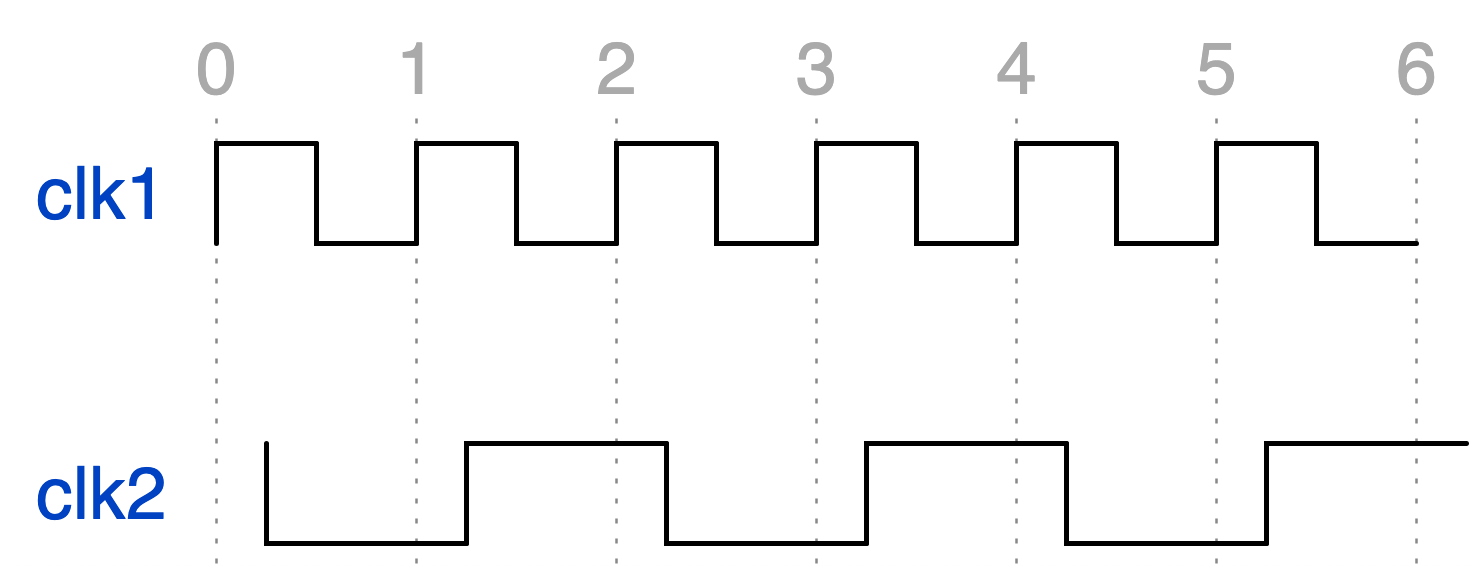
# Why?

Our goal is to build large systems from digital logic. We've already seen combinational logic building blocks. This activity will introduce you to *sequential logic*, whose components hold onto information over time.

# Model 1: Clocks

Consider the following diagrams of two different *clock signals*. The x-axis is time in nanoseconds (ns) and the y-axis is voltage.



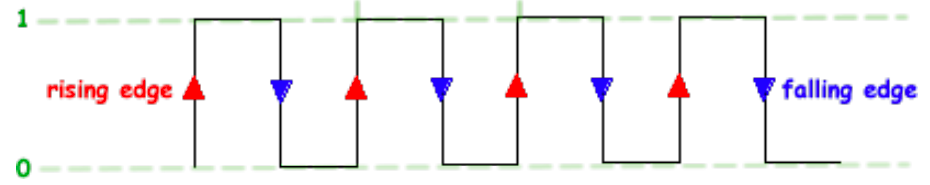
1

0

1

0

Legend:



1. For each clock signal, at what times does the signal transition from 0 to 1? Include units.

Clk1: Every second. Clk2: Every other second with a ¼ second beginning delay.

1. For each clock signal, at what times does the signal transition from 1 to 0? Include units.
2. Given the legend, define ***rising edge***.

When the signal changes from 0 to 1.

1. Given the legend, define ***falling edge***.

When the signal changes from 1 to 0.

