实验二 寄存器堆与存储器及其应用

2022春季

zjx@ustc.edu.cn

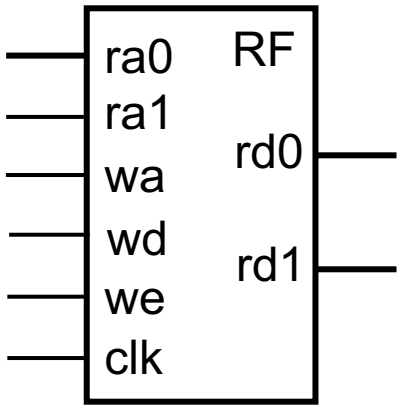
实验目标

- 掌握寄存器堆（**Register File**）和存储器的功能、时序及其应用
- 熟练掌握数据通路和控制器的设计和描述方法

实验内容

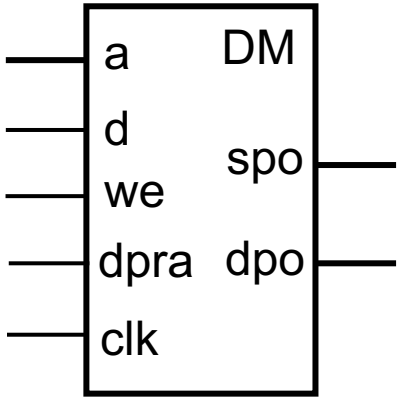
1. 寄存器堆 (Register File)

- ra0, rd0：异步读端口0
- ra1, rd1：异步读端口1
- wa, wd, we：同步写端口
- clk：时钟



2. 双端口RAM存储器

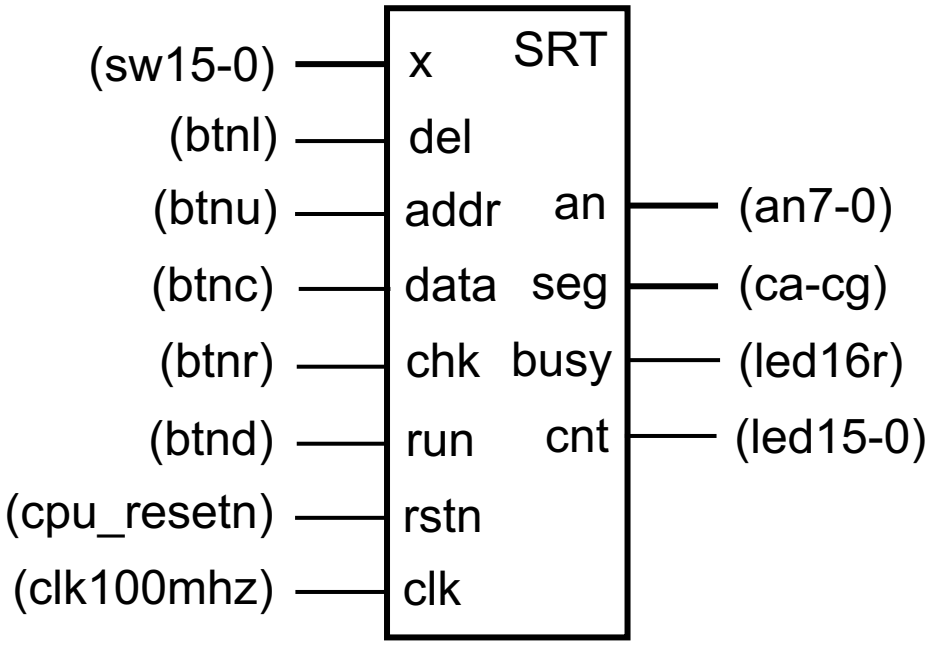
- a：1端口读/写地址
- d：1端口写入数据
- we：1端口写使能
- spo：1端口输出数据
- dpra：2端口读地址
- dpo：2端口输出数据
- clk：时钟



实验内容 (续)

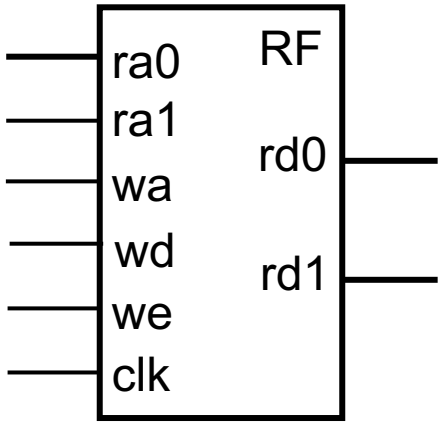
3. 数据排序：实现数据输入、存储、查看、修改和排序等

- 256个16位二进制无符号数
- x：输入1位十六进制数字
- del：删除1位十六进制数字
- addr：设置地址
- data：修改数据
- chk：查看下一项
- run：启动排序
- an, seg：数码管显示
- busy：正在排序中
- cnt：排序耗费时钟周期数



寄存器堆模块

```
module  register_file  #(
    parameter AW = 5,                   //地址宽度
    parameter DW = 32                   //数据宽度
)(
    input  clk,                         //时钟
    input [AW-1:0]  ra0, ra1,           //读地址
    output [DW-1:0]  rd0, rd1,          //读数据
    input [AW-1:0]  wa,                 //写地址
    input [DW-1:0]  wd,                 //写数据
    input we                            //写使能
);
reg [DW-1:0]  rf [0: (1<<AW)-1];    //寄存器堆
assign rd0 = rf[ra0], rd1 = rf[ra1];    //读操作
always  @(posedeg  clk)
    if (we)  rf[wa]  <=  wd;            //写操作
endmodule
```



存储器IP核

- **Vivado中有存储器IP核可以直接使用**
- **两种IP类型：分布式(Distributed)、块式(Block)存储器**
- **定制化方式：ROM/RAM、单端口/简单双端口/真正双端口等**

存储器IP核例化

- **Flow Navigator >> Project Manager >> IP Catalog**
  - Memories & Storage Elements >> RAMs & ROMs >> Distributed Memory Generator
  - 或者 Basic Elements >> Memory Elements >> Distributed Memory Generator
    - Memory config >> Memory Type: Single Port RAM
    - RST & Initialization >> Load COE File

同步写端口：a (地址)，d (数据)，we (写使能)，clk

异步读端口：a (地址)，spo (数据)

存储器IP核例化 (续)

- **Project Manager – display >> Sources >> IP Sources**
  - IP >> dist_mem_gen_0 >>Instantiation Template >> dist_mem_gen_0.veo

```
dist_mem_gen_0   your_instance_name (
  .a(a),              // input wire [4 : 0] a
  .d(d),              // input wire [15 : 0] d
  .clk(clk),          // input wire clk
  .we(we),            // input wire we
  .spo(spo)           // output wire [15 : 0] spo
);
```

