Register Set Design Example (1-bit & 2-bit)

Nahin Ul Sadad
Lecturer
CSE, RUET

Register Set in CPU

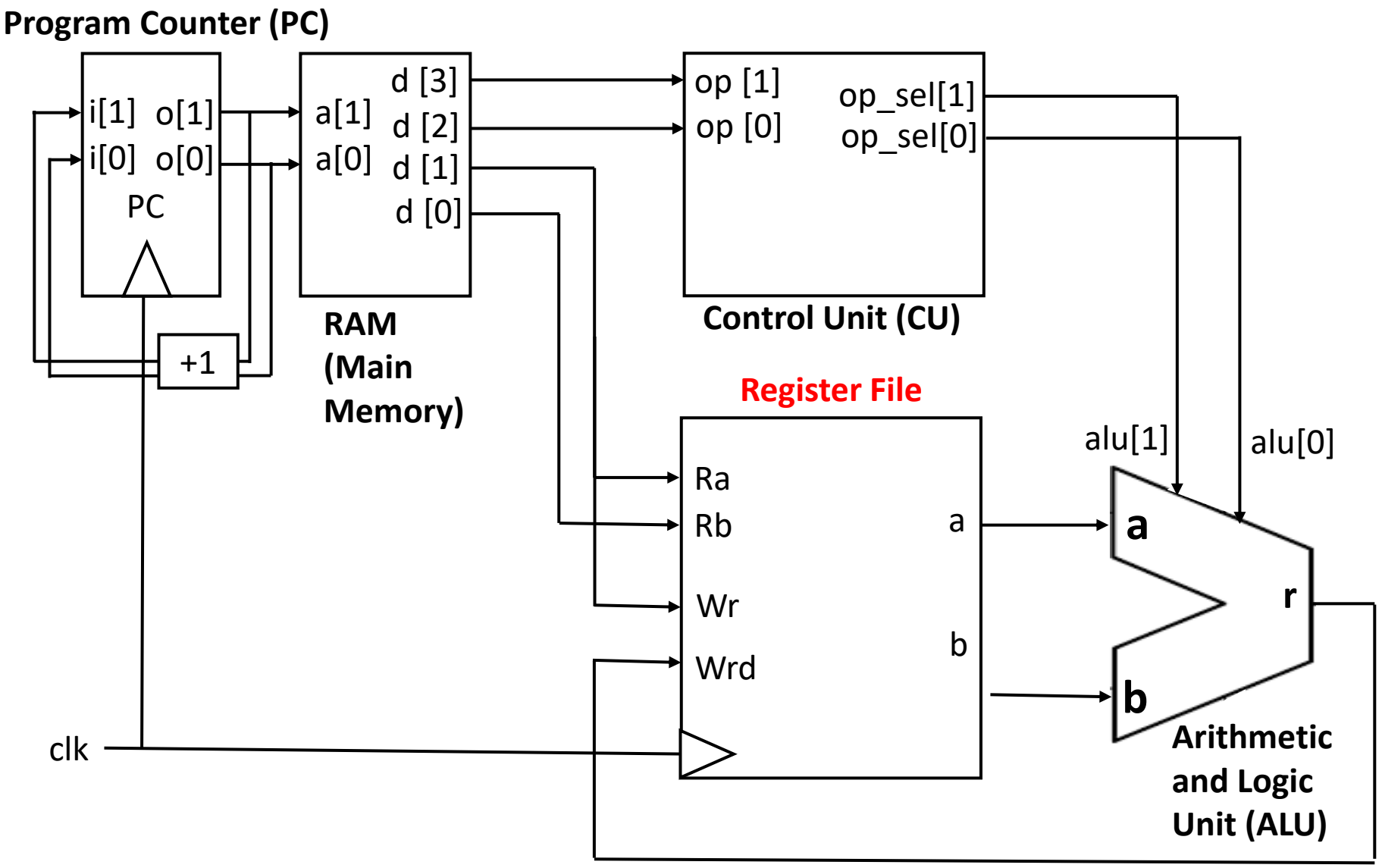


Figure: 1-bit CPU

1. Program Counter will have address of next instruction to be executed in current clock cycle.
2. Address in PC will be sent to RAM to retrieve instruction.
3. Instruction will be decoded by control unit and will select **registers** and/or immediate values.
4. Data within **registers** and/or immediate values will be sent to Arithmetic and Logic Unit (ALU) to perform operations.
5. ALU will perform operation and result will be sent to the **register** to be written.
6. Finally, PC will be incremented to point to the next instruction in next clock cycle.

Prerequisites for Register Design

D Flip-flop

D	Q(t + 1)
0	0
0	1

Fig: D Flip-flop truth table

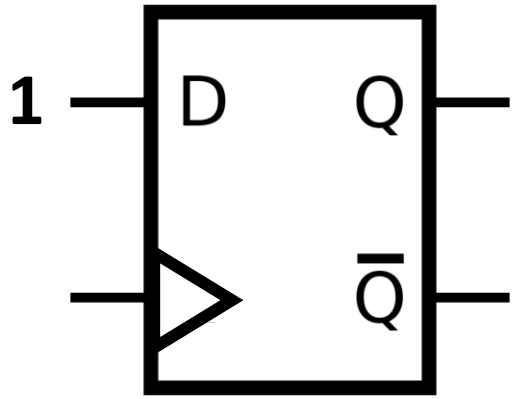


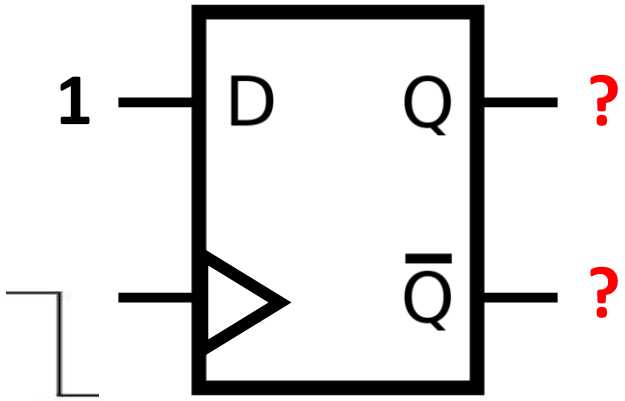
Fig: D Flip-Flop

D Flip-Flop

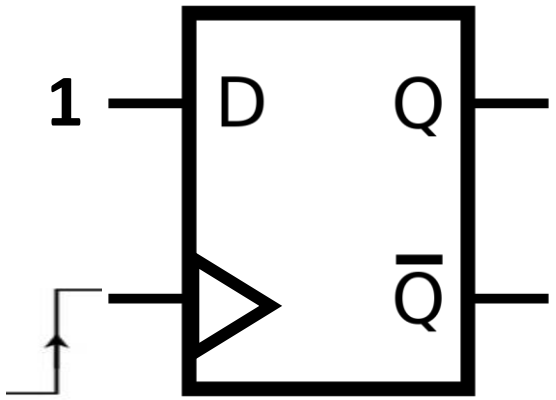
Assume, Q=1 which means D-FF has 1 value stored.

D	Q(t + 1)
0	0
0	1

Fig: D Flip-flop truth table



At Negative Clock Edge



At Positive Clock Edge

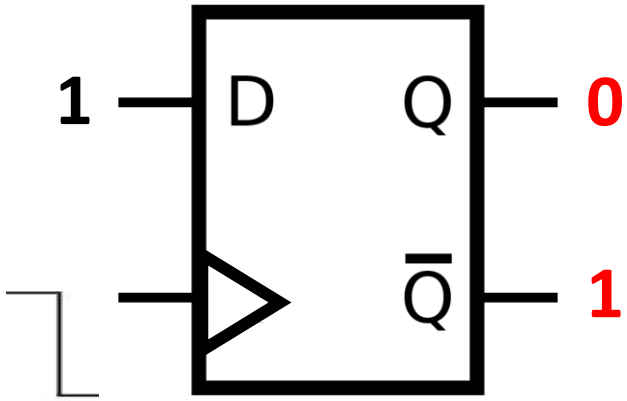
Fig: How D Flip-Flop works.

D Flip-Flop

Assume, Q=1 which means D-FF has 1 value stored.

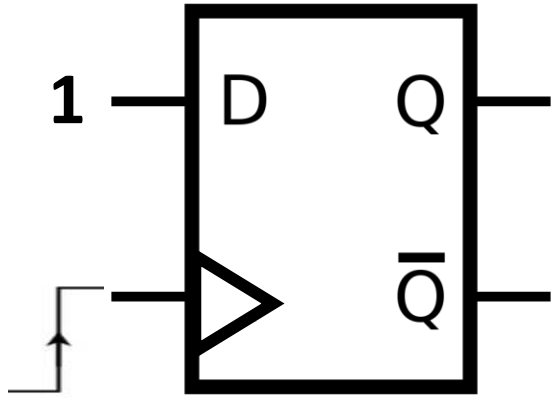
D	Q(t + 1)
0	0
0	1

Fig: D Flip-flop truth table



At Negative Clock Edge

D Flip-flop will not update its content because clock is not on positive edge.



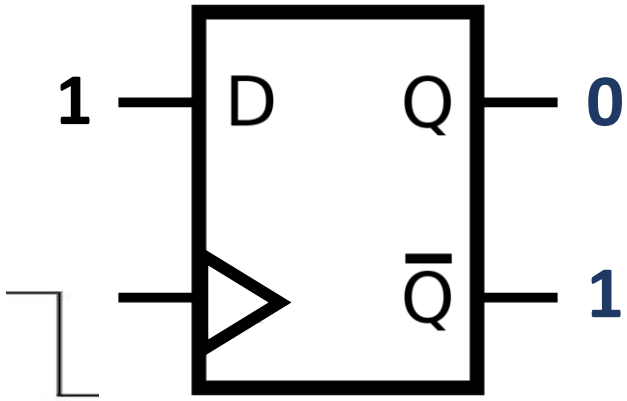
At Positive Clock Edge

Fig: How D Flip-Flop works.

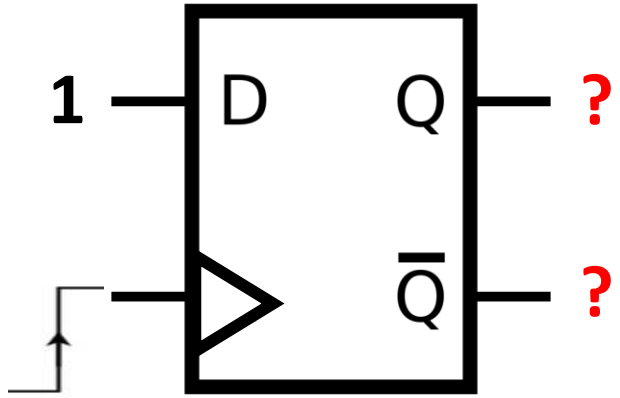
D Flip-Flop

D	Q(t + 1)
0	0
0	1

Fig: D Flip-flop truth table



At Negative Clock Edge



At Positive Clock Edge

Fig: How D Flip-Flop works.

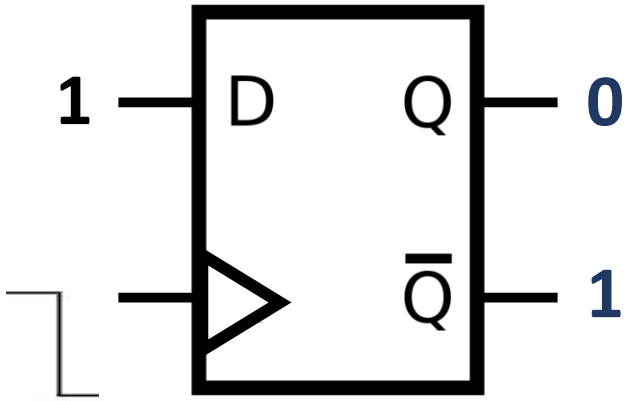
Assume, Q=1 which means D-FF has 1 value stored.

D Flip-flop will not update its content because clock is not on positive edge.

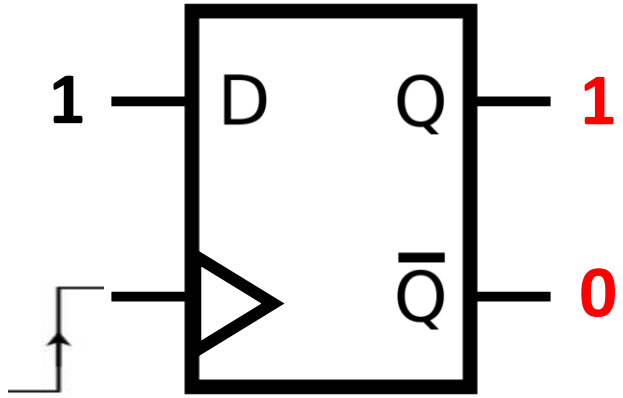
D Flip-Flop

D	Q(t + 1)
0	0
0	1

Fig: D Flip-flop truth table



At Negative Clock Edge



At Positive Clock Edge

Fig: How D Flip-Flop works.

Assume, Q=1 which means D-FF has 1 value stored.

D Flip-flop will not update its content because clock is not on positive edge.

D Flip-flop will update its content because clock is on positive edge.

Register Basics

Register

Register is a very fast computer memory, used to store data/instruction in-execution.

A Register is a group of flip-flops with each flip-flop capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing binary information of n-bits.

There are several types of registers:

1. General Purpose Registers
2. Data Registers
3. Address Registers
4. Condition Codes/FLAG Registers

Register

There are several types of registers:

1. **General Purpose Registers:** General-purpose registers can be assigned to a variety of functions by the programmer.
2. **Data Registers:** Data registers can be used only to hold data and cannot be employed in the calculation of an operand address.
3. **Address Registers:** Address registers can be general purpose or can be devoted to a particular addressing mode.
    a. **Segment Pointer Registers:** In a machine with segmented addressing, a segment register holds the address of the base of the segment. There may be multiple registers: one for the operating system and one for the current process.
    b. **Index Registers:** Index registers are used for indexed addressing and may be auto-indexed.
    c. **Stack Pointer Register:** If there is user-visible stack addressing, then typically there is a dedicated register that points to the top of the stack. This allows implicit addressing which means that push, pop, and other stack instructions need not contain an explicit stack operand.

Register

4. **Condition Codes/FLAG Registers:** Condition codes/FLAG register which is at least partially visible to the user holds condition codes (also referred to as flags). Condition codes are bits set by the processor hardware as the result of operations.

   In addition to the result itself being stored in a register or memory, a condition code is also set. The code may subsequently be tested as part of a conditional branch operation.

   For example, an arithmetic operation may produce a positive, negative, zero, or overflow result.

   Common fields or flags include the following:

   a. **Sign:** It contains the sign bit of the result of the last arithmetic operation.

   b. **Zero:** It sets when the result is 0.

   c. **Carry:** It sets if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high-order bit. Used for multiword arithmetic operations.

   d. **Equal:** It sets if a logical compare result is equality.

Register

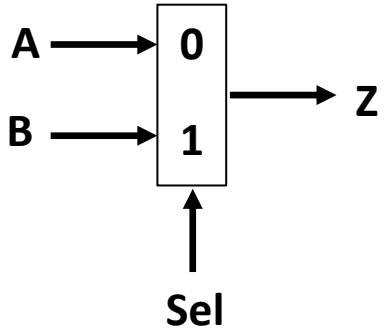
e. **Overflow:** It is used to indicate arithmetic overflow.

f. **Interrupt Enable/Disable:** It is used to enable or disable interrupts.

g. **Supervisor:** It indicates whether the processor is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode and certain areas of memory can be accessed only in supervisor mode.

1-bit Register Design

1-bit Register



2 to 1 MUX

Sel	Z
0	A
1	B

$$Z = \overline{Sel}.A + Sel.B$$

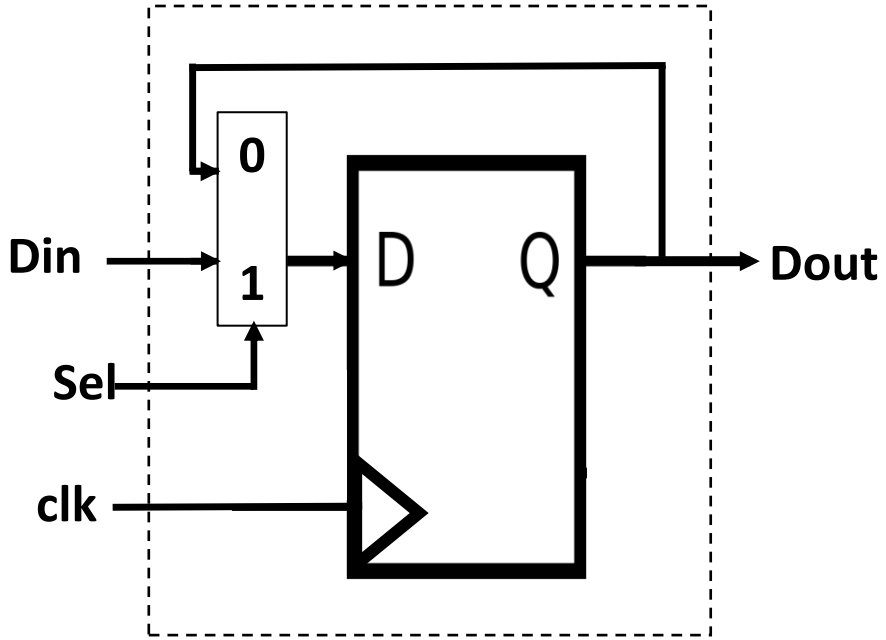
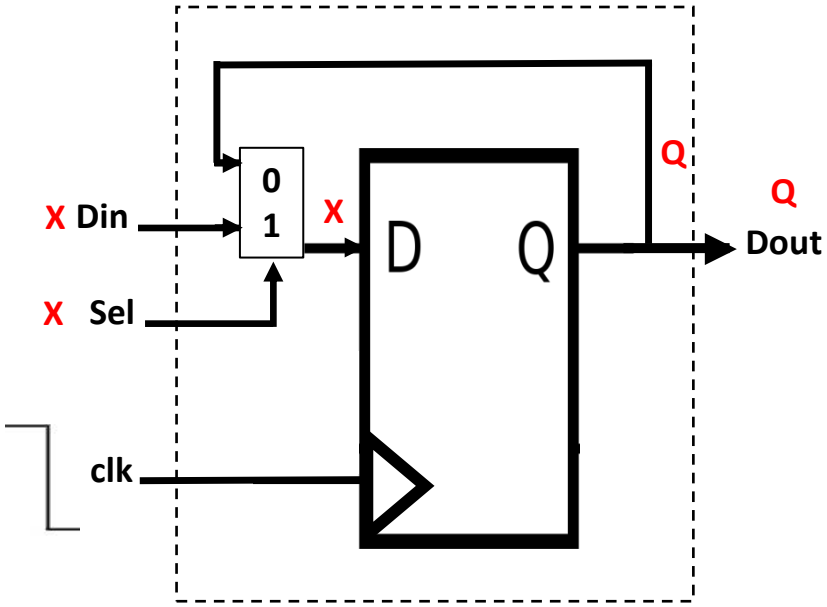


Fig: 1-bit Register

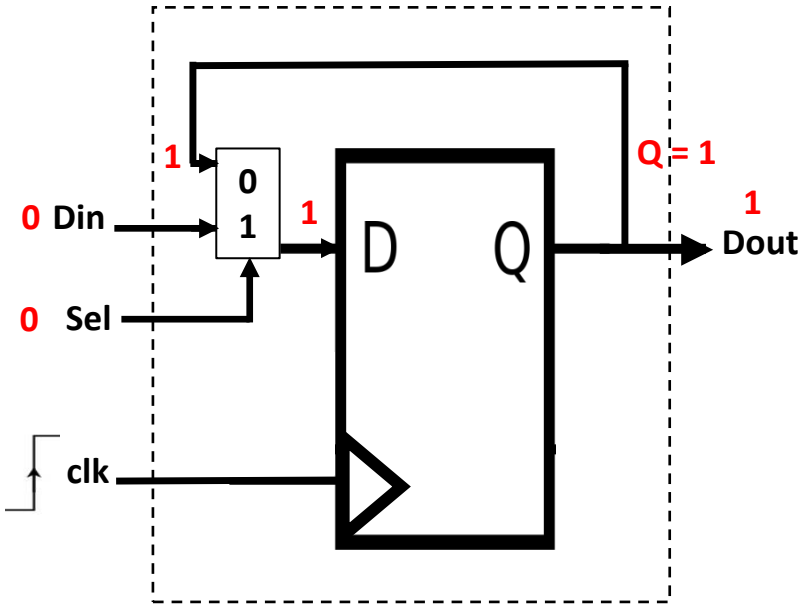
1-bit Register



At negative clock cycle,
D Flip-flop will not update its content regardless of Din and Sel value.

Fig: How 1-bit Register Works

1-bit Register

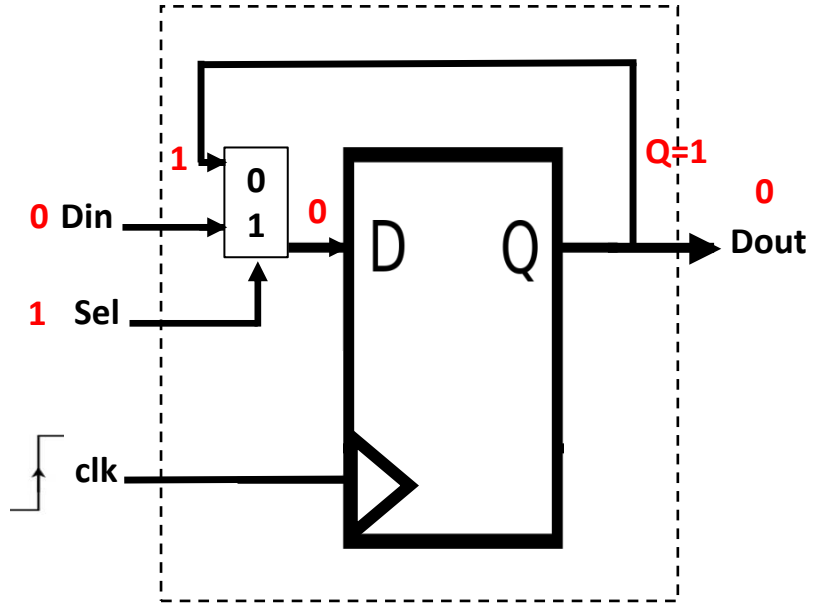


At positive clock cycle,
D Flip-flop will update its content.

When $\mathbf{Sel} = 0, \mathbf{D} = \mathbf{Q}$
So, Data will remain same in D-FF.

Fig: How 1-bit Register Works

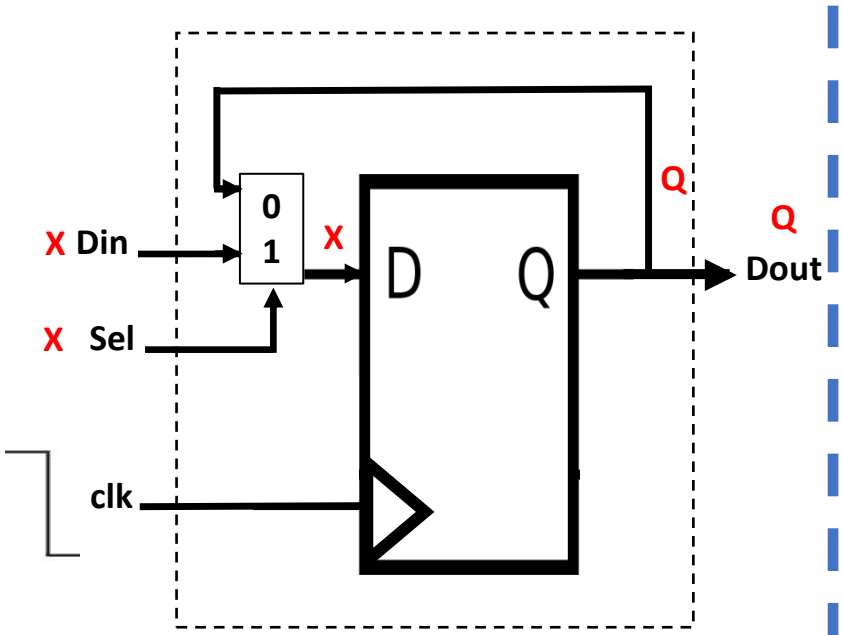
1-bit Register



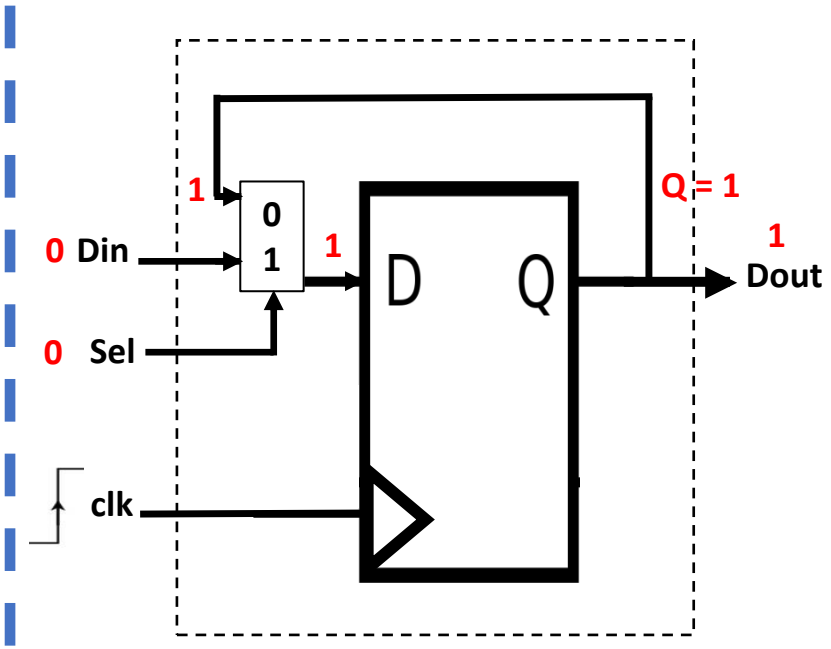
At positive clock cycle,
D Flip-flop will update its content.

