Memory Hierarchy
&
Register Set Design I

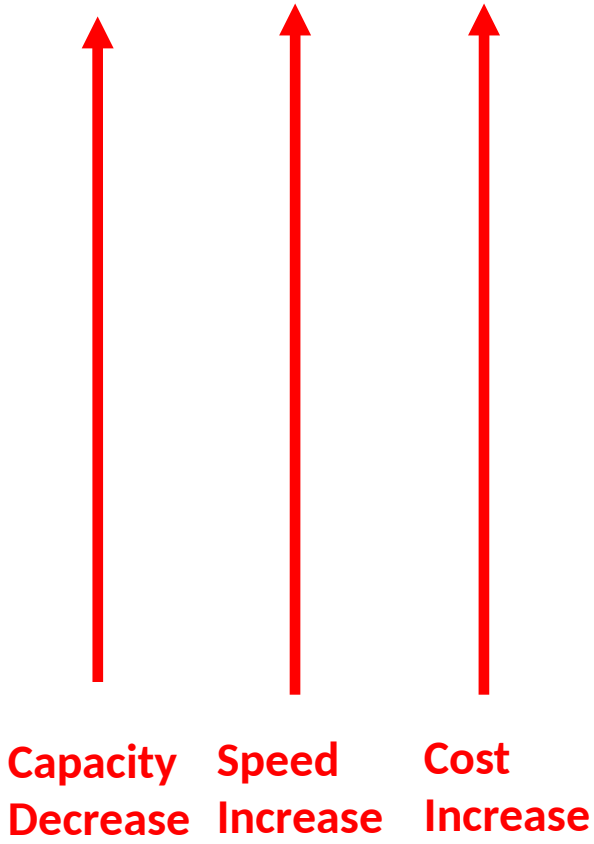
Memory Hierarchy

Memory Technologies

	Capacity	Latency	Cost/GB
Register	1000s of bits	20 ps	$$$$
SRAM	~10 KB-10 MB	1-10 ns	~$1000
DRAM	~10 GB	80 ns	~$10
Flash (SSD)*	~100 GB	100 us	~$1
Hard disk*	~1 TB	10 ms	~$0.10

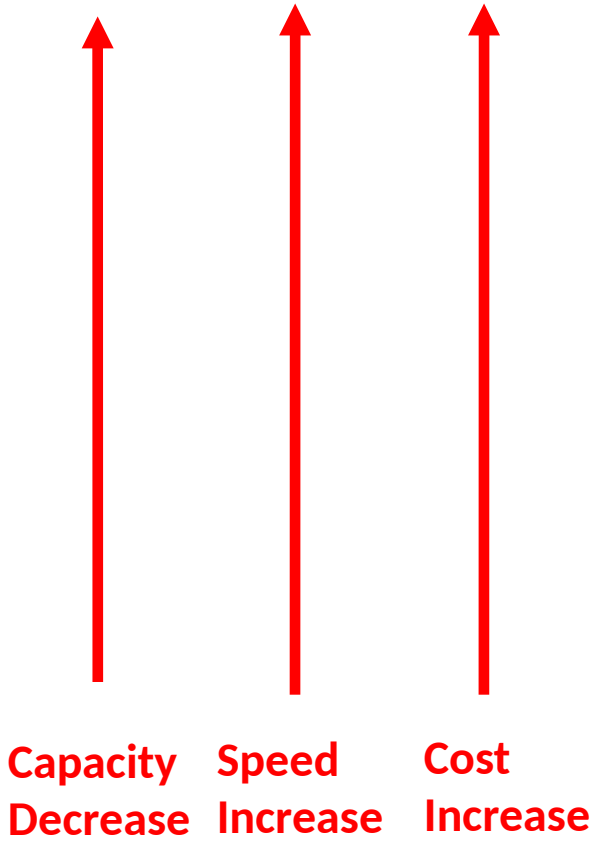
Memory Hierarchy

	Capacity	Latency	Cost/GB
Register	1000s of bits	20 ps	$$$$
SRAM	~10 KB-10 MB	1-10 ns	~$1000
DRAM	~10 GB	80 ns	~$10
Flash (SSD)*	~100 GB	100 us	~$1
Hard disk*	~1 TB	10 ms	~$0.10



Memory Hierarchy

	Capacity	Latency	Cost/GB
Register	1000s of bits	20 ps	$$$$
SRAM	~10 KB-10 MB	1-10 ns	~$1000
DRAM	~10 GB	80 ns	~$10
Flash (SSD)*	~100 GB	100 us	~$1
Hard disk*	~1 TB	10 ms	~$0.10



We want large, fast and cheap memory. But

Large memories are slow!

Fast memories are expensive!

Memory Hierarchy

Idea: Can we use a hierarchal system of memories with different tradeoffs to emulate a large, fast, cheap memory?

Memory Hierarchy

– Programming model: Single memory, single address space

– Machine transparently stores data in fast or slow memory, depending on usage patterns.

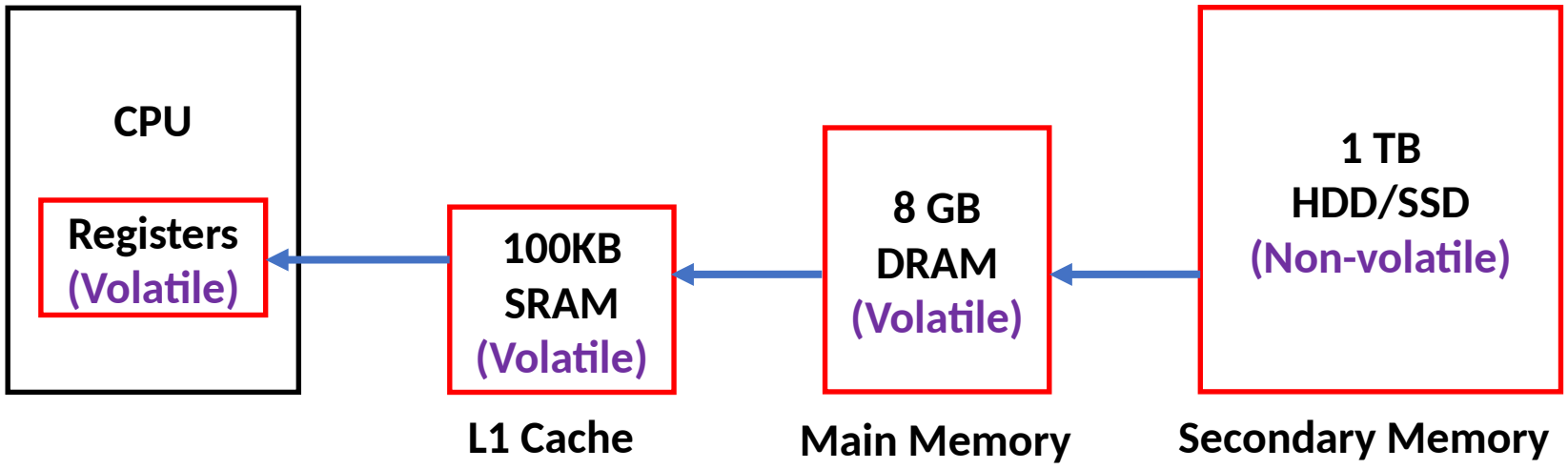
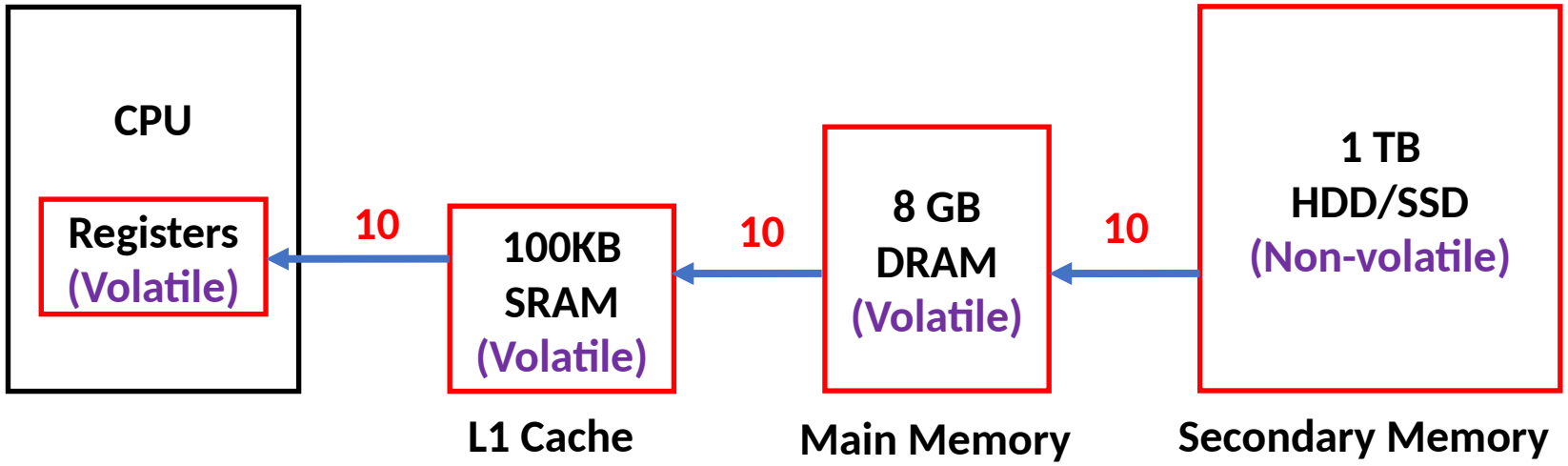


Figure: Memory Hierarchy

Volatile Memory = It only contains data when it is powered ON. It will lose its contents after power off.

Non-volatile Memory = It contains data even after power off.

Memory Hierarchy



Data are stored in Registers before they can be sent to ALU for operation.

Cache memory stores data that are frequently accessed.

Program's instruction and data are temporarily first loaded into Main Memory before running

OS and other programs are stored in Secondary Memory permanently.

Figure: How Memory Hierarchy works

Registers

D Flip-flop

D	Q(t + 1)
0	0
1	1

Fig: D Flip-flop truth table

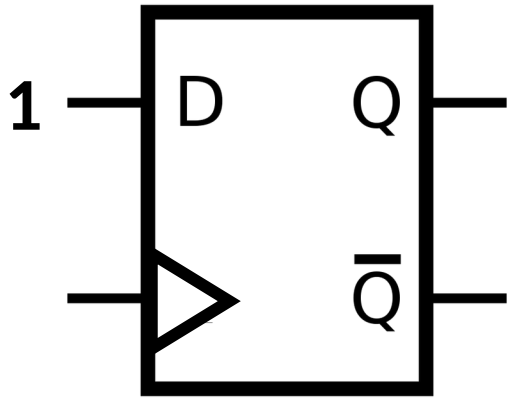


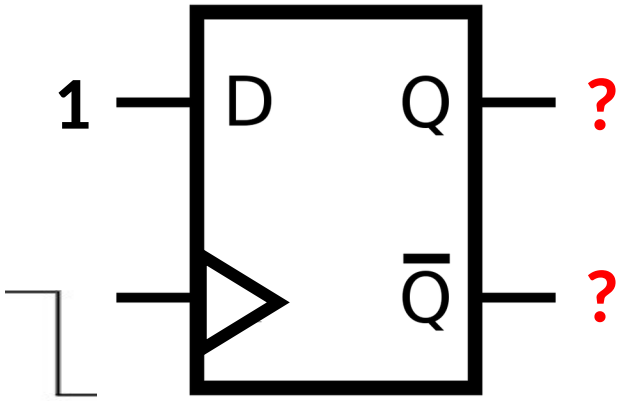
Fig: D Flip-Flop

D Flip-Flop

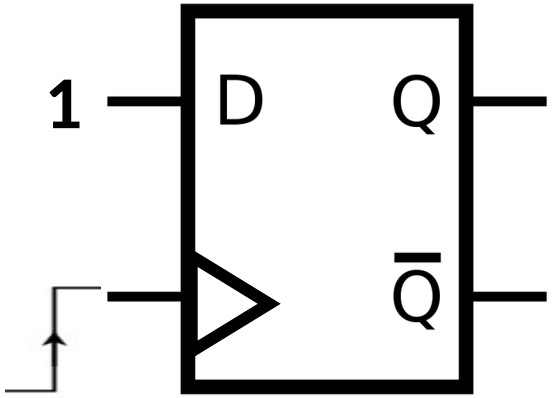
Assume, Q=0 which means D-FF has 0 value stored.

D	Q(t + 1)
0	0
1	1

Fig: D Flip-flop truth table



At Negative Clock Edge



At Positive Clock Edge

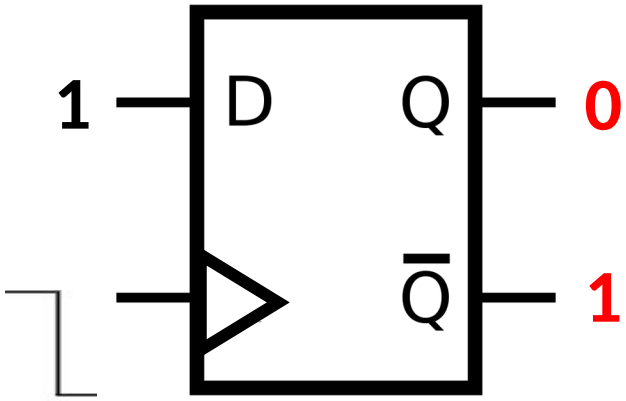
Fig: How D Flip-Flop works.

D Flip-Flop

Assume, Q=0 which means D-FF has 0 value stored.

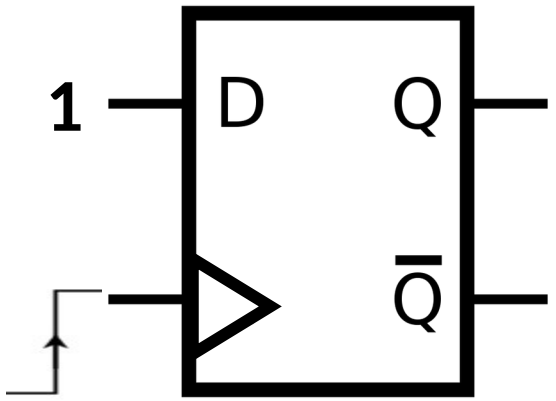
D	Q(t + 1)
0	0
0	1

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

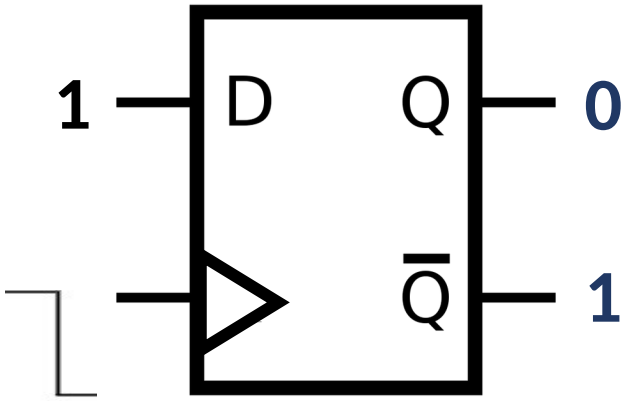
Fig: How D Flip-Flop works.

D Flip-Flop

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

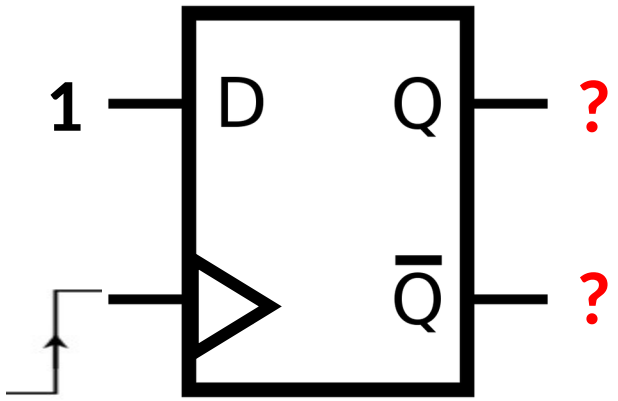
D	Q(t + 1)
0	0
0	1

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

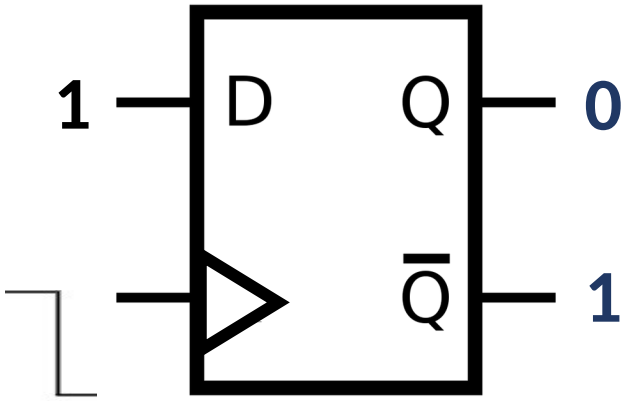
Fig: How D Flip-Flop works.

D Flip-Flop

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

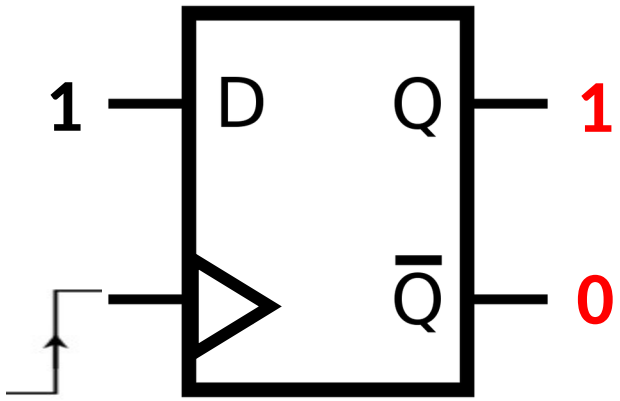
D	Q(t + 1)
0	0
0	1

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.

