+1;

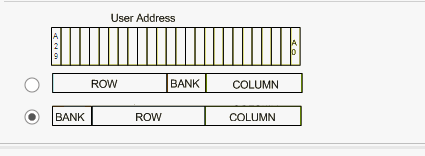
If(cnt==0)begin

Cmd\_w<= `CMD\_ACTIVE;

Addr<=laddr[22:10];

Ba<= laddr[24:23];end

注意：用户操作一般按照线性地址操作存储器，线性地址和实际地址之间映射如下：



端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 描述 | |
| 系统相关 | | |
| clk | | 系统时钟，也是CK的时钟，默认50mhz  目前sdr sdram的ck是频率瓶颈为200mhz |
| rst\_n | | 系统复位 |
| 用户接口 | | |
| req\_w | | 当req\_w==1表示写请求 |
| done\_w | | 当done\_w==1表示当前写完成 |
| addr\_w[ 24:0] | | 写操作线性地址，实际读写一般按照线性地址操作，映射有2种 |
| wrdata[ 4\*16-1:0] | | 写突发数据 |
| Sdr sdram写相关的sdr sdram接口 | | |
| Cmd\_w[4:0] | Cke =Cmd\_w[4]  Cs\_n= Cmd\_w[3]  ras\_n= Cmd\_w[2]  cas\_n= Cmd\_w[1]  we\_n= Cmd\_w[0] | |
| addr\_w[12:0] | Sdr sdram接口行列复用地址信号 | |
| dq\_w[15:0] | Sdr sdram数据信号 | |
| dqm\_w[15:0] | Sdr sdram写掩码 | |
| ba\_w[1:0] | Bank地址 | |

### 2.3.3读操作

Sdr sdram工作有4种独立的操作，写操作就是突发（已设置支持突发）的写入数据，有写潜伏期。

Cl

DBL-1

D…

D1

读命令

D0

`BL+`CL

Sdr sdram写操作按照标准的流程写，我们对写操作流程进行分析

行内操作流程

关闭当前行

读数据，给出bank地址和列地址

给出读数据bank地址和行地址

关闭当前行

读数据，给出bank地址和列地址

给出读数据bank地址和行地址

如果往从当前bank下的行内读数据

读数据，给出bank地址和列地址

读数据，给出bank地址和列地址

给出读数据bank地址和行地址

读数据同时给出列地址

给出行地址

给出bank地址

读数据同时给出列地址

给出行地址

给出bank地址

习惯性的操作流程

非sdr sdram真实操作流程

**多行操作流程**

通过以上的分析，sdr sdram的操作流程分情况，多行操作流程，行内操作流程，但是**行内操作也可以按照多行操作流程操作**，在实际应用中，一般我们都是按照多行操作流程设计，这样提高了设计的 通用性。

#### 2.3.3.1读操作程序设计

sdr\_read

clk

rst\_n

req\_r

done\_r

addr\_r[ 24:0]

cmd\_r[4:0]

addr\_r[12:0]

dq\_r[15:0]

dqm\_r[15:0]

Cke =Cmd\_r[4]

Cs\_n= Cmd\_r[3]

ras\_n= Cmd\_r[2]

cas\_n= Cmd\_r[1]

we\_n= Cmd\_r[0]

ba\_r[1:0]

rddata[ 4\*16-1:0]

cnt==`BL+`CL+`TRP

READ\_COL\_OFF\_ROW

cnt==`TRCD

ON\_ROW

req\_w

rst\_n

IDLE

If(cnt ==`CL+`BL+`TRP)cnt<=0;

Else cnt<=cnt+1;

If(cnt==0)begin

Cmd\_w<= ` CMD\_READ;

Addr<={2’bxx,1’b1,laddr[09:00]};

Ba<= laddr[24:23];end

If(cnt ==`TRCD)cnt<=0;

Else cnt<=cnt+1;

