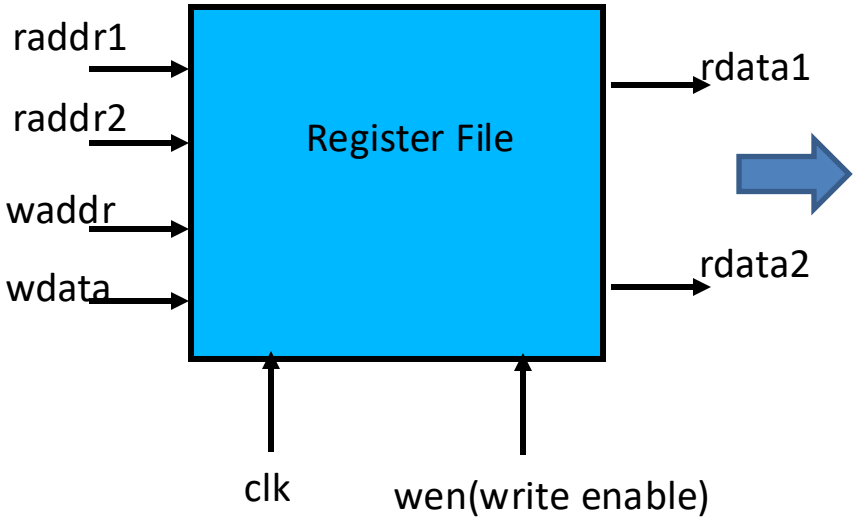
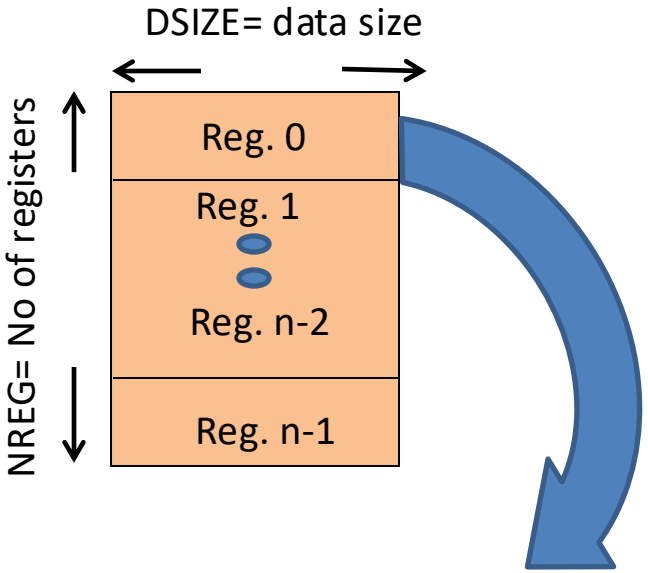
SC3050
Advanced Computer Architecture

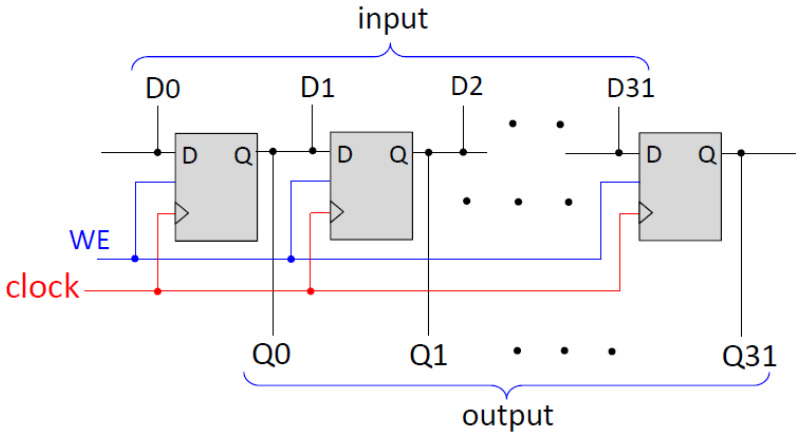
Lab 2 briefing

College of Computing and Data Science
Nanyang Technological University

Register file







Keep in mind

- Synthesis by keeping DSIZE=64
  - Need to change the NREG fro 4 to 64
  - When we change the NREG (no of registers) the address size also need to be changed.
  - Hence when NREG=4, ASIZE(address size)=2
  - NREG=8, ASIZE(address size)=3
  - NREG=2^n, ASIZE(address size)=n
- Synthesis by keeping NREG=32
  - Need to change the DSIZE fro 4 to 64
  - When we change the DSIZE the address size doesn't need to be changed.
- The number of LUTs is directly available in design summary, the delay here is the minimum clock period and not the combinational delay as this is a sequential circuit

datapath simplified- R format

opcode	Rm	shamt	Rn	Rd
31 21	20 16	15 10	9 5	4

Examples:

