实验二 寄存器堆与队列

zjx@ustc.edu.cn

2020.4.29

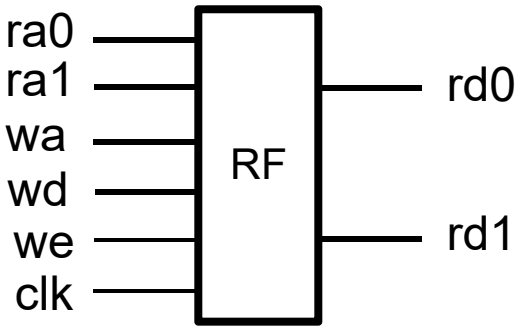
实验目标

- 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

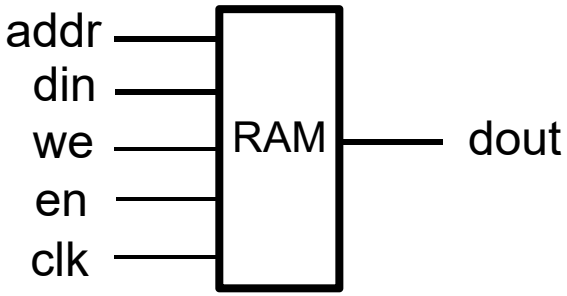
1. 寄存器堆（Register File）

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



2. RAM存储器

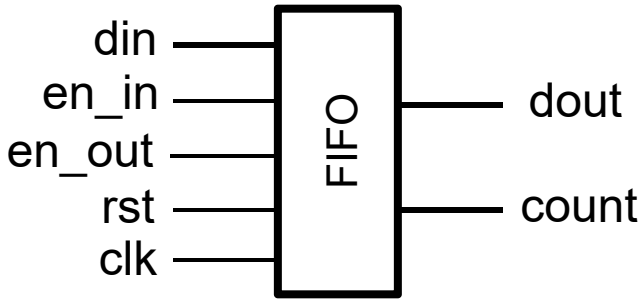
- clk：时钟
- en：总使能
- we：写使能
- addr: 读/写地址
- din：输入数据
- dou：输出数据



实验内容（续）

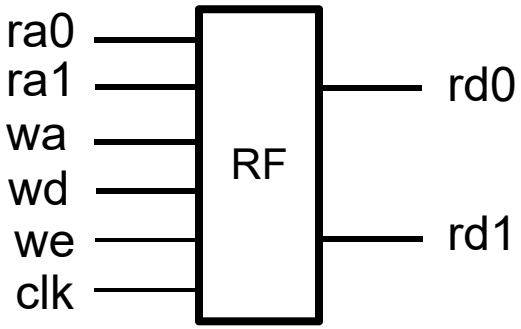
3. FIFO队列：利用例化的存储器IP（16 x 8位块式的单端口RAM）和适当逻辑实现最大长度为16的FIFO队列

- en_in, en_out：入/出队列使能，一次有效仅允许操作一个数据
- din, dout：入/出队列数据
- count：队列中有效数据个数
- clk, rst：时钟，复位



reg_file 端口定义

```
module register_file                    //32 x WIDTH 寄存器堆
    #(parameter WIDTH = 32)             //数据宽度
    (clk,                               //时钟（上升沿有效）
    input [4:0] ra0,                    //读端口 0 地址
    output [WIDTH-1:0] rd0,             //读端口 0 数据
    input [4:0] ra1,                    //读端口 1 地址
    output [WIDTH-1:0] rd1,             //读端口 1 数据
    input [4:0] wa,                     //写端口地址
    input we,                           //写使能，高电平有效
    input [WIDTH-1:0] wd                //写端口数据
    );
```



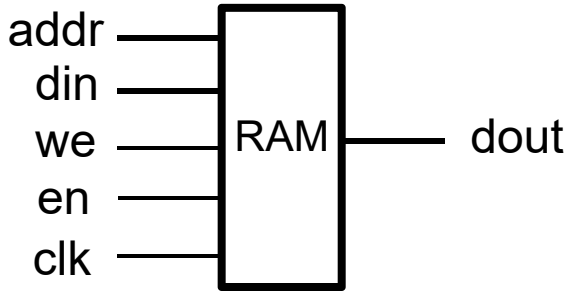
单端口RAM行为方式描述

```
reg [ _____ ] addr_reg;
reg [ _____ ] mem[ _____ ];

//初始化 RAM 的内容
initial
    $readmemh("初始化数据文件名", mem);

assign dout = mem[addr_reg];

always@(posedge clk) begin
   if(en) begin
      addr_reg <= addr;
      if(we)
         mem[addr] <= din;
   end
end
```

