# CSCB58 Lab 6: Datapath

Your task this will will be to build a control unit $AX^2 + BX + C$ using the 8-bit datapath shown below. For this lab we have already built the datapath for you! All you need to do is the design and implement the control unit.

**[TASK] Make sure to download the file `lab6_base.circ` from Quercus. Use it as the base for your solution.**

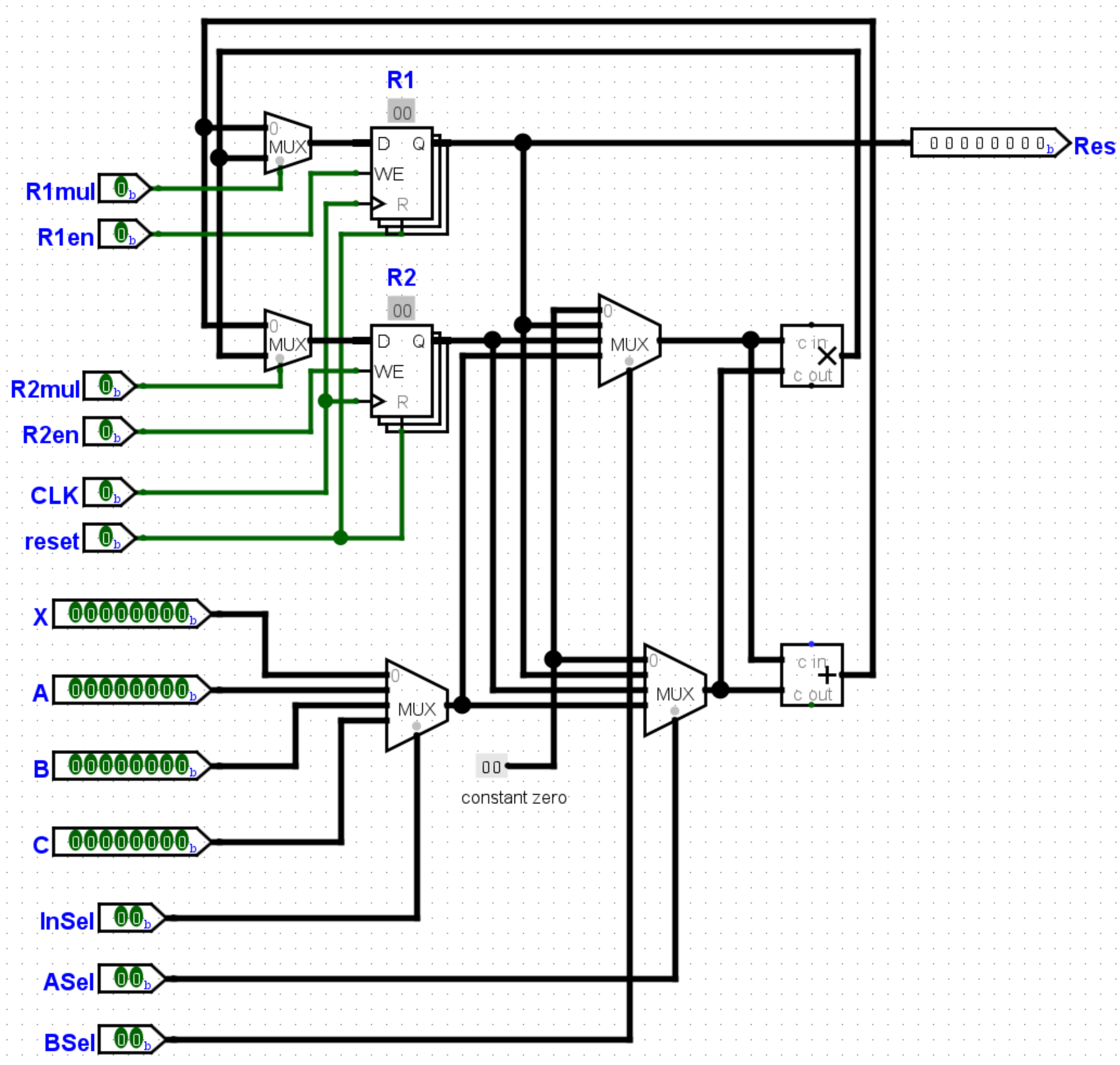



Figure 1: The datapath used for this lab

# 1 Introduction

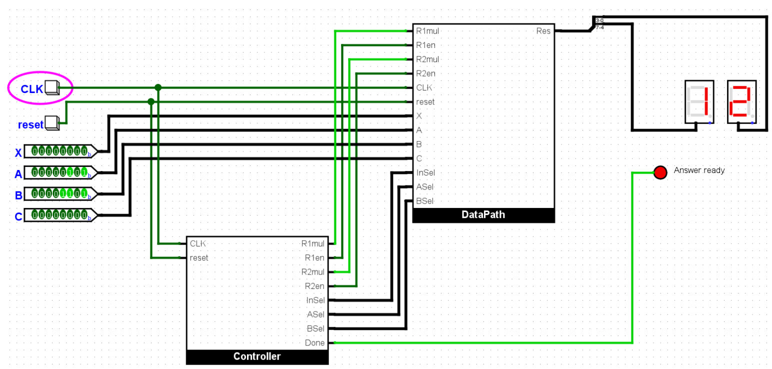
First, open `lab6_base.circ` in Logisim.

You will see that the datapath is implemented for you in the "DataPath" subcircuit. We have implemented a sample control unit that uses this data path to compute $A + B$ in 3 cycles.

