컴퓨터공학 기초 실험2

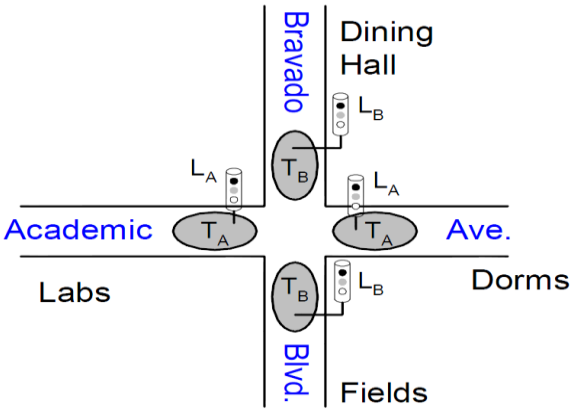
Lab #7

Register file

Register file

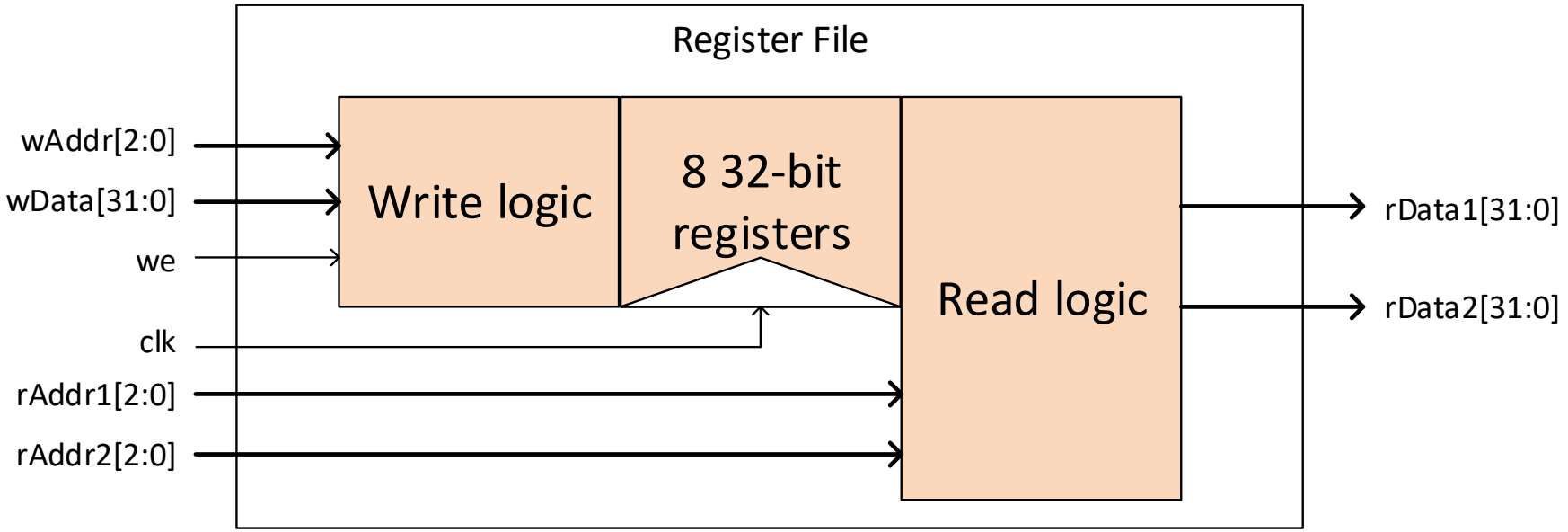
- A register file (RF) is composed of a set of registers that can be read and written by supplying a register number(address) to be accessed
  - ✓ Set of registers
  - ✓ Read operations
  - ✓ Write operations

