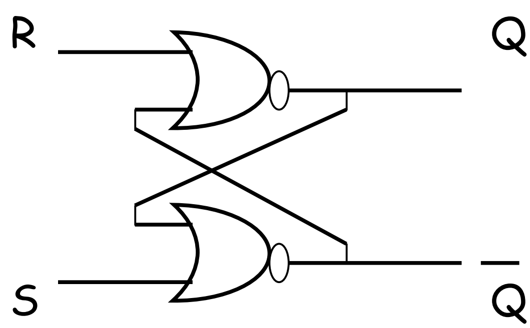
Sequential Circuits

The output depends not only on the current inputs, but also on the past values of the inputs. This is how a digital circuit remembers data. Let us see ho a single bit is stored.

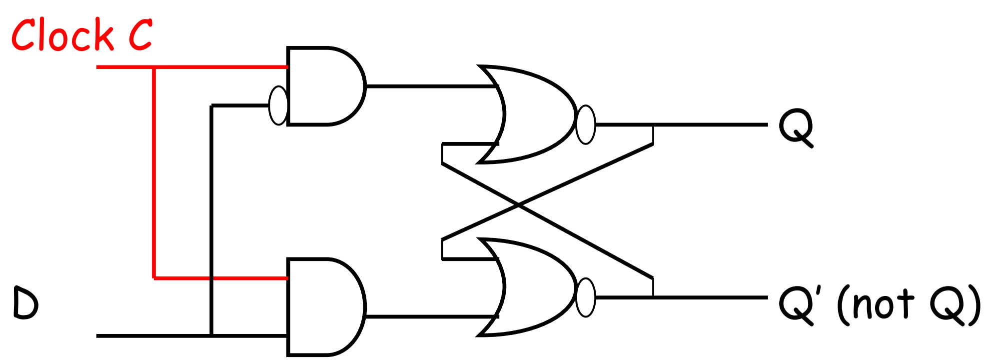


An SR Latch

R = Reset, S= Set

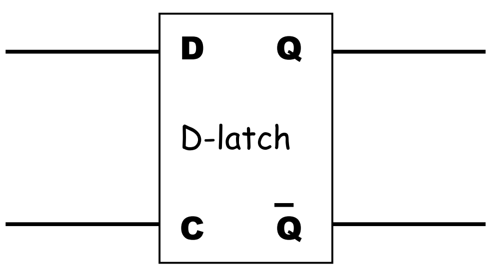
S	R	Q	$\overline{Q}$	Comment
0	0	0/1	1/0	Old state continues
1	0	1	0	Set state
0	1	0	1	Reset state
1	1	0	0	Illegal inputs

A clocked D-latch



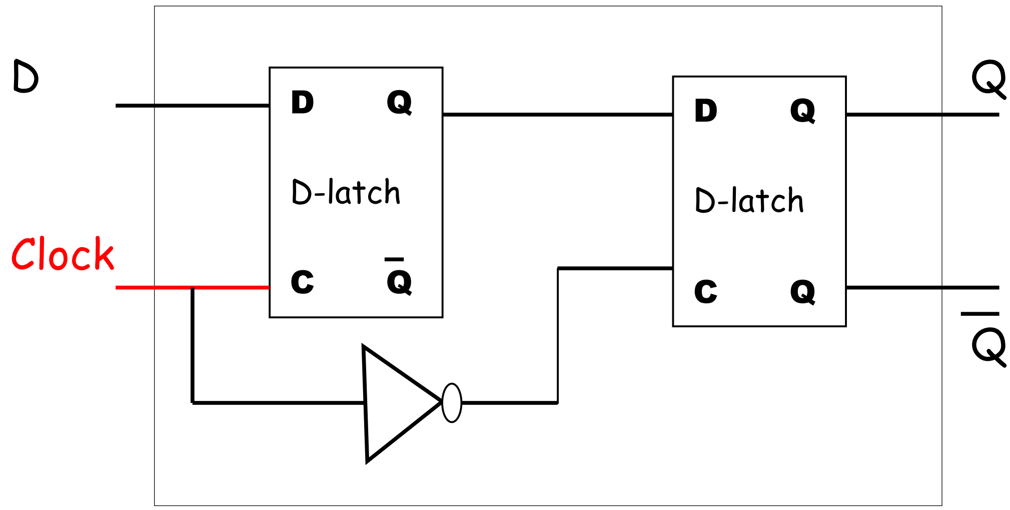
Clock is the enabler. If C=0, Q remains unchanged.