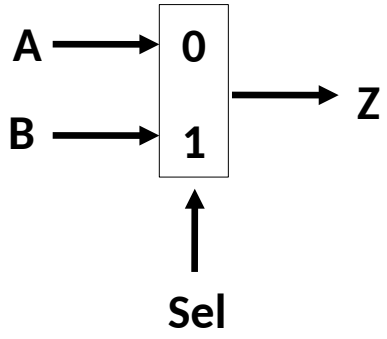


At Positive Clock Edge

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

Fig: How D Flip-Flop works.

1-bit Register



2 to 1 MUX

Sel	Z
0	A
1	B

$$Z = \acute{S}el . A + Sel . B$$

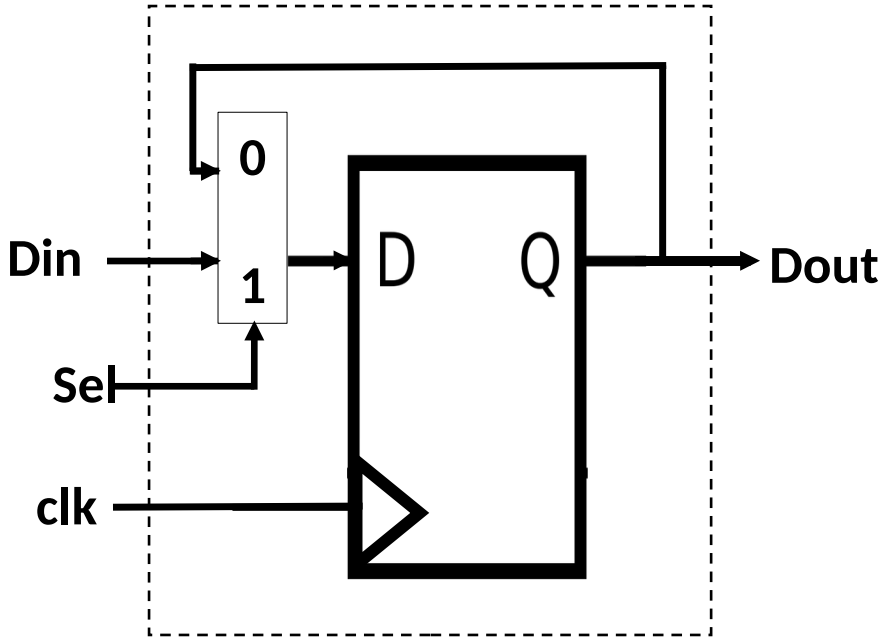
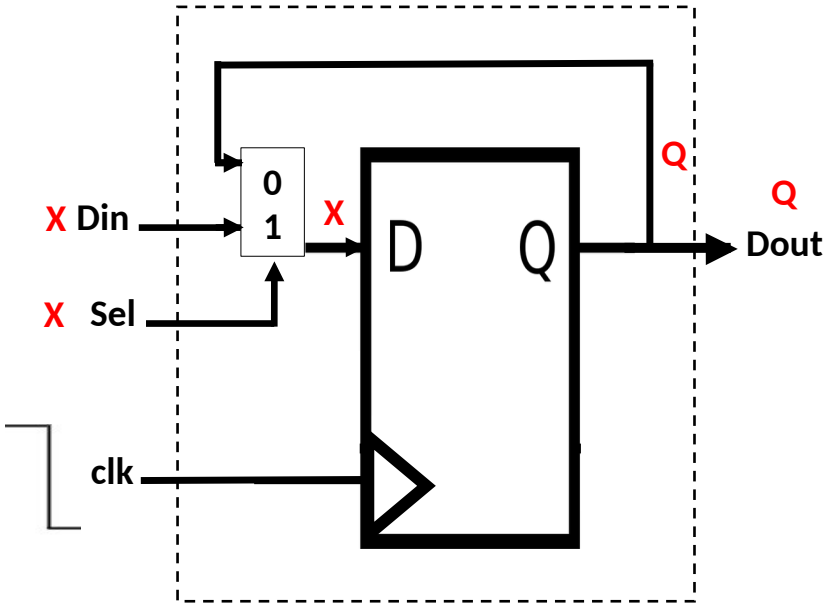


Fig: 1-bit Register

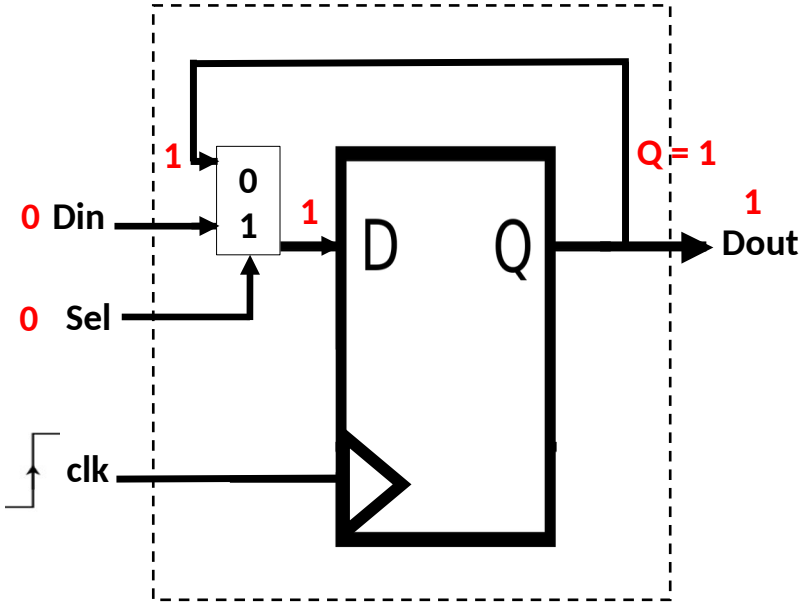
1-bit Register



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

Fig: How 1-bit Register Works

1-bit Register

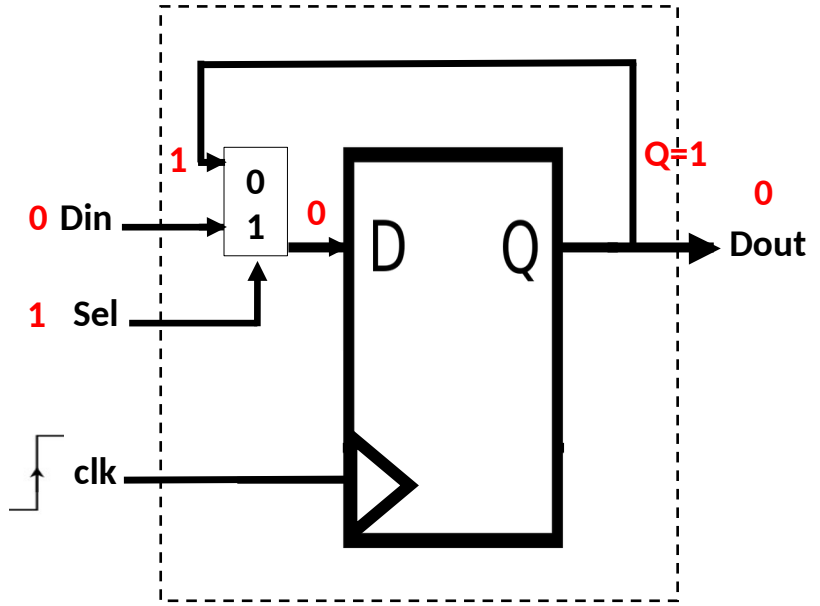


At positive clock cycle,
D Flip-flop will update its content.

When
So, Data will remain same in D-FF.

Fig: How 1-bit Register Works

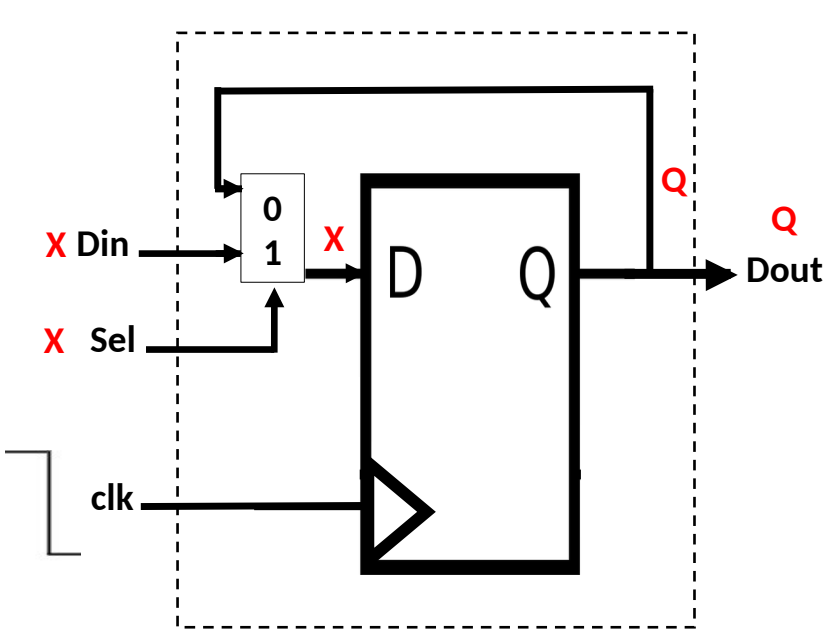
1-bit Register



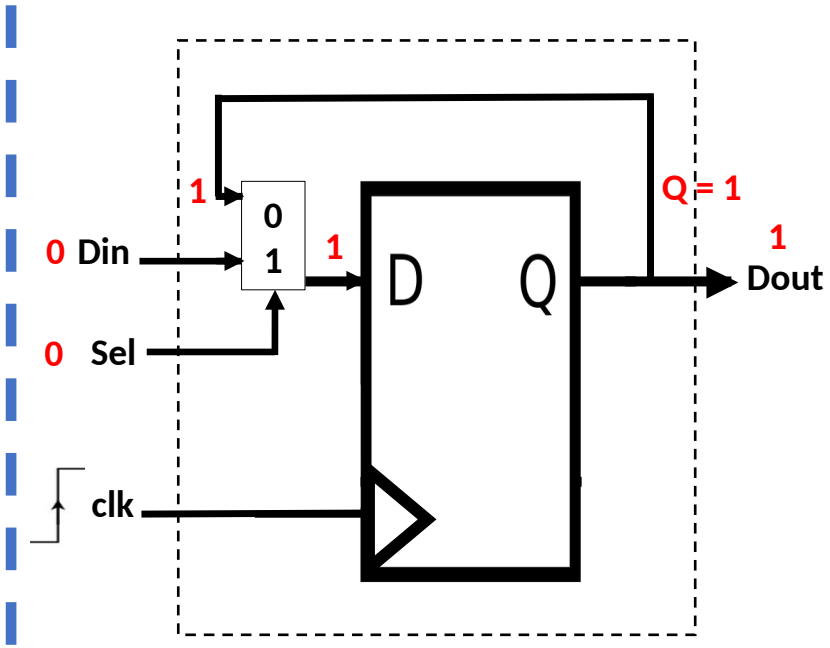
At positive clock cycle,
D Flip-flop will update its content.

When
So, Data will be updated in D-FF.

1-bit Register

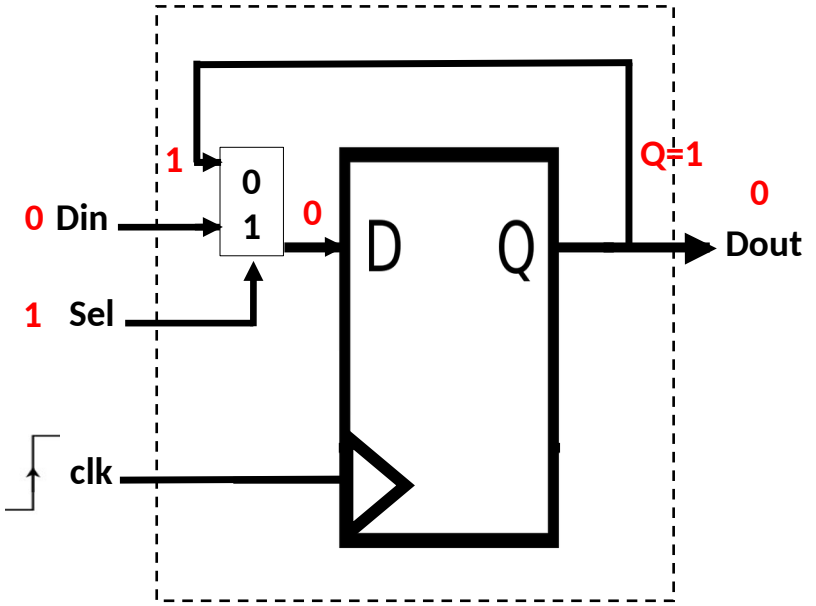


At negative clock cycle,
D Flip-flop will not update its content regardless of and value.



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

When
So, Data will remain same in D-FF.



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

When
So, Data will be updated in D-FF.