- A RF usually can handle multiple read and write time



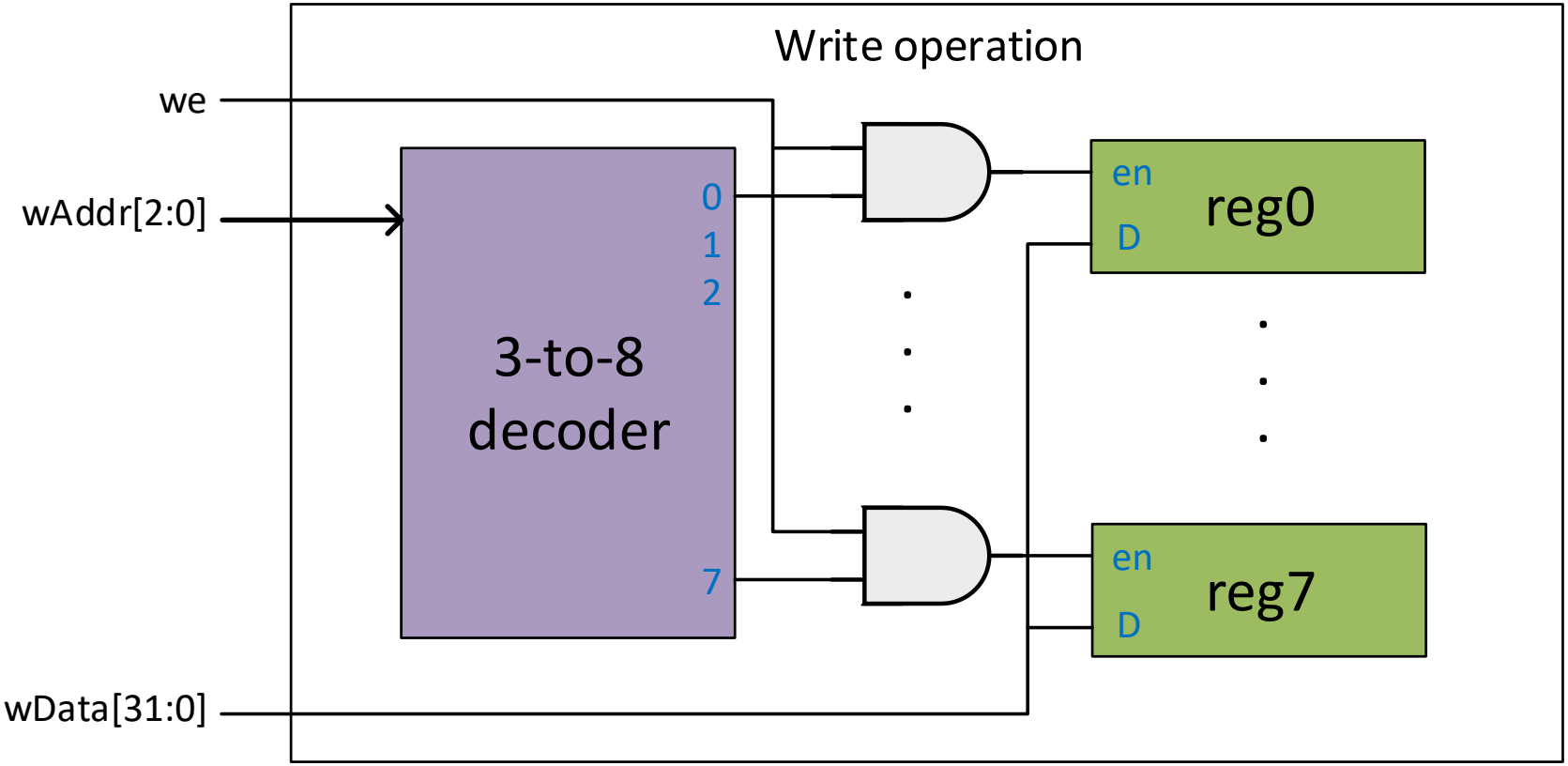
Example

- Consider a register composed of eight 32-bit registers with two read ports and one write port
  - ✓ Eight 32-bit registers
  - ✓ Two read address/data port
  - ✓ One write address/data port



