Continuing on design with datapaths

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).

\[
\begin{array}{c}
1001 \\
\times 1011 \\
\hline
\end{array}
\]

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...
G is Go
Z is Zero

Waiting
X =
Y =
Z =
Done =

X =
Y =
Z =
Done =

X =
Y =
Z =
Done =

---

For RA and RB
0 ⇒ Hold
1 ⇒ Load
2 ⇒ Shift Right
3 ⇒ Shift Left

8-bit Adder
Bob ↓ 8
RS: 8-bit register

Shift Registers shift A[0] into the "open" spot.

For RS
0 ⇒ Load
1 ⇒ Clear
2 ⇒ Hold
3 ⇒ Shift Left

8000 A[3:0] 4 4
000000 8
0 8
RB[0]

Y

RB=zero?

zero

0000 A[3:0] 4 4
B[3:0] 4 4

Data Path

Multiplier

State Machine


Go

Done

Out[7:0]
Another example: laser distance sensor.

- Laser light travels at speed of light, \(3 \times 10^8\) m/sec
- Distance is thus \(D = \frac{T \text{ sec} \times 3 \times 10^8 \text{ m/sec}}{2}\)
- Inputs/outputs
  - \(B\): bit input, from button to begin measurement
  - \(L\): bit output, activates laser
  - \(S\): bit input, senses laser reflection
  - \(D\): 16-bit output, displays computed distance

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| Step 1 | Capture a high-level state machine  
Describe the system’s desired behavior as a high-level state machine.  
The state machine consists of states and transitions. The state machine is “high-level” because the transition conditions and the state actions are more than just Boolean operations on bit inputs and outputs. |
| Step 2 | Create a datapath  
Create a datapath to carry out the data operations of the high-level state machine. |
| Step 3 | Connect the datapath to a controller  
Connect the datapath to a controller block. Connect external Boolean inputs and outputs to the controller block. |
| Step 4 | Derive the controller’s FSM  
Convert the high-level state machine to a finite-state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath. |