4.2 Logic design conventions

To discuss the design of a computer, we must decide how the hardware logic implementing the computer will operate and how the computer is clocked. This section reviews a few key ideas in digital logic that we will use extensively in this chapter. If you have little or no background in digital logic, you will find it helpful to read COD Appendix A (The Basics of Logic Design) before continuing.

The datapath elements in the LEGv8 implementation consist of two different types of logic elements: elements that operate on data values and elements that contain state. The elements that operate on data values are all *combinational*, which means that their outputs depend only on the current inputs. Given the same input, a combinational element always produces the same output. The ALU shown in COD Figure 4.1 (An abstract view of the implementation of the LEGv8 ...) and discussed in COD Appendix A (The Basics of Logic Design) is an example of a combinational element. Given a set of inputs, it always produces the same output because it has no internal storage.

Combinational element: An operational element, such as an AND gate or an ALU.

Other elements in the design are not combinational, but instead contain *state*. An element contains state if it has some internal storage. We call these elements *state elements* because, if we pulled the power plug on the computer, we could restart it accurately by loading the state elements with the values they contained before we pulled the plug. Furthermore, if we saved and restored the state elements, it would be as if the computer had never lost power. Thus, these state elements completely characterize the computer. In COD Figure 4.1 (An abstract view of the implementation of the LEGv8 ...), the instruction and data memories, as well as the registers, are all examples of state elements.

State element: A memory element, such as a register or a memory.

A state element has at least two inputs and one output. The required inputs are the data value to be written into the element and the clock, which determines when the data value is written. The output from a state element provides the value that was written in an earlier clock cycle. For example, one of the logically simplest state elements is a D-type flip-flop (see COD Appendix A (The Basics of Logic Design)), which has exactly these two inputs (a value and a clock) and one output. In addition to flip-flops, our LEGv8 implementation uses two other types of state elements: memories and registers, both of which appear in COD Figure 4.1 (An abstract view of the implementation of the LEGv8 ...). The clock is used to determine when the state element should be written; a state element can be read at any time.

Logic components that contain state are also called *sequential*, because their outputs depend on both their inputs and the contents of the internal state. For example, the output from the functional unit representing the registers depends both on the register numbers supplied and on what was written into the registers previously. Appendix A (The Basics of Logic Design) discusses the operation of both the combinational and sequential elements and their construction in more detail.

PARTICIPATION ACTIVITY 4.2.1: Combinational and state elements.

1) A datapath element whose output values depend only on the present input values is called a _____ element.
   - ○ combinational
   - ○ state

2) A datapath element that has internal storage is called a _____ element.
   - ○ combinational
   - ○ state

3) Element E has two inputs a and b, and one output z. The following fully defines z:

a	b	z
0	0	0
0	1	1
1	0	0
1	1	1

   Element E is a _____ element.
   - ○ combinational
   - ○ state

4) Element F has input b and output z, originally 0. If b changes from 0 to 1, z changes to 1 and stays 1 even after b returns to 0. If b later changes from 0 to 1 again, z changes back to 0. And so on.

   Element F is a _____ element.
   - ○ combinational
   - ○ state

5) An ALU is a _____ element.
   - ○ combinational
   - ○ state

6) A register is a _____ element.
   - ○ combinational
   - ○ state

7) A clock input is present on a _____ element.
   - ○

combinational

state

8) A sequential element is another name for a _____ element.

combinational

state

Clocking Methodology

A *clocking methodology* defines when signals can be read and when they can be written. It is important to specify the timing of reads and writes, because if a signal is written at the same time that it is read, the value of the read could correspond to the old value, the newly written value, or even some mix of the two! Computer designs cannot tolerate such unpredictability. A clocking methodology is designed to make hardware predictable.

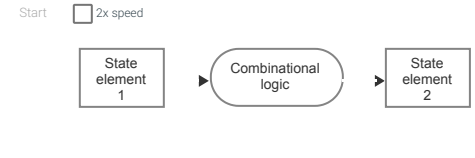
Clocking methodology : The approach used to determine when data is valid and stable relative to the clock.

For simplicity, we will assume an *edge-triggered clocking* methodology. An edge-triggered clocking methodology means that any values stored in a sequential logic element are updated only on a clock edge, which is a quick transition from low to high or vice versa (see the animation below). Because only state elements can store a data value, any collection of combinational logic must have its inputs come from a set of state elements and its outputs written into a set of state elements. The inputs are values that were written in a previous clock cycle, while the outputs are values that can be used in a following clock cycle.

Edge-triggered clocking: A clocking scheme in which all state changes occur on a clock edge.

