实验一 运算器与存储器

2024春

zjx@ustc.edu.cn

实验目标

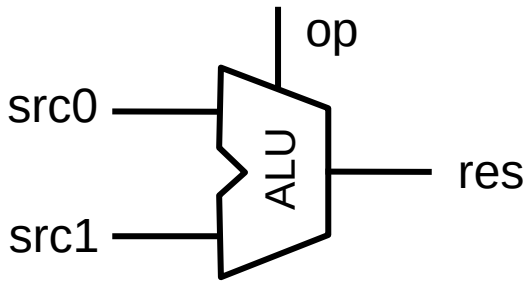
- 熟练掌握算术逻辑单元 (ALU) 、寄存器堆 (RF)和存储器的功能、时序及其应用
- 掌握数据通路和控制器的设计和描述方法
- 了解查看电路性能和资源使用情况

实验内容

- **设计算术逻辑单元(Arithmetic and Logic Unit, ALU)和寄存器堆(Register File, RF)**
- **比较分布式和块式存储器的特性**
- **利用ALU和存储器，设计排序器(Sorter, SRT)**

算术逻辑单元

- **src0, src1：两操作数，32位**
- **op：运算功能，4位**
- **res：运算结果，32位**



op	res	功能说明
0	src0 + src1	加
1	src0 - src1	减
2	src0 $<_s$ src1	有符号小于
3	src0 $<_u$ src1	无符号小于
4	src0 & src1	与
5	src0 | src1	或
6	~(src0 | src1)	或非
7	src0 ^ src1	异或
8	src0 << src1[4:0]	左移
9	src0 >> src1[4:0]	逻辑右移
10	src0 >>> src1[4:0]	算术右移
11	src1	赋值

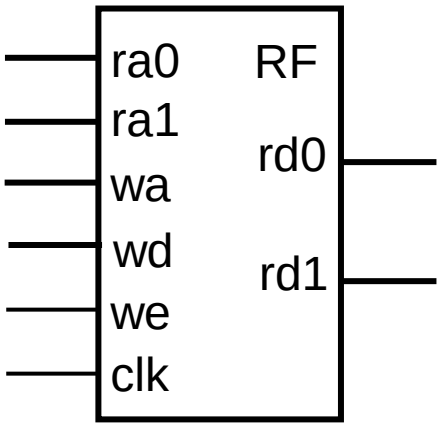
```
module alu (
    input [31:0] src0,    //操作数0
    input [31:0] src1,    //操作数1
    input [3:0] op,       //运算功能
    output [31:0] res     //运算结果
);
```

寄存器堆

```
module  rf (
    input  clk,                  //时钟
    input [4:0]  ra0, ra1,       //读地址
    output [31:0]  rd0, rd1,     //读数据
    input [4:0]  wa,             //写地址
    input [31:0]  wd,            //写数据
    input we                     //写使能
);
     reg [31:0]  x[0:31];        //寄存器堆

     assign rd0 = x[ra0];        //读操作
     assign rd1 = x[ra1];

     always  @(posedeg  clk)
        if (we)  x[wa]  <=  wd;   //写操作
endmodule
```



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

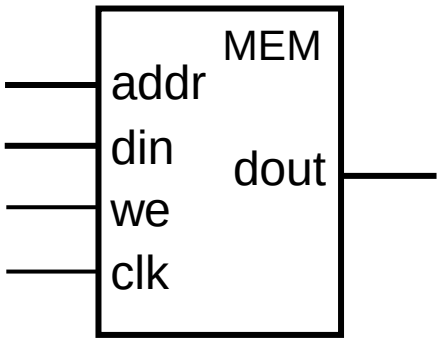
存储器

```
module mem #(
    parameter DATA_WIDTH = 32,
              ADDR_WIDTH = 10,
              INIT_FILE = ""
)(
    input clk,                                  // clock
    input [ADDR_WIDTH-1:0] addr,                // address
    input [DATA_WIDTH-1:0] din,                 // data input
    input we,                                   // write enable
    output reg [DATA_WIDTH-1:0] dout            // data output
);
    reg [DATA_WIDTH-1:0] ram [0: (1 << ADDR_WIDTH) - 1];

    initial $readmemh(INIT_FILE, ram);    // initialize memory

    always @(posedge clk) begin
        dout <= ram[addr];
        if (we) ram[addr] <= din;
    end

endmodule
```



- Flow Navigator >> Project Manager >> Language Templates
  - Verilog >> Synthesis Constructs >> Coding Examples / Example Modules >> RAM

存储器IP核

- **Vivado中有存储器IP核可以直接例化使用**
- **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**

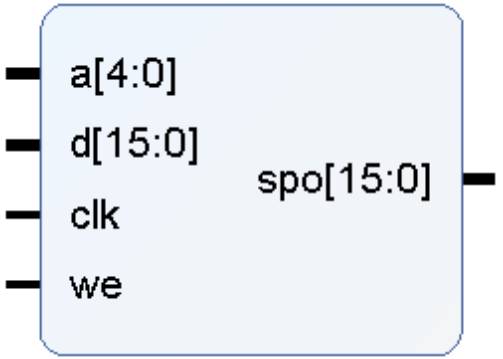
存储器IP核例化

- **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

例如，单端口分布式存储器

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 – xxxx >> 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
);
```

例化模板

COE文件格式

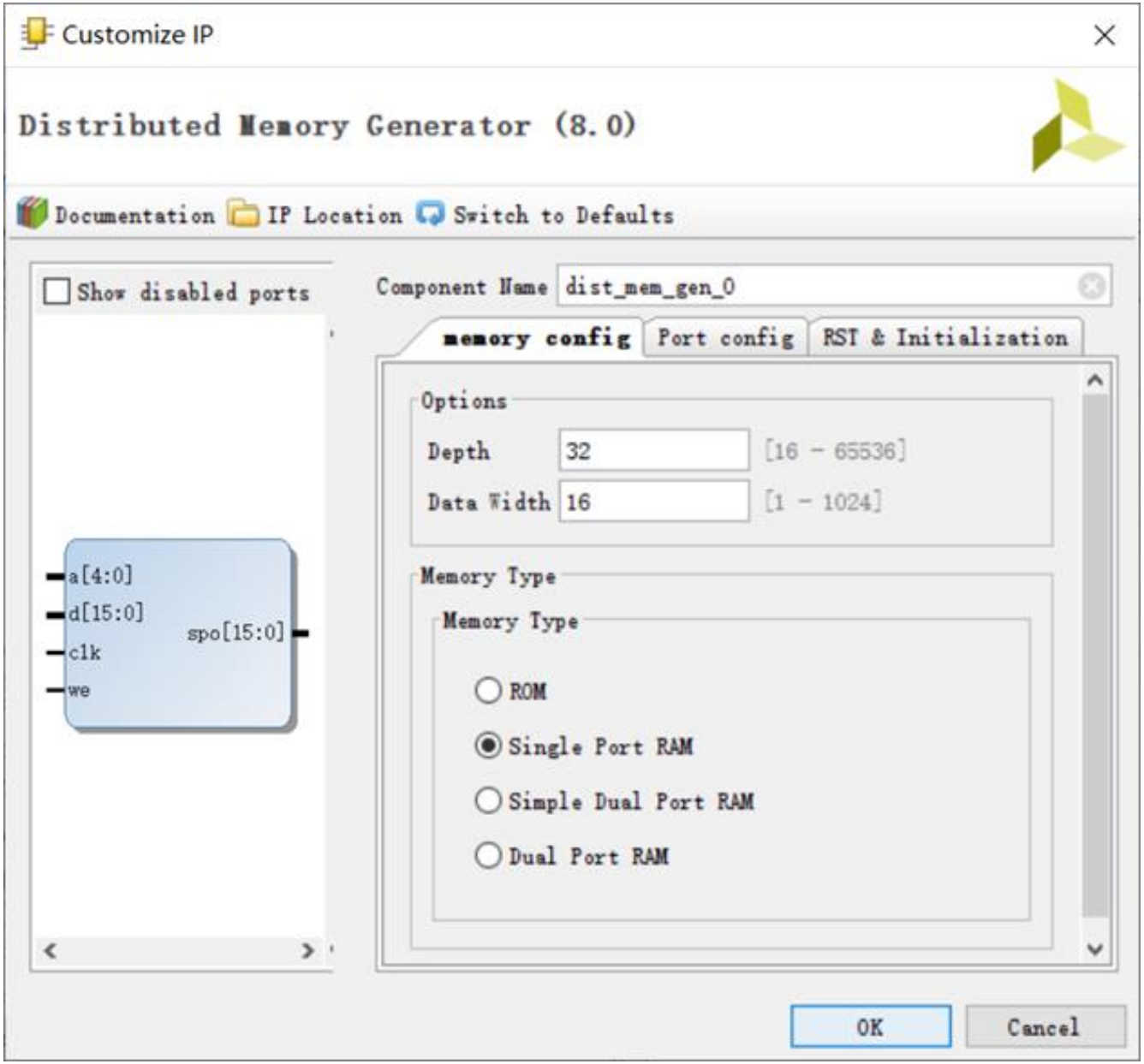
- **An example COE file:**