1. For each clock signal, at what times are all the *rising edges*? Include units.

# Read This!

As you’ll see in subsequent Models, the ***rising edge*** is an important part of the clock signal for sequential logic components.

1. The signals continue the same way into the future. How much time does it take for each signal to repeat?
2. In 5ns, how many times do the signals each repeat?
3. How many times per ***second*** does each signal repeat?

Express your answers in Hertz (1 Hz = 1/second).

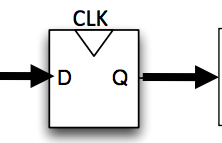
# Read This!

The type of diagram show in the Model is called a **waveform diagram**, or simply: waveform. The particular signal shown in the waveform above is called a **clock signal**. Just as the clock on the wall ticks at a rate of once per second, the clock signal "ticks" at some rate, or **frequency.** The time between two consecutive ticks is the **period**.

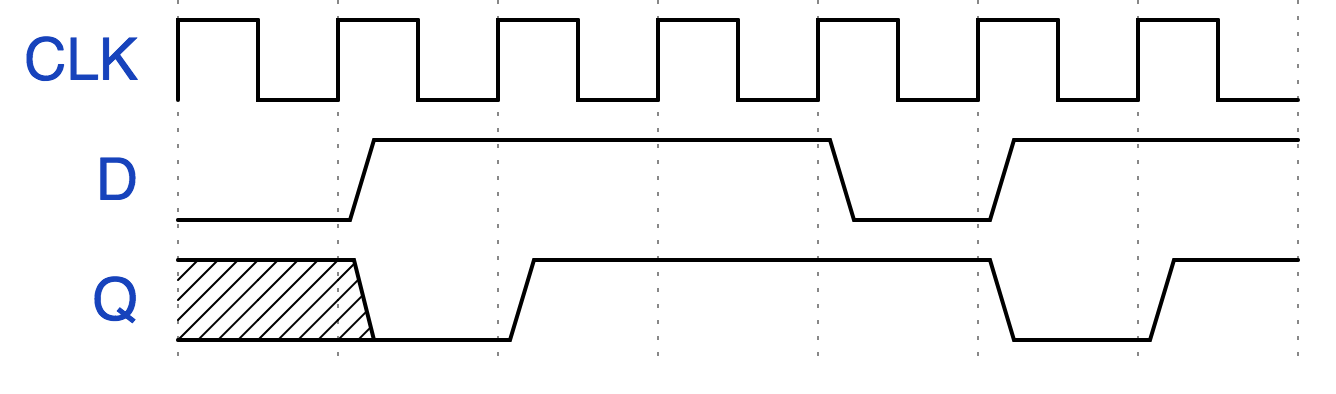
1. What is the *period* of each clock signal in the diagram?
2. What is the *frequency* of each clock signal in the diagram?
3. How are *period* and *frequency* related mathematically?

# Model 2: Flip flop

Consider the following new type of component. To see how its 1-bit output Q is related to its 1-bit input D, refer to the example in the waveform diagram below.



*Some other circuit (not shown) is changing D at various points in time.*



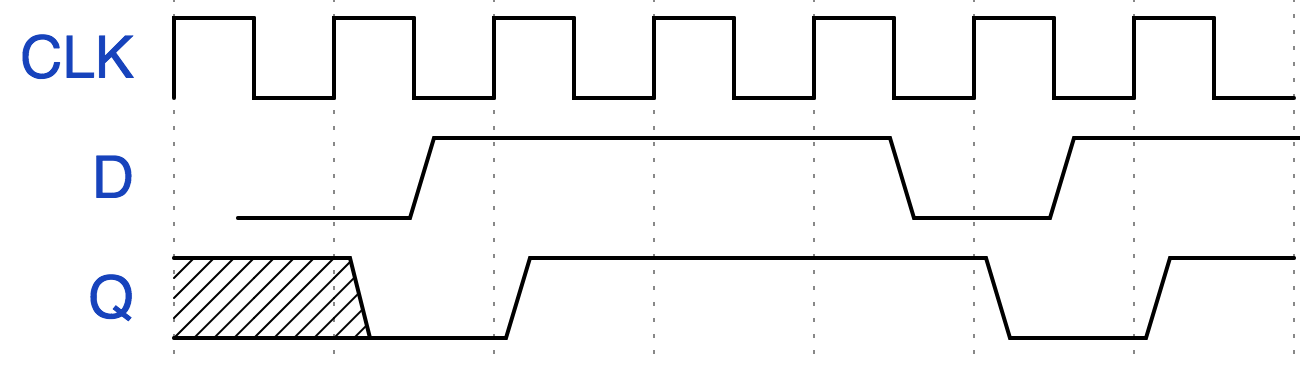
Legend:

* low signal: the value is 0
* high signal: the value is 1
* shaded signal: the value is unknown at that particular time

1. Using appropriate vocabulary from Model 1, describe how output Q behaves with respect to inputs D and CLK.

Q changes to whatever D’s input is only when CLK has a rising edge (changes from 0 to 1).

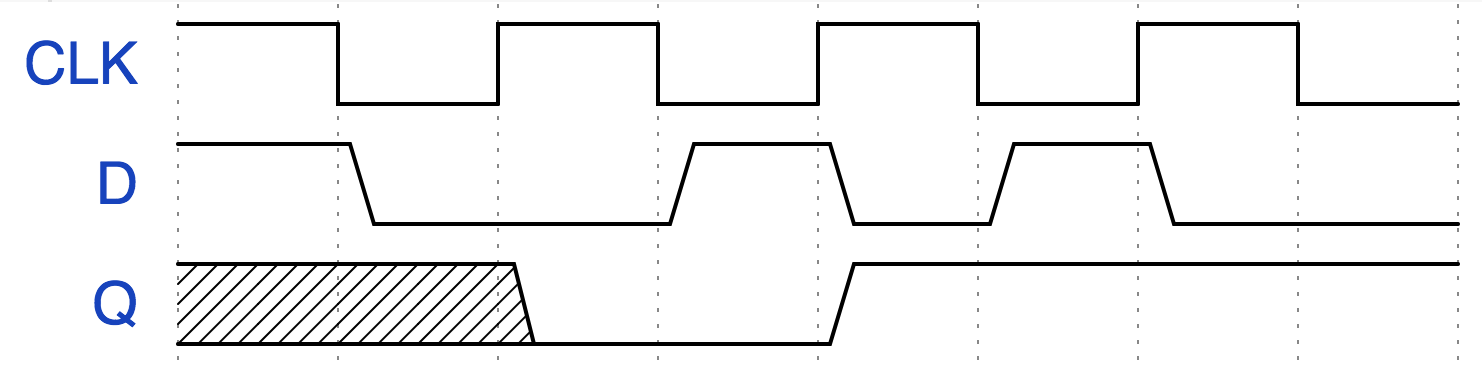
Here is a second, different example.