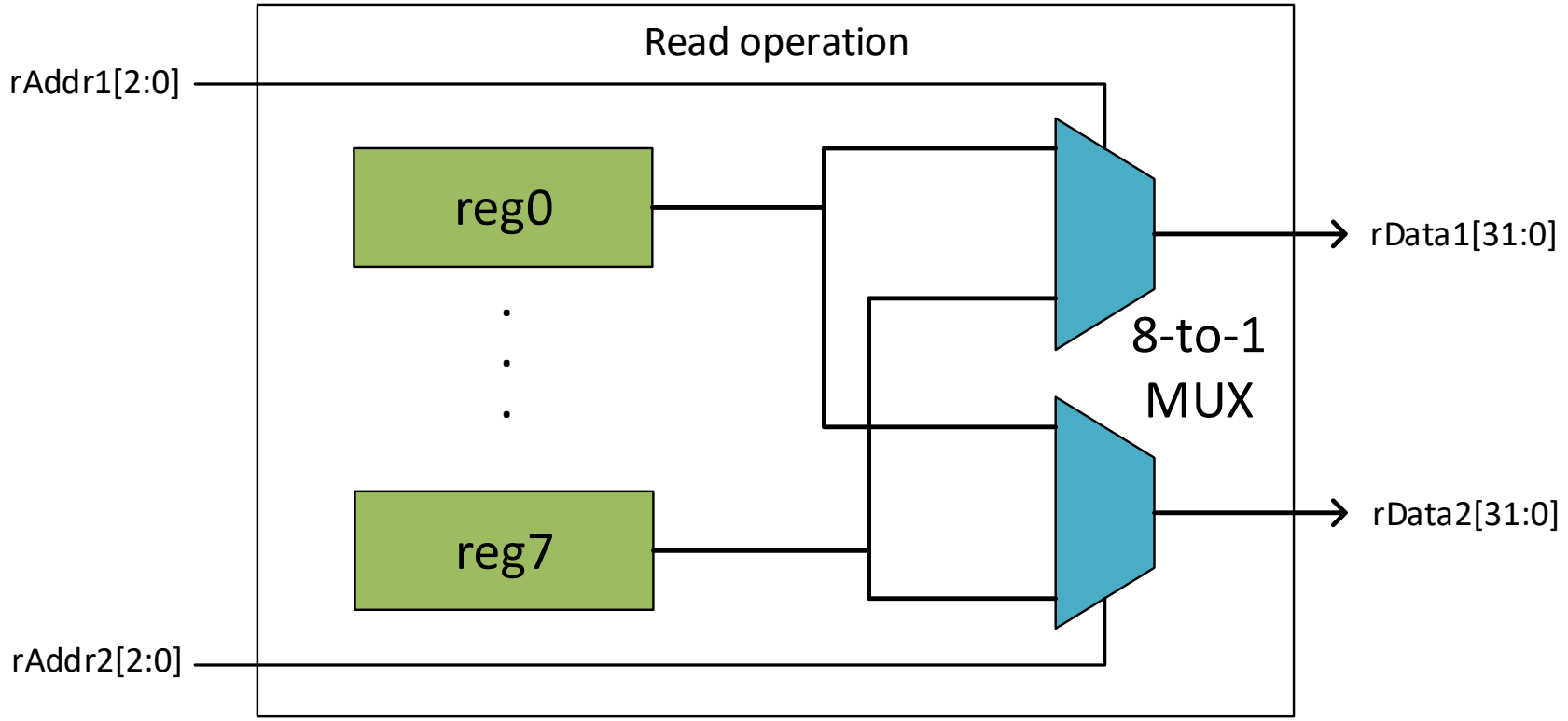
Write operation

- wAddr selects one of eight registers
- wData is written into the selected register only if **we** is active (high)