```
; 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  1  1  A  0
23f4  721  11ff  ABe1  1  1  A  0;
```

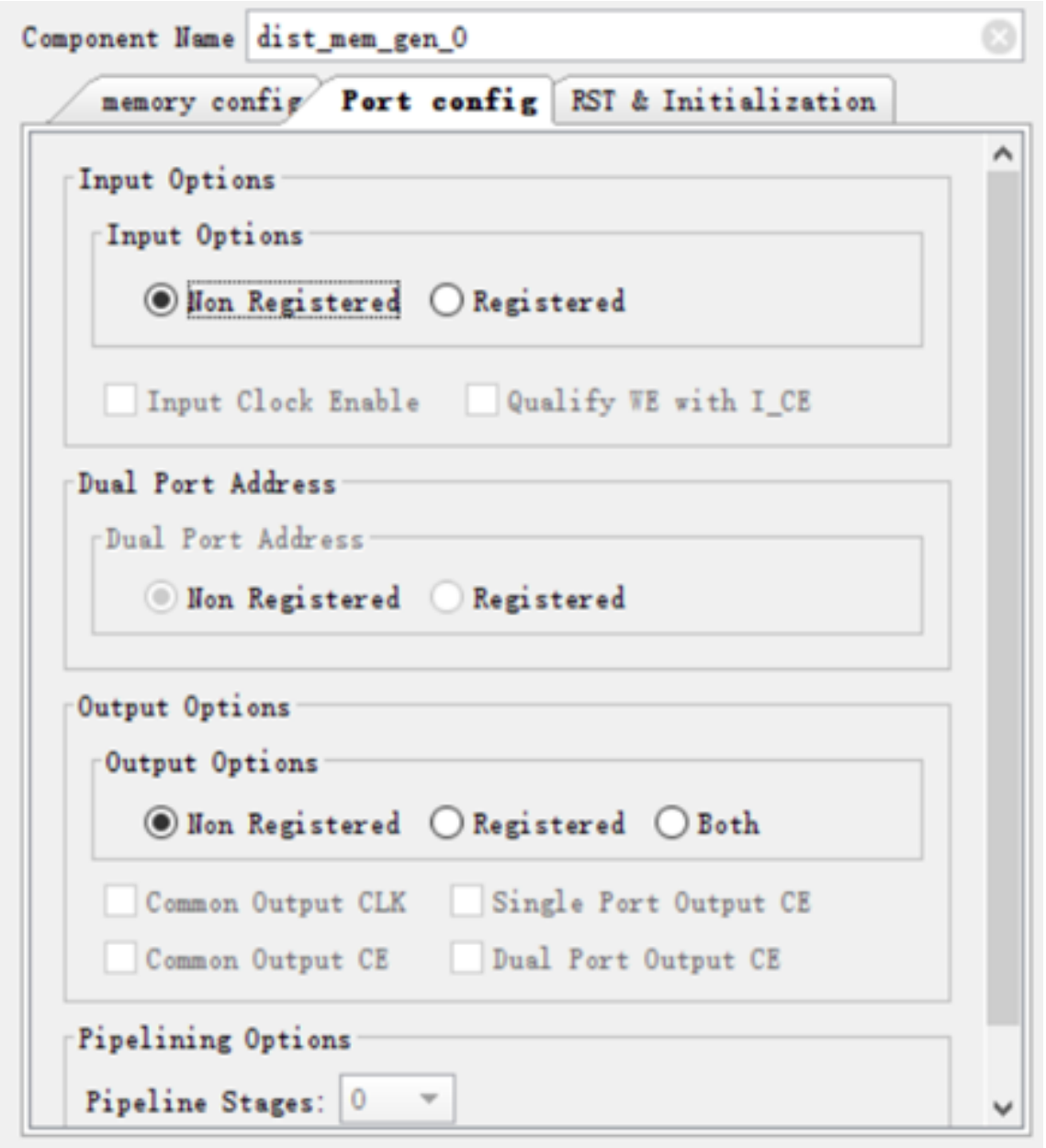
逗号或空格分隔每项数据

（不允许为负数）

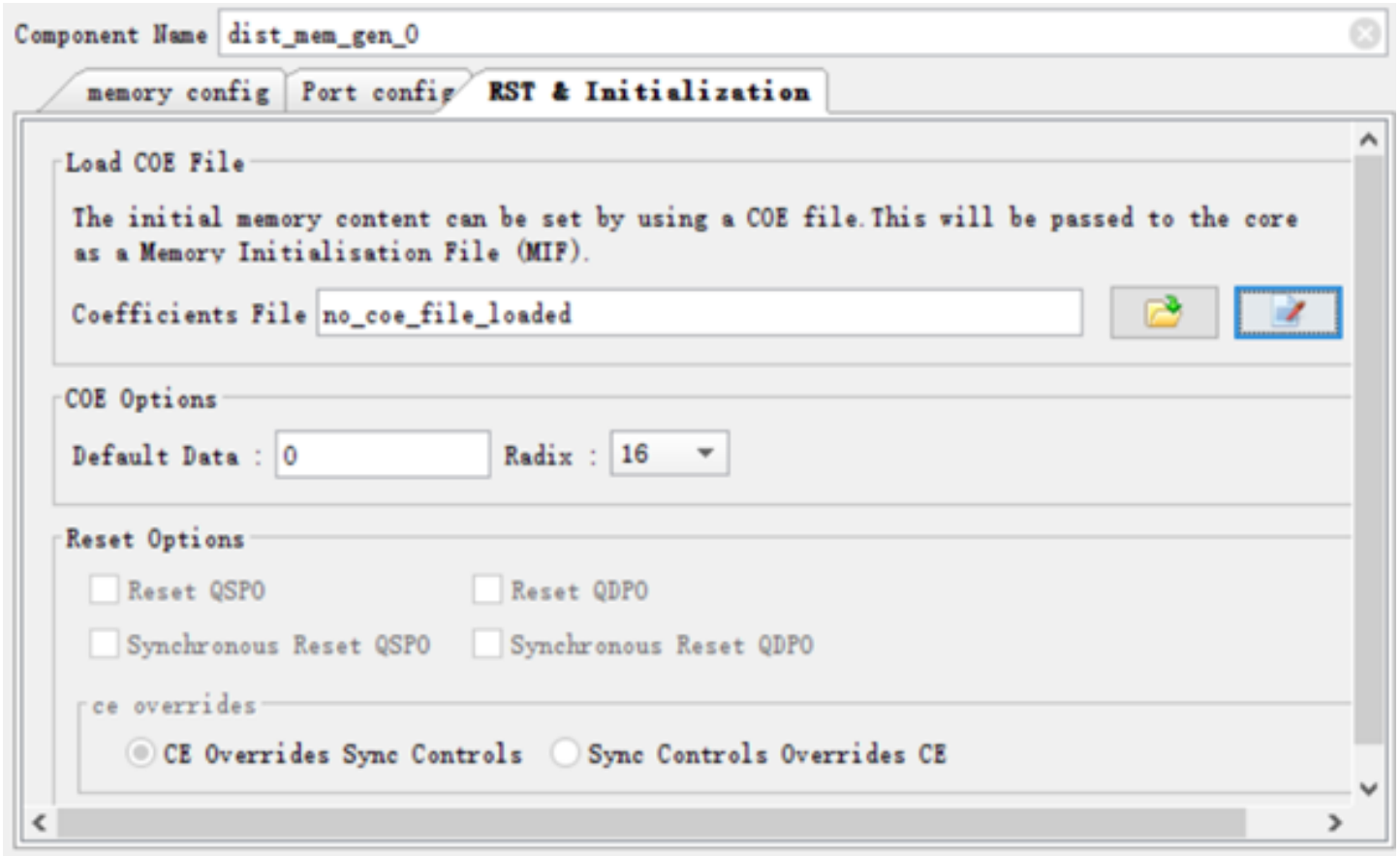
分布式存储器IP


Customize IP
Distributed Memory Generator (8.0)
Documentation
IP Location
Switch to Defaults
Show disabled ports
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

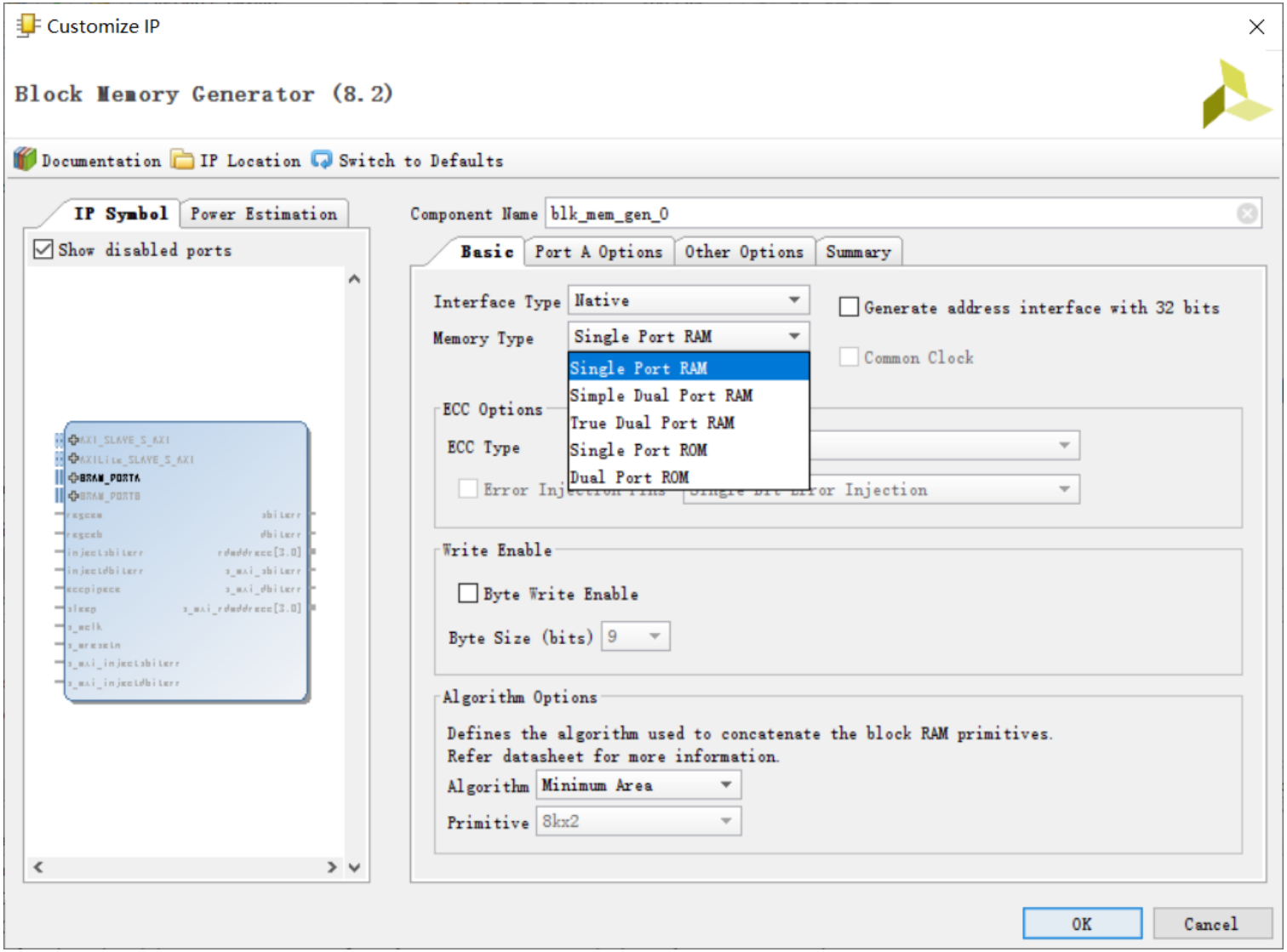
分布式存储器IP


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

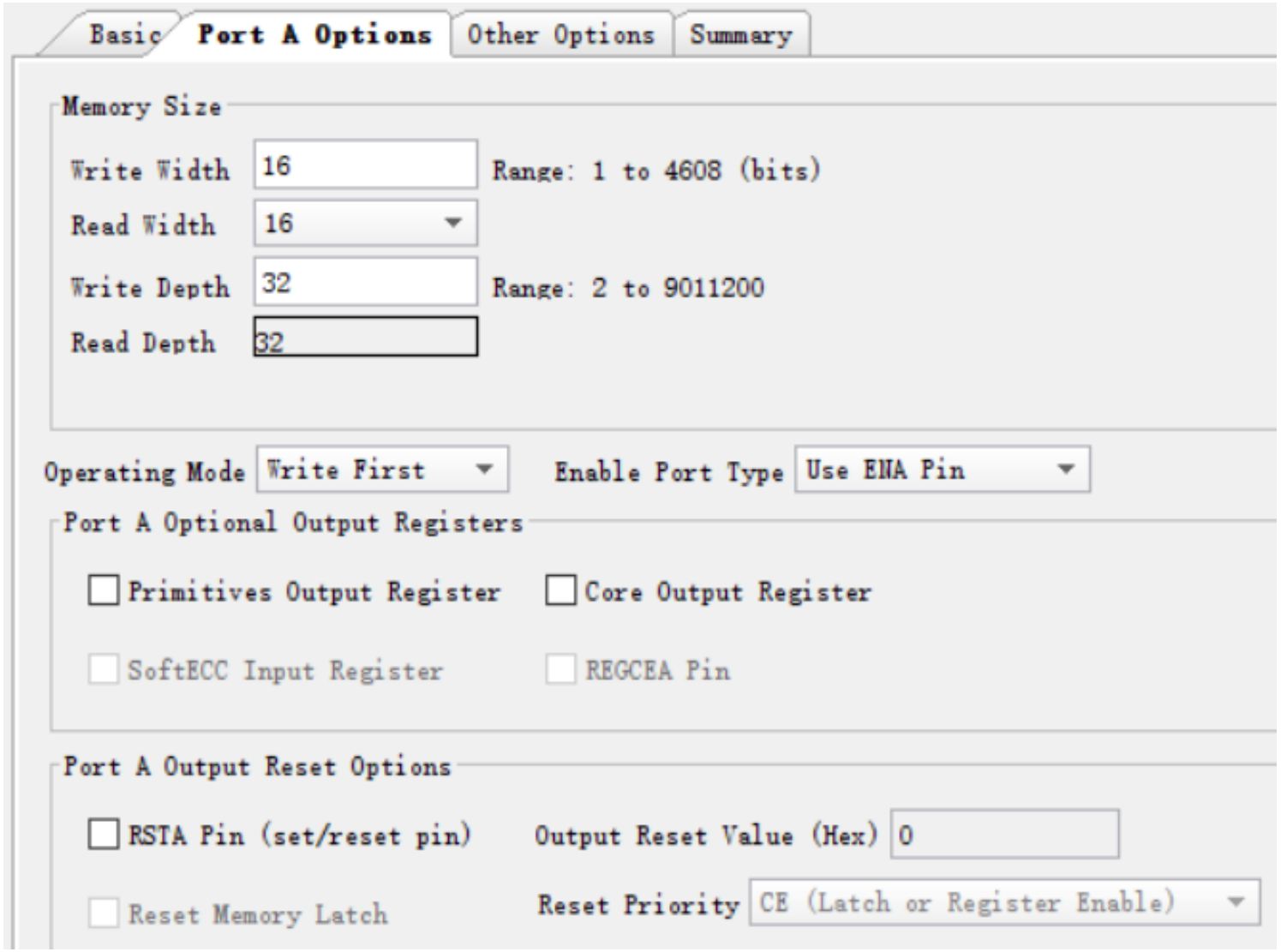
分布式存储器IP