When C=1, then Q acquires the value of D. We will use it as a building block of sequential circuits.

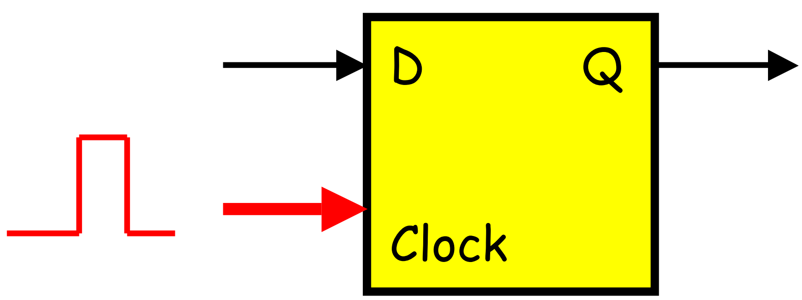


There are some shortcomings of this simple circuit. An edge-triggered circuit (or a master-slave circuit) solves this problem

Master-Slave D flip-flop



Internal details shown above

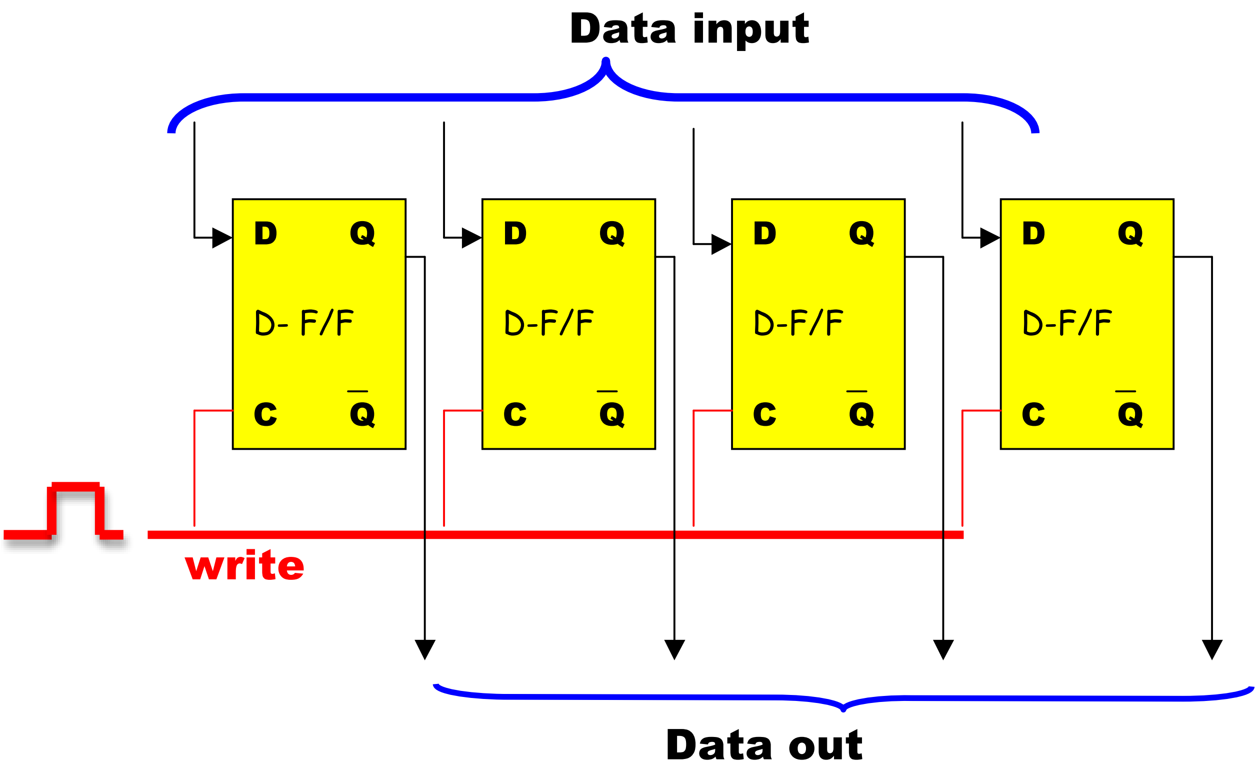


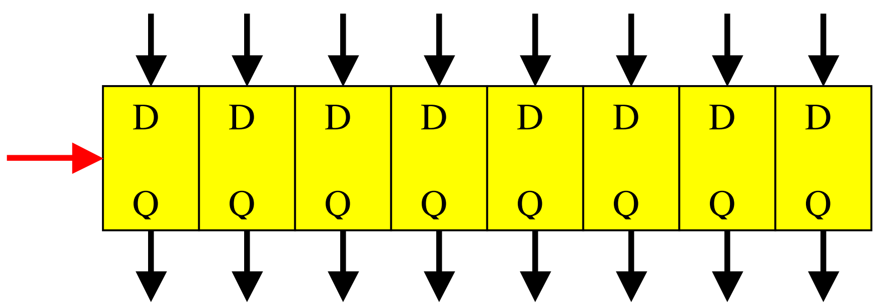
Clock pulse Abstract view

The output Q acquires the value of the input D, only when one complete clock pulse is applied to the clock input.

Register

A 8-bit register is an array of 8 D-flip-flops.