Read operation

- Read does not change values of any register in a register file



A 32bit register file

PRACTICE I

Functional Description

- 32bit register를 8개 가진 register file
- Write는 write enable (we) 에 의해 활성화
  - Read는 read address(rAddr)로 선택된 레지스터의 값이 출력됨
- Write operation
  - Decoder를 통해 address를 해석하여 해당 register enable
- Read operation
  - MUX를 통해 8개의 register 중 한 개 선택

Project properties

- New Project Wizard
  - Project name : Register_file
  - Family & Device : Cyclone V 5CSXFC6D6F31C6 (밑에서 6번째)
- Verilog file
  - Add files: gates.v
  - New files: register32_r_en.v, write_operation.v, read_operation.v, Register_file.v

Top Module

- Implementation

```
module Register_file(clk, reset_n, wAddr, wData, we, rAddr, rData);
  input                    clk, reset_n, we;
  input     [2:0]          wAddr, rAddr;
  input     [31:0]         wData;
  output    [31:0]         rData;

  wire            [7:0]    to_reg;
  wire            [31:0]   from_reg[7:0];
```

Instance of register32_8, write_operation, read_operation

```
endmodule
```

Register32_r_en(1/3)

- Resettable enabled D Flip-flop
  - reset_n이 enable보다 우선순위가 높음

```
module _dff_r_en(clk, reset_n, en, d, q);
  input    clk, reset_n, en, d;
  output reg     q;

  always@(posedge clk or negedge reset_n)
  begin
    if(reset_n == 0)         q <= 1'b0;
    else if(en)              q <= d;
    else                     q <= q;
  end
endmodule
```

Register32_r_en(2/3)

- 32bit register

```
module register8_r_en(clk, reset_n, en, d_in, d_out);
       input        clk, reset_n, en;
       input  [7:0] d_in;
       output [7:0] d_out;

   Instance of dff_r_en

endmodule
```

```
module register32_r_en(clk, reset_n, en, d_in, d_out);
  input             clk, reset_n, en;
  input      [31:0] d_in;
  output     [31:0] d_out;

   Instance of register8_r_en

Endmodule
```

Register32_r_en(3/3)

- 32bit register를 8개 instance

```
module register32_8(clk, reset_n, en, d_in, d_out0, d_out1, d_out2, d_out3, d_out4,
        d_out5, d_out6, d_out7);
  input         clk, reset_n;
  input         [7:0]en;
  input         [31:0]d_in;
  output        [31:0]d_out0, d_out1, d_out2, d_out3, d_out4, d_out5, d_out6, d_out7;

```

Instance of register32_r_en

```
endmodule
```

Write operation(1/4)

- n-to-2^n decoder
  - 입력값을 이용해 알맞은 출력으로 변환

Inputs			outputs							
i[2]	i[1]	i[0]	o[7]	o[6]	o[5]	o[4]	o[3]	o[2]	o[1]	o[0]
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Write operation(2/4)