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

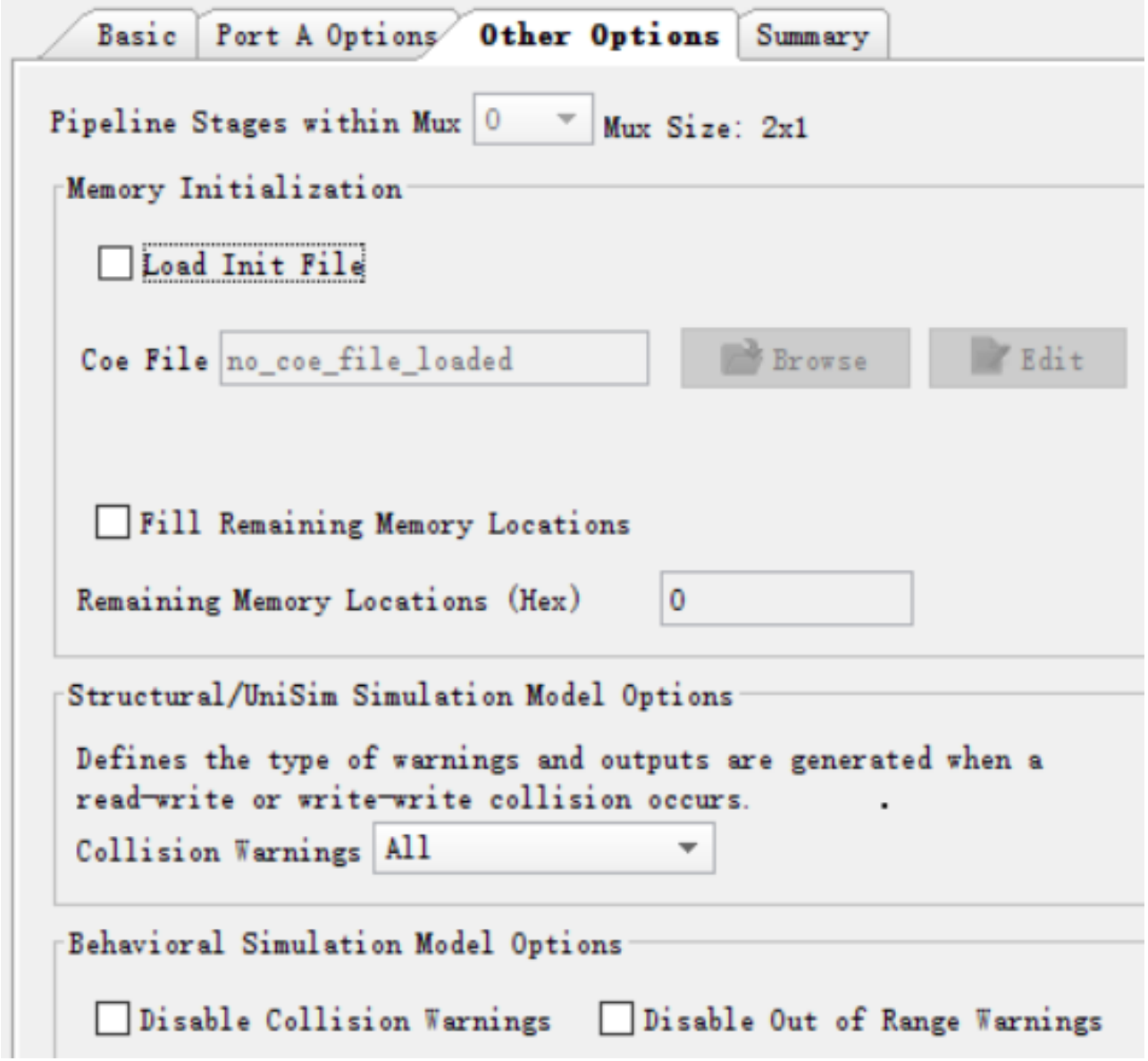
块式存储器IP


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

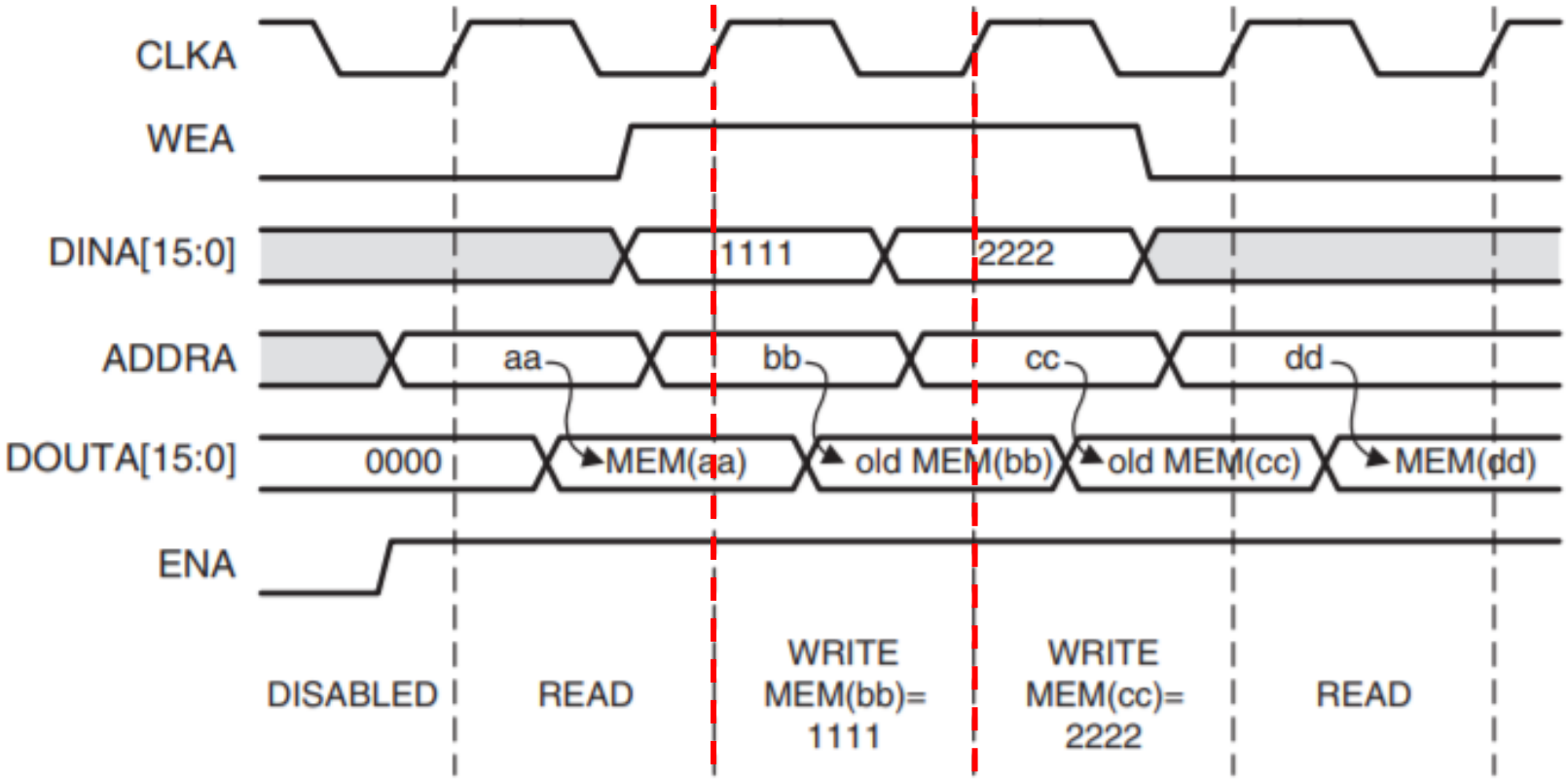

Basic
Port A Options
Other Options
Summary
Memory Size
Write Width
16
Range: 1 to 4608 (bits)
Read Width
16
Write Depth
32
Range: 2 to 9011200
Read Depth
32
Operating Mode
Write First
Enable Port Type
Use ENA Pin
Port A Optional Output Registers
Primitives Output Register
Core Output Register
SoftECC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