Fig: How 1-bit Register Works

1-bit Register

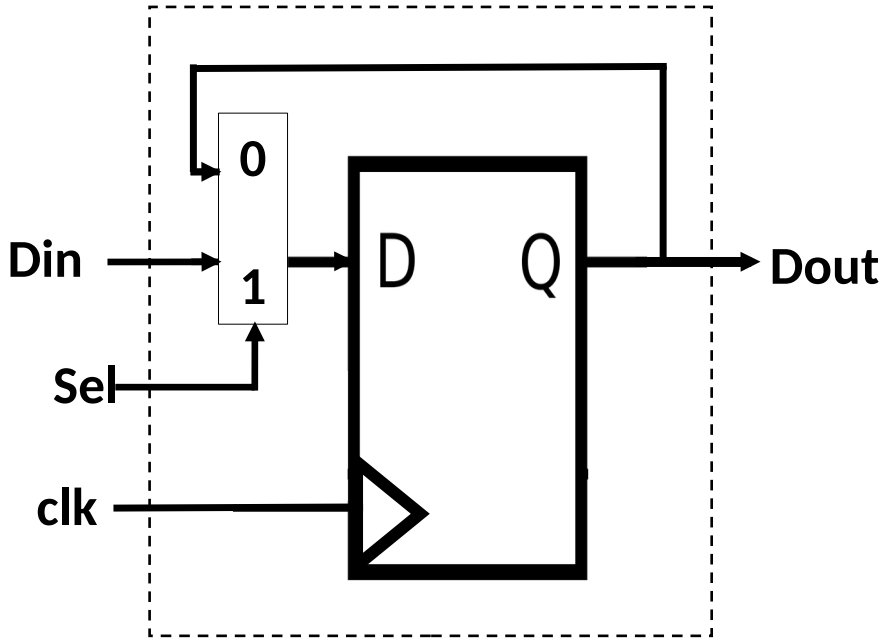


Fig: 1-bit Register

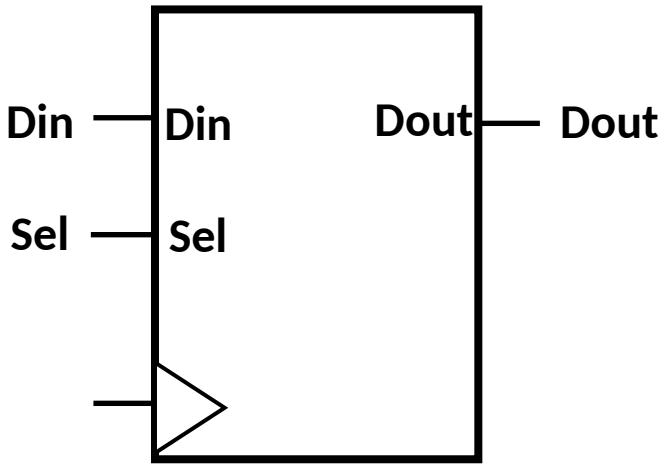


Figure: 1 bit register chip

2-bit Register

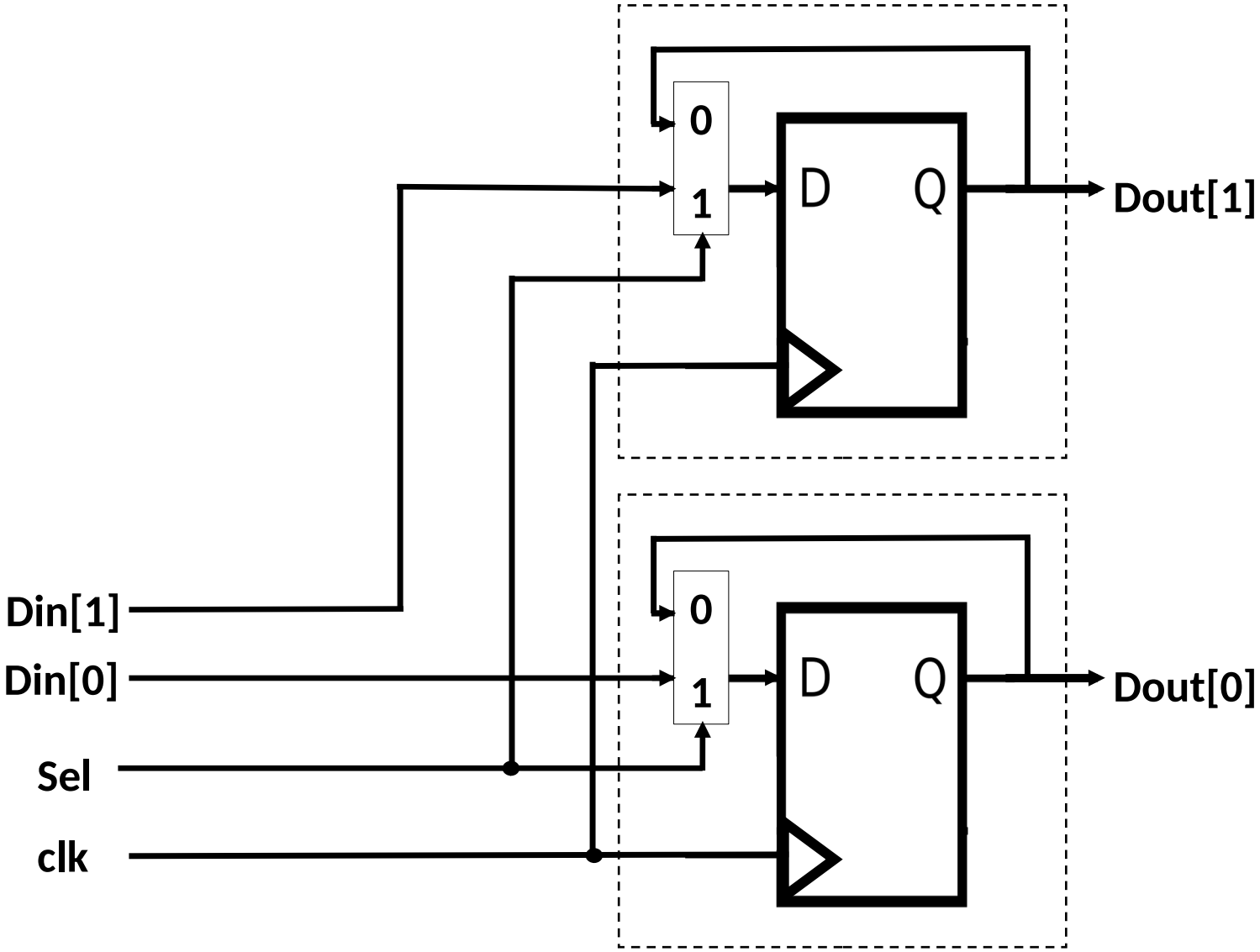


Figure: 2 bit Register

2-bit Register

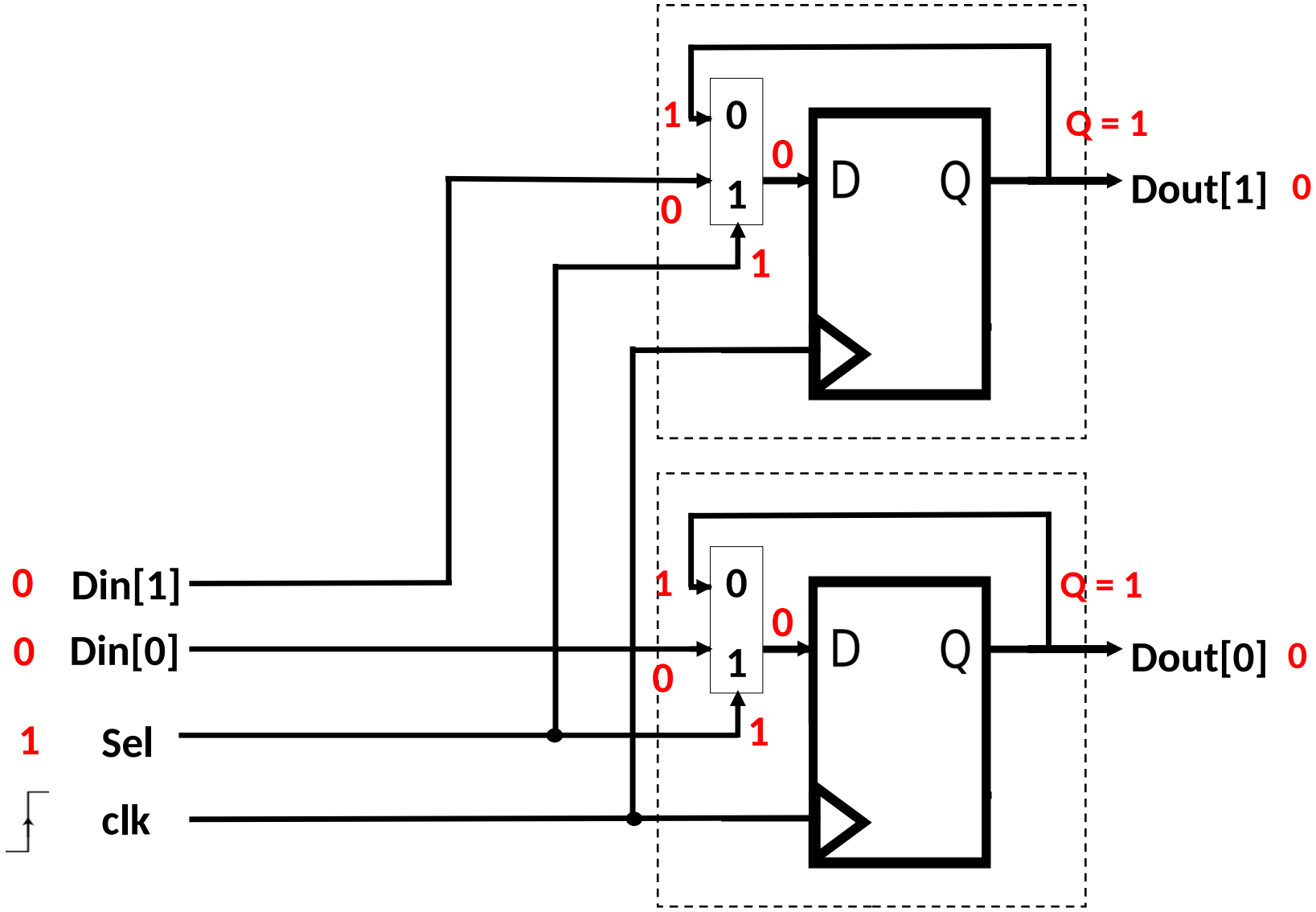


Figure: How 2 bit Register Works

2-bit Register

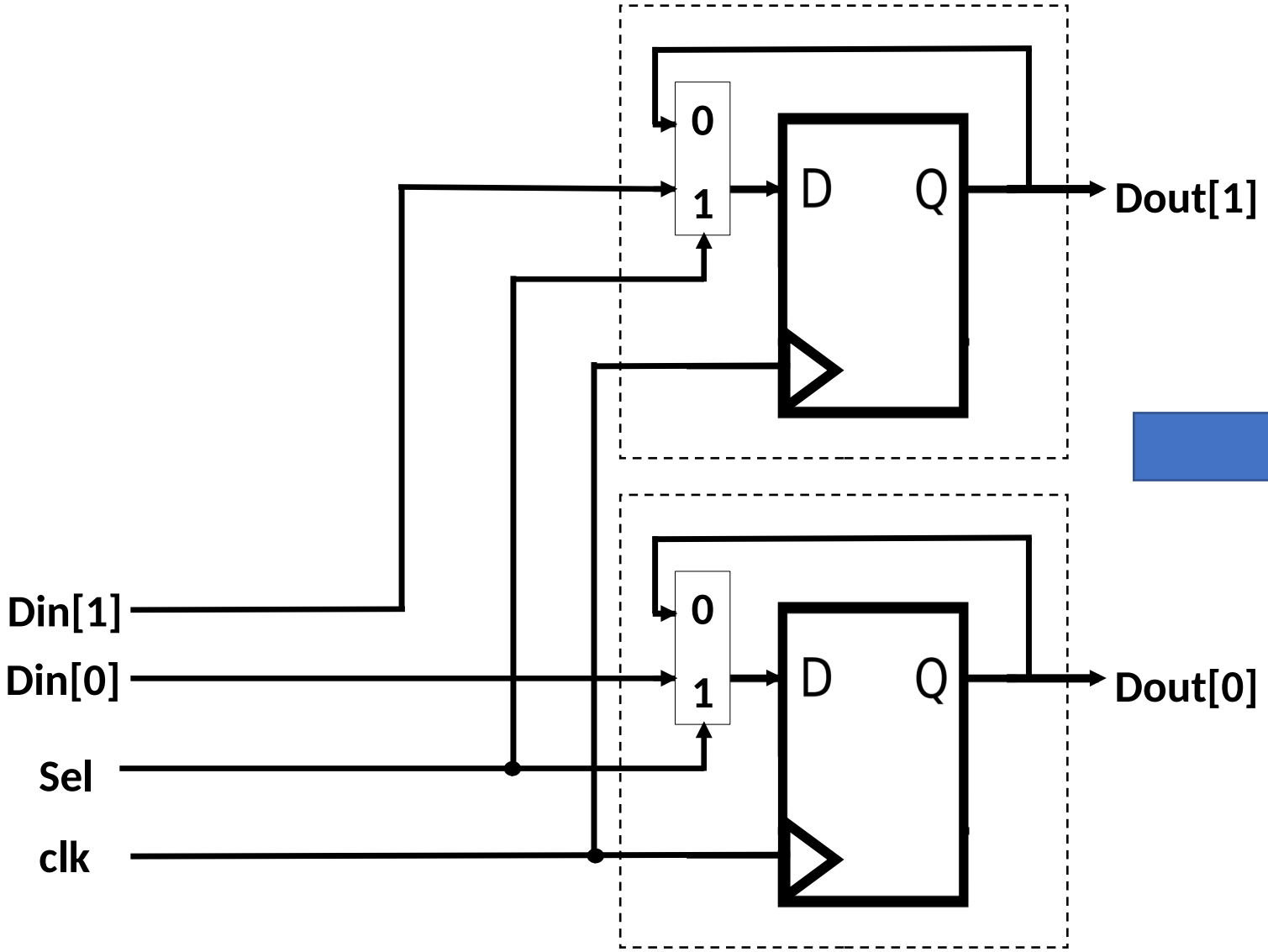


Figure: 2 bit Register

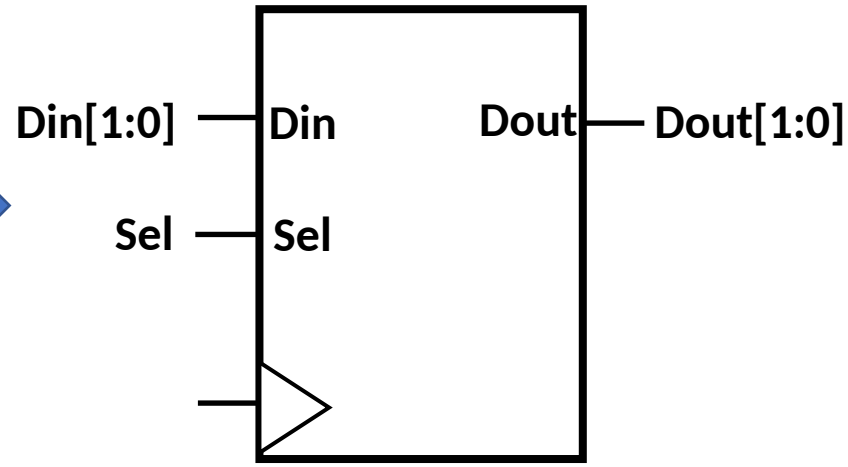


Figure: 2 bit register chip

2-bit Register

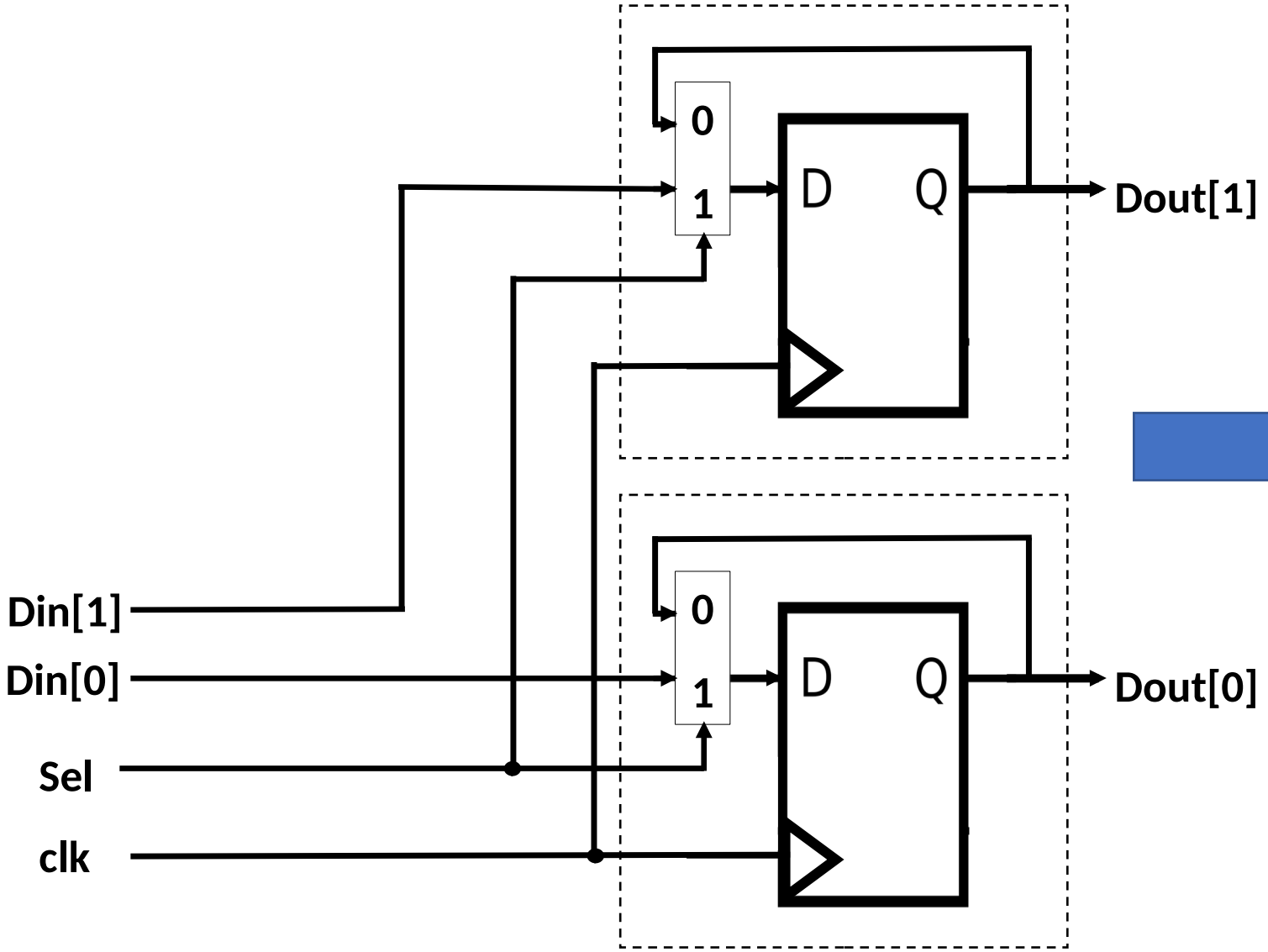


Figure: 2 bit Register

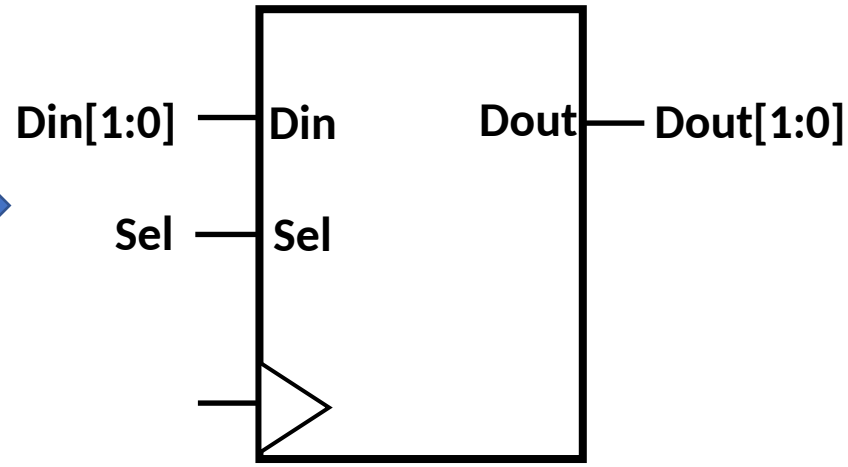


Figure: 2 bit register chip

Homework:
Similarly Design 3bit/4bit Register

Register Set Design I

1-bit CPU

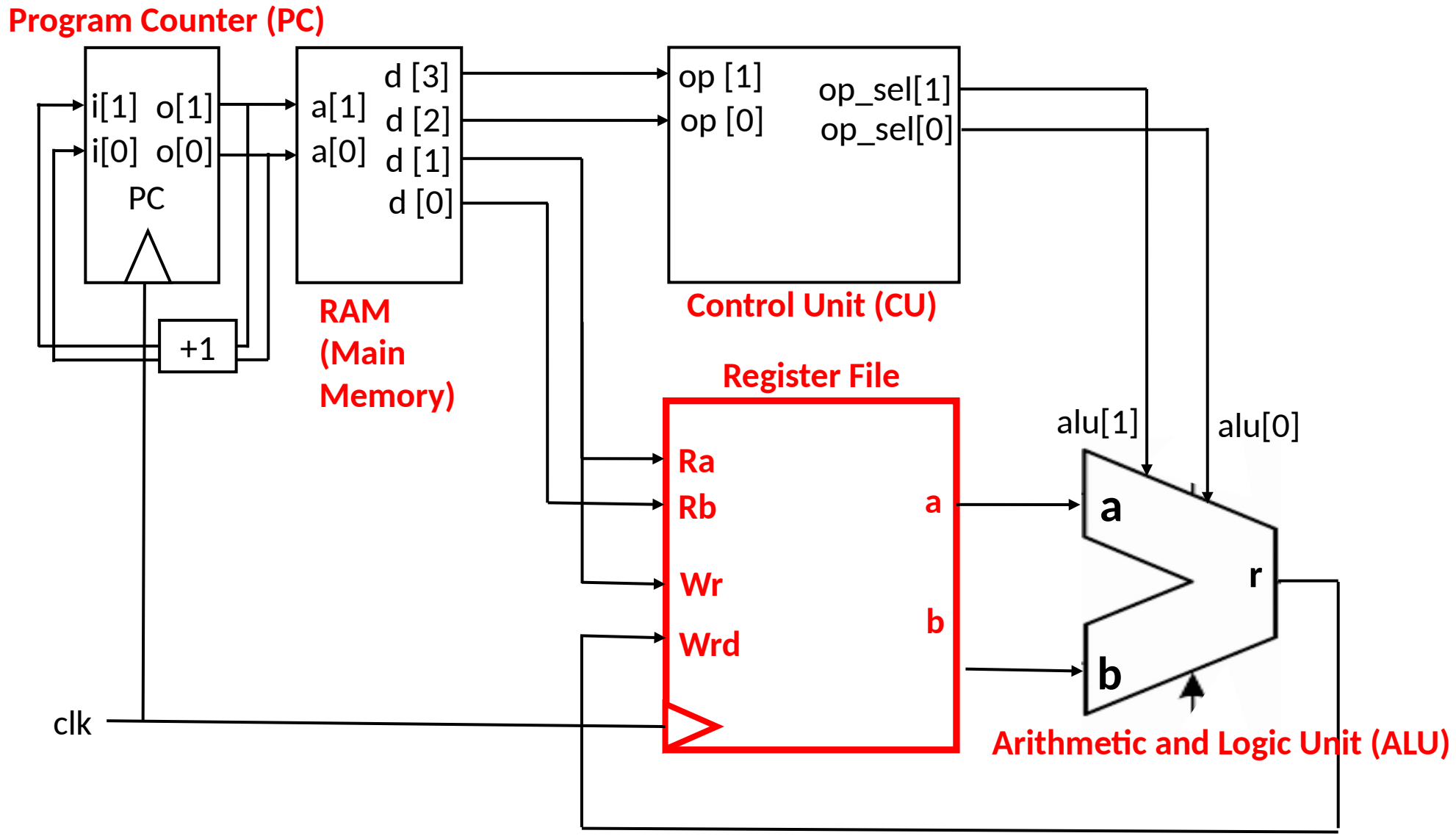


Figure: 1-bit CPU

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.

