Course Design of Computer Organization

Wang Zonghui

College of Computer Science & Technology,
Zhejiang University
March, 2010

topics

- 1. Verilog and Xilinx ISE
- 2. Basic logic component of datapath design
- 3. ALU and the ALU controller design
- 4. R-type instruction processor design
- 5. CPU controller design
- 6. Single-cycle clock CPU design
- 7. Multiple-cycle clock CPU design
- 8. Microprogrammed CPU controller unit design
- 9. Design of datapath of Microprogrammed CPU

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures

Experiment Purpose

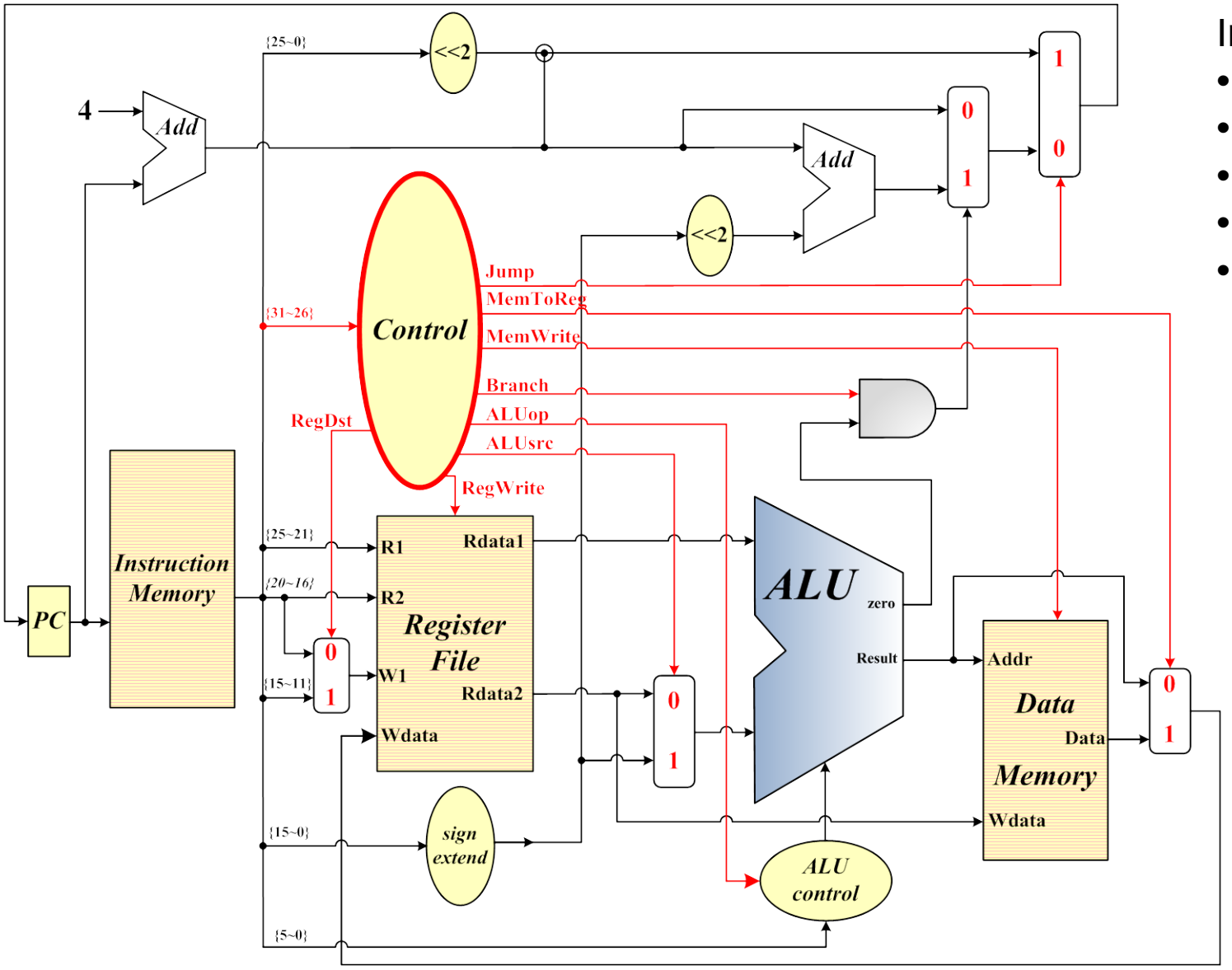
- Understand the principles of Datapath and master methods of Datapath design
- Understand the principles of Single-cycle clock CPU and master methods of Single-cycle clock CPU design
- master methods of program verification of CPU

