



中国科学技术大学
University of Science and Technology of China

计算机组成原理

Lab2 寄存器堆与存储器及其应用

计算机实验教学中心

2023-4-3

实验目标

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

实验原理

1. 寄存器堆

□ 寄存器堆介绍

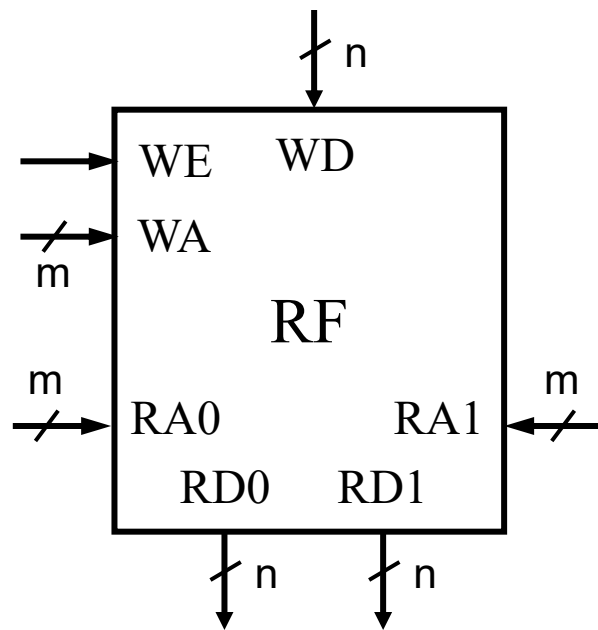
✓ 处理器的32个通用寄存器位于一个叫做寄存器堆 (register file) 的结构中。