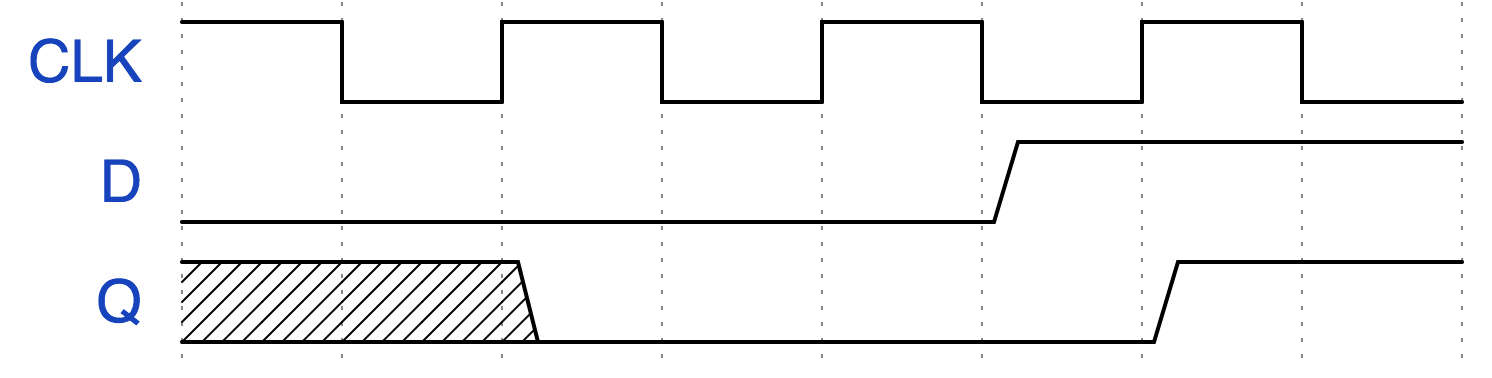


1. Given the examples so far, ***revise*** your definition of how output Q behaves with respect to inputs D and CLK. As necessary, use appropriate vocabulary from Model 1.

Still looks the same.

Here is are two more, different examples.





1. Given the examples so far, ***revise*** your definition of how output Q behaves with respect to inputs D and CLK. As necessary, use appropriate vocabulary from Model 1.

Q checks what D is at every rising edge in CLK and switches to whatever D is at that point.

# Read This!

The new component is called **flip-flop**, with a 1-bit input D, a 1-bit output Q, and an input clock signal.

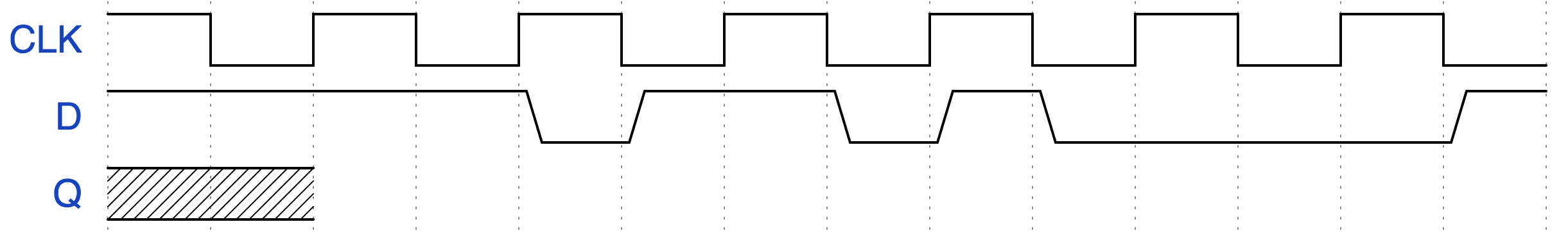
1. Describe the behavior of a **flip-flop**.

Flip-flop’s output Q copies D’s (input) signal but only checks every time CLK has a rising edge.



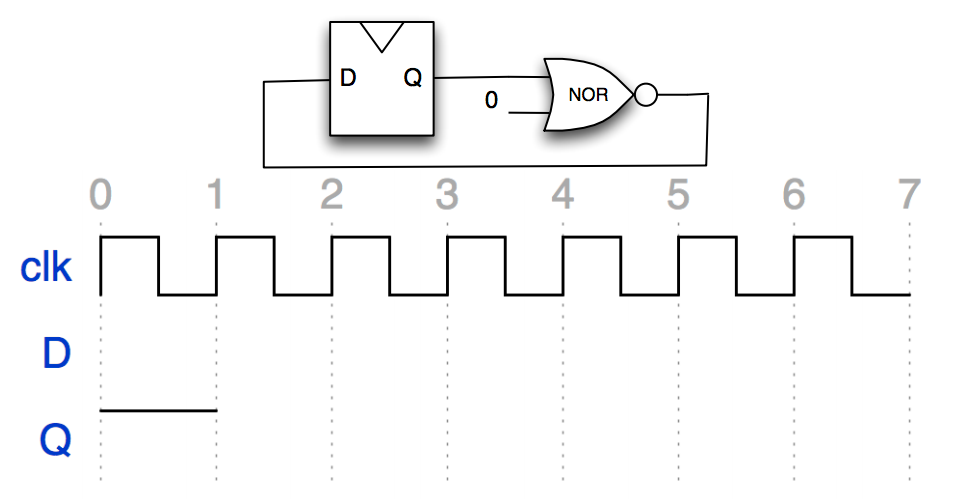
# Exercises

1. Using your definition of a flip-flop, complete Q in this waveform.





1. Now we'll integrate a flip-flop into a complete circuit that also includes some combinational logic. Notice that Q starts at 1. *Complete the signals D and Q.*





