

# **COMP2611: Computer Organization**

## **Building Registers**

# Overview

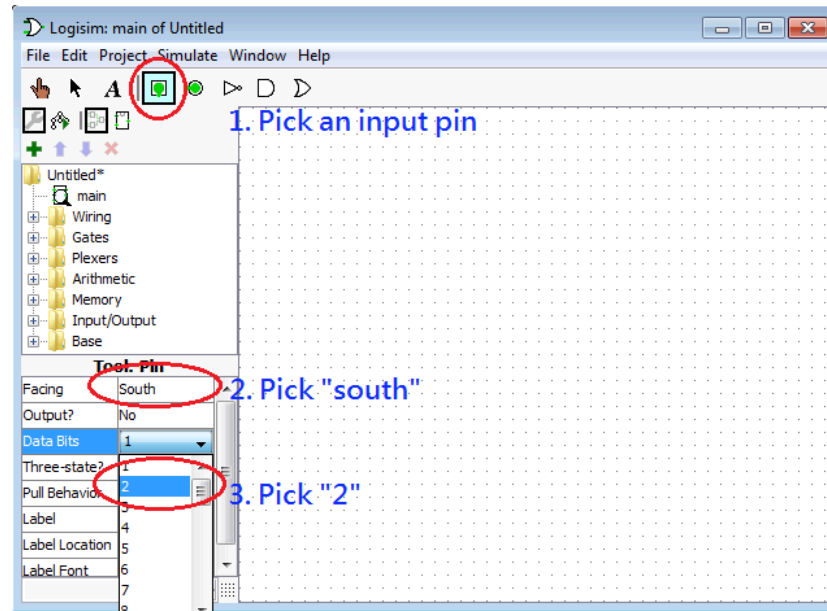
- You will learn the following in this lab:
  - ▣ building a 2-bit register,
  - ▣ building a 2-by-2 register file.


# Building a 2-bit register with the D Flip-Flops

- Continue from the last lab.
- We can build a simple register that holds 2 bits with the D flip-flop (you can use the D flip-flop to build register of any size, we build a 2-bit register here for simplicity).
- Basically, the idea is to use two D flip-flops in the circuit, each of which holds a single bit.
- The input data will be 2-bit in width. We need to separate its individual bits by using a “splitter” and direct the bits to the corresponding D flip-flops.

# Building a 2-bit register with the D Flip-Flops

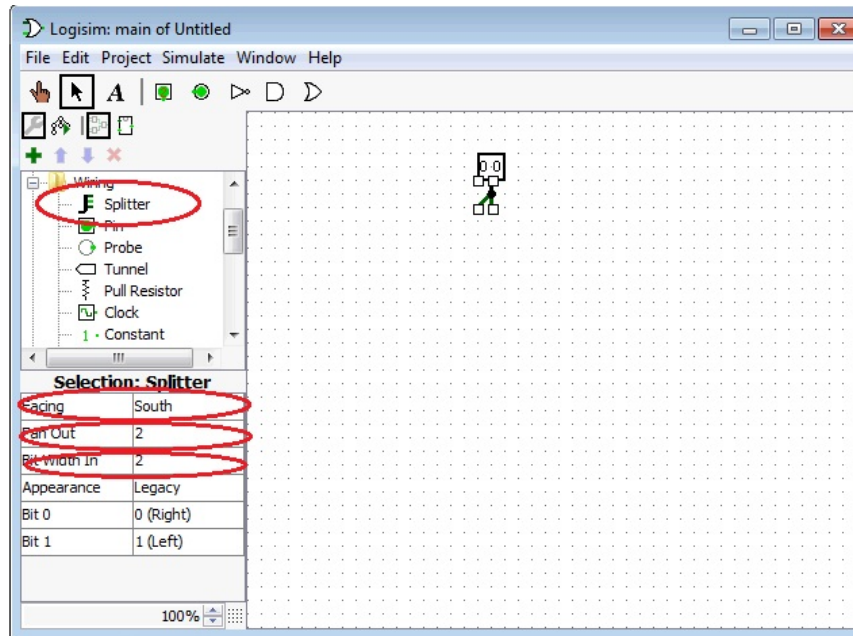
- To have the 2-bit input, you need to follow the 3 steps below to set up the input pin.



- When you pick “south” for “Facing” in step 2, wires can be drawn from the bottom (south) of the input pin. It is up to you whether you want to choose “south”.
- After the 3 steps, you drag the pin to the canvas and you will have this  2-bit input pin on the canvas (values have been initialized to 0,0).