✓ 1个写端口

- WA: 写地址
- WD: 写入数据
- WE: 写使能

✓ 2个读端口

- RA0、RA1: 读地址
- RD0、RD1: 读出数据

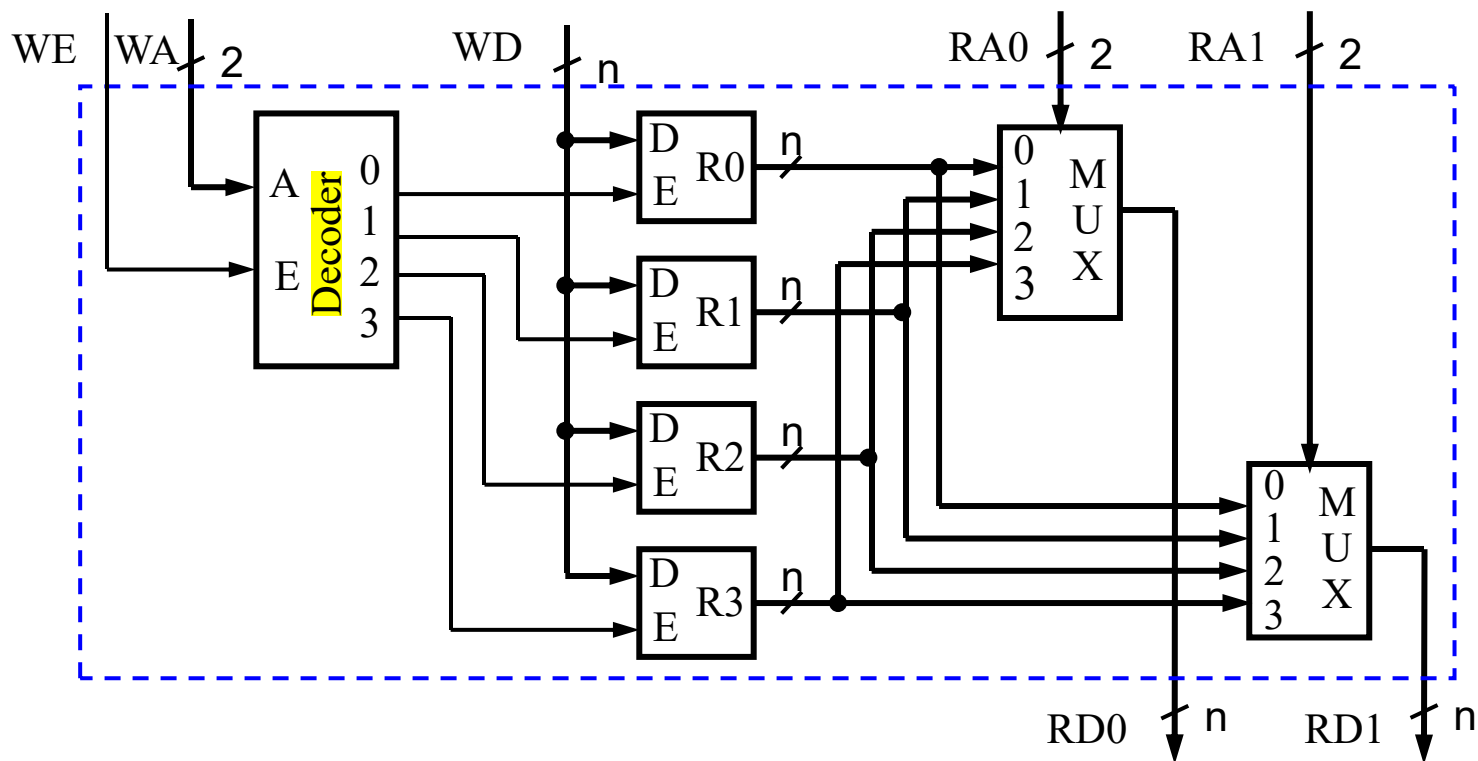


三端口的 $2^m \times n$ 位寄存器堆外形图

实验原理

1. 寄存器堆

□ 寄存器堆结构



三端口4×n寄存器堆

实验原理

2.存储器IP核

□ 存储器IP核介绍

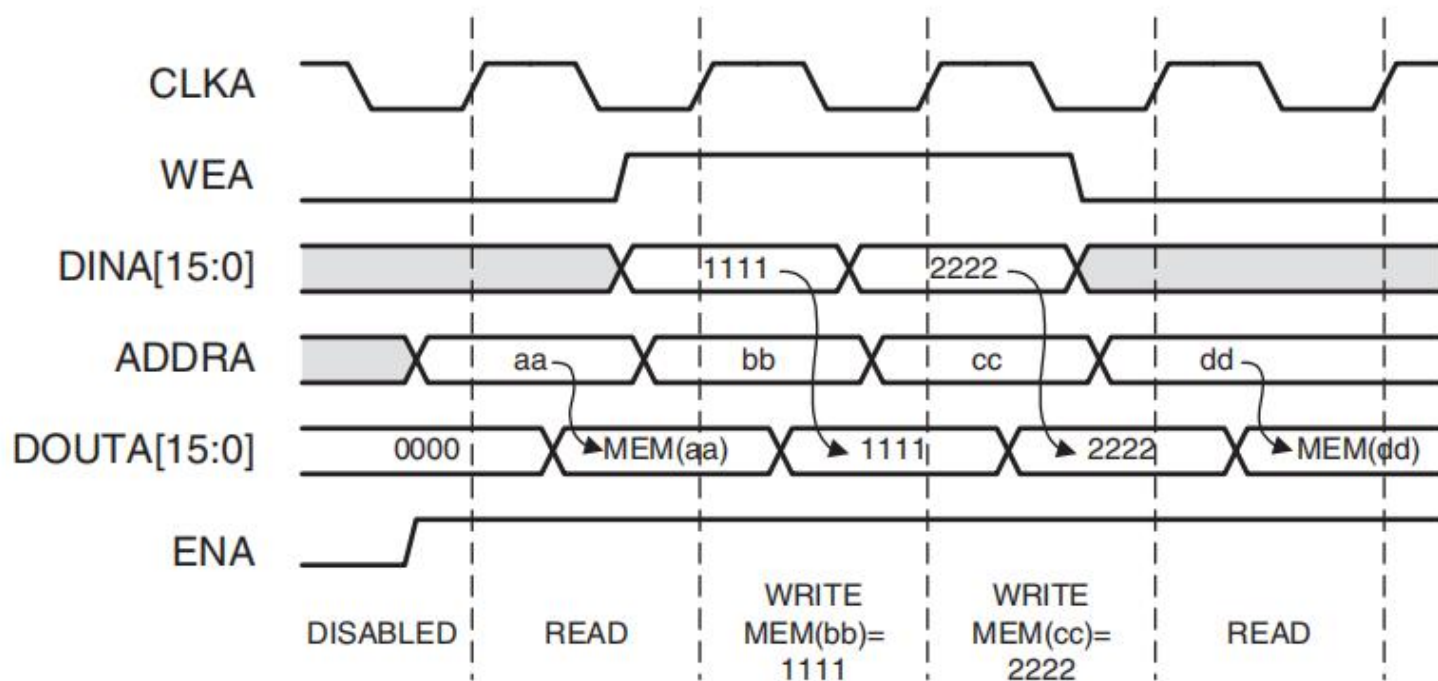
- ✓ 两种IP类型：分布式 (Distributed)、块式 (Block) 存储器
- ✓ 定制化方式：ROM/RAM、单端口/简单双端口/真正双端口等
- ✓ 分布式存储器端口
 - 同步写端口：a (地址), d (数据), we (写使能), clk
 - 异步读端口：a (地址), spo (数据)
- ✓ 块式存储器端口
 - 同步写端口：addr (地址), din (数据), we (写使能), clk
 - 同步读端口：addr (地址), dout (数据), clk
 - 使能端口：en (读、写使能)