When $\mathbf{Sel} = 0, \mathbf{D} = \mathbf{Din}$
So, Data will be updated in D-FF.

1-bit Register

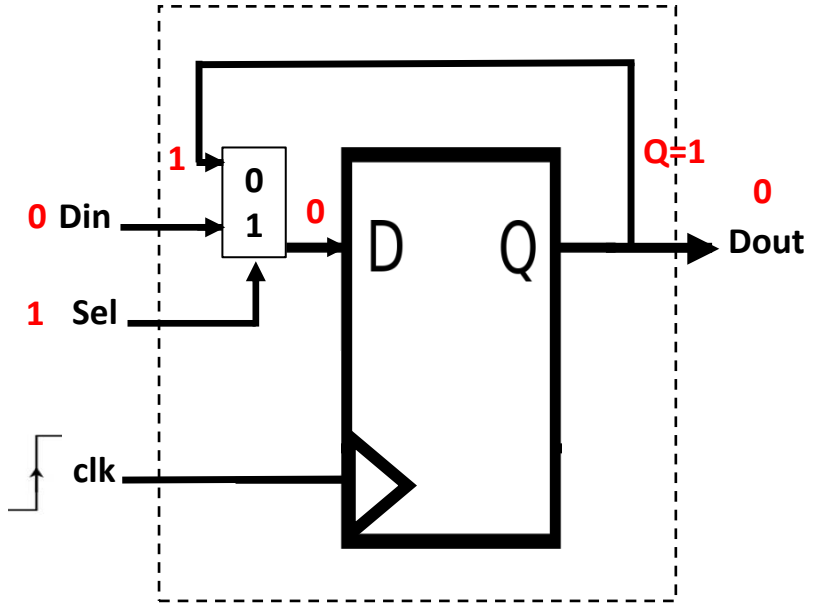


At negative clock cycle,
D Flip-flop will not update its content regardless of Din and Sel value.



At positive clock cycle,
D Flip-flop will update its content.

When $\mathbf{Sel} = 0, \mathbf{D} = \mathbf{Q}$
So, Data will remain same in D-FF.



At positive clock cycle,
D Flip-flop will update its content.

When $\mathbf{Sel} = 0, \mathbf{D} = \mathbf{Din}$
So, Data will be updated in D-FF.