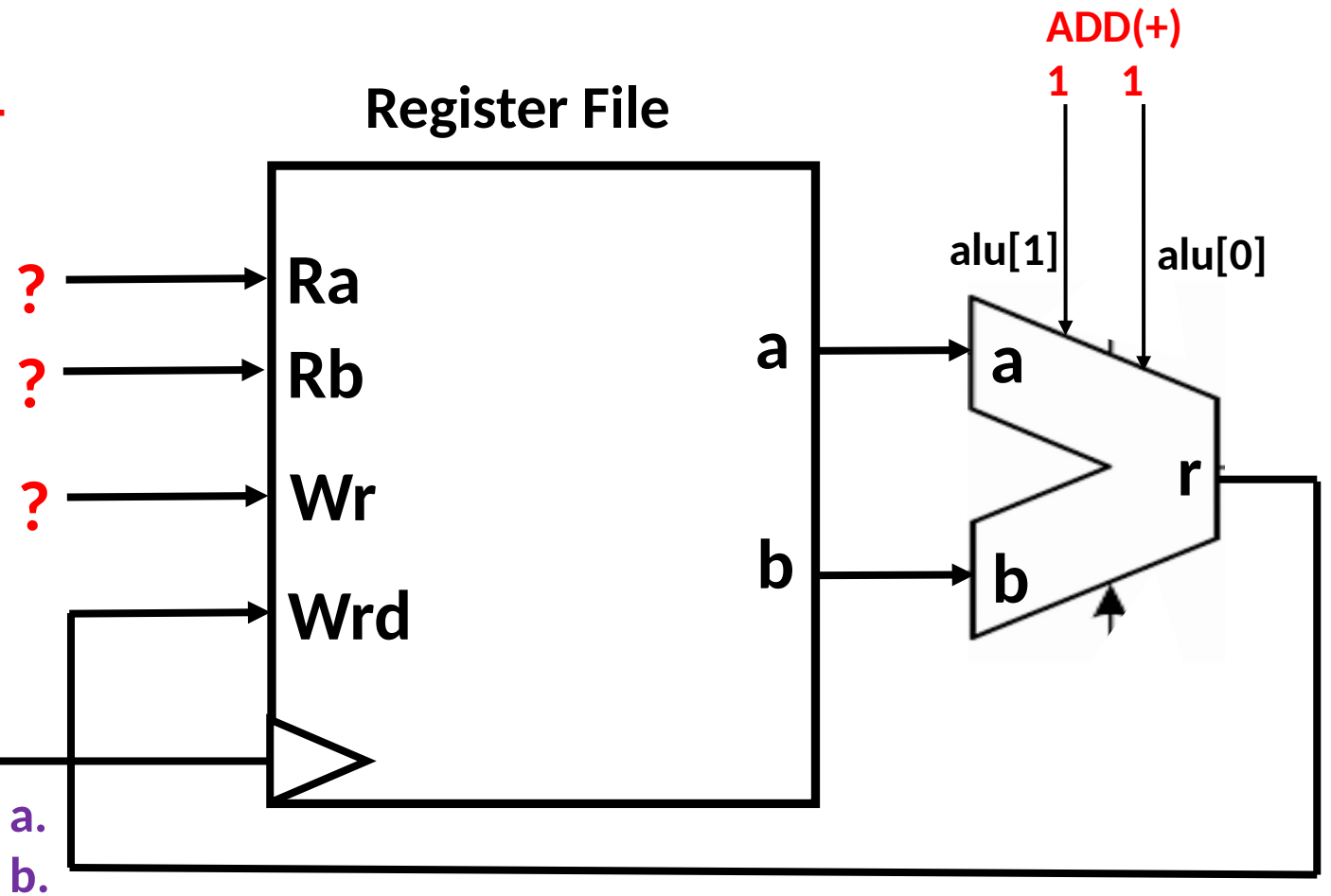


Figure: How Register set operates in CPU

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.

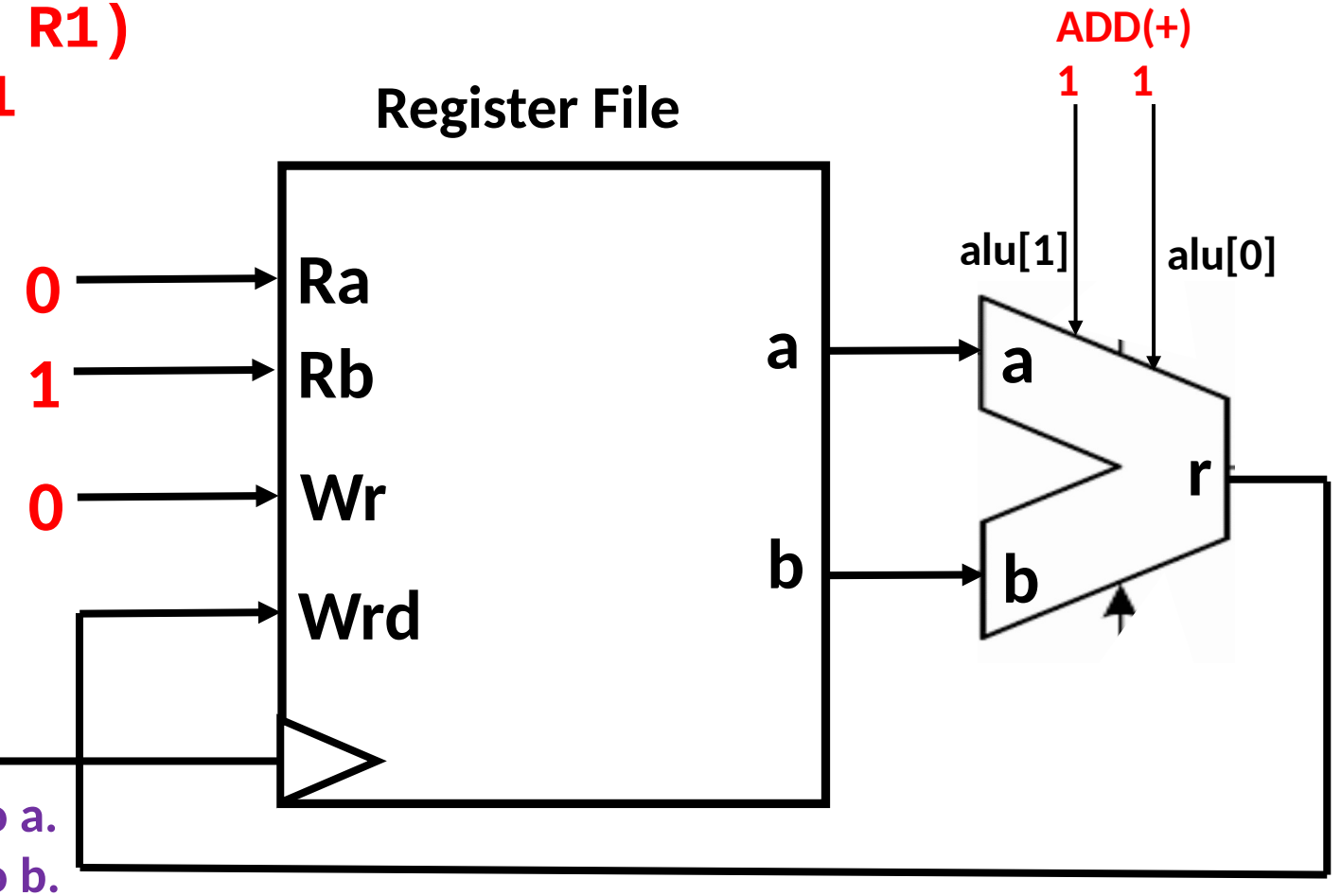


Figure: How Register set operates in CPU

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.

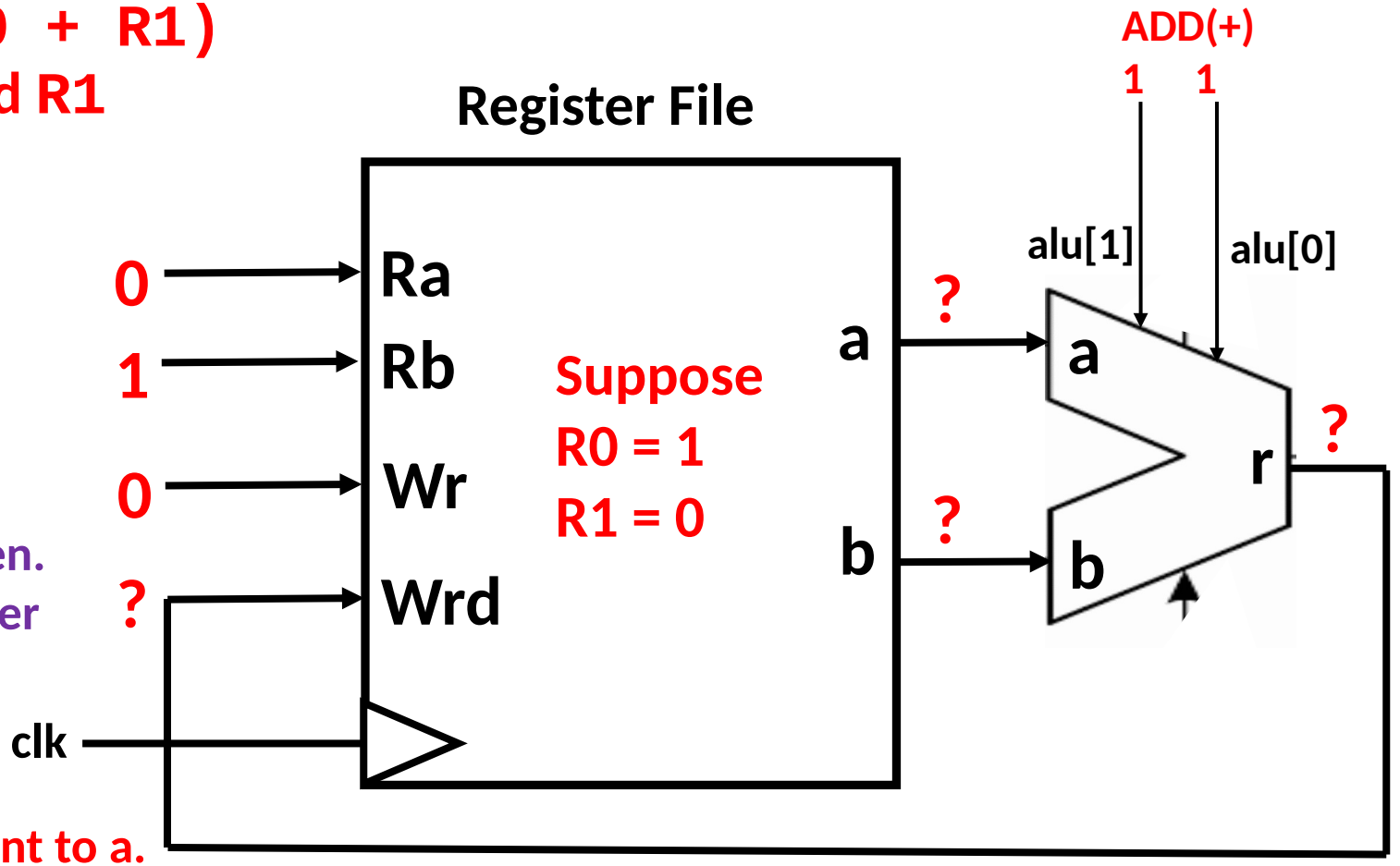


Figure: How Register set operates in CPU

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.

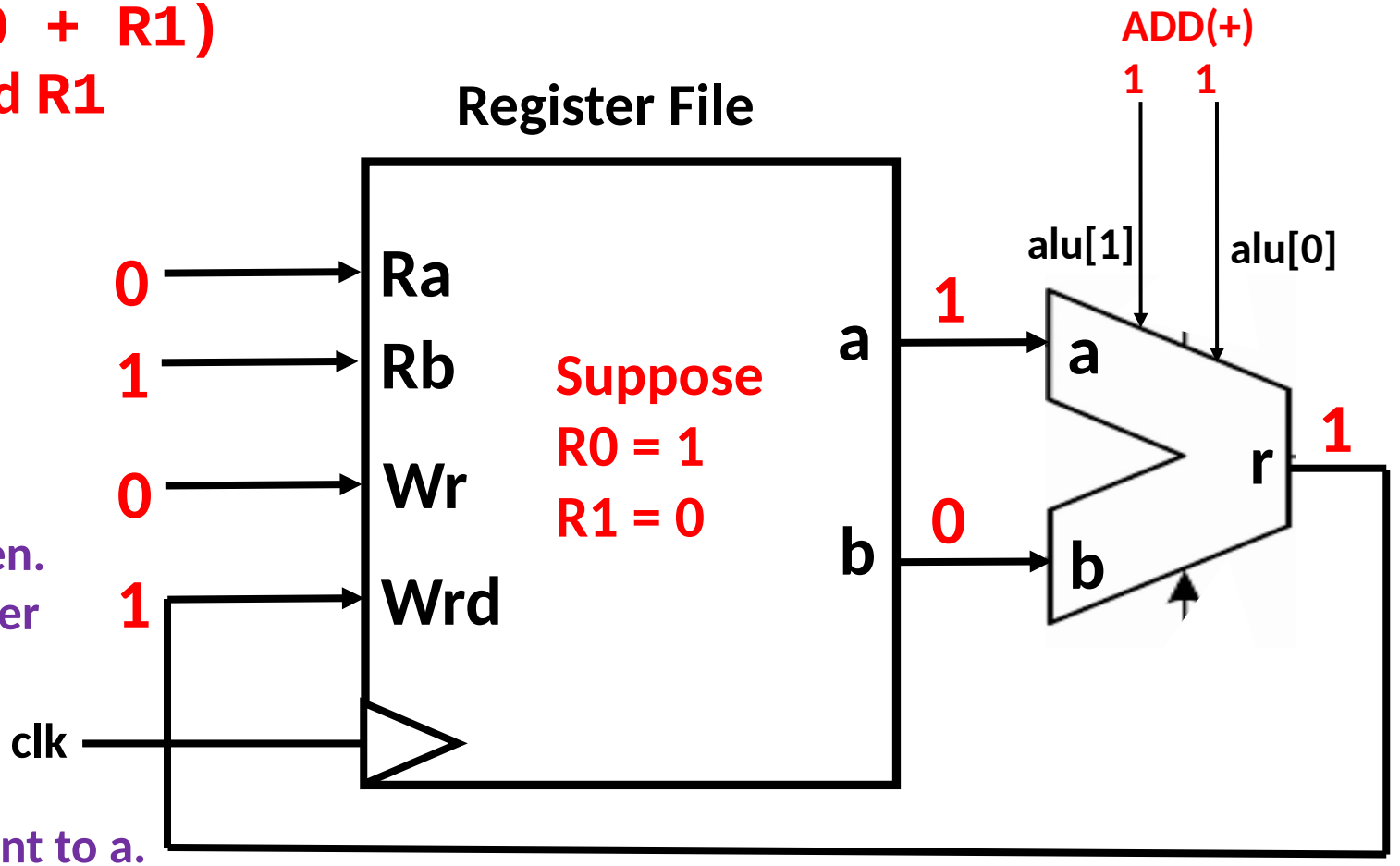
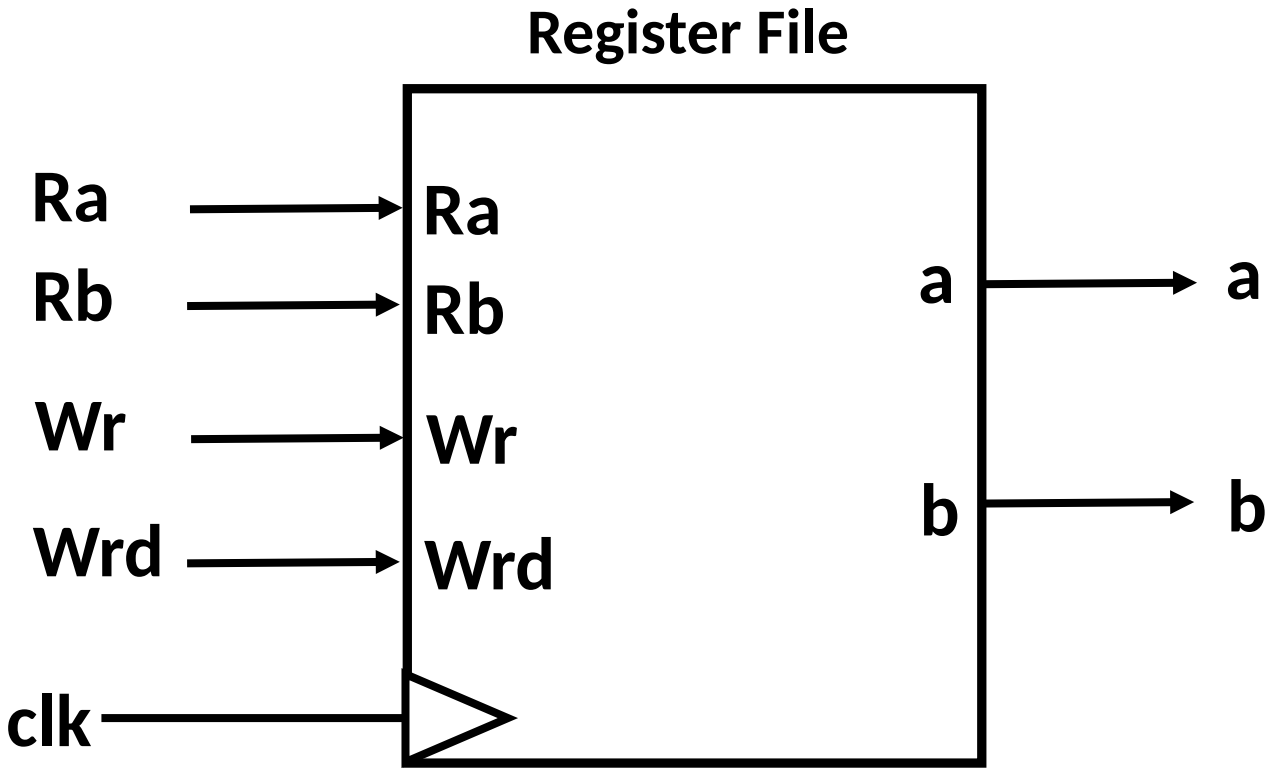