实验原理

2.存储器IP核

□ 存储器时序

✓ Write First Mode

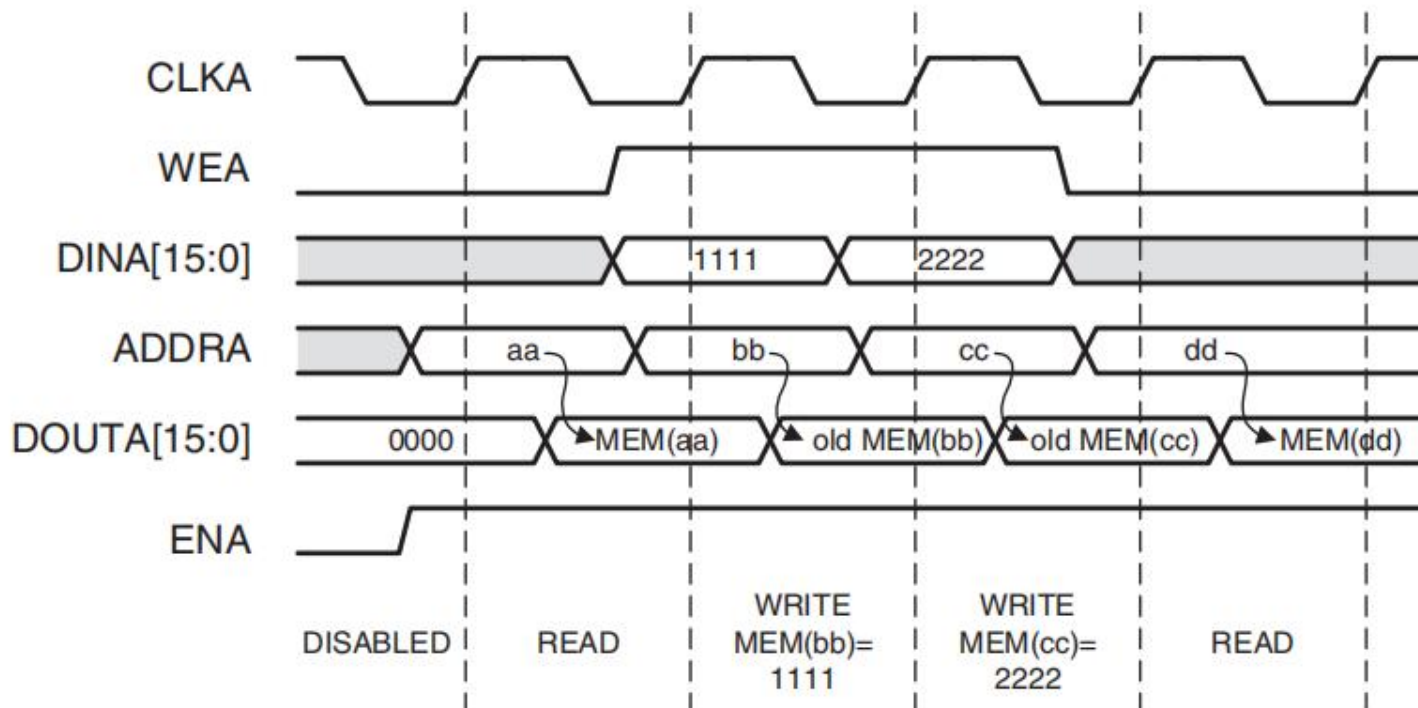


实验原理

2.存储器IP核

□ 存储器时序

✓ Read First Mode

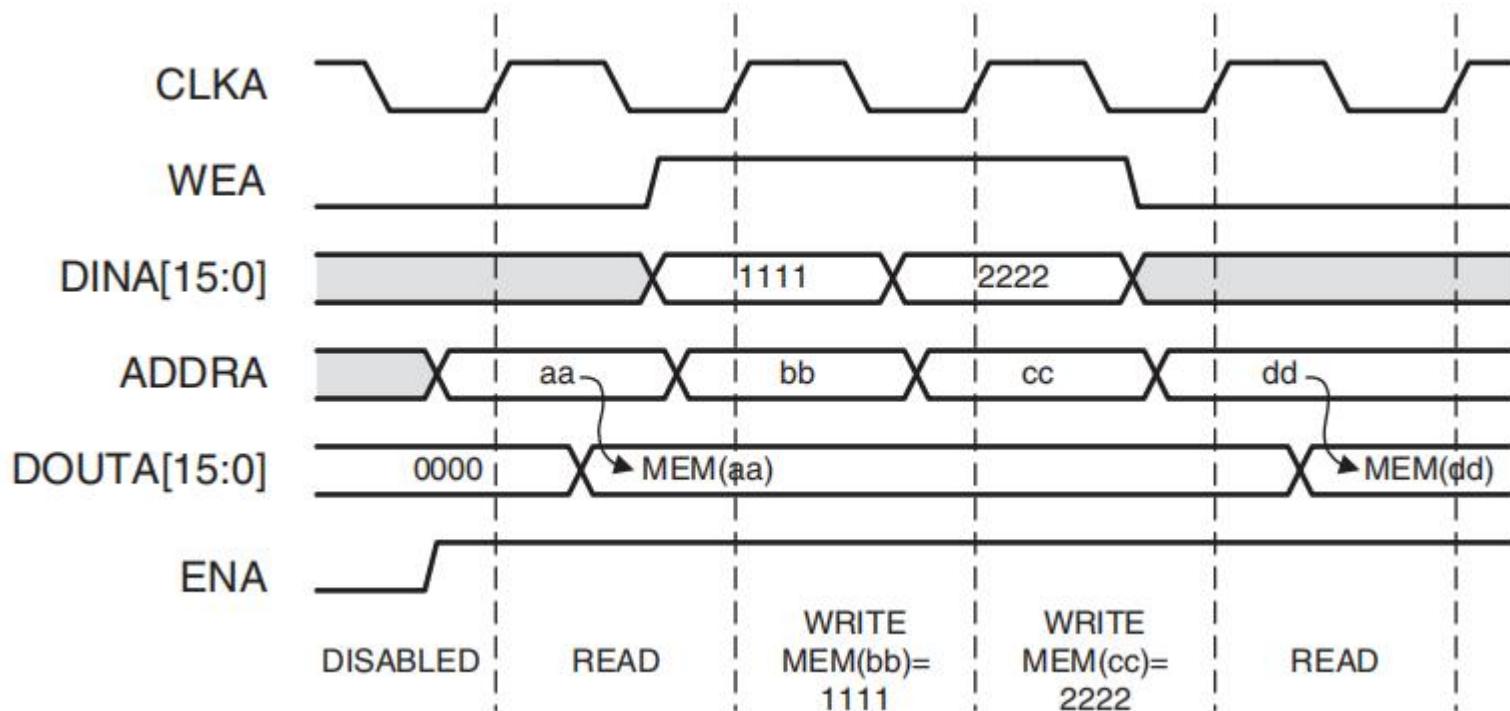


实验原理

2.存储器IP核

□ 存储器时序

✓ No change Mode



实验原理

3.FIFO

□ FIFO定义:

- ✓ First in, First out, 先进先出数据缓存器。

□ FIFO特点:

- ✓ 没有外部读写地址线，数据地址由内部读写指针自动加1完成。

□ FIFO分类:

- ✓ 分为同步FIFO和异步FIFO;
- ✓ 同步FIFO是指读时钟和写时钟为同一个时钟：常用于同步时钟的数据缓存;
- ✓ 异步FIFO是指读写时钟是互相独立的：常用于跨时钟域的数据信号的传递。

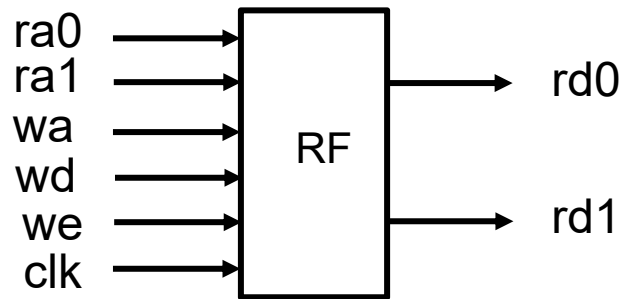
□ FIFO重要参数:

- ✓ 满信号：FIFO里面的信号数量达到了最大深度值;
- ✓ 空信号：FIFO里面的信号全部被读出;
- ✓ 读写指针：总是指向下一个地址。

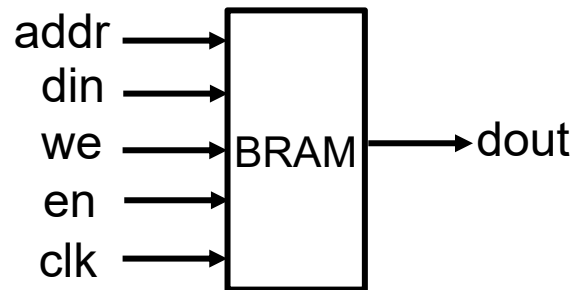
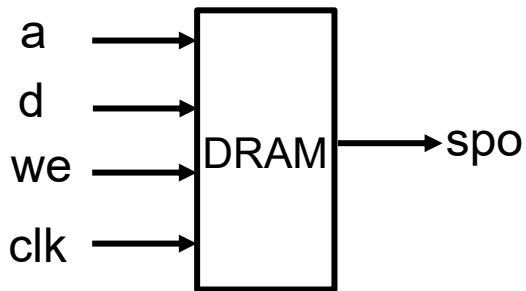