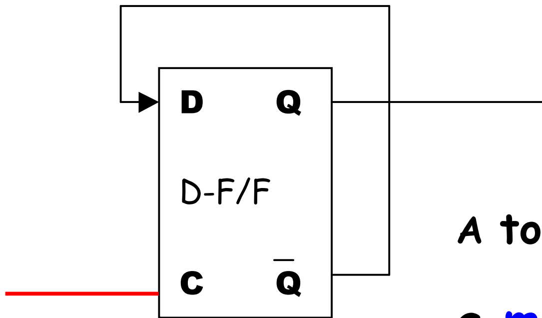




Abstract view of a register

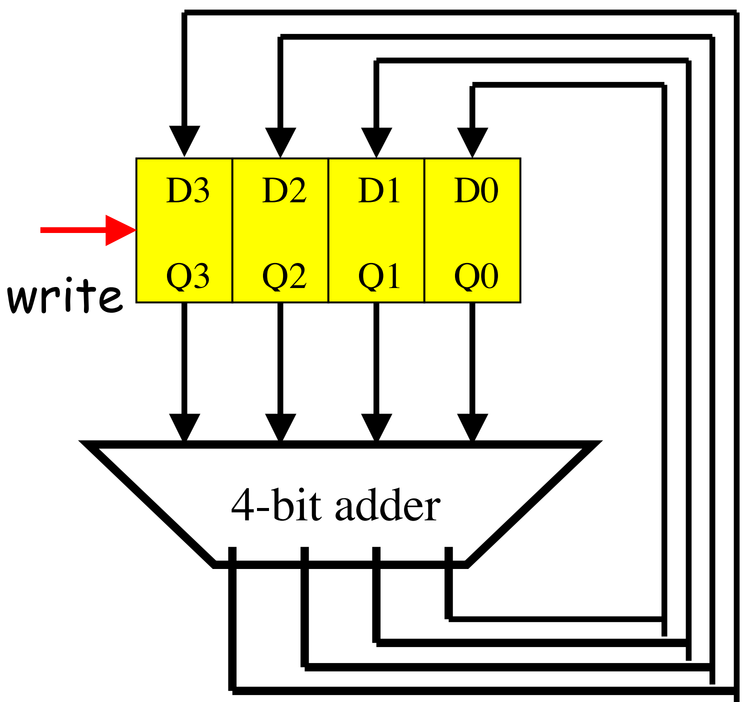
Binary counter

Counts 0, 1, 2, 3, …



A toggle flip-flop (T) is

a **modulo-2 counter**



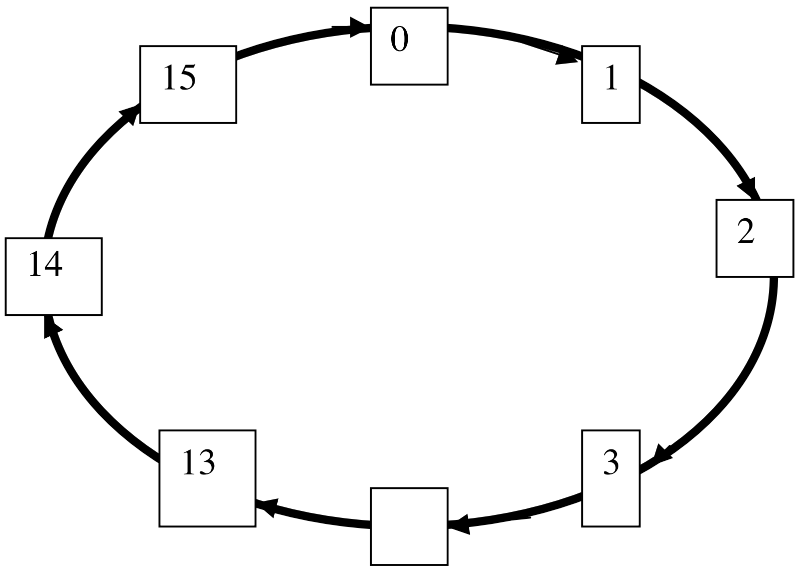
A 4-bit counter

(mod-16 counter)

Observe how Q3 Q2 Q1 Q0 change when pulses are applied to the clock input