Experiment Task

- Design the Datapath, bring together the basic units into Single-cycle clock CPU
- Verify the CPU with program and observe the execution of program

Basic Principle

- Circuit diagram of Single-cycle clock CPU.
- Constitution
- Basic components
- IP Core for Memory



Instr. Type:

- R-type
- LW
- SW
- BEQ
- Jmp

Basic Units of Single-cycle CPU

- CPU controller
- ALU and its controller
- Register file
- Instruction Memory
- Data Memory
- others： Register, adder, sign-extend Unit, shifter, multiplexor。

2 to 1 Multiplexor Module

```
module single_mux(A, B, Ctrl, S);
    parameter N;
    input  wire[N-1:0] A, B;
    output wire[N-1:0] S;
    assign S = (Ctrl == 1'b0) ? A : B;
endmodule
```

The usage of Verilog instantiation's parameter:

- mod_name #(value, …) inst_name(port_map);
- mod_name #(.param(value), …) inst_name (port_map);
- defparam heirarchy_path.param_name = value;

PC + 4 Module

```
module single_pc_plus_4(i_pc, o_pc);
    parameter N;
    input  wire[N-1:0] i_pc;
    output wire[N-1:0] o_pc;
    assign o_pc = i_pc + 1;
endmodule
module single_pc(clk, rst, i_pc, o_pc);
    parameter N;
    intput wire clk, rst;
    input  wire[N-1:0] i_pc;
    output wire[N-1:0] o_pc;
    reg[N-1:0] t_pc;
    assign o_pc = rst ? {N{1'b1}}:t_pc;
    always @(posedge clk)
      t_pc <= i_pc;
endmodule
```

Big Modules

- CPU Controller
- ALU
- ALU Controller
- Register file
- Memory

Register File

```
module single_gpr( clk, rst, i_addr1, i_addr2, i_addr3,
  i_wreg,  i_wdata, i_wen, o_op1,   o_op2,   o_op3);
   input  wire        clk, rst, i_wen;
   input  wire[ 4:0] i_addr1, i_addr2, i_addr3, i_wreg;
   input  wire[31:0] i_wdata;
   output wire[31:0] o_op1, o_op2, o_op3;
          reg [31:0] mem[31:0];
   assign o_op1 = mem[i_addr1];
   assign o_op2 = mem[i_addr2];
   assign o_op3 = mem[i_addr3]; // for test purpose
   always @(posedge clk or posedge rst)
      if (rst == 1'b1) mem[0] = {32{1'b0}};
      else if (i_wen) // write data to register
          mem[i_wreg] = (i_wreg == {5{1'b0}}) ?
                                 {32{1'b0}} : i_wdata;
endmodule
```

Generate IP Core for Mem.

- COE file: in ASCII format, a core must use the file when it will need to configure multiple data.
- XCO file: it is one of a series of output files when CoreGen generates core, and it stores parameters which produce core required.
- VEO file: Verilog template file, the component can be used for instancing a core.
- V File: Verilog package file, and support the Verilog functional simulation for the core.

COE file content

```
Keyword = Value ; Optional Comment
Keyword = Value ; Optional Comment
<Radix_Keyword> = Value ; Optional Comment
<Data_Keyword>  = Data_Val1, Data_Val2, Data_Val3;
```

Radix Keyword	Description
RADIX	Initial the non-mem. core
MEMORY_INITIALIZATION_RADIX	Initial the Virtex series mem.