Try it:

1. Use the poke tool to set A and B to some numbers, for example 3 (00000101) and 13 (00001101). Make the result will not overflow 8-bits.
2. Use the poke tool to click the "reset" button to reset the state of the circuit.
3. Use the poke tool to click "CLK" 3 times, thereby advancing the clock 3 times.

You should see that the "Answer ready" light has turned on, and the result (shown in base 16) is 12, which is 18 in decimal. This should look something like the following




The `lab6_base.circ` file includes the following circuits:

- "main" – wires up the datapath, controller, buttons, and input/output. Looks like the image above.
- "DataPath" – the datapath as shown in the first page.
- "ControllerLogic" – the combinatorial logic for state and output logic of the FSM. Currently, it implements $A + B$ but you will modify it.
- "Controller" – the control unit (FSM) that sends the control signal. It uses ControllerLogic and just adds the necessary flipflops. You may need to modify it.

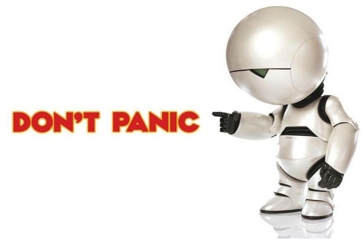
# 2 Design

Your first task is to design a control unit that computes Res $= AX^2 + BX + C$ using this datapath. This datapath has two registers (R1 and R2), an adder (the square marked as "+" in the right side of the datapath), a multiplier (marked as "×"), and bunch of muxes that control what goes into each of these components.

**Requirements:**

- At the end of the computation, your FSM should make the datapath stop and output Done=1 to tell the user the result is ready. If not yet done, output Done=0.
- The user is expected to set up the inputs then press Reset at the beginning of every computation. Make sure to do the right thing using the "Reset" signal.
- The best design will finish the computation using the minimum number of steps. **A design that uses more steps/flipflops than necessary may not receive full marks**. Note that you will need additional flipflops than what is included in the example FSM.

**Guidance:**




1. Don't panic. You can do this.
2. Look at the datapath (shown in page 1). Which inputs of which components do various control signals (for example ASel) connect to? What do these signals **do**? Now figure out what you kind of basic steps you can actually do with the datapath, given the wiring, muxes, and components available.
3. Split $AX^2 + BX + C$ operation into a sequence of basic steps that you can do with the datapath. Make sure these are actually possible with the datapath! Sometimes we wish we could do something, but the wiring in the datapath does not actually allow it.
4. Convert the sequence of steps to a table that shows for each step: (a) the step number (start with zero); (b) what this step does; (c) what the values in R1 and R2 be after this step; (d) what are the control signals you need to achieve that; and (e) what is the number of the next step.

   **[TASK] Submit this table as a PDF to Quercus.**
5. Note that (a), (d) and (e) together make up the truth table for your FSM! You may need to add a Done signal too.
6. Given that truth table, you know know how many flipflops you'll need.

As an example, here is the table for the example FSM that computes $A + B$. (We stress that this is not necessarily the fastest way to compute $A + B$ using this datapath!)

| Step | Operation | In R1 | In R2 | R1mul | R1en | R2mul | R2en | InSel | ASel | BSel | Done | NextStep |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | $R1 = A$ | $A$ | ? | 0 | 1 | x | 0 | 01 | 11 | 00 | 0 | 1 |
| 1 | $R2 = B$ | $A$ | $B$ | x | 0 | 0 | 1 | 10 | 11 | 00 | 0 | 2 |
| 2 | $R1 = R1 + R2$ | $A + B$ | $B$ | 0 | 1 | x | 0 | 10 | 01 | 10 | 0 | 3 |
| 3 | Hold | $A + B$ | $B$ | x | 0 | x | 0 | xx | xx | xx | 1 | 3 |

Note the use of don't-cares (x) when we don't care about a particular MUX signal. This can happen, for example, if the register being "fed" by this MUX is not loading (en=0).

# 3 Implementation

Modify "ControllerLogic" and optionally "Controller" to implement your FSM:

1. You can change "ControllerLogic" (including pins) as much as you want.
2. You can change, add, or remove components in the "Controller" subcircuit. However but you may **not** change (or remove, or add) any of the input/output pins. They are connected to the main circuit, which we already wired.
3. Do not change the DataPath or main circuits. Make sure you do all your changes to the "Controller" and "ControllerLogic" sub-circuit.
4. You are explicitly encouraged to use the circuit analyzer (see Review lecture 5), but note you still need to understand your circuit and be able to answer questions. If you want more practice in using K-maps, you can also do it manually.
5. Test your circuit. Note it is very easy to cause multiplication to overflow (especially with $AX^2$) so **make sure to use appropriately small inputs** so that $AX^2 + BX + C$ does not overflow.

**[TASK] Save your circuit as `lab6.circ` and submit to Quercus**.

## Summary of tasks

1. Design the FSM and submit the table of steps and output signals.
2. Implement and test the circuit in Logisim. Go back to previous steps if you've made a mistake.
3. Submit zip file to Quercus with the table of steps and the circuit.
4. Demonstrate the use of your circuit to the TA and explain why it is working correctly.

### Evaluation

As always, marks are based not only on submitted work but also on oral examination by TA. You need to be able to make reasonable explanations about any details to the TA.

| | | |
|---|---|---|
| Solution | Submitting everything on time | 1 mark |
| | Design | 3 marks |
| | Implementation | 1 marks |
| Understanding | TA oral score | 1 to 3 |

Final lab marks (up to 5) are determined by multiplying your solution subtotal by the oral score (1–3) and dividing by 3:

$$\text{total} = \text{solution marks} \times \frac{\text{oral score}}{3}$$