1. ADD X5, X4, X3 (meaning: X5 ← (X4 + X3). The machine format is given below.

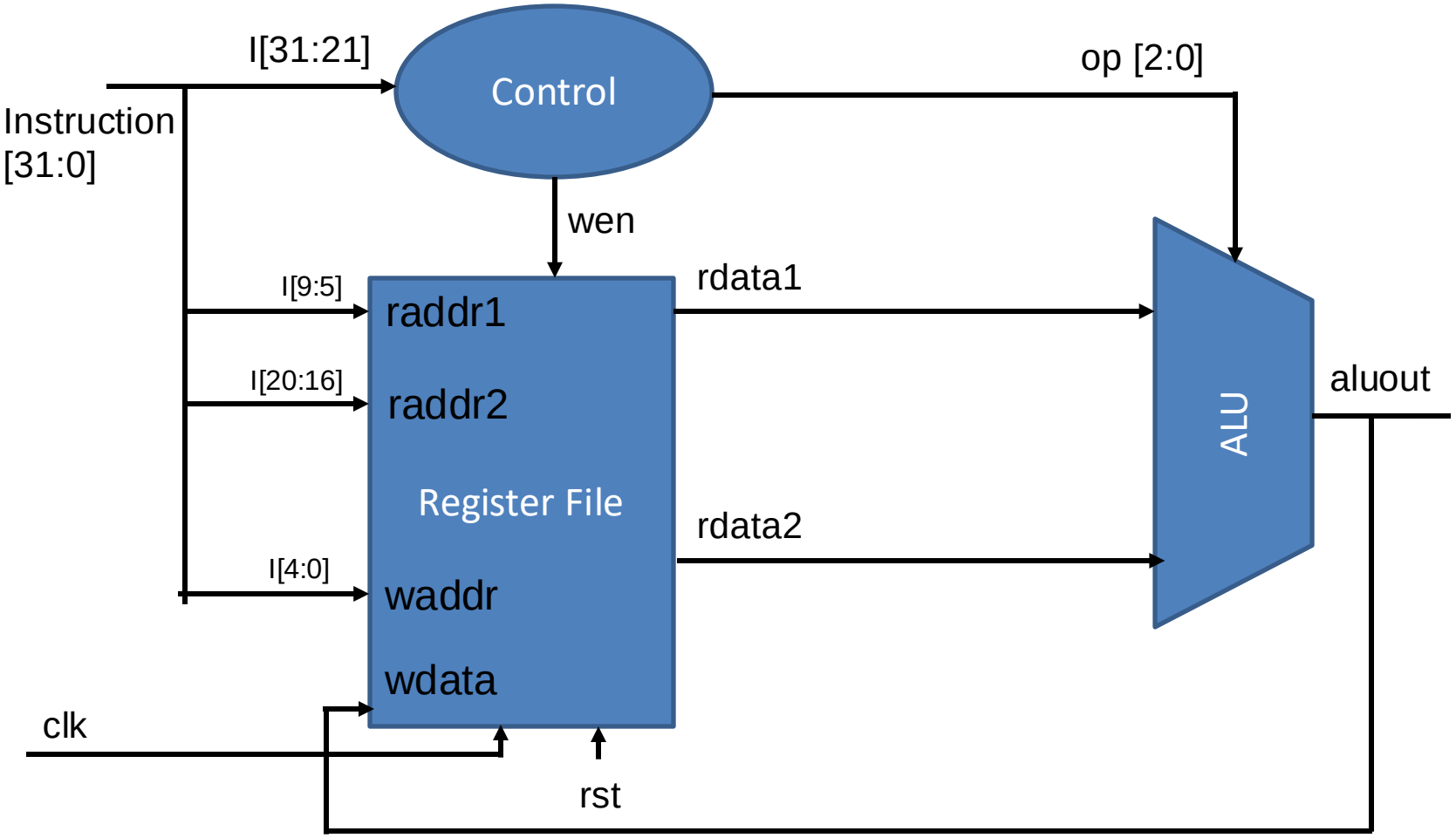
00000000000	00011	000000	00100	00101
31 21	20 16	15 10	9 5	4