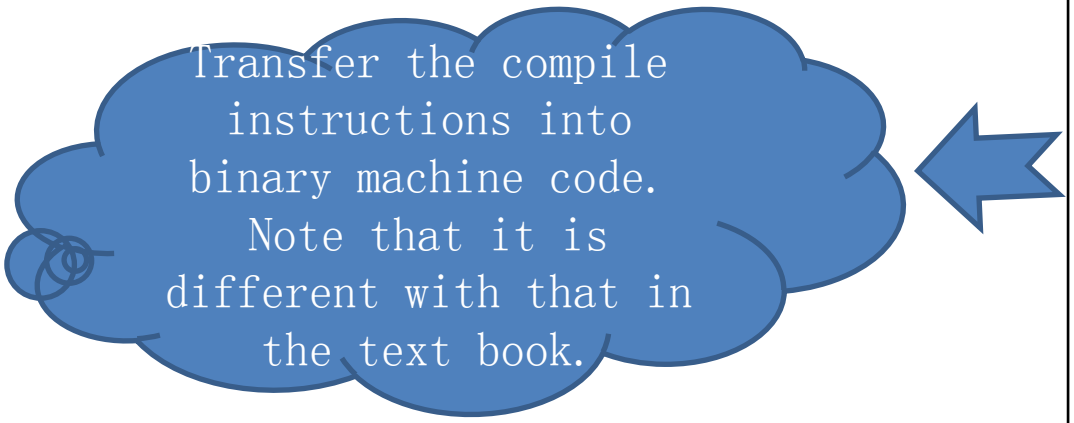
Data Keyword	Description
CoefData	Specify filter's parameter
MemData	For XC4000 series mem.
MEMORY_INITIALIZATION_VECTOR	For Virtex series mem.
PATTERN	For bit-correlator
BRANCH_LENGTH_VECTOR	For Interleaver COE files

Instruction Mem.

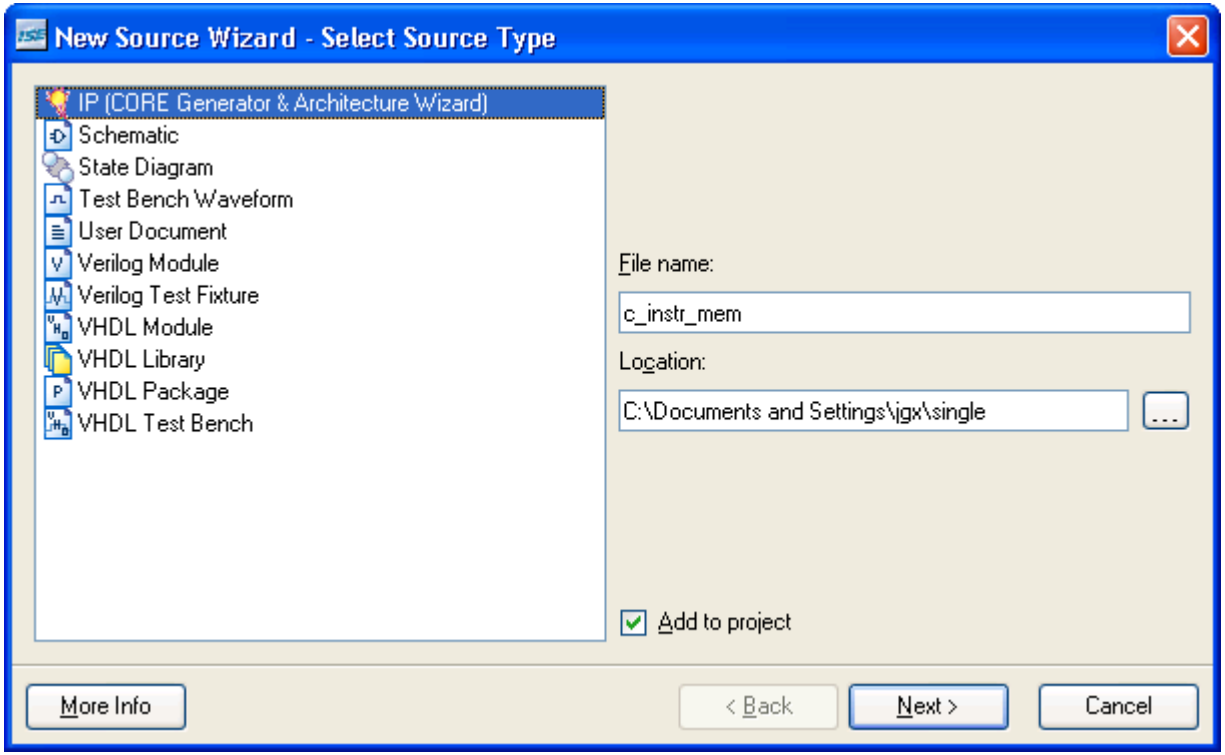
- A simple instance for CPU testing: compute the result of 1+2+3+…+10(=55).