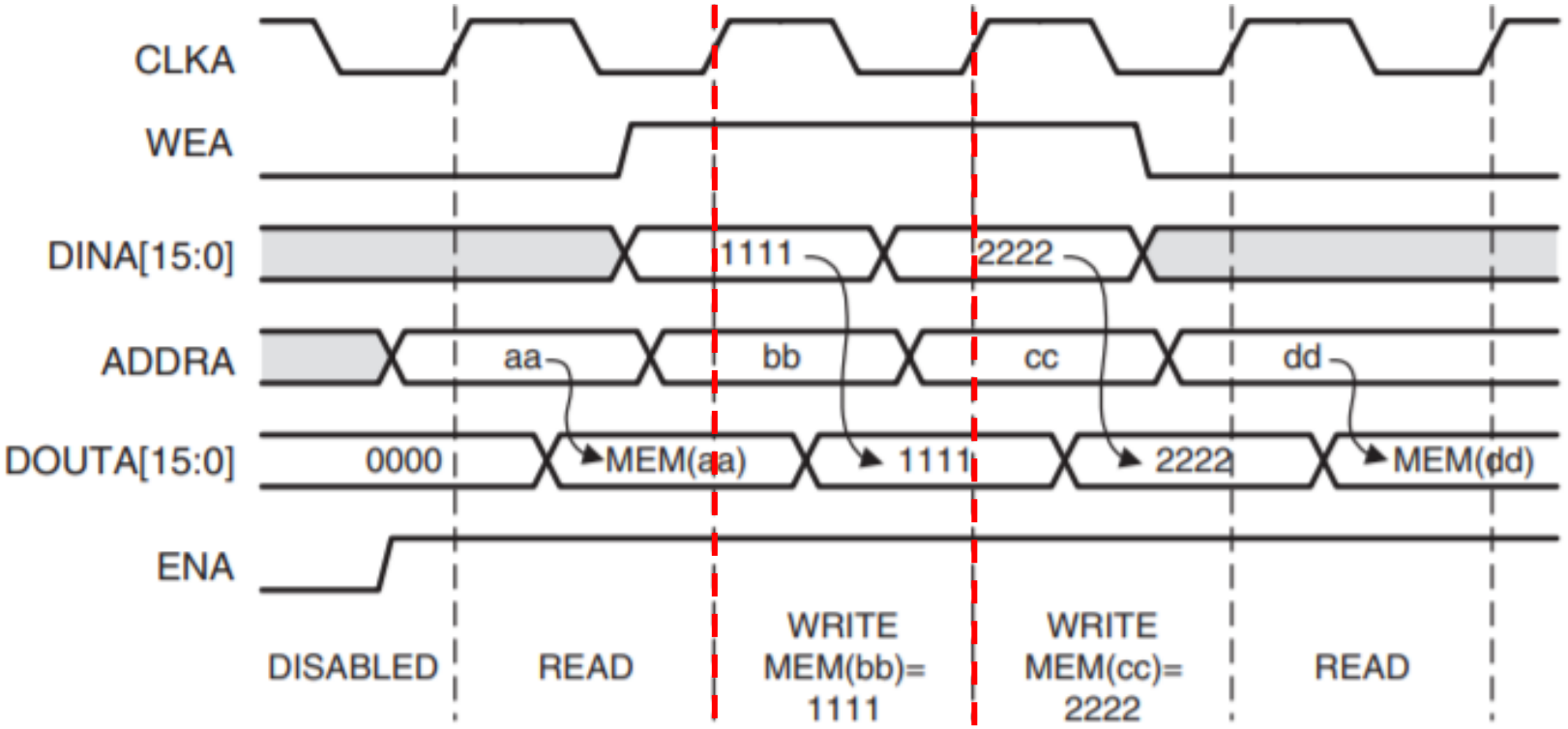
存储器时序

- **Read First Mode**



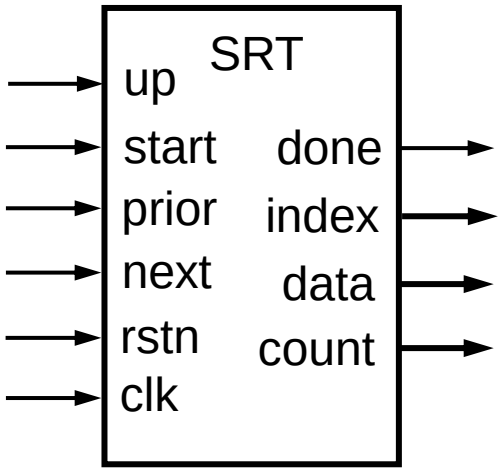
存储器时序 (续)

- **Write First Mode**



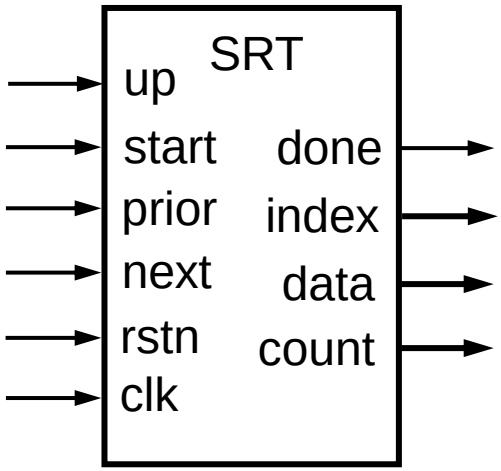
排序器

- **排序存储器中数据，记录排序所耗时间，查看排序结果**
  - up：0 -- 降序，1 -- 升序
  - start：启动排序
  - prior：查看前一个数据
  - next：查看后一个数据
  - done：排序结束标志
  - index：输出数据序号
  - data：输出数据
  - count：时钟周期数
  - clk, rstn：时钟，复位



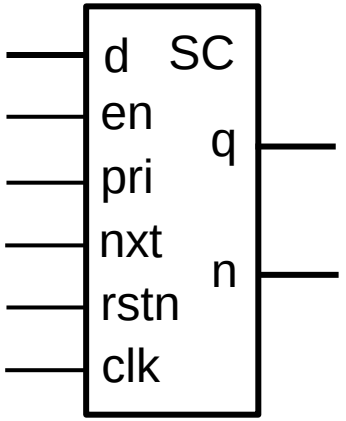
排序器 (续)

- 工作过程
  - 复位后，done =1，index = count =0
  - 启动排序后，done = 0，count递增计数；
  - 排序结束后，done = 1，count停止计数；