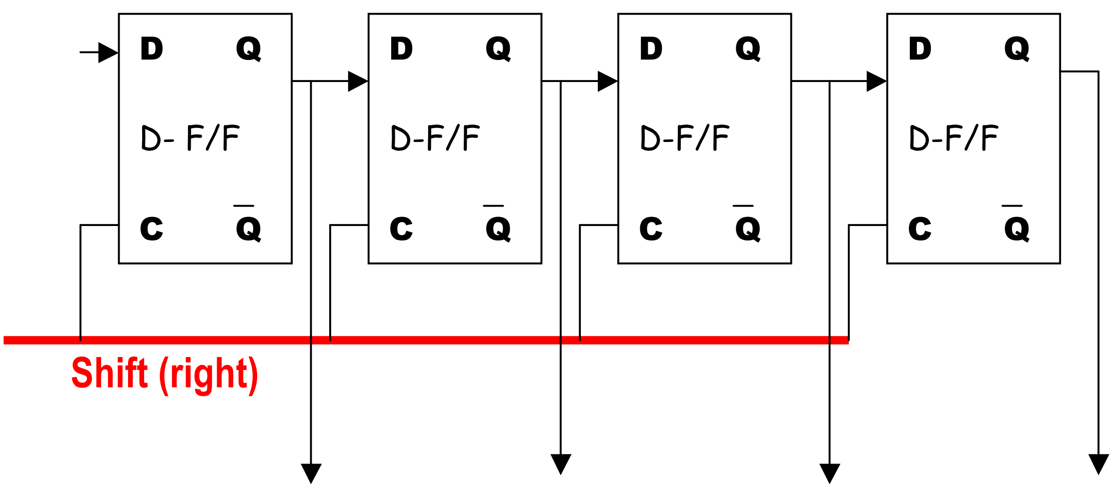
State diagram of a 4-bit counter

Here state = Q3Q2Q1Q0



Recall that the program counter is a 32-bit counter

A shift register



With each pulse

Hardware Multiplication

Multiplicand				1	0	0	1
Multiplier				1	0	1	0
				0	0	0	0
			1	0	0	1	0
		0	0	0	0	0	0
	1	0	0	1	0	0	0
Product	1	0	1	1	0	1	0

