More on minimization, start on datapath

# Register-Transfer Level Design (RTL, Chapter 5)

For the most part, we’ve been tackling fairly simple problems. And for large ones, using a state machine seems like it could get very complex quite quickly. So how *do* we manage complex designs? The answer is a liberal sprinkling of MSI parts and a state machine to control the whole thing.

Controller

Datapath

The Controller is a state machine. The Datapath is a collection of devices (generally MSI devices like adders, registers, etc.) with control points that lets you choose what to do. You could think of your lab 4 as a datapath where the operation (add, subtract, absolute value) might be controlled by a state machine.

Our text argues that the best way to learn this is to consider a problem you want to solve, then figure out a “high-level state machine”. Then design the datapath and controller. See the table below.



## “Simple” example—Fire a laser.

Controller

Datapath

Problem specification:

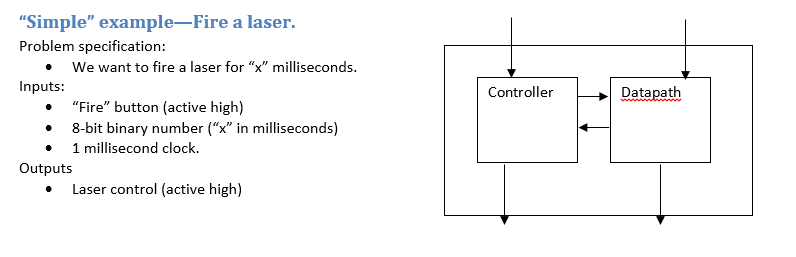
* We want to fire a laser for “x” milliseconds.

Inputs:

* “Fire” button (active high)
* 8-bit binary number (“x” in milliseconds)
* 1 millisecond clock.

Outputs

* Laser control (active high)



## More complex example: multiplier

While that table is the way you want to design these, it’s pretty hard to *start* there with a more complex problem. So we’re going to design the controller for a multiplier given the datapath. First we’ll need to review just what it means to multiply numbers (and binary numbers at that).

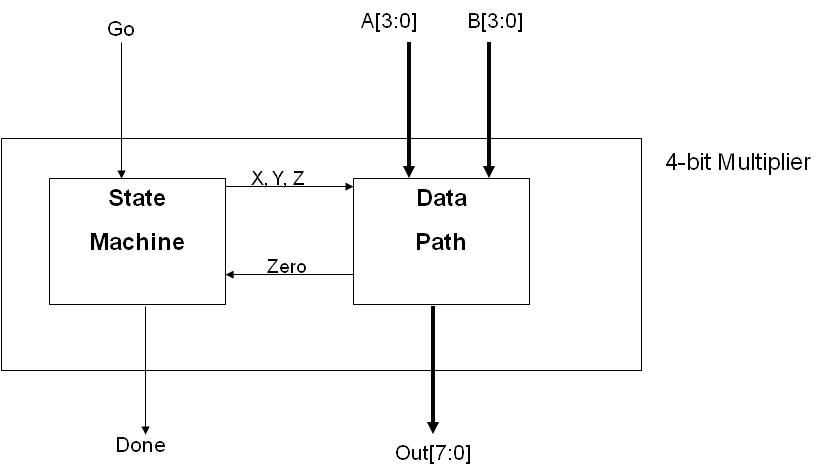
**1001**

**\* 1011**

**======**

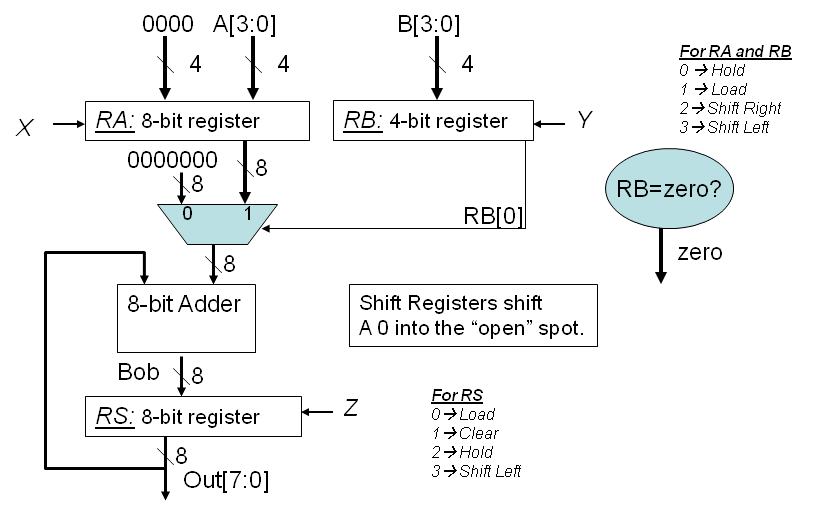
First, note we could build a *combinational* device which does the multiplication. We could brute force it and just build the truth table (for a 4-bit device we’d get \_\_\_\_\_\_\_\_ rows and \_\_\_\_\_\_\_ output columns). But that wouldn’t scale very well. We could certainly come up with something trickier (like we did for N-bit adders). But it’s not obvious what that would be. So let’s build a sequential device (that takes multiple clock ticks to solve the problem).