- 3-to-8 decoder

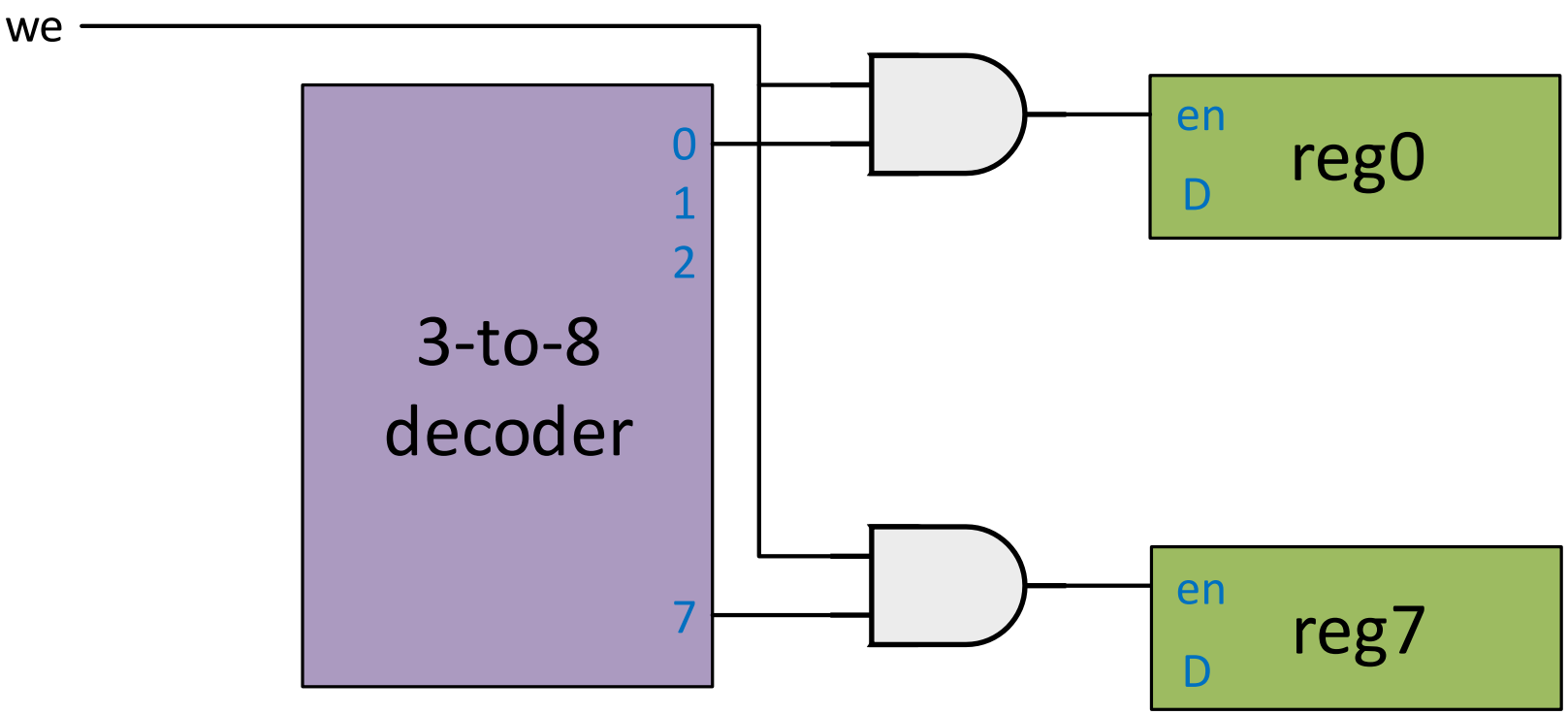
```
module _3_to_8_decoder (d, q);//3to 8 decoder module
  input      [2:0] d;
  output reg [7:0] q;

  always@(d) begin
    case(d)
      3'b000: q =
      3'b001: q =
      3'b010: q =
      3'b011: q =
      3'b100: q =
      3'b101: q =
      3'b110: q =
      3'b111: q =
      default : q = 8'hx;
    endcase
  end
endmodule
```

wAddr

Assign appropriate Output q

Write operation(3/4)

- Decoder에서 받은 값으로 해당 register에 en signal 보냄
  - we (write enable)이 1이어야 register에 값을 쓸 수 있다!



Write operation(4/4)

```
module write_operation(Addr, we, to_reg);
       input               we;
       input  [2:0]  Addr;
       output [7:0]  to_reg;

       wire   [7:0]  w_a;
```

Instance of decoder, 2-input and gates(#8)

```
endmodule
```

Read operation(1/3)