  - 通过prior或next，依次查看数据
- 实现要求
  - 存储器类型：分布式单端口
  - 存储器容量：1024*32位
  - 数据类型：32位无符号数
  - 排序算法：冒泡排序
  - 用ALU实现待排序数据的比较



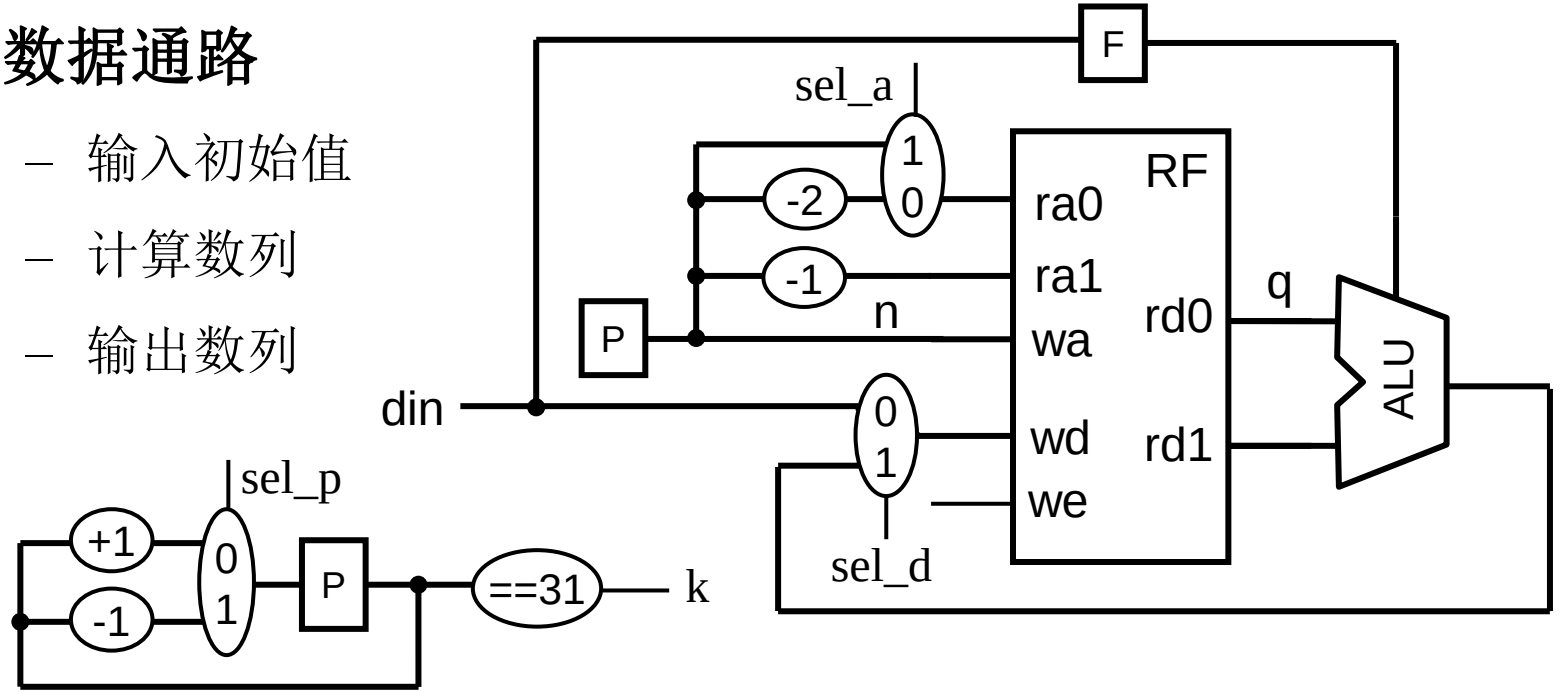
示例：数列计算器

- **利用ALU和RF，设计数列计算器(Series Calculator, SC)**
- **工作过程**
  - 输入运算符(f)和初始项(q_1, q_2), 存入RF中x0 ~ x2
    - 通过d和en依次分时输入
  - 计算数列后续项，存入RF中x3 ~ x31
    - $q_n = q_{n-2}$ ⓕ q_{n-1}, n = 3 ~ 31
    - f 运算功能由ALU定义
  - 输出RF中内容
    - q, n：输出数据和序号
    - pri, nxt：控制输出顺序

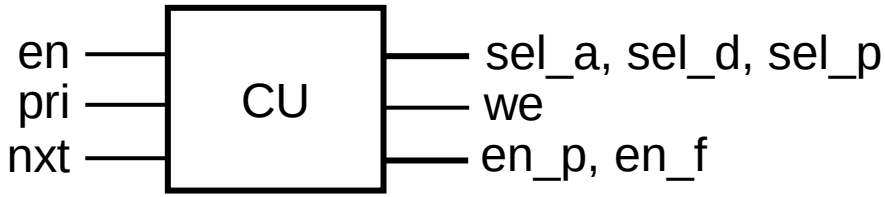


示例：数列计算器 (续1)

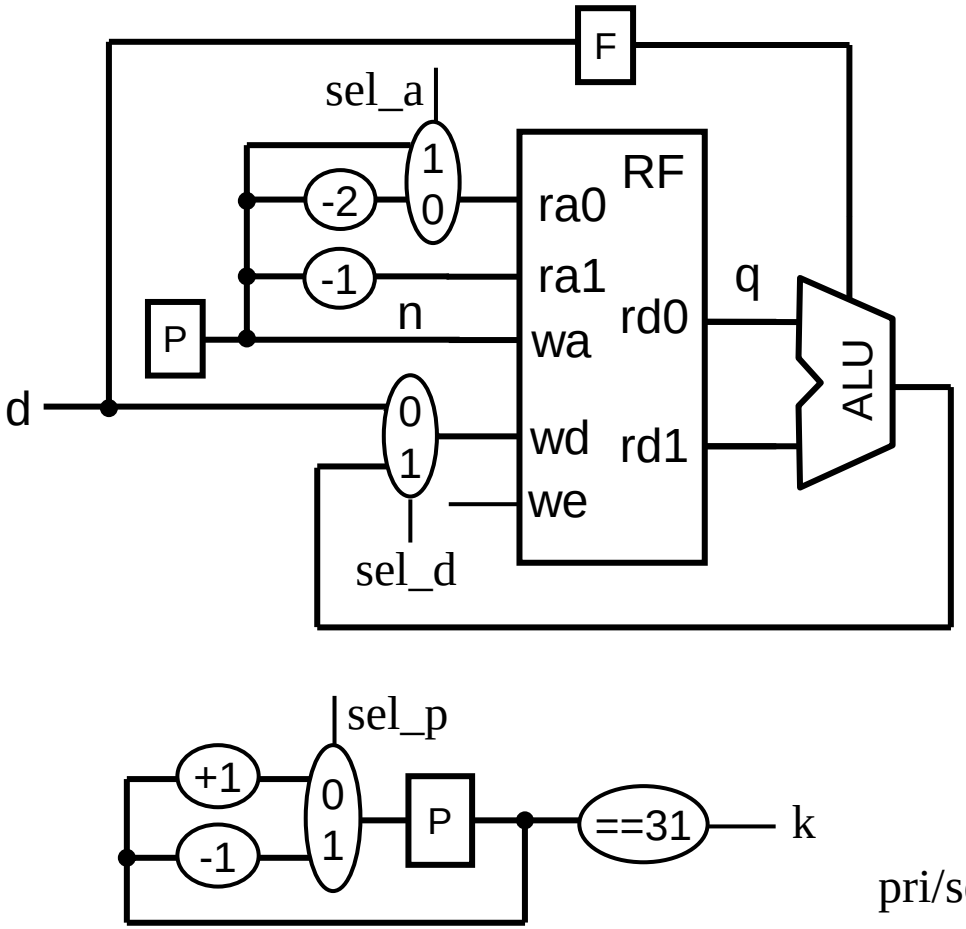
- **数据通路**
  - 输入初始值
  - 计算数列