Fig: How 1-bit Register Works

1-bit Register

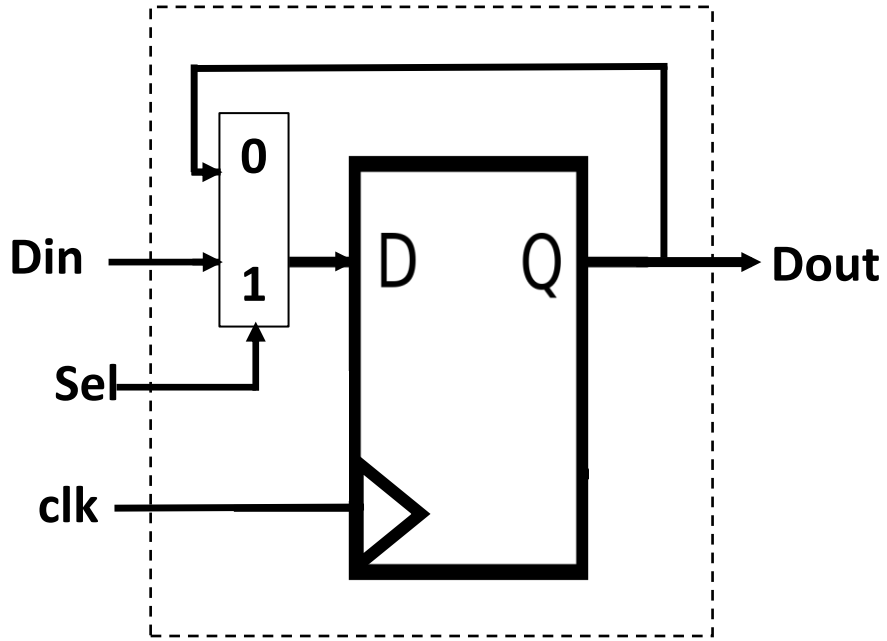


Fig: 1-bit Register

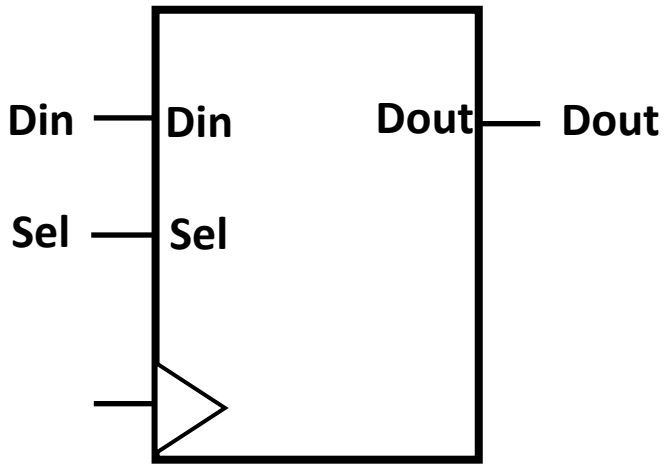


Figure: 1 bit register chip

2-bit Register Design

2-bit Register

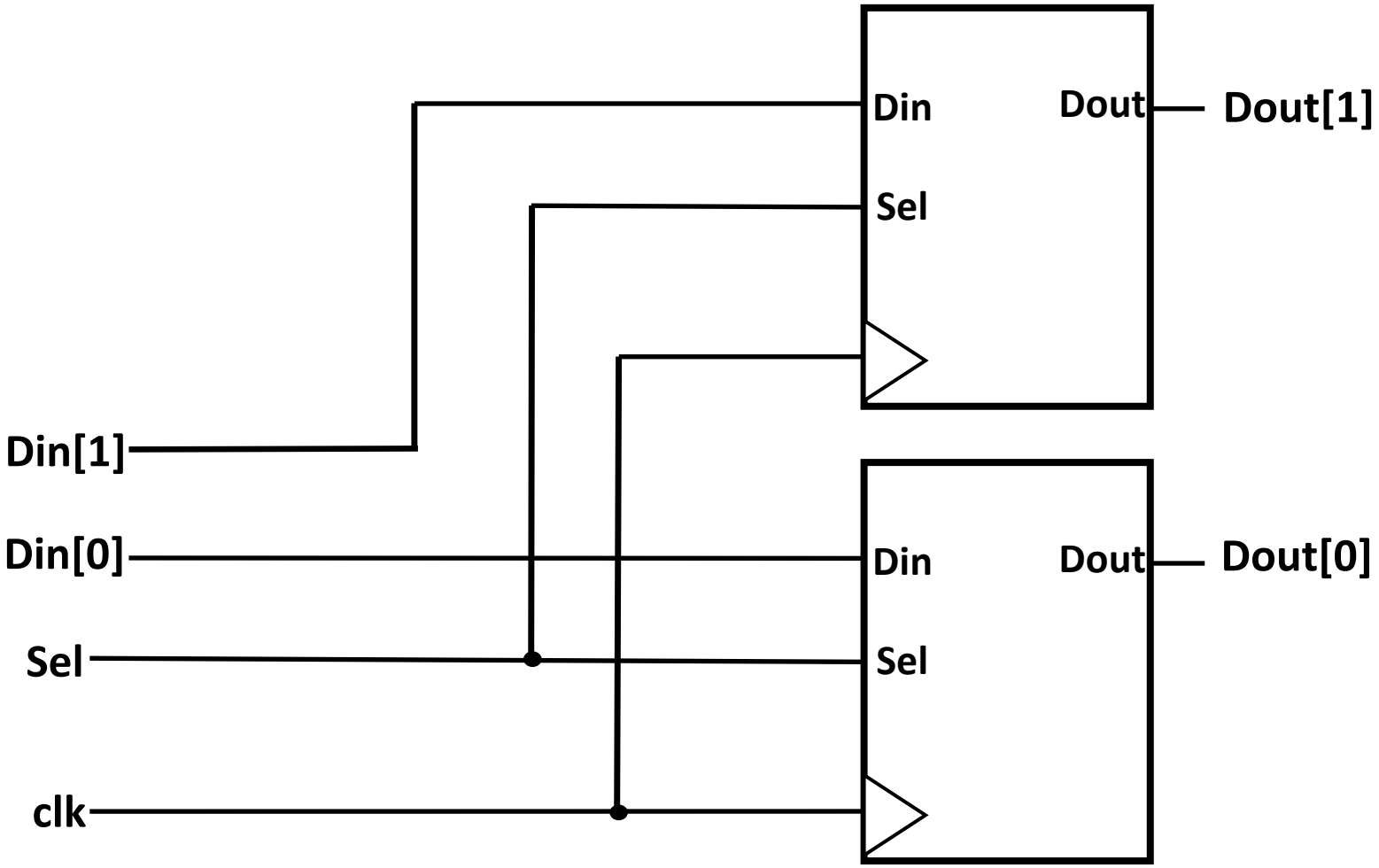


Figure: 2 bit Register

2-bit Register

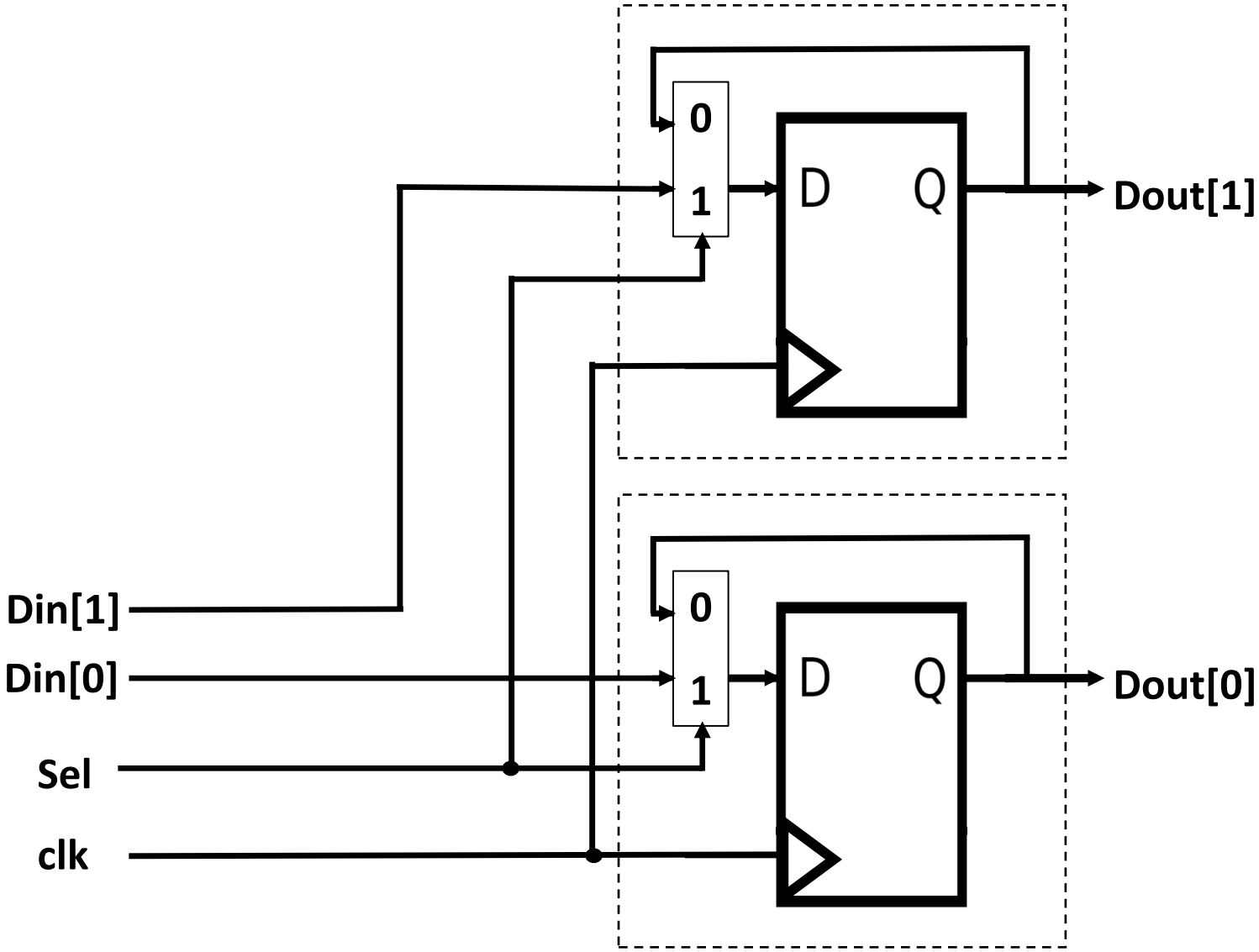


Figure: Inside of 2 bit Register

2-bit Register

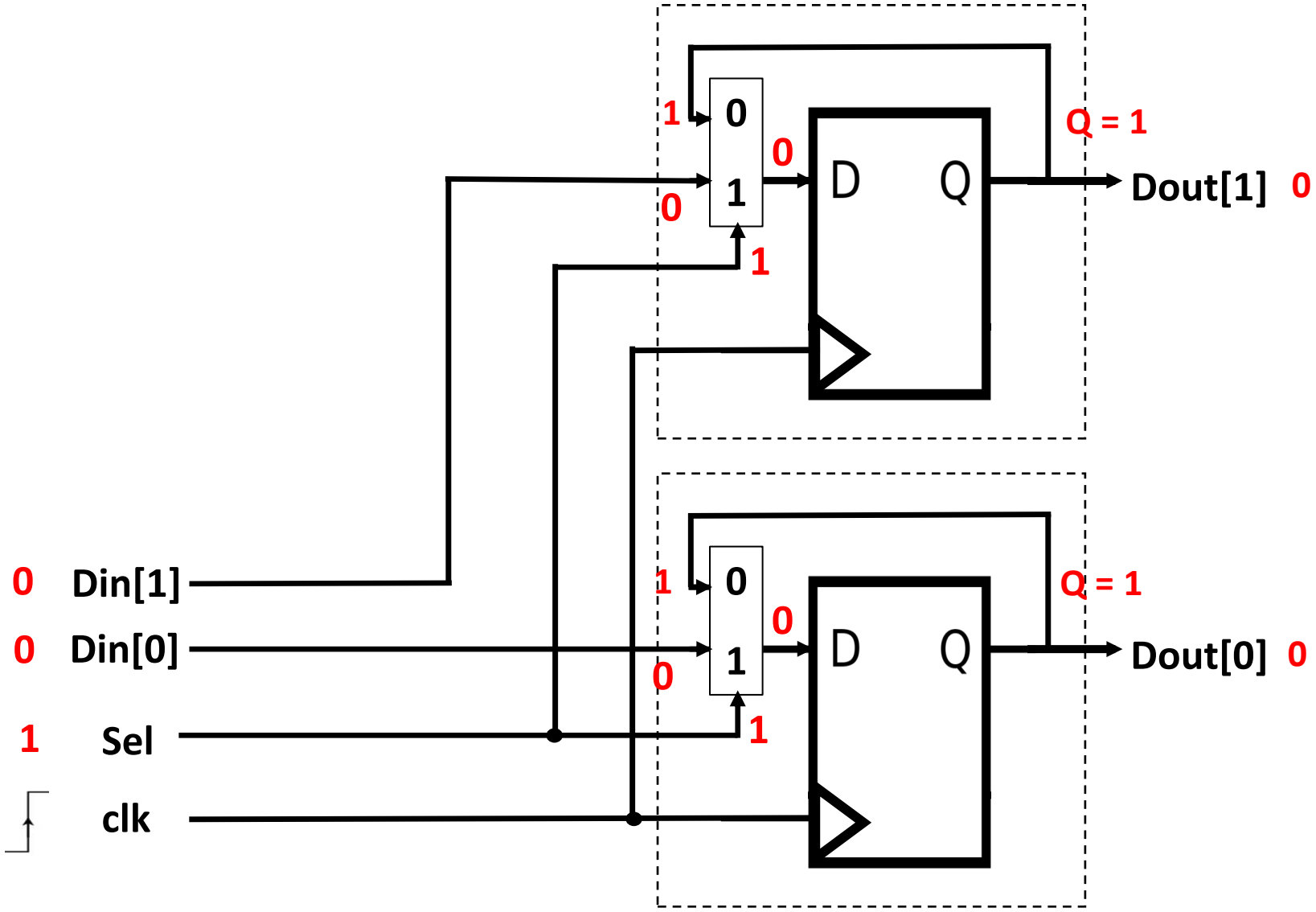


Figure: How 2 bit Register Works

2-bit Register

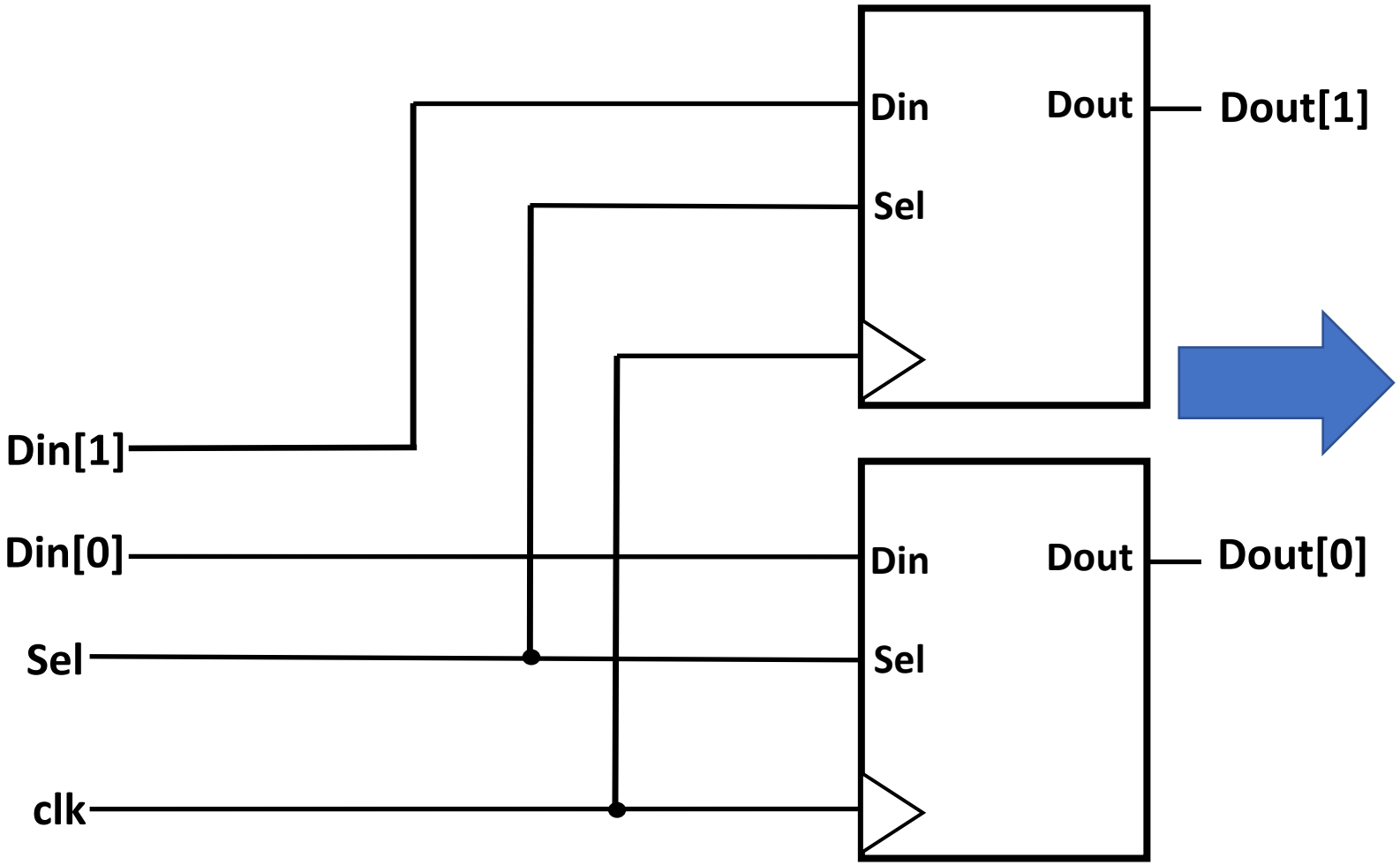


Figure: 2 bit Register

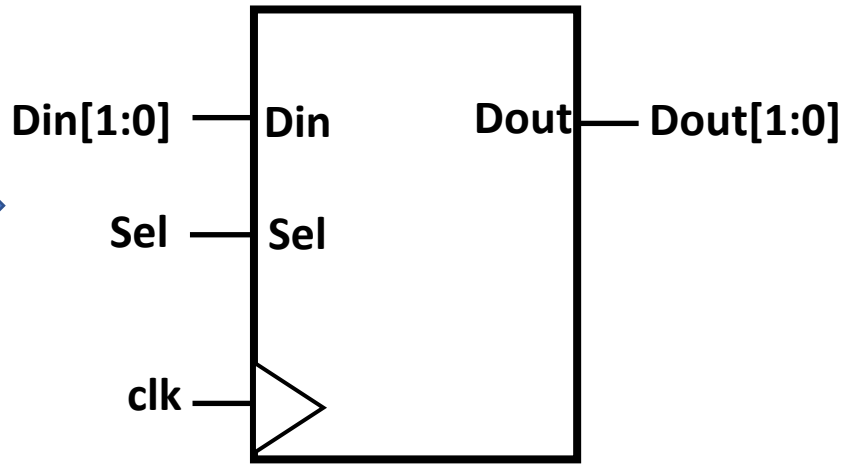


Figure: 2 bit register chip

4-bit Register Design

4-bit Register

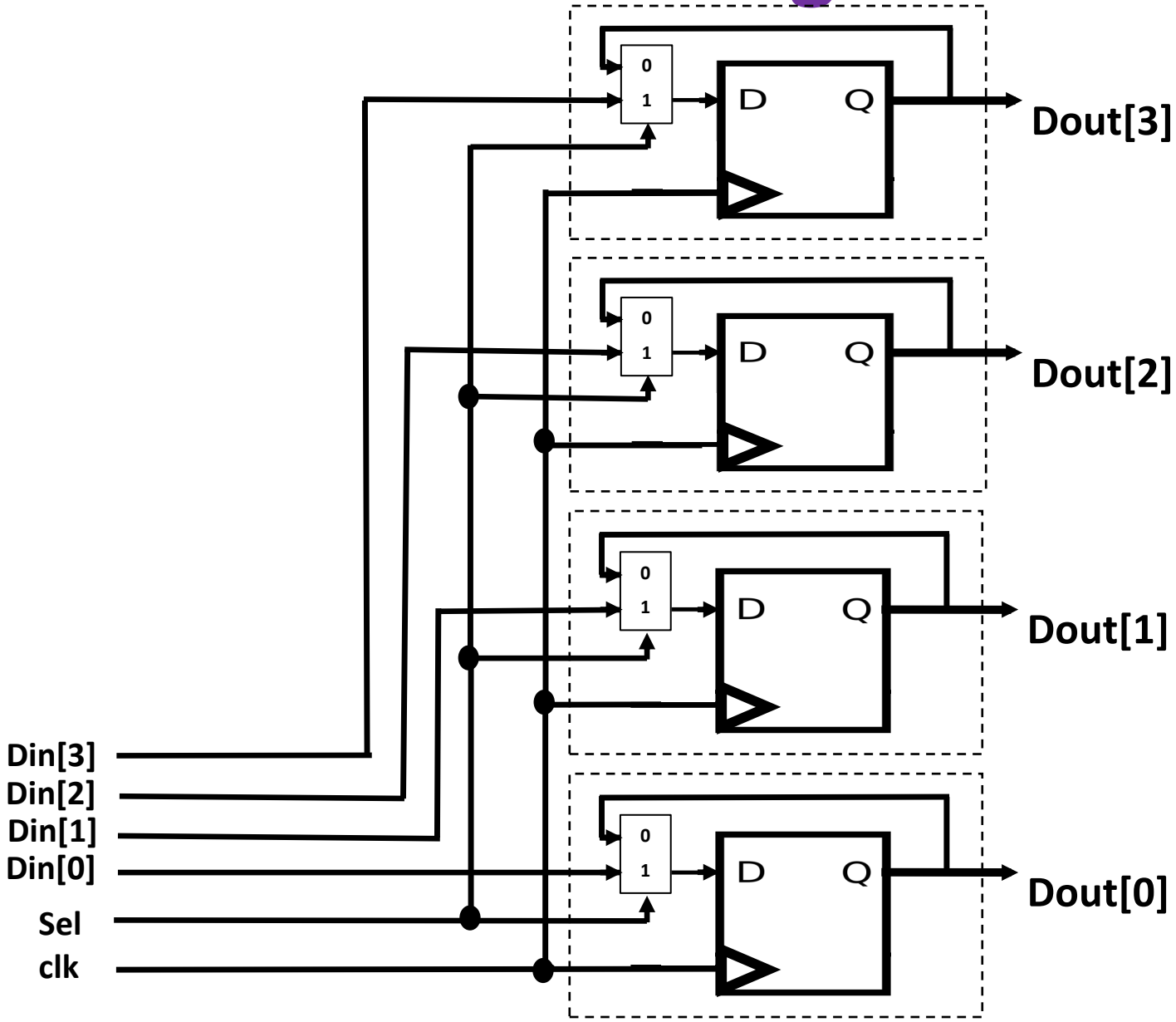


Figure: Inside of 4 bit Register

4-bit Register

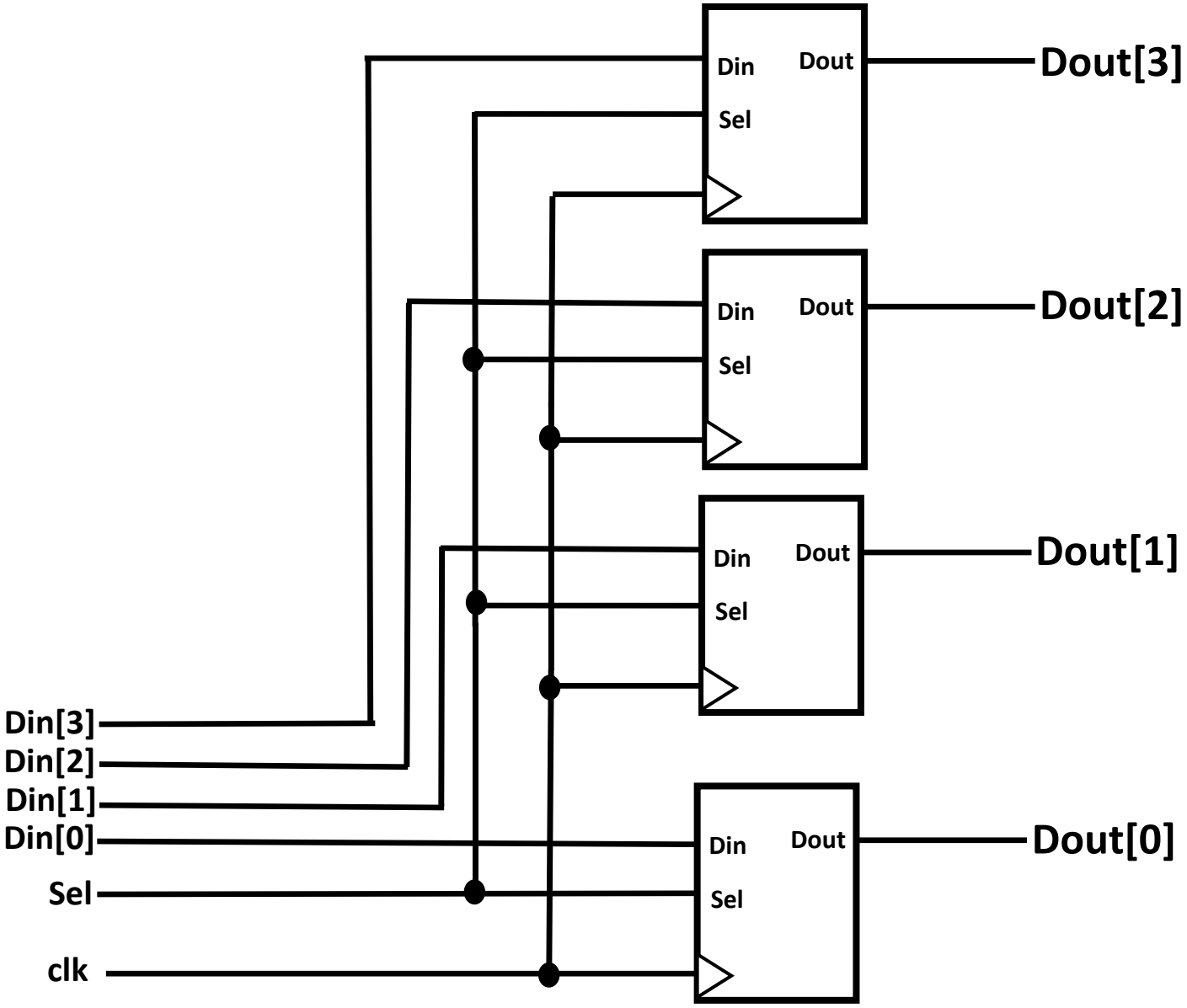


Figure: 4 bit Register

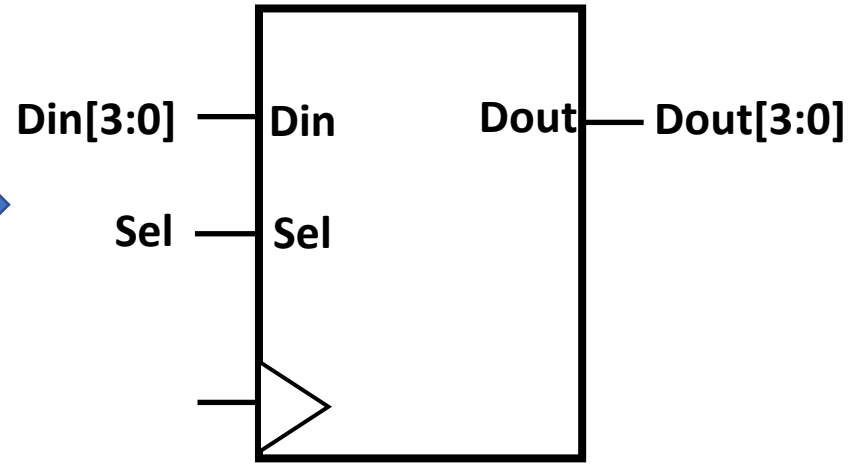


Figure: 4 bit register chip

Example: Register Design

Question: Design a 7-bit Register.

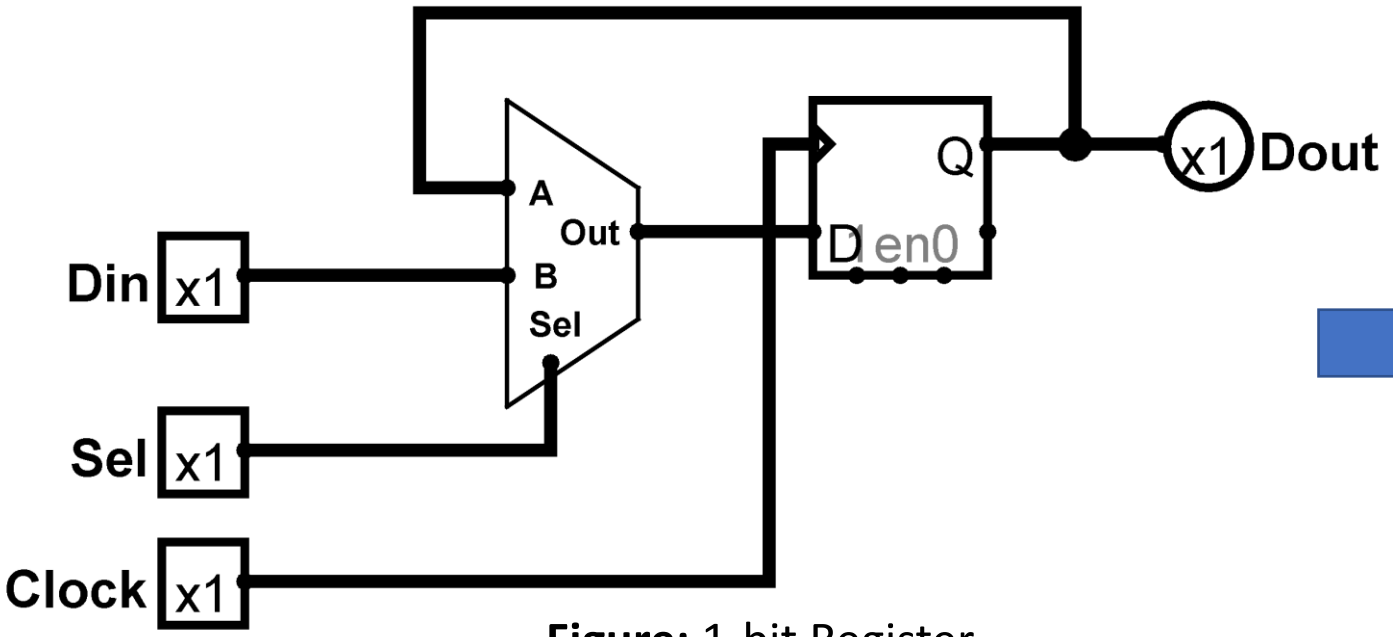


Figure: 1-bit Register

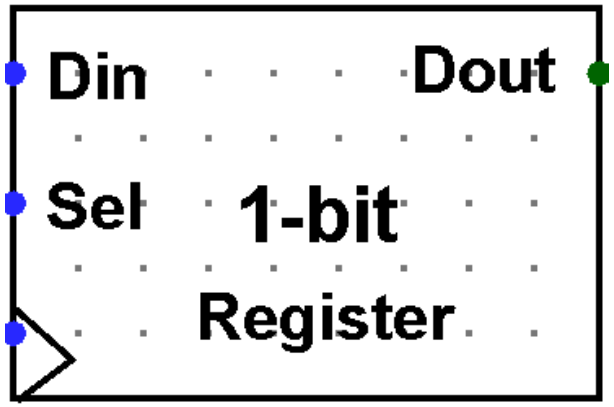


Figure: 1-bit Register Chip

Example: Register Set Design

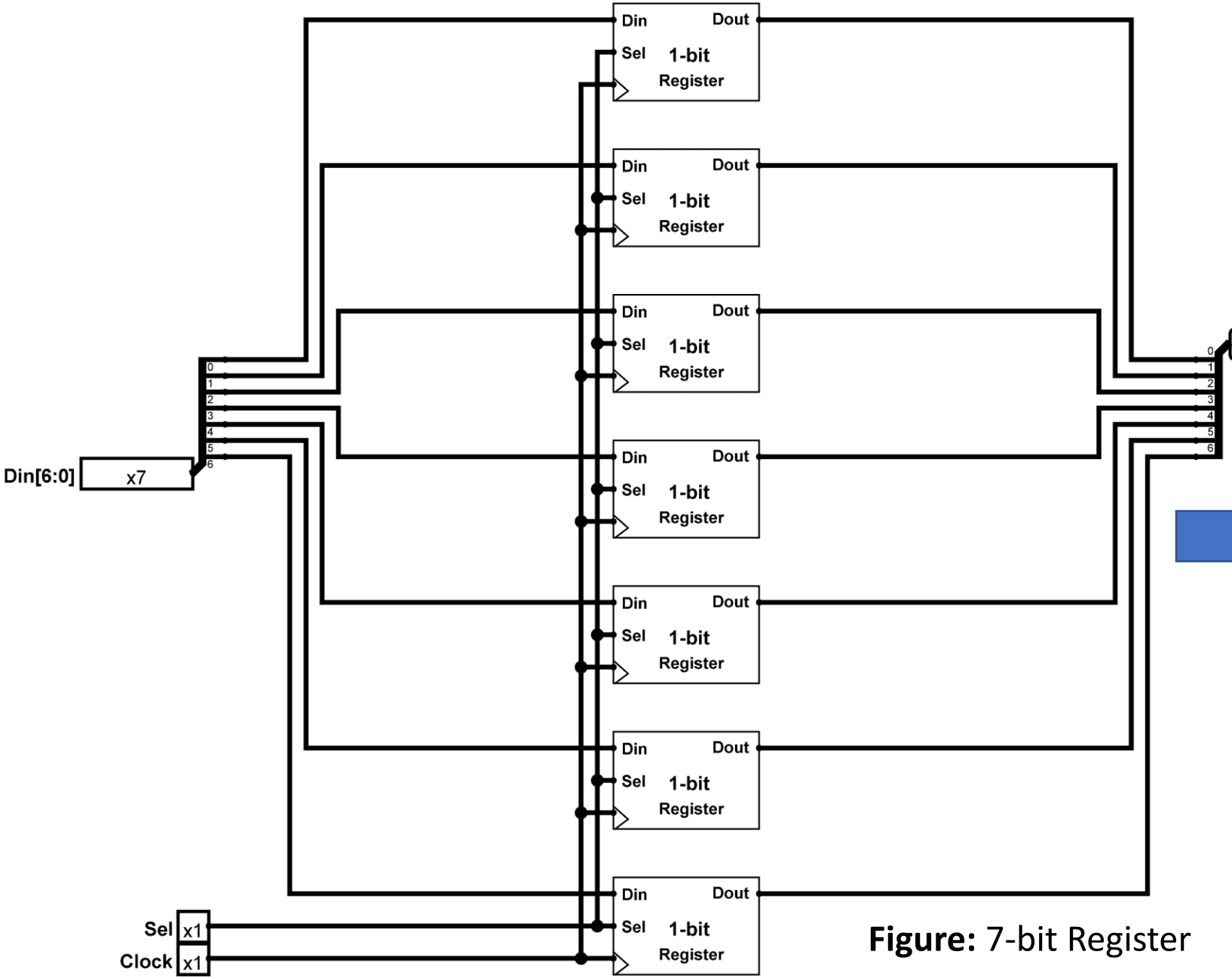


Figure: 7-bit Register

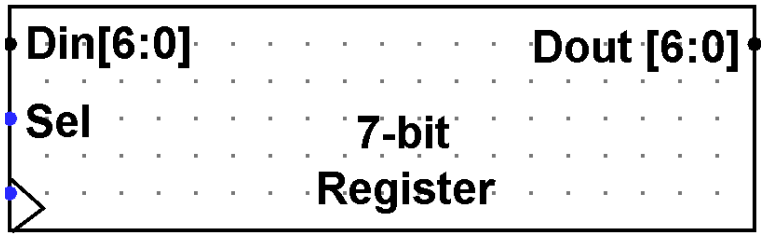


Figure: 7-bit Register Chip

Importance of Register Set Design

Register Set

Register Set contains multiple registers for temporary storages which can be read and write.

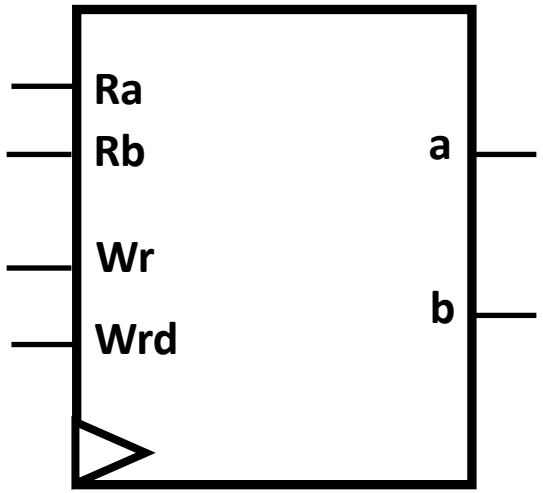


Figure: Register File

1-bit CPU

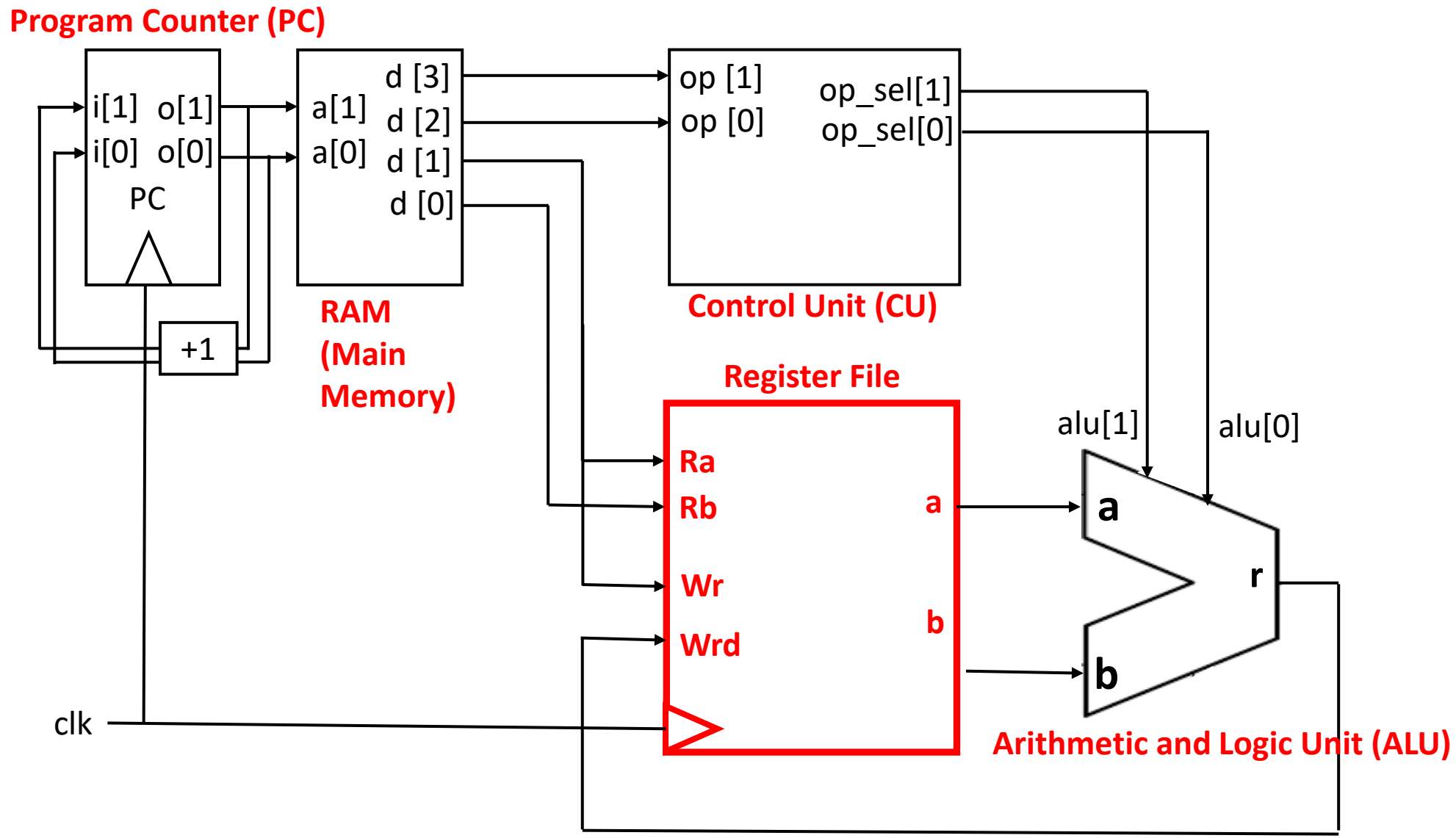


Figure: 1-bit CPU

Register Set Design

In Assembly Language,

Suppose **`ADD R0, R1  (R0 = R0 + R1)`**

Here, We will be adding contents of **`R0`** and **`R1`** and store that result in **`R0`**. It depends on design.

In order to do this, We first have to select:

Registers to be read: `R0` and `R1`

Register to be written: `R0`

Register Set Design

`ADD R0, R1`

Registers to be read: `R0` and `R1`

Register to be written: `R0`

Ra and Rb will select registers whose data to be read.

Wr will select register to be written. Wrd is data to be written in register selected by Wr.

Value within register R0 will be sent to a.
Value within register R1 will be sent to b.

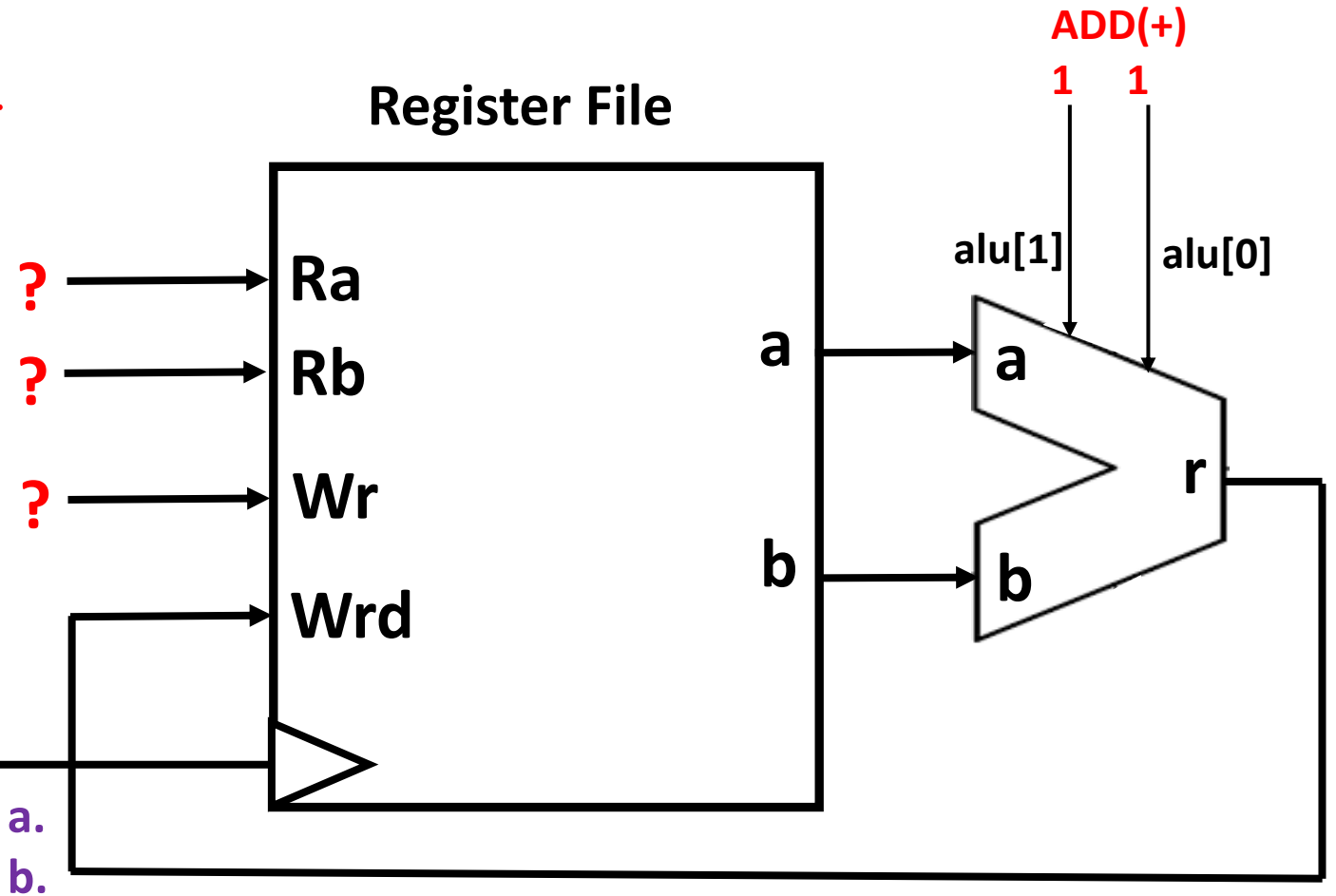


Figure: How Register set operates in CPU

Register Set Design

ADD R0, R1 (R0 = R0 + R1)

Registers to be read: R0 and R1

Register to be written: R0

Ra and Rb will select registers whose data to be read.

Wr will select register to be written.
Wrd is data to be written in register selected by Wr.

Value within register R0 will be sent to a.
Value within register R1 will be sent to b.

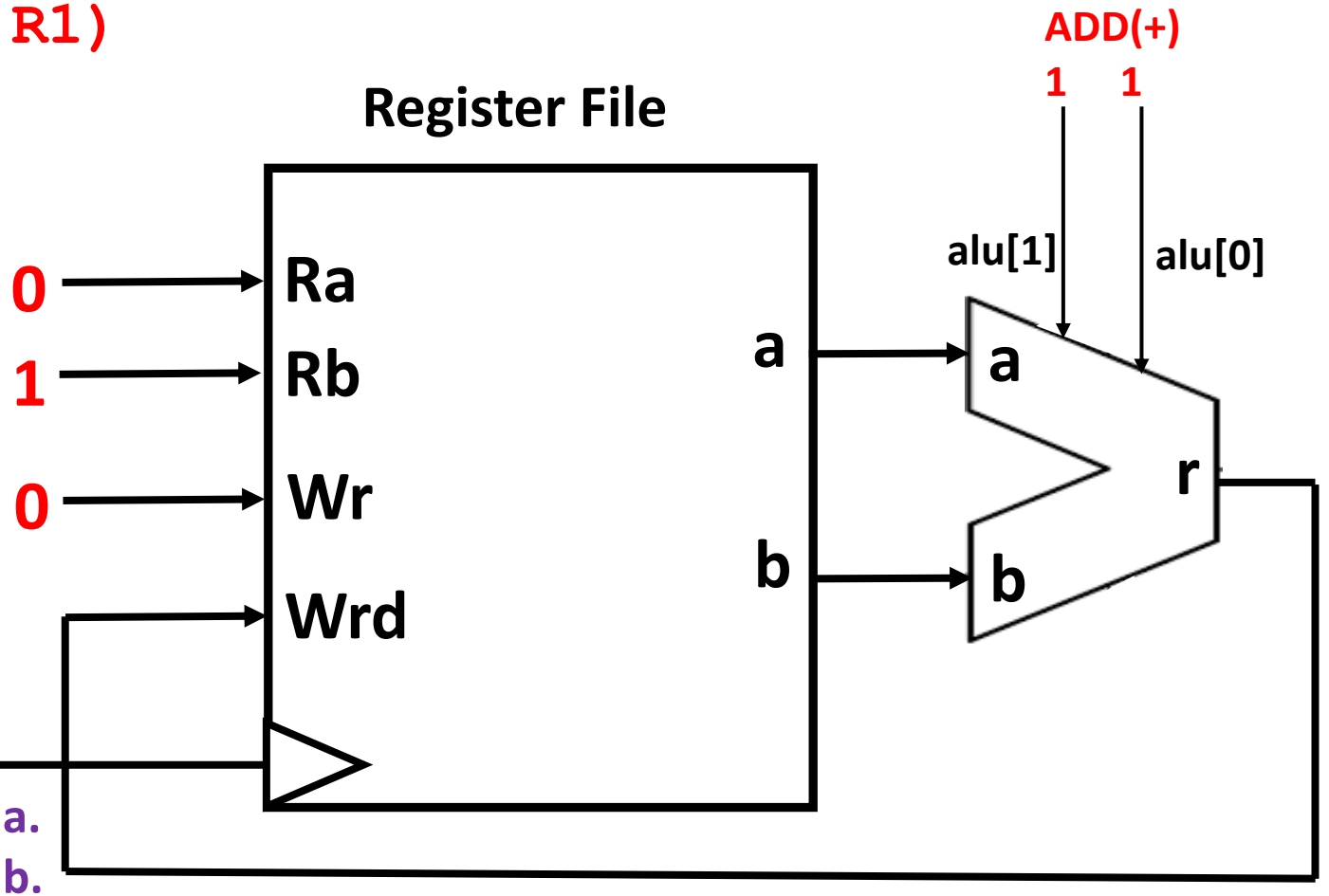


Figure: How Register set operates in CPU

Register Set Design

ADD R0, R1 (R0 = R0 + R1)

Registers to be read: R0 and R1

Register to be written: R0

Ra and Rb will select registers whose data to be read.