The following animation illustrates that in a synchronous digital system, the clock determines when elements with state will write values into internal storage. Any inputs to a state element must reach a stable value (that is, have reached a value from which they will not change until after the clock edge) before the active clock edge causes the state to be updated. All state elements in this chapter, including memory, are assumed to be positive edge-triggered; that is, they change on the rising clock edge.

PARTICIPATION ACTIVITY 4.2.2: Combinational logic, state elements, and the clock are closely related (COD Figure 4.3).



The animation above shows the two state elements surrounding a block of combinational logic, which operates in a single clock cycle: all signals must propagate from state element 1, through the combinational logic, and to state element 2 in the time of one clock cycle. The time necessary for the signals to reach state element 2 defines the length of the clock cycle.

For simplicity, we do not show a write *control signal* when a state element is written on every active clock edge. In contrast, if a state element is not updated on every clock, then an explicit write control signal is required. Both the clock signal and the write control signal are inputs, and the state element is changed only when the write control signal is asserted and a clock edge occurs.

Control signal: A signal used for multiplexor selection or for directing the operation of a functional unit; contrasts with a *data signal*, which contains information that is operated on by a functional unit.

We will use the word *asserted* to indicate a signal that is logically high and assert to specify that a signal should be driven logically high, and deassert or *deasserted* to represent logically low. We use the terms assert and deassert because when we implement hardware, at times 1 represents logically high and at times it can represent logically low.

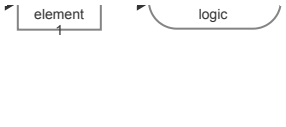
Asserted: The signal is logically high or true.

Deasserted: The signal is logically low or false.

An edge-triggered methodology allows us to read the contents of a register, send the value through some combinational logic, and write that register in the same clock cycle. The animation below gives a generic example. It doesn't matter whether we assume that all writes take place on the rising clock edge (from low to high) or on the falling clock edge (from high to low), since the inputs to the combinational logic block cannot change except on the chosen clock edge. In this book we use the rising clock edge. With an edge-triggered timing methodology, there is *no* feedback within a single clock cycle, and the logic in the animation below works correctly. In COD Appendix A (The Basics of Logic Design), we briefly discuss additional timing constraints (such as setup and hold times) as well as other timing methodologies.

Of course, the clock cycle still must be long enough so that the input values are stable when the active clock edge occurs. Feedback cannot occur within one clock cycle because of the edge-triggered update of the state element. If feedback were possible, this design could not work properly. Our designs in this chapter and the next rely on the edge-triggered timing methodology and on structures like the one shown in the animation below.

PARTICIPATION ACTIVITY 4.2.3: An edge-triggered methodology allows a state element to be read and written in the same clock cycle without creating a race that could lead to indeterminate data values (COD Figure 4.4).



For the 64-bit LEGv8 architecture, nearly all of these state and logic elements will have inputs and outputs that are 64 bits wide, since that is the width of most of the data handled by the processor. We will make it clear whenever a unit has an input or output that is other than 64 bits in width. The figures will indicate *buses*, which are signals wider than 1 bit, with thicker lines. At times, we will want to combine several buses to form a wider bus; for example, we may want to obtain a 64-bit bus by combining two 32-bit buses. In such cases, labels on the bus lines will make it clear that we are concatenating buses to form a wider bus. Arrows are also added to help clarify the direction of the flow of data between elements. Finally, *color* indicates a control signal contrary to a signal that carries data; this distinction will become clearer as we proceed through this chapter.

PARTICIPATION ACTIVITY 4.2.4: Combinational logic, state elements, and rising clock edges.

1) A rising clock edge refers to the clock changing from _____.
   - ○ 0 to 1
   - ○ 1 to 0
   - ○ either 0 to 1, or 1 to 0

2) In the above animations, state element 1 gets written with a new value on the first rising clock edge, causing the combinational logic to output a new value _____.
   - ○ instantly
   - ○ after some amount of time

3) Consider state element 2 in the above animation. A rising clock edge writes a new state element 1 value, serving as input to the combinational logic. If the logic requires 5 ns to output a new value, designers should ensure that rising clock edges are separated by _____.
   - ○ less than 5 ns
   - ○ exactly 5 ns
   - ○ more than 5 ns

4) In the above animation with just one state element, what happens just after a rising clock edge writes a new value to the element?
   - ○ Nothing
   - ○ The logic calculates a new value that then waits at the state element's input for the next rising edge.
   - ○ The logic calculates a new value that is immediately written to the state element.

PARTICIPATION ACTIVITY 4.2.5: Check yourself: Edge-triggered clocking.

1) Because the register file is both read and written on the same clock cycle, any LEGv8 datapath using edge-triggered writes must have more than one copy of the register file.
   - ○ True
   - ○ False

Provide feedback on this section