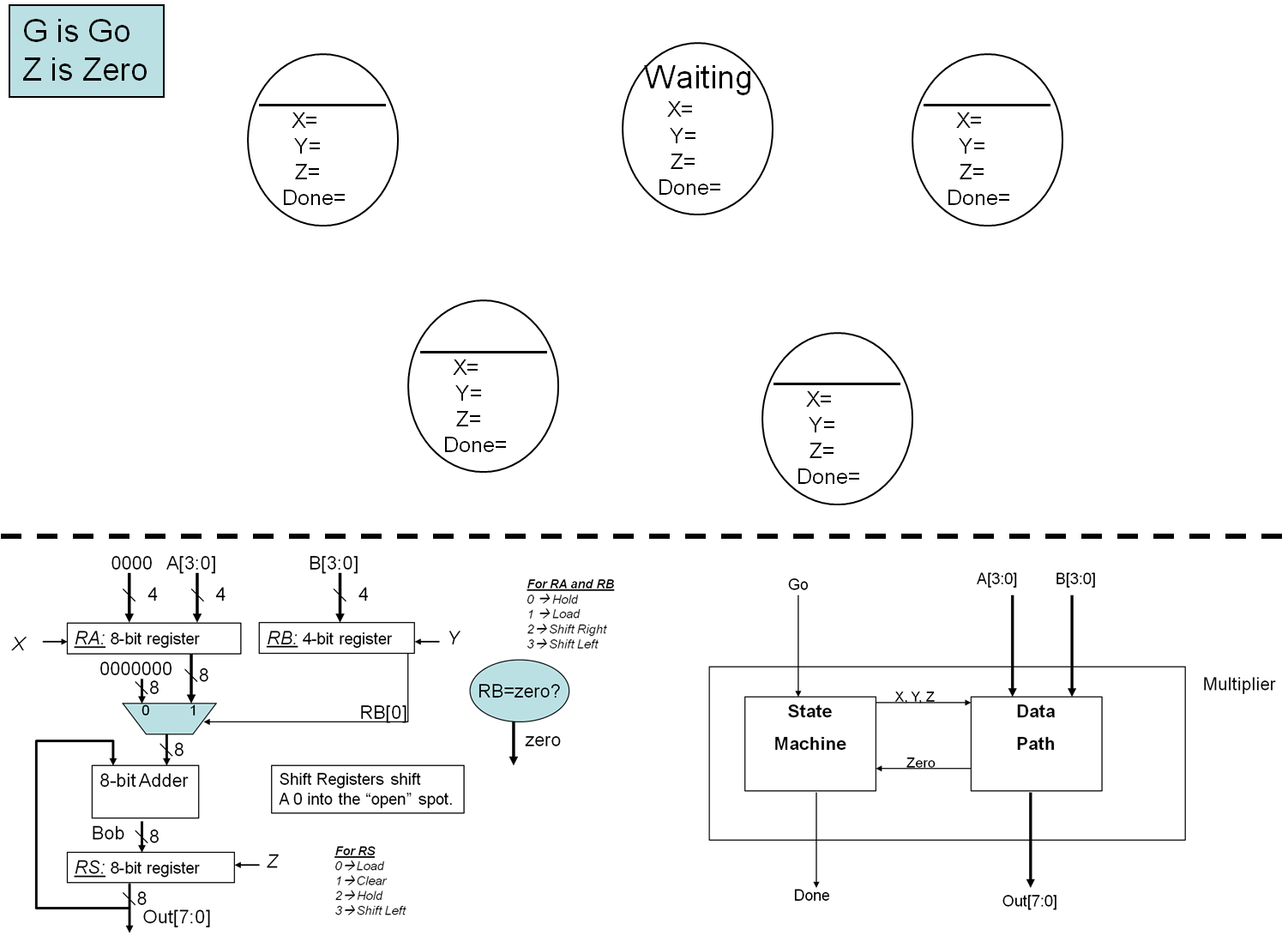
Say our multiplier is going to take some time (clock ticks) to solve the problem. In that case, we’re going to want a signal to tell it to start and one to tell us when it’s done. In addition we need the two inputs (each N bits say) and one output \_\_\_\_\_\_\_\_\_\_\_ bits wide.