Wr will select register to be written. Wrd is data to be written in register selected by Wr.

Value within register R0 will be sent to a.
Value within register R1 will be sent to b.

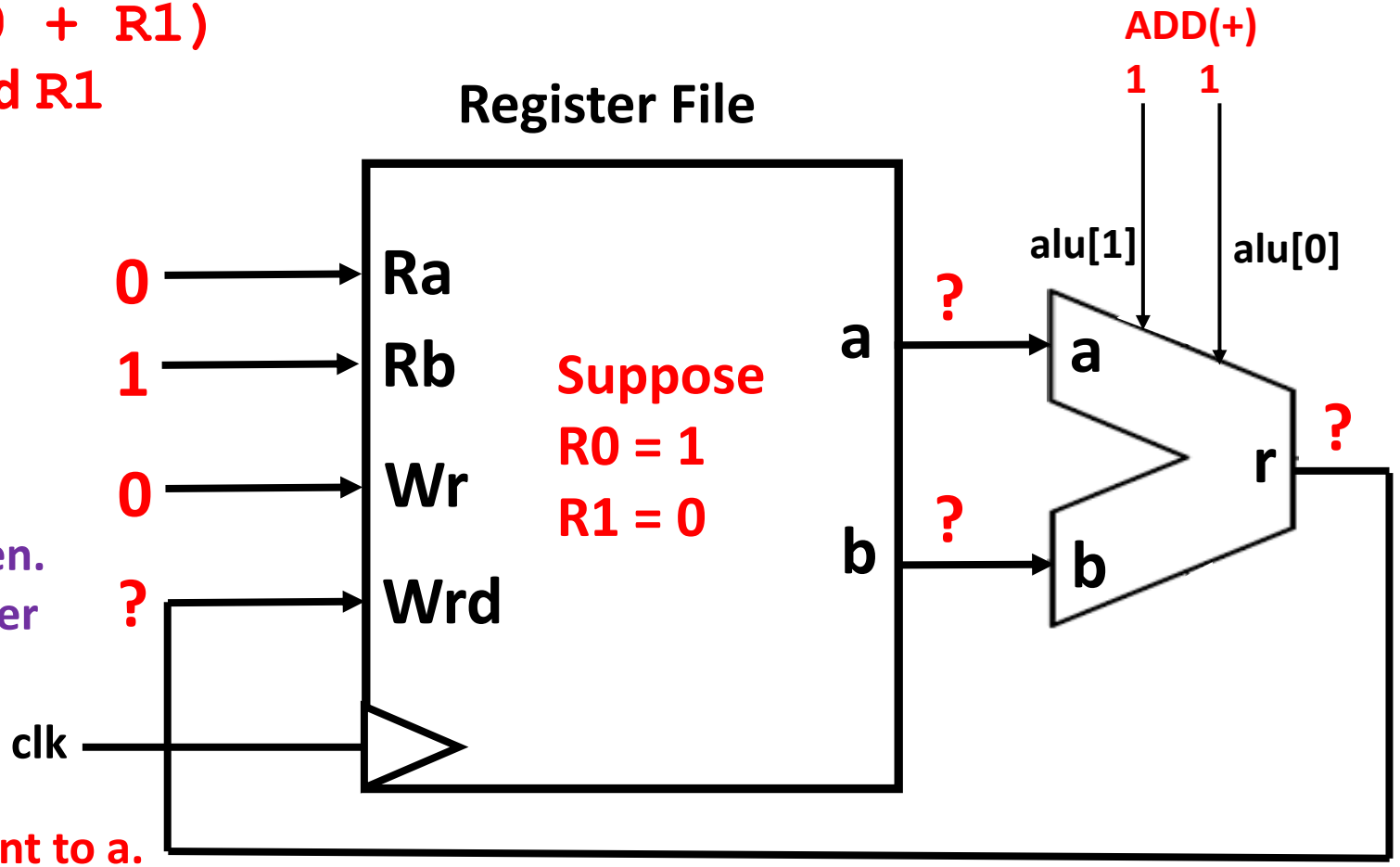


Figure: How Register set operates in CPU

Register Set Design

ADD R0, R1 (R0 = R0 + R1)

Registers to be read: R0 and R1

Register to be written: R0

Ra and Rb will select registers whose data to be read.

Wr will select register to be written.
Wrd is data to be written in register selected by Wr.

Value within register R0 will be sent to a.
Value within register R1 will be sent to b.

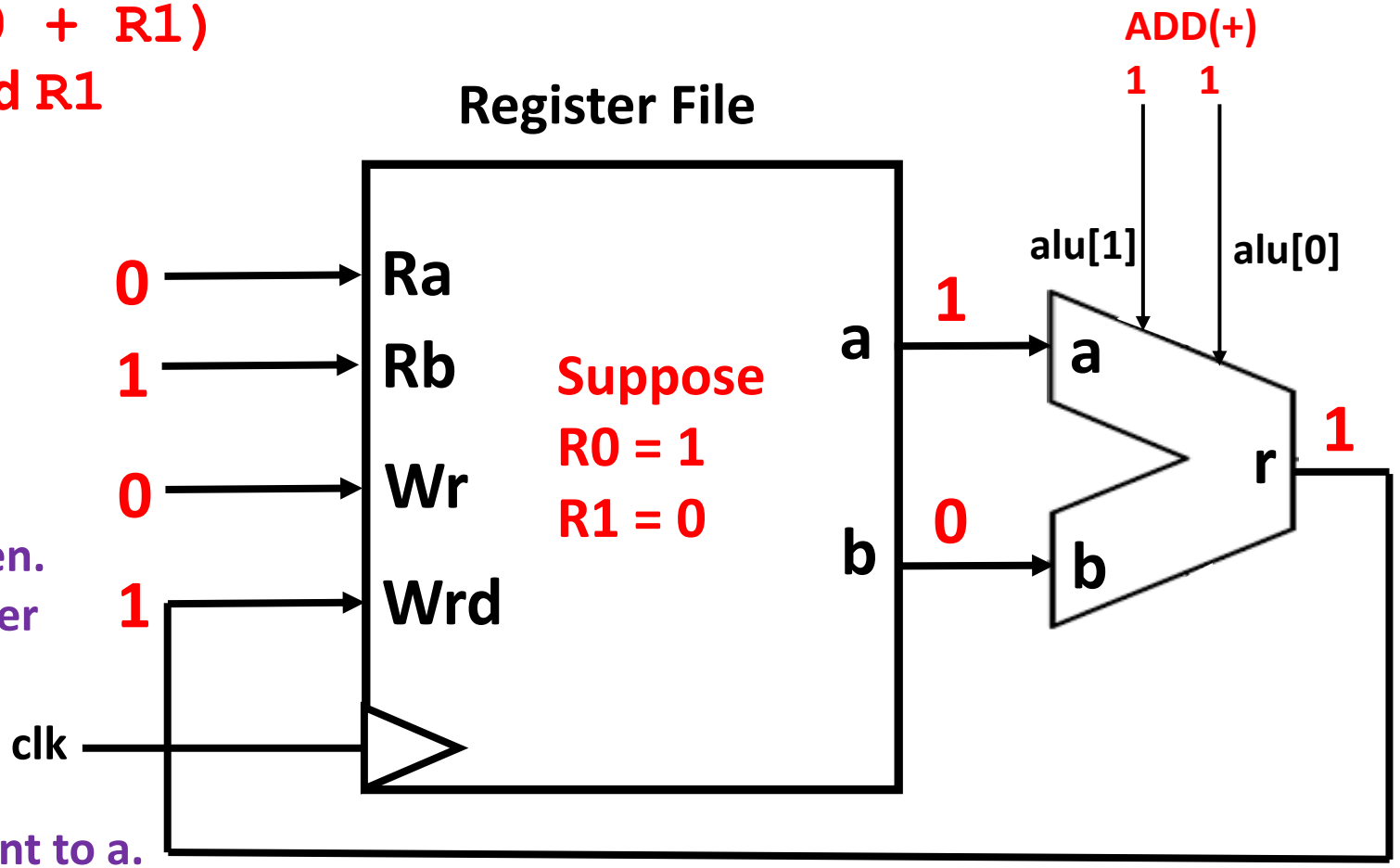
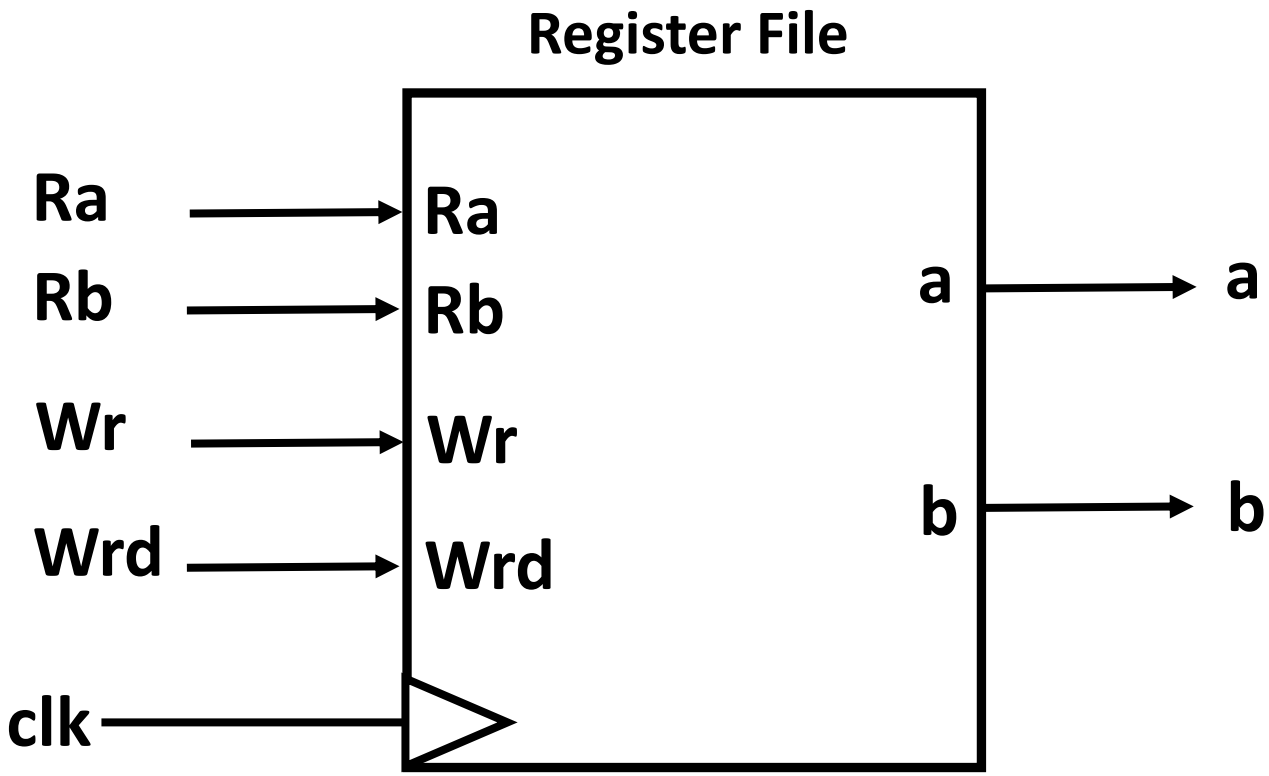


Figure: How Register set operates in CPU

Register Set Design Example (1-bit)

Register Set Design

1-bit Register Set with 2 1-bit registers



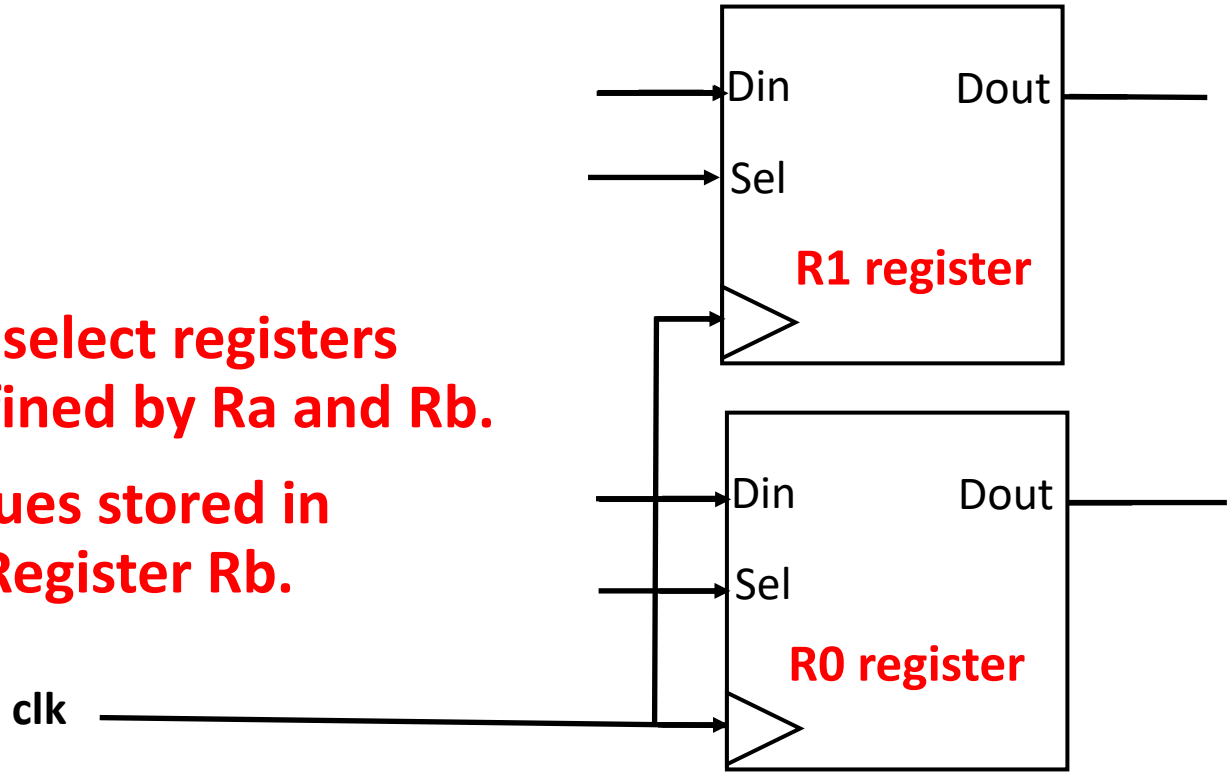
We will be designing this Register File!

Register Set Design

1-bit Register Set with 2 1-bit registers

Suppose we have 2 1-bit registers: R0 and R1.

Ra
Rb



First, we have to select registers which will be defined by Ra and Rb.

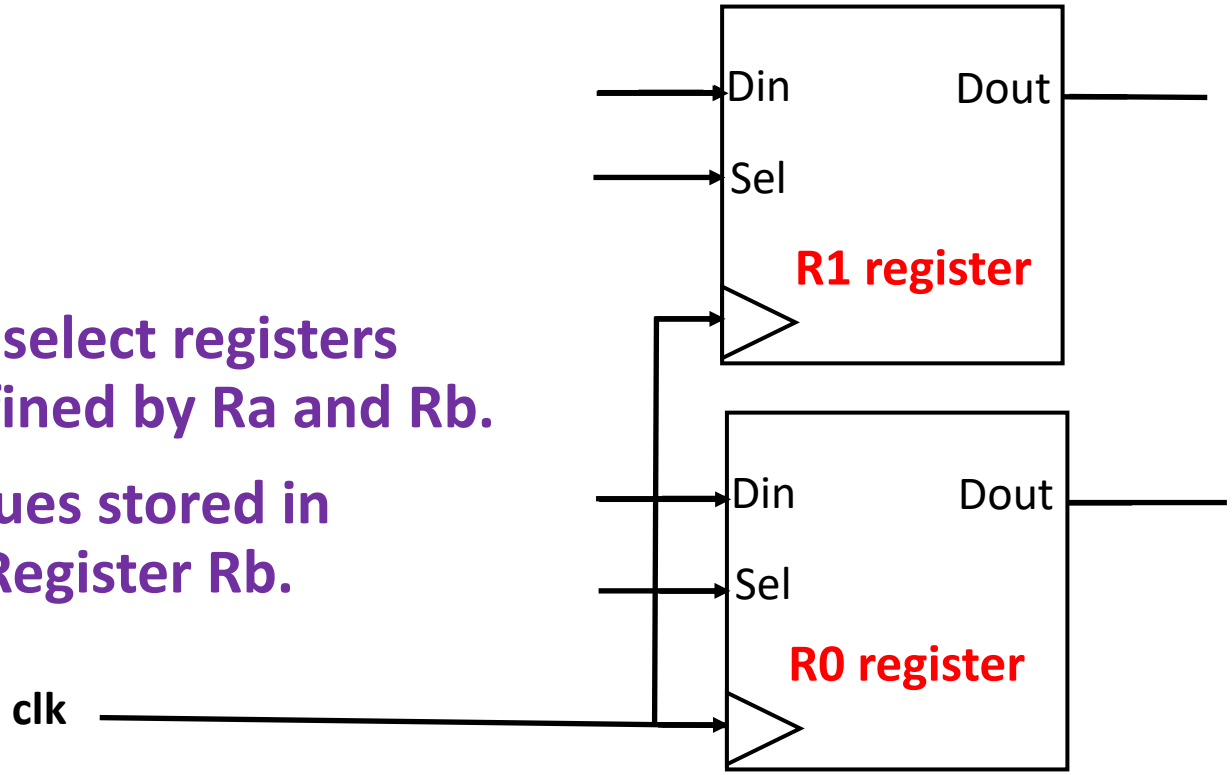
We just need values stored in Register Ra and Register Rb.

Register Set Design

1-bit Register Set with 2 1-bit registers

Suppose we have 2 1-bit registers: R0 and R1.

Ra
Rb



a
b

First, we have to select registers which will be defined by Ra and Rb.

We just need values stored in Register Ra and Register Rb.

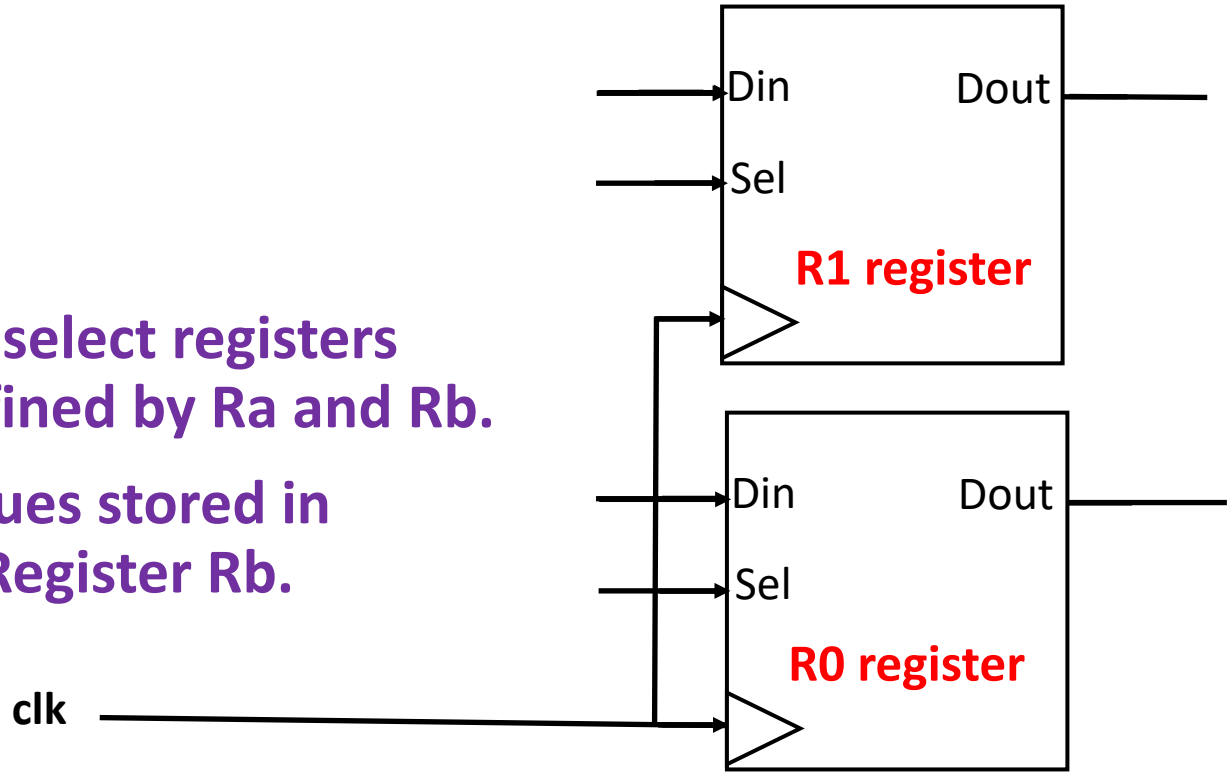
Then, We will have to send data of selected register to a (Ra) and b (Rb).

Register Set Design

1-bit Register Set with 2 1-bit registers

Suppose we have 2 1-bit registers: R0 and R1.

Ra
Rb



a
b

First, we have to select registers which will be defined by Ra and Rb.

We just need values stored in Register Ra and Register Rb.

Then, We will have to send data of selected register to a (Ra) and b (Rb).

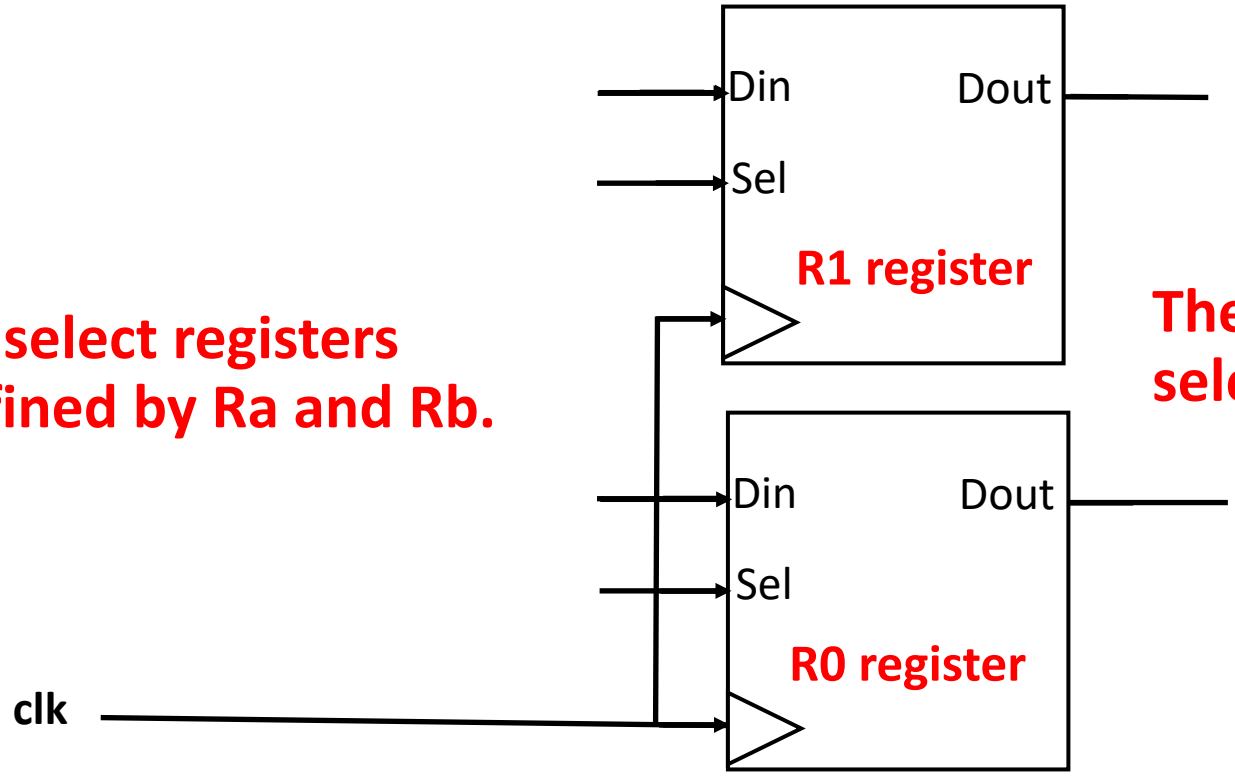
Register Set Design

1-bit Register Set with 2 1-bit registers

Suppose we have 2 1-bit registers: R0 and R1.