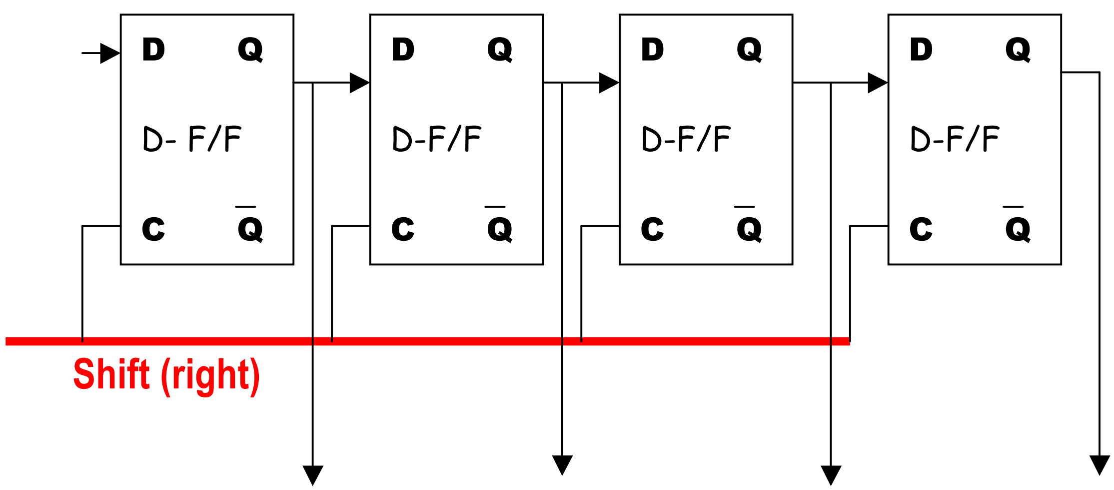
The basic operations are ADD and SHIFT. Now let us see how it is implemented by hardware.

By now, you know all the **building blocks**.