Figure: How Register set operates in CPU

Register Set Design

1bit Register Set with 2 1bit registers



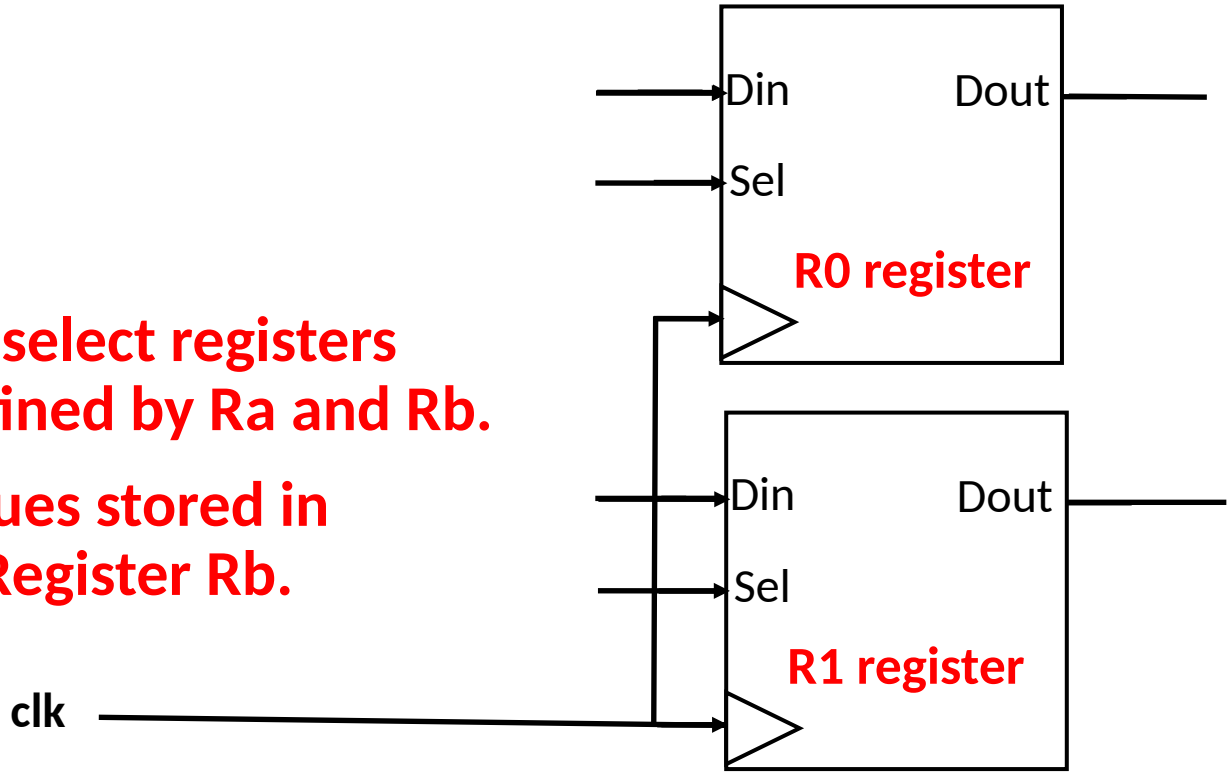
We will be designing this Register File!

Register Set Design

1bit Register Set with 2 1bit 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.

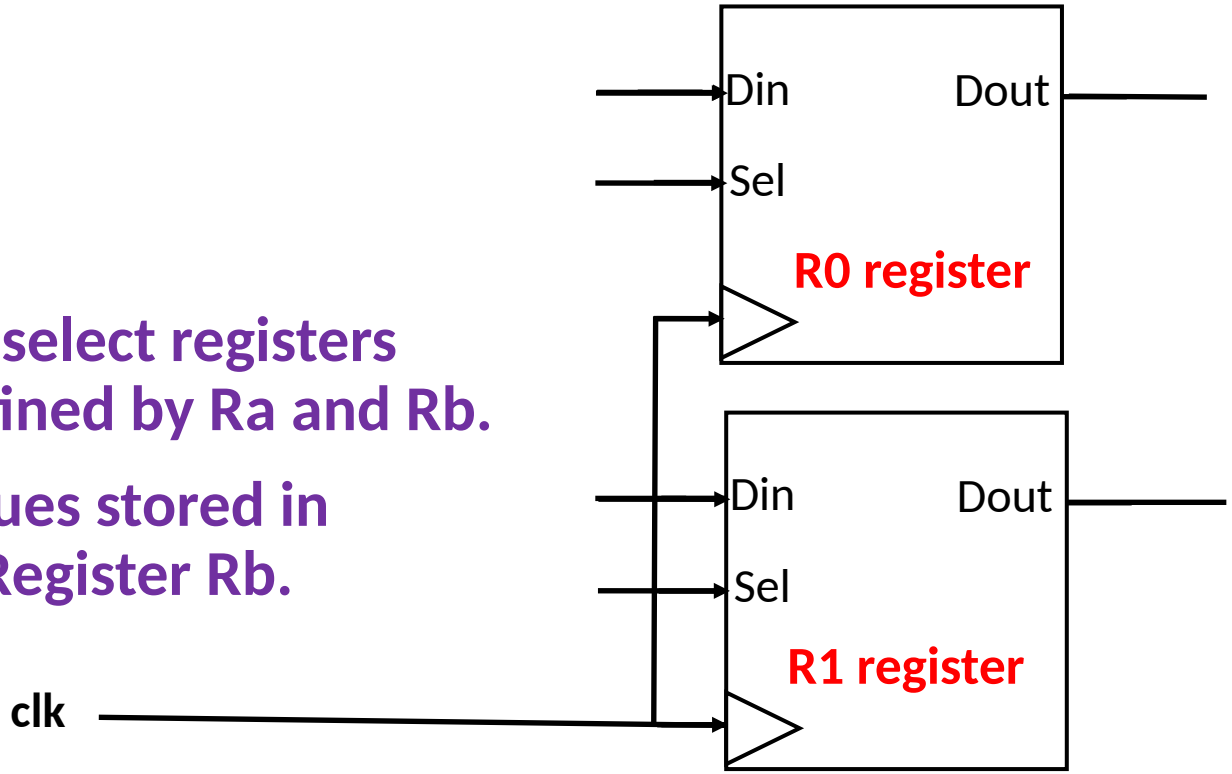
We just need values stored in Register Ra and Register Rb.

Register Set Design

1bit Register Set with 2 1bit 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.

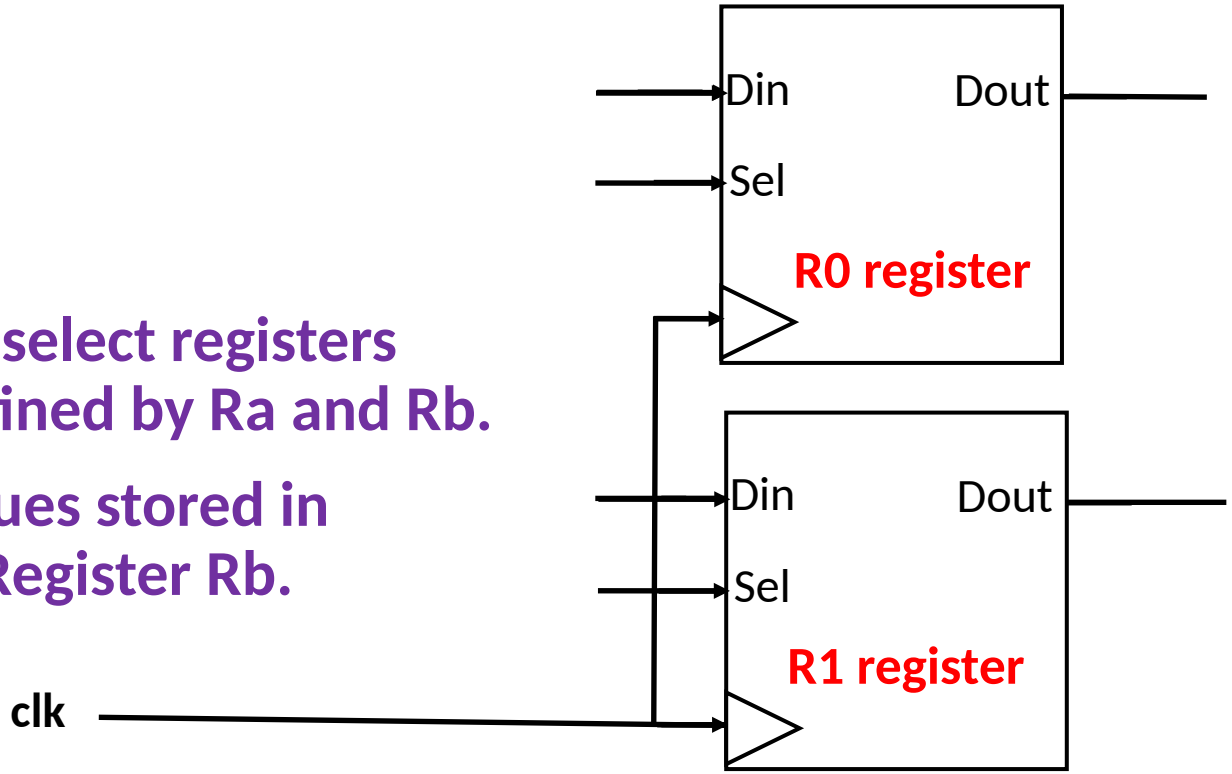
Then, We will have to send data of selected register to a (Ra) and b (Rb).

Register Set Design

1bit Register Set with 2 1bit 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).