实例化模板

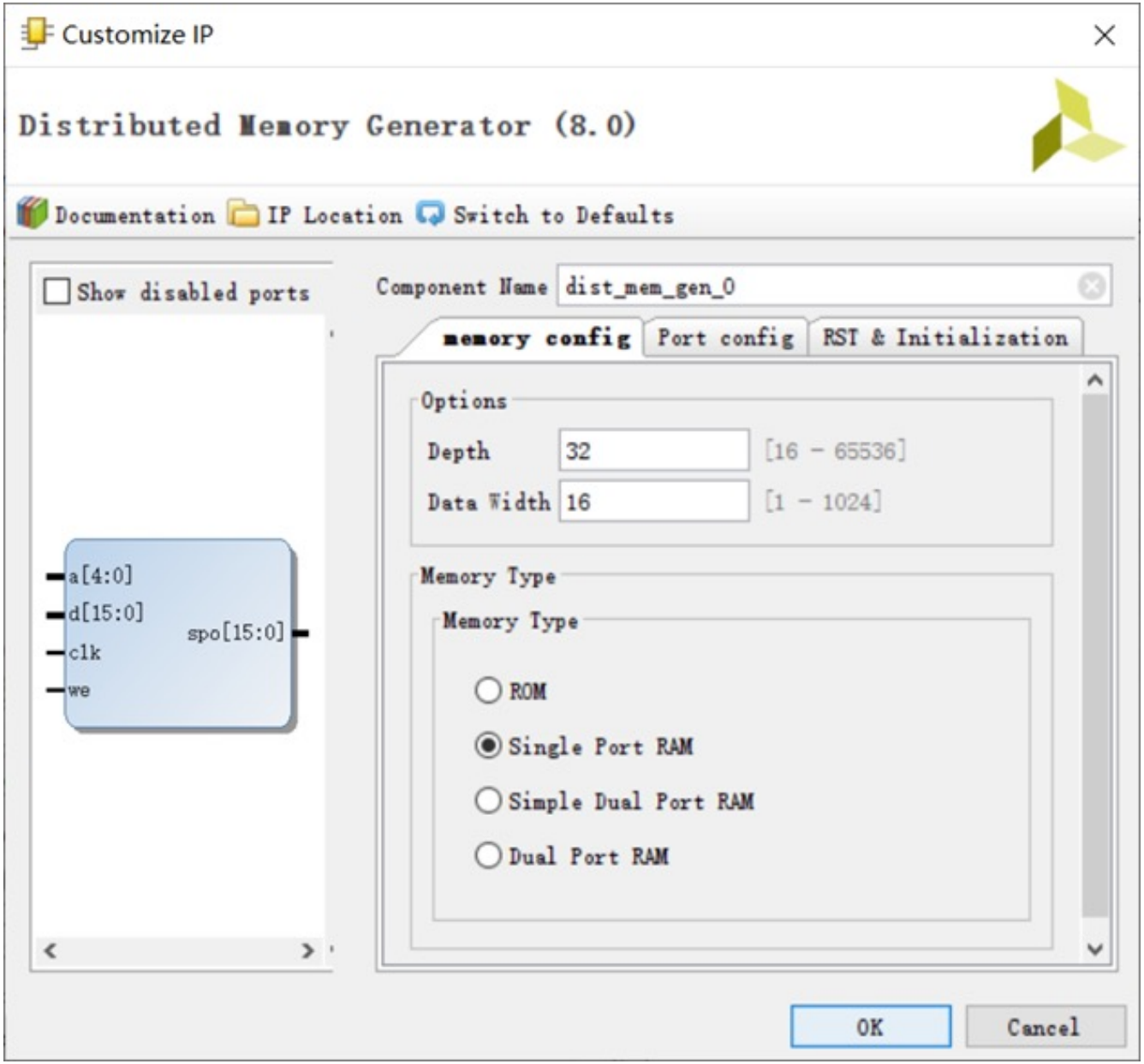
COE文件格式

- **An example COE file:**

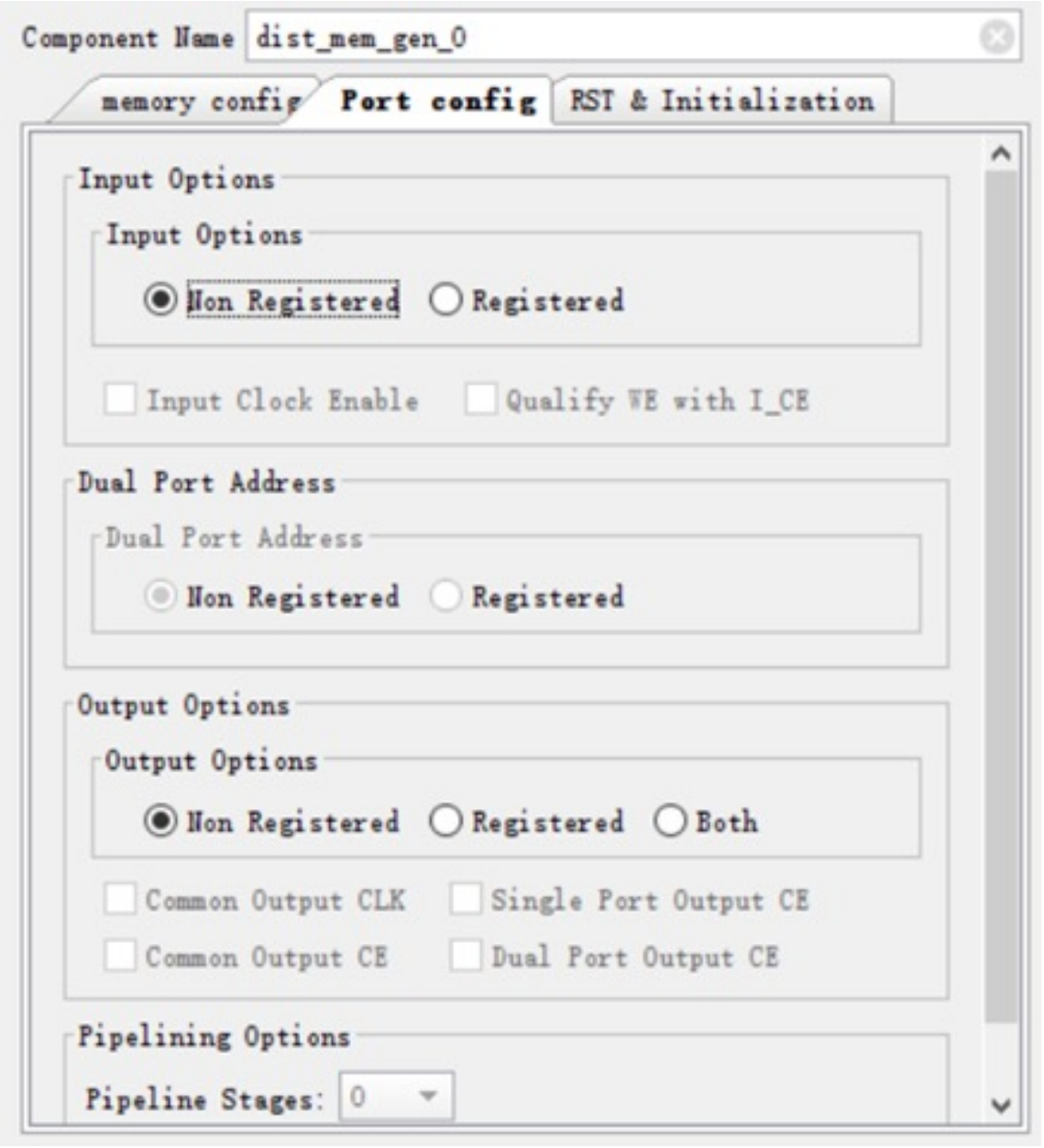
```
; Sample Initialization file for a 32x16 distributed ROM
memory_initialization_radix = 16;
memory_initialization_vector =
23f4 0721 11ff ABe1 0001 1 0A 0
23f4 0721 11ff ABe1 0001 1 0A 0
23f4 721 11ff ABe1 0001 1 A 0
23f4 721 11ff ABe1 0001 1 A 0;
```