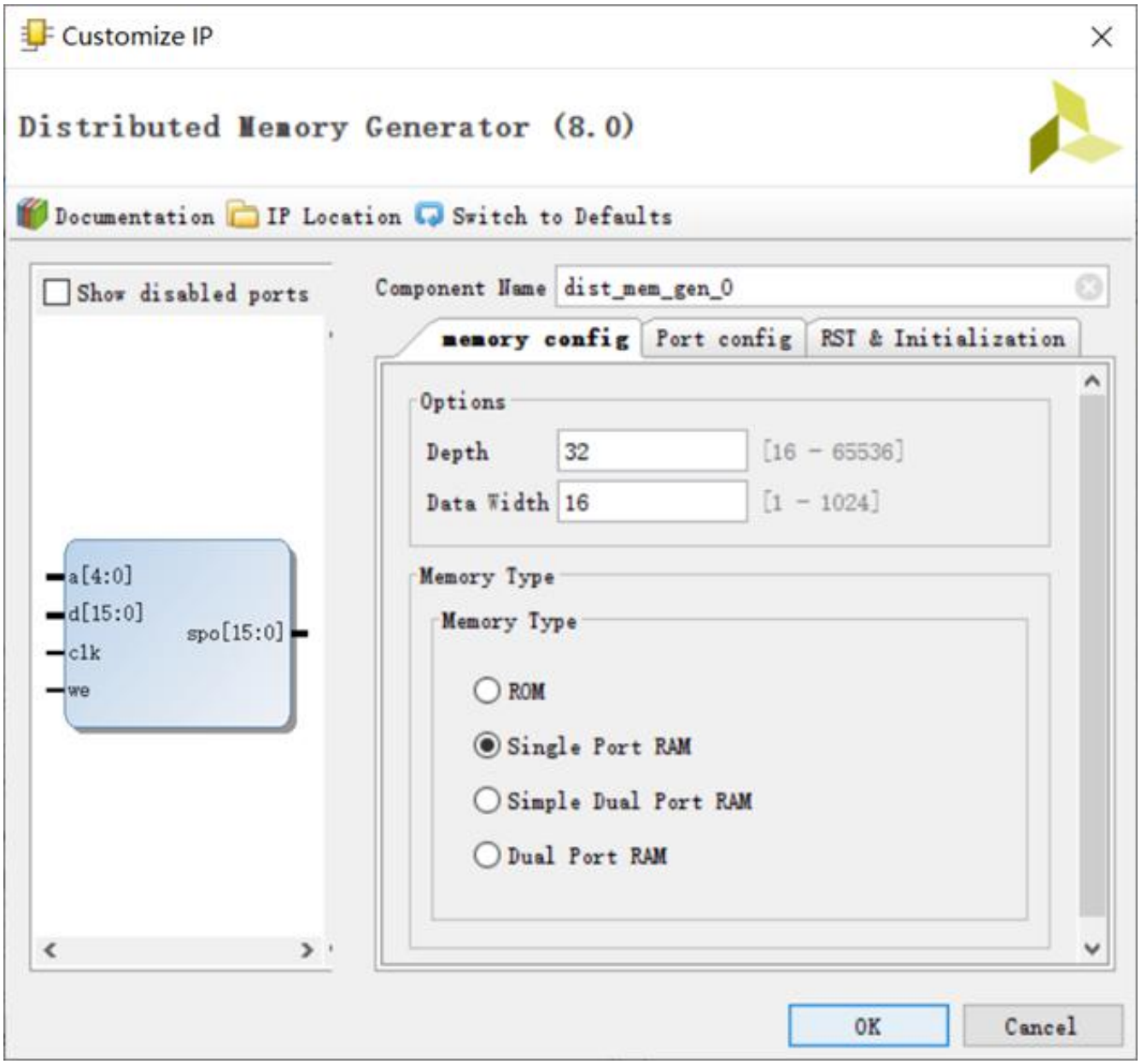


Vivado例化存储器IP

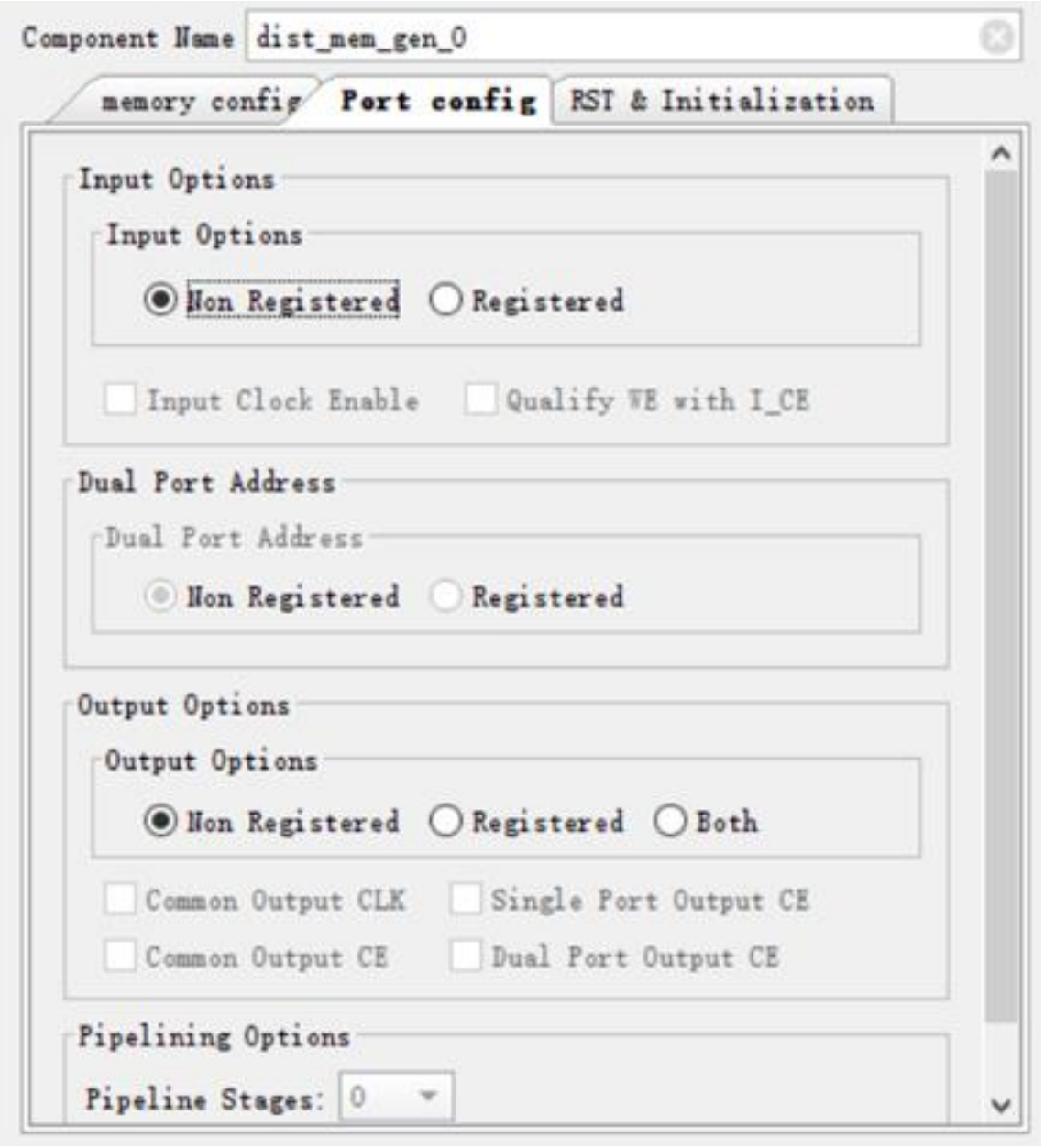
- 两种IP类型：分布式（Distributed）、块式（Block）存储器
- 定制化方式：ROM/RAM、单端口/简单双端口/真正双端口等
- 利用COE文件初始化存储器内容，例如

```
memory_initialization_radix = 16;
memory_initialization_vector =
23f4, 0721, 11ff, ABe1, 0001, 1, 0A, 0;
```