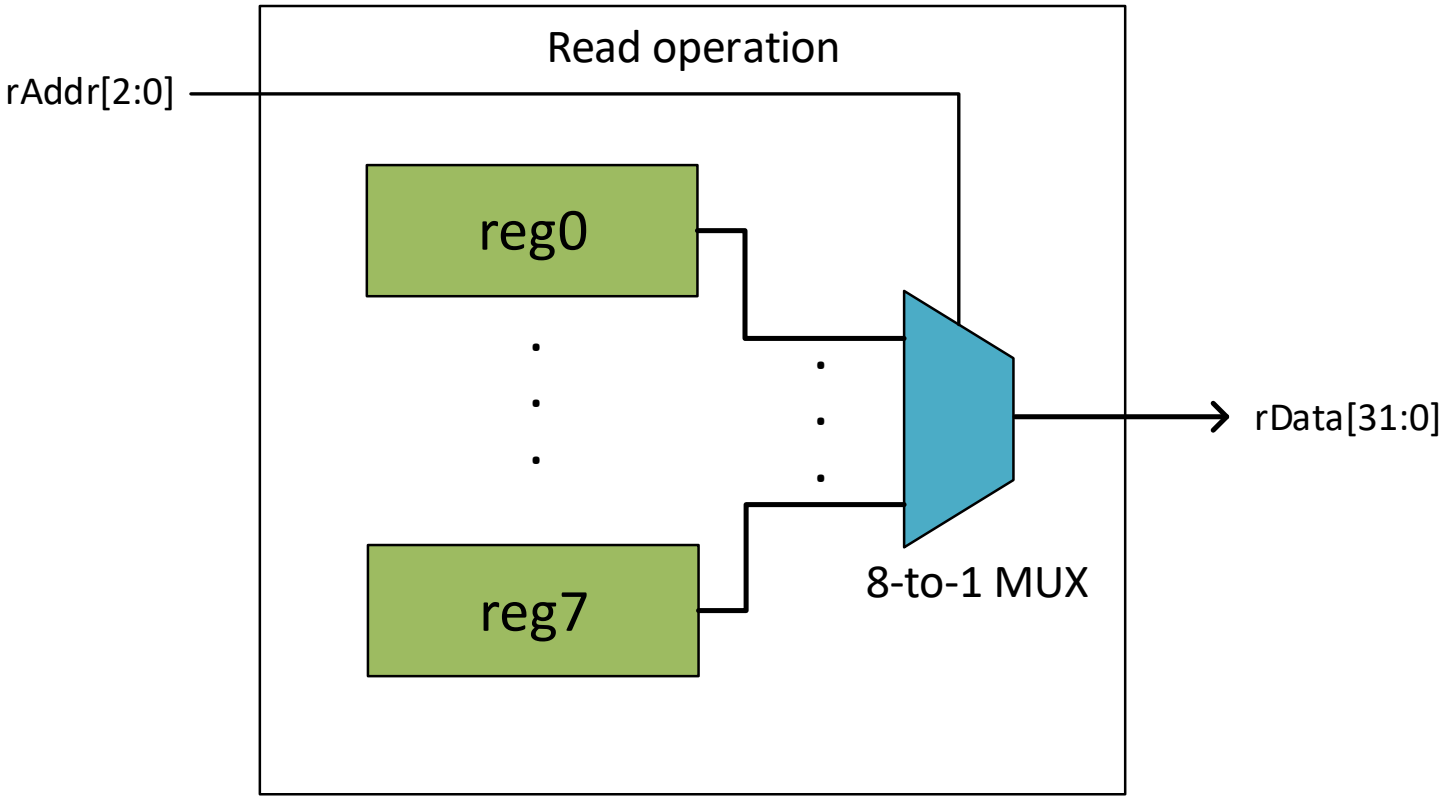
- 8-to-1 MUX를 이용하여 8개의 register 중 1개의 register 출력을 얻어냄

```
module _8_to_1_MUX(a, b, c, d, e, f, g, h, sel, d_out);
  input                  [31:0] a, b, c, d, e, f, g, h;
  input                  [2:0]  sel;
  output reg             [31:0] d_out;


  always@(sel, a, b, c, d, e, f, g, h) begin
    case(sel)
      3'b000       : d_out
      3'b001       : d_out
      3'b010       : d_out
      3'b011       : d_out
      3'b100       : d_out
      3'b101       : d_out
      3'b110       : d_out
      3'b111       : d_out
       default     : d_out
  endcase
end
endmodule
```