逗号或空格分隔每项数据（不允许为负数）

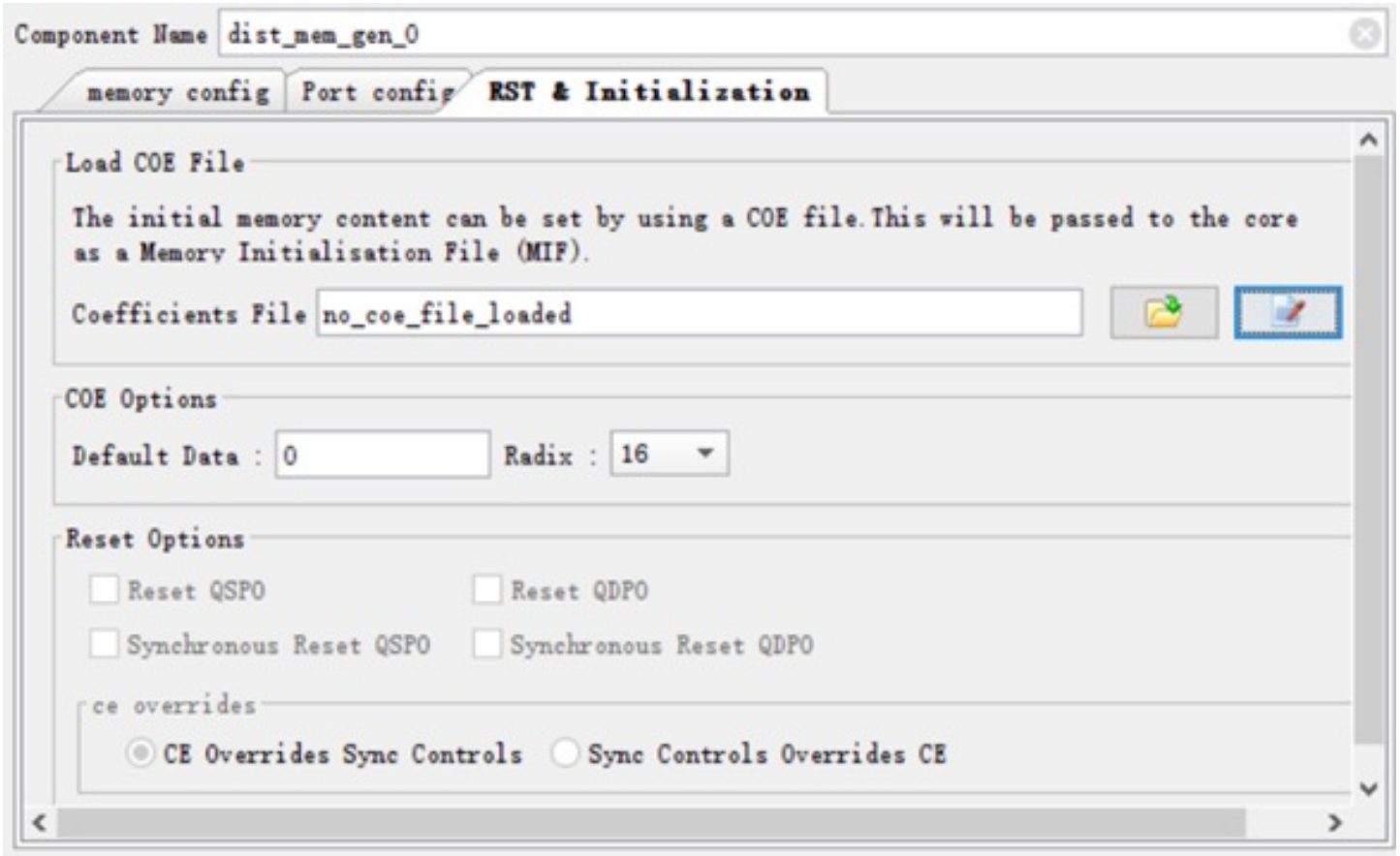
分布式存储器IP


Customize IP
Distributed Memory Generator (8.0)
Documentation
IP Location
Switch to Defaults
Show disabled ports
Component Name
dist_mem_gen_0
memory config
Port config
RST & Initialization
Options
Depth
32
[16 - 65536]
Data Width
16
[1 - 1024]
Memory Type
Memory Type
ROM
Single Port RAM
Simple Dual Port RAM
Dual Port RAM
a[4:0]
d[15:0]
clk
we
spo[15:0]
OK
Cancel