  - 输出数列

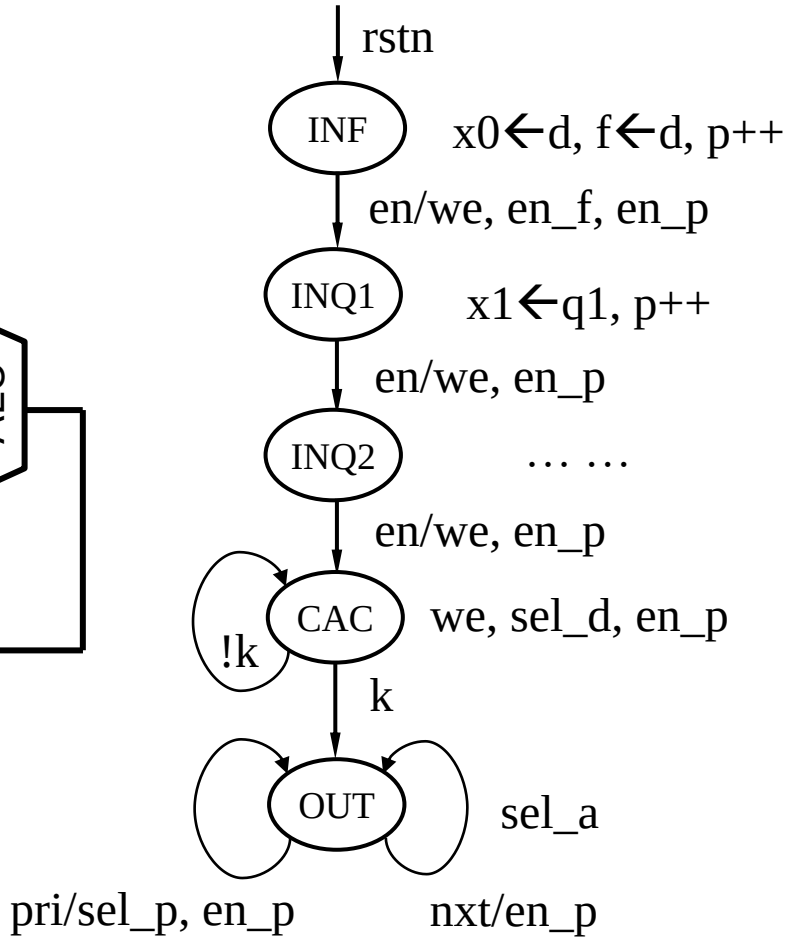


- **控制单元**
  - en, pri, nxt：均为单脉冲信号



示例：数列计算器 (续2)


F
sel_a
1
0
-2
-1
P
n
RF
ra0
ra1
wa
wd
we
rd0
rd1
q
ALU
d
0
1
sel_d
sel_p
+1
-1
0
1
P
==31
k

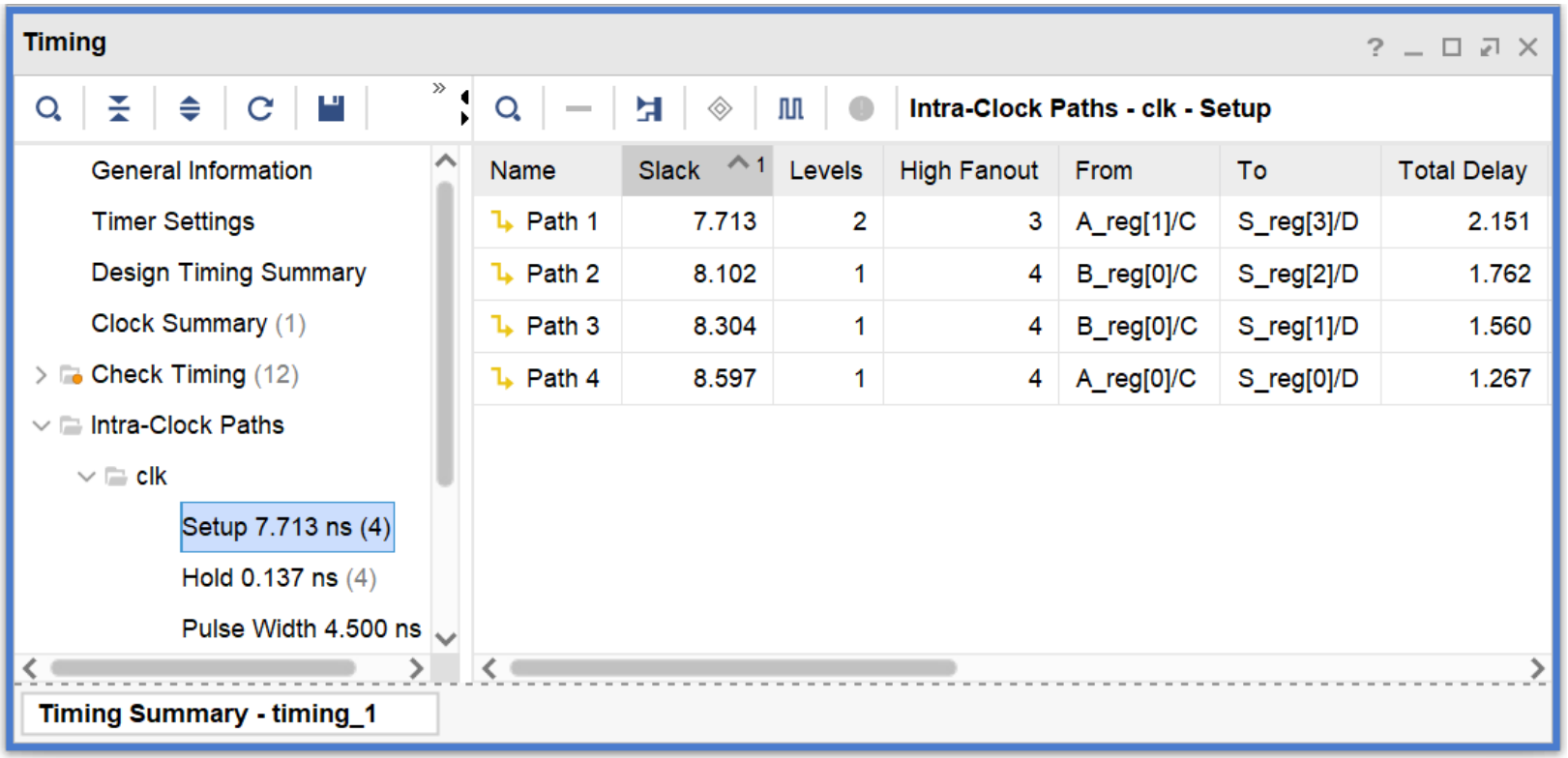

rstn
INF
x0←d, f←d, p++
en/we, en_f, en_p
INQ1
x1←q1, p++
en/we, en_p
INQ2
… …
en/we, en_p
CAC
!k
we, sel_d, en_p
k
OUT
sel_a
pri/sel_p, en_p
nxt/en_p

查看电路资源

- **查看Vivado生成电路**
  - RTL电路：Flow Navigator >> RTL Analysis >> Open Elaborated Design >> Schematic