# Building a 2-bit register with the D Flip-Flops

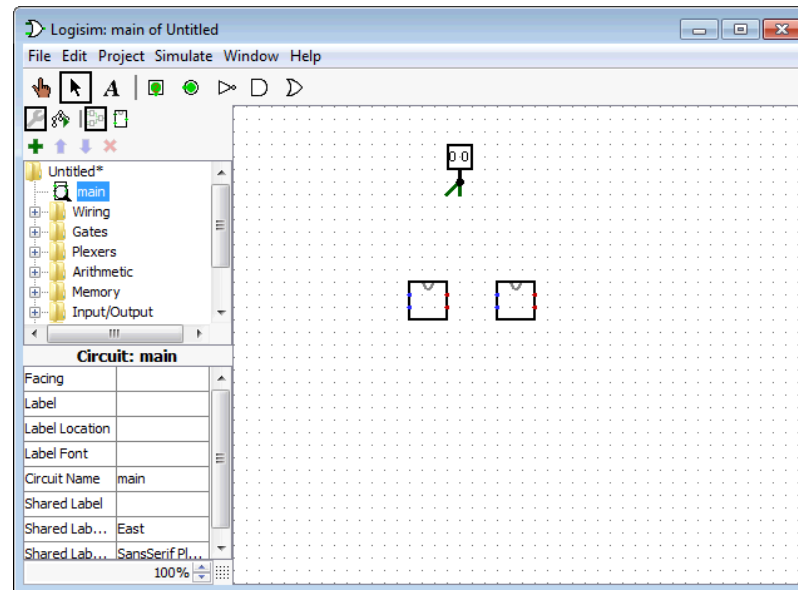
- Select a splitter from the “wiring” folder of the “Explore pane”.



- Make sure you have set both the “Fan-Out” (number of output legs) and “Bit Width In” to 2 so that each leg carries 1 bit of data.

# Building a 2-bit register with the D Flip-Flops

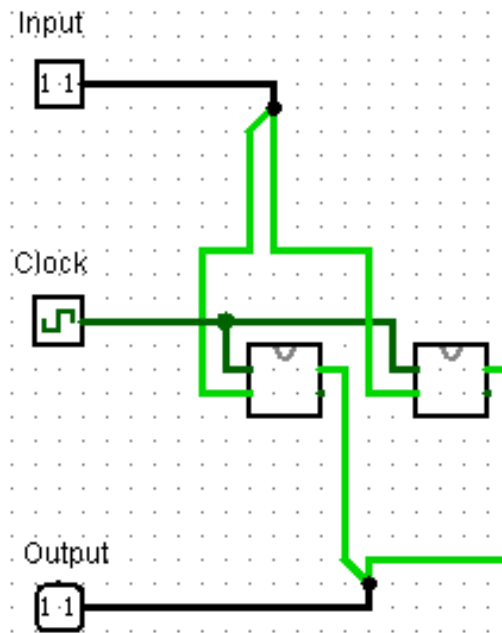
- Now load the “D-flip-flop-withoutclock.circ” as a library and use it as a component of the register. You can load it by clicking “Project”, ”Load Library”, “Logisim library...” and select “D-flip-flop-withoutclock.circ”.
- Drag two D flip-flops from the “Explorer pane”.



- Note the two input dots (in blue) on the left of the flip-flop and the two output dots (in red) on the right.
- The upper input dot is C (Clock), the lower input dot is D (Data).
- The upper output dot is Q and the lower output dot is  $\sim Q$ .

# Building a 2-bit register with the D Flip-Flops

- To store 2 bits into the flip-flops, you need to forward the bits from the splitter to the corresponding D flip-flops.
- You also need to connect a clock signal to both of the D flip-flops.
- Don't forget to retrieve the output from the flip-flops (using an output pin).
- Here is an example circuit for the register:



## Building a 2-bit register with the D Flip-Flops

### ■ Some hints:

- ❑ You may **change the “splitter” facing direction and “Left- or right handed display”** by change the splitter attribute. And, take note on the **consistent position of the bit number**. E.g. if set rightmost bit is bit 0, then the input-splitters-output should be designed such that the bit positions of them consistent
- ❑ The **colors of the wires** hint you on possible errors, make good use of this information. **Any color of wire other than light, dark green means error occur, either connection or logic error.**
- ❑ **The output pin should be set to output, in the attribute table.**
- ❑ Each of the input/output pins can be connected in one direction only, as determined by the “facing” attribute. Make sure you connect it correctly.
- ❑ A splitter splits bits according to their location in the input, the order is preserve on the splitter, (i.e., left-most bit on the left-most split wire and so on)
- ❑ Logisim could sometimes have mysterious errors on the wires even though there is no error at all. An easy way to solve this is to copy and paste the whole circuit to a new canvas. This usually solves the problem.