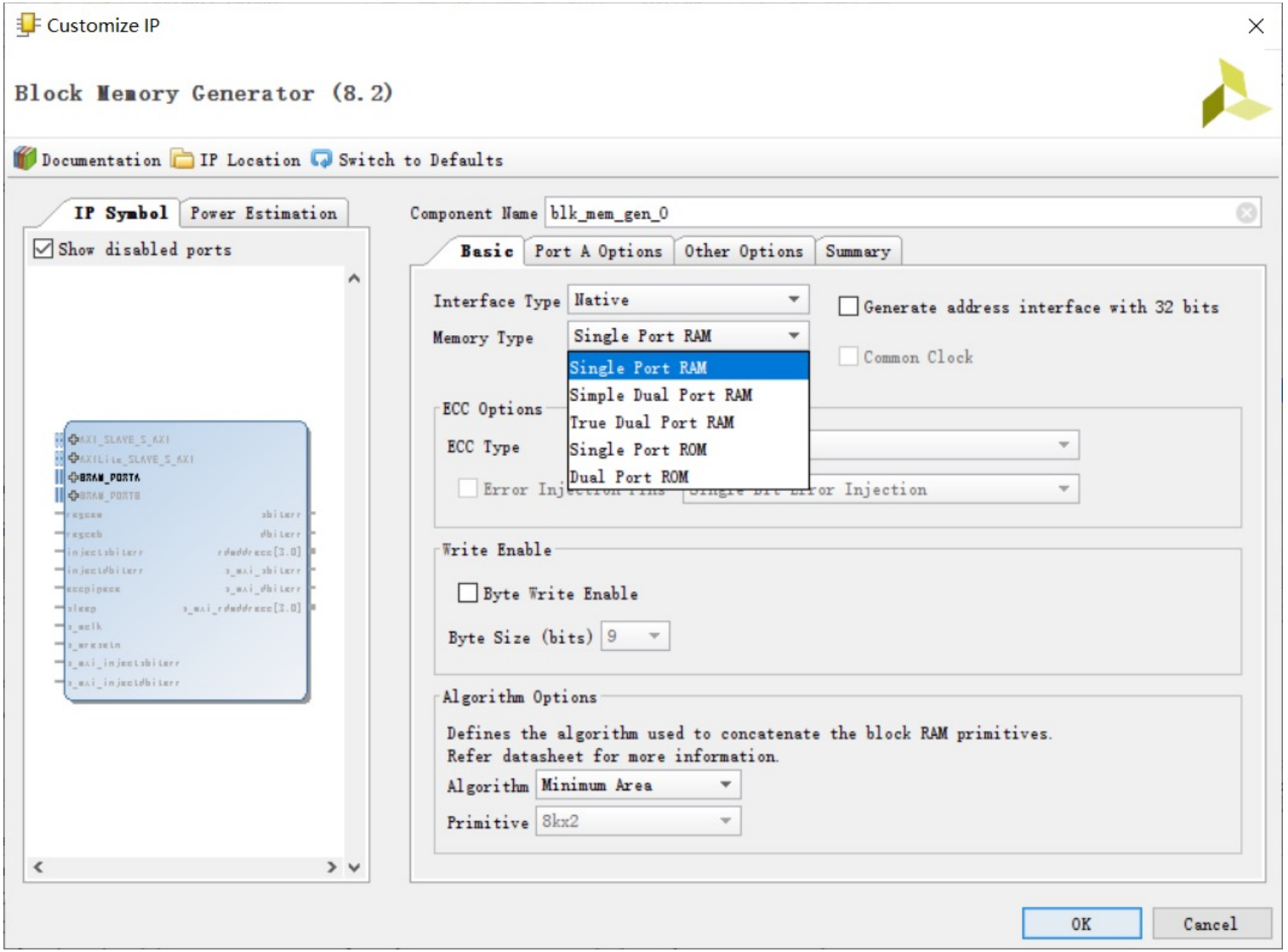
分布式存储器IP


Component Name
dist_mem_gen_0
memory config
Port config
RST & Initialization
Input Options
Input Options
Non Registered
Registered
Input Clock Enable
Qualify WE with I_CE
Dual Port Address
Dual Port Address
Non Registered
Registered
Output Options
Output Options
Non Registered
Registered
Both
Common Output CLK
Single Port Output CE
Common Output CE
Dual Port Output CE
Pipelining Options
Pipeline Stages:
0