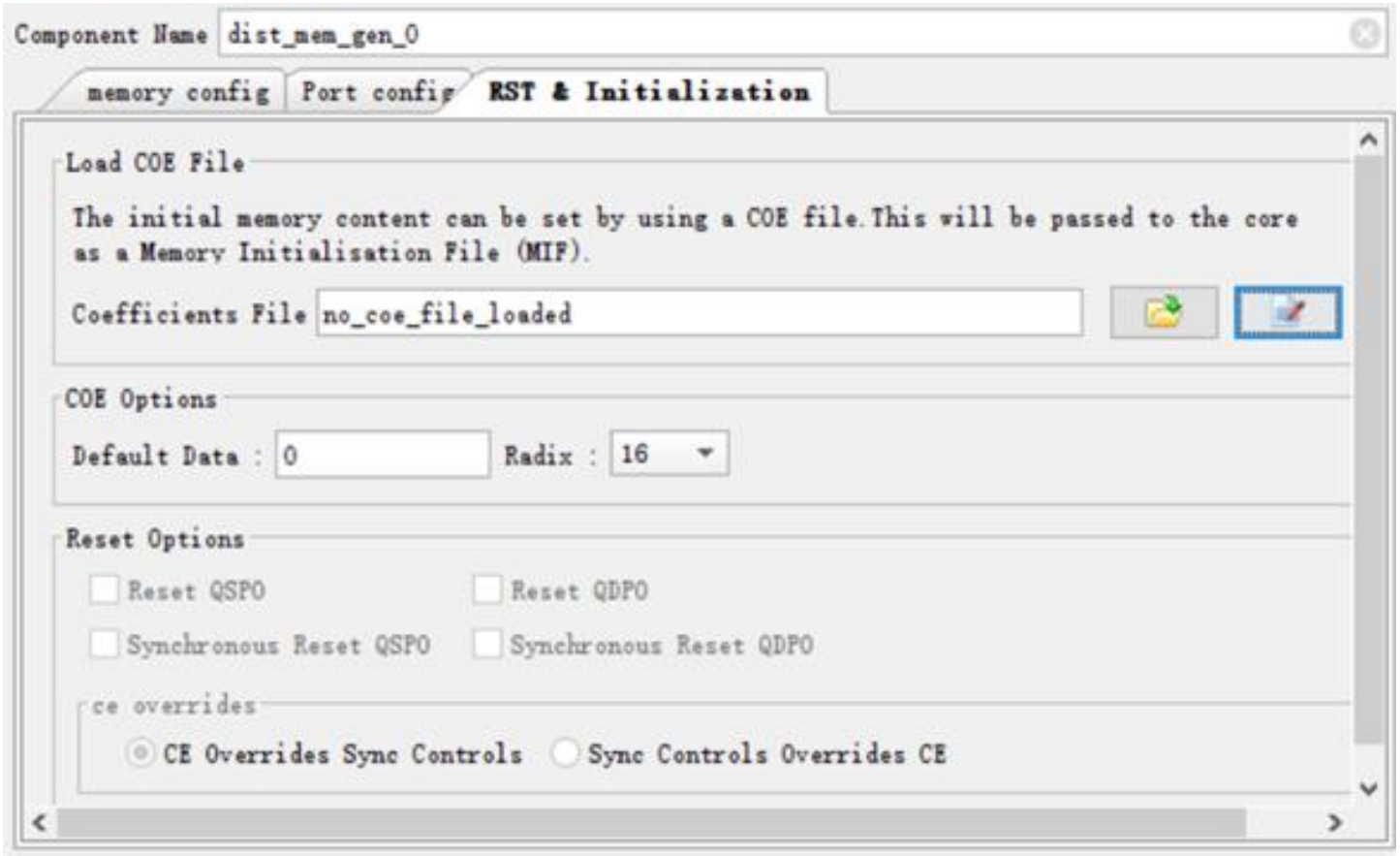
分布式存储器IP


Customize IP
Distributed Memory Generator (8.0)
Documentation
IP Location
Switch to Defaults
Show disabled ports
a[4:0]
d[15:0]
clk
we
spo[15:0]
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
OK
Cancel