2. XOR X8, X9, X10 (meaning: X8← X9 (XOR) X10):

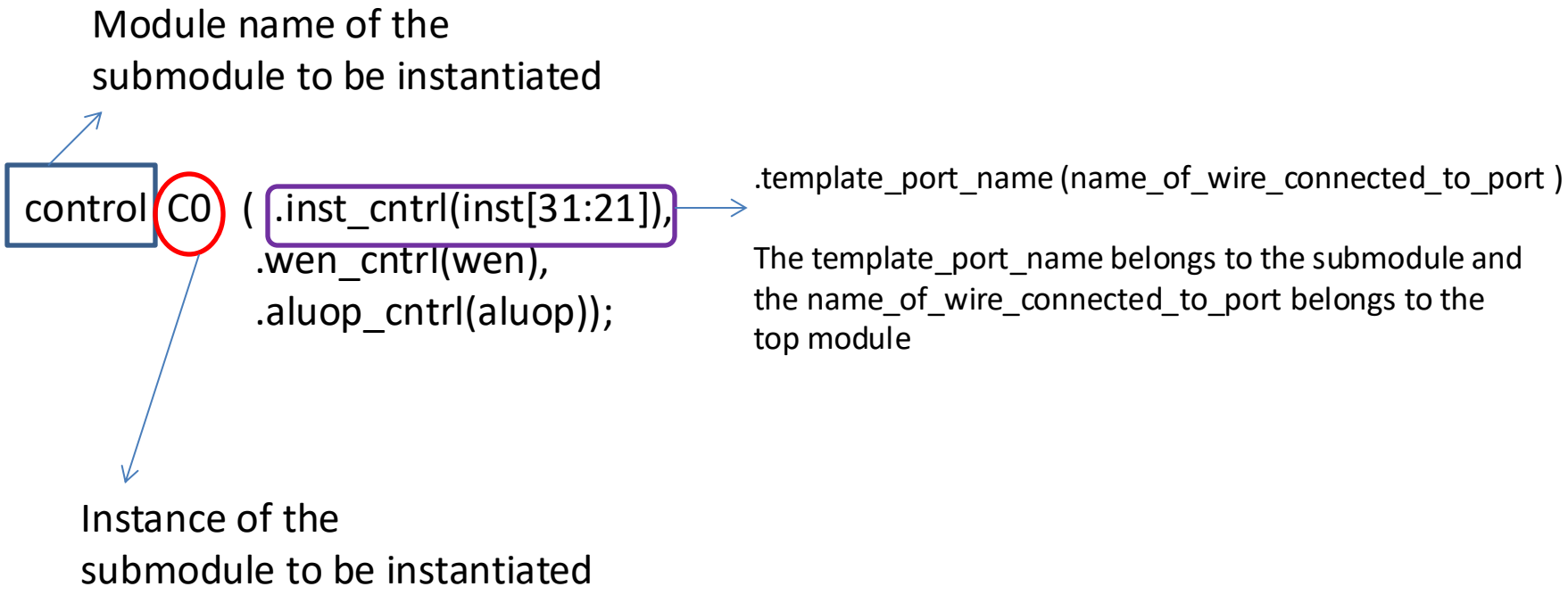
00000000011	01010	000000	01001	01000
31 21	20 16	15 10	9 5	4

R – register type instructions(ADD, SUB, AND, XOR, and ORR)