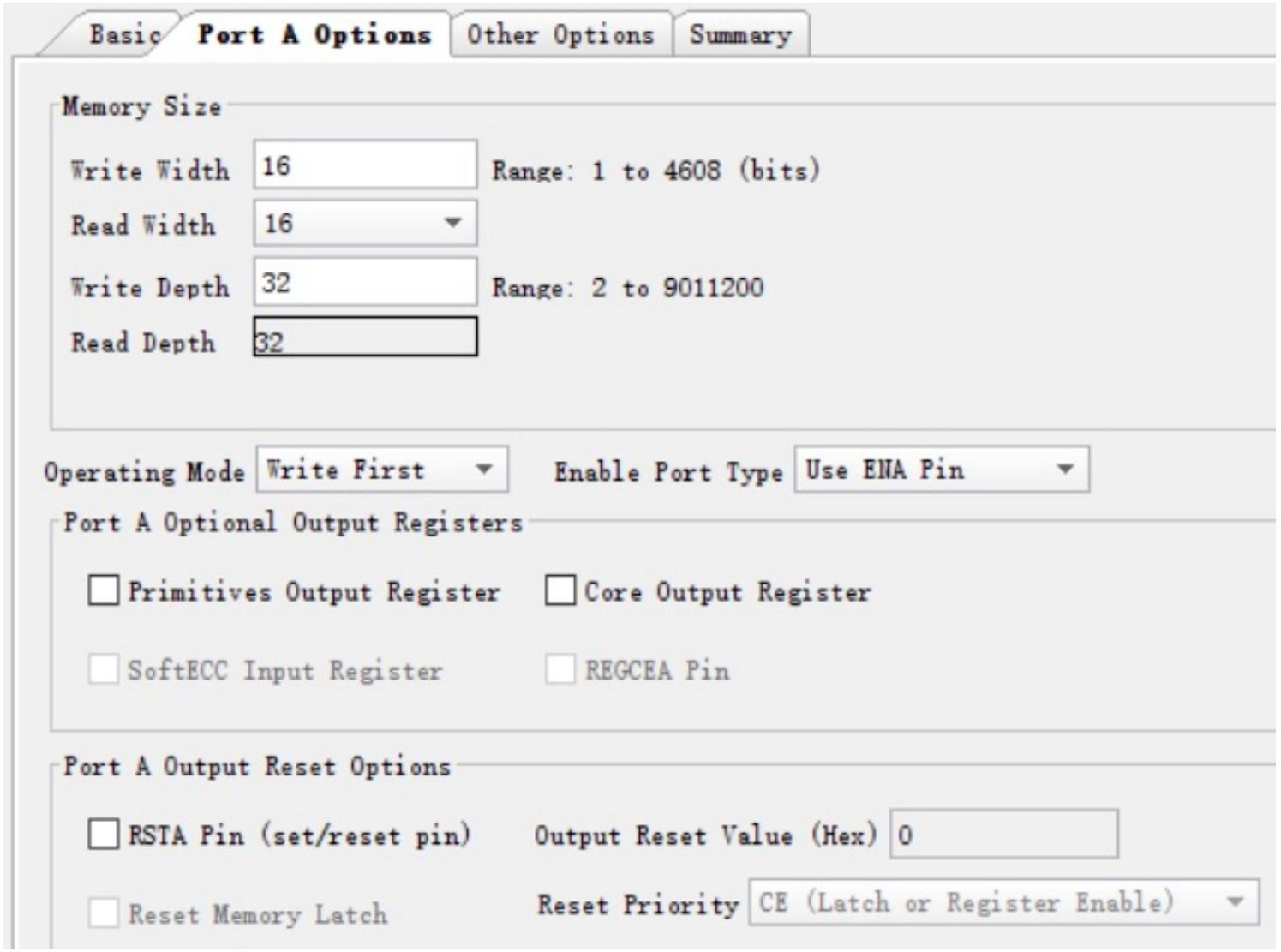
分布式存储器IP


Component Name
dist_mem_gen_0
memory config
Port config
RST & Initialization
Load COE File
The initial memory content can be set by using a COE file. This will be passed to the core as a Memory Initialisation File (MIF).
Coefficients File
no_coe_file_loaded
COE Options
Default Data : 0
Radix : 16
Reset Options
Reset QSPO
Reset QDPO
Synchronous Reset QSPO
Synchronous Reset QDPO
ce overrides
CE Overrides Sync Controls
Sync Controls Overrides CE

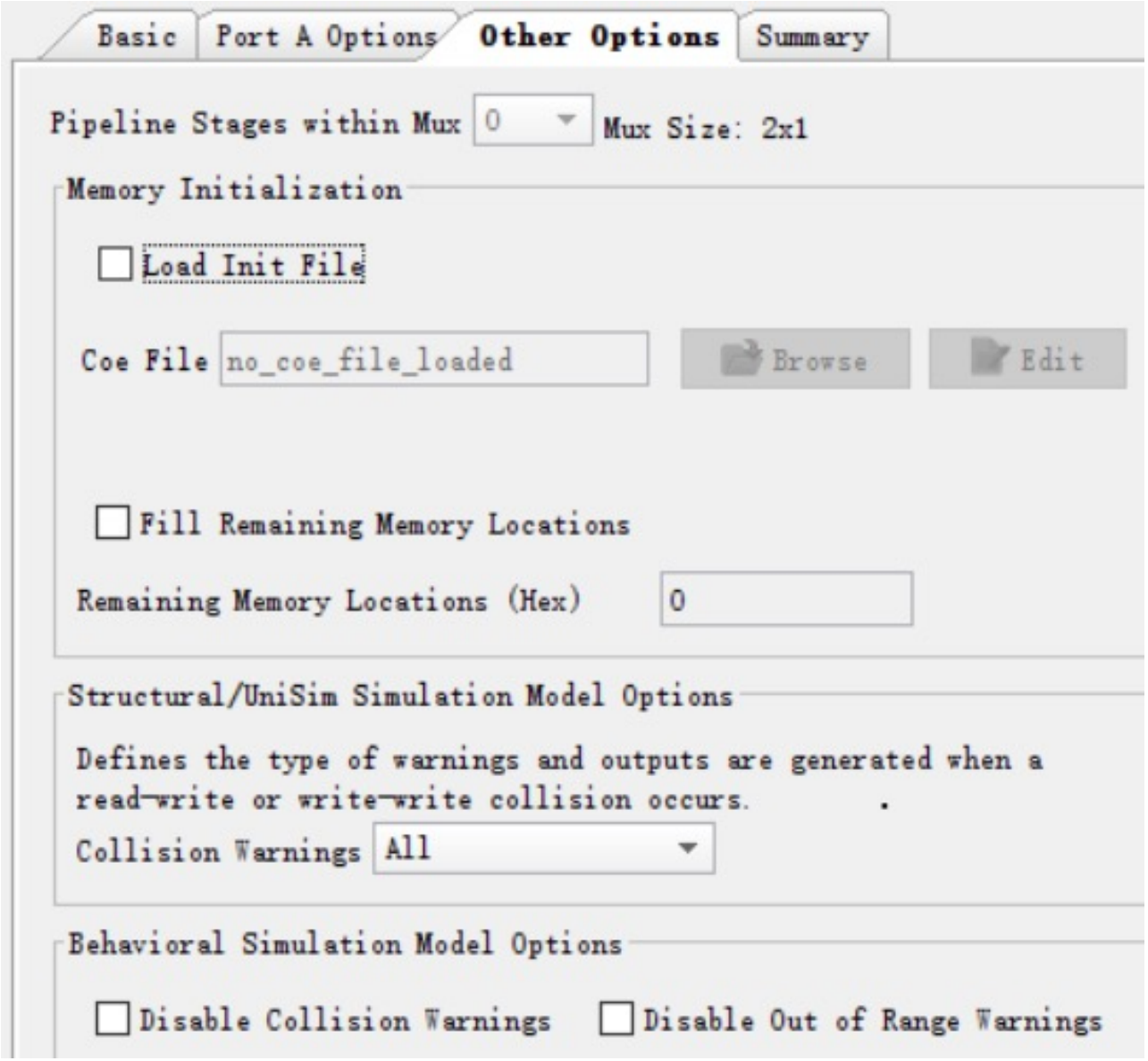
块式存储器IP


Customize IP
Block Memory Generator (8.2)
Documentation
IP Location
Switch to Defaults
IP Symbol
Power Estimation
Show disabled ports
Component Name
blk_mem_gen_0
Basic
Port A Options
Other Options
Summary
Interface Type
Native
Generate address interface with 32 bits
Memory Type
Single Port RAM
Common Clock
Single Port RAM
Simple Dual Port RAM
True Dual Port RAM
Single Port ROM
Dual Port ROM
ECC Options
ECC Type
Write Enable
Byte Write Enable
Byte Size (bits)
9
Algorithm Options
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.
Algorithm
Minimum Area
Primitive
8kx2
OK
Cancel