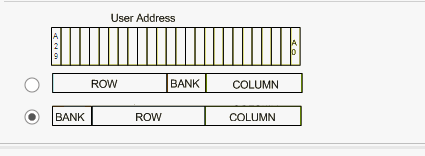
If(cnt==0)begin

Cmd\_w<= `CMD\_ACTIVE;

Addr<=laddr[22:10];

Ba<= laddr[24:23];end

注意：用户操作一般按照线性地址操作存储器，线性地址和实际地址之间映射如下：



端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 描述 | |
| 系统相关 | | |
| clk | | 系统时钟，也是CK的时钟，默认50mhz  目前sdr sdram的ck是频率瓶颈为200mhz |
| rst\_n | | 系统复位 |
| 用户接口 | | |
| req\_r | | 当req\_r==1表示读请求 |
| done\_r | | 当done\_r==1表示当前读完成 |
| addr\_r[ 24:0] | | 写操作线性地址，实际读写一般按照线性地址操作，映射有2种 |
| rddata[ 4\*16-1:0] | | 读突发数据 |
| Sdr sdram写相关的sdr sdram接口 | | |
| Cmd\_r[4:0] | Cke =Cmd\_r[4]  Cs\_n= Cmd\_r[3]  ras\_n= Cmd\_r[2]  cas\_n= Cmd\_r[1]  we\_n= Cmd\_r[0] | |
| addr\_r[12:0] | Sdr sdram接口行列复用地址信号 | |
| dq\_r[15:0] | Sdr sdram数据信号 | |
| dqm\_r[15:0] | Sdr sdram读掩码 | |
| ba\_r[1:0] | Bank地址 | |

### 2.3.4.刷新操作

由于电容本身特性，即使在通电的情况下，状态也不是一直保持不变，它是动态变化的，由于sdram依靠电容状态记忆逻辑值，因此必须保持当前状态不能较大大变化，内部需要一个专用电路不断维持当前电容状态，这个电路叫做刷新放大器，任何sdram内部都有该电路，这个电路不能自己动执行，需要我们控制。

任何sdram刷新电路都是按按行刷新，每一次刷新操作对所有bank的同一行进行刷新，当前行由内部的刷新计数器决定，计数器从0计数到该存储器所有行的最大值。

行地址位

刷新放大器

Cnt(刷新计数器)

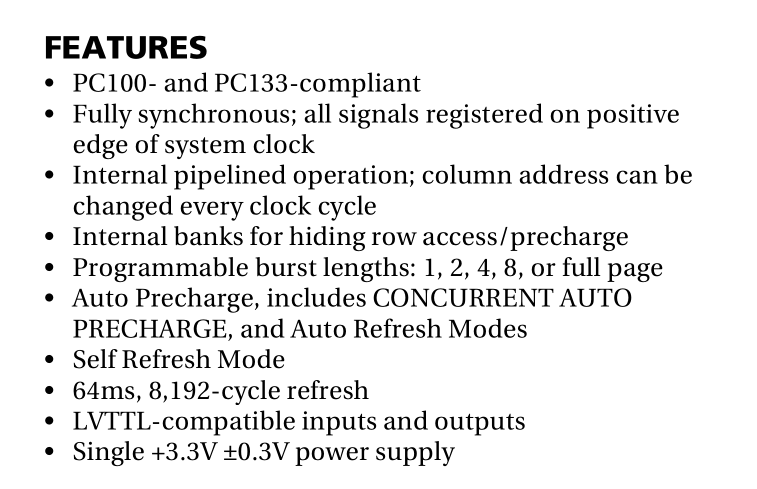
刷新控制

电容状态发生改变的极限事件MAX64ms

刷新工作原理示意图

刷新控制，需要间断性的刷新，刷新时间间隔需要满足电容的极性时间要求

刷新时间间隔的计算按照以下公式计算：Tref <= 64ms/2N(N为行地址位宽)