Java -jar logisim.jar

Optional: Java -jar Logisim.jar fire.circ

Java -jar tests/Logisim.jar file.circ

# Model 3: A sequential logic circuit

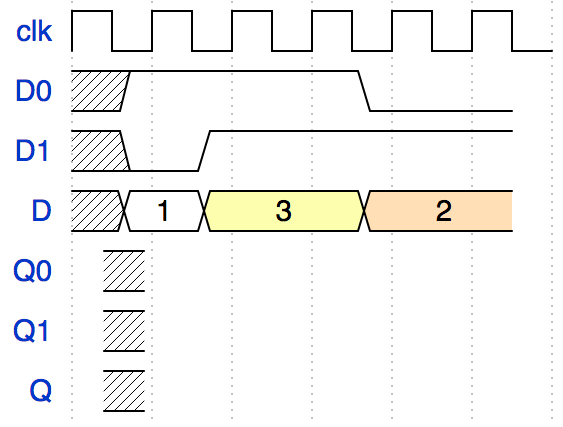
 represents 

Reg = Register

FF = flip-flop

1. In a complete sentence, define what a register is.

Consider the waveform below.





1. On a waveform, how is drawing the value of a multi-bit wire different than drawing the value of a 1-bit wire?

1. Complete the waveform above. The signal names come from Model 3.

# Model 4: Accumulator – proposal 1

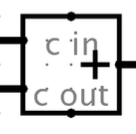
If our goal is to build a programmable processor, we should start with a circuit that just implements *one* program.

|  |  |
| --- | --- |
| **Code** | **Proposed circuit to implement the code** |
| // int3 is a 3-bit int  int3[] A = {5, 1, 3, 1};  // circuit corresponds to  int3 sum = 0;  for (int t=0; t<N; t++) {  sum += A[t];  } | or, with only 1-bit wires, and showing the inside of the 3-bit ripple carry adder |

Delays of components are: 1-bit adder sum=2ns, 1-bit adder carry=1ns

LEGEND: x1 means 1 bit input or output, x3 means 3 bit input or output

Carry in

Input1

ouput

Input2

Carry out

1. What does the program in **Code** do?

Adds all of the number in int3[] A together.

1. In the circuit, at t=0, A[t]=5. At that time (t=0), what is the value of Sum? Explain.

Unknown because no value has come into the circuit.

1. Let’s temporarily assume that the value of Sum is actually 0 at t=0. In the circuit, suppose that a new A[t] is provided every **1ns** (use the values of A given in the code). Label the 1-bit wire schematic with the values on all the wires at these times: t=0ns, 1ns, 2ns, 3ns.
2. Why did the circuit misbehave when the input changed every 1ns?

The circuit will misbehave because it takes longer than 1ns to compute the sum of a 3-bit int.

1. Let’s temporarily assume that the value of Sum is actually 0 at t=0. In the circuit, suppose that a new A[t] is provided every **6ns**. Describe what will happen. Label the 1-bit wire schematic with the values on all the wires at these times: t=0ns, 1ns, 2ns, 3ns.

The 3 adders are out of sync (run at ‘different speeds’ because of carries).