块式存储器IP

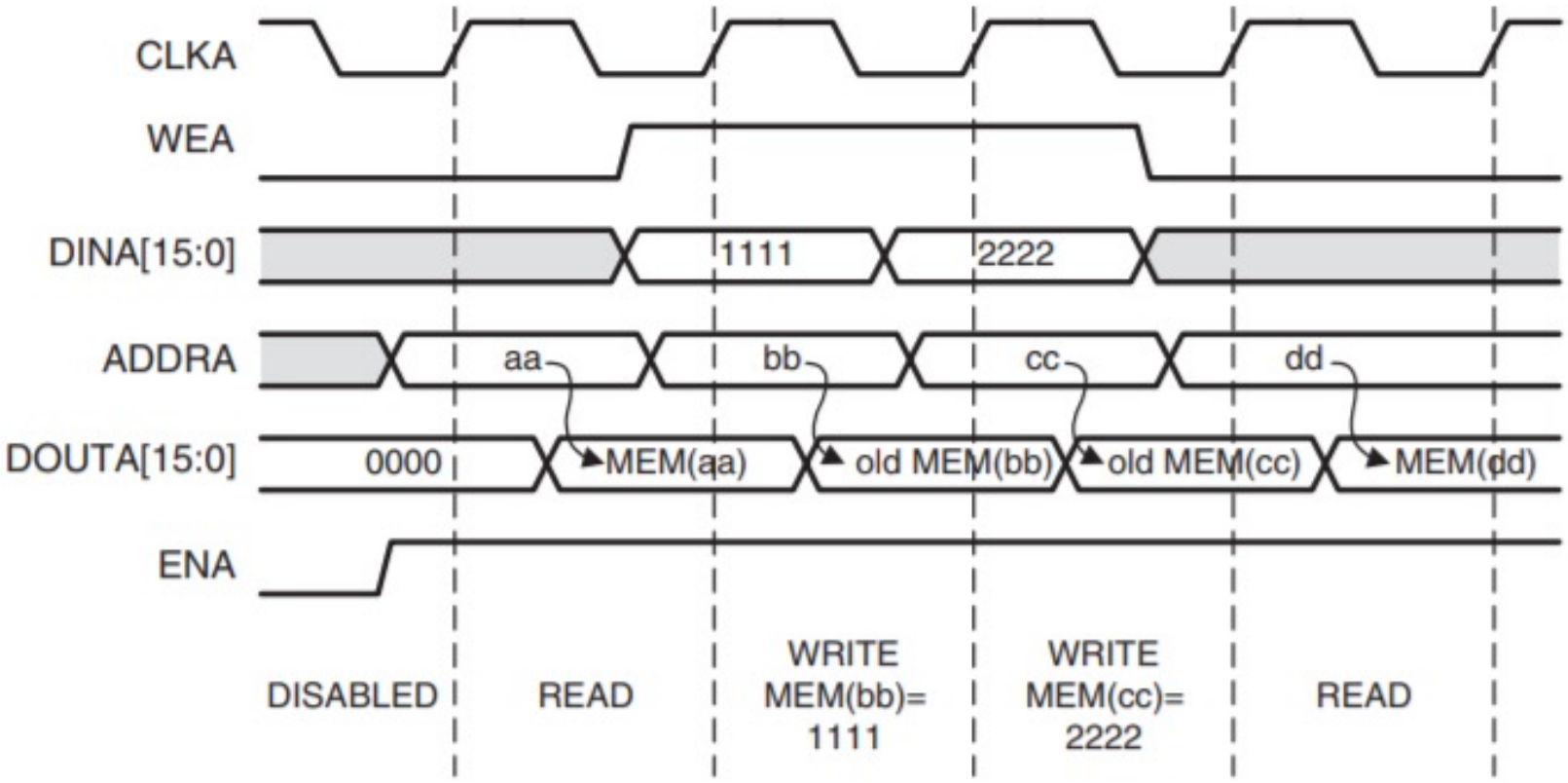

Basic
Port A Options
Other Options
Summary
Memory Size
Write Width
16
Range: 1 to 4608 (bits)
Read Width
16
Write Depth
32
Range: 2 to 9011200
Read Depth
32
Operating Mode
Write First
Enable Port Type
Use ENA Pin
Port A Optional Output Registers
Primitives Output Register
Core Output Register
SoftECC Input Register
REGCEA Pin
Port A Output Reset Options
RSTA Pin (set/reset pin)
Output Reset Value (Hex)
0
Reset Memory Latch
Reset Priority
CE (Latch or Register Enable)

块式存储器IP


Basic
Port A Options
Other Options
Summary
Pipeline Stages within Mux
0
Mux Size: 2x1
Memory Initialization
Load Init File
Coe File
no_coe_file_loaded
Browse
Edit
Fill Remaining Memory Locations
Remaining Memory Locations (Hex)
0
Structural/UniSim Simulation Model Options
Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.
Collision Warnings
All
Behavioral Simulation Model Options
Disable Collision Warnings
Disable Out of Range Warnings