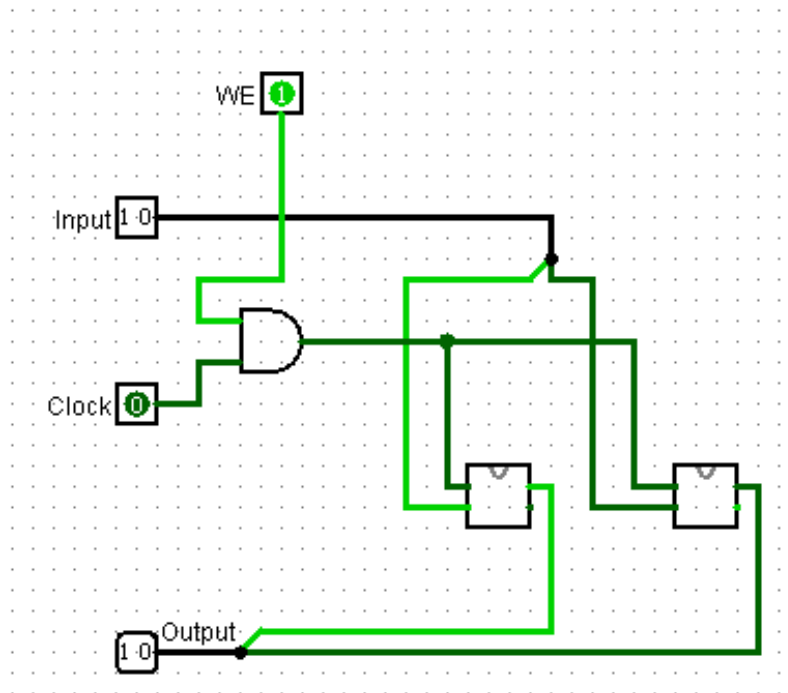


# Building a 2-bit register with Write-Enabled input

- In some usage of a register, we may want to disable the writing of the input data to the register temporarily at some falling edge of the clock signal.
- To do that, a register can include an extra input (say WE for Write Enabled) so that at a falling edge of the clock signal,
  - ☐ If WE is 1, the register will be updated with the input data;
  - ☐ If WE is 0, the register will NOT be updated.

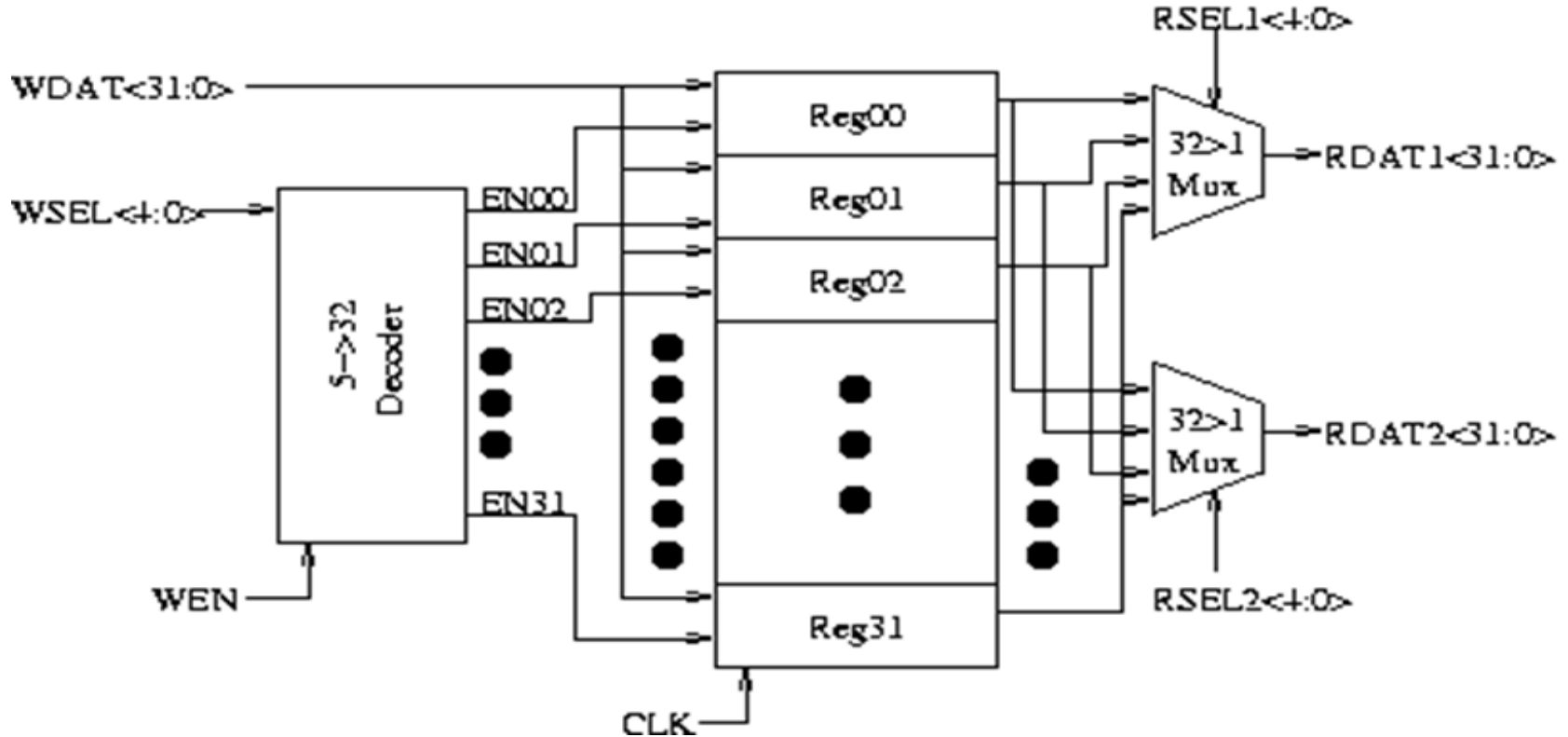
# Building a 2-bit register with Write-Enabled input