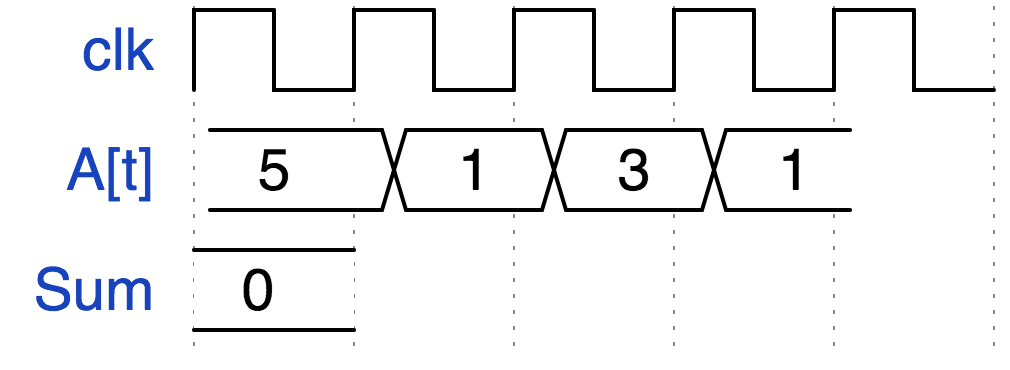
1. Why did the circuit ***still*** misbehave when the input changed every 6ns?
2. Let’s assume the situation in #24, where input changes quickly, is unfair. Given the answers to questions #22, 26, in complete sentences describe the problems with our proposed implementation of the accumulator.

# Model 5: Accumulator – proposal 2

|  |  |
| --- | --- |
| **Code** | **Proposed circuit to implement the code** |
| int[] A = {5, 1, 3, 1};  // circuit corresponds to  int sum = 0;  for (int t=0; t<N; t++) {  sum += A[t];  } | or, with only 1-bit wires, and showing the inside of the 3-bit ripple carry adder and 3-bit register |

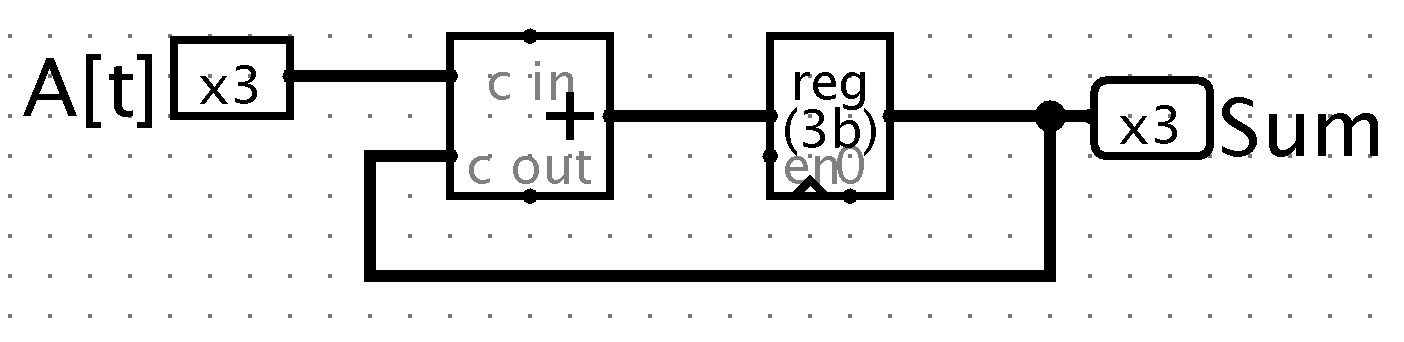
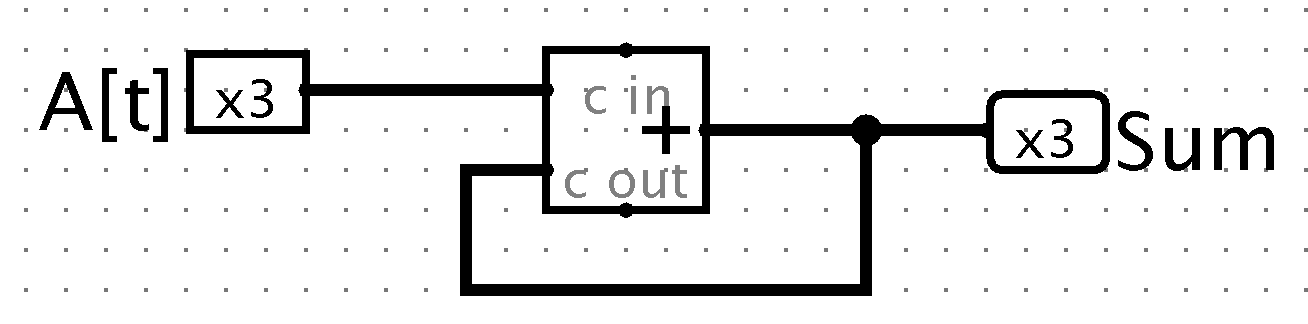
LEGEND: reg means register, 3b means 3 bits, 1b means 1 bit. Note also that in illustrating registers, we assume the clock is wired to it but often don’t show it.