Ra

Rb



a

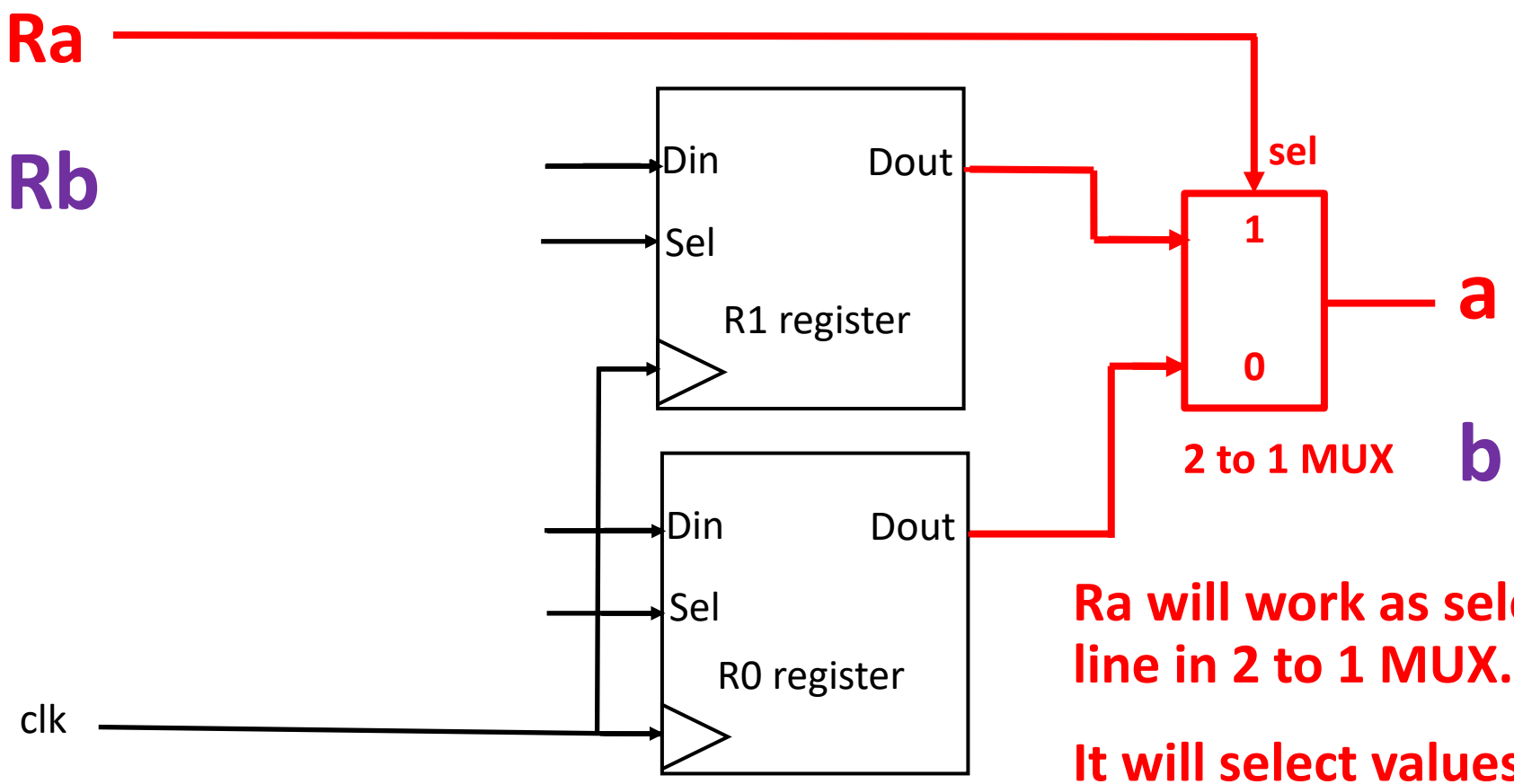
b

First, we have to select registers which will be defined by Ra and Rb.

Then, We will have to send data o selected register to a and b

Register Set Design

1-bit Register Set with 2 1-bit registers

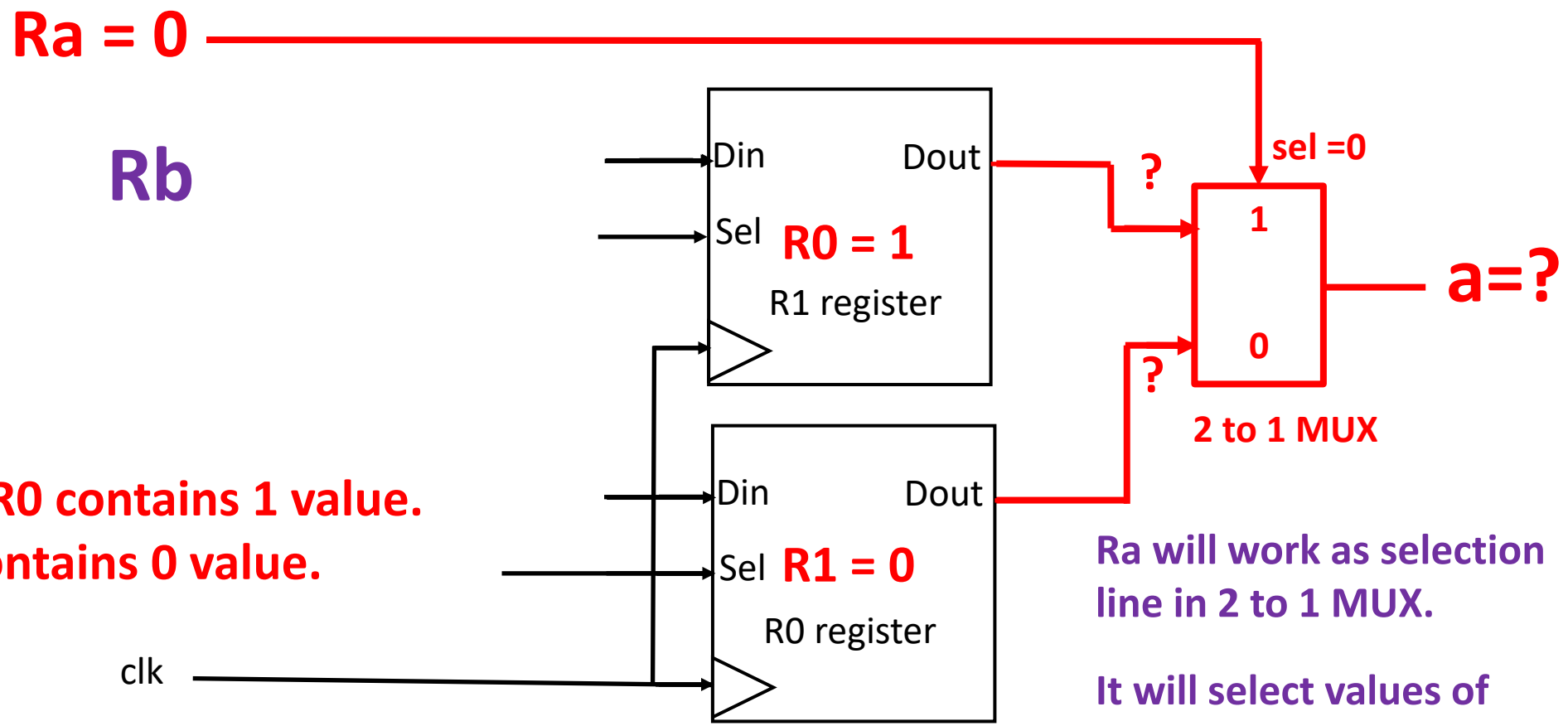


Ra will work as selection line in 2 to 1 MUX.

It will select values of one of registers selected by Ra.

Register Set Design

1-bit Register Set with 2 1-bit registers



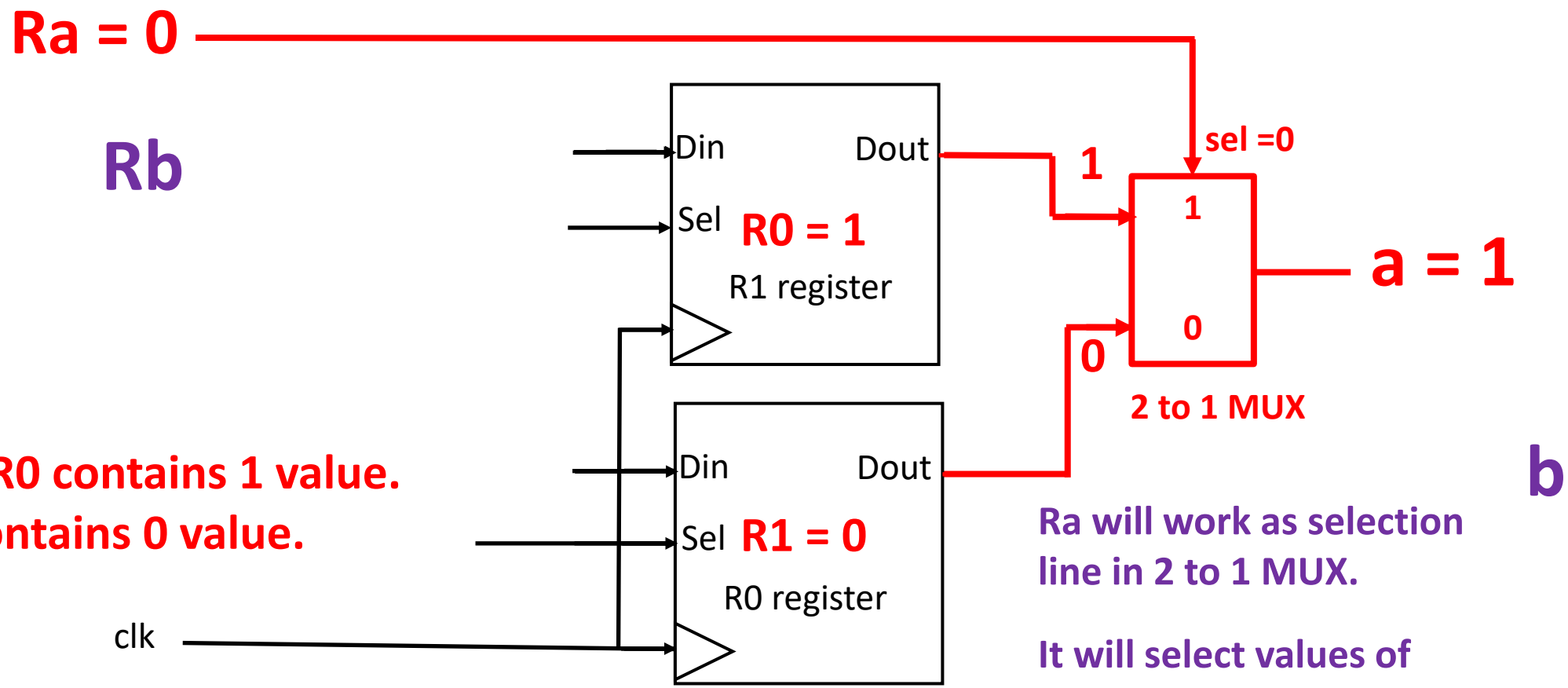
Suppose
R0 = 1
R1 = 0
It means R0 contains 1 value.
And R1 contains 0 value.

Ra will work as selection line in 2 to 1 MUX.

It will select values of one of registers selected by Ra.

Register Set Design

1-bit Register Set with 2 1-bit registers



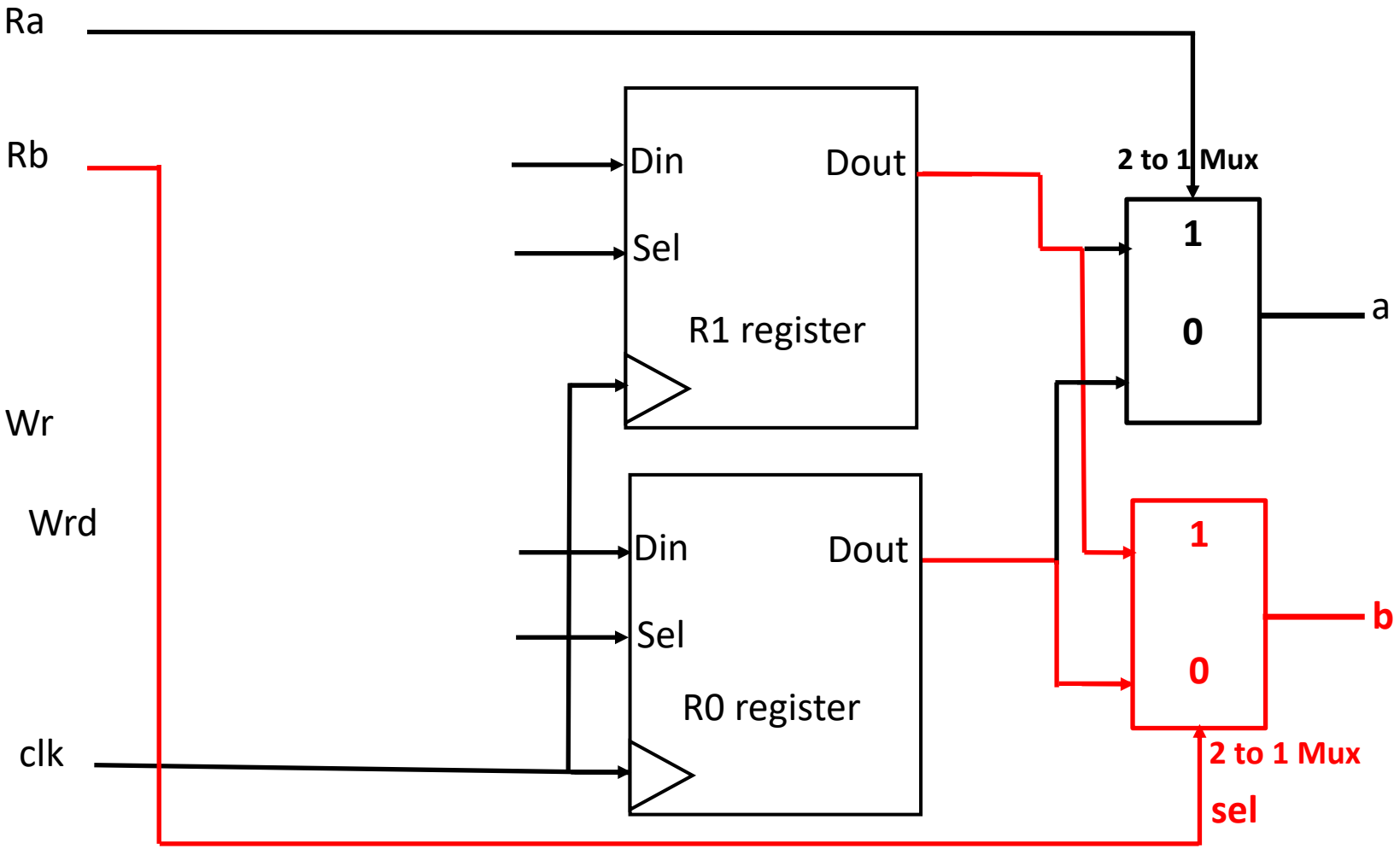
Suppose
R0 = 1
R1 = 0
It means R0 contains 1 value.
And R1 contains 0 value.

Ra will work as selection line in 2 to 1 MUX.

It will select values of one of registers selected by Ra.

Register Set Design

1-bit Register Set with 2 1-bit registers

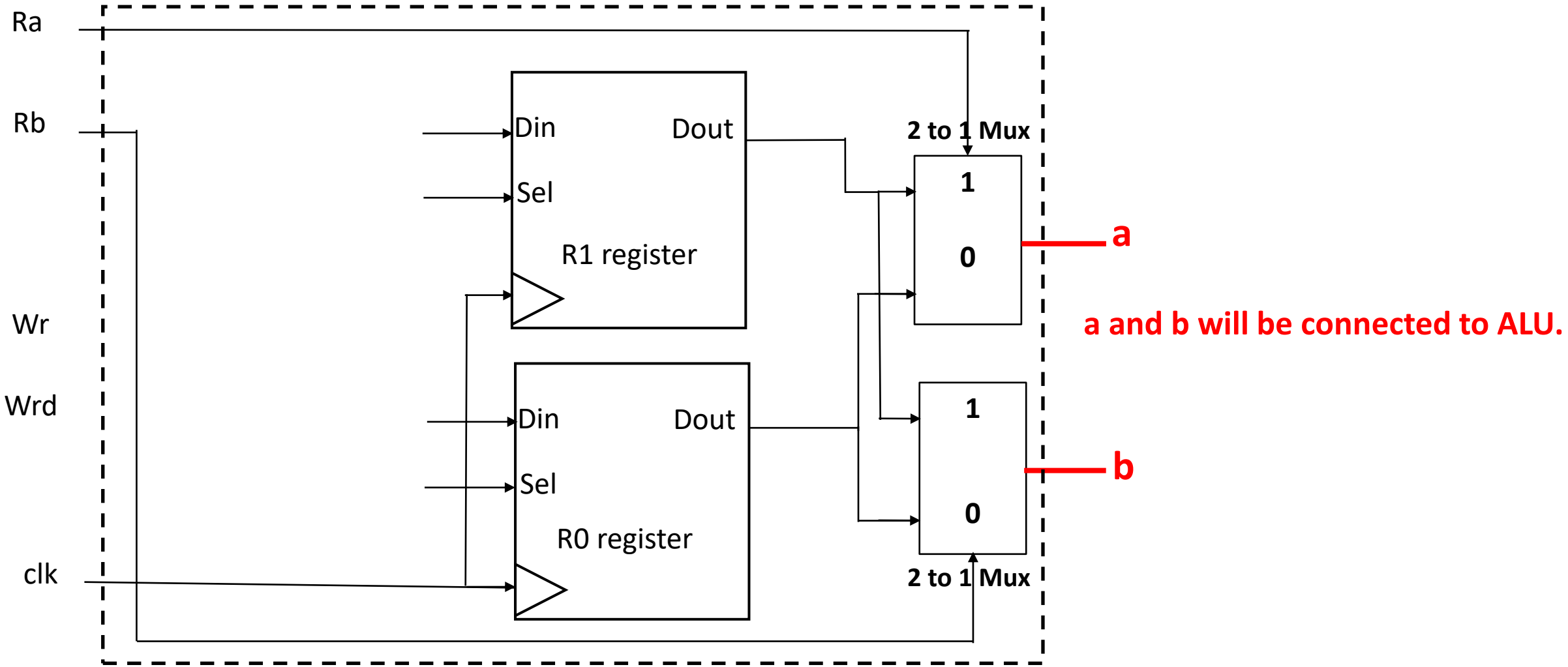


Similarly, Rb will work as selection line in 2 to 1 MUX.

It will select values of one of registers selected by Rb.

Register Set Design

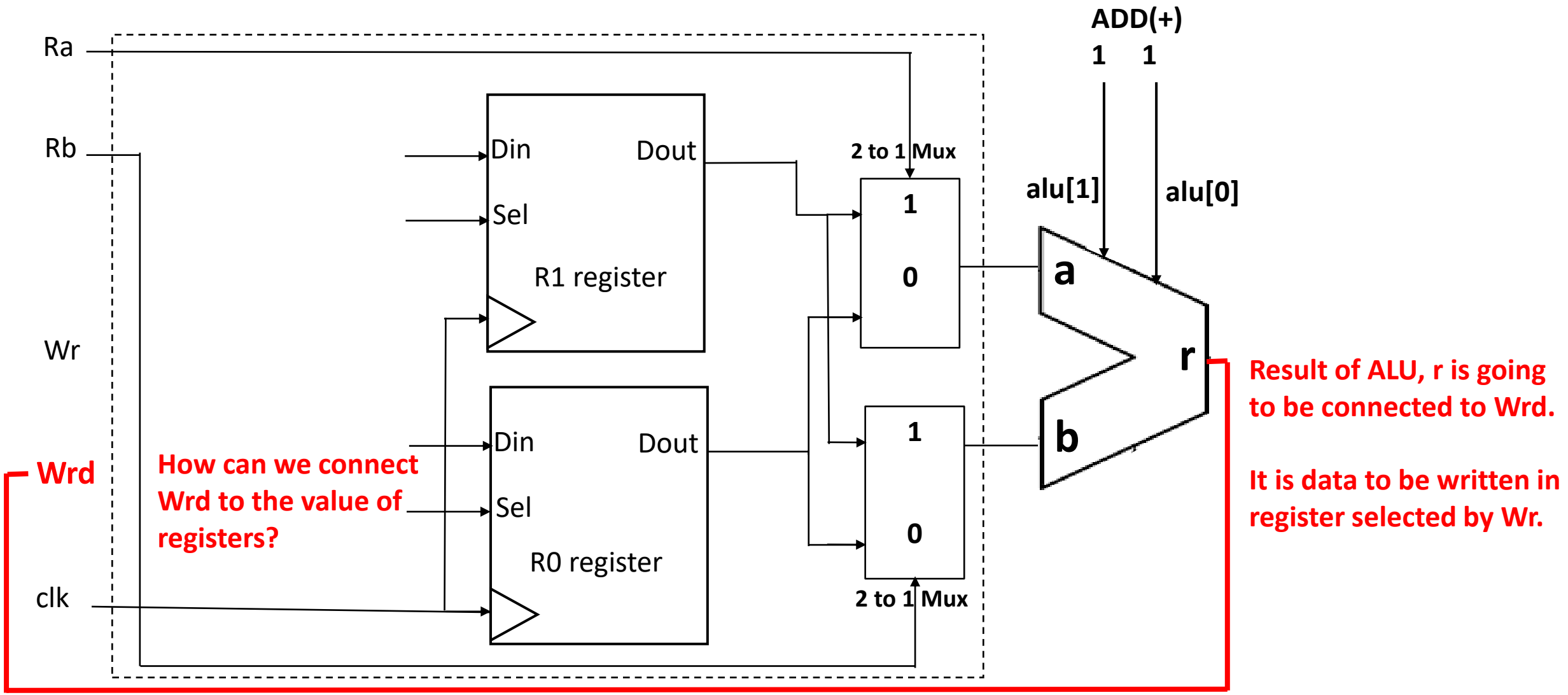
1-bit Register Set with 2 1-bit registers



a and b will be connected to ALU.

