← Lab 8: Making a multiplier/divider

Finish by midnight on Sunday, 11/18

Your task

You will make an **8-bit by 8-bit multiplier** which will take (a maximum of) 8 clock cycles to produce the product. It will also stop running and indicate that it is finished multiplying.

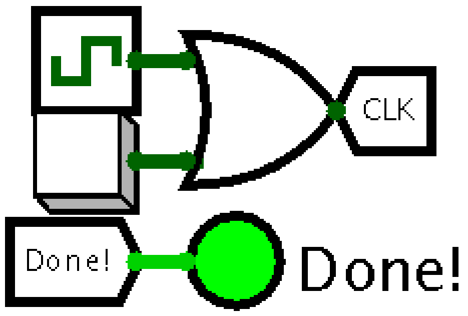
TL;DR

The bullet-list of requirements:

- Your circuit must have the clock, "Done!" LED, and registers named appropriately as shown below.
- Your circuit should use a splitter to get the LSB of the multiplier.
- Your circuit should use the write enable inputs on the registers to stop updating them when the multiplication is finished.
- The multiplication is finished when the multiplier is 0.

Starting off

To make it easier for the grader, please start off by placing the following things near the top-left of the circuit:

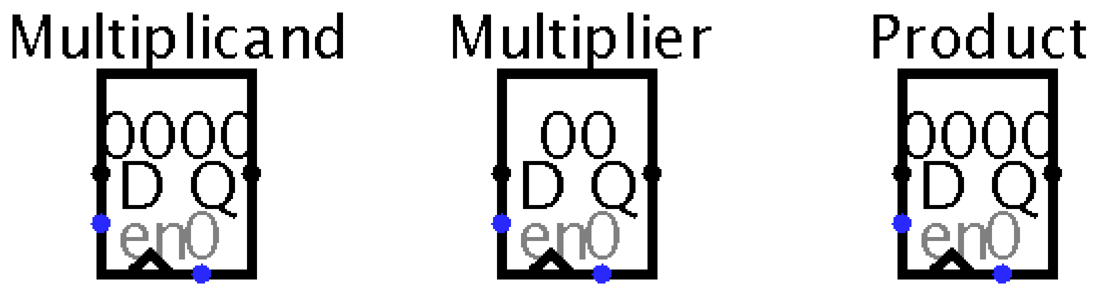


This is:

- a clock and an **Input/Output > Button** going into an OR gate, to produce the clock signal
  - (by doing this, you can tick the clock by clicking the button)

- An **Input/Output > LED** labeled “Done!”

In addition, please name your three registers like so:



Time to multiply

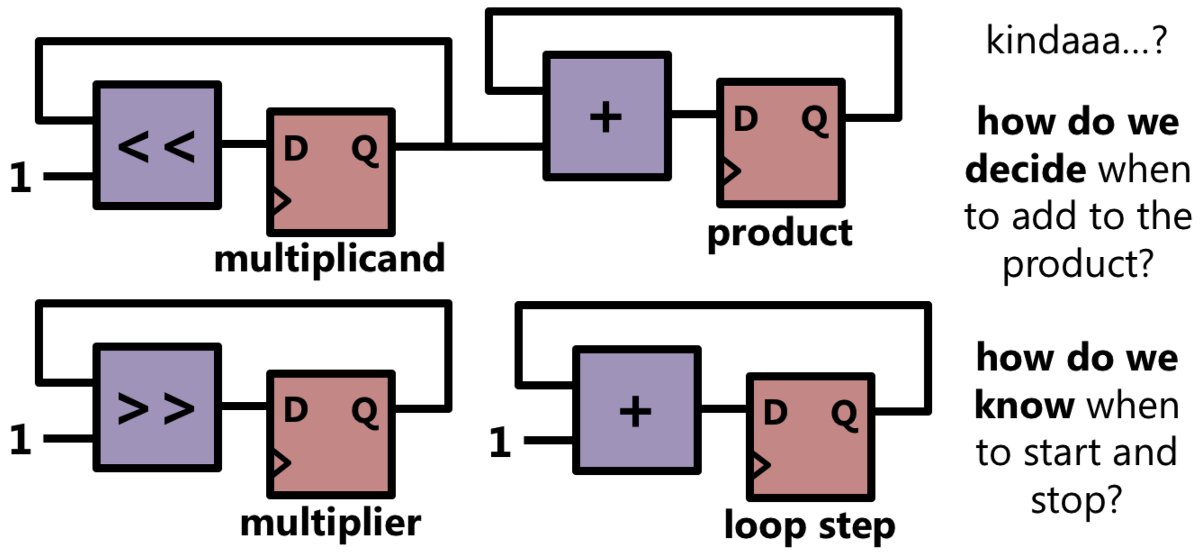
Here is the algorithm you’ll implement in circuitry:

```
while(multiplier != 0) { // condition is a little different th
    if(multiplier[0] is 1) {
        product += multiplicand
    }
    multiplicand <<= 1
    multiplier >>= 1
}
```

The condition of the loop is now `multiplier != 0` - we don’t *have* to go through all the bits of the multiplier, if the remaining bits are all 0s. It’s just a little optimization.