MIPS compile				pseudocode
	LW	$0, 0($0)		$0 = 1 ←$0[0];R only
	LW	$1, 1($0)		$1 = 11←$0[1];R only
	LW	$2, 2($0)		$2=0←$0[2];initial sum
	ADD	$3, $0, $2		$3 = 1;
REP:	ADD	$2, $2, $3		do{ $2 += $3;
	ADD	$3, $3, $0		$3 += $0;
	BEG	$3, $1, OUT		} while ($3 != $1);
	J	REP		
OUT:	SW	$2, 3($0)		$2 → $0[3];save result

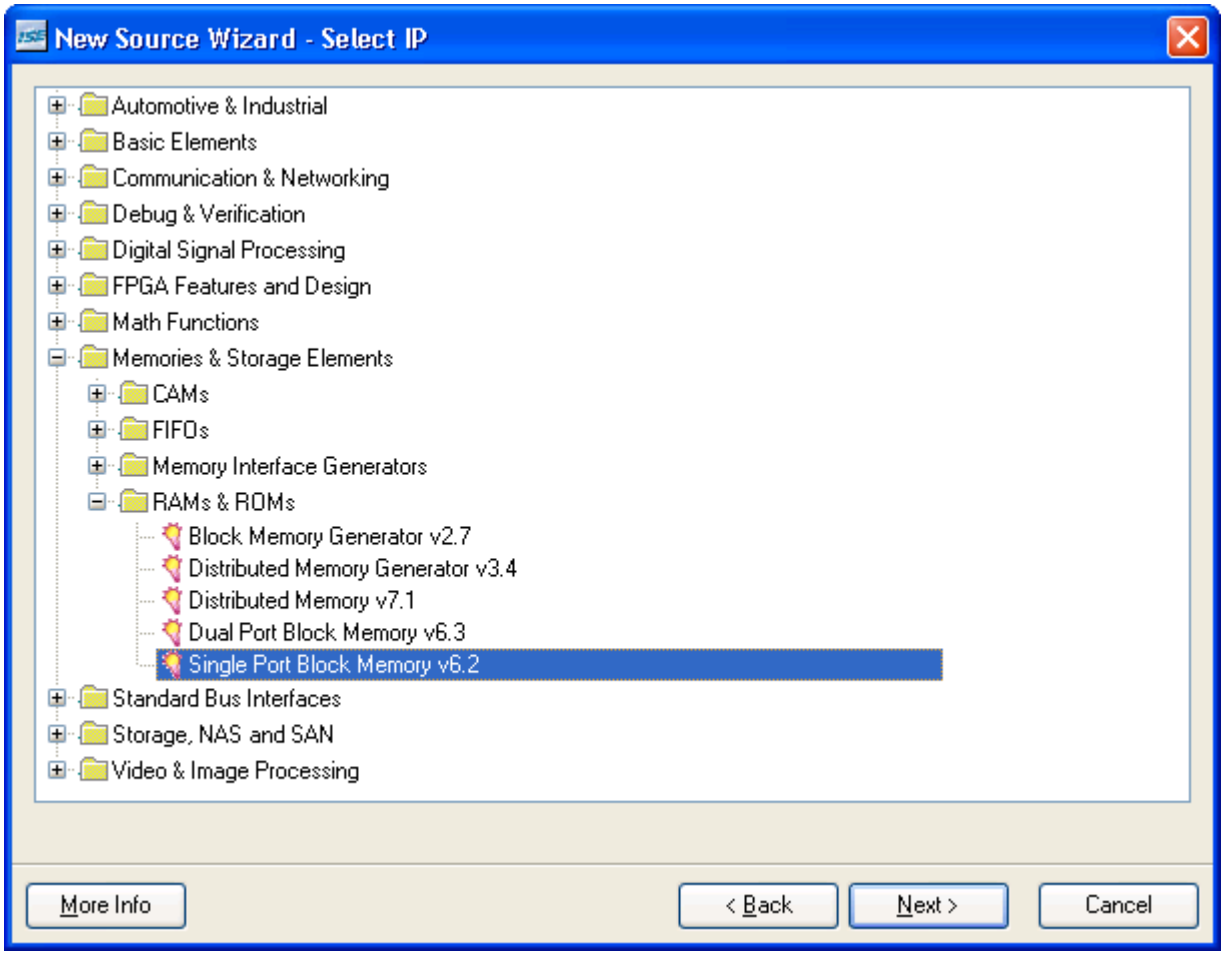
Coe and instructions