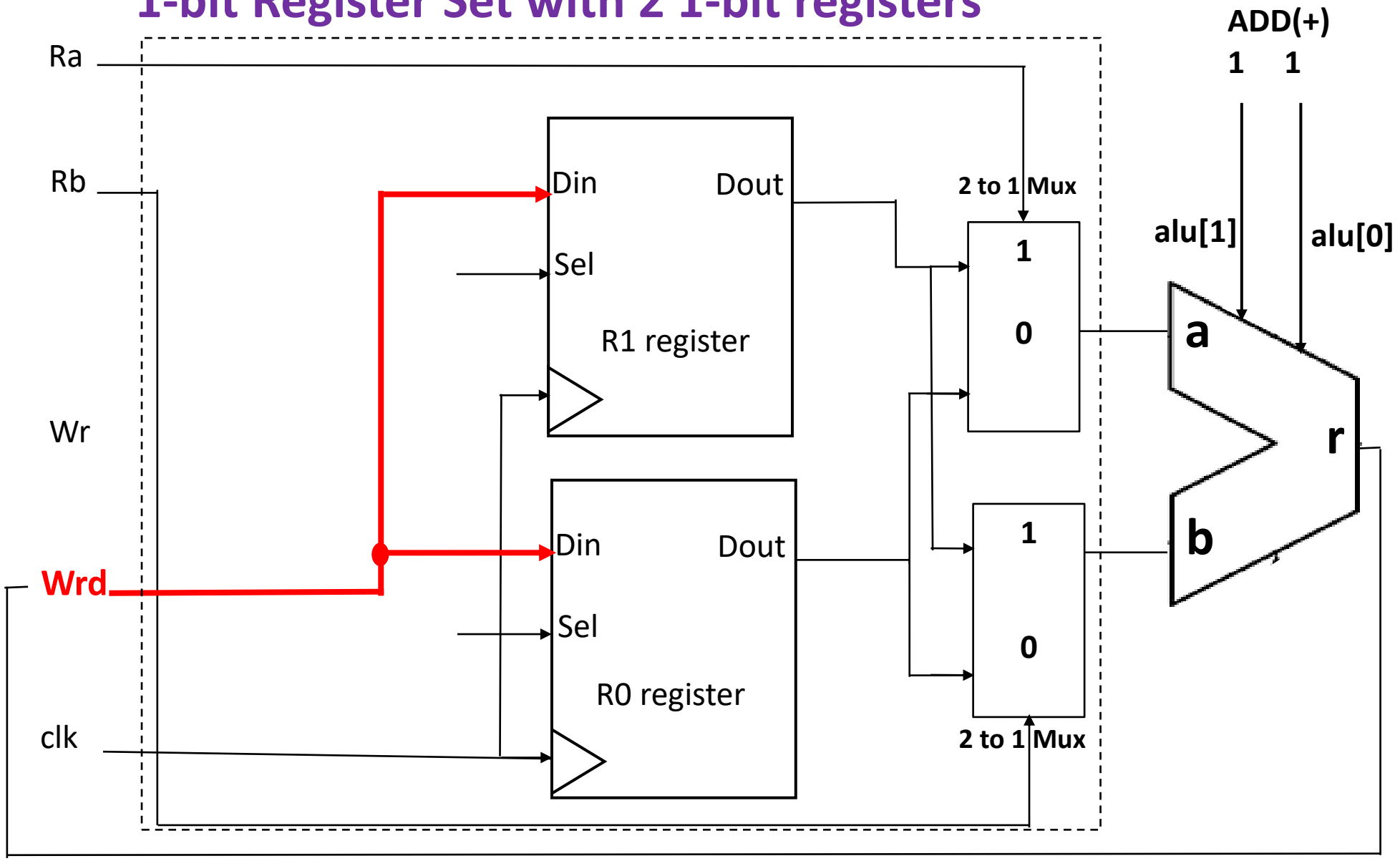
Register Set Design

1-bit Register Set with 2 1-bit registers



Register Set Design

1-bit Register Set with 2 1-bit registers



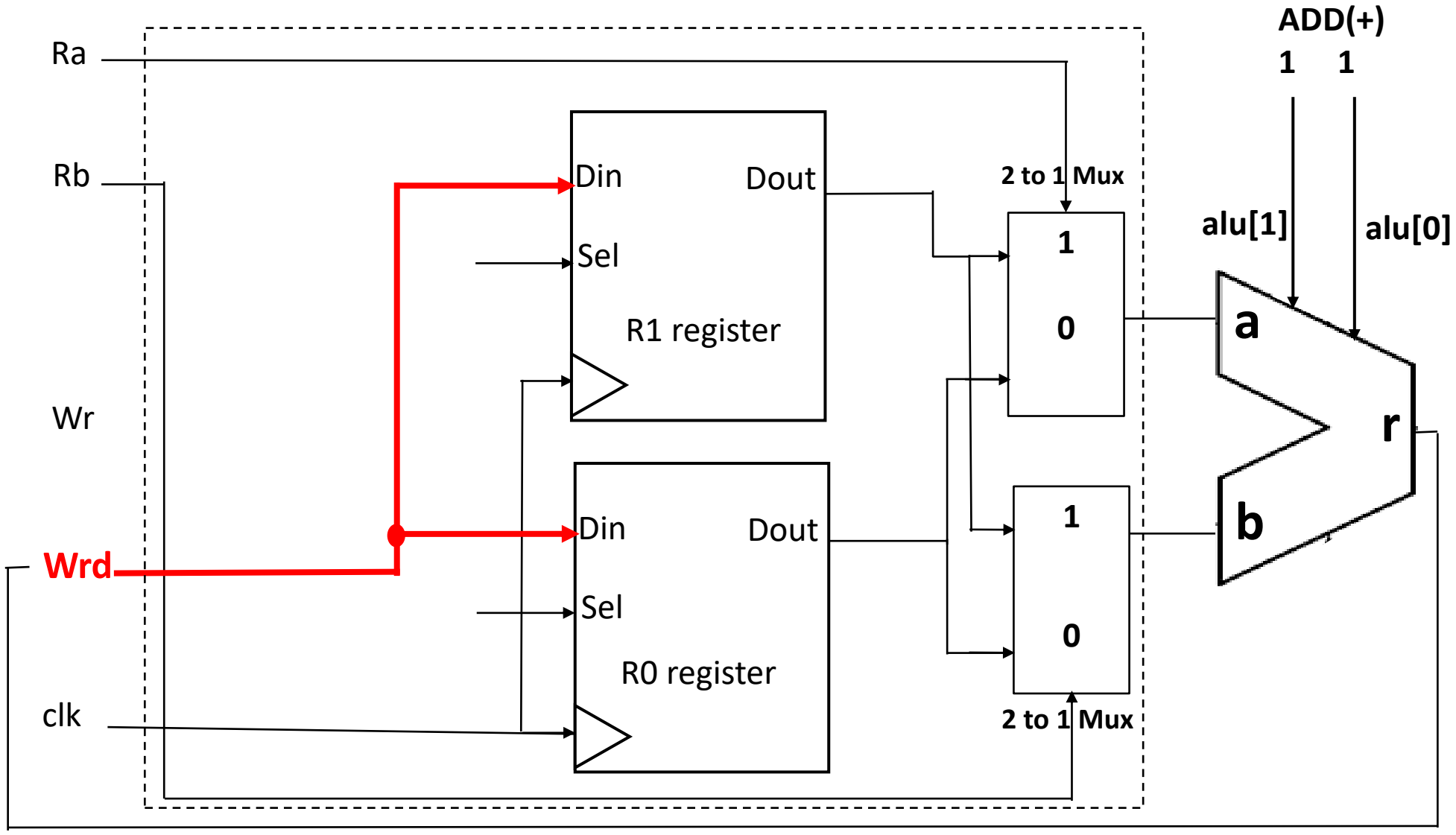
We will directly connect Wrd to Din pin of each register.

Even though Wrd is connected to Din, It cannot update data in register UNLESS Sel = 1 AND CLOCK IS POSITIVE EDGE.

Register Set Design

1-bit Register Set with 2 1-bit registers

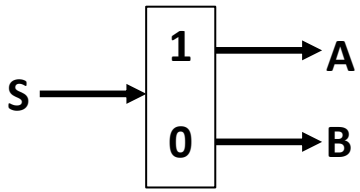
How are we going to select one of registers defined by Wr?



Register Set Design

1-bit Register Set with 2 1-bit registers

1 to 2 Decoder

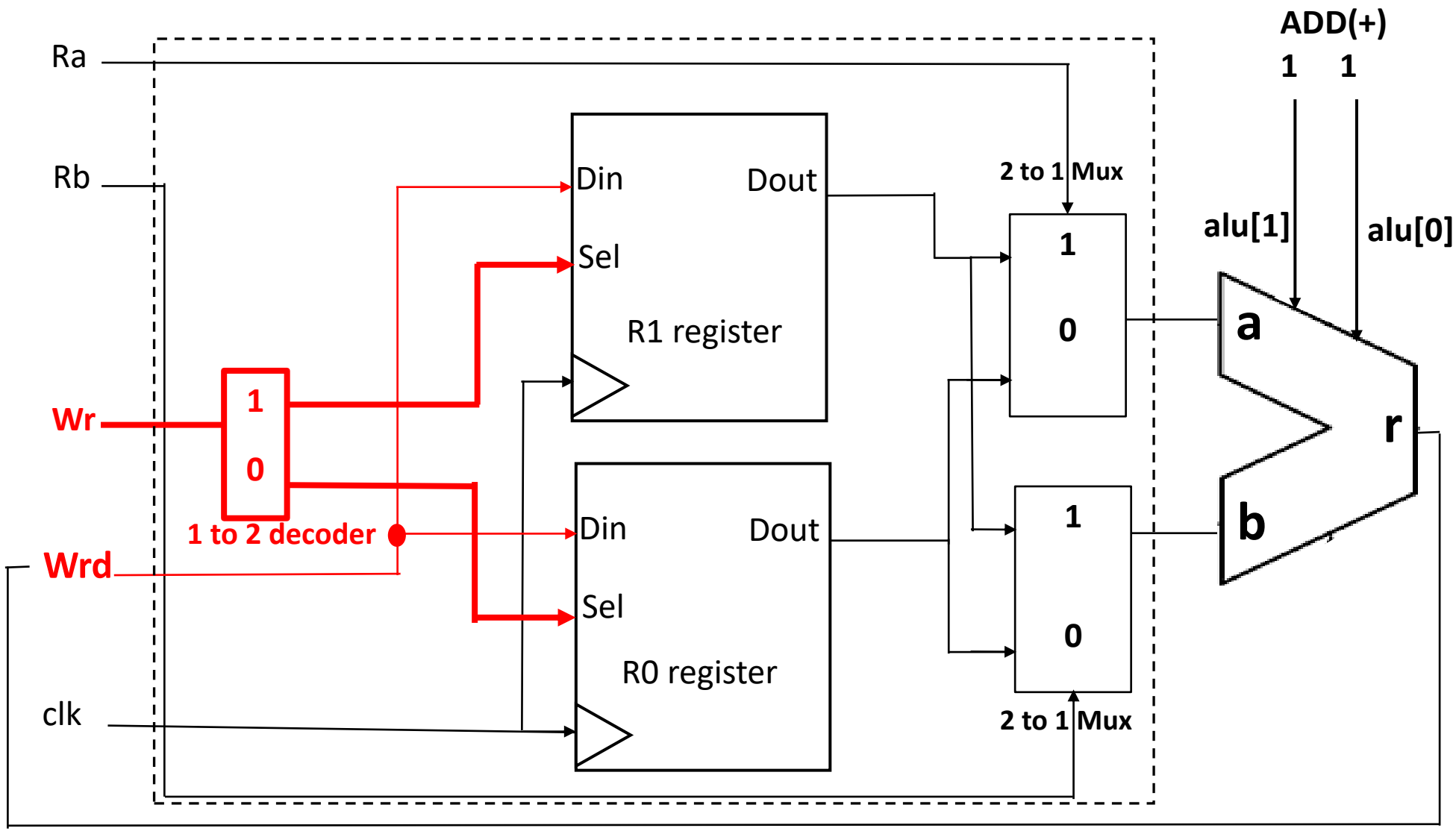


S	A	B
0	1	0
1	0	1

$$A = \bar{S}$$

$$B = S$$

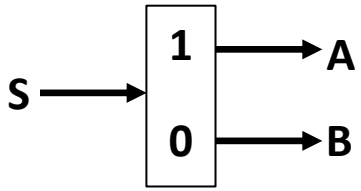
We can use a 1 to 2 decoder to select one of register.



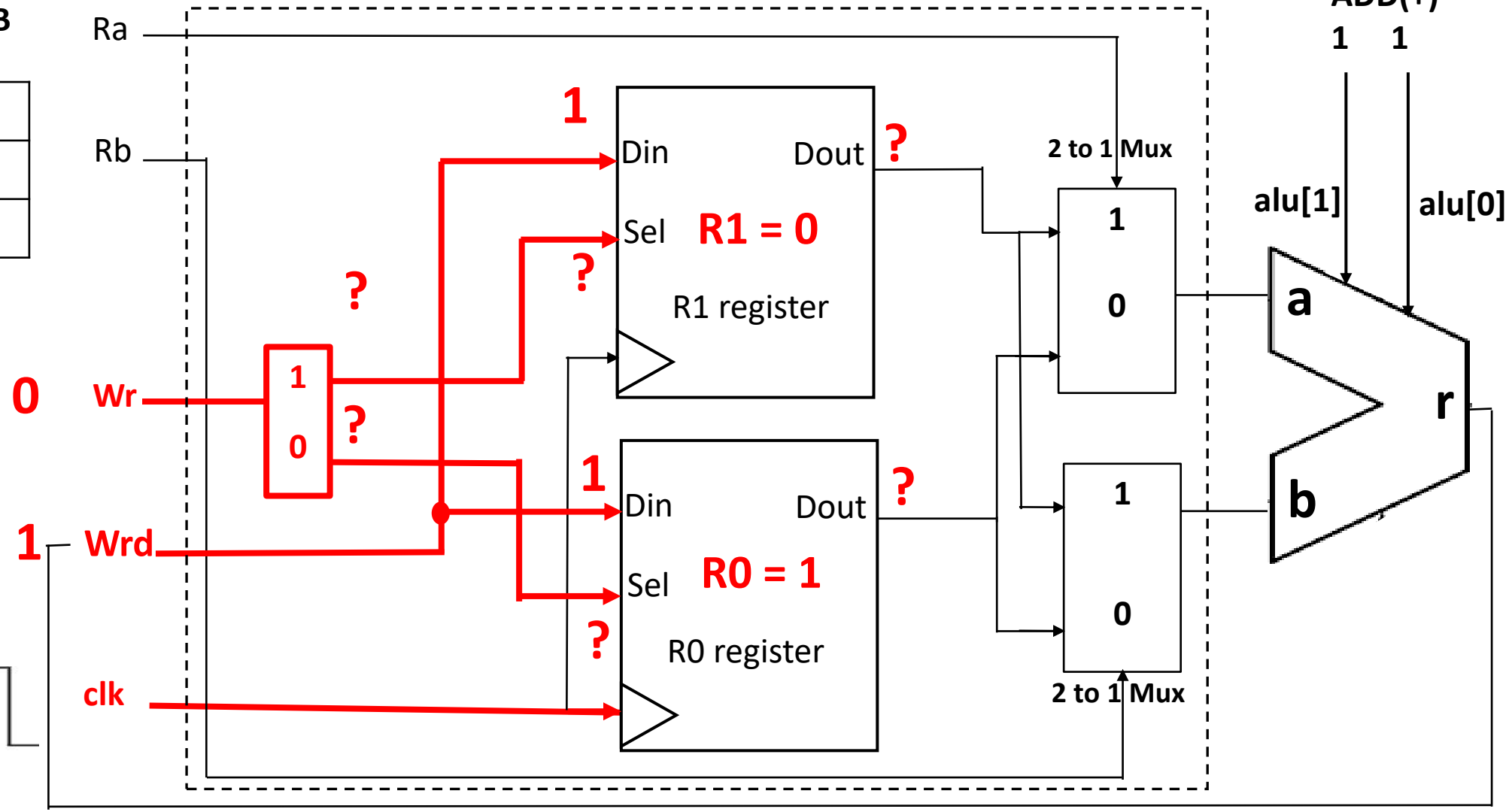
Register Set Design

1-bit Register Set with 2 1-bit registers

1 to 2 Decoder



S	A	B
0	1	0
1	0	1

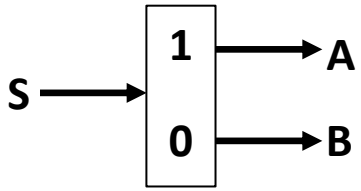


Clock is on negative edge.

Register Set Design

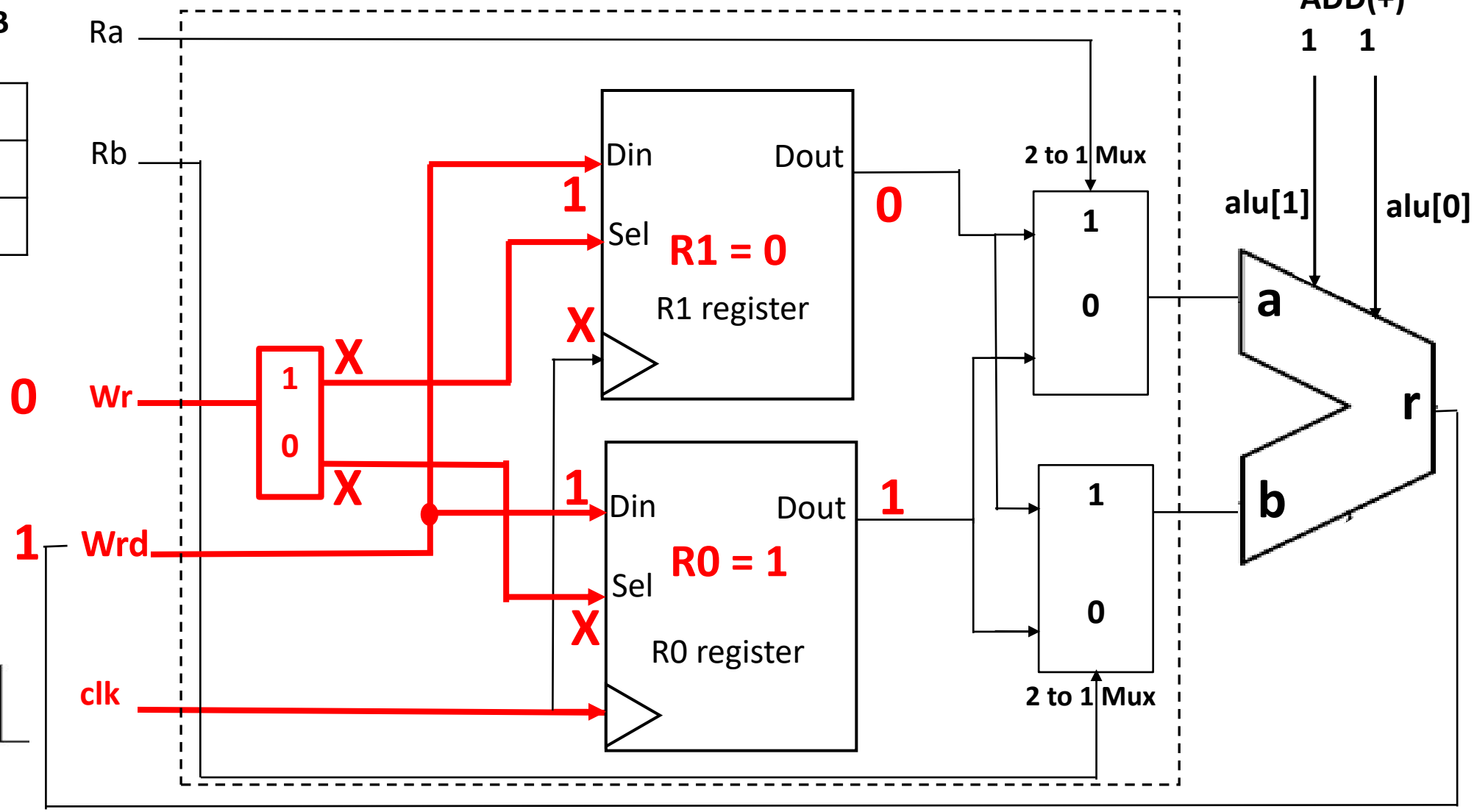
1-bit Register Set with 2 1-bit registers

1 to 2 Decoder



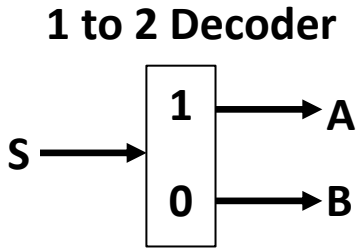
S	A	B
0	1	0
1	0	1

Since clock is on negative edge, value of registers will not be updated!