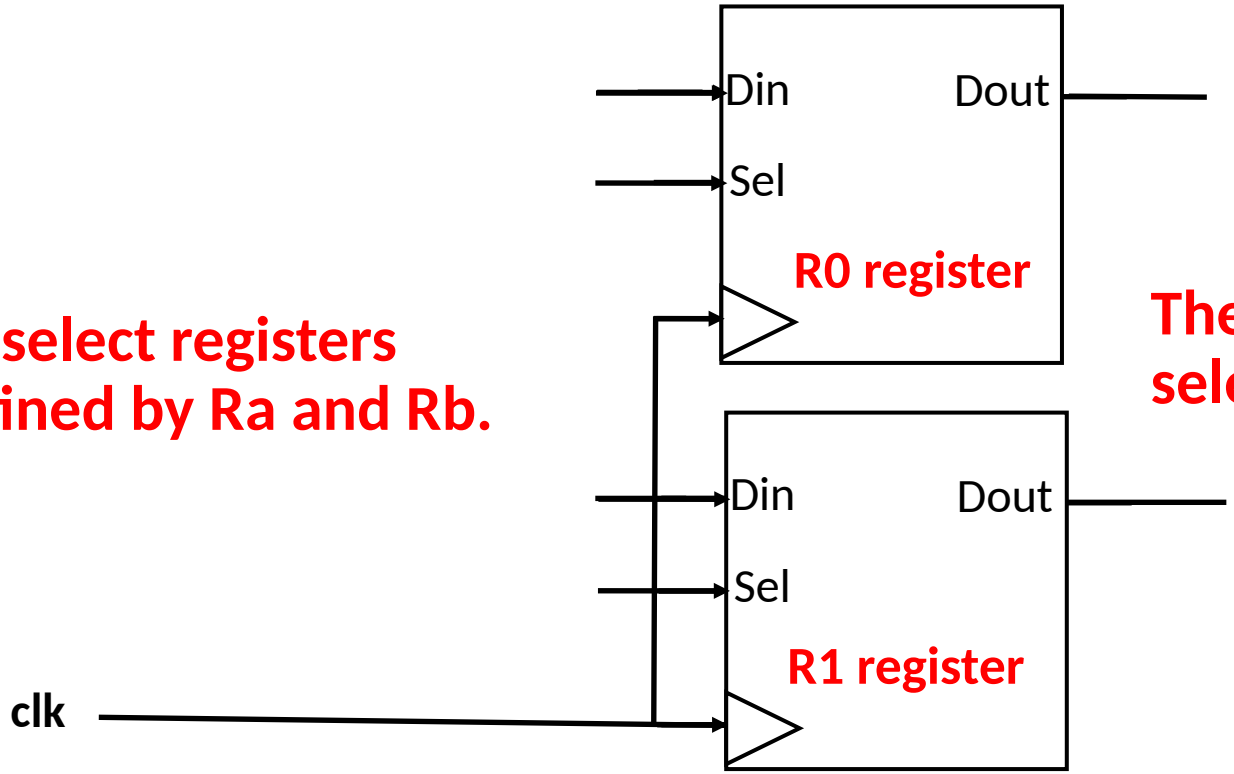
Register Set Design

1bit Register Set with 2 1bit registers

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

Ra

Rb



a

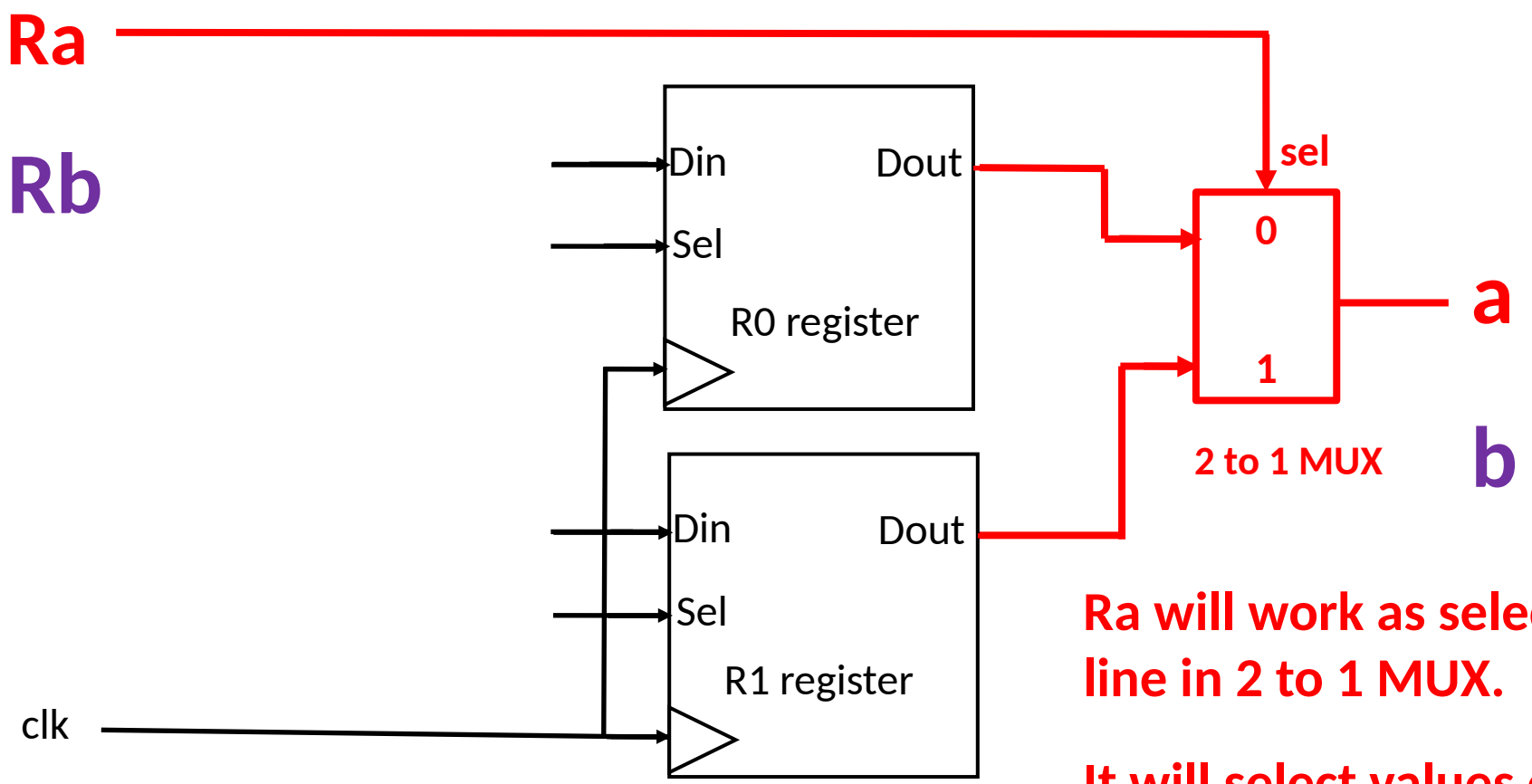
b

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

1bit Register Set with 2 1bit registers

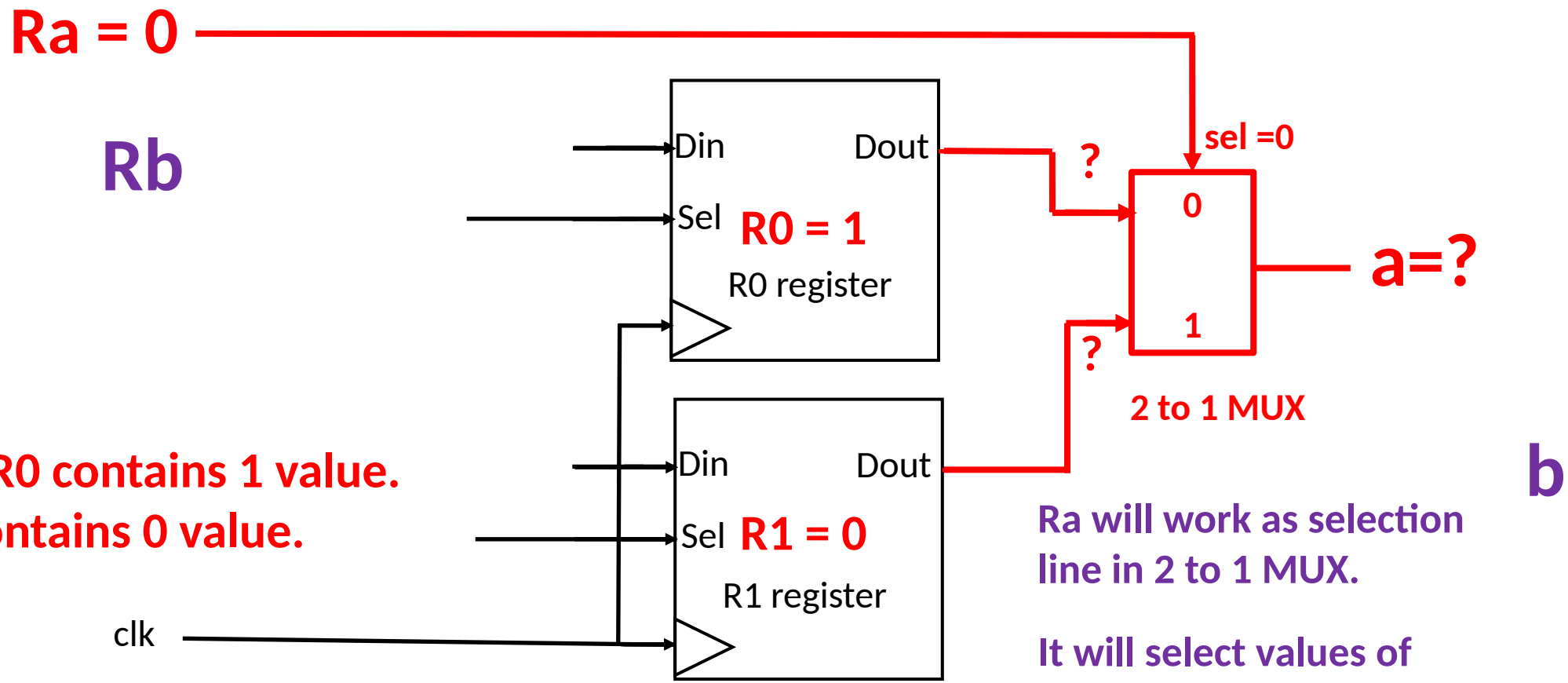


Ra will work as selection line in 2 to 1 MUX.

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

Register Set Design

1bit Register Set with 2 1bit registers



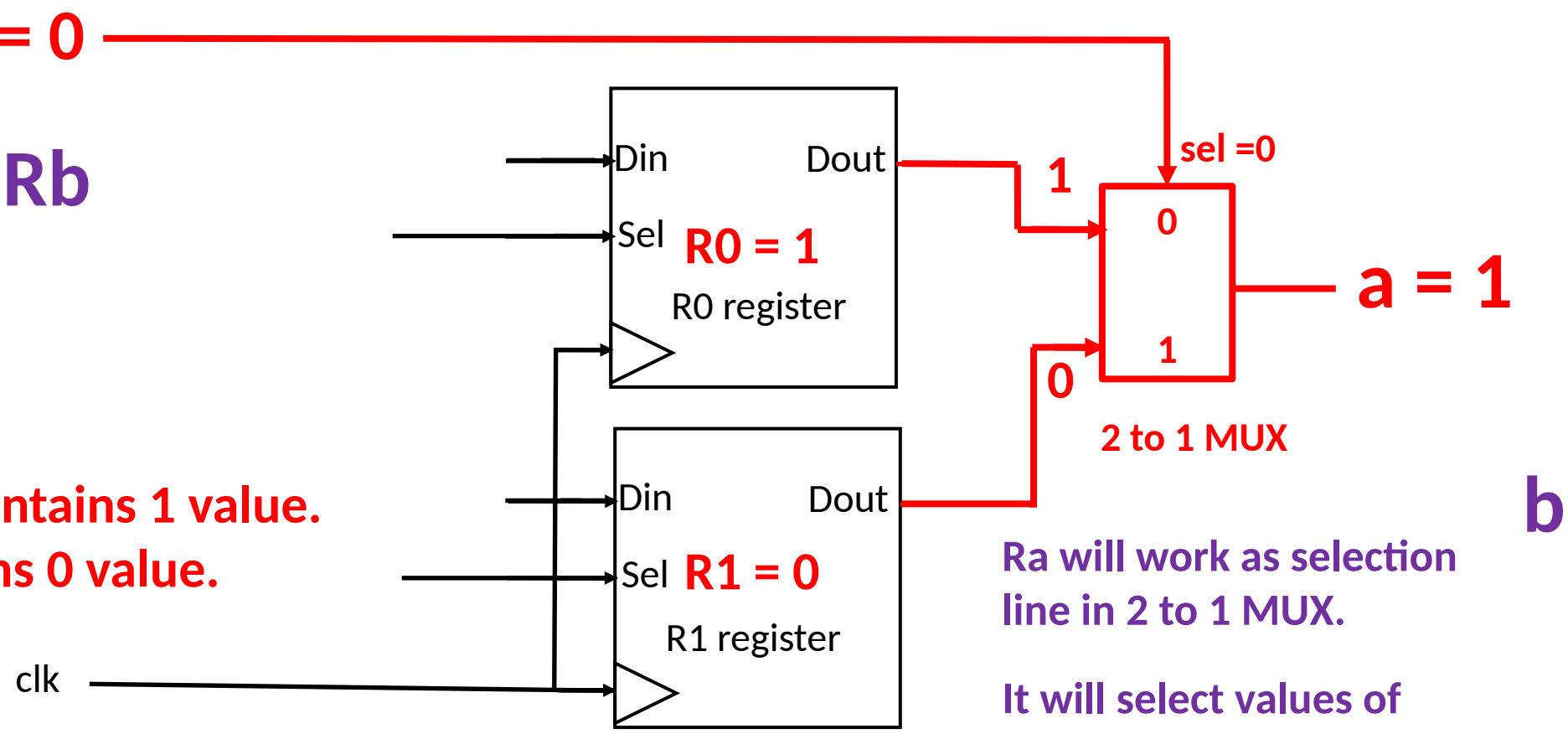
Suppose
R0 = 1
R1 = 0
It means R0 contains 1 value.
And R1 contains 0 value.

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

1bit Register Set with 2 1bit registers



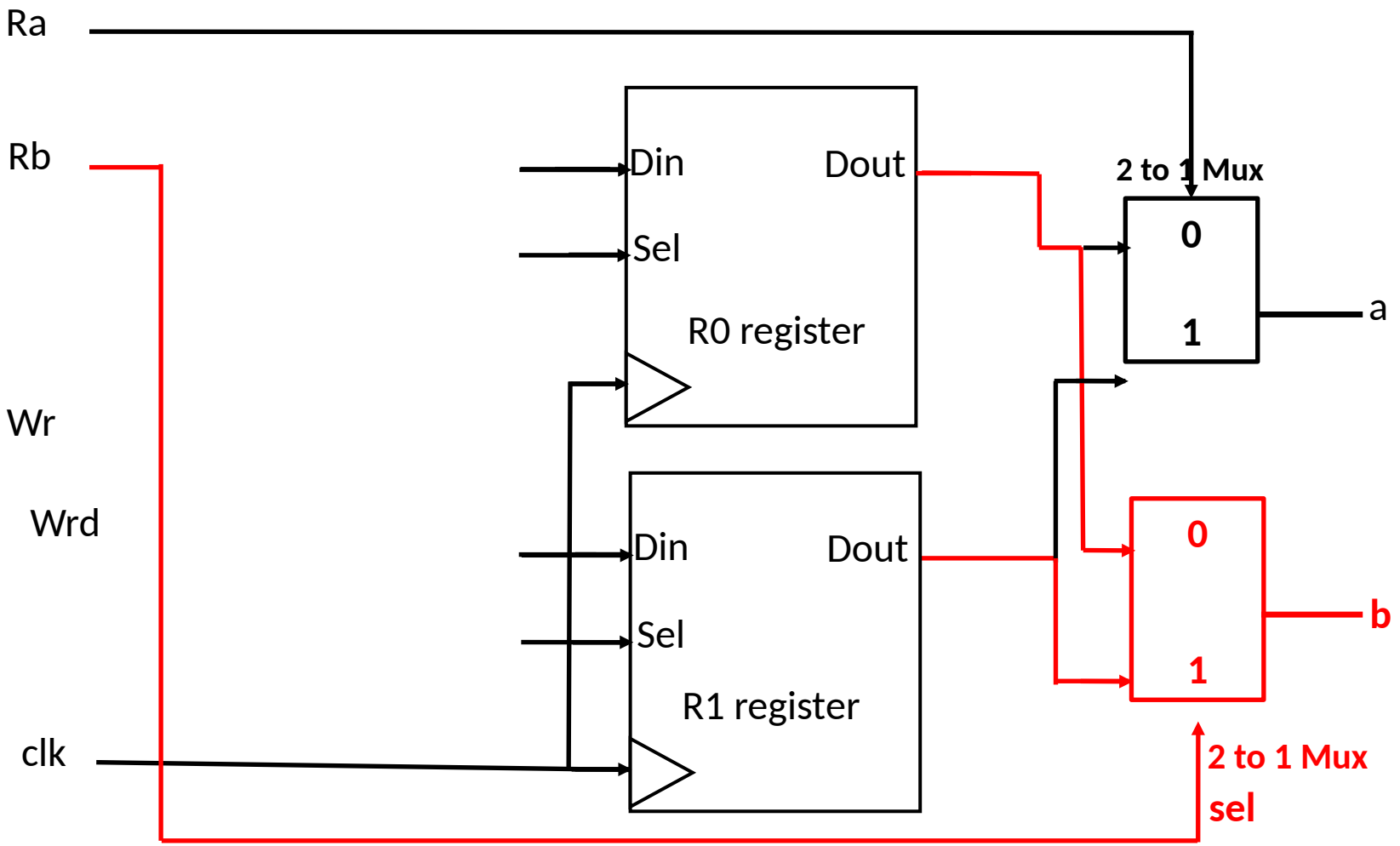
Suppose
R0 = 1
R1 = 0
It means R0 contains 1 value.
And R1 contains 0 value.

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

1bit Register Set with 2 1bit registers

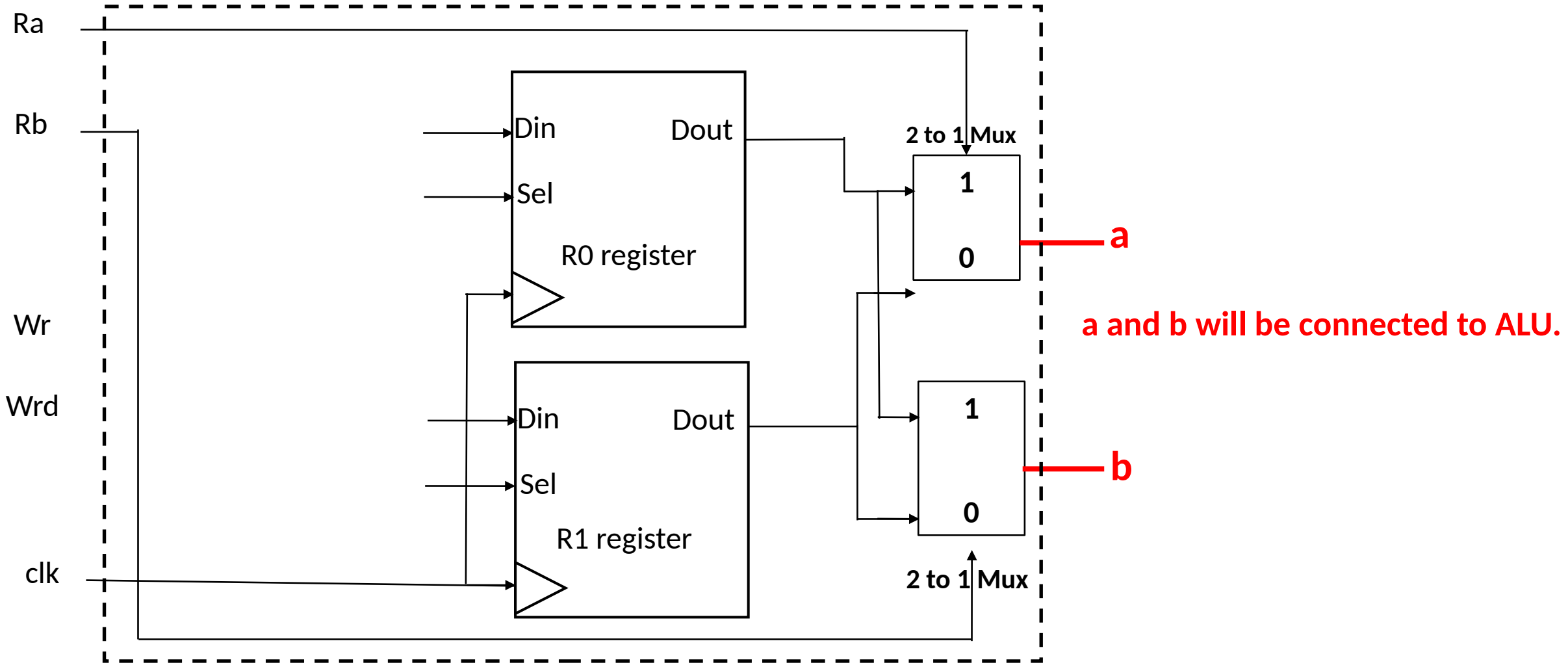


Similarly, Rb will work as selection line in 2 to 1 MUX.