#	COE files' content(machine instruction)	description
0		LW $0, 0($0)
1		LW $1, 1($0)
2		LW $2, 2($0)
3		ADD $3, $0, $2
4		ADD $2, $2, $3
5		ADD $3, $3, $0
6		BEG $3, $1, 1
7		J 4
8		SW $2, 3($0)
9		J 0
10	00000000000000000000000000000000	
	...	
510	00000000000000000000000000000000	
511	00000000000000000000000000000000	

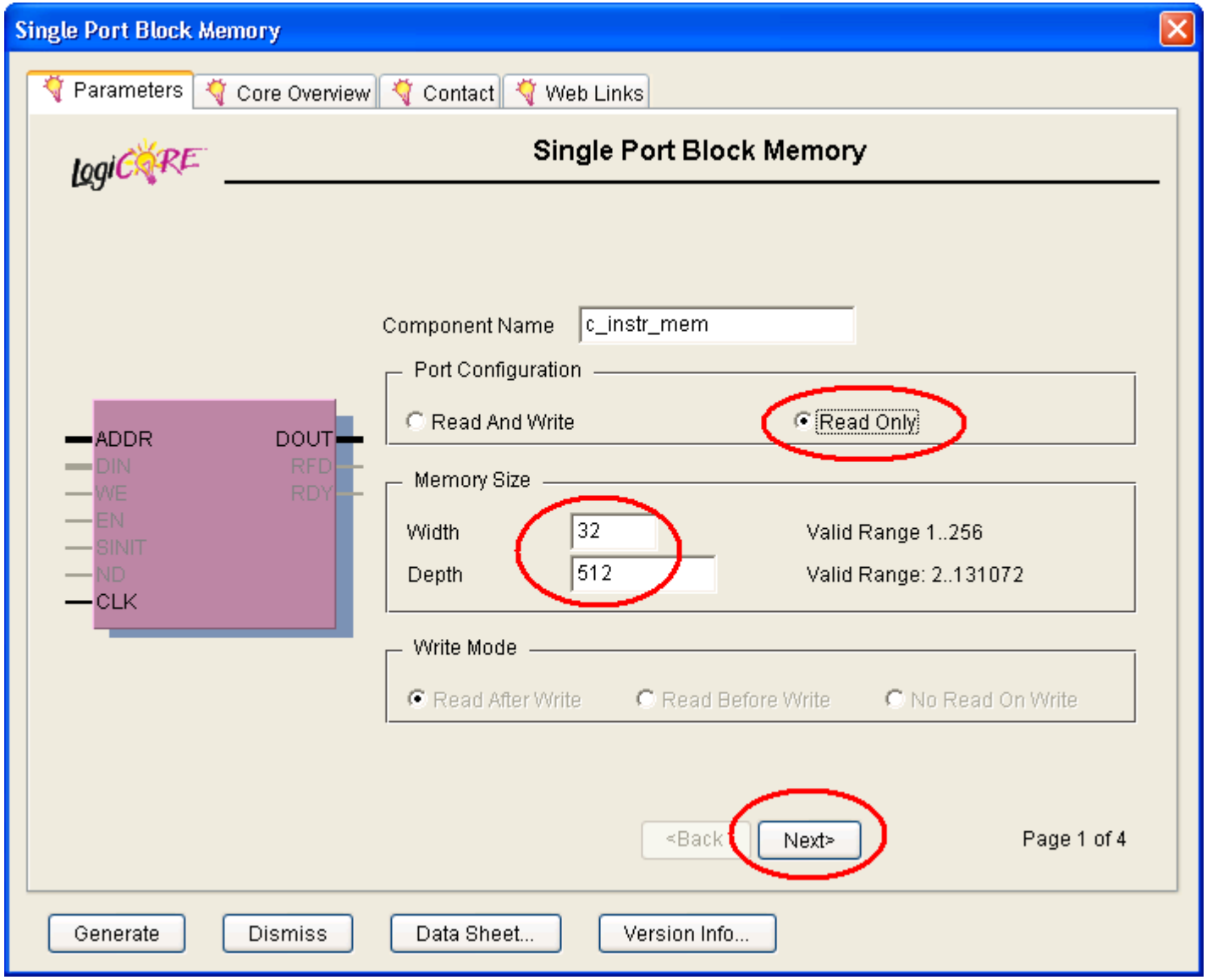
Instruction Mem. (1)