Assign appropriate Output d_out

Read operation(2/3)

- MUX에 의해 선택된 값은 출력됨



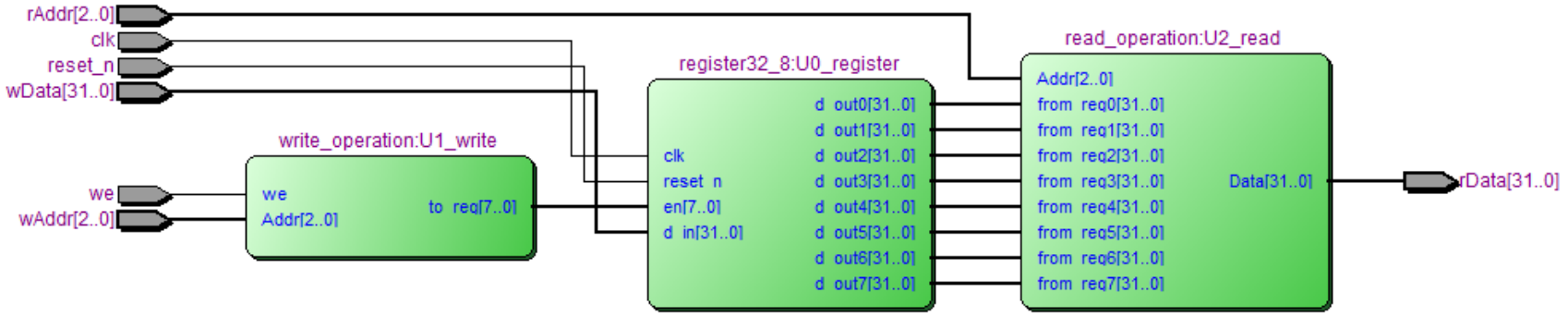
Read operation(3/3)

```
module read_operation (Addr, Data, from_reg0, from_reg1, from_reg2, from_reg3,
      from_reg4, from_reg5, from_reg6, from_reg7);
  input      [31:0] from_reg0, from_reg1, from_reg2, from_reg3,
                    from_reg4, from_reg5, from_reg6, from_reg7;
  input      [2:0]         Addr;
  output     [31:0] Data;
```