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

Register Set Design

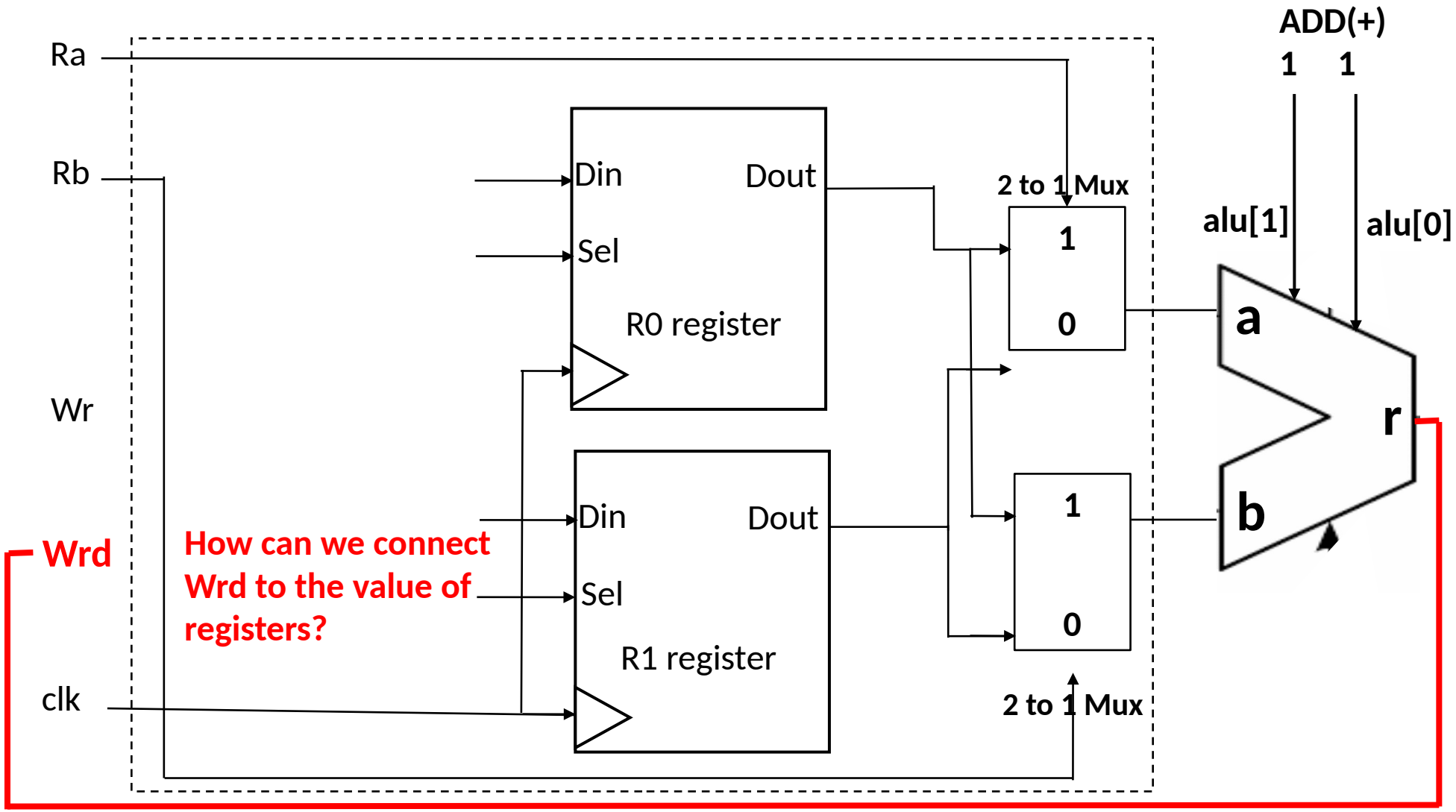
1bit Register Set with 2 1bit registers



a and b will be connected to ALU.

Register Set Design

1bit Register Set with 2 1bit registers

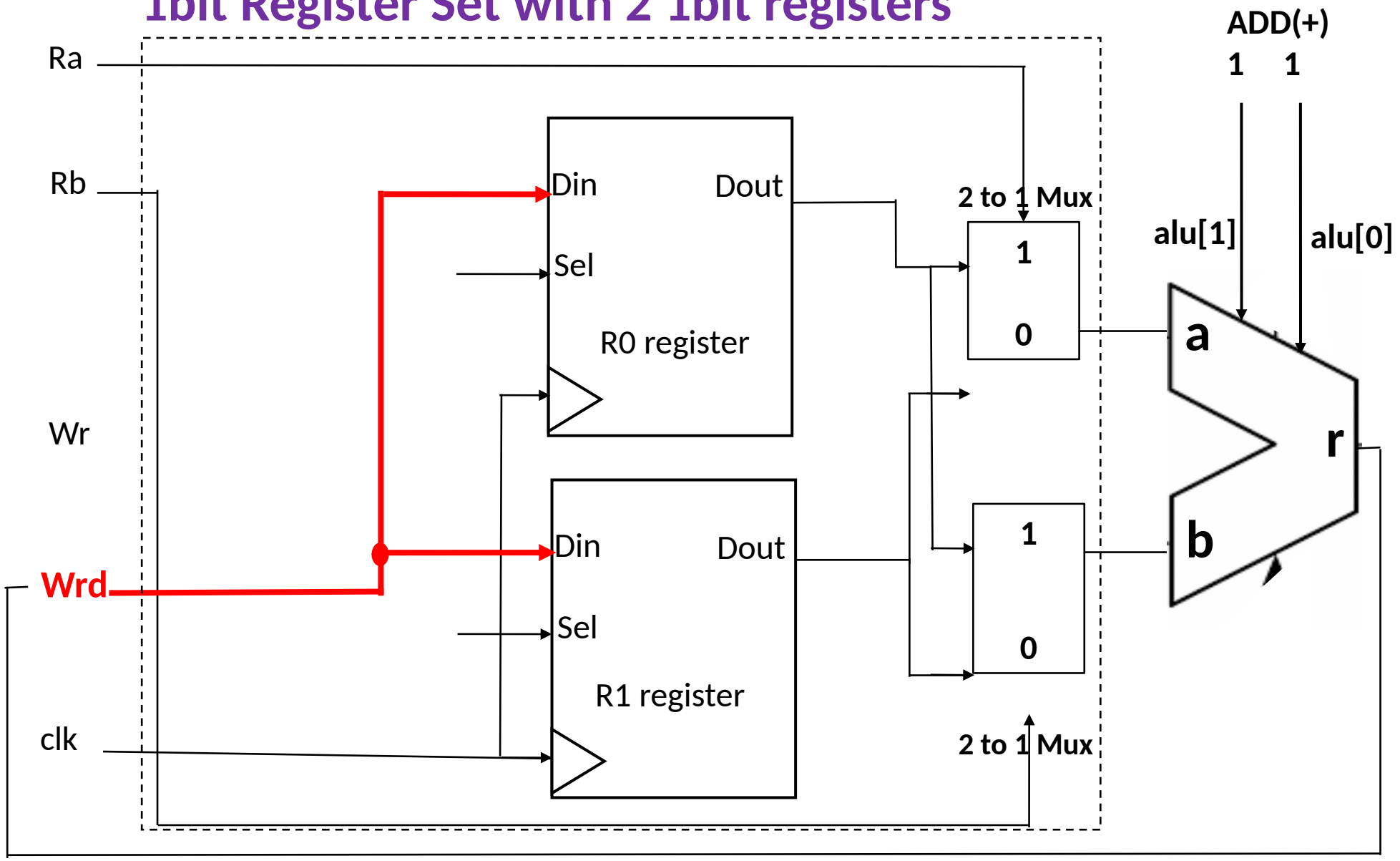


Result of ALU, r is going to be connected to Wrd.

It is data to be written in register selected by Wr.

Register Set Design

1bit Register Set with 2 1bit registers



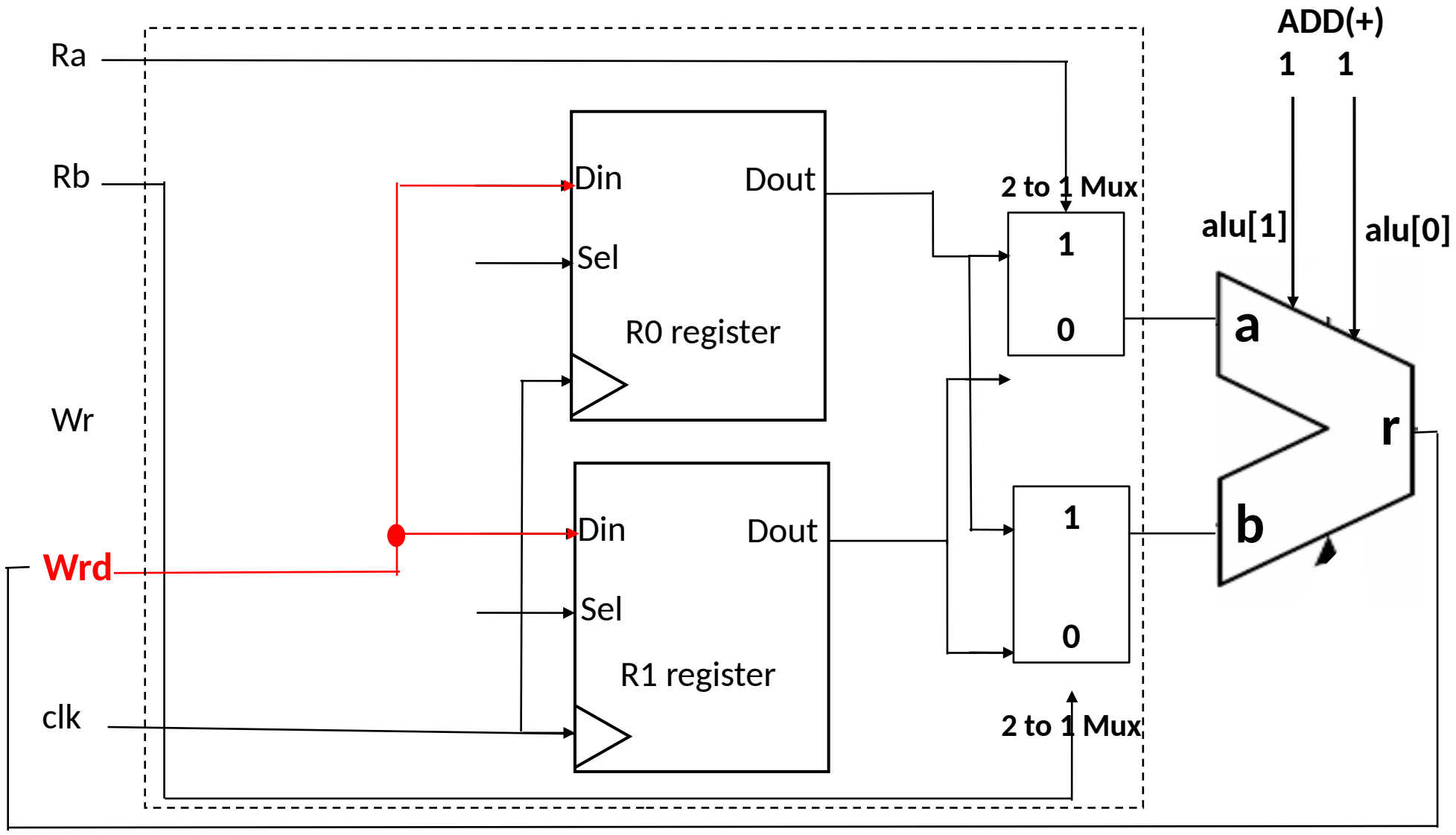
We will directly connect Wrd to Din pin of each register.

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

Register Set Design

1bit Register Set with 2 1bit registers

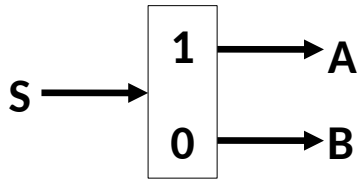
How are we going to select one of registers defined by Wr?