实验内容

1. 寄存器堆设计及仿真

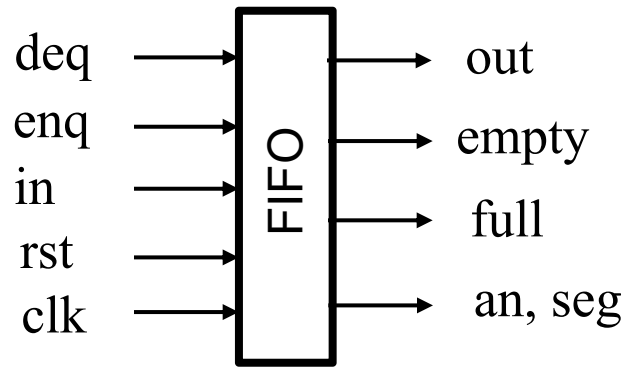


2. 存储器IP核例化及仿真



实验内容

3. 寄存器堆应用：FIFO队列



实验内容

1. 寄存器堆设计及仿真

□ 寄存器堆端口定义

```
module register_file
#(parameter WIDTH = 32)
( input clk,
  input[4 : 0] ra0,
  output[WIDTH - 1 : 0] rd0,
  input[4 : 0] ra1,
  output[WIDTH - 1 : 0] rd1,
  input[4 : 0] wa,
  input we,
  input[WIDTH - 1 : 0] wd
);

reg [WIDTH - 1 : 0] regfile[0 : 31];
assign rd0 = regfile[ra0],
       rd1 = regfile[ra1];
always @(posedge clk) begin
  if (we) regfile[wa] <= wd;
end
endmodule
```

//三端口32 x WIDTH寄存器堆
//数据宽度和存储器深度
//时钟（上升沿有效）
//读端口0地址
//读端口0数据
//读端口1地址
//读端口1数据
//写端口地址
//写使能，高电平有效
//写端口数据

□ 寄存器堆功能设计要求

- ✓ 更改上述寄存器堆代码，使得寄存器堆的0号地址恒为0.

实验内容

2. 存储器IP核例化及仿真

□ IP核生成方式

Flow Navigator >> Project Manager >> IP Catalog

✓ 途径1:

- Memories & Storage Elements >> RAMs & ROMs >> Distributed Memory Generator (分布式存储器)
- Memories & Storage Elements >> RAMs & ROMs& BRAMs >> Block Memory Generator (块式存储器)

✓ 途径2:

- Basic Elements >> Memory Elements >> Distributed Memory Generator(分布式存储器)
- Basic Elements >> Memory Elements >> Block Memory Generator(块式存储器)

实验内容

2. 存储器IP核例化及仿真

□ IP核生成方式

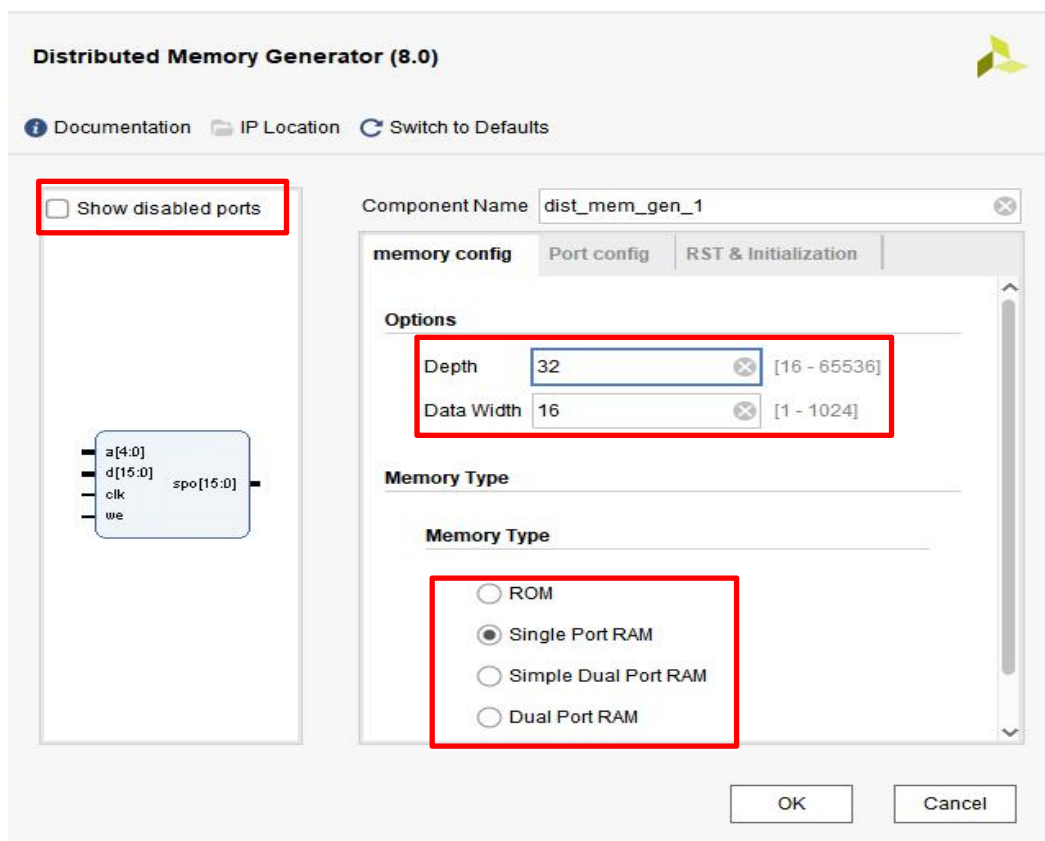
The screenshot shows the Vivado IP Catalog interface. On the left, the 'IP Catalog' tab is selected in the 'PROJECT MANAGER' section. The search bar at the top contains 'mem', resulting in 7 matches. The search results are displayed in a table with columns: Name, AXI4, Status, License, and VLNV. The results are categorized into folders: Vivado Repository, AXI Infrastructure, Basic Elements, Memory Elements, Communication & Networking, Embedded Processing, Memories & Storage Elements, RAMs & ROMs, RAMs & ROMs & BRAM, and Video & Image Processing. The 'Basic Elements' and 'Memory Elements' folders are expanded, showing 'Block Memory Generator' and 'Distributed Memory Generator' respectively. The 'RAMs & ROMs' folder is also expanded, showing 'Distributed Memory Generator' and 'Block Memory Generator'.