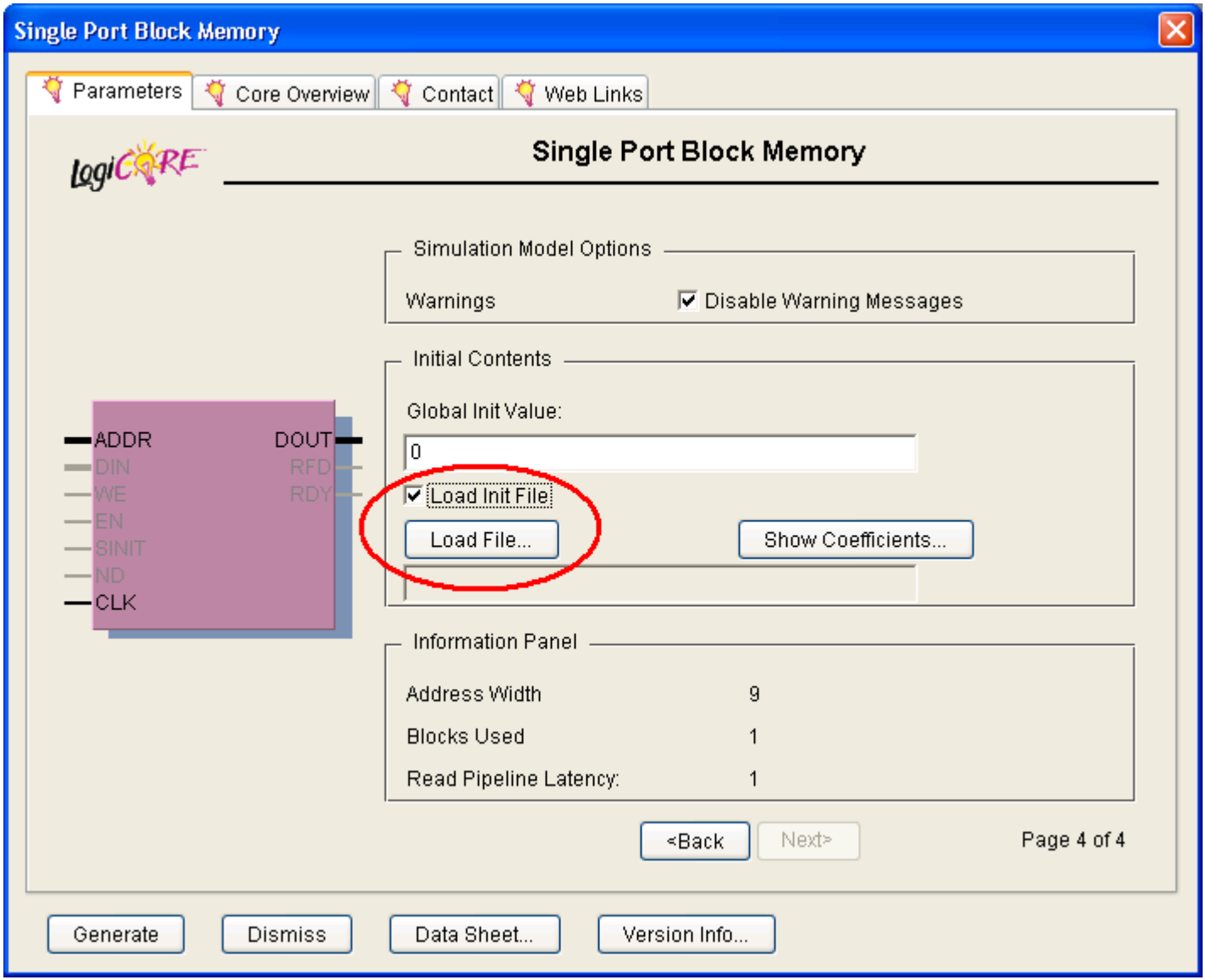
Instruction Mem. (2)