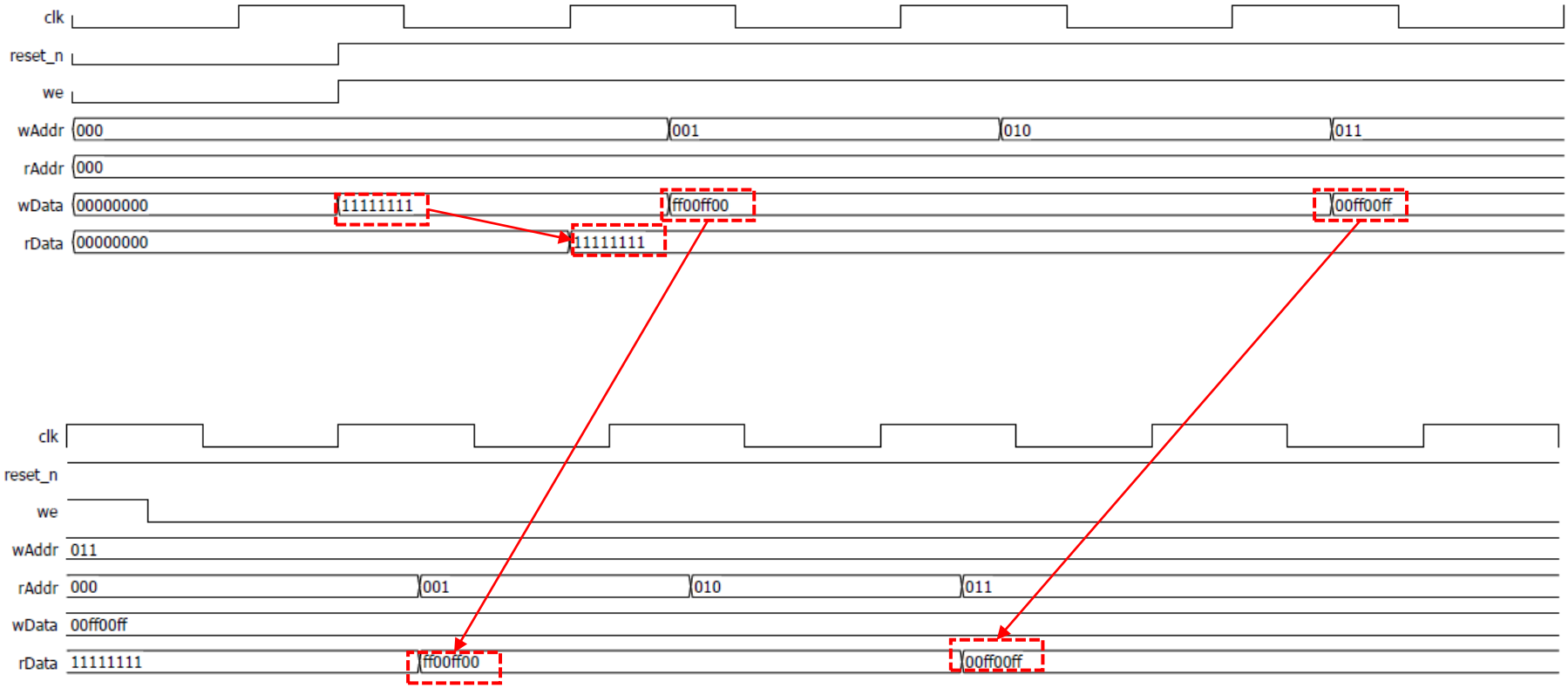
Instance of mux

```
endmodule
```

RTL viewer


rAddr[2..0]
clk
reset_n
wData[31..0]
write_operation:U1_write
we
wAddr[2..0]
we
Addr[2..0]
to reg[7..0]
register32_8:U0_register
clk
reset_n
en[7..0]
d in[31..0]
d out0[31..0]
d out1[31..0]
d out2[31..0]
d out3[31..0]
d out4[31..0]
d out5[31..0]
d out6[31..0]
d out7[31..0]
read_operation:U2_read
Addr[2..0]
from reg0[31..0]
from reg1[31..0]
from reg2[31..0]
from reg3[31..0]
from reg4[31..0]
from reg5[31..0]
from reg6[31..0]
from reg7[31..0]
Data[31..0]
rData[31..0]

Testbench


clk
reset_n
we
wAddr
000
001
010
011
rAddr
000
wData
00000000
11111111
ff00ff00
00ff00ff
rData
00000000
11111111
clk
reset_n
we
wAddr
011
rAddr
000
001
010
011
wData
00ff00ff
rData
11111111
ff00ff00
00ff00ff

Verification

- Testbench
  - 작성한 module에 대하여 testbench를 작성하여 ModelSim에서 검증 수행
- RTL Viewer
  - 확인 후 레포트에 이에 대하여 정리한다.
- Flow Summary
  - 확인 후 레포트에 보고한다.

Assignment 7

- Report
  - ✓ 자세한 사항은 lab document 참고
- Submission
  - ✓ Soft copy
    - 강의 당일 후 1주까지(delay 2 days 20%감점)
    - 실습 미수강은 디지털 논리2 조교 공지에 따름

References

- Altera Co., www.altera.com/
- D. M. Harris and S. L. Harris, Digital Design and Computer Architecture, Morgan Kaufmann, 2007
- 이준환, 디지털논리회로2 강의자료, 광운대학교, 컴퓨터 공학과, 2019

Q&A

THANK YOU