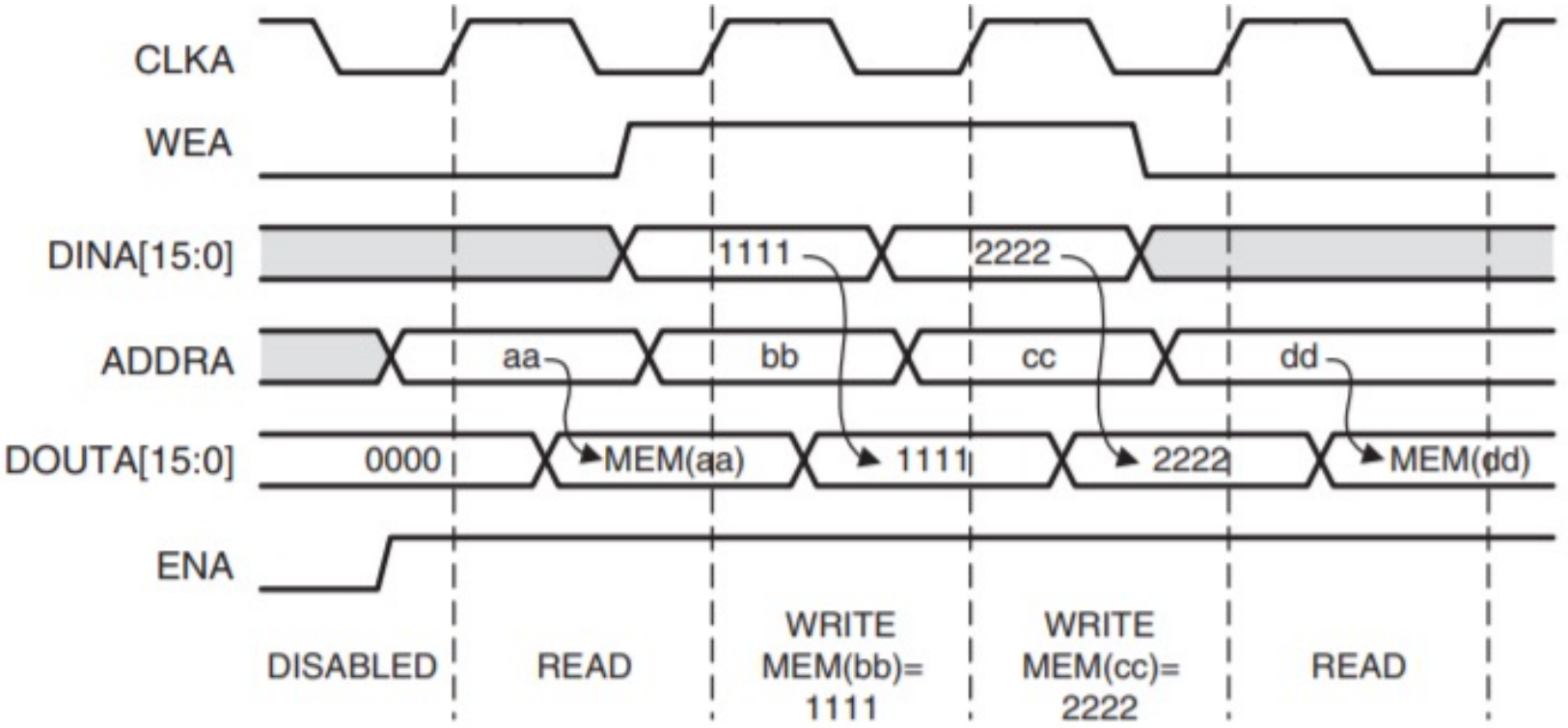
存储器时序

- **Read First Mode**



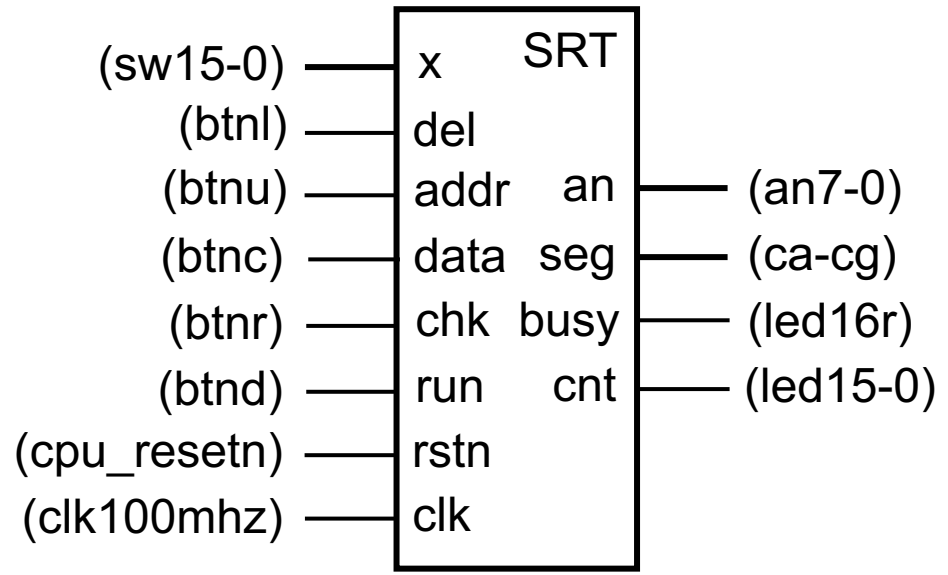
存储器时序 (续)

- **Write First Mode**



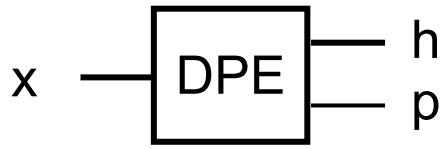
数据排序

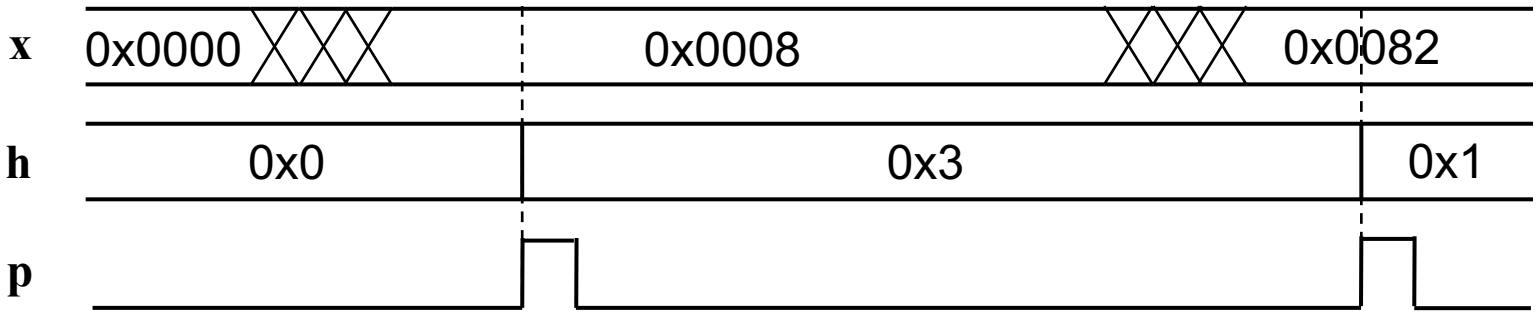
- **数据输入/输出**
  - 采用分布式双端口存储器保存数据，例化时可以初始化数据
  - 利用chk查看数据，数码管显示存储器的地址和数据
  - 利用x、del、addr、data设置地址和修改数据
- **数据排序**
  - run启动排序, 同时启动时钟计数cnt，busy置1
  - 排序时不能人工查看和修改数据
  - 排序结束后停止计数，busy清零



开关数字输入

- **假定任何时刻只改变16个开关x(sw15-0)中一个开关状态**
- **每次向上或向下拨动一次开关，生成1位十六进制数h，即4位二进制数，同时产生持续1个时钟周期的脉冲p**
  - DPE：去抖动、取双边沿、编码
    - h：1位十六进制数字，p：单次脉冲