Instruction Mem. (3)



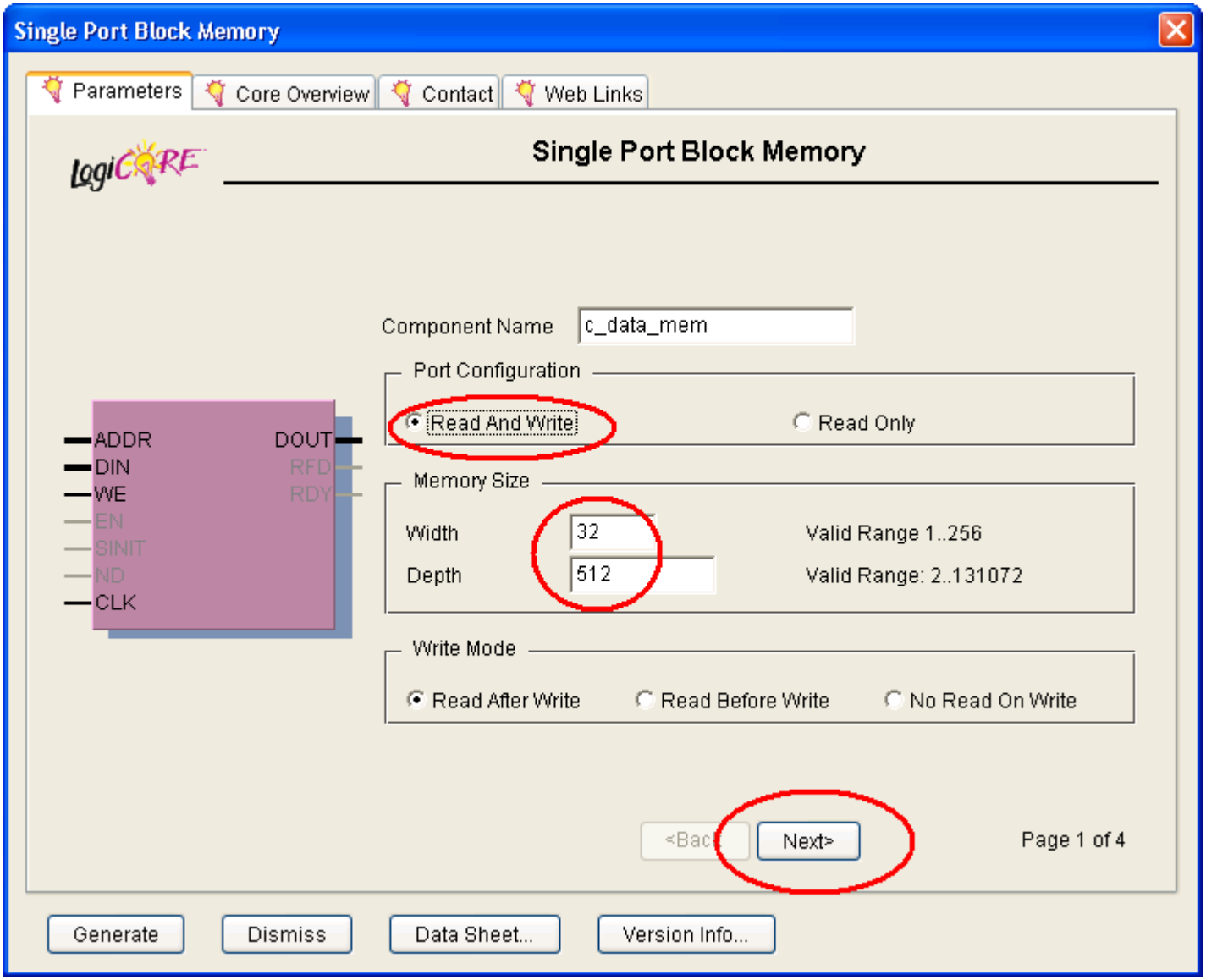
Instruction Mem. (4)