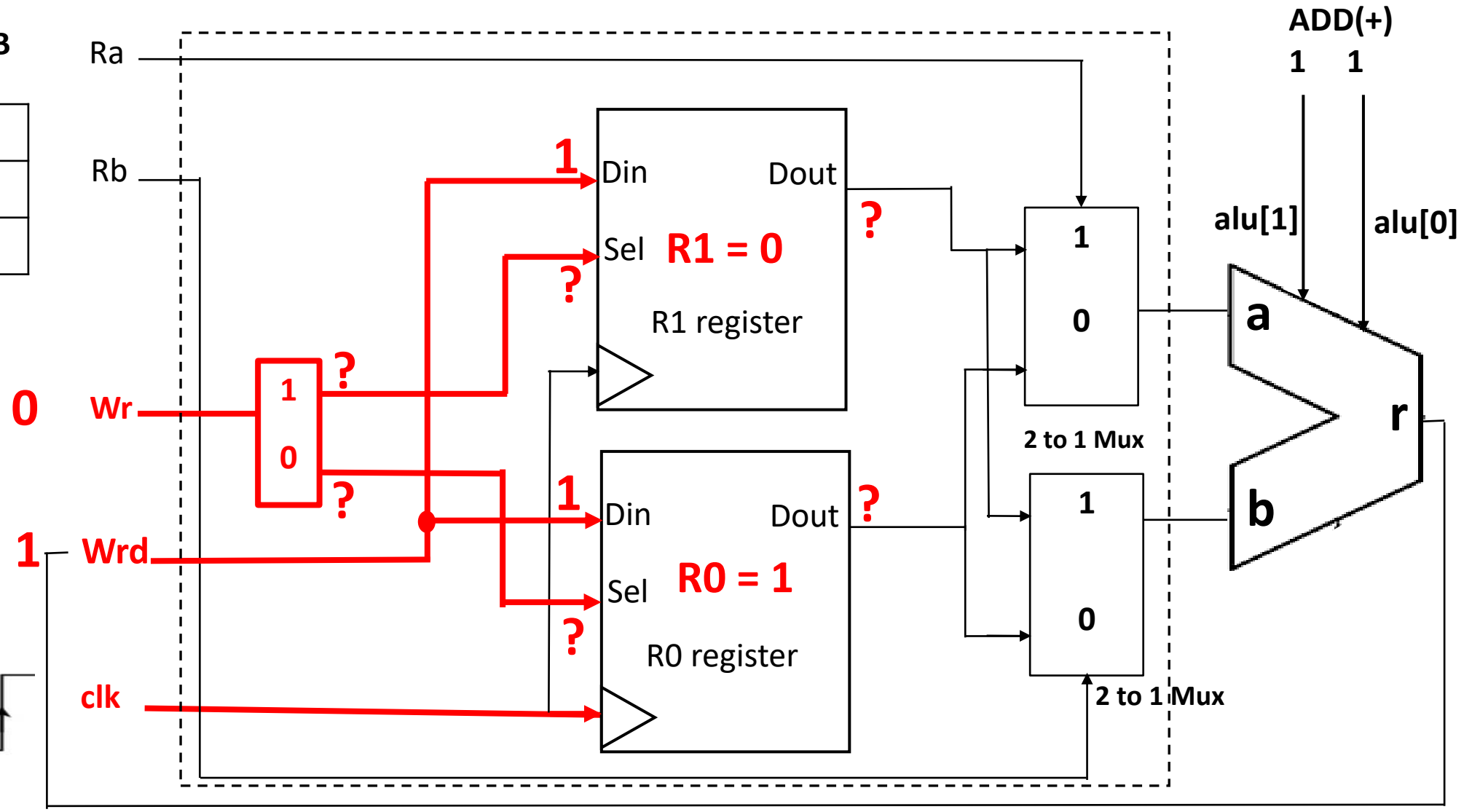


Register Set Design

1-bit Register Set with 2 1-bit registers

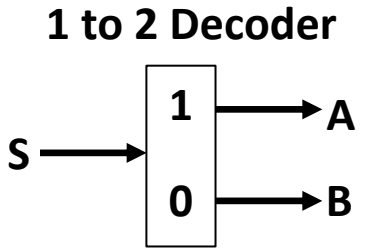


S	A	B
0	1	0
1	0	1

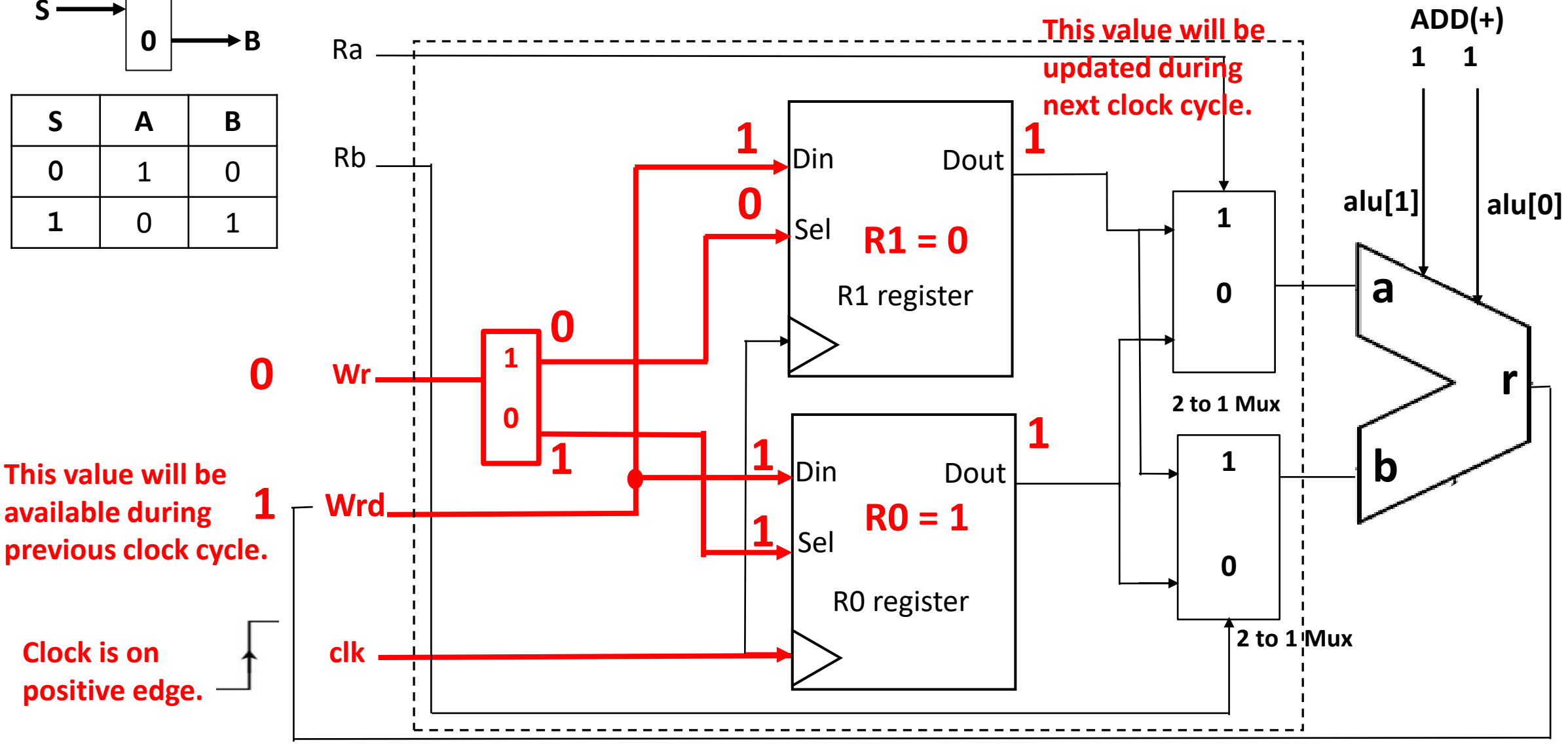


Register Set Design

1-bit Register Set with 2 1-bit registers



S	A	B
0	1	0
1	0	1



Register Set Design

1-bit Register Set with 2 1-bit registers

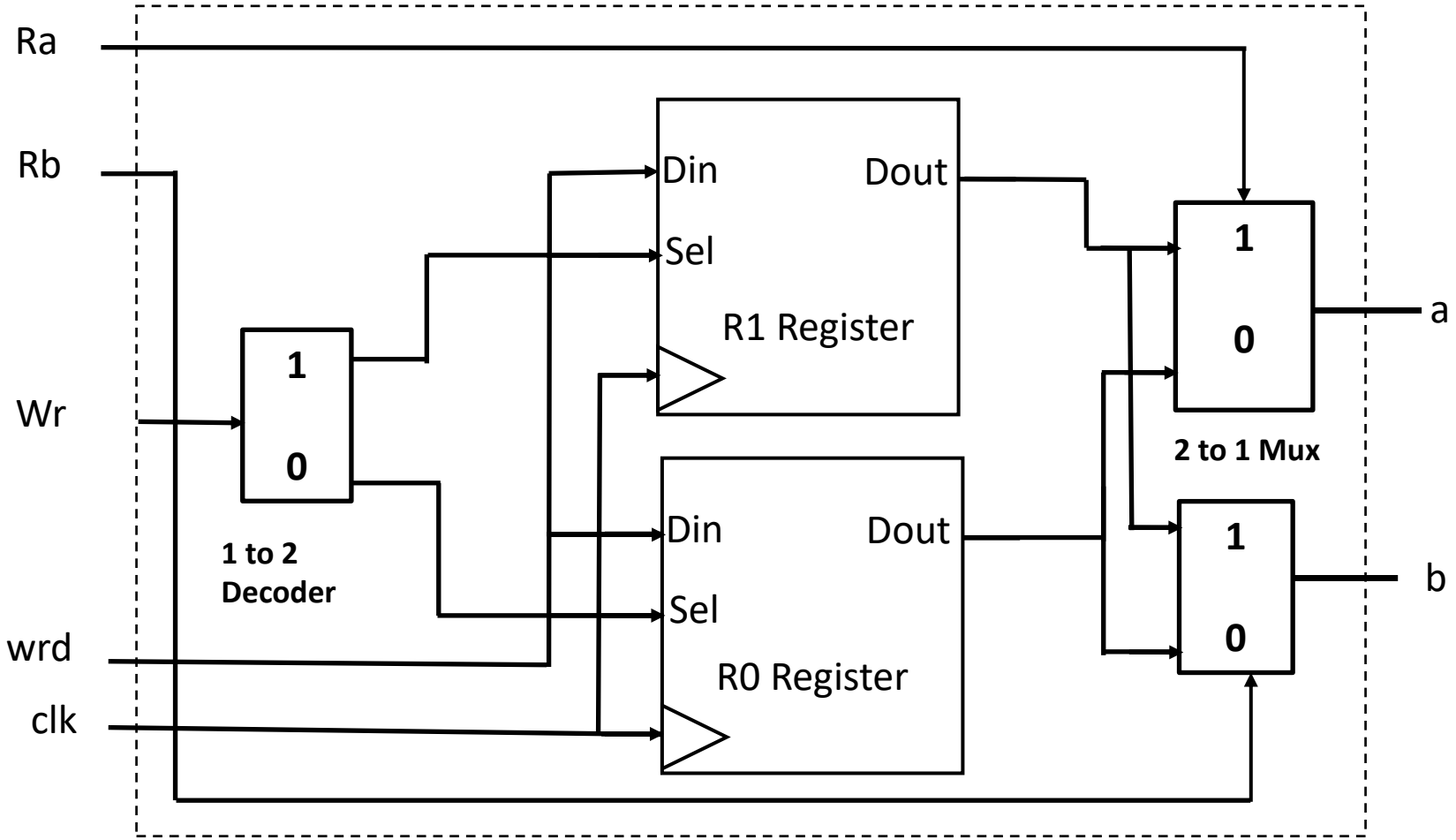


Figure: 1-bit Register set with 2 1-bit registers (Final Design)

Register Set Design

1-bit Register Set with 2 1-bit registers

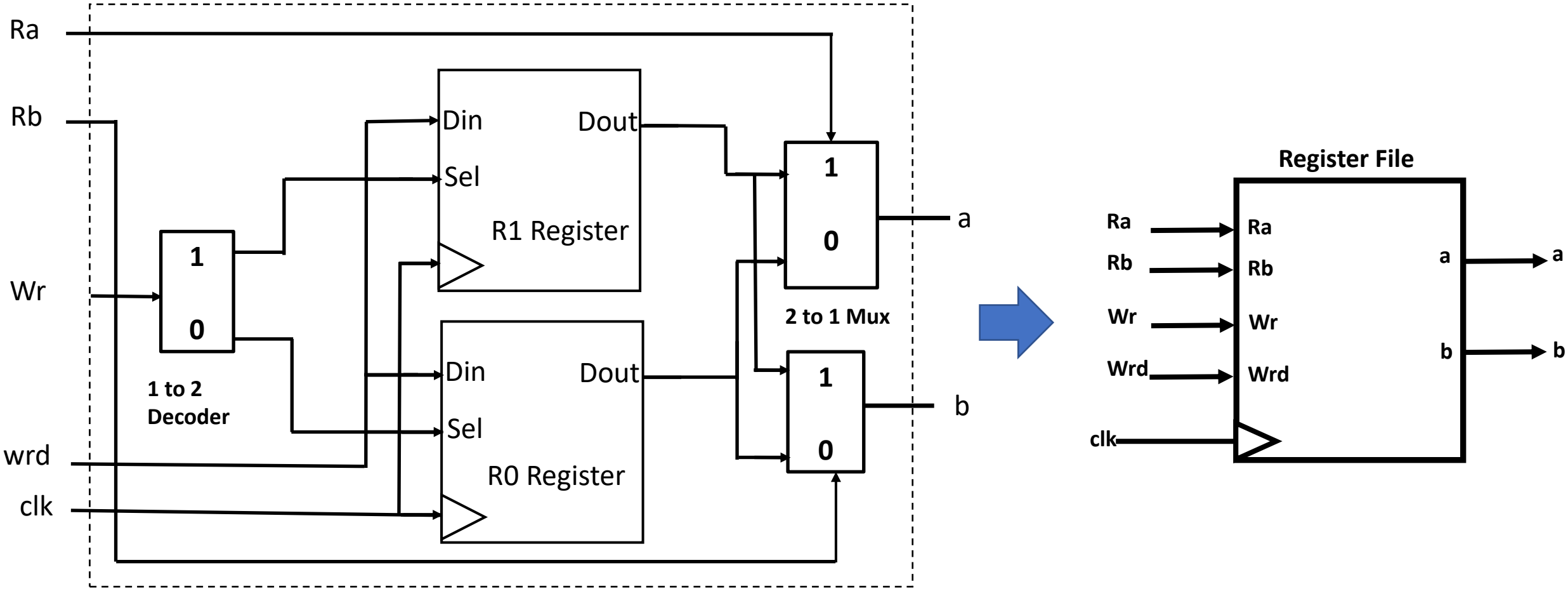


Figure: 1-bit Register Set with 2 1-bit registers

Register Set Design Example (2-bit)

2-bit Register

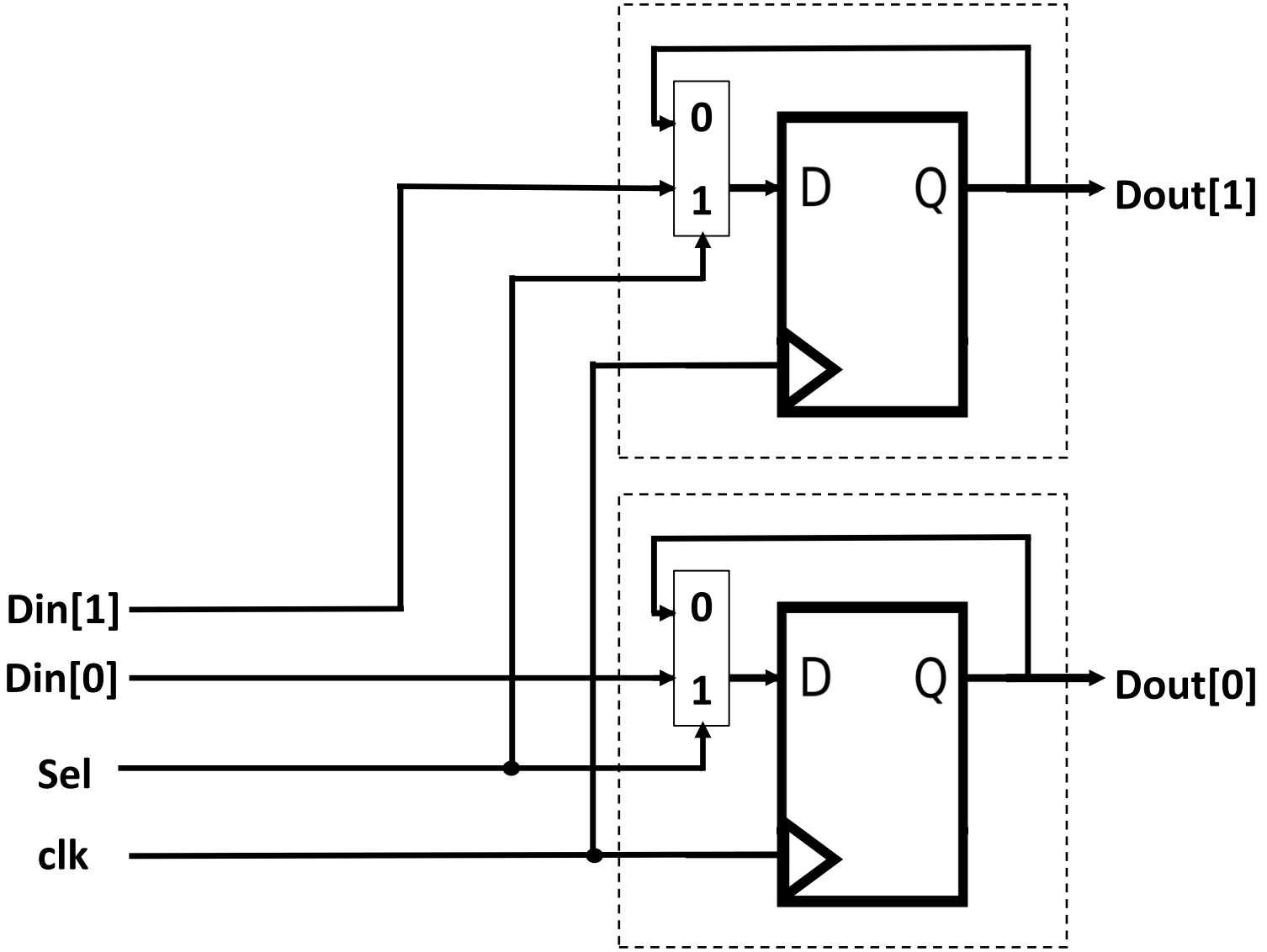
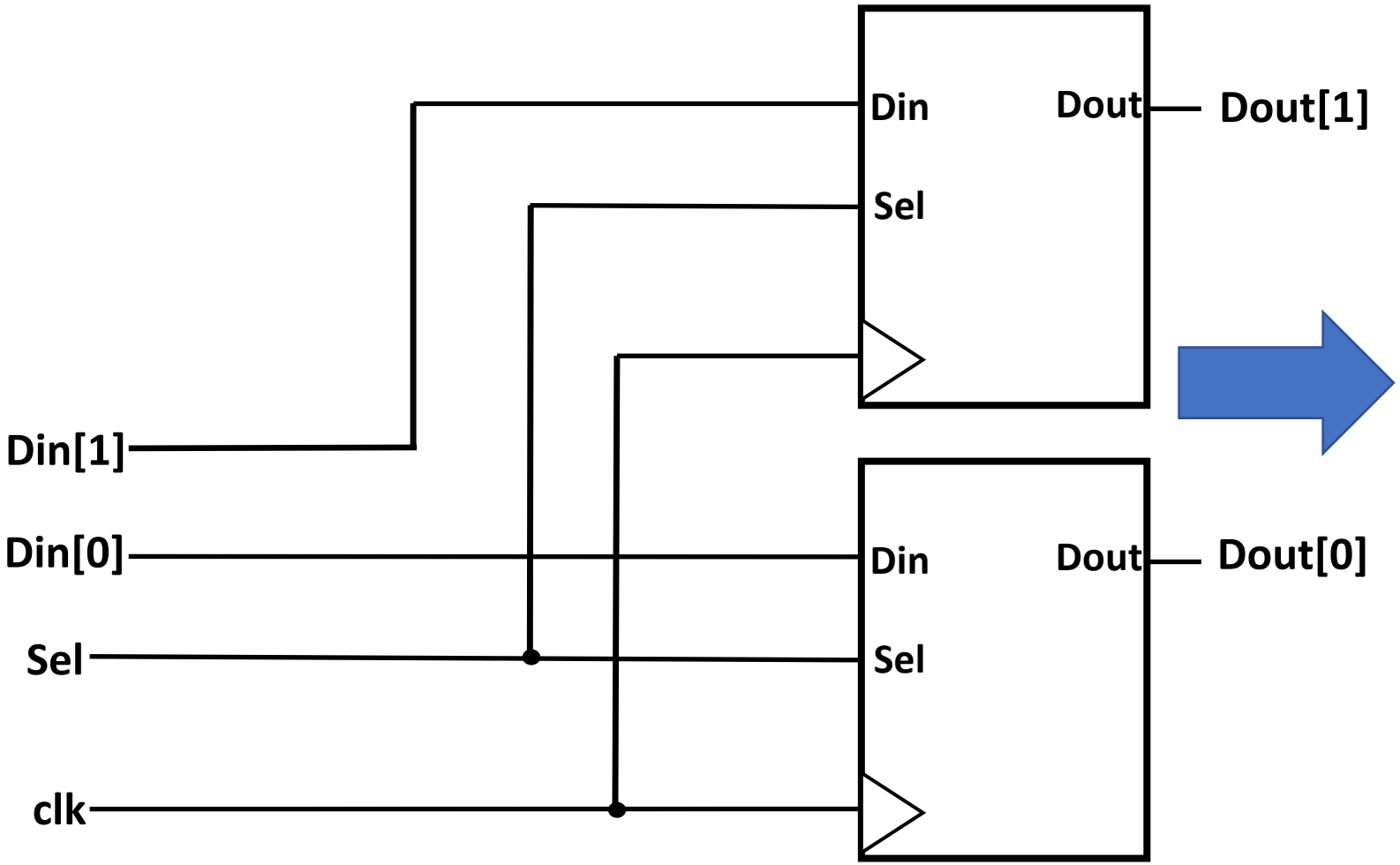


Figure: Inside of 2 bit Register

2-bit Register



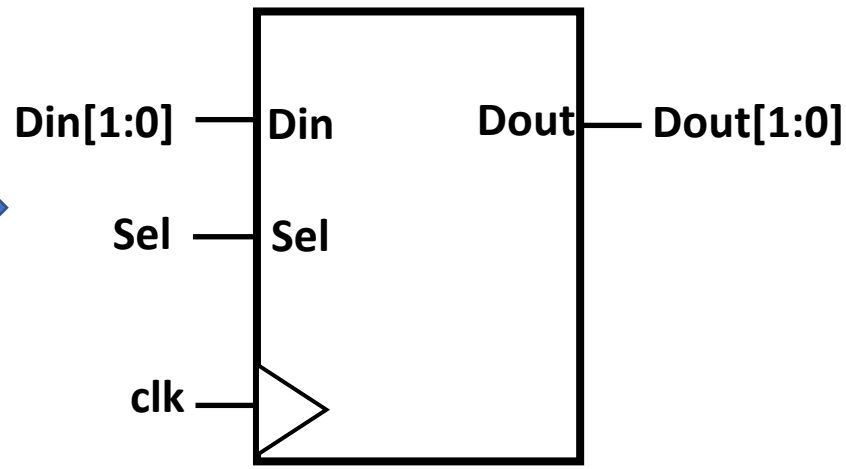


Figure: 2 bit register chip

Figure: 2 bit Register

Register Set Design

1bit Register Set with 2 1bit registers

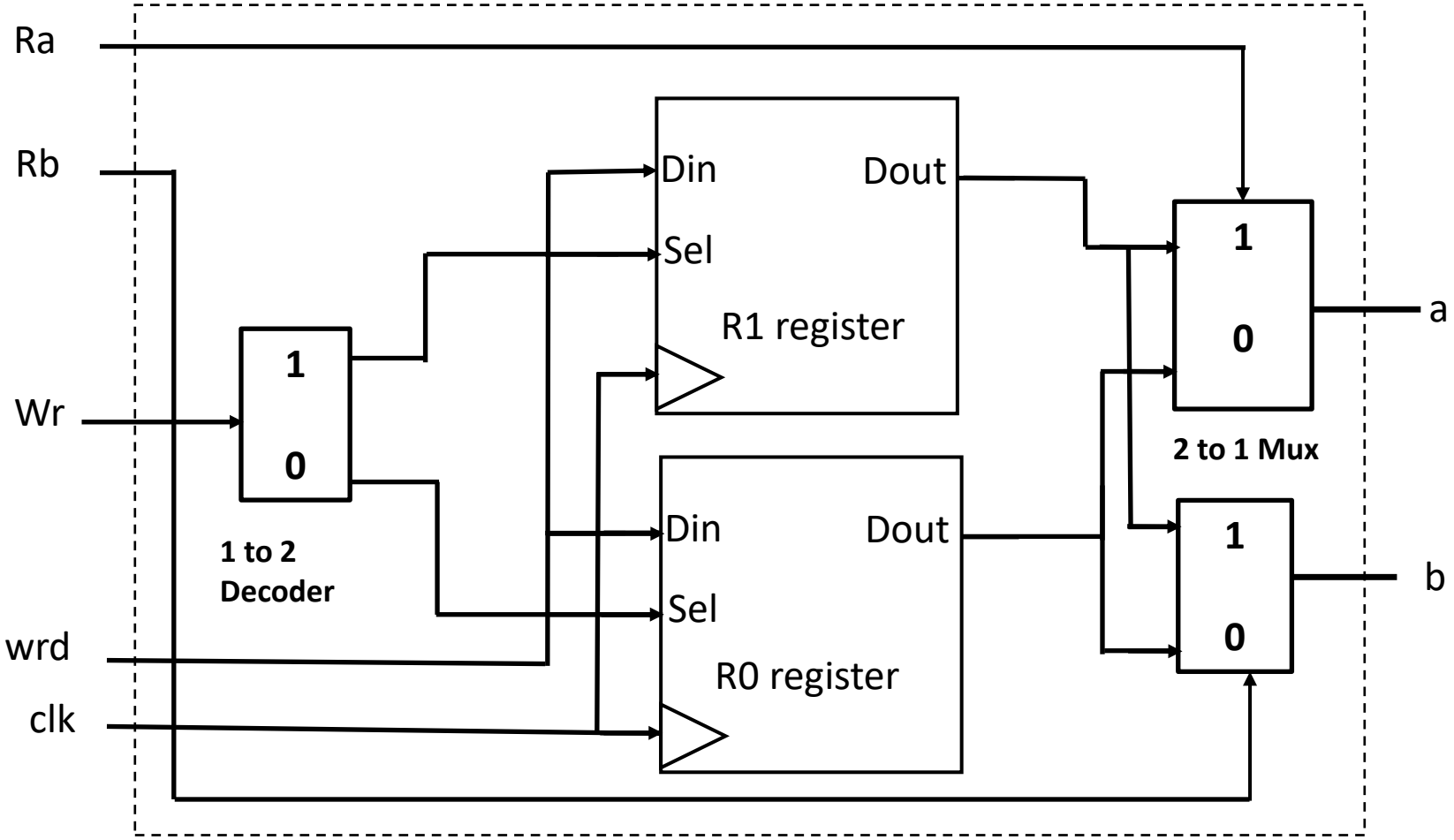


Figure: 1-bit Register Set with 2 1-bit registers

There will be always two MUX.

Because there is always two operands (A contained in Ra and B contained in Rb) in ALU.