Name	AXI4	Status	License	VLNV
Vivado Repository				
AXI Infrastructure				
Basic Elements				
Memory Elements				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4
Distributed Memory Generator		Production	Included	xilinx.com:ip:dist_mem_gen:8.0
Communication & Networking				
Embedded Processing				
Memories & Storage Elements				
ECC		Production	Included	xilinx.com:ip:ecc:2.0
FIFOs				
Memory Interface Generators				
RAMs & ROMs				
Distributed Memory Generator		Production	Included	xilinx.com:ip:dist_mem_gen:8.0
RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4
Video & Image Processing				

实验内容

2. 存储器IP核例化及仿真

□ 分布式存储器IP核参数设置及初始化



实验内容

2. 存储器IP核例化及仿真

□ 分布式存储器IP核参数设置及初始化

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



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  

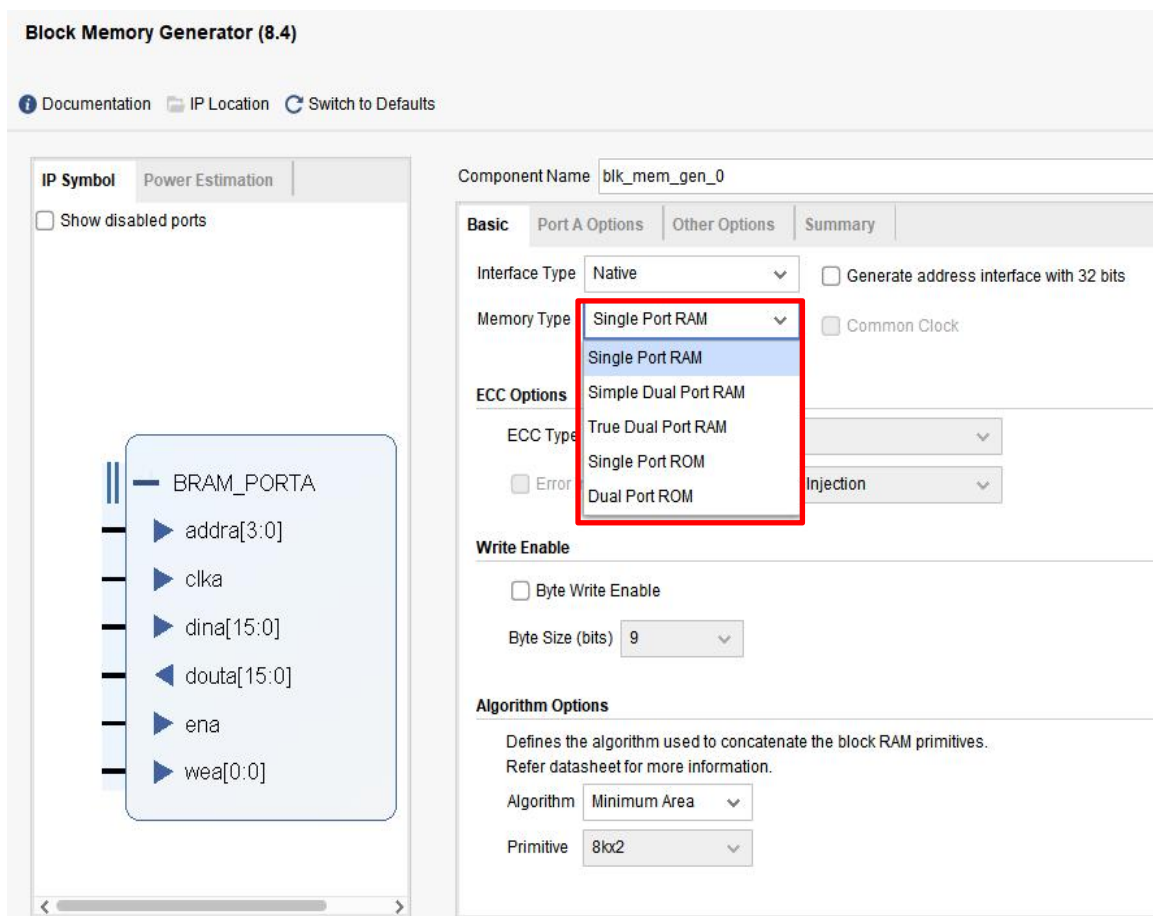
COE Options

Default Data :  Radix : 

实验内容

2. 存储器IP核例化及仿真

□ 块式存储器IP核参数设置及初始化



实验内容

2. 存储器IP核例化及仿真

□ 块式存储器IP核参数设置及初始化

Basic	Port A Options	Other Options	Summary
-------	----------------	---------------	---------

Memory Size

Write Width	16	Range: 1 to 4608 (bits)
Read Width	16	
Write Depth	16	Range: 2 to 1048576
Read Depth	16	

Operating Mode: Write First

Enable Port Type: Always Enabled

Port A Optional Output Registers

<input type="checkbox"/> Primitives Output Register	<input type="checkbox"/> Core Output Register
<input type="checkbox"/> SoftECC Input Register	<input type="checkbox"/> REGCEA Pin

Basic	Port A Options	Other Options	Summary
-------	----------------	---------------	---------

Pipeline Stages within Mux: 0 Mux Size: 1x1

Memory Initialization

☒ Load Init File

Coe File: o proj/CS EXP2023/LAB2/RAM/BRAM/BRAM/data_ram.coe

实验内容

2. 存储器IP核例化及仿真

□ IP核初始化COE文件格式

; Sample Initialization file for a 32x16 distributed ROM (注释行)

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;

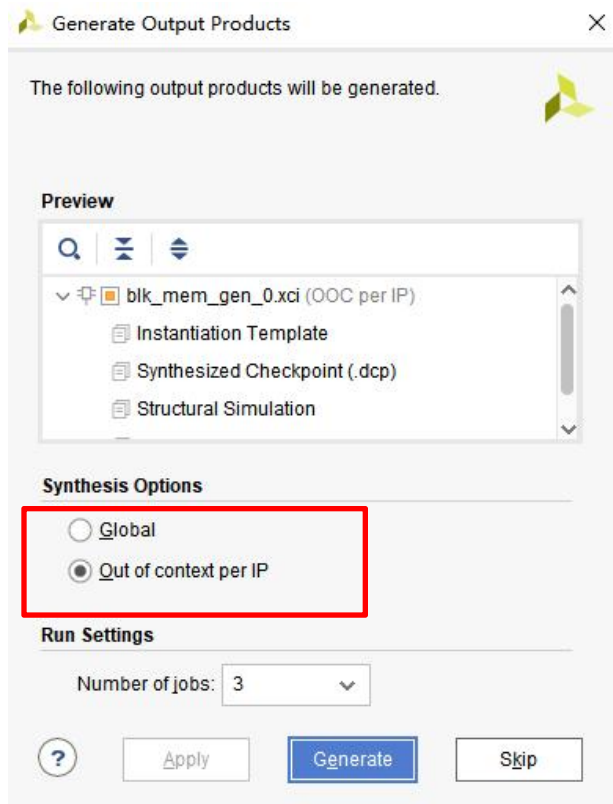
逗号或空格分隔每项数据
数据不一定是4位

最后用;结尾

实验内容

2. 存储器IP核例化及仿真

□ IP核综合方式



✓ Global模式:

- 全局综合：每次工程综合时，IP核源码都会被综合；
- 不会产生.dcp文件；
- 综合的时间长。

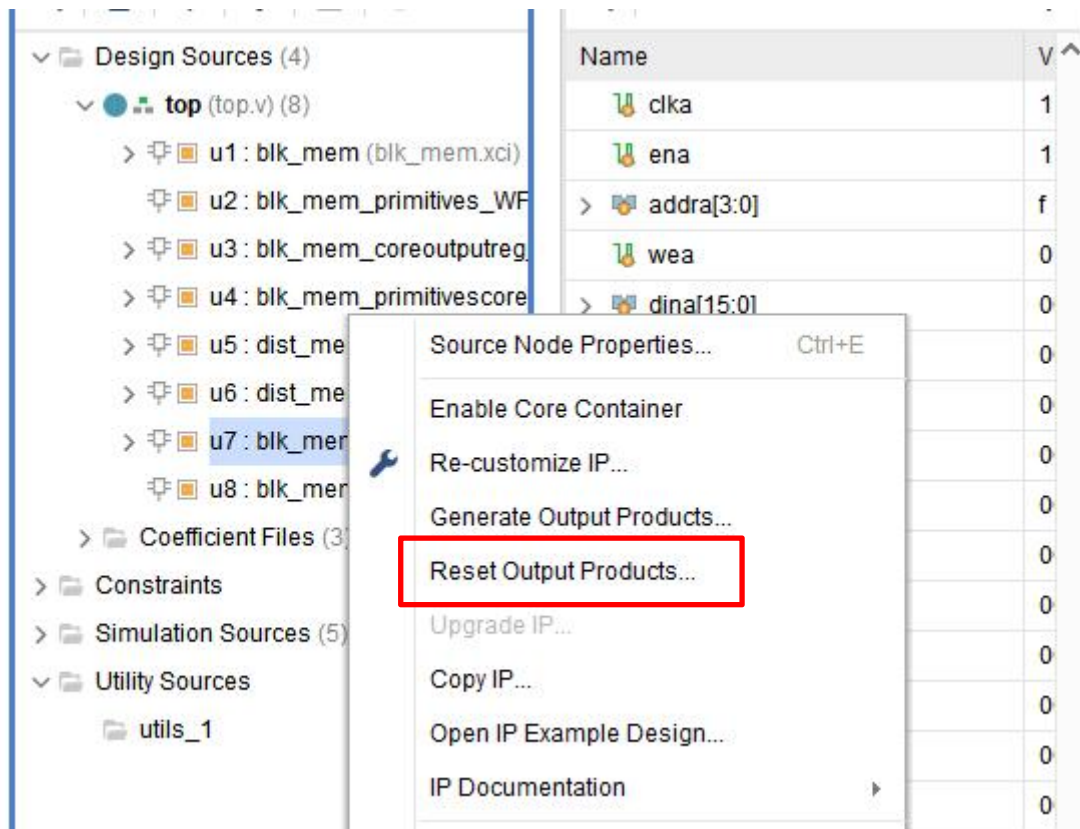
✓ OOC模式:

- 对IP进行单独综合，生成.dcp文件；
- 工程用到IP时，只需从.dcp文件中解析出对应IP的网表文件，而不再对IP核的源文件重新综合；
- 可以加快综合的速度；
- 生成IP核后，如果还需要对IP的参数或初始值进行修改，则需要先对IP核进行复位，然后重新选择OOC模式生成IP核。

实验内容

2. 存储器IP核例化及仿真

□ IP核综合方式



实验内容

2. 存储器IP核例化及仿真

□ 实例化IP核

Project Manager – display >> Sources >> IP Sources

– IP >> dist_mem_gen_0 >> Instantiation Template >> dist_mem_gen_0.vco

```
dist_mem_gen_0 your_instance_name (  
    .a(a),           // input wire [3 : 0] a  
    .d(d),           // input wire [7 : 0] d  
    .clk(clk),       // input wire clk  
    .we(we),         // input wire we  
    .spo(spo)        // output wire [7 : 0] spo  
);
```

实验内容

2. 存储器IP核例化及仿真

□ 实例化IP核

The screenshot displays the Vivado IDE interface. On the left, the 'Sources' window shows a project hierarchy. Under 'IP (1)', the 'dist_mem_gen_0' core is expanded, showing its 'Instantiation Template (2)'. The files 'dist_mem_gen_0.vho' and 'dist_mem_gen_0.veo' are listed, with 'dist_mem_gen_0.veo' highlighted by a red box. The bottom tab bar shows 'IP Sources' selected, also highlighted by a red box. On the right, the 'dist_mem_gen_0.veo' file is open, showing Verilog code. A red box highlights the instantiation block starting at line 57: `dist_mem_gen_0 your_instance_name (`. The code includes comments for input wires `a`, `d`, `clk`, `we`, and output wire `spo`. The file path at the top is `d:/Project_zyn/Vivado/2022/project_lab2_fifo/project_lab2_fifo.srcs/sources`.

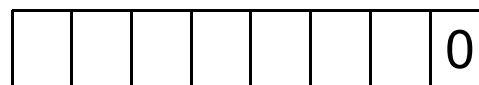
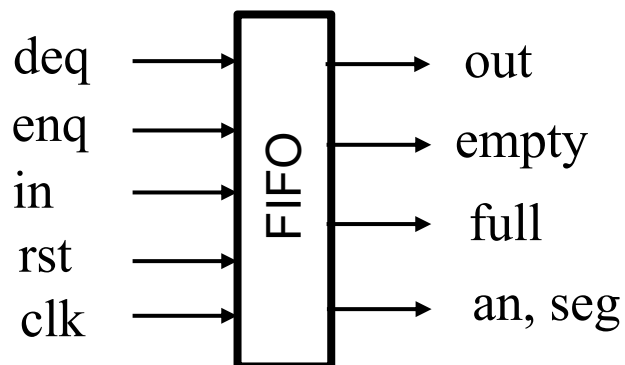
```
46 //
47 // DO NOT MODIFY THIS FILE.
48
49 // IP VLNV: xilinx.com:ip:dist_mem_gen:8.0
50 // IP Revision: 13
51
52 // The following must be inserted into your Verilog file for
53 // core to be instantiated. Change the instance name and port
54 // (in parentheses) to your own signal names.
55
56 //----- Begin Cut here for INSTANTIATION Template -----//
57 dist_mem_gen_0 your_instance_name (
58   .a(a),      // input wire [3 : 0] a
59   .d(d),      // input wire [7 : 0] d
60   .clk(clk),  // input wire clk
61   .we(we),    // input wire we
62   .spo(spo)   // output wire [7 : 0] spo
63 );
64 // INST_TAG_END ----- End INSTANTIATION Template -----
65
```