Register Set Design

1bit Register Set with 2 1bit registers

1 to 2 Decoder

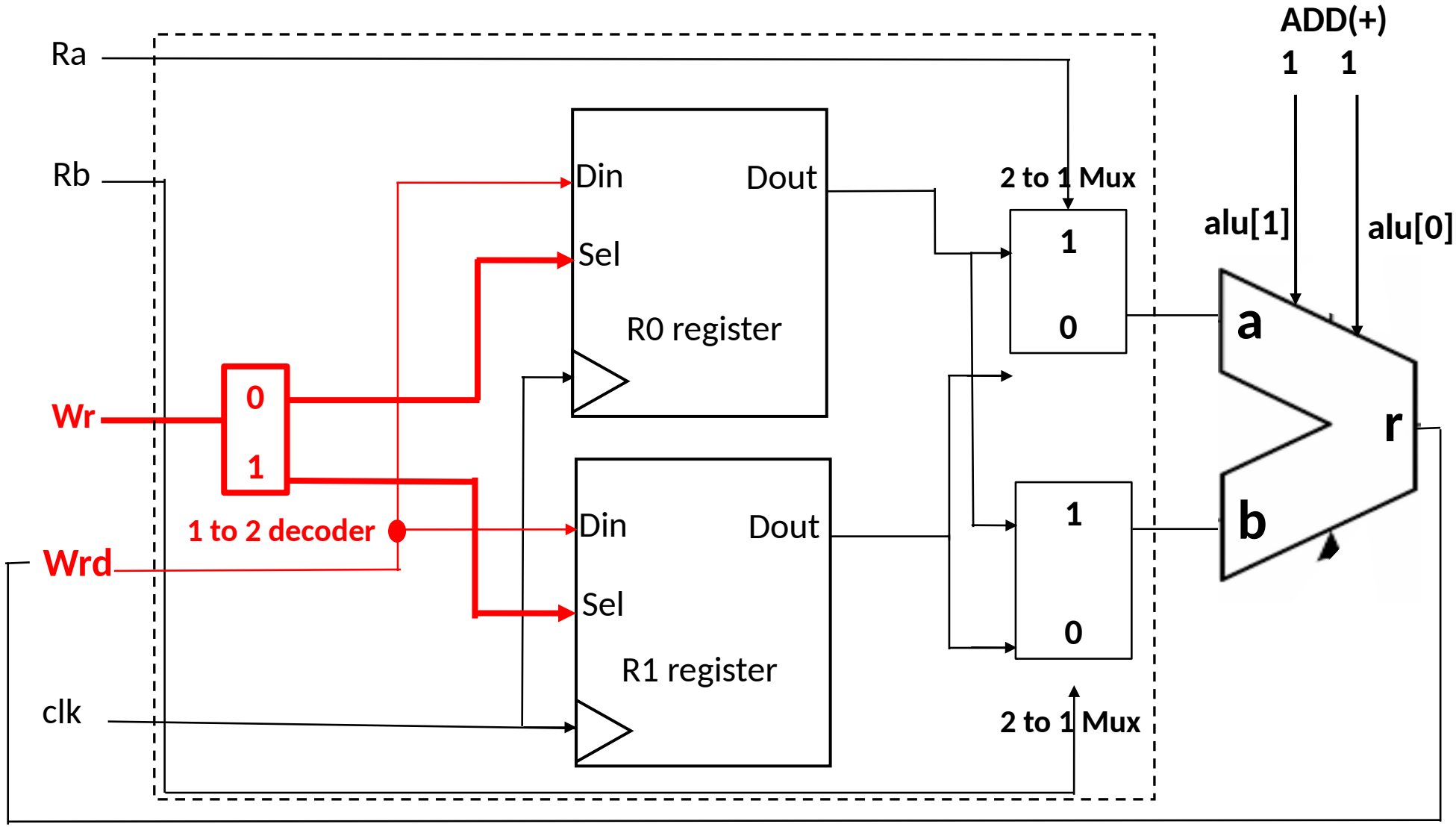


S	A	B
0	1	0
1	0	1

$A = \acute{S}$

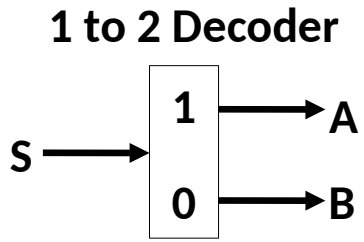
$B = S$

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

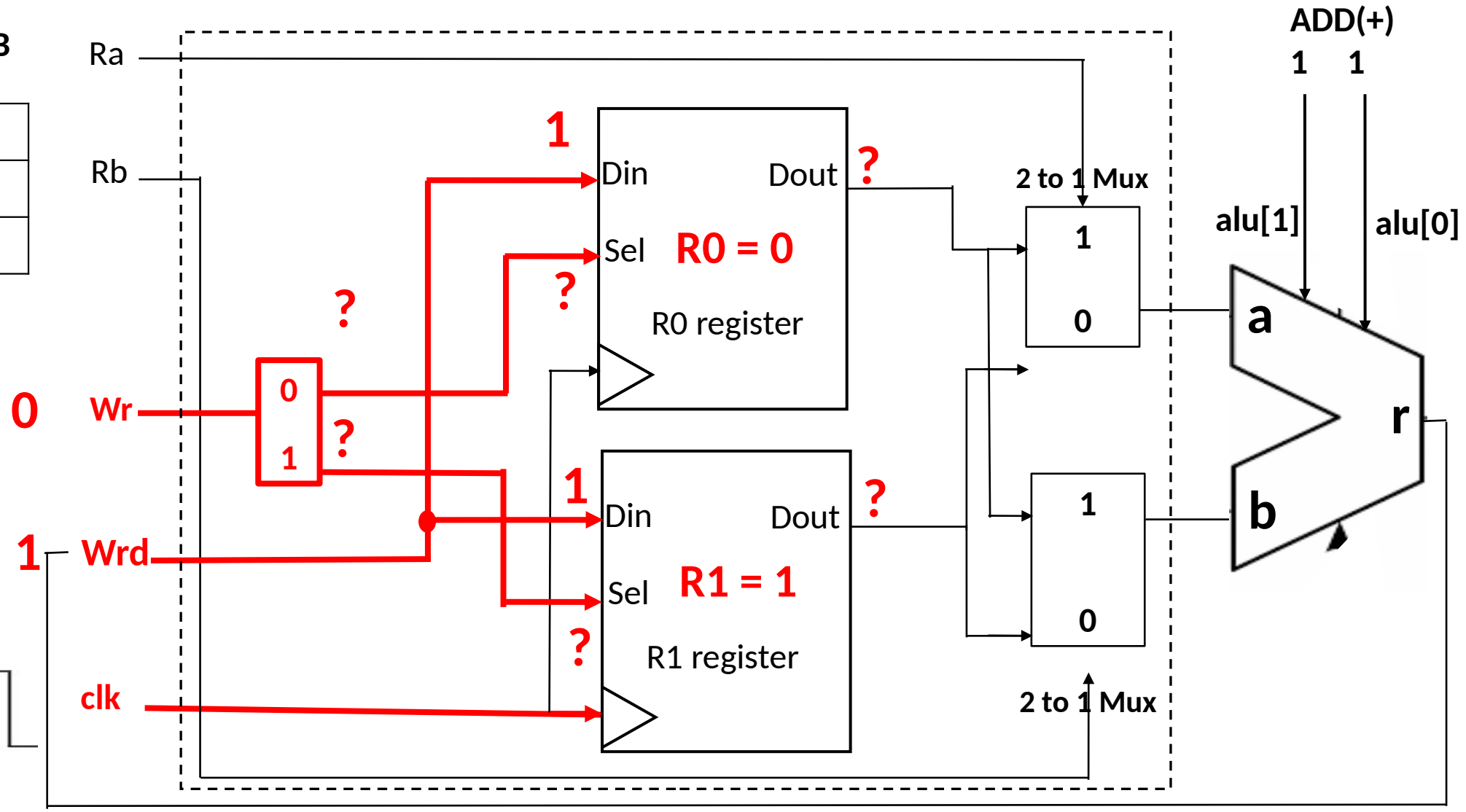


Register Set Design

1bit Register Set with 2 1bit registers



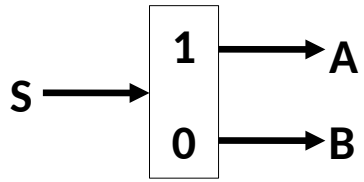
S	A	B
0	1	0
1	0	1



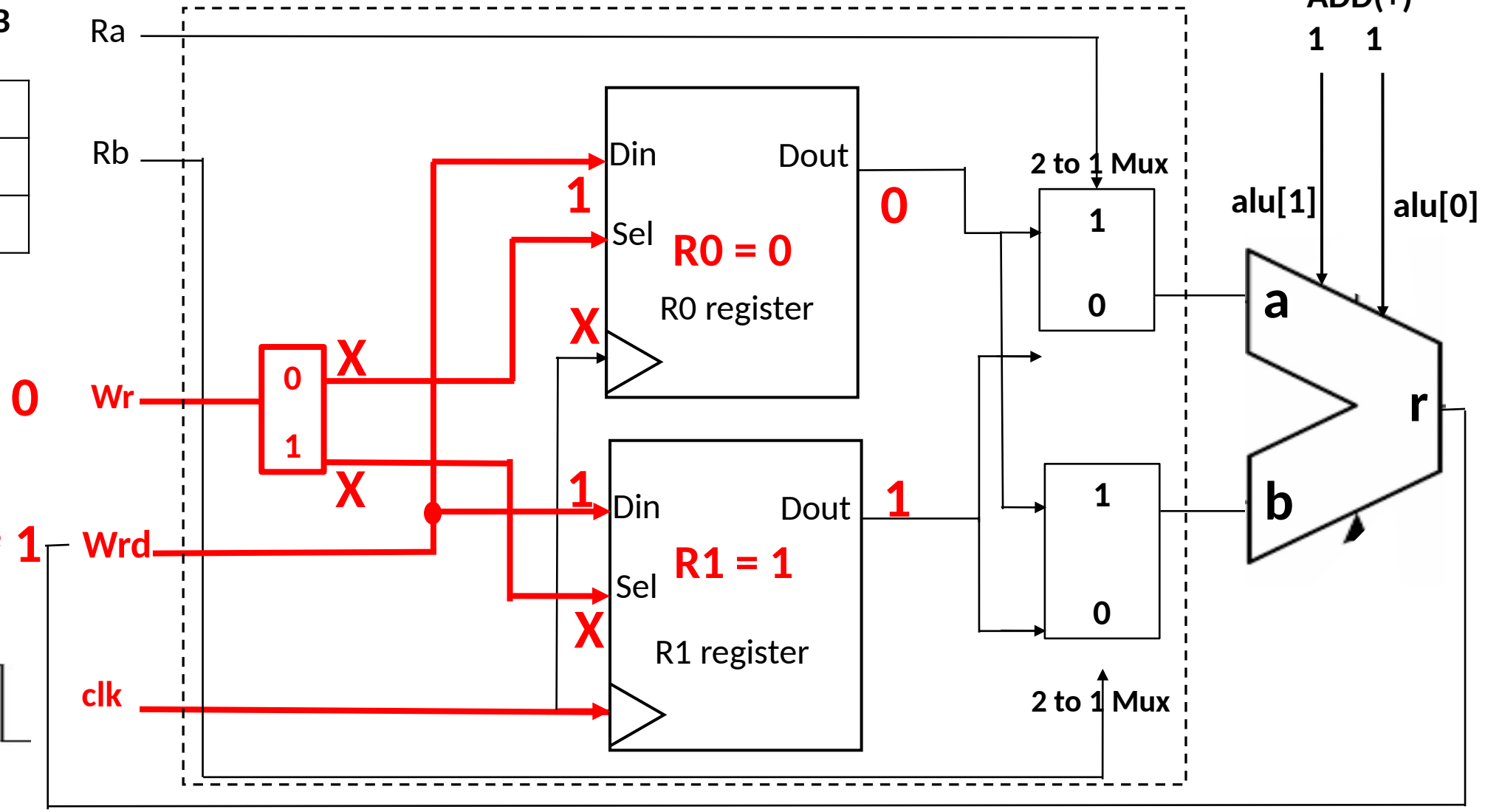
Register Set Design

1bit Register Set with 2 1bit registers

1 to 2 Decoder



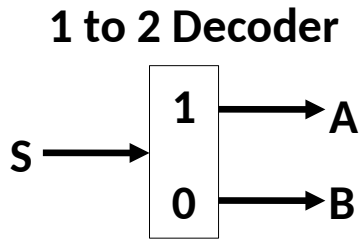
S	A	B
0	1	0
1	0	1



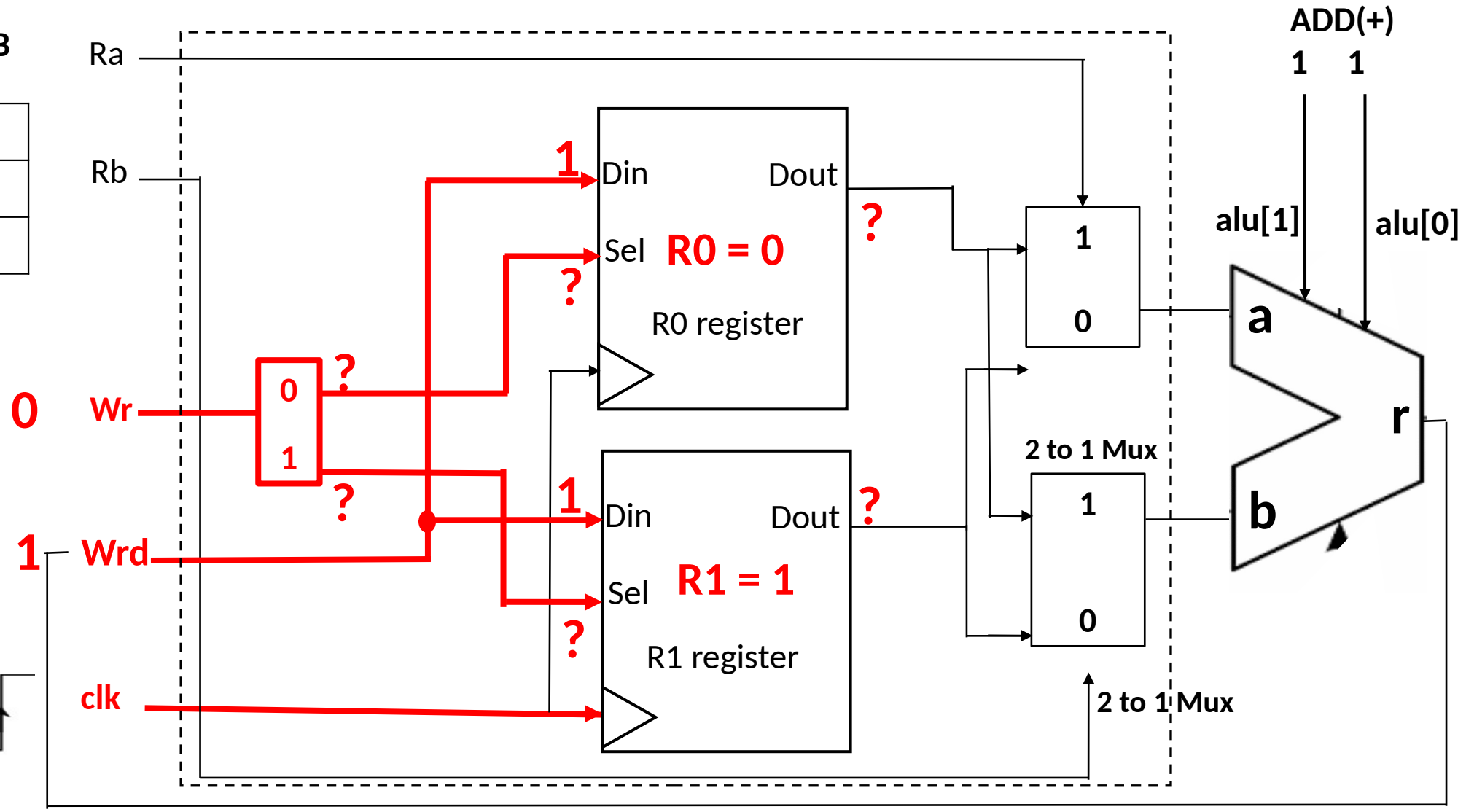
Since clock is on negative edge, value of registers will not be updated!

Register Set Design

1bit Register Set with 2 1bit registers

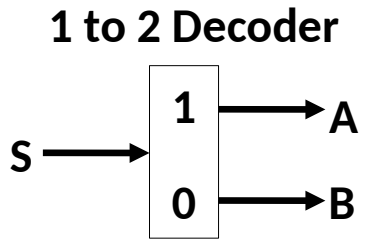


S	A	B
0	1	0
1	0	1

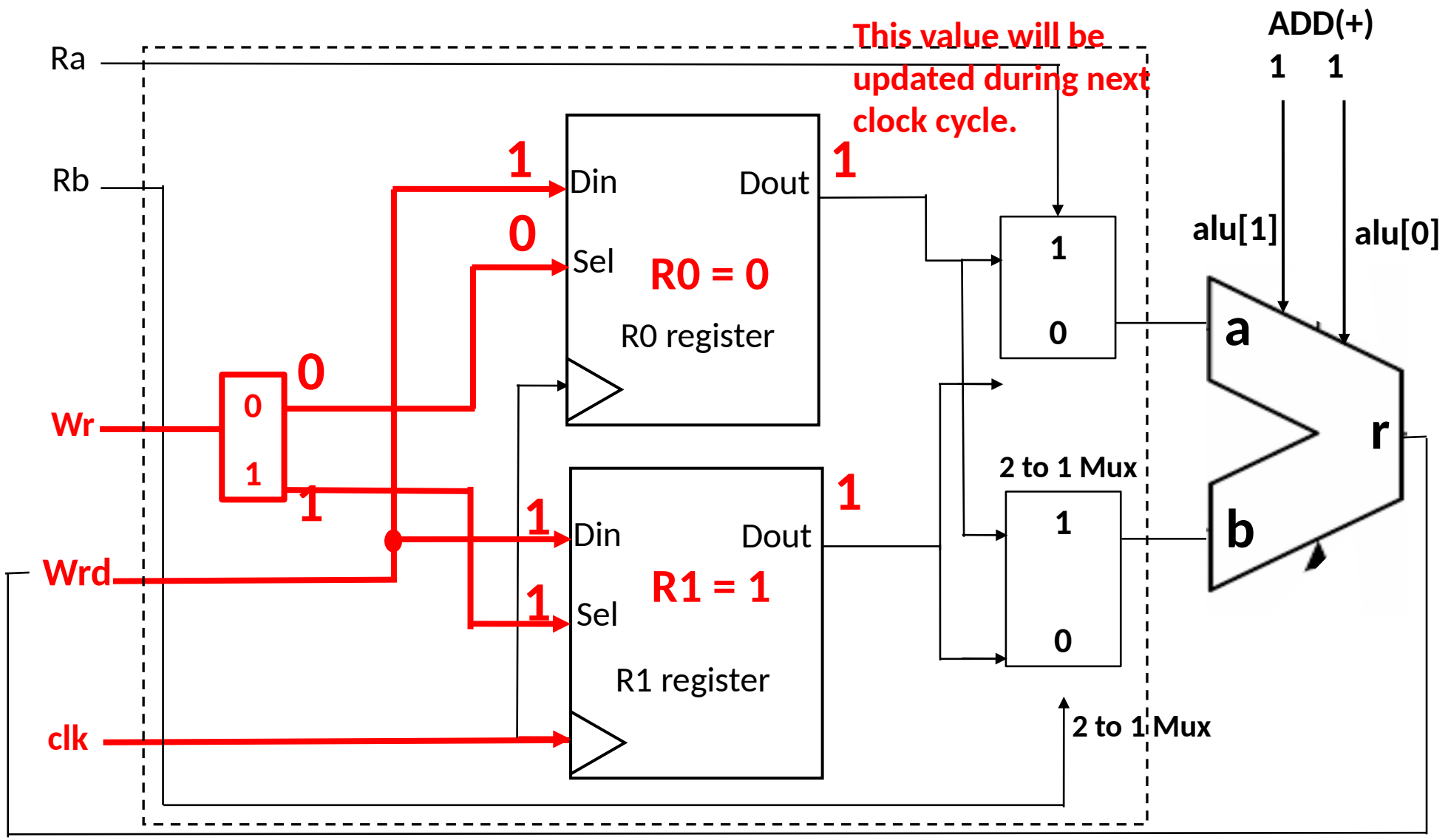


Register Set Design

1bit Register Set with 2 1bit registers



S	A	B
0	1	0
1	0	1



Register Set Design

1bit Register Set with 2 1bit registers

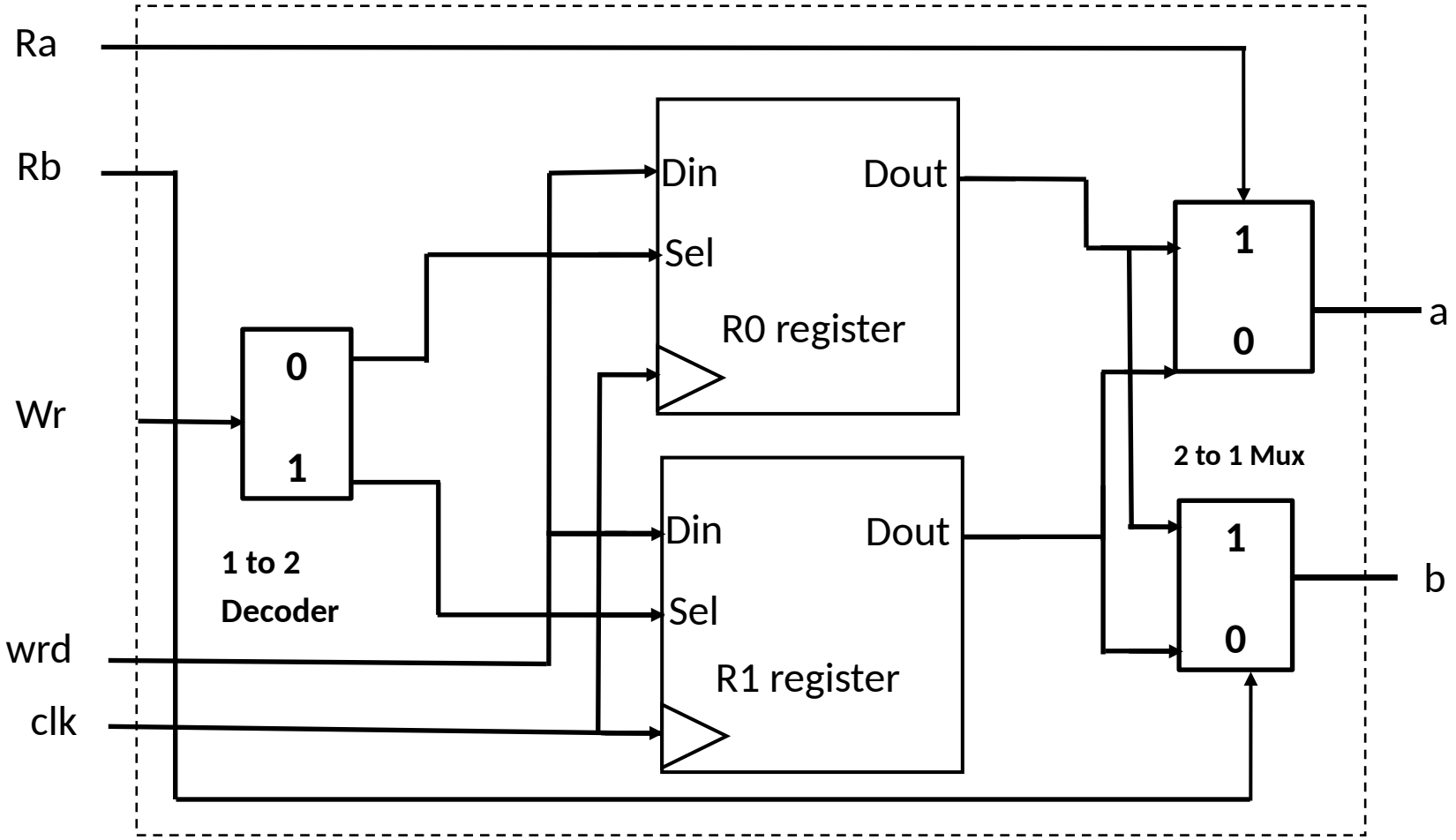


Figure: Register set with 2 1-bit registers (Final Design)

Homework:
Similarly try to design 2bit/3bit/4bit Register Set with 1/2/3/4 register/registers

Next Day:
Register Design II
&
SRAM

Thank You ㅆ