Tref <= 64ms/8192<= 7.812us

#### 2.3.4.1刷新操作程序设计

Cke =Cmd\_rf[4]

Cs\_n= Cmd\_rf[3]

ras\_n= Cmd\_rf[2]

cas\_n= Cmd\_rf[1]

we\_n= Cmd\_rf[0]

rst\_n

clk

done\_rf

cmd\_rf[4:0]

req\_rf

sdr\_ref

端口说明

|  |  |
| --- | --- |
| 信号名 | 描述 |
| 系统相关 | |
| clk | 系统时钟，也是CK的时钟，默认50mhz  目前sdr sdram的ck是频率瓶颈为200mhz |
| rst\_n | 系统复位，低电平有效 |
| 用户接口 | |
| req\_rf | 当req\_rf==1的时候表示刷新请求 |
| done\_rf | 当done\_rf==1表示当前刷新完成 |
| sdr sdram刷新操作相关接口信号 | |
| cmd\_rf[4:0] | Cke =Cmd\_rf[4]  Cs\_n= Cmd\_rf[3]  ras\_n= Cmd\_rf[2]  cas\_n= Cmd\_rf[1]  we\_n= Cmd\_rf[0] |

If(cnt ==`TRFC)cnt<=0;

Else cnt<=cnt+1;

If(cnt==0)cmd\_rf<=`CMD\_RF;

rst\_n

req\_rf

Cnt==`TRFC

RF\_SANT

IDLE

### 2.3.5控制程序架构设计

基于以上4种操作，4个独立的模块设计一个基于MT48LC32M16A2存储器芯片设计一个控制器的架构。

clk

rst\_n

clk

rst\_n

sdr\_init

sdr\_write

sdr\_ctl

Ck

we

done

req

done\_rf

req\_rf

laddr\_r[24:0]

done\_r

req\_r

laddr\_w[24:0]

done\_w

req\_w

done\_init

req\_init

cmd\_rf[4:0]

dqm\_r[1:0]

ba\_r[1:0]

addr\_r[12:0]

cmd\_r[4:0]

dqm\_w[1:0]

ba\_w[1:0]

addr\_w[12:0]

cmd\_w[4:0]

cmd\_init[4:0]

addr\_init[12:0]

rddata[63:0]

Wrdata[63:0]

cas\_n

ras\_n

cs\_n

Cke

sdr\_sdtam

Laddr[24:0]

data\_r[63:0]

data\_w[63:0]

dq [15:0]

dqm[1:0]

ba[1:0]

addr[12:0]

we\_n

dq\_w[15:0]

dq\_w[15:0]

sdr\_rf

sdr\_read

bus\_sel[1:0]

0：初始化操作

1：刷新操作

2：写操作

3：读操作

#### 2.3.5.1sdr控制电路程序设计

Sdr sdram4种独立操作，同一时刻只能有一种操作，并且这四种操作协同工作才能使存储器正常读写数据。

bus\_sel[1:0]

sdr\_ctl

req\_init

done\_init

req\_w

done\_w

laddr\_w[24:0]

data\_w[63:0]

req\_r

done\_r

laddr\_r[24:0]

data\_r[63:0]

req\_rf

done\_rf

rst\_n

clk

req

done

we

Wrdata[63:0]

rddata[63:0]

laddr[24:0]

If(done\_rf)

Req\_rf<=0;

Else

Req\_rf<=1;

done\_w

done\_rf

tm\_ok

RF\_S

rst\_n

If(done\_r)

Req\_r<=0;

Else

Req\_r<=1;

Laddr\_r<=laddr;

If(done\_w)

Req\_w<=0;

Else

Req\_w<=1;

Data\_w<=wrdata;

Laddr\_w<=laddr;

done\_r

~tm\_ok & req & ~we

~tm\_ok & req &we

done\_init

If(done\_init)

Req\_ini<=0;

Else

Req\_init<=1;