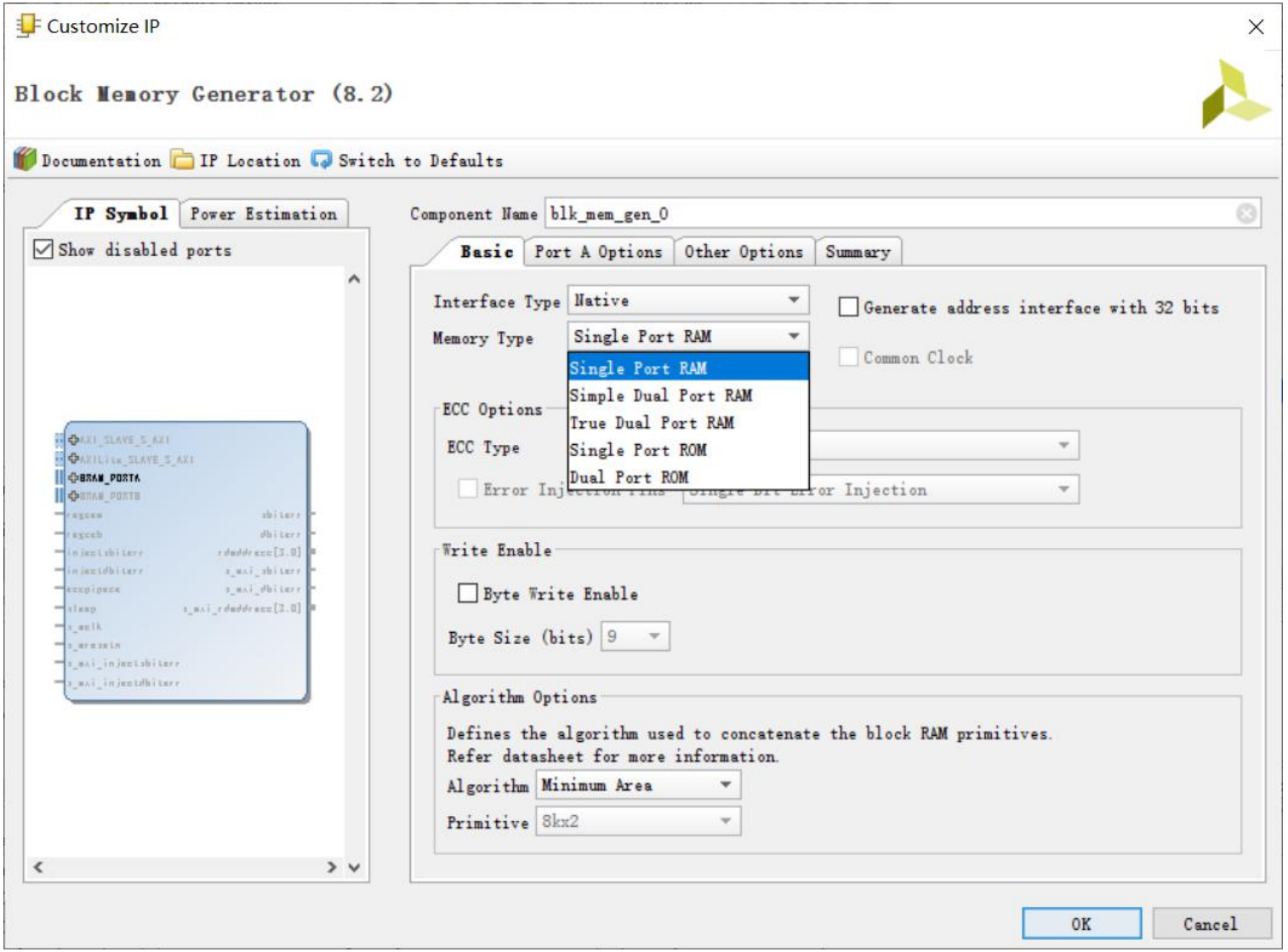
分布式存储器IP


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

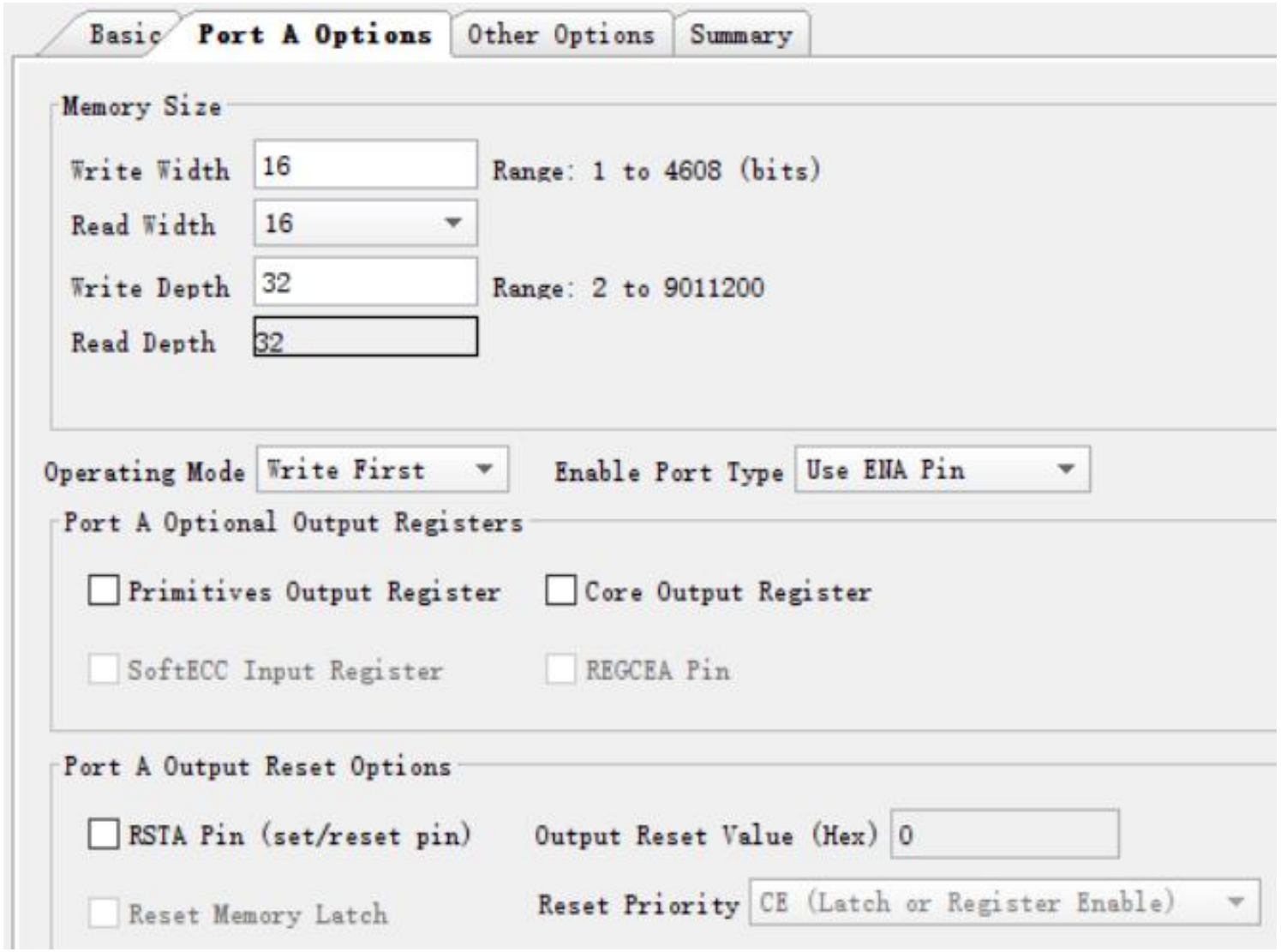
分布式存储器IP


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

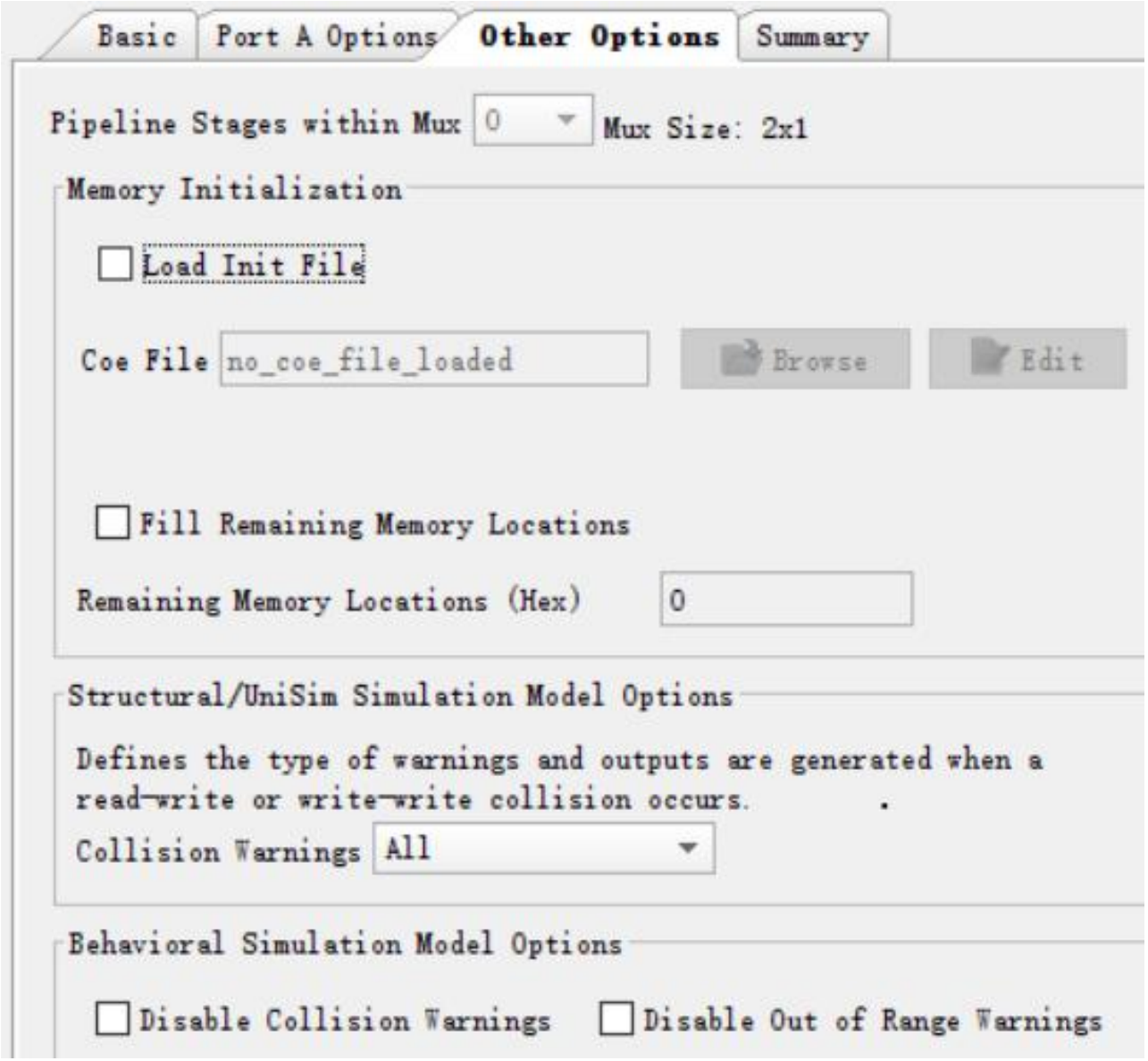
块式存储器IP


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