Datapath for R type instructions



Module instantiation- example



To make the datapath

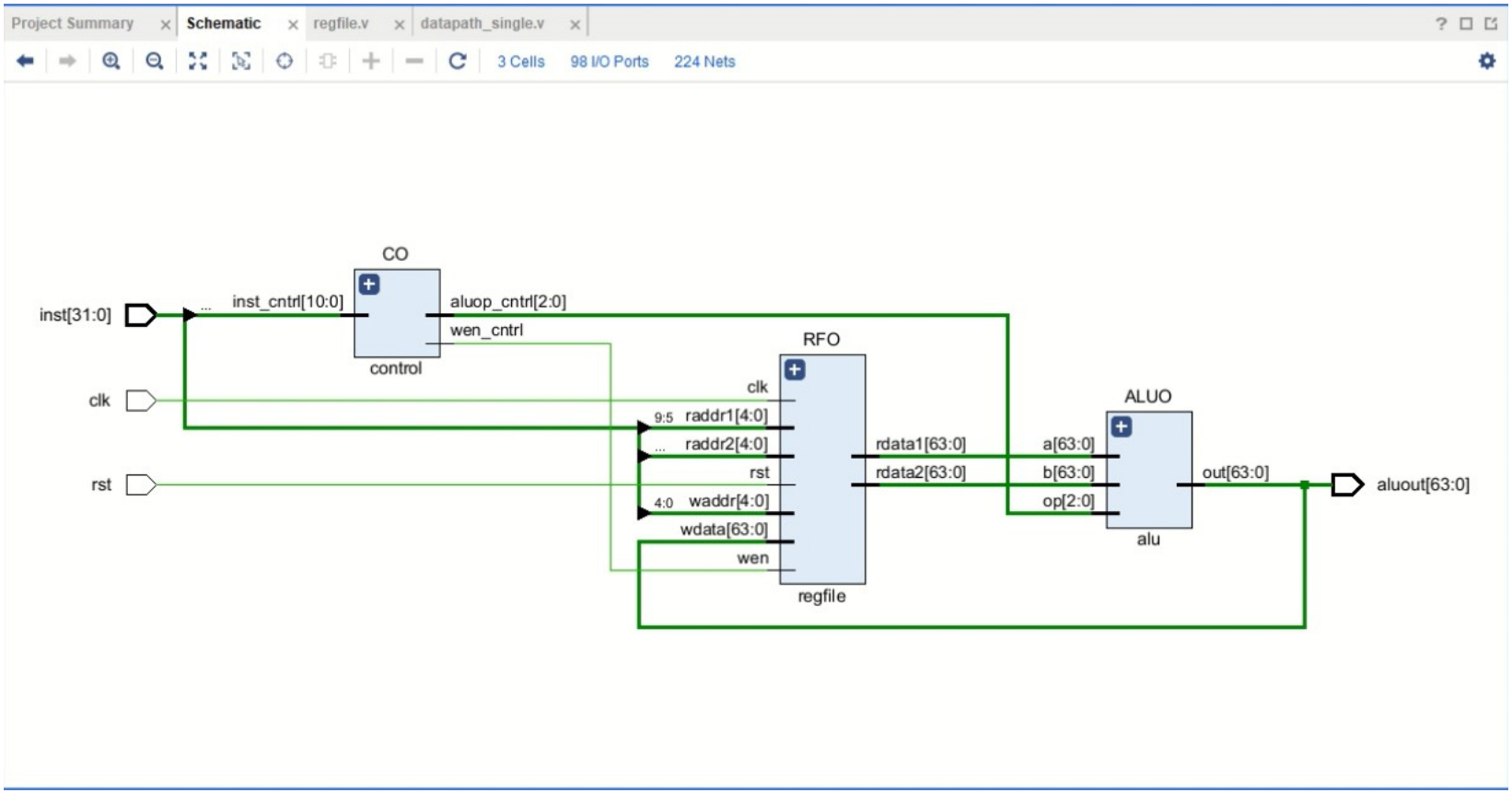
- You need to instantiate all the following modules and connect then as according to the datapath diagram

```
control C0
(.inst_cntrl(inst[31:21]),.wen_cntrl(wen),.aluop_cntrl(aluop));

regfile  RF0 (.clk(clk), .rst(rst), .wen(wen), .raddr1(inst[9:5]),
.raddr2(inst[20:16]), .waddr(inst[4:0]), .wdata(aluout),
.rdata1(rdata1), .rdata2(rdata2));

alu  ALU0 ( .a(rdata1), .b(rdata2), .op(aluop), .out(aluout));
```

RTL schematic



Testing the datapath(change)

- Hardcoded in “regfile.v”

regdata[1] <=3

regdata[2] <=2;

Rest all the registers from 0-31 are initialized to zero

Opcode **Rn=source1**

inst=32'b00000000000001000000000001000011; // ADD X3, X1, X4

Rm=source2 **Rd=dest**

Add X3, X1, X4= [X3] <= [X1] + [X4]

As reg[1] has value 3 and reg 2 has value 0,

the result stored in reg[3] =3