READ\_S

WRITE\_S

IDLE

INIT\_S

端口说明

|  |  |
| --- | --- |
| 信号名 | 描述 |
| 系统相关 | |
| clk | 系统时钟 |
| rst\_n | 系统复位 |
| Sdr sdram读写相关信号 | |
| req | Sdr sdram存储器请求 |
| done | 当前Sdr sdram存储器读写完成 |
| we | 写使能 |
| laddr[24:0] | 存储器线性地址 |
| Wrdata[63:0] | 存储器写数据 |
| rddata[63:0] | 存储器读数据 |
| 初始化相关信号 | |
| req\_init | 初始化请求 |
| done\_init | 当前初始化完成 |
| **写相关信号** | |
| req\_w | 写请求 |
| done\_w | 当前写完成 |
| laddr\_w[24:0] | 写地址 |
| data\_w[63:0] | 写数据 |
| 读相关信号 | |
| req\_r | 读请求 |
| done\_r | 当前读完成 |
| laddr\_r[24:0] | 读地址 |
| data\_r[63:0] | 读数据 |
| 刷新相关信号 | |
| req\_rf | 刷新请求 |
| done\_rf | 当前刷新完成 |
| 接口选择 | |
| Bus\_sel[1:0] | bus\_sel[1:0]  0：初始化操作  1：刷新操作  2：写操作  3：读操作 |

### 2.3.6仿真验证

rddata[63:0]

wrdata[63:0]

laddr[24:0]

we

done

req

dqm[1:0]

dq[15:0]

addr[12:0]

ba[1:0]

mt48lc32m16a2

Sdr\_sdram

we\_n

cas\_n

ras\_n

cs\_n

cke

ck

Laddr<={%damom,2’b00}

done

done

WRITE\_DATA

rst\_n

If(done)

Req<=0;

Else

Req<=1;

We<=0;

If(done)

Req<=0;

Else

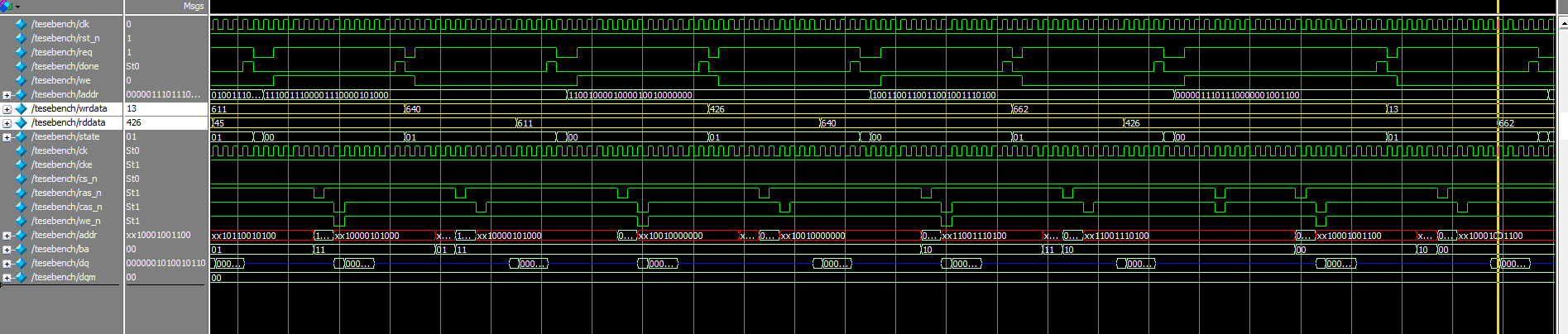
Req<=1;