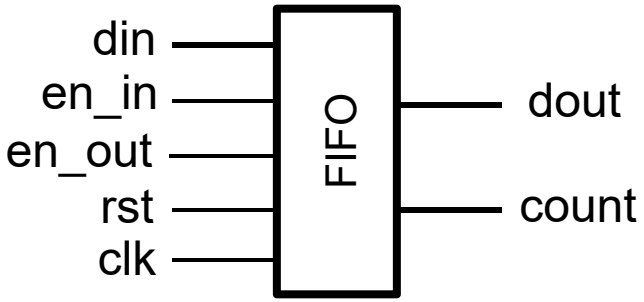

Basic
Port A Options
Other Options
Summary
Memory Size
Write Width
16
Range: 1 to 4608 (bits)
Read Width
16
Write Depth
32
Range: 2 to 9011200
Read Depth
32
Operating Mode
Write First
Enable Port Type
Use ENA Pin
Port A Optional Output Registers
Primitives Output Register
Core Output Register
SoftECC 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

FIFO端口定义

```
module fifo
    (input clk, rst,        //时钟（上升沿有效）、异步复位（高电平有效）
     input [7:0] din,       //入队列数据
     input en_in,           //入队列使能，高电平有效
     input en_out,          //出队列使能，高电平有效
     output [7:0] dout,     //出队列数据
     output [4:0] count     //队列数据计数
    );
```