- Add the additional circuit to our previous 2-bit register, and then create the 2-bit register with the extra WE input as shown below.
  - Test it afterward.



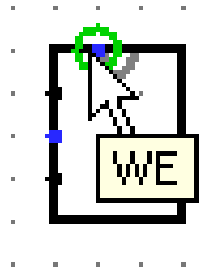
# Register file

- The Decoder (5→32bit) input WSEL control which one of the EN00 to EN31 will be enabled (i.e. EN??=1), that EN?? in turn decides which register (reg00-reg31) will be enabled to **write the input data into the specified register**
- RSEL control from which register to **Read from and output to RDAT1 and RDAT2**



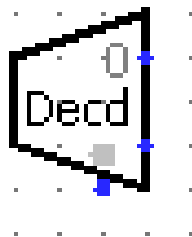
# Building a 2-by-2 register file

- We now build a **2-by-2 register file**, which consists of 2 2-bit registers, using the previous register.
  - The register file's registers are numbered 0, 1, .... (like the location addresses in memory). It has an extra input, **RegNum**, for specifying the no. (in binary format) of the target register for reading and writing data.
- Store the previous register with the clock input replaced by a 1-bit input pin to the file **"2bit-reg.circ"**.
- Start a new Logisim project, and load this file as a Logisim library in it.
- Add two such registers from the library to the project.
- On each register, note the top dot for the WE input pin and the left three blue dots for the input data, clock and output pins (from top to bottom).



# Building a 2-by-2 register file

- Select a decoder from the “Plexers” folder of the “Explore pane”.
  - ☐ Its outputs 0, 1, ... are positioned from top to bottom on its right side (with the “0” label on it for the output 0).
  - ☐ Its “Include Enable?” attribute is for including/excluding an Enable input on it, which is used to enable/disable all the output wire connections of the decoder. Just set this attribute to “no” since that input is not needed in our case.