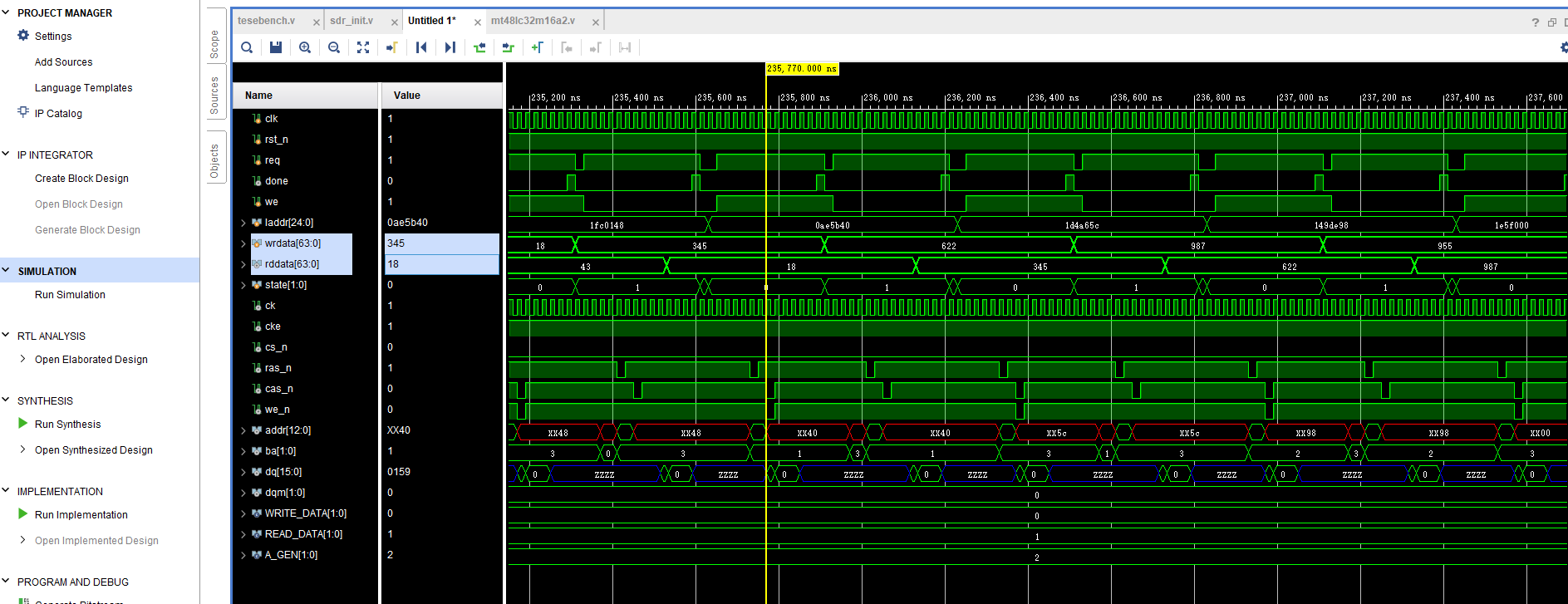
Wrdata<={$random}%1023;

We<=1;

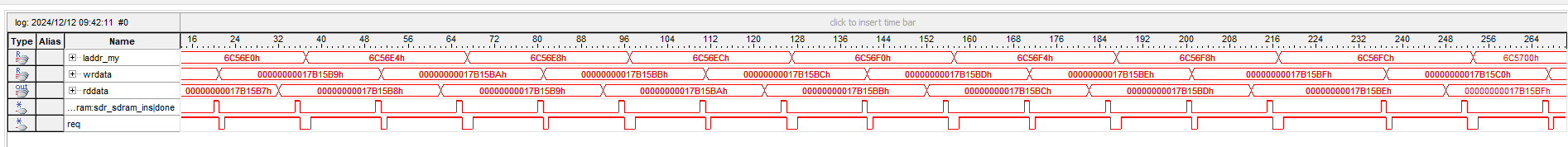
A\_GEN

READ\_DATA





### 2.3.7在线验证



# 第三节：DDRx

作为集成电路存储器，它采用同步操作，接口需要时钟，随着技术的发展，我们希望操作同步时钟频率越高越好，即ck的频率越高越好，实际中该时钟频率会受到存储器本身的限制，以目前的计数，尚未突破200mhz，并且从显示的角度去看很突破该限制，我们即希望能改变操作频率，又在ck限制的情况下达到目的，为了达到这种目的，DDX就在这样的背景下诞生。