1. Given the sequence of A[t] values, complete the Sum signal.





1. In complete sentences, describe ***how*** adding a register to the accumulator circuit fixes the problems of the first one, which you stated in #27.



It fixes it because all of it is timed now, so there’s no ‘feedback’ loop and unnecessary add’s being done. It also fixes the problem that there was no second input for the first adder (register inputs a 0 now).

# Exercises

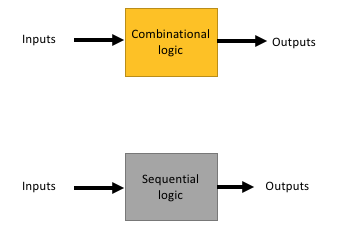
1. Implement a sequential logic circuit that outputs the Fibonacci numbers (next number is the sum of the previous two) 1,1,2,3,5,8,... . The circuit has no inputs. The output should be 32 bits and should change once per clock period, like an accumulator. You may use 32-bit adders and 32-bit registers and you may assume the registers can start holding any value you want (draw the initial value inside the register).

**Recorder:** should draw the circuit on the board as directed by the team

**Manager:** should be the skeptic and test the circuit on the board (there is only 1 test case since the circuit has no inputs!

# Model 6: Comparison between combinational and sequential logic

Consider combinational logic and sequential logic each as a black box with inputs and outputs.



(Assume the sequential logic black box uses a clock signal inside of it, **but you cannot see it**)

1. Suppose the inputs are 1 bit and the outputs are 1 bit. Describe all possible behaviors (input-output relationships) the combinational logic box could have.

Combinational logic: output is a function of the current input.

* Leave the input the same; combinational output CAN’T change after it computes the answer; while Sequential output COULD change (e.g., Model 5 circuit).

1. Suppose the inputs are 1 bit and the outputs are 1 bit. Describe all possible behaviors (input-output relationships) the sequential logic box could have.

Sequential logic: output is a function of previous+current inputs (has memory).

1. How is the behavior of the combinational logic different than the behavior of the sequential logic?
2. If you didn’t know whether a black box was combinational logic or sequential logic, how might you test it to decide?

The diagrams can be reproduced using <https://wavedrom.com/> and the code below.

{signal: [

{name:'clk1', wave: 'p.....' },

{name:''},

{name:'clk2', phase: 0.75, wave: 'hlhlhlh' },

],

head:{

tick:0,

},

}

{signal: [

{name: 'CLK', wave: 'hlhlhlhl'},

{name: 'D', wave: '10.1010.'},

{name: 'Q', wave: '0...1...'}

],

}

{signal: [

{name: 'CLK', wave: 'hlhlhlhlhlhlhl'},

{name: 'D', wave: '1...01.010...1'},

{name: 'Q', wave: 'xx'}

],

}

Model 2 flip-flop definition progression

{signal: [

{name: 'CLK', wave: 'p......'},

{name: 'D', wave: '01..01.'},

{name: 'Q', wave: 'x01..01'}

],

}

{signal: [

{name: 'CLK', wave: 'p......'},

{name: 'D', wave: '01..01.', phase: '-0.4'},

{name: 'Q', wave: 'x01..01'}

],

}

{signal: [

{name: 'CLK', wave: 'hlhlhlhl'},

{name: 'D', wave: '10.1010.'},

{name: 'Q', wave: 'xx0.1...'}

],

}

{signal: [

{name: 'CLK', wave: 'hlhlhlhl'},

{name: 'D', wave: '0....1..'},

{name: 'Q', wave: 'xx0...1.'}

],

}

Model 5

{ "signal" : [ { name: "clk", wave:"p...."},

{ name: "A[t]", wave: "2222", data: "5 1 3 1", phase:-0.1},

{ name: "Sum", wave: "2", data:"0"}

], }