  - 综合/实现电路：Flow Navigator >> Synthesis/Implementation >> Open Synthesized/Implemented Design >> Schematic
- **查看电路资源使用情况**
  - 综合/实现电路：Flow Navigator >> Synthesis /Implementation >> Open Synthesized/Implemented Design >> Report Utilization

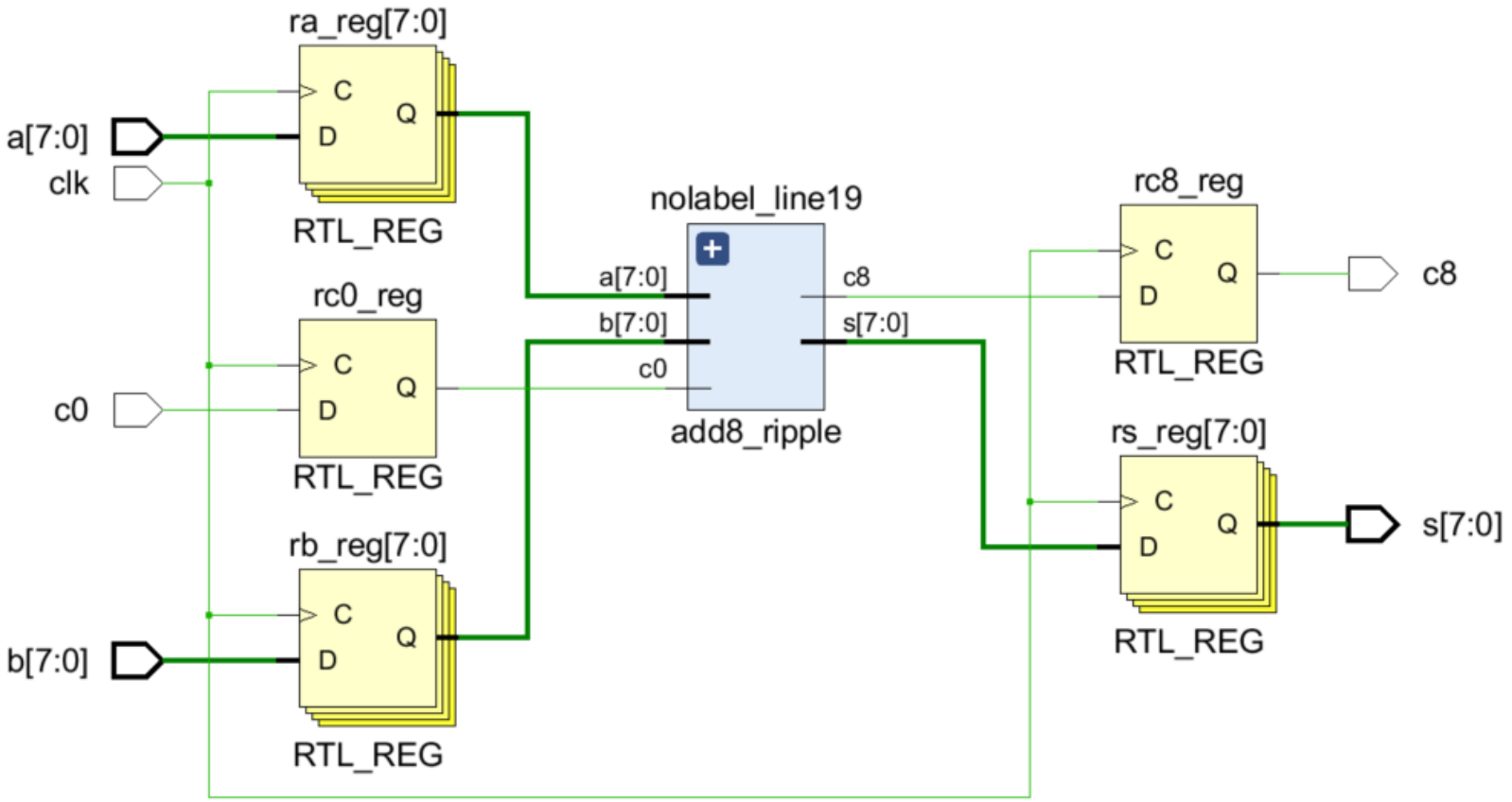
查看电路性能

- 查看综合电路性能
  - Flow Navigator >> Synthesis >> Open Synthesized Design >> Report Timing Summary



示例：加法器

- **8位串行进位加法器：** {c8, s} = a + b + c0;



示例：加法器 (续1)

- 设置时钟约束
  - 设置尽量高时钟频率，让Vivado工具优化电路和布局

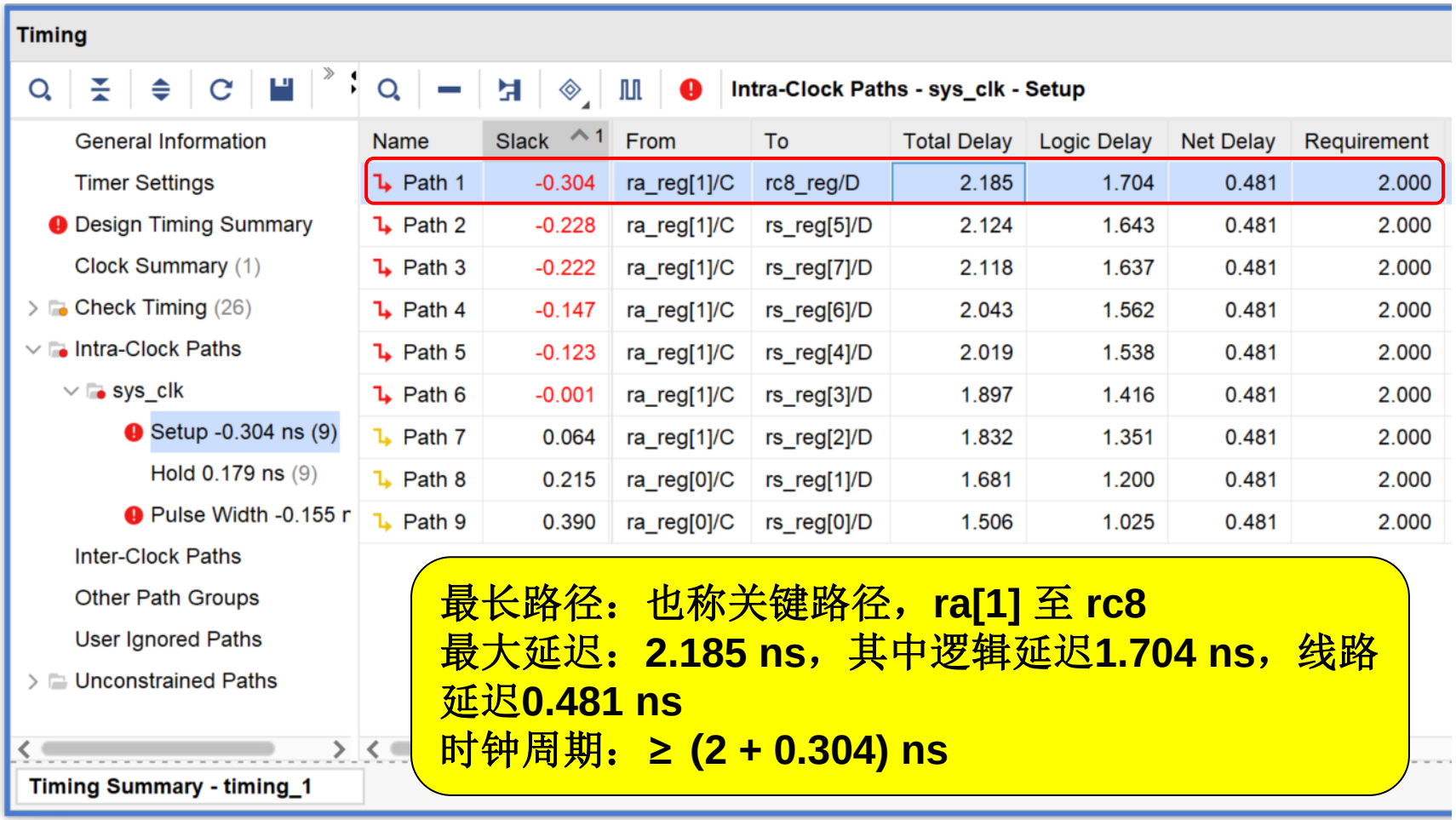
```
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }];