Data Mem.

COE file’s content(data mem.)	description
MEMORY_INITIALIZATION_RADIX = 2; MEMORY_INITIALIZATION_VECTOR = 00000000000000000000000000000001, 00000000000000000000000000001010, 00000000000000000000000000000000, 00000000000000000000000000000000, 00000000000000000000000000000000, ... 00000000000000000000000000000000, 00000000000000000000000000000000, 00000000000000000000000000000000, 00000000000000000000000000000000, 00000000000000000000000000000000, 00000000000000000000000000000000;	**binary** **Start of computation: 1** **End of computation: 11** **Sum’s initiatory value: 0** **Sum’s memory address**

Data Mem.


Single Port Block Memory
Parameters
Core Overview
Contact
Web Links
LogiCORE
Single Port Block Memory
Component Name
c_data_mem
Port Configuration
Read And Write
Read Only
ADDR
DIN
WE
EN
SINIT
ND
CLK
DOUT
RFD
RDY
Memory Size
Width
32
Valid Range 1..256
Depth
512
Valid Range: 2..131072
Write Mode
Read After Write
Read Before Write
No Read On Write
<Back
Next>
Page 1 of 4
Generate
Dismiss
Data Sheet...
Version Info...

Experiment Content

- 1. Create project and write code.
- 2. Generate Instr. Mem. and Data Mem..
- 3. Write UCF file.
- 4. Synthesize /Implement.
- 5. Program the bit file and verify it.

Input and Output

- Input
  - Button 0: Step execute
  - Button 1: Reset
  - Slide Button 0-1: Display type
  - Slide Button 2-6: Register Index
- Output
  - LED 0: Execution
  - LED 1-5: Instruction type
  - LED Tubes: (display type 00-low 16bit of register of selection ; 01-high 16bit; 10-PC; 11-Clock Count).

Experiment Report Requirement

- Draw out the organization chart of all modules.
- Give the description of the following code：
  - Including the detailed comment of all the modules’ Verilog code.
  - Including the detailed comment of UCF.
- Describe the detail running status of each instructions and analyze the result (do 10 steps for loop).
- Voiding the following：
  - Put Full-page screenshot of the courseware into the report.
  - Put ISE Notes and code automatically generated into the report.
  - The total number of the experiment report exceed 25 pages.