The Building Blocks

A shift register

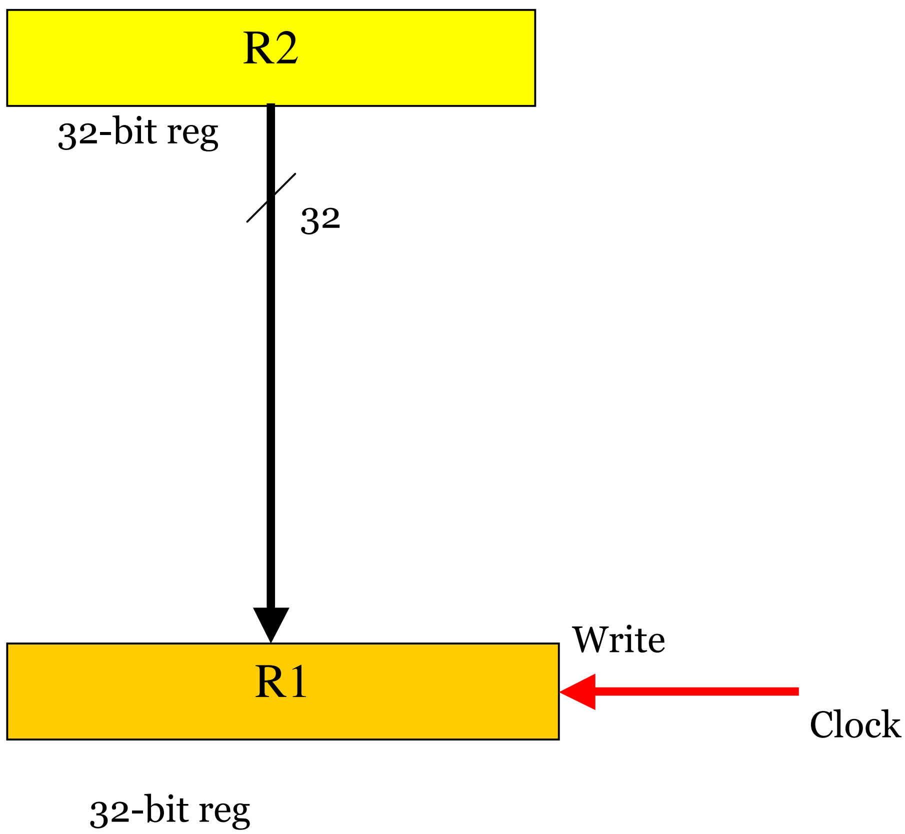
Review how a D flip-flop works



With each clock pulse on the shift line, data moves one place to the right.

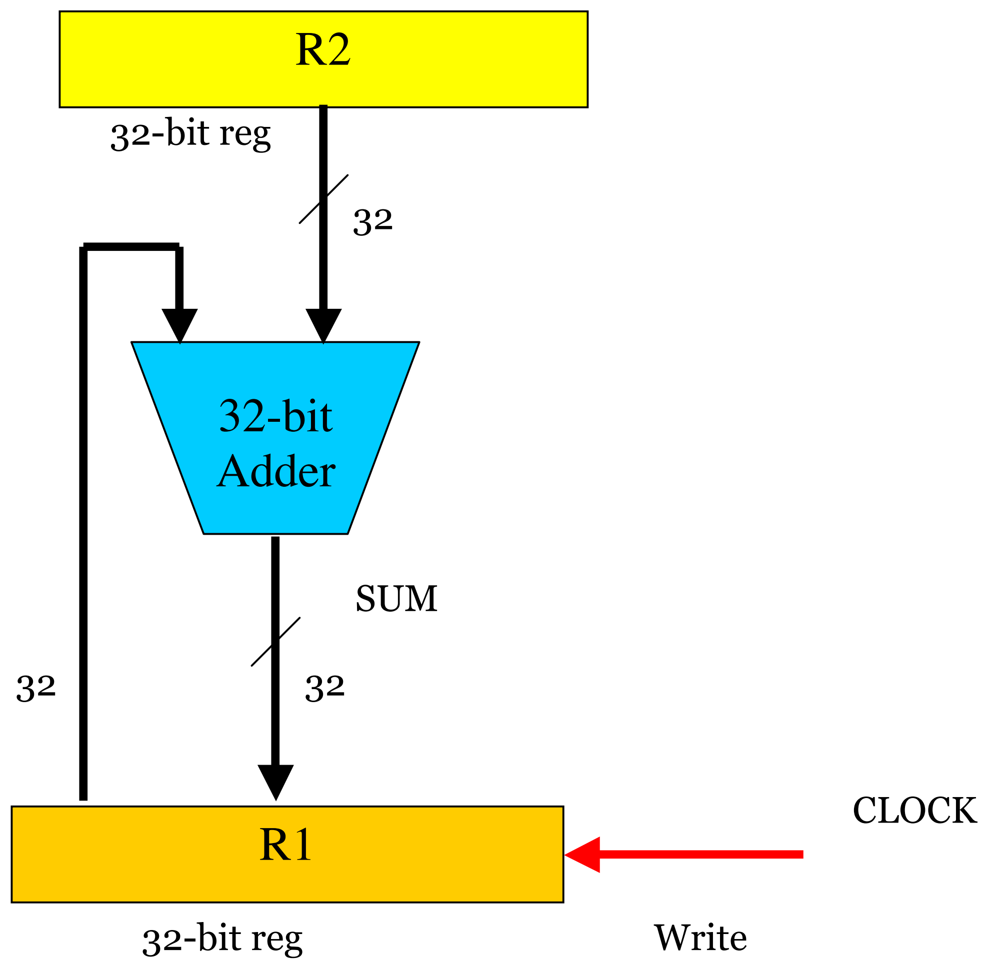
Executing r1:= r2

How to implement a simple register transfer r1:= r2?