Making the Multiplicand and Multiplier registers

Remember the circuit sketch from the FSMs lecture?


1
<<
D Q
multiplicand
+
D Q
product
kindaaa...?
how do we decide when to add to the product?
1
>>
D Q
multiplier
1
+
D Q
loop step
how do we know when to start and stop?

You won't need the "step counter" register in your circuit. Seeing if the multiplier is 0 is enough.

- how big does the multiplier register need to be?
- what about the multiplicand register?
- what do we do with the values in those registers?

The **Arithmetic > Shifter** component does bit shifting.

- set its **Shift Type** property to either "Logical Left" or "Logical Right"
- the second input is the distance to shift, which you can make a **Wiring > Constant**
  - what do you notice about the *number of bits* for the second input?
  - why do you think that is?

Make sure the registers have their **clock inputs hooked up** (use tunnels to connect them).

Then try **putting some values in the registers.** You can use the poke tool, click the register value, and type in whatever you want on the keyboard.

Then, **tick the clock.** You should see the values in the Multiplicand **increase,** and the values in the Multiplier **decrease.**

Make sure you get the above stuff working first!

Implementing the "condition"

Your multiplier has to indicate when it's done by lighting up that LED.

Look at the loop in the pseudocode above. That's the condition you want to test for.

You can check the condition with an **Arithmetic > Comparator** component. It outputs three signals for <, =, and >, but you can just use the one you need.

Remember that the multiplication is done when the condition is **false (0).**

When the multiplication is done, the Multiplier and Multiplicand registers should be disabled, and the LED should light.

Now, when the multiplier register contains 0, the Done LED should light, and ticking the clock should have no effect on the Multiplier and Multiplicand registers.

Making it multiply

You'll need the third and final register to hold the product.

Look back at the multiplier circuit diagram from class. See how things are hooked up? Do that. :)

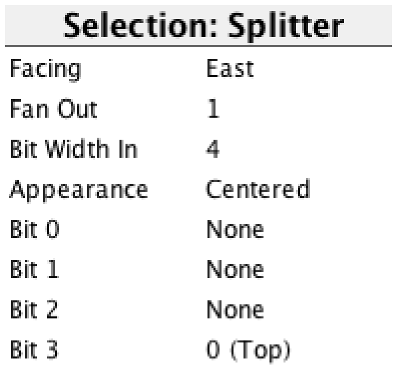
When do we write to the product register? That will tell you what needs to go into its *write enable* input.

> Make sure the product register is only written to when your circuit is running (not when it's done!).

Tip: if you want to get a *single bit* from a wire, try this:

1. make a **splitter**
2. set its **Bit Width In** to the size of the input value
3. set its **Fan Out** to 1
4. for the **Bit 0, Bit 1, etc.** properties, set **the bit that you want** to "0 (Top)" and set the rest to "None"

As an example, this extracts the MSB of a 4-bit number:



Selection: Splitter

Facing	East
Fan Out	1
Bit Width In	4
Appearance	Centered
Bit 0	None
Bit 1	None
Bit 2	None
Bit 3	0 (Top)

Testing it

Now you can see if it works. Try putting some values in the Multiplier and Multiplicand registers, and then tick the clock until the Done LED lights up. See if the product is what you expect.

Use **Wiring > Probes** with the "Radix" set to "Unsigned Decimal" to make it easier to see the values going into and out of the circuit.

If your circuit isn't giving you what you expect, try doing the binary multiplication on paper, and see if your circuit is giving the same results on each step. Also make sure you're doing the shifting at the *right spot in the circuit*. Read the pseudocode carefully. Do you shift before or after the `if`?

Extra credit (+1% of final grade): Division

Make a new subcircuit (**Project > Add Circuit...**) and name it "Division."

Now implement an 8-bit by 8-bit division circuit based on the circuit sketch that was shown in class.

Hints:

- You can start with your multiplier circuit as a base - division is just a slight rearrangement of the same basic parts.
- Unlike with multiplication, there's no real way to "stop early". So your circuit will always take 8 clock cycles to do its job.
- When testing, put the dividend (numerator) in the **remainder** register, and the divisor (denominator) in the **upper** 8 bits of the **divisor** register, and .
    - So if you want to do `4E ÷ 09` then you'd put `4E` in the remainder and `0900` in the divisor.

Submitting

Once you're sure your circuit works, you can submit.

Name your circuit file `username_lab8.circ`, like `jfb42_lab8.circ`.

Submit here.

Drag your asm file into your browser to upload. **If you can see your file, you uploaded it correctly!**

You can also re-upload if you made a mistake and need to fix it.