实验内容

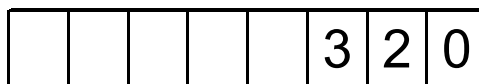
3. 寄存器堆应用：FIFO队列

□ 功能要求

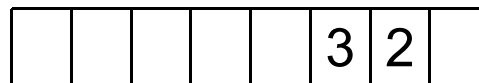
- ✓ 用三端口8×4寄存器堆实现最大长度为8的FIFO队列
- ✓ deq,enq: 出/入队列使能 (互斥), 高电平有效, 一次有效仅允许操作一项数据
- ✓ out, in: 出/入队列数据
- ✓ full, emp: 队列满/空, 满/空时无法进行入/出队操作
- ✓ an, seg: 数码管控制信号, 显示队列数据和出入状态
- ✓ clk, rst: 时钟, 复位



复位(数码管显示0)



数据1, 2, 3依次入队列,
数据1无法入队



数据0出队列



队列空

实验内容

3. 寄存器堆应用：FIFO队列

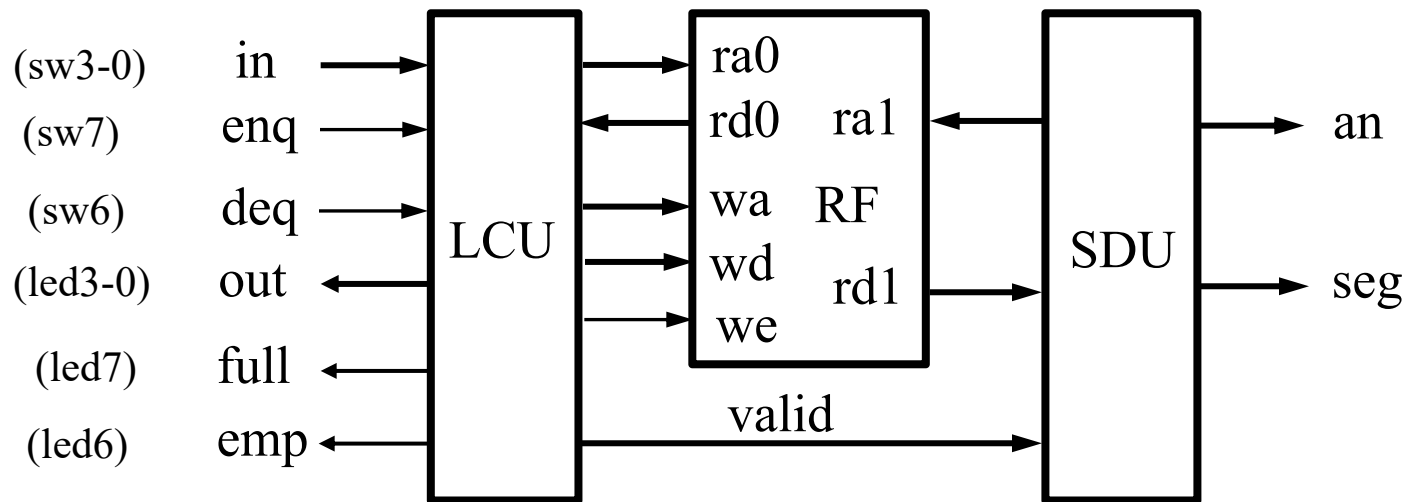
□ 逻辑结构

✓ 队列控制单元 LCU (List Control Unit)

- 处理出/入队操作，显示队列空/满状态，读出数据

✓ 数码管显示单元 SDU (Segment Display Unit)

- 显示队列数据和出入状态



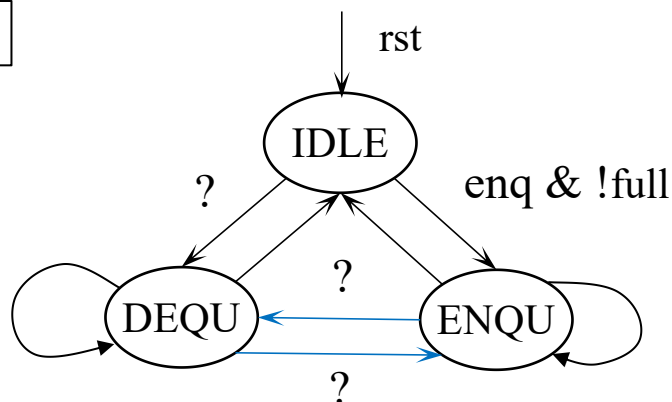
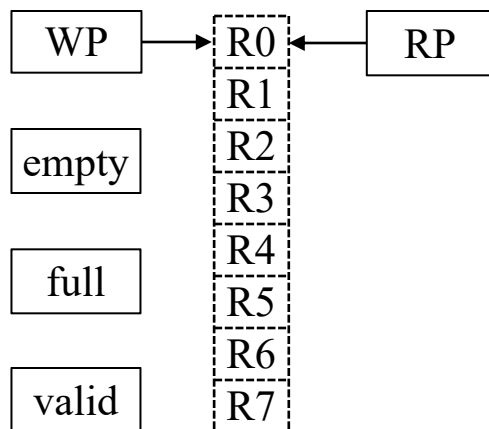
* 省略了clk (100MHz) 和 rst (button)

实验内容

3. 寄存器堆应用：FIFO队列

□ 控制单元LCU设计

- ✓ RP: 3位, 读指针, **总是指向下一个时钟要读的地址**, 指针可循环移动;
- ✓ WP: 3位, 写指针, **总是指向下一个时钟要写的地址**, 指针可循环移动;
- ✓ full, empty: 各1位, 满和空标志, 满或空时, 为“1”;
- ✓ valid: 8位, 数据有效标志: “1” 数据有效 (入队), “0” 数据无效 (出队);
valid[i]对应R[i]是否有效。



ENQU:

$RF[WP] \leftarrow in$ (RF写使能we),
 $WP \leftarrow WP + 1$,
 $Valid[WP] \leftarrow 1$,
 $full \leftarrow ((WP + 1) == RP)$,
 $empty \leftarrow 0$.