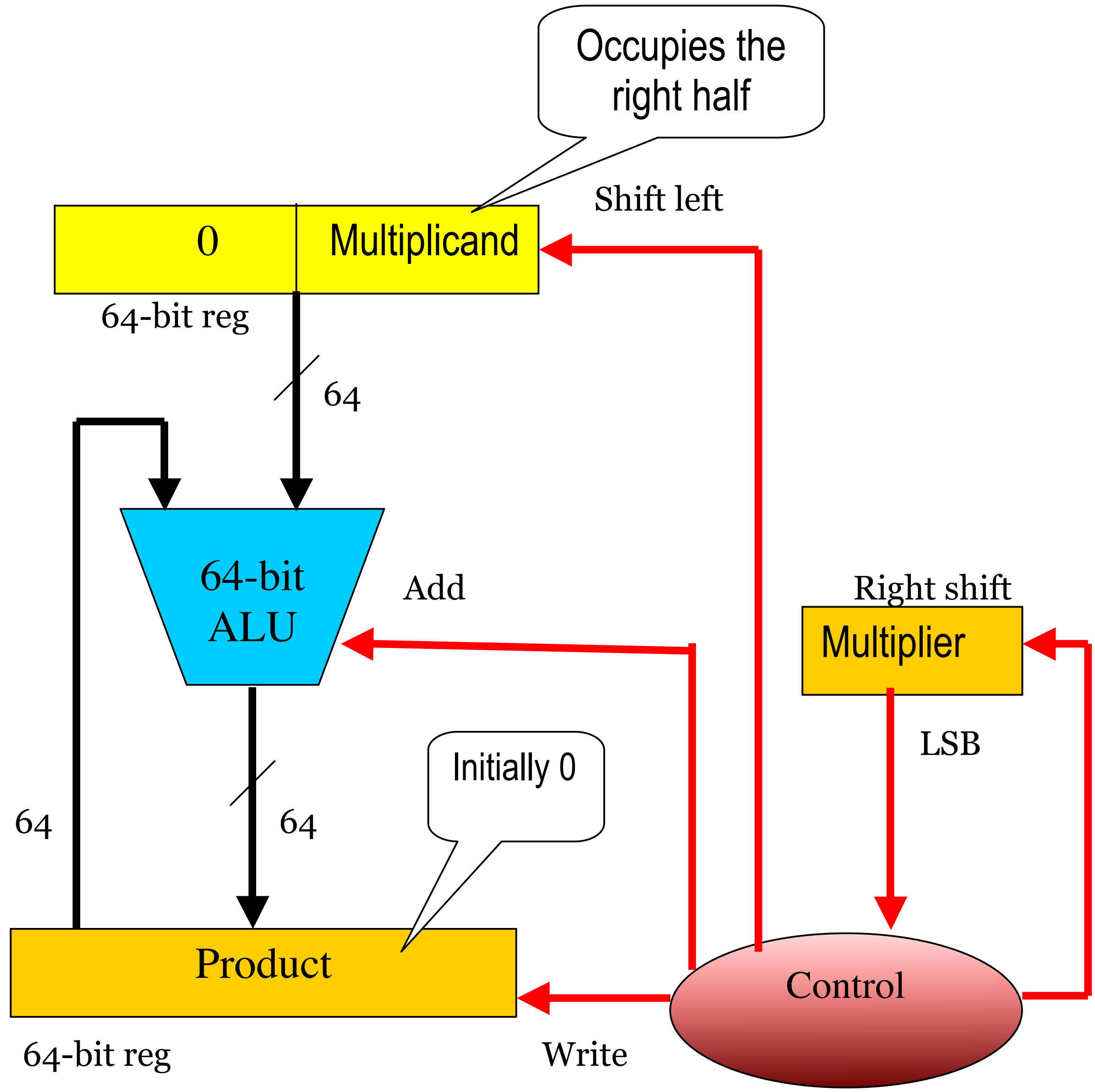
It requires only one clock pulse to complete the operation.

Executing r1:= r1 + r2



It requires only one clock pulse to complete the operation.