Data0[15:00]

Data1[31:16]

sdram

sdram

Ck1=200mhz

Ck1=400mhz

sdram

sdram

Ck1=400mhz

Data0[15:00]

Data1[31:16]

Ck1=200mhz

2bit预取说明示意图

2bit预存说明示意图

data[15:0]

预存取缓存

预存取缓存

Data3[15:00]

Data3[31:16]

Data3[47:32]

Data3[63:48]

sdram

sdram

sdram

sdram

Ck1=200mhz

Ck1=800mhz

data[15:0]

sdram

sdram

sdram

Ck1=200mhz

Ck1=800mhz

sdram

预存取缓存

预存取缓存

4bit预取说明示意图

data[15:0]

Data3[15:00]

Data3[31:16]

Data3[47:32]

Data3[63:48]

4bit预存说明示意图

通过以上的分析可以看出，传统的sdram由于本身ck的限制，如果想要更高速率的ck操作接口，解决的方式通过预存取技术实现，引入了该技术之后就提高了操作频率，它是ddrx的核心技术。

传统同步方式单时钟沿采样，如果我们2个沿同时采样数据，我们将预存取缓存分为奇数组和偶数组，上升数据放入计数组寄存器中，下降沿采样数据放入偶数组寄存器中，这样实现了双沿采样，操作时钟频率也同时降为原来的一半，由于双沿采样我们称之为ddr(双倍速率同步动态随机[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8/1583185?fromModule=lemma_inlink)) sdram.

根据ddx的工作原理衍生出3个主要参数，分别是核心工作频率，总线频率，等效频率

核心频率：sdram本身的操作频率

总线频率：ddr接口的同步时钟频率高，是等效频率的一半

等效频率：理论上操作频率

Ddr1 ： 2比特预存取（<400mhz）

Ddr2 ： 4比特预存取（<800mhz）

Ddr3 ： 8比特预存取(<1600mhz)



可以看出，ddrx的同步时钟频率高很高，一般FPGA的时钟很难达到，所以都是集成控制，IC是可以达到，

## 3.1DDRx接口

Ddx由于采用了双眼采样的预存取计数，因此控制上要比sdr发复杂的多，也会采用一种不同于sdr的接口标准。

ck : 接口同步时钟

Ck# : 接口同步时钟

cke ： 时钟屏蔽信号

cs\_n : 片选信号

ras\_n ：行选通信号

cas\_n : 列选通

we\_n : 写使能

addr[k0-1:0] : 行列复用地址

ba[B-1:0] : bank地址

**dqs ：数据选通信号**

差分时钟

差分时钟

**dqs# ：数据选通信号**

**dq[d-1:0] :双向数据信号**

**dqm[d/8-1:0] :DDRX种只保留了写掩码**

odt :ODR开关信号，为了保证dq,dqm，dqs等与数据相关信号的信号指令

一般有50欧姆，75欧姆，150欧姆

rzq :内部集成RZQ校验电路，作用就是动态天正ODT电阻值，维持在合适的阻值范围内，DDRX要求引脚下拉一个240欧姆的精准电阻。

Reset :功耗管理信号，主要用户功耗管理，

## 3.2ddrx在FPGA中的应用

Ddrx在FPGA中实际应用，对于总线频率较高的存储器，在里面集成了IC控制器，我们的芯片选型的时候选择带控制器的芯片，以xilinx为例，不同的系列的FPGA芯片，控制器也不一样，并且同一系列，不同型号的控制器也有所不同，目前xilin将复杂的IP核都统一采用AXI总线。