create_clock -add -name sys_clk -period 2.00 -waveform { 0 1 } [get_ports { clk }];
```

假设要求电路时钟频率能够达到500MHz

- 查看时序报告
  - Report Timing Summary
  - 了解电路最长路径和最大延迟
  - 若出现红色警示，则表示违反了时钟约束，生成电路的工作速度将可能达不到该频率

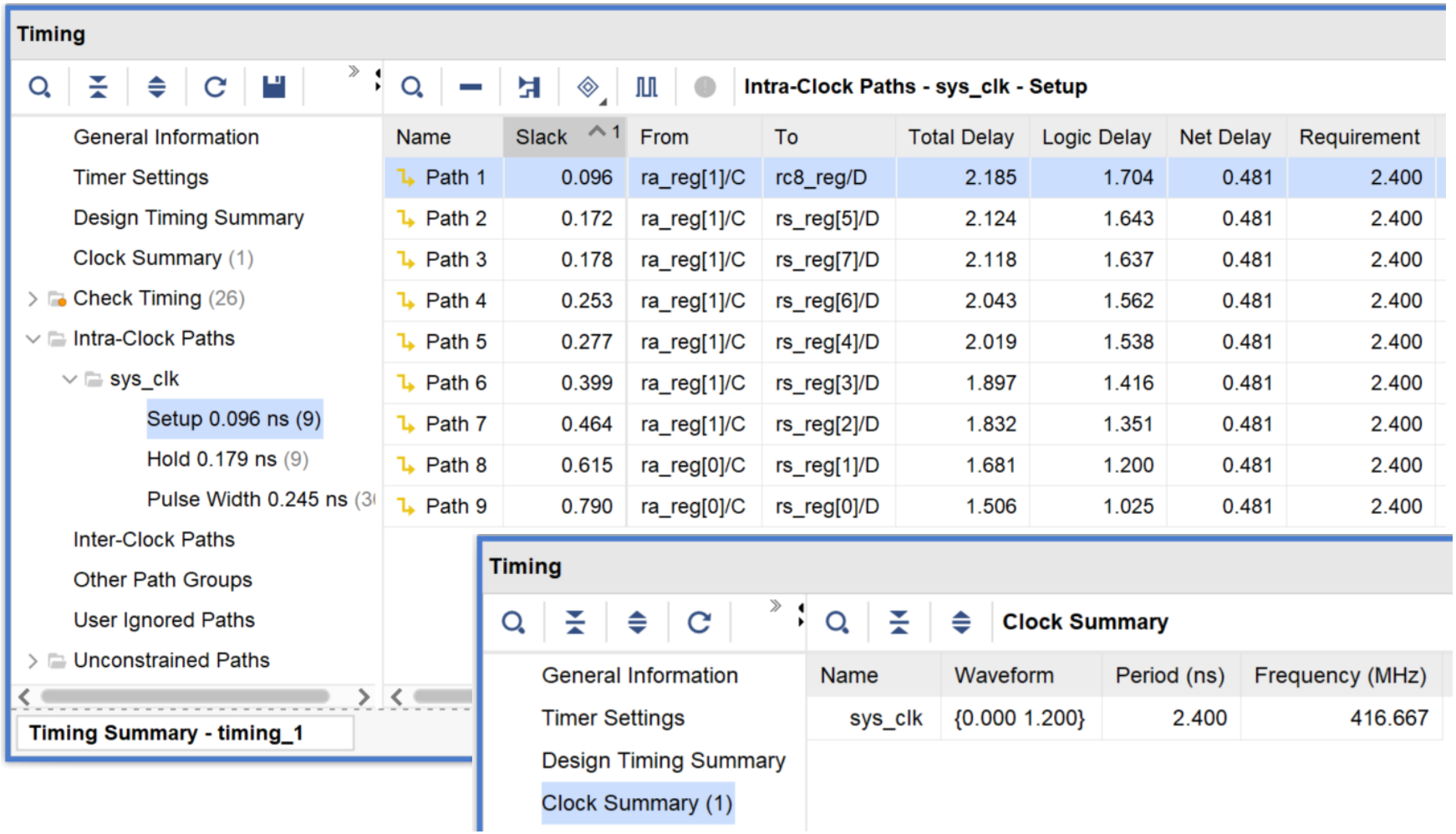
示例：加法器 (续2)



Name	Slack	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	-0.304	ra_reg[1]/C	rc8_reg/D	2.185	1.704	0.481	2.000
Path 2	-0.228	ra_reg[1]/C	rs_reg[5]/D	2.124	1.643	0.481	2.000
Path 3	-0.222	ra_reg[1]/C	rs_reg[7]/D	2.118	1.637	0.481	2.000
Path 4	-0.147	ra_reg[1]/C	rs_reg[6]/D	2.043	1.562	0.481	2.000
Path 5	-0.123	ra_reg[1]/C	rs_reg[4]/D	2.019	1.538	0.481	2.000
Path 6	-0.001	ra_reg[1]/C	rs_reg[3]/D	1.897	1.416	0.481	2.000
Path 7	0.064	ra_reg[1]/C	rs_reg[2]/D	1.832	1.351	0.481	2.000
Path 8	0.215	ra_reg[0]/C	rs_reg[1]/D	1.681	1.200	0.481	2.000
Path 9	0.390	ra_reg[0]/C	rs_reg[0]/D	1.506	1.025	0.481	2.000

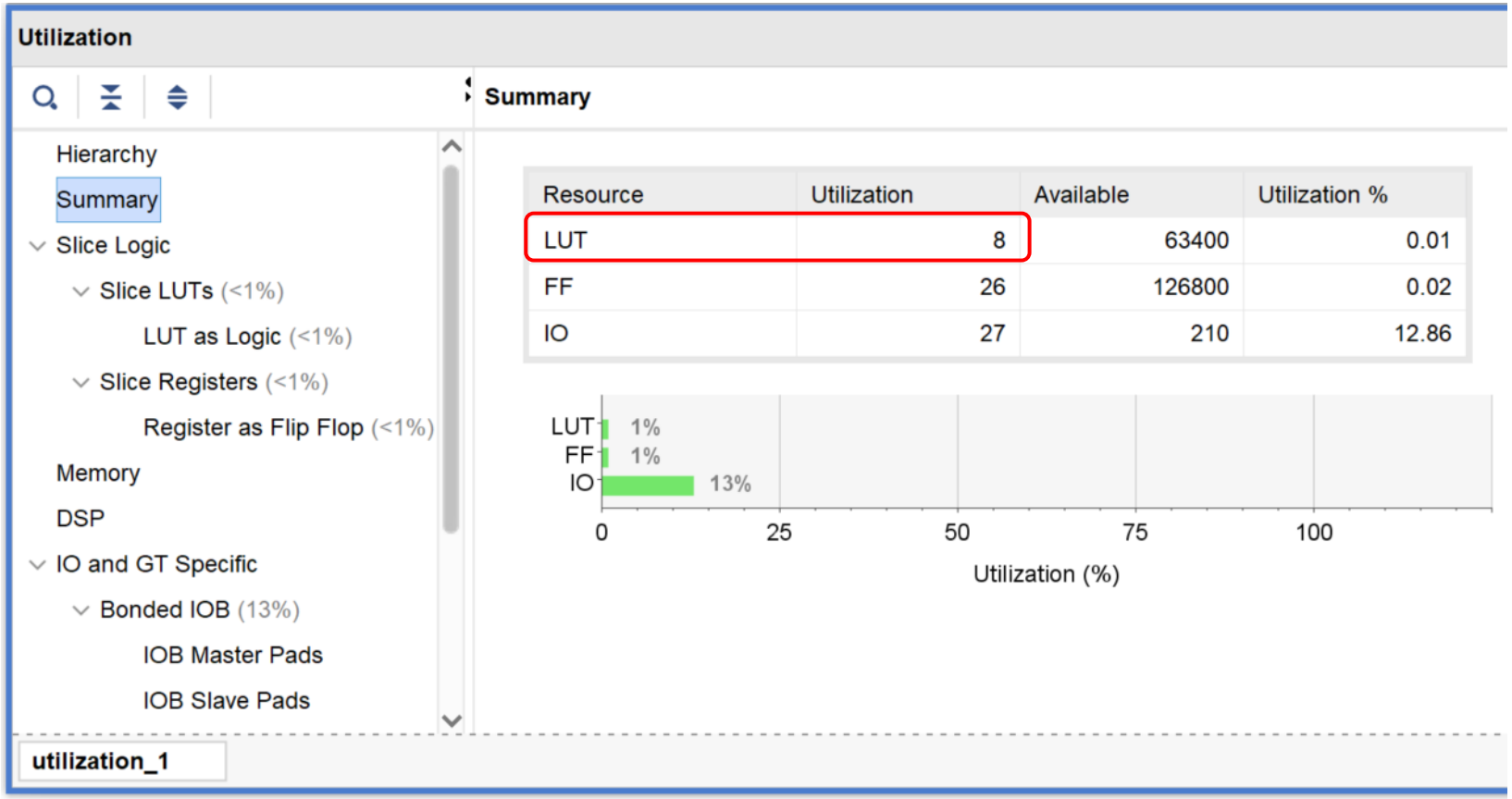
最长路径：也称关键路径，ra[1] 至 rc8
最大延迟：2.185 ns，其中逻辑延迟1.704 ns，线路延迟0.481 ns
时钟周期：≥ (2 + 0.304) ns

示例：加法器 (续3)


Timing
Intra-Clock Paths - sys_clk - Setup
General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (26)
Intra-Clock Paths
sys_clk
Setup 0.096 ns (9)
Hold 0.179 ns (9)
Pulse Width 0.245 ns (3
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
Timing Summary - timing_1
Name Slack 1 From To Total Delay Logic Delay Net Delay Requirement
Path 1 0.096 ra_reg[1]/C rc8_reg/D 2.185 1.704 0.481 2.400
Path 2 0.172 ra_reg[1]/C rs_reg[5]/D 2.124 1.643 0.481 2.400
Path 3 0.178 ra_reg[1]/C rs_reg[7]/D 2.118 1.637 0.481 2.400
Path 4 0.253 ra_reg[1]/C rs_reg[6]/D 2.043 1.562 0.481 2.400
Path 5 0.277 ra_reg[1]/C rs_reg[4]/D 2.019 1.538 0.481 2.400
Path 6 0.399 ra_reg[1]/C rs_reg[3]/D 1.897 1.416 0.481 2.400
Path 7 0.464 ra_reg[1]/C rs_reg[2]/D 1.832 1.351 0.481 2.400
Path 8 0.615 ra_reg[0]/C rs_reg[1]/D 1.681 1.200 0.481 2.400
Path 9 0.790 ra_reg[0]/C rs_reg[0]/D 1.506 1.025 0.481 2.400
Timing
Clock Summary
General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Name Waveform Period (ns) Frequency (MHz)
sys_clk {0.000 1.200} 2.400 416.667

示例：加法器 (续4)

- 查看电路资源使用情况：Report Utilization



实验要求

- **完成ALU和RF的设计**
  - RF的x0内容恒为零，读模式为写优先
  - 查看RTL电路图，综合后功能仿真和查看电路资源
- **比较分布式与块式存储器的特性**
  - 单端口RAM存储器，容量1024 x 32位
  - 实现后时序仿真和查看电路资源
- **完成SRT的模块化设计**
  - 查看RTL电路图，综合后功能仿真、查看电路资源和性能
  - 下载测试
- **选项**
  - 使用块式存储器实现SRT，比较两种实现后的电路资源和性能

The End