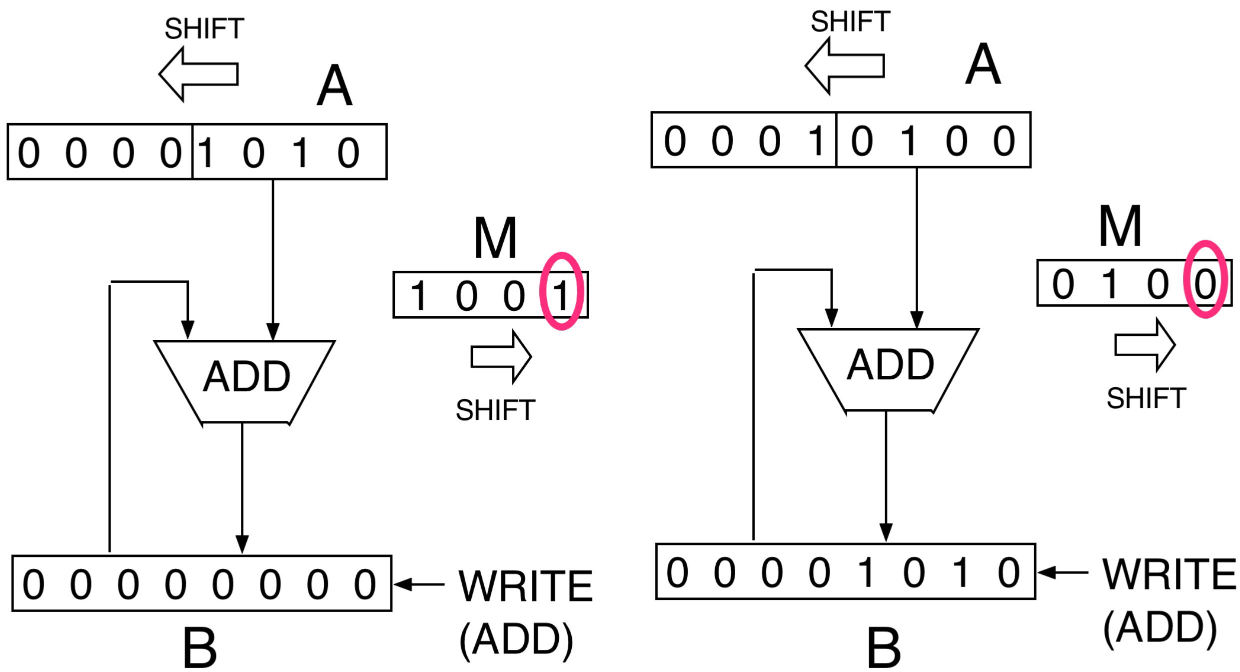
A Hardware Multiplier



If LSB of Multiplier = 1 then *add* else *skip*;

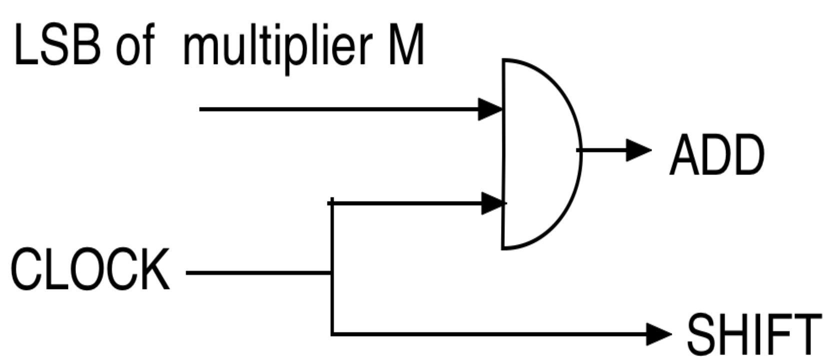
Shift left multiplicand & shift right multiplier

How to implement the control unit?



if LSB (M) = 1 then ADD, SHIFT LEFT A, SHIFT RIGHT M

else SHIFT LEFT A, SHIFT RIGHT M



Division

The restoring division algorithm follows the simple idea from the elementary school days. It involves subtraction and shift. Here is an implementation by hardware