Register Set Design

2 bit Register Set with 2 2bit registers

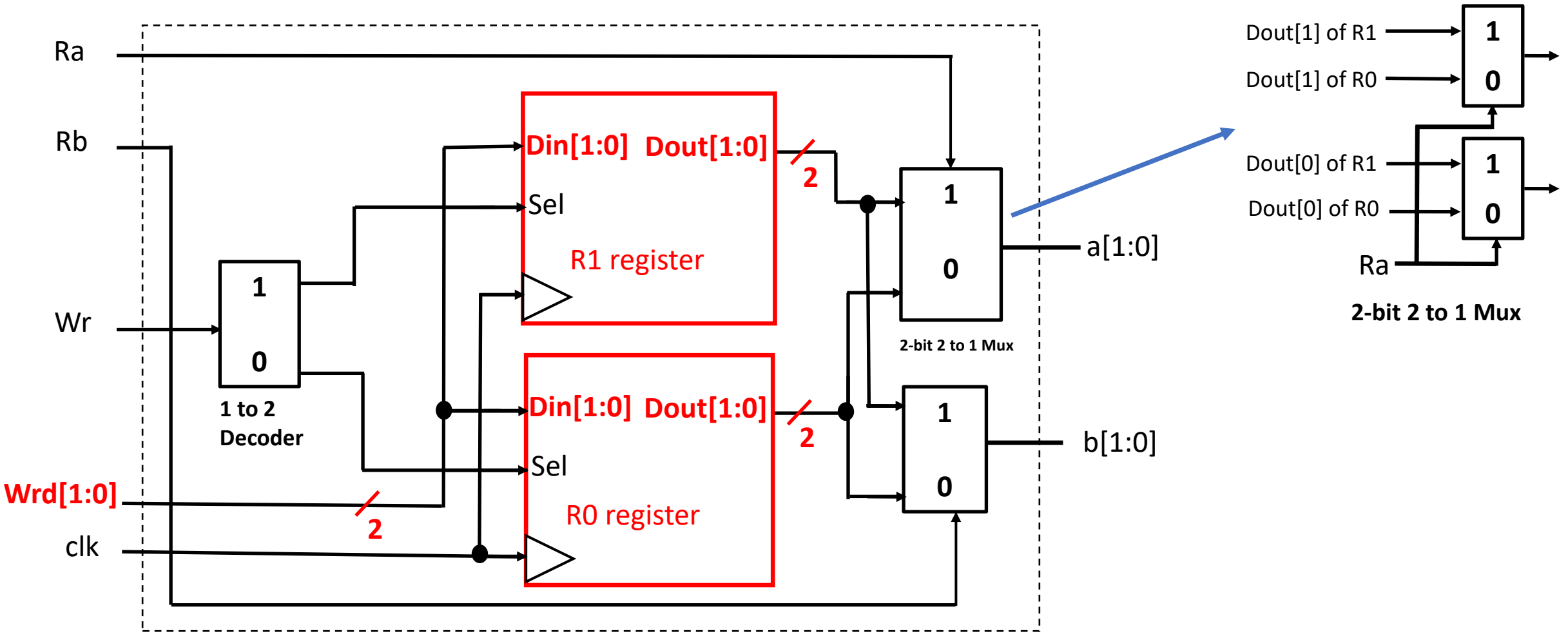


Figure: 2 bit Register set with 2 2-bit registers

Register Set Design

2 bit Register Set with 4 2bit registers

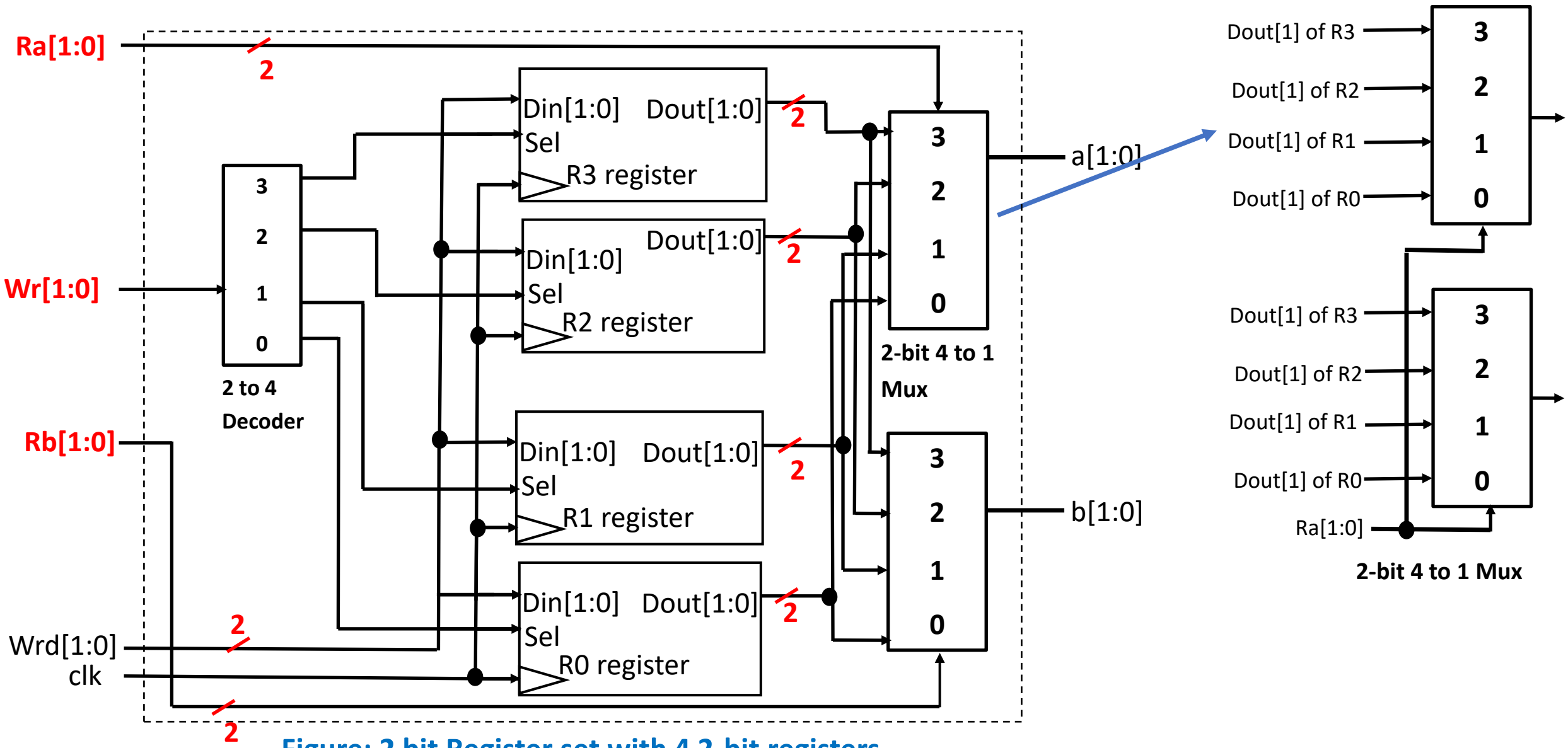


Figure: 2 bit Register set with 4 2-bit registers

Register Set Design
2 bit Register Set with 4 2bit registers

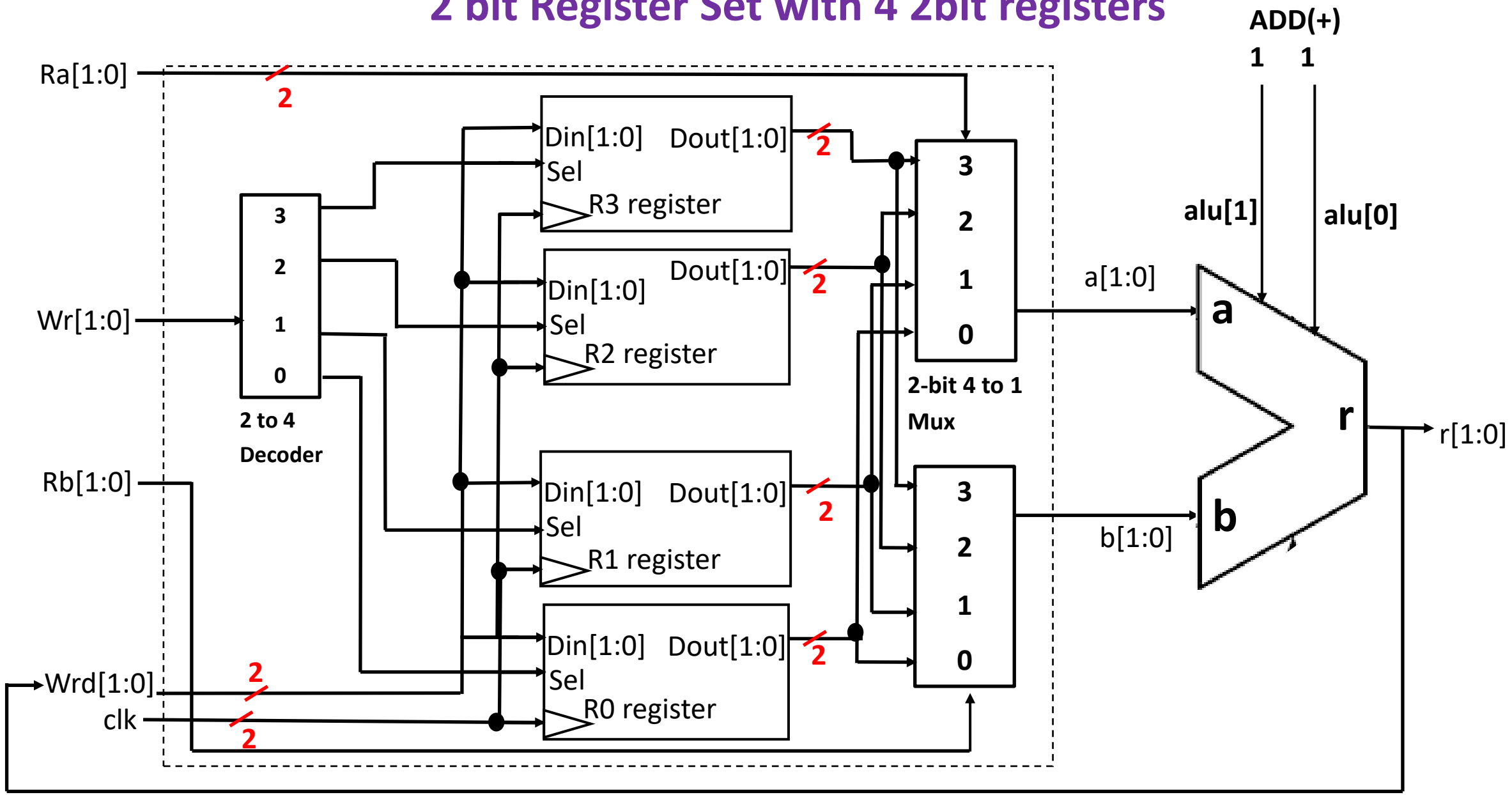


Figure: 2 bit Register set with 4 2-bit registers

Register Set Design
2 bit Register Set with 4 2bit registers

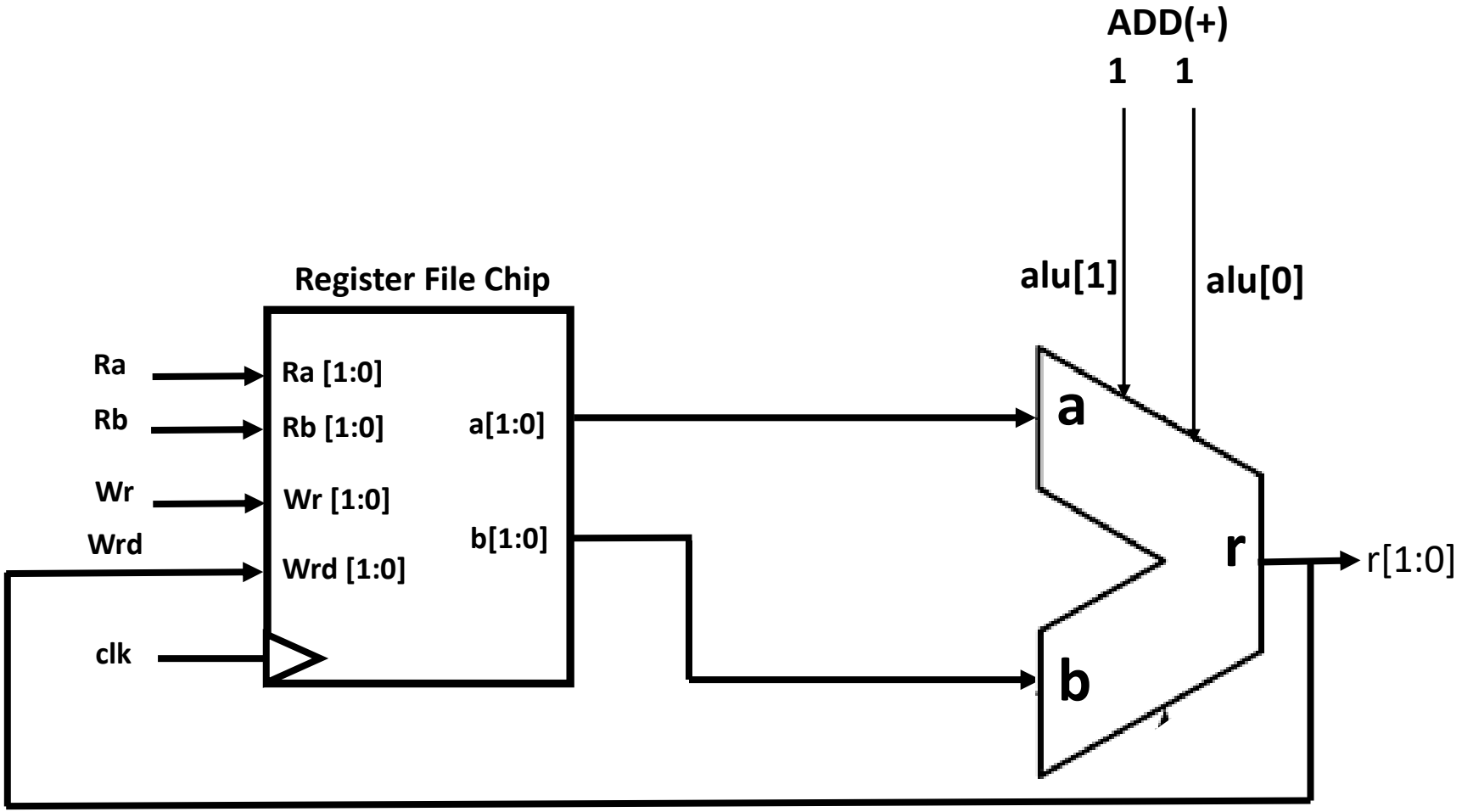


Figure: 2 bit Register set with 4 2-bit registers

Example: Register Set Design

Question: Design a 7-bit Register Set with 10 registers.

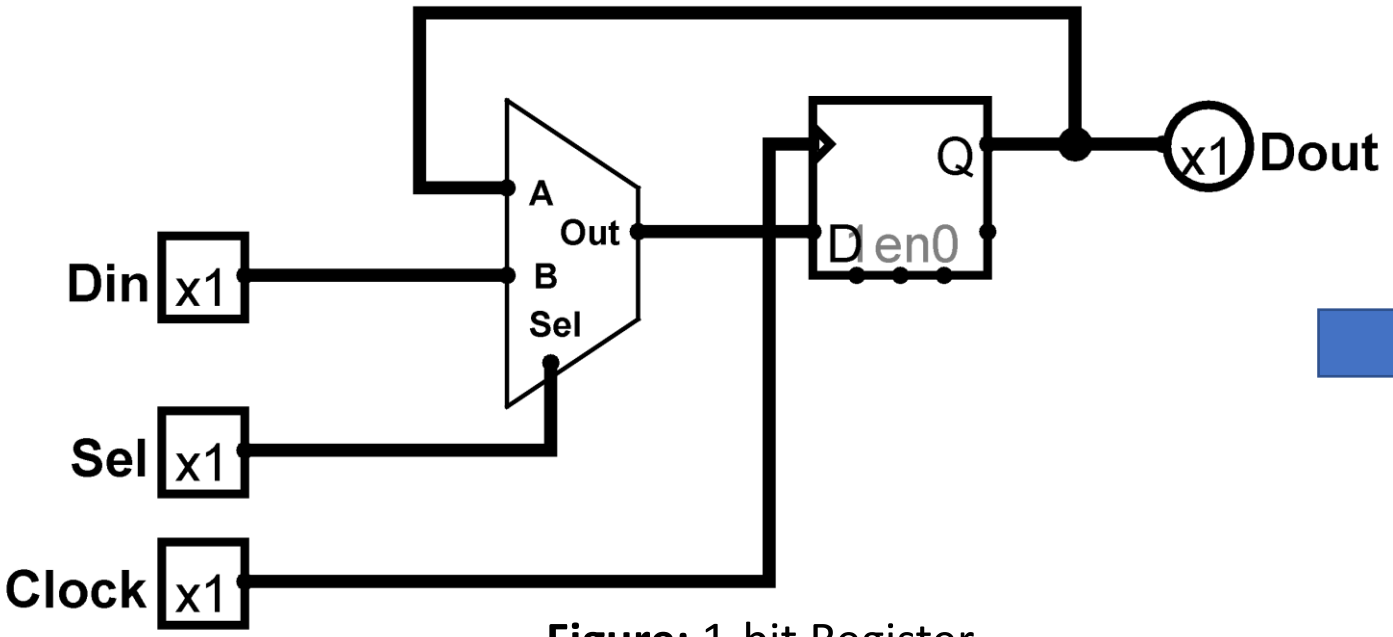


Figure: 1-bit Register

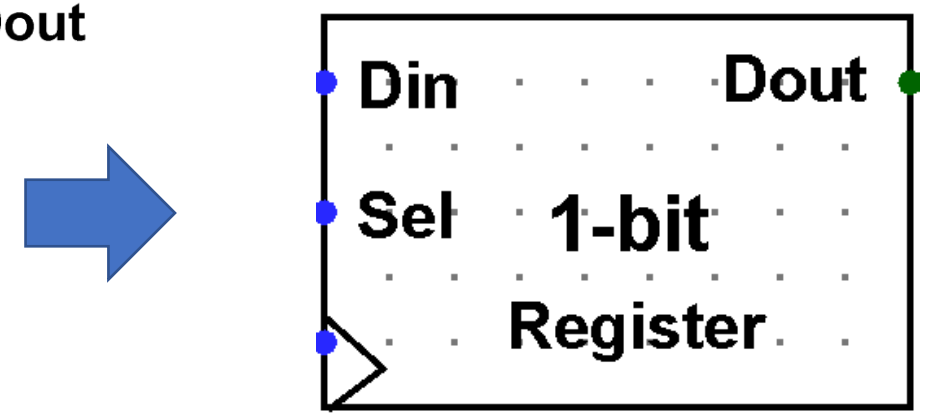


Figure: 1-bit Register Chip

Example: Register Set Design

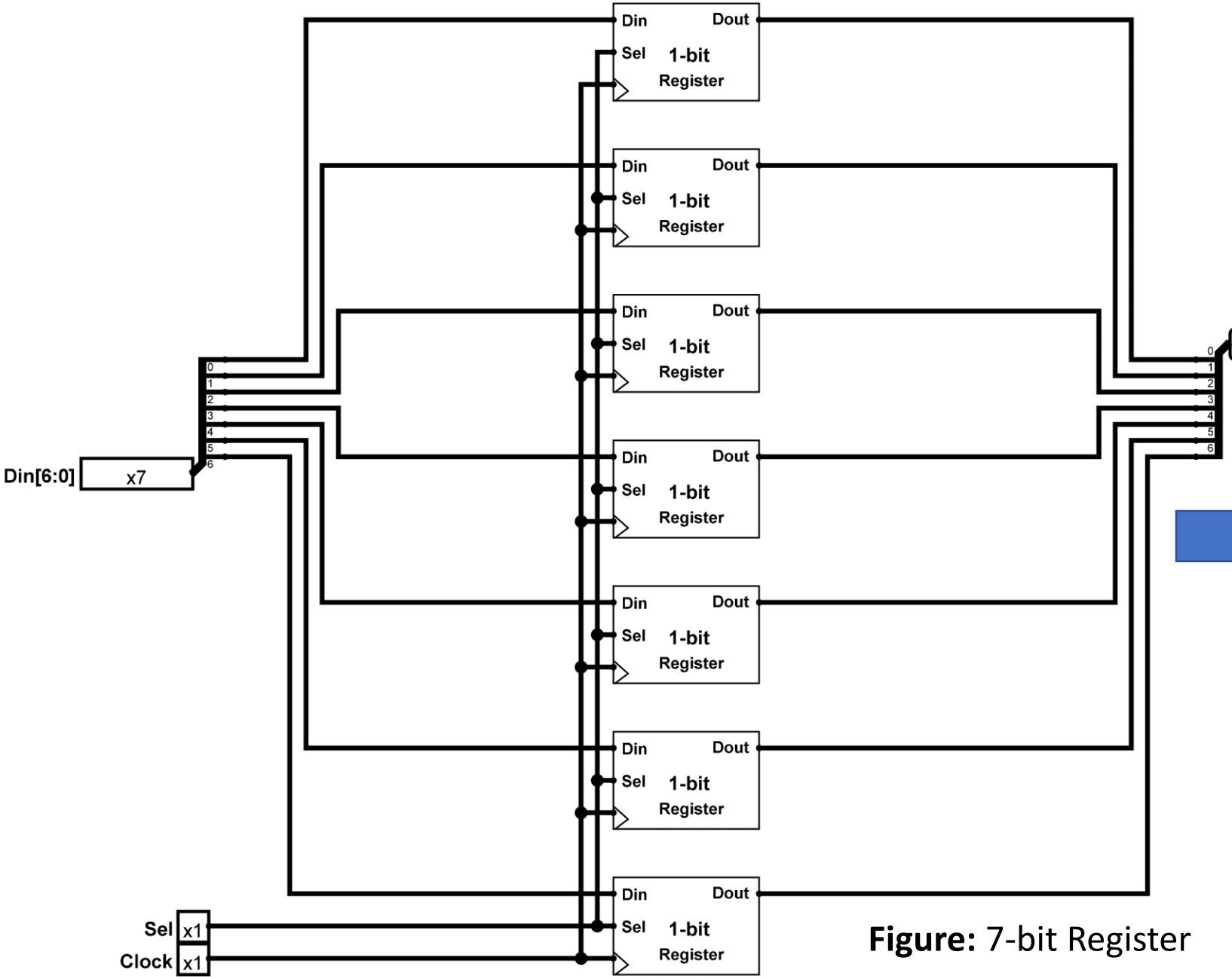


Figure: 7-bit Register

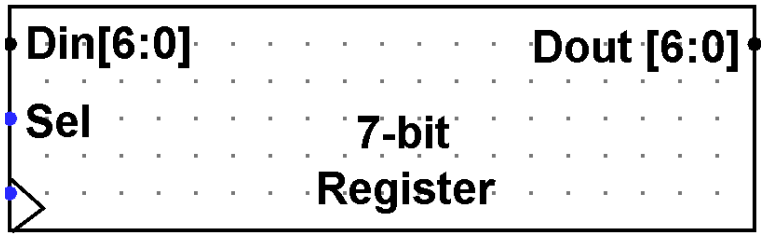


Figure: 7-bit Register Chip

Example: Register Set Design

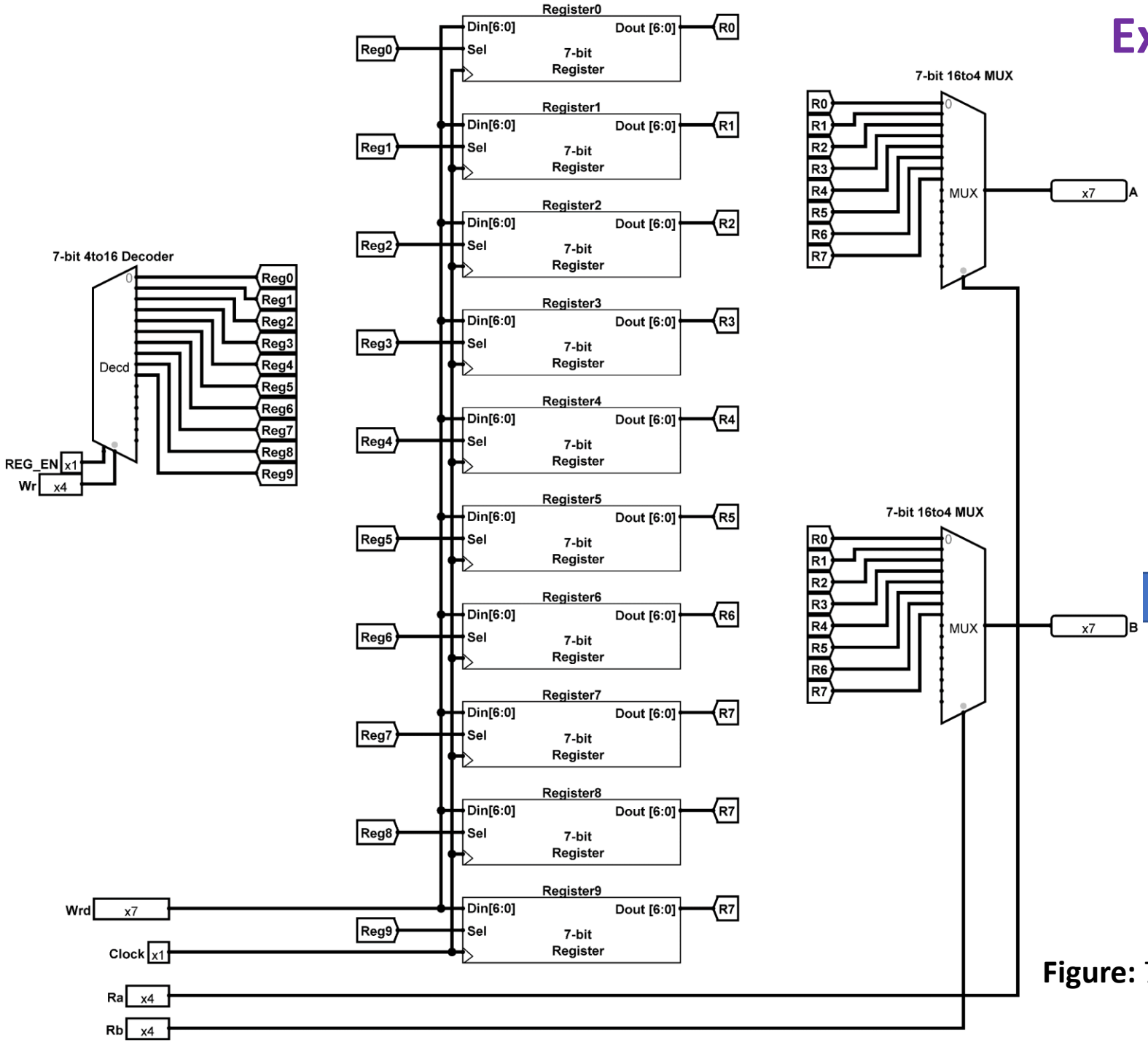


Figure: 7-bit Register Set with 10 Registers

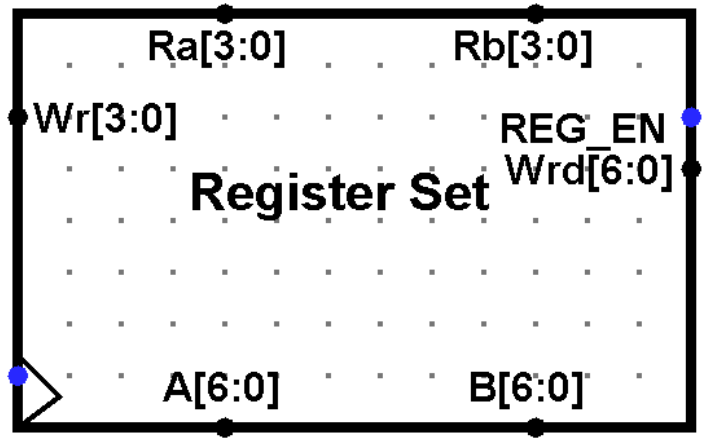


Figure: 7-bit Register Set with 10 Registers Chip

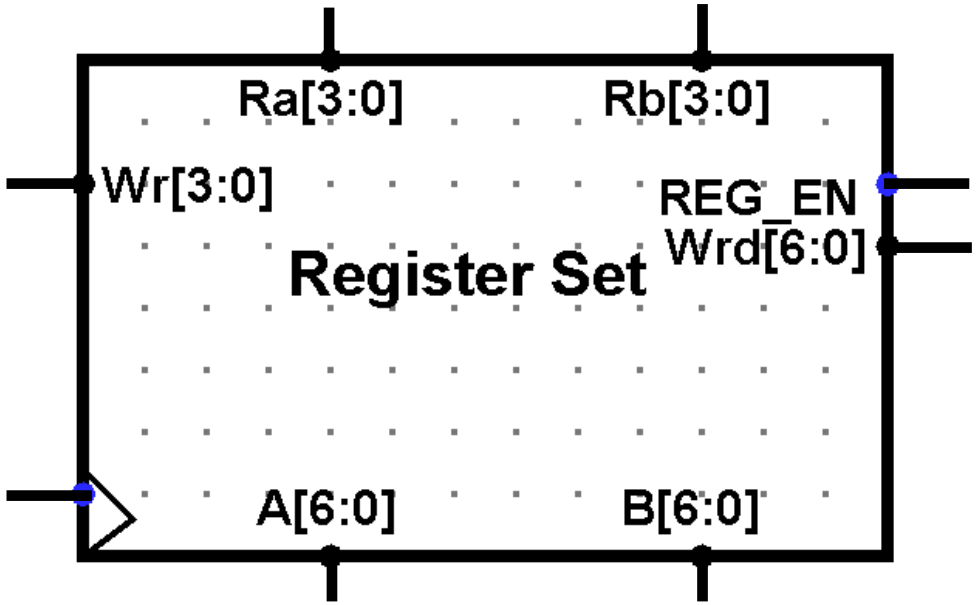
Exercises

1. Draw 2-bit/3-bit/4-bit/5-bit/6-bit/7-bit/8-bit Register.

2. Design 2-bit/3-bit/4-bit/5-bit/6-bit/7-bit/8-bit Register Set with 1/2/3/4/5/6/7/8 register/registers.

Exercises

3. Consider the following Register Set chip



	Data						
	6	5	4	3	2	1	0
R2	1	0	0	1	0	0	1
R5	1	0	0	0	1	0	1
R6	0	0	0	0	0	0	0

i. What is the size and word size of Register Set?
ii. If Ra = 0101 and Rb = 000, then what will the value of A and B?
iii. If Wr = 0010, Wrd = 1110000, REG_EN = 0, Ra = 010, Rb = 110, then what will be the value of A and B after next clock pulse?
iv. Design this Register Set chip.

Thank You ☺