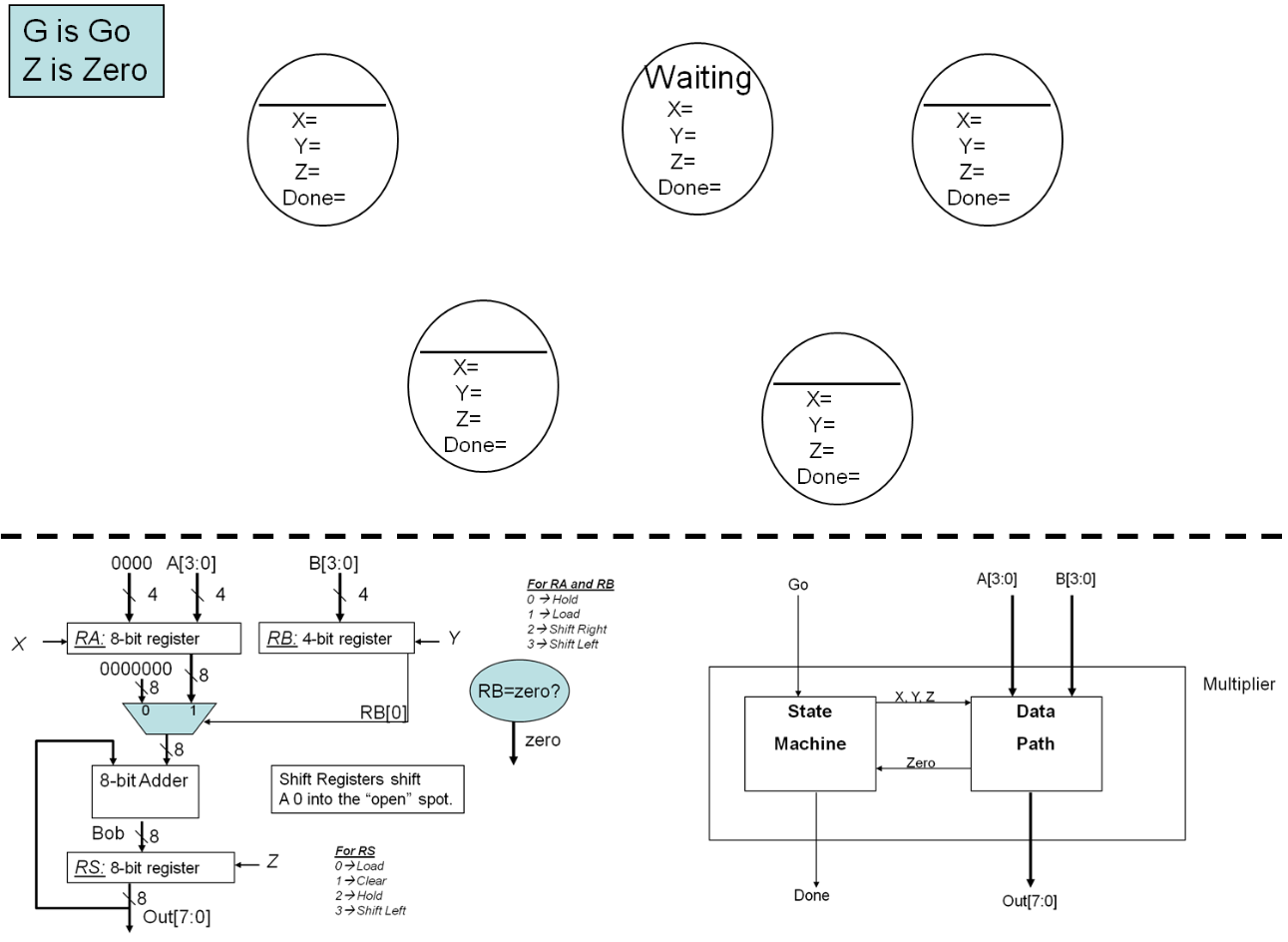
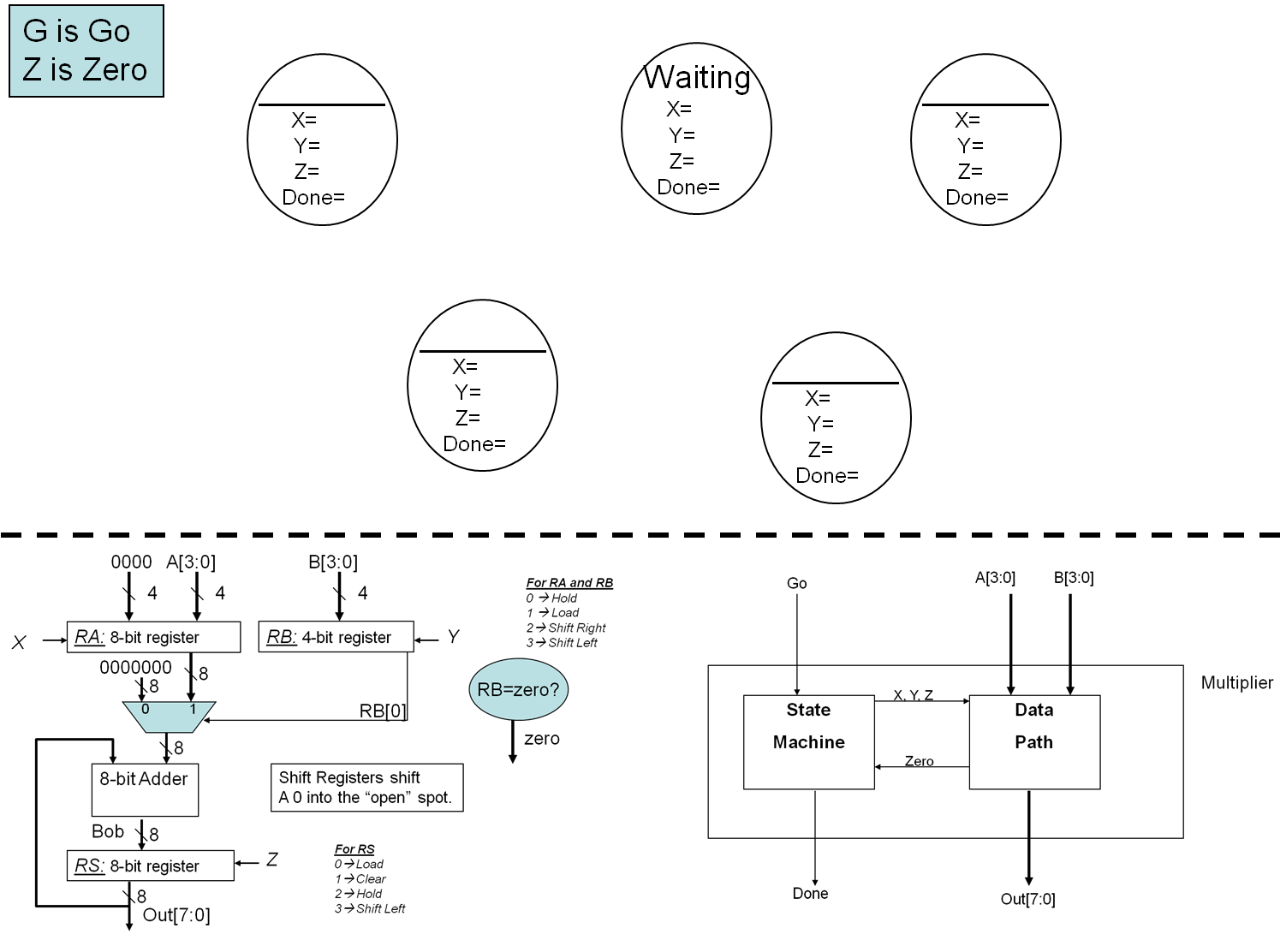


Let’s say we use the following datapath:

Now we need a state machine to control this…







CLK

Go

X

Y

Z

Bob

RA

RB

RS