DEQU:

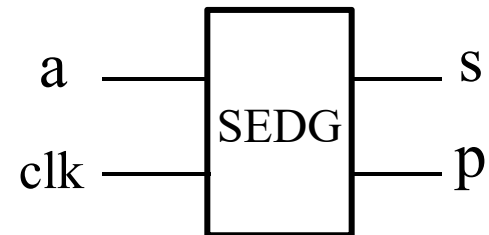
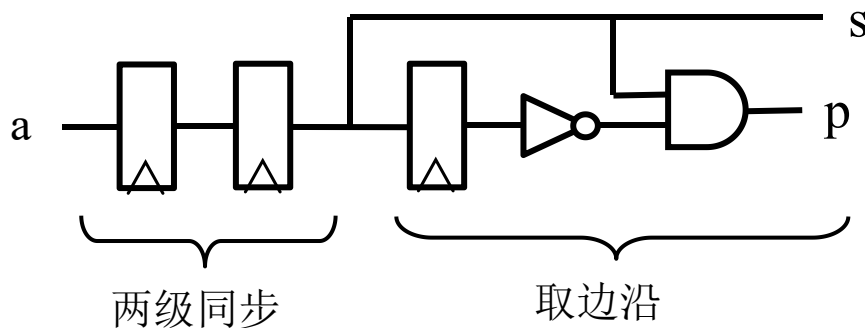
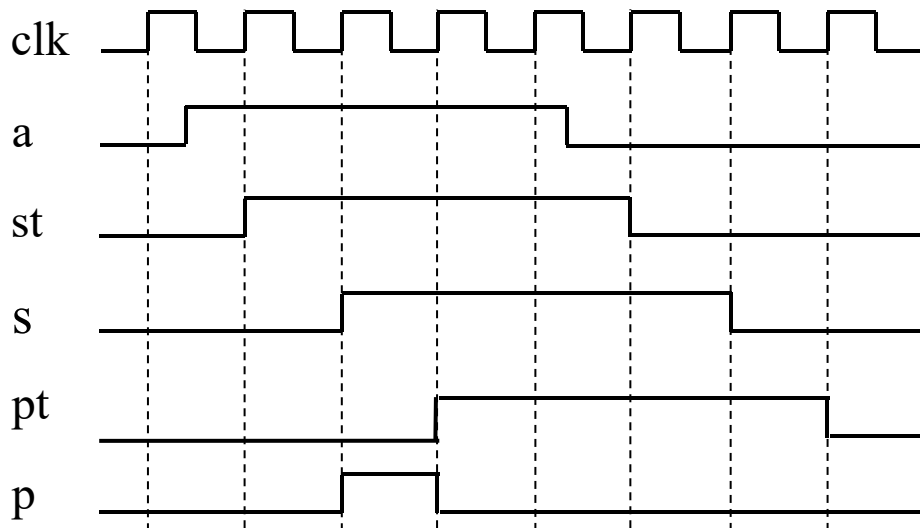
IDLE:

*读写指针初始都位于第一个存储器位置, 且FIFO队列为空。

实验内容

3. 寄存器堆应用：FIFO队列

□ 出/入队使能信号处理-异步信号同步和取边沿



组合逻辑

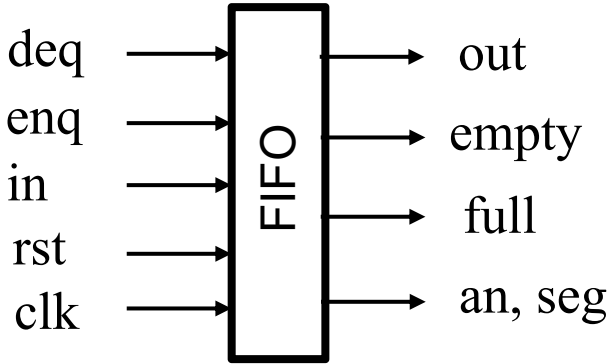
`assign p= s && (~pt)`

实验内容

3. 寄存器堆应用：FIFO队列

□ FIFO队列端口定义

```
module fifo (  
    input clk, rst,           //时钟（上升沿有效）、同步复位（高电平有效）  
    input enq,                //入队列使能，高电平有效  
    input [3:0] in,           //入队列数据  
    input deq,                //出队列使能，高电平有效  
    output [3:0] out,          //出队列数据  
    output full,empty,        //队列满和空标志  
    output [2:0] an,           //数码管选择  
    output [3:0] seg          //数码管数据  
);
```



FIFO队列外设端口分配表

端口	外设
clk	100MHz
rst	button
enq	sw7
in	sw3-0
deq	sw6
out	led3-0
full	led7
empty	led6
an	an
seg	seg

实验要求[必做]

1. 寄存器堆设计及仿真

- 采用行为方式参数化设计32 x WIDTH寄存器堆 (X0恒等于0) ;
- 完成寄存器堆功能仿真。

2. 存储器IP核例化及仿真

- 顶层模块分别IP例化分布式和块式16 x 8位单端口RAM，完成功能仿真，并对比两者之间的时序差异

- ✓ 分布式RAM参数设置:

- Input Options: non register;
- Output Options: non register.

- ✓ 块式RAM参数设置:

- Operationg Mode: write first;
- Enable Port Type: always enabled;
- Optional Output Registers: 不勾选.

实验要求[必做]

2. 存储器IP核例化及仿真

□ 顶层模块分别IP例化块式16 x 8位单端口RAM三种不同的操作模式 (operating mode) , 通过功能仿真对比时序差异

✓ 块式RAM参数设置:

- Operationg Mode: 1) write first; 2) read first; 3) no change.

- Enable Port Type: always enabled;

- Optional Output Registers: 不勾选.

实验要求[必做]

3. 寄存器堆应用：FIFO队列

- 设计LCU模块的数据通路和控制器：
画出数据通路及状态机， Moore型FSM描述控制器；
- 设计SDU模块；
- 结构化方式描述顶层模块FIFO：实例化LCU， SDU， RF模块；
- 对整个工程进行功能仿真；
- 将FIFO队列电路下载至FPGAOL中测试。

实验要求[选做]

1. 存储器IP核例化及仿真

□ 仿真比较16 x 8位块式单端口RAM输出寄存器不同设置的时序差异

✓ 块式RAM参数设置:

- Operation Mode: write first;
- Enable Port Type: always enabled;
- Optional Output Registers:

1) 勾选Primitives Output Register

2) 勾选Core Output Register

3) 同时勾选Primitives Output Register和Core Output Register

□ 例化分布式存储器，将结果显示到LED

✓ 在FIFO队列中，开关sw4尚未使用，现添加功能如下:

- 开关sw4关闭时，FIFO队列功能如前，此时(led3-0)显示出队列的数据out;
- 开关sw4开启时，将FIFO队列头部的元素作为读地址，输入例化的单端口分布式存储器，将存储器读出的内容显示到(led3-0)。

实验要求[选做]

2. 寄存器堆应用：FIFO队列

□ 将队列的头部元素始终显示在数码管的最右侧



复位(数码管显示0)



数据1, 2, 3依次入队列, 数据1无法入队



数据0出队列



队列空



中国科学技术大学
University of Science and Technology of China

The End