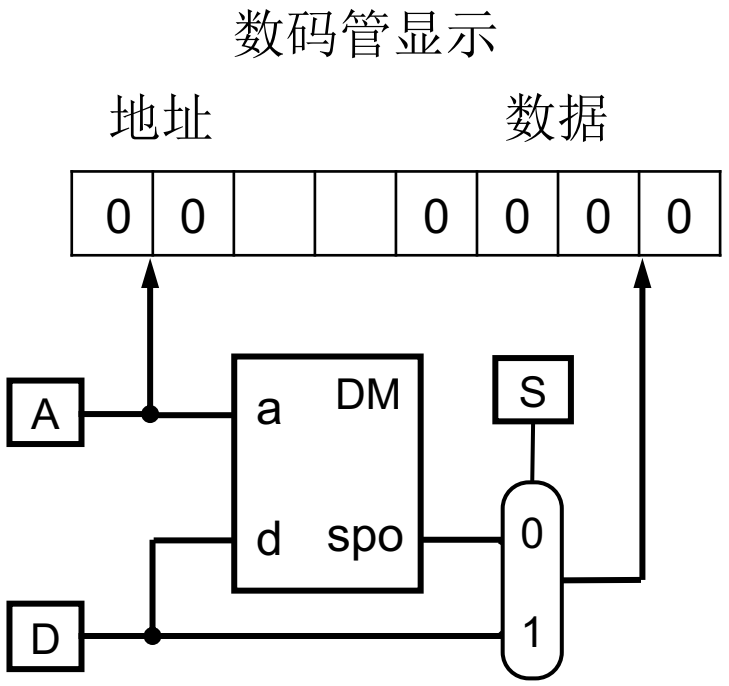
数据输入/输出

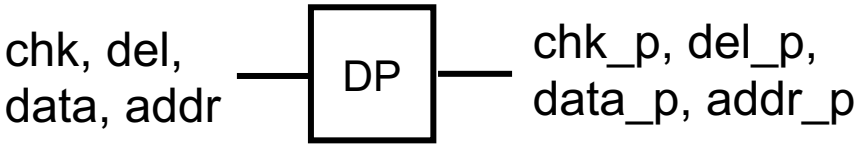
- 假定初始状态，存储器各单元的数据与其地址相同

序号	操作	数码管显示	操作说明
1	复位	00 0000	显示0单元
2	chk (btnr)	01 0001	查看下一单元
3	x (sw2)	01 0002	编辑数据或地址：输入2
4	del (btnl)	01 0000	删除2
5	x (sw10)	01 000A	输入A
6	data (btnc)	02 0002	修改数据，并查看下一单元
7	chk (btnr)	03 0003	查看下一单元
8	x (sw11)	03 000B	编辑数据或地址：输入B
9	x (sw15)	03 00BF	输入F
10	del (btnl)	03 000B	删除F
11	x (sw5)	03 00B5	输入5
12	addr (btnu)	02 000A	设置地址，并查看该单元
13	chk (btnr)	03 0003	查看下一单元

数据输入/输出 (续1)

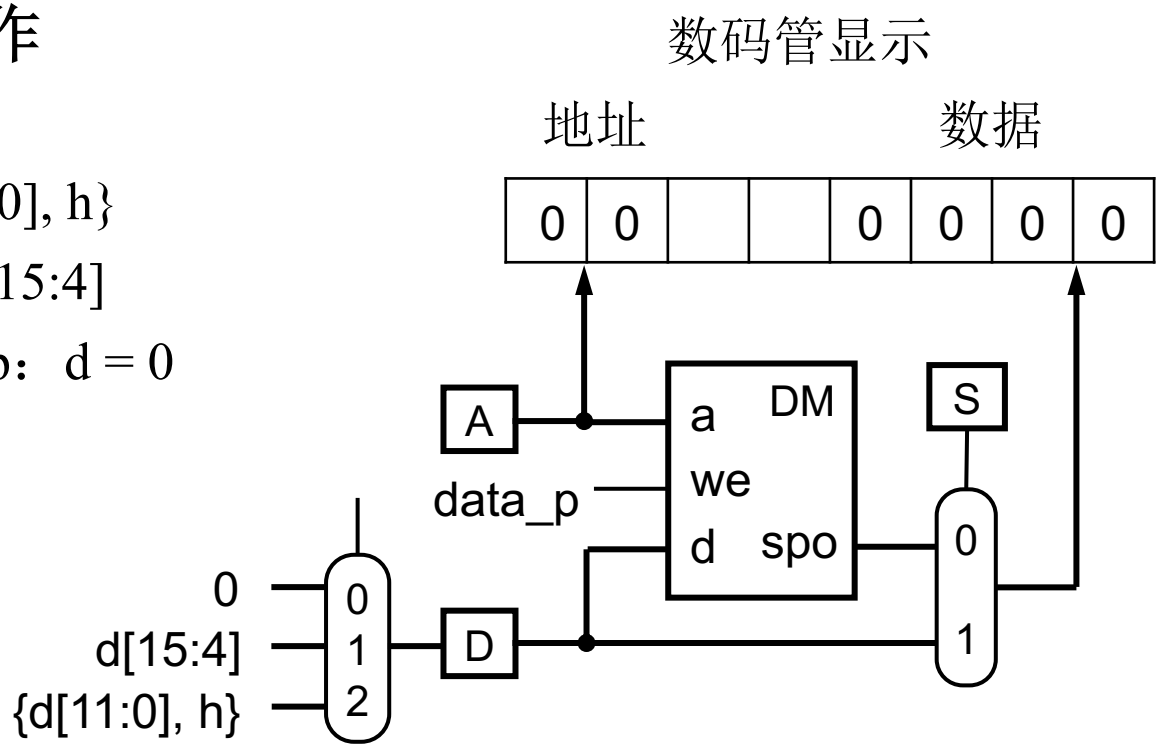
- 数据通路及其操作
  - rstn：a = 0, d = 0, s = 0
  - chk_p：a = a + 1, s = 0
  - p：d = {d[11:0], h}, s = 1
  - del_p：d = d[15:4], s = 1
  - data_p：M[a] = d, d = 0, a = a + 1, s = 0
  - addr_p：a = d[7:0], d = 0, s = 0

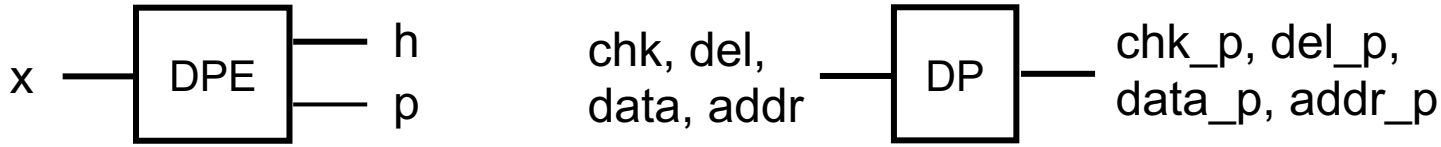




数据输入/输出 (续2)

- **寄存器D的操作**
  - rstn：d = 0
  - p：d = {d[11:0], h}
  - del_p：d = d[15:4]
  - data_p, addr_p：d = 0
- **DM的写操作**
  - we = data_p
- **……**





排序模块接口

```
module  sort (
  input  clk,
  input  rstn,

  input [15:0]  x,            //输入1位十六进制数字
  input   del,                //删除1位十六进制数字
  input  addr,                //设置地址
  input  data,                //修改数据
  input chk,                  //查看下一项
  input run,                  //启动排序

  output [7:0]  an,           //数码管显示存储器地址和数据
  output [6:0]  seg,
  output busy,                //1—正在排序，0—排序结束
  output [15:0]  cnt          //排序耗费时钟周期数
);
```

实验步骤

1. **完成32x32位的寄存器堆的功能仿真**
   - 寄存器堆的0号寄存器内容恒定为零
   - 寄存器堆的写操作优先于读操作
2. **完成256x16位的分布式和块式单端口RAM IP核的功能仿真和对比**
   - 分布式和块式存储器的读操作
   - 块式存储器写操作优先和读操作优先
3. **完成排序电路的数据通路和控制器设计和功能仿真，并将排序电路下载至FPGA中测试**
4. **选项：将数据量增大至4096x16位，分别采用分布式和块式存储器保存数据并排序，对比电路资源和性能**

The End