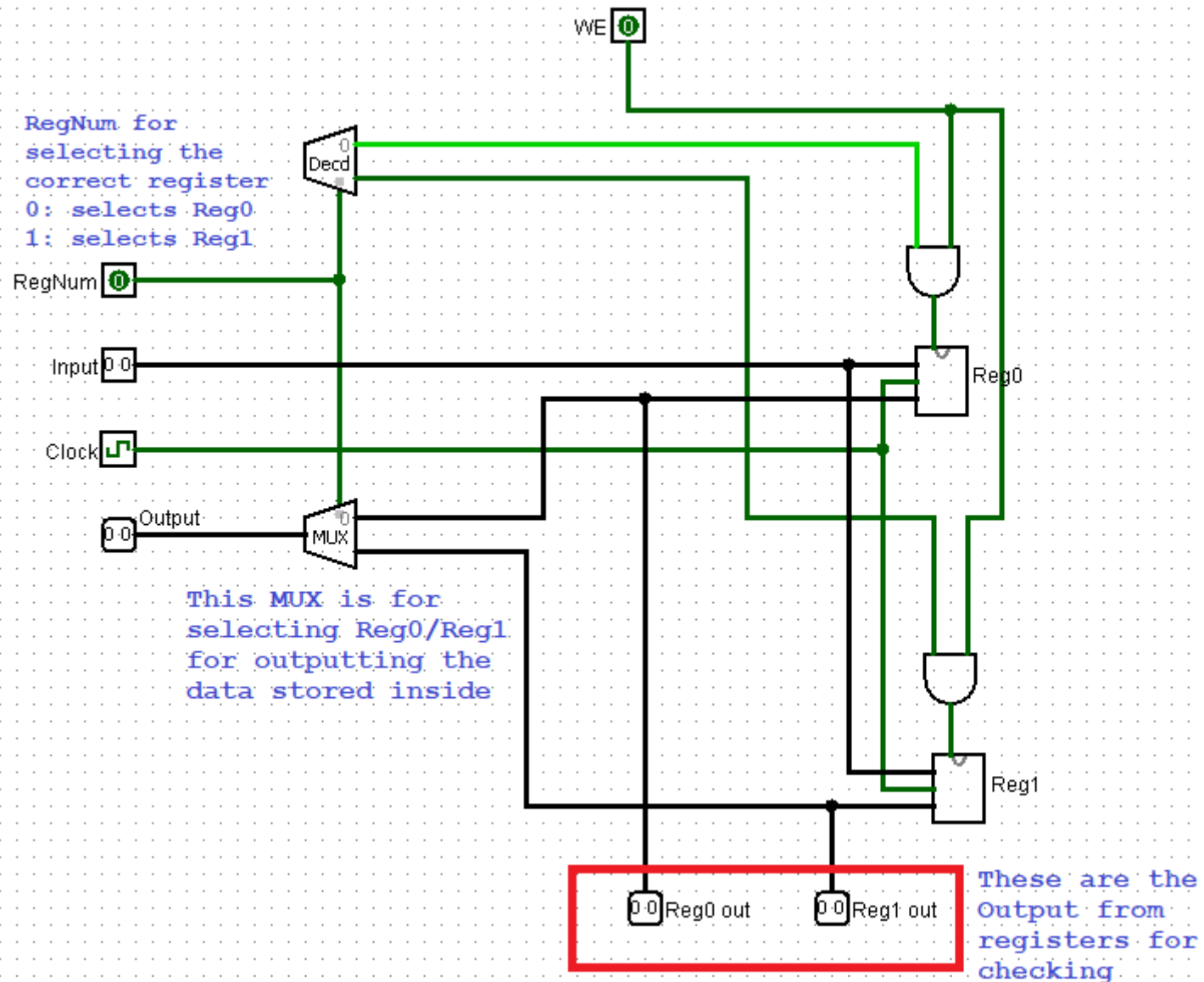


| Tool: Decoder   |             |
|-----------------|-------------|
| Facing          | East        |
| Select Location | Bottom/Left |
| Select Bits     | 1           |
| Three-state?    | No          |
| Disabled Output | Floating    |
| Include Enable? | No          |

# Building a 2-by-2 register file

- Add a decoder and a multiplexer to select the target register in the register file based on RegNum for writing and reading its data, respectively. The decoder sets its WE input to 1 to enabling the writing.
  - Register 0 is positioned above Register 1 on the canvas.
- Create the register file as shown below and test it.
  - Change the “Facing” attribute of any component to get its wire dots on the positions convenient for wiring.
- This is a 2-by-2 Register file (2 bits register, with 2 register):
- Compare with the previous schematic diagram of 32-bit Register File
  - Input (2-bit) ~ WDAT (32-bit)
  - Decoder (1>2) ~ Decoder (5>32)
  - RegNum ~ WSEL, RSEL; WE-0 ~ EN00; WE-1 ~ EN01
  - MUX( 2>1) ~ MUX (32>1)
  - Output ~ RDAT1 (one of the output)
  - WE ~ WEN (Write Enabled)

# Building a 2-by-2 register file



# More register file features

- The previous register file always performs both writing and reading in the target register at every falling edge of the clock.
- To separate the writing and reading in two (possibly different) registers, two inputs (instead of one) that specify the no. of those registers will be connected to the decoder and multiplexer, respectively.
- To allow disabling the writing in all the registers temporarily at some falling edge of the clock, an extra input will be used in the register file to set all the registers' WE inputs to 0 at that edge, regardless of the values of the decoder's outputs.



# Conclusion

- We covered the following topics today:
  - building a 2-bit register,
  - building a 2-by-2 register file.