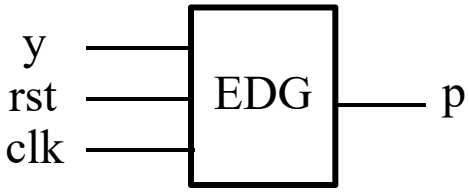


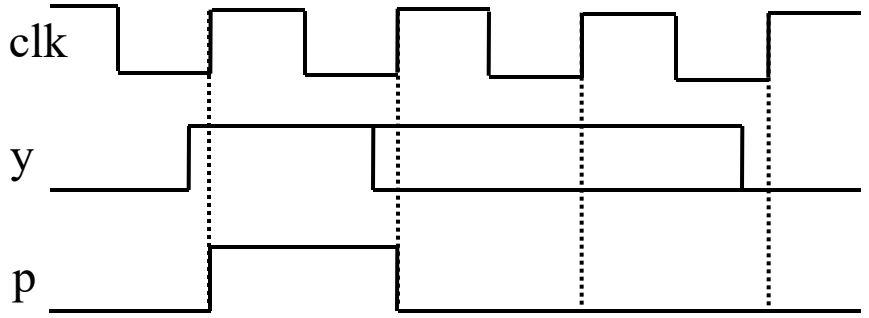
示例：取边沿电路

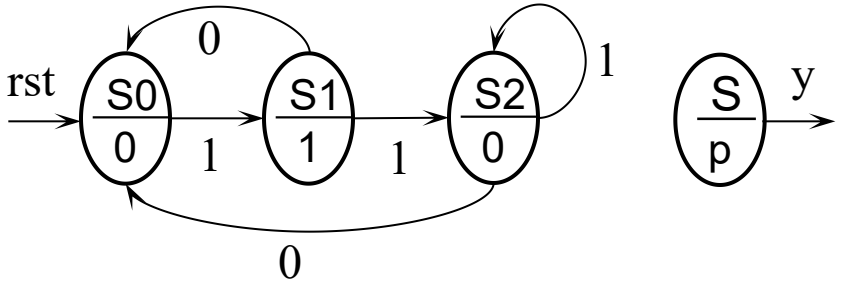
```
//output logic
assign  p = (state==S1);

//state logic
always @(posedge clk, posedge rst)
  if (rst) state <= S0;
  else state <= next_state;

//next state logic
always @* begin
  next_state = state;
  case (state)
    S0: if (y) next_state = S1;
    ……
    default: next_state = S0;
  endcase
end
```

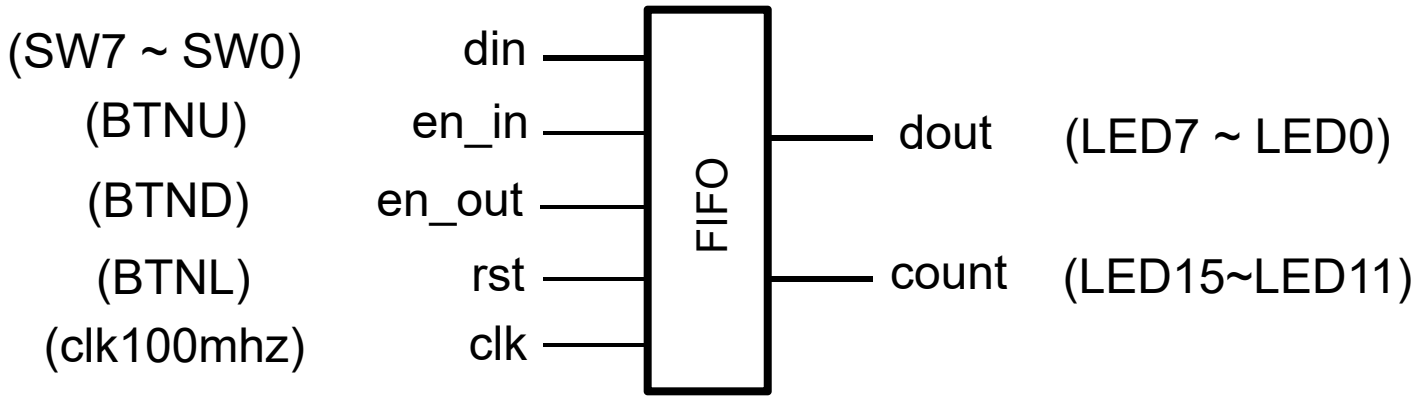






实验步骤

1. 行为方式描述参数化寄存器堆，功能仿真
2. IP例化分布式和块式16 x 8位单端口RAM，功能仿真和对比
3. 设计FIFO队列电路的数据通路和控制器，结构化方式描述数据通路，两段式FSM描述控制器，功能仿真
4. FIFO队列电路下载至FPGA中测试



实验检查

- 检查寄存器堆和RAM的功能仿真；
- 检查FIFO队列电路的功能仿真；
- 检查FIFO队列电路下载到FPGA后的运行功能

思考题

- 如何利用寄存器堆和适当电路设计实现可变个数的数据排序电路？

The End